SPERRY

PUBLICATIONS RELEASE

MAPPER 10/System 11

Type 3065

Processor and Storage

Reference

UP-9955

This Library Memo announces the release and availability of *Type 3065, Processor and Storage, Reference,* UP-9955.

The Type 3065 central complex components described in this manual are required components in some smaller Series 1100 Systems (for example, the Distributed Data Processing System 11).

This manual provides hardware-oriented information for the following Type 3065 central complex components:

- K3649 Instruction Processor
- K3650 Disk Controller Channel
- K3651 Byte Bus Channel
- K3652 Block Multiplexer Channel
- K3653 Main Storage Unit

This manual also includes information for the integrated Streaming Tape Subsystem; and in the appendices, a listing of instructions by function code and mnemonic.

This manual is intended for system programmers and system analysts who are familiar with similar equipment.

This manual consists of:

1. Introduction
2. Instruction Processor
3. Channel Input/Output Processors
4. Main Storage Unit
5. Integrated Streaming Tape Subsystem

Appendix A. Instructions Listed by Function Code
Appendix B. Instructions Listed by Mnemonic
Appendix C. Abbreviations, Acronyms, and Symbols
Index

Copies of the manual may be requisitioned through your Sperry representative.

LIBRARY MEMO ONLY	LIBRARY MEMO AND ATTACHMENTS	THIS SHEET IS
To Mailing Lists AC, BZ, CZ (less DE, GZ, HA), MZ, 8, 30, 38, 62, MBR, MI00, and MI40.	Library Memo plus UP-9955 (283 pages and cover) to Mailing Lists DE, GZ, HA, and 82.	Library Memo for UP-9955
		RELEASE DATE: April 1984

Type 3065
Processor and Storage

Reference

SPERRY

This document contains the latest information available at the time of preparation. Therefore, it may contain descriptions of functions not implemented at manual distribution time. To ensure that you have the latest information regarding levels of implementation and functional availability, please consult the appropriate release documentation or contact your local Sperry representative.

Sperry reserves the right to modify or revise the content of this document. No contractual obligation by Sperry regarding level, scope, or timing of functional implementation is either expressed or implied in this document. It is further understood that in consideration of the receipt or purchase of this document, the recipient or purchaser agrees not to reproduce or copy it by any means whatsoever, nor to permit such action by others, for any purpose without prior written permission from Sperry.

Page Status Summary

Section	Pages	Update
Cover/Disclaimer		
PSS	1	
Preface	1 - 2	
Contents	1 - 10	
Section 1	1 - 11	
Section 2	1 - 87	
Section 3	1 - 83	
Section 4	1 - 24	
Section 5	1 - 28	
Appendix A	1 - 22	
Appendix B	1 - 5	
Appendix C	1 - 3	
Index	1 - 5	
User Comment Sheet		

Section	Pages	Update

Preface

The Type 3065 central complex components described in this manual are required components in some smaller Series 1100 Systems (for example, the Distributed Data Processing System 11).

This manual provides hardware-oriented information for the following Type 3065 central complex components: Instruction Processor, Channel Input/Output Processors, and Main Storage Unit. This manual also includes information for the integrated Streaming Tape Subsystem.

This manual is intended for system programmers and system analysts who are familiar with similar equipment.

This manual consists of:

SECTION	CONTENTS
1	Introduction Presents an overview of the central complex components, and provides illustrations of the standard system configurations.
2	Instruction Processor Describes general operation, including: processor state, storage structures, processor operation, interrupt processing, classes, and status.
3	Channel Input/Output Processors Describes input/output operations, including: I/O order code initiation and operation, channel command word operation, and status tabling operation.
4	Main Storage Unit Describes general main storage operation, including: functional characteristics, system bus, addressing modes, and storage instructions. Error reporting, error detection, and error handling are also described.

SECTION CONTENTS

Documents referenced in this manual are:

- *Series 1100 Assembly Instruction Mnemonics (AIM), Supplementary Reference,* UP-9047 (applicable version*)
- *System Support Processor (SSP), Operator Reference,* UP-9123 (applicable version*)
- *8436 Disk Subsystem, Reference,* UP-10058.

** Use the version needed for the software level in use at your site.*

Contents

Figures

Tables

1. Introduction

1.1. Functional Description

This system is organized around two busses: a System bus (S-bus) and a byte-multiplexing bus. The S-bus is a high-performance, word-multiplexing storage bus that provides the main data and control paths for the Instruction Processor (IP), Channel Input/Output Processors (Channel IOPs), Main Storage Units (MSUs), and System Support Processor (SSP). The byte-multiplexing bus is an 8-bit input/output (I/O) bus that supports a set of integrated peripheral control units and integrated line modules. Figure 1-1 shows the general arrangement of the system components.

1.1.1. Instruction Processor

The IP is a 36-bit word general purpose processor. It has four subsystems: a processor module, a unit control module, a power system, and a cooling system. The power system and cooling systems support the processor module and the unit control module. The processor module has an instruction buffer and an operand buffer that are used for internal instruction fetch and operand fetch. The unit control module is the interface between the IP and the SSP, and between the IP and the system panel.

The IP capabilities include:

- a 128-word General Register Set (GRS);
- 32 Base (B) registers for addressing program and system storage banks;
- a program (logical) address range of 68,719,476,740 words;
- user program compatibility with previous Series 1100 processors;
- additional instructions and optional instruction format changes to use the addressing space expansion;

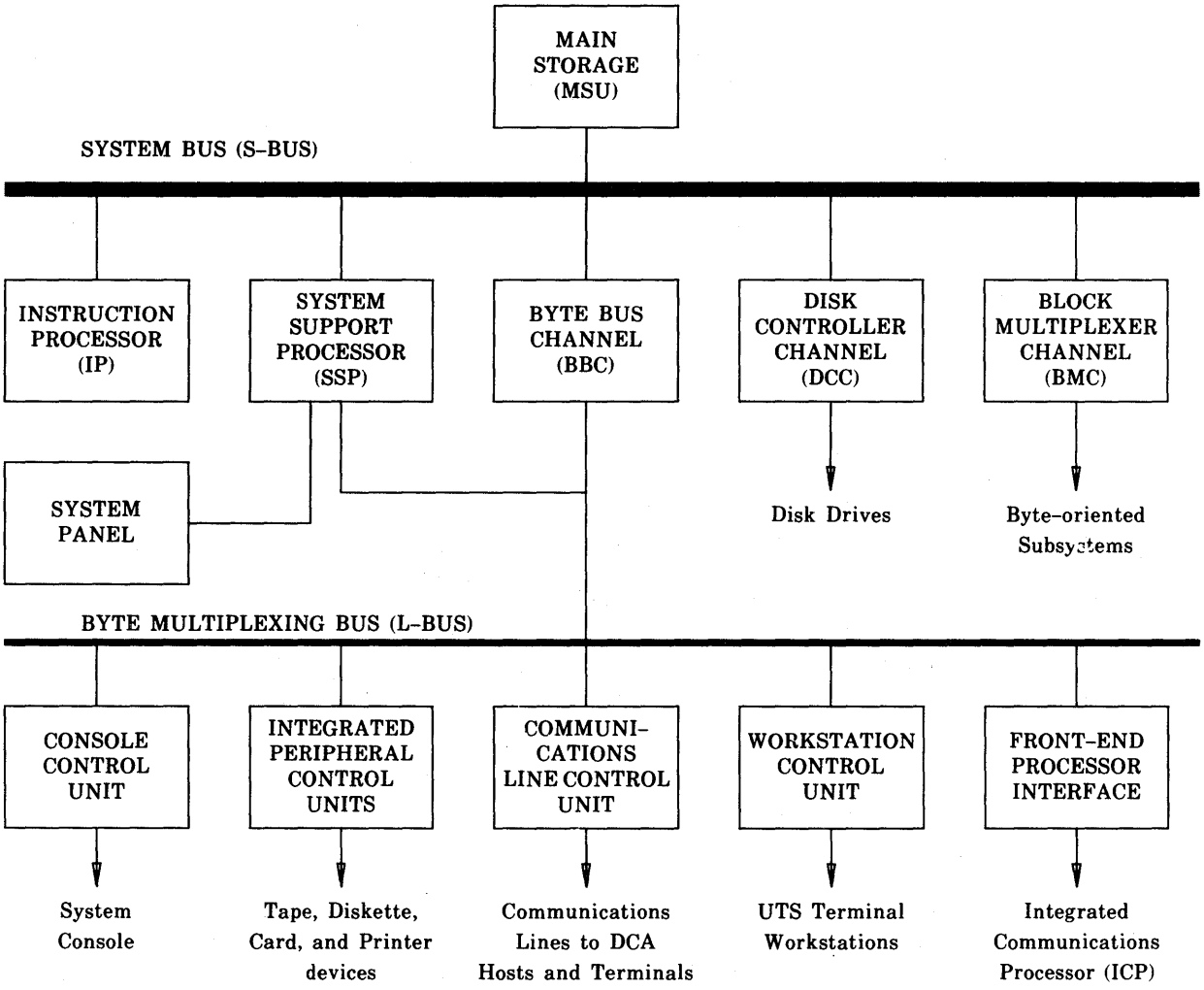


Figure 1-1. General System Block Diagram

- security features for the separation and protection of information in the system;
- program storage and protection mechanisms that control the accessing and addressing of that storage;
- a basic mode instruction set and a remapped (extended mode) instruction set;
- access control mechanisms to ensure that only authorized activities are permitted to obtain or modify the contents of a bank; and
- a physical address range of 16 million words.

The basic mode of operation executes the standard Series 1100 Systems programs, while the extended mode supports the new hardware features. The extended mode has 32 Base (B) registers. Sixteen are for users and sixteen are for Executive use. Twenty-four bit relative addressing is provided in extended mode only. Absolute addressing is not provided in extended mode, but it is provided in basic mode.

The IP has binary and decimal arithmetic sections that manipulate arithmetic and logical data, and perform other decision making procedures required to execute instructions. There is also a high-speed multiplication section in the binary arithmetic section.

The IP's interface to other components in the central complex is through the S-bus. The IP communicates with the MSU, Channel IOPs, and the SSP through the S-bus. The IP also has a Universal Processor Interface (UPI) that provides a standard interface for communications between active components in the central complex.

The S-bus provides 36-bit word parallel data and message exchanges between the IP and the MSU, Channel IOPs, and the SSP. The IP has two connections (drops) to the S-bus lines for 36-bit information flow. The IP can interface with up to four MSUs.

1.1.2. Channel Input/Output Processors

There are three Channel IOPs: a Disk Controller Channel (DCC), a Byte Bus Channel (BBC), and a Block Multiplexer Channel (BMC). The DCC supports word-oriented, high-speed I/O operations, and performs a combination of system channel and device control functions. The BBC interfaces byte-oriented, low-speed I/O operations equipment. It will support byte-wide I/O line busses that support a set of integrated peripheral control units and integrated communications equipment; such as a Distributed Communications Processor (DCP). The BMC supports freestanding peripheral subsystems, operating on a block multiplexer channel. Each of these units is a requester on the S-bus. I/O control and data exchange with the IP is through main storage.

The Channel IOP capabilities include:

- integrated I/O hardware (except peripheral devices) in the central complex cabinet;
- block multiplexer channel interface supported as a standard external I/O interface to provide connection for very large peripheral configurations;
- I/O expansion of one channel at a time at the storage interface without requiring intervening I/O-type multiplexing hardware to combine multiple channels into a single storage interface;
- disk controller channels that look like word channels to the host system;
- integrated communications support that includes the function of front-end processors of the DCP family;
- block multiplexer channels that support device path selection, status tabling, transparent control unit busy, command chaining, and data chaining I/O operations;
- block multiplexer channel data support for formats A and C;
- automatic sense information retrieval for devices on block multiplexer channels; and
- dual access to the device level supported on the disk controller channels and block multiplexer channels.

The interface between the Channel IOPs and other components of the central complex is through the common S-bus. The channels communicate with the MSU, the SSP, and the IP through the S-bus. The Channel IOP interfaces are preemptive ports and have priority over IP interfaces at the MSU.

A universal processor interface exists between each pair of active IPs and Channel IOPs, and consists of a send request-acknowledge pair and a receive request-acknowledge pair. A Channel IOP has an internal universal processor interface consisting of a single request-acknowledge pair.

Each of the Channel IOPs is described below:

- Disk Controller Channel (DCC)

  Each DCC provides direct data path and control for up to 16 disk storage units and the MSU through the S-bus. The DCC is a microprogrammable unit that operates as a combined word channel and a disk controller in the same hardware. Each DCC occupies the space of a single printed-circuit card in the S-bus.

  The DCC is logically equivalent to a combination input/output processor, Internally Specified Index (ISI) channel, and a disk control unit. The DCC performs disk I/O operations in support of software executing in the IP.

  The DCC's disk command set, command formats, and sense formats are compatible with a subset of the SPERRY 5056 Disk Control Unit command set. Each DCC contains a 4096-word internal data buffer that is used to minimize system data overruns.

- Byte Bus Channel (BBC)

  The BBC performs a byte multiplexer function for communications equipment and low-speed I/O devices. Each BBC provides I/O data paths and control for a combination of communications control units, peripheral control units, and a front end processor interface. Each BBC references main storage through the S-bus and interfaces with control units through the byte multiplexing bus (L-bus).

  The BBC supports the following integrated control units:

  - tape control unit,
  - printer control unit,
  - card reader control unit,
  - diskette adapter,
  - workstation control unit,
  - communications line control unit, and
  - an Integrated Communications Processor (ICP).

  The BBC is similar to a Series 1100 I/O block multiplexer channel. (It can also be viewed as a channel with L-bus control units operating under block multiplexer subchannel protocols.) Data transfers are not in blocks from each L-bus unit, instead they are on a multiplexed byte basis. Each BBC operates under IP control, but may alternately be controlled by the SSP software when the Executive System is not in control of the system.

  The BBC performs byte-to-word assembly/disassembly in formats A (quarter word) and C (8-bit packed). Format B (6-bit packed) is not supported. A subchannel control word set controls the operation of this channel.

- Block Multiplexer Channel (BMC)

  The BMC provides the standard I/O bus for freestanding subsystems external to the central complex. Any freestanding peripheral subsystem, supported by the host system, that is capable of operation in block multiplexer mode on a compatible channel interface may be attached to this channel. The BMC supports:

  - high performance tape and disk subsystems,
  - freestanding distributed communications subsystems, and
  - dual-access subsystems.

  Each BMC provides a single I/O bus interface for up to two freestanding subsystems. Each of the BMCs is a separate unit on the S-bus.

  The BMC performs I/O operations under control of the IP software. A microprocessor controls the channel hardware operation and provides a block multiplexer functional capability with 256 nonshared subchannels. Byte multiplexer operations are not supported on this channel.

  The BMC supports byte-to-word assembly/disassembly in formats A and C (format B is not supported). Each channel supports command retry operations, but does not support truncated search operations.

1.1.3. Main Storage Unit

The MSU consists of control logic and main storage printed-circuit cards mounted in the central complex backpanel. Each main storage card has a capacity of 524,288 words (2,097,152 bytes).

Main storage requests are broadcast on the system bus to all MSUs. An individual MSU responds when the address on the S-bus falls within its address range. A register establishes the MSU's starting address and is loaded by the SSP when it initializes the system. Thus the SSP software controls main storage address assignments, including the ability to configure contiguous physical addresses between MSUs up to and including the maximum storage capacity of the system.

The MSU provides error detection and correction of single-bit data errors and detection of double-bit data errors (also all even multiples and some odd multiples). The MSU also detects single-bit address errors and provides internal through checking on the address and data paths.

1.1.4. System Support Processor

The SSP is the hardware focal point of the system. It accomplishes no part of the customer's workload, yet is required for both system initialization and operation. The SSP allows substantial amounts of system control and maintenance operations to be implemented through programming techniques. The SSP controls the hardware during hardware initialization and the Series 1100 Operating System (1100 OS) initial program load. After the 1100 OS is operational, the SSP reverts to a backup role whose function is invoked only when a system fault occurs that the 1100 OS cannot handle. If this happens, the SSP intercedes to provide increased fault tolerance.

The SSP is a microprocessor printed-circuit assembly installed in the S-bus. It consists of:

- 131,072 bytes of Random Access Memory (RAM) storage with error correction code,
- S-bus and L-bus interface logic,

- time-of-day clock logic,
- auto-recovery timer,
- serial scan/set control logic, and
- power control and system panel interface.

The SSP is part of a set of ancillary hardware, called the support complex. The support complex performs hardware system control, maintenance, and man-machine interface functions in support of the processing complex. Refer to the SSP Operator Reference, UP-9123 (see Preface).

1.1.5. System Bus

The S-bus is the primary data path between central complex components. This 36-bit data interface is used for main storage references, universal processor interface messages, system control messages, error status transfers, initialization, and test. The S-bus is the interface between the IP, disk controller channel, block multiplexer channel, byte bus channel, main storage, and the SSP.

The S-bus can transfer data every bus cycle. This is accomplished by overlapping bus request, information transfer, and acknowledge in a three-deep pipeline operation. Each unit interface on the S-bus will latch and test the information on the bus every busy cycle. When a unit matches the destination bus address with its own unit identifier, the unit responds by activating the common acknowledge line and placing bus transfer status information on the appropriate bus lines.

Two types of bus addressing are provided, direct and indirect. Direct addressing is used when a sending unit predetermines that there is a single destination unit for its information, as when an MSU returns read data to the specified unit that requested that data. Indirect addressing is used when the sending unit cannot predetermine which one of several units is the correct single destination, as occurs for main storage references where the address and function and/or command are broadcast to all MSUs.

The bus uses a fixed priority evaluation scheme. Each unit drop contains priority logic, with all unit priority logic operating in parallel. Each unit determines when it has access to the bus without need for a central priority arbitrator.

Parity is checked on all information and addresses transferred on the S-bus. Acknowledge timeouts are used to detect errors that result in improperly routed bus transfers. S-bus hang conditions that can be caused by faults in any units request or acknowledge logic are detected by the SSP.

1.1.6. Byte Multiplexing Bus

The byte multiplexing bus (L-bus) provides a full-duplex byte-multiplexing interface between the central complex and various peripheral and communications controllers that are physically integrated into the central complex cabinet. Both the byte bus channel and the SSP provide L-bus interfaces.

A multiple level priority scheme is used, arbitrated by the L-bus's parent channel (byte bus channel or SSP). Each L-bus unit has three levels of request priority: high, medium, and low. When two units access the bus with identical request priorities, preemptive priority takes precedence based on drop address.

1.1.7. Integrated Streaming Tape Subsystem

The streaming tape has two modes of operation, start-stop and continuous tape motion. In the start-stop mode, it functions as a conventional drive, stopping after each block of data has been transferred. In the continuous tape motion mode, the tape moves continuously at high speed, passing over the interblock gaps. In the event that a command overrun occurs while operating in continuous motion mode, the automatic tape repositioning time (recovery time) is approximately 1.5 seconds. Data is recorded on 9-tracks at 1600 bpi (phase encoded) at speeds of 25 ips (start-stop mode) and 100 ips (continuous tape motion mode).

This subsystem consists of an Integrated Tape Control Unit (ITCU) that is inserted into the L-bus of the central complex cabinet. It interfaces to a formatter unit via the 8-bit bi-directional byte interface for magnetic tapes.

Up to two streaming tape drives can be housed in a peripheral cabinet and up to two streaming tape drives may be attached to each ITCU.

1.2. Instructions

The instructions are listed in Appendix A by function code with a brief description of each instruction. For a detailed explanation of each instruction, refer to the AIM Supplementary Reference, UP-9047 (see Preface). A cross reference by mnemonic versus function code is given in Appendix B.

1.3. Major System Components

Table 1-1 lists the major system components, along with their minimum and maximum quantities for each cabinet.

Table 1-1. Major System Components

Unit	Basic Cabinet		Expansion Cabinet	
	Min	Max	Min	Max
Instruction Processors	1	1	0	1
Main Storage Units*	1	2	0	2
System Support Processor	1	1	0	0
Byte Bus Channels	1	1	0	1
Disk Controller Channels	1	2	0	3
Block Multiplexer Channels	0	1	0	1

* *This table considers a Main Storage Unit (MSU) as the MSU control and one or two expansion modules. Every MSU control must have a minimum of one expansion or a maximum of two expansions. Four MSU control units are configurable with each containing 1024KW (two expansion modules) of storage.*

1.4. Standard System Configurations

Figures 1-2 through 1-5 show four standard system configurations.

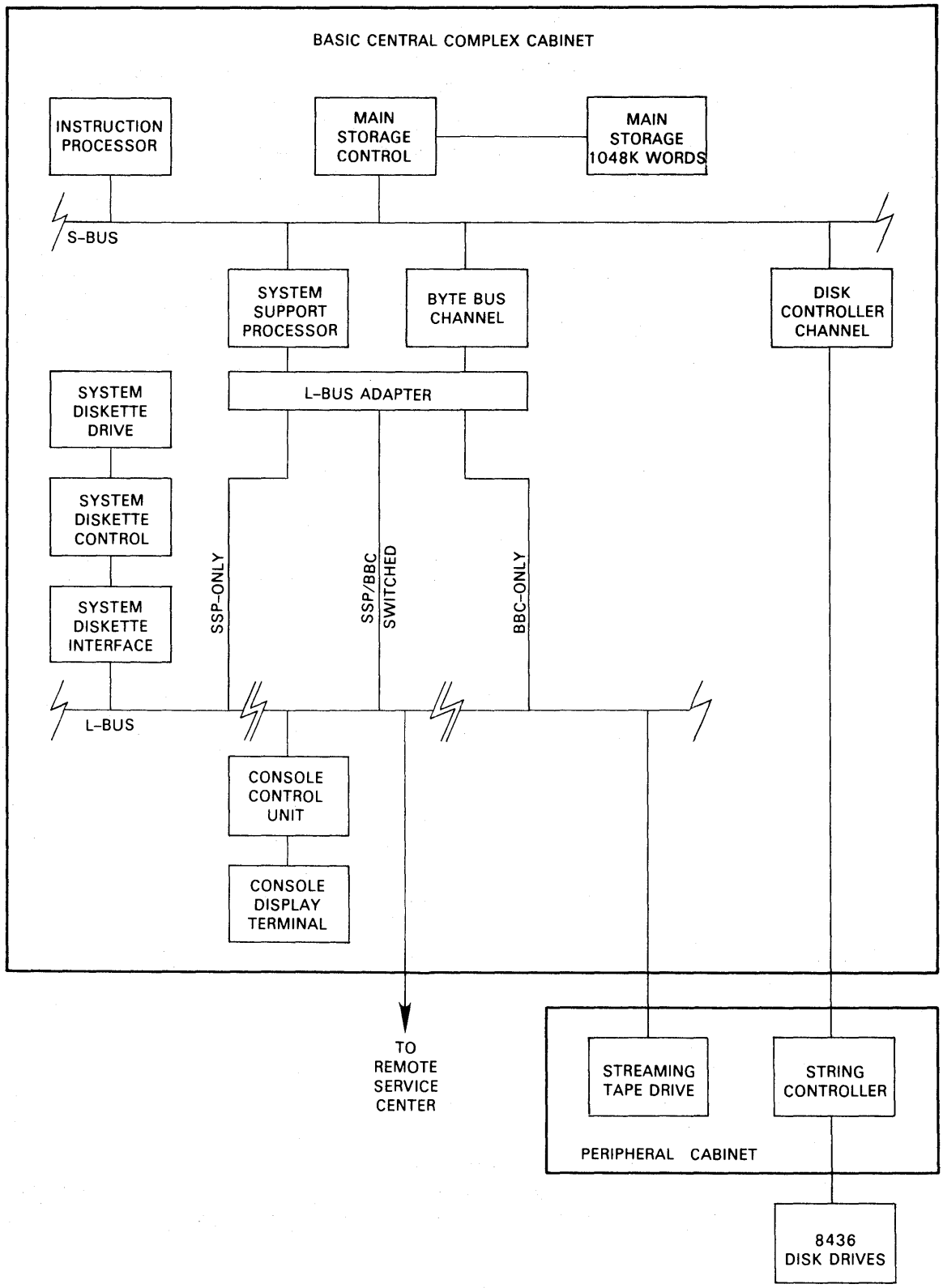


Figure 1-2. Standard Small System Configuration

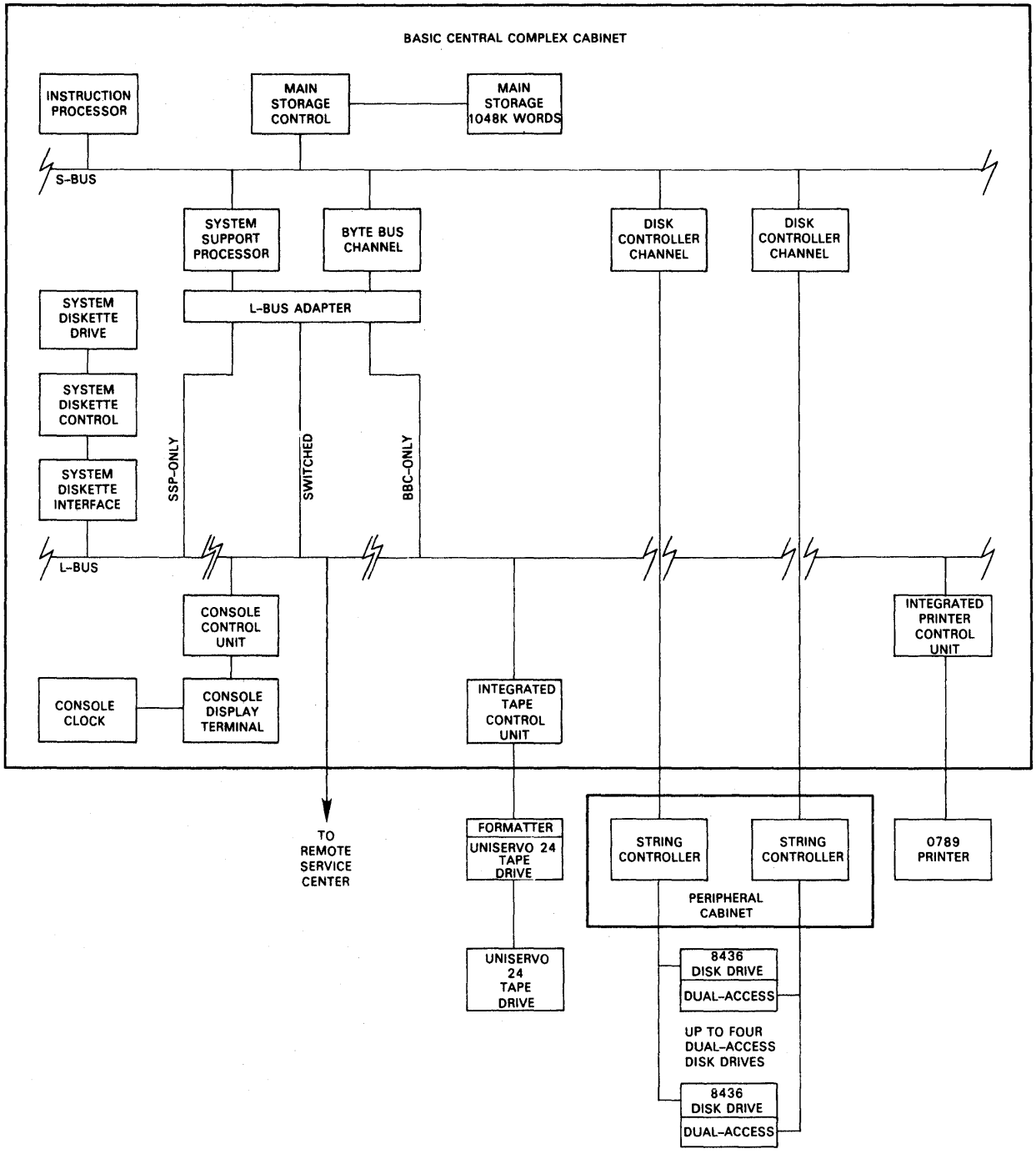


Figure 1-3. Standard Medium System Configuration

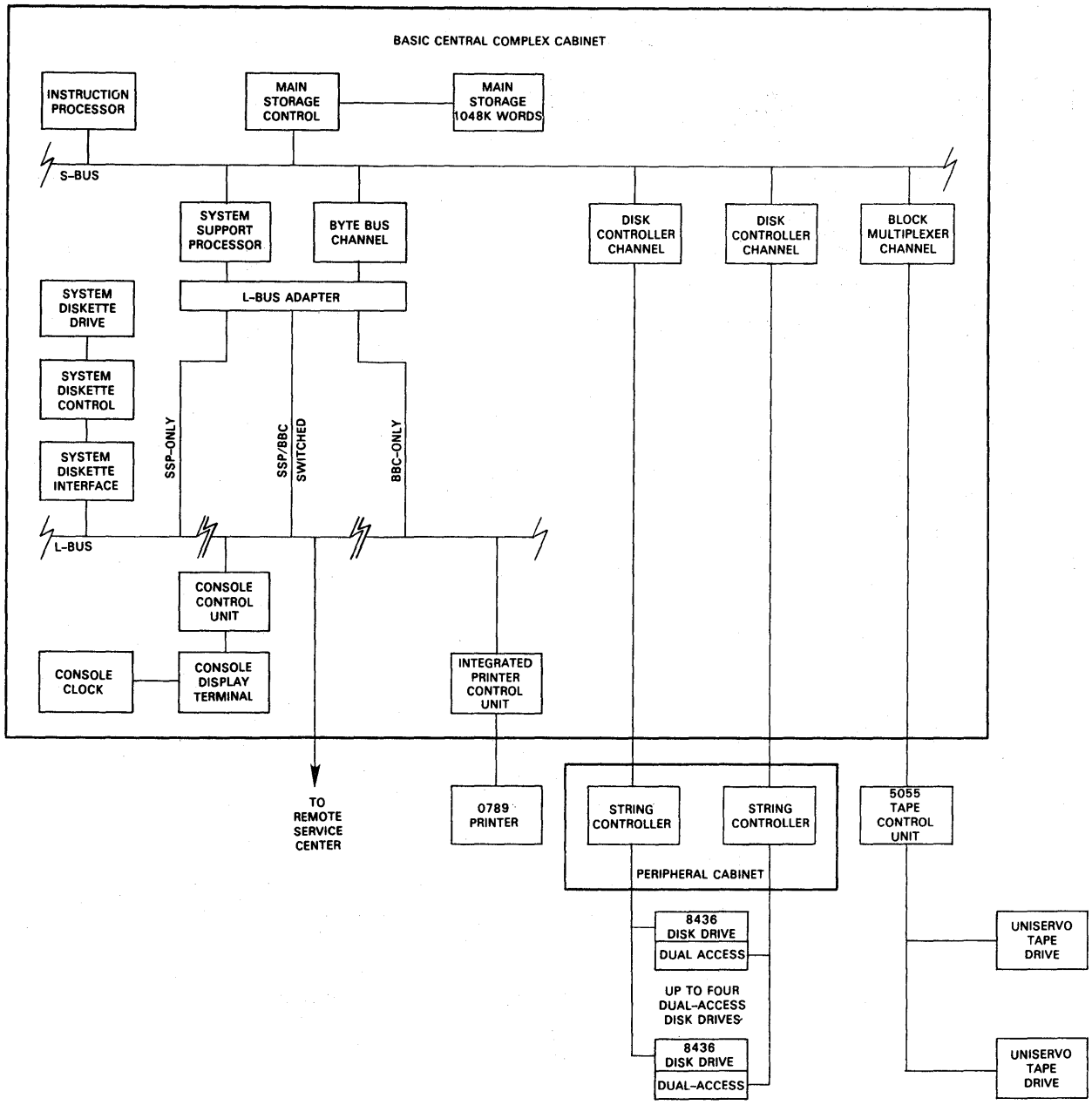


Figure 1-4. Standard Large System Configuration

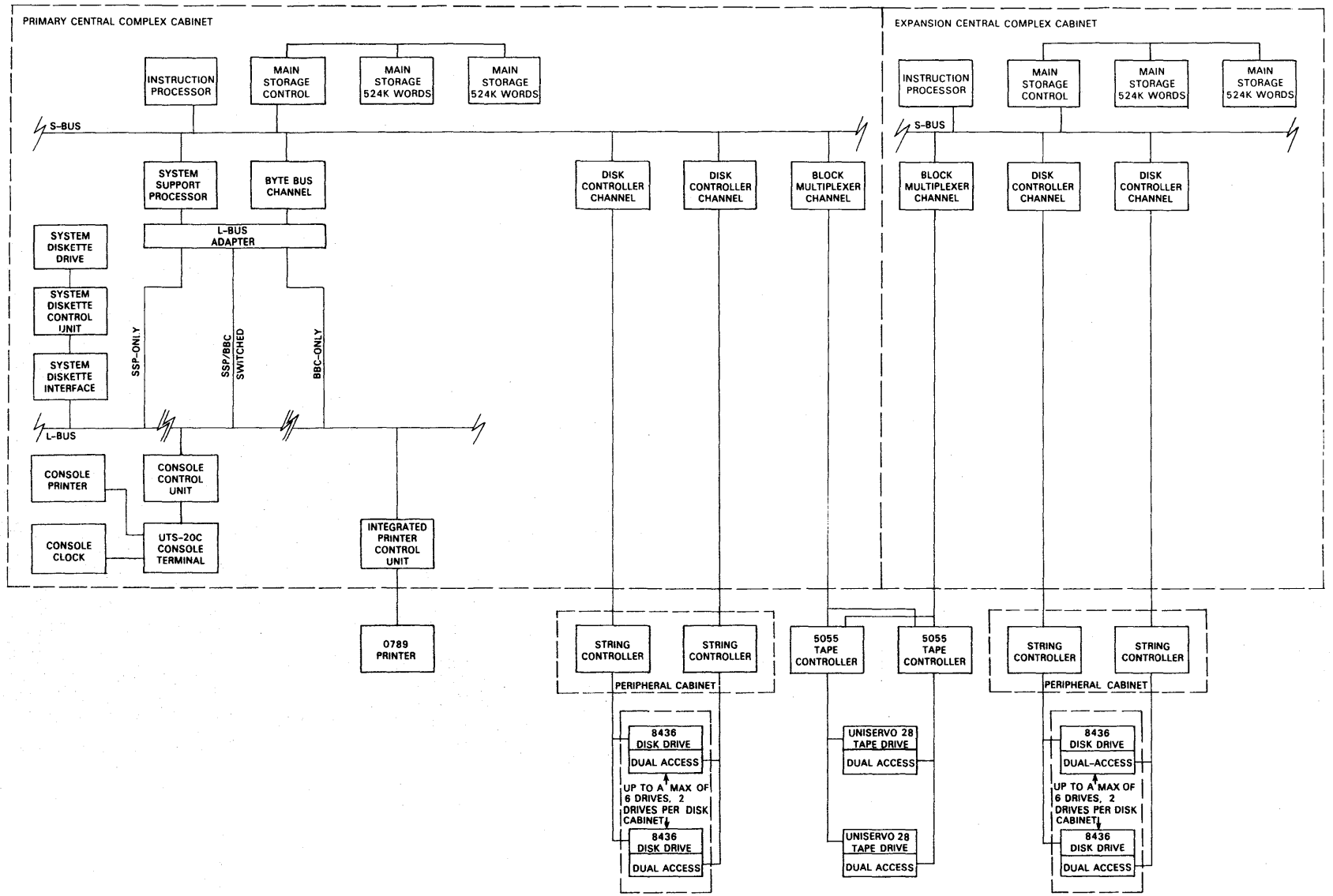


Figure 1-5. Standard Multiprocessor System Configuration

2. Instruction Processor

2.1. General

The Instruction Processor (IP) performs the logical, arithmetic, and instruction sequencing operations in the system.

The IP has two modes of operation: basic mode and extended mode. Basic mode addressing is program compatible with earlier Series 1100 systems. Extended mode is used with the many new hardware features, including an expanded visible address space, additional programmable registers, and new instructions. Basic or extended mode operation is selected by a designator register bit.

2.2. Operands

Each instruction occupies a single 36-bit word of storage and is capable of generating a single storage address that is used either to select an operand in storage, or develop an operand immediately. Operands in storage can be part of a word, a single word, a double word, or a block of words. The storage-to-storage instructions can generate multiple storage addresses. The second operand for all other instructions is selected, if required, from the General Register Set (GRS) of storage locations.

Operands may be of two lengths, depending on the instruction: single precision or double precision. Single-precision operands are 36 bits long and occupy either a single storage or GRS location. Double-precision operands are 72 bits long and occupy consecutive storage or GRS locations. For some instructions, the operand address can be used as a single-precision operand, and the single-precision operand can be developed from the contents of a portion of the storage location selected by the instruction.

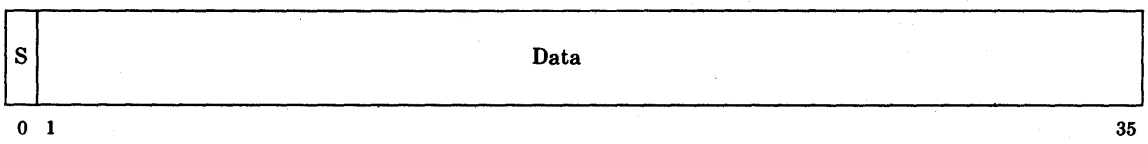
2.2.1. Data Formats

The IP operates on a set of single-precision (36 bits), double-precision (72 bits), fractional (12 or 18 bits), and character (6, 9, 12, or 18 bits) data types. Partial-word and character operands are expanded to fixed-point single-precision data prior to their use in computations. A list of single-precision and double-precision data types and their format is given in the following subsections. A negative value has a sign value of 1 and is the ones complement of the corresponding positive number. Fixed-point binary computations are done as integral operations. In each diagram, the location of the sign bit or bits is indicated by an "S".

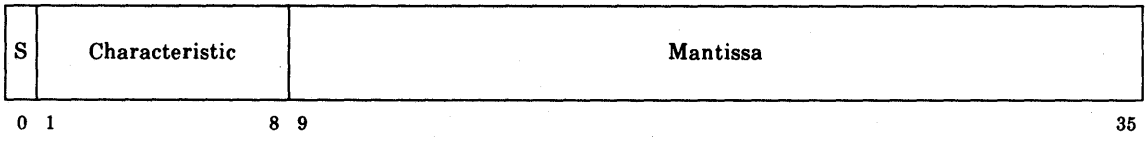
2.2.1.1. Single-Precision Binary

The single-precision binary formats are:

Fixed-Point



Floating-Point

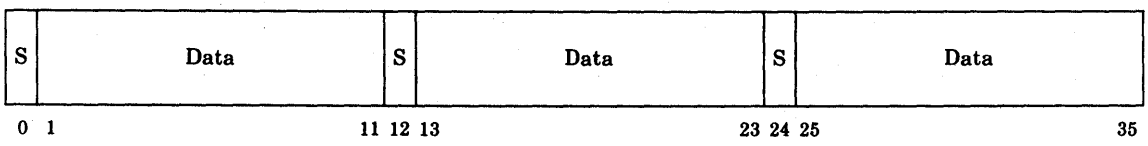


2.2.1.2. Fractional-Precision Binary

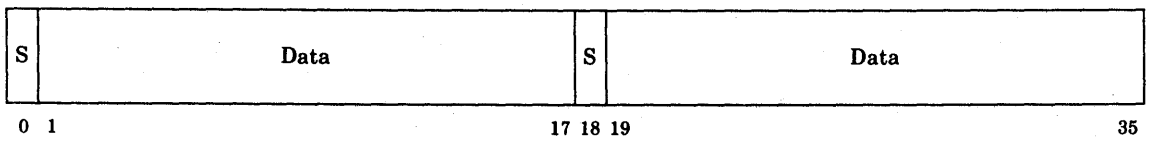
The IP provides a set of fixed-point add and add-negative instructions that manipulate the fractional-precision binary data formats. The operations are performed in parallel on portions of the word. The sign bit is the leftmost bit of each portion.

The fractional-precision binary formats are:

Third-Word



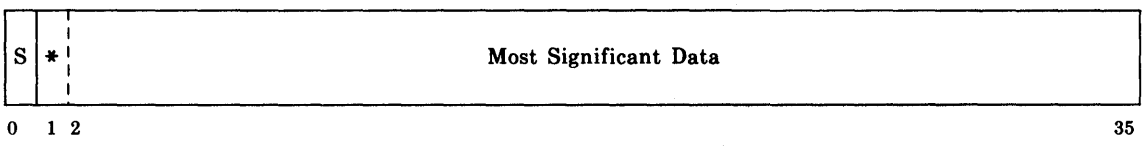
Half-Word

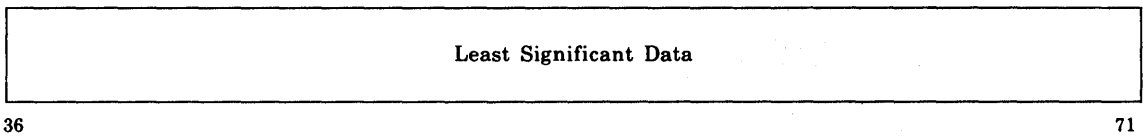


2.2.1.3. Double-Precision Binary

The double-precision binary formats are:

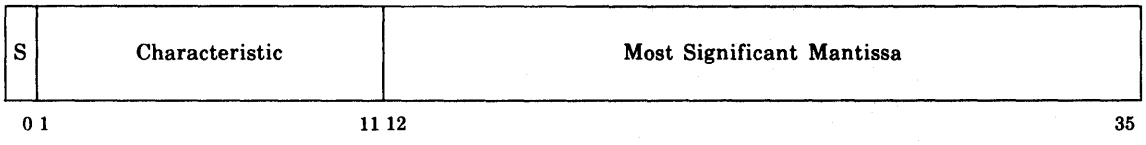
Fixed-Point

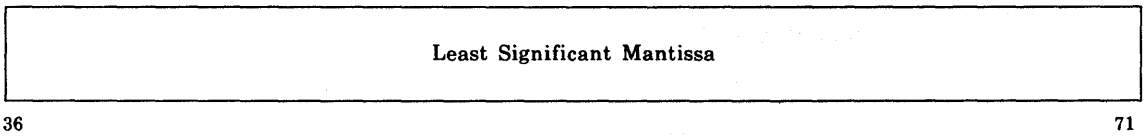




** A double sign bit is the result of Multiply Integer.*

Floating-Point



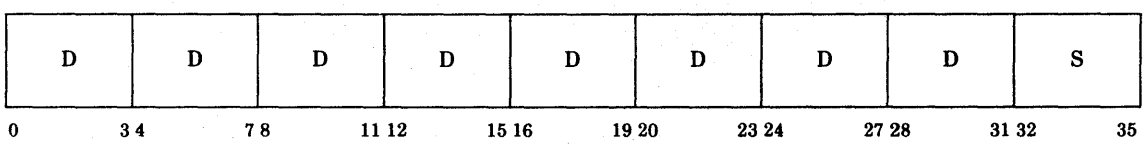


2.2.1.4. Decimal Data

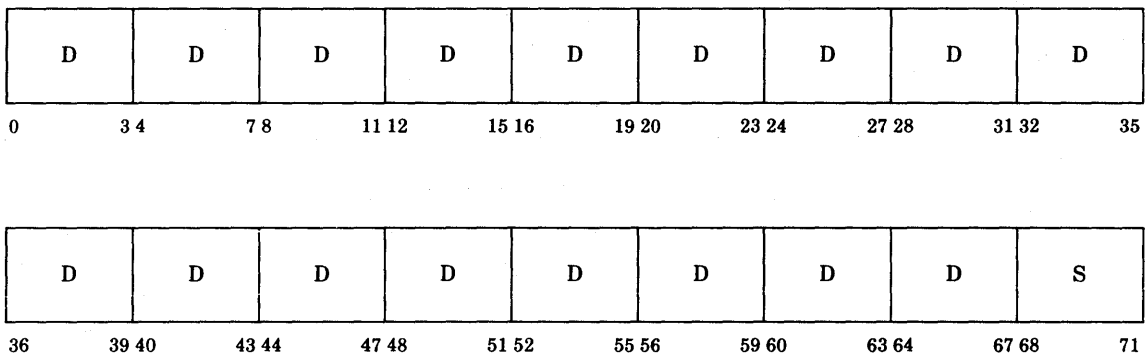
The format of decimal data is Binary Coded Decimal (BCD) using absolute value and sign, also called magnitude and sign representation. Each decimal digit occupies four bits with decimal digits 1 through 9 being represented by the binary values 0000 through 1001. The sign occupies the least significant four bits of the highest addressed word of the data. Decimal data occupies only full words, with single- double- and triple-precision formats supported.

The decimal data formats are:

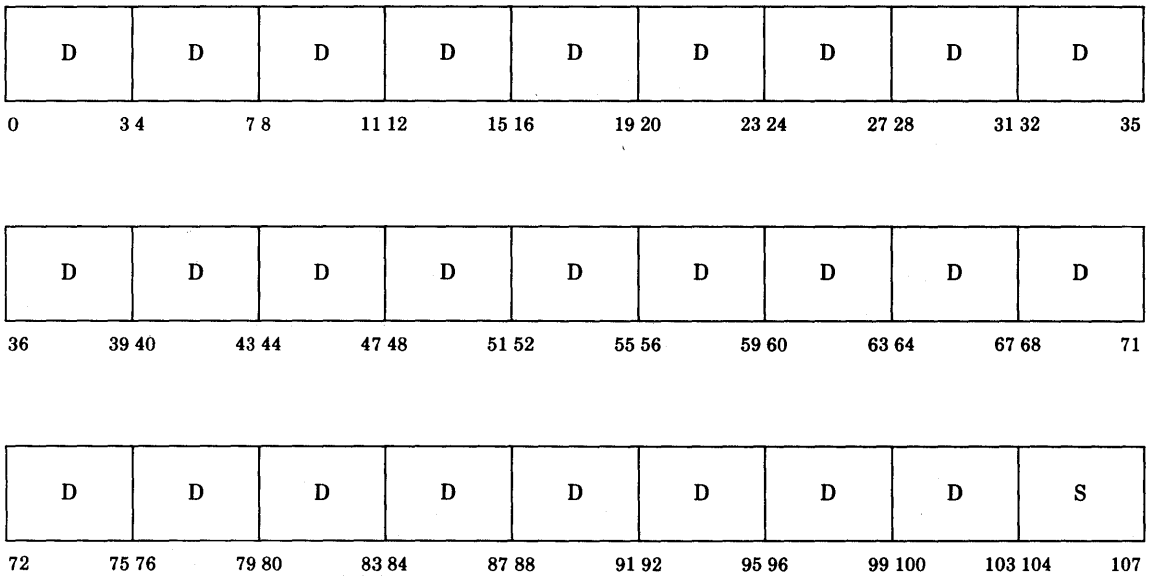
Single-Precision Decimal Format



Double-Precision Decimal Format



Triple-Precision Decimal Format



D - single decimal digit in binary representation 0000 through 1001

S - sign code

The octal sign codes accepted on input for positive signs are 0_8, 2_8, 4_8, 6_8, 10_8, 12_8, 14_8, 16_8, and 17_8; and for negative signs are 1_8, 3_8, 5_8, 7_8, 11_8, 13_8, 15_8. The only sign codes generated as a result of an operation on decimal data are 14_8 for plus, and 15_8 for minus. A zero result is always stored with a positive sign unless overflow occurs.

2.3. General Register Set

The General Register Set (GRS) is the set of storage locations with addresses 0000_8 through 01778_8. The GRS locations are divided into a jump history, a user set, and an Executive set. The user set and the Executive set are each divided into three 16-register subsets according to their uses and capabilities. Within each subset, the registers are numbered sequentially from 0-15. The A-registers are capable of providing input for and accepting results from arithmetic

operations. The X-registers are index registers and are used to modify the u-field of an instruction. Index registers 12-15 occupy the same GRS locations as arithmetic registers 0-3. These overlapped registers can both index and perform as arithmetic registers. The R-registers have a variety of purposes including repeat count for the Search and Block Transfer instructions and masking for the Mask Search instructions.

Registers are addressed using two methods. Both are available to most instructions, and exceptions are noted:

- The location of the register ($0-177_8$) can be the operand address of an instruction.
- The register location within a subset can be specified in an instruction a- or x-field. The user or Executive register set is determined by the setting of the EXEC Register Set bit Selection Designator Bit, DB17. When selecting a register using the a-field of an instruction, the subset is indicated by the function code of the instruction; for example, a Load A (LA) instruction selects one of the 16 A-registers using the a-field. The x-field always selects one of the 16 index registers, with the exception of an x-field value of zero, which specifies no indexing.
- The Jump Greater and Decrement (JGD) instruction uses the j- and a-fields together to select one of the GRS locations.

When the relative operand address, after indirect addressing, is less than 200_8 and the instruction permits operands in the GRS, relative to absolute address translation is not performed and the operand address selects a GRS location.

If the value of the a-field is 17_8 and the instruction uses more than one arithmetic register (A_{a+1} or A_{a+2}), these registers are located at GRS locations 34_8 and 35_8, or 174_8 and 175_8, depending on the value of DB17.

Rules concerning conflicts, when A_a, A_{a+1}, A_{a+2} are required and select the same physical GRS location as X_x, are noted as part of the instruction description.

Table 2-1 shows the GRS register assignments and the access to the GRS allowed by the Processor Privilege (PP) designator bits (DB14 and DB15).

Table 2-1. GRS Layout with Processor Privilege Designator

Address (octal)	Read Access Allowed	Write Access Allowed	Register
0000–0007	PP ≤ 3	PP ≤ 3	User X0–X7
0010–0017	PP ≤ 3	PP ≤ 3	User X8–X11; X12/A0–X15/A3
0020–0027	PP ≤ 3	PP ≤ 3	User A4–A11
0030–0037	PP ≤ 3	PP ≤ 3	User A12–A15; A15+1, +2, +3, +4
0040–0047	PP ≤ 2	PP = 0	JH words 0–7
0050–0057	PP ≤ 2	PP = 0	JH words 8–15
0060–0067	PP ≤ 2	PP = 0	JH words 16–23
0070–0077	PP ≤ 2	PP = 0	JH words 24–31
0100–0107	PP ≤ 3	PP ≤ 3	User R0–R7
0110–0117	PP ≤ 3	PP ≤ 3	User R8–R15
0120–0127	PP ≤ 2	PP = 0	Executive R0–R7
0130–0137	PP ≤ 2	PP = 0	Executive R8–R15
0140–0147	PP ≤ 2	PP = 0	Executive X0–X7
0150–0157	PP ≤ 2	PP = 0	Executive X8–X11; X12/A0–X15/A3
0160–0167	PP ≤ 2	PP = 0	Executive A4–A11
0170–0177	PP ≤ 2	PP = 0	Executive A12–A15; A15+1, +2, +3, +4

2.3.1. Index (X) Registers

The index (X) registers provide the programmer with address modification capability (indexing).

An index register contains a modifier field (X_m) that modifies the operand address (indexing), and an increment field (X_i) that modifies the modifier field (automatic incrementation). There are two register formats: 18-bit format and 24-bit format. In 24-bit format, X_m is the lower (least significant) 24 bits of the index register (bits 12–35), and X_i is the upper 12 bits of the index register (bits 0–11). In 18-bit format, X_m is the lower 18 bits of the index register (bits 18–35), X_i is the upper 18 bits of the index register (bits 0–17).

Index register zero (X0) has indexing capability when selected by the a-field of the Block Transfer (BT) instruction. It does not have indexing capability when addressed by the x-field of an instruction. It may also be used as a general-purpose register.

2.3.2. Arithmetic (A) Registers

The arithmetic (A) registers provide intermediate storage for arithmetic operands and results. To the programmer, the A-registers effectively function as accumulators. Any two or three consecutively addressed accumulators can hold double- or triple-length operands.

Four of the A-registers are also assigned as index registers. Their dual role provides the IP with additional flexibility because it permits the result of a given operation to be readily used for address modification in a following instruction.

2.3.3. Special (R) Registers

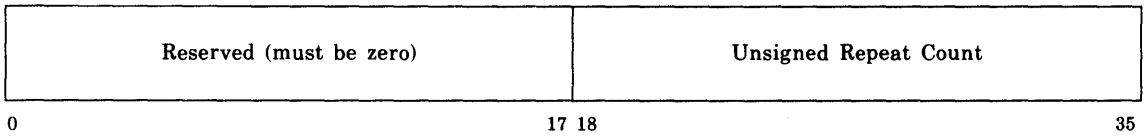
For some instructions, the special registers are referred to as R-registers. Two of the R-registers serve special purposes and are the repeat count register (R1) and mask register (R2). The remaining R-registers are not specifically assigned and may be used as temporary storage locations.

2.3.3.1. Repeat Count Register (R1)

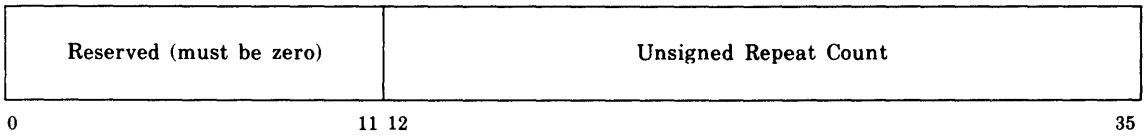
The contents of the repeat count register define the maximum number of times a repeated instruction is executed. During execution of a repeated instruction, the contents of the lower portion (bits 18–35) of the repeat count register is decreased by one each time the repeated instruction is executed. The number of bits in the repeat count varies depending on whether the processor is executing in basic or extended mode. If an interrupt occurs during the sequence of repeated executions of an instruction, the repeat sequence is suspended to process the interrupt, and the current count is left in R1. The repeated sequence may be resumed after the interrupt has been processed. The final value of the count after the repeat sequence terminates is always available in R1. If the contents of the repeat count register is 0, the repeated instruction is not executed and the execution of the next instruction is initiated.

The R1 register formats are:

Basic Mode R1



Extended Mode R1



The contents of R1 is undefined during the execution of a repeated instruction. Upon interruption or termination, R1 is updated to reflect the progress or completion of the instruction.

2.3.3.2. Mask Register (R2)

The bits in the mask register specify the fields of operands to be operated upon in certain instructions. A logical AND is performed with the operand and the mask register and/or its complement. The portions of the operand so selected are used in the instruction operation.

2.3.4. Register Selection Designator

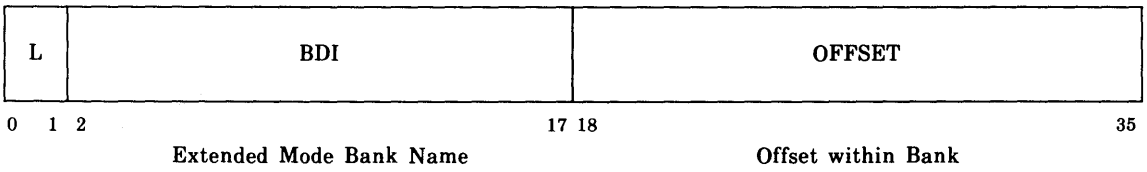
Among the 128 addressable general registers is a set of registers referred to as A-, X-, and R-registers that are designated for use by the user. Another set of A-, X-, and R-registers are designated for use by the Executive. The designator register bit 17 (DB17) defines which set of A-, X-, and R-registers is addressed by the a- and x-fields of an instruction. When DB17 is 0, the a- and x-fields of an instruction reference the set of A-, X-, and R-registers designated for use by the user. When DB17 is 1, the a- and x-fields of an instruction reference the set of A-, X-, and R-registers designated for use by the Executive. (This distinction does not hold when the operand address (U) addresses general registers, since they are directly addressed as storage locations 0000 through 0177.)

2.4. Virtual Address Space

The virtual address space is 2^{36} (about 69,000 million) words long. This allows a program to address an area that is several times larger than the physical main storage. For this reason, the address space is divided into a maximum of 262,144 containers of variable length. These containers are known as banks. Only a subset of the available banks must be residing in main storage at any one time for an individual program to proceed. Appropriate hardware assists enable Executive software to determine which banks must be resident so that it can effectively use the available physical main storage.

The location of a bank in physical main storage is unknown to the program being executed by the IP. Instead, the program expresses the address of all operands using a virtual address. Every virtual address is validated and the program's ability to access the address is checked by the IP.

The virtual address format is:



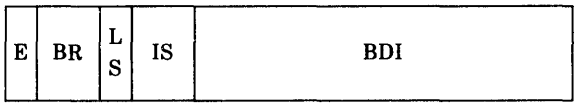
where:

Bits 0-1 — Level-field (L-field) divides the address space into four portions. See 2.4.1.

Bits 2-17 — Bank Descriptor Index-field (BDI-field) is used to locate the bank in which the information is stored. This field can be treated by the programmer as the name or identifier of the bank. Provisions are made for up to 64 bank descriptors to be grouped so that the maximum bank is 16,777K words in length. See 2.4.2.1.

Bits 18-35 — OFFSET-field locates the word being addressed within the bank selected by the BDI field.

However, not all instructions can directly express a virtual address that is 36 bits long. The Load I-Bank Base and Jump (LIJ), Load D-Bank Base and Jump (LDJ), and Load Bank Descriptor and Jump (LBJ) instructions express a virtual address (known as a basic mode virtual address) of the following format:



Basic Mode Bank Name

where:

Bit 0 — EXEC (E) bank descriptor table sector. The value of this bit, together with Level Specification-field (LS-field) field determines the L-field value of the full virtual address corresponding to this truncated virtual address.

Bits 1-2 — Base Register (BR) field is not related to the virtual address. It is used by these instructions to select the base register to be loaded. The base register number is computed by the IP by adding 12 to the value found in this field.

Bit 3 — Level Specification (LS) field is used together with the E-field to determine the L-field value according to the following table:

L	E	LS
0	1	1
1	1	0
2	0	0
3	0	1

Bits 4-5 — Interface Specification (IS) field is not related to the virtual address. It specifies processor operation at the interface between basic mode and extended mode banks. The execution mode is determined automatically by the processor from the type associated with the target bank. If this field contains the value 2, an extended mode Return (RTN) operation is performed. Other values do not effect the operation of these instructions unless the virtual address is within a bank containing extended mode instructions. In the case, when the field is 0, a CALL operation is performed and when its value is 1, a GOTO operation is performed.

Bits 6-17 — Bank Descriptor Index (BDI) field is a shortened form of the BDI-field found in a full virtual address. Four bits with the value zero are linked on the left to form a full BDI.

These instructions cannot express an OFFSET-field. The full virtual address corresponding to a truncated virtual address has an OFFSET-field value of zero.

2.4.1. Address Control

The address space is divided into four portions of equal size, called levels. These levels are assigned numbers 0 through 3 corresponding to the four possible values of the L-field in the virtual address. The IP treats those addresses with L=0 and BDI<32 specially, but otherwise the operating system software is free to assign any meaning to the four levels.

The four levels provide the operating system software with the ability to control access to shared information and isolate unrelated programs from one another. The use of the L-field by the 1100 Operating System is described in the following paragraphs.

Sharing of the information and code is controlled by assigning an appropriate L-field value to the address of the structure containing the information. Those addresses with an L-field value of zero are visible to all programs in the system. An L-field value of one indicates that the address is a member of a small set of banks that are each visible only to a set of subordinate programs. A set of banks with L=1 is known as an application group. This is because these banks are shared by a group of application programs. The programs are assigned address with L=2. Address with L=3 are set aside for storage local to the various activities (loosely, a thread of instruction execution and associated register contents) that are executing within the program. Figure 2-1 shows these various groups of addresses. Each circle represents a set of addresses with the same sharing attributes (L-field value).

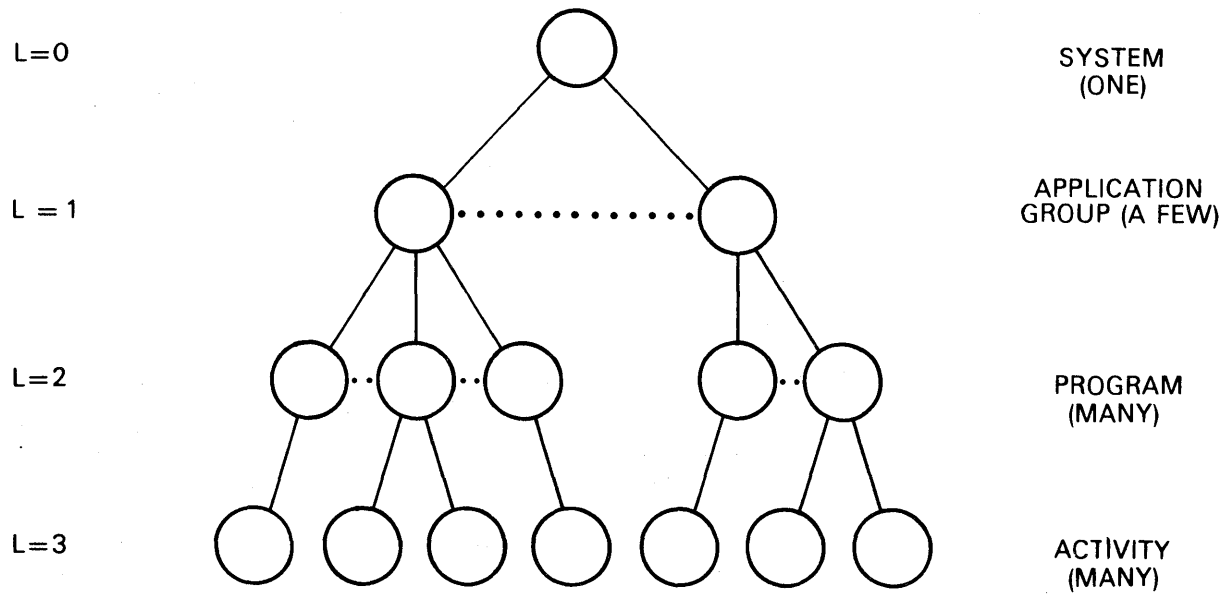


Figure 2-1. Address Tree

Figure 2-1 is generally referred to as the address tree. Each activity can access only one set of banks at each level, and the information visible to each activity and program is precisely controlled. Specifically that an activity cannot express an address local to another activity, a program cannot express an address within another program, and an application group cannot express an address within another application group. Thus, total isolation is achieved where necessary.

Programs that were constructed for previous 1100 Systems use the basic mode virtual address. These programs will always have the LS-field clear (zero). This means the virtual addresses with L=0 and L=3 are not available to these programs. Operating system software is responsible for providing an appropriate and compatible addressing environment for these programs.

2.4.2. Storage Objects

Storage objects consist of banks, bank descriptions, bank descriptor tables, and gates.

2.4.2.1. Banks

Each level of the address space is divided into groups of contiguous addresses known as banks. Under program control not all banks are used and they may vary in size up to 16,777,777 words. Since there is seldom a requirement for banks as large as 16,777,777, the virtual address is interpreted by the IP as if there were a maximum of 65,536 banks, each with a maximum size of 262,144 words. If a bank that exceeds 262,144 words is needed, Executive software can mark several adjacent banks as being each a part of a group that together describes a single bank up to the maximum of 16,777,777 words. The mechanics of this designation are invisible to the programmer.

The important concerns for the programmer are that the banks can vary in length and that there are both upper and lower address bounds associated with each bank. Every address expressed by a program is checked by the IP to make sure that it is within a set of visible banks and that it is within the address bounds of the selected bank. The specific bank within the selected level is determined by the IP using the BDI-field within the virtual address. The IP automatically handles the case where this BDI selects a portion of a bank whose site or address bounds exceeds 262,144 words, (see 2.4.2.1.3).

A single structure cannot span banks. That is, once a base address has been established (by loading a virtual address into a base register), any displacement relative to that base register must not result in an address outside the bank containing the original base virtual address. This includes the case where if the sum of the virtual address and the displacement were computed, the result would be a defined virtual address. This poses no real restriction on programming and result in early detection of a large class of errors.

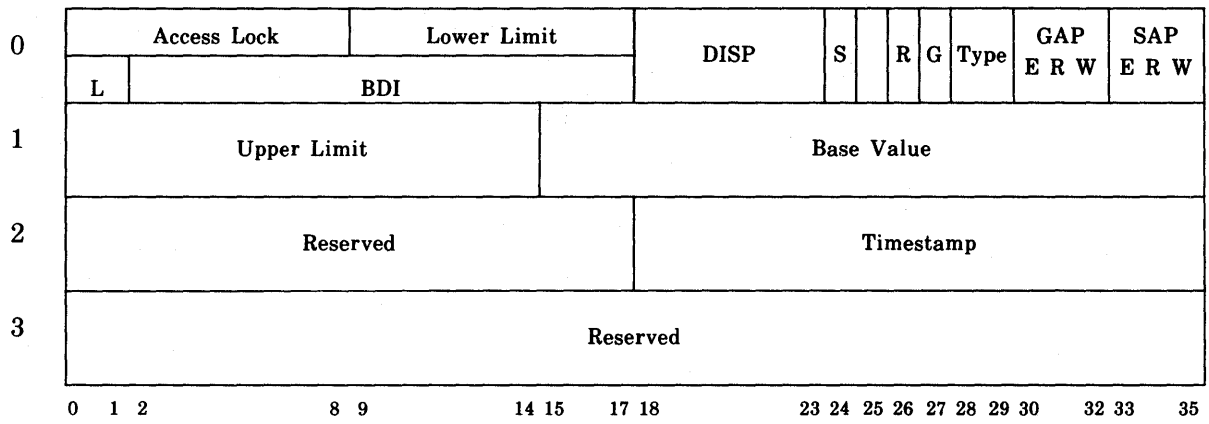
2.4.2.1.1. Bank Descriptors

Bank descriptors contain bank size, location, protection, and condition information. They are referenced by the instructions that interpret virtual addresses. These instructions use the information in the descriptor to establish the contents of the physical base register. This modified copy of the bank descriptor is made to save repetitive references to the bank descriptor that would otherwise be necessary during each instruction execution.

2.4.2.1.2. Bank Descriptor Format

A bank descriptor contains the storage management and protection information required to load a base register. A Bank Descriptor has the following format:

Words



where:

Word 0

Bits 0–17 The L,BDI field is the virtual address of a Bank Descriptor (BD) and is used in place of the current one if type=3 (indirect). Otherwise;

Bits 0–8 If type is 0, 1, or 2, the Access Lock field is compared with the Access Key field in the Indicator/Key register during every reference to the bank to determine whether GAP or SAP (see bits 30–35) is applicable.

Bits 9–17 The Lower Limit value has a granularity of 512 words if the S-bit is clear, or 32,768 words if the S-bit is set.

Bits 18–23 Displacement (DISP) is an unsigned integer in the range 0–63. DISP specifies the location of this Bank Descriptor relative to the first one that describes the bank in question. DISP is only relevant for large banks and must be set to 0 when Size (S) = 0 (small bank). DISP has explicit significance only for the Load Bank Name instruction.

Bit 24 Size (S) bit indicates, if set, that the bank may be as large as 16,777,216 words and, if cleared, that the bank does not exceed 262,144 words.

Bit 25 Not used

Bit 26 When Residency (R) fault is set, an interrupt occurs if load base usage is attempted.

Bit 27 When the General fault bit (G) is set, an interrupt occurs if load base usage is attempted. Differs from R in that G is not examined by the LAE and UR instructions.

Bits 28–29 Indicates the type of bank:

0 Extended Mode Bank (expects extended mode addressing/execution environment)
1 Basic Mode Bank
2 Gate Bank (changed to type 0 when loaded into B register)
3 Indirect bank descriptor (Bits 0–17 of word 0 contain the L,BDI of the bank descriptor to be substituted for this one. Only the Type, DISP, R, G, and L, BDI fields are meaningful in a Type 3 BD. Only one level of indirection is allowed.)

