60458920

CONTROL DATA CORPORATION

CDC® CYBER 170 COMPUTER SYSTEMS MODELS 865 AND 875

HARDWARE REFERENCE MANUAL

REVISION RECORD

REVISION	DESCRIPTION
A	Manual released.
(04-23-82)	
Publication No. 60458920	

REVISION LETTERS I, O, Q, S, X AND Z ARE NOT USED

Address comments concerning this manual to:
Control Data Corporation
Publications and Graphics Division
4201 North Lexington Avenue
St. Paul, Minnesota 55112

or use Comment Sheet in the back of this manual.

LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual, are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

PAGE	REV
Front Cover	-
Inside Front Cover	A
Title Page	-
2	A
3/4	A
5/6	A
7	A
8	A
9	A
10	A
1-1	A
1-2	A
1-3	A
1-4	A
1-5	A
1-6	A
2-1	A
2-2	A
2-3	A
2-4	A
2-5	A
2-6	A
2-7	A
2-8	A
2-9	A
2-10	A
2-11	A
2-12	A
2-13	A
2-14	A
2-15	A
2-16	A
3-1	A
3-2	A
3-3	A
3-4	A
3-5	A
3-6	A
3-7	A
3-8	A
3-9	A
3-10	A
3-11	A
4-1	A
4-2	A
4-3	A
4-4	A
4-5	A
4-6	A
4-7	A
4-8	A
4-9	A
4-10	A
4-11	A
4-12	A
4-13	A
4-14	A
4-15	A
4-16	A
4-17	A
4-18	A
4-19	A
4-20	A
4-21	A
4-22	A
4-23	A
4-24	A
4-25	A
4-26	A
4-27	A
4-28	A
4-29	A
4-30	A
4-31	A
4-32	A
4-33	A
4-34	A
4-35	A
4-36	A
5-1	A
5-2	A
5-3	A
5-4	A
5-5	A
5-6	A
5-7	A
5-8	A
5-9	A
5-10	A
5-11	A
5-12	A
5-13	A
5-14	A
5-15	A
5-16	A
5-17	A
5-18	A
5-19	A
5-20	A
5-21	A
5-22	A
5-23	A
5-24	A
5-25	A
5-26	A
5-27	A
5-28	A
5-29	A
5-30	A
5-31	A
5-32	A
5-33	A
5-34	A
5-35	A
5-36	A
5-37	A
5-38	A
5-39	A
5-40	A
5-41	A
5-42	A
5-43	A
5-44	A
5-45	A
5-46	A
5-47	A
5-48	A
5-49	A
5-50	A
5-51	A
5-52	A
5-53	A
5-54	A
A-1	A
Index-1	A
Index-2	A
Index-3	A
Index-4	A
Index-5	A
Index-6	A
Comment Sheet	A
Back Cover	-

PREFACE

This manual contains hardware reference information for the CDC® CYBER 170 Models 865 and 875 Computer Systems.

The manual describes the functional, operational, and programming characteristics of the computer system hardware.

This manual is for use by customer, marketing, training, programming, and Engineering Services personnel who operate, program, and maintain the computer systems.

Other manuals that are applicable to the CYBER 170 computer systems are:

Control Data Publication	Publication Number
NOS Version 2 Operator/Analyst Handbook	60459310
NOS Version 2 Systems Programmer's Instant	60459370
NOS Version 1 Operator's Guide	60493900
NOS/BE Version 1 System Programmer's Reference Manual, Volume 1	60494100
NOS/BE Version 1 System Programmer's Reference Manual, Volume 2	60457370
CYBER 170 Models 175, 740, 750 and 760 Refrigeration Manual	60427800
CYBER 170 Models 720, 730, 740, 750, 760 Power Distribution and Warning System Hardware Maintenance Manual	60456160
Extended Semiconductor Memory (ESM) Hardware Reference Manual	60455990

Publication ordering information and latest revision levels are available from the Literature Distribution Services catalog, publication number 90310500.

WARNING

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

CONTENTS

APPENDIX

INDEX

FIGURES

TABLES

SYSTEM DESCRIPTIONS

This section introduces the CYBER 170 Models 865 and 875 Computer Systems, gives physical and functional characteristics, and provides descriptions of major system components.

INTRODUCTION

Models 865 and 875 (figures 1-1 and 1-2) are large scale, high-speed computer systems for both business and scientific applications. Both systems include the following components.

- Central processor (CP).
- Central memory (CM).
- Input/output unit (IOU).
- Display station.

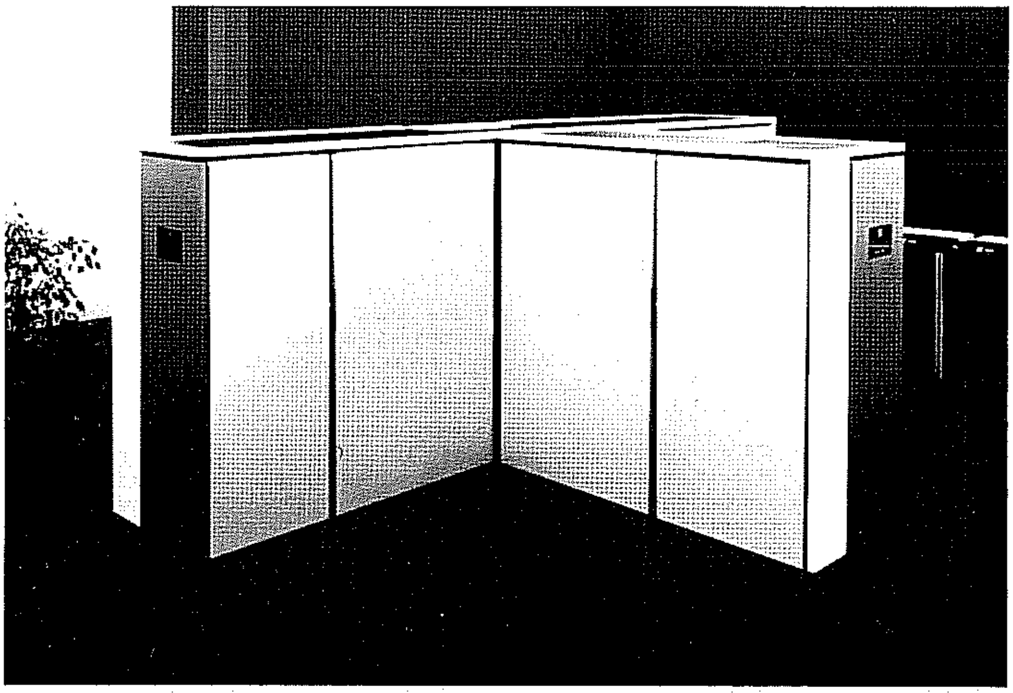


Figure 1-1. Model 865 Computer System

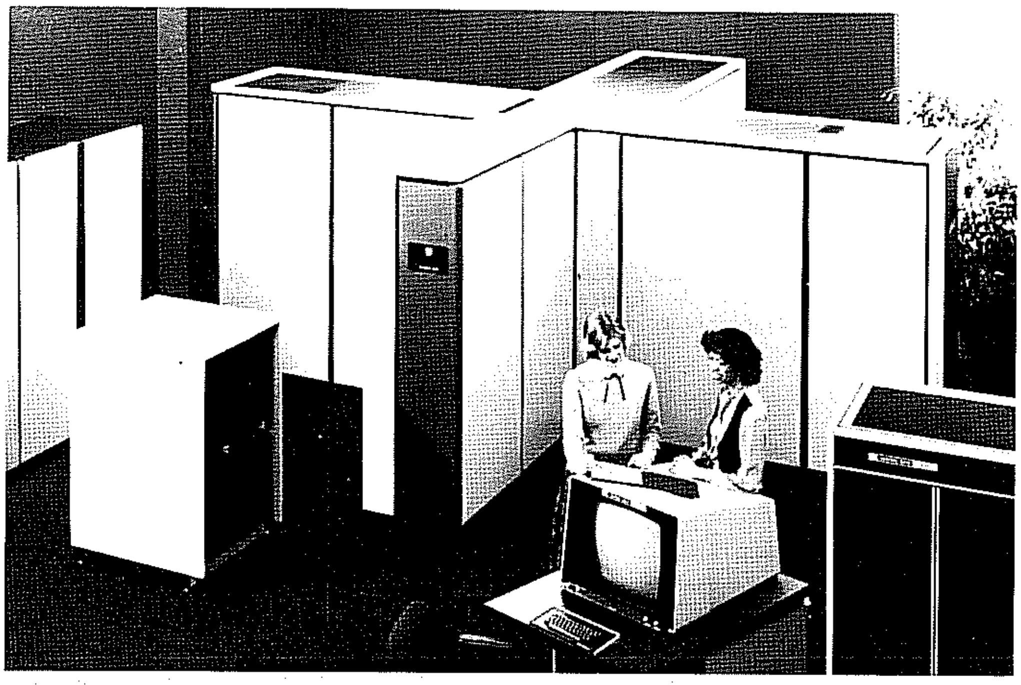


Figure 1-2. Model 875 Computer System

PHYSICAL CHARACTERISTICS

The model 865 basic mainframe configuration includes a display station and a two-bay cabinet for the CP, CM, and IOU. The model 875 basic mainframe configuration includes a display station and a three-bay cabinet for the CP, CM, and IOU. With both models, an additional one-bay cabinet contains the optional second CP (figures 1-3 and 1-4).

Each bay in the models 865 and 875 cabinets contains a logic chassis with plug-in circuit boards. The logic chassis in the IOU also contains a deadstart panel with initialization and maintenance controls. A stand-alone condensing unit provides logic chassis cooling for the models 865 and 875 cabinets. Also provided are dc power supplies and an ac/dc control section with voltage margin testing facilities. For additional cooling or power information, refer to the cooling system and power distribution and warning system manuals listed in the preface.

FUNCTIONAL CHARACTERISTICS

The models 865 and 875 use emitter-coupled logic (ECL) to achieve high computation speeds. The CP uses nine independent functional units in parallel operation. In this operation, the specialized arithmetic units provide simultaneous retrieval and execution of instructions. The CP design is based on the assumption that instructions are, in most cases, accessed from successive memory locations. Accordingly, the CP prefetches instructions expected to be used next while the current instruction is being processed.

The models 865 and 875 offer a second CP as an option. It operates in a manner parallel to the first CP, but with separate, individual access to central memory and the capability of executing the entire instruction set independently.

The model 865 CM is a semiconductor memory divided into eight independent banks. The model 875 CM is a bipolar memory divided into 16 independent banks. For both models, these banks may all be simultaneously in the process of completing read/write requests which are queued and distributed at ECL speeds. System input/output speeds are determined by the capabilities of existing external devices. Both models have the option of using extended memory, either logically partitioned within CM (unified extended memory), or physically separate from CM (external extended memory).

MODEL 865 CHARACTERISTICS

Central Processor

- 60-bit internal word.
- Eight 60-bit operand (X) registers.
- Eight 18-bit address (A) registers.
- Eight 18-bit index (B) registers.
- Two registers that isolate user central memory (RAC, FLC).
- Two registers that isolate user extended memory (RAE, FLE).
- Register exchanging instructions (exchange jumps).

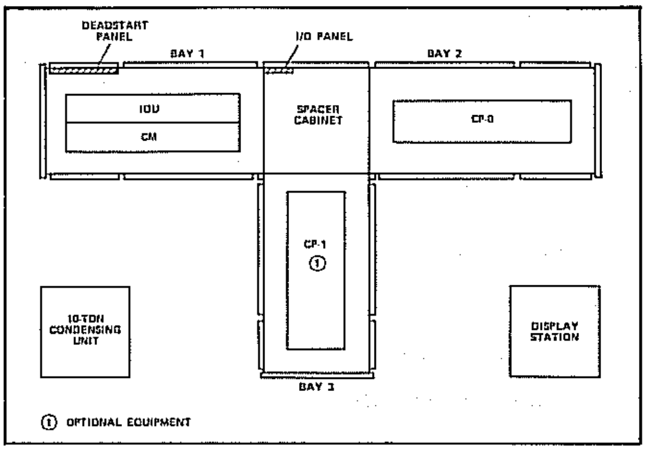


Figure 1-3. Model 865 Chassis Configuration (Top Cutaway View)

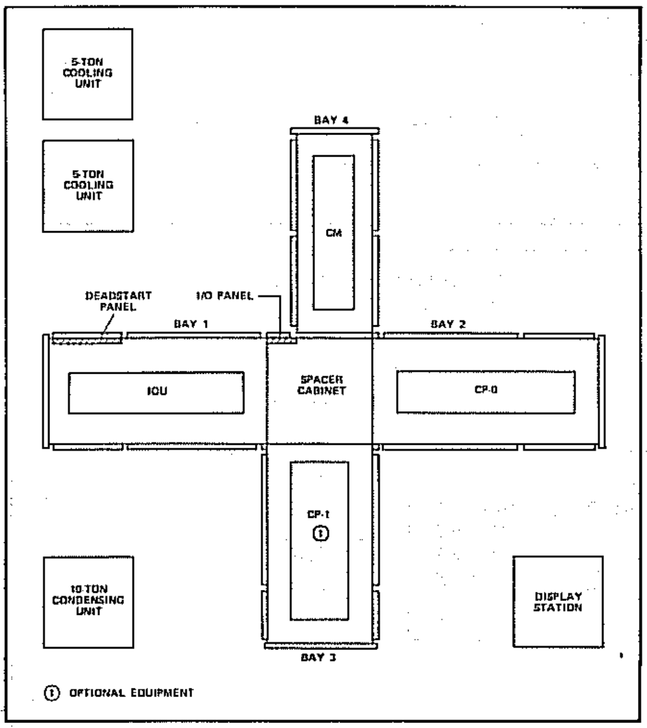


Figure 1-4. Model 875 Chassis Configuration (Top Cutaway View)

- Floating-point arithmetic (11-bit exponent, 48/96-bit coefficient).
- Integer arithmetic (60/18-bit operands).
- Packed instructions (15/30/60-bit instructions in 60-bit words).
- 12-word instruction word stack.
- Synchronous internal logic.
- 25-nanosecond clock period.
- Branch instruction lookahead.
- Parity checking of all major data and address paths crossing bay boundaries.

Central Memory

- 68-bit data word (60 data bits, 8 single-error correction double-error detection bits).
- 262K words of static metal-oxide semiconductor (MOS) memory with options available to 1048K words.
- Organization of eight independent banks.
- Clock period of 50 nanoseconds.
- Maximum data transfer rate of one word every 50 nanoseconds.
- Read access time to CP of 375 nanoseconds in both single and dual CP systems.
- Read/write cycle time of 200 nanoseconds.
- Single-error correction double-error detection (SECDED) on stored data.
- Parity checking of all major data and address paths.
- Unified extended memory (UEM).

Input/Output Unit

- Ten peripheral processors (PPs), 15-PP/20-PP options available. Each PP has independent memory (PPM) comprised of 4K 12-bit words.
- Port to central memory via central memory control (CMC).
- Twelve 12-bit channels to external devices, 24-channel option available.
- Real-time clock (channel 14_8).
- Display controller (channel 10_8).
- Communications interface for remote technical assistance (RTA) (channel 15_8).
- Parity checking on all major data and address paths.
- Operating speed of 500 nanoseconds and a minor cycle of 50 nanoseconds.

MODEL 875 CHARACTERISTICS

Central Processor

The model 875 CP is the same as that of model 865. Refer to the description of the CP under Model 865 Characteristics.

Central Memory

- 68-bit data word (60 data bits, 8 single-error correction double-error detection bits).
- 262K words of bipolar memory with options available to 1048K words.
- Organization of 16 independent banks.
- Two memory ports with 512K increments.
- Clock period of 25 nanoseconds.
- Maximum data transfer rate of one word every 25 nanoseconds.
- Read access time to CP of 200 nanoseconds in both single and dual CP systems.
- Read/write cycle time of 75 nanoseconds.
- Single-error correction double-error detection (SECDED) on stored data.
- Parity checking of all major data and address paths.
- Unified extended memory (UEM).

Input/Output Unit

The model 875 IOU is the same as that of model 865. Refer to the description of the IOU under Model 865 Characteristics.

MAJOR SYSTEM COMPONENT DESCRIPTIONS

The following are the major system components.

- Central processor (CP).
- Central memory (CM).
- Input/output unit (IOU).
- Display station.

CENTRAL PROCESSOR

The CP hardware (figure 1-5) consists of the following:

- Central processing unit (CPU).
- Nine functional units.

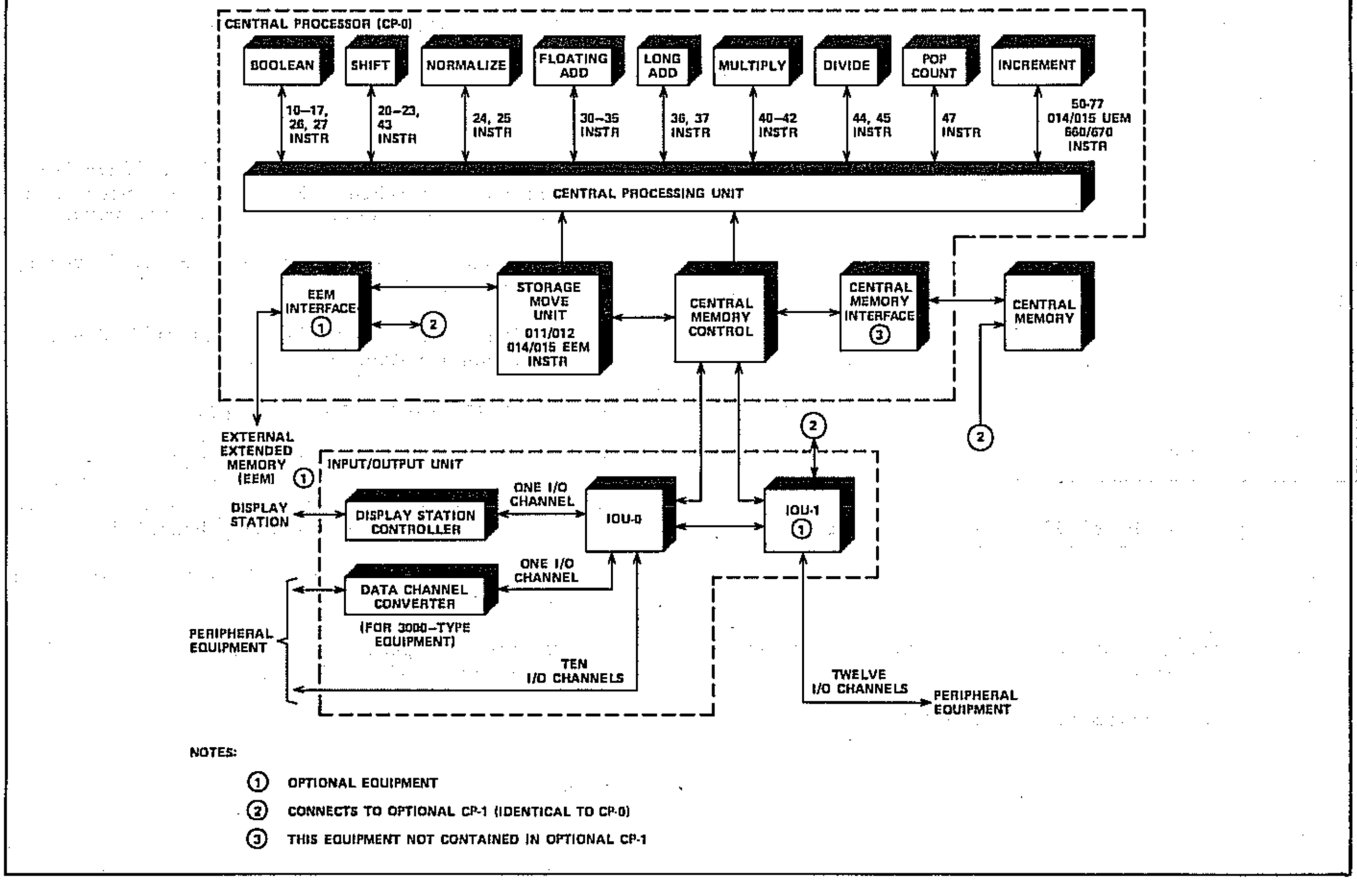


Figure 1-5. Models 865 and 875 Computer System Block Diagram

- Storage move unit (SMU).
- Central memory control (CMC).

The CP is isolated from the IOU unit and is thus able to carry on computation or character manipulation unencumbered by I/O requirements.

Central Processing Unit (CPU)

The CPU is comprised of the following:

- Control section.
- Registers.
- Execution section.
- Instruction work stack (IWS).

Control Section

The control section directs the arithmetic and manipulative functions for instruction execution. The control section prefetches instruction words from memory and disassembles them into instructions.

Registers

Operating registers reduce the number of the number of storage accesses for operands used during the execution of an instruction. These registers are:

- Eight 60-bit X registers (X0 through X7) which hold operands used for computation.
- Eight 18-bit A registers (A0 through A7) which use A0 primarily for indexing and A1 through A7 for CM operand addressing.
- Eight 18-bit B registers (B0 through B7) which are primarily indexing registers to control program execution. The B0 register always contains all zeros.

Eight support registers support the operating registers during program execution. These registers are:

- 18-bit program address (P) register.
- 21-bit reference address for CM (RAC) register.
- 21-bit field length for CM (FLC) register.
- 6-bit exit mode (EM) register.
- 6-bit flag register.
- 21-bit reference address for extended memory (RAE) register.
- 24-bit field length for extended memory (FLE) register.
- 18-bit monitor address (MA) register.

The registers store data and control information, present operands to the execution section, and store results.

Execution Section

The execution section combines the operands to achieve the result.

Instruction Word Stack (IWS)

The IWS is a group of 12 60-bit registers that hold program instruction words for execution. It is essentially a moving window in the program code. The IWS fills two words ahead of the program address currently being executed. A program loop of up to 10 instruction words may be entirely contained within the IWS. When this happens, the instruction loop may be executed repeatedly without further references to the CM.

Functional Units

Each CP functional unit is an independent, specialized arithmetic unit with an algorithm for performing a portion of the CP instructions in a fixed amount of time (refer to Functional Units in section 2).

Storage Move Unit (SMU)

The SMU performs all block copy transfers to/from unified extended memory and external extended memory. The SMU also performs the single-word transfers to/from EEM.

Central Memory Control (CMC)

The CMC, which is part of the CP chassis, controls the flow of data between the CM and the requesting system components. The CMC also generates and checks parity on addresses transferred within the system.

CENTRAL MEMORY

The CM (figure 1-5) consists of the following:

- Eight memory banks (model 865).
- Sixteen memory banks (model 875).

The model 865 CM is a static metal-oxide semiconductor (MOS) memory, which is organized into eight independent banks. The model 875 CM is a bipolar memory, which is organized into 16 independent banks.

EXTENDED MEMORY

Extended memory (figure 1-6) is used for its large storage capacity and high transfer rates. When a program requests extended memory, a block is assigned for the duration of the program. The block is designated by the RAE and FLE registers, which are included in the CP exchange package. In addition to block copies, CP instructions may read/write single words in extended memory. Two forms of extended memory exist for the models 865 and 875: unified extended memory (UEM) and the optional external extended memory (EEM).

Unified Extended Memory (UEM)

The models 865 and 875 CM may be logically partitioned to have a portion reserved for extended memory, called UEM. This partition is set by the operating system software; there is no physical distinction. The amount of UEM available matches CM memory size, or a maximum of 1024K words. Refer to the COMPASS Version 3 Reference Manual, publication number 60492600, for further information.

External Extended Memory (EEM)

The models 865 and 875 offer EEM as an option. This memory is located in a physically separate cabinet. The EEM interface, required with the EEM option, expands the SMU to add an external interface to EEM. The models 865 and 875 support two versions of EEM:

- Extended semiconductor memory (ESM), with a maximum size of 16 million words.

- Extended core storage (ECS), with a maximum size of 2 million words.

ESM and ECS are commonly referred to as EEM in this manual, except where it is necessary to use the specific names to distinguish the two versions.

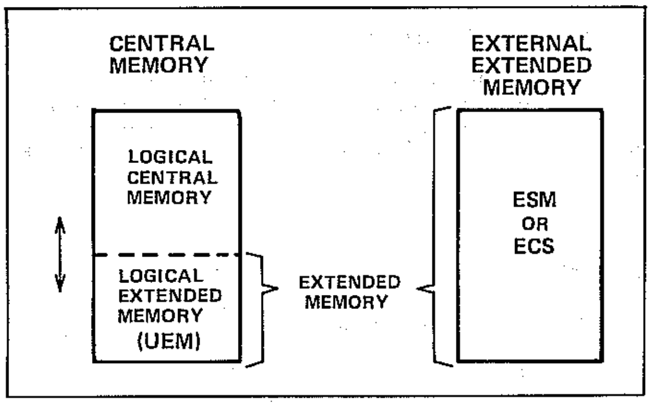


Figure 1-6. Extended Memory

ADDRESSING MODES

Two addressing modes are used to reference UEM and EEM. These are standard addressing mode and expanded addressing mode. The addressing modes differ in how RAE and FLE are defined in an exchange package. Refer to Extended Memory Addressing Modes, section 5, for complete description.

Standard Addressing Mode

This mode provides addressing up to 21 bits in a 24-bit format, or a maximum of 2 million 60-bit words.

Expanded Addressing Mode

This mode provides addressing up to 24 bits in a 30-bit format, or a maximum of 16 million 60-bit words.

INPUT/OUTPUT UNIT

The IOU (figure 1-5) consists of the following:

- Ten logically independent peripheral processors (PPs), called IOU-0. Options are available to increase total to 15 or 20 PPs, called IOU-1.

- Internal interface to 12 I/O channels. A 24-channel option is available.

- External interfaces to I/O channels

  - 11 or 23 channel interfaces.

  - Display controller interface (channel 10_8).

  - Real-time clock interface (channel 14_8).

  - Communications interface (channel 15_8).

- Interface to central memory.

IOU-0 and the optional IOU-1 consist of PPs organized in groups of ten, called barrels. The PPs in a barrel time-share common hardware. Each PP has its own independent memory, and communicates with all I/O channels and with central memory.

DISPLAY STATION

The display station provides a visual, alphanumeric readout for the computer. The receipt of symbol and position information from the computer enables displaying program information on a 21-inch cathode-ray tube (CRT). The station also contains an alphanumeric keyboard which enables an operator to send data to the computer. The keyboard and CRT combination permits the computer operator to modify computer programs and view the result on the screen. The computer outputs two alternate, nonrelated data streams. The display station keyboard has a switch which enables the operator to select either of the data streams or to select both for presentation on the CRT. Except for programming information in section 5, refer to the display station manual listed in the preface for further display station information.

FUNCTIONAL DESCRIPTIONS

This section provides functional descriptions of the central processor (CP), central memory (CM), and input/output (IOU) parts as shown in block diagram in section 1.

Functional descriptions for the system display station and the cooling system are in their respective manuals listed in the preface.

CENTRAL PROCESSOR

The CP consists of a central processing unit (CPU), nine functional units, a storage move unit (SMU), and the central memory control (CMC).

CENTRAL PROCESSING UNIT

The CPU consists of the control section, 24 operating registers, and 8 support registers. The CPU includes the registers and control logic to direct the arithmetic operations and provide interface between the functional units and CMC. In addition to instruction execution, the CPU performs instruction fetching, address preparation, memory protection, and data fetching and storing. Figure 2-1 illustrates the general flow of information.

Program execution begins with execution of an exchange jump, which loads the CPU operating registers with a 16-word block from CM. An exchange jump also stores the original contents of the CPU operating registers in the same 16-word block in CM. The operating system can use an exchange jump to switch program execution between two CM programs, leaving the first program in a usable state for later reentry into the CPU. For further information, refer to Exchange Jump in section 5.

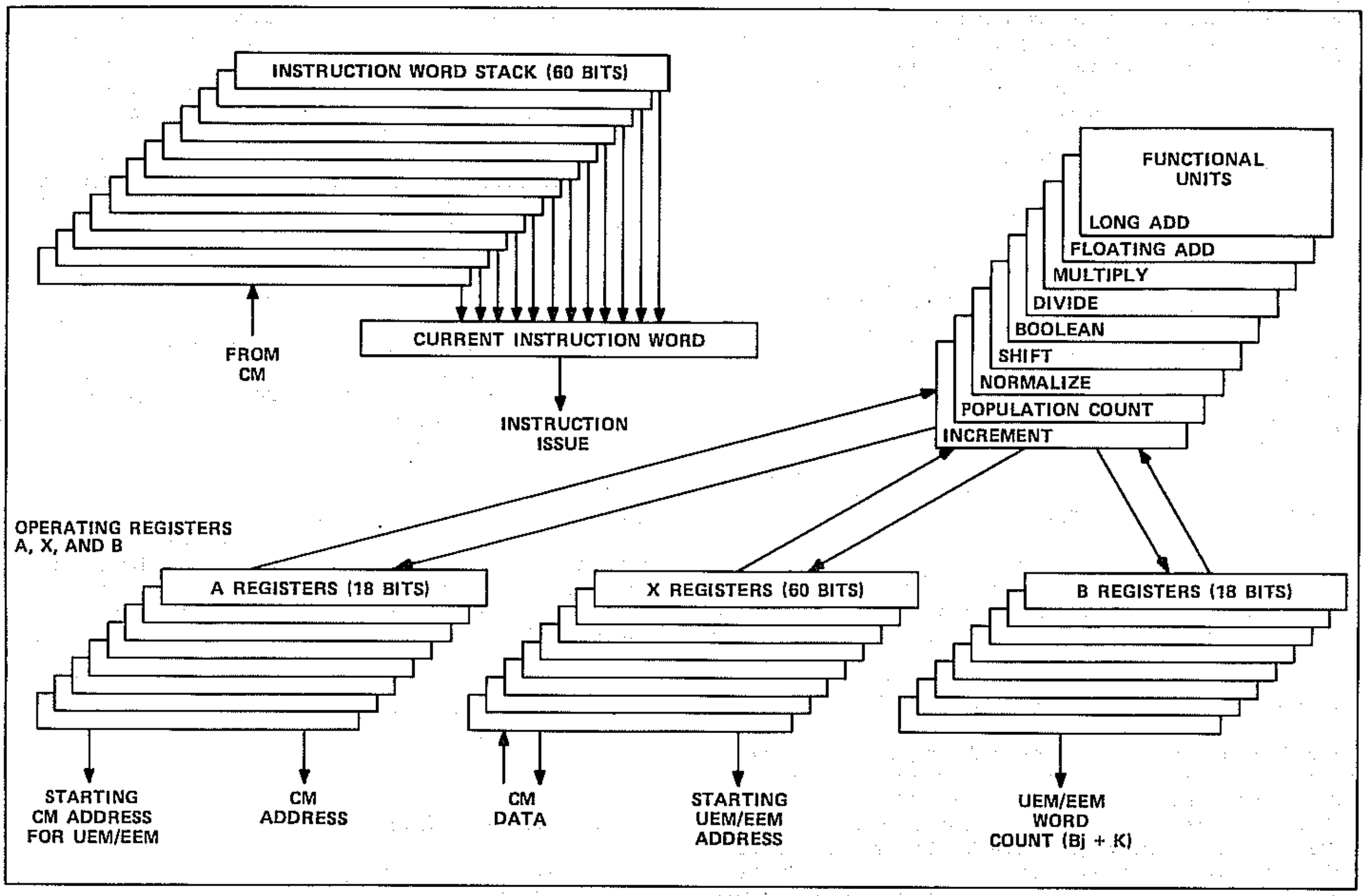


Figure 2-1. CP Information Flow

The CPU reads 60-bit words from CM and enters them in the instruction word stack (IWS), which is capable of holding up to twelve 60-bit words. Each instruction word, in turn, leaves the IWS and enters the current instruction word (CIW) register for interpretation and testing. The CIW register holds four 15-bit instructions, two 30-bit instructions, or combinations of the two types of instructions. The 15- or 30-bit instructions issue individually from the CIW register. The functional units obtain the instruction operands from and store results in 24 operating registers. Reservation control keeps an account of active operating registers to resolve conflicts.