Bits 30–35 General Access Permission (GAP) and Special Access Permission (SAP) fields define the types of access (Enter/Read/Write) allowed. Within each field, the leftmost (E) bit enables enter access when set, the middle (R) bit enables read access, and the rightmost (W) bit enables write access. The GAP or SAP field selection is made by comparing the Access Lock field with the access key.

Word 1

Bits 0–14 The Upper Limit value has a granularity of 8 words if the S-bit is clear, or a granularity of 512 words if the S-bit is set.

Bits 15–35 The Base Value field is the left most 21 bits of a 24-bit word granular address of the bank in main storage.

Word 2

Bits 0–17 Reserved for software

Bits 18–35 The Timestamp field is updated with the value of a system global clock at the time this BD is fetched in the course of any Base register manipulation instruction.

Word 3

Bits 0–35 Reserved for software

2.4.2.1.3. Banks Larger Than 262,143 Words

The system supports banks of up to 2^{24} words and this requires certain software conventions to handle offsets larger than 18 bits:

- Based on information supplied by the user, system software must determine the maximum size (relative upper limit) in words of a bank; it must then allocate n consecutive Bank Descriptors (in the appropriate BDT) where:

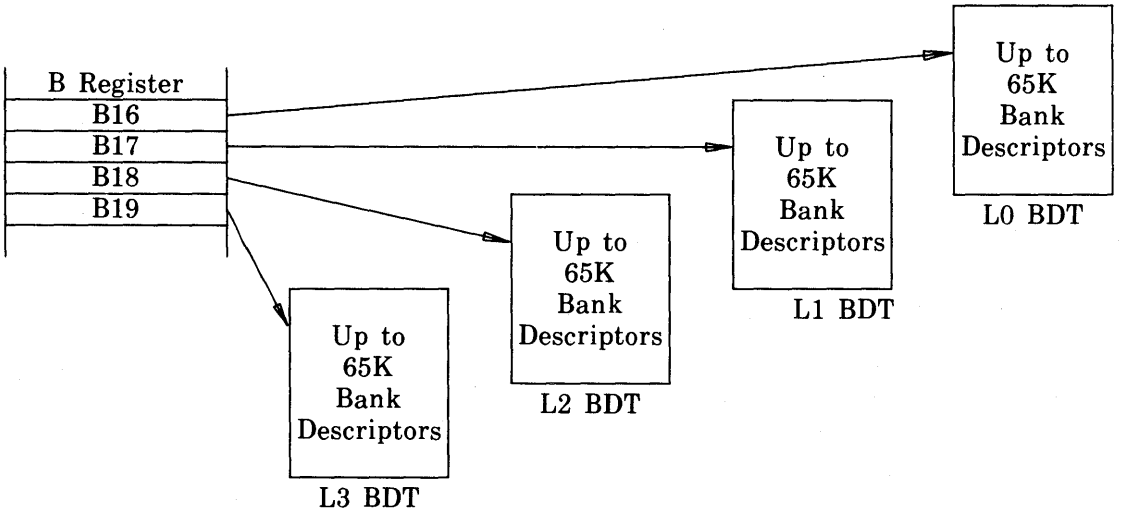
$$n = (\text{size} + 777777_8)/1000000_8$$

The first BD describes the entire bank and its associated L,BDI is the "name" of the bank; its DISP is zero. Each successive BD (if any) has its lower limit decreased by 10_8, upper limit decreased by 1000_8, and base value increased by 1000000_8, and DISP increased by one. For a large bank, all BDs must have their S bits set, and their timestamp fields must be searched to determine the "timestamp" of the entire bank.

- Software (including user programs) is expected in the general case to construct a virtual address by adding offset and bank name (rather than catenating them) after proper alignment and padding. A distinct bank name is an L,BDI (18 bits) catenated with 18 trailing zero bits, and an offset is 12 or 18 leading zero bits catenated with 24 or 18 significant offset bits. The sum of the two is a virtual address.

2.4.2.1.4. Bank Descriptor Tables

A bank descriptor table (BDT) is a linear list of bank descriptors. Four such tables are available in the processor, one for each active mode in the four-level address tree described in 2.4.1. Each table is described in the processor by a certain base register as shown below. These tables are at most 262,143 words in length, giving a maximum of 65,536 four-word bank descriptors.



2.4.2.2. Gates

Gates are structures in main storage that allow Executive software to control the IP when jumps between banks are attempted. This control applies to the LIJ, LDJ, and LBJ instructions in basic mode and to the CALL and GO TO instructions in extended mode. If the target of any of these instructions is a gate bank, as marked in the bank descriptor, the IP interprets the target address as a gate rather than an instruction. If the target is not a gate bank, no special control checks are made. The RTN instruction processes entries on the Return Control Stack. These entries are similar in format to gates. Gate banks are treated as data banks by the LBU and LBE instructions.

Since gates allow the IP to change protection environments, it is expected that the ability to construct gates will be tightly controlled by Executive software. By establishing appropriate access permission fields in all gates, Executive software can constrain some users to a limited set of banks while allowing others to access a larger set. Once the IP has passed the entry access

check for a gate, the target bank whose address is in the gate is always available. The Access Permission field in the descriptor for that bank is ignored. Also, any bank that is based in B0 can be read by the code executing within that bank, even if the bank descriptor prohibits read access.

2.4.2.2.1. Gate Use

The gate processing access path is described by Figure 2-2, assuming no faults.

Source Operands:

L,BDI	Offset	
(U)0-17 or Xa0-17	(U)18-35 U	for CALL and GOTO for LIJ, LDJ, and LBJ

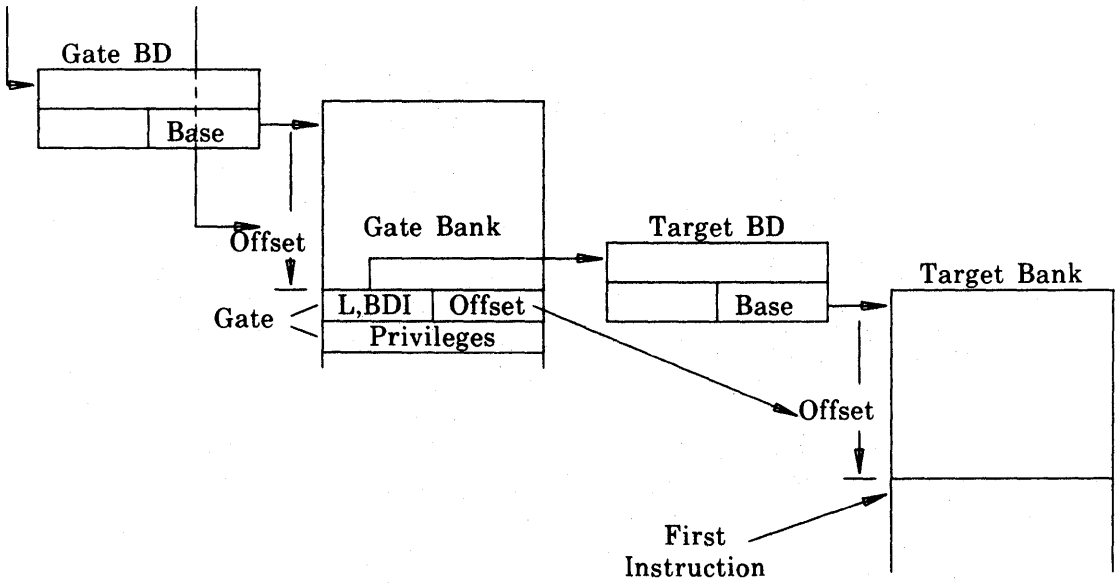


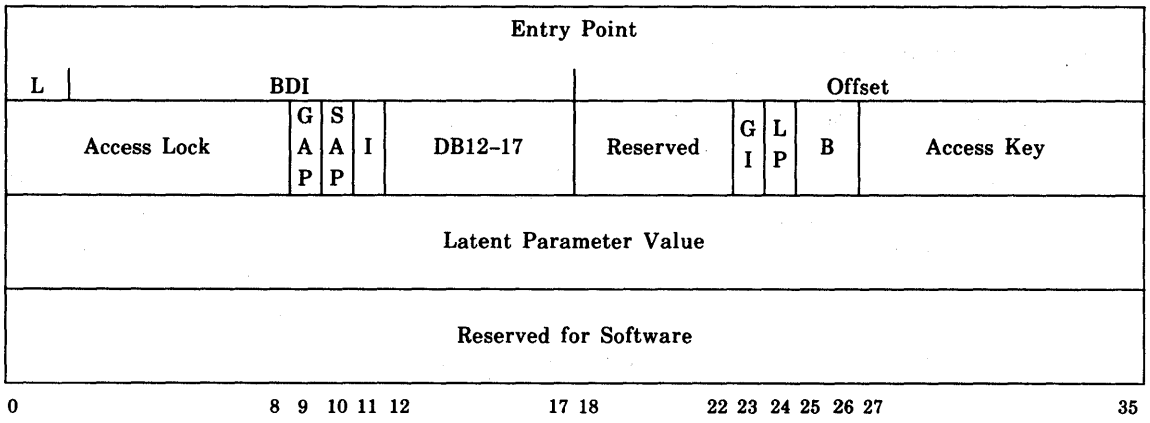
Figure 2-2. Data Processing Access Paths

2.4.2.2.2. Gate Format

Gates provide a mechanism whereby both storage and processor privilege can be changed in a controlled and protected fashion without requiring the invoker of the mechanism to have any special privileges. The important features of such a mechanism are:

- The description of the privilege changes, hereafter called simply the gate, can be addressed but not written (nor, preferably, read) by the invoker.
- The gate, rather than the invoker, must define where instruction execution is to begin following the privilege changes; otherwise a less privileged procedure could cause a more privileged one to malfunction by beginning execution at the wrong location.

A gate is processed by most inter-bank jump instructions (Call, COTO, LIJ, LDJ, and LBJ) and is four words having the format:



where:

Entry Point	Target L,BDI, Offset
Access Lock,GAP,SAP	Define privilege required to access this gate. GAP and SAP correspond to the GAP/SAP fields of a BD, but just for Enter access. Inhibits domain and designator change when set.
Designator Bits 12-17	Define new designator bit settings (for QT Enable, Interrupt Enable, PP, Register Set) if domain change is to occur. Note bit 16 (execution mode) is ignored.
GI	GOTO Inhibit indicator. If set during a GOTO operation, an addressing exception interrupt occurs to prevent execution of the operation. If clear, operation continues normally. This bit is ignored during CALL operations.
LP	Latent Parameter indicator; see Latent Parameter Value.
B	Base register to be modified on CALL or GOTO to basic mode bank. The B field is ignored otherwise.
Access key	The new access key is loaded into the indicator/key register of the ASP when the gate I-bit is clear.
Latent Parameter Value	If a latent parameter is indicated (LP set) the latent parameter value is stored in R0.

The gate processing access path is described by Figure 2-2 assuming no faults.

2.4.2.3. Storage Stacks

A stack is a linear address space contained in a bank. Stacks are Last In-First Out (LIFO) queue facilities. Stacks that the hardware supports are a portion of main storage that is obtainable in some word length, and are referred to as a stack frame.

Three types of stacks are supported. Two of these, the Return Control Stack (see 2.17.2) and the Interrupt Control Stack, (see 2.17.1) are special purpose stacks that the hardware uses directly. The third type is the Generic Stack that is provided for general purpose programming usage.

2.4.2.3.1. Stack Structure

Five parameters are required for stack control:

1. Specification of the base register that describes the bank containing the stack to be manipulated. This may be explicit (generic stacks) or fixed values (return control stack, interrupt control stack).

2. The empty position of the stack. This is defined by the upper limit (UL) of the stack base register, the empty position being the first out-of-limits word address beyond UL.*

   * *Stacks expand towards the lower limit (LL) field (that is, lower numbered addresses). Expansion of the stack towards the lower numbered addresses provides two features:*

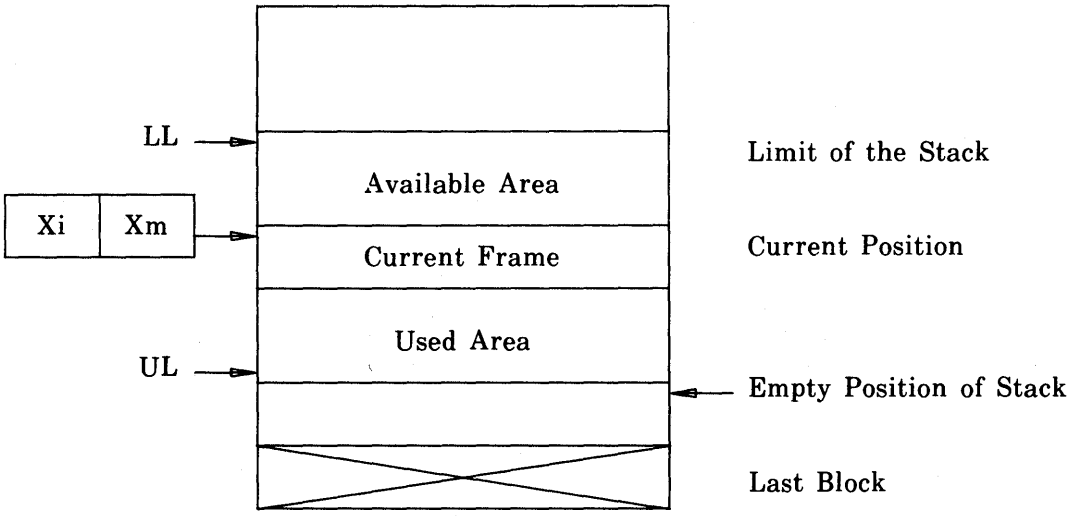
     a. *normal u + Xm addressing to access objects currently in the stack, and*

     b. *the size of the last frame acquired is not required to properly acquire subsequent frames.*

3. The limit of the stack; that is, the maximum extent to which the stack can grow. This is defined by the lower limit (LL) of the stack base register.

4. The current position of the stack; that is, the address of the starting word of the current stack frame. This is specified in the modifier portion of an X-register, again explicitly for generic stacks and fixed values for return control stack and interrupt control stack.

5. The size of a stack frame when stack space is being bought (acquired) or sold (released). A stack frame is one or more consecutive words of space, and is the basic unit of stack management. For generic stacks the size is variable and programmer specified by the stack manipulation instructions; the return control stack and interrupt control stack frame sizes are fixed values. In the most straightforward cases, however, the frame size will be in the increment portion of the X-register containing the current position, and that is assumed in the following:



2.4.2.3.2. Stack Manipulation

The user may define stacks and manipulate them with the BUY and SELL instructions. The user selects either the B- or X-register. The stack frame size is also user defined and can be negative; if so, care must be exercised in choosing the starting (empty) address to ensure proper detection of stack overflow or underflow.

User stacks unlike the interrupt control stack and return control stack, may be accessed by using 24-bit addressing or 18-bit addressing.

User manipulation of B1 is restricted to help implementation of an activity - local storage stack. However, the BUY and SELL instructions may be used to manipulate this stack in the normal manner.

2.4.3. Operand Protection

Operand protection is provided by program isolation using address control (see 2.4.1) and access control. Access control allows the IP to enforce rules defining the permitted access to operands. The rules for data are established separately for each bank and apply uniformly to each operand within the bank. The rules for instruction access can be optionally established to limit entry into a bank to a list of one or more addresses known as entry points. Each entry point is described by a gate structure.

2.4.3.1. Access Key

The IP has a key that is used to enforce the access rules described by each gate and bank. The key has two parts: a domain number and a ring number. The domain number determines which of all visible banks and gates (objects) are owned by the IP at the moment (see 2.4.3.2). The ring number determines how many other objects the IP can access even though they are not within its domain. That is, the ring number determines the current data access privilege level of the IP.

2.4.3.2. Accesss Lock

Each gate and bank has a lock value that has the same two parts as the access key: a domain number and a ring number. Domain number lets the user declare what group the object is a member of; all objects in a group (domain) have the same domain number. Ring number in a lock allows the user to declare how sensitive the information within the bank or behind the gate is. These definitions result from the rules used to select access permission rules.

2.4.3.3. Access Permission

There are two sets of access permission controls associated with each gate and bank. These are called the Special Access Permissions (SAP) and General Access Permissions (GAP) fields. For banks, there is a bit within each field to control read, write, and enter. Enter allows the IP to cross into the bank from another bank and execute any instruction within the bank. For gates, each control field only has one bit, enter, that permits or denies the use of the gate. If the bit is set (has a value of one) the access is allowed; if clear, the access is denied.

Access checks are made when the object is referenced for data or entry is attempted. No access validation is done when data banks are based. Only the interpretation of the virtual address is done at that time.

2.4.3.4. Access Permission Field Selection

Only one of the two available access permission fields are used to check each access attempt. The selection is based on the comparison of the current access key held by an IP with the access lock associated with the bank or gate being referenced. IF the key "matches" the lock, the Special Access Permission (SAP) field is used, otherwise the General Access Permission (GAP) field is used.

The current key "matches" the lock when the following relation is satisfied:

KEY.DOMAIN = LOCK.DOMAIN

OR

KEY.RING < LOCK.RING

This is illustrated with the example in Figure 2-3.

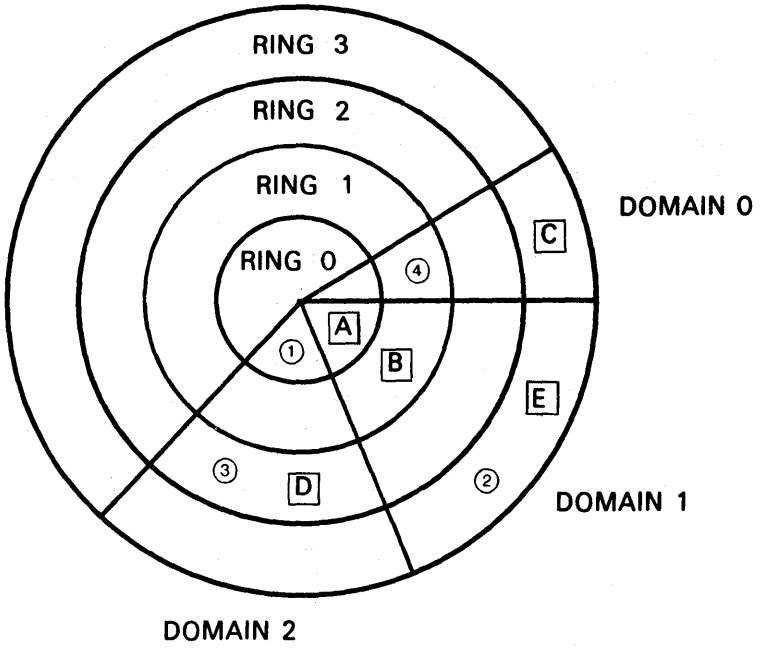


Figure 2-3. Example of Access Permission Field Selection

The numbered circles represent several possible key values that can be held by an IP. When in position 1, the IP is holding a key with ring zero and domain 2; specifically, KEY.RING=0 and KEY.DOMAIN=2.

Likewise, the lettered boxes represent some possible lock values associated with gates and banks. An object in position A has LOCK.RING=0 and LOCK.DOMAIN=1.

Table 2-2 shows the access permission field (S = special or G = general selected for each object location and each IP location in the figure.

Table 2-2. Access Permission Field Selection

IP* Location	Ring / Domain	Object				
		A	B	C	D	E
		Ring, Domain				
		0,1	1,1	3,0	2,2	3,2
1	0 2	G	S	S	S	S
2	3 2	S	S	G	G	S
3	2 2	G	G	S	S	S
4	1 0	G	G	S	S	S

**See Figure 2-3*
NOTE: G = General, S = Special

2.4.3.5. Address Control and Access Control

Access control applies to the list of banks that are currently visible. This list is defined by the four bank descriptor tables currently known to the IP (see 2.4.1). Thus, a single domain number can be freely assigned to more than one set of banks as long as each set is in a separate sub-tree of the address tree in Figure 2-1. The address control properties of the address tree keep these separate sets of bank from being visible to the IP at the same time.

2.5. Base Registers

The IP base register set consists of 32 Base (B) registers; the first sixteen (B0 through B15) are user Base registers, and the last sixteen (B16 through B31) are Executive Base registers.

Instructions are provided to load virtual addresses into Base registers and retrieve the last address placed in the register. During the load of the virtual address, it is interpreted to locate the descriptor for the bank containing the address. Portions of the bank descriptor are moved into the actual register in the IP. The programmer should treat the register as containing only a virtual address.

2.5.1. Base Register Properties

User programs express all storage address in terms of offsets relative to the contents of a specified base register. Thus, the addresses of a storage operands are virtual addresses. It is required that an operand address and the virtual address in the base register be in the same bank. This enables the association of address bounds and the contents of each base register, so that the references using that register are restricted to one of storage structure.

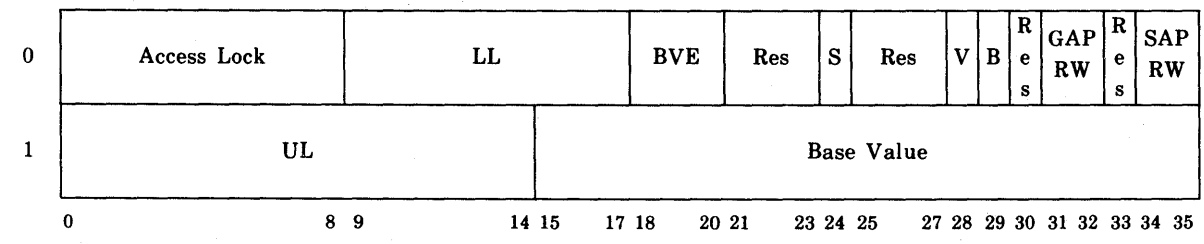
The address bounds are derived from the upper and lower address limit fields. These fields are found in the bank descriptor for the bank containing the virtual address loaded into the registers. The address limits are adjusted to account for the OFFSET-field of the virtual address and are saved as the address bounds associated with the register contents. These adjusted address limits are retained only to avoid repetitive examination of the bank descriptor whenever the base register is referenced. The adjusted base address of the structure in physical main storage is also retained, along with the access lock, access permission fields, and some of the information describing the bank type.

When interpreting instructions in basic mode, the processor may bypass address translation. This occurs only when executing within the Executive and when the i-field of the instruction is set. The specified index register has a 24-bit (rather than 18-bit) modifier field and the generated operand address is the physical main storage address. No access checks are performed.

The use of the base registers depends on the IP instruction execution mode. When executing code while in basic mode, only base registers B12-B15 are available and the one selected is determined implicitly from the operand address (see 2.12.2). When executing code in extended mode, either the first 16 or all 32 base registers are available depending on the current processor privilege level. User programs can only address base registers B0 through B15.

2.5.2. Base Register Format

The base register format with description of each bit position follows:



where:

Word 0

Bits 0-8 — The Access Lock field is compared to the requester's access key to determine whether the General Access Permission (GAP) or the Special Access Permission (SAP) fields are to be used for access permission validations.

Bits 9-17 — Lower Limits (LL) has 512 word granularity if the size (S)-bit is clear; if the S-bit is set, 32,768 word granularity is in effect. LL is used during address generation limits testing.

Bits 18–20 — Base Value Extension (BVE) is the rightmost three bits of the 24-bit word granular base address value. This field is nonzero only if subsetting was used in loading the base register. This occurs only if the virtual address has a nonzero affect field. In BO, BVE is always zero.

Bits 21–23 — Reserved

Bit 24 — The Size (S)-bit indicates, if set, that the bank can be as large as 16,777,216 words and, if clear, that the bank does not exceed 262,144 words in size. The value of this bit selects the granularity of the Upper and Lower Limits fields. If basic mode address generation is being performed and the base register is other than B15 and the S-bit is set, the base register is treated as void.

Bits 25–27 — Reserved

Bit 28 — The Void (V)-bit indicates, if clear, that the B-register contains valid information; if set, it indicates that the B register is void. A void B register requires no time stamp maintenance, has a value of zero or one in bits 0–17 of its Active Base Table (ABT) entry (B0–B15 only), and is treated as if the lower limit exceeded the upper limit (squashed limits). Bits 0–17 of the entry for B0 are located in the leftmost 18 bits of the Program Address Register. When the V-bit is set, only the B-bit has meaning and all other fields are undefined.

Bit 29 — The Basic Mode (B) bit, when set, indicates implicit base selection and address generation is to be performed using the B-register selected from B12–15 according to the Basic Mode base selection algorithm; if clear, indicates that explicit base selection is in effect, and address generation is to be performed using this B-register.

This bit is treated as zero in any B-registers when used for implicit (hardware generated other than by the instruction b-field or a-field) storage references.

Bit 30 — Reserved

Bits 31–32 — The General Access Permissions (GAP) field validates the storage reference when the access lock does not "match" the access key of the requester; the bits are defined as follows:

R (bit 31) - Reads allowed, if set.
W (bit 32) - Writes allowed, if set.

Bit 33 — Reserved

Bits 34–35 — The Special Access Permissions (SAP) field is the same as GAP, except that it is used instead of GAP when the access lock matches the access key of the requester.

Word 1

Bits 0–14 — Upper Limit (UL) is 8 word granularity if the Size bit is clear; if the Size bit is set, 512-word granularity is allowed.

Bits 15–35	The Base Value field is the leftmost 21 bits of the 24 bit word granular. The Base Value and BVE fields together form the Base Address Value; that is, Base Address Value = Base Value/BVE.

When the contents of a base register is moved to storage, the unused bits are cleared to 0.

2.5.3. Fixed Base Register Assignments

Certain base registers are set aside for specific purposes, which are described as follows:

Register	Use
B0	This register points to the current bank (instruction stream) when in extended mode. The address bounds for this bank are never modified and the OFFSET–field of the virtual address loaded into B0 is retained as the program counter field in the program address register.
B1	This register points to the programming language automatic stack and can only be modified by very privileged Executive routines.
B2–B11	These registers are available for user data banks.
B12–B15	The processor uses these registers differently depending on whether it is executing in basic mode or extended mode. In basic mode the program allocates these registers to point at either code or data. The processor implicitly selects the particular back for address generation. This is described in 2.12.2. In extended mode, the program can only allocate these registers so that they point at data. Selection is either explicit by the instructions b–field or implicit if the referenced base register (which may be any of B1–B15) contains a virtual address within a bank containing basic mode information.
B16	This register points to the interrupt vector and the bank descriptor table for the virtual address whose L–field contains 0.
B17–B19	These registers point to the bank descriptor tables for virtual addresses whose L–fields contain 1–3, respectively.
B20	This register is the active base table with the base numbered 1 through 15 as the offset.
B21	This register points to the current return control stack.
B22	This register points to the interrupt control stack.
B23–B31	These registers are available to Executive code.

2.6. Base-Relative Address Space

All addresses calculated by instruction interpretation are known as relative addresses. These addresses are always used in conjunction with a base register to locate the operand in main storage. Specifically, the BASE-field in the selected base register is added to the relative address specified by the instruction to calculate the address of the operand in main storage. The calculation of the relative address and the selection of the base register depends on processor execution mode and base register contents. See 2.12 for the details of the calculation and selection.

2.7. General Registers as Storage Operand Locations

GRS locations are used as storage operands selected by the relative address when that address is less than 128 and certain other conditions are satisfied.

When the processor is in extended mode, the b-field of the instruction must be zero. Thus, relative addresses 0-127 for all data base registers can be in banks within main storage. However, the first 128 relative addresses for the current code bank (B0) are only available for instructions. Jumps to these addresses will execute instructions within main storage while data references will address GRS locations.

When the processor is in basic mode, any data reference with a relative address less than 128 will be satisfied within GRS. Only jumps and absolute (untranslated - see 2.5.1) address references will always be satisfied in main storage.

The access rules for GRS locations are always enforced. These are shown in Table 2-1.

2.8. Immediate Operands

An immediate operand is formed by developing an operand address into a 36-bit operand. This operand is termed an immediate operand because it does not require a storage access and it is immediately available. This is not to imply that such an instruction necessarily executes in less time than one whose operand is in storage. Immediate operands are not subject to storage limits checking. The specification of the immediate operands is discussed in 2.11.1.2.

If the operand is an immediate operand, and the x-field of the instruction is zero, the h- and i-fields are added to with the u-field to form an 18-bit operand address that will be immediately developed into a 36-bit operand. As no indexing is performed, the value +0 is added to the 18-bit operand address. This is significant only if the operand address is -0, in which case +0 will result. Therefore it is impossible to generate an immediate operand whose 18 least significant bits have a value of 1.

Regardless of the value of the x-field, the value in the j-field determines the manner in which the operand address is to be extended from 18 to 36 bits in width. If the j-field contains 16_8, the operand address is extended with 18 zero bits on the left, and if the j-field contains 17_8, the value of the sign bit (most significant bit) of the operand address is used to fill the leftmost 18 bits of the generated 36-bit operand.

For some f-field and j-field combinations where the j-field is a minor function code, for example, shift instructions ($f=73_8$, j=0-5 and 10_8-13_8), the operand is generated according to the rules for immediate operands, except that the h- and i-fields are not combined with the u-field and no extension is performed. In these cases, the length of the operand is dependent on the particular instruction.

2.9. Sequence of Operand References

All non-interative instructions, in either basic or extended mode, that access multiple main storage words (as opposed to GRS) do so in an undefined order; furthermore, they are subject to retry from the beginning (for example, hardware error) and thus must not be used for activity (process) synchronization (for example, to clear a Test and Set cell). Such instructions include DS, DTE, DL, SRS, DCEL, CALL, GOTO, LIJ, LBJ, LBU, and LBUI.

For instructions of any type or mode that have multiple word storage operands, except basic mode searches, the IP is free to access all words of the operand for acceleration purposes, even though the particular values involved would not require such access, and any resulting reference violation interruptions (limits errors) need not be suppressed. For example, on EXR of a test instruction, the IP may look ahead beyond a word which satisfies the test, but not beyond the limit implied by R1. Some operand lengths are defined by the operand, such as with a stop character; the IP must not look ahead beyond the end of the operand specified in such fashion.

Non-interative operations that make multiple word operand references are subject to the restriction that the operation is undefined if the operand address range (for example, U, U+1, etc.) includes or spans a value of all ones. The restriction applies to the following cases:

Case	Restriction Applies
ICS Writes	All Interrupts
RCS Accesses	CALL, RTN
Gate Accesses	CALL, GOTO, LIJ, LDJ, LBJ
DA, DAN	Extended Mode Only
DL, DLN, DLM, DS	Extended Mode Only
DTE	Extended Mode Only
LRS, SRS	Extended Mode Only
DLSC	Extended Mode Only
DFA, DFAN, DFM, DFD	Extended Mode Only
DFU, DLCF, FCL	Extended Mode Only
ACEL, DCEL	Always
LAE, UR	Always
TVAE	Always

2.10. Instruction Execution Mode

The IP interprets instructions in two formats. The format depends on the value of Basic Mode Enable Designator Bit 16 (DB16). When DB16 is set to 1, the processor executes in basic mode and the instruction format is compatible with previous Series 1100 Systems. Whenever DB16 is 0, the processor executes in extended mode and the instruction format is altered. This is necessary to enable the use of the 32 base registers and the 36-bit virtual address space.

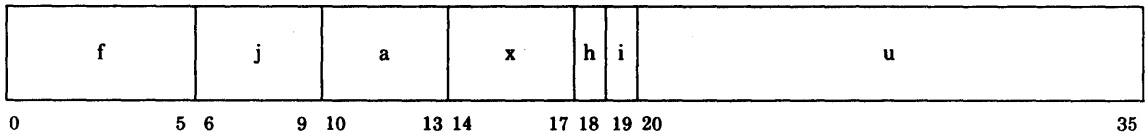
When the processor encounters an inter-bank transfer instruction (LIJ, LDJ, LBJ, CALL, or GOTO instructions), the type field found in the bank descriptor of the target bank is interrogated. If that field indicates that the bank contains instructions in the basic (compatible) format, the processor sets DB16. Thereafter, all instructions are interpreted in basic mode format and are obtained from any of the four banks based on B12 through B15. Bits 18 through 35 of the program address register contain the relative address of the instruction and the particular base register used is determined according to the same rules as operand base register selection discussed in 2.12.2.

If the type field in the bank descriptor indicates that the bank contains extended mode instructions, DB16 is cleared to 0. Subsequent instructions are interpreted in extended mode format and obtained from the bank based on B0.

2.11. Instruction Word Formats

2.11.1. Basic Mode Instruction Word Format

The following illustration shows the format of an instruction word when DB16 is set. An explanation of each field follows. Some fields have alternative meanings, depending on the class of instruction.



2.11.1.1. Function Code f-Field

The 6-bit f-field specifies the operation that the IP is to perform. For function codes of 07_8, 37_8, or above 70_8, the f- and j-fields combine to produce a 10-bit function code. When the function code is 05_8, the f- and a-fields are combined to produce a 10-bit function code. Further, for some of these function codes, the a-field is also included to form a 14-bit function code. An invalid function code generates an interrupt.

2.11.1.2. Partial-Word or Immediate-Operand Designator j-Field

The j-field is an extension to the function code if the value of the f-field is 07, 37 or greater than 70. In all other cases, the j-field represents an operand qualifier as a partial-word specification. As an operand qualifier, the j-field indicates whether the actual operand is the entire 36-bit word specified by the operand address (j=0); an 18, 12, 9 or 6 bit subfield of the word (j=01-15); or the operand address itself (j=16-17). Figures 2-4 and 2-5 show the relationship between the j-field values and the resulting operand formation.

The j-field values of 4, 5, 6, and 7 each have more than one meaning, depending on the value of Quarter Word Selection Designator Bit 32 (DB32). If DB32 is 0, these values indicate certain half-, third-, and sixth-word operations. If DB32 is 1, j-values of 4 through 7 indicate quarter-word operations.

j (Octal)	Quarter-Word Designator*	Storage Location → Arithmetic Register
0	0 or 1	0 35 → (36) → 0 35
1	0 or 1	18 35 → (18) → 0 zeros 17 | 18 35
2	0 or 1	0 17 → (18) → 0 zeros 17 | 18 35
3	0 or 1	18 35 → (18) → 0 signs 17 | 18 35
4	0	0 17 → (18) → 0 signs 17 | 18 35
5	0	24 35 → (12) → 0 signs 23 | 24 35
6	0	12 23 → (12) → 0 signs 23 | 24 35
7	0	0 11 → (12) → 0 signs 23 | 24 35
4	1	9 17 → (9) → 0 zeros 26 | 27 35
5	1	27 35 → (9) → 0 zeros 26 | 27 35
6	1	18 26 → (9) → 0 zeros 26 | 27 35
7	1	0 8 → (9) → 0 zeros 26 | 27 35
10	0 or 1	30 35 → (6) → 0 zeros 29 | 30 35
11	0 or 1	24 29 → (6) → 0 zeros 29 | 30 35
12	0 or 1	18 23 → (6) → 0 zeros 29 | 30 35
13	0 or 1	12 17 → (6) → 0 zeros 29 | 30 35
14	0 or 1	6 11 → (6) → 0 zeros 29 | 30 35
15	0 or 1	0 5 → (6) → 0 zeros 29 | 30 35
16**	0 or 1	18 35 → (18) → 0 zeros 17 | 18 35
17**	0 or 1	18 35 → (18) → 0 signs 17 | 18 35

* *The Quarter-Word Designator Bit (DB32) is held in the designator register.*

** *If x = 0, then h, i, and u are transferred.*

If x ≠ 0, then u + (X_m) is transferred.

Figure 2-4. Data Transfers from Storage

j (Octal)	Quarter-Word Designator*	Arithmetic Register → Storage Location
0	0 or 1	0 35 → 36 → 0 35
1	0 or 1	18 35 → 18 → 18 35
2	0 or 1	18 35 → 18 → 0 17
3	0 or 1	18 35 → 18 → 18 35
4	0	18 35 → 18 → 0 17
5	0	24 35 → 12 → 24 35
6	0	24 35 → 12 → 12 23
7	0	24 35 → 12 → 0 11
4	1	27 35 → 9 → 9 17
5	1	27 35 → 9 → 27 35
6	1	27 35 → 9 → 18 26
7	1	27 35 → 9 → 0 8
10	0 or 1	30 35 → 6 → 30 35
11	0 or 1	30 35 → 6 → 24 29
12	0 or 1	30 35 → 6 → 18 23
13	0 or 1	30 35 → 6 → 12 17
14	0 or 1	30 35 → 6 → 6 11
15	0 or 1	30 35 → 6 → 0 5
16	0 or 1	No Data Transfer
17	0 or 1	No Data Transfer

* *The Quarter-Word Designator Bit (DB32) is held in the designator register.*

Figure 2-5. Data Transfers to Storage

For store instructions, if the j-field specifies a full transfer, the word in the register is transferred to the specified location in main storage. If the j-field specifies a partial-word transfer, the partial word is transferred from the least significant bit positions of the register to the j-field specified portion of the storage location; the remaining portion of the storage location is undisturbed. If the j-field value is 16 or 17 for a store instruction, no transfer occurs.

For other instructions, if the j-field specifies a full word transfer (00), the word at the locations specified by the operand address is the actual operand. If the j-field specifies a partial-word transfer (01–15), the partial operand and the remaining more significant bit positions are filled with 0's or sign bits (depending on the j-field value) forming a full word (36 bit) operand. When the value is the j-field specifies sign extension, the remaining more significant bit positions of the operand are filled with bits that are identical to the most significant bit of the partial word. When U is less than 0200, j-field values of 01–15 are ignored and a full 36-bit operand is formed.

2.11.1.3. Control Register Designator a-Field

For the majority of instructions, the a-field of the instruction contains an A-, X-, or R-register address for use as an operand. The EXEC Register Set Selection Designator Bit 17 (DB17) selects whether the register is selected from the user or Executive register set. In some instructions the value in the a-field references two or three A-registers. When two A-registers are referenced, the value in the a-field explicitly references the register A, and implicitly references the register A+1. When three A-registers are referenced, the value in the a-field explicitly references the register A, and implicitly references the register A+1 and A+2.

In the Jump on Greater and Decrement (JGD) instruction ($f=70_8$), the value in the j-field and a-field combine to form a single numeric value. This value specifies which one of the 128 addressable control registers is used as the counter in the execution of the instruction.

If the value of the f- and j-fields of the instruction are 05, 0–17; 50, 0–17; 37, 04–06; 73, 14–17; or 74, 4–5:14–15, the a-field of the instruction is an extension to the function code.

2.11.1.4. Index Register Designator x-Field

The x-field of the instruction contains an index register address. If the value of the x-field is not 0, the contents of the modifier field of the index register X_x is added to the operand address to form the relative address or immediate operand. If the value of the x-field is 0, no indexing occurs. The modifier field consists of bits 18–35 for most instructions and bits 12–35 for extended mode instructions with the i-field set.

2.11.1.5. Index Incrementation Designator h-Field

The single-bit h-field controls the update of the index value (X_x-modifier) by the increment field (X_x-increment) after indexing. The signed increment portion is added to the modifier portion, and the result stored back into the index register with the increment field left unchanged. The format of the index register depends on Processor Privilege (PP) Designator Bits 14-15 (DB14-DB15) and the instruction i-field as follows:

Processor Privilege	i-field	X_x-increment	X_x-modifier
0-3	0	bits 0-17	bits 18-35
2,3	1	bits 0-17	bits 18-35
0,1	1*	bits 0-11	bits 12-35

* *If the instruction executed is a jump, the i-field is ignored and treated as zero, and the index register has the normal 18-bit increment and modifier.*

2.11.1.6. Indirect Address Designator i-Field

The single-bit i-field normally controls the use of indirect addressing during instruction execution. If the processor privilege is either 0 or 1, the i-field determines index register format as described in 2.11.1.5 and also bypasses normal address translation and checking. The following description applies only to user-mode execution. If the i-field is 0, the instruction functions without indirect addressing. If the i-field is 1, the 22 least significant bit positions of the instruction (x-, h-, i-, and u-fields) are replaced in the instruction register within the processor with the contents of the 22 least significant bit positions of the operand word. The contents of the instruction register is then interpreted as a new instruction. Indirect addressing continues as long as i=1 in the instruction register, with full indexing capability at each level.

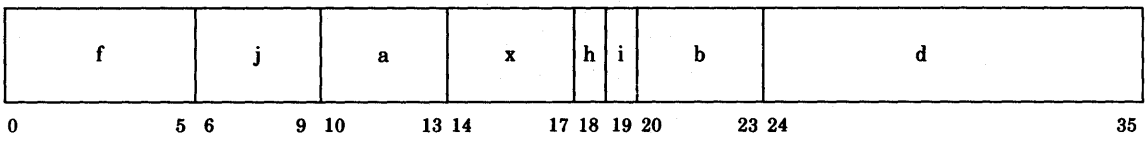
2.11.1.7. Operand Address u-Field

The 16-bit u-field normally specifies the operand relative address prior to indexing. However, for certain instructions it may hold a constant. For example, the shift instructions use the seven least significant bit positions to produce the shift count. For all instructions, the value in the u-field may be modified by the contents of an index register. This occurs when the x-field is not 0. The modification is performed by appending two bits of 0 to the left of the u-field and adding the result to the modifier part of the selected index register. When the function code and j-field specified immediate operand selection (see 2.11.1.2) and the x-field is 0, the h- and i-fields are linked with the normal 16-bit u-field to form an 18-bit u-field.

When the calculation of the relative address yields a value with all bits set, +0 is used instead. Thus, the last address is not available either as an immediate operand or to locate a storage operand.

2.11.2. Extended Mode Instruction Word Format

In extended mode, the majority of the instructions use the following format:



The jump instructions, shift instructions, and immediate operand specifications (j = 16_8, 17_8) use the basic mode format (see 2.11.1).

The effect of the f–, j–, a–, x–, h–fields on instruction interpretation is the same in extended mode as in basic mode. However, the meaning of f–field; or f–, j–field; or f–, j–, a–field; or f–,,a–field as a function code may be different in extended mode (that is, the repertoire has, in a number of cases, been reorganized). The next subsections describes only those fields that have an interpretation different from basic mode.

2.11.2.1. Operand Address d–Field

The x– and d–fields are used in operand relative address formation. The 12–bit value in d–field is zero extended on the left and added to the modifier field of the index register selected by the x–field value. The addition is ones complement with negative zero suppression.

If this addition yields a result with all bits set (either 18 or 24 bits), positive zero is used instead. It is therefore not possible to reference the last address in any bank.

2.11.2.2. Index Register Format Selector i–Field

When the processor privilege is either 2 or 3, the i–field selects the index register format. In this case, if the i–field is 0, the index register has 18–bit increment and modifier fields compatible with basic mode, and if the i–field is 1, the index register has a 12–bit increment field and a 24–bit modifier field. See 2.11.1.6 for interpretation of the i–field when the processor privilege is 0 or 1.

The format of the index register depends on Processor Privilege Designator Bits (DB14–15) and the i–field with different rules from basic mode.

Processor Privilege	i–field	X_x–increment	X_x–modifier
2,3	0	bits 0–17	bits 18–35
2,3	1*	bits 0–11	bits 12–35
0,1	1**	bits 0–17	bits 18–35

* *If the instruction executed is a jump, the i–field is ignored and treated as zero, and the index register has the normal 18–bit increment and modifier.*

** *The i–field is merged into the b–field when PP is zero or one.*

2.12. Base Register Selection

The mechanism used to select the base register used to locate a storage operand depends primarily on processor execution mode. In extended mode, the instruction explicitly selects the base register. However, if upon reference to that base register, the B-bit is found set (see 2.5.2) implicit base register selection is then used to locate the actual base register. In basic mode, implicit base register selection is always used. This method performs an ordered search of the limits fields in physical B12–B15. The search stops when the specified relative address falls within the address limits found in one of the registers.

2.12.1. Explicit Base Register Selection

The b-field selects the Base (B) register, which is used to form and validate the storage address. If the processor privilege is either 2 or 3, the selection is always from registers B0–B15. If the processor privilege is either 0 or 1, the instruction i-field is treated as the high-order bit of a 5-bit b-field to select from registers B0–B31. If the instruction is a jump, there is no b-field and B0 is always used.

If a virtual address in a basic mode bank was previously loaded into the selected base register, one of the B-registers (B12 through B15) is selected as the B-register to be used. The selection is according to the discussion in 2.12.2.

If the specified relative address is not within the address limits associated with the base register or the attempted access in not allowed, the instruction execution is aborted and an appropriate interrupt is generated by the IP.

2.12.2. Implicit Base Register Selection

The base register selected to compute the physical main storage location of the operand is either B12, B13, B14, or B15. The selection is made by comparing the relative operand address (sum of the u-field and Xx-modifier) with the upper and lower address field found in each of these base registers. The BDR Selection Designator Bit (DB31) specifies which pair (B12, B14 or B13, B15) is to be examined first. Within each pair, the lowest numbered register is checked first. This designator bit is toggled to the opposite value when a jump instruction has its target in the pair checked second in the algorithm. If the address is not within the limits of any base register or the selected register does not permit the attempted access, an interruption of instruction execution occurs.

When in basic mode and if the current processor privilege value is 0 or 1 and the i-field has the value of 1, base register selection and access validation is bypassed and the specified address is used as the physical main storage address of the operand.

Figure 2-6 is a diagram of the algorithm used to select one of the four base registers.

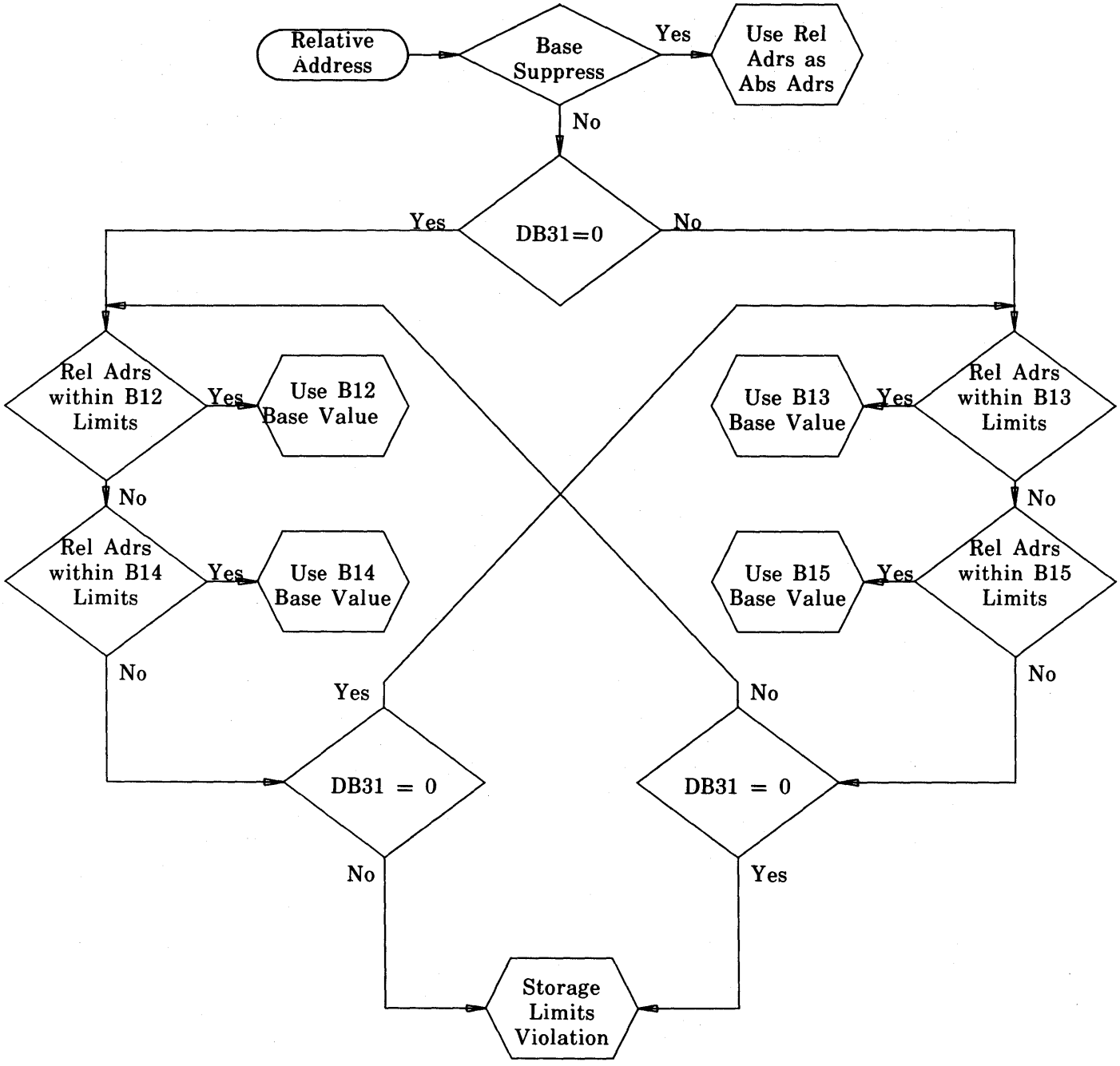


Figure 2-6. Base Value Selection

2.13. Activity State Packet

The activity state packet includes the registers internal to the processor required for operational control of an activity. These include the Program Address Register (PAR), the Designator register (D), the Indicator/Key register (I/K), the Quantum Timer value (QT), and four words of mid-execution state (F0 and ISW0–ISW2). These eight words, in the order stated above, are found in each of the following data structures:

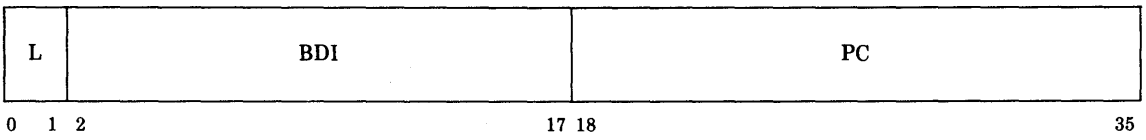
Structure	Location
Interrupt Stack Frame	Words 0–7
Activity State Packet	Words 16–23
Virtual Processor Control Block (VPCB)	Words 8–15
User Return (UR) instruction operand	Words 0–7

The format of an activity state packet is:

Word	Activity State Packet
0	Program Address Register
1	Designator Register
2	Indicator/Key Register
3	Quantum Timer Value
4	F0
5	ISW0
6	ISW1
7	ISW2

2.13.1. Program Address Register

The Program Address Register (PAR) contains the virtual address of the current instruction. If an interrupt prevents complete instruction execution, the captured PAR value is the address of that instruction. However, if an interrupt occurs after the termination of an instruction and prior to the interpretation of the next instruction, the captured value is the address of the new instruction.



where:

Bits 0–17 — The L,BDI 18-bit field contains the name of the bank currently based by B0. There is no offset field retained. Its value is always treated as zero. The L,BDI field is changed by interrupts and also by transfers between banks except when in basic mode. In extended mode, the hardware ensures that B0 always points to the currently active instruction code bank. B0 is not used in basic mode and neither is this field, and B0 cannot be subsetted.

Bits 18–35 The Program Counter (PC) is the offset of the instruction currently in execution. This offset is relative to the virtual start of the B0 bank if the processor is in extended mode, and is relative to the virtual start of the address space described by B12–B15 if the processor is in basic mode. There is no connection between the bank based on B0 and PC when the processor is in basic mode. PC is always 18 bits independent of the size (S-bit value) of the bank based on B0.

2.13.2. Designator Register

The designator register contains information for controlling the basic operational mode of the IP. The designator register can be loaded by instruction, although certain bit combinations are not valid or may not be available to the user. In general, no hardware checks are made for these invalid combinations. When an interrupt occurs, the current value of the designator register is stored in GRS, and the register is cleared, unless otherwise specified in the following paragraphs, to establish the proper interrupt handling environment. The designator register bit description is given in Table 2–3.

Table 2–3. Designator Register Bit Description

Bit Number	Bit Names	Description
DB0	Software Performance Monitor Change Enable	If the Software Performance Monitor Change Enable bit is set, the Load Performance Monitor instruction may alter the three software Performance Monitor designator bits DB3–DB5. If clear, the instruction may not alter DB3–DB5, it operates as a No Operation instruction (NOP).
DB1–DB2		Reserved
DB3–DB5	Software Performance Monitors	Software Performance Monitor bits are used with Processor Privilege designators (DB14–DB15) to specify sixteen software states.
DB6	Fault Handling In Progress	Fault Handling In Progress (FHIP) is set by hardware on Hardware Check interrupts; it may also be set by Executive software. FHIP is cleared by software when fault handling is complete.
DB7–DB11		Reserved.
DB12	Quantum Timer Enable	If the Quantum Timer Enable designator is 1, the value in the Quantum Timer is decremented by one for every 60 nanoseconds that the IP is actually executing instructions. When the Quantum Timer value is 0, or negative, a Quantum Timer interrupt is generated.
DB13	Deferrable Interrupt Enable	If the Deferrable Interrupt Enable designator is set, external interrupts are allowed; when clear, external interrupts are locked out (held).

Table 2-3. Designator Register Bit Description (continued)

Bit Number	Bit Names	Description
DB14–DB15	Processor Privilege	The Processor Privilege (PP) designators form the Processor Privilege field; the four values have the following correspondence to software structure: PP=0 Executive Kernel PP=1 Executive Worker PP=2 Proprietary User PP=3 Ordinary User Each instruction and protected function (for example, accessing Executive GRS) is assigned a privilege value, and that instruction or function is performed only when the PP of the executing activity is less than or equal to the assigned value; otherwise, an Invalid Instruction or Reference Violation interrupt occurs. Processor Privilege for other functions is defined as follows: PP=0 Writing Executive GRS PP≤1 Using B16–B31 (extended mode only) Absolute addressing (basic mode only) PP≤2 Reading Executive GRS
DB16	Basic Mode	Basic Mode designator when cleared indicates extended mode.
DB17	EXEC Register Set Selection	When the EXEC Register Set Selection designator is cleared the user register set is selected; when set, the Executive register set is used. When DB17 is set, meaningful programming cannot be achieved if PP is not zero.
DB18	Carry	The Carry designator is set to 1 if a carry condition is detected on the execution of a single- or double-precision integer arithmetic instruction. This bit may also be set by the BIC, BICL, AND EDDE instructions.
DB19	Overflow	The Overflow designator is set to 1 if an overflow condition is detected on the execution of a single- or double-precision integer arithmetic instruction. This bit may be set by the four decimal arithmetic instruction as well as by the BN, BBN, BIC, BICL, BMTC, and EDDE instructions.
DB20		Reserved

Table 2-3. Designator Register Bit Description (continued)

Bit Number	Bit Names	Description
DB21	Characteristic Underflow	The Characteristic Underflow designator is set to 1 when the characteristic of a floating-point result is less than -200_8 (single-precision) or -2000_8 (double-precision).
DB22	Characteristic Overflow	The Characteristic Overflow designator is set to 1 when the characteristic of a floating-point result is greater than 177_8 (single-precision) or 1777_8 (double-precision).
DB23	Divide Check	The Divide Check designator is set to 1 when the characteristic of a floating-point result is less than -200_8 in a single-precision instruction, or if DB29 or DB34 is clear and the characteristic is less than -2000_8 in a double-precision instruction. If the characteristic is less than -2000_8 in a double-precision instruction, and DB29 and DB34 are both set, DB21 is unchanged.
DB24-28		Reserved.
DB29	Arithmetic Exception Enable	If the Arithmetic Exception Enable designator is 0, and if an arithmetic exception occurs (DB23, DB22, or DB21 set to 1), the specified A-registers are cleared to zeros and no interrupt occurs. If DB29 is 1, and if an arithmetic exception occurs, the specified A-registers are left unchanged (except as specified by DB34), and an interrupt occurs.
DB30	Floating-Point Residue Enable	If the Floating-Point Residue Enable designator is 1, it enables the residue store for single-precision floating-point instructions. In extended mode (DB16 clear), this designator is reserved (i.e., treated as zero). *NOTE: If a residue of -0 is generated as the result of a FA or FAN operation, +0 is stored in A_{a+1} when the operation yields an overflow in the result or an underflow in the residue.*
DB31	Bank Descriptor Register Selection	If the BDR Selection designator is 1, B12 and B14 are selected as the primary pair of Bank Descriptor Registers; when DB31 is zero, B13 and B15 are selected as the primary pair. The primary pair is selected over the secondary pair if the storage limits overlap. DB31 is toggled during a jump that is taken (excluding User Return) if the jump to address falls exclusively within the limits of the secondary pair. DB31 is not altered when an interrupt occurs. This designator is used during basic mode addressing only.
DB32	Quarter Word Selection	Quarter Word Selection Designator (see Figures 2-1 and 2-2). The j-field values of 4, 5, 6 and 7 each have more than one meaning, depending on the value of Quarter Word Selection Bit 32 (DB32). If DB32 is 0, these values

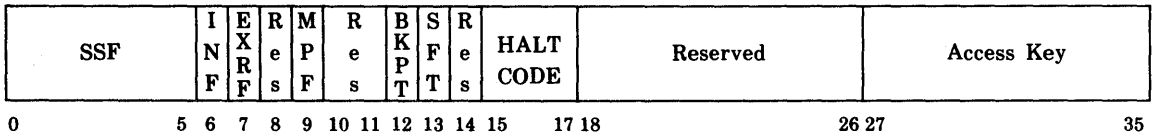
Table 2-3. Designator Register Bit Description (continued)

Bit Number	Bit Names	Description
		indicate certain half-, third-, and sixth-word operations. If DB32 is 1, j-values of 4 through 7 indicate quarter-word operations.
DB33		Reserved.
DB34	Double-Precision Floating-Point Underflow Mask	If the Double-Precision Floating-Point Underflow Mask designator is 0, an Arithmetic Exception interrupt occurs if characteristic underflow is detected during the execution of a double-precision floating-point instruction. The contents of the specified A-registers remain unchanged. When DB34 is 1, the interrupt does not occur; however, the contents of the specified A-registers are cleared to zeros and the normal instruction sequence is continued.
DB36		Reserved.

All reserved bits except those explicitly reserved for software (DB1 and DB2) are ignored when the Designator register is loaded (LD instruction, UR instruction, etc.) and are cleared to zero when the register is stored (SD instruction or interrupt).

2.13.3. Indicator/Key Register

The Indicator/Key register contains mid-execution control indicators, exigent pending interrupt indicators, and an access key value. The format of the register is:



where:

Bits 0–5 — Short Status Field (SSF) contains interrupt status when found in an interrupt stack frame; not used by the User Return (UR) instruction. The interpretation of SSF depends on the interrupt.

Bits 6–9 — Mid-Instruction Descriptor (MID) flags defined as follows:

Bit 6 — The instruction in F0 (INF) is set in the Indicator/Key register upon instruction initiation; it is cleared upon successful instruction completion. Consequently, INF is set in an Interrupt Stack Frame for all cases where the interrupt occurred following instruction initiation

and prior to instruction completion. Backing-up an instruction to the interrupt point at the start of the instruction does not result in the INF being cleared.

When INF is set to one, the activity state packet describes a mid-execution state (interrupted instruction) and the next instruction to be executed is in F0 within the activity state packet. In this case program address register points to the original instruction that was interrupted.

When INF is zero, the activity state packet describes the uninitiated instruction state. The Program Address register points to the next instruction to be executed, and the contents of F0 is undefined.

The INF being set causes pending of certain events, such as various Break and Breakpoint interrupts until the current instruction is complete. (See 2.14.)

Bit 7 If EXR instruction target in F0 (EXRF) is set to 1, the activity state packet describes the mid-execution state of an EXR instruction. EXRF is always set in conjunction with INF; if INF is clear when EXRF is set, operation is undefined. Because EXR is an extended-mode-only instruction, EXRF cannot be set in a basic mode activity state packet (one with DB16=1) except by operating system software intervention. When a UR instruction is executed with an activity state packet that has INF and EXRF set in the Indicator/Key register, and DB16 set in the Designator register, the result will be an invalid instruction (Class 14) interrupt.

There are three valid combinations of INF and EXRF. They are summarized as follows:

INF	EXRF	Hardware Action on Resumption
0	0	Normal. Fetch Instruction addressed by program address register and execute it.
1	0	All mid-execution states except EXR. Obtain instruction from F0 (rather than using program address register) then proceed normally.
1	1	EXR mid-execution. Enter normal EXR logic at the point where the target instruction has just been fetched (but not decoded), using F0 as the target instruction.

NOTE: In the special case where EXR is itself the target of an EX instruction, mid-execution state will have EXRF clear until the first interrupt point after the EXR instruction has been fetched.

Bit 8 Reserved

Bit 9 Macro-Procedure in FO (MPF). Indicates that a macro procedure was interrupted in mid-execution when MPF is set with INF also set. This bit is ignored if when INF is clear.

Bit 10–11 Reserved

Bits 12–17 Contains (Pending) interrupt and halt conditions. Once set, these bits are not cleared until the corresponding interrupt has been selected. The bits have the following interpretation:

Bit 12 Breakpoint (BKPT) Register Match Condition. This condition is not deferrable (by DB13), but is held until taken. This cannot happen before the end of the current instruction. Breakpoint halts are handled identically to Breakpoint interrupts, the only difference being that whenever the Breakpoint interrupt would have occurred, the processor halts instead of performing the interrupt sequence.

The breakpoint hardware mechanism sets this PEND bit directly upon detecting a match.

Bit 13 Soft (SFT) Break (set only by Executive). This condition is deferrable (by DB13) and held until the end of the current instruction and then held until taken, at which time it is cleared.

The Breakpoint and Soft Break interrupt conditions are preserved in this field across mid-execution interruption and return, but those interrupts cannot occur at mid-execution interrupt points; thus at mid-execution interrupt points, it is possible for lower priority interrupts to occur even though Breakpoint and/or Soft Break interrupts are pending. If a P Breakpoint interrupt occurs, the breakpointed instruction is executed prior to taking the breakpoint interruption.

The Breakpoint and Soft Break interrupt conditions may also be pended at between-instructions interrupt points, but in this case due only to pre-emption by higher priority interrupts (or DB13 being clear in the case of Soft Break). In either case of pending, it should be understood that the PEND bits are captured on the Interrupt Control Stack and then cleared in the hard-held activity state packet; that is, the pended condition does not persist after the interrupt is taken, but upon the User Return to the captured activity state packet.

Bit 14 Reserved

Bits 15–17 Halt Code. Whenever the processor is halted, the reason is indicated here prior to halting. The codes are:

0 No true halt condition exists (IP could be halted during IPL or various maintenance modes).

1 Halt Jump instruction.

2 Operating system software assumes this code reflects an auto recovery.

3 Breakpoint.

4 Software-causeable failure during interrupt sequence. May be due to hardware or software. Results from one or more of the following interrupt sequence failures: ICS overflow; ICS boundary violation;

limits violation on interrupt vector access; addressing exception on the User Return instruction.

5 Catastrophic hardware failure. This case includes, but is not limited to: Class 1 interrupt condition when DB6 is set (that is, a second hardware error before software has handled the first one); hardware failure during interrupt sequence. Some detected failures may be severe enough as to preclude proper setting of the halt code.

6 Reset failure. Demand interrupt condition while Reset Indicator is set.

7 External halt (operator pressed STOP).

The codes are in priority order, the lower-numbered condition having precedence. If resumption of operation of the IP commences with an interrupt sequence, the halt code is captured in the activity state packet on the ICS, then cleared; otherwise it is cleared immediately.

NOTE: Results are undefined if an attempt is made to resume operation of a IP (by pushing START) after halts other than for codes 0, 1, 3, or 7, without an intervening initialization.

A halted IP can be directly restarted by software only with a UPI action initiated by another IP in the application and only when stopped via a Halt Jump (code 1).

For code 4 and those cases of code 5 involving interrupt sequence failures, the exact degree of progress through the sequence is undefined.

Bits 18–26 Reserved

Bits 27–35 The current ACCESS KEY associated with the activity. It is compared to the access lock found in the referenced B-register to select between the GAP and SAP fields for use in access rights validation.

2.13.4. Quantum Timer

The Quantum Timer is a signed ones complement 36-bit internal register. A copy of the timer is maintained in the activity save area Each IP instruction has an internally stored base execution time, which is deducted from the Quantum Timer when the next instruction is loaded into the primary instruction holding register. When the timer is decremented to zero (or becomes negative) a Quantum Timer interrupt is generated. The timer does not stop decreasing at zero; rather, it continues decreasing until the interrupt is fielded by the Executive and the timer value is reset. This provides more accurate accounting in that when the timer reaches zero in the middle of a lengthy instruction, the balance of that instruction is not unaccounted.

The Quantum Timer is used by the Executive to control the amount of time used by an activity, and to implement its resource allocation algorithms.

2.13.5. Current Instruction Register (F0)

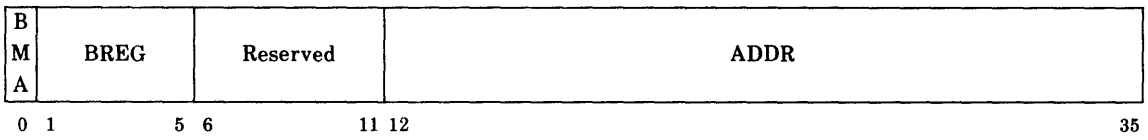
The F0-register contains the instruction being executed when the interrupt condition was detected or is undefined.

2.13.6. Interrupt Status Words (ISW0-2)

Three Interrupt Status Words (ISW0-2) are used for reporting interrupt status conditions. ISW0 normally contains information on the addressing mode, base register selected, and relative operand address for synchronous interrupts. ISW1 contains auxiliary status for program and MSU errors. ISW2 contains auxiliary status for processor faults.

The format and interpretation of ISW0 is:

ISW0



where:

BMA (Bit 0) — Basic Mode Address. Set if basic mode address generation was invoked during address translation. This includes the case where an extended mode instruction specified a base register that had the B-bit set as well as when the processor is executing in basic mode.

BREG (Bits 1-5) — Base Register. The base register selected during address translation. This may be different from the one specified by an extended mode instruction when BMA is set. The base register is undefined when the instruction that provoked the interrupt specified absolute addressing or a GRS operand.

Reserved (Bits 6-11) — Not used.

ADDR (Bits 12-35) — Address. The relative address computed either by adding u+Xm or d+Xm as determined by processor execution mode and the instruction operation.

For indirect addressing, ADDR and BREG, when not otherwise undefined for the given interrupt, will correspond to the last successful operand fetch, that is, ADDR and BREG will correspond to bits 14-35 of F0.

- Auxiliary status for program and MSU errors: ISW1
- Auxiliary status for processor faults: ISW2

Interrupt classes 1, 22, 23, and 26 relate to IP and MSU hardware malfunctions and abnormalities. (I/O processor errors are reported via UPI interrupts.) See 2.21.2.

2.14. Instrumentation State

Instrumentation state includes the Breakpoint register, the Jump History (JH) facility and the software performance monitoring controls.

NOTE: The use of instrumentation facilities for purposes other than instrumentation or debugging is not recommended and voids assurance of compatibility on future architectures; for example, using breakpoints as flags to drive program logic is not recommended.

2.14.1. Address Breakpoint

Address breakpoint provides the method for software to detect certain types of reference to a specified address. Breakpoint operation is controlled, match conditions are enabled, and the breakpoint address is specified by the breakpoint register.

2.14.1.1. Breakpoint Operation

Each main storage absolute address generated is classified as:

1. Program fetch instruction – addresses generated from the PAR, to resolve indirect addresses, and to obtain the operand of an EX or EXR instruction; or

2. Read – addresses generated to obtain data from storage other than the addresses classified as program fetches; or

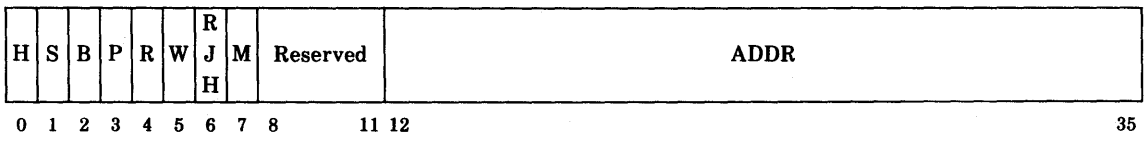
3. Write – addresses generated to store data into storage.