Control Section

The instruction control section consists of instruction lookahead and instruction control sequences.

Instruction Lookahead

The instruction lookahead hardware consists of the instruction stack (IS) and the CIW register. The IS consists of the IWS and the instruction address stack (IAS).

The IS holds up to twelve 60-bit instruction words for program execution along with the relative CM address associated with each word. All instructions executed by the CPU must pass through the IS. The IS fetches and stores up to two instruction words ahead of program execution. The instruction stack also provides the capability of executing program loops (10-word limit) without referencing memory once the loop is loaded in the IS.

Instruction Word Stack

The IWS is a group of twelve 60-bit registers that hold program instruction words for execution by the CPU. The IWS is accessed by entering instruction fetch data and by reading instructions to be entered in the CIW. The IWS fills two words ahead of the program address currently being executed. A program loop of up to 10 instruction words may be entirely contained within the IWS. When this happens, the instruction loop may be executed repeatedly without further references to CM.

The 12 IWS registers are individually identified by rank. The rank 1 register contains the oldest data. In both IWS and IAS, the rank number and the physical register number may differ from each other. If the IWS contains sequential program instruction words, the rank 1 register corresponds to the lowest CM address in the IAS.

The IWS has separate read and write address ports such that data can enter and instructions can leave during the same clock cycle. The IWS write address comes from IAS control logic, and corresponds to the IAS register number (1 through 12) associated with the instruction stored in the IWS. The IWS read address is generated based on coincidence between the IAS rank address and the current program address. The IWS may be purged under certain conditions. Purging the stack means that the IWS is not accessible, and the IAS clears. New instructions must then be read from CM into the IWS and IAS. Purging the stack results from an exchange jump, return jump, jump to Bi plus K (instruction 02), or a branch to a location not in the stack (instructions 03 through 07). The stack always contains a sequential code.

The IWS shifts to accommodate a new word arriving from CM. New information arriving from CM enters rank 12. Ranks 11 through 1 clear and are entered with information from the next highest-order rank. The information in rank 1 discards.

Instruction Address Stack

The IAS is a group of twelve 18-bit address registers associated with the IWS. It holds relative CM program addresses on a one-for-one basis with the program words in the IWS. The IAS address registers also have twelve 18-bit comparators and their associated control, used for coincidence tests.

An address from the next stack address (NSA) register enters one of the 12 address registers as data corresponding to that address enters the IWS. IAS control chooses one of the 12 address registers with which to replace the oldest (rank 1) data and address. The selected register then becomes rank 12 and the other registers shift accordingly. A coincidence test occurs for each of the 12 IAS ranks, in which the rank addresses compare with the current program address. When coincidence occurs for a rank, the data in the corresponding IWS rank gates to the 60-bit CIW register.

IAS logic senses when a word has been read from the most recent entry (rank 12 coincidence). The instruction fetch control uses rank 11 and rank 12 coincidence terms to attempt to maintain instruction requests two instruction words ahead of program execution.

Each IAS address register has a full bit contained in a shift register. The full bit enables the coincidence test for that address register. As addresses are assigned, full bits shift in to indicate that the address register contains a valid address.

The IAS is purged by clearing the IAS full bits. A purged stack causes the next instruction (P + 1) to read from CM into the IWS before execution. After the stack is purged, instructions in remaining parcels of CIW [current instruction word (P)] execute without being reloaded from memory.

Purging of the IAS is caused by the following:

- Exchange jump execution (XSF).
- Return jump execution (010).
- Jump instruction execution (02).
- Any jump to an address not in IAS (jump out of stack condition).

- Extended memory (UEM/EEM) block read (011).
- Branch instructions 03 through 07 (branch taken and branch not taken) while stack purge flag set.
- Increment write instructions 5X6 and 5X7 while stack purge flag set.

A maintenance switch disables either IAS ranks 1 through 10 or ranks 1 through 4. Disabling the ranks prevents the full bits from setting for the disabled ranks.

Current Instruction Word (CIW) Register

The CIW register is divided into four 15-bit parcels. All four parcels load when an instruction word reads from the IWS. An instruction, consisting of one, two, or four parcels, issues from the CIW register when conditions in the functional units and operating registers permit the instruction to execute without conflicting with previously issued instructions.

Instruction Control Sequences

Block Copy Sequence

The storage move unit performs block copies of data between CM and extended memory. If the UEM flag is set, the copy is between CM and UEM. If the UEM flag is clear, the copy is between CM and EEM. The CM starting address is determined by A0 or the upper 30 bits of X0, depending on the state of the enhanced block copy flag. The number of words transferred is determined by the sum of (Bj) + K. The block copy instructions are:

011	Block copy (Bj) + K from UEM or EEM to CM	REC Bj + K
012	Block copy (Bj) + K from CM to UEM or EEM	WEC Bj + K

Direct Read/Write Sequence (CM, UEM)

The increment unit performs direct read/write sequences between CM and UEM. The direct read/write sequences are:

660	Read central memory at (Xk) to Xj	CR Xj,Xk
670	Write central memory at (Xk) from Xj	CW Xj,Xk
014	Read one word from UEM at (Xk + RAE) to Xj (UEM flag set)	RX Xj,Xk
015	Write one word from Xj to UEM at (Xk + RAE) (UEM flag set)	WX Xj,Xk

Direct Read/Write Sequence (EEM)

The storage move unit performs the direct read/write references to external extended memory. If the UEM flag is clear, the storage move unit performs single-word direct read and write operations to/from EEM. The direct read/write operations are:

014	Read one word from EEM at (Xk + RAE) to Xj (UEM flag clear)	RX Xj,Xk
015	Write one word from Xj to EEM at (Xk + RAE) (UEM flag clear)	WX Xj,Xk

Exchange Sequence

The exchange sequence generates timed CM reference signals to implement the exchange of data between the CP and CM, as required by the exchange jump instruction. In addition, the exchange sequence provides internal control signals to the operating and control registers to systematically enter the contents of an exchange jump package. For further information, refer to Exchange Jump in section 5.

Normal Jump Sequence

The normal jump sequence controls the execution of branch instructions 02 through 07. The 02 instruction performs an unconditional jump to the Bi register address plus K. The branch address is K when i equals 0. Bi + K and the P register are 18-bit quantities which restrict the executable portion of the program to the first 262K of relative CM addresses. The 02 instruction is:

02	Jump to (Bi) + K	JP

The conditional jump instructions 03 through 07 branch to address K if the jump condition is met. These instructions are:

030	Branch to K if (Xj) = 0	ZR
031	Branch to K if (Xj) = 0	NZ
032	Branch to K if (Xj) positive	PL
033	Branch to K if (Xj) negative	NG
034	Branch to K if (Xj) in range	IR
035	Branch to K if (Xj) out of range	OR
036	Branch to K if (Xj) definite	DF
037	Branch to K if (Xj) indefinite	ID
04	Branch to K if (Bi) = (Bj)	EQ
05	Branch to K if (Bi) ≠ (Bj)	NE
06	Branch to K if (Bi) ≥ (Bj)	GE
07	Branch to K if (Bi) < (Bj)	LT

Return Jump Sequence

The return jump sequence controls the execution of three instructions.

00	Error exit to MA or program stop	PS
010	Return jump to K	RJ
013	Central exchange jump to (Bj) + K or monitor exchange to MA	XJ Bj + K

Registers

The CP contains the operating and support registers described in the following paragraphs. The contents of these registers can be written into memory and reloaded from memory as an exchange package by a single CP instruction (exchange jump).

The time an exchange package resides in CP hardware is called an execution interval. During this interval, the contents of X, A, B and P registers can be changed by CP instructions. The contents of other support registers change only as a result of a exchange jump.

Operating Registers

The operating registers consist of operand (X), address (A), and index (B) registers. These registers minimize memory references for arithmetic operands and results.

X Registers

The CP contains eight 60-bit X registers, X0 through X7. The X0 through X7 registers are primarily data handling registers for computation. Operands and results transfer between CM and the X registers as a result of placing CM addresses into corresponding A registers.

The X0 register provides the relative starting address in extended memory for normal block copy transfers, and in CM for an enhanced block copy transfer (refer to section 5). X1 through X5 are used to input data from CM, and X6 and X7 are used to transmit data to CM. X0 through X7 can also directly reference CM using the Xk register as an address to read or write data to or from the Xj register.

A Registers

The CP contains eight 18-bit A registers, A0 through A7. The A0 register serves as an intermediate register for the user's discretion. Also, the A0 register provides the relative CM starting address in a normal block copy operation.

The A1 through A7 registers are essentially CM operand address registers associated one-for-one with the X registers. Placing a quantity into an address register (A1 through A5) causes a CM read reference to that address and transmits the CM word to the corresponding X register (X1 through X5). Similarly, placing a quantity into the A6 or A7 register causes the word in the corresponding X6 or X7 register to be written into that relative address of CM.

B Registers

The CP contains eight 18-bit B registers, B0 through B7. These registers are primarily indexing registers to control program execution. Program loop counts may also be incremented or decremented in these registers.

Program addresses may be modified on the way to an A register by adding or subtracting B register quantities. The B registers also hold shift counts for the nominal Bj shifts, the resultant exponent for the unpack, the operand exponent for the pack, and the resultant shift count from a normalize. The B0 register always contains positive zero which can be used as an operand. This register cannot hold results from instructions.

Support Registers

Eight support registers assist the operating registers during the execution of programs. The contents of the support registers are stored in CM, and their new contents are loaded from CM during an exchange sequence. With the exception of the program address (P) register, the contents of the support registers cannot be altered during the execution interval of an exchange package. When the execution interval completes, the data in the support registers is sent back to CM through an exchange jump.

P Register

The 18-bit P register loads from CM during the first word of an exchange sequence and contains the current program execution address. The register serves as a program address counter and holds the relative CM address for each program step.

RAC Register

The 21-bit CM reference address (RAC) register loads from CM during the second word of an exchange sequence. An absolute CM address forms by adding RAC to a relative address determined by the instruction. The content of the P register is added to RAC to form the program address in CM. A P-equal-to-zero condition specifies relative address zero and, therefore, (RAC). This CM location is reserved for recording error exit conditions and should not be used to store data or instructions.

FLC Register

The 21-bit CM field length (FLC) register loads from CM during the third word of an exchange sequence. The FLC register defines the size of the field of the program in execution. Relative CM addresses are compared with FLC to check that the program is not going out of its allocated memory range.

EM Register

The 6-bit exit mode (EM) register loads from CM during the fourth word of an exchange sequence. The EM register holds 6 exit mode selection bits that control individual error conditions for a program. Selected EM register bits cause the CP to error exit when the corresponding conditions occur. Any or all of the 6 bits can be set at one time. Clear EM register bits allow the CP to continue, without error processing, when most of the corresponding conditions occur. The exit mode selection bits appear in the exchange package as bits 48 through 50, and 57 through 59. The bits and their corresponding conditions are:

Mode Selection Bit	Condition Sensed
48	Address out of range.
49	Infinite operand.
50	Indefinite operand.
57	EEM flag function parity error or direct parity error.
58	CMC input error.
59	CM data error.

Flag Register

The 6-bit flag register loads from CM during the fourth word of an exchange sequence. The flag register holds 6 bits that function as control flags.

Bits	Condition
51	Not used.
52	Instruction stack purge flag. If set, enables extended purging of instruction stack.
53	Not used.
54	Enhanced block copy flag. If set, block copy instructions (011, 012) use bits 30 through 50 of X0 rather than A0 to determine the CM address.
55	Expanded addressing select flag. If set, UEM or EEM operate in expanded addressing mode; if clear, UEM or EEM operate in standard addressing mode.
56	UEM enable flag. If set, UEM is used. If clear, EEM is used.

RAE Register

The RAE register contains the relative address for extended memory references (UEM or EEM). The 21-bit UEM reference address (RAE) register loads from CM during the fifth word of an exchange sequence. The lower 6 bits of this register are always zero. An absolute UEM or EEM address forms by adding RAE to the relative address which is determined by the instruction. The addressing mode selection is made via the expanded addressing select flag, which is bit 55 of the third word of an exchange sequence. This flag selects standard or expanded addressing for extended memory direct read/write and block copy transfers.

Extended memory references use EEM if UEM mode is not selected. Standard addressing mode is used if expanded addressing mode is not selected. In any extended memory reference, RAE contents add to extended memory addresses specified by instructions to form an absolute extended memory address. If the UEM flag is clear, the absolute extended memory address is located in EEM. If the UEM flag is set, the absolute extended memory address is located in CM.

FLE Register

The 24-bit extended memory field length (FLE) register loads from CM during the sixth word of an exchange sequence. The lower 6 bits of this register are always zero. The FLE register defines the size of the field in UEM or EEM for the program in execution. Relative UEM or EEM addresses are compared with FLE.

MA Register

The 18-bit monitor address (MA) register loads from CM during the seventh word of an exchange sequence. The MA register contains the absolute starting address of an exchange package which is used when executing a central exchange jump (013) instruction with the monitor flag set, or when honoring a monitor exchange jump to MA (262x) instruction with the monitor flag clear. For further information, refer to Exchange Jump in section 5.

FUNCTIONAL UNITS

Each of the nine functional units in the CP is a specialized arithmetic unit with an algorithm for performing a portion of the CP instructions. Each unit is independent of the other units, and a number of functional units may be in operation at the same time. No visible registers are in the functional units from a programming standpoint. A functional unit receives one or two operands from operating registers at the beginning of instruction execution and delivers the result to the operating registers when the function has been performed. No information is retained in a functional unit for reference in subsequent instructions.

All functional units, with the exception of the multiply and divide units, have a 1-clock-period segmentation. This means that the information arriving at a unit, or moving within a unit, is captured and held in a new set of registers every clock period. It is possible to start a new set of operands for unrelated computation in a functional unit each clock period even though the unit may require more than 1 clock period to complete the calculation. This process may be compared to a delay line in which data moves through the unit in segments to arrive at the destination in the proper order but at a later time. All functional units perform their algorithms in a fixed amount of time. No delays are possible once the instruction issues.

The multiply unit has a 2-clock-period segmentation. Operands may enter the multiply unit in any clock period providing there was no multiply instruction initiated in the preceding clock period. There is a 1-clock-period delay in initiating a multiply instruction if another multiply instruction has just started.

The divide unit is the only functional unit in which an iterative algorithm executes. No segmentation is possible in this unit although the beginning of a new operation can overlap the completion of the previous operation by 2 clock periods.

Boolean Unit

The Boolean unit performs instructions that require bit-by-bit data manipulation. This includes both the logical and transmissive operations. The instructions requiring logical operations are:

11	Logical product (Xj) and (Xk)to Xi	BXi Xj * Xk
12	Logical sum of (Xj) and (Xk) to Xi	BXi Xj + Xk
13	Logical difference of (Xj) and (Xk) to Xi	BXi Xj - Xk
15	Logical product of (Xj) with complement of (Xk) to Xi	BXi -Xk * Xj
16	Logical sum of (Xj) with complement of (Xk) to Xi	Xi -Xk + Xj
17	Logical difference of (Xj) with complement of (Xk) to Xi	BXi -Xk - Xj

The instructions requiring transmissive operations are:

10	Transmit (Xj) to Xi	BXi Xj
14	Transmit complement of (Xk) to Xi	BXi - Xk

Shift Unit

The shift unit performs instructions that require shifting the 60-bit field of data within the operand word. The shift instructions are:

20	Left shift (Xi) by jk	LXi jk
21	Right shift (Xi) by jk	AXi jk
22	Left shift (Xk) nominally (Bj) places to Xi	LXi Bj Xk
23	Right shift (Xk) nominally (Bj) places to Xi	AXi Bj Xk
43	Form mask of jk bits to Xi	MXi jk

The shift unit also controls the pack and unpack instructions. In the packed floating format, the coefficient is contained in the lower 48 bits. The sign and biased exponents are contained in the upper 12 bits. The unpack instruction obtains the packed word from the Xk register, delivers the coefficient to the Xi register, and delivers the exponent to the Bj register. The unpack and pack instructions are:

26	Unpack (Xk) to Xi and Bj	UXi Bj Xk
27	Pack (Xk) and (Bj) to Xi	PXi Bj Xk

Normalize Unit

The normalize unit executes instructions that require rearranging operands in floating-point format. The unit left shifts the coefficient so that the most significant bit shifts into the highest-order bit position of the coefficient. The exponent adjusts by subtracting the shift count. The normalize instructions are:

24	Normalize (Xk) to Xi and Bj	NXi Bj Xk
25	Round normalize (Xk) to Xi and Bj	ZXi Bj Xk

Floating-Add Unit

The floating-add unit performs the operations necessary to form the 48-bit floating sum with a 12-bit exponent of the floating-point sum or difference of two floating-point operands. The floating-add instructions are:

30	Floating sum of (Xj) and (Xk) to Xi	FXi Xj + Xk
31	Floating difference of (Xj) and (Xk) to Xi	FXi Xj - Xk
32	Floating double-precision sum of (Xj) and (Xk) to Xi	DXi Xj + Xk
33	Floating double-precision difference of (Xj) and (Xk) to Xi	DXi Xj - Xk

34	Round floating sum of (Xj) and (Xk) to Xi	RXi Xj + Xk
35	Round floating difference of (Xj) and (Xk) to Xi	RXi Xj - Xk

Long-Add Unit

The long-add unit executes instructions that require integer addition of two 60-bit operands. The long-add instructions are:

36	Integer sum of (Xj) and (Xk) to Xi
37	Integer difference of (Xj) and (Xk) to Xi

Multiply Unit

This unit performs multiply operations.

The multiply instructions are:

40	Floating product of (Xj) and (Xk) to Xi	FXi Xj * Xk
41	Round floating product of (Xj) and (Xk) to Xi	RXi Xj * Xk
42	Floating double-precision product of (Xj) and (Xk) to Xi	DXi Xj * Xk

Divide Unit

The divide unit executes instructions that require division of two operands in floating-point format. The divide instructions are:

44	Floating divide (Xj) by (Xk) to Xi	FXi Xj/Xk
45	Round floating divide (Xj) by (Xk) to Xi	RXi Xj/Xk

Population-Count Unit

The population-count unit executes an instruction that requires counting the number of one bits in a 60-bit operand. The population-count instruction is:

47	Population count of (Xk) to Xi	CXi Xk

Increment Unit

The increment unit executes instructions 50 through 77 that require arithmetic operations on two 18-bit operands. During the 50 through 57 instructions, the result transmits to an A register. The same result plus RAC is sent to CM for the increment read or write address. The increment unit also directs the read/write sequences with central/extended memory [014 (UEM), 015 (UEM), 660, 670].

The increment instructions are:

50	Set Ai to (Aj) + K	SAi Xj K
51	Set Ai to (Bj) + K	SAi Bj K
52	Set Ai to (Xj) + K	SAi Xj K
53	Set Ai to (Xj) + (Bk)	SAi Xj + Bk
54	Set Ai to (Aj) + (Bk)	SAi Aj + Bk
55	Set Ai to (Aj) - (Bk)	SAi j - Bk
56	Set Ai to (Bj) + (Bk)	SAi Bj + Bk
57	Set Ai to (Bj) - (Bk)	SAi Bj - Bk
60	Set Bi to (Aj) + K	SBi Aj K
61	Set Bi to (Bj) + K	SBi Bj K
62	Set Bi to (Xj) + K	SBi Xj K
63	Set Bi to (Xj) + (Bk)	SBi Xj + Bk
64	Set Bi to (Aj) + (Bk)	SBi Aj + Bk
65	Set Bi to (Aj) - (Bk)	SBi Aj - Bk
66	Set Bi to (Bj) + (Bk)	SBi Bj + Bk
67	Set Bi to (Bj) - (Bk)	SBi Bj - Bk
70	Set Xi to (Aj) + K	SXi Aj K
71	Set Xi to (Bj) + K	SXi Bj K
72	Set Xi to (Xj) + K	SXi Xj K
73	Set Xi to (Xj) + (Bk)	SXi Xj + Bk
74	Set Xi to (Aj) + (Bk)	SXi Aj + Bk
75	Set Xi to (Aj) - (Bk)	SXi Aj - Bk
76	Set Xi to (Bj) + (Bk)	SXi Bj + Bk
77	Set Xi to (Bj) - (Bk)	SXi Bj - Bk

STORAGE MOVE UNIT (SMU)

The SMU (not a functional unit) performs the direct read/write references to EEM, and performs all block copy sequences to/from UEM and EEM. Refer to Block Copy Sequence and Direct Read/Write Sequence (EEM) in this section for additional information.

CENTRAL MEMORY CONTROL (CMC)

The CMC provides an interface between CM, CP and the IOU. The CMC has provisions to allow connection with a second CP/CMC for dual CP systems. CMC provides the following functions:

- Assigns priority to CM requests from CP, IOU and extended memory.
- Resolves CM bank conflicts including bank busy and reservations.

- Provides control for CM read/write data.
- Increments addresses for exchange jumps and extended memory transfers.
- Controls data transfers, during an exchange jump, between CM and CP.
- Parity checks addresses from IOU.
- Parity checks data from IOU and EEM.
- Generates parity (parity mode only) on data to IOU and EEM.
- Generates parity for address to CM.
- Breakpoint checks.
- Controls CM reconfiguration.
- Resolves conflicts caused by dual CP memory requests.
- Contains 21-bit address.
- Provides memory degrade selections.

SECDED

The SECDED logic corrects single-bit errors during a CM read, permitting unimpeded computer operation (figure 2-2). The SECDED logic prepares for the error correction by generating error correction code (ECC) bits for each data word, and by storing these ECC bits in CM with the data word during the CM write. Table 2-1 lists the hexadecimal codes for all the combinations of syndrome bits with the number of the data bit assigned each code or a note categorizing the code. Then, during a CM read, CM performs the following SECDED sequence.

1. Read one CM word and generate new ECC bits for data portion of CM word.
2. Compare new ECC bits with CM word ECC bits.
3. If old and new ECC bits match, no error exists. Send data to requesting unit.
4. If bits do not match, generate syndrome bits from result of ECC compare.
5. Decode syndrome bits to determine if single or multiple bit failure.
6. If single bit failure, correct by inverting failing bit in data word. Send corrected word to requesting unit.
7. If multiple bit or other uncorrectable error, send uncorrectable error response code to CP or IOU. A PP in the IOU may then analyze the syndrome bits using the maintenance channel.

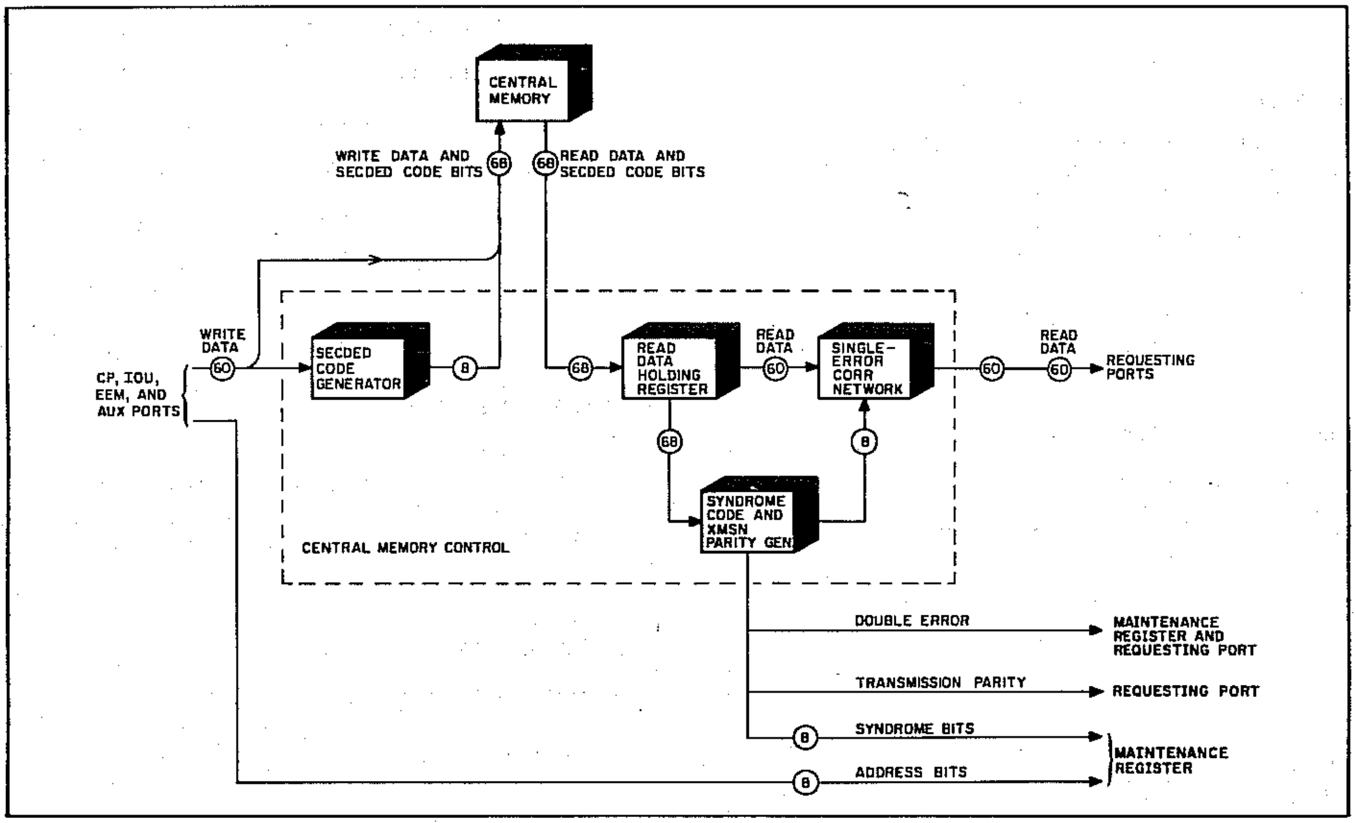

CENTRAL MEMORY
WRITE DATA AND SECDED CODE BITS
68
68
READ DATA AND SECDED CODE BITS
WRITE DATA
60
CP, IOU, EEM, AND AUX PORTS
SECDED CODE GENERATOR
8
68
READ DATA HOLDING REGISTER
READ DATA
60
SINGLE-ERROR CORR NETWORK
60
READ DATA
60
REQUESTING PORTS
68
8
SYNDROME CODE AND XMSN PARITY GEN
CENTRAL MEMORY CONTROL
DOUBLE ERROR
MAINTENANCE REGISTER AND REQUESTING PORT
TRANSMISSION PARITY
REQUESTING PORT
8
SYNDROME BITS
8
ADDRESS BITS
MAINTENANCE REGISTER

Figure 2-2. SECDED Network Block Diagram (SECDED Mode)

Table 2-1. SECDED Syndrome Codes/Corrected Bits

Code	Bit	Code	Bit	Code	Bit	Code	Bit	Code	Bit	Code	Bit	Code	Bit	Code	Bit
000	⑤	040	65①	100	66①	140	②	200	67①	240	②	300	②	340	50
001	60①	041	②	101	②	141	53	201	②	241	57	301	58	341	②
002	61①	042	②	102	②	142	54	202	②	242	59	302	④	342	②
003	②	043	0	103	1	143	②	203	2	243	②	303	②	343	③
004	62①	044	②	104	②	144	40	204	②	244	④	304	④	344	②
005	②	045	23	105	3	145	②	205	5	245	②	305	②	345	③
006	②	046	22	106	8	146	②	206	9	246	②	306	②	346	③
007	10	047	②	107	②	147	③	207	②	247	44	307	③	347	②
010	63①	050	②	110	②	150	41	210	②	250	43	310	48	350	②
011	②	051	47	111	7	151	②	211	6	251	②	311	②	351	28
012	②	052	27	112	31	152	②	212	11	252	②	312	②	352	③
013	13	053	②	113	②	153	③	213	②	253	③	313	③	353	②
014	②	054	29	114	30	154	②	214	16	254	②	314	②	354	③
015	17	055	②	115	②	155	③	215	②	255	③	315	③	355	②
016	18	056	②	116	②	156	③	216	②	256	③	316	③	356	②
017	②	057	③	117	52	157	②	217	③	257	②	317	②	357	③
020	64①	060	②	120	②	160	42	220	②	260	45	320	49	360	②
021	②	061	46	121	51	161	②	221	56	261	②	321	②	361	③
022	②	062	32	122	55	162	②	222	15	262	②	322	②	362	③
023	14	063	②	123	②	163	③	223	②	263	③	323	36	363	②
024	②	064	33	124	35	164	②	224	39	264	②	324	②	364	20
025	19	065	②	125	②	165	③	225	②	265	③	325	③	365	②
026	21	066	②	126	②	166	③	226	②	266	③	326	③	366	②
027	②	067	③	127	③	167	②	227	③	267	②	327	②	367	③
030	②	070	34	130	37	170	②	230	38	270	②	330	②	370	③
031	24	071	②	131	②	171	③	231	②	271	③	331	③	371	②
032	25	072	②	132	②	172	12	232	②	272	③	332	③	372	②
033	②	073	③	133	③	173	②	233	③	273	②	333	②	373	③
034	26	074	②	134	②	174	③	234	②	274	③	334	③	374	②
035	②	075	4	135	③	175	②	235	③	275	②	335	②	375	③
036	②	076	③	136	③	176	②	236	④	276	②	336	②	376	③
037	③	077	②	137	②	177	③	237	②	277	③	337	③	377	②

Syndrome codes are octal representations of eight syndrome code bits. Circled numbers in the bit columns refer to the following.