   NOTE: Addresses generated to obtain GRS operands are specifically excluded.

If the generated address is equal to one held in the breakpoint register and the classification is enabled in the breakpoint register, a breakpoint-match condition occurs. This condition is held in the PEND field of the ASP until the next between-instruction interrupt point. At this point, if the H-bit in the breakpoint register is set, the processor halts; otherwise, a breakpoint interrupt occurs.

2.14.1.2. Breakpoint Register Format

The breakpoint register is loaded by a Load Breakpoint Register (LBRX) instruction and controls the operation of the address breakpoint and jump history facilities. The breakpoint register format and the meaning of its fields are:



where:

H (Bit 0) If Halt enable (H) is set, the IP halts on a breakpoint match condition. If a higher priority interrupt occurs in the same instruction, the match condition is pended in the activity state packet. If clear, a Breakpoint interrupt occurs.

S (Bit 1) If Stack full interrupt enable (S) is set, the detection of a jump history full condition results in a Jump History interrupt. If clear, the condition is ignored.

B (Bit 2) If stacking disable on interrupt (B) is set, the recording of jump history is discontinued when any interrupt occurs, but only after the interrupt entry is made. If clear, the jump history operation is not affected by interrupts. The entry caused by an interrupt is entered regardless of the value of this bit.

P (Bit 3) If Program instruction breakpoint enable (P) is set, the addresses generated to fetch instructions and resolve indirect addresses are tested for quality with the Breakpoint register ADDR field. If clear, these addresses are ignored. Operands of the EX and EXR instructions are treated as instructions for this purpose.

R (Bit 4) If Read breakpoint enable (R) is set, the addresses generated to obtain (read) words from storage are tested for equality with the ADDR field. If clear, these addresses are ignored.

W (Bit 5) The Write breakpoint enable (W) is identical to R except that the addresses must be generated to store (write) words into storage. This includes the operand address to SLJ.

RJH (Bit 6) If Reset Jump History (RJH) is set, and "M" is not set, the JH0 value and pass flag associated with the jump history are reset to zero when the LBRX is executed. If clear, the JH0 value and pass flags are not modified. This field is not interpreted when the LBRX is executed with M=1.

M (Bit 7) If Main storage JH-buffering enable (M) is set, jump history entries are written to the main-storage buffer defined by B_{23}. Also, if set, the content of GRS locations 040 - 077 is undefined. If clear, JH entries are written in GRS locations 040 - 077. The RJH field is not interpreted when M is set.

Bits 8-11 Reserved

ADDR (Bits 12-35) The Address (ADDR) field contains the absolute address that causes a breakpoint match condition if equal to an instruction, a read address, or a write address as controlled by the P-, R-, and W-bits, respectively. Processor action upon the detection of such a condition is specified by the H-bit.

2.14.2. Jump History

The jump history is a circular buffer with optional wrap-around detection maintained either in GRS or in main storage and containing: the addressed within the instruction stream where a jump or interrupt occurred, certain associated control information when appropriate, and entries explicitly created using a special instruction.

2.14.2.1. Operation

Jump history entries are created in four circumstances:

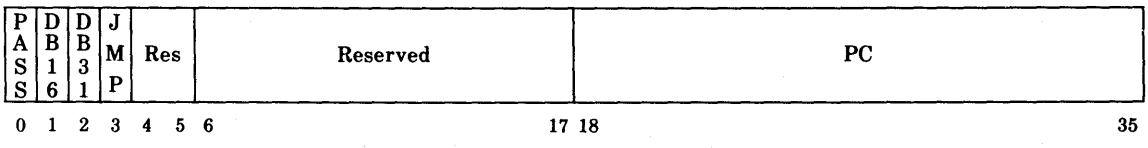
1. when an interbank transfer instruction is executed - all cases of CALL, RTN, GOTO, LIJ, LDJ, LBJ, and UR,
2. when a UR instruction is executed;
3. when interrupt is taken;
4. when a simple jump instruction is executed (only if jump is taken for conditional jumps), or
5. when a CJHE (Create Jump History Entry) instruction is executed.

In cases 1, 2, and 3, two entries are actually made. The first (a jump entry) records the occurrence of a transfer of control. The second (a control entry) records the L,BDI being displaced. When two (or more) interrupts happen to be processed in mid instruction at the same interrupt point, there is no jump entry between the control entries and the second and subsequent control entries are specially marked.

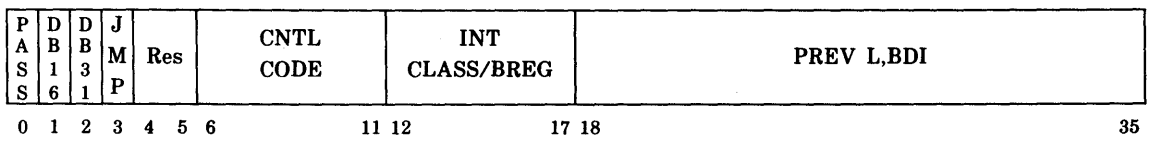
2.14.2.2. Entry Format

The format of a jump history entry is independent of the destination (GRS or main storage) of the entry. The format for each type of entry is as follows:

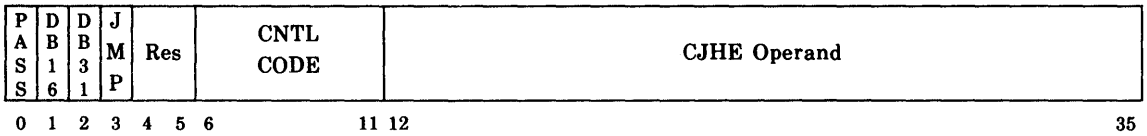
Jump



XFER/INT



CJHE



where:

Bit 0 — PASS flag specifies which of the two possible jump history groups this entry is part of and used by software interpreting the jump history to determine the most recent entry. Entry is found in GRS.

Bit 1 — Basic/Extended flag is a copy of DB16 when the entry was created.

Bit 2 — Basic mode base register pair selector is a copy of DB31 when the entry was created.

Bit 3 — If the Jump (JMP) entry flag is cleared, this is a control entry made by a CJHE instruction, an interrupt, or an interbank transfer instruction (see CNTL CODE field). If set, this is a jump entry created by a jump instruction or the jump portion of an interrupt or interbank transfer operation. The value of this bit also determines the interpretation of remaining bits.

Bits 4,5 — Reserved

Bits 6–11 — Control Code (CNTL CODE) field is zero when the JMP bit is set but specifies the source of the entry when JMP is clear as follows:

Value	Entry Caused By
0	Unused
1	Interrupt
2	Interrupt with no preceding jump entry
3	CJHE instruction
4	CALL instruction
5	RTN instruction
6	GOTO instruction
7	LIJ instruction
8	LDJ instruction
9	LBJ instruction
10	UR instruction

When this field indicates an interrupt entry (values 1 and 2); the INT CLASS field contains the interrupt class.

Bits 12–17 — Interrupt Class (INT CLASS/BREG) field is only meaningful when the CNTL CODE field indicates that this is a control entry generated by an interrupt.

Bits 18–35 — Program Counter (PC) field contains the PC value (from program address register) of the jump instruction or interrupt point that caused the creation of the entry. This value points to the jump instruction for jump created entries, to the instruction interrupted for mid-instruction interrupt created entries, or to the next instruction for between-instruction-interrupt created entries.

Bits 18–35 Previous L,BDI (PREV L,BDI) field contains the L,BDI of the bank displaced and is only meaningful when an interbank transfer instruction or interrupt caused the creation of the control entry. This field is undefined when the entry is created by a UR instruction executed in the basic mode.

Bits 12–35 CJHE operand field contains the low order 24 bits of the operand of the CJHE instruction that caused the entry.

A non-jump entry with the CNTL-CODE field equal to zero is to be ignored by software when interpreting jump-history contents. Such entries are used to contain the jump-history-offset (JHO) value, the JH-full interrupt-pending condition, and the wrap-around word. (See 2.14.2.3 for details.)

2.14.2.3. Main Storage Buffer Operation

When the M-bit is set in the breakpoint register, main storage jump-history buffering is enabled (see 2.14.1.2), and the contents of GRS locations 40_8–77_8 are not defined. The main storage buffer is described in B23. An unsigned current jump-history-offset (JHO) value is located in the lower-order 18 bits of word 0 of the main storage buffer. The JHO is accelerated from the main storage buffer upon execution of LBRX with M=1 and maintained internally by the IP. It is decelerated back to the main storage buffer upon the occurrence of any interrupt while M=1 in the breakpoint register, and upon execution of LBRX with M=0 when the previous breakpoint register setting had M=1.

The LL field in B23 must contain zero for proper operation. The S-bit in the base register must be clear.

Entries are placed in the buffer according to the following algorithm:

1. Address generation (using current JHO) storage-limits check. A storage limits failure here results in a Class 8 (reference violation) interrupt (see 2.21.2.3). Whenever a reference violation is detected here, a JH full condition is detected at step 4, and the JH full interrupt is taken before the reference violation is taken.

2. Increment JHO by 1.

3. Store the current word of the current entry in the main-storage buffer at the address computed in step 1.

4. Limits check on the updated JHO computed in step 2.

   a. Set JHO = 1.

   b. If step 3 above was for the first word of a two-word entry write the second word to word 1 of the main storage buffer, otherwise write zero to word 1 of the main storage buffer. This prevents the second-word of a two-word entry from ever being lost, and in effect, that word 1 of the main storage buffer is reserved for wrap-around.

   c. Increment JHO by 1.

   d. If the breakpoint register S-bit is set, disable jump-history operation and indicate that a jump-history-full interrupt condition exists.

If a JH-full interrupt cannot be processed due to a higher priority interrupt request or due to DB13=0 the JH-full interrupt is deferred until either interrupts are allowed (DB13=1) or until

the LBRX instruction es executed by the IP. When the LBRX instruction is executed if a JH-full is currently deferred, the JH-full interrupt is "pended" by writing the SSF value to bits 12-17 of word 0 of the main storage buffer described by B23; if M=1 in the new breakpoint register value, and if there is a non-zero value in bits 12-17 of word 0 of the main storage buffers described by B23, a JH-full interrupt, with SSF= bits 12-17 word 0, occurs following the LBRX instruction, or is deferred if DB13=0. When the interrupt is deferred (because BD13=0), the entering of jump-history entries is disabled until DB13=1. Bits 12-17 of word 0 of the main storage buffer described by B23 are cleared to zero by the LBRX instruction after they have been inspected for a pending interrupt condition. This means that software cannot switch main storage JH buffers with single LBRX instruction. In order to switch main storage JH buffers, software must execute the LBRX instruction with M=0 (to "pend" any existing deferred JH-full interrupt into the old buffer), followed by LBE B23 for the new buffer, followed by LBRX with M=1 to initiate jump-history operation into the new buffer (and detect any pending JH-full interrupt in the new buffer).

NOTES: 1. *Carefully note the above discussion concerning switching main storage buffers.*

2. *A user main storage buffer contains some superfluous entries caused by interrupts and UR instructions. These entries may not be of interest to users; the CNTL CODE field in the non-jump entries provides an easy means for a jump-history editing program to bypass these entries.*

3. *An EXEC JH (GRS or main storage) will not contain the interrupt and UR entries that are placed in user JH bank. This deficiency can overcome by executive software (if desired) by usage of the CJHE instruction in interrupt handlers and in the dispatcher.*

4. *Note again the restrictions placed upon B23 main storage JH buffer description: the LL field must contain zero, the S-bit in the base register must be clear. The Executive must enforce these restrictions, or main storage JH will not function properly. Main storage JH operation is undefined if the S-bit is set in B23. Reference-violation interrupts occur during JH operation if the B23 limits are not properly set up.*

5. *When main storage jump-history operation is initialized by software, the JHO field buffer of word zero is set to zero.*

6. *When GRS JH is in operation and JH-full interrupts are enabled, the second word of a two-word entry is lost if the GRS JH buffer became full as a result of the entering of the first word of the entry.*

2.14.3. Software Performance Monitoring

If set, DB0 indicates that Software Performance monitoring will occur. Three designator bits, DB3, DB4, and DB5, are used along with DB14-15 for software monitoring. The three bits are software controllable by the LD instruction. These designator bits in conjunction with DB14-15 provide eight software state indicators in both user and EXEC mode and provide considerable flexibility in profiling selected software packages or portions of software.

The above bits are sampled and a counter associated with the appropriate combination is increased. The 16 counters can be used to accumulate time spent in a particular state. The combinations of values of the Processor Privilege (PP) and software performance monitor designators are:

PP	DB3	DB4	DB5	Meaning
0 or 1	0	0	0	Interrupt routine – forced by hardware on interruption.
0 or 1	1	1	1	Idle – tested by external interrupt distribution hardware.

2.15. Instruction Interrupt Points

Interrupts occur at well-defined interrupt points during instruction processing. The machine state that is in effect (or captured on the interrupt stack) upon occurrence of the interrupt reflects the logical progress of the interrupted instruction sequence so as to allow proper resumption of that sequence after the interrupt has been processed.

Exceptions to this are limited to unretryable machine faults (in which case the interrupt status indicates that the faulting sequence cannot be retried) and a few special cases deemed harmless to software (such as spurious stack writes, etc.). Hardware, upon detection of an interrupt condition, either defers occurrence until an interrupt point is reached if the condition is classified as a non-fault or backs up the machine state to the previous interrupt point if the condition is classified as a fault.

An interrupt point always exists between instructions, and most interrupts occur then. The captured machine state reflects the interrupt point between instructions, and the captured program address register usually points to the next instruction. interrupt sequence.

Most instructions have no mid-execution interrupt points. However, certain instructions are potentially so lengthy and/or complex that neither backing out nor waiting until completion is feasible for handling the interrupts that can occur and thus must be interruptible in the middle of execution. These instructions are:

- All instructions using indirect addressing (via the instruction i-bit). An interrupt point exists just prior to fetching the target word for each iteration of indirection.
- Execute (EX).
- BT and all search instructions. An interrupt point exists between each iteration, assuming there is more than one iteration.
- Bit Move (BIM), Bit Compare Long (BICL), and Bit Move with Translation Control (BMTC).
- Execute Repeated (EXR).

Jumps require special discussion. A jump instruction is technically an instruction that loads P (of the program address register) with a value, usually U. The jump instruction is then complete (INF cannot be set) and an interrupt point is reached. The next instruction commences by using the new contents of P to fetch an instruction; this is consistent with what happens if an interrupt occurs following the jump and the interrupted sequence is resumed via a User Return instruction. The special consideration is this: in basic mode (DB16=1), a jump may alter DB31 (BDR Selection) depending on the target address; this is done before the interrupt point, since UR is required to restore the activity state packet exactly, and the fact that a jump occurred is lost when the interrupt is taken. Hardware translates the target address prior to the interrupt point, to the extent of determining whether to toggle DB31.

2.16. GRS Conflicts

There are no undetected inter-instruction register (GRS) conflicts for any user visible (PP=2 or 3) instructions in either basic or extended mode. The IP operates as if each instruction is completely executed before the next instruction commences, as far as GRS is concerned.

Inter-instruction conflicts are cases where a single instruction executed in extended mode specifies the same GRS location more than once. The ways a GRS location may be specified are:

- a-field (including a+1, etc.)
- x-field (including x+1, etc.)
- ja-field on JGD instruction
- operand addresses (U, U+1, etc.)
- implicit or indirect operand (for example, CALL modifies EXO; LRS/SRS A_a specifies GRS areas; field instruction descriptors specify X-registers; repeats use R1; ACEL/DCEL accesses all user GRS, etc.)

For non-interative instructions (all except EXR, BT, and the Character Manipulation instructions) the extended mode rules are:

1. The operand address (U) is generated only once, using X_x, before any possible GRS modification; multi-word operand addresses (for example, U+1 for DL) are all calculated according to the original contents of X_x.

2. X-incrementation may occur at any time with respect to other GRS accesses, constrained only by rule 1. Thus, operation is undefined if x-incrementation is specified for a register that is also referenced in any other way except address generation.

3. If the same GRS location is specified to be modified in two different ways, operation is undefined.

4. Multi-word GRS operands (specified by U, for example, on DS, etc.) are fetched and stored in undefined order, both with respect to each other and any associated GRS references specified in other ways, constrained only by rule 1.

5. When the order of reference to GRS is explicitly given as part of the basic instruction definition, that order applies regardless of rules 2, 3, or 4; for example, TLEM specifies that the test is performed prior to incrementing X_a. However, the definition of DL is not precise with regard to order, since both words could be fetched first and then both written to GRS, or it could be treated as two sequential load operations, without changing the basic semantics of the instruction. Also, arithmetic and scaling operations clearly require acquisition of the entire operand before any result can be determined, so instructions like DA or DLSC or DFM are not affected by rule 4.

2.17. Control Structures

2.17.1. Interrupt Control Stack

The interrupt control stack stores processor state and status when an interrupt occurs. The size of frames on the interrupt control stack is always assumed to be 16 and interrupt processing hardware ignores but does not modify the content of X_i. There is no defined method for removing frames from the interrupt control stack . Several methods are available, including SELL, UR (with index register incrementation specified), NOP (again, using index register incrementation) or any of the instructions that modify index registers explicitly (AX, ANX, etc.).

The interrupt control stack pointers are:

- B22 - interrupt control stack limits and base
- EX1 - current stack position address (always treated as 18 bits)

The Executive software establishes and maintains an interrupt control stack for each IP.

2.17.2. Return Control Stack

The return control stack holds dynamic procedure linkage information in a protected manner. The CALL instruction adds frames to the return control stack and the Return (RTN) instruction removes frames from the return control stack. The size of frames on the return control is always assumed to be 2 and the contents of X_i is ignored but not modified by the CALL and RTN instructions.

The return control stack pointers are:

- B21 - return control stack limits and base
- EX0 - current stack position address (always treated as 18 bits)

The Executive software establishes and maintains a return control stack for each activity.

2.17.3. User Stacks

The user may define stacks and manipulate them with the BUY and SELL instructions. The user selects either the B- or X-register. The stack frame size is also user defined and may be negative; if so, care must be exercised in choosing the starting (empty) address to ensure proper detection of stack overflow or underflow.

User stacks, unlike the interrupt control stack and return control stack, may be accessed by using either 24-bit addressing or 18-bit addressing.

User manipulation of base register B1 is restricted to facilitate implementation of an activity-local storage stack. However, the BUY and SELL instructions may be used to manipulate this stack in the normal manner.

2.17.4. Activity Save Area

The activity save area is not an architectural entity. However, since many of its components have architectural definition, and since the architecture attempts to anticipate Series 1100 Operating System usage of those components, this section treats them as a whole in order to give a coherent overview of intended maintenance of activity environment. Distinctions between architectural requirements and mere intention are carefully drawn. See Figure 2-7 for the format of the activity save area.

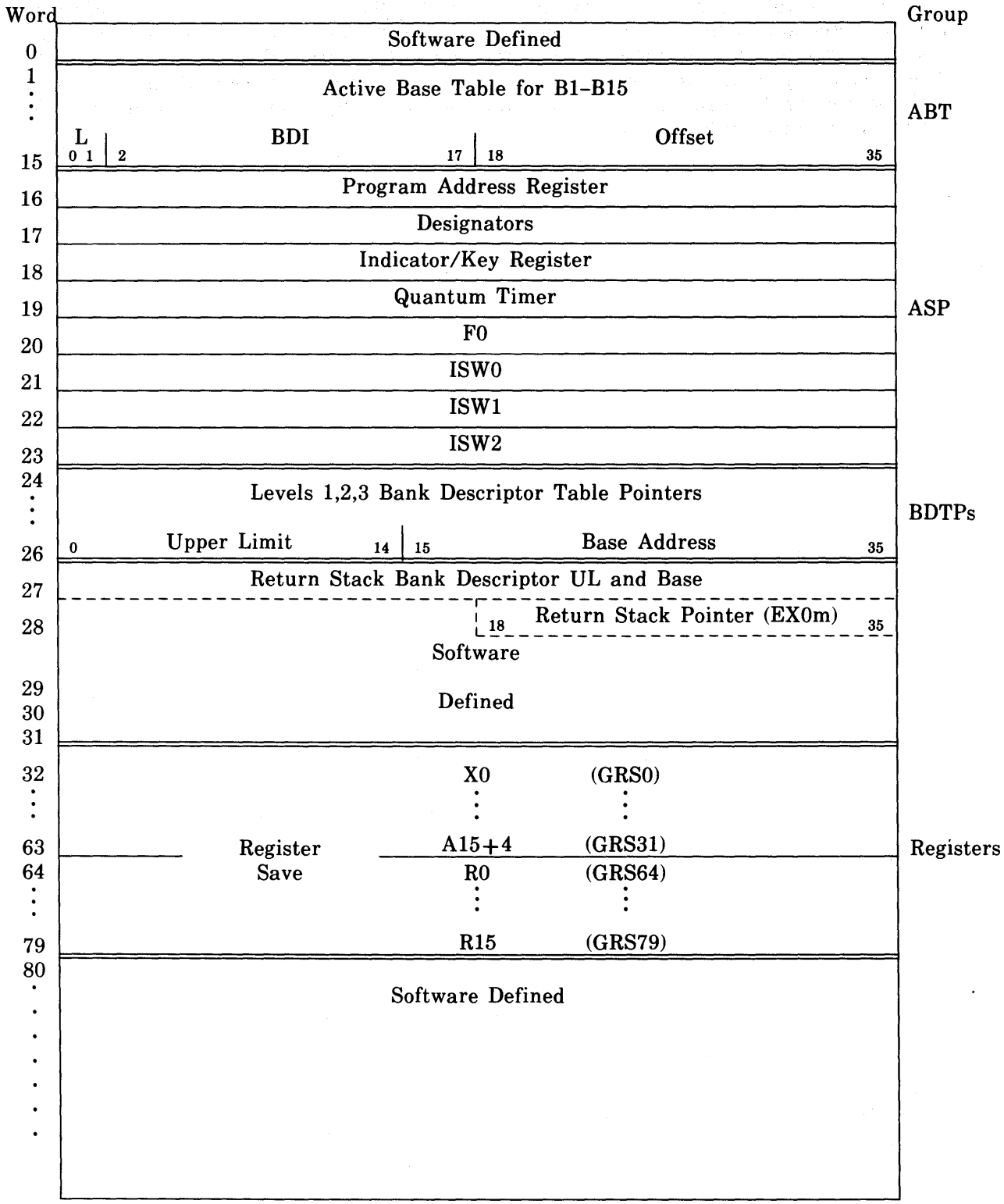


Figure 2-7. Activity Save Area

Group	Word(s)	Description
-----	0	Software (all such fields may be used by software in any manner). Word 0 is a buffer control word.
ABT	1-15	Active Base Table. This is a copy of the virtual address currently active (loaded) in B1-B15. The ABT always contains sufficient information to establish the complete explicit address space for an activity, with the exception of B0; the specification of the descriptor loaded into B0 is contained in the activity state packet. The Base Register manipulation instructions that involve implicit access to the ABT (for example, the LBJ, LBU, etc. instructions) use B20 as the ABT base with the base number (1-15) as the offset. Software has B20 loaded with a bank that maps the entire activity save area, when there is a possibility of such instructions being executed. The LAE instruction is used to restore B1-B15 by using an offset of zero in conjunction with B20 (that is, LAE 0,,B20). Saving the B-registers or the ABT is not ordinarily needed, since the latter is automatically updated by hardware to reflect changes to B1-15 during program execution.
ASP	16-23	Activity State Packet. This block of eight words corresponds exactly to both the first eight words of an Interrupt Stack Frame and the operand of a User Return (UR) instruction. Software saves this activity state by moving it from the Interrupt Stack to the activity save area, and restores it via a User Return instruction. ISWs need not be saved.
BDTPs	24-26	Words 24-26 define the Bank Descriptor Tables for levels 1-3, respectively. Each word has the format of Word 1 of a Bank Descriptor. Since the actual Bank Descriptor Table Pointers (BDTPs) for levels 1, 2, 3 lie in B17-B19, software restores B17-B19 by using LBE1 instructions that provide direct loading of the base and upper limits. Saving of BDTPs is not ordinarily needed. The location and ordering of the BDTPs has no architectural significance except in relation to the TVAE instruction.
-----	27-31	Software. Word 27 has been chosen (any free word is permissible) as the place for software to maintain the Return Control Stack Bank Descriptor Base. Just as for the BDTPs, this word is in a format that facilitates use of the LBE1 instruction. The Return Stack is implicitly based by B21. Software restores B21 by executing LBE1. Saving of the Return Control Stack Base is not ordinarily required. Software sets Word 0 of B17, 18, 19 and 21 to zeros except possibly access permissions and the access lock. The right half of Word 28 has been chosen (any free halfword is permissible) as the place for software to save the Return Control Stack Pointer from EX0 (GRS 140_8); only the modifier portion is saved, since the increment field is ignored by the CALL and RTN instructions.

NOTE: Software must save and restore EX0 explicitly, for each user activity.

Registers	32–79	Register Save. For software to capitalize on the ACEL and DCEL instructions, any register save area must be in this format.
-----	80 and beyond	Software.

The locations of individual activity save area groups are related in one regard: the Test Virtual Address in Environment instruction (TVAE) takes an operand of 27 consecutive words having the order and format of activity save area Words 0–26, although not all fields are accessed.

2.18. Dayclock

The system has one Dayclock located in the SSP.

The Dayclock has the following logical components:

1. **Clock:** a 51-bit counter. The clock is updated by adding binary 100000 every microsecond. The five low order bits are a count of previous reads since the clock was last updated and thus are cleared on each update. It is expected that no more than 31 clock reads can possibly occur between updates – if the 32nd read does occur (that is, low order five bits are all ones) then the read must be delayed until the next update occurs; this mechanism guarantees that each read of the Dayclock yields a unique and increasing value.

2. **Comparator:** a 51-bit register, whose contents are compared to the high order 51 bits of the clock each time the clock is updated (that is, each microsecond). If the clock value becomes greater than or equal to the comparator value (as loaded by the Executive System), a Dayclock interrupt is generated and distributed via the Broadcast Interrupt Service Flag interrupt distribution mechanism. No further interruptions are generated by this Dayclock until the following events have occurred in sequence:

   a. The previous interruption has been accepted by a processor.

   b. The Comparator has been loaded by a Load Real Comparator (LRC) instruction.

   c. A greater or equal condition again exists.

3. **Speed Mode:** The clock has three speeds; normal, fast, and slow, with accuracy as follows:

Speed	Error Rate
Normal	±1/10000 (error does not exceed one part in 10000, about 8.5 seconds per day).
Fast	+1/E±1/10000 (clock gains one part in E, again with normal error not exceeding one part in 10000).
Slow	-1/E±1/10000 (clock loses one part in E, with normal error).
	where: $2^{9.5} < E < 2^{10.5}$

The Dayclock runs continuously when power is applied. When a Dayclock is powered up after any kind outage, its clock is set to all zeros, its comparator is set to all zeros, and its speed is set to normal. No manual controls are required or provided.

2.19. Arithmetic Operations

All arithmetic computation can be performed in either fixed-point, floating-point, or decimal mode. Fixed-point arithmetic instructions provide single-precision, double-precision, half-word, and third-word addition and subtraction, and fraction and integer multiplication and division. Floating-point instructions provide both single-precision and double-precision operation. Decimal computation of fixed-point numbers is provided. The arithmetic section also performs certain logical operations such as shifting and comparisons. The instruction word may be used to specify the transfer of any chosen portion of a word (half, third, quarter, or sixth) to the arithmetic section. The ability to transfer only the selected portion of a word minimizes the number of masking and shifting operations required.

The shift matrix in the arithmetic section permits the completion of an entire single-word shift operation in one main storage cycle time. By use of the matrix, the shift operation can shift a single- or double-word operand in either direction up to 72-bit positions.

2.19.1. General Operation

During the execution of logical and arithmetic instructions, the IP performs the following steps:

1. Transfers input data from instruction word specified storage locations or general registers to input registers in the arithmetic section.

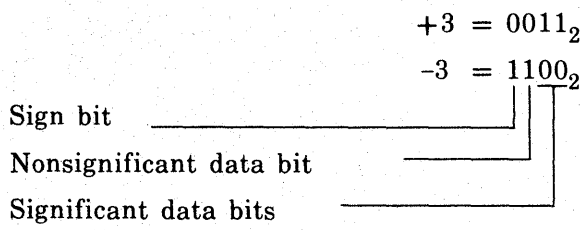
2. Performs the arithmetic operations of addition, subtraction (add negative), multiplication, division, skip detection, etc., as specified by the instruction word.

3. Transfers final results from the arithmetic section to general registers or indicate an instruction skip or jump condition.

2.19.1.1. Data Word

Numbers for the binary arithmetic computations are expressed in ones complement notation. The highest order binary bit represents the sign of the value contained in the remaining bit positions. If the sign bit contains a 0, the word is positive; 0's or 1's in the remaining bit positions represent significant data. If the sign bit contains a 1, the word is negative; 0's or 1's in the remaining bit positions represent significant data. A binary data word containing all 0's is referred to as positive zero (+0). A binary data word containing all 1's is referred to as negative zero (-0).

Example: (assume a 4-bit word length)



2.19.1.2. Data Word Complement

The ones complement of any binary arithmetic data word is obtained when all 0's in the word are changed to 1's and all the 1's are changed to 0's. An arithmetic data word of positive value, when complemented, becomes a negative value, and a negative value, when complemented, becomes a positive value.

2.19.1.3. Absolute Values

The absolute value of an arithmetic number is the magnitude of the number regardless of the sign.

Example:

Binary Value	Absolute Value
001110_2 $(+14_{10})$	001110_2 (14_{10})
110001_2 (-14_{10})	001110_2 (14_{10})

2.19.2. Main Adder Characteristics

The main adder of the IP arithmetic section performs single- or double-precision addition, subtraction, and half-word and third-word addition and subtraction. It also does single or double-word logical operations.

2.19.3. Fixed-Point Arithmetic Overflow and Carry Conditions

In fixed-point arithmetic, the execution of certain instructions can result in an overflow or a carry condition. During execution, the Overflow designator (DB19) and the Carry designator (DB18) bits are cleared to 0's; the overflow and carry conditions set bits DB19 and DB18, respectively, in the designator register. These bits can be sensed by certain other instructions. Each of these designators, when set to 1, remain in the set condition until the next time any one of the instructions in Table 2-4 is executed or until the Load Designator Register instruction is executed.

2.19.3.1. Overflow

An overflow condition is detected when one of the instructions in Table 2-4 is executed and the numeric value of the result obtained exceeds the maximum numeric value that can be contained in the register holding the final result. This is significant when the operands for an additive process are of the same sign or when the operands for a subtractive process have different signs. If overflow occurs in these cases, the sign of the result is unnatural. The condition of DB19 can be tested by executing either the Jump Overflow or the Jump No Overflow instruction.

2.19.3.2. Carry

A carry condition is detected when an end-around carry occurs during the execution of an instruction listed in Table 2-4. The detection of a carry condition indicates that a carry was propagated out of the sign bit position and automatically added into the low-order bit position. The detection of the carry condition is significant when programming multiple-precision

routines. In ones complement subtractive arithmetic, the carry condition can be equated to the no-borrow condition, and the no-carry condition to the borrow condition.

The condition of the carry designator can be tested by executing either the Jump Carry or Jump No Carry instructions. Table 2-5 lists the sign bit combinations for which the designator DB18 would be set to 1 indicating that a carry has occurred.

Table 2-4. Instructions that Condition the Carry and Overflow Designators

Function Code (Octal)	Instruction
f = 05 j = 00-15 a = 15	Add One to Storage
05,,16	Subtract One from Storage
07,00	Add Decimal
07,01	Double Add Decimal
07,02	Subtract Decimal
07,03	Double Subtract Decimal
14	Add to A
15	Add Negative to A
16	Add Magnitude to A
17	Add Negative Magnitude to A
20	Add Upper
21	Add Negative Upper
24	Add to X
25	Add Negative to X
33,05 Extend Mode	Floating Decimal Scale Pack
37,11 (BM) 73,14,12 (EM)	Bit Compare
37,13 (BM) 73,14,13 (EM)	Bit Compare Long
37,15 (BM) 72,10 (EM)	Byte to Decimal
37,16 (BM) 72,11 (EM)	Decimal to Byte
37,17 (BM) 73,16 (EM)	Edit Decimal
71,10	Double-Precision Fixed-Point Add
71,11	Double-Precision Fixed-Point Add Negative
72,12	Bit Normalize
72,14	Byte to Bit Normalize

BM = Basic Mode
EM = Extended Mode

Table 2-5. Sign Bit Combinations That Set the Carry Designator

Operation	Input Operation Sign		Resultant Sign
	Augend	Addend	
Addition	+	-	+
	-	+	+
	-	-	+
	-	-	-
	Minuend	Subtrahend	
Subtraction (Add Negative)	+	+	+
	-	-	+
	-	+	+
	-	+	-

2.19.3.3. Arithmetic Interrupt

The arithmetic section cannot independently cause a system interrupt. But when an arithmetic fault occurs, it generates a fault condition signal that allows the control section to set the appropriate designator bit. Other processor conditions in conjunction with those arithmetic fault conditions determine whether or not control generates an interrupt.

2.19.4. Fixed-Point Division

The process of dividing one fixed-point number by another consists of transferring the numbers to the arithmetic section, calculating a quotient and a remainder, transferring the properly signed quotient to a register and, if the remainder is to be saved, transferring the properly signed remainder to another register. All divide operations use the main adder and shifter.

2.19.5. Fixed-Point Multiplication

The arithmetic section contains a high-speed multiplier unit to handle multiplications. and ending cycles for input and output data adjustments.

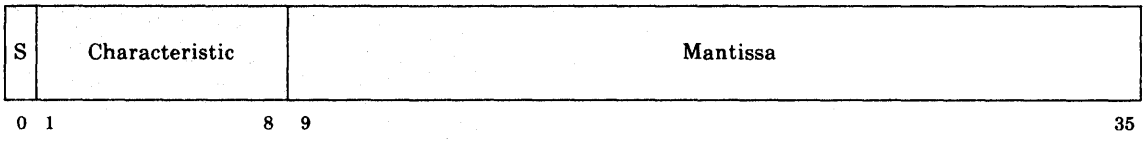
2.19.6. Floating-Point Arithmetic

Floating-point arithmetic handles the scaling problems that arise in computations involving numbers that vary widely in range. In floating-point arithmetic, the numbers are represented in a special format so that the computer can automatically handle the scaling.

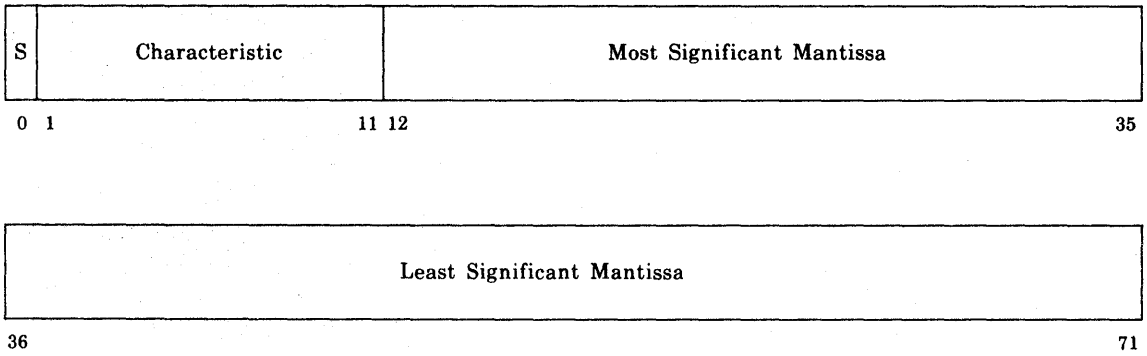
2.19.7. Floating-Point Numbers and Word Formats

Floating-point numbers in the instructions are represented in single-precision format as a 27-bit fractional quantity multiplied by the appropriate power of two, or in the double-precision format as a 60-bit fractional quantity multiplied by the appropriate power of two. The power of two is called the exponent. In machine representation, the exponents are biased to make them lie in the range of positive numbers or zero. These biased exponents are called characteristics. The fractional part is referred to as the mantissa. The two format types, single-precision and double-precision, are:

Single-Precision Floating-Point Format



Double-Precision Floating-Point Format



An explanation of the sign bit, characteristic, and mantissa follows:

- Sign bit - The sign bit expresses the sign (S) of the numerical quantity represented by the floating-point number.

  If S = 0, the numerical quantity is positive (+).

  If S = 1, the numerical quantity is negative (-).

- Characteristic - The characteristic represents both the numerical value and the sign of the exponent.

  1. Single-Precision Characteristic - The 8-bit characteristic of a single-precision floating-point number represents an exponent value in the range +127 through -128. The characteristic is formed by adding a bias of +128 (200_8) to the exponent. Table 2-6 shows the range of characteristic values and corresponding exponent values.

Table 2-6. Single-Precision Floating-Point Characteristic Values and Exponent Values

Decimal Values		Octal Values	
Characteristic	Unbiased Exponent	Characteristic	Unbiased Exponent
255	+127	377	+177
128	000	200	000
000	-128	000	-200

2. Double-Precision Characteristic - The 11-bit characteristic of a double-precision floating-point number represents an exponent value in the range +1023 through -1024. The characteristic is formed by adding a bias of +1024 (2000_8) to the exponent. Table 2-7 shows the range of characteristic values and the corresponding exponent values.

Table 2-7. Double-Precision Floating-Point Characteristic Values and Exponent Values

Decimal Values		Octal Values	
Characteristic	Unbiased Exponent	Characteristic	Unbiased Exponent
2047	+1023	3777	+1777
1024	0000	2000	0000
0000	-1024	0000	-2000

- Mantissa - The mantissa portion of a floating-point number represents the fractional part of the number. In the instructions, the fractional part is normalized so that the absolute values represented are greater than or equal to 1/2 but less than one. Zero cannot be represented in this range, and it is considered to be normalized as it stands. The binary point of a floating-point number is assumed to lie between the last bit of the characteristic and the first bit of the mantissa. The mantissa of a single-precision floating-point number contains 27 bits; for a double-precision floating-point number, the mantissa contains 60 bits. The mantissa need not be normalized for all instructions.

2.19.7.1. Single-Precision Floating-Point Numbers

A single-precision floating-point number can be derived from a positive decimal number.

Example:

Given number = $+12_{10}$

$+12_{10} = 1100_2 = .1100_2 \times 10_2^4$

Sign = + = 0

Characteristic = exponent + bias

$= 00\ 000\ 100_2 + 10\ 000\ 000_2$

$= 10\ 000\ 100_2$

Mantissa $= .110\ 000\ ....\ 000_2$

The format for the floating-point number is (sign included):

Sign	Characteristic	Mantissa
0	10 000 100	1100 0

0 1 8 9 35

$= 204600000000_8$

2.19.7.2. Double-Precision Floating-Point Numbers

A double-precision floating-point number can be derived from a positive decimal number following the same steps that were used for single-precision, with these two exceptions:

- A bias value of 2000_8 is added to the exponent to form the characteristic. For single-precision the value is 200_8.
- The mantissa is 60 bits instead of 27 bits.

2.19.7.3. Negative Floating-Point Numbers

A floating-point number can be derived to represent a given negative number as follows:

- Represent the given number as a positive floating-point number.
- Form the ones complement of the entire positive floating-point number.

Example:

Given number $= -12_{10}$

The single-precision floating-point number for $+12_{10}$ (including sign) is $204\ 600\ 000\ 000_8$.

The single-precision floating-point number for -12_{10} (including sign) is $573\ 177\ 777\ 777_8$.

2.19.7.4. Residue

During single-precision floating-point Add or Add Negative instructions, the bits shifted off the right end of the register during alignment of the mantissas are not included in the addition but saved, and they become the residue. After the addition is performed, the sum and the residue are each packed into floating-point format, the sum is stored, and the residue is stored if enable residue store for Floating-Point Residue Enable designator (DB30) is 1.

When the two 36-bit input operands for an Add or Add Negative instruction are transferred to the arithmetic section, their characteristics are examined, and the mantissa of the input operand with the smaller characteristic is right-shifted a number of bit positions equal to the difference between the characteristics. The bits shifted out of the 36-bit arithmetic register are saved in an auxiliary register. The portion of the mantissa saved in the auxiliary register is used to form the residue and it is not included in the algebraic addition. After completion of the addition and any shifting necessary to normalize the sum, the sum and the residue are packed into single-precision floating-point format and transferred to two consecutive A-registers. If the normalize shifting of the sum is to the right, the least significant bit of the sum is shifted into the most significant bit of the residue mantissa. If the shift is to the left, the residue is not shifted into the sum.

2.19.8. Normalized/Unnormalized Floating-Point Numbers

A floating-point number is normalized when the leftmost bit of the mantissa is not identical to the sign bit or when all bits of the mantissa are identical to the sign bit. A floating-point number is not normalized when all bits of the mantissa are not sign bits and the leftmost bit of the mantissa is identical to the sign bit.

All floating-point operations produce a normalized result when the input operands are normalized. The sums produced by Floating Add and Floating Add Negative instructions and the result produced by the Load and Convert To Floating instruction are always normalized, regardless of whether or not the input operands are normalized. When either or both input operands are not normalized, the result of Add and Add Negative instructions may be less accurate than if normalized input operands had been used.

Normalized input operands must be used for the Floating Multiply, Divide, Compress and Load, and Expand and Load instructions. If normalized input operands are not used for these instructions, the results are undefined.

2.19.9. Floating-Point Characteristic Overflow/Underflow

Floating-point characteristic overflow/underflow occurs when the characteristic does not lie in the range represented in the number of bits allowed for the characteristic.

When any of the Floating-Point Add, Add Negative, Multiply, Divide, or Load and Convert instructions, or the Compress and Load instruction are performed, overflow or underflow may occur.

2.19.9.1. Floating-Point Characteristic Overflow

Single-precision floating-point characteristic overflow occurs when the 8-bit characteristic of the resultant most significant single-precision floating-point word represents a number greater than 377_8 and the associated mantissa is not zero.

Double-precision floating-point characteristic overflow occurs when the 11-bit characteristic of the resultant double-precision floating-point number represents a number greater than 3777_8 and the associated mantissa is not zero.

When overflow is detected, the action taken depends on the Arithmetic Exception Enable designator (DB29). The Characteristic Overflow designator (DB22) is always set.

2.19.9.2. Floating-Point Characteristic Underflow

Single-precision floating-point characteristic underflow occurs when the resultant floating-point word represents a negative number and the associated mantissa of the result is not zero. This means that the exponent of the result is less than -0200; thus, the attached sign (positive - because absolute value is used) changes due to the borrow. If the characteristic of the residue (Floating Add, Floating Add Negative), remainder (Floating Divide), or the least significant single-precision word of the product (Floating Multiply) represents a negative number, this fact by itself does not result in underflow. Instead, the residue, remainder, or least significant word of the product is cleared to all zero bits or set to all one bits (to reflect the appropriate sign).

Double-precision floating-point characteristic underflow occurs when the 11-bit characteristic of the result represents a negative number, i.e., the exponent of the result is less than 2000_8, the mantissa of the result is not zero, and the Double-Precision Underflow Mask designator (DB34) is cleared.

When underflow is detected, the Characteristic Underflow designator (DB21) is always set and the action taken by the IP depends on the state of the Arithmetic Exception Enable designator DB29.

2.19.9.3. Floating-Point Divide Fault

For single- or double-precision floating-point division, a divide fault condition is detected when the mantissa of the divisor is zero. The action taken depends on the Arithmetic Exception Enable designator DB29. The Divide Check designator (DB23) is always set.

2.19.10. Fixed-Point to Floating-Point Conversion

Conversion of a fixed-point number to floating-point number is performed in the arithmetic section. The first input operand contains a characteristic (biased exponent) that defines the location of the binary point for the fixed-point number with respect to the standard position of the binary point for a floating-point number. The second input operand is the signed fixed-point number to be converted.

The conversion process consists of:

- transferring the two operands to the arithmetic section,
- shifting the fixed-point number, if necessary, to position its bits as the mantissa for a normalized floating-point number,
- modifying the characteristic to reflect the magnitude and direction of the normalizing shift,
- packing the shifted fixed-point number (mantissa) and the modified characteristic in floating-point format, and
- loading the packed results in a register (conversion to single-precision floating-point format) or into two consecutive registers (conversion to double-precision floating-point format).

2.19.11. Floating-Point Addition

The process of adding two floating-point numbers consists of:

- loading the numbers into the arithmetic section,
- determining the difference between the characteristics of the two numbers,
- shifting (right) the mantissa of the number having the smaller characteristic,
- adding the mantissas,
- normalizing the result and correcting the characteristic for the shift,
- combining the results in floating-point format, and
- transferring the resulting floating-point numbers to GRS.

The input operands for floating-point addition need not be normalized numbers. For single-precision addition, the sum (most significant word produced) is always a normalized number. The residue word may or may not be a normalized number. For double-precision addition, the sum is always a normalized number.

2.19.12. Double-Precision Floating-Point Addition

The steps performed for double-precision floating-point addition are similar to those for the single-precision addition with these six differences:

1. Each of the two operands occupy two 36-bit registers in the arithmetic section. In single-precision addition, both operands are contained in two 36-bit registers.
2. The mantissa sum can contain a maximum of 60 bits in double-precision addition, instead of 27 bits as in single-precision addition.
3. The bits that are shifted out of the right end of the 36-bit register when the operands are lined up prior to addition are lost. There is no residue.
4. Double-precision characteristic overflow occurs when the characteristic is greater than 3777_8 and the mantissa is not zero.
5. Double-precision underflow occurs when the exponent is less than -2000_8 and the mantissa is not zero. In single-precision the value is -200_8.
6. The sum is stored in two consecutive registers, A_a and A_{a+1}. No residue is stored.

2.19.13. Floating-Point Subtraction (Add Negative)

Floating-point subtraction (both single-precision and double-precision) uses the same steps as for the floating-point add operation (see 2.19.11), except that the subtrahend is complemented before the addition is performed.

2.19.14. Floating-Point Multiplication

The process of multiplying two floating-point numbers consists of loading normalized input operands into the arithmetic section, unpacking, multiplying the mantissas, adding the characteristics, packing the results into floating-point format, and transferring the result to GRS. The results obtained for all cases is that either or both input operands that are not normalized numbers are undefined.

2.19.15. Floating-Point Division

The process of dividing one floating-point number by another consists of loading the normalized input operands into the arithmetic section, unpacking, dividing one mantissa by the other, subtracting the characteristics, packing the results into floating-point format, and transferring the result to GRS. If the operands are not normalized numbers, the results are undefined.

2.19.16. Floating-Point Zero

Floating-point zero can be defined as a floating-point number having all mantissa bits identical to the sign bit. The characteristic sign and all mantissa bits are forced to zeros when this is the result of a floating-point instruction.

2.20. Universal Processor Interface

The Universal Processor Interface (UPI) provides a standard interface for communications between the host system IPs and Channel IOPs. Each IP-to-IP and IP-to-IOP link has a UPI. Each UPI can send and receive requests.

Each UPI is a pair of interfaces between two active components. One interface provides signalling from the first component (sender) to the second component (receiver). The other interface provides signalling in the reverse direction. These two interfaces operate independently.

The UPI interfaces provide no data transmission. The only information passed is a request to communicate. Data that is to be exchanged is placed in main storage at locations agreed upon between the sender and receiver.

An active component only needs to know three things about a UPI interface:

1. When receiving, its priority relative to other UPI interfaces; that is, when two UPI requests are made, which is honored first.

2. Whether a request is directed or broadcast. A directed request always goes to a particular receiver explicitly identified by the sender. A broadcast request goes to whichever potential receiver will accept it first.

   IOPs send broadcast requests and receive directed requests. IPs send directed requests, but can receive either type (that is, for receiving, IPs must know whether the sender is directing or broadcasting interrupts for the receiving interface (see 2.20.3).

3. When receiving, whether the sender is in the same cluster or not.

2.20.1. UPI Number Assignments

The UPI number consists of six bits:

- The leftmost two bits indicate the type of active component:

  0 = SSP or IP
  1 = reserved
  2 = I/O
  3 = I/O

- The rightmost four bits indicate the component number.

- All values, except the IP, consist of S-bus address bits 1 through 3 and 5 through 7 into UPI bits 0 through 6, respectively.

- IP UPI values consists of S-bus address bits 1 and 2 mapped to UPI bits 0 and 1. A value of 0 is mapped to UPI 2 and S-bus address bits 5 through 7 are mapped to UPI bits 3 through 5. IP UPI values are always 0 and 1 even though the corresponding S-bus address bits are 1 through 3, 5 through 7, and 8 and 9.

UPI selector interpretation is:

UPI Number	Active Component
0	IP0
1	IP1
2-6	Reserved
7	SSP
8-39	Reserved
40-63	I/O 0-10,23

2.20.2. UPI Instructions

Four instructions are provided for IP control of UPI operations:

1. Test Previous Send Acknowledge (TPSA)

   TPSA determines whether a specified send interface is busy by testing whether or not a previously sent message (if any) has been acknowledged by the receiver.

2. Send (SEND)

   SEND signals a specified receiver that a message is being sent (that is, has been placed in mutually agreed main storage locations).

3. ACK signals to a specified sender that a message has been received (that is, it is safe for the sender to store a new message in the mutually agreed main storage locations).

4. Clear Previous Send (CPS)

   CPS causes the executing IP to drop its output request for that interface.

2.20.3. External Interrupt Distribution

Two types of interrupts can occur:

1. Directed – a single receiver is specified by the component making the interrupt request.

2. Broadcast – there are potentially multiple receivers. The requester does not specify a receiver; rather, the potential receivers must resolve which one is to accept the request.

All IP external interrupts initiated by I/O are broadcast interrupts (this does not include interrupts that are provoked because of condition detected by the IP when it requests service from I/O). All other interrupts are directed interrupts.

Initial Program Load (IPL) interrupts are broadcast interrupts. The distribution of these interrupts is resolved by load path selection.

The distribution of all other broadcast interrupts is resolved by a Broadcast Interrupt Service flag. The Broadcast Interrupt Service flag operates as follows:

- At any instant in time, this flag belongs to only one IP in an application. Any IP in the application may be the first to have this flag at initialization, but because a halted IP does not retain the flag, the load path IP will get the flag first.

- While the IP has the flag, it must scan at least once for interrupts of all kinds. If there are any interrupts it can currently accept (some may be inhibited by DB13, Deferrable Interrupt Enable designator bit), it accepts the highest priority interrupt.

- The IP that has the flag retains it as long as the IP:

  – is running (not halted),

  – has interrupts enabled (DB13 set), and

  – is idle.

  If any of these conditions cease to be true, the IP must pass the flag promptly.

- Passing the flag means giving it to the next IP.

- An IP may accept a directed IPL interrupt with or without the Broadcast Interrupt Service flag.

2.20.3.1. Broadcast Interrupt Request Sequence

The steps in a broadcast interrupt request reference are:

1. The sender broadcasts a request to all IPs.

2. A wait occurs until an IP that has both the Broadcast Interrupt Service flag and is able to accept the interrupt. The IP then accepts and acknowledges the interrupt request.

3. The sender drops the request.

4. The receiving IP passes the Broadcast Interrupt Service flag.

5. An Acknowledge (ACK) instruction is executed on the receiving IP. This drops the acknowledge on UPI broadcast interrupts. On non-UPI sequences, hardware drops the acknowledge as soon as the sender has dropped the request.

2.20.3.2. Directed Interrupt Request Sequence

The steps in a directed interrupt request sequence are:

1. The sender executes a Send instruction to initiate a request.
2. The receiver detects the request, generates an interrupt, and acknowledges the request.
3. The sender drops the request.
4. An ACK instruction is executed on the receiving IP. This drops the acknowledge for UPI interrupts only.

NOTE: An IP that is halted, but not cleared, can be restarted (interrupted) only by a directed UPI interrupt, and only if the halt is caused by a Halt Jump instruction.

2.21. Interrupts

2.21.1. Interrupt Processing

The interrupt control stack stores the processor state and status when an interrupt occurs. The size of frames on the interrupt control stack is always assumed to be 16 and interrupt processing hardware ignores and does not modify the content of the increment portion of Executive Index Register 1 (EX1). There is no specific architecturally defined method for removing frames from the interrupt control stack. Several methods are available, including SELL, UR (with index register incrementation specified), NOP (again, using index register incrementation) or any of the instructions that modify index registers explicitly (AX, ANX, etc.).

The interrupt control stack pointers are:

- B22 - interrupt control stack limits and base
- Executive Index Register 1 - current stack position address (always treated as 18 bits)

The Executive software establishes and maintains an interrupt control stack for each IP.

For normal interrupts, the hardware performs the equivalent of a stack BUY on the interrupt control stack, with certain variations:

- A true stack overflow condition (resultant stack pointer out of limits) causes the processor to halt, setting a halt code of 4 in bits 15–17 of the PEND field of the activity state packet. EX1 is not updated in this case.
- If the resultant stack pointer is equal to B22 LL//200_8 (that is, pointer is 128 words above its lower limit), then an interrupt control stack overflow warning interrupt condition persists (not deferrable by DB13) until taken (at least waiting for the current interrupt sequence to be completed).

2.21.2. Interrupt Classes and Status

Table 2-8 summarizes the IP interrupt classes and gives their general characteristics. The column headings of Table 2-8 are defined as follows:

Column **Description**

Class — Class number serves three purposes:

- It provides digital identification of the interrupt class.
- During the interrupt sequence, it is used as an index into the interrupt vector starting at Word 0 of the bank described by B16 (normally reflects MSR).
- It defines priority in situations where interrupts for multiple classes can be taken, the lower numbered class having precedence. It must be understood that this priority-driven precedence applies only when conflicting interrupt conditions exist at the same conceptual point in time; for example, if a P breakpoint is set on an instruction that jumps out of limits, a breakpoint interrupt occurs rather than a reference violation interrupt, because the breakpoint condition arose during the jump instruction whereas the reference violation is associated with the next (jump to) instruction.

Sync — Synchrony. Indicates the degree to which the interrupt condition is synchronous with respect to the current instruction sequence, according to the following symbol definitions:

S Fully synchronous, that is, related to the current instruction sequence. Taken at next interrupt point, unless pre-emptied by a higher priority interrupt in which case this interrupt condition is lost (discarded).

A Asynchronous. Not necessarily related to the current instruction sequence. If deferred (by DB13) or pre-emptied, this interrupt condition persists until taken. Asynchronous interrupts are associated with a particular processor.

B Broadcast. Same as asynchronous, except that the interrupt mechanism request is external to the processor and is serviced in accordance with the protocol of the broadcast interrupt flag*. In this case, if the interrupt is pre-emptied or deferred by a processor (having the flag), it still persists until taken, but may be taken by another processor (when it has the flag.) An interrupt class may include both asynchronous and broadcast conditions (for example, IP-to-IP and I/O Processor-to-IP UPI requests); in this case, asynchronous interrupts are taken regardless of the possession of the flag.

* *The determination of which IP accepts the interrupt is based on which IP in the application is currently assigned to servicing such interrupts. This assignment rotates, or is passed, from one IP to the next as each interrupt is fielded.*

P Pended. Related to the current instruction sequence, but differs from fully synchronous in that, depending on the particular condition, pended conditions can be deferred, can be held until a between-instructions interrupt point, and if some other interrupt occurs first, the pended condition is retained in the activity state packet captured on the interrupt control stack and then cleared in the hard-held activity state packet.

Table 2-8. Interrupt Classes

Class	Name	Sync	Defer	Int Pt	Cond Cat	Determ	Source	Association
0	Reserved-Hardware Default	-	-	-	-	-	-	-
1	Unretryable Hardware Check	S	E	M[1]	F[1]	I	I,E	IP/MSU Hardware
3-7	Reserved	-	-	-	-	-	-	-
8	Reference Violation	S	E	M[2]	F[2]	D	I	Storage/GRS Reference
9	Addressing Exception	S	E	I,E	F,NF[3]	D	I	Storage Reference, Instruction
10	Interrupt Stack Overflow Warning	A[4]	E	B[4]	NF[4]	D	I	Interrupt Stack
11	RCS/Generic Stack Under/Overflow	S	E	I,E	F	D	I	Instruction
12	Signal	S	E	B	NF	D	I	Instruction
13	Test & Set	S	E	I,E	F	D	I	Instruction
14	Invalid Instruction	S	E	I,E	F	D	I	Instruction
15	Reserved	-	-	-	-	-	-	-
16	Arithmetic Exception	S	E	I,E	F	D	I	Instruction
17	Character Manipulation Exception	S	E	I,E	F	D	I	Instruction
18	Reserved	-	-	-	-	-	-	-
19	Breakpoint	P	E	B[5]	NF	D	I	Storage Reference by Activity
20	Quantum Timer	P[6]	E[6]	M	NF	D	I	Activity
21	Reserved	-	-	-	-	-	-	-
22	Critical Alert	A,B	E	M	NF	I	I,E	Hardware Environment
23	General Alert	A,B	E	M	NF	I	I,E	Hardware Environment
24	Software Break	P	D	B	NF	D	I	Activity
25	Jump History Full	A	D,I[8]	B	NF	D	D,I[8]	Instruction/Interrupt
26	Delayed Hardware Check	A,B	D	M	NF	I	I,E	IP/MSU Hardware
27	Dayclock	B	D	M	NF	I	E	Dayclock (MSU)
28	Reserved	-	-	-	-	-	-	-
29	IPL	-[7]	-[7]	M[1]	NF[1]	I	E	MSU
30	UPI Initial	-[7]	-[7]	M[1]	NF[1]	I	E	MSU
31	UPI Normal	A,B	D	M	NF	I	I,E	IP/IOP
32-63	Reserved	-	-	-	-	-	-	-

NOTES for Table 2-8:

1. *The nature of these interrupts is such that captured state is not trustworthy and should be used for diagnosis only.*

2. *In the case of a jump (or fall-through) out of limits, captured P is the offending address, not the address of the jump instruction (or preceding instruction). In this case, any previously outstanding interrupt requests take precedence.*

3. *NF applies only for terminal addressing exceptions.*

4. *This class is peculiar in that the condition can only arise during an interrupt sequence. Since it is exigent, it will always be taken before any further instruction execution occurs.*

5. *May arise during interrupt sequence.*

6. *Quantum timer interrupts do not strictly fit the definition of pended synchrony, in that the condition (negative quantum timer) is never cleared by hardware in either the hard-held or interrupt control stack activity state packet. Note that while exigent, quantum timer interrupts are inhibited when DB12 is cleared (which occurs on every interrupt sequence.)*

7. *Since IPL and UPI interrupts are initial interrupts, synchrony and deferability do not apply.*

8. *The value of short status field tells whether state is determinate.*

9. *The Jump History Full interrupt normally occurs only at the between-instructions interrupt point. However, in cases where the Jump History Full interrupt is deferred by DB13 (the indeterminate cases), it may appear that the interrupt has occurred at a mid-instruction interrupt point, that is, the INF bit may be set on the interrupt control stack.*

Defer — Deferability. Defined by the following symbols:

- D Deferrable. Such interrupts are inhibited when DB13 is clear.
- E Exigent. Not affected by DB13.

Int Pt — Interrupt point at which the interrupt can be taken, according to the following symbols:

- B Between instructions only. The INF bit cannot be set on the interrupt control stack for such classes.
- M Mid-execution. Such interrupts can be taken in the middle of instruction execution (in which case the INF bit is set on the interrupt control stack) as well as between instructions.
- I,E Same as M, but restricted to cases involving indirect addressing, the Execute instruction, or the Execute Repeated instruction.

Cond Cat — Condition Category. Defines the action taken upon detection of the interrupt condition, according to the following symbols:

- F Fault. The interrupt condition prevents further instruction execution. The program address register and appropriate GRS locations (X_x, register operands, and the repeat count register, R1) are restored (backed-up) to the values they contained at the previous interrupt point such that the

instruction may be properly restarted at that point. This requires that X_x (if incrementation was specified) must be restored to the value it held prior to instruction execution to properly locate the original operand. If an interrupt condition is detected coincident with an interrupt point, state must reflect progress up to that interrupt point.

NF Non-fault. Instruction execution may continue for a defined and restricted duration. The interrupt is processed at the next interrupt point (or immediately if the condition is coincident with an interrupt point). If this interrupt point is between instructions, processor state reflects the completion of the instruction. If the interrupt point is encountered prior to instruction completion, the program address register captured on the interrupt control stack and appropriate GRS locations as described for the fault case above must reflect progress up to the interrupt point and provide for eventual resumption of the instruction. This includes restoration of X_x if incrementation was specified.

Determ Determinacy. Indicates whether the point of occurrence of the interrupt class is predictable from visible processor and main storage state, assuming a unit processor and no main storage traffic other than from that processor, as follows:

I Indeterminate.
D Determinate.

Source
I Internal to the processor.
E External to the processor.

Association Indicates the system entity or operation with which the interrupt class is associated.

The explicit information associated with a particular interrupt is captured in the I/K register Short Status Field (SSF) field, the I/K register ICF field, and the Interrupt Status Words 0–2 of the activity status packet written to Words 0–7 of the interrupt control stack. Other parts of the activity state packet, especially F0, may serve as additional implicit status to assist software in handling the interrupt. The contents of SSF and the individual ISWs are undefined except as specified for particular interrupts; the same applies to reserved fields within defined ISWs; nevertheless the preferred value is zeros.

Within the following sections, references to the different fields of the activity state packet are made without specifying that they are part of the current interrupt control stack frame.

For consistency of style, where applicable, particular kinds of status are reported as follows:

- Basic status: Short Status Field (SSF).
- Address status: ISW0; includes address generation mode, selected base register, and relative operand address for all fully synchronous interrupts (see Table 2–8). Specific exceptions are enumerated in the discussion of each affected interrupt.

2.21.2.1. Hardware Default - Class 0

This interrupt can't happen. Class 0 is permanently reserved as an ARM feature, acting as a trap for hardware loss of interrupt class.

2.21.2.2. Unretryable Hardware Check - Class 1

This class is comprised of all interrupt-reported hardware errors that alter processor state or main storage data in such a way as to preclude correct resumption of the failing instruction sequence, on this or any other IP in the application.

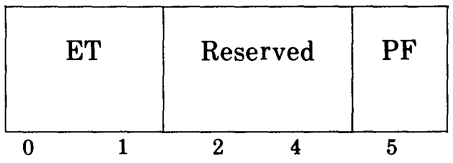
Detailed status indicates the scope of the error. Based on this and other context, software must determine recovery action (may abort user job, do a reboot, etc.)

Occurrence of this interrupt sets DB6 (Fault Handling In Progress).

2.21.2.3. Reference Violation - Class 8

A Reference Violation interrupt is caused by an improper reference to GRS or storage. Status is reported in the SSF and ISW0.

Short Status Field (SSF):



Bits **Meaning**

0–1 Error Type (ET). In case of two errors detected simultaneously, the lower-numbered code will apply. The error type codes are:

0: GRS violation. Attempt to read or write Executive GRS when that type of reference is prohibited by PP.

This error can occur in three circumstances:

1. JGD instruction. The GRS address referenced is the value of the j-field catenated with the a-field. JGD is a jump instruction and therefore cannot otherwise reference GRS.

2. SRS and LRS. The GRS address referenced can be determined from the contents of A_a. The operand of SRS and LRS is always in storage.

3. When GRS is referenced as an instruction operand. The GRS address referenced is in ISW0. This case includes all GRS reference violations caused by instructions other than JGD, SRS, and LRS.

NOTE: A reference violation cannot be caused by GRS references using either the a- or x-fields of an instruction or by the writing of a latent parameter into EXEC R0 as part of gate processing.

1: Storage Limits violation. Attempt to reference a storage location using a relative address that lies outside the storage limits of the B-register(s) used to translate the address.

2: Read Access violation. Attempt to read a storage location, using a relative address, when that type of reference is prohibited by the access permissions (GAF and SAP) in the base register used to translate the address.

3: When both read and write references are prohibited by the access permissions, and an instruction that can both read and write is executed, a read or write access violation is reported depending on whether the storage references are performed with storage lock. Read and write instructions performed with storage lock have both read and write permission validated before the storage lock is set and before any operand reference is made. When both access permissions are denied, the higher number violation (write) is reported. Read and write instructions performed without storage lock are reads followed by writes. When both access types are prohibited, only the first encountered failure (read) is reported, except for the ADD1 and SUB1 instructions, in which case, only the write access violation is reported. For read and write class instructions, when one type of access is prohibited, only that type is reported.

2-4 Reserved

5 P Fetch (PF). When set, indicates that the error occurred attempting to fetch an instruction. This includes P-incrementation (P+1) jumps, indirect references, and references to the target of an EX or EXR instruction. When PF is set, and INF is not set, ISW0 is undefined. When P-fetch is not set or when P-fetch is set along with INF. ISW0 is undefined.

When P-fetch is set, ISW0 is undefined and the relative address for the instruction fetch is in the program address register. When P-fetch is clear, ISW0 is defined as stated in 2.21.2 except that ISW0 is undefined for the LAE and TVAE instructions. However, for GRS violation (ET=0) ISW0 may not point to the failing GRS address. The failing GRS address may have been specified in the a- or ja- field of the instruction (that is, JGD, LRS, SRS), in which case ISW0 would point to the storage operand of the instruction. For storage limits violation (ET=1), when ISW0.BMS=1, ISW0.BREG is undefined. Also when the reference violation was committed by either the ACEL or DCEL instructions, ISW0 ADDR is undefined.

Reference violations may occur in mid-execution; that is, INF and EXRF may be set. Generally, captured activity state reflects successful instruction execution progress, and the processor "backs up" to the last interrupt point. However, INF cannot be set when PF is set; this includes the case of the target address of a jump being the offending reference. In basic mode, this case causes captured DB31 to be undefined.

2.21.2.4. Addressing Exception - Class 9

This interrupt primarily reports exception conditions associated with base register manipulation. It may also be caused by the TVAE, ACEL, DCEL, LBN, and SBU instructions. Status is given by (or inferred from) SSF, F0, ISW0, and ISW1 from the ICS frame.

Short Status Field (SSF):

			Access Conditions		
F	T	R	G	E/V/S	
0	1	2	3	4	5

Value	Meaning
F=1	Fatal (F). This condition can only be the result of hardware or executive malfunction.
F=0	The non-fatal interrupt is the result of a detected exception in the interrupted instruction sequence. This bit can only be set for certain error conditions as defined in the ISW1 description and is otherwise clear.
T=1	Terminal exception condition (T). The exception is reported in the addressing environment established by the normal termination of the instruction in F0 but a detected exception prevents instruction execution in that environment. One or more of the R,G,E,V, and S access conditions are indicated and F is not set.
T=0	The addressing exception prevented execution of the instruction. The interpretation of the Error Condition and Qualifier fields of ISW1 are dependent on the access condition status in SSF.

If any of the Access Conditions are indicated (either of R or G set or E/V/S not zero), the error condition field of ISW1 is meaningless and the F-bit in SSF is always clear. If no access conditions are indicated, the error condition field of ISW1 is meaningful and the F-bit is set or clear as appropriate. In either case, the QUAL field qualifies the condition.

R=1	Residency fault (R). The Residency interrupt bit (R) was found set in a BD. The T-field of and QUAL field of ISW1 indicate whether this is a target, indirect, or gate BD. A gate BD is treated as a target BD by LBU, LBUI, LBE, and LAE.
R=0	No residency fault encountered.
G=1	General fault (G). The General interrupt bit (G) was found in a BD and is handled as described for an R-bit.
G=0	No general fault encountered.

Enter, Validated entry point, and base register Selection access conditions encoding (E/V/S).

0 None of the above detected.

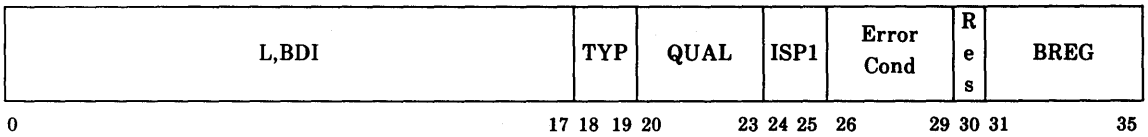
1 Enter access denied.

2 Validated entry point violation. This condition is always terminal.

3 Selection-of-base-register violation. This condition is always terminal and is detected prior to entry point validation.

ISW0 is as defined in 2.13.6 and 2.21.2, with the following exceptions: For the RTN instruction, ADDR=(EX0M) and BREG=21; for SBU, ISW0.ADDR = the instruction a-field and ISW0.BREG=20; for TVA and TVAE, ISW0 is undefined; BMA and BREG are undefined for the LIJ, LDJ, LBJ, and LOCL instructions. For class 9 interrupts, ISW0 may not provide definitive information, as it usually points to source L,BDI (bits 0–17) and does not serve to pinpoint the actual cause of the interrupt. More specific information is provided by ISW1, as follows:

ISW1



L,BDI The L,BDI (bits 0–17) of either the bank in error, the indirect BD containing the error, or the gate bank containing the gate in error. The Error Condition and QUAL fields together indicate the meaning of this field.

TYP Type (bits 18,19). The type field of the BD fetched using the L,BDI field, that is, a copy of bits 28–29 of BD WD0, and not the interpretation of the bank descriptor type loaded in the base registers. This field is undefined for the following errors:

- L,BDI <32;
- boundary errors on the Return Control Stack and Bank Descriptor Table structures;
- BDT limit errors where L,BDI is from the source or from the RCS;
- ABT limits violations;
- LBJ interface specification errors; or
- illegal BD type for the BD pointed to by a gate.

QUAL Qualifier (bits 20–23). This field provides qualification of both the access conditions in SSF and the error conditions of ISW1 by indicating the phase of the base register algorithm where the condition was detected. The QUAL field is undefined when the Error Condition field contains a value of 4. QUAL is also undefined for Addressing Exception interrupts resulting from operand boundary errors encountered by the ACEL and DCEL instructions.

Value	Meaning	L,BDI
0	Target BD	Target
1	Indirect BD	Source
2	Gate BD	Source or from indirect BD
3	Gate	Source or from indirect BD
4	Active Base Table	Undefined
5	Reserved	-
6	Source BD	Source
7	Return Control Stack	Undefined
8	User Return	Target

When the qualifier indicates user return, the activity state packet captured on the interrupt control stack describes the environment that was the operand of the UR instruction, not the environment in the UR instruction that was executed. Therefore, F0 does not necessarily contain a UR instruction.

ISP1 Interface Specification Plus One (bits 24,25). This field is zero except when an LIJ, LDJ, or LBJ instruction caused the interrupt, in which case this field contains a value that is one greater than the IS field found in X_a.

This field can be used to determine which extended mode operation was executed by an LBJ-class instruction on a mixed mode transfer.

Value	Condition
0	Not LBJ-class or invalid IS (never terminal); therefore, no LBJ-class mixed mode transfer occurred.
1	CALL operation if DB16=0 in the captured activity state packet and this is a terminal (SSF.T=1) exception.
2	GOTO operation if DB16=0 in the captured activity state packet and this is a terminal exception.
3	RTN operation.

Error Cond Error Condition (bits 26-29). This field is meaningful only if T and the Access Condition fields are both zero. The conditions may or may not be fatal (see F-bit in the SSF) as indicated. If multiple conditions are detected simultaneously, the condition with the least status value is reported.

Value	Meaning	Fatal/Non-Fatal
0	Inappropriate BD type	Always fatal.
1	Invalid BDI value	Fatal only when a return control stack frame, the program address register of an activity state packet, indirect BD, gate, or active base table entry contains L,BDI<32.
2	Invalid IS value	Never fatal.
3	Invalid fast RTN	Never fatal.

4	Invalid GOTO operation (GI-bit set)	Never fatal.
5	Gate error conditions with MP-bit set	The value 5 reports error conditions that are specific to a gate that has the MP-bit set. The error conditions are: ■ The gate was not invoked with a call instruction. This condition is never fatal (F-bit in SSF is not set). ■ Word 0 of the gate contains zero or specifies a macro-procedure that is not defined or supported. This condition is always fatal (F-bit in the SSF is set).
6-7	Reserved	
8	Limit error (gate, BDT, or active base table)	Fatal when a return control stack frame, the program address register of an activity state packet, indirect BD, gate, or active base table entry contains a BDI that is out of limits for the BDT selected by L or when the active base table is not long enough to have an entry for the base register being modified or stored (SBU).
9	Absolute boundary error	Fatal only when the return control stack (RTN, CALL and LOCL, operations), any BD, or a gate does not lie on the boundary required.
10	Upper limit boundary error	Never fatal.
11-15	Reserved	

BREG Base register (bits 31-35). This is the number of the register being manipulated when the condition was detected. For all instructions except interbank transfers between execution modes and LAE, there is only one base register operated on during the entire instruction. In the mixed-mode transfer case, the base register is set as follows: for non-terminal errors encountered by LBJ with IS=2 (RTN operation), the base register is undefined; for all other non-terminal errors except the active base table limits violations, the base register is one of B12 through B15; for all terminal errors base register corresponds to the destination mode. The base register is undefined for all non-terminal errors encountered by LBJ, IS=2 operations (that is, for those with basic mode target as well as those with extended mode target). In the case of LAE, the base register contains a number from 1-15 indicating the register being loaded when the condition was detected. The base register content is undefined when the interrupt is caused by any of the instructions LBN, SBU, TVA, or TVAE. It is also undefined for operand boundary errors.

2.21.2.5. Interrupt Stack Overflow Warning - Class 10

The Interrupt Stack Overflow Warning interrupt is generated if the creation of an interrupt stack frame for another interrupt causes the following conditions: bits 18-26 of EX1 (GRS 0141) are equal to the lower limit value of B22, and bits 27-35 of EX1 have a value of 200_8, that is, eight more stack frames are available.

The value of the SSF is zero. SSF values 1-63 are reserved, and no ISWs are defined.

It is recommended that the S-bit of B22 be clear.

2.21.2.6. Return Control Stack/Generic Stack Underflow/Overflow - Class 11

This interruption is caused during the following instructions if a stack overflow or underflow condition is detected during:

1. The execution of a SELL or BUY instruction.
2. The execution of a LIJ, LBJ, or LDJ instruction that causes the creation or deletion of a return control stack frame.
3. The execution of a CALL or LOCL instruction.
4. The execution of a RTN instruction.

The contents of the SSF is defined as follows:

Value	Meaning
0	Generic stack overflow
1	Generic stack underflow
2	Return control stack overflow
3	Return control stack underflow
4-63	Reserved

The ISW0 contents are defined as follows:

SSF Value	Meaning
0	ADDR=X_m-X_i-d; BMA and BREG defined normally.
1	ADDR=X_m; BMA and BREG defined normally.
2,3	ISW0 is undefined. This is the return control stack with the failing pointers implicitly defined as EX0 and B21.

Both ISW1 and ISW2 are undefined. The instruction causing the interrupt may be safely retried.

2.21.2.7. Signal - Class 12

This interruption occurs as the result of the execution of an ER or SGNL instruction.

The SSF is interpreted as follows:

Value	Meaning
0	ER instruction caused the interrupt
1	SGNL instruction caused the interrupt
2-63	Reserved

The operand address (U) of the instruction is stored into bits 12-35 of ISW0 and the BMA and BREG fields of ISW0 and both ISW1 and ISW2 are undefined.

2.21.2.8. Test and Set - Class 13

This interrupt condition arises during the execution of a TS instruction if bit 5 of the operand is set.

The value of SSF is defined as follows:

Value	Meaning
0	Relative addressing
1	Absolute addressing
2-63	Reserved

ISW0 is defined normally.

2.21.2.9. Invalid Instruction - Class 14

This interrupt occurs in the following circumstances, identified by SSF values:

Value	Meaning
0	An attempt was made to execute either an instruction whose opcode is not defined for current execution mode (DB16), or an LBJ, LIJ, or LDJ instruction selects X0 as the linkage register (a-field is zero), or an LBU or LBUI instruction selects either B0 or B1 (a-field is zero or one), or an LBU0 or LBU1 instruction selects B0 (a-field is zero).
1	An attempt was made to execute an instruction with a defined operation code but whose associated Processor Privilege (PP) requirement is lower-valued than current PP (DB14-15.)
2	An instruction that permits the j-field to specify a partial word operand was encountered in extended mode which specified an immediate operand in conjunction with 24-bit indexing, that is, when j specifies partial word operations and $j=16_8$ or 17_8, and DB16=0 and i=1, and X>0 and PP>1.
3	An Execute Repeated instruction (EXR) attempted to execute an invalid target instruction or a UR instruction was executed with a target activity state packet that had INF and EXRF fields set in the indicator/key register, and DB16 set in the designator register.

4 The operand of a User Return instruction specified resumption of a macro-procedure (IWF and MPF are both set) and the MPID field is zero or specified a macro-procedure that is not defined or supported.

5 The operand of a User Return instruction specified resumption of a macro-procedure and the SUB-IND field was non-zero.

6-63 Reserved

The detection and reporting of these conditions are prioritized, with higher priority conditions having lower SSF values. For example, a normally undefined operation code that is invoked by an EXR is reported as an undefined operation (SSF=0), not an illegal EXR operand (SSF 2), which it also is.

When the SSF value is 0, 1, or 3, there are four cases for which ISW0 is undefined. These occur when any of the operation codes for the extended mode instructions BUY, SELL, SBU, or RTN are encountered in basic mode execution or as target of an EXR instruction. When the SSF value is 2 or 3, ISW0 is undefined; otherwise, ISW0 is as specified in 2.21.2.

NOTES: 1. Instruction i-field indirection is never resolved for invalid instructions, and no user state is altered.

2. The primary intent of the invalid instruction mechanism is for protection and debugging. However, it is recognized that a standard software engineering technique is to use invalid operation codes to achieve repertoire extensions or retrofits, and the architecture caters to this. Nevertheless, future systems may unavoidably define such operation codes, thus causing object-level incompatibility. It is therefore recommended that such usage be done in a manner that facilitates conversion (for example, by using symbolic names for the operation codes).

2.21.2.10. Arithmetic Exception - Class 16

If the Arithmetic Exception designator bit (DB29) is set, an Arithmetic Fault interruption occurs in the following cases:

1. If the exponent of a floating-point result is greater than +127 (Single Precision) or +1023 (Double Precision), it is Characteristic Overflow.

2. If the exponent of a floating-point result is less than -128 (Single Precision) or -1024 (Double Precision), it is Characteristic Underflow. For Double Precision, DB34 must be clear also.

3. If the quotient of a divide function (DI, DF, DSF) may not be represented in a single register, it is Divide Check. A divisor with value 0 will always cause a Divide Check to occur.

The appropriate Designator bit indicating that the fault occurred is set independently.

In extended mode, if the instruction that caused the arithmetic exception was the operand of a EXR instruction, as indicated by the value of the EXRF bit in the activity status packet, the contents of R1 must always be restored to its proper value by adding 1. This is necessary because R1 is decremented in preparation for the next execution of the operand instruction prior to the detection and processing of the exception generated during the current execution cycle.

The value of the SSF is defined as follows:

Value	Meaning
0	Characteristic Overflow
1	Characteristic Underflow
2	Divide Check
3–63	Reserved

No ISWs are defined.

2.21.2.11. Character Manipulation Exception - Class 17

This interrupt condition arises during the execution of certain instructions. Many different conditions can cause the generation of this class of interruption as identified by the following values in the SSF:

Value	Meaning
0	An illegal control character was encountered during the execution of BMTC or EDDE.
1	More than 256 storage references were attempted during the execution of the EDDE.
2	Applies to BMTC only. The remaining bit count in R1 is not a multiple of the number of bits in a destination string character. When this occurs, the instruction results are undefined.
3–63	Reserved.

ISWs are undefined.

2.21.2.12. Breakpoint - Class 19

This interrupt condition exists if the H-bit of the Breakpoint register is clear, and one of the following conditions is met:

1. The absolute address contained in the Breakpoint register is equal to the absolute address of the current instruction, and the breakpoint P-bit is set.

2. The absolute address contained in the Breakpoint register is equal to an absolute operand address and either the operation is a read and the breakpoint R-bit is set, or the operation is a write and the break point W-bit is set.

The instruction causing the address match to occur is completed before the interruption is taken.

On a mid-execution interruption, a breakpoint condition is recorded in the PEND field.

The value of the short status field is zero. SSF values 1–63 are reserved, and ISWs are undefined.

2.21.2.13. Quantum Timer - Class 20

If the contents of the Quantum Timer value are negative, the Quantum Timer interrupt condition exists.

The current value of the Quantum Timer is stored as part of the activity state packet on any interruption.

The SSF value is zero. SSF values 1-63 are reserved, and ISWs are undefined.

2.21.2.14. Critical Alert - Class 22

This class is used to report abnormal hardware conditions, unrelated to the interrupted instruction sequence, which presage an imminent, or very likely, major loss of state, data or capability. Such conditions generally relate to environmental factors such as power, cooling, etc.

The interrupted instruction sequence is always resumable; however, software must decide whether to continue operation in light of the critical circumstances.

2.21.2.15. General Alert - Class 23

This class is similar to Critical Alert, but is used to inform the software about abnormal hardware conditions, unrelated to the interrupted instruction sequence, which do not represent an immediate hazard to continued operation. It includes notification of cessation of abnormal conditions (for example, power restoration.)

The interrupted instruction sequence is always resumable, and software will presumably continue operation after taking appropriate logging and advisory action.

2.21.2.16. Software Break - Class 24

The Software Break interruption indicates that the Soft Break bit of the PEND field of the Indicator/Key register was set. This bit is encountered in an activity state packet during a UR instruction, and causes an interruption before the first instruction is executed, if the MID INF bit is clear, or after the current instruction is completed, if the MID INF bit was set.

The SSF value is zero. SSF values 1-63 are reserved, and no ISWs are defined.

2.21.2.17. Jump History Full - Class 25

If the S-bit of the Breakpoint register is set, this interrupt condition will exist when the jump history becomes full.

Detection of the JH full condition will disable any further stacking, and that the JH pointer will point to the first JH location after the last word has been stored independently of the S-bit value.

The SSF has meaning as follows:

Value	Meaning
0	Normal Jump History Full interrupt. The point of the interrupt occurrence is determinate (see the meaning of determ for Table 2-8). Note that the entry created by the jump history full interrupt is always lost, but since this entry can be reconstructed from program address register, etc., captured on the interrupt control stack, no significant information is lost.
1	The Jump History Full interrupt was deferred due to DB3=0, one or more entries may have been lost, and the point of the interrupt occurrence is indeterminate.
2-63	Reserved.

ISWs are undefined.

2.21.2.18. Delayed Hardware Check - Class 26

This class is used to report all instances of processor and main storage hardware failures that have been successfully overcome by hardware in one of the following ways:

- Successful retry by hardware.
- Successful data reconstruction by hardware (for example, using error correction code).
- Successful retry initiated by software.
- Graceful degradation by hardware (presumably in conjunction with one of the above).

Detailed status indicates the nature of the error, affected components, etc.

2.21.2.19. Dayclock - Class 27

This interrupt is normally caused by a Dayclock's Comparator being less than or equal to its clock value.

The SSF value is the number of the MSU that contains the Dayclock causing the interrupt. ISWs are undefined.

2.21.2.20. Initial Program Load - Class 29

The IPL interrupt is a bootstrap interrupt broadcast by an MSU.

The SSF contains the MSU number. ISWs are undefined.

2.21.2.21. UPI Initial - Class 30

This interrupt occurs when a directed, as opposed to broadcast, Universal Processor Interface (UPI) interrupt request is made to a IP that is in the cleared state. It is intended for activating non-load-path IPs during system IPL (bootstrap), and for activating a IP newly added (partitioned) to the application.

The SSF contains the UPI number of the sending IP. ISWs are undefined.

2.21.2.22. UPI Normal - Class 31

This interrupt occurs when a UPI interrupt request is honored by an IP that is not in the cleared state.

The SSF contains the UPI number of the sender. ISWs are undefined.

3. Channel Input/Output Processors

3.1. General

The Channel Input/Output Processors (Channel IOPs) consist of three separate units that reduce computer system hardware requirements for peripheral subsystems. These Channel IOPs are:

- Integrated Disk Controller Channel (DCC)
- Integrated Byte Bus Channel (BBC)
- Integrated Block Multiplexer Channel (BMC)

Each of these is a requester on the System bus (S-bus). Channel IOP control and data exchange with the Instruction Processor (IP) is accomplished through main storage.

Each DCC performs a combination of system channel and device control unit functions for two strings of disk drives. The BBC performs a channel function for all character/byte oriented input/output (I/O) devices using a lower speed I/O that is amenable to data path sharing. Complementing the BBC is a series of integrated microprogrammed peripheral control units and communications line modules. These interface to the channel via the BBC I/O bus that is electrically and logically equivalent to the Distributed Communications Processor (DCP)/20, and DCP/40 L-Bus. The BMC is used for nonstandard configurations that require a compatible block multiplexer mode interface.

3.1.1. Error Detection

System error detection capabilities are:

- All adders (data path and control) are either duplicated and compared or protected by parity predict.
- User data paths are parity through-checked where possible. The exception to this is unique format data packing and unpacking which is protected by parity accumulate logic.
- All Random Access Memory (RAM) is parity checked.
- All RAM addressing paths are parity checked.

3.1.2. Interface

The Channel IOPs provide interfaces for:

- Up to 16 disk drives may be attached per DCC.
- Up to 16 logical couplings are provided for low-speed peripherals and communications lines that may be coupled to the BBC through the integrated controllers is dependent upon the configuration, mix of peripheral types, communications line speeds, and type of communications line module.
- Up to eight control units may be attached to each BMC.

All communications between the IP and the I/O hardware is accomplished through the Universal Processor Interface (UPI). Control information is passed between the software and the I/O units through control tables located in the Main Storage Unit (MSU). The I/O hardware uses Channel Address Words (CAW), I/O order codes, subchannel snapshot format, Channel Command Words (CCW), and I/O status reporting mechanisms.

3.2. Input/Output Operations

These operations include subchannel addressing during instruction initiation. At this time, an index is specified that is related to the physical location of a device. Also included in I/O operations is Channel IOP control information entered and accessed in a control table, which is located in main storage. A typical input/output operation results in a basic series of events.

3.2.1. Subchannel Addressing

The 12-bit subchannel address, issued by the software during instruction initiation, specifies an index into a channel-descriptor table. Since the index is oriented on a subchannel basis, the Channel IOP interface to the MSU is effectively established. The channel-descriptor table also contains information relating to the physical location of the addressed device. The content of the channel-descriptor table follows.

The relative position of the logical fields may change; however, their bit content ( ) remains as shown.

(5)	(1)	(3)	(6)	(1)	(1)	(12)	(1)
Subchannel Mode	Subchannel Enable	Channel Type	Channel Number	Status Table Select	Device Path Select	DPSTO	TCUB Enable

No bit positions are entered in the table since these designations are unique for each host system application.

Subchannel Mode — Specifies the mode of operation of the subchannel. The different modes with their appropriate bit positions are defined in Table 3-1.

The subchannel field contains five bits. Bits 1 through 4 determine the subchannel mode for the DCC, BBC, and BMC. Bit 0 is used to create modes unique from the subchannel mode for microprogram use.

Table 3-1. Subchannel Mode Bit Designation

Bit Position*					
0	1	2	3	4	Definition
0	0	0	0	0	Idle
0	0	0	0	1	Terminate Out
0	0	0	1	0	Command Retry
0	0	0	1	1	Active Command Chain
0	0	1	0	0	Status Pending
0	0	1	0	1	Start I/O Fast release (SIOF) Pending**
0	0	1	1	1	Active Out
0	0	1	1	1	Active In
0	1	0	0	0	External Interrupt (EI) Pending/Unit Check Device End Pending
0	1	0	0	1	Terminate In
0	1	0	1	0	Reserved
0	1	0	1	1	Reserved
0	1	1	0	0	Reserved
0	1	1	0	1	Control Unit Busy (CUB)
0	1	1	1	0	Mode External Function (EF)
0	1	1	1	1	Reserved for Device Path Selection Operations
1	0	0	0	0	Channel Status Pending (BMC)
1	0	0	0	1	Auto Sense Pending (BMC)
1	0	0	1	0	Auto Sense Status Pending (BMC)
1	0	0	1	1	Reserved
1	1	0	1	1	Reserved
1	1	1	0	0	Active in Forward Special (DCC)
1	1	1	0	1	Reserved
1	1	1	1	1	Reserved

* *Binary number is related to octal value.*

** *Pending indicates that status can not be written in storage.*

Subchannel Enable — Specifies that the subchannel is operational (online) if set; the subchannel is disabled (offline) if clear.

Channel Type — Specifies the type of channel associated with this subchannel. The following channels are available, shown with their appropriate digital code.

00 - Unassigned or non-existent

01 - 1100 Block Multiplexer

02 - 1100 Word

Channel Number — Specifies the logical number, 0, of the channel.

Status Table Select — Specifies that all status for this subchannel be reported by means of status tabling. The presence of a logical 0 causes all status to be reported using the Universal Processor Interface (UPI) discipline.

Device Path Select	Enables the Device Path Selection feature if set (logical 1).
Device Path Selection Table Offset (DPSTO)	Specifies an offset into the Device Path Selection table when the device path select bit is set (logical 1).
Transparent Control Unit Busy (TCUB)	Enables the optional Transparent Control Unit Busy feature if set (logical 1).

3.2.2. Control Information

Communications between the software and the Channel IOP is accomplished using the UPI logical protocol. Control information is entered and accessed in a control table, which is located in main storage. The Channel IOP has a 32-word control table for Channel Address Words (CAWs), Channel Status Words (CSWs), Channel Fault Words (CFWs), and Unit Check CSWs. A Sense Information buffer of eight words is appended to the address, status, and fault words. This buffer is capable of expansion to bring the total length of the control table to 436_8 words. See Table 3-2 for a configuration of the Channel IOP Control Table.

The Sense Information buffer is used in automatic sense information retrieval. Peripheral fault data is stored as a result of device status reporting. The termination of an auto sense operation as the result of an error results in the channel IOP issuing a selective reset to the device. This ensures that the control unit is not left in the contingent connective state.

The control table starts on any 16-word absolute address boundary. When an Channel IOP is master cleared, the control table basic address is set to Module Select Register location MSR+120_8. The MSR can be changed, since it is a memory location.

The Channel IOP provides UPI interfaces for each IP in the system, as well as for the SSP. Also, the software can determine the control table location from the UPI number.

3.2.3. Sequence of Events

The basic sequence of events that occur during the initiation of an order code includes the following:

1. The IP writes the CAW into the I/O control table in main storage.
2. The IP executes a SEND command.
3. The Channel IOP interprets the order code contained in the CAW.
4. The Channel IOP executes the particular order and writes a condition code in the CAW in main storage.
5. The IOP then transmits a UPI Acknowledge (ACK) command to the IP.
6. If the order code is a Start I/O Fast Release (SIOF), then:
   a. The Channel Command Word (CCW) list (specified by the CAW) is transferred and executed by the IP.
   b. Data is transferred (if necessary) between the peripheral device and main storage.

Table 3-2. Channel IOP Control Table Configuration

Address	Contents
000	Channel Access Word 0 for UPI 0
001	Channel Access Word 1 for UPI 0
002	Channel Access Word 0 for UPI 1
003	Channel Access Word 1 for UPI 1
004	Channel Access Word 0 for UPI 2
005	Channel Access Word 1 for UPI 2
006	Channel Access Word 0 for UPI 3
007	Channel Access Word 1 for UPI 3
010	Reserved
011	Reserved
012	Reserved
013	Reserved
014	Channel Address Word 0 for SSP 1
015	Channel Address Word 1 for SSP 1
016	Channel Address Word 0 for SSP 0
017	Channel Address Word 1 for SSP 0
020	Channel Status Word 0
021	Channel Status Word 1
022	Channel Status Word 2
023	Channel Status Word 3
024	Channel Fault Word 0
025	Channel Fault Word 1
026	Channel Fault Word 2
027	Channel Fault Word 3
030	Unit Check Channel Status Word 0
031	Unit Check Channel Status Word 1
032	Unit Check Channel Status Word 2
033	Unit Check Channel Status Word 3
034	Unit Check Channel Fault Word 0
035	Unit Check Channel Fault Word 1
036	Unit Check Channel Fault Word 2
037	Unit Check Channel Fault Word 3
040	First Word of Sense Information Buffer
041	Second Word of Sense Information Buffer
042	Third Word of Sense Information Buffer
043	Fourth Word of Sense Information Buffer
044	Fifth Word of Sense Information Buffer
045	Sixth Word of Sense Information Buffer
046	Seventh Word of Sense Information Buffer
047	Eighth Word of Sense Information Buffer
.	.
.	.
.	.
0436	Last Word of a Maximum Sense Information Buffer

c. A CSW is written either in a status table or in the Channel IOP control table.

d. If specified, a UPI interrupt is sent to the IP when the Channel IOP transmits a UPI Send command to the IP.

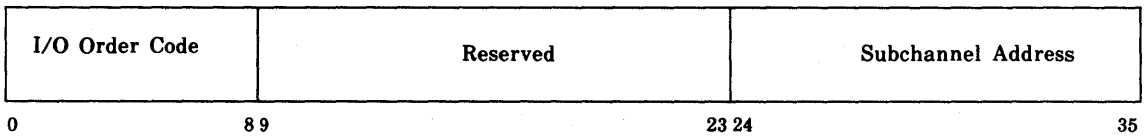
e. The IP sends an acknowledge to the Channel IOP by transmitting a UPI Acknowledge command.

3.2.4. Order Code Initiation

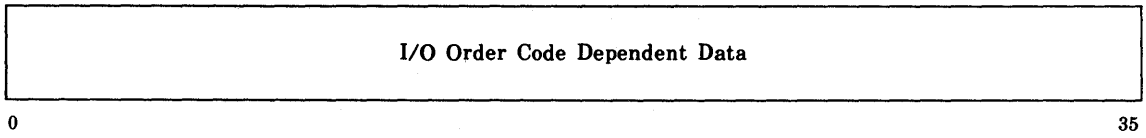
An order code is a command to the Channel IOP to do something. This control information is transferred between the software and I/O hardware through control tables located in main storage. UPI routines are converted to hardware addresses by microcode in the IP.

The general format of a Channel Address Word (CAW) is:

CAW0



CAW1



The Channel IOP always returns a condition code in CAW1 indicating the success or failure of the relevant I/O operation. Although the fields are order code dependent, the first five bits are common. Bits 0–2 contain the condition code; bit 3 is set (logical 1) if the order code is illegal or not implemented; bit 4 is set if a hardware fault is subchannel related and does not involve the entire Channel IOP.

3.2.5. I/O Order Codes

The Channel IOPs execute 19 order codes to initiate, control, and halt I/O operations. These codes are listed in Table 3-3, together with the related mnemonic and octal value.

Table 3-3. Channel IOP Order Codes

Order Code	Mnemonic	Value (Octal)
Start I/O Fast Release	SIOF	001
Test Subchannel	TSC	002
Halt Subchannel	HSC	003
Clear Subchannel	CSC	004
Clear Channel	CCH	005
Load Interrupt Mask Register	LIMR	006
Load Control Table Address	LCTA	007
Activate Status Table	AST	010
Stop Status Table	SST	011
Update Status Table	UST	012
Read Fault Log	RFL	013
Load Device Path Selection Base Register	LDPS	014
Select Device Path Selection	SDPS	015
Write Channel Descriptor Table	WCDT	016
Inject MSU Fault	IMF	017
Inject I/O Internal Fault	IIF	020
Select Status Tabling	SLST	021
Enable/Disable Subchannel	EDSC	022
Read Channel Descriptor Table	RCDT	023

The Channel IOPs interpret the following I/O order codes as illegal:

Order Code	**Value (Octal)**
Undefined	0,024 - 777

3.2.5.1. Subchannel Status

The Channel IOP's execution of an I/O order code is determined by the state of the specified subchannel. A subchannel has four states:

- Idle - A subchannel is placed in the idle state by:
  - a master clear operation,
  - an interrupt or status tabling sequence that relieves the subchannel of status, or
  - a successful halt subchannel, clear subchannel, or clear channel order code.
- Busy - A subchannel is placed in the busy state by:
  - a SIOF instruction when in the idle state.
- Status Pending - A subchannel is placed in the status pending state by:
  - the detection of a hardware fault,
  - the detection of a software fault, or

- the presentation of status by a control unit or device.

■ Disabled - A subchannel is placed in the disabled state by:

- the execution of an EDSC order code with the EN bit clear.

3.2.5.2. Order Code Response

The IOP interprets and responds to all order codes by writing a three-bit condition code into the CAW1, bits 0-2, before acknowledging the UPI request. The general format of the condition code is:

Condition Code	Explanation
0	The order code was successfully executed.
1	The order code encountered a hardware fault.
2	The fault was detected while attempting to read the CAWs.
3	The addressed subchannel was not available.
4	The addressed subchannel was busy.
5	The addressed subchannel contained pending status.
6,7	Order code dependent.

The return of a non-zero condition code leaves the subchannel in a state as if the I/O order code had never been executed. If the Channel IOP recognizes an illegal order code in CAW0, it responds by setting the illegal order code indicator (bit 03) in CAW1. When this bit is set (logical 1), the condition code field is set to 7. Illegal order code include codes not implemented.

3.2.6. Order Code Operations

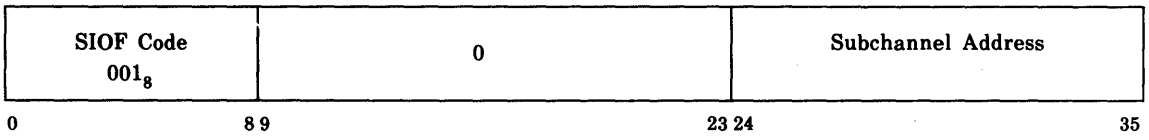
The operations performed by the order codes are described in the subsequent paragraphs.

3.2.6.1. Start I/O Fast Release start I/O fast release

The Start I/O Fast Release (SIOF) order code initiates the execution of a subchannel program on the subchannel specified by entering appropriate data in designated fields of CAW0 and CAW1.

The CAW format, as well as a description of each field, for the SIOF order code is:

CAW0



CAW1

where, for CAW0:

Bits 0–8 = 001_8	Octal value of SIOF order code.
Bits 9–23 = Zero	The value of this field is set at all zeros.
Bits 24–35 = Subchannel Address	Specifies the address of device–related subchannel.

where, for CAW1:

Bit 0 = Priority (PRI)	Specifies that after a status table entry is made for this operation, the IOP is to interrupt the IP. The address in the table of the entry is reported in the CSW.
Bits 12–35 = MSU Data Address of First CCW	Specifies the location of the CCW in main storage, where it was written by the IP.

A three–bit condition code, returned to software in bits 0–2 of CAW1, indicates the response to the SIOF order code. Bit 3 of CAW1 is cleared (logical 0).

The general format of the condition codes is:

Condition Code	Explanation
0	The order code was successfully executed.
1	The order code encountered a hardware fault.
2	A fault was detected while attempting to read the CAWs.
3	The addressed subchannel was not available.
4	The addressed subchannel was busy.

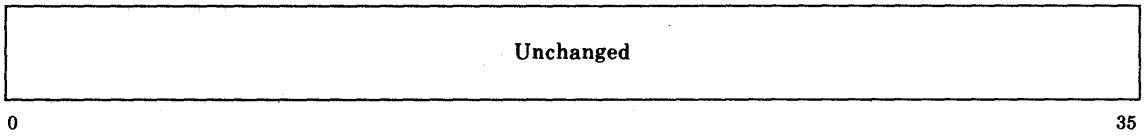
5 The addressed subchannel contained pending status.

6 Reserved.

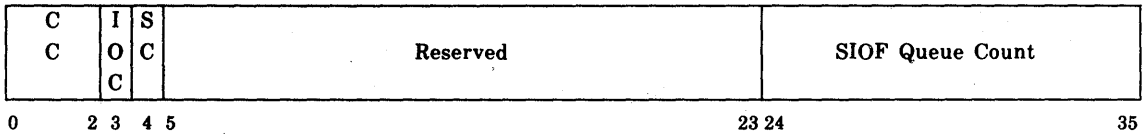
7 The CCW address specified was not on a double word boundary.

The CAW format by which the IOP presents a condition code is:

CAW0



CAW1



where, for CAW1:

Bits 0–2 = Condition Code (CC) The IOP interprets and responds to all order codes by writing a three-bit condition code before acknowledging the UPI request.

NOTE: The return of a non-zero condition code leaves the subchannel in a state as if the I/O order code had never been executed. The SIOF order code initiates a state change on the subchannel only if a clear (logical 0) condition code is returned. This may not be true in the case of a hardware fault.

Bit 3 = Illegal Order Code (IOC) If the IOP recognizes an illegal order code in CAW0, it responds by setting the Illegal Order Code indicator bit to 1. When this bit is set, the condition code field is specified 7, as a "don't care" selection.

NOTE: Illegal order codes include order codes not implemented.

Bit 4 = Subchannel (SC) Fault If this bit is set (logical 1) and the Condition Code is 1, hardware fault, then the fault is subchannel related and does not involve the entire IOP.

Bits 5–23 = Reserved Reserved for software

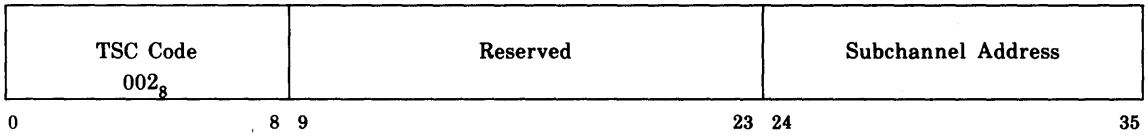
Bits 24-35 = SIOF Queue Count Specifies the number of outstanding SIOFs for the channel associated with the addressed subchannel. The current command is included in the count. This field is only valid if a condition code of zero is returned.

3.2.6.2. Test Subchannel

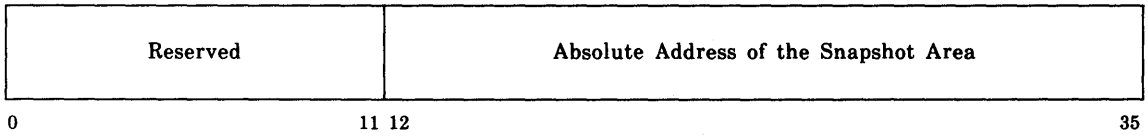
The Test Subchannel (TSC) order code stores a snapshot of the addressed subchannel's control words at the absolute address specified by bits 12-35 of CAW1.

The format of the CAW for the TSC order code is:

CAW0

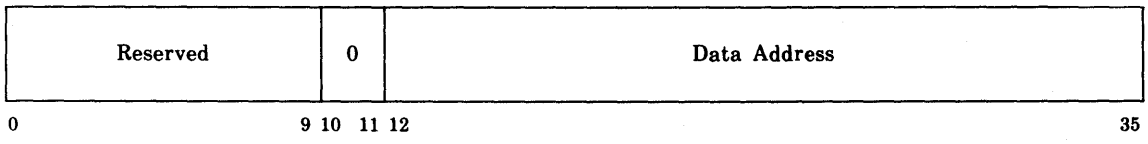


CAW1

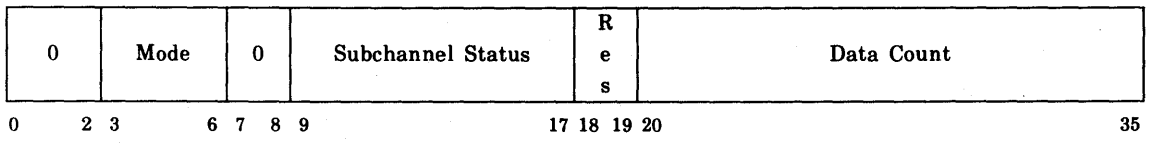


The format of the first four snapshot words used by the BMC is:

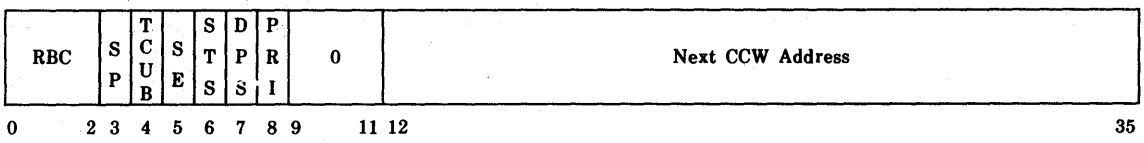
Snapshot Word 0



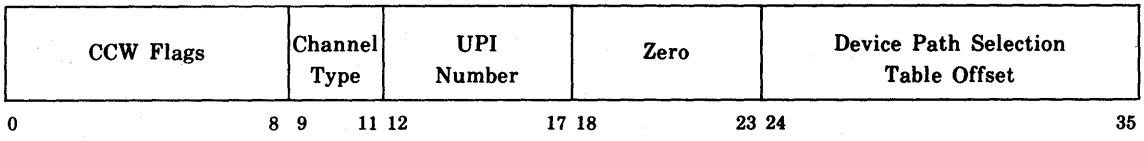
Snapshot Word 1



Snapshot Word 2

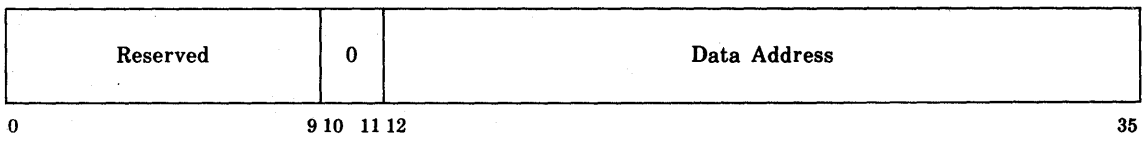


Snapshot Word 3

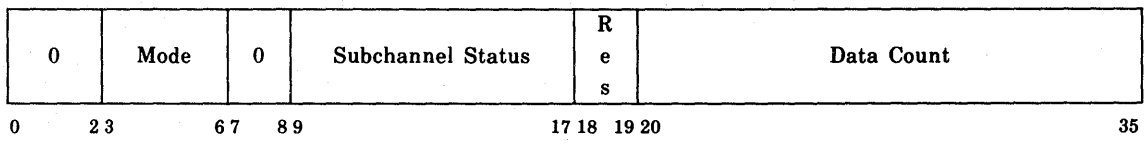


The format of the Snapshot words as provided by the BBC is:

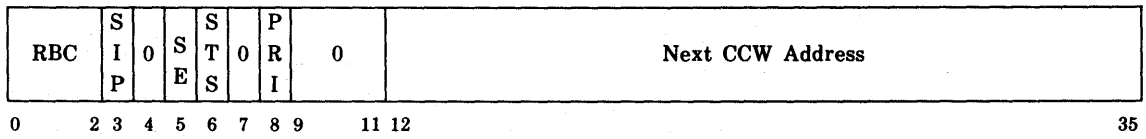
Snapshot Word 0



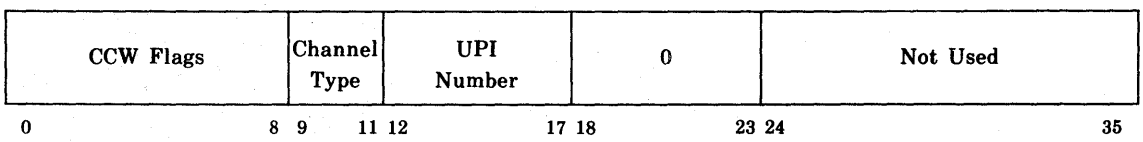
Snapshot Word 1



Snapshot Word 2

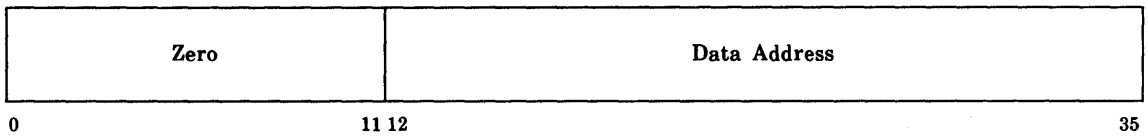


Snapshot Word 3

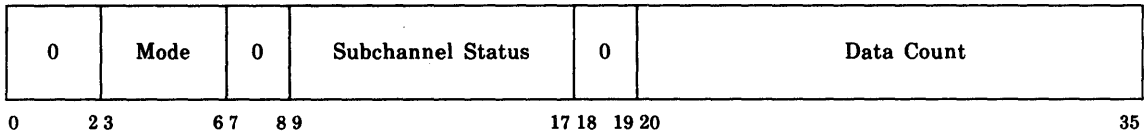


The format of the Snapshot words as provided by the DCC is:

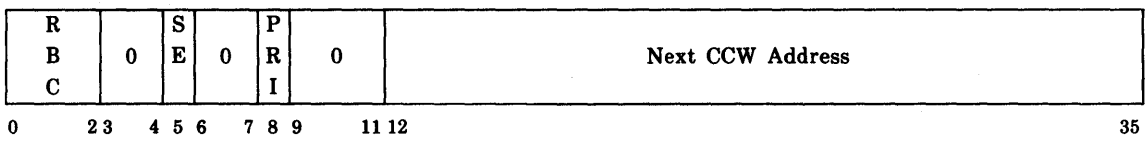
Snapshot Word 0



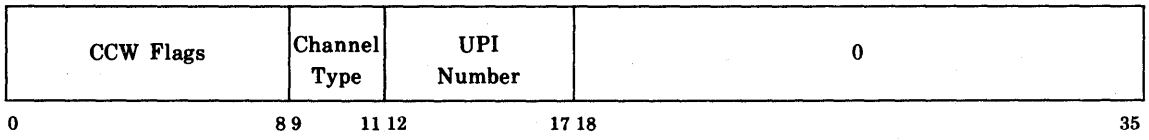
Snapshot Word 1



Snapshot Word 2



Snapshot Word 3



The meaning of the various fields appearing in the TSC order code snapshot words are given in Table 3-4.

A three-bit condition code, returned to software in bits 0-2 of CAW1, indicates the response to the TSC order code. Bit 3 (IOC) of CAW1 is cleared (logical 0). The general format of the condition code is:

Condition Code	Explanation
0	The order code was successfully executed.
1	The order code encountered a hardware fault.
2	The fault was detected while attempting to read the CAWs.
3	The addressed subchannel was not available.

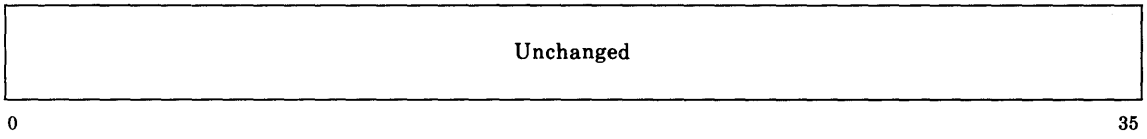
Table 3-4. Test Subchannel Order Code Fields

Field	Resulting Action
Data Address	Reports the current value of the address being used to read/write data.
Mode	Reports subchannel mode
Subchannel Status	Reports subchannel status 09 Device Not Available (Block MUX) 10 Incorrect Length 11 Program Check 12 Device Status Fault (DSF)(Block MUX) 13 Unsolicited Status 14 MSU Interface Check 15 Deferred Condition Code 16 Internal Channel IOP Fault 17 Channel Fault Word (CFW) Written
Data Count	Reports the current value of the Word Count remaining for the current CCW.
Residual Byte Count (RBC)	Reports the current value of the Residual Byte Count of BMCs.
Sense in Progress	If set, indicates a sense operation is in progress (not used by DCC).
Transparent Control Unit Busy (TCUB) Enable	If set, indicates that the TCUB option is enabled (not used by DCC and BBC).
Subchannel Enable	Always set in Snapshot Word, indicates that the subchannel is enabled.
Status Table Select	If set, indicates that the subchannel has Status Tabling enabled (not used by DCC).
Device Path Select	If set, indicates that the subchannel has Device Path Selection enabled (not used by DCC and BBC).
Priority	If set, indicates that the subchannel is acting on an SIOF that had Priority set.
Next CCW Address	Reports the address of the next CCW.
CCW Flags	Reports the CCW flags of the currently active CCW (excluding format flags of Block Multiplexer CCW).
Channel Type	Identifies the type of channel: ■ 01 = BMC and BBC ■ 02 = DCC
UPI Number	Reports the UPI number for this Channel IOP.
Device Path Selection Table Offset	Specifies the offset into the Device Path Selection table when the device path selection is enabled (not used by DCC and BBC).

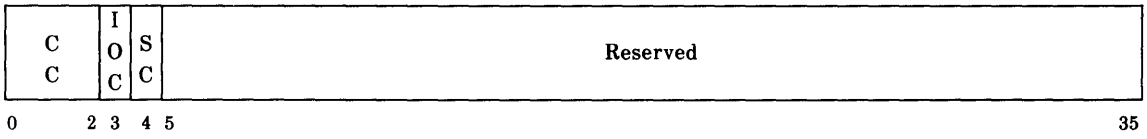
4,5	Not applicable.
6	An MSU/Channel IOP fault was encountered while writing the snapshot or the storage location specified by the absolute address in CAW1 was not available.
7	The snapshot address specified in CAW1 was not on a double-word absolute address boundary.
Subchannel (SC) Fault	If this bit set and the condition code is equal to 1, then the hardware fault is subchannel related and does not involve the entire IOP.

The CAW format after the Channel IOP presents a condition code is:

CAW0



CAW1



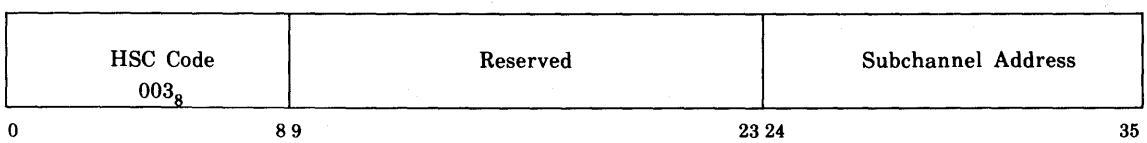
3.2.6.3. Halt Subchannel

The Halt Subchannel (HSC) order code sets the specified subchannel to the idle state.

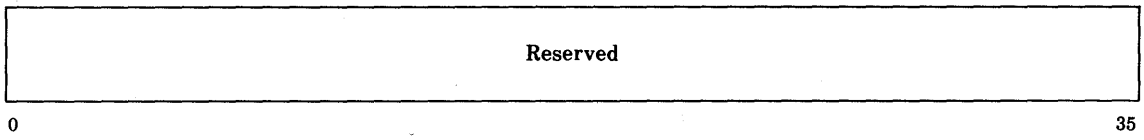
If the subchannel is in the status pending state, its status is cleared and the subchannel is returned to the idle state. A device status, initiated by the device following a HSC order code may be presented to the software as unsolicited status. It may lead to the continuation of an I/O operation, if allowed by the presented status and the device path selection. All state associated with the subchannel is cleared to a no operation pending condition.

The format of the CAW for the HSC order code is:

CAW0



CAW1



A three-bit condition code, returned to software in bits 0–2 of CAW1, indicates response to the HSC order code. Bit 3 (IOC) of CAW1 is cleared (logical 0). The general format of the condition code is:

Condition Code	Explanation
0	The order code was successfully executed.
1	The order code encountered a hardware fault.
2	A fault was detected while attempting to read the CAWs.
3	The addressed subchannel was not available.
4,5	Not applicable.
6,7	Undefined.

3.2.6.4. Clear Subchannel

The Clear Subchannel (CSC) order code sets the addressed subchannel to the idle state. If the device associated with the subchannel is logically connected on a block multiplexer channel or byte bus channel, the device is disconnected from the channel by a selective reset sequence.

If the channel associated with the subchannel is an Internally Specified Index (ISI) word channel, such as a disk controller channel, the CSC order code sets the subchannel to the idle state. No selective reset is issued.

If the subchannel is in the status pending state, its status is cleared and the subchannel is returned to the idle state.

A device status initiated by the device following a CSC order code, may be presented to the software as unsolicited status. Or, it may lead to the continuation of an I/O operation if allowed by the presented status and the device path selection.

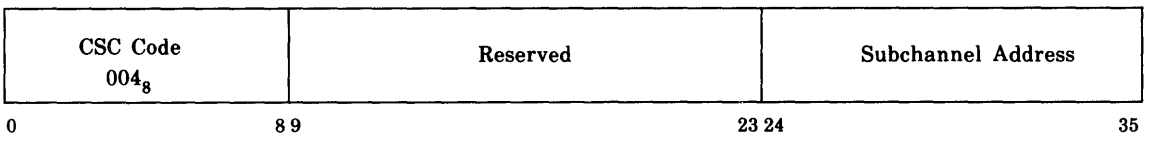
As a result of a CSC order code being successfully executed, the Channel IOP will not initiate further operations on that subchannel until either:

- An order code is received specifying further operations.
- An unsolicited interrupt is received by the device.

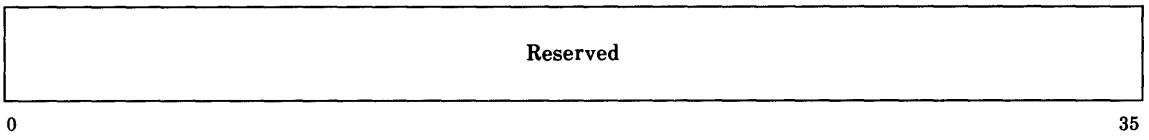
All current and pending operations are cleared.

The format of the CAW for the CSC order code is:

CAW0



CAW1



A three-bit condition code, returned to software in bits 0-2 of CAW1 indicates response of the CSC order code. Bit 3 (IOC) of CAW1 if cleared (logic 0). The general format of the condition code is:

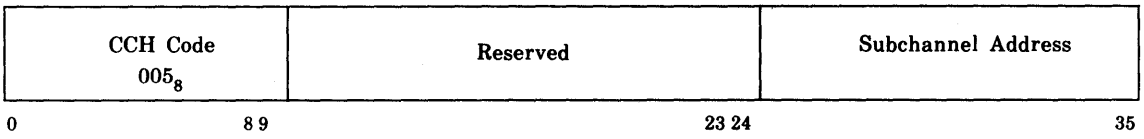
Condition Code	Explanation
0	The order code was successfully executed. The control unit or DCC device is not issued a selective reset sequence.
1	The order code encountered a hardware fault.
2	A fault was detected while attempting to read the CAWs.
3	The addressed subchannel was not available.
4,5	Not applicable.
6	The order code was successfully executed. The associated channel is a block multiplexer channel or byte bus channel; therefore, a selective reset sequence has been executed.
7	Undefined.

3.2.6.5. Clear Channel

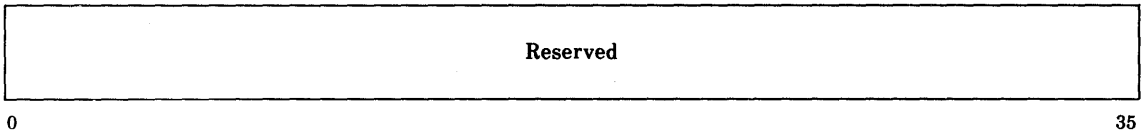
The Clear Channel (CCH) order code sets the addressed subchannel to the idle state. If the channel associated with the subchannel is a block multiplexer channel or byte bus channel, a system reset sequence is executed to all connected control units. If the channel associated with the subchannel is a word channel, such as a disk controller channel, a master clear is issued only to the one addressed channel and its associated control unit. All state associated with the subchannel is cleared to a no operation pending condition.

The format of the CAW for the CCH order code is:

CAW0



CAW1



A three-bit condition code, returned to software in bits 0-2 of CAW1, indicates the response of the CCH order code. Bit 3 (IOC) of CAW1 is cleared (logical 0). The general format of the condition code is:

Condition Code	Explanation
0	The order code was successfully executed.
1	The order code encountered a hardware fault.
2	A fault was detected while attempting to read the CAWs.
3	The addressed subchannel was not available.
4,5	Not applicable.
6,7	Undefined.

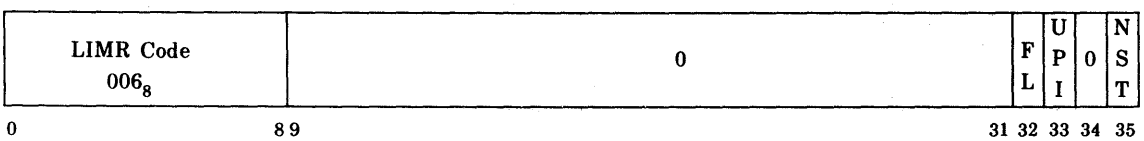
The reset of the device path select table is software responsibility.

3.2.6.6. Load Interrupt Mask Register

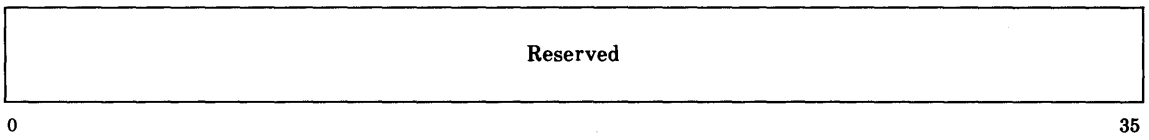
The Load Interrupt Mask Register (LIMR) order code allows software to inhibit/enable Channel IOP interrupts.

The format of the LIMR CAW is:

CAW0



CAW1



where, for CAW0:

Bits 0–8 = 006_8	Octal value of LIMR order code.
Bits 9–31	Reserved for software.
Bit 32 = 0	Allow Fault Log (FL) entry mode interrupts.
Bit 32 = 1	Prevent all interrupts that result from an entry being made in the Fault Log.
Bit 33 = 0	Allow all interrupts from subchannels using the UPI for status reporting.
Bit 33 = 1	Prevent all interrupts from subchannels using the UPI for status reporting.
Bit 34 = 0	Unused.
Bit 35 = 0	Allow Noncommunications Status Table (NST) interrupts.
Bit 35 = 1	Prevent all interrupts from the Noncommunications Status Table. Table entries continue to be made.

A three–bit condition code, returned to software in bits 0–2 of CAW1, indicates the response of the LIMR order code. Bit 3 (IOC) of CAW1 is cleared (logical 0). The general format of the condition code is:

Condition Code	Explanation
0	The order code was successfully executed.
1	The order code encountered a hardware fault.
2	A fault was detected while attempting to read the CAWs.
3–5	Not applicable.
6,7	Undefined.

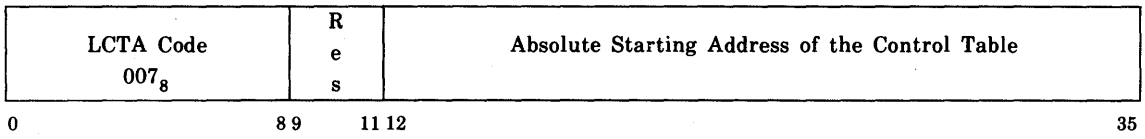
3.2.6.7. Load Control Table Address

The Load Control Table Address (LCTA) order code allows software to move the location of a UPI control table in storage. The control table base address, specified in bits 12-35 of CAW0, is an absolute address that must specify a 16-word boundary. Bits 32-35 must be zero.

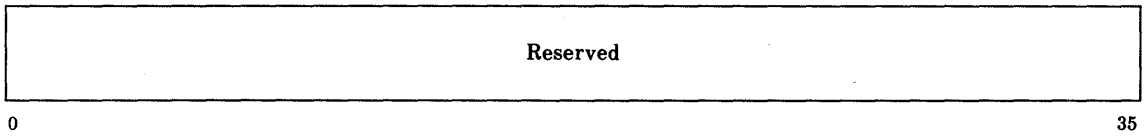
The Channel IOP makes an MSU request to the first and last address of the specified table location to determine MSU availability prior to returning a condition code.

The format of the CAW for the LCTA order code is:

CAW0



CAW1



A Channel IOP master clear operation sets the starting address of the UPI control table to Module Select Register (MSR) plus 120_8. The MSR can be changed, since it is a storage location. A new storage address once executed, defaults to MSR+120_8 on a master clear.

A three-bit condition code, returned to software in bits 0-2 of CAW1 indicates the response of the LTCA order code. Bit 3 (IOC) of CAW1 is cleared (logical 0).

The condition code is returned under control of the previous, current table. Thus, the operation does not take effect until after the condition code is stored successfully.

Condition Code	Explanation
0	The order code was successfully executed.
1	The order code encountered a hardware fault.
2	A fault was detected while attempting to read the CAWs.
3-5	Not applicable.
6	The specified control table was "Not Available" in the MSU, or an MSU/Channel IOP interface fault was encountered while performing the table available check.
7	The control table address specified was not on a 16-word boundary.

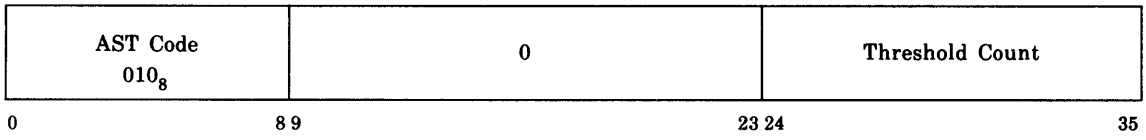
NOTE: For reliability and ease of subsequent partitioning, the UPI table should be allocated to be completely contained in one physical storage unit. It is therefore recommended to arrange tables on absolute address boundaries.

3.2.6.8. Activate Status Table

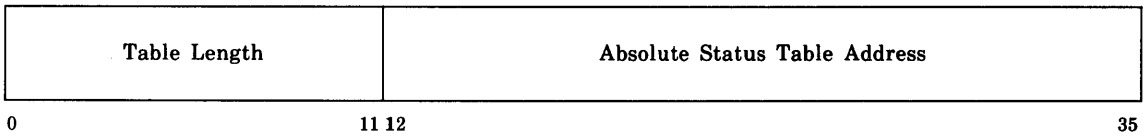
The Activate Status Table (AST) order code allows software to define the parameters for the stopped or inactive status table of the Channel IOP. This table is located in the main storage unit. Bit 9 of CAW0 must be zero for the AST order code to be executed. After the Channel IOP complex executes the AST order code, status tabling continues using the new table parameters. The AST order is not executed in the DCC; bit 3 is set (logical 1) in CAW1 to indicate illegal order code should this happen.

The format of the CAW for the AST order code is:

CAW0



CAW1



where, for CAW0:

Bits 0–8 = 010_8	Octal value of AST order code.
Bits 9–23 = 0	Field of all zeros, including bit 9 which enables executing of AST order code.
Bits 24–35 = Threshold Count	Specifies the number of 4-word status table entries the IOP is to make in the status table before generating a status table threshold interrupt.

NOTE: Zero threshold count inhibits all threshold count interrupts.

where, for CAW1:

Bits 0–11 = Table Length	Specifies the number of 4-word entries available in the status table; must be at least two words.
Bits 12–35 = Absolute Status Table Address	Specifies the starting address of the status table. This address must select a multiple of four words as a boundary.

NOTE: No threshold interrupts are generated if a threshold count of zero is selected, or a number larger than table size.

A three-bit condition code, returned to software in bits 0–2 of CAW1, indicates the response of the AST order code. Bit 3 (IOC) of CAW1 becomes set if the AST order code is executed in the DCC. This results in an illegal order code indication.

The general format of the condition code is:

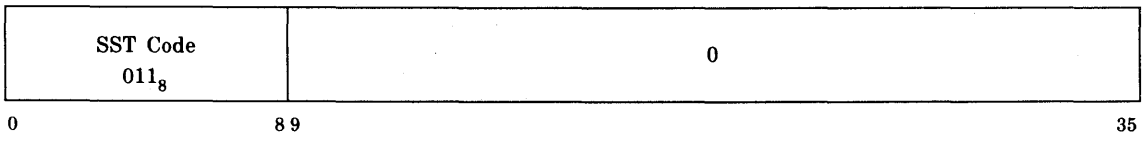
Condition Code	Explanation
0	The order code was successfully executed.
1	A hardware fault was encountered during execution of the order.
2	A fault was detected while attempting to read the CAWs.
3	Not Applicable.
4	The status table was active or bit 9 of CAW0 was not zero.
5	Not Applicable.
6	The status table length field contained a value that was less than two.
7	The specified status table starting address was not on a four-word boundary.

3.2.6.9. Stop Status Table

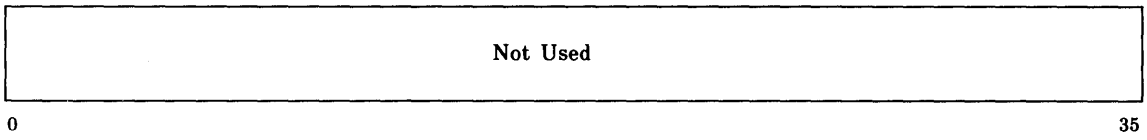
The Stop Status Table (SST) order code allows software to inhibit the status tabling feature in the Channel IOP. Subsequent status data, that would normally be tabled, is stacked. Bit 9 of CAW0 must be zero for the SST order to be executed. Subchannel status data is stacked in the subchannel buffer and device status data is stacked in the device buffer. The SST order is not executed in the DCC; bit 3 is set (logical 1) in CAW1 to indicate illegal order code should this happen.

The format of the CAW for an SST order is:

CAW0



CAW1



where, for CAW0:

Bits 0–8 = 011_8	Octal value of SST order code.
Bits 9–35 = 0	Field of all zeros, including bit 9 which enables execution of SST order code.

A three-bit condition code, returned to software in bits 0–2 of CAW1, indicates the response of the SST order code. Bit 3 (IOC) of CAW1 becomes set if the SST order code is executed in the DCC. This results in an illegal order code indication.

The general format of the condition code is:

Condition Code	Explanation
0	The order code was successfully executed.
1	A hardware fault was encountered during execution of the order.
2	A fault was detected while attempting to read the CAWs.
3–5	Not Applicable.
6	Bit 9 of CAW0 was not zero.
7	Not Applicable.

3.2.6.10. Update Status Stable

The Update Status Table (UST) order code allows software to replace the current status table threshold count and remaining-table-entries count in the Channel IOP. Bit 9 of CAW0 must be zero for the UST order code to be executed. If the table has not been activated or has been stopped, no attempt is made by the Channel IOP to update the threshold count and the remaining-table-entries count parameters.

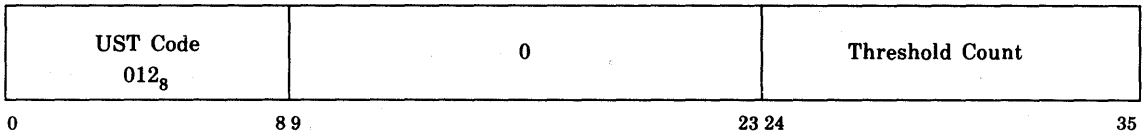
The number of status entries processed by software (contained in CAW1) is added to the number of remaining entries (the Channel IOP has this count) and the total is compared to the maximum table length (saved by the Channel IOP from the previous AST order code). If this total is greater than the maximum table length, no attempt is made to update the table parameter. If this total is less than or equal to the maximum table length, this total becomes the new remaining-table-entries count in the Channel IOP.

If the specified threshold count (CAW0) is zero, this field is improved and the Channel IOP continues with the threshold count in CAW0 replaces the current threshold count in the Channel IOP.

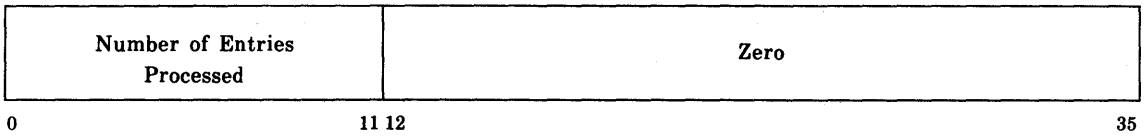
The UST order code is not executed in the DCC; bit 3 is set (logical 1) in CAW1 to indicate an illegal order code should this happen.

The format of the CAW for an UST order code is:

CAW0



CAW1



where, for CAW0:

Bits 0–8 = 012_8	Octal value of UST order code.
Bits 9–23 = 0	Field of all zeros, including bit 9 which enables execution of UST order code.
Bits 24–35 = Threshold Count	Specifies the number of 4-word status table entries the IOP is to enter in the status table before generating a status table threshold interrupt.

NOTE: Zero threshold count inhibits all threshold count interrupts.

where, for CAW1:

Bits 0–11 = Number of Entries Processed	Specifies the number of 4-word entries for software processing.
Bits 12–35 = 0:	Field of all zeros.

NOTE: No threshold interrupts are generated if a threshold count of zero is selected, or a number larger than table size.

A three-bit condition code, returned to software in bits 0–2 of CAW1, indicates the response of the UST order code. Bit 3 (IOC) of CAW1 becomes set if the UST order code is executed in the DCC. This results in an illegal order code indication.

The general format of the condition code is:

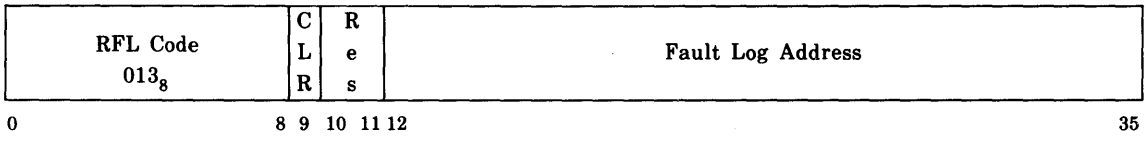
Condition Code	Explanation
0	The order code was successfully executed.
1	A hardware fault was encountered during execution of the order.
2	A fault was detected while attempting to read the CAWs.
3–5	Not applicable.
6	The status table to be updated was not active, or bit 9 of CAW0 was not zero.
7	The number of entries processed by software plus the number of available table entries was greater than the table length established by the AST order.

3.2.6.11. Read Fault Log

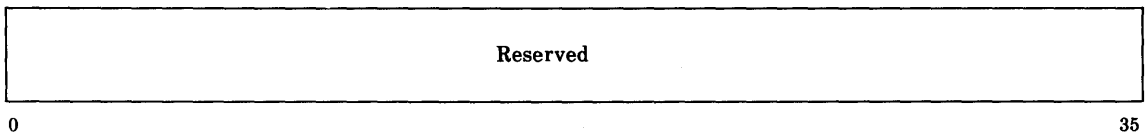
The Read Fault Log (RFL) order code is used by software to read the contents of the IOP fault log. The entries in the fault log are written into the MSU, beginning at the address designated by CAW0.

The format of the CAW for the Read Fault Log order code is:

CAW0



CAW1



where, for CAW0:

Bits 0–8 = 013_8	Octal value of RFL order code.
Bit 9 = Clear (CLR)	If set (logical 1), this bit indicates that the Channel IOP is to reset the fault log to its empty state after writing the contents into the MSU. If clear (logical 0), the Channel IOP is to maintain the present fault log contents for future software use.

Bits 10–11 = Reserved — Reserved for software.