① Syndrome code bit failed (single code bit set).
② Double error or multiple error (even number of code bits set).
③ Multiple error reported as single error (five or seven code bits set).
④ Not used because of 64-bit algorithm.
⑤ No error detected.

ERROR DETECTION AND RESPONSE

CMC checks for address parity errors, data parity errors, SECDED errors, and breakpoint conditions. When errors occur or breakpoint conditions are met, information is sent to the maintenance register and to the requesting port. Refer to figure 2-3 for all CMC error communications.

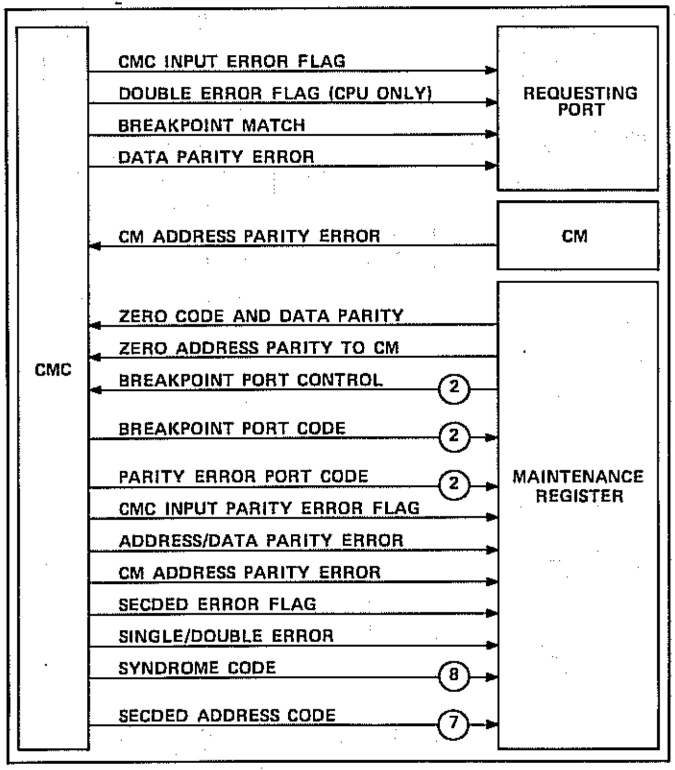


Figure 2-3. CMC Error Communications

Address Parity

The CMC checks parity on the address paths from:

- IOU-0 (PPs 1-10).
- IOU-1 (PPs 11-20), if present.

If an address parity error occurs at the CMC, applicable error information is sent to the maintenance register as follows:

- CMC input parity error flag.
- Requesting port code.
- Address error.

If the address parity error occurs on a write request, the write signal is blocked (not sent to CM) to protect memory.

If the address parity error occurs on a read request, the read data is replaced by a word of all ones.

Address parity is generated in CMC for the address going to CM. If CM detects an error, the error signal is sent back to CMC. The CMC then sends a CM address parity error to the maintenance register.

If the CP is the requesting port to CM, a CMC error signal sets the parity error condition. If the exit mode bit 59 sets, additional action is taken in the CP. Refer to Error Response under CP Programming in section 5 for additional information.

Data Parity

The CMC checks parity on the data paths from:

- IOU-0 (PPs 1-10) to CMC.
- IOU-1 (PPs 11-20), if present, to CMC.

If a data parity error occurs at the CMC, a CMC input error signal is sent to the requesting port which initiated the reference, and applicable error information is sent to the maintenance register as follows:

- CMC input parity error flag.
- Requesting port code or EEM error flag.

The previous signals are the same as the address parity information with the exception of the address error. The absence of the address error indicates a data error. Refer to Maintenance Register in section 5 for additional parity information.

In parity mode on a write operation, the data parity generates in the CMC for transmission to CM and substitutes in place of SECDED code bit 0.

Parity is checked on the data from CM, and code bit 0 is used as the parity bit. When a parity error occurs, the CMC sends only an error signal to the CPU if the CPU is the requesting unit. For other ports, the parity bit propagates for interrogation by the requesting unit.

In SECDED mode on a write operation, data parity is checked at the input requesting ports (except for the CPU and SMU). SECDED code bits then generate for transmission to CM.

In SECDED mode on a read operation, all models send data through the SECDED network. A parity bit is generated in CMC and transmits to the requesting unit (except the CPU) along with the read data.

CENTRAL MEMORY

The model 865 CM consists of a single chassis in bay 1. The memory is organized into eight banks of static metal-oxide semiconductor (MOS) memory. The basic memory size is 262K words, expandable to one million words in 262K increments. The word size is 60 bits plus eight SECDED bits.

The model 865 CM provides the following performance features:

- 50-nanosecond data transfer rate.
- 200-nanosecond memory cycle time.
- 375-nanosecond CP increment read to X for single and dual CP system.

The model 875 CM consists of a single chassis in bay 4. The basic memory size is 262K words, expandable to one million words in increments of 262K. The CM is accessed through up to four ports. Each port accesses an independent 262K words of high-speed bipolar memory. Each 262K increment adds a memory port. The word size is 60 bits plus eight SECDED bits.

The model 875 CM provides the following performance features:

- 25-nanosecond data transfer rate (12.5-nanosecond rate on dual CP systems where two ports are referenced in parallel).
- 75-nanosecond memory cycle time.
- 200-nanosecond CP increment read to X for single and dual CP system.

CM PRIORITIES

Model 865

CM references from central memory control (CMC) normally pass through the storage address stack (SAS) to CM. The SAS is a group of three registers which can stack up to three addresses for buffering of memory references, if necessary. Addresses enter SAS in the order in which they arrive from CMC. CMC requests received simultaneously are given a priority which determines the sequence of CM access via SAS. These priorities to SAS are:

- CPU return jump, exchange jump - IOU-0 (PPs 1-10).
- CPU return jump, exchange jump - IOU-1 (PPs 11-20, if present).
- Increment, storage move (CP).
- Instruction fetch (CP).

All memory references appear the same to CM. CMC provides hardware tags to identify the source or destination of any CM word reference. A CM reference sends the address to all CM banks, where the selected bank accepts the address. If the CM address is unavailable, an address backup condition occurs in SAS. Address backup results from the stacking of two or more words in the three-word SAS due to reference conflicts. Address backup disables subsequent requests to CM. These requests represent issue of instructions 50 through 57, 660, 670, 014 and 015. Storage move and IOU requests are also not honored. Address backup is caused by the following reference conflicts:

- Bank busy.
- Simultaneous CM requests from each CP in a dual CP system.

If the selected bank is busy, the request waits in the SAS until that bank is free and the address is available. Simultaneous requests from each CMC in a dual CP system result in an arbitrary assignment of priority to each CMC. Loss of priority appears the same as a bank busy for 50 nanoseconds or two clock cycles. Once a backup condition occurs, up to two requests may be waiting in SAS at the same time. Instruction issue does not start again until all addresses in SAS are accepted by CM.

CMC provides eight bank busy registers to correspond to the eight banks of CM. The bank busy registers set when a memory reference initiates for a particular bank, and remain set until the CM bank becomes available for another request.

Model 875

CM references from CMC normally pass through the SAS to all CM banks with ports installed. The SAS is a group of three registers which can stack up to three addresses for buffering of memory references, if necessary. Addresses enter SAS in the order in which they arrive from CMC. CMC requests received simultaneously are given a priority which determines the sequence of CM access via SAS. These priorities to SAS are:

- CPU return jump, exchange jump - IOU-0 (PPs 1-10).
- CPU return jump, exchange jump - IOU-1 (PPs 11-20, if present).
- Increment, storage move (CP).
- Instruction fetch (CP).

All memory references appear the same to CM. CMC provides hardware tags to identify the source or destination of any CM word reference. A CM reference sends the address to all ports of CM banks. The selected bank in the port with a request to one of its two cages accepts the address.

If the CM address is unavailable, an address backup condition occurs in SAS. Address backup results from the stacking of two or more words in the three-word SAS due to reference conflicts. Address backup disables subsequent requests to CM. These requests represent issue of instructions 50 through 57, 660, 670, 014 and 015. Storage move and IOU requests are also not honored. Address backup is caused by the following reference conflicts:

- Bank busy.
- Simultaneous CM requests from each CP in a dual CP system (single or multiple CM ports).

If the selected bank is busy, the request waits in the SAS until that bank is free and the address is available. Simultaneous requests from each CMC in a dual CP system with one CM port result in an arbitrary assignment of priority to each CMC. Simultaneous requests from each CMC in a dual CP system with multiple CM ports first cause a bank busy test so that priority is not assigned to a request to a busy bank. If the bank is not busy, the port conflict is resolved by assigning CP-0 priority to even-numbered ports and CP-1 priority to odd-numbered ports. A port conflict appears the same as a 25-nanosecond bank busy conflict. Loss of priority appears the same as a bank busy for 50 nanoseconds or two clock cycles. Once a backup condition occurs, up to two requests may be waiting in SAS at the same time. Instruction issue does not start again until all addresses in SAS are accepted by CM.

CMC provides 16 bank busy registers to correspond to the 16 banks of CM. The bank busy registers set when a memory reference initiates for a particular bank, and remain set until the CM bank becomes available for another request.

ADDRESS FORMATS

The address contained in SAS represents the address used to reference memory. Before addresses are sent to CM, bits 4, 5, 18 and 19 are operated on by the memory reconfiguration switches which contain information about the number of available memory quadrants. The memory address also includes three parity bits (parity B, C, and D) which are generated after the address is altered by memory reconfiguration.

Figures 2-4 and 2-5 show models 865 and 875 CM address formats before reconfiguration. The address formats are the same as the address format for a one-million-word system.

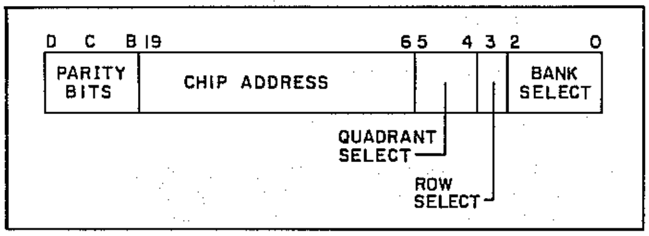


Figure 2-4. Model 865 CM Address Format

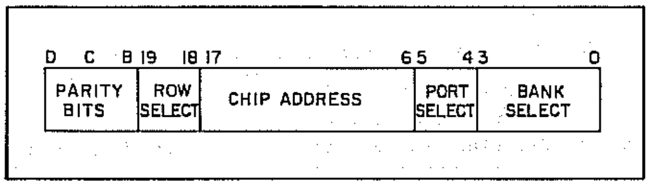


Figure 2-5. Model 875 CM Address Format

The following list defines the address formats:

Model 865

- Bank select specifies one of eight banks. Since the bank address is the lowest-order three bits of the storage address, sequential addressing results in a phased-bank operation. This allows a maximum data transfer rate of one word each 2 clock periods. A read or write bank cycle time is 8 clock periods.
- Chip address specifies the address of one word in 16K MOS memory chips for the selected bank.
- Row select selects one of four word rows in a quadrant.
- Quadrant select selects one of four quadrants. It is used only for storage units larger than 524K words.
- For a 262K system, the quadrant select is forced to quadrant zero and bits 4 and 5 are copied to bit positions 18 and 19.
- For a 524K system, bit 5 into quadrant selection is forced to zero such that only quadrants 0 and 1 are selected. Bit 5 is also copied to the bit 19 position.
- For a 786K system, bits 18 and 19 are used for quadrant selection if bits 4 and 5 specify quadrant 3.

Model 875

- Bank select specifies one of 16 banks. Since the bank address is the lowest-order three bits of the storage address, sequential addressing results in a phased-bank operation. This allows a maximum data transfer rate of one word each clock period. A read or write bank cycle time is 3 clock periods.
- Chip address specifies the address of one word in 4K bipolar memory chips for the selected bank.
- Row select selects one of four word rows in a port.
- Port select selects one of two ports. It is used only for storage units larger than 524K words.
- For a 262K system, the quadrant select is forced to port zero and bits 4 and 5 are copied to bit positions 18 and 19.
- For a 524K system, bit 5, port selection, is forced to zero such that only ports 0 and 1 are selected. Bit 5 is also copied to the bit 19 position.
- For a 786K system, bits 18 and 19 are used for port selection if bits 4 and 5 specify quadrant 3.

CENTRAL MEMORY RECONFIGURATION

Central memory reconfiguration is a manually performed function that permits the computer operator to restructure the CM addresses so that the failing part of CM can be quickly relocated to provide a continuous block of usable CM. CM reconfiguration consists of memory degrades, accomplished by setting switches to manipulate address bits 4, 5, 18, and 19 on chassis 5. Refer to Table 3-5, Central Memory Reconfiguration, for possible memory degrade options.

Certain restrictions exist for reconfiguration, depending on the location of the failing CM quadrant (model 865) or port (model 875):

- If the failing quadrant (port) is not the top 262K of CM, only one degrade from a nominal configuration is available.
- If the top 262K quadrant (port) is degraded, the next highest 262K quadrant (port), if failing, may be degraded since the first degrade reduced CM size but did not reconfigure CM.
- In a 1024K CM, a third degrade may be used if the first two degrades are performed on the top two quadrants (ports).

Memory reconfiguration and degrade sometimes result in a memory reference being made to a reconfigured quadrant (port), instead of the quadrant (port) originally specified by address bits 4, 5, 18, and 19. For these references, error address information pertains to the actual post-degrade memory address.

The models 865 and 875 do not support memory address wraparound. Memory references which exceed the amount of CM installed may result in an undefined operation. Predefined memory wraparound requirements are not required because the maintenance register bits $176/260_8$ through $182/266_8$ define how much CM is available. Refer to maintenance register bit assignments in section 5 for further detail.

INPUT/OUTPUT UNIT

The IOU performs the functions required to locate, select, and initialize the external devices connected to the system, and controls the transfer of data between a selected device and CM. The IOU also performs system maintenance functions. The IOU contains the following functional areas:

- Peripheral processor (PP).
- I/O channels.
- Real-time clock.
- Communications interface.
- Maintenance register.
- CM access.
- Data channel converter (DCC).

PERIPHERAL PROCESSOR

The basic IOU contains 10 PPs and can be expanded t 20 PPs in 5-PP increments. The first group of 10 PP is referred to as IOU-0. The optional second group, which may contain 5 or 10 PPs, is called IOU-1. Eac PP is a logically independent computer with its ow memory. Each 10-PP group is organized into multiplexing system which allows the PPs to shar common hardware for arithmetic, logical, and I/ operations without losing independence. Thi multiplexing system is comprised of 10 ranks o registers termed a barrel. Each rank contain information related to the instruction being execute by one PP.

Each PP can communicate with the CP through CM usin the exchange jump. The PPs may communicate with eac other over the I/O channels.

Each PP executes programs alone or with other PPs t control data transfers between external devices an CM. These programs are comprised of instructions fro the IOU instruction set and respond to requests issue through CM by the operating system. The program translate generalized operating system requests int control functions for accessing the external device and may also perform device scheduling an optimization. The programs use PP memory as a buffe for the data transfer between external devices and C to isolate IOU data transfer from variations in C transfer rate.

Deadstart

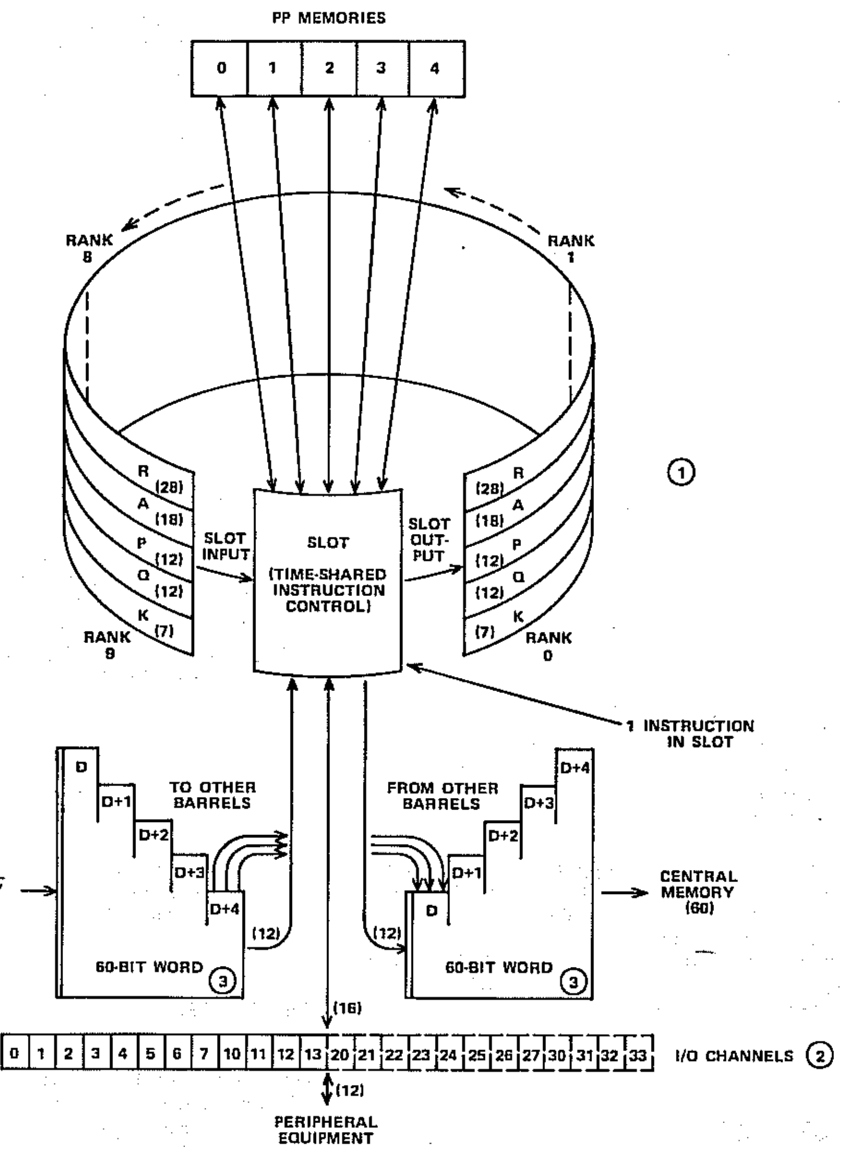
Deadstart sequence allows the IOU to initialize itself. Deadstart sequence is initiated by the DEA START switch on the deadstart panel or the DEAD STAR switch on the display station. The panel include controls for assigning any PPM to PPO.

Barrel and Slot

The barrel consists of the R, A, P, Q, and K registers, each one of which has 10 ranks 0 through 9 (figure 2-6). Information in these registers move from one rank to the next at a uniform 20-megahert rate, providing a multiplexed system of ten PPs, eac operating at a 2-megahertz rate. The registers are stationary while the PPs rotate. For example, rank 4 registers contain PP0 through PP9 in succession, eac consuming 50 nanoseconds of the total cycle time of 500 nanoseconds. Since PP memories operate at a slower rate, independent memory with its own address and data registers is provided for each PP.

Each time data enters the slot, a portion of the instruction for that data is executed. The slot performs tasks such as arithmetic and logic operations and program address manipulation. Complete execution of an instruction may require the R, A, P, Q, and K register quantities to go more than one trip around the barrel and through the slot.

The PPM may be referenced once each time the PP passes around the barrel and through the slot. During its slot time, the PP may also communicate with CM or with any of the I/O channels.



(1) OPTIONS OF 10, 15 OR 20 PPS AVAILABLE

(2) OPTIONS OF 12 OR 24 I/O CHANNELS AVAILABLE

(3) D IS THE ADDRESS OF THE FIRST PP WORD. THE LEFTMOST 4 BITS OF EACH CM WORD ARE UNDEFINED AND ARE IGNORED BY THE PP.

Figure 2-6. Barrel and Slot

PP Registers

R Register

The 21-bit R register, in conjunction with the A register, forms an absolute CM address for CM read/write instructions. When bit 17 of the A register is set, the absolute CM address is formed by appending six zeros to the lower end of the contents of the R register and adding to the result bits 0 through 16 of the contents of the A register. Refer to figure 2-7.

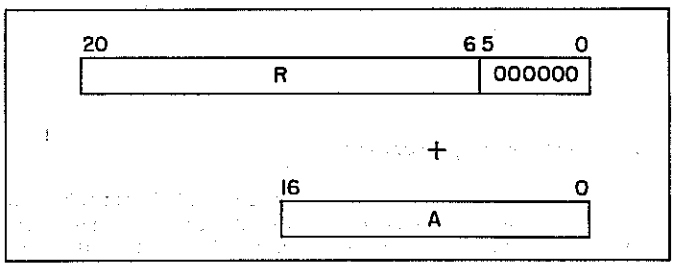


Figure 2-7. Formation of Absolute CM Address

A Register

The 18-bit A register holds one operand for arithmetic, logic, or selected I/O operations. The content of A may be an arithmetic operand, CM address, I/O function, or I/O data word. Various instructions operate on 6, 12, or 18 bits of the A register.

When the A register is used as the CM address, parity is generated for transmission with the address to central memory control. At deadstart, the A register is set to 10000 (octal). When bit 17 of the A register is clear, the A register is used as the CM address; however, when bit 17 is set, the R register is added to the A register as described above to obtain the absolute CM address for CM read/write instructions.

P Register

The 12-bit P register is the program address register, except during the execution of instructions 61, 63, 71, and 73. For these instructions, the P register contains the PPM address of the data transfer. At deadstart, the P register is set to zero.

Q Register

The 12-bit Q register holds data for several functions such as the address of the operand during direct addressing and indirect addressing, peripheral address of data used during one-word central read or write instructions, upper 6 bits during constant mode instructions, channel number on all I/O and channel instructions, shift count, and relative jump designator. At deadstart, each rank of the Q register is set to a corresponding PP number. Rank 0 is set to PP0, rank 2 is set to PP2, and so on.

K Register

The 9-bit K-register holds a 6-bit F portion of a instruction word and a 3-bit trip count. The tri count determines the operation of an instruction a different stages of completion. At deadstart, in loa mode the K register is forced to 710; in sweep mod the K register is forced to 505; and in dump mode th K register is forced to 730.

PP Numbering

PPs are numbered as follows:

Barrel	PPs
0	00 to 11 (octal)
1	20 to 31 (octal)

The deadstart sequence decodes deadstart panel switches to determine PP numbering within a barrel. The sequence assigns barrel numbers according to the switch settings and, during the first minor cycle after deadstart, loads a zero into the Q register in barrel 0. This defines all the data in that rank of the barrel as belonging to PP0 and since Q is the channel selector, assigns PP0 to channel 0. During the next minor cycle, Q loads with a 1. This defines PP1 and assigns it to channel 1. This process occurs in parallel in all barrels until the IOU assigns each rank of each barrel with a PP number and a channel number. Reassignment can be done only during a deadstart.

PP Memory

Each PP has an independent 4K word memory; each word contains 12 data bits and 1 parity bit. PP0 reads the deadstart program from the deadstart panel during the deadstart operation. Therefore, PP memory 0 must be operational. A PP memory reconfiguration feature allows the user to restore IOU operation if the IOU detects a fault in the PP memory normally assigned to PP0.

To reconfigure, the operator assigns a good PP memory to PP0 and the operating system removes the failing PP memory. Computer operation can continue without the failing PP memory, and repairs can be made during scheduled maintenance. The system must be deadstarted to reconfigure PPMs.

I/O CHANNELS

The I/O channels are comprised of an internal interface that allows common hardware and software to control the external devices, and an external interface that allows the IOU to communicate with the external devices using 12-bit data channels.

The internal interface can transfer 12-bit data words between two PPs, or between a PP and an external device at a maximum rate of one word every 500 nanoseconds. This rate can be sustained for the maximum practical channel transfer (4096 words). During transfers between PPs, if the PPs are in the slot at the same time, the transfer rate is 500 nanoseconds.

Any PP can access any of the bidirectional I/O channels. All PPs communicate with external devices through the independent I/O channels. Each channel may be connected to one or more pieces of external equipment, but only one piece of equipment can use a channel at one time. All channels can be active simultaneously.

The display station controller (DSC) is attached to channel 10_8. The DSC is the IOU interface between the PPs and the display station, servicing both the keyboard and the cathode-ray tube (CRT). It transmits function words and digital symbol size/position data to the display station, and receives digital character codes from the keyboard. It also receives digital symbol codes from the PPs and converts these to analog signals to the CRT.

REAL-TIME CLOCK

The real-time clock is a 12-bit free-running counter, incrementing at a 1-megahertz rate. It is permanently attached to channel 14_8. This channel may be read at any time as its active and full flags are always set.

COMMUNICATIONS INTERFACE

An RS232-C communications interface provides communication capability between a PP and one attached terminal. This interface is reserved for maintenance purposes. The communications interface is permanently attached to channel 15_8.

MAINTENANCE REGISTER

The maintenance register consists of the maintenance channel interface on channel 16_8 (IOU-0) and channel 36_8 (optional IOU-1), and a set of interconnecting cables. The maintenance register has bit assignments to monitor all parity error and SECDED networks, and also is the source of control for testing these networks. Status bits can be externally set. Control bits can be set and tested individually, or read as 12-bit words. Status bits can also be read in 12-bit words using function codes and an input instruction. Refer to Maintenance Register in section 5 for complete description and bit assignments.

CM ACCESS

Any PP can access CM. During a write from the IOU to CM, the IOU assembles five successive 12-bit PP words into a 60-bit CM word. During a CM read, the IOU disassembles the 60-bit CM word into five PP words. A PP forms a 21-bit CM address by adding the 21-bit base relocation address from the R register to the 17-bit relative address from the A register.

A maximum of 20 PPs in various stages of assembly/disassembly can simultaneously read CM words, and five PPs can write CM words. Each of the PPs transfers a 60-bit word to or from the CM every 50 nanoseconds.

DATA CHANNEL CONVERTER

Each system data channel converter (DCC) attaches to a data channel of the IOU. A DCC may share the data channel with up to seven other pieces of CDC 6000 Computer Systems peripheral equipment. As many as eight 3000 Computer Systems controllers can be attached to one DCC.

To prepare any of the 3000 Computer Systems equipment for operation, the DCC must first be selected. The desired 3000 Computer Systems equipment is selected (connected). The two select operations are made by function codes sent from a PP through the data channel. A data channel is part of the I/O channel that exists between a PP and external equipment which uses the same type transmitters and receivers for information interchange. The DCC differs from other 6000 Computer Systems equipment as follows:

- The DCC must be attached to the data channel before all other 6000 Computer Systems devices.
- The DCC does not relay (pass on) information to other equipment on the same data channel when selected. This prevents unwanted activity in the other equipment caused by identical function codes.
- The DCC must be deselected (2100) before other 6000 Computer Systems equipment sharing the data channel can be selected.
- A master clear (MC) signal on deadstart operations selects all DCCs in the computer system.

OPERATING INSTRUCTIONS

This section describes mainframe controls and indicators and the operating procedures which are hardware-dependent. Software-dependent procedures are in system software reference manuals listed in the preface.

CONTROLS AND INDICATORS

This section describes IOU deadstart panel controls and indicators and CM configuration switches used by the system operator. Other controls used by maintenance personnel are described in the hardware maintenance manuals of the power distribution and warning system, the cooling system, and the display console listed in the preface.

DEADSTART PANEL CONTROLS/INDICATORS

The deadstart panel for both models (figure 3-1) is in the IOU, located in bay 1 (refer to figures 1-3 and 1-4). The panel contains peripheral processor (PP) control switches which are only active during a deadstart. The switches and their functions are listed in table 3-1. The deadstart panel also contains PP register selection and display facilities, deadstart controls, error indicators, and a switch matrix, which is the source for a short PP program for initialization or troubleshooting. The switches, indicators, and their functions are listed in table 3-1.