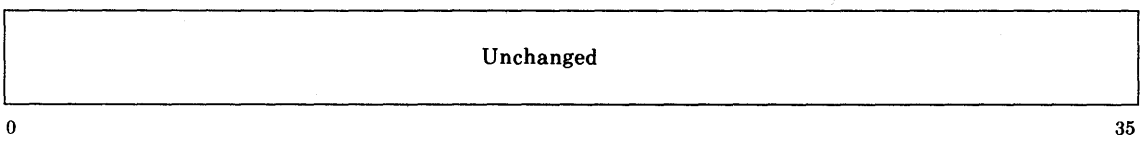
Bits 12–35 = Fault Log Address — Specifies absolute starting address for writing log entries into the MSU.

where, for CAW1:

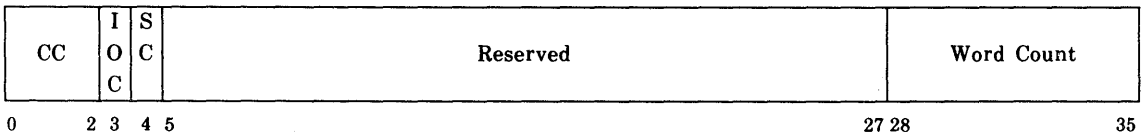
Bits 0–35 = Reserved — Reserved for software.

The data fields of the RFL order code are configured for information return as follows. This is the response of Channel IOP to software.

CAW0



CAW1



where, for CAW1:

Bits 0–2 = Condition Code (CC) — The Channel IOP interprets and responds to all order codes by writing a three-bit condition code before acknowledging the UPI request.

NOTE: The return of a non-zero condition code leaves the subchannel in a state as if the I/O order code had never been executed. The RFL order code initiates a state change on the subchannel only if a zero condition code is returned.

The format of the RFL condition code is:

Condition Code	Explanation
0	The order code was successfully executed.
1	The order code encountered a hardware fault.
2	A fault was detected while attempting to read the CAWs.
3–5	Not applicable.
6	An MSU/Channel IOP fault was detected while writing the fault log contents into the MSU.

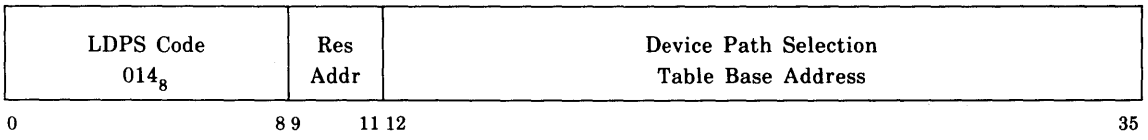
7	The specified address was not on a double–word multiple (boundary) or was "not available".
Bit 3 = Illegal Order Code (IOC)	If the Channel IOP recognizes an illegal order code in CAW0, it responds by setting the Illegal Order Code indicator bit to 1. When this bit is set, the condition code field is specified 7, as a "don't care" selection. *NOTE: Illegal order codes include order codes not implemented.*
Bit 4 = Subchannel Fault (SC)	If this bit is set and the Condition Code is 1, hardware fault, then the fault is subchannel related and does not involve the entire IOP.
Bits 5–27 = Reserved	Reserved for software.
Bits 28–35 = Word Count	During execution of an RFL order code, only the valid (existing) fault log entries are written into the MSU. Software is informed of the number of valid words written by the Word Count.

3.2.6.12. Load Device Path Selection Base Register

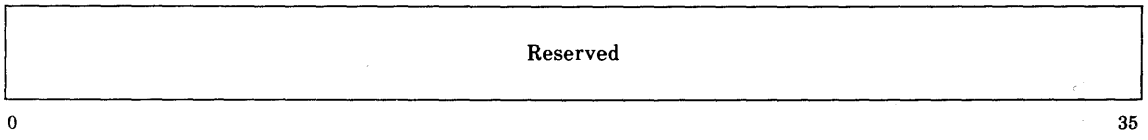
The Load Device Path Selection (LDPS) base register order code permits Software to establish or relocate the IOP's device path selection control table. Only the BMC can execute the LDPS order code.

The format of the CAW for an LDPS order code is:

CAW0



CAW1



NOTE: It is recommended that the Device Path Selection Table Base Address be selected so that the lowest n bits are zeros, where n is a number such that the length of the table minus one can be represented in n bits.

Thus, if bits 2 through 35 are zeros, a table 255 words in length can be addressed by this number (n). The address, then, should be an even multiple of the table size designator.

A three-bit condition code, returned to software in bits 0–2 of CAW1, indicates the response of the LDPS order code. Bit 3 (IOC) of CAW1 becomes set if the LDPS order code is executed in the BBC or DCC. This results in an illegal order code indication.

The Channel IOP makes an MSU request to the first and last address of the specified control location to determine availability prior to returning a condition code to software. Rather than a separate routine, the response to this request is indicated with the return of Condition Code 6.

The general format of this condition code is:

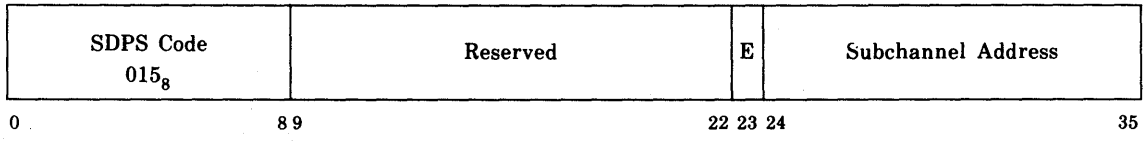
Condition Code	Explanation
0	The order code was successfully executed.
1	The order code encountered a hardware fault.
2	A fault was detected while attempting to read the CAWs.
3–5	Not applicable.
6	Control table address specified was not available.
7	Undefined.

3.2.6.13. Select Device Path Selection

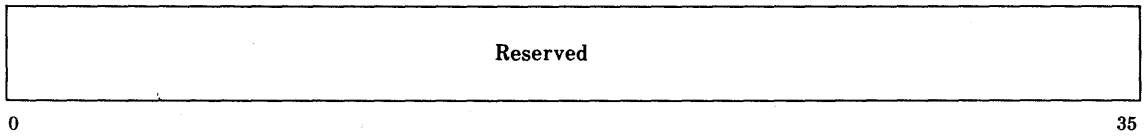
The Select Device Path Selection (SDPS) order code allows software to enable or disable the device path selection function on a subchannel basis. Only the BMC can execute the SDPS order code.

The format of the CAW for a SDPS order code is:

CAW0



CAW1



where, for CAW0:

Bits 0–8 = 015_8 Octal value of SDPS order code.

Bits 9-22 = Reserved	Reserved for software.
Bit 23 = Enable (E)	If set, this bit specifies the Channel IOP enabling the device path selection function on the addressed subchannel. If cleared (logical 0), the Channel IOP disables the device path selection function on the addressed subchannel.
Bits 24-35 = Subchannel Address	Specifies the address of device-related subchannel.

where, for CAW1:

Bits 0-35 = Reserved	Reserved for software.

A three-bit condition code, returned to software in bits 0-2 of CAW1, indicates the response of the SDPS order code. Bit 3 (IOC) of CAW1 becomes set if the SDPS order code is executed in the BBC or DCC. This results in an illegal order code indication.

The general format of the SDPS condition code is:

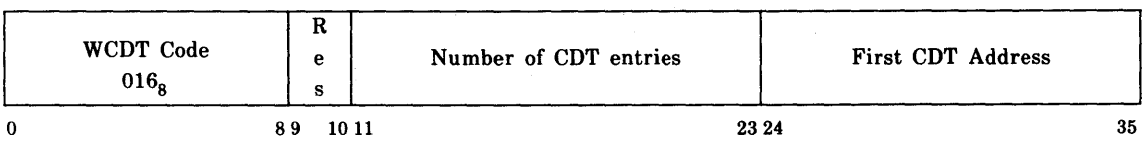
Condition Code	Explanation
0	The order code was successfully executed.
1	The order code encountered a hardware fault.
2	A fault was detected while attempting to read the CAWs.
3	The addressed subchannel was unavailable.
4,5	Not applicable.
6	Undefined.
7	The subchannel address specified a subchannel that does not provide the device path selection function.

3.2.6.14. Write Channel Descriptor Table

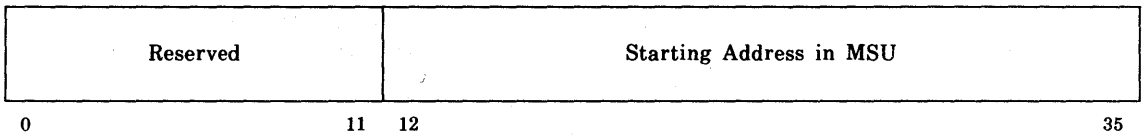
The Write Channel Descriptor Table (WCDT) order code allows software to write the contents of one or more of the Channel Descriptor Table (CDT) words.

The format of the CAW for a WCDT order code is:

CAW0



CAW1



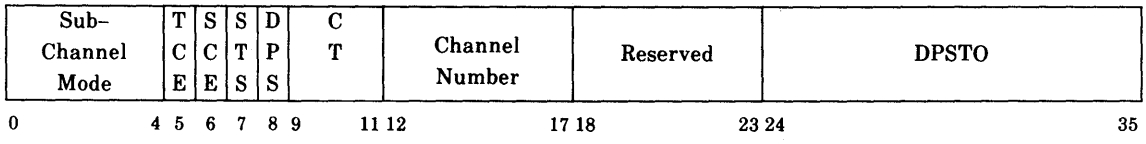
where, for CAW0:

Bits 0–8 = 016_8	Octal value of WCDT order code.
Bits 9–10 = Reserved	Reserved for software.
Bits 11–23 = Number of CDT Entries	The value of this field is equal to the number of CDT entries to be written.
Bits 24–35 = First CDT Address	The value of this field is equal to the first CDT address in Channel IOP.

where, for CAW1:

Bits 0–11 = Reserved	Reserved for software.
Bits 12–35 = Starting Address	The value of this field is equal to the starting storage address in MSU.

The CDT information is written from the MSU storage locations to Channel IOP storage in the following format.



NOTE: When this order code is executed, the Subchannel Mode of the CDT address is always set to Idle, regardless of the contents of bits 0–4 of the storage location.

A three–bit condition code, returned to software in bits 0–2 of CAW1, indicates the response of the WCDT order code. Bit 3 (IOC) of CAW1 is cleared (logical 0).

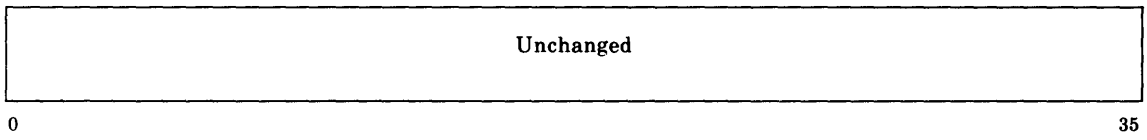
The general format of the WCDT condition code is:

Condition Code	Explanation
0	The order was successfully executed.
1	The order code encountered a hardware fault.
2	A fault was detected while attempting to read the CAWs.

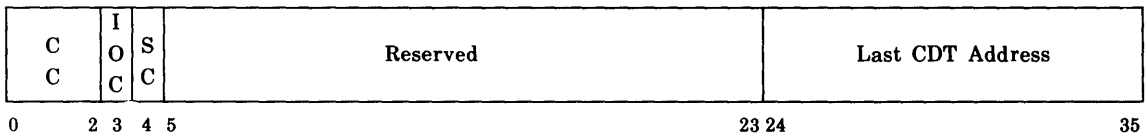
3-5	Not applicable.
6	An MSU/Channel IOP fault was detected while reading from MSU.
7	First CDT address plus N-1 exceeds table length; or N = 0.

The CAW format by which the Channel IOP presents a condition code is:

CAW0



CAW1



where, for CAW1:

Bits 0-2 = Condition Code (CC) — The Channel IOP interprets and responds to all order codes by writing a three-bit condition code before acknowledging the UPI request.

NOTE: The return of a non-zero condition code leaves the subchannel in a state as if the I/O order code had never been executed. The WCDT order code initiates a state change or the subchannel only if a zero condition is returned.

Bit 3 = Illegal Order Code (IOC) — If the Channel IOP recognizes an illegal order code in CAW0, it responds by setting the illegal order code indicator bit to 1. When this bit is set, the condition code field is specified 7, as a 'don't care' selection.

NOTE: Illegal order codes include order codes not implemented.

Bit 4 = Subchannel Fault (SC) — If this bit is set and the condition code is 1, hardware fault, then the fault is subchannel related and does not involve the entire IOP.

Bits 5-23 = Reserved — Reserved for software.

Bits 24-35 = Last CDT Address — The value of this field is equal to the last CDT address in Channel IOP.

NOTE: If a condition code of 1 or a condition code of 6, with bit 4 set (logical 1), is returned from the Channel IOP to software, the CDT address involved is reported in the bits 24-35 of CAW1.

3.2.6.15. Inject MSU Fault

The Inject MSU Fault (IMF) order code insures the fault detecting and reporting logic associated with the IOP/MSU interface is operating properly. Faults are injected on a subchannel basis to provide concurrent normal operation and confidence testing. Only the most recent IMF order which is successfully executed (that is, condition code = 0) is effective in an IOP at any given time.

An IMF order with an activity code of zero cancels any fault injection currently in effect. A Reset Clear or Orderly Halt command also clears any fault injection currently in effect.

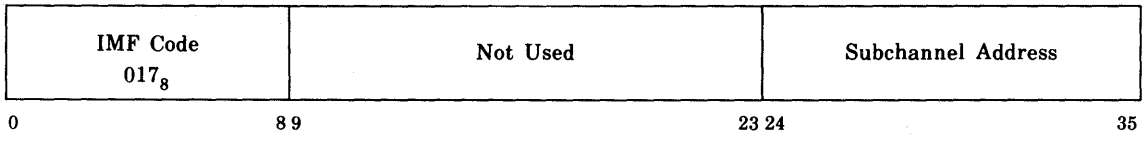
If single occurrence is specified, and an error other than the error designated by the IMF occurs on the specified subchannel, then fault injection is cancelled. The IMF can also be used to inject faults out the S-bus during information transfers to other control units by correctly specifying the Fault Type and Activity.

An IMF for a particular subchannel must be transferred to the IOP before that subchannel is activated. If the subchannel is currently active when the IMF is received, a condition code of zero is returned to software, but the fault is never injected. Fault injection is never implemented until the Inject MSU Fault order code is successfully executed.

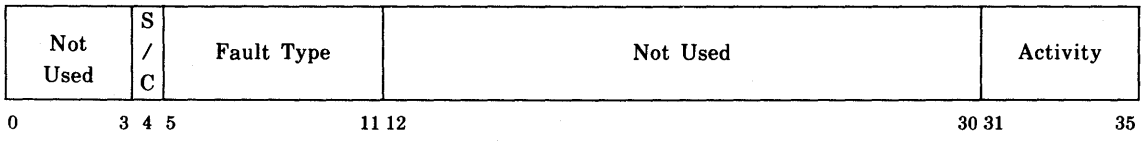
The locations where faults may be injected do not represent all the detection points in an IOP. The injection locations do reflect all instances where different methods of hardware, microcode, or operating system software recovery are effective.

The format of the CAW for an IMF order code is:

CAW0



CAW1



where, for CAW0:

Bits 0–8 = 017_8 — Octal value of IMF order code.

Bits 9–23 = Not Used — These bits are not used in this application.

Bits 24–35 = Subchannel Address — Specifies the address of device-related subchannel.

where, for CAW1:

Bits 0–3 = Not Used — These bits are not used in this application.

Bit 4 = Set/Clear (S/C) — Specifies whether the fault is to be injected as a continuous condition or as a single occurrence; when set (logical 1), continuous is specified.

Bit 5–11 = Fault Type — Identifies the specific type of MSU fault to be injected. Fault type codes in the range 101–105 identify faults to be injected by the Channel IOP onto the S–Bus; fault type code 110 specifies that a fault is to be injected onto the S–Bus by the MSU.

The code 110 causes the Channel IOP to substitute Diagnostic Read or Diagnostic Write MSU functions in place of the normal Read and Write. The specific function is determined by the selected activity.

The External Function register in the MSU must be conditioned by the software to inject the appropriate error prior to the occurrence of the activity in the Channel IOP. Refer to Table 3–5 for a list of fault type codes and a description of each.

Table 3–5. Fault Type Code

Fault Type Code (Octal)	Description
000	Cancel Fault Inject
001–100	Not Used
101	Inject parity fault on bits 00–05 of the first word of an S–bus message.
102	Inject parity fault on bits 12–17 of the first word of an S–bus message.
103	Inject parity fault on the Format field of an S–bus message.
104	Inject parity fault on the Destination Address field of an S–bus message.

Table 3-5. Fault Type Code (continued)

Fault Type Code (Octal)	Description
105	Inject parity fault on the Source Address field of an S-bus message.
106-107	Not Used
110	Use the Diagnostic Write MSU function for an activity requiring an MSU write or use the Diagnostic Read MSU function for an activity requiring an MSU read.
111-177	Not Used

Bits 12-30 = Not Used — These bits are not used in this application.

Bits 31-35 = Activity — Specifies the task to be in progress when the fault is injected. If an activity is selected that does not make use of the hardware related to the fault type code, no error is injected. Table 3-6 lists the activity code, a description of the activity, and the fault type codes that produces an error for the activity.

Table 3-6. Activity Codes

Activity Code (octal)	Description	Related Fault Type Code (octal)
00	Cancel fault injection	Don't Care
01	Transfer Channel Address Word (CAW)1 from MSU	101-105,110
02	Transfer Channel Command Word (CCW) 0 from MSU	101-105,110
03	Transfer External Function (EF) 0 from MSU (DCC only)	101-105,110
04	Transfer Device Path Selection Table (DPSTW) from MSU (BMC only)	101-105,110
05	Write CAW1 into MSU	101-105,110
06	Write Channel Status Word (CSW) into MSU	101-105,110
07	Write Table Status Word (TSW) 0 into MSU	101-105,110

Table 3-6. Activity Codes (continued)

Activity Code (octal)	Description	Related Fault Type Code (octal)
10	Write DPSTW into MSU (BMC only)	101-105,110
11	Channel IOP transmits a UPI Send Command	101-105
12	Channel IOP transmits a UPI Acknowledge (ACK) Command	101-105
13	Not used	None
14	Output data transfer (S-Bus)	101-105,110
15	Write the first word of the fault log into MSU	101,105,110
16-21	Not used	None
22	Input data transfer (S-Bus)	101-105,110

A three-bit condition code returned to software in bits 0-2 of CAW1, indicates the response of the IMF order code. Bit 3 (IOC) of CAW1 is cleared (logical 0).

The general format of the IMF condition code is:

Condition Code	Explanation
0	The order was successfully executed.
1	A hardware fault was encountered during execution of the order.
2	A hardware fault was detected while attempting to read the CAW.
3	The addressed subchannel was unavailable.
4-7	Not applicable.

3.2.6.16. Inject I/O Internal Fault

The Inject I/O internal Fault (IIF) order code assures that internal IOP fault detecting, reporting logic, and microcode is functioning properly. Faults are injected on a subchannel basis to provide concurrent normal operation and confidence testing. Only the most recent IIF order code, which is successfully executed (i.e., condition code = 0), is in force in an IOP at any one time. An IIF order with an injection code of 000 cancels any fault injection, if successfully executed.

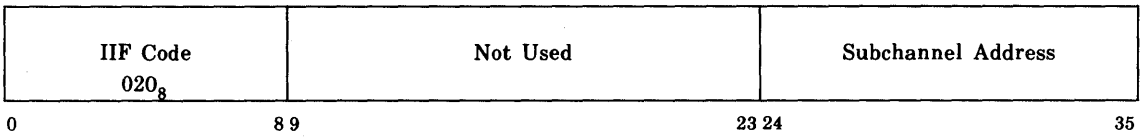
If single occurrence is specified and an error other than the error designated by the IIF occurs on the specified subchannel, then fault injection is cancelled. A Reset Clear or Orderly Halt command also clears any fault injection currently in effect.

The IIF for a particular subchannel must be sent to the Channel IOP before that subchannel is activated. If the subchannel is currently active when the IIF is received, a condition code of 0 is returned but the fault is never injected.

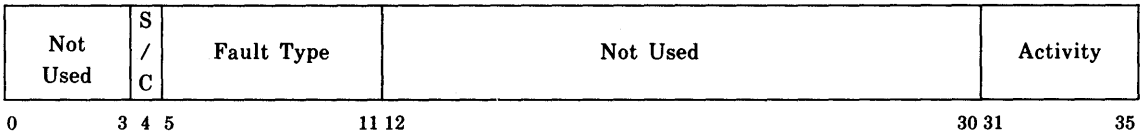
The locations that may be injected do not represent all of the detection points in a Channel IOP. The locations do not reflect all cases where different methods of hardware microcode or operating system software recovery are in effect.

The format of the CAW for an IIF order code is:

CAW0



CAW1



where, for CAW0:

Bits 0–8 = 020_8	Octal value of IIF order code.
Bits 9–23 = Not Used	These bits are not used in this application.
Bits 24–35 = Subchannel Address	Specifies the address of device-related subchannel.

where, for CAW1:

Bits 0–3 = Not Used	These bits are not used in this application.
Bit 4 = Set/Clear (S/C)	Specifies whether the fault is to be injected as a continuous condition or as a single occurrence; when set (logical 1), continuous is designated. If single occurrence is specified, the fault is injected the first time through the activity and then cancelled. A continuous fault is injected each time through an activity until cancelled by a Reset Clear, Orderly Halt, or an IIF order with a fault type code of zero.
Bits 5–11 = Fault Type	Specifies the fault injection point in the logic. Table 3–7 lists the fault type codes that are common to all three Channel IOPs. The exception is codes 67 and 72, which are common to the DCC and BMC only. The table also contains a description of the fault that occurs as a result of each code's use.

Table 3-8 contains a list of valid fault types that are unique to the DCC. A description of the fault that occurs as a result of each code's use is also presented.

Table 3-9 contains a list of valid fault types that are unique to the BMC. A description of the fault that occurs as a result of each code's use is also presented.

Table 3-10 contains a list of valid fault types that are unique to the BBC. A description of the fault that occurs as a result of each code's use is also presented.

Table 3-7. Common I/O Channel Fault Type Codes

Injection Code (Octal)	Description of the fault that occurs
060	Parity fault on bits 00-05 of the Buffer Memory Bus
061	Parity fault on bits 06-11 of the Buffer Memory Bus
062	Parity Fault on bits 12-17 of the Buffer Memory Bus
063	Parity fault on bits 18-23 of the Buffer Memory Bus
064	Parity fault on bits 24-29 of the Buffer Memory Bus
065	Parity fault on bits 30-35 of the Buffer Memory Bus
066	Not used
067	Parity fault on the Data Buffer Amount Adder (DCC and BMC only). Parity fault on the Buffer Memory Bus during an S-bus sequencing element initiated transfer (BBC only).
070	Parity fault on S-bus information bits 00-05 indicated by the S-bus common logic.
071	Parity fault on S-bus information bits 12-17 indicated by the S-bus common logic.
072	Parity fault on the Data Buffer Address Selector (DCC and BMC)
073	On output an indication is present that the word from the S-Bus Data Register did not get written into the Data Buffer
074	Parity fault on the Word Count Adder
075	Parity fault on bits 00-11 of the MSU Address Adder
076	Parity fault on bits 12-23 of the MSU Address Adder
077	Not used
106	Parity fault on bits 00-05 of the Interval Timer
107	Parity fault on data bits 00-05 from Control Store Memory
110-177	Not used

Table 3-8. Fault Type Codes for the DCC

Fault Type Code (Octal)	Description of the fault that occurs
000	Cancels fault injection
001	Parity fault on input data from String Controller Bus A
002	Parity fault on input data from String Controller Bus B
003	On output, the total parity of a word does not agree with the accumulated parity of each byte.
004	Parity fault on output data from bits 00–17 of the Staging Register to bits 00–17 of the Disk Assy Register
005	Parity fault on output data from bits 00–17 of the Staging Register to bits 18–35 of the Disk Assy Register
006	On input, the total parity of a word did not agree with the accumulated parity of the bytes from the disk interface.
007	Parity fault on the disk interface Residual Register
010	Parity fault on output data in bits 00–17 of the Staging Register
011	Parity fault on output data in bits 00–17 of the Staging Register
012	Parity fault in the Byte Pair Counter
013	Parity fault in the Data Count Register
014	Incorrect Length indication on a data transfer
015	Parity error on incoming device status
016	Parity fault is injected onto Byte Bus A. This fault should be detected and reported by the String Controller.
017–057	Not used

Table 3-9. Fault Type Codes for the BMC

Fault Type Code (Octal)	Description of the fault that occurs
000	Cancels fault injection
001	Not Used
002	Not Used
003	Channel Data Overrun condition, input or output operation
004	Parity fault in the Device Interface Residual Register
005	Parity fault in bits 00–17 of the Device Interface Staging Register
006	Parity fault in bits 18–35 of the Device Interface Staging Register
007	Parity fault in bits 00–35 of the Device Interface Assembly Register
010	Illegal inbound tags from the control unit during initial selection
011	Parity fault in the Device Interface Bus Out Register
012	Parity fault in the Device Interface Byte Count Rank B Register
013	Parity fault in the Device Interface SI Byte Count Register or the DI Byte Count Register
014	Parity fault in the Device Interface Bus in Register
015	Parity fault in the Device Interface Status in Register
016	Parity fault injected onto Bus Out. This fault should be detected by the control unit.
017–057	Not Used

Table 3-10. Fault Type Codes for the BBC

Fault Type Code (Octal)	Description of the fault that occurs
000	Cancels fault injection
001	Parity error on the L-bus data register on inputting status
002	L-Bus data parity error on output data transfers
003	L-Bus data parity error on outputting Block Multiplex command
004	Parity error on bits 0-5 of the State RAM Bus when writing to State RAM during data transfers
005	Parity error on bits 0-3 of the Data RAM Bus during an output data transfer
006	L-Bus data parity error on outputting device address
007	Parity error on the L-bus data register during input data transfer
010-057	Not used

Bits 12-30 = Not Used — These bits are not used in this application.

Bits 31-35 = Activity — Specifies the task in program during which the fault is to be injected. If an activity is selected which does not make use of the hardware related to the fault type code, then no error is injected. Table 3-11 contains a listing of the activity code, a description of the activity, and fault type codes that produce an error for that activity in each I/O channel.

Table 3-11. Activity Code and Related Fault Type Code for Each Channel

Activity Code (Octal)	Description	Related Injection Codes (Octal)		
		DCC	BMC	BBC
00	Cancel fault injection	None	None	None
01	Transfer CAW1 from main storage	70, 71, 106, 107	70, 71, 106, 107	70, 71, 106, 107
02	Transfer CCW 0 from main storage	70, 71, 106, 107	70, 71, 106, 107	70, 71, 106, 107
03	Transfer External Function (EF) 0 from main storage (DCC only)	70, 71, 106, 107	None	None
04	Transfer Device Path Selection Table Word (DPSTW) from main storage (BMC only)	None	70, 71, 106, 107	None
05	Write CAW1 into main storage	70, 71, 106, 107	70, 71, 106, 107	70, 71, 106, 107
06	Write Channel Status Word (CSW) 0 into main storage	70, 71, 106, 107	70, 71, 106, 107	70, 71, 106, 107
07	Write Table Status Word (TSW) 0 into main storage	None	70, 71, 106, 107	70, 71, 106, 107
10	Write DPSTW into main storage (BMC only)	None	70, 71, 106, 107	None
11	I/O transmits a Universal Processor Interface (UPI) Send message	70, 71, 106, 107	70, 71, 106, 107	70, 71, 106, 107
12	I/O transmits a UPI Acknowledge (ACK) message	70, 71, 106, 107	70, 71, 106, 107	70, 71, 106, 107
13	Output data transfer Device Interface	3-5, 7-14, 16, 60-67 72, 106, 107	3-7, 11-13, 16, 60-67 72, 106, 107	2, 4, 5, 60-65, 106, 107
14	Output data transfer (S-bus I/F)	70, 71, 73-76, 106, 107	70, 71, 73-76, 106, 107	70, 71, 73-76, 106, 107

Table 3-11. Activity Code and Related Fault Type Code for Each Channel (continued)

Activity Code (Octal)	Description	Related Injection Codes (Octal)		
		DCC	BMC	BBC
15	Write the first word of the fault log into main storage	70, 71, 106, 107	70, 71, 106, 107	70, 71, 106, 107
16	Output an address during a subsystem selection sequence	16	10, 11, 16	6
17	Output a command to a subsystem	16	10, 11, 16	3
20	Status response from a subsystem	15	15	1
21	Input data transfer (Device I/F)	1, 2, 6-14, 60-67, 72, 106, 107	3-7, 12-14, 60-67, 72, 106, 107	4, 7, 60-65, 106, 107
22	Input data transfer (S-bus I/F)	70, 71, 73-76, 106, 107	70, 71, 73-76, 106, 107	70, 71, 73-78, 106, 107
23-37	Not Used			

A three-bit condition code, returned to software in bits 0-2 of CAW1, indicates the response of the IIF order code. Bit 3 (IOC) of CAW1 is cleared (logical 0).

The general format of the IIF condition code is:

Condition Code	Explanation
0	The order was successfully executed.
1	A hardware fault was encountered during execution of the order.
2	A fault was detected while attempting to read the CAWs.
3	The addressed subchannel was unavailable.
4-7	Not applicable.

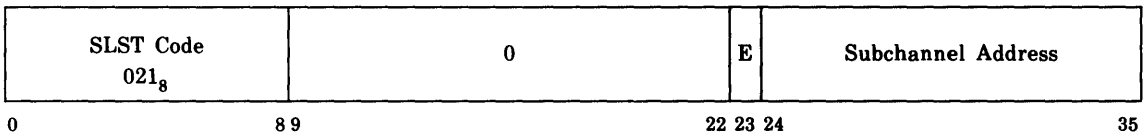
3.2.6.17. Select Status Tabling

The Select Status Tabling (SLST) order code allows software to enable and disable the tabling of status on a subchannel basis.

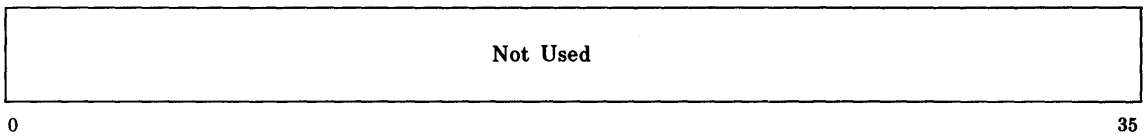
The SLST order code is not executed in the DCC; bit three is set (logical 1) in CAW1 to indicate illegal order code should this happen.

The format of the CAW for an SLST order code is:

CAW0



CAW1



where, for CAW0:

Bits 0–8 = 021_8	Octal value of SLST order code.
Bits 9–22 = Zero	The value of this field is set at all zeros.
Bit 23 = Enable (E)	This bit specifies, if set (logical 1), that all ensuing status for this subchannel be reported through status tabling. If clear (logical 0), all ensuing status is reported through the UPI interface.
Bits 24–35 = Subchannel Address	Specifies the address of device-related subchannel.

where, for CAW1:

Bits 0–35 = Not Used	These bits are not used in this application.

A three-bit condition code, returned to software in bits 0–2 of CAW1, indicates the response of the SLST order code. Bit 3 (IOC) of CAW1 becomes set if the SLST order code is executed in the DCC. This results in an illegal order code indication.

The general format of the condition code is:

Condition Code	Explanation
0	The order code was successfully executed.
1	The order encountered a hardware fault.
2	A fault was detected while attempting to read the CAWs.
3	The addressed subchannel was unavailable.
4–7	Not applicable.

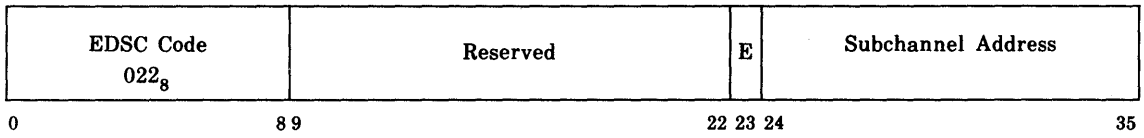
NOTE: The return of a non-zero condition code leaves the subchannel in a state as if the I/O order code had never been executed. The SLST order code initiates a state change on the subchannel only if a zero condition code is returned.

3.2.6.18. Enable/Disable Subchannel

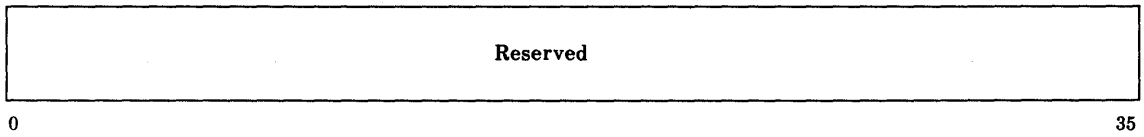
The Enable/Disable Subchannel (EDSC) order code allows software to disable the subchannel specified. Software may also change a disabled subchannel back to normal operation by setting the CAW0 enable bit to logical one.

The format of the CAW for an EDSC order code is:

CAW0



CAW1



where, for CAW0:

Bits 0–8 = 022_8	Octal value of EDSC order code.
Bits 9–22 = Reserved	Reserved for software.
Bit 23 = Enable (E)	Places the addressed subchannel in the enabled state with a logical 1; places the subchannel in a disabled state with logical 0.
Bit 24–35 = Subchannel Address	Specifies the address of device-related subchannel.

where, for CAW1:

Bits 0–35 = Reserved	Reserved for software.

A three-bit condition code, returned to software in bits 0–2 of CAW1, indicates the response of the EDSC order code. Bit 3 (IOC) of CAW1 is cleared (logical 0).

The general format of the condition code is:

Condition Code	Explanation
0	The order code was successfully executed.
1	The order code encountered a hardware fault.
2	A fault was detected while attempting to read the CAWs.
3	The addressed subchannel was not available.
4	The addressed subchannel was busy.
5	Not applicable.
6,7	Undefined.

NOTE: The return of a non-zero condition code leaves the subchannel in a state as if the I/O order code had never been executed. The EDSC order code initiates a state change on the subchannel only if a zero condition code is returned.

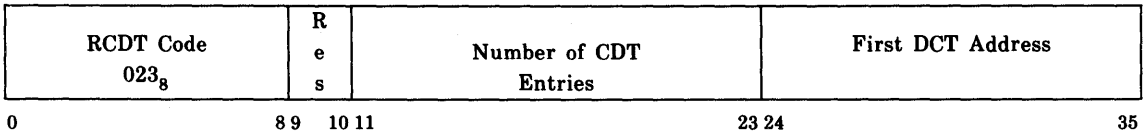
If a subchannel is disabled, all order codes specifying that subchannel, except an EDSC, are rejected with a condition code of 3. The EDSC order code is rejected with a condition code of 3 if the channel type of the addressed subchannel is unassigned or undefined. All status from the device associated with the disabled subchannel is acknowledged by the channel and discarded by the Channel IOP.

3.2.6.19. Read Channel Descriptor Table

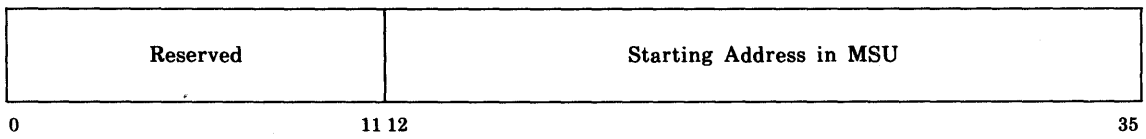
The Read Channel Descriptor Table (RCDT) order code allows software to read the contents of one or more of the Channel Descriptor Table (CDT) words.

The format of the CAW for an RCDT order code is:

CAW0



CAW1



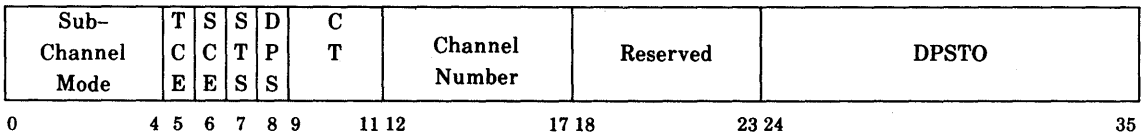
where, for CAW0:

Bits 0–8 = 016_8	Octal value of RCDT order code.
Bits 9–10 = Reserved	Reserved for software.
Bits 11–23 = Number of CDT Entries	The value of this field is equal to the number of CDT entries to be written.
Bits 24–35 = First CDT Address	The value of this field is equal to the first CDT address in IOP.

where, for CAW1:

Bits 0–11 = Reserved	Reserved for software.
Bits 12–35 = Starting Address	The value of this field is equal to the starting storage address in MSU.

The CDT information is read into the MSU storage locations from Channel IOP storage in the following format:



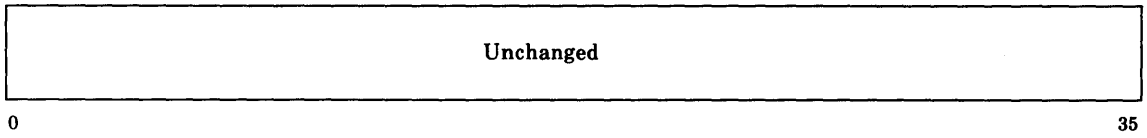
A three-bit condition code, returned to software in bits 0–2 of CAW1, indicates the response of the RCDT order code. Bit 3 of CAW1 is cleared (logical 0).

The general format of the RCDT condition code is:

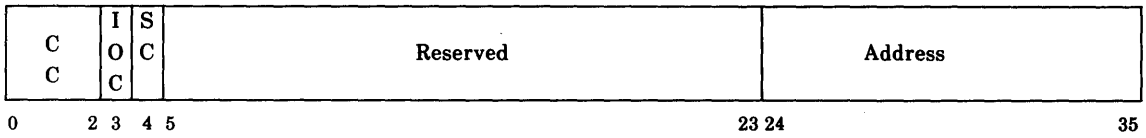
Condition Code	Explanation
0	The order was successfully executed.
1	The order code encountered a hardware fault.
2	A fault was detected while attempting to read the CAWs.
3–5	Not applicable.
6	An IOP/MSU fault was detected while reading from MSU.
7	First CDT address plus N–1 exceeds table length; or N = 0.

The CAW format by which the Channel IOP presents a condition code is:

CAW0



CAW1



where, for CAW1:

Bits 0–2 = Condition Code (CC) — The Channel IOP interprets and responds to all order codes by writing a three-bit condition code before acknowledging the UPI request.

NOTE: *The return of a non-zero condition code leaves the subchannel in a state as if the I/O order code had never been executed. The RCDT order code initiates a state change on the subchannel only if a zero condition code is returned.*

Bit 3 = Illegal Order Code (IOC) — If the Channel IOP recognizes an illegal order code in CAW0, it responds by setting the illegal order code indicator bit to 1. When this bit is set, the condition code field is specified 7, as a "don't care" selection.

NOTE: *Illegal order codes include order codes not implemented.*

Bit 4 = Subchannel Fault (SC) — If this bit is set and the Condition Code is 1, hardware fault, then the fault is subchannel related and does not involve the entire Channel IOP.

Bits 5–23 = Reserved — Reserved for software.

Bits 24–35 = Address — The value of this field is equal to the last CDT address in the Channel IOP.

NOTE: *If a condition code of 1 or 6, with bit 4 set (logical 1), is returned from Channel IOP to software, the CDT address involved is reported in bits 24–35 of CAW1.*

3.3. Channel Command Word

The Channel Command Word (CCW) is a 72-bit control word used to govern data transfers to and from a specific peripheral. The CCW is written into main storage by the IP. It is specified by the address field of CAW1 for the Start I/O Fast order code operation.

The double-word CCW may be located anywhere in storage, but it must be on a double-word boundary. A channel program is comprised of one or more CCWs chained together by CCW control flags and Transfer in Channel (TIC) commands. Normally, the CCWs are stored as a list in contiguous double-word storage locations and are read and executed sequentially by the Channel IOP. However, a TIC command code in the currently active CCW causes the Channel IOP to read the next CCW from the storage location specified by the address in the CCW containing the TIC command.

A CCW specifies the operation to be performed by a device or subchannel. When the designated operation is due to be executed, the CCW is read by the Channel IOP and moved to the appropriate subchannel storage location. The CCW image in the subchannel is then modified and updated to control the operation as it progresses.

3.3.1. Format

The general CCW format for the Channel IOP is dependent on the application. The operation of each I/O channel is determined by whether it is byte oriented or word oriented. Format conventions and protocol which govern this operation are:

- The DCC operates following the protocol and format conventions defined for the ISI word channel. However, the DCC responds to the monitor bit with program check subchannel status.

  The format of the CCW used by the DCC is shown in 3.3.2 with a description of each bit position.

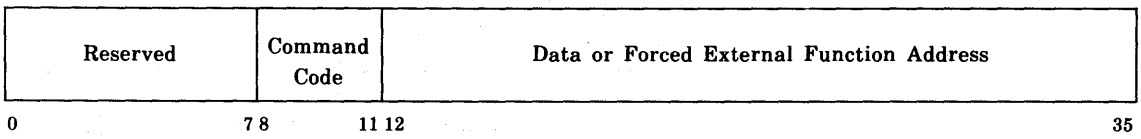
- The BBC and BMC operate following the protocol and format conventions defined for the Block Multiplexer CCW. However, data format B (six-bit packed) is not supported by either channel.

  The format of the CCW used by the BBC and BMC is shown in 3.3.3 with a description of each bit position.

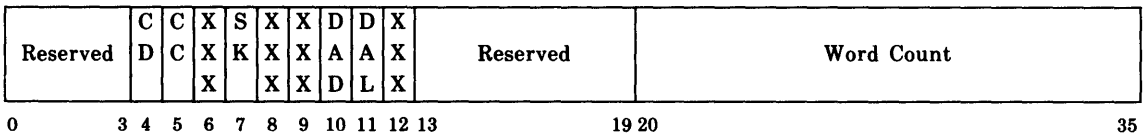
3.3.2. Internally Specified Index Word Channel

The ISI word channel provides the protocol and format conventions for operation of the DCC. The CCW format for a word channel module shared ISI interface is:

CCW0



CCW1



where, for CCW0:

Bits 0–7 = Reserved — Reserved for software.

Bits 8–11 = Command Code — Specifies the operation performed by the subchannel. The command executed by the device is contained in the Forced External Function (FEF) word. In the command codes listed below, the letter X indicates that the bit position is ignored.

Bit Positions				Command
08	09	10	11	
0	0	0	0	Invalid
0	1	0	0	Invalid
1	0	0	0	Transfer In Channel
1	1	0	0	Invalid
X	X	0	1	Activate Output
X	X	1	0	Activate Input
X	X	1	1	Forced External Function

Bits 12–35 = Data or FEF Address — Contains the absolute storage address for the first data or FEF word transferred, unless the command code is Transfer In Channel command. In this case, the field contains the absolute storage address of the next CCW.

where, for CCW1:

Bits 0–3 = Reserved — Reserved for software.

Bits 4–12 = CCW Flags

Bit 4 = Chain Data (CD) — Specifies that upon exhaustion of the word count of current CCW, a new CCW is read from storage and the operation is continued under control of the new CCW.

Bit 5 = Chain Command (CC) — Specifies that upon exhaustion of the word count of the current CCW, a new CCW is read from storage and the operation specified by the new command code is initiated. If the Chain Data flag is set, the Chain Command flag is ignored.

Bit 6 = XXX — Bit position not used.

Bit 7 = Skip Data (SK) — Specifies that data is not written in storage for input operations. However, the subchannel and control words are handled in the same manner as during conventional

input operation. The Skip Data flag is ignored on output operations.

Bit 8-9 = XXX — Bit positions not used.

Bit 10 = Data Address Decrement (DAD) — Specifies that the data address be decreased by one for each data word transferred. This flag is ignored if the Data Address Lock (DAL) flag is set. If neither the DAL nor DAD flag is set, the data address is increased by one for each data word transferred. Increasing or decreasing the data address is a 24-bit twos-complement arithmetic operation.

Bit 11 = Data Address Lock (DAL) — Specifies that the contents of the data address field remain unchanged for each data word transferred under control of the current CCW.

Bit 12 = XXX — Bit position not used.

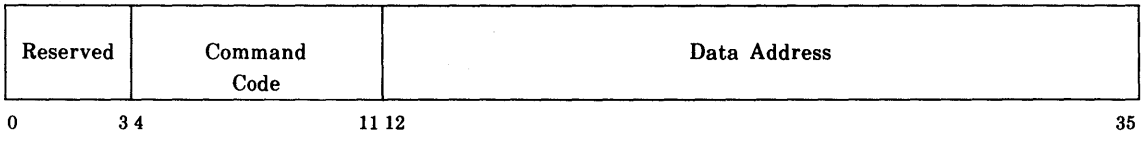
Bits 13-19 = Reserved — Reserved for software.

Bits 20-35 = Word Count — Specifies the number of words to be transferred to or from storage.

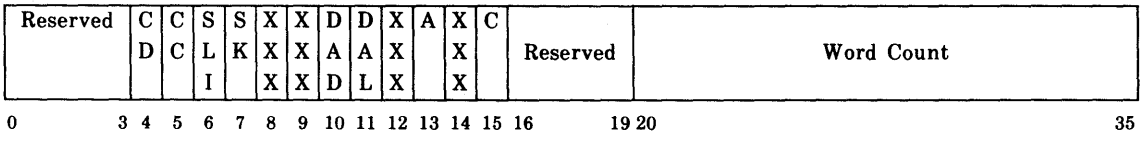
3.3.3. Block Multiplexer

The Block Multiplexer provides the protocol and format conventions for operation of the BBC and BMC. The CCW format for a byte/block oriented channel module shared interface is:

CCW0



CCW1



where, for CCW0:

Bits 0-3 = Reserved — Reserved for software.

Bits 4-11 = Command Code — Specifies the operation performed by the device and subchannel. In the command codes listed below, the letter X is ignored and the letter M identifies a modifier bit. The meaning of the modifier bits depends upon the type of I/O device.

Bit Position								Command
04	05	06	07	08	09	10	11	
X	X	X	X	0	0	0	0	Invalid
M	M	M	M	0	1	0	0	Sense
X	X	X	X	1	0	0	0	Transfer In Channel
M	M	M	M	1	1	0	0	Read Backward
M	M	M	M	M	M	0	1	Write
M	M	M	M	M	M	1	0	Read Forward
M	M	M	M	M	M	1	1	Control

Bits 12–35 = Data Address — Contains the absolute storage address for the first data word transferred, unless the command code is Transfer in Channel command. In this case, the field contains the absolute storage address of the new CCW.

where, for CCW1:

Bits 0–3 = Reserved — Reserved for software.

Bits 4–15 = CCW Flags

Bit 4 = Chain Data (CD) — Specifies that upon exhaustion of the word count of the current CCW, a new CCW is read from storage and the operation is continued under control of the new CCW.

Bit 5 = Chain Command (CC) — Specifies that upon completion of the operation at the device (status code contains Device End), a new CCW is read from storage and the operation specified by the new command is initiated. If chain data is set or incorrect length conditions are detected, the chain command flag is ignored.

Bit 6 = Suppress Length Indication (SLI) — The SLI flag disables the checking of subchannel word count versus the number of bytes presented by a device. If the SLI flag is not set and the number of bytes that a device attempts to transfer to the Channel IOP is not exactly equal to the CCW word count, the subchannel program is immediately terminated and incorrect length subchannel status is reported to the software. An SLI error does not occur if the last byte presented or requested by a device is partially contained in the last valid word. If the SLI flag is set and the number of bytes that a device attempts to transfer is not exactly equal to the CCW word count, the operation is unaffected and the execution of the subchannel program proceeds normally.

During execution of a block multiplexer data chain operation, the SLI flag is tested during initiation of the first CCW only. The state of the flag will be maintained throughout the CCW Data Chain list.

The SLI flag is tested and its condition saved out of the CCW pair read if one of the following conditions is met.

1. The previous CCW specified command chaining.
2. The CCW is the first one of a channel program.

Bit 7 = Skip Data (SK)	Specifies that data is not written in storage for input operations. However, the subchannel and control words are handled in the same manner as during conventional input operation. The Skip Data flag is ignored on output operations.
Bits 8-9 = XXX	Bit positions not used.
Bit 10 = Data Address Decrement (DAD)	Specifies that the data address be decreased by one for each data word transferred. This flag is ignored if the Data Address Lock (DAL) flag is set. If neither the DAL nor DAD flag is set, the data address is increased for each data word transferred. Increasing or decreasing the data address is a 24-bit twos complement arithmetic operation.
Bit 11 = Data Address Lock (DAL)	Specifies that the contents of the data address field remain unchanged for each data word transferred under control of the current CCW.
Bit 12 = XXX	Bit position not used.
Bit 13 = Format A	Specifies the quarter-word format for packing bytes into words and unpacking bytes from words. One byte is right justified in each 9-bit quarter word. If the leftmost bit is set in any quarter word during an output operation, the data transfer is terminated.
Bit 14 = XXX	Bit position not used.
Bit 15 = Format C	Specifies 8-bit packed format for packing and unpacking bytes from words. Four and one-half bytes are packed in each 36-bit word.
Bits 16-19 = Reserved	Reserved for software.
Bits 20-25 = Word Count	Specifies the number of words transferred to or from storage. All writes or reads to storage are full word transfers.

3.3.4. Unconditional Branching

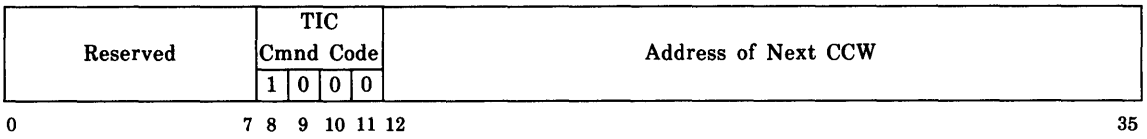
Unconditional CCW branching may be performed with the DCC, BBC, and BMC Channel IOPs. The Transfer in Channel (TIC) command provides the branching function for each channel. It allows for sequentially executing CCWs that are not at adjacent storage locations. It also allows for command and buffer loops.

A TIC is specified for a particular command code in a CCW.

- ISI Word Channel

  The word format for the TIC command follows. This is for the DCC, or word-oriented application.

  CCW0

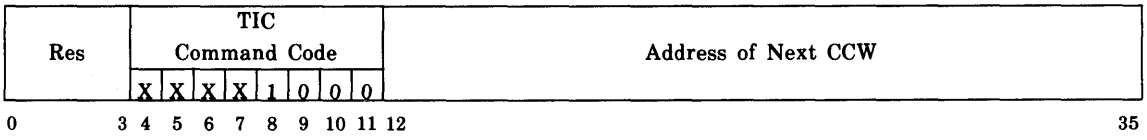


  As shown, bits 12–35 contain the absolute storage addressing of the next CCW.

- Block Multiplexer

  The word format for the TIC command follows. This is for the BBC and BMC, or byte/block-oriented application.

  CCW0



Bit positions 4 through 7 are to be ignored. As shown, bits 12–35 contain the absolute storage address of the next CCW.

NOTE: Observe that bit eight of the TIC Command Code is set (logical 1) for both ISI word channel and Block Multiplexer word formats. As long as this bit continues set, and the chain data bit is clear, branching remains unconditional with device end status indications.

3.3.5. Conditional Branching

Conditional CCW branching may be performed with the DCC, BBC, and BMC Channel IOPs.

The TIC instruction provides the branching command function for each channel. It allows for sequentially executing CCWs which are not at adjacent storage locations. It also allows for command and buffer loops. A TIC is specified by a particular command code in a CCW.

The storage address of the CCW is contained in the data address field of the TIC CCW.

Two conditional branching modes are described in the following paragraphs.

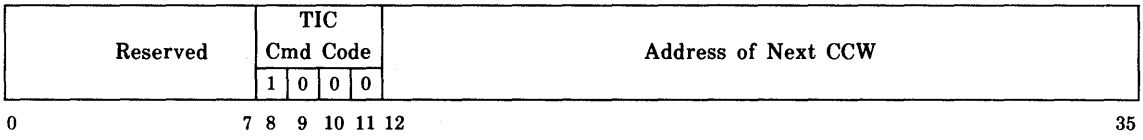
3.3.5.1. Command Chaining

Chaining commands are structured according to the IOP application.

- Word Orientation

  The word oriented DCC performs command chaining in the manner described for the ISI word channel (see 3.3.2). The word format for the TIC command is:

  CCW0

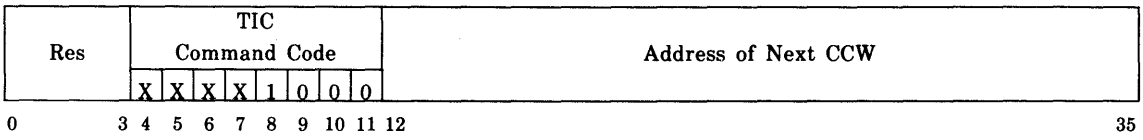


  As shown, bits 12–35 contain the absolute storage address of the next CCW.

- Block/Byte Orientation

  The block/byte oriented BBC and BMC perform command chaining in the manner described for the Block Multiplexer Channel (see 3.3.3). The word format for the TIC command is:

  CCW0



Bit positions 4 through 7 are to be ignored. As shown, bits 12–35 contain the absolute storage address of the next CCW.

3.3.5.2. Data Chaining

A subchannel executes a data chain by retrieving a new CCW from storage and continuing the previous operation under control of the data address and data count fields of the new CCW.

During a data chain operation, the command code field of the new CCW is ignored unless it contains a transfer in channel command.

For an ISI word channel, all CCW flags are replaced with the CCW flags in the new CCW.

For a Block Multiplexer Channel, the Suppress Length Indication (SLI) flag and all three format flags are carried from the first CCW through the entire CCW data chain list. All other CCW flags are replaced with the CCW flags in the new CCW.

On all subchannels, data chaining occurs if:

- the subchannel is active,
- the data count is exhausted,
- the Chain Data (CD) CCW flag is set, or
- no hardware fault is detected.

3.4. Status Reporting

The Channel IOPs report status by generating interrupts and writing status words. Their functioning is consistent with the ability of the System Support Processor (SSP) to initiate an order to an I/O device using the Universal Processor Interface (UPI) protocol. When this happens, the Channel IOP directs the I/O termination interrupt to the UPI assigned to the SSP rather than transmit the interrupt to the IP.

All I/O interrupts are presented to software by way of the UPI. All Channel IOP status is either reported through the UPI or through status tables.

A noncommunications status table is defined for each Channel IOP. Status reported through the UPI is translated with Channel Status Words (CSWs). Status words reported by status tabling are referred to as Table Status Words (TSWs).

A CSW/TSW is either a four-word or an eight-word entry. If no faults are detected during the I/O operation, a four-word CSW/TSW is written to report status. If a fault is detected, four Channel Fault Words (CFWs) are appended to the CSW/TSW and reported to software.

NOTE: If a fault is detected, and the subchannel has status tabling selected, the IOP ensures that two four-word entries are available prior to entering any information into the status table.

3.4.1. Input/Output Interrupts

Channel IOP interrupts are transmitted to the IP through the UPI. Each Channel IOP interrupt is uniquely defined by a 9-bit code in bits 0–8 of CSW0.

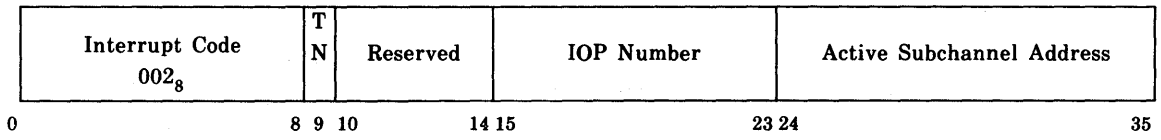
The following listing contains a description of each of the codes.

Interrupt Code	Description
000_8	Used for status table entries only.
001_8	I/O termination status is being reported through the UPI.
002_8	Channel IOP status pertaining to a status table is being reported.
003_8	Channel IOP status pertaining to a fault log is being reported.
004_8	Channel IOP Automatic Sense termination status for a subchannel that reports status through the Status Table.
005_8	Channel IOP Automatic Sense termination status for a subchannel that reports status through the UPI.
006_8–0777_8	Reserved.

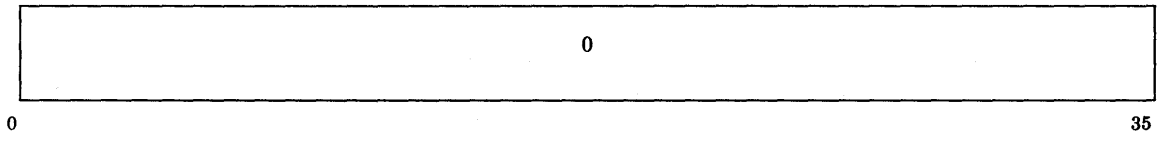
3.4.2. Status Table Presented CSW

All status pertaining to the operation of the status table is reported to software by the UPI. Only the BMC and BBC support status tabling, the DCC does not. The format of the CSW for an interrupt by a status table entry is:

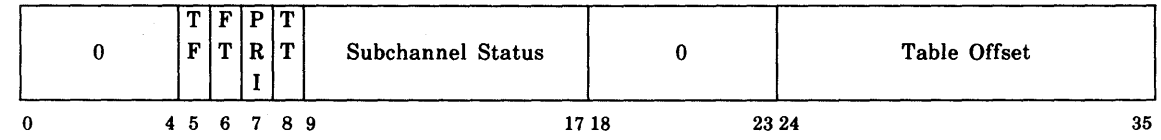
CSW0



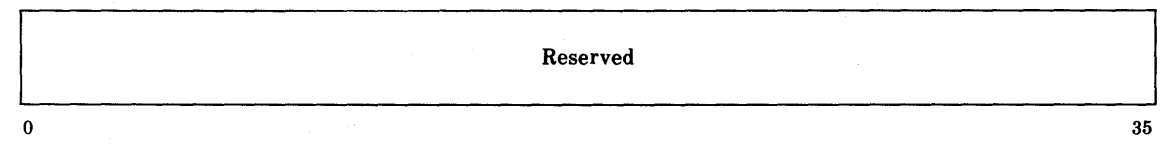
CSW1



CSW2



CSW3



where, for CSW0:

Bits 0–8 = Interrupt Code — Code 002 specifies that CSW was written and an interrupt generated due to status of table entry.

Bit 9 = Table Number (TN) — Specifies table number:
0 = noncommunications status table.
1 = not used.

Bits 10–14 = Reserved — Reserved for software.

Bits 15–23 = IOP Number — Identification of Channel IOP. Designation is UPI number in the range 0–511.

Bits 24–35 = Active Subchannel Address	Location in main storage.

where, for CSW1:

Bits 0–35 = Zero	Value of word field is all zeros.

where, for CSW2:

Bits 0–4 = Zero	Value of field is zero.
Bit 5 = Table Fault (TF)	Fault detected during status table operation that stopped table.
Bit 6 = Full Table (FT)	Status table entry specified by CSW0, bit 9 is full. The full condition is set when the number of available entries is less than two.
Bit 7 = Priority (PRI)	A priority table entry was made in the status table.
Bit 8 = Threshold (TT)	The number of table entries is equal to or greater than the table threshold value specified by software in the last AST order code.
Bits 9–17 = Subchannel Status	Refer to Table 3-12 for a detailed description of each bit for this field.
Bits 18–23 = Zero	Value of field is zero.
Bits 24–35 = Table Offset	Specifies the number of entries from the starting address where the table entry was made.

where, for CSW3:

Bits 0–35 = Reserved	Reserved for software.

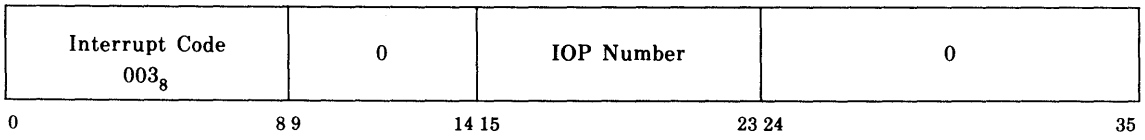
Table 3-12. Subchannel Status Field of Status Table CSW

Bit Position	Designation	Function
09	Zero	Value is zero.
10	Zero	Value is zero.
11	Program check	Set (logical 1) if the Channel IOP attempted to make a table entry is not available storage.
12	Zero	Value is zero.
13	Zero	Value is zero.
14	MSU Interface Fault	Set if a fault is detected on the MSU/Channel IOP interface. The MSU may detect a multiple uncorrectable error, or the Channel IOP detect a read data parity error, or an internal MSU check may occur during processing the addressed word.
15	Zero	Value is zero.
16	Internal Channel IOP Fault	Set if the IOP detects a hardware fault during a status table operation.
17	CFW Written	Set if the four channel fault words are written after the four word CSW.

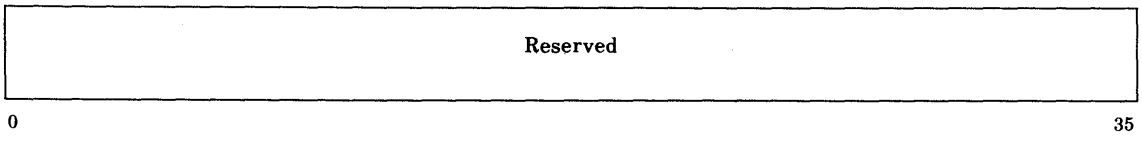
3.4.3. Fault Log Entry Presented CSW

The DCC, BBC, and BMC report fault log entry mode CSWs through the following word formats:

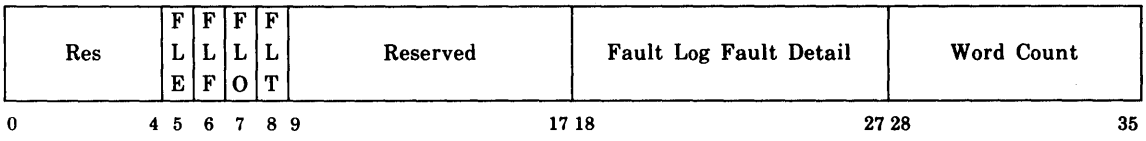
CSW0



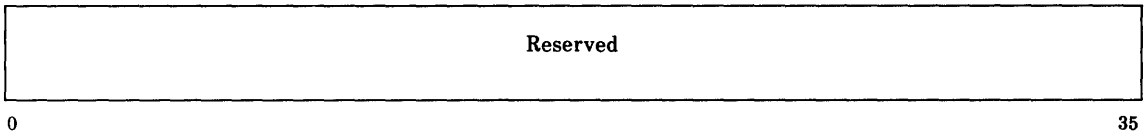
CSW1



CSW2



CSW3



where, for CSW0:

Bits 0–8 = 003_8	Specifies that the CSW is written and an interrupt generated due to a fault log entry.
Bits 9–14 = Zero	Value of field is zero.
Bits 15–23 = IOP Number	Identification of Channel IOP designation is UPI number in the range 0–511.
Bits 24–35 = Zero	Value of field is zero.

where, for CSW1:

Bits 0–35 = Reserved	Reserved for software.

where, for CSW2:

Bits 0–4 = Reserved	Reserved for software.
Bit 5 = Fault Log Error (FLE)	Set if a fault log read error occurred.
Bit 6 = Fault Log Full (FLF)	Set if the log is full or has overflowed.
Bit 7 = Fault Log Overflow (FLO)	Set if the fault log overflowed. When this condition exists, the threshold and full condition also exists.
Bit 8 = Fault Log Threshold (FLT)	Set if the threshold condition exists in the fault log.
Bit 9–17 = Reserved	Reserved for software.
Bits 18–27 = Fault Log Fault Detail	System dependent.
Bits 28–35 = Word Count	Specifies the number of valid words in the fault log at the time the fault log entry CSW was stored.

where, for CSW3:

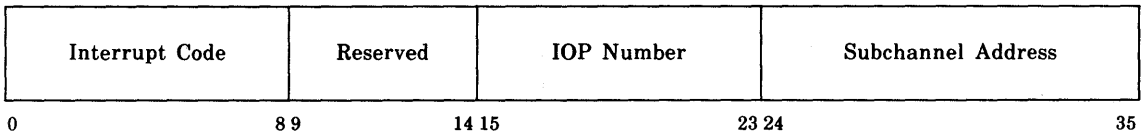
Bits 0–35 = Reserved — Reserved for software.

3.4.4. BBC and BMC Status Words

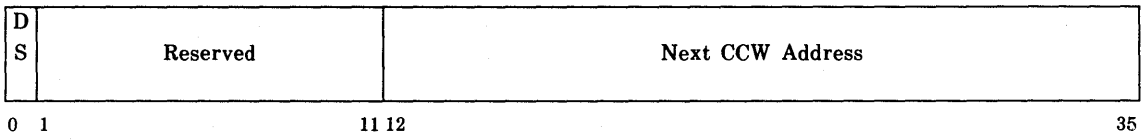
The BBC and BMC report Channel Status Words (CSWs) and Tabled Status Words (TSWs) using the following word format for a Block Multiplexer Channel:

NOTE: A Program Check subchannel status is generated whenever data format B is specified in a CCW for data transfer.

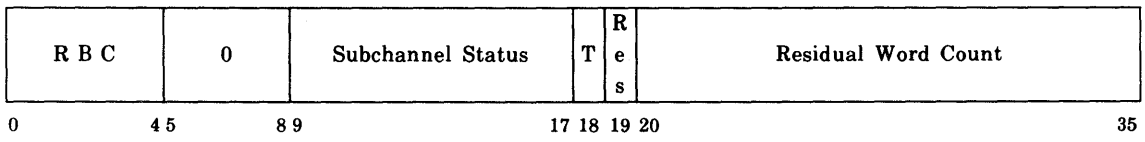
CSW0/TSW0



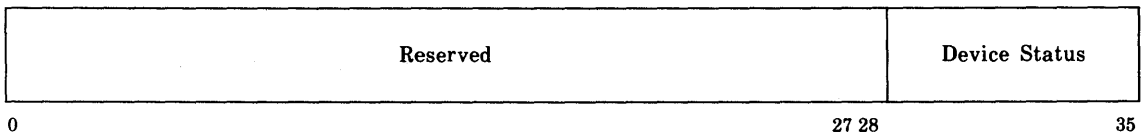
CSW1/TSW1



CSW2/TSW2



CSW3/TSW3



where, for CSW0/TSW0:

Bits 0–8 = Interrupt Code — Set if a CSW is written in the Control Table and an interrupt generated due to termination of an I/O operation. This field is set to zero (000) when the TSW is written in a Status Table.

Bits 9–14 = Reserved	Reserved for software.
Bits 15–23 = IOP Number	Identification of IOP. Designation is UPI Number in the range 0–511.
Bits 24–35 = Subchannel Address	Location in main storage.

where, for CSW1/TSW1:

Bit 0 = Device Status (DS)	Set if the Device Status field in bits 28–35 of CSW3/TSW3 contains status issued by the device or its control unit.
Bits 1–11 = Reserved	Reserved for software.
Bits 12–35 = Next CCW Address	This field contains the address of the next Channel Command Word (CCW), to be used at the time the status information is stored.

where, for CSW2/TSW2:

Bits 0–4 = Residual Byte Count (RBC)	The Channel IOP deals strictly with word counts. Any remaining bytes not transferred are reported by the Residual Byte Count. It indicates how may bytes remain in the last word or output, or how many unfilled byte locations are in the last word or output.
Bits 5–8 = Zero	The value of this field is zero.
Bits 9–17 = Subchannel Status	See Table 3–13.
Bits 18 = Toggle (T)	Used for TSWs only. This bit is set to one for the first pass through the status table and is toggled during each consecutive cycle.
Bit 19 = Reserved	Reserved for software.
Bits 20–35 = Residual Word Count	Contains the number of words not transferred under the active CCW to the MSU during an input operation, or not fully or partially transferred to the device during an output operation.

where, for CSW3/TSW3:

Bits 0–27 = Reserved	Reserved for software.
Bits 28–35 = Device Status	Contains a device or control unit generated status byte, if certain conditions are detected and reported by the device or control unit.