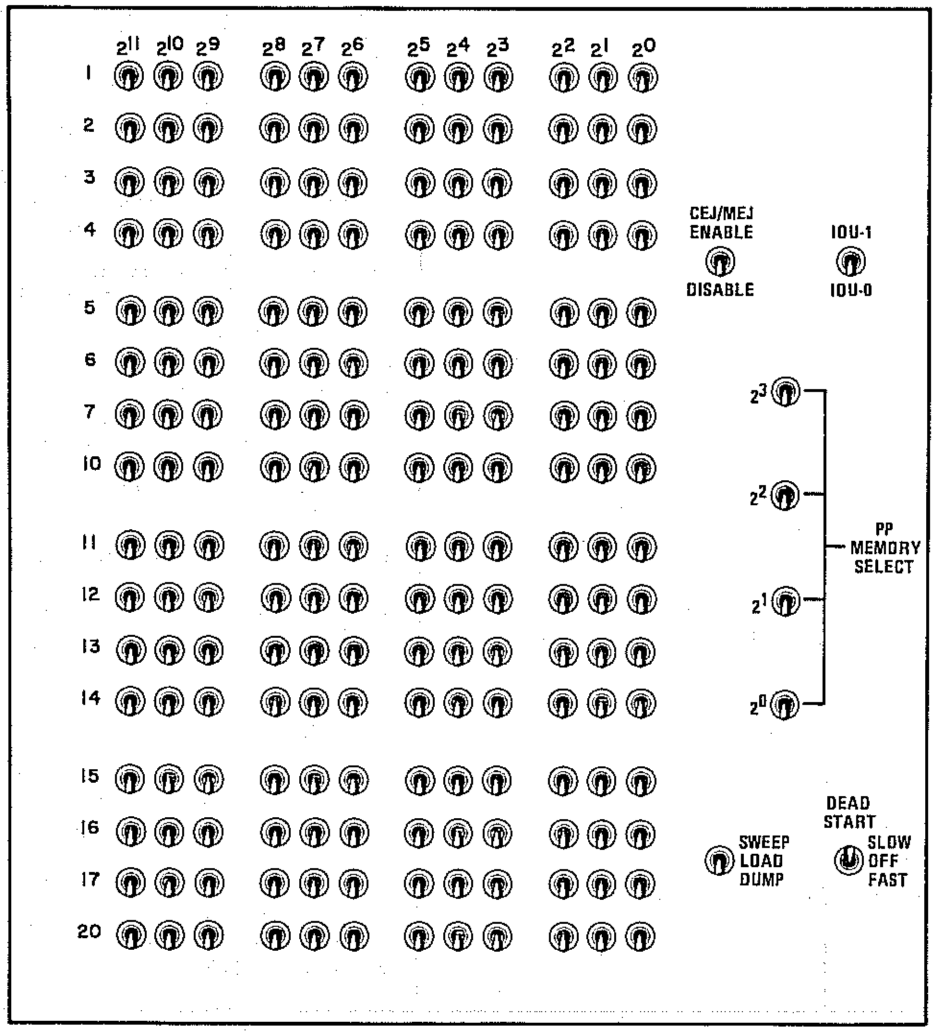


Figure 3-1. Deadstart Panel

Table 3-1. Deadstart Panel Controls/Indicators

Panel Nomenclature	Description	Function
2^0 through 20_{11} by 1 through 20	Toggle switch matrix (two-position switches)	Provides a 16-word deadstart program for PP0. Switches 2^0 through 2^{11} set 12 bits for each of the program words, labeled 1 through 20 (octal). Up position sets bit. Down position clears bit.
CEJ/MEJ ENABLE DISABLE	Toggle switch (two-position)	Enables or disables the central exchange jump (CEJ) instruction for the CP and the monitor exchange jump (MEJ) instructions for the peripheral processors (PPs). The switch position is set prior to a deadstart. Resetting the switch after a deadstart does not affect the computer operation until the next deadstart.
IOU-0 IOU-1	Toggle switches (two-position)	Select IOU-0 and IOU-1 to contain the controlling PP-0. For PP-0 to be in IOU-1,IOU-1 must contain all 10 PPs. Resetting the switch after a deadstart does not affect the computer operation until the next deadstart.
PP MEMORY SELECT 2^3, 2^2, 2^1, 2^0	Toggle switches (two-position)	Permit the assignment of any peripheral processor memory (PPM) to PP-0. PP-0 has a special control function at deadstart time. If the PPM for PP-0 malfunctions, the user may set the switches to assign any of the other nine PPMs to PP-0. The selection retains the PP order of rotation in the barrel and slot matrix (refer to Barrel and Slot in section 2). The assignment is made by enabling the switches to form a binary number of the PPM chosen for PP-0 (for example, 0101 selects PPM-5). These switches do not affect the IOU-1 chassis unless that chassis contains all 10 PPs. For software debugging purposes, these switches may be used prior to a deadstart dump to move the logical position of PPM-0 so its contents can be saved. Additional information on this capability is in the NOS Version 1 Operator's Guide and the NOS Version 1 System Programmer's Instant (refer to the preface for publication numbers). Up position sets bit. Down position clears bit.
SWEEP LOAD DUMP	Toggle switch (three-position)	Selects PP-0 mode of operation (refer to Deadstart in this section).
DEADSTART	Toggle switch (three-position)	Provides system deadstart. The deadstart stops the CP.
SLOW		Causes deadstart to repeat each 4096 microseconds, which includes a master clear duration of 1.0 microsecond.
OFF		Sets deadstart to off.
FAST		Causes deadstart to repeat each 256 microseconds, which includes a master clear duration of 1.0 microsecond.

I/O CHANNEL PARITY SWITCHES

The IOU has input/output (I/O) channel parity switches for the 12 channels 0 through 13 (octal) on the UE module at location I10 (figure 3-2). Channel parity switches for the 12 channels 20 through 33 (octal), if installed, are on a second UE module at location J10.

Switch X0 at the top of the module controls the parity selection for channel 0 or 20, depending on the respective module location. In top to bottom order, the following switches control successive channels 1 through 13 or 21 through 33.

Channel parity is enabled when the PARITY switch is set to ON and disabled when the switch is set to OFF.

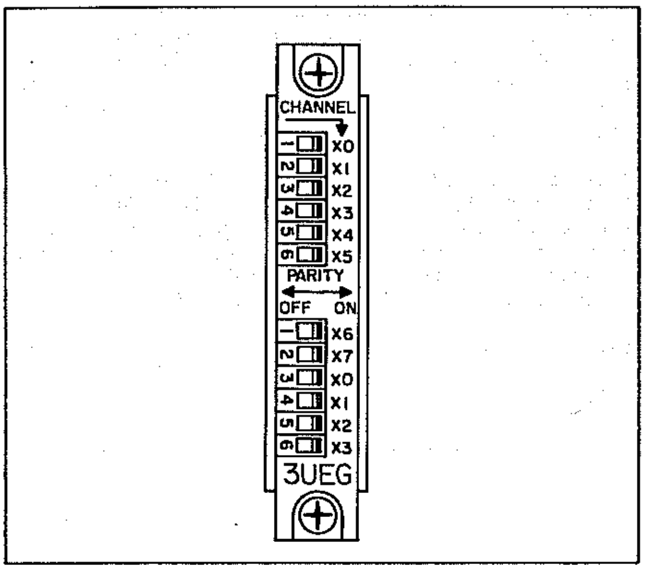


Figure 3-2. Module at I10 and J10

CLOCK SELECTION SWITCHES AND INDICATORS

The clock selection switches are on modules at locations 2A26 and 2A27 (figures 3-3 and 3-4).

The module at 2A26 has one two-position toggle switch, a momentary-contact switch (pushbutton or toggle), and three red, light-emitting diode indicators.

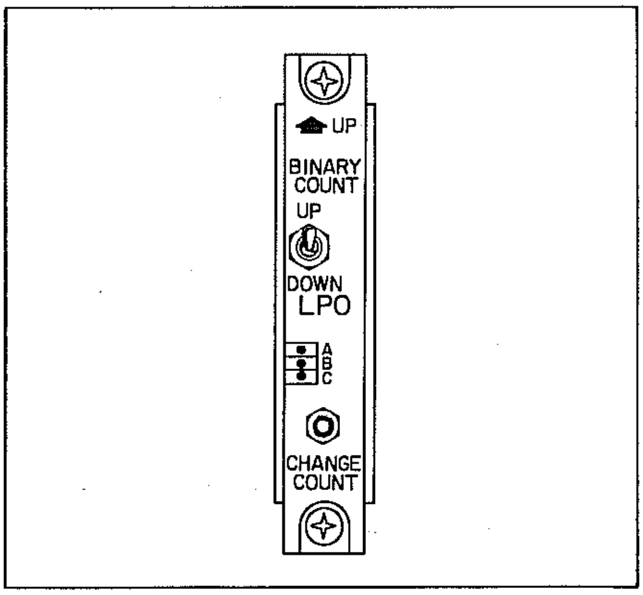


Figure 3-3. Module at 2A26

The module at 2A27 has two two-position toggle switches.

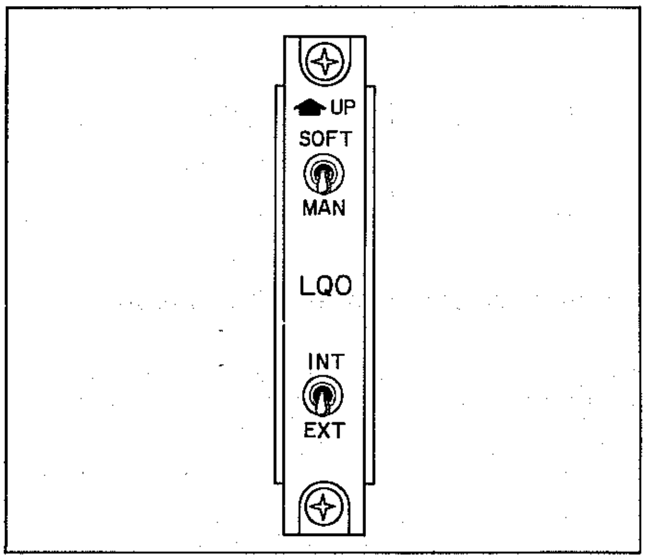


Figure 3-4. Module at 2A27

Table 3-2 lists the switch and indicator functions of both modules.

Table 3-2. Function of CP Modules at 2A26 and 2A27

Panel Nomenclature	Description	Function
INT EXT	Toggle switch	Selects internal or external clock source. Internal source is required for clock-margin checks. External source is required for use with EEM.
SOFT MAN	Toggle switch	Selects software or manual control of the clock frequency margins. Software control is described by the maintenance register bits 141 through 143 in section 2. Manual control is described by the following switches. After a removal of power from the CP or ECS/ESM controller, this switch must be set to SOFT to ensure the selection of the external clock during deadstart.
CHANGE COUNT	Pushbutton switch	Permits the clock to be manually incremented to a fast, slow, or normal operating frequency, when the SOFT/MAN switch is set to the MAN position. Normal operation requires the clock to be set at the normal operating frequency as indicated by light-emitting diodes A, B, and C.
A B C	Indicators	Indicate a binary count. C is bit 0, B is bit 1, and A is bit 2. The count for normal clock operating frequency is 3, where A does not light and B and C do light. Further use of these indicators is described in the hardware maintenance manuals listed in the preface.
BINARY COUNT UP/DOWN	Toggle switch	Selects an increment or decrement of the binary count, changed with the CHANGE COUNT switch.

P REGISTER AND STATUS BIT SELECTION SWITCHES

The peripheral processor program (P) register and status bit selection switches are located in chassis 2. The switches are on modules at locations 2D33 (IOU-0) and 2P34 (IOU-1) (figure 3-5).

The CP module has four two-position toggle switches. The switches permit the selection of the contents of any of the IOU P registers for display on indicators on BZ modules at IOU locations 2C28 (IOU-0) and 2P32 (IOU-1). The switches also define which PP is enabled for status bits 125 and 126 when maintenance register status bit 124 is not set.

The switches form a binary number code with the bottom switch as code bit 0 and the top switch as code bit 3. Table 3-3 lists the switches and their functions.

The switches are active when control bit 124 clears and disabled when bit 124 sets.

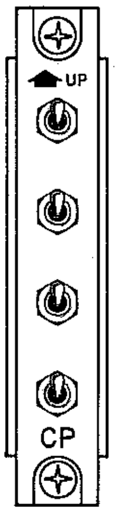


Figure 3-5. Module at 2D33 and 2P34

Table 3-3. Function of CP Modules at 2D33 and 2P34

Panel Location	Description	Function
Top	Toggle switch	UP position enables code bit 3. Down position disables code bit.
Next-to-top	Toggle switch	UP position enables code bit 2. Down position disables code bit.
Next-to-bottom	Toggle switch	UP position enables code bit 1. Down position disables code bit.
Bottom	Toggle switch	UP position enables code bit 0. Down position disables code bit.

KEYBOARD DISPLAY SELECTION SWITCHES

The keyboard display selection switch is on a module at location 2R36 (figure 3-6).

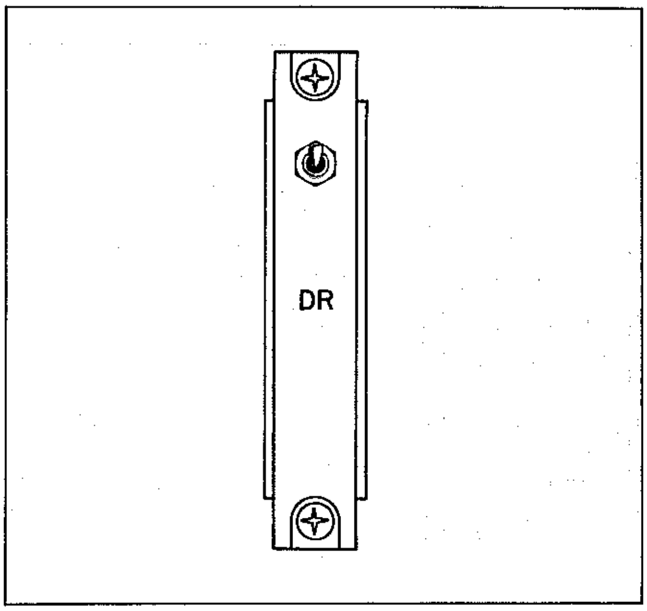


Figure 3-6. Module at 2R36

The DR module is a keyboard input receiver with one two-position toggle switch. The switch enables (down position) or disables (up position) the keyboard of the display station. The switch is always active.

IOU-0 MAINTENANCE REGISTER INDICATORS

The maintenance register indicators for IOU-0 (PP-0 through PP-9) are on modules at locations 2C41, 2D40, 2B37, 2C28, 2C31, and 2E40 (figures 3-7 through 3-12).

Each module has two columns of nine, red, light-emitting diodes. The diodes indicate the condition of certain maintenance register bits. Each diode represents a bit in the register and lights when the bit sets. A pushbutton switch on the module permits testing all the diodes on themodule without changing any of the data bits. The light displays are useful in debugging software.

Figures 3-7 through 3-12 show the modules, diode locations (decimal and octal), usage (status or control), and descriptions for IOU-0. The light-emitting diodes which are not used but have bit number designations are wired to the maintenance register. The diodes without bit designators are not wired.

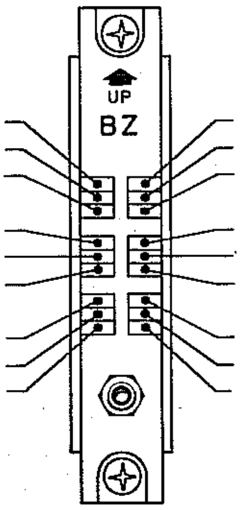
IOU-1 MAINTENANCE REGISTER INDICATORS

The maintenance register indicators for IOU-1 (PP-10 through PP-19) are located in chassis 2, when installed. The indicators are on modules at locations 2N35, 2O38, and 2P32 (figure 3-13 through 3-15).

Each module has two columns of nine, red, light-emitting diodes. The diodes indicate the condition of certain maintenance register bits. Each diode represents a bit in the register and lights when the bit sets. A pushbutton switch on the module permits testing all the diodes on the module without changing any of the data bits. The light displays are useful in debugging software.

Figures 3-13 through 3-15 show the modules, diode locations (decimal and octal), usage (status or control), and descriptions for IOU-1. The light-emitting diodes which are not used but have bit number designations are wired to the maintenance register. The diodes without bit designators are not wired.

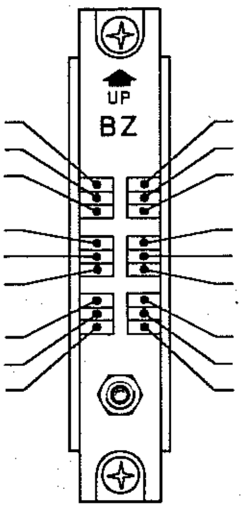
BIT NO. DEC.	BIT NO. OCT.	US-AGE	DESCRIPTION
10	12	S	ANY ERROR BIT EQUALS ONE
11	13	S	ECS TRANSFER ERROR
12	14	S	CP–O PE
13	15	S	CP–I PE (USED ONLY IN DUAL CP MODELS)
14	16	S	PPMO PE
15	17	S	PPM1 PE
16	20	S	PPM2 PE
17	21	S	PPM3 PE
18	22	S	PPM4 PE



BIT NO. DEC.	BIT NO. OCT.	US-AGE	DESCRIPTION
0	0	S	CM PARITY ERROR
1	1	S	CM ADRS PE
2	2		SECDED ERROR CP-1
3	3	S	SECDED ERROR
183	267	S	DOUBLE ERROR
5	5	S	CMC PE
7	7		NOT USED
8	10		NOT USED
9	11		EEM-TRANSFER ERROR

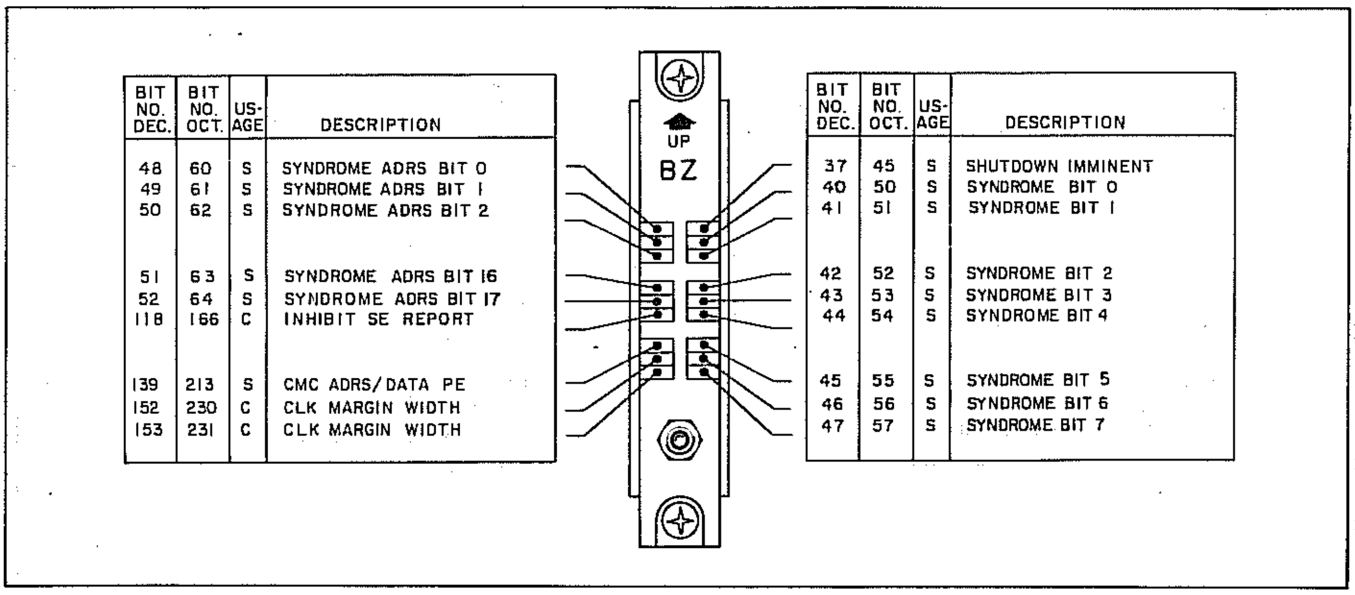
Figure 3-7. Module at 2C41

BIT NO. DEC.	BIT NO. OCT.	US-AGE	DESCRIPTION
28	34	S	CHANNEL 4 PE
29	35	S	CHANNEL 5 PE
30	36	S	CHANNEL 6 PE
31	37	S	CHANNEL 7 PE
32	40	S	CHANNEL 10 PE
33	41	S	CHANNEL 11 PE
34	42	S	CHANNEL 12 PE
35	43	S	CHANNEL 13 PE
36	44	S	MAINS PWR FAILURE



BIT NO. DEC.	BIT NO. OCT.	US-AGE	DESCRIPTION
19	23	S	PPM5 PE
20	24	S	PPM6 PE
21	25	S	PPM7 PE
22	26	S	PPM8 PE
23	27	S	PPM9 PE
24	30	S	CHANNEL 0 PE
25	31	S	CHANNEL 1 PE
26	32	S	CHANNEL 2 PE
27	33	S	CHANNEL 3 PE

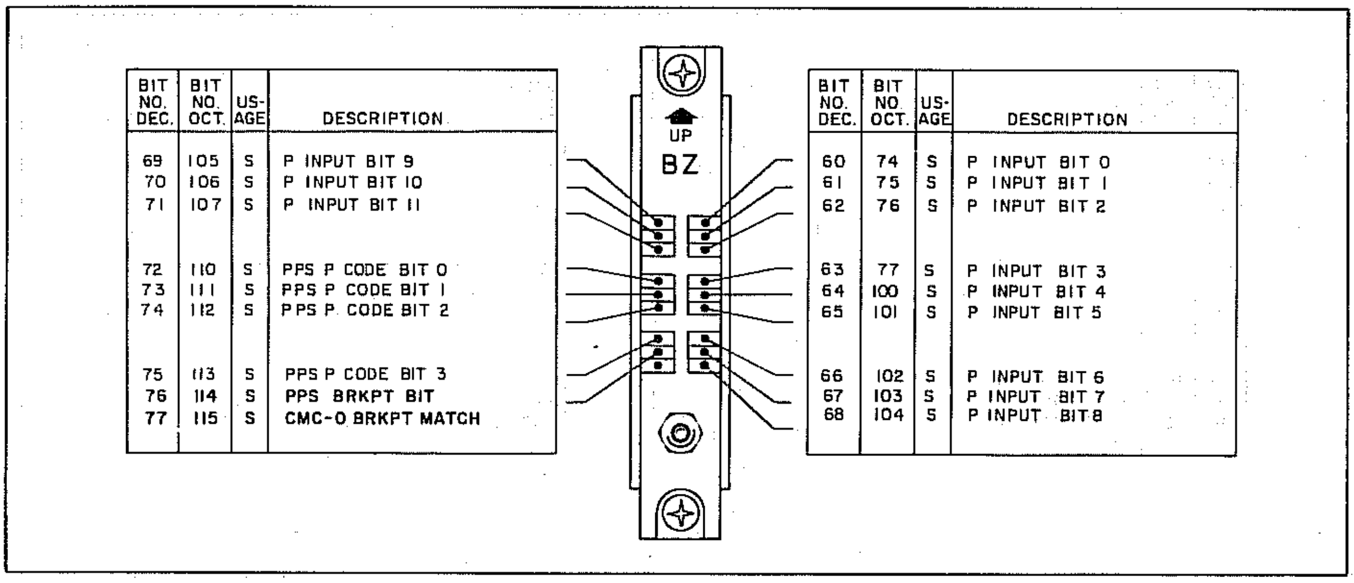
Figure 3-8. Module at 2D40



BIT NO. DEC.	BIT NO. OCT.	US-AGE	DESCRIPTION
48	60	S	SYNDROME ADRS BIT 0
49	61	S	SYNDROME ADRS BIT 1
50	62	S	SYNDROME ADRS BIT 2
51	63	S	SYNDROME ADRS BIT 16
52	64	S	SYNDROME ADRS BIT 17
118	166	C	INHIBIT SE REPORT
139	213	S	CMC ADRS/DATA PE
152	230	C	CLK MARGIN WIDTH
153	231	C	CLK MARGIN WIDTH

BIT NO. DEC.	BIT NO. OCT.	US-AGE	DESCRIPTION
37	45	S	SHUTDOWN IMMINENT
40	50	S	SYNDROME BIT 0
41	51	S	SYNDROME BIT 1
42	52	S	SYNDROME BIT 2
43	53	S	SYNDROME BIT 3
44	54	S	SYNDROME BIT 4
45	55	S	SYNDROME BIT 5
46	56	S	SYNDROME BIT 6
47	57	S	SYNDROME BIT 7

Figure 3-9. Module at 2B37

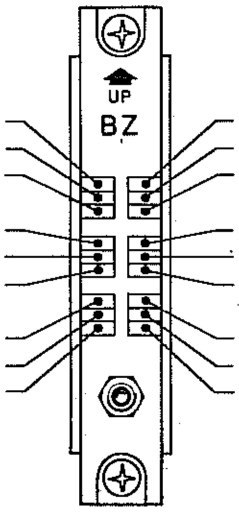


BIT NO. DEC.	BIT NO. OCT.	US-AGE	DESCRIPTION
69	105	S	P INPUT BIT 9
70	106	S	P INPUT BIT 10
71	107	S	P INPUT BIT 11
72	110	S	PPS P CODE BIT 0
73	111	S	PPS P CODE BIT 1
74	112	S	PPS P CODE BIT 2
75	113	S	PPS P CODE BIT 3
76	114	S	PPS BRKPT BIT
77	115	S	CMC-O BRKPT MATCH

BIT NO. DEC.	BIT NO. OCT.	US-AGE	DESCRIPTION
60	74	S	P INPUT BIT 0
61	75	S	P INPUT BIT 1
62	76	S	P INPUT BIT 2
63	77	S	P INPUT BIT 3
64	100	S	P INPUT BIT 4
65	101	S	P INPUT BIT 5
66	102	S	P INPUT BIT 6
67	103	S	P INPUT BIT 7
68	104	S	P INPUT BIT 8

Figure 3-10. Module at 2C28

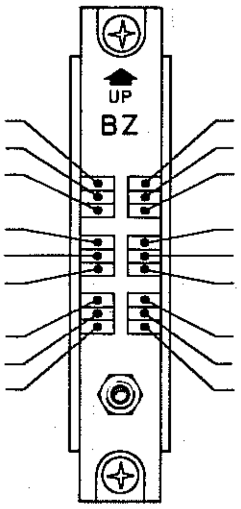
BIT NO. DEC.	BIT NO. OCT.	US-AGE	DESCRIPTION
95	137	C	STOP ON PPM PE
190	276	C	SYNDROME ADRS BIT 3 (M 171, 172, 173) BIT 4 (M 175)
191	277		NOT USED
192	300	S	CP-0 STOPPED
193	301	S	CP-1 STOPPED
194	302	S	ECS IN PROGRESS FLAG
195	303	S	MONITOR FLAG CP-0
196	304	S	MONITOR FLAG CP-1
202	312	S	EXP ADRS MODE



BIT NO. DEC.	BIT NO. OCT.	US-AGE	DESCRIPTION
54	66	S	PE PORT CODE BIT 0
55	67	S	PE PORT CODE BIT 1
56	70	S	BRKPT PORT CODE BIT 0
57	71	S	BRKPT PORT CODE BIT 1
58	72	S	BRKPT FCTN CODE BIT 0
59	73	S	BRKPT FCTN CODE BIT 1
79	117		CMC-1 BRKPT MATCH
85	125	C	INHIBIT CMC REQUEST TO CMC
94	136	C	STOP ON ERROR

Figure 3-11. Module at 2C31

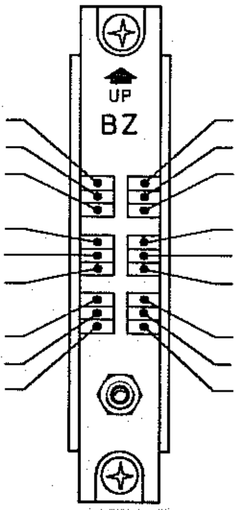
BIT NO. DEC.	BIT NO. OCT.	US-AGE	DESCRIPTION
148	224	S	RVM ADRS BIT 4 STATUS
149	225	S	RVM ADRS BIT 5 STATUS
150	226	S	RVM ADRS HI/LO
151	227	S	RVM ALL/ONE
88	130	S	DIAGNOSTIC AID
89	131	S	DIAGNOSTIC AID
90	132	S	DIAGNOSTIC AID
91	133	S	DIAGNOSTIC AID
92	134	S	DIAGNOSTIC AID



BIT NO. DEC.	BIT NO. OCT.	US-AGE	DESCRIPTION
120	170	C	PP SEL CODE BIT 0
121	171	C	PP SEL CODE BIT 1
122	172	C	PP SEL CODE BIT 2
123	173	C	PP SEL CODE BIT 3
124	174	C	PP SEL AUTO/MNL MODE
144	220	S	RVM ADRS BIT 0 STATUS
145	221	S	RVM ADRS BIT 1 STATUS
146	222	S	RVM ADRS BIT 2 STATUS
147	223	S	RVM ADRS BIT 3 STATUS

Figure 3-12. Module at 2E40

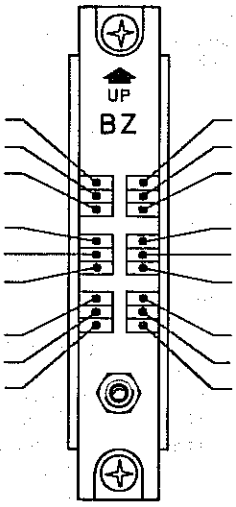
BIT NO. DEC.	BIT NO. OCT.	US-AGE	DESCRIPTION
18	22	S	PPM 24 PE
19	23	S	PPM 25 PE
20	24	S	PPM 26 PE
21	25	S	PPM 27 PE
22	26	S	PPM 28 PE
23	27	S	PPM 29 PE
24	30	S	CHANNEL 20 PE
25	31	S	CHANNEL 21 PE
26	32	S	CHANNEL 22 PE



BIT NO. DEC.	BIT NO. OCT.	US-AGE	DESCRIPTION
0	0	S	READ DISASSEMBLER PE
6	6		NOT USED
7	7		NOT USED
12	14	S	CP-0 RGTR PE
13	15	S	CP-1 RGTR PE
14	16	S	PPM 20 PE
15	17	S	PPM 21 PE
16	20	S	PPM 22 PE
17	21	S	PPM 23 PE

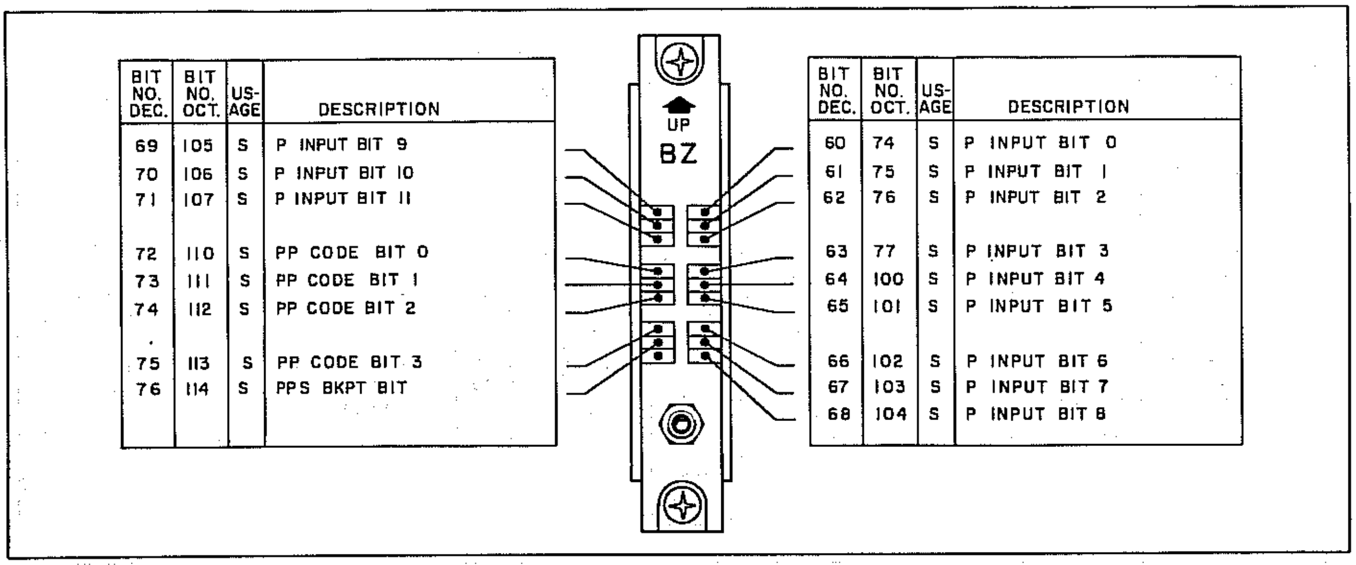
Figure 3-13. Module at 2N35

BIT NO. DEC.	BIT NO. OCT.	US-AGE	DESCRIPTION
85	125	C	INHIBIT CMC REQUEST
94	136	C	STOP ON DOUBLE SECDED ER
95	137	C	STOP ON PP MEMORY ER
			NOT USED
120	170	C	PP SEL CODE BIT 0
121	171	C	PP SEL CODE BIT 1
122	172	C	PP SEL CODE BIT 2
123	173	C	PP SEL CODE BIT 3
124	174	C	PP SEL AUTO/MNL MODE



BIT NO. DEC.	BIT NO. OCT.	US-AGE	DESCRIPTION
27	33	S	CHANNEL 23 PE
28	34	S	CHANNEL 24 PE
29	35	S	CHANNEL 25 PE
30	36	S	CHANNEL 26 PE
31	37	S	CHANNEL 27 PE
32	40	S	CHANNEL 30 PE
33	41	S	CHANNEL 31 PE
34	42	S	CHANNEL 32 PE
35	43	S	CHANNEL 33 PE

Figure 3-14. Module at 2038



BIT NO. DEC.	BIT NO. OCT.	US-AGE	DESCRIPTION
69	105	S	P INPUT BIT 9
70	106	S	P INPUT BIT 10
71	107	S	P INPUT BIT 11
72	110	S	PP CODE BIT 0
73	111	S	PP CODE BIT 1
74	112	S	PP CODE BIT 2
75	113	S	PP CODE BIT 3
76	114	S	PPS BKPT BIT

BIT NO. DEC.	BIT NO. OCT.	US-AGE	DESCRIPTION
60	74	S	P INPUT BIT 0
61	75	S	P INPUT BIT 1
62	76	S	P INPUT BIT 2
63	77	S	P INPUT BIT 3
64	100	S	P INPUT BIT 4
65	101	S	P INPUT BIT 5
66	102	S	P INPUT BIT 6
67	103	S	P INPUT BIT 7
68	104	S	P INPUT BIT 8

Figure 3-15. Module at 2P32

CM CONFIGURATION AND CLOCK SWITCHES AND INDICATORS

The CP contains clock switches and indicators at module locations 5A1 through 5A3 (figure 3-16). Table 3-4 lists the switches, indicators, and their functions. Table 3-5 lists the switch settings for a normal or reconfigured CM.