NOTE: The Residual Word Count field and the Residual Byte Count field values are undefined for CCWs that terminate because of a hardware fault.

Table 3-13. Subchannel Status Field of BBC and BMC Status Word

Bit Position	Designation	Function
9	Device not available	Addressed control unit or device was offline or powered down.
10	Incorrect length	Number of bytes specified in the CCW for an I/O operation does not equal the number of bytes requested or offered by the device. The suppress length indication CCW flag disables the reporting of an incorrect length condition.
11	Program check	The Program Check Subchannel status bit is set whenever a software error is detected. This provides an indication of one of the following conditions. ■ CCW with a TIC command did not specify a CCW address on a double word boundary. ■ CCW specified an invalid command. ■ CCW, with a command other than TIC, specified a word count of zero. ■ Two successive CCWs specify the TIC command. ■ Data format B was specified in a CCW that specifies data packing format for a data transfer. ■ Block Multiplexer CCW specifies either multiple format flags no format flags, or unsupported formats. *NOTE: The BMC and BBC do not report program check when a CCW specifies a data address that is not available, or when a channel program directs the Channel IOP to read a CCW from "not available" storage. In these cases, the BMC and BBC report an MSU interface fault.*
12	Device Status Fault (DSF)	The Device Status Fault bit is set whenever a Channel End or Device End routine is in error. This provides an indication of one of the following conditions. ■ Device End status is received without Channel End or a previous Channel End. ■ Device End only status is received when subchannel is in Active In or Active Out mode. ■ Two subsequent Channel End status presentations are received on split status termination. ■ Channel End only status is received when subchannel is in Idle mode or Active Command Chain mode. ■ Delayed Command Retry is terminated on the second status presentation because Device End only status is not presented.

Table 3-13. Subchannel Status Field of BBC and BMC Status Word (continued)

Bit Position	Designation	Function
		When the DSF bit is set, Channel Fault Word (CFW) written bit (17) is also set and four CFWs are written. The check code is Channel Fault Word 0 identifies the error explicitly.
13	Unsolicited Status	If set, this bit indicates that the subchannel mode is Idle when the device presents status. *NOTE: If the subchannel is in Device Path Selection mode, the associated Device Path Selection table entry is checked for pending operations.*
14	MSU Interface Fault	Indicates that a fault is detected on the MSU/Channel IOP interface as follows: ■ MSU detects multiple uncorrectable errors on the addressed word. ■ A CCW specified a data address that is not available. ■ A channel program directed the Channel IOP to read a CCW from "not available" storage. ■ IOP detects a read data parity error. ■ An internal MSU check occurs during processing of the addressed word.
15	Deferred Condition Code	Indicates that an SIOF previously reported a condition code of zero when it was placed in the stack. A status condition is detected prior to transferring the first byte of data to and/or from the control unit. This is either before or during the retrieval of the SIOF from the stack.
16	Internal Channel IOP Fault	If set, this bit indicates that the operation for the addressed subchannel encountered an internal Channel IOP hardware fault.
17	CFW Written	If set, this bit indicates that four channel fault words are written after the four-word CSW/TSF.

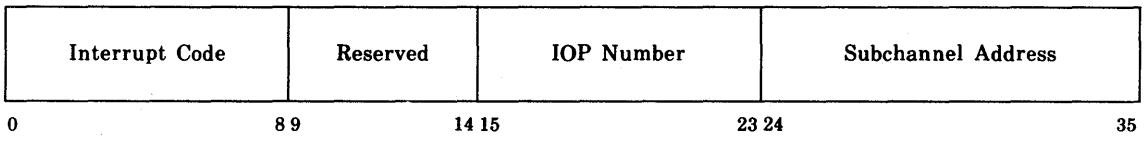
For more information on the format and operation of the Block Multiplexer instruction, see 3.3.3.

3.4.5. DCC Status Words

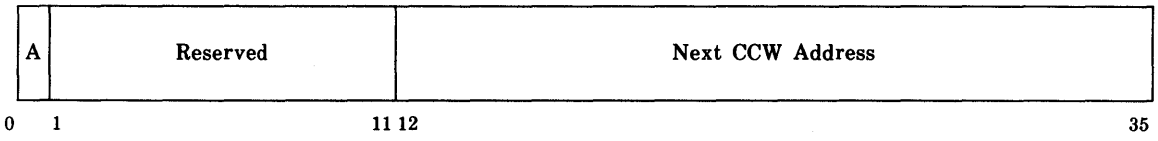
The DCC reports Channel Status Words (CSWs) using the following word format for an ISI word channel:

NOTE: The DCC does not table status words.

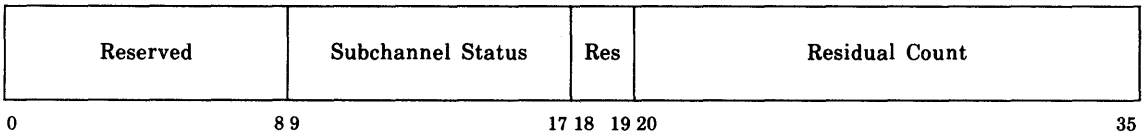
CSW0



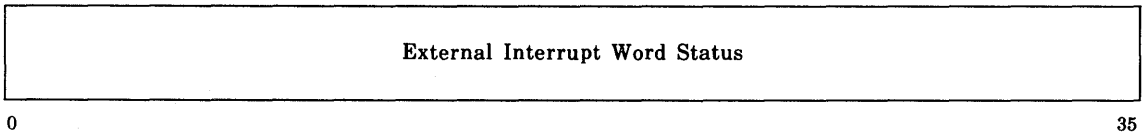
CSW1



CSW2



CSW3



where, for CSW0:

Bits 0–8 = Interrupt Code	If set to one, specifies that a CSW was written in the UPI and an interrupt generated due to termination of an I/O operation.
Bits 9–14 = Reserved	Reserved for software.
Bits 15–23 = IOP Number	Identification of Channel IOP. Designation is UPI number in the range 0–511, of the Channel IOP presenting interrupt.
Bits 24–35 = Subchannel Address	Contains the address in main storage of the subchannel presenting the status information.

where, for CSW1:

Bits 0 = Attention (A)	If set (logical 1), indicates that a valid external interrupt status word is stored as CSW.
Bits 1–11 = Reserved	Reserved for software.
Bits 12–35 = Next CCW Address	This field contains the address of the next CCW. At the time the status information is stored, the operation is to be under control of the new CCW.

where, for CSW2:

Bits 0–8 = Reserved	Reserved for software.
Bits 9–17 = Subchannel Status	See Table 3–14.
Bits 18–19 = Reserved	Reserved for software.
Bits 20–35 = Residual Count	The value of this field is the number of words not transferred to the MSU on input, or to the device on output, with operations under control of the current CCW.

where, for CSW3:

Bits 0–35 = External Interrupt Word Status	This field contains the 36-bit status word returned from the device or its control unit.

Table 3–14. Subchannel Status Field of DCC Status Word

Bit Position	Designation	Function
9	Zero	The value of this bit is zero.
10	Monitor	If set to one, this bit position indicates that the subchannel detects a Monitor condition. The Monitor condition encompasses: ■ Word count exhausted. ■ Chain Command (CC) flag clear. ■ Chain Data (CD) flag clear. ■ Monitor (MON) flag set. ■ No hardware fault detected.
11	Program Check	The Program Check subchannel status bit is set whenever a software error is detected. The indicated error could take any of the following forms: ■ CCW with a TIC command did not specify a CCW address on a double word boundary.

Table 3-14. Subchannel Status Field of DCC Status Word (continued)

Bit Position	Designation	Function
		■ A CCW specified an invalid command. ■ A CCW, with a command other than TIC, specified a word count of zero. ■ Two successive CCWs specified a TIC. *NOTE: The DCC does not report program check when a CCW specifies a data address that is not available, or when a channel program directs the Channel IOP to read a CCW from "not available" storage. In these cases, the DCC reports an MSU interface fault.*
13	Unsolicited Status	If set, this bit indicates that the subchannel mode is Idle when the device presents status.
14	MSU Interface Fault	Indicates that a fault is detected on the MSU/Channel IOP interface. ■ MSU detects multiple uncorrectable errors on the addressed word. ■ IOP detects a read data parity error. ■ Internal MSU check occurred during processing of the addressed word. ■ A CCW specified a data address that is not available. ■ A channel program directed the Channel IOP to read a CCW from "not available" storage.
15	Deferred Condition Code	Order Code SIOF reports a condition code of zero when placed in Channel IOP internal storage. When it is read from the Channel IOP internal storage, a status condition is detected prior to transferring this first word of data to the control unit.
16	Internal Channel IOP Fault	If set, this bit indicates the operation for the addressed subchannel encountered an internal hardware fault.
17	Channel Fault Word (CFW) Written	Indicates that four channel fault words are written after the four-word CSW.

For more information on the format and operation of the Internally Specified Index (ISI) instruction, see 3.3.2.

3.5. Automatic Sense Information Retrieval

The receipt of Unit Check status from a device on a Block Multiplexer Channel causes the Channel IOP to initiate an Automatic Sense operation. This retrieves the device's sense information in Format A and places it in the Sense Information Buffer of the Control Table.

Before initiating the Automatic Sense operation, the Channel IOP generates a Channel Status Word (CSW) with the device's Unit Check status and the subchannel's termination results and places it in the Control Table. This is regardless of the subchannel's status reporting technique (Universal Processor Interface or Status Table). The CSW generated uses the format of the Block Multiplexer Channel status word but sets the Interrupt Code to 4 or 5 (see 3.4.1). When the Automatic Sense operation terminates, an I/O interrupt is presented to software by the Universal Processor Interface.

The length of the Sense Information Buffer must be set at Channel IOP initialization time by the System Support Processor (SSP). When the Channel IOP microprogram is initiated, the Sense Information Buffer length (L) is adjusted to a value of 8 by the microprogram. This is only after determining that the length does not fall in the range greater than 7 or equal to or less than 255 ($7 < L \leq 255$).

The termination of an Automatic Sense operation as the result of an error results in the Channel IOP issuing a selective reset to the device to ensure that the control unit is not left in the contingent connection state.

The IOP parameterizes the device's subchannel Command Word for the Automatic Sense operation as follows:

Parameter	Description
Command Code	Sense command (004_8)
Data Address	Beginning Address of Sense Information Buffer
Word Count	Equal to the Sense Information Buffer length
Flags	The only flag set is Format A
Mode	Active In
Control Bits	Sets the Sense in Progress (SIP) bit.

3.6. Status Tabling

3.6.1. Description

Status information is reviewed in 3.4, Status Reporting. Transfer of data is accomplished more efficiently through the addition of Status Tabling. The table, or "mail box", accumulates peripheral device responses. A group of entries in the Status Table is then transferred to the IP in one exchange operation. An interrupt may be used if a status word has priority.

Both the BBC and BMC, as block/byte channels, provide Status Tabling. The DCC, a word channel interfacing with disk storage devices, does not provide Status Tabling. The host system supports noncommunications Status Tabling of Channel Status Words (CSWs). This is because the system offers a Distributed Communications Processors (DCP) and so does not require a Status Table for communications subchannels.

Status Tabling is a technique used for I/O requests that are not "time critical". The Universal Processor Interface (UPI) is provided for I/O requests that require immediate software response. Tabling of status words allows multiple exchanges to be presented and accumulated before sending a UPI interrupt to the IP.

A priority bit is provided in the CAW to permit software to receive an interrupt on selected table entries. A threshold count is also provided to allow software a means by which to specify the number of table entries that may be made before the I/O channel transmits a UPI interrupt.

After a status table's threshold condition is reported to software, a subsequent threshold condition may not be reported for that table until after the receipt of an Update Status Table (UST) order code (or the table has been stopped and reactivated). When a UST order code is executed, the number of processed entries is added to the number of available entries and subtracted from the number of accumulated entries for the selected table. If the accumulated number of entries is equal to or greater than the table's threshold count the Channel IOP notifies software of the table's new threshold condition. If the accumulated entry count is less than the table's threshold count, no threshold condition exists.

Any status table condition, such as full, threshold, priority (SIOF order code), or fault, which occurs as the result of the current table entry or entries, is always reported after the table entry or entries have been made. Two entries are used when reporting a fault with the status presentation.

3.6.2. Operation

Selection of the status table to be used by each subchannel is accomplished by the System Support Processor (SSP) during system initialization. Status tabling is enabled/disabled, on a subchannel basis, by system software through the Select Status Tabling order code, or by the SSP during system initialization.

Initiation, control, and stopping of the Status Table is accomplished on a subchannel basis by the three Order Codes provided. These codes are Activate Status Tabling, Update Status Table, and Stop Status Table.

The Activate Status Table (AST) order code initiates a Status Table by specifying the table location, length, and threshold count. The Update Status Table (UST) order code is used to inform the IOP of the number of table locations processed by software and to change the threshold count. The Stop Status Table (SST) order code halts a status table.

Following a system initialization, all noncommunications status is presented to software via the UPI interrupt mechanism until the noncommunications status table has been activated. Once the Status Table is enabled, I/O requests are recorded as entries in the table. If a status table is halted for any reason, or is full, all status that would normally be presented to that status table is stacked in the control unit memory for the device until the status table is reactivated.

The noncommunications status tabling stops as the result of an MSU Interface Fault during entry.

Tabling for noncommunications status stops as the result of any one or the following errors reported by a CSW.

- Program Check on the Status Table
  - IOP attempted to make a table entry in not available storage.

- Internal IOP Fault on the Status Table
  - IOP detected hardware fault.
- Table Full condition
  - When Status Tabling is stopped as the result of an error, the CSW reporting the condition has the Table Fault bit set, indicating the tabling has been stopped.

3.7. Data Word Formats

There are three word formats for unpacking/packing data from or into the 36-bit word of the Instruction Processor (IP). These conversion techniques, designated Format A, B, and C are:

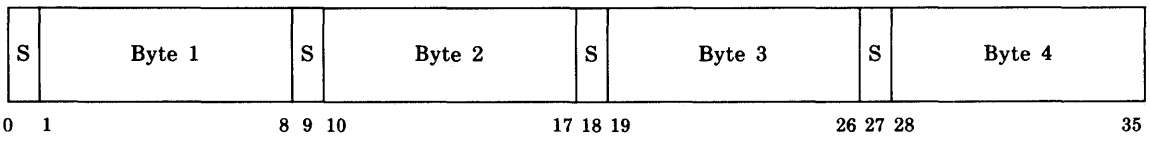
- Format A: Enter four 8-bit bytes in 36-bit word; leave other four bits blank.
- Format B: Enter six 6-bit characters in 36-bit word. This configuration is not supported by the host system.
- Format C: Enter four and one-half 8-bit bytes in 36-bit word; remaining half byte carried to next word with four full bytes.

The Channel IOPs do not support Format B data manipulation. The Byte Bus Channel (BBC) and Block Multiplexer Channel (BMC), being block/byte oriented, both support Formats A and C in input as well as output. A more detailed description, together with word formats, follows.

- An output transfer (write) consists of unpacking data from a 36-bit word to 8-bit bytes. This is accomplished in the following manner.

  Format A: Four 8-bit bytes in one 36-bit word.

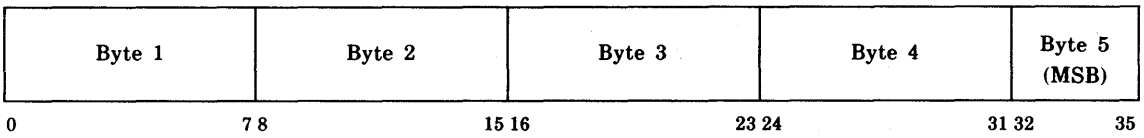
  This data manipulation technique specifies the quarter-word format for unpacking bytes from words. One byte is right justified in each 9-bit quarter word. If the left most bit is set in any quarter word during an output operation, the data transfer is terminated.

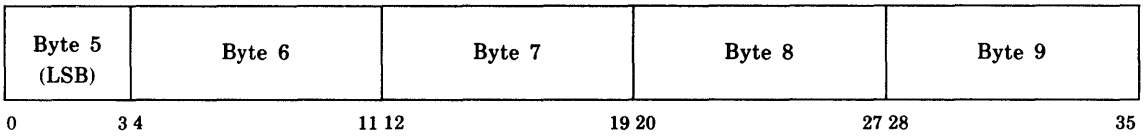


NOTE: If any Stop (S) bit is set, the byte following the Stop bit does not transfer and the operation is terminated. It must be assured that the Stop bit being set before the first byte works properly.

Format C: Nine bytes in two 36-bit words.

This data manipulation technique specifies 8-bit packed format for unpacking bytes from words. Four and one-half bytes are packed in each 36-bit word.





MSB = Most Significant Bits

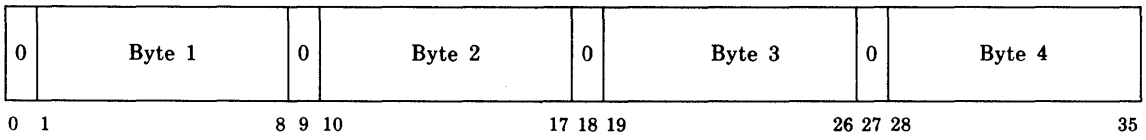
LSB = Least Significant Bits

If a CCW word count specifies an odd number of words, the byte being sent to the device contains bits 32–35 of the last word and four bits of undetermined data. The Channel IOP does not read the next word from the MSU.

- An input transfer (read forward) consists of packing data into a 36-bit word from 8-bit bytes. This is accomplished in the following manner:

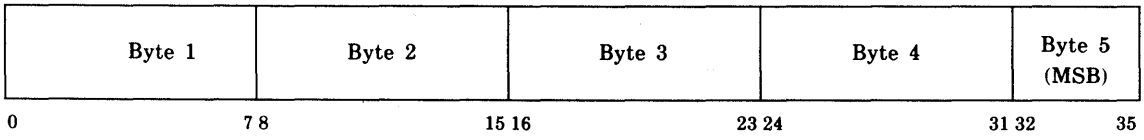
Format A: Four 8-bit bytes in one 36-bit word.

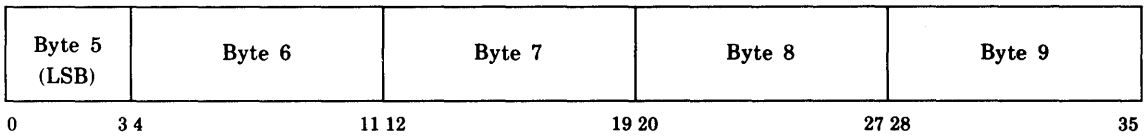
This data manipulation technique specifies the quarter-word format for packing bytes into words. One byte is right justified in each 9-bit quarter word.



Format C: Nine bytes in two 36-bit words.

This data manipulation technique specifies 8-bit packed format for packing bytes into words. Four and one-half bytes are packed in each 36-bit word.





MSB = Most Significant Bits

LSB = Least Significant Bits

If the CCW word count and the device record size are equal and end on an odd word boundary, the LSB of the last byte of data transferred from the device are discarded and not written into the MSU.

If the CCW word count is greater than the device record size, all data received from the device are transferred to the MSU with all unused portion of the word being zero filled.

All transfers to the MSU are full-word transfers regardless of the number of bytes in the word or the data format specified.

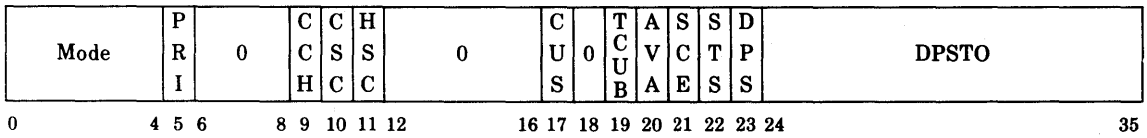
For a read backward operation, the bytes are packed into the 36-bit word in reverse direction. All termination cases remain the same.

3.8. Channel Descriptor Table

Each Channel IOP is uniquely identified by a Channel Descriptor Table. The 12-bit subchannel address issued by the software during instruction initiation specifies an index into a Channel Descriptor Table.

The format of the Channel Descriptor Table is common for the DCC, BMC, and BBC. The Channel Descriptor Table is a 246 word table beginning at address 37300_8 in all S-bus Channel IOP control stores. Not every field is used by every S-Bus Channel IOP. After the initial control store load, each entry is zero except for the Available bit (20), which is a 1. Subsequent to initial load the table may be loaded or read by software using the Write Channel Descriptor Table or Read Channel Descriptor Table order codes, respectively. The following is an illustration of a single entry in the table and a description of each field in the entry.

Channel Descriptor Table Entry



Mode Is the subchannel mode. The least significant four bits determine the modes defined for the DCC, BBC, IP, and BMC. The fifth bit (bit 0) is used to create unique modes from the defined modes for microprogram use. The modes are defined as:

000 - Idle
001 - Terminate Out

002 - Command Retry
003 - Active Command Chain
004 - Status Pending
005 - SIOF Pending
006 - Active Out
007 - Active In Forward
010 - EI Pending
011 - Terminate In
012 Undefined
013 Undefined
014 - Undefined
015 - Control Unit Busy
016 - Mode EF (DCC only)
017 - Reserved for device path selection operations
020 - Channel Status Pending (BMC only)
021 - Undefined
022 - Automatic sense status pending (BMC only)
Indicates that: (1) Sense command status for automatic sense has been stacked in the control unit, or (2) the Unit Check CSW for automatic sense has been written, but the sense command to the device has not yet been issued.

PRI PRiority status bit from CAW0 for this subchannel. Only valid if subchannel has been activated by SIOF. (BMC and BBC only.)

CCH Indicates that a Clear Channel order code has been executed.

CSC Indicates that a Clear Subchannel order code has been executed.

HSC Indicates that a Halt Subchannel order code has been executed.

CUS Control Unit Sequence in progress. Used to indicate that the continuation of a channel program was initiated by a control unit on the interface or that device status was presented by a control unit initiated sequence. (BMC only.)

TCUB Transparent Control Unit Busy. Indicates that the channel should not report Control Unit Busy to the IP but wait until the control unit is not busy to initiate the SIOF for this subchannel. (BMC only.)

AVA This is the Available bit. It designates which subchannels are available for this Channel IOP. It is set by the microcode when a reset clear is received. The BMC and BBC will have this bit set in all 256 Channel Descriptor Table entries. The DCC will have this bit set in only the first Channel Descriptor Table entry (DCC supports only subchannel 0). Thus, the DCC rejects all subchannel based order codes (including EDSC) for subchannels 1–256 with a condition code of 3. (Subchannel Not Available.)

SCE This is the Subchannel Enable bit. It is manipulated by the EDSC order code and initialized by the SSP.

STS This is the Status Table Select bit. If set it indicates that status tabling is enabled. It is altered by the Activate Status Table. This bit is ignored by the DCC. (BMC and BBC only.)

DPS This is the Device Path Selection bit. It is only used by the BMC to enable device path selection.

DPSTO This is the Device Path Selection Table Offset value. It is used only by the BMC to determine where to look in the Device Path Selection Table for the channel program (CCW list for this subchannel).

Channel Type and Channel Number (UPI number) - These values are the same for every subchannel in a given Channel IOP and therefore are not kept in the Channel Descriptor Table.

3.9. Disk Controller Channel

The Disk Controller Channel (DCC) is a microprogrammable logic unit that combines the functional capability of a word channel and a disk controller in the same hardware. This functional capability is a product of the microcode.

3.9.1. Functional Description

The DCC hardware consists of four distinct elements:

- Processor element
- Data buffer
- S-bus sequencer
- Disk sequencer

The data buffer, S-bus sequencer, and disk sequencer are independent sections of hardware that operate under control of the processor element. Figure 3-1 shows this relationship.

The processor element uses the other elements to accomplish I/O requests between disk units and main storage.

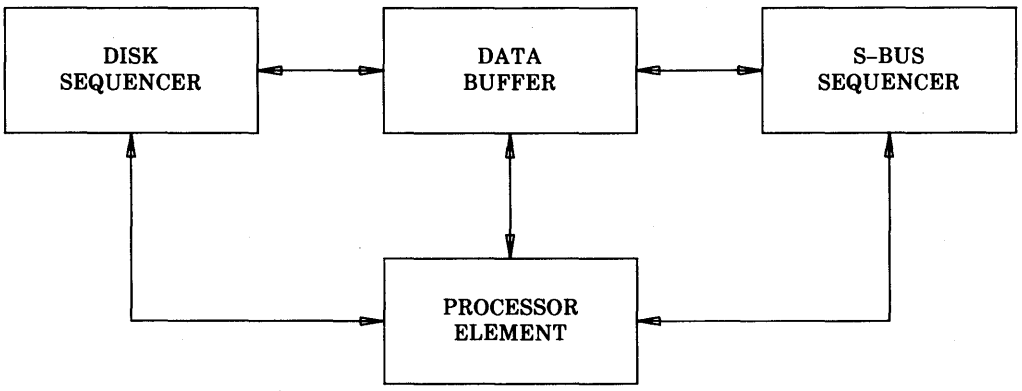


Figure 3-1. Hardware Elements

3.9.1.1. Processor Element

The processor element consists of a Random Access Memory (RAM) and an Arithmetic Logic Unit (ALU)/Shifter/Priority Encoder, eight general purpose registers, instruction addressing logic, operand addressing logic, and interval timer, and a single-level interrupt. One of the general purpose registers can be used as an index register during operand addressing sequences, and another can be used as a repeat counter for the repeat microinstruction. The RAM contains both instructions and data. The internal data paths through the ALU/Shifter/Priority Encoder are 14 bits (12 data bits plus 2 parity bits) wide except on control store operand calculations. On those operations, the ALU operates on 14 bits of data.

The processor element executes channel programs by reading control words, initializing and monitoring the data path during data transfer, formulating and writing status words, and implementing retry and recovery procedures.

3.9.1.2. Data Buffer

The data buffer consists of a RAM and programmable addressable logic. The RAM is 42 bits (36 bits of data plus 6 parity bits) wide and 4096 words deep. The data buffer is used as a First-In, First-Out (FIFO) memory device for data transfers between the disk and main storage, disk IDentification (ID) count field and command parameters between the disk and the data buffer, and back-to-back transfers between the data buffer and main storage.

The data buffer also contains limit registers that allow the programmer to configure up to 4096 logically independent FIFO buffers with complete wrap-around, and overflow and underflow detection features.

3.9.1.3. S-Bus Sequencer

The S-bus sequencer is used by the processor element to communicate through the S-bus to other system components such as main storage, Instruction Processor, and System Support Processor. The S-bus sequencer transfers information between these components and the data buffer and other S-bus sequencing registers. The S-bus sequencer is controlled by registers and flip-flops connected to the processor element data paths and provides feedback information by registers and testable variables connected to the processor's logic.

3.9.1.4. Disk Sequencer

The disk sequencer is used by the processor element to communicate with the disk string controllers and the disks. The processor element can directly communicate with the peripherals by entering commands in interface registers connected to its output data paths and testing status from interface registers on its input data paths. The processor element also controls the transfer of data between the disks and the data buffer by registers and directly testable variables. The interface to the disk string controller is a two-byte-wide bidirectional interface, called the Standard Control Interface (SCI). The disk sequencer contains hardware to pack these bytes into 36-bit words on input and extract them from 36-bit words on output.

3.9.2. DCC Characteristics

3.9.2.1. Logical Characteristics

The microcode in the DCC provides the logical characteristics of an ISI word channel connected to a 5056 Disk Control Unit. See 3.3.2, for a description of the ISI word channel routine and format.

The logical relationship of the channel components is shown in Figure 3-2.

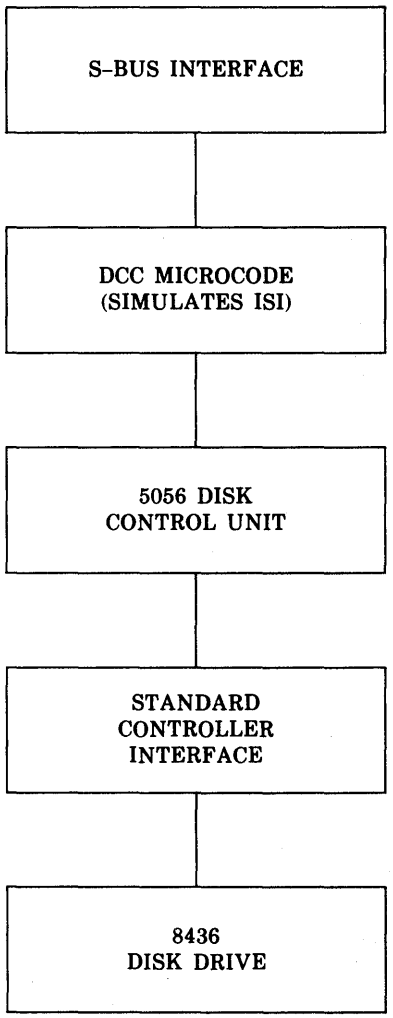


Figure 3-2. Logical Relationship of the Disk Controller Channel (DCC)

The DCC is driven by 5056/ISI software. It appears to the program as a word channel with one shared subchannel. The address of this subchannel is 0000_8. This is a permanent subchannel assignment, fixed by convention with the operating software.

3.9.2.2. Controller Characteristics

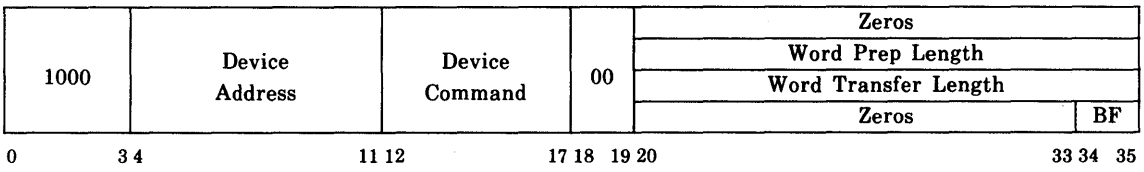
The DCC functions as a disk controller as well as a word channel. This means that the DCC decodes and executes external function words rather than pass them on to the control unit, as a word channel would do.

3.9.3. Disk Commands

When the command code field in CCW0 specifies a Forced External Function (FEF), the DCC reads a one or two word External Function (EF) from main storage starting at the address specified in CCW0. The EF word or words specify the disk to be selected and the operation to be performed on that disk. The EF words take the following general form.

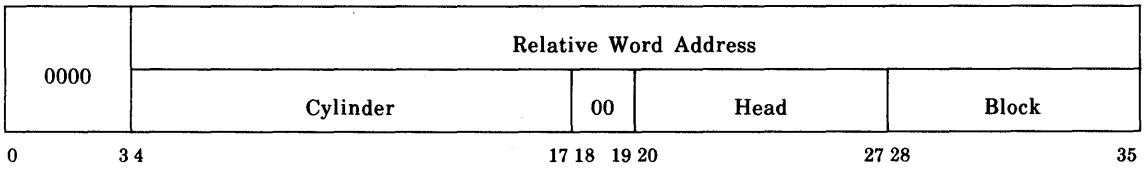
Command Function Word

EF0



Address Function Word

EF1



where, for EF0:

Bits 0–3 = 1000	This value indicates a command function word.
Bits 4–11 = Device Address	Designations of device or peripheral (disk unit).
Bits 12–17 = Device Command	Instruction selected from DCC repertoire.
Bits 18–19 = Zero	The value of this field is zero. It is not used.
Bits 20–35(A) = Zero	The application requires this field to have a value of zero.
Bits 20–35(B) = Word Prep Length	"Prep track" formats ID and data fields for a full track starting with block zero through the maximum number of blocks allowable for the specified word prep length.

NOTE: For the DCC, the only valid number in the prep word length field is 112. Any other number causes the Unit Check bit to be set in the status word.

The control unit rejects with Unit Check (invalid track format) any attempt to prep an alternate track where the data length of block zero disagrees with the specified prep length.

Bits 20–35(C) = Word Transfer Length — Consecutive data records are transferred, starting at the word whose address is specified in the address function word. The exchange continues until the block containing the last word specified by the word transfer length subfield has been processed.

Bits 20–33(D) = Zero — The application requires this field to have a value of zero.

Bits 34–35(D) = Backing Factor (BF) — The value of this field provides compensation for varying reconnect and buffer prefill delays. When the control unit receives this command, it interrogates this field and advances the presentation of Device End by an amount determined by the coding in the BF field.

where, for EF1:

Bits 0–3 = 0000 — This value indicates an address function word.

Bits 4–35 = Relative Word Address (A) — Location in Main storage of EF word.

Bits 4–17 = Cylinder (B) — Specifies location in disk drive of desired information.

Bits 18–19 = Zero (B) — The value of this field is zero. It is not used.

Bits 20–27 = Head (B) — Specifies position of Head to retrieve desired information.

Bits 28–35 = Block (B) — Specifies physical block length - 52 on disk surface.

The command set for the 36-bit DCC is outlined in Table 3–15. Only those command and address fields whose content changes with different routines are listed. See the EF word formats previously presented for alphabetical designations.

Table 3-15. Command Repertoire for 36-Bit DCC

Category	Command Name	Command Word (Bits 20–25)	Address Word (Bits 4–35)	Command Code
Control	Recalibrate	A	Unused	13_{16}
	Sector Position Relative	D	A	$2B_{16}$
	Track Position Relative	A	A	$0B_{16}$
	Untagged Track Position Relative	A	A	$1B_{16}$
	Untagged Sector Position Relative	D	A	$3B_{16}$
Write	Prep Track	A	B	01_{16}
	Write Data	C	A	$0D_{16}$
Read	Read Initial Program Load	A	Unused	00_{16}
	Read Full Track	C	B	16_{16}
	Read Count	A	Unused	12_{16}
	Read Data	C	A	$0E_{16}$
	Read Data Special	C	A	$3E_{16}$
	Read Label	A	Unused	32_{16}
Sense	Sense Input/Output	A	Unused	04_{16}
	Read Buffered Record	A	Unused	14_{16}
	Read Track Identification	A	B	36_{16}
	Write Identification	*	B	15_{16}
Recovery	Assign Alternate Block	A	A	35_{16}

* *Displacement code contained in bits 34 and 35*

If any other command codes are entered in the Device Command field, the Unit Check status bit is set (logical 1) in the status word.

NOTE: A detailed description of the disk commands is given in the 8436 Disk Subsystem Reference, UP-10058 (current version).

3.9.4. Disk Status

The Channel IOP configures a 5056 Disk Control Unit with the 8436 Disk Drive. The status word presented by the disk control unit is compatible with the External Interrupt word reported by the DCC.

3.9.4.1. DCC Status Word

As described in 3.4, Status Reporting, the DCC reports outcome of an I/O operation by writing a four-word Channel Status Word (CSW) in main storage. The fourth word (CSW3) is the External Interrupt (EI) status word. This word contains information describing the outcome of the I/O operation at the device level.

3.9.4.2. Disk Control Unit Status Word

The disk control unit controls and reports on data transfer to its associated disk drive.

- End Data Transfer

  The disk control unit determines when to stop requesting data transfer and presents a status word to indicate termination of the associated command. This occurs when the disk control unit has decremented a transfer count to zero, when an error condition is detected, or when channel truncation is detected. Channel truncation is detected by the disk control unit when the channel has not responded to a word transfer request within 18 milliseconds or an FEF is detected.

- Status Indication

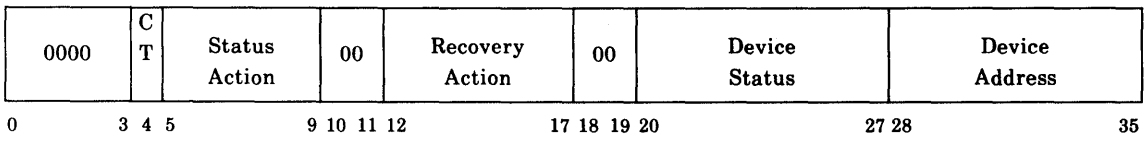
  The disk control unit presents a status word for every command. Transfer of a status word always implies that the channel interface is available for initiation of another command. The status word is generated by the disk control unit micro-program and sent to the channel together with an EI signal. A secondary status word is usually presented upon completion of a device positioning operation.

An associated status word may be presented upon detection of a pack change condition. The disk control unit has a five second time out on all status word transfers.

Normal command completion is indicated by a status word containing Channel End with Device End status. Command termination with error is usually indicated by unit check status alone. A status word containing Channel End status alone indicates that the operation is being continued independently and that a secondary status will be presented later for the associated device. A status word containing Busy Status alone indicates that the device is busy and that a secondary status word will be presented when the associated device is available.

3.9.4.3. Channel Status Word for DCC

The fourth CSW (CSW3) of a DCC I/O operation is the EI. The format of the EI instruction with bit designations is shown below:



A detailed description of the status word format is given in the *8436 Disk Subsystem, Reference,* UP-10058 (current version).

3.9.5. Disk Sense Information

Disk sense information comes in the form of 24 bytes packed in six 36-bit words using Format A. This sense information is valid whenever the Unit Check bit is set in the device status field in the EI word.

There are nine sense data formats. The first eight bytes (bytes 0–7) of all nine formats are created by the DCC and have a common form. A detailed description of the sense data formats are given in the *8436 Disk Subsystem, Reference,* UP-10058 (current version).

3.10. Byte Bus Channel

The Byte Bus Channel (BBC) is a microprogrammable logic unit that offers compatible block multiplexer mode. This functional capability is a product of microcode.

3.10.1. Functional Description

The BBC hardware consists of three distinct elements:

- Processor element
- Device sequencer
- S-Bus sequencer

The processor element and the S-bus sequencer are identical to the to the DCC and are described in 3.9.1.1 through 3.9.1.3, respectively.

The device sequencer is unique to the BBC and is described below.

The BBC device sequencer contains a 1024-word, 14-bit (12 bits of data and 2 bits of parity) RAM. Each of 16 possible line modules has a 16-word block in the RAM. The 16-word block contains information necessary to execute command, status, and data sequences. A second RAM is used to buffer data between the L-bus peripherals and main storage. This RAM has 1024 words by 42 bits (36 bits of data and 6 bits of parity). Each of the 16 possible line modules has a 32-word data buffer in this RAM.

The device sequencer packs bytes into 36-bit words on input and unpacks bytes from 36-bit words on output in data formats A and C (see 3.7). The device sequencer also provides L-bus interface parity checks where possible.

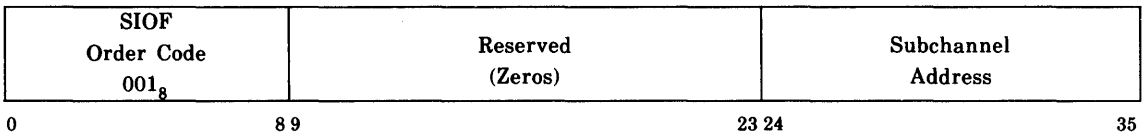
3.10.2. Logical Description

Being byte oriented, the BBC is compatible with the block multiplexer mode of operation. See 3.3.3 for a description of the Block Multiplexer routine and format. The information that follows describes implementation details related to programming of the BBC.

3.10.2.1. Subchannel Addressing

The BBC subchannel address is specified in Channel Address Word (CAW) 0. The least significant 12 bits are allotted for address designation. Field arrangement of the word for a typical configuration is shown in the following format:

CAW0 Channel Address Word (CAW)



In typical usage, two Channel Address Words are used, CAW0 and CAW1. For more information, see 3.2.4. For simplicity only CAW0 is shown, since it contains the field assigned for subchannel address.

The 12-bit address is decoded as shown in Table 3-16. The binary representation is converted to its 8-bit byte equivalent in decimal, hexadecimal, and octal.

Table 3-16. Byte Bus Channel Address Decode

12-Bit Subchannel Address												8-Bit Device Address		
00	01	02	03	04	05	06	07	08	09	10	11	Octal	Hex	Decimal
0	0	0	0	0	0	0	0	0	0	0	0	000	00	0
0	0	0	0	0	0	0	0	0	0	0	1	001	01	1
0	0	0	0	0	0	0	0	0	0	1	0	002	02	2
0	0	0	0	0	0	0	0	0	0	1	1	003	03	3
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
0	0	0	0	1	1	1	1	1	1	0	1	375	FD	253
0	0	0	0	1	1	1	1	1	1	1	0	376	FE	254
0	0	0	0	1	1	1	1	1	1	1	1	377	FF	255
X	X	X	1	X	X	X	X	X	X	X	X		Not Available	
X	X	1	X	X	X	X	X	X	X	X	X		Not Available	
X	1	X	X	X	X	X	X	X	X	X	X		Not Available	
1	X	X	X	X	X	X	X	X	X	X	X		Not Available	

NOTE: Bits 0-3 are always zero by convention established with the operating software.

3.10.2.2. General Operations

Byte Bus Channel operation consists of four main activities. These activities are:

- Input/Output order code execution,
- Channel program initiation,
- Data transfer, and
- Status reporting.

All 256 subchannels may be concurrently active and involved in any of the four operations. The significant part of I/O order code implementation is the execution of the Start I/O Fast (SIOF)

order. This order results in the BBC getting a job from the operating software and placing it in a "stack". This stack is called an SIOF stack. The BBC maintains 16 SIOF stacks, one for each potential attached line module.

The SIOF stacks are First-In, First-Out (FIFO) stacks. The depth of each stack is 16. If a line module has no subchannels currently transferring data or waiting to report status, and has no outstanding interrupts, the BBC checks its SIOF Stack. If that SIOF Stack is not empty, the BBC removes a job from the top of the stack and initiates that channel program.

After the channel program has been initiated, a line module may begin transferring data. The BBC multiplexes bytes of data between the L-bus and the data buffer. Line module 0 has the highest priority on the L-bus, and line module 15 has the lowest. When the data buffer in the BBC for a particular line module contains more than 8 words of data on input or fewer than 24 words on output, an interrupt is generated to the BBC micro-controller. This interrupt indicates that a line module requires the S-bus interface to the MSU to transfer data. When a line module gains access to the S-bus eight words are transferred between the MSU and the BBC Data Buffer. The BBC multiplexes 8-word blocks of data from all line modules through the S-bus. Line Module 0 has the highest priority for data transfer at the S-bus.

The channel program continues executing (transferring data) until:

- It is suspended due to the receipt of Device End status from the line module.
- It is terminated because of a BBC or line module error.
- It reaches normal termination.

When a channel program terminates, the status of that program is reported in a status table or the control table in main storage. If the status reporting mechanism is not available, the BBC does not stack status in the line modules. Instead, the BBC maintains a status queue. This queue is a FIFO stack of depth 16. It is possible to make one entry in the stack per line module. If a line module has an entry in the status queue for an associated subchannel, then no new jobs will be initiated on that line module. When the status reporting mechanism becomes available, the status queue is emptied on a FIFO basis.

Whenever a channel program is suspended or terminates and status has been reported, the BBC enables asynchronous status at the line module. If the line module has no outstanding interrupts, the BBC checks the line module's SIOF stack for new entries. If the SIOF stack is empty, asynchronous status remains enabled until an interrupt occurs or the BBC gets an SIOF for a subchannel associated with the line module.

3.11. Block Multiplexer Channel

The Block Multiplexer Channel (BMC) is a microprogrammable logic unit that offers compatible block multiplexer mode. This functional capability is a product of microcode.

3.11.1. Functional Description

The BMC hardware consists of four distinct elements:

- Processor element,
- Device sequencer,
- S-Bus sequencer, and

- Data buffer.

The processor element, data buffer, and the S-bus sequencer are identical to the DCC and are described in 3.9.1.1, 3.9.1.2, and 3.9.1.3, respectively. The device sequencer is unique to the BMC and is described below.

The BMC device sequencer is used by the processor element to communicate with peripheral control units. The processor element communicates directly with control units by putting commands in the Bus Out register, manipulating interface control lines, and reading the response from the Bus In register. The processor element also controls data transfer between control units and the data buffer using parameters passed in internal registers, flip-flops set by microinstructions, and directly testable hardware variables.

The device sequencer does the byte-to-word and word-to-byte data packing and unpacking as described in 3.7.

3.11.2. Logical Description

This Channel IOP, operating in the block multiplexer mode, is compatible with byte oriented devices. See 3.3.3 for a description of the Block Multiplexer routine and format. See 3.10.2.1 for subchannel addressing.

4. Main Storage Unit

4.1. General

The Main Storage Unit (MSU) is a single-port storage unit capable of operating on a synchronous bussed interface. The MSU provides double error detection and single error correction on read data internal to storage. Byte parity is checked and provided on interface data. The MSU performs writes, reads, partial writes, and an internal refresh that the user sees as destination busy. The MSU is controlled by function code.

4.2. Functional Characteristics

Performance characteristics for main storage are specified in Table 4-1. All times are measured at the MSU interface from address valid time on the system bus and are based on a 108 nanosecond clock rate. Read access parameters have bus priority. Read access is the time interval between valid read address out of the bus receiver to valid input data to the bus driver.

Table 4-1. Main Storage Performance Characteristics

Parameters	Performance
Single-word read access without correction	324 nanoseconds
Single-word read access with correction	432 nanoseconds
Read/write cycle time	432 nanoseconds
Partial write cycle	648 nanoseconds
Double-word read access to first word	648 nanoseconds
Double-word read cycle	756 nanoseconds
Double-word write cycle	756 nanoseconds
Refresh Cycle	324 nanoseconds
Refresh Rate	25 microseconds

4.3. System Bus Lines for Information Flow

The MSU operates on the System bus (S-bus). The S-bus provides the communications path and protocol required to control the flow of information between units of the system (see Figure 4-1).

The S-bus to MSU interface has 85 bidirectional lines. Even parity is used where parity lines are provided.

4.3.1. Request Lines (23 Lines)

The MSU uses these lines to determine priority in the next bus cycle.

4.3.2. Priority Disable (1 Line)

If this bidirectional line is active during a bus unit time, the requester that has access to the bus maintains access during the following bus unit time. The MSU uses this line for block-read transfers and for read operations within an enabled 64K SBC Boundary.

4.3.3. Request Inhibit (1 Line)

When this line is activated by an IP or Channel IOP, new requests from other IPs or Channel IOPs are inhibited. Requests already initiated are serviced according to normal priority assignments. The MSU does not use this line.

4.3.4. Source Address

The source address is a bidirectional 9-bit field including even parity that identifies the transmitting unit.

4.3.5. Destination Address

The destination address is a bidirectional 9-bit field including even parity that identifies the receiving unit and determines the address mode.

4.3.6. Information Lines

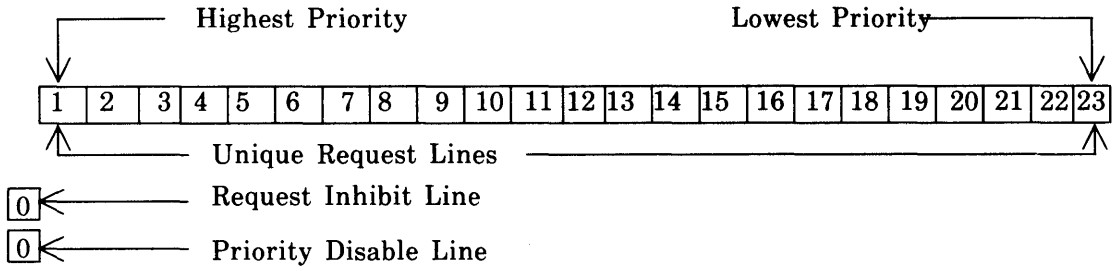
The information lines are made up of a bidirectional 42-bit field including even parity on 6-bit data boundaries for transmitting and receiving 36 information bits.

4.3.7. Format Lines

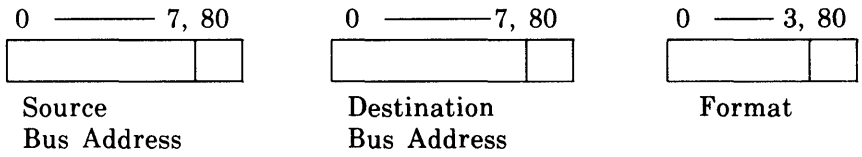
The format lines are made up of a bidirectional 5-bit field including even parity that identifies the format of the information lines. The information formats are: data, message, error report, and nonspecific. Bit 3 of the format field indicates a transfer extension follows during the next bus unit time.

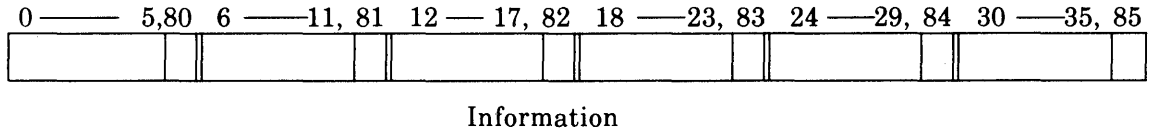
- Data format 1000 and data extended format 1001 - The MSU receives information to be stored in the MSU by the data formats. The MSU transmits using the data formats.

Bus Access (25 lines)

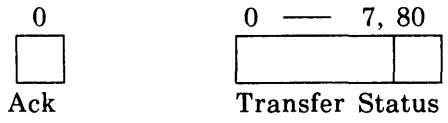


Transfer

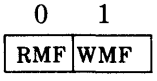




Bus Acknowledge (10 lines)



Memory Fault (2 lines)



Bits 80–85 = Even Parity

NOTE: RMF - Read Memory Fault, WMF - Write Memory Fault

Figure 4–1. System Bus Lines For Information Flow

- Nonspecific format 1110 and nonspecific extended format 1111 – The nonspecific format is associated with broadcast address mode requests only. The MSU receives a nonspecific instruction set (see Figure 4-2) via the nonspecific format.

- Message format 1010 and message extended format 1011 – The message formats are associated with direct address mode requests only. The MSU receives the message instruction set (see Figure 4-3) via the message format.

- Error report format 1100 or 1101 – The error report format is not recognized or used by the MSU.

4.3.8. Acknowledge and Bus Status

A bus scan sequence is completed every bus unit time (108 nanoseconds) by the MSU. A bus scan sequence captures the contents of the source, destination, format, and information fields. The MSU responds with an ACKNOWLEDGE signal and a bus status for the following conditions:

- In the direct addressing mode for destination addresses without parity errors that compare to the MSU drop address.

- Broadcast within the MSU address range.

- The extended transfer of an acknowledged-broadcasted request is unconditionally acknowledged in the broadcast addressing mode, when the address transmitted by the requester is within the address range of the MSU and without parity errors:

  - in address bits 12–23,
  - the format field, and
  - the destination address.

  A data format is assumed by the MSU.

The bus status reports the status of the acknowledged transfer as follows:

Bus Status Bit	Status
0	Destination busy
1	Parity Error in Transfer
2	Parity Error in Source Address
3	Parity Error in Format
4	Parity Error Information bits 00–11
5	Parity Error Information bits 12–23
6	Parity Error Information bits 24–35
7	Reserved
80	Even Parity

4.3.9. Bus Priority

All units on the S-bus, including the MSU, contain priority resolution circuitry. Each unit on the S-bus has a request line for priority on the S-bus.

The relative priority of each request line is established by five switches internal to each unit. Of the 32 binary combinations of the switches, 23 are valid in the system. Switch combination 00001 is reserved for the System Support Processor (SSP), and the remaining are associated with

MSU, Channel IOP, and IP units on the S-bus. Switch combination 00001 is highest priority, with relative priority descending in order.

When requesting priority, the MSU raises its request line one bus cycle prior to data transmission. This disables all requests of lower priority. The MSU uses its priority determination circuitry to detect any requests of higher priority than that of the MSU as follows:

- If none exists, the MSU transmits on the next bus cycle.
- If a higher priority request or a priority disable exists, the MSU tries for priority the next cycle and repeats this until priority is granted.

A priority disable line retains system bus priority during read block transfers and during the transfer of corrected data.

4.3.10. Information Transfers

Information transfers are handled by the MSU as follows:

1. The MSU tries for priority of the cycle prior to data transmission by raising its request line. If no higher priority request exists, the MSU transmits on the next S-bus cycle. If a higher priority request exists or a priority disable, the MSU continues to try for priority until it is granted.

2. The requester's source address is returned as the destination address with the destination address bit 0 cleared (directed instruction).

3. The MSU drop address is returned as the source address.

4. The format is either data 1000 or data extended 1001. For extended transfers, the MSU raises the PRIORITY DISABLE line in the bus cycle in which it was first granted priority. This maintains S-bus priority for the MSU. The PRIORITY DISABLE line is deactivated for the last extended data transfer.

5. If a read request is to a disabled 64K Single Bit error Correction (SBC) boundary, the MSU transmits uncorrected read data to the requester the first bus cycle that the MSU gains priority.

   If a read request is to an enabled 64K SBC boundary, the MSU transmits uncorrected read data to an all zeroes destination address in the first bus cycle that the MSU has priority, using the PRIORITY DISABLE line. In the S-bus cycle, that follows, the MSU transmits corrected read data to the requester.

6. The MSU ignores the acknowledge and bus status associated with the transferred data.

4.3.11. Read Memory Fault (1 line)

The Read Memory Fault (RMF) signal is returned one clock cycle after the last requested MSU information word is valid on the system bus.

4.3.12. Write Memory Fault (1 line)

The MSU activates this line for errors occurring in cycles where the MSU does not transmit data.

4.3.13. Power Up Clear (1 line)

The POWER UP CLEAR signal is an S-bus signal which when active:

- clears all MSU registers,
- sets the alternate data in the Scan/Set registers,
- disables the system bus Interface line, and
- disables 64K SBC boundaries, (single bit correction disabled).

The MSU must be initialized after the POWER UP CLEAR signal.

4.3.14. Cycle Step (1 line)

The cycle step line when active conditions the MSU to operate in a cycle step mode.

4.3.15. Scan Set (5 lines)

The MSU scan set registers and header register are controlled by the Scan/Set lines.

4.4. Addressing Modes

Main storage is addressed through the 8-bit destination address, address bits 12-17 in the broadcast mode, and the 8-bit destination address in the direct address mode as determined by bit 0 of the destination address.

4.4.1. Direct Address Mode (Message)

In the direct address mode:

- the destination address bit 0 is cleared,
- bit 1-3 (010 specifies an MSU),
- bits 5-7 specify a particular MSU, and
- bit 4 is always zero.

Destination bits 0-3 equals 0010, respectively. All MSUs on the bus compare the destination bits to their own 4-7 MSU Drop Address switches, and only the MSU with an address match responds. In the direct address mode, the information lines of the bus transfer are interpreted using the message format independent of the format lines. If an extended transfer occurs, the information lines of the extended transfer or transfers are interpreted by the MSU using the data format, independent of the format lines.

4.4.2. Broadcast Mode (Nonspecific)

In the broadcast mode:

- destination address bit 0 is set,
- bits 1-3 identify MSU (010),
- bits 5-7 are cleared,
- requester uses the nonspecific format (initial transfer), and
- bit 4 is always zero.

All MSUs on the bus test address bits 12-16 determine if the address is within the MSU's address range. Only the "in range" MSU responds. The range of the MSU is defined by the MSU starting address which is the contents of the MSU's Base Address register and the MSU's maximum address which is the MSU capacity bits added to the Base Address register. If an extended transfer nonspecific format is used on the extended transfer or transfers, the MSU interprets the information lines as a data format independent of the format lines.

4.5. Storage Instructions (Nonspecific Format)

The nonspecific instruction set of the MSU is associated with broadcasted requests. The information field of an acknowledged broadcasted request is interpreted in Figure 4-2. It is the requester's responsibility to use only the nonspecific instructions of Figure 4-2. All undefined nonspecific instructions have unknown results in the MSU.

4.5.1. Read

Read with/without Single Bit Error Correction (SBC)

- Read without SBC - For read requests with an address within a disabled 64K SBC boundary, the MSU returns uncorrected read data with correct system bus parity. A READ MEMORY FAULT signal is transmitted if the MSU detects an error.

- Read with SBC - For read requests with an address within an enabled 64K SBC boundary, the MSU returns single bit error corrected data with correct S-bus parity.

4.5.2. Write Cycle

The 36 information bits are stored in the specified storage address with generated check bits. Errors detected during the MSU write cycle inhibits the write cycle.

Information Lines

0	1–3	4–11	12–35
unlock	Function Code	Write Control	MSU Address

1	2	3	4	5	6	7	8	9	10	11	
0	0	0	0	0	0	0	0	0	0	0	Read instruction
0	0	0	0	0	0	0	0	0	0	1	Partial Write instruction
			252 more combinations								
0	0	0	1	1	1	1	1	1	1	0	Partial Write instruction
0	0	0	1	1	1	1	1	1	1	1	Write Full Word instruction
0	0	1	0	0	0	0	0	0	0	0	Two Word Block Read
0	0	1	1	1	1	1	1	1	1	1	Two Word Block Write
0	1	1	0	0	0	0	0	0	0	0	Read Diagnostic instruction
0	1	1	0	0	0	0	0	0	0	1	Partial Write Diagnostic instruction
			252 more combinations								
0	1	1	1	1	1	1	1	1	1	0	Partial Write Diagnostic instruction
0	1	1	1	1	1	1	1	1	1	1	Write Diagnostic instruction
1	0	0	0	0	0	0	0	0	0	0	Read Maintenance Register
1	0	0	1	1	1	1	1	1	1	1	Load Error Function Register
1	0	0	1	0	0	0	0	0	0	0	Read Base Address Register
1	0	0	0	0	0	0	1	0	0	0	Read Error Function Register
1	0	0	0	0	0	0	0	0	0	1	Read 64K SBC register bits 0–11
1	0	0	0	0	0	1	0	0	0	0	Read 64K SBC register bits 12–15
0	1	0	1	0	0	0	0	0	0	0	Read with Program Lock instruction
0	1	0	1	0	0	0	0	0	0	1	Partial Write with Prog Lock instruction
			252 more combinations								
0	1	0	1	1	1	1	1	1	1	0	Partial Write with Prog Lock instruction
0	1	0	1	1	1	1	1	1	1	1	Write Full Word with Prog Lock instruction
	1	1	0	0	0	0	0	0	0	0	Test and Clear
	1	1	0	1	1	1	1	1	1	1	Test and Set
	1	1	1	0	0	0	0	0	0	0	Read and set 64K SBC word boundary
	1	1	1	1	1	1	1	1	1	1	Clear 64K SBC word boundary

NOTE: When cleared, Bit 0 of the information line will have no function in the MSU and when set, it unlocks the MSU.

Figure 4-2. Nonspecific Format for Broadcast Addressing Mode

4.5.3. Partial Write

The stored word is read and corrected by the MSU, and the corrected word is merged with the write word. The fields, specified by bits 4–11 of the instruction word, substitute write data if set. After merging write data, the MSU generates the appropriate Error Correction Code (ECC) check bits and writes the merged word with the ECC check bits into storage. Multiple bit errors detected during this cycle inhibits the storage write.

Write Control Bit	Write Data Bits Merged
4 = 1	0–5
5 = 1	6–8
6 = 1	9–11
7 = 1	12–17
8 = 1	18–23
9 = 1	24–26
10 = 1	27–29
11 = 1	30–35

4.5.4. Block Writes

The MSU, via the function code and the extended data transmission format, captures and acknowledges two write data transfers on consecutive bus cycles. The consecutive data words are received after the address. The MSU uses the address field transmitted by the requester that is incremented internally by the MSU. The MSU writes the first word indicated by bit 35 and writes the successive word after toggling bit 35 (only bit 35 of the address is changed). The MSU stores the data block with the generated ECC and aborts the sequence if an error is detected.

4.5.5. Block Read

The MSU reads and transfers a two-word block of single bit corrected data via the function code. The MSU uses the address transmitted by the requester and internally toggles bit 35 for a two-word block transfer. Block read data from the MSU is transmitted on consecutive bus cycles using the extended data transmission format. The requested storage word of the block is always the first word transferred followed by the next block words; that is, if the requested word was Word 1, the order of transferred words is Word 1 followed by word 0.

4.5.6. Test and Set

The MSU returns the corrected read data to the requester, and clears information bits 0–4 and sets bit 5 of the internal word. This information is stored with the correct ECC. Testing of bit 5 is done by the requester.

4.5.7. Test and Clear

The MSU returns the corrected read data and internally clears information bits 0–5 before rewriting the modified word with the correct ECC. Testing of bit 5 is done by the requester.

4.5.8. Program Lock

The MSU for a function code 101 and with the Unlock bit cleared, performs the operation defined by the Write Control field and sets the MSU Program Lock register. In a program lock condition, the MSU is busy to all requesters except the requester that initiated the program lock condition. The requester that initiated the Program Lock instruction clears the program lock condition by accessing the MSU with a broadcast instruction with the Unlock bit set. The MSU provides a Lock Time Out (see 4.9.4).

4.5.9. Unlock Bit

For a Program Lock, the MSU only services the requester that set the program lock condition and clears the program lock condition if the unlock bit is set. A set unlock bit overrides a program lock condition. (See 4.9.3 for Maintenance Lock.)

4.5.10. Read Diagnostic

The MSU performs a Read instruction that is modified by the contents of the Error Function register.

4.5.11. Read Base Address Registers

The MSU transmits the contents of the Base Address register to the requester.

NOTE: All broadcasted requests including the Read Base Address Register instruction, must be within the address range of the MSU (see 4.8.5).

4.5.12. Read Error Function Register

The MSU transmits the contents of the Error Function register to the requester.

4.5.13. Read 64K SBC Boundary Register Bits 0–11

The MSU transmits 64K SBC boundary register bits 0–11 to the requester.

4.5.14. Read 64K SBC Boundary Register Bits 12–15

The MSU transmits 64K SBC boundary register bits 12–15 to the requester.

4.5.15. Write Diagnostic

The MSU performs a Write instruction that is modified by the contents of Error Function register.

4.5.16. Read Maintenance Register

The MSU returns the contents of the Maintenance register to the requester and clears the silent error portion of the Maintenance register. A broadcasted Maintenance register read, conditions the MSU Maintenance register to be overwritten by the next error. A broadcast Maintenance register read with the Unlock bit set is required to clear a maintenance lock condition.

4.5.17. Load Error Function Register

The MSU loads bus information into the Error Function register (bits 6–23,81–83).

4.5.18. Clear the 64K SBC Boundary

The MSU responds to this instruction by clearing the 64K SBC boundary that encompasses the address transmitted by the requester. Uncorrected read data is transmitted for all succeeding requests to the cleared 64K SBC boundary.

4.5.19. Set the 64K SBC Boundary

The MSU responds to this instruction by setting the 64K SBC boundary that encompasses the address transmitted by the requester. The MSU also returns corrected read data.

4.6. Directed Instruction Set (Message Format)

The MSU message instruction set is a subset of the system bus Message Format instruction set. Message Format instructions not within the MSU Message instruction set have an unknown result in the MSU. The message instruction set is associated with direct addressing mode requests. Information fields of an acknowledged directed request are interpreted from the format in Figure 4-3.

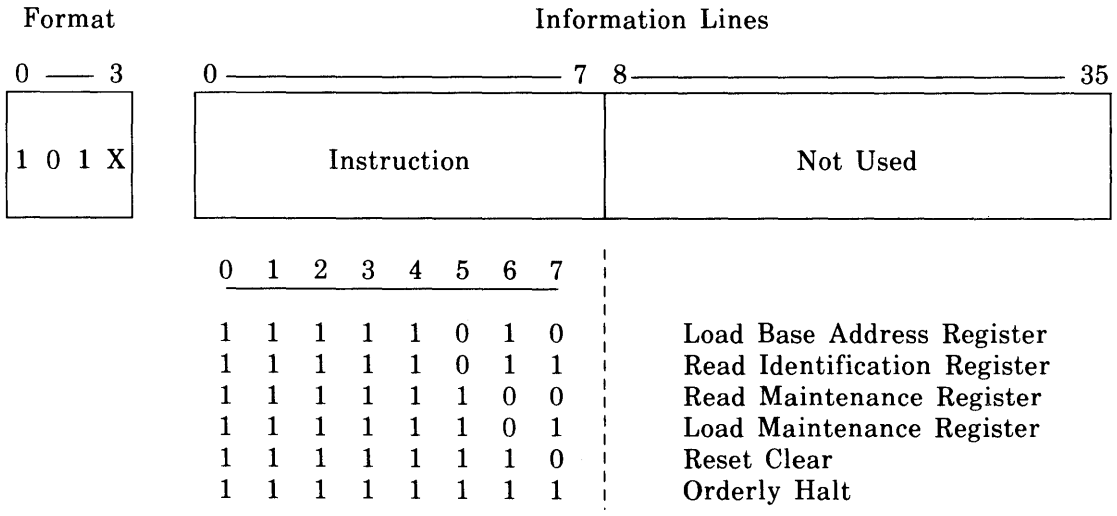


Figure 4-3. Message Format for Direct Addressing Mode

4.6.1. Load Base Address Register

The MSU loads bus information into the Base Address register (bits 12-17,82). The Base Address register is described in 4.8.5.

4.6.2. Read Unit Identification

The MSU returns the unit identification to the requester. See 4.8.8 for Identification Register operation.

4.6.3. Read Maintenance Register

The MSU returns the contents of the Maintenance register to the requester and clears the Maintenance register (see 4.9.2).

4.6.4. Load Maintenance Register

The MSU loads bus information into the Maintenance register.

4.6.5. Reset Clear

A Reset Clear instruction is not locked out by a refresh, program lock, or a maintenance lock. The MSU is never busy to a Reset Clear instruction. The MSU response is:

- Clear all MSU Main Rank registers (S-bus Enable remains unchanged)
- Set a 1 in the first Scan/Rank register to designate a gate array boundary.
- Clear 64K boundaries (disable SBC)
- Require restoring correct ECC in all storage locations (initialization)

4.6.6. Orderly Halt

The MSU responds to an Orderly Halt instruction by clearing the Cycle Compare Error, Sequence Control Error, or Refresh Error holding registers. If not cleared, certain nonspecific instructions receive READ MEMORY FAULT and WRITE MEMORY FAULT signals (see 4.9.1.1 and 4.9.1.2).

4.7. MSU Control

Control in main storage is achieved through the use of the system clock, unconditional phases, and three shift registers (Refresh Shift register, Bus Control Shift register, and Array Control Shift register.)

4.7.1. System Clock

The system synchronous clock has four phases. It is a 108 nanosecond square wave and is active as long as power is applied to the system. The four system phases associated with the system clock are controlled by the System Support Processor. The four system phases are deactivated for Scan/Set and start/stopped for cycle step operations.

4.7.2. Unconditional Phases

The system generates four unconditional phases that are active as long as power is applied to the system. The four unconditional phases have the same timing relationships as the four system clock phases. The four unconditional phases control MSU refresh, the refresh interval counter, and other storage device related functions. The four unconditional phases allow deactivation

of the four system clock phases without corruption of the storage data. The unconditional phases are deactivated for Scan/Set operations to string four. Initialization of the MSU is required after a Scan/Set operation to string four.

4.7.3. Refresh Shift Register

The Refresh Shift Register (RSR) is a 3-bit shift register that controls the refresh cycle of main storage. The RSR is normally cleared. The refresh cycle is started by a refresh interval counter that generates a carry every 23 microseconds. This carry initiates a refresh request. Refresh is performed on the next available cycle (refresh has highest priority). To start a refresh cycle the MSU shifts a bit into the RSR. This bit is shifted through the RSR by phase 3 of the unconditional phases.

4.7.4. Bus Control Shift Register

The Bus Control Shift Register (BCSR) is an 11-bit shift register that controls the MSU system bus and non-storage array operations. A single bit is shifted through the BCSR by phase 03 of the system clock. The position of the bit in the BCSR, along with the decode of the write controls and function codes, determines what and when an event occurs.

The first bit in the BCSR is set when an S-bus broadcast request is accepted by the MSU. Directed requests do not start the BCSR.

Instruction that do not require information be returned to the requester, cause the bit in the BCSR to be shifted on each phase 3 of the system clock until an End Of Cycle Clear (EOCC) is decoded. The EOCC clears the BCSR and opens the MSU to new requests.