All CM quadrants (model 865) or ports (model 875), for a particular CM size, are available for use by the system when the CM configuration switches are set to the normal operation positions shown in table 3-5.

If one of the CM quadrants (model 865) or ports (model 875) becomes defective, the CM may be reconfigured to operate without the quadrant (port). The reconfiguration is performed by determining the defective quadrant (port) and then setting the CM configuraton switches for that quadrant (port) to the reconfigured operation switch positions shown in table 3-5. Any one quadrant (port) may be reconfigured at one time. The switches accomplish a logical reconfiguration by manipulating the storage address stack (SAS) bits 2^4, 2^5, 2^{18}, and 2^{19}.

Any errors detected while CM is operating in a reconfigured (degraded) mode appear to the maintenance register in translated form, giving the physical address (bank, quadrant, and so on) of the error.

The switches are always active.

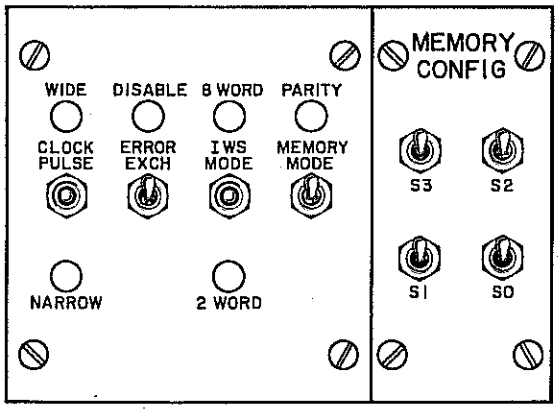


Figure 3-16. Modules 5A1 through 5A3

POWER-ON AND POWER-OFF PROCEDURES

In case of an emergency, use the system EMERGENCY OFF switch. The power-on and power-off procedures are described in the power distribution and warning system manual listed in the preface.

CAUTION

Improper application or removal of power may damage system circuits and/or air conditioning system. Power must be turned on/off by designated personnel only, except for the system EMERGENCY OFF switch. Use only for extreme emergency, not for normal shutdown.

OPERATING PROCEDURES

Refer to the operator/analyst handbook listed in the preface. The system is initialized by setting its control switches, and then by running a deadstart sequence. After initialization, the keyboard is used to instruct the system further, under program control.

Table 3-4. Functions of Controls on Modules at 5A1 Through 5A3 - Models 865 and 875

Panel Nomenclature	Description	Function
MEMORY CONFIG S0, S1, S2, S3	Toggle switches	Control CM quadrant (model 865) or port (model 875) configuration. Up positions set bits. Down positions clear bits.
CLOCK PULSE	Toggle switch	Controls clock pulse width. Up position provides wide pulse. Middle position enables software control of pulse width. Down position provides narrow pulse.
WIDE, NARROW	Indicators	Light to show respective clock pulse widths.
ERROR EXCH	Toggle switch	Up position disables CEJ on error exit. Down position enables CEJ on error exit.
DISABLE	Indicator	Lights to show CEJ disabled on error exit condition.
IWS MODE	Toggle switch	Selects size of instruction word stack (IWS). Up position selects 8 words. Middle position selects 12 words. Down position selects 2 words.
8 WORD, 2 WORD	Indicators	Light to show respective IWS words selected. Neither indicator lighted denotes a 12-word IWS selection.
MEMORY MODE	Toggle switch	Selects a parity or single-error correction double-error detection (SECDED) mode. Changing the position of this switch requires CM to be rewritten. Up position selects parity mode. Down position selects SECDED mode.
PARITY	Indicator	Lights to show parity mode selection.

Table 3-5. Central Memory Reconfiguration - Models 865 and 875

Central Memory Size	Range of Address	Normal Operation Switch Positions †				Faulty Quadrant/ Port		Degrade Operation Switch Positions†				Resulting Memory Size
		Memory Configuration Switches						Memory Configuration Switches				
		S0	S1	S2	S3	Quad	Port	S0	S1	S2	S3	
262K	0-777777	1	0	0	0	1	1	No Degrade				-
524K	0-1777777	1	1	0	0	1	1	0	1	0	0	
						1	0	1	0	0	0	262K
786K	0-2777777	1	1	1	0	1	1	0	1	1	0	
						1	0	1	0	1	0	512K
						0	1	1	1	0	0	
1048K	0-3777777	1	1	1	1	1	1	0	1	1	1	
						1	0	1	0	1	1	
						0	1	1	1	0	1	786K
						0	0	1	1	1	0	

†Switches generate a 1 when in the up position and a 0 when in the down position.

INSTRUCTION DESCRIPTIONS

CP INSTRUCTIONS

CP INSTRUCTION FORMATS

NOTE

CP instructions use a 60-bit word. For these instructions, the most significant bit is bit 59 and the least significant bit is bit 0.

Program instruction words are divided into 15-bit fields called parcels. The first parcel (parcel 0) is the highest-order 15 bits of the 60-bit word. The second, third, and fourth parcels (parcels 1, 2, and 3) follow in order. Figure 4-1 shows possible parcel arrangements for instructions within a program instruction word.

An instruction may occupy one, two, or four parcels. This arrangement depends upon the instruction format. When an instruction occupies two parcels, it must occupy two parcels within the same program word. A program word may be filled with a one-parcel pass instruction or an instruction acting as a two-parcel pass instruction. These instructions are used to fill a program word when necessary to place a particular instruction in the first parcel of a program word or to avoid starting a two-parcel instruction in the fourth parcel of a program word. Pass instructions may also be used for branch entry points because a branch instruction destination address must begin with a new word. One-parcel pass instructions are 460xx through 463xx. Instructions 60xxx through 62xxx may be used as two-parcel pass instructions by setting the i instruction designator to zero. Refer to table 4-1 for CP instruction designators.

CP instructions 011 and 012 have special properties. They are 60-bit double instructions which must start at parcel 0. The programmer has the option of providing a branch instruction at parcels 2 and 3 in the same instruction word, (to an error handling software routine), or filling this space with pass instructions. Refer to instructions 011 and 012.

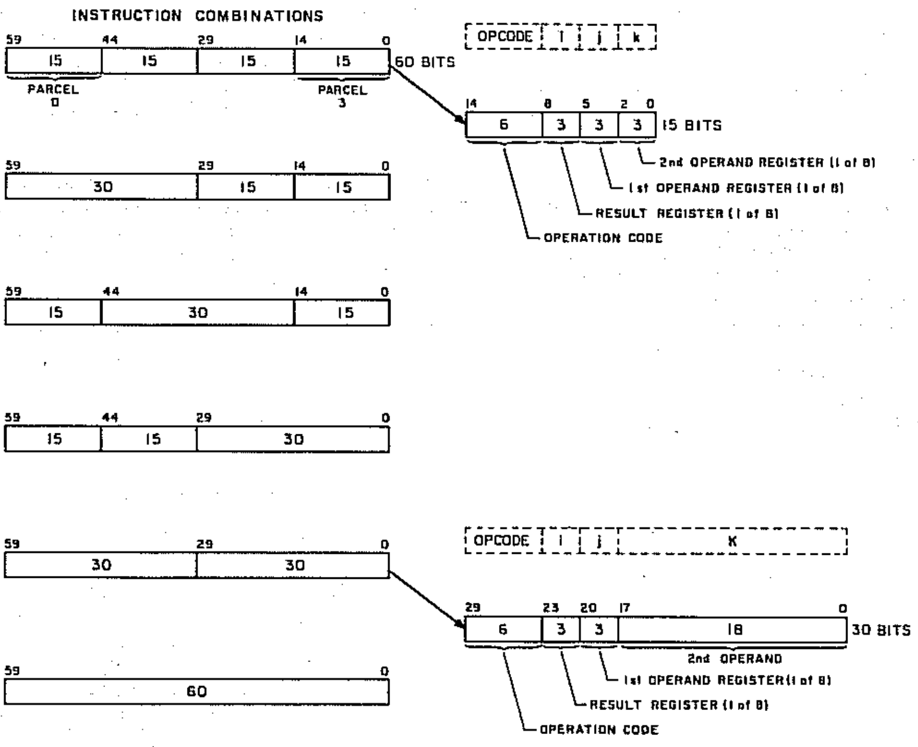


Figure 4-1. CP Instruction Parcel Arrangement

Instruction 013 is a 60-bit instruction that must start at parcel 0. It ignores any information in parcels 2 and 3; however, these parcels are normally set to all zeros.

Table 4-1. Central Processor Instruction Designators

Designator	Use
Opcode	6-bit/9-bit field specifying instruction operation code.
i	3-bit code specifying one of eight registers.
j	3-bit code specifying one of eight registers.
jk	6-bit code specifying amount of shift or mask.
k	3-bit code specifying one of eight registers.
K	18-bit operand or address.
x	Unused designator.
A	One of eight 18-bit address registers.
B	One of eight 18-bit index registers; B0 is fixed and equal to zero.
X	One of eight 60-bit operand registers.
()	Content of the word at a CM address.

CP OPERATING MODES

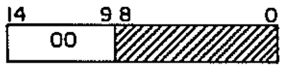
The CP executes instructions in job mode or monitor mode, and changes these operating modes through exchange jumps. Such exchanges are caused by CP instruction 013, PP instructions 2600, 2610, 2620, or by hardware-detected error conditions. Hardware-caused exchanges are also called error exits; most of these can be enabled/disabled by setting/clearing bits in the exchange package. Refer to Exchange Jump and Error Response, section 5.

A hardware flag called the monitor flag (MF) when set/clear indicates that the CP is in the monitor/job mode.

CP INSTRUCTION DESCRIPTIONS

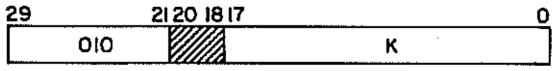
The instruction descriptions are in numerical order. The shaded areas, like those in the following 00xxx and 010xK instruction formats, indicate unused bits. The unused bits are ignored by the CP.

00xxx Error Exit to MA or Program Stop PS



The CEJ/MEJ switch determines which functions this instruction can perform. When the switch is in the DISABLE position, the system has no central exchange or monitor exchange jump capability so this instruction stops the CP. When this stop occurs, the content of the P register may not correspond to the address of the 00 instruction. The P register may have been incremented prior to the execution of the 00 instruction. When the switch is in the ENABLE position, this instruction causes an error exit response that is the same as an illegal instruction. Refer to Error Response under CP Programming in section 5.

010xK Return Jump to K RJ



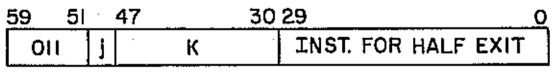
This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction writes a special word into CM at relative address K. The current program sequence then terminates by a jump to address K plus 1. The word stored in memory contains a jump instruction which causes an unconditional jump to the address of this return jump instruction plus 1.

This instruction calls a subroutine and inserts execution of the subroutine between execution of this instruction word and the following instruction word. Instructions appearing after the return jump instruction in the instruction word are not executed. The called subroutine exit must be at address K. The called subroutine entrance address must be K plus 1.

This instruction stores a 60-bit word at address K in memory. The upper half of this word contains an unconditional jump (0400) instruction with an address which is equal to the current program address plus 1. The lower half of the stored word is all zeros. The octal digits in the stored word then appear as illustrated with the x field indicating the location of the current program address plus 1.

K	0400x	xxxxx	00000	00000	Subroutine exit
K + 1	yyyyy	yyyyy	yyyyy	yyyyy	Subroutine entrance

011jK Block Copy (Bj) + K Words from (X0 + RAE) to (A0 + RAC) REC Bj + K



This instruction copies a block of Bj plus K consecutive words from unified extended memory (UEM) or external extended memory (EEM) to CM, beginning at address A0 plus RAC in CM and address X0 plus RAE in UEM or EEM. It leaves Bj, X0, and A0 unchanged. Bj and K are both signed 18-bit one's complement numbers, making it possible to transfer a maximum of 131 071 60-bit words. If Bj plus K is zero, the instruction acts as a 60-bit pass instruction.

With the exception of IOU requests and requests by the opposite CP on dual CP systems, all other CP activity stops during this block transfer. All instructions which have issued prior to this instruction execute to completion. No further instructions are issued until the block transfer completes. As a result of these restrictions, the maximum rate of data flow from UEM to CM is 100 nanoseconds (model 865) or 50 nanoseconds (model 875). The maximum rate of data flow from EEM to CM is 100 nanoseconds for both models. When an IOU request for CM occurs during the block transfer, the IOU word address is inserted in the stream of addresses to the storage address stack (SAS).

The selection of UEM or EEM depends on the state of the UEM enable flag bit. If the flag is set, the transfer is between UEM and CM; if the flag is clear, the transfer is between EEM and CM.

A fake read/write and exit can occur only with EEM references. A fake read/write and exit occurs in standard addressing mode if X0 plus RAE bit 21 or 22 is set. A fake read/write and exit occurs in expanded addressing mode if X0 plus RAE bit 28 is set. In both cases, a half exit to parcel 2 of the same instruction word occurs (refer to Fake Read/Write and Exit in section 5). If this is not the case, the next instruction is taken from parcel 0 of the next instruction word.

In standard addressing mode, a set condition of bit 23 of the X0 address and FLE register causes a flag register operation (refer to Flag Register Operation in section 5). In expanded addressing mode, a set condition of bit 29 of the X0 address and FLE register causes a flag register operation. Flag register operations exist only for EEM references, and result in an address range error for UEM references.

Enhanced block copy addressing permits accessing of a block of CM memory words more than 262K from RAC. This is achieved by using the upper 30 bits of X0 as a CM starting address instead of the customary 18 bits of A0. X0 bits 30 through 50 provide a 21-bit format for the CM starting address, as compared to the 18-bit format of A0. The enhanced block copy mode flag is bit 54 of word 3 of an exchange package. A clear flag selects A0 as the starting CM address; a set flag selects the upper 30-bit parcel of X0 as the starting CM address.

0012jK	Block Copy (Bj) + K Words from (A0 + RAC) to (X0 + RAE)	WEC Bj + K

59	51	47	30 29	0
012	j	K	INST. FOR HALF EXIT	

This instruction is identical to block copy instruction 011jK, except that the transfer is a block of Bj plus K consecutive words from CM to UEM or EEM, rather than vice versa. Refer to instruction 011jK, preceding.

013jK	Central Exchange Jump to Bj + K when MF Set	XJ Bj + K
013xx	Monitor Exchange Jump to MA when MF Clear	XJ

59	51	47	30 29	0
013	j	K	▨	

This instruction must start at parcel 0. Also, an exchange package must be ready at address Bj plus K or at address MA.

This instruction stores P plus 1 into the outgoing exchange package in hardware and then exchanges this exchange package with the exchange package stored in memory. If, at the beginning of the instruction, the MF is set, then the incoming exchange package starts at absolute address Bj plus K. If, at the beginning, the MF is clear, then the j and K fields of the instruction are ignored, and the incoming exchange package starts at absolute address MA which is obtained from the outgoing exchange package. In either case, the MF is toggled and the outgoing exchange package is stored beginning at the same CM address from where the incoming exchange package is obtained. Also, the jump is always to relative address P, parcel 0, from the new exchange package. Refer to Exchange Jump, section 5.

014jk	Read Extended Memory Xk to Xj	RX Xj,Xk

14	6 5	3 2 0
014	j	k

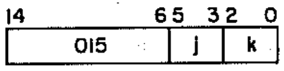
This is a 15-bit instruction that reads one 60-bit word from UEM or EEM address (Xk) to X register Xj. If the UEM mode flag is clear, EEM is read. If the UEM mode flag is set, UEM is read per RAE/FLE.

If EEM is referenced, the instruction is executed by the storage move unit and cannot segment or overlap execution with other instructions until the data enters the Xj register. If UEM is referenced, the instruction is executed by increment and is fully segmented.

Instruction 014 may execute in standard or expanded addressing mode to access UEM or EEM. The addressing mode is selected by a set or clear condition of the expanded addressing select flag (refer to Extended Memory Addressing Modes in section 5). In standard addressing mode, instruction 014 uses the memory address Xk (21 bits) plus RAE (21 bits) to address UEM or EEM, depending on the condition of the UEM mode flag. In expanded addressing mode, instruction 014 uses the memory address Xk (24 bits) plus RAE (24 bits) to address UEM or EEM, depending on the condition of the UEM mode flag.

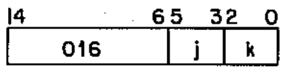
A fake read/write and exit may occur with EEM references in a direct read/write instruction. A fake read/write and exit occurs in standard addressing mode if X0 plus RAE bit 21 or 22 is set. A fake read/write and exit occurs in expanded addressing mode if X0 plus RAE bit 28 is set. In both cases, Xj in CM is entered with zero and the next instruction is taken from parcel 0 of the following instruction word.

015jk Write Extended Memory Xk from Xj WX Xj,Xk



This 15-bit instruction is identical to direct read/write instruction 014jK, except that the one-word transfer is from CM address Xj to UEM or EEM address Xk, rather than vice versa. Refer to instruction 014jK, preceding.

016jk Read Microsecond Counter to Xj RC

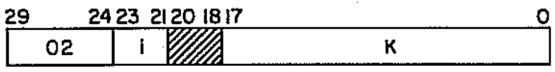


This instruction reads the contents of the 32-bit microsecond counter to register Xj. The upper 28 bits of Xj are entered with zeros.

017jk Illegal Instruction

Refer to Illegal Instructions, section 5.

02ixK Jump to (Bi) + K JP

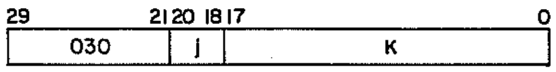


This two-parcel instruction uses the lower-order 18 bits as operand K. The instruction causes the current program sequence to terminate with a jump to address Bi plus K in CM.

This instruction allows computed branch point destinations. This is the only instruction in which a computed parameter can specify a program branch destination address. All other jump instructions have preassigned destination addresses.

The quantities in Bi and operand K are added in an 18-bit one's complement mode. The result is treated as an 18-bit positive integer which specifies the beginning address in CM for the new program sequence. The remaining instructions, if any, in the instruction word do not execute.

030jK Branch to K if (Xj) = 0 ZR

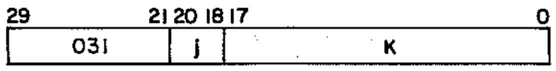


This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending upon the content of Xj. The branch to address K occurs only on the following conditions. The current program sequence continues for all other cases.

```
Jump to K if: (Xj) = 0000 0000 0000 0000 0000
                     (positive zero)
              (Xj) = 7777 7777 7777 7777 7777
                     (negative zero)
```

This instruction branches on a zero result from either a fixed-point or a floating-point operation.

031jK Branch to K if (Xj) $\neq$ 0 NZ

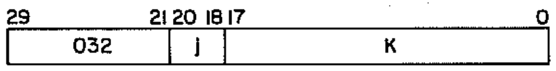


This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending upon the content of Xj. The program sequence continues only on the following conditions. The branch to address K occurs for all other cases.

```
Continue if: (Xj) = 0000 0000 0000 0000 0000
                    (positive zero)
             (Xj) = 7777 7777 7777 7777 7777
                    (negative zero)
```

This instruction branches on a nonzero result from either a fixed-point or a floating-point operation.

032jK Branch to K if (Xj) is Positive PL



This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending upon the content of Xj. The branch decision for this instruction is based on the value of the sign bit in Xj.

Jump to K if: Bit 59 of Xj = 0 (positive)

Continue if: Bit 59 of Xj = 1 (negative)

This instruction branches on a positive result from either a fixed-point or a floating-point operation.

033jK Branch to K if (Xj) is Negative NG

29 21	20 18	17 0
033	j	K

This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending upon the content of Xj. The branch decision for this instruction is based on the value of the sign bit in Xj.

```
Jump to K if:  Bit 59 of Xj = 1 (negative)

Continue if:   Bit 59 of Xj = 0 (positive)
```

This instruction branches on a negative result from either a fixed-point or a floating-point operation.

034jK Branch to K if (Xj) is in Range IR

29 21	20 18	17 0
034	j	K

This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending upon the content of Xj. The program sequence continues only on the following conditions. The branch to address K occurs for all other cases.

```
Continue if:    (Xj) =   3777 xxxx xxxx xxxx xxxx
                         (positive overflow)
                (Xj) =   4000 xxxx xxxx xxxx xxxx
                         (negative overflow)
```

This instruction branches on a floating-point quantity within the floating-point range. The value of the coefficient is ignored in making this branch test. An underflow quantity is considered in range for purposes of this test.

035jK Branch to K if (Xj) is Out of Range OR

29 21	20 18	17 0
035	j	K

This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending upon the content of Xj. The branch to address K occurs only on the following conditions. The current program sequence continues for all other cases.

```
Jump to K if: (Xj) = 3777 xxxx xxxx xxxx xxxx
                     (positive overflow)
              (Xj) = 4000 xxxx xxxx xxxx xxxx
                     (negative overflow)
```

036jK Branch to K if (Xj) is Definite DF

29 21	20 18	17 0
036	j	K

This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending upon the content of Xj. The program sequence continues only on the following conditions. The branch to address K occurs for all other cases.

```
Continue if:  (Xj) = 1777 xxxx xxxx xxxx xxxx
                     (positive indefinite)
              (Xj) = 6000 xxxx xxxx xxxx xxxx
                     (negative indefinite)
```

This instruction branches on a floating-point quantity which may be out of range but is still defined. The value of the coefficient is ignored in making this branch test. An overflow quantity or an underflow quantity is considered defined for purposes of this test.

037jK Branch to K if (Xj) is Indefinite ID

29 21	20 18	17 0
037	j	K

This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending upon the content of the Xj register. The branch to address K occurs only on the following conditions. The current program sequence continues for all other cases.

```
Jump to K if: (Xj) = 1777 xxxx xxxx xxxx xxxx
                     (positive indefinite)
              (Xj) = 6000 xxxx xxxx xxxx xxxx
                     (negative indefinite)
```

This instruction branches on a floating-point quantity which is not defined. The value of the coefficient is ignored in making this branch test. An overflow quantity or an underflow quantity is considered defined for purposes of this test.

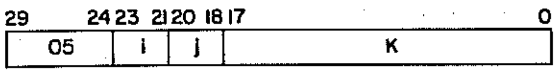
04ijK Branch to K if (Bi) = (Bj) EQ

29 24	23 21	20 18	17 0
04	i	j	K

This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending upon a comparison of the contents of the Bi and Bj registers. The branch to address K occurs only if the two quantities are identical on a bit-by-bit comparison basis. The current program sequence continues for all other cases.

This instruction branches on an index equality test. A quantity consisting of all zeros and a quantity consisting of all ones are not equal for this test.

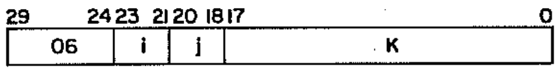
05ijK Branch to K if (Bi) ≠ (Bj) NE



This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending upon a comparison of the contents of the Bi and Bj registers. The program sequence continues only if the two quantities are identical on a bit-by-bit comparison basis. The branch to address K occurs for all other cases.

This instruction branches on an index inequality test. A quantity consisting of all zeros and a quantity consisting of all ones are not equal for this test.

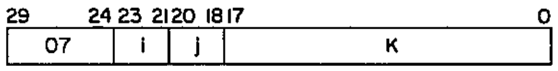
06ijK Branch to K if (Bi) is ≥ (Bj) GE



This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending upon a comparison of the contents of Bi and Bj. Both quantities are treated as signed integers. The branch to address K occurs if the content of Bi is greater than or equal to the content of Bj. The current program sequence continues if the content of Bi is less than Bj.

This instruction branches on an index threshold test. A positive zero quantity is considered greater than a negative zero quantity.

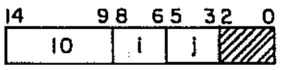
07ijK Branch to K if (Bi) < (Bj) LT



This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending upon a comparison of the contents of Bi and Bj. Both quantities are treated as signed integers. The branch to address K occurs if the content of Bi is less than the content of Bj. The current program sequence continues if the content of Bi is greater than or equal to the content of Bj.

This instruction branches on an index threshold test. A positive zero quantity is considered greater than a negative zero quantity.

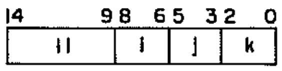
10ijx Transmit (Xj) to Xi BXi Xj



This instruction transfers a 60-bit word from Xj into Xi.

This instruction moves data from one X register to another X register. No logical function is performed on the data.

11ijk Logical Product of (Xj) and (Xk) to Xi BXi Xj * Xk

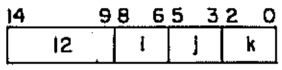


This instruction reads operands from two X registers, operates upon them to form a result, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. The result delivered to Xi is the bit-by-bit logical product of the two operands. Each of the 60 bits in Xj is compared with the corresponding bit in Xk to form a single bit in Xi. A sample computation is listed in octal notation to illustrate the operation performed and includes the four possible bit combinations that may occur.

```
(Xj) = 7777 7000 0123 4567 1010

(Xk) = 0123 4567 0077 7700 1100

(Xi) = 0123 4000 0023 4500 1000
```

This instruction extracts portions of a 60-bit word during data processing.

12ijk Logical Sum of (Xj) and (Xk) to Xi BXi Xj + Xk

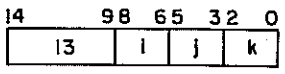


This instruction reads operands from two X registers, operates upon them to form a result, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. The result delivered to Xi is the bit-by-bit logical sum of the two operands. Each of the 60 bits in Xj is compared with the corresponding bit in Xk to form a single bit in Xi. A sample computation is listed in octal notation to illustrate the operation performed and includes the four possible bit combinations that may occur.

```
(Xj) = 0000 7777 0123 4567 1010

(Xk) = 0123 4567 7777 0000 1100

(Xi) = 0123 7777 7777 4567 1110
```

This instruction merges portions of a 60-bit word into a composite word during data processing.

13ijk Logical Difference of (Xj) and (Xk) to Xi — BXi Xj - Xk

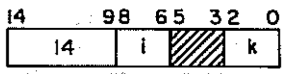


This instruction reads operands from two X registers, operates upon them to form a result, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. The result delivered to Xi is the bit-by-bit logical difference of the two operands. Each of the 60 bits in Xj is compared with the corresponding bit in Xk to form a single bit in Xi. A sample computation is listed in octal notation to illustrate the operation performed and includes the four possible bit combinations that may occur.

```
(Xj) = 0123 7777 0123 4567 1010

(Xk) = 0123 4567 7777 3210 1100

(Xi) = 0000 3210 7654 7777 0110
```

This instruction compares bit patterns or complements bit patterns during data processing.

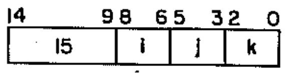
14ixk Transmit Complement of (Xk) to Xi — BXi -Xk



This instruction reads a 60-bit word from Xk, complements the word, and writes the result into Xi.

This instruction changes the sign of a fixed-point or floating-point quantity. The instruction also inverts an entire 60-bit field during data processing.

15ijk Logical Product of (Xj) with Complement of (Xk) to Xi — BXi -Xk * Xj

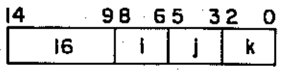


This instruction reads operands from two X registers, operates upon them to form a result, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. The result delivered to Xi is the bit-by-bit logical product of the value in Xj and the complement of the value in Xk. Each of the 60 bits in Xj is compared with the corresponding bit in Xk to form a single bit in Xi. A sample computation is listed in octal notation to illustrate the operation performed and includes the four possible bit combinations that may occur.

```
(Xj) = 7777 7000 0123 4567 1010

(Xk) = 0123 4567 0007 7700 1100

(Xi) = 7654 3000 0120 0067 0010
```

This instruction extracts portions of a 60-bit word during data processing.

16ijk Logical Sum of (Xj) with Complement of (Xk) to Xi — BXi -Xk + Xj

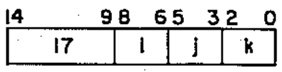


This instruction reads operands from two X registers, operates upon them to form a result, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. The result delivered to Xi is the bit-by-bit logical sum of the value in Xj and the complement of the value in Xk. Each of the 60 bits in Xj is compared with the corresponding bit in Xk to form a single bit in Xi. A sample computation is listed in octal notation to illustrate the operation performed and includes the four possible bit combinations that may occur.

```
(Xj) = 0000 7777 0123 4567 1010

(Xk) = 0123 4567 7777 0000 1100

(Xi) = 7654 7777 0123 7777 7677
```

This instruction merges portions of a 60-bit word into a composite word during data processing.

17ijk Logical Difference of (Xj) with Complement of (Xk) to Xi — BXi -Xk - Xj

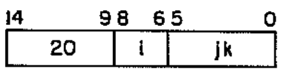


This instruction reads operands from two X registers, operates upon them to form a result, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. The result delivered to Xi is the bit-by-bit logical difference of the value in Xj and the complement of the value in Xk. Each of the 60 bits in Xj is compared with the corresponding bit in Xk to form a single bit in Xi. A sample computation is listed in octal notation to illustrate the operation performed and includes the four possible bit combinations that may occur.

```
(Xj) = 0123 7777 0123 4567 1010

(Xk) = 0123 4567 7777 3210 1100

(Xi) = 7777 4567 0123 0000 7667
```

This instruction compares bit patterns or complements bit patterns during data processing.

20ijk Left Shift (Xi) by jk — LXi jk



This instruction reads one operand from Xi, shifts the 60-bit word left circularly by jk bit positions, and writes the resulting 60-bit word back into the same Xi register. The j and k designators are treated as a single 6-bit positive integer operand in this instruction.

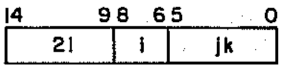
A left-circular shift implies that the bit pattern in the 60-bit word is displaced towards the highest-order bit positions. The bits shifted off the upper end of the 60-bit word are inserted in the lowest-order bit positions in the same sequence. The resulting 60-bit word has the same quantity of bits with values of one and zero as in the original operand.

A sample computation is listed in octal notation to illustrate the operation performed.

```
Initial (Xi) = 2323 6600 0000 0000 0111

jk           = 12 (octal)

Final (Xi)   = 7540 0000 0000 0022 2464
```

This instruction, together with instruction 21, may be used whenever a data word is to be shifted by a predetermined amount. If the amount of shift is derived in the execution of the program, instruction 22 or 23 should be used.

21ijk Right Shift (Xi) by jk AXi jk



This instruction reads one operand from Xi, shifts the 60-bit word right with sign extension by jk bit positions, and writes the resulting 60-bit word back into the same Xi register. The j and k designators are treated as a single 6-bit positive integer operand in this instruction.

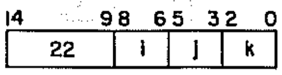
A right shift with sign extension implies that the bit pattern in the 60-bit word is displaced toward the lowest-order bit positions. The bits shifted off the lower end of the word are discarded. The highest-order bit positions are filled with copies of the original sign bit.

Two sample computations are listed in octal notation to illustrate the operation performed. The first example contains a positive operand, and the second example contains a negative operand.

```
Initial (Xi) = 2004 7655 0002 3400 0004

jk           = 30 (octal)

Final (Xi)   = 0000 0000 2004 7655 0002


Initial (Xi) = 6000 4420 2222 0000 5643

jk           = 10 (octal)

Final (Xi)   = 7774 0011 0404 4440 0013
```

This instruction, together with instruction 20, may be used whenever a data word is to be shifted by a predetermined amount. If the amount of shift is derived in the execution of the program, instruction 22 or 23 should be used.

22ijk Left Shift (Xk) Nominally (Bj) Places to Xi LXi Bj Xk



This instruction reads a 60-bit operand from Xk, shifts the data either left or right as specified by Bj, and writes the resulting 60-bit word into Xi. If the value in Bj is positive, the data is left shifted circularly the number of bit positions designated by the value in Bj. If the value in Bj is negative, the data is right shifted with sign extension the number of bit positions designated by the value in Bj. The sign of Bj is determined by Bj bit 17.

A left circular shift implies that the bit pattern in the 60-bit word is displaced towards the highest-order bit positions. The bits shifted off the upper end are inserted in the lowest-order bit positions in the same sequence. The resulting 60-bit word has the same quantity of bits with values of one and zero as in the original operand.

A right shift with sign extension implies that the bit pattern in the 60-bit word is displaced towards the lowest-order positions. The bits shifted off the lower end are discarded. The highest-order bit positions are filled with copies of the original sign bit.

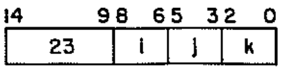
Two sample computations are listed in octal notation to illustrate the operation performed. The first example contains a positive shift count resulting in a left circular shift, and the second example illustrates the right shift with sign extension.

```
(Xk)   =   2323 6600 0000 0000 0111

(Bj)   =   00 0012

(Xi)   =   7540 0000 0000 0022 2464


(Xk)   =   1327 6000 0000 3333 2422

(Bj)   =   77 7771

(Xi)   =   0013 2760 0000 0033 3324
```

If Bj bits 6 through 10 are different from Bj bit 17 and Bj bit 17 is set, the shift count is greater than 63 (decimal) places right, and a result of positive zero is returned to Xi. Bj bits 11 through 16 are not tested by this instruction.

This instruction is used when the amount of shift is derived in the computation. The instruction is also used for correcting the coefficient of a floating-point number when the exponent has been unpacked into a B register.

23ijk Right Shifted (Xk) Nominally (Bj) Places to Xi AXi Bj Xk



This instruction reads a 60-bit operand from Xk, shifts the data either left or right as specified by the content of Bj, and writes the resulting 60-bit word into Xi. If the value in Bj is positive, the data is right shifted with sign extension the number of bit positions designated by the value in Bj. If the value in Bj is negative, the data is left shifted circularly the number of bit positions designated by the value in Bj. The sign of Bj is determined by Bj bit 17.

A left circular shift implies that the bit pattern in the 60-bit word is displaced towards the highest-order bit positions. The bits shifted off the upper end are inserted in the lowest-order bit positions in the same sequence. The resulting 60-bit word has the same quantity of bits with values of one and zero as in the original operand.

A right shift with sign extension implies that the bit pattern in the 60-bit words is displaced towards the lowest-order bit positions. The bits shifted off the lower end of the word are discarded. The highest-order bit positions are filled with copies of the original sign bit.

Two sample computations are listed in octal notation to illustrate the operation performed. The first example contains a positive shift count resulting in a right shift with sign extension, and the second example contains a negative shift count resulting in a left circular shift.

```
(Xk)   =   1327 6000 0000 3333 2422

(Bj)   =   00 0006

(Xi)   =   0013 2760 0000 0033 3324

(Xk)   =   2323 6600 0000 0000 0111

(Bj)   =   77 7765

(Xi)   =   7540 0000 0000 0022 2464
```

If Bj bits 6 through 10 are different from Bj bit 17 and Bj bit 17 is clear, the shift count is greater than 63 (decimal) places right, and a result of positive zero is returned to Xi. Bj bits 11 through 16 are not tested by this instruction.

This instruction is used when the amount of shift is derived in the computation. The instruction is also used for correcting the coefficient of a floating-point number when the exponent has been unpacked into a B register.

24ijk Normalize (Xk) to Xi and Bj NXi Bj Xk

14 … 9	8 … 6	5 … 3	2 … 0
24	i	j	k

This instruction reads one operand from Xk, performs a normalizing operation on this word in floating-point format, and delivers the normalized result to Xi. In addition, a positive integer shift count is sent to Bj. This shift count is the number of bit positions of shift required to normalize the original operand coefficient.

The normalizing operation consists of repositioning the coefficient portion of the operand and then adjusting the exponent portion of the operand to leave the value of the result unaltered. The coefficient is shifted towards the higher-order bit positions of the word. The coefficient is shifted the minimum number of bit positions required to make bit 47 different from sign bit 59. This places the most significant bit of the coefficient in the highest-order position. The exponent is then decreased by the number of bit positions shifted.

Two sample computations are listed in octal notation to illustrate the operation performed. The first example involves a positive floating-point number, and the second example involves a negative number.

```
(Xk)   =   2034 0047 6500 0000 2262

(Xi)   =   2026 4765 0000 0022 6200

(Bj)   =   00 0006

(Xk)   =   5743 7730 1277 7777 5515

(Xi)   =   5751 3012 7777 7755 1577

(Bj)   =   00 0006
```

Normalizing a number with either a positive or negative zero coefficient sets a shift count in Bj to 48 (decimal) and enters Xi with positive zero. If Xk contains an infinite quantity (3777xxx...x or 4000xxx...x) or an indefinite quantity (1777xxx...x or 6000xxx...x), no shift takes place. The content of Xk is copied to Xi, and Bj is set to zero. Corresponding infinite and indefinite exit conditions are also set in the CP for exit mode action. For further information, refer to Floating-Point Arithmetic under CP Programming in section 5.

25ijk Round Normalize (Xk) to Xi and Bj ZXi Bj Xk

14 … 9	8 … 6	5 … 3	2 … 0
25	i	j	k

This instruction reads one operand from Xk, performs a rounding and then a normalizing operation in floating-point format, and delivers the round normalized result to Xi. In addition, a positive integer shift count is sent to Bj. This shift count is the number of bit positions of shift required to normalize the original operand coefficient.

The rounding operation consists of adding a bit to the coefficient portion of the operand in a bit position immediately below the least significant bit position. This round bit has a value equal to the complement of the operand sign bit. The result increases the magnitude of the coefficient by one-half the value of the least significant bit.

The normalizing operation consists of repositioning the coefficient and adjusting the exponent to leave the value of the resulting floating-point quantity unaltered. The coefficient is shifted towards the higher-order bit positions. The round bit is shifted along with the coefficient. The displacement is the minimum number of bit positions required to make bit 47 different from sign bit 59. This places the most significant bit of the coefficient in the highest-order bit position. The exponent is decreased by the number of bit positions shifted.

Two sample computations are listed in octal notation to illustrate the normalizing operation performed. The first example involves a positive floating-point number, and the second example involves a negative number.

(Xk)	=	2034 0047 6500 0000 2262
(Xi)	=	2026 4765 0000 0022 6420
(Bj)	=	00 0006

(Xk)	=	5743 7730 1277 7777 5515
(Xi)	=	5751 3012 7777 7755 1537
(Bj)	=	00 0006

If Xk contains either an infinite quantity (3777xxx...x or 4000xxx...x) or an indefinite quantity (1777xxx...x or 6000xxx...x), no shift takes place. The content of Xk is copied to Xi, and Bj is set to zero. Corresponding infinite and indefinite exit conditions are also set in the CP for exit mode action. For further information, refer to Floating-Point Arithmetic under CP Programming in section 5.

26ijk Unpack (Xk) to Xi and Bj UXi Bj Xk

14 9	8 6	5 3	2 0
26	i	j	k

This instruction reads one operand from Xk, unpacks this word from floating-point format, and delivers the coefficient and exponents to Xi and Bj, respectively. The 60-bit word delivered to Xi consists of the lowest 48 bits unaltered from the original operand plus the upper 12 bits, each equal to the original sign bit. This is a signed integer equal to the value of the coefficient in the original operand. The 18-bit quantity delivered to Bj is a signed integer equal to the value of the exponent in the original operand. The 11-bit exponent field in the operand is altered to remove the bias and then sign extended to fill out the 18-bit quantity. The sign of the coefficient is removed in this process.

Four sample sets of operands and unpacked results are listed in octal notation to illustrate the operation performed. These examples contain the four combinations of coefficient sign and exponent sign.

(Xk)	=	2034 4500 3333 2000 0077
(Xi)	=	0000 4500 3333 2000 0077
(Bj)	=	00 0034

(Xk)	=	1743 4500 3333 2000 0077
(Xi)	=	0000 4500 3333 2000 0077
(Bj)	=	77 7743

(Xk)	=	5743 3277 4444 5777 7700
(Xi)	=	7777 3277 4444 5777 7700
(Bj)	=	00 0034

(Xk)	=	6034 3277 4444 5777 7700
(Xi)	=	7777 3277 4444 5777 7700
(Bj)	=	77 7743

This instruction converts a number from floating-point format to fixed-point format. For further information, refer to Floating-Point Arithmetic under CP Programming in section 5.

27ijk Pack (Xk) and (Bj) to Xi PXi Bj Xk

14 9	8 6	5 3	2 0
27	i	j	k

This instruction reads the contents of Xk and Bj, packs them into a single word in floating-point format, and delivers this result to Xi. The coefficient for the value in Xi is obtained from the content of Xk, which is treated as a signed integer. The exponent for the value in Xi is obtained from the content of Bj, which is treated as a signed integer.

The lowest-order 48 bits in Xi are copied directly from the lowest-order 48 bits in Xk. The sign bit in Xi is copied directly from the sign bit in Xk. The exponent field in Xi is derived from the value in Bj by extracting the lowest-order 11 bits in Bj and modifying this quantity for exponent bias and coefficient sign.

Four sample sets of operands and packed results are listed in octal notation to illustrate the operation performed. These examples contain the four combinations of coefficient sign and exponent sign.

(Xk)	=	0000 4500 3333 2000 0077
(Bj)	=	00 0034
(Xi)	=	2034 4500 3333 2000 0077

(Xk)	=	0000 4500 3333 2000 0077
(Bj)	=	77 7743
(Xi)	=	1743 4500 3333 2000 0077

(Xk)	=	7777 3277 4444 5777 7700
(Bj)	=	00 0034
(Xi)	=	5743 3277 4444 5777 7700

(Xk)	=	7777 3277 4444 5777 7700
(Bj)	=	77 7743
(Xi)	=	6034 3277 4444 5777 7700

This instruction converts a number in fixed-point format to floating-point format.

30ijk Floating Sum of (Xj) and (Xk) to Xi FXi Xj + Xk

14 9	8 6	5 3	2 0
30	i	j	k

This instruction reads operands from two X registers, operates upon them to form a floating-point sum, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are in floating-point format and are not necessarily normalized. The sum of the quantities in Xj and Xk is delivered to Xi in floating-point format and is not necessarily normalized.

The two operands are unpacked from floating-point format, and the exponents are compared. The coefficient with the smaller exponent is right shifted by the difference of the two exponents such that both coefficients are the same significance. The two coefficients are then added to form a 96-bit result. The upper half of the result is then selected as a coefficient and packed along with the larger exponent to form the result sent to Xi. If coefficient overflow occurs, the sum is right shifted one place, and the exponent is increased by one.

If the two operands have unlike signs, the result coefficient may have leading zeros. No normalize operation is built into this instruction to correct this situation. A separate normalize instruction must be programmed if the result is to be kept in a normalized form.

When the difference between the exponents is greater than 128 (decimal), the shifted sign bit is extended to the entire shifted operand. Infinite (3777xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands cause corresponding exit conditions to set in the CP for exit mode action. For further information, refer to Floating-Point Arithmetic under CP Programming in section 5.

31ijk Floating Difference of (Xj) and (Xk) to Xi FXi Xj - Xk

14 9	8 6	5 3	2 0
31	i	j	k

This instruction reads operands from two X registers, operates upon them to form a floating-point difference, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are in floating-point format and are not necessarily normalized. The result of subtracting the quantity in Xk from the quantity in Xj is delivered to Xi in floating-point format and is not necessarily normalized.

The two operands are unpacked from floating-point format, and the exponents are compared. The coefficient with the smaller exponent is right shifted by the difference of the two exponents such that both coefficients are the same significance. The Xk coefficient is then subtracted from the Xj coefficient to form a 96-bit result. The upper half of the result is then selected and packed along with the larger exponent to form the result sent to Xi. If coefficient overflow occurs, the result is right shifted one place, and the exponent is increased by one.

If the two operands have like signs, the result coefficient may have leading zeros. No normalize operation is built into this instruction to correct this situation. A separate normalize instruction must be programmed if the result is to be kept in a normalized form. Infinite (3777xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands cause corresponding exit conditions to set in the CP for exit mode action. For further information, refer to Floating-Point Arithmetic under CP Programming in section 5.

32ijk Floating Double-Precision Sum of (Xj) and (Xk) to Xi DXi Xj + Xk

14 9	8 6	5 3	2 0
32	i	j	k

This instruction reads operands from two X registers, operates upon them to form a double-precision floating-point sum, and delivers the lower half of this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are in floating-point format and are not necessarily normalized. The sum of the quantities in Xj and Xk is delivered to Xi in floating-point format and is not necessarily normalized.

The two operands are unpacked from floating-point format, and the exponents are compared. The coefficient with the smaller exponent is right shifted by the difference of the two exponents such that both coefficients are the same significance. The two coefficients are then added to form a 96-bit result. The lower half of the result is then selected and packed along with the larger exponent minus 48 (decimal) to form the result sent to Xi. If coefficient overflow occurs, the result is right shifted by one place, and the exponent is increased by one. Infinite (3777xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands cause corresponding exit conditions to set in the CP for exit mode action. For further information, refer to Floating-Point Arithmetic under CP Programming in section 5.

33ijk Floating Double-Precision Difference of (Xj) and (Xk) to Xi DXi Xj - Xk

14 9	8 6	5 3	2 0
33	i	j	k

This instruction reads operands from two X registers, operates upon them to form a double-precision floating-point difference, and delivers the lower half of this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are in floating-point format and are not necessarily normalized. The result of subtracting the quantity in Xk from the quantity in Xj is delivered to Xi in floating-point format and is not necessarily normalized.

The two operands are unpacked from floating-point format, and the exponents are compared. The coefficient with the smaller exponent is right shifted by the difference of the two exponents such that both coefficients are the same significance.

The Xk coefficient is then subtracted from the Xj coefficient to form a 96-bit result. The lower half of the result is then selected and packed along with the larger exponent minus 48 (decimal) to form the result sent to Xi. If coefficient overflow occurs, the result is right shifted one place, and the exponent is increased by one.

Infinite (3777xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands cause corresponding exit conditions to set in the CP for exit mode action. For further information, refer to Floating-Point Arithmetic under CP Programming in section 5.

34ijk Round Floating Sum of (Xj) and (Xk) to Xi — RXi Xj + Xk

14 9	8 6	5 3	2 0
34	i	j	k

This instruction reads operands from two X registers, operates upon them to form a rounded floating-point sum, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are in floating-point format and are not necessarily normalized. The result is delivered to Xi in floating-point format and is not necessarily normalized.

The round floating-point sum is a single-precision floating-point sum with a round bit (or bits) inserted before the add operation takes place. A round bit is always inserted in the coefficient with the larger exponent. If the two exponents are equal, the round bit is inserted in the coefficient for Xk. The round bit is equal to the complement of the sign bit and is inserted immediately to the right of the lowest-order bit in the coefficient. This has the effect of increasing the magnitude of the coefficient by one-half of the least significant bit. A second round bit is inserted in a corresponding manner to the other coefficient if both operands are normalized or have unlike signs. The second round bit is inserted before the coefficient has been shifted by the difference of the exponents. Infinite (3777xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands cause corresponding exit conditions to set in the CP for exit mode action. For further information, refer to Floating-Point Arithmetic under CP Programming in section 5.

35ijk Round Floating Difference of (Xj) and (Xk) to Xi — RXi Xj - Xk

14 9	8 6	5 3	2 0
35	i	j	k

This instruction reads operands from two X registers, operates upon them to form a rounded floating-point difference, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are in floating-point format and are not necessarily normalized. The result of subtracting the quantity in Xk from the quantity in Xj is delivered to Xi in floating-point format and is not necessarily normalized.

The round floating-point difference is a single-precision floating-point difference with a round bit (or bits) inserted before the subtract operation takes place. A round bit is always inserted in the coefficient with the larger exponent. If the two exponents are equal, the round bit is added to the coefficient for Xk. The round bit is equal to the complement of the sign bit and is inserted immediately to the right of the lowest-order bit in the coefficient. This has the effect of increasing the magnitude of the coefficient by one-half of the least significant bit. A second round bit is inserted in a corresponding manner to the other coefficient if both operands are normalized or have like signs. The second round bit is inserted before the coefficient has been shifted by the difference of the exponents. Infinite (3777xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands cause corresponding exit conditions to set in the CP for exit mode action. For further information, refer to Floating-Point Arithmetic under CP Programming in section 5.

36ijk Integer Sum of (Xj) and (Xk) to Xi — IXi Xj + Xk

14 9	8 6	5 3	2 0
36	i	j	k

This instruction reads operands from two X registers, operates upon them to form a 60-bit integer sum, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are signed integers. The resulting integer sum is delivered to Xi. Overflow is not detected.

This instruction adds integers too large for handling by 50 through 77 instructions. The instruction also merges and compares data fields during data processing. For further information, refer to Floating-Point Arithmetic under CP Programming in section 5.

37ijk Integer Difference of (Xj) and (Xk) to Xi — IXi Xj - Xk

14 9	8 6	5 3	2 0
37	i	j	k

This instruction reads operands from two X registers, operates upon them to form a 60-bit integer difference, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are signed integers. The result of subtracting the quantity in Xk from the quantity in Xj is delivered to Xi. Overflow is not detected.

This instruction subtracts integers too large for handling by 50 through 77 instructions. The instruction also compares data fields during data processing. For further information, refer to Floating-Point Arithmetic under CP Programming in section 5.

40ijk Floating Product of (Xj) and (Xk) to Xi — FXi Xj * Xk

14 9	8 6	5 3	2 0
40	i	j	k

This instruction reads operands from two X registers, operates upon them to form a floating-point product, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are in floating-point format and are not necessarily normalized. The result is delivered to Xi in floating-point format. If both operands are normalized, the result is also normalized. If both operands are not normalized, the result is not normalized.

The two operands are unpacked from floating-point format. The exponents are added with a correction factor to determine the exponent for the result. The coefficients are multiplied as signed integers to form a 96-bit integer product. The upper half of this product is extracted to form the coefficient for the result. If the original operands are normalized and the product has only 95 significant bits, a 1-bit left shift to normalize the result coefficient is done. The resulting exponent is reduced by one count in this case.

If both operands are not normalized, the resulting double-precision product has less than 96 significant bits. No test is made for the position of the most significant bit. The upper 48 bits are read from the double-precision product register. Leading zeros occur in this result coefficient.

This instruction is used in floating-point calculations where rounding of operands is not desired, such as in multiple-precision arithmetic and in calculations involving error analysis. Infinite (3777xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands cause corresponding exit conditions to set in the CP for exit mode action. For further information, refer to Floating-Point Arithmetic under CP Programming in section 5.

41ijk Round Floating Product of (Xj) and (Xk) to Xi — RXi Xj * Xk

14 9	8 6	5 3	2 0
41	i	j	k

This instruction reads operands from two X registers, operates upon them to form a rounded floating-point product, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are in floating-point format and are not necessarily normalized. The result is delivered to Xi in floating-point format. If both operands are normalized, the result is also normalized. If both operands are not normalized, the result is not normalized.

The two operands are unpacked from floating-point format. The exponents are added with a correction factor to determine the exponent for the result. The coefficients are multiplied as signed integers to form a 96-bit integer product. A rounding bit is added to bit position 46 of this product. The upper half of this product is extracted to form the coefficient for the result. If the original operands are normalized and the product has only 95 significant bits, a 1-bit left shift to normalize the result coefficient is done. The resulting exponent is reduced by one count in this case.

If both operands are not normalized, the resulting double-precision product has less than 96 significant bits. No test is made for the position of the most significant bit. The upper 48 bits are read from the double-precision product register. Leading zeros occur in this result coefficient.

This instruction is used in single-precision floating-point calculations. For multiple-precision calculations, the 40 and 42 instructions must be used. Infinite (3777xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands cause corresponding exit conditions to set in the CP for exit mode action. For further information, refer to Floating-Point Arithmetic under CP Programming in section 5.

42ijk Floating Double-Precision Product of (Xj) and (Xk) to Xi — DXi Xj * Xk

14 9	8 6	5 3	2 0
42	i	j	k

This instruction reads operands from two X registers, operates upon them to form a double-precision floating-point product, and delivers the lower half of this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are in floating-point format and are not necessarily normalized. The lower half of the double-precision product is delivered to Xi in floating-point format and is not necessarily normalized.

The operands are not rounded in this operation. The two operands are unpacked from floating-point format. The exponents are added to determine the exponent for the result. The result exponent is exactly 48 less than the exponent for a 40 instruction. The coefficients are multiplied as signed integers to form a 96-bit integer product. The lower half of this product is extracted to form the coefficient for the result. If the original operands are normalized and the double-precision product has only 95 significant bits, a 1-bit left shift to normalize the result coefficient is done. The resulting exponent is reduced by one count in this case.

If both operands are not normalized, the resulting double-precision product has less than 96 significant bits. No test is made for the position of the most significant bit. The lower 48 bits are always read from the 96-bit product register.

This instruction is used in multiple-precision floating-point calculations. This instruction also provides for integer multiplication capabilities where both operands have an exponent value of plus or minus zero, and neither coefficient has been normalized. The integer result sent to Xi is 48 bits with 60-bit sign extension. If the result exceeds 48 bits, the hardware does not detect an overflow. An overflow check can be made by executing a 40 instruction using the same two operands. If the result is nonzero, overflow is then indicated. An integer multiply operation is not intended to be used with normalized operands. Infinite (3777xxx...x and 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands cause corresponding exit conditions to set in the CP for exit mode action. For further information, refer to Floating-Point Arithmetic under CP Programming in section 5.

43ijk Form Mask of jk Bits to Xi MXi jk

```
14      98 65 32 0
|  43   | i | j | k |
```

This instruction generates a masking word using the j and k designators as parameters. No operands are read from operating registers. The j and k designators are treated as a single 6-bit octal quantity to designate the width of the masking field. A field of ones, beginning at the highest-order end of the word, is extended downward on a background of zeros. The completed masking word consists of one bits in the highest-order jk bit positions and zero bits in the remainder of the word. This masking word is then delivered to Xi. The following are sample parameters.

j = 2

k = 4

Xi = 7777 7760 0000 0000 0000

This instruction generates variable width masks for logical operations. This instruction, together with a shift instruction, generally creates an arbitrary field mask faster than reading a pregenerated mask from CM.

44ijk Floating Divide (Xj) by (Xk) to Xi FXi Xj/Xk

```
14      98 65 32 0
|  44   | i | j | k |
```

This instruction reads operands from two X registers, operates upon them to form a floating-point quotient, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are in floating-point format. The result of dividing the content of Xj by the content of Xk is delivered to Xi. If both operands are normalized, the quotient is also normalized. The remainder from the division process is discarded.

The two operands are unpacked from floating-point format. The exponents are subtracted with a correction factor to determine the exponent for the result. The coefficient from Xj is positioned in a dividend register. The coefficient from Xk is trial-subtracted repeatedly from the dividend. The quotient bits are assembled in a quotient register. When 48 bits of the quotient are assembled, they are packed with the result exponent into floating-point format and delivered to Xi.

If the exponent subtraction causes an underflow or overflow, an underflow or overflow result is returned even with the occurrence of a divide fault.

If the dividend is not normalized, the quotient cannot be normalized. However, the quotient is correct even though there may be leading zeros in the coefficient. If the divisor is not normalized, the quotient may be incorrect. If the coefficient for the content of Xj is larger than the coefficient for the content of Xk by a factor of two or more, a divide fault causes an indefinite result to be returned to Xi. For further information, refer to Floating-Point Arithmetic under CP Programming in section 5.

This instruction is used in floating-point calculations where rounding of operands is not desired. In multiple-precision division, this instruction must be followed by a multiplication of the quotient by the divisor and subtracted from the dividend to reconstruct the remainder.

If infinite (3777xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands are used, corresponding exit conditions are set in the CP for exit mode action. For further information, refer to Floating-Point Arithmetic under CP Programming in section 5.

45ijk Round Floating Divide (Xj) by (Xk) to Xi RXi Xj/Xk

```
14      98 65 32 0
|  45   | i | j | k |
```

This instruction reads operands from two X registers, operates upon them to form a rounded floating-point quotient, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are in floating-point format. The result of dividing the content of Xj by the content of Xk is delivered to Xi. If both operands are normalized, the quotient is also normalized. The remainder from the division process is discarded.

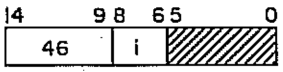
The two operands are unpacked from floating-point format in this operation. The exponents are subtracted with a correction factor to determine the exponent for the result. The coefficient from Xj is positioned in a dividend register. The Xj quantity is modified by inserting a 2525...25 round pattern below the lowest-order bit of the dividend coefficient. The coefficient from Xk is trial-subtracted repeatedly from the dividend. The quotient bits are assembled in a quotient register. When 48 bits of the quotient are assembled, they are packed with the result exponent into floating-point format and delivered to Xi.

If the dividend is not normalized, the quotient cannot be normalized. However, the quotient is correct even though there may be leading zeros in the coefficient. If the divisor is not normalized, the quotient may be incorrect. If the coefficient for the content of Xj is larger than the coefficient for the content of Xk by a factor of two or more, a divide fault occurs. A divide fault causes an indefinite result to be returned to Xi. Refer to Floating-Point Arithmetic under CP Programming in section 5.

This instruction is used in single-precision floating-point calculations where rounding of operands is desired to reduce truncation errors.

If infinite (3777xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands are used, corresponding exit conditions are set in the CP for exit mode action. For further information, refer to Floating-Point Arithmetic under CP Programming in section 5.

460xx through 463xx Pass NO

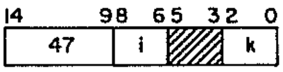


These instructions fill program instruction words where necessary to match jump destinations with word boundaries. The j and k designators are ignored, and a nonzero value has no effect in this instruction.

464 through 467 Instructions

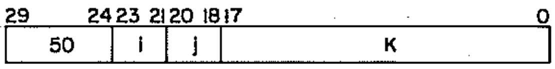
These instructions are illegal instructions. Refer to Illegal Instructions under CP Programming in section 5.

47ixk Population Count of (Xk) to Xi CXi Xk



This instruction reads one operand from Xk, counts the number of one bits in the operand, and stores the count in Xi. The count delivered to Xi is a positive integer. If the operand is all ones, a count of 60 (decimal) is delivered to Xi. If operand is all zeros, a zero word is delivered to Xi.

50ijK Set Ai to (Aj) + K SAi Aj K

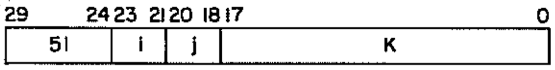


This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction reads an operand from Aj, forms the sum of the operand plus K, and delivers the result to Ai. If the i designator is nonzero, a reference is made to CM using the result as the relative address. The type of reference is a function of the i designator value.

i = 0	No CM reference
i = 1,2,3,4,5	Read from CM to Xi
i = 6,7	Write into CM from Xi

This instruction obtains operands from CM for computation and delivers the result back into CM.

51ijK Set Ai to (Bj) + K SAi Bj K

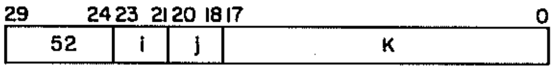


This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction reads an operand from Bj, forms the sum of the operand plus K, and delivers the result to Ai. If the i designator is nonzero, a reference is made to CM using the result as the relative address. The type of reference is a function of the i designator value.

i = 0	No CM reference
i = 1,2,3,4,5	Read from CM to Xi
i = 6,7	Write into CM from Xi

This instruction obtains operands from CM for computation and delivers the result back into CM.