Instructions that require information be returned to the requester, cause the bit in the BCSR to be shifted on phase 3 of the system clock until a vie for S-bus priority is decoded. The BCSR is then inhibited from shifting until priority for the system bus is granted. The BCSR is then shifted until an EOCC is decoded.

4.7.5. Array Control Shift Register

The Array Control Shift Register (ACSR) is an 11 bit shift register clocked by the unconditional clock phases and controls the storage array-RAS,CAS, and Partial Write Merging.

- Normal Operation - During normal operation, the ACSR and BCSR start and shift bit positions at the same time. The first bit in the ACSR and the BCSR is set when S-bus broadcast request is accepted by the MSU. The ACSR is then shifted by unconditional phase 3 until an end of cycle clear is decoded. The EOCC clears the ACSR. The MSU does not accept a bus request if a refresh request is present or a refresh is in process.

- The BCSR is shifted by phase 03 until a cycle hold or end cycle is decoded. A cycle hold results from a busy S-bus preventing the MSU from gaining access to the bus. An S-bus Transmit enable resumes the BCSR operation. The END CYCLE signal clears the BCSR and opens the MSU to new requests.

- Cycle Step Operation - During cycle step operation, the ACSR is out of phase with the BCSR. The BCSR starts with an S-bus request and is shifted every cycle step. After three cycle steps (three shifts of the BCSR), a bit is shifted into the ACSR starting the array control. The ACSR is shifted every system clock cycle by unconditional phase 3. By definition, the ACSR must be allowed to shift to completion. Some overlap between the ACSR and the

next cycle step is allowed. However, a minimum of seven sync clock cycles is needed between cycle steps.

NOTE: Cycle step operation is the only time the BCSR and the ACSR are independent of each other.

- The MSU initiates a refresh cycle if the refresh request occurs prior to the starting of the ACSR. A refresh request occurring while the ACSR is shifting is accepted by the cycle after the ACSR has finished.

4.8. MSU Functions

4.8.1. Error Function Register

The Error Function register is used to inject errors into the MSU and onto the S-bus interface. An Error Function Register command loads the Error Function register for the error injecting operation. The MSU will respond to diagnostic write or read operations with normal write or read operations modified by the contents of the Error Function register. A Broadcast instruction with function code bits 1, 2, and 3 (011), activate the Error Function register. This register can be read with the Read Error Function Register instruction.

4.8.2. Cycle Step

Cycle step is a continuous system clock operation with periodic stepping of the four phases. The timing relationship between the four phases is not changed during cycle step operation. The time between steps; that is, phase one to phase one, is under control of the SSP and is variable. This time interval (or pause) allows the MSU to refresh, if necessary, before proceeding to the next cycle step.

4.8.3. Initialization

All storage locations must be full word written prior to the execution of reads or write partials. The MSU address range must be traversed twice during initialization to ensure the proper operation. Initialization is required after a Power Up Clear, Reset Clear, Refresh Fault, or a scan/set operation to the MSU string 4.

4.8.4. Refresh

The MSU internally controls refresh cycles that are used to restore information in the dynamic storage device. Refresh cycles cause the requester to see a destination busy. Stopping the system clock, results in the loss of refresh and corruption of storage data. Refresh has the highest priority and occurs once every 25 microseconds. The refresh rate and interval addresses are controlled by internal counters.

4.8.5. Base Address Register

For broadcasted requests, the MSU provides a Base Address Register (BAR) that contains the starting address of the MSU and provides system address continuity. The BAR determines the address range of the MSU that is defined as the addresses from the starting address (contents of the BAR) to the starting address plus the MSU capacity inclusively. In broadcasted accesses, the MSU responds only if the broadcasted address is within the address range of the MSU.

The binary address range of the BAR is 000000 to 111110 or 111100 inclusively for 1/2 megaword or 1 megaword, respectively. BAR addresses above this range with MSU capacity results in a BAC error. Bit 17 of the BAR must always be loaded in the cleared state. A set bit 17 of the BAR results in an ARP error.

The BAR is loaded by the Load BAR command and is loaded independent of parity errors in the data of the system bus Information transfer. The BAR can be read by the Read BAR instruction.

4.8.6. Drop Address

The Drop Addresses are switches that define the MSU drop for the direct addressing mode. The destination address bits 1–7 are compared to the Drop Address switches. Bits 1–3 are equal to 010, respectively and define MSUs in general. Bits 5–7 define a particular MSU. Bits 0 and 4 are zeros. When transmitting data, the MSU transmits the MSU drop address as the source address. Bit 0 and 4 of the source address are transmitted as zeros.

4.8.7. Address Selection

The address structure normally has bits 20–35 associated with the 64K RAM address. Bits 22–29 are the row selection bits and bits 20, 21, 30–35 are the column selection bits. Bits 16–19 select the 64K word sections and bits 12–16 identify the storage drop being accessed. For broadcasted instructions, bits 12–16 are compared to the MSU address range determined by the Base Address register and the MSU capacity. The MSU internal address equals the received address minus the Base Address register contents. (see Figure 4–4.).

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
Row Select				C_7	C_6	R_7	R_6	R_5	R_4	R_3	R_2	R_1	R_0	C_5	C_4	C_3	C_2	C_1	C_0
0	0	0	0	65K															
0	0	0	1	131K		C = Column Address													
0	0	1	0			R = Row Address													
0	0	1	1																
0	1	0	0																
0	1	0	1																
0	1	1	0																
0	1	1	1																
1	0	0	0																
1	0	0	1																
1	0	1	0																
1	0	1	1																
1	1	0	0																
1	1	0	1																
1	1	1	0																
1	1	1	1	1 Meg															

Figure 4–4. Row Select Bits

4.8.8. Identification Register

The Identification (ID) register identifies the storage capacity, FCO level, and identification straps. The ID register is read directly by the requester using the direct address mode (see Figure 4–5).

The ID register is implemented as follows:

- 0–19 Primary features are embedded in the gate array logic and are unchangeable.
- 20–27 Revision is implemented in the switches.
- 28–35 Except for bit 35, modifier bits are embedded as zeros in the gate array logic. Bit 35 is the 1 megaword capacity and is set automatically when the MSU capacity is increased to 1 megawords.

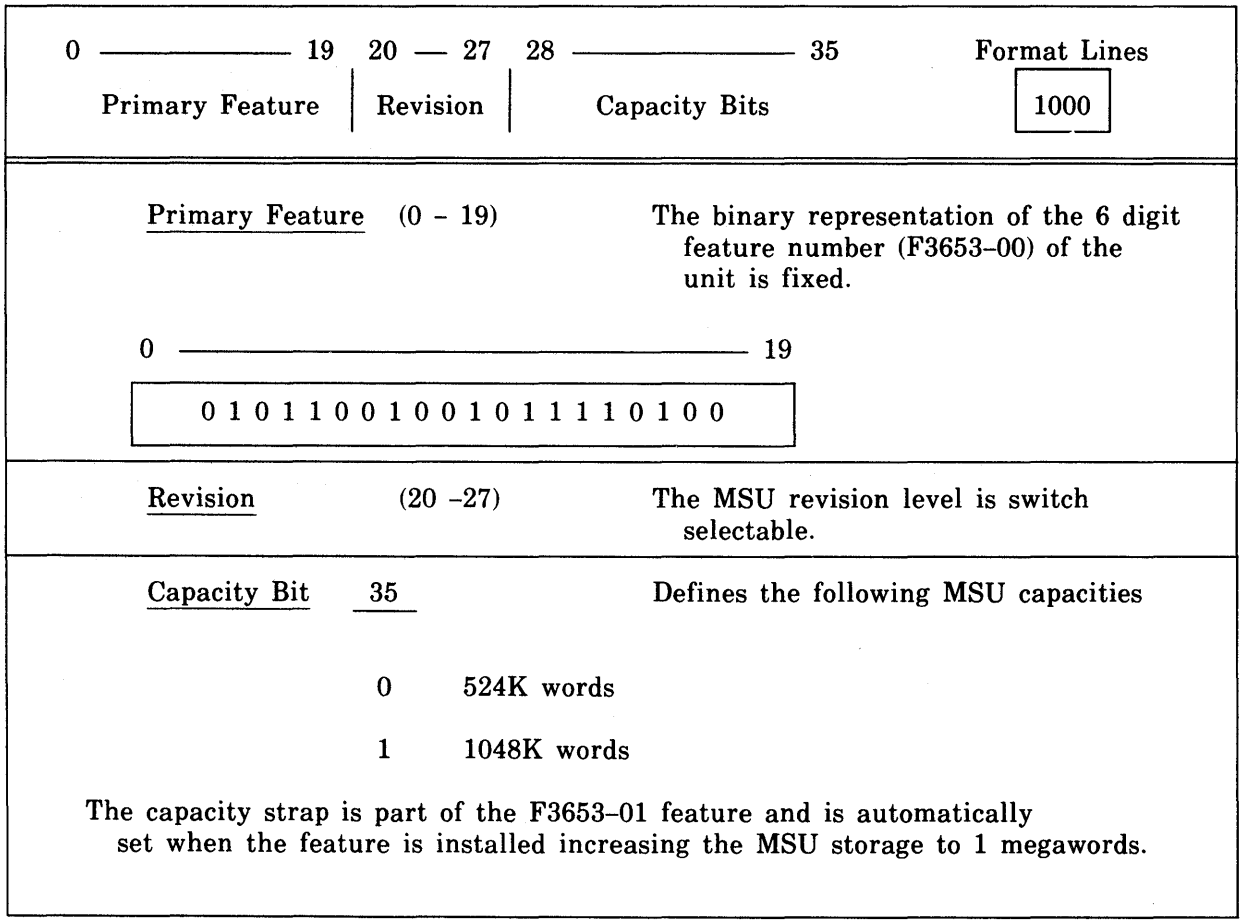


Figure 4–5. Identification Register

4.9. Error Reporting

Errors are reported by the MSU as described in the following subsections.

4.9.1. Error Signals

4.9.1.1. Read Memory Fault

The READ MEMORY FAULT (RMF) signal notifies the requester that the MSU has detected an error, and the error is stored in the Maintenance register. This signal is transmitted in MSU cycles in which data is transmitted to the requester. It is synchronized to data and is valid one bus cycle after read data is valid. Only one RMF signal is transmitted per MSU instruction. For a Block Read instruction, the RMF signal is synchronized to the last data transfer.

The RMF signal is activated for the following errors:

- Single Bit Error detected during read accesses to addresses within a disable 64K SBC boundary
- Multiple Bit Error
- Refresh Error*
- Sequence Control Error*
- Address Control Error
- Error Function register (Read diagnose instructions only)
- Data through Check and False Error Correction errors
- Cycle Compare Error*
- Address Register Parity error
- Instruction Register Parity error
- Byte Correction Error

* *Requires a POWER UP signal, a Reset Clear instruction, or an Orderly Halt instruction to clear the Error Holding registers. If the Error Holding registers are not cleared, all the following nonspecific instructions except those with a function code value of 100 receive an RMF signal.*

NOTES: 1. *The RMF signal is generated for all Broadcast instructions except nonspecific instructions with function code bits 1, 2, and 3 set with a value of 100, respectively. Directed instructions do not result in an RMF fault signal.*

2. *If an RMF signal occurs during Read and Write instructions, the instruction cycle time is extended by two clock cycles.*

4.9.1.2. Write Memory Fault

The WRITE MEMORY FAULT (WMF) signal notifies the requester that the MSU has detected an error, and the error status is stored in the Maintenance register. This signal is transmitted four cycles after the ADDRESS ACKNOWLEDGE signal, and in MSU cycles in which data is not transmitted to the requester.

The WMF signal is activated for the following errors:

- Multiple Bit Error (during partial write)
- Refresh Error*
- Sequence Control Error*

- Address Control Error
- Error Functions register (Write diagnose instructions only)
- Partial Through Check and Data Through Check errors
- Cycle Compare Error*
- Address Register Parity error
- Instruction Register Parity error
- Byte Correction Error during partial write

* *Requires a POWER UP signal, a Reset Clear instruction, or an Orderly Halt instruction to clear the Error Holding registers. If the Error Holding registers are not cleared, all the following nonspecific instructions except those with a function code value of 100 receive an WMF signal.*

NOTES: 1. *The WMF signal is generated for all Broadcast instructions except nonspecific instructions with function code bits 1, 2, and 3 set with a value of 100, respectively. Directed instructions do not result in a WMF signal.*

2. *If a WMF signal occurs during Read and Write instructions, the instruction cycle time is extended by two clock cycles.*

4.9.1.3. Silent Errors

The following errors are silent because the requester is not notified that the error is detected:

- Destination Address Parity (DAP)
- Broadcast Address Check (BAC)
- Information Parity Error (IPE)
- Lock Time Out (LTO)

The requester determines if a silent error has occurred by reading the MSU Maintenance register. The first IPE detected by the MSU is loaded into the Maintenance register and remains valid until the Maintenance register is read or cleared by a POWER UP signal or a Reset Clear instruction. The first DAP, BAC, or LTO detected by the MSU is loaded into the Maintenance register and remains valid until the Maintenance register is read or cleared by a POWER UP signal or a Reset Clear instruction.

4.9.1.4. S-Bus Transfer Status

For every S-bus cycle acknowledged, the MSU transmits a Bus Transfer Status byte. The following statuses are reported:

Status Bit	Status
0	Destination Busy
1	Parity error in transfer
2	Parity error is source address
3	Parity error in format
4	Parity error in Information Bits 0–11
5	Parity error in Information Bits 12–23
6	Parity error in Information Bits 24–35

7	Reserved
8	Even Parity

4.9.2. Maintenance Register

The MSU Maintenance register for stores the status of errors detected by the MSU. The Maintenance register is loaded with all the errors detected by the MSU except those reported in the Bus Transfer Status and the errors that and are single bit corrected. For a Broadcasted Maintenance register read, the silent errors of the Maintenance register are cleared. Subsequent errors will overwrite the Maintenance register. For a directed Maintenance register read, the entire Maintenance register is cleared.

4.9.3. Maintenance Lock Register

RMF and WMF errors set the Maintenance Lock Register and lock out the MSU to all requesters (Destination Busy), except the requester receiving the RMF or the WMF signal. During the Maintenance lock, all requests from the requester receiving the RMF or the WMF signal are acknowledged and bus statuses are returned but result in NO-OP by the MSU unless the instruction is a Maintenance register read with the Unlock bit set. In which case, the Maintenance Lock register is cleared and the contents of the Maintenance register is transmitted to the requester. It is the responsibility of the requester receiving the RMF or WMF signal to unlock the MSU. If the requester does not read the Maintenance register with an unlock, after 6 milliseconds the MSU performs a Lock Time Out.

4.9.4. Lock Time Out

A timeout is provided on either a program or maintenance lock condition. A requester has 6 milliseconds to clear a lock condition before a Lock Time Out (LTO) occurs. An LTO is a silent error and results in the Maintenance Lock register and Program Lock register being cleared and the LTO bit being set in the Maintenance register without transmitting an RMF or a WMF signal. An LTO does not clear the Maintenance register but subsequent errors (excluding silent errors) overwrite the Maintenance register. The LTO is inhibited if bit 9 in the Error Function register is set. or a parity error exists for byte one of the EFR or a cycle step line is active.

4.10. Error Detection and Handling

Errors detected by the MSU are handled with one of the following techniques:

- The MSU aborts the cycle (NO-OP) and reports the error in an acknowledge status (Information and Source errors). Two exceptions are Load EFR and Load BAR instructions, which are not aborted by errors in the information fields of a data transfer.
- The MSU aborts the cycle (NO-OP) and loads the MSU Maintenance register without transmitting an RMF or a WMF signal (DAP and BAC).
- The MSU loads the Maintenance register without an RMF or a WMF signal (LTO and IPE).
- The MSU inhibits storage writes, loads the Maintenance register, sets the Maintenance Lock register, and transmits RMF or WMF signals (REF, SCE, 64K SBC Boundary, CCE, ACE, IRP, ARP, MBE, PTC, FEC, DTC, BCE, and EFR).

4.10.1. Interface (System Bus) Errors

4.10.1.1. Destination Address Parity Error

The MSU checks the parity of the destination address and the format fields. A Destination Address Parity (DAP) error occurs for parity errors detected in the destination address for broadcasted and directed requests or for a parity error detected in the format field for broadcasted requests. Format errors in directed requests are reported in the system bus transfer status.

MSUs detecting DAP errors shall abort the cycle and set the DAP bit in the Maintenance Register. The DAP error is a silent error. The MSU does not respond with an acknowledge, a status, RMF, or WMF.

4.10.1.2. Broadcast Address Check

Through checking is provided on broadcasted requests by:

- checking parity on the system bus address bits 12–17,
- checking parity on the Base Address register,
- checking the maximum allowable Base Address Register versus capacity, and
- checking parity on the Base Address Adder.

The detection of a Broadcast Address Check (BAC) error results in the MSU not responding to the requester with an ACKNOWLEDGE, STATUS, RMF, or WMF signal. The MSU sets the BAC bit in the Maintenance Register. For a DAP error in the format field, the response in the same, and a Bus Status is returned. The DAP bit remains set until the Maintenance register is read. The BAC is a silent error.

4.10.1.3. Information and Source Parity Errors

Parity errors detected by the MSU in the source address field or in the information field (except BAC) shall result in a BUS ACKNOWLEDGE signal. The error is reported in the system bus transfer status. The MSU aborts the cycle without an RMF or a WMF signal, without loading the MSU Maintenance register, and without altering the contents of the MSU. Load EFR and Load BAR instructions are exceptions which the MSU executes with parity errors in the information field of the data transfer.

4.10.1.4. Information Parity Errors

During transmission to the requester, parity is checked on the system bus information. Detection of a parity error in the transferred information results in the appropriate code being stored in the IPE bits of the Maintenance register. IPE errors are silent.

4.10.2. ECC Detection and Handling

4.10.2.1. ECC Detection

The MSU uses a seven bit Error Correction Code (ECC). The MSU can correct all single bit errors, detect all double bit errors, and detect odd multiple bit errors that do not define a valid SBE. The MSU transmits uncorrected read data with correct interface parity for read cycles unless the address of the requested word is within an enabled 64K SBC boundary; in which case, single bit corrected data is transmitted. Read data in partial write, test and set, test and clear, and block read cycles shall be single bit corrected without transmitting an RMF or a WMF signal independent of 64K SBC boundaries.

4.10.2.2. ECC Error Handling

ECC errors are handled by:

- activating either the RMF or WMF line;
  - The MSU activates the RMF line for:
    1. SBEs or MBEs detected for read requests with an address within a disabled 64K SBC boundary.
    2. MBEs detected for read requests with an address within an enabled 64K SBC boundary.
    3. MBEs detected during test and set, or test and clear (write is inhibited).
    4. MBEs detected during block read cycles.
  - The MSU activates the WMF line and inhibits storage writes for MBEs detected during partial writes.
- loading the MSU Maintenance register with error information
- rewriting SBEs detected in read cycles and
- setting the maintenance lock register.

4.10.2.3. 64K SBC Boundary

Single Bit Correction (SBC) is provided on 64K address boundaries that encompass the address range of the 64K storage device. The MSU provides sixteen 64K SBC boundaries; one for each row of 64K storage devices in a one million word storage. A 64K SBC boundary is enabled by the Set 64K SBC Boundary instruction and is cleared by a Reset Clear or a Clear 64K SBC Boundary instruction or a POWER UP CLEAR signal. Only the 64K SBC Boundary register encompassing the requester's address is set or cleared by the Set or Clear 64K SBC Boundary instructions.

The MSU transmits corrected read data for read requests to an enabled 64K SBC boundary.

During read cycles the MSU requests system bus priority at the same time in the MSU read cycle independent of whether or not the MSU is in SBC mode.

- If the request is to a disabled 64K SBC boundary, the MSU transmits uncorrected read data to the requester the first bus cycle that the MSU gains priority.

- If the request is to an enabled 64K SBC boundary, the MSU transmits uncorrected read data to an all zeroes destination address in the first bus cycle that the MSU has priority, using priority disable. In the following system bus cycle, the MSU transmits corrected read data to the requester.

The 64K SBC Boundary registers can be read by the Read 64K SBC Boundary register instruction.

4.10.2.4. SBE Rewrite

The following SBE rewrites occur:

- SBE Rewrite Read

  The MSU rewrites single bit corrected data for SBEs detected during read instructions. The read cycle time is extended by two clock cycles.

- SBE Rewrite Partial Write

  The MSU rewrites SBEs during partial writes independent of the 64K SBC boundary.

4.10.3. Through Checking

4.10.3.1. Partial Through Checking

When Partial Through Checking (PTC) errors are detected, the appropriate PTC bits are set in the Maintenance register, the requester receives an RMF or a WMF signal, and storage writes are inhibited.

- Write Byte Parity - The system bus parity received on the 6-bit byte write data fields is used internally by the MSU and is checked at internal MSU registers until parity is replaced with ECC check bits.

- Merge Parity Predict (PTC) - The MSU provides merge parity prediction on partial write data merging by generating parity on the Write and Read data prior to merging and on the merged word after merging. The parity is generated on the data boundaries of the write control fields (63366336). The parities of the merged word is compared to the respective Write and Read data parities as determined by the requester's Write Control field. In the process of generating parity on the write control boundaries, the 6-bit Data Slice parity is checked on Write, Read, and merged data.

4.10.3.2. Data Through Checking

When Data Through Checking (DTC) errors are detected, the requester receives an RMF or a WMF signal, and inhibits storage writes.

- Overall Check Bit Parity - The 6-bit byte write data partitioning of the interface is used internally by the MSU. Each 6-bit byte (data slice) generates seven partial check bits. The seven check bits stored with the write data are the resultant Exclusive ORing of the

respective seven partial check bits from each of the six data slices. Each data slice has one partial check bit, which is the overall parity of the data slice. This overall partial check bit is compared to the S-bus parity received on the 6-bit data slice.

- Data Through Check - The over all parity of the stored check bits equals the complement of the overall parity of generated 6-bit byte parities. The MSU checks for this condition for both write and read data.

- Information Parity - Prior to transmission to the requester, the MSU checks the parities on the S-bus information lines. Detection of a parity error in the transferred information results in the appropriate code being stored in the IPE bits of the Maintenance register. IPE errors are silent.

4.10.3.3. Address Through Checking

Address through checking is provided as follows:

- Address Compare Errors (ACE) - The address path in the MSU is paralleled and compared at the input to the storage device drivers. The ACE also includes a gate array to gate array Refresh Address Parity check. Detection of this error causes the MSU to set the ACE bit in the Maintenance register, inhibit storage write, and transmit an RMF or a WMF signal.

- Refresh Faults (REF) - The MSU uses parity prediction on the refresh address and on the refresh cycle shifter. Each refresh counter has even parity. Detection of either error causes the MSU to set the REF bit in the maintenance register, and transmit an RMF or WMF signal. The MSU continues to transmit RMF and WMF signals to all requesters until the MSU is cleared by a POWER UP CLEAR signal, or a Reset Clear or Orderly Halt instruction.

- Address Register Parity (ARP) - The MSU carries the S-bus parity on address bits 12-35 internal to the MSU and checks for ARP errors in register to register transfers. ARP errors also include gate array to gate array BAR parity check. The ARP error is valid only in the absence of a BAC error. ARP errors cause an RMF or a WMF signal, abort the cycle, and set the ARP bit in the Maintenance register.

- MOR Driver Through Checking - The loading on the storage device drivers (MOS) is divided between 2 different half words. This increases the probability of detecting MOS driver faults, by forcing Multiple Bit Errors.

4.10.3.4. Error Function Register Through Checking

The MSU stores S-bus parity with the Error Function Register (EFR) and checks parity during Write and Read Diagnostic instructions. Detection of this error sets the EFR bit in the Maintenance register, aborts the cycle, and transmits an RMF or WMF signal.

4.10.3.5. 64K SBC Boundary Through Checking

The MSU provides parity on the sixteen 64K SBC Boundary registers. Detection of this error sets the 64K error bit in the Maintenance register, aborts the cycle, and transmits an RMF or WMF signal.

4.10.3.6. Sequence Control Through Checking

The MSU checks parity on internal sequence circuitry. Detection of an error sets the Sequence Control Error (SCE) bit in the Maintenance register, aborts the cycle, and sends an RMF or WMF signal to the requester. The MSU continues to transmit RMF or WMF signals until cleared by a POWER-UP CLEAR signal, or a Reset Clear or Orderly Halt instruction.

4.10.3.7. Instruction Register Through Checking

The MSU provides parity on the internal MSU Instruction register. Detection of this error sets the Instruction Register Parity (IRP) bit in the Maintenance register, aborts the cycle, and transmits an RMF or WMF signal.

4.10.3.8. Cycle Compare Through Checking

The MSU provides parity checking on internal MSU bus cycle circuitry. The detection of a Cycle Compare Error (CCE) sets the CCE bit in the Maintenance register, aborts the cycle, and transmits an RMF or WMF signal. The MSU continues to transmit RMF or WMF signals until cleared by a POWER-UP CLEAR signal, or a Reset Clear or Orderly Halt instruction.

4.10.3.9. Byte Correction Through Checking

The Byte Correction Error (BCE) is based on the fact that the MSU error correction circuitry is duplicated for each data byte and that each data byte decodes a unique BYTE CORRECTION signal. For a BCE, the Maintenance register is loaded with the ECC syndrome bits and the status of the six BYTE CORRECTION signals.

5. Integrated Streaming Tape Subsystem

5.1. Subsystem Components

The streaming tape subsystem consists of an F3674 Integrated Tape Control Unit contained in the host system central complex cabinet and one or two K3782 Streaming Tape Drives contained in a separate peripheral cabinet adjacent to the central complex cabinet. The first tape drive has a power control module that provides compatible power control for the tape drives in the peripheral cabinet.

5.1.1. Integrated Tape Control Unit

The Integrated Tape Control Unit (ITCU) provides interface and control logic to the host system Byte Bus Channel (BBC) on the L-bus, and to the streaming tape drives.

5.1.1.1. Data Rates

The ITCU can accommodate data rates up to 200 kilobytes/second.

5.1.1.2. Recording Mode

The ITCU supports Non Return to Zero Inverted (NRZI) and Phase Encoded (PE) recording modes.

5.1.1.3. Data Checking and Read Correction

The ITCU and the formatter together provide facilities for checking and correction for NRZI and PE modes. Single-track Read errors are automatically corrected in PE mode. In NRZI mode, single track errors are reported in sense byte 2 and are recoverable by software. Multi-track errors are reported but are not correctable.

5.1.1.4. Tape Formatting

The tape formatter controls the tape format during writing—this includes interblock gap timing, preamble/postamble recording, and generation of file marks. Gap detection and detection of preamble/postamble on reading is also done in the formatter.

5.1.2. Streaming Tape Drive

The streaming tape drive provides for dual speed operation. The high speed streaming mode operates at 100 inches per second with a data rate of 160 kilobytes per second. The low speed streaming mode operates at 25 inches per second with a data rate of 40 kilobytes per second. The operational speed mode is selected by command under program control.

The streaming tape drive does not have the capability to stop and start within the interblock gap as defined in ANSI X3.39-1973. A repositioning cycle is required whenever the streaming mode is interrupted. This repositioning cycle creates inefficiencies in the data throughput rate and should be avoided.

This drive uses the 1600 bits per inch Phase Encoded recording format and permits compatible interchange of tapes per ANSI X3.39-1973. Tapes written in either the high speed streaming mode or low speed streaming mode meet the requirements of ANSI X3.39-1973.

5.1.2.1. Formatter

Each streaming tape drive has built in formatter electronics.

5.1.2.2. Streaming Tape Power Control Module

This feature must be installed with the first streaming tape drive. It provides power for up to two streaming tape drives.

5.2. Streaming Tape Characteristics

Table 5-1 provides streaming tape functional characteristics.

Table 5-1. Functional Characteristics for Streaming Tape

Function	Characteristic
Operating Functions	Reads and writes 9-track tape. Reads forward and backwards, writes forward only. Read after write check capability.
Recording Density	1600 bits/inch
Recording Format	9-track Phase Encoded (PE)
Tape Speeds	25 and 100 inches/second
Transfer Rate	40,000 bytes/second at 25 inches/second 160,000 bytes/second at 100 inches/second
Access Times	225 milliseconds at 100 inches/second 55 milliseconds at 25 inches/second
Tracks on Tape	8 data, 1 parity
Interblock Gap	0.6 inch
Interblock Gap Time	6 milliseconds at 100 inches/second 24 milliseconds at 25 inches/second
Rewind Time (2400 feet)	190 seconds (maximum)

5.2.1. Dual Speed Streaming Tape Operation

5.2.1.1. High Speed Streaming Operation

High speed streaming mode is the primary operational mode and is especially suited to back up data stored on nonremovable disk packs.

The operational objectives provide a maximum data throughput in the streaming mode and increased capabilities to maintain the streaming mode to avoid the inefficiency of a repositioning cycle.

The subsystem can operate in a streaming (non-stop) mode with interblock gap passing time of six milliseconds when writing 9-track tapes at 100 inches per second with a nominal 0.6 inch interblock gap.

The maximum time to go from a stopped state to the transfer of the first data byte is 225 milliseconds.

The time from completion of a command execution to the latest point that the tape transport can accept another command without a repositioning cycle is:

- Read to Read commands - 4.7 milliseconds, minimum
- Write to Write commands - 2.7 milliseconds, minimum

Software must provide a stop to allow repositioning for a Write command following a Read Forward operation. A Read Forward after a Write operation is illegal.

Repositioning occurs on all direction changes.

The maximum time from a command overrun to the transfer of the first byte of data from a command received during a recovery cycle is 1020 milliseconds.

The maximum time to recover from a command overrun and return to the stopped state to await the next operation command is 750 milliseconds.

5.2.1.2. Low Speed Streaming Operation

The low speed streaming mode is provided for those dump/restore backup operations where a lower effective data rate is needed to avoid the inefficient repositioning cycles.

Low speed streaming mode also allows the emulation of traditional start/stop operations. However, since the streaming tape drive does not have the capability to stop and start within the interblock gap, a repositioning cycle is required after each stop operation. The emulation of a start/stop operation is more effective where the stop time between run commands is significant and absorbs the repositioning time.

The subsystem can operate in a streaming (non-stop) mode with interblock gap passing times of 24 milliseconds when writing 9-track tapes. Low tape speed is 25 inches per second with a nominal 0.6 inch interblock gap.

The maximum time to go from a stopped state to the transfer of the first data byte is 55 milliseconds.

The time from completion of a command execution to the latest point that the tape transport can accept another command without a repositioning cycle is:

- Read to Read commands - 19.7 milliseconds, minimum
- Write to Write commands - 11.7 milliseconds, minimum

Software must provide a stop to allow repositioning for a Write command following a Read Forward operation. A Read Forward after a Write operation is illegal.

Repositioning occurs on all direction changes.

The maximum time from a command overrun to the transfer of the first byte of data from a command received during a recovery cycle is 225 milliseconds.

The maximum time to recover from a command overrun and return to the stopped state to await the next operation command is 160 milliseconds.

5.2.2. Load/Unload Operation

The threading and loading sequence is fully automatic with selfcorrecting capabilities. The sequence of events consists of seating the reel, testing for reel size, threading the tape and calibrating the reel servo control.

When no threading retries are required, the time from activating the LOAD switch on the operator panel until the tape is stopped at the Beginning Of Tape (BOT) with tension established

is 40 seconds. If threading problems occur, the unit makes a total of three attempts to thread before aborting. Each threading retry adds 16 seconds to the total time. On reels that are seated upside down, the unit aborts after one threading attempt.

The maximum time to rewind 2400 feet of tape is 190 seconds.

The maximum time from a reversal command occurring during:

	Hi Speed	Low Speed
■ Command reconstruct time to the transfer of the first data byte is:	450 ms	110 ms
■ Recovery cycle to the transfer of the first data byte is:	1500 ms	310 ms
■ Stopped state to the transfer of the first data byte is:	750 ms	170 ms

The maximum command construction times are:

	Hi Speed	Low Speed
■ Read to read operations:	4.7 ms	19.7 ms
■ Write to write operations:	2.7 ms	11.2 ms

The maximum repositioning times are:

■ To stop position:	850 ms	170 ms
■ From stop to data in same direction:	325 ms	70 ms
■ From stop to data in opposite direction:	900 ms	190 ms

5.2.3. Operational Capabilities

5.2.3.1. Reading/Writing

Tape reading or writing operations occur only when tape is moving at high or low speeds (±4%).

5.2.3.2. Simultaneous Operation

Only one Read or one Write (with write check) operation can be performed at one time on any streaming tape configuration. Any number of streaming tape drives can simultaneously be engaged in a Rewind Tape operation without restricting read or write capability.

5.2.3.3. Phase Encoding Format and Conventions

A tape that was written in Phase Encoding mode has an identification burst written on it in the area of the load point or BOT indicator. This identification burst consists of 1600 flux reversals per inch on track 4 and dc erasure on all other tracks. It begins 1.7 inches minimum before the trailing edge of the BOT marker and continues past the trailing edge of the BOT marker, but ending at least 0.5 inch before the first block. This ensures a proper identification of a tape when reading from the BOT marker. A data block in Phase Encoding mode consists of three basic sections:

1. A "preamble" or synchronizing pattern consisting of 40 bytes of binary zeros in all tracks followed by a single byte of binary ones in all tracks.

2. The data portion of the block consisting of N to M bytes. M is limited only by program conventions and the size of the storage in the host system. The formatter does not limit the number of data bytes in the block. A suggested limit for the range N to M is 18 to 2048 bytes in accordance with ANSI X3.39-1973 for data interchange. However, longer data block lengths may be desirable for more effective streaming operations. During the data portion of the block, the bytes are written with odd parity.

3. A "postamble" consisting of a single byte of binary ones in all tracks followed by 40 bytes of binary zeros in all tracks.

   With this tape format, blocks on tapes are completely symmetrical, facilitating reading in either direction. Interblock gaps in this mode of recording are measured from the end of the postamble to the beginning of the preamble. Thus, when calculating average data transfer rates, a time equivalent to 82 bytes must be added to the interblock gap passing time to obtain the correct data rate.

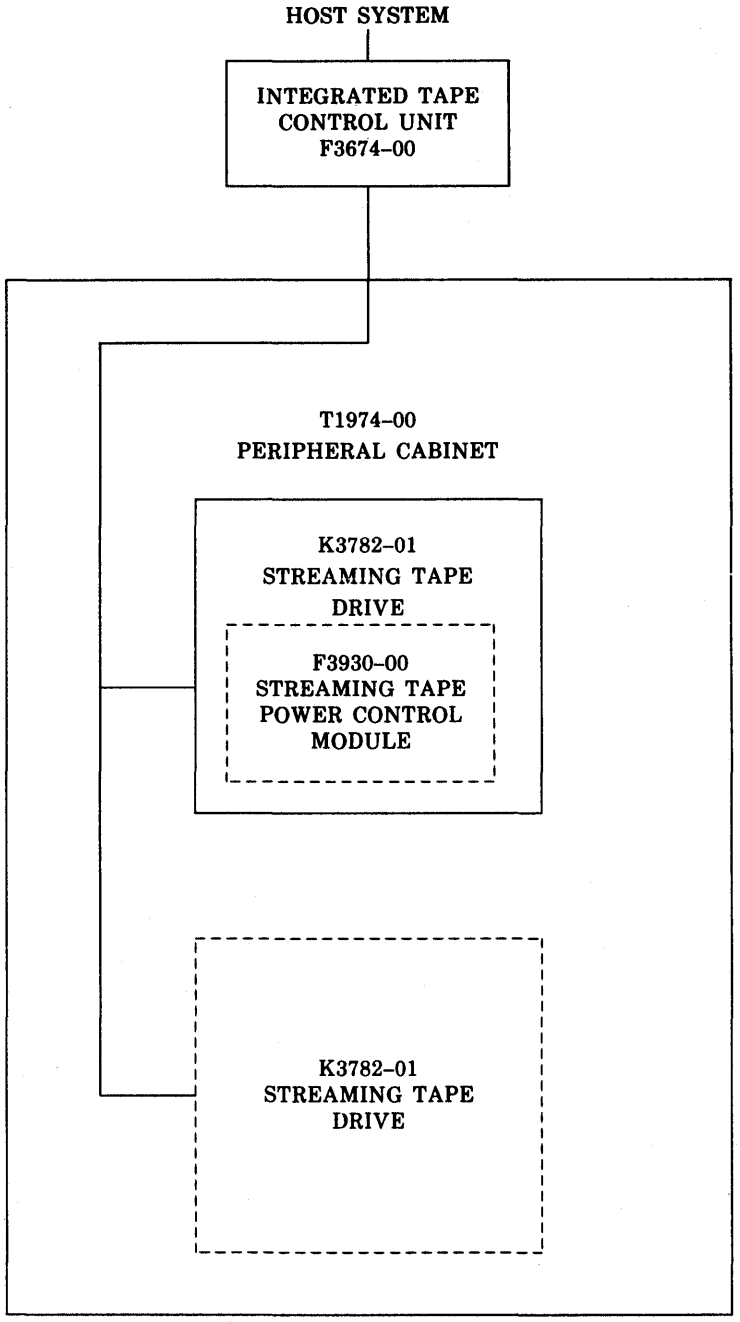
   A tape mark in 9-track Phase Encoding mode consists of a multibyte block that follows a 3.5 inch interblock gap. Assume the tracks on tape are numbered 1 through 9, beginning from the reference edge. Tape mark blocks consist of 64 to 256 flux reversals (at 3200 flux reversals per inch) in tracks 2, 5, and 8 and at least tracks 3, 6 and 9 are dc erased. If information exists in tracks 1, 4, and 7, a tape mark will still be detected.

5.2.4. File Protection

The streaming tape drive provides for file protection by means of the write-enable ring. Write operations are possible if a write-enable ring has been inserted in the supply reel.

5.3. Subsystem Configuration

Figure 5-1 shows the streaming tape configuration.



NOTES:

1. *Two K3782 drives can be mounted in cabinet T1974-00 and be controlled by one ITCU F3674-00.*
2. *One F3930-30 Power Control Module is required to power two K3782-01 drives.*
3. *This subsystem could be duplicated so that the host system can have four streaming tape drives and two ITCUs.*
4. *Each Streaming Tape has a built in Formatter.*

Figure 5-1. Streaming Tape Configuration

5.4. Formatter Control and Interface Characteristics

The streaming tape drive contains its own formatter electronics. Data is exchanged in 9-bit parallel form. Streaming tape only uses 1600 bits per inch Phase Encoded recording format. Repositioning cycles are handled automatically by internal electronics and do not require external commands.

As long as new commands are received in the same direction during the command reinstruct period, the control responds to the commands and continue to stream. If the command reinstruct period expires without a new command, automatic repositioning prepares for a new command in the same direction. Any command received during a repositioning cycle is accepted but may not be acted upon until the repositioning cycle has been completed. Any command requiring a change in direction causes a repositioning cycle to occur. On a write operation or any Erase operation command following any Read Forward operation, software must provide a stop operation to allow an automatic repositioning cycle. Any Read Forward command following any Write or Erase operation is illegal and it is the responsibility of the external software to avoid this operation.

The signal interface lines (see Figure 5-2) are:

- Data Lines

  These bidirectional lines have eight data bits and one "even" parity bit.

- Clear Line

  An active CLEAR signal causes any existing command in operation to be terminated immediately. It master clears the formatter and performs all Power On Confidence (POC) tests. At the end of POC tests, the POC completed indicator bit is set in bit 5 of first status byte in response to the next command.

  The online/offline status remains the same as it was before the CLEAR signal.

- Last Character (LC)

  The LC signal is generated by the host to indicate:

  - that the data on the bus is the last byte of a command and no data transfer takes place from the host to the peripheral device for the remaining part of the current operation, or

  - the data on the bus is the last byte of output data for the current command and an interrupt with ending status is expected from the device, or

  - the host does not accept any more data for the current input command. Interrupt with ending status is not presented until end-of-block is reached if the input is a Read command.

- Select Line

  Each device on the interface has a unique SELECT line. The presence of the signal is an enable to the device to perform an operation.

  - The leading edge of SELECT indicates to the device that a command is ready. The device puts up an OUTPUT REQUEST to receive that command. After acknowledging status, the host drops SELECT.

- The SELECT signal to the device is dropped by the host immediately after a Status byte is received and acknowledged. When the SELECT signal is dropped, it must remain off for at least 4 microseconds before it is raised again.

- The device performs only one command per Select sequence.

- If SELECT is dropped prior to the end of the status presentation sequence, the device immediately terminates any activity in progress and raises the ATTENTION signal. Upon receipt of the next command, the device immediately presents status containing the Interface Error indication.

■ Acknowledge (ACK)

The ACK signal is raised by the host in response to an OUTPUT REQUEST (OR), or an INPUT REQUEST (IR) to indicate to the device that these signals have been received.

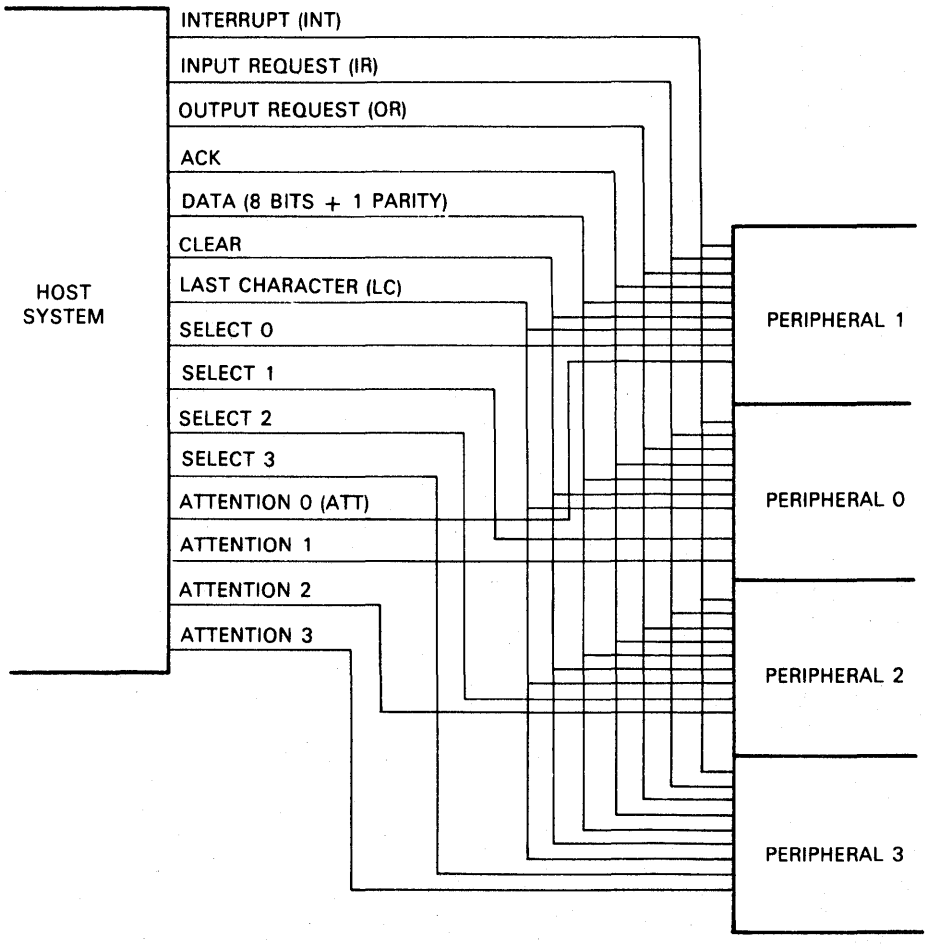


Figure 5-2. 8-Bit Bi-directional Byte Interface Signals

- Output Request (OR)

  The OR signal is raised by the device to request a command function or output data. When the host raises SELECT, the device raises OR to request a command. For an output data command the device raises OR again to indicate that the device is expecting data. OR drops with each ACK from the host and is raised again until the LC Signal is raised by the host.

- Input Request (IR)

  The IR signal is raised by the device for each byte of input data placed on the data lines. IR is dropped with each ACK signal. When IR and Interrupt are both active, the data on the data lines is status.

- Interrupt (INT)

  The INT signal is raised at the end of each command sequence. IR is raised in conjunction with INT and status is placed on the Data lines. The ACK signal in response to the last status byte causes both IR and INT to drop.

- Attention (ATT)

  The ATT line is used by the formatter to signal the host that an unsolicited status is ready to be presented. This occurs on the completion of all control commands (when ending status is ready for presentation). ATT, once asserted, remains on until the formatter has been selected and drops before INT drops during the next command sequence.

5.5. Commands

The Integrated Tape Control Unit (ITCU) responds to the commands listed in Table 5–2. Commands not listed or commands without a corresponding feature, such as NRZI in the ITCU, are invalid. Invalid commands set Command Reject. Command Reject sets the Unit Check bit in the status byte.

Table 5-2. ITCU Commands

Command	Hexadecimal Code	Binary Code
Basic Commands		
Sense	04	0000 0100
Write	01	0000 0001
Read*	02,12	000I 0010
Read Backward*	0C,1C	000I 1100
Load Translate Table	FD	1111 1101
Read Translate Table	FA	1111 1010
Control Commands		
Rewind	07	0000 0111
Rewind Unload	0F	0000 1111
Erase Gap	17	0001 0111
Write Tape Mark	1F	0001 1111
Backspace Block	27	0010 0111
Backspace File	2F	0010 1111
Forward Space Block	37	0011 0111
Forward Space File	3F	0011 1111
Data Security Erase	97	1001 0111
Set High-Speed Mode	F9	1111 1001
Reset High-Speed Mode	E9	1110 1001
Set 1600 BPI Mode	F3	1111 0011
Set Translate Mode	43	0100 0011
Diagnostic Commands		
Diagnostic Control	4B	0100 1011
Loop Write to Read	8B	1000 1011

* *When I is set to 1, Unit Check status is set if sense byte 3, bit 4 is set.*
When I is set to 0, normal operation occurs.

5.5.1. Programming Note

It is a program responsibility to avoid the execution of improper read and write sequences since the integrated streaming tape subsystem has no interlocking to prevent such improper sequences. The following two basic sequences must be avoided:

1. A Write operation after a Forward Read operation except when the block or tape mark read is known:

   - to be followed by a tape mark;
   - to have been followed by an erase gap operation when written; or
   - to have been the last block written before a Backward operation.

   Example:

   ```
   RRW*    avoid
   WBRW*   allowed
   ```

2. A Read Forward operation following Write operations.

   Example:

   ```
   RBWR*   avoid
   WBRR*   avoid
   ```

   where:

   W Indicates a Write operation: write, write tape mark, or erase gap.

   R Indicates a Forward Read operation: read forward, forward space block, or forward space file.

   B Indicates a Backward Read operation: read backward, backspace block, or backspace file.

   * *Indicates the logical record on which problems might occur.*

Because it is difficult to determine these unsafe situations, a write after a read-forward sequence should occur only in applications where strict control of format and sequence exists.

A write can follow a backspace. Assume the following tape format with labels, where * is used to denote a tape mark:

```
, VOL HDR*DATA SET*EOF*HDR*DATA SET*EOF**.
```

A rewrite of the last data set involves the following safe and proper sequence. After processing the next-to-last End Of File (EOF) and the tape mark, read forward to verify the HDR label of the last data set, backspace, write a new HDR, and rewrite the data set. If a new data set is being added, the read forward verifies the second consecutive tape mark, thus the true end of data set on this tape. A backspace, write new HDR, and rewrite the data set complete the sequence.

5.5.2. Basic Commands

The basic commands (see Table 5-2) for the ITCU unit are described in the following subsections.

5.5.2.1. Sense Command (04_{16})

The Sense command transmits up to 14 sense bytes to the Byte Bus Channel (BBC). The bits of the sense bytes specify the conditions present in the ITCU and addressed device at the time of this command. Information presented includes error and unusual conditions associated with the last command that was executed. Sense information being held in the control unit is reset by the system reset pushbutton or the acceptance of any command other than a no-op or sense command.

The tape drive interface used during a sense command retrieves tape-unit-oriented sense information. This information is available even if the addressed tape drive is not ready.

5.5.2.2. Write Command (01_{16})

The Write command moves tape forward on the selected drive and writes the data obtained from the BBC on the tape. Each byte is checked for odd vertical redundancy as it is received from the BBC. Nondata-characters used for synchronization and error checking are recorded on the tape. The block written on the tape is checked for validity as it passes under the read head. The BBC determines the volume of data written. If the End-Of-Tape (EOT) marker is encountered during the Write operation, unit exception is included during the Write operation. A subsequent Write, Write Tape Mark, or Erase operation sets unit exception if EOT has not been reset.

Upon successful completion of a write, final status contains Channel End and Device End. Unit Exception, if found in the final status, indicates an unusual condition. Final status under fault conditions contains Channel End, Device End, and Unit Check.

Tape drive motion is terminated after the readback check is complete. The readback check is completed during the deceleration time of the tape drive.

The ITCU's primary mode of operation is the streaming mode. Stop time always overruns start time gap spacing. An automatic repositioning cycle is triggered on start/stop operations. Substantial throughput overheads are required for repositioning in the interblock gap. However, a reinstruction time for the last word (write mode) signal or trailing edge of data busy (read mode) signal allows processing of subsequent motion commands without stopping tape motion. Throughput rates are significantly increased if commands occur during this reinstruction time. Reinstruction times are: 16 milliseconds for 25 ips and 4 milliseconds for 100 ips.

5.5.2.3. Read Command (02_{16} or 12_{16})

The Read command moves the tape forward on the selected tape drive. It reads the bytes of data recorded on the tape and transfers them to the BBC. Nondata characters used for synchronization and error checking are not transferred to the BBC. The integrity of each byte is checked and data errors are corrected before passing to the BBC. At the completion of reading all data within a block, that tape unit is positioned in the interblock gap immediately following the block just read. If the BBC terminates data early, the remaining bytes are not checked for parity error, data is discarded by the control unit, and the tape is positioned in the interblock gap.

Normal status for a Read command is Channel End and Device End. Unit Check is set on ending status if a data check was detected, or Unit Exception is set if the block read was a tape mark. If bit 3 in the Read command is set, and a correctable error occurs during a read operation, the ending status byte contains a Unit Check indication. A subsequent sense operation indicates no Data Check (sense byte 0, bit 4). Bit 3 should be used only as a diagnostic tool to indicate how often error corrections are occurring in the PE mode.

5.5.2.4. Read-Backward Command ($0C_{16}$ or $1C_{16}$)

The Read-Backward command reverses the direction of the tape on the selected tape drive. The bytes of data recorded on the tape are read and transferred to the BBC. Nondata characters used for synchronization and error checking are not transferred to the channel. While reading, the integrity of each byte is checked, and data may be corrected before it is passed to the BBC. At the completion of reading all the data within a block, the tape drive is positioned in the interblock gap immediately preceding the block just read. If the BBC fails to accept all the data bytes within a block, the remaining bytes are checked and discarded by the control unit and the tape is positioned in the interblock gap.

Normal ending status for a Read-Backward command is Channel End and Device End. Unit Check is set in ending status is a data check was detected, if the Read-Backward command was issued to a tape unit positioned at load point, or if load point was detected during the Read Backward operation. If bit 3 in the Read-Backward command op code is set to 1 and a correctable error occurs during the Read Backward operation, the ending status byte contains a Unit Check indication. A subsequent sense operation indicates no Data Check (sense byte 0, bit 4). Bit 3 should be used only as a diagnostic tool to indicate how often error corrections are occurring.

5.5.2.5. Load-Translate-Table Command (FD_{16})

The Load-Translate-Table command turns on translate functions for the tape drive. System clear turns off the translate tables. The controller provides loadable translate tables: 512 bytes for write translate and 512 bytes for read translate. The standard microcode load includes ASCII (controller)-to-EBCDIC (tape) translation tables.

The controller requests 1024 bytes of output data to load the write and read translate tables. During translation, data to be translated is used as an offset address into the tables. The first 512 bytes load the write translate table. The command normally terminates with Device End and Channel End. If the BBC terminates the transfer before the tables are completely loaded, Unit Check is set in the ending status, and Data Check and Translate Error sense is set.

5.5.2.6. Read-Translate-Table Command (FA_{16})

The Read-Translate-Table command turns on translate function for the tape drive. The controller provides loadable translate tables; 512 bytes for write translate and 512 bytes for read translate.

System clear turns the translate mode off. The standard microcode load includes ASCII (controller)-to-EBCDIC (tape) translation tables. This command lets the controller transmit the contents of the translate tables to the BBC. The command normally terminates with Channel End and Device End. The BBC may terminate transfer at any point.

5.5.3. Control Commands

Control commands make the addressed tape drive move tape but not transfer data between the BBC and ITCU. Control commands present two status bytes to the BBC: an initial status when the command is initiated and an asynchronous ending status after the command is executed. Control commands present Channel End in initial status, as soon as the command has been accepted by the ITCU. The BBC is then free to address other control units or other drives on the same control unit. On completion of the control command, the ITCU presents asynchronous ending status with Device End set when asynchronous status is enable by the BBC. If the controller receives another command for a device for which asynchronous ending status has not been presented to the BBC, the controller sends Busy in the status byte.

5.5.3.1. Rewind Command (07_{16})

The Rewind command rewinds the addressed tape drive to the Beginning-Of-Tape (BOT) load point. Initial status contains only Channel End status if the addressed tape drive is not ready, not busy, and not at load point. After presenting initial status, the ITCU is busy for a maximum period of a few microseconds (excluding turnaround delay if required) to initiate the Rewind command in the tape drive. Channel End and Device End is presented in final status immediately after the tape drive accepts the command. If any error is detected after Channel End and Device End has been presented, but before the tape drive has reached BOT load point, the proper sense information is set and automatically reports through the ending status of the next command.

5.5.3.2. Rewind-Unload Command ($0F_{16}$)

The Rewind-Unload command rewinds the addressed tape drive to load point, removes the tape from the tape path, and rewinds it completely onto the file reel. This leaves the tape drive in a not-ready condition until operator intervention makes the tape drive ready again. The control unit remains busy until the tape drive has accepted the command and is rewinding, which may require a turnaround delay. At this point, the control unit presents ending status of Channel End, Device End, and Unit Check.

5.5.3.3. Erase-Gap Command (17_{16})

The Erase-Gap command moves the addressed tape drive forward and passes a current through all the erase heads, resulting in a dc erasure of the tape. The formatter unit remains busy for the duration of the command. During the erase gap operation, the read detection circuitry verifies whether a complete erasure has occurred. A single erase gap erases approximately 4.2 inches of tape and subsequent erase gaps erase approximately 3.6 inches of tape for all tape speeds. Upon acceptance of the command, Channel End is provided in initial status. At the completion of the command, Device End is presented to the BBC. If the EOT marker is encountered during the Erase operation, Unit Exception is presented in ending status.

5.5.3.4. Write-Tape-Mark Command ($1F_{16}$)

The Write-Tape-Mark command lets the addressed tape drive move tape forward, erase tape for a distance equal to a single erase gap, and write a tape mark identifier appropriate to the recording mode in effect at the time the command was issued. No data is transferred during the write portion of the command. During the entire write operation, the detection circuitry verifies whether a complete erasure has occurred and whether the tape mark written is correct. At the completion of the command, Channel End and Device End status is presented. If the EOT marker is encountered during this operation, Unit Exception is presented in ending status.

In PE mode, a tape mark is 128 flux reversals at a rate of 3200 flux reversals per inch written in bit positions P, 0, 2, 5, 6, and 7. This corresponds to track positions 4, 7, 5, 1, 8, and 2, respectively.

5.5.3.5. Backspace-Block Command (27_{16})

The Backspace-Block command lets the addressed tape drive move the tape backward and positions it in the next interblock gap. The device remains busy during the period of time that the tape is being moved to the next interblock gap. No data is transferred to the BBC and no validity checking is performed on the data. Upon completing the command and positioning it in the next interblock gap, the control unit provides Device End status. The device must detect the minimal conditions for data present before the search for an interblock gap is performed. If the block backspaced over was a tape mark, Unit Exception is included in ending status. If the load point marker is detected before interblock gap, Device End, Control Unit End, and Unit Check are provided in ending status.

5.5.3.6. Backspace-File Command ($2F_{16}$)

The Backspace-File command lets the addressed tape drive move tape backward. The device searches for the next tape mark. The device remains busy for the duration of the tape mark search. No data is transferred to the BBC and no validity testing is performed on the data read. At the completion of the command, the tape is positioned in the interblock gap immediately preceding the tape mark. When the command is completed, the control unit provided Device End status. If the load point marker is sensed before a tape mark, Control Unit End and Unit Check are presented as ending status.

5.5.3.7. Forward-Space-Block Command (37_{16})

The Forward-Space-Block command lets the addressed tape drive move tape forward and positions it in the next interblock gap. The device remains busy during the period of time that tape moves to the next interblock gap. No data is transferred to the BBC and no validity testing of data is performed. The device must detect the minimal conditions for data present before it searches for an interblock gap. When the tape is positioned in the next interblock gap, the control unit provides Device End status. If the block that has been spaced over is recognized as a tape mark, Unit Exception is presented in ending status. Unit Check is presented with ending status if the tape reaches the EOT marker.

5.5.3.8. Forward-Space-File Command ($3F_{16}$)

The Forward-Space-File command lets the addressed tape drive move tape forward. The device searches for the next tape mark character and remains busy for the duration of this search. No data is transferred to the BBC and no validity check is performed on the data read. At the completion of the command, the tape is positioned in the interblock gap immediately following the next tape mark. When the command is completed, the control unit provides Device End status. No indication is presented if the tape reaches the EOT marker.np

5.5.3.9. Data-Security-Erase Command (97_{16})

The Data-Security-Erase command lets the selected tape drive erase from the point at which the operation was initiated to the EOT marker. This command is accepted by the ITCU only when chained to a preceding Erase-Gap command. Command Reject (sense byte 0, bit 0) is set with Unit Check in initial status if the Data-Security-Erase command is issued to a tape drive positioned at EOT.

Device End status is signaled at the detection of an EOT during a normal completion. However, a sense command should be issued to assure the EOT was reached. Upon the completion of this command, the streaming tape automatically erases four feet of tape beyond the EOT.

5.5.3.10. Set-High-Speed-Mode Command ($F9_{16}$)

The Set-High-Speed-Mode command sets the specified tape drive to the higher of the two speeds provided. On system reset or return to load point, the ITCU resets the device to low speed. This is a command immediate operation that returns Channel End and Device End in the initial status byte.

5.5.3.11. Reset-High-Speed-Mode Command ($E9_{16}$)

The Reset-High-Speed-Mode command sets the addressed tape drive to the lower of the two speeds provided. This is a command immediate operation that returns Channel End and Device End in the initial status byte.

5.5.3.12. Set-1600-BPI-Mode Command ($F3_{16}$)

The Set-1600-BPI-Mode command sets the selected tape drive to PE, 1600-bits per inch mode and returns Channel End and Device End immediately after the condition code/function code byte. It is effective only when the tape on the selected tape drive is situated at load point. Whenever a tape arrives at load point, the tape drive is automatically placed into PE mode.

If the Set-1600-BPI-Mode command is issued when the tape on the selected tape drive is not at load point, it is treated as a No-Operation (op code 03) command.

5.5.3.13. Set-Translate-Mode Command (43_{16})

The Set-Translate-Mode command sets the translate mode control for the specified tape drive. When this mode is set, all data written to or read from this tape unit is translated according to the previously loaded translate tables. This mode is reset by system reset or by the tape arriving at load point.

5.5.4. Diagnostic Commands

5.5.4.1. Diagnostic Control Command ($4B_{16}$)

The Diagnostic Control (Set Diagnose) command is used to transmit 2 or 9 bytes subcommand to the ITCU. It is used for placing the ITCU in diagnostic mode and executing special subcommands. When the bytes have been received Channel End and Device End status is returned to the BBC. The subcommand is not performed. Following completion of the Diagnostic

Control command, a chained read or a write command is received if any data transfer is required. At this time the subcommand is performed. The two least significant bytes of the subcommand are usually ignored. The following chart shows the two most significant bytes in hexadecimal, the function, and whether the command is followed by a read of a write command.

Subcommand	Function	Read or Write
00C4	Read Controller ID	Read
01C4	Read Device ID	Read
00F4	Read Microcode ID	Read
00F1	Load Microcode	Write
00F2	Read Microcode	Read
00D2	Read Display	Read
00F3	Start Microcode	
00D4	Read Monitor Sense	Read

If any command other than the one shown in the chart follows the Diagnostic Control command, Unit Check status is returned with Command Reject indicated in sense byte 0.

5.5.4.2. Loop-Write-to-Read Command ($8B_{16}$)

The Loop-Write-to-Read command provides a means to check the read/write data paths through the ITCU.

The Loop Write to Read command transmits a data byte to the ITCU and sets the Loop Write to Read mode. If the addressed tape drive is in translate mode, the data will pass through the translate paths. Data is not transferred to the tape drives. Successful completion of the test causes Channel End and Device End status to be returned to the BBC. A Read command can be chained to Loop Write to Read command to return the data to the BBC. The chained Read command returns the three bytes to the BBC and resets the Loop Write to Read mode. Any other command resets the Loop Write to Read mode.

Channel End and Unit Check is presented if a parity error is detected in the data path.

Read data can be compared with written data for data path chain.

5.6. Status Byte

The status byte indicates status conditions detected by the ITCU and transmitted to the BBC. Offline operations and diagnostics do not present status information to the BBC or prevent the presentation of status from an online operation. Figure 5-3 shows the status byte and Table 5-3 defines each bit.

Attention 0	Status Modifier 1	Control Unit End 2	Busy 3	Channel End 4	Device End 5	Unit Check 6	Unit Exception 7

Figure 5-3. Status Byte Format

Table 5-3. Status Byte Bit Descriptions

Name/Bit	Description
Attention 0	Presented by the ITCU to indicate that a device has become ready and the tape has reached the BOT load point. This condition occurs after the tape: ■ has been loaded at initial power on; ■ is moved to the BOT load point; and ■ is moved to BOT load point following the Rewind-Unload command.
Status Modifier 1	When set to 1 along with Busy (bit 3) and with or without Control Unit End (bit 2), indicates the control unit is busy rather than the device. Status Modifier, with any other bits set, indicates that a recovered error has occurred and auto sense follows.
Control Unit End 2	May be presented alone, or along with Control Unit Busy status, or with Status-Modifier status. It is never merged with Channel End and Device End status, which it follows if that status is to be presented. Control Unit End indicates that the control unit previously responded to a command initiation sequence with Control Unit Busy and is now available for a new command. If the ITCU receives a chained command initiation sequence prior to presenting Control Unit End, it holds the Control Unit End status until the end of the chain.
Busy 3	Indicates that the device cannot accept a command because it is executing a previously initiated operation or that a status condition is pending or stacked at a command selection time. Busy status can be presented only during a command selection sequence. If the status condition that causes a busy indication is for the addressed device, the Busy bit without Status Modifier is set along with the existing status. If the status is not for the addressed device, Status Modifier is set along with the Busy bit to indicate the control unit is busy and may also include Control Unit End (bit 2).
Channel End 4	Indicates that the transfer of data or the control information portion of an I/O operation between the channel and control unit is complete. Channel End is generated only once per I/O operation. Channel End is always set with Device End at the completion of all commands. Channel End and Device End are set for a Rewind command as soon as the command is initiated by the device.
Device End 5	Is always set along with Channel End (bit 4) at the completion of all commands.
Unit Check 6	Indicates that the control unit or tape drive detected an unusual condition and is always followed by an auto sense sequence. The conditions causing Unit Check are detailed by information available as sense data. Unit Check is set when any of the following occurs: ■ Any bit in sense byte 0 is set. ■ A read backward, backspace block, or backspace file is initiated at or into load point. ■ A Rewind Unload operation is completed at the control unit. ■ Sense byte 1, bit 7 is set (Not Compatible).

Table 5-3. Status Byte Bit Descriptions (continued)

Name/Bit	Description
	■ Sense byte 5, bit 3 is set (ID Burst Check). ■ Bit 3 of the Read or Read-Backward command is set and error correction is indicated (corrected error).
Unit Exception 7	Indicates an unusual condition that is not necessarily an error. The conditions that set this bit are: ■ End of tape (EOT, SB1-5) is set during a Write, Write-Tape-Mark, or Erase-Gap operation. ■ A tape mark is sensed during a Read, Read-Backward, Forward-Space-Block, or Backspace-Block operation. *NOTES:* *1. The tape drive sets end of tape when it senses the trailing edge of the EOT reflective marker while tape is moving forward.* *2. A subsequent write, write-tape-mark, a erase-gap command sets unit exception again with device end if end of tape has not been reset.* *3. A command that moves the tape backward, so that the tape drive again senses the trailing edge of the end-of-tape marker, resets the end of tape, thereby possibly preventing unit exception from occurring again.* *4. A Rewind or Rewind-Unload command also resets the end of tape.* *5. In Read and Space-Block operations, unit exception is set only for the tape mark read and is reset for any subsequent command. Therefore, it is important to handle a unit exception when it is recognized.*

5.7. Sense Data

Seven bytes of sense data are stored in the control unit and are available to the BBC with a Sense command. The bit positions in each byte define the conditions present in the control unit or tape drive. The information in these bytes includes error and unusual conditions associated with the last operation and provides information about the current conditions present in the control unit and tape drive. Figure 5-4 shows the format of the sense bytes, and Table 5-4 defines the sense bits in sense bytes 0 through 6.

Byte	Bit 0	1	2	3	4	5	6	7
0	COMMAND REJECT	INTER-VENTION REQUIRED	BUS-OUT CHECK	EQUIPMENT CHECK	DATA CHECK	OVERRUN	WORD COUNT ZERO	0
1	NOISE	Tape Drive STATUS A	Tape Drive STATUS B	0	LOAD POINT	0	FILE PRO-TECTED	NOT COM-PATIBLE
2	TRACK IN ERROR IF NRZI DEAD TRACK INFORMATION IF MTE OR ENV ERROR OVERSKEWED TRACK IF SKEW ERROR							
3	R/W VRC	MTE/ LRC	SKEW	END/ CRC	ENV	1600-BPI	BACKWARD	C/P COMPARE
4	RUN-AWAY	REJECT TU	TAPE INDICATE	0	TRANSLATE ERROR	LOOP WRITE TO READ	0	0
5	NO MOTION	0	WTM CHECK	ID BURST CHECK	0	PARTIAL RECORD	POST-AMBLE ERROR	0
6	FIFO ERROR	RAM PARITY ERROR	MICROCODE NOT LOADED	DATA BUS PARITY ERROR	TRANSLATOR COMP ERROR	DEVICE REG. ERROR	FORMATTER CHECK	AUX INTF ERROR

Figure 5-4. Sense Data Bytes

Table 5-4. Sense Bytes Bit Definitions

Bit	Designation	Definition
		Sense Byte 0
0	Command Reject	Set as a result of any of the following: ■ A Data-Security-Erase command is not chained to an erase-gap command. ■ A command is not recognized by the ITCU or is for a feature not installed in the ITCU. (This bit is not set if a bus-out check occurs without the appropriate feature installed.) ■ A mode-set command to an ITCU without the appropriate feature installed. ■ A received command is issued to a tape that is not preceded by the proper sequence of commands. ■ A Write, Write-Tape-Mark, Erase-Gap, or Data-Security-Erase command is issued to a file-protected tape unit. ■ Not compatible; sense byte 1, bit 7 is set.
1	Intervention Required	Set when the addressed tape drive is not ready or nonexistent. This occurs when bit 1 is not set in sense byte 1.
2	Bus-Out Check	Set when the byte on the BBC indicates incorrect (even) parity for a command of data byte. If this condition occurs during a data transfer for a Write operation, the operation is terminated and the error byte is not written on tape.
3	Equipment Check	Set as a result of any of the following: ■ Reject Tape Unit (sense byte 4, bit 1) is set. ■ Write Tape Mark operations: – The tape mark block was not detected at the read head within 0.30 inch (0.76 cm) after writing terminated. – The tape mark block was not written correctly, and backspacing is not possible without losing tape position. – An interblock gap was not detected following the tape mark. – Sense byte 4, bit 0 or 1 is set.
4	Data Check	Set for any of the following read or write operations: 1. Read/write Vertical Redundancy Check (VRC), sense byte 3, bit 0 is set.

Table 5-4. Sense Bytes Bit Definitions (continued)

Bit	Designation	Definition
		2. Multitrack error (MTE), sense byte 3, bit 1 is set. 3. Skew, sense byte 3, bit 2 is set. 4. End, sense byte 3, bit 3 is set. 5. ENVelope/Single-Track Error (ENV/STE) sense byte 3, bit 4 is set. 6. Compare, sense byte 3, bit 7 is set. 7. Partial Record, sense byte 5, bit 5 is set. 8. Write Tape Mark Check, sense byte 5, bit 2 is set. 9. Postamble Error, sense byte 5, bit 6 is set. 10. Noise, sense byte 1, bit 0 is set. 11. Hardware error occurs in the streaming tape subsystem if operations 1 through 5 cannot be deferentiated. Data Check is set with no other sense bytes. 12. ID Burst Check, sense byte 5, bit 3 is set.
5	Overrun	Set when the ITCU data transfer rate exceeds that of the BBC for a read or write operation. For a write operation, the operation is terminated, tape motion is halted, and no further data is requested from the BBC. For a read operation, data transfer is terminated and tape motion continues until an interblock gap is detected.
6	Word Count Zero	Word Count Zero is set for the following conditions: 1. A Terminate response to the first data request for a write operation. 2. The L-bus terminates the operation. 3. The first data byte for a Write operation indicates incorrect (even) parity. (Bus Out Check sense byte 0, bit 2 is also set.) Word Count Zero indicates that no tape motion has occurred.
7	Not Used	Set to 0.
Sense Byte 1		
0	Noise	Set as a result of any of the following: ■ No data was transferred on a Read or Read-Backward operation, and the block was not detected as a tape mark.

Table 5-4. Sense Bytes Bit Definitions (continued)

<table>
<tr><th>Bit</th><th>Designation</th><th>Definition</th></tr>
<tr><td></td><td></td><td>■ Data was detected during an erase-gap or during the erase portion of the Write-Tape-Mark operation.
■ A data check condition occurs during a read or read-backward operation.
■ A gap was detected in data read which exceeded 2.5 frame times during a write, write tape mark, read, read backward, space block operation in NRZI.</td></tr>
<tr><td>1</td><td>Tape Drive Status A</td><td>Set if the addressed tape drive selected is ready or not busy.</td></tr>
<tr><td>2</td><td>Tape Drive Status B</td><td>Set if the addressed tape drive is not ready or is ready but rewinding (busy).

The following shows the tape drive status and initial selection response for the various conditions of tape drive status A and B.

<table>
<tr><th>Tape Drive Status A</th><th>Tape Drive Status B</th><th>Tape Drive Status</th><th>Initial Selection Response</th></tr>
<tr><td>0</td><td>0</td><td>Nonexistent</td><td>Unit Check*</td></tr>
<tr><td>0</td><td>1</td><td>Not ready</td><td>Unit Check*</td></tr>
<tr><td>1</td><td>0</td><td>Available</td><td>Clear Status</td></tr>
<tr><td>1</td><td>1</td><td>Busy or rewinding</td><td>Busy</td></tr>
</table>
* Sense byte 0, bit 1 is also set.</td></tr>
<tr><td>3</td><td>Not used</td><td>Set to 0.</td></tr>
<tr><td>4</td><td>Load Point</td><td>Set if the addressed tape drive is positioned at the beginning of tape.</td></tr>
<tr><td>5</td><td>Not used</td><td>Set to 0.</td></tr>
<tr><td>6</td><td>File Protected</td><td>Set when the addressed tape drive does not have the write enable ring installed on the file reel.</td></tr>
<tr><td>7</td><td>Not Compatible</td><td>Set if the tape drive or ITCU features are not compatible with the data format of the tape being read. For example, no ID Burst is detected on a Read operation from load point or tape unit does not have NRZI feature.</td></tr>
</table>

Table 5-4. Sense Bytes Bit Definitions (continued)

Bit	Designation	Definition		
		Sense Byte 2		
0-7	Track In Error byte	PE mode		Indicates the dead track register information for both Read and Write operations.
		NRZI mode		TIE byte indicates a single track in error. Bits 6 and 7 with Data Check indicates a multiple track error. Normal operation is indicated by bits 6 and 7 without Data Check. Data Check and zeros in bit 0-7 indicate a D-bit error.
		Sense Byte 3		
0	R/W VRC	PE mode		Set for Read or Write operations when the data byte indicates incorrect parity and there are no phase errors or dead tracks.
		NRZI mode		Set for Read type operation if the data byte indicates incorrect parity.
1	MTE/LRC	MTE/LRC is set for the following conditions:		
		NRZI mode	1 2	WTM Check, sense byte 5, bit 2. or an LRC error.
		PE mode	3	Any combination of two or more dead tracks, or phase error conditions for Read or Write operations.
2	Skew	PE mode		Set because of excessive skew on Read, Read Backward or Write operations.
		NRZI mode		Set if the bit spacing within a byte exceeds 25% of nominal bit cell time for write operations.
3	END/CRC	PE mode	1.	Set if beginning of postamble was not detected.
			2.	Set if beginning of postamble was detected before or after the actual end of data from a Read operation.
			3.	Set if an interblock gap is detected before the end of data during a Read operation.
		NRZI mode		Set if a CRC error is detected during Read or Write operations.

Table 5-4. Sense Bytes Bit Definitions (continued)

Bit	Designation	Definition
4	ENV	PE mode – ■ Set if the interblock gap is detected while writing data for Write operations. ■ Set if an error correction was required for Read or Read-Backward operations. Data check is not set for this condition. ■ Set if the tape mark block was not written correctly but can be backspaced without losing tape position from all WTM operations.
5	1600 bpi	Set to indicate addressed tape drive is in PE mode 1600-bpi.
6	Backward	Set to indicate addressed tape drive is in backward mode.
7	C/P Compare	Set by data checks internal to the controller.
	Sense Byte 4	
0	Runaway	Set if data is not detected within 25 feet for a Read operation.
1	Reject TU	Set as a result of any of the following: ■ Runaway, sense byte 1, bit 0 is set. ■ No data detected within 0.30 inch (0.76 cm) after writing terminated for a Write or Write Tape Mark operation. ■ The tape drive drops ready during execution of a command. ■ The tape drive changed from write to read status during a write operation or from read to write status during a Read operation. ■ An interblock gap not detected following an ID burst written after load point.
2	Tape Indicate	Set when EOT marker is sensed during forward tape motion.
3	Not used	Always Zero.
4	Translate Error	Set if translate table is not loaded or if Translate Mode selecting table cannot be performed.
5	Loop Write to Read	Set when a Loop-Write-to-Read command is executed.
6-7	Not used	Always Zeros.

Table 5-4. Sense Bytes Bit Definitions (continued)

Bit	Designation	Definition
Sense Byte 5		
0	No Motion	Set when a command which should cause tape motion results in no motion.
1	Not used	Always zero.
2	WTM Check	Set when the tape mark is not written properly but can be backspaced without loading tape position.
3	ID Burst Check	Set if an error exists in the ID burst when written after the load point.
4	Not used	Set to 0.
5	Partial Record	PE mode Set when an interblock gap is detected before the end of data for read operations.
6	Postamble Error	Set when the postamble exceeds its Read operation boundary.
7	Not used	Always zero.
Sense Byte 6		
0	FIFO Error	Set if First In First Out (FIFO) control or parity error occurred.
1	RAM Parity Error	Set with Unit Check on the first command after Power On Confidence (POC) test if a RAM parity was detected during POC tests.
2	Microcode Not Loaded	Set if RAM microcode is not loaded.
3	Data Bus Parity Error	Set if a parity error is detected on the internal data bus.
4	Translator Comparator Error	Set when translator has comparator error.
5	Device Register Error	Set with Unit Check on the first device related command after POC test if a device register error was detected during POC tests.
6	Formatter Check	Set if formatter does not respond to an access by the ITCU.
7	Auxiliary Interface Error	Set if error occurs on auxiliary interface.

5.8. Invalid Command Sequences

5.8.1. Commands Executed prior to Load Microcode

After power up or system clear, and prior to executing load microcode, the following lists the order for properly executing the command sequence:

- Diagnostic Control ($4B_{16}$)
- Sense (04_{16})
- No Op (03_{16})

Attempts to execute any other commands before loading microcode makes the ITCU present Command Reject sense with Unit Check status.

5.8.2. Translate Sequences

Attempts to use the translate functions before loading the translator after a power–on clear results in Command Reject sense with Unit Check status.

Appendix A. Instructions Listed by Function Code

This appendix lists the basic and extended mode instructions by function code. Basic-mode only instructions are indicated by "(BM)" in the mnemonic (fourth) column of the table. Extended-mode only instructions are indicated by "(EM)". All other instructions are used in both modes.

The abbreviations and symbols in this appendix are defined in Appendix C.

An explanation of each table heading follows:

Heading	Explanation
f	The major function code (f-field) of the instruction (bits 0 through 5).
j	The defined j-field (bits 6 though 9) values for the indicated f-field value. If a range of values is indicated, the instruction will operate only on the partial words indicated by that range. A blank j column indicates that the full range (00 though 17_8) of partial word or immediate operands are defined.
a	The defined a-field (bits 10 through 13) value for the indicated f-field and j-field values. A blank 'a' column indicates that the full range (00-17_8) of values are available for GRS register selection.
Mnemonic	The instruction mnemonic defined for the indicated function code and IP mode, BM indicates Basic mode and EM indicates Extended mode.
Instruction	The full name of the instruction.
Description	Briefly describes instruction operation. The function codes that perform no operation, are undefined, or result in a invalid instruction interrupt are indicated in this column as follows: * This operation code is not defined, and an invalid instruction interrupt occurs.

** No operation is performed, but index incrementation and indirection, if specified, occurs normally. If f=22_8 (BT), register R1 is decreased to 0.

*** This operation code is not defined, but an invalid instruction interrupt will not occur (processor privilege level is verified) and results are undefined.

NOTE: A detailed description of each instruction is given in the Assembly Instruction Mnemonic (AIM) Supplementary Reference, UP-9047 (see Preface).

Instructions Listed by Function Code

f	j	a	Mnemonic	Instruction	Description
00					*
01	00-15		SA	Store A	$(A_a) \rightarrow U$
01	16		PRBA	Probe A	Causes designated hardware point A to execute a trigger pulse. Used to trigger external monitor.
01	17		PRBB	Probe B	Causes designated hardware point B to execute a trigger pulse. Used to trigger external monitors.
02	0-15		SNA	Store Negative A	$-(A_a) \rightarrow U$
02	16,17				**
03	00-15		SMA	Store Magnitude A	$\lvert (A_a) \rvert \rightarrow U$
03	16,17				**
04	00-15		SR	Store R	$(R_a) \rightarrow U$
04	16,17				**
05	00-15	00	SZ	Store Zero	Stores constant 000000 000000, zeros, in location specified by operand address
05	16,17	00			**
05	00-15	01	SNZ	Store Negative Zero	Stores constant 777777 777777, negative zero, in location specified by operand address
05	16,17	01			**
05	00-15	02	SP1	Store Positive One	Stores constant 000000 000001, positive one, in location specified by operand address

Instructions Listed by Function Code (continued)

f	j	a	Mnemonic	Instruction	Description
05	16,17	02			**
05	00-15	03	SN1	Store Negative One	Stores constant 777777 777776, negative one, in location specified by operand address
05	16,17	03			**
05	00-15	04	SFS	Store Fieldata Spaces	Stores constant 050505 050505, Fieldata spaces, in location specified by operand address
05	16,17	04			**
05	00-15	05	SFZ	Store Fieldata Zeros	Stores constant 606060 606060, Fieldata zeros, in location specified by operand address
05	16,17	05			**
05	00-15	06	SAS	Store ASCII Spaces	Stores constant 040040 040040, ASCII spaces, in location specified by operand address
05	16,17	06			**
05	00-15	07	SAZ	Store ASCII Zeros	Stores constant 060060 060060, ASCII zeros, in location specified by operand address
05	16,17	07			**
05	00-15	10	INC	Increase Operand by One	Increases operand by one under storage lock. If initial operand or result is zero, execute NI; if not zero, skip NI
05	16,17	10			**
05	00-15	11	DEC	Decrease Operand by One	Decreases operand by one under storage lock. If initial operand or result is zero, execute NI; if not zero, skip NI
05	16,17	11			**
05	00-15	12	INC2	Increase Operand by Two	Increases operand by two under storage lock. If initial operand or result is zero, execute NI; if not zero, skip NI
05	16,17	12			**
05	00-15	13	DEC2	Decrease Operand by Two	Decreases operand by two under storage lock. If initial operand or result is zero, execute NI; if not zero, skip N

Instructions Listed by Function Code (continued)

f	j	a	Mnemonic	Instruction	Description
05	16,17	13			**
05	00–15	14	ENZ	Eliminate Negative Zero	Increases operand by zero. If initial operand or result is zero execute NI; if not zero, skip NI
05	16,17	14			**
05	00–15	15	ADD1	Add One to Storage	Adds one to operand; goes to NI
05	16,17	15			*
05	00–15	16	SUB1	Subtract One from Storage	Subtracts one from operand; goes to NI
05	16,17	16			**
05		17			*
06	00–15		SX	Store X	$(X_a) \rightarrow U$
06	16,17				**
07	00		ADE	Add Decimal	Adds (U) to an A-register and stores result in A-register
07	01		DADE	Double Add Decimal	Adds two-word operand (U, U+1) to two A-registers (A_a, A_{a+1}) and stores results in A_a and A_{a+1}
07	02		SDE	Subtract Decimal	Subtracts (U) from an A-register and stores result in A-register
07	03		DSDE	Double Subtract Decimal	Subtracts two-word operand from A-register (A_a, A_{a+1}) and stores result in A_a and A_{a+1}
07	04		LAQW	Load A Quarter Word	(Selected quarter word) $\rightarrow A_{a27-35}$
07	05		SAQW	Store A Quarter Word	$(A_a)_{27-35} \rightarrow$ Selected quarter word
07	06		DEI	Decimal to Integer	Converts (U) from signed magnitude format to a ones complement binary format and stores in A-register
07	07		DDEI	Double Decimal to Integer	Converts (U, U+1) from decimal to ones complement binary format and stores in A_a and A_{a+1}

Instructions Listed by Function Code (continued)

f	j	a	Mnemonic	Instruction	Description
07	10		IDE	Integer to Decimal	Converts (U) in ones complement to decimal format, stores in a pair of A-registers
07	11		DIDE	Double Integer to Decimal	Converts (U, U+1) from ones complements to decimal, stores in A_a, A_{a+1}, A_{a+2}
07	12		LDJ (BM)	Load D-Bank Base and Jump	Ignores $X_{a\ 1-2}$; if DB31 is zero; B14 is selected; if DB31 is one, B15 is selected; BDI from $X_{a\ 6-17}$ Jump to U
07	13		LIJ (BM)	Load I-Bank Base and Jump	Ignores $X_{a\ 1-2}$; if DB31 is zero, B12 is selected; if DB31 is one, B13 is selected; BDI from $X_{a\ 6-17}$ Jump to U
07	14		LPD (BM)	Load Program Control Designator	U bits 11,12,14,16 → Designator Register; bit 11 → DB29 bit 12 → DB30 bit 14 → DB32 bit 16 → DB34
07	15		SPD (BM)	Store Program Control Designator	Designator Register Bit → U_{27-35} DB27-35 → Bit 27-35
07	12-15		(EM)		*
07	16		(BM)		*
07	16	00	LOCL (EM)	Local Procedure Call	Identical to CALL, except that the target is the instruction operand address ($U+X_m$) within the bank currently based on B0.
07	16	01-12	(EM)		***
07	16	13	CALL (EM)	Procedure Call	Captures essential activity environment on the return control stack, such that a subsequent return to the original environment can be done by a RTN instruction
07	16	14-17	(EM)		***
07	17	00	(BM)		*
07	17	01-17	LBJ (BM)	Load Bank and Jump	Loads L,BDI specified by $X_{a\ 0,3,6}$ - 17 to B12-B14 selected by $X_{a\ 1,2}$ and jump to U. If a = 0, an invalid instruction interrupt occurs.
07	17	00	GOTO (EM)	GO TO	This instruction is the extended mode counterpart of LBJ in terms of instruction bank switching, and it operates in a similar fashion except that there is no link register specified

Instructions Listed by Function Code (continued)

f	j	a	Mnemonic	Instruction	Description
10			LA	Load A	$(U) \rightarrow A_a$
11			LNA	Load Negative A	$-(U) \rightarrow A_a$
12			LMA	Load Magnitude A	$\lvert (U) \rvert \rightarrow A_a$
13			LNMA	Load Negative Magnitude A	$-\lvert (U) \rvert \rightarrow A_a$
14			AA	Add to A	$(A_a) + (U) \rightarrow A_a$
15			ANA	Add Negative to A	$(A_a) - (U) \rightarrow A_a$
16			AMA	Add Magnitude to A	$(A_a) + \lvert (U) \rvert \rightarrow A_a$
17			ANMA	Add Negative Magnitude to A	$(A_a) - \lvert (U) \rvert \rightarrow A_a$
20			AU	Add Upper	$(A_a) + (U) \rightarrow A_{a+1}$
21			ANU	Add Negative Upper	$(A_a) - (U) \rightarrow A_{a+1}$
22	00–15		BT	Block Transfer	$(X_x + u) \rightarrow X_a + u$; repeat k times
22	16,17				**
23			LR	Load R	$(U) \rightarrow R_a$
24			AX	Add to X	$(X_a) + (U) \rightarrow X_a$
25			ANX	Add Negative to X	$(X_a) - (U) \rightarrow X_a$
26			LXM	Load X Modifier	$(U) \rightarrow X_{a\ 18-35}$; $X_{a\ 0-17}$ unchanged
27			LX	Load X	$(U) \rightarrow X_a$
30			MI	Multiply Integer	$(A_a) \bullet (U) \rightarrow A_a, A_{a+1}$
31			MSI	Multiply Single Integer	$(A_a) \bullet (U) \rightarrow A_a$
32			MF	Multiply Fractional	$(A_a) \bullet (U) \rightarrow A_a, A_{a+1}$, left circular one bit
33			(BM)		*
33	00		FDA (EM)	Floating Decimal Add	$(A_a, A_{a+1}, A_{a+2}) + (U, U+1, U+2) \rightarrow A_a, A_{a+1}, A_{a+2}$
33	01		FDAN (EM)	Floating Decimal Add Negative	$(A_a, A_{a+1}, A_{a+2}) - (U, U+1, U+2) \rightarrow A_a, A_{a+1}, A_{a+2}$

Instructions Listed by Function Code (continued)

f	j	a	Mnemonic	Instruction	Description
33	02		FDM (EM)	Floating Decimal Multiply	$(A_a,A_{a+1},A_{a+2}) \bullet (U,U+1,U+2) \rightarrow A_a,A_{a+1},A_{a+2}$
33	03		FDD (EM)	Floating Decimal Divide	$(A_a,A_{a+1},A_{a+2}) \div (U,U+1,U+2) \rightarrow A_a,A_{a+1},A_{a+2}$
33	04		DESF (EM)	Decimal Scale and Float	$(A_a),A_{a+1},A_{a+2} \bullet 10^{u(\text{including bias})} \rightarrow A_a,A_{a+1},A_{a+2}$
33	05		FDSP (EM)	Floating Decimal Scale and Pack	$\lvert(A_a,A_{a+1},A_{a+2})\rvert \div 10^{u(\text{including bias}}$ right justified with zero fill $\rightarrow A_a,A_{a+1},A_{a+2}$ $^{\text{bits 0-31}}$ if $(A_a)_0 = 0$ then $14_8 \rightarrow A_{a+2}$ $^{\text{bits 32-35}}$ else $15_8 \rightarrow A_{a+2}$ $^{\text{bits 32-35}}$
33	06		FDR (EM)	Floating Decimal Round	$(A_a,A_{a+1},A_{a+2}) + (-1)^{(A_a)0}$ X.5X10 $^{(\text{explAa1})-u}$ (truncate to U digits) $\rightarrow A_a,A_{a+1},A_{a+2}$
33	07		FDT (EM)	Floating Decimal Truncate	Signs $\rightarrow [A_a,A_{a+1},A_{a+2}]$ (12+4*U)-107
33	08-17		(EM)		*
34			DI	Divide Integer	(A_a,A_{a+1}) divided by (U) $\rightarrow A_a$; REMAINDER $\rightarrow A_{a+1}$
35			DSF	Divide Single Fractional	[(A_a), 36 sign bits) right algebraic shift 1 place] divided by (U) $\rightarrow A_{a+1}$
36			DF	Divide Fractional	[(A_a,A_{a+1}) right algebraic shift 1 place] divided by (U) $\rightarrow A_a$; REMAINDER $\rightarrow A_{a+1}$
37	00		LRD (EM)	Load Real Dayclock	Loads clock register with contents of bits 15–66 of A_a, A_{a+1}
37	01		RRD (EM)	Read Real Dayclock	Transfers clock register to bits 15–71 of A_a, A_{a+1}, bits 0–14 are cleared to zeros
37	02		LRC (EM)	Load Real Comparator	Loads comparator register with contents of bits 15–66 of A_a, A_{a+1}
37	03		RRC	Read Real Comparator	Transfer the comparator component of the dayclock to bits 15–66 of A_a, A_{a+1}
37	04	00	SMD (EM)	Select Master Dayclock	Selects the specified real Dayclock is selected as the Master Dayclock in the IP on which instruction is executed
37	04	01	SDMN (EM)	Set Dayclock Mode Normal	Sets Dayclock to normal speed mode
37	04	02	SDMF (EM)	Set Dayclock Mode Fast	Sets Dayclock to fast speed mode

Instructions Listed by Function Code (continued)

f	j	a	Mnemonic	Instruction	Description
37	04	03	SDMS (EM)	Set Dayclock Mode Slow	Sets Dayclock to slow speed mode
37	04	04–17			***
37	05		RSS	Read System Status	Stores 72 bits of system status (load path, partitioning, and auto-recovery information) into A_a, A_{a+1}
37	06	00	SLP0 (EM)	Select Load Path 0	Directs MSU with Load Path 0 to control auto-recovery function using Load Path 0
37	06	01	SLP1 (EM)	Select Load Path 1	Directs MSU with Load Path 1 to control auto-recovery function using Load Path 1
37	06	02–17			***
37	07				*
37	10		BIM (BM)	Bit Move	Moves a source string of bits which starts on any bit boundary to a destination string which also starts on any bit boundary
37	11		BIC (BM)	Bit Compare	Compares a source string of bits to a destination string of bits
37	12		BMTC (BM)	Bit Move with Translation and Control	Moves a source string of characters to a destination string of characters
37	13		BICL (BM)	Bit Compare Long	Compares a source string of characters to a destination string
37	14		BIML (BM)	Bit Move Long	Moves a source string of bits which starts on any bit boundary to a destination which also starts on any bit boundary
37	15		BDE (BM)	Byte to Decimal	Converts a string of either ASCII or External Comp 3 characters to a signed magnitude format, results stored in A_a, A_{a+1}, A_{a+2}
37	16		DEB (BM)	Decimal to Byte	Converts a string of decimal characters in BCD signed magnitude format to either ASCII or External Comp 3 format
37	17		EDDE (BM)	Edit Decimal	Moves a source string to a destination string
37	10–17		(EM)		*

Instructions Listed by Function Code (continued)

f	j	a	Mnemonic	Instruction	Description
40			OR	Logical OR	(A_a) OR (U) → A_{a+1}
41			XOR	Logical Exclusive OR	(A_a) XOR (U) → A_{a+1}
42			AND	Logical AND	(A_a) AND (U) → A_{a+1}
43			MLU	Masked Load Upper	[(U) AND (R2)] OR [(A_a) AND NOT (R2)] → A_{a+1}
44			TEP	Test Even Parity	Skips NI if (U) AND (A_a) has even parity
45			TOP	Test Odd Parity	Skips NI if (U) AND (A_a) has odd parity
46			LXI	Load X Increment	(U) → $(X_a)_{0-17}$; $(X_a)_{18-35}$ unchanged
47			TLEM	Test Less Than or Equal to Modifier	Skips NI if $(U)_{18-35} \leq (X_a)_{18-35}$; always $(X_a)_{18-35}$ + $(X_a)_{0-17}$ → $X_{a\ 18-35}$
			TNGM	Test Not Greater Than Modifier	Alternate mnemonic and name for preceding instruction
50			TZ (BM)	Test Zero	Skips NI if (U) = ±0
50		00	TNOP (EM)	Test No Operation	If the a-field is zero no condition will satisfy the test and the next instruction will always be executed, however, the operand of this instruction is always fetched from storage
50		01	TGZ (EM)	Test Greater Than Zero	Skips NI if (U) > +0
50		02	TPZ (EM)	Test Positive Zero	Skips NI if (U) = +0
50		03	TP (EM)	Test Positive	Skips NI if (U) ≥ +0
50		04	TMZ (EM)	Test Minus Zero	Skips NI if (U) = -0
50		05	TMZG (EM)	Test Minus Zero or Greater Than Zero	Skips NI if (U) = -0 or (U) > +0
50		06	TZ (EM)	Test Zero	Skips NI if (U) = -0 or (U) = +0
50		07	TNLZ (EM)	Test Not Less Than Zero	Skips NI if (U) ≤ -0
50		10	TLZ (EM)	Test Less Than Zero	Skips NI if (U) < -0

Instructions Listed by Function Code (continued)

f	j	a	Mnemonic	Instruction	Description
50		11	TNZ (EM)	Test Not Zero	Skips NI if (U) ≠ −0 or (U) ≠ +0
50		12	TPZL (EM)	Test Positive Zero or Less Than Zero	Skips NI if (U) = +0 or (U) < −0
50		13	TNMZ (EM)	Test Not Minus Zero	Skips NI if (U) ≠ −0
50		14	TN (EM)	Test Negative	Skips NI if (U) ≤ −0
50		15	TNPZ (EM)	Test Not Positive Zero	Skips NI if (U) ≠ +0
50		16	TNGZ (EM)	Test Not Greater Than Zero	Skips NI if (U) ≥ +0
50		17	TSKP (EM)	Test - Always Zero	If the a-field is equal to 17_8, all conditions satisfy the test and the next instruction will always be skipped, however, the operand of this instruction is always fetched from storage
51			TNZ (BM)	Test Nonzero	Skips NI if (U) ≠ ±0
51			LXSI (EM)	Load X Register Short Increment	Transfers rightmost 12 bits of operand to leftmost 12 bits of X_a
52			TE	Test Equal	Skips NI if (U) = (A_a)
53			TNE	Test Not Equal	Skips NI if (U) ≠ (A_a)
54			TLE	Test Less Than or Equal	Skips NI if (U) ≤ (A_a)
			TNG	Test Not Greater	Alternate mnemonic and name for preceding instruction
55			TG	Test Greater	Skips NI if (U) > (A_a)
56			TW	Test Within Range	Skips NI if (A_a) < (U) ≤ (A_{a+1})
57			TNW	Test Not Within Range	Skips NI if (U) ≤ (A_a) or (U) > (A_{a+1})
60			TP (BM)	Test Positive	Skips NI if $(U)_0 = 0$
61			TN (BM)	Test Negative	Skips NI if $(U)_0 = 1$

Instructions Listed by Function Code (continued)

f	j	a	Mnemonic	Instruction	Description
62			SE (BM)	Search Equal	Skips NI if (U) = (A_a), else repeat
63			SNE (BM)	Search Not Equal	Skips NI if (U) ≠ (A_a), else repeat
64			SLE (BM)	Search Less Than or Equal	Skips NI if (U) ≤ (A_a), else repeat
			SNG (BM)	Search Not Greater	Alternate mnemonic and name for preceding instruction
65			SG (BM)	Search Greater	Skips NI if (U) > (A_a), else repeat
66			SW (BM)	Search Within Range	Skips NI if (A_a) < (U) ≤ (A_{a+1}), else repeat
67			SNW (BM)	Search Not Within Range	Skips NI if (U) ≤ (A_a) or (U) > (A_{a+1}), else repeat
60 – 67			(EM)		*
70			JGD	Jump Greater and Decrement	Jumps to U if $(\text{General Register})_{ja}$ > 0; goes to NI if $(\text{General Register})_{ja}$ ≤ 0; always $(\text{General Register})_{ja}$ − 1 → $\text{General Register}_{ja}$
71	00		MSE (BM)	Masked Search Equal	Skips NI if (U) AND (R2) = (A_a) AND (R2), else repeat
71	01		MSNE (BM)	Masked Search Not Equal	Skips NI if (U) AND (R2) ≠ (A_a) AND (R2), else repeat
71	02		MSLE (BM)	Masked Search Less Than or Equal	Skips NI if (U) AND (R2) ≤ (A_a) AND (R2), else repeat
			MSNG (BM)	Masked Search Not Greater	Alternate mnemonic and name for preceding instruction
71	03		MSG (BM)	Masked Search Greater	Skips NI if (U) AND (R2) > (A_a) AND (R2), else repeat
71	04		MSW (BM)	Masked Search Within Range	Skips NI if (A_a) AND (R2) < (U) AND (R2) ≤ (A_{a+1}) AND (R2), else repeat
71	05		MSNW (BM)	Masked Search Not Within Range	Skips NI if (U) AND (R2) ≤ (A_a) AND (R2) or (U) AND (R2) > (A_{a+1}) AND (R2), else repeat
71	06		MASL (BM)	Masked Alphanumeric Search Less Than or Equal	Skips NI if (U) AND (R2) ≤ (A_a) AND (R2), else repeat

Instructions Listed by Function Code (continued)

f	j	a	Mnemonic	Instruction	Description
71	07		MASG (BM)	Masked Alphanumeric Search Greater	Skips NI if (U) AND (R2) > (A_a) AND (R2), else repeat
71	00		MTE (EM)	Masked Test Equal	Skips NI if (U) AND (R2) = (A_a) AND (R2)
71	01		MTNE (EM)	Masked Test Not Equal	Skips NI if (U) AND (R2) ≠ (A_a) AND (R2)
71	02		MTLE (EM)	Masked Test Less Than or Equal	Skips NI if (U) AND (R2) ≤ (A_a) AND (R2)
			MTNG (EM)	Masked Test Not Greater	Alternate mnemonic and name for preceding instruction
71	03		MTG (EM)	Masked Test Greater	Skips NI if (U) AND (R2) > (A_a) AND (R2)
71	04		MTW (EM)	Masked Test Within Range	Skips NI if (A_a) AND (R2) < (U) AND (R2) ≤ (A_{a+1}) AND (R2)
71	05		MTNW (EM)	Masked Test Not Within Range	Skips NI if (U) AND (R2) ≤ (A_a) AND (R2) or (U) AND (R2) > (A_{a+1}) AND (R2)
71	06		MATL (EM)	Masked Alphanumeric Test Less Than or Equal	Skips NI if (U) AND (R2) ≤ (A_a) AND (R2)
71	07		MATG (EM)	Masked Alphanumeric Test Greater	Skips NI if (U) AND (R2) > (A_a) AND (R2)
71	10		DA	Double-Precision Fixed-Point Add	(A_a, A_{a+1}) + (U, U+1) → A_a,A_{a+1}
71	11		DAN	Double-Precision Fixed-Point Add Negative	(A_a, A_{a+1}) - (U, U+1) → A_a, A_{a+1}
71	12		DS	Double Store A	(A_a, A_{a+1}) → U, U+1
71	13		DL	Double Load A	(U, U+1) → A_a, A_{a+1}
71	14		DLN	Double Load Negative A	- (U, U+1) → A_a, A_{a+1}
71	15		DLM	Double Load Magnitude A	| (U, U+1) | → A_a, A_{a+1}

Instructions Listed by Function Code (continued)

f	j	a	Mnemonic	Instruction	Description
71	16		DJZ	Double-Precision Jump Zero	Jumps to U if $(A_a, A_{a+1}) = \pm 0$; goes to NI if $(A_a, A_{a+1}) \neq \pm 0$
71	17		DTE	Double-Precision Test Equal	Skips NI if $(U, U+1) = (A_a, A_{a+1})$
72	00				*
72	01		SLJ (BM)	Store Location and Jump	Relative $P+1 \rightarrow U_{18-35}$; jump to U+1
72	01		(EM)		*
72	02		JPS	Jump Positive and Shift	Jump to U if $(A_a)_0 = 0$; goes to NI if $(A_a)_0 = 1$; always shift (A_a) left circularly one bit position
72	03		JNS	Jump Negative and Shift	Jumps to U if $(A_a)_0 = 1$; goes to NI if $(A_a)_0 = 0$; always shift (A_a) left circularly one bit position
72	04		AH	Add Halves	$(A_a)_{0-17} + (U)_{0-17} \rightarrow (A_a)_{0-17}$; $(A_a)_{35-18} + (U)_{35-18} \rightarrow A_{a\ 35-18}$
72	05		ANH	Add Negative Halves	$(A_a)_{0-17} - (U)_{0-17} \rightarrow (A_a)_{0-17}$; $(A_a)_{18-35} - (U)_{18-35} \rightarrow A_{a\ 18-35}$
72	06		AT	Add Thirds	$(A_a)_{0-11} + (U)_{0-11} \rightarrow A_{a0-11}$;$(A_a)_{12-23} + (U)_{12-23} \rightarrow A_{a\ 12-23}$;$(A_a)_{24-35} + (U)_{24-35} \rightarrow A_{a\ 24-35}$
72	07		ANT	Add Negative Thirds	$(A_a)_{0-11} - (U)_{0-11} \rightarrow A_{a0-11}$;$(A_a)_{12-23} - (U)_{12-23} \rightarrow A_{a\ 12-23}$;$(A_a)_{24-35} - (U)_{24-35} \rightarrow A_{a\ 24-34}$
72	10		EX (BM)	Execute	Executes the instruction at U
72	11		ER (BM)	Executive Request	Generates signal interrupt with Executive Request status
72	10		BDE (EM)	Byte to Decimal	Converts a string of either ASCII or External Comp 3 characters to a signed magnitude format, results stored in A_a, A_{a+1}, A_{a+2}
72	11		DEB (EM)	Decimal to Byte	Converts a string of decimal characters in BCD signed magnitude format to either ASCII or External Comp 3 format
72	12		BN	Bit Normalize	$(36_{10} \bullet X_{a\ 12-35} + X_{a\ 0-5} + \text{signed } U_{0-35})/36_{10} \rightarrow$ result and remainder
72	13		(BM)		*

Instructions Listed by Function Code (continued)

f	j	a	Mnemonic	Instruction	Description
72	13		THC (EM)	Test Hash Code	Skips NI if bits 0–11 of X_a are 0. Skip NI if bits 0–11 of X_a are not 0 and equal to operand. If unequal, execute NI
72	14		BBN	Byte to Bit Normalize	$(36_{10} \bullet X_{a\ 12-35} + 9 \bullet (X_{a\ 4-5} +$ signed $U_{35-0})/36_{10}$ → result and remainder
72	15		TRA	Test Relative Address	Determines whether a relative address specified in X_a is within the given relative addressing range
72	16		SRS	Store Register Set	Transfers GRS areas defined in A_a to consecutive storage starting at address U
72	17		LRS	Load Register Set	Transfers from consecutive storage, starting at location U, to GRS areas defined in A_a
73	00		SSC	Single Shift Circular	Shifts (A_a) right circularly *n* places → A_a
73	01		DSC	Double Shift Circular	Shifts (A_a, A_{a+1}) right circularly *n* places → A_a, A_{a+1}
73	02		SSL	Single Shift Logical	Shifts (A_a) right *n* places, zero fill
73	03		DSL	Double Shift Logical	Shifts (A_a, A_{a+1}) right *n* places A_a, A_{a+1} zero fill vacated bits
73	04		SSA	Single Shift Algebraic	Shifts (A_a) right *n* places → A_a sign fill
73	05		DSA	Double Shift Algebraic	Shifts (A_a, A_{a+1}) right *n* places A_a, A_{a+1} sign fill
73	06		LSC	Load Shift and Count	(U) → A_a; shift (A_a) left circularly until $(A_a)_0 \neq (A_a)_1$; number of shifts → A_{a+1}
73	07		DLSC	Double Load Shift and Count	(U, U+1) → A_a, A_{a+1}; shift (A_a, A_{a+1}) left circularly until $(A_a, A_{a+1})_{36} \neq (A_a, A_{a+1})_{37}$; number of shifts → A_{a+2}
73	10		LSSC	Left Single Shift Circular	Shifts (A_a) left circularly *n* places → A_a
73	11		LDSC	Left Double Shift Circular	Shifts (A_a, A_{a+1}) left circularly *n* places → A_a, A_{a+1}
73	12		LSSL	Left Single Shift Logical	Shifts (A_a) left *n* → A_a, places zero fill vacated bits
73	13		LDSL	Left Double Shift Logical	Shifts (A_a, A_{a+1}) left *n* places → A_a, A_{a+1}, zero fill

Instructions Listed by Function Code (continued)

f	j	a	Mnemonic	Instruction	Description
73	14	00	NOPI (BM)	No operation Increment	No operation is performed; but index incrementation, if specified, occurs normally
73	14	00	NOP (EM)	No Operation	Proceeds to next instruction
73	14	01	LPM	Load Performance Monitors	Transfers operand bits 3-5 in DB3-5
73	14	02-13	(BM)		*
73	14	02	BUY (EM)	BUY Stack Frame	Acquires a new stack frame from the stack defined by B_b (base, LL, and UL of the stack) using stack pointer defined Xx-modifier and frame size defined by a combination of d-field and X_x-increment
73	14	03	SELL (EM)	SELL Stack Frame	Releases a stack frame
73	14	04	ER (EM)	Executive Request	Generates signal interrupt with Executive Request status
73	14	05	EX (EM)	Execute	Executes the instruction at U
73	14	06	EXR (EM)	Execute Repeated	Executes the instruction at U the number of times specified by R1. Only certain instructions can be executed. If the target is a test class instruction, it terminates, repeating if it skips.
73	14	07	BMTC (EM)	Bit Move with Translation and Control	Moves a source string of characters to a destination string of characters
73	14	10	BIM (EM)	Bit Move	Moves a source string of bits which starts on any bit boundary to a destination string which also starts on any bit boundary
73	14	11	BIML (EM)	Bit Move Long	Moves a source string of bits which starts on any bit boundary to a destination which also starts on any bit boundary
73	14	12	BIC (EM)	Bit Compare	Compares a source string of bits to a destination string of bits
73	14	13	BICL (EM)	Bit Compare Long	Compares a source string of characters to a destination string
73	14	14-17			*

Instructions Listed by Function Code (continued)

f	j	a	Mnemonic	Instruction	Description
73	15	00-01			*
73	15	02	LBRX	Load Breakpoint Register	Transfers operand to Breakpoint Register
73	15	03	ACEL	Accelerate User Register Set	Transfers 48-word storage operand to user register set
73	15	04	DCEL	Decelerate User Register Set	Transfers user register set to 48-word storage operand
73	15	05	SPID	Store Processor Identification	Stores IP equipment identification in U
73	15	06			*
73	15	07	SEND	Send	Signals a specified IP or IOP that a message is being sent
73	15	10	ACK	Acknowledge	Signals a specified sender that a message has been received
73	15	11	TPSA	Test Previous Send Acknowledged	Tests if previous SEND message has been acknowledged by receiver
73	15	12	(BM)		*
73	15	12	LAE (EM)	Load Addressing Environment	Loads B1-B15 according to words 1-15 of a 16 word storage operand (ABT), word 0 is ignored
73	15	13			*
73	15	14	LD	Load Designator Register	Places full-word operand in Designator register
73	15	15	SD	Store Designator Register	Stores Designator register contents at location specified by operand address
73	15	16	UR	User Return	Loads Activity State Packet and loads B0
73	15	17	SGNL	Signal Condition	Notifies the Executive that some previously defined condition has been detected and that appropriate action is required
73	16		(BM)		*
73	16		EDDE (EM)	Edit Decimal	Moves a source string to a destination string

Instructions Listed by Function Code (continued)

f	j	a	Mnemonic	Instruction	Description
73	17	00	TS	Test and Set	If $(U)_5 = 1$, generate Test and Set interrupt; if $(U)_5 = 0$, goes to NI; if U ≥ 200, then $01_8 \rightarrow U_{0-5}$; $(U)_{6-35}$ unchanged
73	17	01	TSS	Test and Set and Skip	If $(U)_5 = 1$, goes to NI; if $(U)_5 = 0$, skip NI; if U ≥ 200, then $01_8 \rightarrow U_{0-5}$; $(U)_{6-35}$ unchanged
73	17	02	TCS	Test and Clear and Skip	If $(U)_5 = 0$, perform NI; if $(U)_5 = 1$, skip NI; if U ≥ 200 clears $(U)_{0-5}$; $(U)_{6-35}$ unchanged
73	17	03-05	(BM)		*
73	17	03	RTN (EM)	Procedure Return	RTN is the complementary instruction to CALL and LOCL, and overrides the most recent CALL operation
73	17	04	LUD (EM)	Load User Designators	Copies bits 18-30 and 32-35 of the operand into corresponding bit positions of the designator register
73	17	05	SUD (EM)	Store User Designators	Stores bits 18-35 of the designator register into bit positions 18-35 of U; bits 0-17 are cleared to zero
73	17	06	IAR	Initiate Auto-Recovery	Causes the IP to halt and initiates an auto-recovery
73	17	07	CPS	Clear Previous Send	Unconditionally clears the IP output request signal in the addressed universal processor interface
73	17	10	IPC	IP Control	Provides a communications path from the operating system to the supporting IP hardware and firmware.
73	17	11	(BM)		*
73	17	11	CJHE (EM)	Create Jump History Entry	The rightmost 24 bits of the 36-bit operand are used to construct an entry on the jump history.
73	17	12	SYSC	System Control	Provides a facility for communicating for diagnostic purposes with system components at a level closer to the hardware interfaces than is provided by the universal processor interface network.
73	17	13-17			*
74	00		JZ	Jump Zero	Jumps to U if $(A_a) = \pm 0$; goes to NI if $(A_a) \neq \pm 0$
74	01		JNZ	Jump Non Zero	Jumps to U if $(A_a) \neq \pm 0$; goes to NI if $(A_a) = \pm 0$

Instructions Listed by Function Code (continued)

f	j	a	Mnemonic	Instruction	Description
74	02		JP	Jump Positive	Jumps to U if $(A_a)_0 = 0$; goes to NI if $(A_a)_0 = 1$
74	03		JN	Jump Negative	Jumps to U if $(A_a)_0 = 1$; goes to NI if $(A_a)_0 = 0$
74	04	00	J (BM)	Jump	The IP performs an unconditional jump to the instruction located at the operand address
74	04	01-17	JK (BM)	Jump Keys	No operation performed; the IP operates as if jump key tested were present, but cleared.
74	05	00	HJ (BM)	Halt Jump	The IP performs an unconditional jump to the instruction located at the operand address and never halts
74	05	01-17	HKJ (BM)	Halt Keys and Jump	The IP performs an unconditional jump to the instruction located at the operand address and never halts
74	06		NOP (BM)	No Operation	Proceeds to next instruction
74	07		AAIJ (BM)	Allow All Interrupts and Jump	Allows all interrupts and jump to U
74	04-07		(EM)		*
74	10		JNB	Jump No Low Bit	Jumps to U if $(A_a)_{35} = 0$; goes to NI if $(A_a)_{35} = 1$
74	11		JB	Jump Low Bit	Jumps to U if $(A_a)_{35} = 1$; goes to NI if $(A_a)_{35} = 0$
74	12		JMGI	Jump Modifier Greater and Increment	Jumps to U if $(X_a)_{18-35}$ 0; goes to NI if $(X_a)_{18-35} \leq 0$; always $(X_a)_{18-35} + (X_a)_{0-17} \rightarrow X_{a\ 18-35}$
74	13		LMJ	Load Modifier and Jump	1 + bits 18 - 35 of PAR $\rightarrow (X_a)_{18-35}$; jump to U
74	14	00	JO	Jump Overflow	Jumps to U if DB19 = 1; goes to NI if DB19 = 0
74	14	01	JFU	Jump Floating Underflow	Jumps to U if DB21 = 1, clears DB21; goes to NI if DB21 = 0
74	14	02	JFO	Jump Floating Overflow	Jumps to U if DB22 = 1, clears DB22; goes to NI if DB22 = 0
74	14	03	JDF	Jump Divide Fault	Jumps to U if DB23 = 1, clears DB23; goes to NI if DB23 = 0
74	14	04-06	(BM)		*

Instructions Listed by Function Code (continued)

f	j	a	Mnemonic	Instruction	Description
74	14	04	JC (EM)	Jump Carry	Jumps to U if DB18 = 1; goes to NI if DB18 = 0
74	14	05	JNC (EM)	Jump No Carry	Jumps to U if DB18 = 0; goes to NI if DB18 = 1
74	14	06	AAIJ (EM)	Allow All Interrupts and Jump	Allows all interrupts and jumps to U
74	14	07	PAIJ	Prevent all Interrupts and Jump	Prevents all interrupts and jumps to U
74	14	10–17			*
74	15	00	JNO	Jump No Overflow	Jumps to U if DB19 = 0; goes to NI if DB19 = 1
74	15	01	JNFU	Jump No Floating Underflow	Jumps to U if DB21 = 0; goes to NI if DB21 = 1; clears DB21
74	15	02	JNFO	Jump No Floating Overflow	Jumps to U if DB22 = 0; goes to NI if DB22 = 1; clears DB22
74	15	03	JNDF	Jump No Divide Fault	Jumps to U if DB23 = 0; goes to NI if DB23 = 1; clears DB23
74	15	04	(BM)		*
74	15	04	J (EM)	Jump	The IP performs an unconditional jump to the instruction located at the operand address
74	15	05	HLTJ	Halt Jump	Unconditional jump to the operand address and the IP halts before executing that instruction
74	15	06–17			*
74	16		JC (BM)	Jump Carry	Jumps to U if DB18 = 1; goes to NI if DB18 = 0
74	17		JNC (BM)	Jump No Carry	Jumps to U if DB18 = 0; goes to NI if DB18 = 1
74	16–17		(EM)		*
75	00		LBU	Load Base Register, User	Loads user base register B_a
75	01		LBUI	Load Base Register, User Indirect	Loads user base register B_a; except when a = 2–11 and L,BDI refers to basic bank; then B_a is marked void and to cause implicit base register selection
75	02		SBU	Store Base User	$B_a \rightarrow U$

Instructions Listed by Function Code (continued)

f	j	a	Mnemonic	Instruction	Description
75	03		LBE	Load Base Register, Executive	U → B_{a+16}
75	04		LBE0	Load Base Register Executive/0	U → B_{a+16} word 0
75	05		LBE1	Load Base Register Executive/1	U → B_{a+16} word 1
75	06		LBU0	Load Base Register User/0	U → user B_a word 0
75	07		LBU1	Load Base Register User/1	U → user B_a word 1
75	10		TVA	Test Virtual Address	Interprets a virtual address and validates access to that address in the current addressing environment
75	11		TVAE	Test Virtual Address in Environment	Same as TVA except that an alternate addressing environment can be specified
75	12				*
75	13		LXLM	Load X Register Long Modifier	Transfers operand rightmost 24 bits to X_a rightmost 24 bits
75	14		LBN	Load Bank Name	Uses U_{0-17} as L, BDI to determine true bank name in $X_{a\ 0-17}$
75	15		CR	Conditional Replace	If the contents of A_a are equal to (U) then store the contents of A_{a+1} into the location specified by the operand address and skip the next instruction. If the contents of A_a are not equal to (U) then do not store A_{a+1} and do not skip the next instruction. If the operand is in GRS ($U<200_8$), the operation is undefined. The entire CR is performed under storage lock.
75	16				*
75	17		RMD	Read Master Dayclock	Retrieves contents of dayclock designated as master for the IP on which RMD is executed
76	00		FA	Floating Add	(A_a) + (U) → A_a; RESIDUE → A_{a+1} if DB30 = 1
76	01		FAN	Floating Add Negative	(A_a) - (U) → A_a; RESIDUE → A_{a+1} if DB30 = 1
76	02		FM	Floating Multiply	(A_a) ● (U) → A_a (and A_{a+1} if DB30 = 1)

Instructions Listed by Function Code (continued)

f	j	a	Mnemonic	Instruction	Description
76	03		FD	Floating Divide	(A_a) divided by (U) → A_a; REMAINDER → A_{a+1} if DB30 = 1
76	04		LUF	Load and Unpack Floating	$\lvert (U) \rvert_{1-8}$ → A_{a28-35}, zero fill; $(U)_{9-35}$ → $A_{a+1 \text{ bits } 9-35}$ sign fill
76	05		LCF	Load and Convert to Floating	$(U)_0$ → $A_{a+1 \text{ bit } 0}$, [NORMALIZED (U)]$_{26-0}$ → $A_{a+1 \text{ bits } 9-35}$ if $(U)_0 = 0$, $(A_a)_{28-35}$ ± NORMALIZING COUNT → $A_{a+1 \text{ bits } 1-8}$ if $(U)_0 = 1$, ones complement of [$(A_a)_{28-35}$ ± NORMALIZING COUNT] → $A_{a+1 \text{ bits } 1-8}$
76	06		MCDU	Magnitude of Characteristic Difference to Upper	$\lvert \lvert (A_a) \rvert_{0-8} - \lvert (U) \rvert_{0-8} \rvert$ → $A_{a+1 \text{ bits } 27-35}$; zeros → $A_{a+1 \text{ bits } 0-26}$
76	07		CDU	Characteristic Difference to Upper	$\lvert (A_a) \rvert_{0-8} - \lvert (U) \rvert_{0-8}$ → $A_{a+1 \text{ bits } 27-35}$; sign bits → $A_{a+1 \text{ bits } 0-26}$
76	10		DFA	Double-Precision Floating Add	$(A_a, A_{a+1}) + (U, U+1)$ → A_a, A_{a+1}
76	11		DFAN	Double-Precision Floating Add Negative	$(A_a, A_{a+1}) - (U, U+1)$ → A_a, A_{a+1}
76	12		DFM	Double-Precision Floating Multiply	$(A_a, A_{a+1}) \bullet (U, U+1)$ → A_a, A_{a+1}
76	13		DFD	Double-Precision Floating Divide	(A_a, A_{a+1}) divided by $(U, U+1)$ → A_a, A_{a+1}
76	14		DFU	Double Load and Unpack Floating	$\lvert (U, U+1) \rvert_{37-47}$ → $A_{a\ 25-35}$, zero fill; $(U, U+1)_{48-71}$ → $A_{a+1 \text{ bits } 12-35}$, sign fill; $(U, U+1)_{0-35}$ → A_{a+2}
76	15		DLCF	Double Load and Convert to Floating	$(U)_0$ → $A_{a+1 \text{ bit } 0}$; [NORMALIZED $(U, U+1)]_{0-35, 48-71}$ → $A_{a+1 \text{ bits } 12-35}$ and A_{a+2}; if $(U)_0 \neq 0$, $(A_a)_{25-35}$ ± NORMALIZING COUNT → $A_{a+1 \text{ bits } 1-11}$; if $(U)_0 = 1$, ones complement of [$(A_a)_{25-35}$ ± NORMALIZING COUNT] → $A_{a+1 \text{ bits } 1-11}$
76	16		FEL	Floating Expand and Load	If $(U)_0 = 0$; $(U)_{0-8} + 1600_8$ → $A_{a\ 0-11}$; if $(U)_0 = 1$; $(U)_{0-8} - 1600_8$ → A_{a0-11} $(U)_{9-32}$ → $A_{a\ 12-35}$; $(U)_{33-35}$ → $A_{a+1 \text{ bits } 0-2}$; $(U)_0$ → $A_{a+1 \text{ bits } 3-35}$
76	17		FCL	Floating Compress and Load	If $(U)_0 = 0$; $(U)_{0-11} - 1600_8$ → $A_{a\ 0-8}$; if $(U)_0 = 1$; $(U)_{0-11} + 1600_8$ → $A_{a\ 0-8}$ $(U)_{12-35}$ → A_{a9-32}; $(U+1)_{0-2}$ → $A_{a\ 33-35}$
77			(BM)		*

Instructions Listed by Function Code (continued)

f	j	a	Mnemonic	Instruction	Description
77			MOP (EM)	Macro-Operation	Executes the macro operation selected by the identifier obtained by catering the contents of the instruction's j- and a-fields.

Appendix B. Instructions Listed by Mnemonic

This appendix lists the instruction mnemonics for the basic and extended modes sorted in alphabetical order with the operation code corresponding to each mnemonic. Along with the operation code (f-field value), the j-field, a-field, and Processor Privilege (PP) values are also given for each mnemonic.

The definition of the column headings are the same as those given in Appendix A, with the exception of the PP value which is described as follows:

PP – The processor privilege value required for the execution of the instruction; for example, the relation PP≤n must be satisfied, where n is the number in this column. A blank in this column implies a PP value of 3.

At the end of this appendix, the operation codes that perform no operation, are undefined, or result in an invalid operation are listed with a single asterisk, double asterisk, or triple asterisk, respectively, in the INST column. The meaning of these symbols is defined in Appendix A for the Description column heading.

Instructions Listed by Mnemonic

	Basic Mode				Extended Mode			
INST	f	j	a	PP	f	j	a	PP
AA	14				14			
AAIJ	74	07			74	14	06	0
ACEL	73	15	03	2	73	15	03	2
ACK	73	15	10	0	73	15	10	0
ADD1	05	00-15	15	0	05	00-15	15	
ADE	07	00			07	00		
AH	72	04			72	04		
AMA	16				16			
ANA	15				15			
AND	42				42			
ANH	42	05			72	05		
ANMA	17				17			
ANT	72	07			72	07		
ANU	21				21			
ANX	25				25			
AT	72	06			72	06		
AU	20				20			
AX	24				24			
BBN	72	14			72	14		
BDE	37	15			72	10		
BIC	37	11			73	14	12	
BICL	37	13			73	14	13	
BIM	37	10			73	14	10	
BIML	37	14			73	14	11	
BMTC	37	12			73	14	07	
BN	72	12			72	12		
BT	22	00-15			22	00-15		
BUY					73	14	02	
CALL					07	16	13	
CDU	76	07			76	07		
CJHE					73	17	11	
CPS	73	17	07	0	73	17	07	0
CR	75	15		0	75	15		
DA	71	10			71	10		
DADE	07	01			07	01		
DAN	71	11			71	11		
DCEL	73	15	04	2	73	15	04	2
DDEI	07	07			07	07		
DEB	37	16			72	11		
DEC	05	00-15	11		05	00-15	11	
DEC2	05	00-15	13		05	00-15	13	
DEI	07	06			07	06		

	Basic Mode				Extended Mode			
INST	f	j	a	PP	f	j	a	PP
DESF					33	04		
DF	36				36			
DFA	76	10			76	10		
DFAN	76	11			76	11		
DFD	76	13			76	13		
DFM	76	12			76	12		
DFU	76	14			76	14		
DI	34				34			
DIDE	07	11			07	11		
DJZ	71	16			71	16		
DL	71	13			71	13		
DLCF	76	15			76	15		
DLM	71	15			71	15		
DLN	71	14			71	14		
DLSC	73	07			73	07		
DS	71	12			71	12		
DSA	73	05			73	05		
DSC	73	01			73	01		
DSDE	07	03			07	03		
DSF	35				35			
DSL	73	03			73	03		
DTE	71	17			71	17		
EDDE	37	17			73	16		
ENZ	05	00-15	14		05	00-15	14	
ER	72	11			73	14	04	
EX	72	10			73	14	05	
EXR					73	14	06	
FA	76	00			76	00		
FAN	76	01			76	01		
FCL	76	17			76	17		
FD	76	03			76	03		
FDA					33	00		
FDAN					33	01		
FDD					33	03		
FDM					33	02		
FDR					33	06		
FDSP					33	05		
FDT					33	07		
FEL	76	16			76	16		
FM	76	02			76	02		
GOTO					07	17	00	
HJ	74	05	00					

	Basic Mode				Extended Mode			
INST	f	j	a	PP	f	j	a	PP
HKJ	74	05	01-17					
HLTJ	74	15	05	0	74	15	05	0
IAR	73	17	06	0	73	17	06	0
IDE	07	10			07	10		
INC	05	00-15	10		05	00-15	10	
INC2	05	00-15	12		05	00-15	12	
IPC					73	17	10	0
J	74	04	00		74	15	04	
JB	74	11			74	11		
JC	74	16			74	14	04	
JDF	74	14	03		74	14	03	
JFO	74	14	02		74	14	02	
JFU	74	14	01		74	14	01	
JGD	70				70			
JK	74	04	01-17					
JMGI	74	12			74	12		
JN	74	03			74	03		
JNB	74	10			74	10		
JNC	74	17			74	14	05	
JNDF	74	15	03		74	14	03	
JNFO	74	15	02		74	15	02	
JNFU	74	15	01		74	15	01	
JNO	74	15	00		74	15	00	
JNS	72	03			72	03		
JNZ	74	01			74	01		
JO	74	14	00		74	14	00	
JP	74	02			74	02		
JPS	72	02			72	02		
JZ	74	00			74	00		
LA	10				10			
LAE					73	15	12	0
LAQW	07	04			07	04		
LBE	75	03		0	75	03		0
LBE0	75	04		0	75	04		0
LBE1	75	05		0	75	05		0
LBJ	07	17	01-17					
LBN	75	14		0	75	14		
LBRX	73	15	02	0	73	15	02	0
LBU	75	00		0	75	00		
LBUI	75	01		0	75	01		
LBU0	75	06		0	75	06		0
LBU1	75	07		0	75	07		0
LCF	76	05			76	05		

	Basic Mode				Extended Mode			
INST	f	j	a	PP	f	j	a	PP
LD	73	15	14	0	73	15	14	0
LDJ	07	12						
LDSC	73	11			73	11		
LDSL	73	13			73	13		
LIJ	07	13						
LMA	12				12			
LMJ	74	13			74	13		
LNA	11				11			
LNMA	13				13			
LOCL					07	16	00	
LPD	07	14						
LPM	73	14	01		73	14	01	
LR	23				23			
LRC	37	02		0	37	02		0
LRD	37	00		0	37	00		0
LRS	72	17			72	17		
LSC	73	06			73	06		
LSSC	73	10			73	10		
LSSL	73	12			73	12		
LUD					73	17	04	
LUF	76	04			76	04		
LX	27				27			
LXI	46				46			
LXLM	75	13		0	75	13		
LXM	26				26			
LXSI					51			
MASG	71	07						
MASL	71	06						
MATG					71	07		
MATL					71	06		
MCDU	76	06			76	06		
MF	32				32			
MI	30				30			
MLU	43				43			
MOP					77			
MSE	71	00						
MSG	71	03						
MSI	31				31			
MSLE	71	02						
MSNE	71	01						
MSNG	71	02						
MSNW	71	05						
MSW	71	04						
MTE					71	00		

	Basic Mode				Extended Mode			
INST	f	j	a	PP	f	j	a	PP
MTG					71	03		
MTLE					71	02		
MTNE					71	01		
MTNG					71	02		
MTNW					71	05		
MTW					71	04		
NOP	74	06			73	14	00	
NOPI	73	14	00	0				
OR	40				40			
PAIJ	74	14	07	0	74	14	07	0
PRBA	01	16			01	16		
PRBB	01	17			01	17		
RMD	75	17		2	75	17		2
RRC					37	03		0
RRD	37	01		0	37	01		0
RTN					73	17	03	
SA	01	00–15			01	00–15		
SAQW	07	05			07	05		
SAS	05	00–15	06		05	00–15	06	
SAZ	05	00–15	07		05	00–15	07	
SBU	75	02		0	75	02		
SD	73	15	15	1	73	15	15	1
SDE	07	02			07	02		
SDMF	37	04	02	0	37	04	02	0
SDMN	37	04	01	0	37	04	01	0
SDMS	37	04	03	0	37	04	03	0
SE	62							
SELL					73	14	03	
SEND	73	15	07	0	73	15	07	0
SFS	05	00–15	04		05	00–15	04	
SFZ	05	00–15	05		05	00–15	05	
SG	65							
SGNL	73	15	17		73	15	17	
SLE	64							
SLJ	72	01						
SLP0	37	06	00	0	37	06	00	0
SLP1	37	06	01	0	37	06	01	0
SMA	03	00–15			03	00–15		
SMD	37	04	00	0	37	04	00	0
SNA	02	00–15			02	00–15		
SNE	63							
SNG	64							
SNW	67							
SNZ	05	00–15	01		05	00–15	01	
SN1	05	00–15	03		05	00–15	03	
SPD	07	15						
SPID	73	15	05	2	73	15	05	2
SP1	05	00–15	02		05	00–15	02	
SR	04	00–15			04	00–15		
SRS	72	16			72	16		
SSA	73	04			73	04		
SSC	73	00			73	00		
SSL	73	02			73	02		
SUB1	05	00–15	16	0	05	00–15	16	
SUD					73	17	05	
SW	66							
SX	06	00–15			06	00–15		
SYSC					73	17	12	0
SZ	06	00–15	00		05	00–15	00	
TCS	73	17	02		73	17	02	
TE	52				52			
TEP	44				44			
TG	55				55			
TGZ					50		01	
THC					72	13		
TLE	54				54			
TLEM	47				47			
TLZ					50		10	
TMZ					50		04	
TMZG					50		05	
TN	61				50		14	
TNE	53				53			
TNG	54				54			
TNGM	47				47			
TNGZ					50		16	
TNLZ					50		07	
TNMZ					50		13	
TNOP					50		00	
TNPZ					50		15	
TNW	57				57			
TNZ	51				50		11	
TOP	51				45			
TP	60				50		03	
TPSA	73	15	11	0	73	15	11	0
TPZ					50		02	
TPZL					50		12	

	Basic Mode				Extended Mode			
INST	f	j	a	PP	f	j	a	PP
TRA	72	15			72	15		1
TS	73	17	00		73	17	00	
TSKP					50		17	
TSS	73	17	01		73	17	01	
TVA	75	10		0	75	10		
TVAE	75	11		1	75	11		1
TW	56				56			
TZ	50		00		50		06	
TZ	50		01-17					
UR	73	15	16	0	73	15	16	0
XOR	41				41			
*	00				00			
*	05	16,17	15	0	05	16,17	15	
*	05		17	0	05		17	
*					07	12-15		
*	07	16						
*	07	17	00					
*	33				33	08-17		
*	37	07			37	07		
*					37	10-17		
*					60-67			
*	72	00			72	00		
*					72	01		
*	72	13						
*	73	14	02-13					
*	73	14	14-17	0	73	14	14-17	0
*	73	15	00,01	0	73	15	00,01	0

	Basic Mode				Extended Mode			
INST	f	j	a	PP	f	j	a	PP
*	73	15	06	0	73	15	06	0
*	73	15	12					
*	73	15	13		73	15	13	
*	73	16						
*	73	17	03-05					
*	73	17	11					
*	73	17	13-17		73	17	13-17	
*					74	04-07		
*	74	14	04-06					
*	74	14	10-17					
*	74	15	04					
*	74	15	06-17		74	15	06-17	
*					74	16,17		
*	75	12			75	12		
*	75	16			75	16		
*	77							
**	02	16,17			02	16,17		
**	03	16,17			03	16,17		
**	04	16,17			04	16,17		
**	05	16,17	00		05	16,17	00	
**	05	16,17	01-14		06	16,17	01-14	
**	05	16,17	16		05	16,17	16	
**	06	16,17			06	16,17		
**	22	16,17			22	16,17		
***					07	16	01-12	
***					07	16	14-17	
***	37	04	04-17		37	04	04-17	
***	37	06	02-17		37	06	02-17	

Appendix C. Abbreviations, Acronyms, and Symbols

The following abbreviations, acronyms, and symbols are frequently used in this manual.

a	a (arithmetic register) field of instruction
A_a	Arithmetic register specified by the a-field of an instruction
AND	Symbol denoting logical product, or logical AND
b	b (base) field of an extended mode instruction
B	Base register
BBC	Byte Bus Channel
B bit	Basic mode addressing indicator, in a base register
B_a	Base register specified by a-field of an instruction
B_b	Base register specified by b-field (extended mode) of an instruction
BM	Basic Mode
BMC	Block Multiplexer Channel
CAW	Channel Address Word
CCW	Channel Command Word
CFW	Channel Fault Word
CSW	Channel Status Word
d	d (displacement) field of an extended mode instruction
DB	Designator Bit
DCC	Disk Controller Channel
DCP	Distributed Communications Processor

ECC	Error Correction Code
EF	External Function
EI	External Interrupt
EM	Extended Mode
f	f (function) field of instruction
FEF	Forced External Function
GRS	General Register Set
h	h (index incrementation) field of instruction
i	i (indirect) field of instruction or increment portion of x-register
ICP	Integrated Communications Processor
IOP	Input/Output Processor
IP	Instruction Processor
ISI	Internally Specified Index
ITCU	Integrated Tape Control Unit
ISW	Interrupt Status Word
j	j (partial word) field of instruction
K	Used for notational convenience to replace the low order digits of an integral power of 2 or an integral multiple thereof. Thus, 262K is used to represent 262,144 (2^{18}).
k	Represents 1000 in decimal notation
L-bus	The interface used by the attaching integrated control units. The L-bus communicates with line modules that support communication activity and peripheral control unit activity. (Also called byte multiplexing bus.)
m	Modifier portion of x-register
MSR	Module Select Register
MSU	Main Storage Unit
NI	Next Instruction
NRZI	Non-Return to Zero Inverted
OR	Symbol denoting logical sum, or inclusive OR
PE	Phase Encoded

PP	Processor Privilege
R	R-register, GRS addresses 100_8–137_8
R_a	R-register specified by the a-field of an instruction
RAM	Random Access Memory
s-bit	Size indicator in B-registers and bank descriptors
SBC	Single Bit Correction
System bus	The primary data path between central complex units. (Also called S-bus.)
SSP	System Support Processor
TIC	Transfer In Channel
TSW	Table Status Word
U	The effective address or value of the operand after application of indexing and indirect addressing.
u	u (displacement) field of a basic mode instruction
UPI	Universal Processor Interface
v-bit	Void indicator in a base register
X	Index register. A control register in the GRS specified by the x-field of an instruction.
X_a	Index register specified by the a-field of an instruction.
Xi	Increment portion of an index register (bits 0–17)
Xm	Modifier portion of an index register (bit 18–35)
x	x (index register) field of instruction
XOR	Symbol denoting logical difference, or exclusive OR
( )	Contents of
( )'	Complement of contents of
| ( ) |	Absolute value of magnitude of contents of
$( )_{00\text{–}17}$	Subscripts indicate the bit positions involved. A full word is normally not subscripted. Subscripts are also used to designate octal or decimal notation.
→	Direction of data flow

Index

CUT

SPERRY

USER COMMENT SHEET

We will use your comments to improve subsequent editions.

NOTE: Please do not use this form as an order blank.

_____

(Document Title)

_____ _____ _____

(Document No.) *(Revision No.)* *(Update No.)*

Comments:

From:

_____

(Name of User)

_____

(Business Address)

Fold on dotted lines, and mail. (No postage stamp is necessary if mailed in the U.S.A.)
Thank you for your cooperation

CUT

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 21 BLUE BELL, PA.

POSTAGE WILL BE PAID BY ADDRESSEE

SPERRY CORPORATION

ATTN.: SOFTWARE SYSTEMS PUBLICATIONS

P.O. BOX 64942
ST. PAUL, MINNESOTA 55164

FOLD