52ijK Set Ai to (Xj) + K SAi Xj K

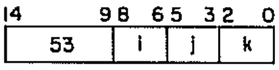


This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction reads an operand from Xj, forms the sum of the operand plus K, and delivers the result to Ai. If the i designator is nonzero, a reference is made to CM using the result as the relative address. The type of reference is a function of the i designator value.

i = 0	No CM reference
i = 1,2,3,4,5	Read from CM to Xi
i = 6,7	Write into CM from Xi

This instruction obtains operands from CM for computation and delivers the result back into CM.

53ijk Set Ai to (Xj) + (Bk) SAi Xj + Bk

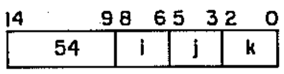


This instruction reads operands from Xj and Bk, forms the sum of the operands, and delivers the result to Ai. If the i designator is nonzero, a reference is made to CM using the result as the relative address. The type of reference is a function of the i designator value.

i = 0	No CM reference
i = 1,2,3,4,5	Read from CM to Xi
i = 6,7	Write into CM from Xi

This instruction obtains operands from CM for computation and delivers the result back into CM.

54ijk Set Ai to (Aj) + (Bk) SAi Aj + Bk

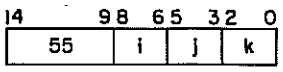


This instruction reads operands from Aj and Bk, forms the sum of the operands, and delivers the result to Ai. If the i designator is nonzero, a reference is made to CM using the result as the relative address. The type of reference is a function of the i designator value.

i = 0	No CM reference
i = 1,2,3,4,5	Read from CM to Xi
i = 6,7	Write into CM from Xi

This instruction obtains operands from CM for computation and delivers the result back into CM.

55ijk Set Ai to (Aj) - (Bk) SAi Aj - Bk

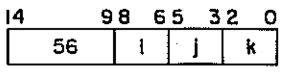


This instruction reads operands from Aj and Bk, subtracts the Bk operand from the Aj operand, and delivers the result to Ai. If the i designator is nonzero, a reference is made to CM using the result as the relative address. The type of reference is a function of the i designator value.

i = 0	No CM reference
i = 1,2,3,4,5	Read from CM to Xi
i = 6,7	Write into CM from Xi

This instruction obtains operands from CM for computation and delivers the results back into CM.

56ijk Set Ai to (Bj) + (Bk) SAi Bj + Bk

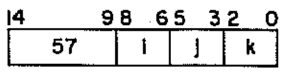


This instruction reads operands from Bj and Bk, forms the sum of the operands, and delivers the result to Ai. If the i designator is nonzero, a reference is made to CM using the result as the relative address. The type of reference is a function of the i designator value.

i = 0	No CM reference
i = 1,2,3,4,5	Read from CM to Xi
i = 6,7	Write into CM from Xi

This instruction obtains operands from CM for computation and delivers the results back into CM.

57ijk Set Ai to (Bj) - (Bk) SAi Bj - Bk

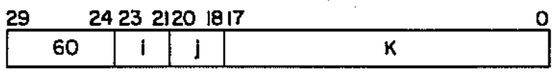


This instruction reads operands from Bj and Bk, subtracts the Bk operand from the Bj operand, and delivers the result to Ai. If the i designator is non-zero, a reference is made to CM using the result as the relative address. The type of reference is a function of the i designator value.

i = 0	No CM reference
i = 1,2,3,4,5	Read from CM to Xi
i = 6,7	Write into CM from Xi

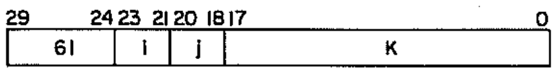
This instruction obtains operands from CM for computation and delivers the result back into CM.

60ijK Set Bi to (Aj) + K SBi Aj K



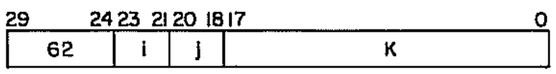
This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction reads an operand from Aj, forms the sum of the operand plus K and delivers the result to Bi. The sum is formed in an 18-bit one's complement mode. This instruction is for address modification in the increment registers.

61ijK Set Bi to (Bj) + K SBi Bj K



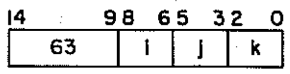
This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction reads an operand from Bj, forms the sum of the operand plus K, and delivers the result to Bi. The sum is formed in an 18-bit one's complement mode.

62ijK Set Bi to (Xj) + K SBi Xj K



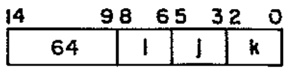
This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction reads an operand from Xj, forms the sum of the operand plus K, and delivers the result to Bi. The sum is formed in an 18-bit one's complement mode.

63ijk Set Bi to (Xj) + (Bk) SBi Xj + Bk



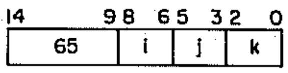
This instruction reads operands from Xj and Bk, adds the operands, and delivers the result to Bi. The sum is formed in an 18-bit one's complement mode.

64ijk Set Bi to (Aj) + (Bk) SBi Aj + Bk



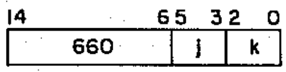
This instruction reads operands from Aj and Bk, adds the operands, and delivers the result to Bi. The sum is formed in an 18-bit one's complement mode.

65ijk Set Bi to (Aj) - (Bk) SBi Aj - Bk



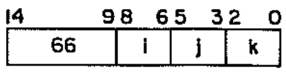
This instruction reads operands from Aj and Bk, subtracts the Bk operand from the Aj operand, and delivers the result to Bi. The difference is formed in an 18-bit one's complement mode. If the i designator is zero, this becomes a pass instruction.

660jk Read Central Memory at (Xk) to Xj CRXj Xk



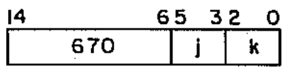
This instruction places into Xj the word at location (Xk), where Xk is a right-justified 21-bit relative word address. Bits 21 through 59 of Xk are ignored. RAC plus Xk is less than FLC.

66ijk Set Bi to (Bj) + (Bk) SBi Bj + Bk



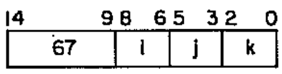
This instruction reads operands from Bj and Bk, adds the operands, and delivers the result to Bi. The sum is formed in an 18-bit one's complement mode.

670jk Write Xj into Central Memory at (Xk) CWXj Xk



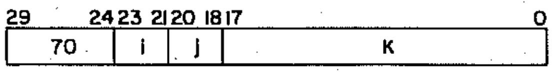
This instruction places Xj into location (Xk) where Xk is a 21-bit relative word address. Bits 21 through 59 of Xk are ignored. RAC plus Xk must be less than FLC.

67ijk Set Bi to (Bj) - (Bk) SBi Bj - Bk



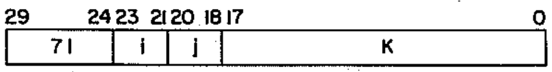
This instruction reads operands from Bj and Bk, subtracts the Bk operand from the Bj operand, and delivers the result to Bi. The difference is formed in an 18-bit one's complement mode.

70ijK Set Xi to (Aj) + K SXi Aj K



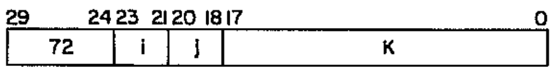
This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction reads an operand from Aj, forms the sum of the operand plus K, and delivers the result to Xi. The sum is formed in an 18-bit one's complement mode. The 18-bit result is sign-extended by copying the highest-order bit or the result into the upper 42 bit positions in Xi.

71ijK Set Xi to (Bj) + K SXi Bj K



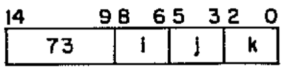
This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction reads an operand from Bj, forms the sum of the operand plus K, and delivers the result to Xi. The sum is formed in an 18-bit one's complement mode. The 18-bit result is sign-extended by copying the highest-order bit of the result into the upper 42 bit positions in Xi.

72ijK Set Xi to (Xj) + K SXi Xj K



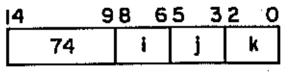
This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction reads an operand from Xj, forms the sum of the operand plus K, and delivers the result to Xi. The sum is formed in an 18-bit one's complement mode. The 18-bit result is sign-extended by copying the highest-order bit of the result into the upper 42 bit positions in Xi.

73ijk   Set Xi to (Xj) + (Bk)   SXi Xj + Bk



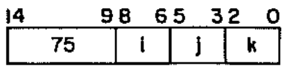
This instruction reads operands from Xj and Bk, adds the operands, and delivers the result to Xi. The sum is formed in an 18-bit one's complement mode. The 18-bit result is sign-extended by copying the highest-order bit of the result into the upper 42 bit positions in Xi.

74ijk   Set Xi to (Aj) + (Bk)   SXi Aj + Bk



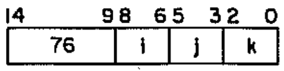
This instruction reads operands from Aj and Bk, adds the operands, and delivers the result to Xi. The sum is formed in an 18-bit one's complement mode. The 18-bit result is sign-extended by copying the highest-order bit of the result into the upper 42 bit positions in Xi.

75ijk   Set Xi to (Aj) - (Bk)   SXi Aj - Bk



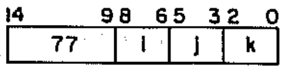
This instruction reads operands from Aj and Bk, subtracts the Bk operand from the Aj operand, and delivers the result to Xi. The difference is formed in an 18-bit one's complement mode. The 18-bit result is sign-extended by copying the highest-order bit of the result into the upper 42 bit positions in Xi.

76ijk   Set Xi to (Bj) + (Bk)   SXi Bj + Bk



This instruction reads operands from Bj and Bk, adds the operands, and delivers the result to Xi. The sum is formed in an 18-bit one's complement mode. The 18-bit result is sign-extended by copying the highest-order bit of the result into the upper 42 bit positions in Xi.

77ijk   Set Xi to (Bj) - (Bk)   SXi Bj - Bk



This instruction reads operands from Bj and Bk, subtracts the Bk operand from the Bj operand, and delivers the result to Xi. The difference is formed in an 18-bit one's complement mode. The 18-bit result is sign-extended by copying the highest-order bit of the result into the upper 42 bit positions in Xi.

INSTRUCTION EXECUTION TIMING

Execution times for the models 865 and 875 CP instructions are listed in table 4-2. The execution times are listed with the assumption that no conflicts occur. Execution delays result unless all the conditions listed in the timing notes column exist for the particular instruction. The numbers in the timing notes column refer to notes listed at the end of the table. Execution time groups are as follows where one clock cycle equals 25 nanoseconds.

Table 4-2. CP Instruction Timing (Sheet 1 of 5)

Instruction Code	Description	Functional Unit	Execution Time (CP)†	Timing Notes
00xxx	Error exit to MA or program stop	-	-	-
010xK	Return jump to K	-	18 [11]	1,2,3,18,19
011jK UEM flag clear	Block copy (Bj) + K words from EEM to CM	Storage move	((Bj) + K)4	4,5,6,7,9
012jK UEM flag clear	Block copy (Bj) + K K words from CM to EEM	Storage move	((Bj) + K)4	4,5,6,7,9, 20,21,22
011jK UEM flag set	Block copy (Bj) + K words from UEM to CM	Storage move	((Bj) + K)4 [((Bj) + K)2]	2,3,4,7,22
012jK UEM flag set	Block copy (Bj) + K words from CM to UEM	Storage move	((Bj) + K)4 [((Bj) + K)2]	2,3,4,7,22
013jK	Central exchange jump to (Bj) + K (monitor flag set)	-	82 [43]	1,2,4,22,23
013xx	Central exchange jump to MA (monitor flag not set)	-	82 [43]	1,2,4,22, 23,24
014jk UEM flag clear	One word read EEM to Xj	Storage move	††	2,3,4
015jk UEM flag clear	One word write Xj to EEM	Storage move	††	2,3,4
014jk UEM flag set	One word read UEM to Xj	Increment	15 [8]	2,3,8,12, 18,19,25
015jk UEM flag set	One word write Xj to UEM	Increment	2	2,3,8,12
016jk	Read microsecond counter to Xj	-	2	8,12,13
02ixK	Jump to (Bi) + K	-	17 [12]	1,2,3,8,18,19
030jK	Branch to K if (Xj) = 0	-	17 [12]	1,2,3,10, 11,18,19
031jK	Branch to K if (Xj) ≠ 0	-	17 [12]	1,2,3,10, 11,18,19
032jK	Branch to K if (Xj) positive	-	17 [12]	1,2,3,10, 11,18,19
033jK	Branch to K if (Xj) negative	-	17 [12]	1,2,3,10, 11,18,19
034jK	Branch to K if (Xj) in range	-	17 [12]	1,2,3,10, 11,18,19
035jK	Branch to K if (Xj) out of range	-	17 [12]	1,2,3,10, 11,18,19
036jK	Branch to K if (Xj) definite	-	17 [12]	1,2,3,10, 11,18,19
037jK	Branch to K if (Xj) indefinite	-	17 [12]	1,2,3,10, 11,18,19

†The nonbracketed execution times apply to model 865. Execution times in brackets [ ] apply to model 875 (single or dual CP); in this case, notes 18 and 19 do not apply. A single execution time for an instruction (25 ns) applies to both models (single or dual CP).
††Not available at this time.

Table 4-2. CP Instruction Timing (Sheet 2 of 5)

Instruction Code	Description	Functional Unit	Execution Time (CP)†	Timing Notes
04ijK	Branch to K if (Bi) = (Bj)	-	17 [12]	1,2,3,10, 11,18,19
05ijK	Branch to K if (Bi) ≠ (Bj)	-	17 [12]	1,2,3,10, 11,18,19
06ijK	Branch to K if (Bi) ≥ (Bj)	-	17 [12]	1,2,3,10, 11,18,19
07ijK	Branch to K if (Bi) ≤ (Bj)	-	17 [12]	1,2,3,10, 11,18,19
10ijx	Transmit (Xj) to Xi	Boolean	2	8,12,13
11ijk	Logical product of (Xj) and (Xk) to Xi	Boolean	2	8,12,13
12ijk	Logical sum of(Xj) and (Xk) to Xi	Boolean	2	8,12,13
13ijk	Logical difference of (Xj) and (Xk) to Xi	Boolean	2	8,12,13
14ijk	Transmit complement of (Xk) to Xi	Boolean	2	8,12,13
15ijk	Logical product of (Xj) and complement of (Xk) to Xi	Boolean	2	8,12,13
16ijk	Logical sum of (Xj) and complement of (Xk) to Xi	Boolean	2	8,12,13
17ijk	Logical difference of (Xj) and complement of (Xk) to Xi	Boolean	2	8,12,13
20ijk	Left shift (Xi) by jk	Shift	2	8,12,13
21ijk	Right shift (Xi) by jk	Shift	2	8,12,13
22ijk	Left shift (Xk) nominally (Bj) places to Xi	Shift	2	8,12,13
23ijk	Right shift (Xk) nominally (Bj) places to Xi	Shift	2	8,12,13
24ijk	Normalize (Xk) to Xi and Bj	Normalize	3	8,12,13
25ijk	Round normalize (Xk) to Xi and Bj	Normalize	3	8,12,13
26ijk	Unpack (Xk) to Xi and Bj	Boolean	2	8,12,13
27ijk	Pack (Xk) and (Bj) to Xi	Boolean	2	8,12,13
30ijk	Floating sum of (Xj) and (Xk) to Xi	Floating add	4	8,12,13
31ijk	Floating sum of (Xj) and (Xk) to Xi	Floating add	4	8,12,13
32ijk	Floating double-precision sum of (Xj) and (Xk) to to Xi	Floating add	4	8,12,13

† The nonbracketed execution times apply to model 865. Execution times in brackets [ ] apply to model 875 (single or dual CP); in this case, notes 18 and 19 do not apply. A single execution time for an instruction (25 ns) applies to both models (single or dual CP).

Table 4-2. CP Instruction Timing (Sheet 3 of 5)

Instruction Code	Description	Functional Unit	Execution Time (CP)†	Timing Notes
33ijk	Floating double-precision difference of (Xj) and (Xk) to Xi	Floating add	4	8,12,13
34ijk	Round floating sum of (Xj) and (Xk) to Xi	Floating add	4	8,12,13
35ijk	Round floating difference of (Xj) and (Xk) to Xi	Floating add	4	8,12,13
36ijk	Integer sum of (Xj) and (Xk) to Xi	Long add	2	8,12,13
37ijk	Integer difference of (Xj) and (Xk) to Xi	Long add	2	8,12,13
40ijk	Floating product of (Xj) and (Xk) to Xi	Floating multiply	5	8,12,13
41ijk	Round floating product of (Xj) and (Xk) to Xi	Floating multiply	5	8,12,13
42ijk	Floating double-precision product of (Xj) and (Xk) to Xi	Floating multiply	5	8,12,13
43ijk	Form mask of jk bits to Xi	Shift	2	8,12,13
44ijk	Floating divide (Xj) by (Xk) to Xi	Divide	20	8,12,13,15
45ijk	Round floating divide (Xj) by (Xk) to Xi	Divide	20	8,12,13,15
460xx-463xx	Pass	-	1	-
47ixk	Population count of (Xk) to Xi	Population count	2	8,12,13
50ijk	Set Ai to (Aj) + K	Increment	15 [8]	2,3,8,12,16, 17,18,19,25
51ijK	Set Ai to (Bj) + K	Increment	15 [8]	2,3,8,12,16, 17,18,19,25
52ijk	Set Ai to (Xj) + K	Increment	15 [8]	2,3,8,12,16, 17,18,19,25
53ijk	Set Ai to (Xj) + (Bk)	Increment	15 [8]	2,3,8,12,16, 17,18,19,25
54ijk	Set Ai to (Aj) + (Bk)	Increment	15 [8]	2,3,8,12,16, 17,18,19,25
55ijk	Set Ai to (Aj) - (Bk)	Increment	15 [8]	2,3,8,12,16, 17,18,19,25
56ijk	Set Ai to (Bj) - (Bk)	Increment	15	2,3,8,12,16, 17,18,19,25
57ijk	Set Ai to (Bj) - (Bk)	Increment	15 [8]	2,3,8,12,16, 17,18,19,25

†The nonbracketed execution times apply to model 865. Execution times in brackets [ ] apply to model 875 (single or dual CP); in this case, notes 18 and 19 do not apply. A single execution time for an instruction (25 ns) applies to both models (single or dual CP).

Table 4-2. CP Instruction Timing (Sheet 4 of 5)

Instruction Code	Description	Functional Unit	Execution Time (CP)†	Timing Notes
60ijk	Set Bi to (Aj) + K	Increment	2	8,12,13
61ijk	Set Bi to (Bj) + K	Increment	2	8,12,13
62ijk	Set Bi to (Xj) + K	Increment	2	8,12,13
63ijk	Set Bi to (Xj) + (Bk)	Increment	2	8,12,13
64ijk	Set Bi to (Aj) + (Bk)	Increment	2	8,12,13
65ijk	Set Bi to (Aj) - (Bk)	Increment	2	8,12,13
66ijk I≠0	Set Bi to (Bj) - (Bk)	Increment	2	8,12,13
67ijk I≠0	Set Bi to (Bj) - (Bk)	Increment	2	8,12,13
660jk	Read CM to XJ	Increment	15 [8]	2,3,8,12,19, 25
670jk	Write CM with XJ	Increment	2	2,3,8,12,18
70ijk	Set Xi to (Aj) + K	Increment	2	8,12,13
71ijk	Set Xi to (Bj) + K	Increment	2	8,12,13
72ijk	Set Xi to (Xj) + K	Increment	2	8,12,13
73ijk	Set Xi to (Xj) + (Bk)	Increment	2	8,12,13
74ijk	Set Xi to (Aj) + (Bk)	Increment	2	8,12,13
75ijk	Set Xi to (Aj) - (Bk)	Increment	2	8,12,13
76ijk	Set Xi to (Bj) + (Bk)	Increment	2	8,12,13
77ijk	Set Xi to (Bj) - (Bk)	Increment	2	8,12,13

† The nonbracketed execution times apply to model 865. Execution times in brackets [ ] apply to model 875 (single or dual CP); in this case, notes 18 and 19 do not apply. A single execution time for an instruction (25 ns) applies to both models (single or dual CP).

Table 4-2. CP Instruction Timing (Sheet 5 of 5)

Timing Notes:

The times listed in the Execution Time column assume that no conflicts occur. However, a delay results unless all of the conditions listed in the Timing Notes column exist for that particular instruction. The Timing Notes column is keyed to the following conditions.

1. All previous instruction fetches are complete.

2. No CM conflicts or SAS backup caused by CM conflicts exist.

3. No IOU request occurs.

4. All operating registers are free.

5. EEM is not busy.

6. All EEM banks have completed previously initiated read/write cycles.

7. Does not include startup time/end time.

8. The requested operating register(s) is free.

9. Assumes no EEM partial records.

10. If the address is in the IAS, the execution time is 3 clock periods.

11. If the branch conditions are not met, the execution time is 2 clock periods.

12. The requested destination register(s) input data path is free during the required clock period.

13. No further delay is possible after the instruction is issued to the functional unit.

14. The multiply unit is free.

15. The divide unit is free.

16. If no storage reference is required [i=0], or if a write storage reference is selected [i=6 or 7], the execution time is 2 clock periods.

17. After the instruction has issued to the increment unit, no further delays are possible in the delivery of data to the Ai register. However, CM conflicts may delay the resulting storage reference.

18. If memory enable is not present when the address is gated to SAS, one additional clock period is required (model 865 only).

19. If optional CP-1 is installed, 2 additional clock periods are required (model 865 only).

20. Assumes no shared usage of the EEM coupler by the opposite CP on dual CP systems.

21. Assumes no bank conflicts in EEM caused by other equipment attached to EEM.

22. Assumes no exchange jump requests being processed by the opposite CP on dual CP systems.

23. All CM banks must be not busy and SAS in both CP-0 and CP-1 must be empty before the exchange sequence can start.

24. On dual CP systems, an exchange jump request that will cause the monitor play to set will wait until the monitor flag for the opposite CP to clear.

25. Memory requests in SAS attempting to access memory are delayed for 2 clock periods if divide time 8 is present (a 1 clock period delay is seen at divide time 15 with model 875).

PP INSTRUCTIONS

PP INSTRUCTION FORMATS

Figure 4-2 shows PP instruction formats. PP instructions are 12 or 24 bits long. In instruction descriptions, the operation code is given either by two or three octal digits. The third digit, when used, indicates the state of the s-bit (zero or one) in I/O instruction (refer to table 4-3).

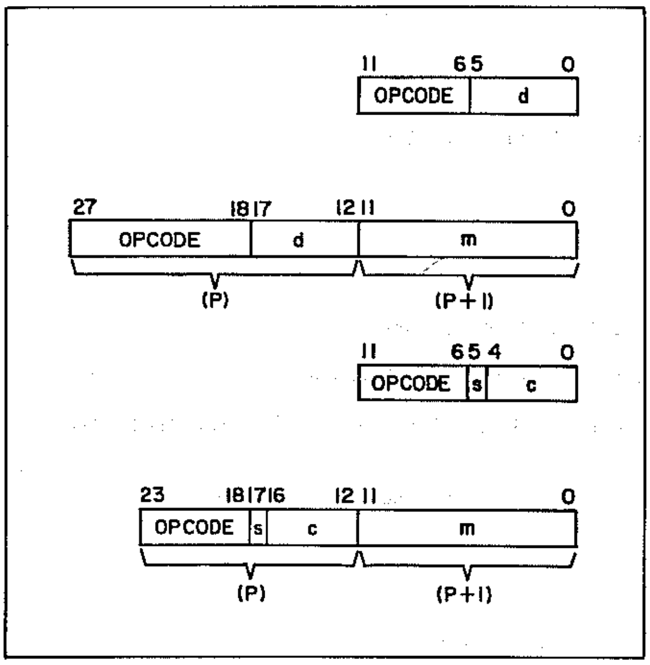


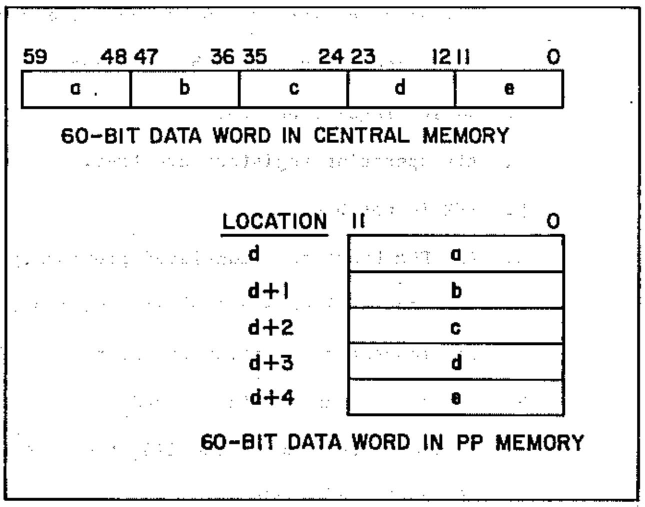
Figure 4-2. PP Instruction Formats

Table 4-3. PP Nomenclature

Term	Description
f	6-bit operation code.
d	6-bit operand or address.
m	12-bit operand or address.
fd	12-bit instruction code.
dm	18-bit operand.
x	Unused register.
A	Arithmetic register.
P	Program register.
Q	Q register.
( )	Content of a register or location.
(( ))	Indirect addressing which specifies the content of a location whose address is specified by a designator inside the parentheses.

PP DATA FORMAT

Figure 4-3 shows PP data format and how 12-bit data is packed into 60-bit CM words or unpacked from 60-bit CM words.

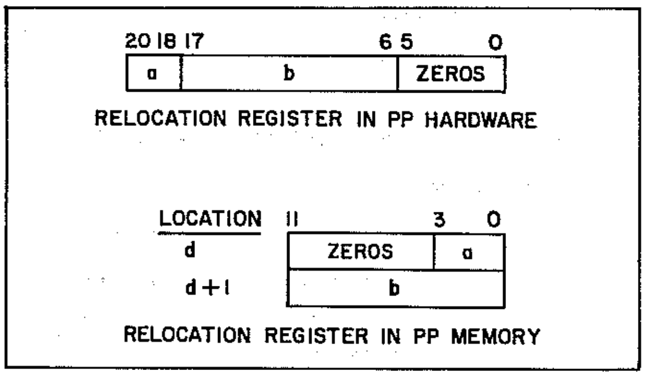


59 48 47 36 35 24 23 12 11 0
a
b
c
d
e
60-BIT DATA WORD IN CENTRAL MEMORY
LOCATION
11 0
d
d+1
d+2
d+3
d+4
a
b
c
d
e
60-BIT DATA WORD IN PP MEMORY

Figure 4-3. PP Data Format

PP RELOCATION (R) REGISTER FORMAT

Figure 4-4 shows PP relocation (R) register format. This register is loaded-from/stored-into PP memory by instructions 24 and 25 (load/store R register).



20 18 17 6 5 0
a
b
ZEROS
RELOCATION REGISTER IN PP HARDWARE
LOCATION
11 3 0
d
d+1
ZEROS
a
b
RELOCATION REGISTER IN PP MEMORY

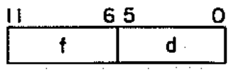
Figure 4-4. PP Relocation (R) Register Format

PP INSTRUCTION DESCRIPTIONS

The PP instructions have separate descriptions. Shaded areas, like those in the 260x and 261x instruction formats, indicate unused bits. The unused bits are ignored by the PPs.

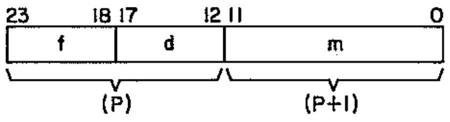
Timing information follows the instructions.

00xx Pass PSN



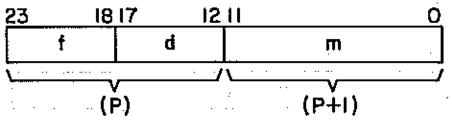
This instruction specifies that no operation is to be performed. The instruction provides a means of padding out a program.

01dm Long jump to m + (d) LJM m,d



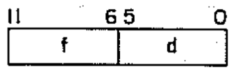
This instruction jumps to the address given by m plus the content of location d. If d equals zero, m is not modified.

02dm Return Jump to m + (d) RJM m,d



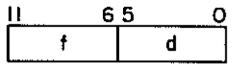
This instruction jumps to the address given by m plus the content of location d. If d equals zero, m is not modified. The current program address (P) plus 2 is stored at the jump address. The next instruction starts at the jump address plus 1. The subprogram exits with a long jump or normal sequencing to the jump address minus 1, which in turn contains a long jump, 0100. This returns the original program address plus 2 to the P register.

03d Unconditional Jump d UJN d



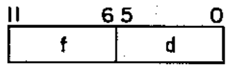
This instruction provides an unconditional jump to any address up to 31 (decimal) locations forward or backward from the current program address. The value of d is added to the current program address. If d is positive (01 through 37), 0001 through 0037 is added, and the jump is forward. If d is negative (40 through 76), 7740 through 7776 is added, and the jump is backward. When d equals 00 or 77, the PP hangs; a deadstart is required to restart the PP.

04d Zero Jump d ZJN d



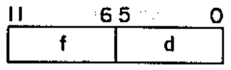
This instruction provides a conditional jump to any address up to 31 (decimal) locations forward or backward from the current program address. If the content of the A register is zero, the jump is taken. If the content of A is nonzero, the next instruction executes from P plus 1. Negative zero (777777) is treated as nonzero. For interpretation of d, refer to the 03 instruction.

05d Nonzero Jump d NJN d



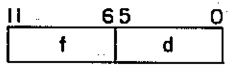
This instruction provides a conditional jump to any address up to 31 (decimal) locations forward or backward from the current program address. If the content of the A register is nonzero, the jump is taken. If the content of A is zero, the next instruction executes from P plus 1. Negative zero (777777) is treated as nonzero. For interpretation of d, refer to the 03 instruction.

06d Plus Jump d PJN d



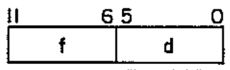
This instruction provides a conditional jump to any address up to 31 (decimal) locations forward or backward from the current program address. If the sign of the A register is positive, the jump is taken. If the sign of A is negative, the next instruction executes from P plus 1. Positive zero is treated as a positive quantity. Negative zero is treated as a negative quantity. For interpretation of d, refer to the 03 instruction.

07d Minus Jump d MJN d



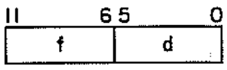
This instruction provides a conditional jump to any address up to 31 (decimal) locations forward or backward from the current program address. If the content of the A register is negative, the jump is taken. If the content of A is positive, the next instruction executes from P plus 1. Positive zero is treated as a positive quantity. Negative zero is treated as a negative quantity. For interpretation of d, refer to the 03 instruction.

10d Shift d SHN d



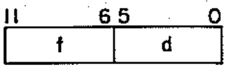
This instruction shifts the content of the A register right or left d places. If d is positive (00 through 37), the shift is left circular. If d is negative (40 through 77), the shift is right (end-off with no sign extension). Thus, d equal to 06 requires a left shift of six places; d equal to 71 requires a right shift of six places.

11d   Logical Difference d   LMN d



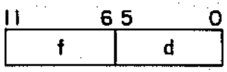
This instruction forms the bit-by-bit logical difference of d and the lower 6 bits of A in the register in A. This is equivalent to complementing individual bits of A that correspond to bits of d that are one. The upper 12 bits of A are not altered.

12d   Logical Product d   LPN d



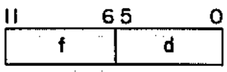
This instruction forms the bit-by-bit logical product of d and the lower 6 bits of the A register and leaves this quantity in the lower 6 bits of A. The upper 12 bits of A are zero.

13d   Selective Clear d   SCN d



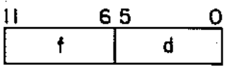
This instruction clears any of the lower 6 bits of the A register where corresponding bits of d are one. The upper 12 bits of A are not altered.

14d   Load d   LDN d



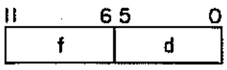
This instruction clears the A register and loads d. The upper 12 bits of A are zero.

15d   Load Complement d   LCN d



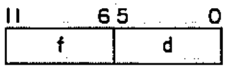
This instruction clears the A register and loads the complement of d. The upper 12 bits of A are one.

16d   Add d   ADN d



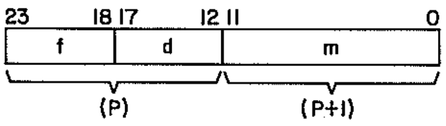
This instruction adds d (treated as a 6-bit positive quantity) to the content of the A register.

17d   Subtract d   SBN d



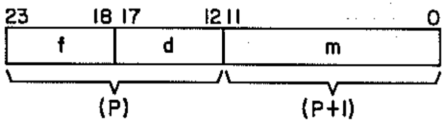
This instruction subtracts d (treated as a 6-bit positive quantity) from the content of the A register.

20dm   Load dm   LDC dm



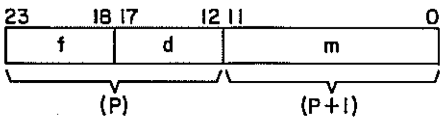
This instruction clears the A register and loads an 18-bit quantity consisting of d as the upper 6 bits and m as the lower 12 bits. The content of the location (P plus 1) which follows the present program address (P) is read to provide m.

21dm   Add dm   ADC dm



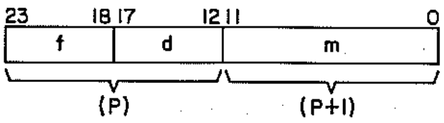
This instruction adds to the A register the 18-bit quantity consisting of d as the upper 6 bits and m as the lower 12 bits . The content of the location (P plus 1) which follows the present program address (P) is read to provide m.

22dm   Logical Product dm   LPC dm



This instruction forms the bit-by-bit logical product of the content of the A register and the 18-bit quantity dm in A. The upper 6 bits of this quantity consist of d, and the lower 12 bits are the content of the location (P plus 1), which follows the present program address (P).

23dm   Logical Difference dm   LMC dm



This instruction forms the bit-by-bit logical difference of the content of the A register and the 18-bit quantity dm in A. This is equivalent to complementing individual bits of A which correspond to bits of dm that are one. The upper 6 bits of the quantity consist of d, and the lower 12 bits are the content of the location (P plus 1), which follows the present program address (P).

24d Load R Register LRD d

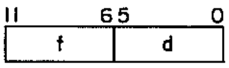


Figure 4-4 shows R register format. If d is not equal to zero, this information loads the R register from PP memory locations d (rightmost 4 bits) and d plus 1 (next 12 bits). The lowest 6 bits of R are always zero and are not entered by this instruction.

25d Store R Register SRD d

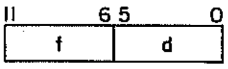
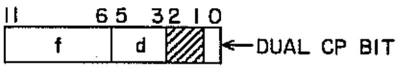


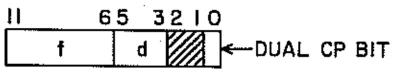
Figure 4-4 shows R register format. If d is not equal to zero, this information stores the R register from PP memory locations d (rightmost 4 bits) and d plus 1 (next 12 bits). The lowest 6 bits of R are always zero and are not stored by this instruction.

2600 Exchange Jump EXN



This instruction causes an unconditional exchange jump in the CP, leaving the CP monitor flag unaltered. The new exchange package begins at central memory location R plus A when the leftmost bit in A is set. When this bit is clear, A specifies the address. The PP waits until the exchange has been completed before proceeding with the next instruction. With dual CP systems, the lowest order bit of the instruction specifies which of the two CPs the exchange jump interrupts. With single CP systems this bit is not interpreted. Exchange jump addresses are truncated to 18 bits by the CP such that all exchange operations take place in the lower 262K words of central memory.

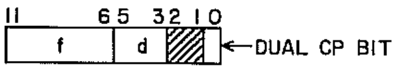
2610 Monitor Exchange Jump MXN



This instruction is enabled or disabled by the CEJ/MEJ switch. If the CEJ/MEJ switch is in the disable position, this instruction performs as a 260 instruction. When the switch is in the enable position, this instruction causes a conditional exchange jump of the CP. With dual CP systems, the lowest order bit of the instruction specifies which of the two CPs the exchange jump interrupts. With single CP systems this bit is not interpreted. Exchange jump addresses are truncated to 18 bits by the CP such that all exchange operations take place in the lower 262K words of central memory.

If the CP is in monitor mode, this instruction is a pass. If the CP is in job mode, it causes an exchange jump in the CP, switching the CP to monitor mode (MF equals 1). The new exchange package begins at central memory location R plus A when the leftmost bit of A is set. When this bit is clear, A specifies the address. The PP waits until the exchange has completed before proceeding with the next instruction.

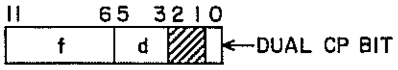
2620 Monitor Exchange Jump to MA MAN



This instruction is enabled or disabled by the CEJ/MEJ switch. If the CEJ/MEJ switch is in the disable position, this instruction performs as a 260 instruction. When the switch is in the enable position, this instruction causes a conditional exchange jump of the CP. With dual CP systems, the lowest order bit of the instruction specifies which of the two CPs the exchange jump interrupts. With single CP systems this bit is not interpretted. Exchange jump addresses are truncated to 18 bits by the CP such that all exchange operations take place in the lower 262K words of central memory.

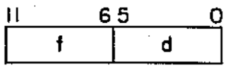
If the CP is in monitor mode, this instruction is a pass. If the CP is in job mode, it causes an exchange jump in the CP, switching the CP to monitor mode (MF equals 1). The new exchange package begins at the absolute address given in the MA field of the outgoing exchange package. The PP waits until the exchange has completed before proceeding with the next instruction.

27d Read Program Address KPT d



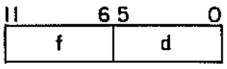
This instruction transfers the content of the CP P register to the PP A register; this allows the PP to determine whether the CP is running. For information on the dual-CP bit, refer to the 2600 instruction.

30d Load (d) LDD d



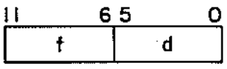
This instruction clears the A register and loads the content at location d. The upper 6 bits of A are zero.

31d Add (d) ADD d



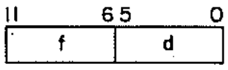
This instruction adds the content at location d (treated as a 12-bit positive quantity) to the A register.

32d Subtract (d) SBD d



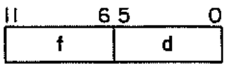
This instruction subtracts the content at location d (treated as a 12-bit positive quantity) from the A register.

33d Logical Difference (d) LMD d



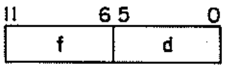
This instruction forms in the A register the bit-by-bit logical difference of the lower 12 bits of the A register and the content at location d. This is equivalent to complementing individual bits of A which correspond to bits in location d that are ones. The upper 6 bits are not altered.

34d Store (d) STD d



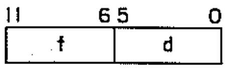
This instruction stores the lower 12 bits of the A register at location d.

35d Replace Add (d) RAD d



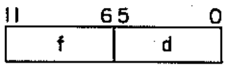
This instruction adds the quantity at location d to the content of the A register and stores the lower 12 bits of the result at location d. The result remains in A at the end of the operation and the original content of A is destroyed.

36d Replace Add One (d) AOD d



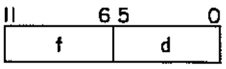
This instruction replaces the quantity at location d with its original value plus 1. The result remains in the A register at the end of the operation, and the original content of A is destroyed.

37d Replace Subtract One (d) SOD d



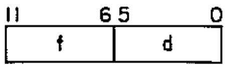
This instruction replaces the quantity at location d with its original value minus 1. The result remains in the A register at the end of the operation, and the original content of A is destroyed.

40d Load ((d)) LDI d



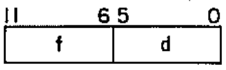
This instruction clears the A register and loads a 12-bit quantity that is obtained by indirect addressing. The upper 6 bits of A are zero. Location d is read from PPM, and the word read is used as the operand address.

41d Add ((d)) ADI d



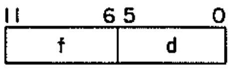
This instruction adds to the content of the A register a 12-bit operand (treated as a positive quantity) obtained by indirect addressing. Location d is read from PPM, and the word read is used as the operand address.

42d Subtract ((d)) SBI d



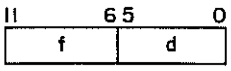
This instruction subtracts from the A register a 12-bit operand (treated as a positive quantity) obtained by indirect addressing. Location d is read from PPM, and the word read is used as the operand address.

43d   Logical Difference ((d))   LMI d



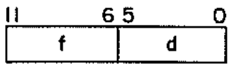
This instruction forms in the A register the bit-by-bit logical difference of the lower 12 bits of the A register and the 12-bit operand read by indirect addressing. Location d is read from PPM, and the word read is used as the operand address. The upper 6 bits of A are not altered.

44d   Store ((d))   STI d



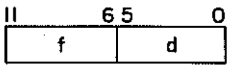
This instruction stores the lower 12 bits of the A register at the location specified by the content of location d.

45d   Replace Add ((d))   RAI d



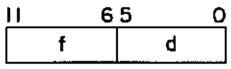
This instruction adds the operand, which is obtained from the location specified by the content at location d, to the content of the A register. The lower 12 bits of the sum replace the original operand. The result remains in A at the end of the operation.

46d   Replace Add One ((d))   AOI d



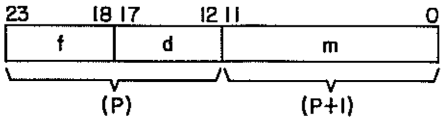
This instruction replaces the operand, which is obtained from the location specified by the content at location d, by its original value plus 1. The result remains in the A register at the end of the operation, and the original content of A is destroyed.

47d   Replace Subtract One ((d))   SOI d



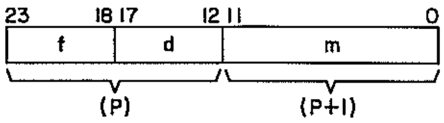
This instruction replaces the operand, which is obtained from the location specified by the content at location d, by its original value minus 1. The result remains in the A register at the end of the operation, and the original content of A is destroyed.

50dm   Load (m + (d))   LDM m,d



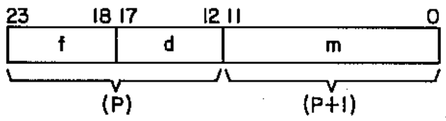
This instruction clears the A register and loads a 12-bit quantity. The upper 6 bits of A are zeros. The 12-bit operand is obtained by indexed direct addressing. The quantity m, read from PPM location P plus 1, serves as the base operand address to which the content of d is added. If d equals 0, the operand address is m, but if d is not equal to 0, m plus the content in d is the operand address. Thus, location d may be used as an index quantity to modify operand addresses.

51dm   Add (m + (d))   ADM m,d



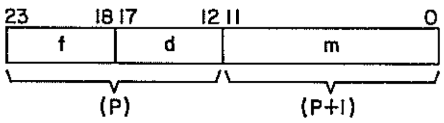
This instruction adds the 12-bit operand (treated as a positive quantity) read by indexed direct addressing (refer to the 50 instruction) to the A register.

52dm   Subtract (m + (d))   SBM m,d



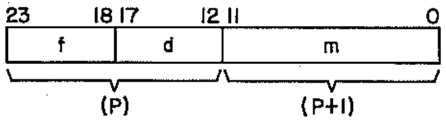
This instruction subtracts the 12-bit operand (treated as a positive quantity) read by indexed direct addressing (refer to the 50 instruction) from the A register.

53dm   Logical Difference (m + (d))   LMM m,d



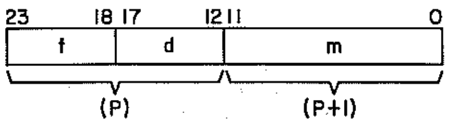
This instruction forms the bit-by-bit logical difference of the lower 12 bits of the A register and a 12-bit operand obtained by indexed direct addressing (refer to the 50 instruction) in A. The upper 6 bits of A are not altered.

54dm   Store (m + (d))   STM m,d



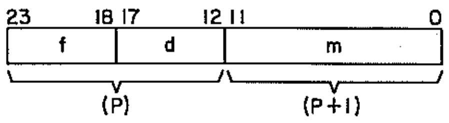
This instruction stores the lower 12 bits of the A register in the location determined by indexed direct addressing (refer to the 50 instruction).

55dm Replace Add (m + (d)) RAM m,d



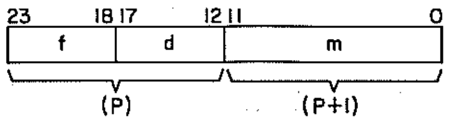
This instruction adds the operand, which is obtained from the location determined by indexed direct addressing (refer to the 50 instruction), to the A register. The lower 12 bits of the sum replace the original operand in PPM. The result remains in A at the end of the operation, and the original content of A is destroyed.

56dm Replace Add One (m + (d)) AOM m,d



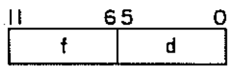
This instruction replaces the operand, which is obtained from the location determined by indexed direct addressing (refer to the 50 instruction), by its original value plus 1. The result remains in the A register at the end of the operation, and the original content of A is destroyed.

57dm Replace Subtract One (m + (d)) SOM m,d



This instruction replaces the operand, which is obtained from the location determined by indexed direct addressing (refer to the 50 instruction), by its original value minus 1. The result remains in the A register at the end of the operation, and the original content of A is destroyed.

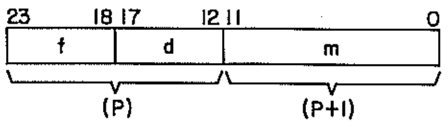
60d Central Read from (A) to d CRD d



This instruction dissassembles one 60-bit word from central memory into five 12-bit words and stores these in five consecutive PP memory locations, beginning with the leftmost 12 bits of the 60-bit word. The parameters of the transfer are as follows:

If bit 17 of A is zero, A bits 0 through 16 contain the absolute address of the 60-bit word transferred. If bit 17 of A is one, hardware adds relocation register R to zero-extended A bits 0 through 16 to obtain the absolute address of the 60-bit word transferred. For further information, refer to R Register under Input/Output Unit in section 2. Field d gives the PP location which receives the first 12-bit word transferred. PP memory addressing is cyclic and location 0000 follows location 7777.

61dm Central Read (d) Words from (A) to m CRM d,m



PP location 0000 is used by hardware. This instruction disassembles 60-bit words from central memory into 12-bit words, and places these in consecutive PP memory locations, beginning with the leftmost 12 bits of the first 60-bit word. The parameters of the transfer are as follows:

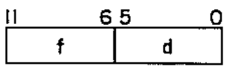
If bit 17 of A is zero, A bits 0 through 16 contain the absolute address of the first 60-bit word transferred. If bit 17 of A is one, hardware adds relocation register R to zero-extended A bits 0 through 16 to obtain the absolute address of the first 60-bit word transferred. For further information, refer to R Register under Input/Output Unit in section 2. PP location d must contain the number of 60-bit words transferred. Field m gives the PP location into which the first 12-bit word is placed.

This instruction stores P plus 1 into PP location 0000 before beginning the transfer. After the transfer is completed, the next instruction is taken from one plus whatever address is stored in location 0000. If the transfer overwrites location 0000, execution resumes at the location specified by (0000) plus 1. PP memory addressing is cyclic and location 0000 follows location 7777.

Register A is incremented by one after each 60-bit word is read from central memory. If the incrementing changes A bit 17, the central memory addressing is switched between direct address and relocation address modes. After the transfer is completed, the A register contains the address of the last word transferred plus one (direct addressing) or the same address less the contents of the relocation address register (relocation addressing), except as follows:

If the last word transferred is from a relative address 377776_8 and relocation is in effect, then the A register is cleared, and the value returned in A may not point to the last word transferred plus one.

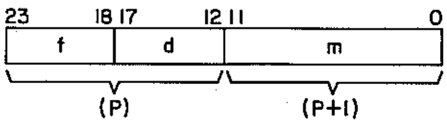
62d Central Write to (A) from d CWD d



This instruction assembles five 12-bit words from consecutive PP memory locations into one 60-bit word and stores the 60-bit word in central memory. The first 12-bit word is stored in the leftmost 12 bits of the 60-bit word. The parameters of the transfer are as follows:

If bit 17 of A is zero, A bits 0 through 16 contain the absolute address of the 60-bit word stored. If bit 17 of A is one, hardware adds relocation register R to zero-extended A bits 0 through 16 to obtain the absolute address of the 60-bit word stored. Field d gives the PP location of the first 12-bit word transferred. For further information, refer to R Register under Input/Output Unit in section 2. PP memory addressing is cyclic and location 0000 follows location 7777. The transfer is subject to the CM bounds test.

63dm Central Write (d) Words to (A) from m CWM m,d



PP location 0000 is used by hardware. This instruction assembles 12-bit words from consecutive PP memory locations into 60-bit words and stores these in central memory. The first 12-bit word is stored in the left-most 12 bits of the 60-bit word. The parameters of the transfer are as follows:

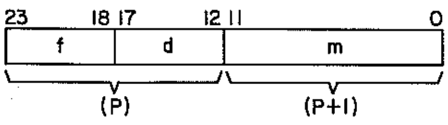
If bit 17 of A is zero, A bits 0 through 16 contain the absolute address of the first 60-bit word transferred. If bit 17 of A is one, hardware adds relocation register R to zero-extended A bits 0 through 16 to obtain the absolute address of the first 60-bit word transferred. PP location d must contain the number of 60-bit words transferred. Field m gives the PP location from where the first 12-bit word is obtained.

This instruction stores its own address plus two into PP location 0000 before beginning the transfer. Memory address is cyclic and location 0000 follows location 7777. The transfer is subject to the CM bounds test.

The A register is incremented by one after each 60-bit word is written into central memory. If the incrementing of A changes A bit 17, the central memory addressing is switched between direct address and relocation address modes. Refer to Central Memory Addressing by PPs, section 5. After the transfer is completed, the A register contains either the address of the last word transferred plus one (direct addressing), or the same address less the contents of the relocation address register (relocation addressing), except as follows:

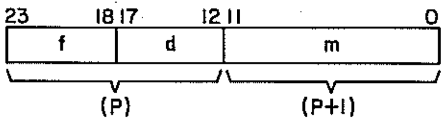
If the last word transferred is from a relative address 377776$_8$ and relocation is in effect, then the A register is cleared and the value returned in A may not point to the last word transferred plus one.

64dm Jump to m if Channel d Active AJM



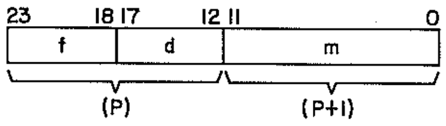
This 24-bit instruction provides a conditional jump to a new address specified by m. The jump is taken if the channel specified by d is active. The next instruction is at P plus 2 if the channel is inactive.

65dm Jump to m if Channel d Output Word Flag Not Set IJM



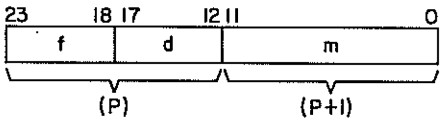
This conditional branch instruction continues the current program sequence or jumps to a new program sequence, depending upon the condition of the channel d output word flag. A new program sequence initiates beginning at address m if the channel d output word flag is not set. The current program sequence continues if the flag is set.

66dm Jump to m if Channel d Output Record Flag Set FJM



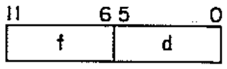
This conditional branch instruction continues the current program sequence or jumps to a new program sequence, depending upon the condition of the channel d output record flag. A new program sequence initiates beginning at address m if the channel d output record flag is set. The current program sequence continues if the flag is not set.

67dm Jump to m if Channel d Output Record Flag Not Set EJM



This conditional branch instruction continues the current program sequence or jumps to a new program sequence, depending upon the condition of the channel d output record flag. A new program sequence initiates beginning at address m if the channel d output record flag is not set. The current program sequence continues if the flag is set.

70d Input to A from Channel d IAN d

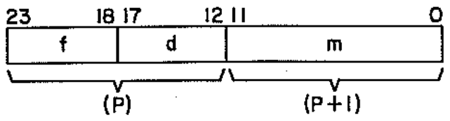


This instruction transfers a word from input channel d to the lower 12 bits of the A register. The upper 6 bits of A are cleared to zero.

NOTE

If bit 5 of d is clear and the channel is inactive, this instruction hangs the PP, waiting for the channel to go active and full, if executed. If bit 5 of d is set and the channel is inactive or is deactivated before a full is received, the instruction exits. The word is not accepted, and the A register clears.

71dm Input A Words to m from Channel d IAM m,d



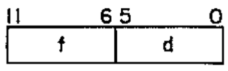
This instruction transfers a block of 12-bit words from input channel d to PPM. The first word goes to the PPM address specified by m. The A register holds the block length. A reduces by one as each word is read. The input operation completes when A equals zero or the data channel becomes inactive. If the operation terminates by the channel becoming inactive, the next storage location in PPM is set to zero. However, the word count is not affected by this empty word. Therefore, A holds the block length minus the number of real data words read.

During this instruction, address 0000 temporarily holds P while m is held in the P register. P advances by one to hold the address for the next word as each word is stored.

NOTE

If this instruction executes when the data channel is inactive, no input operation is accomplished, and the program continues at P plus 2. However, the location specified by m is set to zero.

72d Output from A on Channel d OAN d

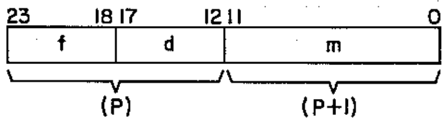


This instruction transfers a word from the A register (lower 12 bits) to output channel d.

NOTE

If bit 5 of d is clear and the channel is inactive, this instruction hangs the PP, waiting for the channel to go active and full, if executed. If bit 5 of d is set and the channel is inactive, the program continues at P plus 1. The word is not transferred.

73dm Output A Words from m on Channel d OAM m,d



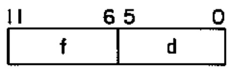
This instruction transfers a block of words from PPM to channel d. The first word is read from the address specified by m. The A register holds the number of words to be sent. A reduces by one as each word is read. The output operation completes when A equals zero or the channel becomes inactive.

During this instruction, address 0000 temporarily holds P while m is held in the P register. P advances by one to give the address of the next word as each word is read from the PPM.

NOTE

If this instruction executes when the data channel is inactive, no output operation is accomplished, and the program continues at P plus 2.

74d Activate Channel d ACN d

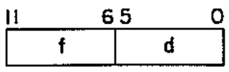


This instruction activates the channel specified by d and sends the active signal on the channel to equipment connected to the channel. Activating a channel, which must precede a 70 through 73 instruction, prepares I/O equipment for the exchange of data.

NOTE

If this instruction executes when the data channel is already active and if bit 5 of d is set, the program continues at P plus 1. Otherwise, activating an already active channel causes the PP to wait until the channel goes inactive. The PP hangs if the channel does not go inactive.

75d Deactivate Channel d DCN d



This instruction deactivates the channel specified by d. As a result, the I/O data transfer stops.

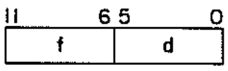
NOTE

If this instruction executes when the data channel is already inactive and bit 5 of d is set, the program continues at P plus 1. The channel remains inactive, and no inactive signal is sent to the I/O equipment. Deactivating an already inactive channel causes the PP to hang until the channel becomes active.

If an output instruction is followed by a disconnect instruction without first establishing that the information has been accepted by the input device (check for channel empty), the last word transmitted may be lost.

Do not deactivate a channel before putting a useful program in the associated PP. PPs other than 0 are hung on an input instruction (71) after deadstart. Deactivating a channel after deadstart causes an exit to the address specified by the content of location 0000 plus 1 and execution of that program. If the channel is deactivated without a valid program in that PP, the PP executes whatever program was left in PPM. Therefore, the PP could run wild.

76d Function A on Channel d FAN d

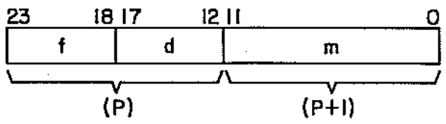


This instruction sends the external function code in the lower 12 bits of the A register on channel d.

NOTE

If this instruction executes with bit 5 of d clear and the channel active, PP execution stops until a deadstart or another PP causes the channel to become inactive. If bit 5 of d is set and the channel is active, the program continues at P plus 1. Neither the function signal nor the function word transmits. The channel remains active, and execution continues.

77dm Function m on Channel d FNC m,d



This instruction sends the external function code specified by m on channel d.

NOTE

If this instruction executes with bit 5 of d clear and the channel active, PP execution stops until a deadstart or another PP causes the channel to become inactive. If bit 5 of d is set and the channel is active, the program continues at P plus 2. Neither the function signal nor the function word transmits. The channel remains active, and execution continues.

INSTRUCTION EXECUTION TIMING

Approximate execution times for the PP instructions are listed in table 4-4. These times are listed with the assumption that no conflicts occur. The numbers in the timing notes column refer to the notes at the end of the table. Execution times are given in 250-nanosecond major cycles.

NOTE

These execution times are approximations only and subject to change without notice. Accurate timings can come only from benchmark tests. Control Data Corporation is not responsible for assumptions made based on the times listed here.

Table 4-4. PP Instruction Timing (Sheet 1 of 3)

Instruction Code	Description	Execution Time in 250-ns Cycles	Timing Notes
00xx	Pass	1	-
01dm	Long jump to m + (d)	2	1
02dm	Return jump to m + (d)	3	2
03d	Unconditional jump d	1	-
04d	Zero jump d	1	-
05d	Nonzero jump d	1	-
06d	Plus jump d	1	-
07d	Minus jump d	1	-
10d	Shift d	1	-
11d	Logical difference d	1	-
12d	Logical product d	1	-
13d	Selective clear d	1	-
14d	Load d	1	-
15d	Load complement d	1	-
16d	Add d	1	-
17d	Subtract d	1	-
20dm	Load dm	2	-
21dm	Add dm	2	-
22dm	Logical product dm	2	-
23dm	Logical difference dm	2	-
24d	Load R register from (d) and (d) + 1	3	-
2400	Pass	1	-
25d	Store R register at (d) and (d) + 1	3	-
2500	Pass	1	-
260x	Exchange jump	1	3
261x	Monitor exchange jump	2	4
262x	Monitor exchange jump to MA	2	4
27d	Read program address	1	-
30d	Load (d)	2	-
31d	Add (d)	2	-

Timing Notes:

1. 3 if d = 0.
2. 4 if d = 1.
3. Assumes no CMC conflict.
4. No assembly-disassembly unit (ADU) conflicts and no outstanding exchange jump request in the ADU.

Table 4-4. PP Instruction Timing (Sheet 2 of 3)

Instruction Code	Description	Execution Time in 250-ns Cycles	Timing Notes
32d	Subtract (d)	2	-
33d	Logical difference (d)	2	-
34d	Store (d)	2	-
35d	Replace add (d)	3	-
36d	Replace add one (d)	3	-
37d	Replace subtract one (d)	3	-
40d	Load ((d))	3	-
41d	Add ((d))	3	-
42d	Subtract ((d))	3	-
43d	Logical difference ((d))	3	-
44d	Store ((d))	3	-
45d	Replace add ((d))	4	-
46d	Replace add one ((d))	4	-
47d	Replace subtract one ((d))	4	-
50dm	Load (m + (d))	3	5
51dm	Add (m + (d))	4	-
52dm	Subtract (m + (d))	3	5
53dm	Logical difference (m + (d))	3	5
54dm	Store (m + (d))	3	5
55dm	Replace add (m + (d))	4	6
56dm	Replace add one (m + (d))	4	6
57dm	Replace subtract one (m + (d))	4	6
60d	Central read from (A) to d	8	7
61dm	Central read (d) words from (A) to m	6	3,7,8
62d	Central write to (A) from d	6	7
63dm	Central write (d) words to (A) from m	6	7,8
64dm	Jump to m if channel d active	2	-
65dm	Jump to m if channel d output word flag not set	2	-
66dm	Jump to m if channel d output record flag set	2	-
67dm	Jump to m if channel d output record flag not set	2	-

Timing Notes:

3. Assumes no CMC conflict.
5. 4 if d = 0.
6. 5 if d = 0.
7. Add one major cycle to this instruction. When it is preceded by a 10d (Shift d) instruction.
8. 5 major cycles per CM word.

Table 4-4. PP Instruction Timing (Sheet 3 of 3)

Instruction Code	Description	Execution Time in 250-ns Cycles	Timing Notes
70d	Input to A from channel d	2	-
71dm	Input A words to m from channel d	5	8
72d	Output from A on channel d	2	-
73dm	Output (A) words from m on channel d	5	8
74d	Activate channel d	2	-
75d	Deactivate channel d	2	-
76d	Function A on channel d	2	-
77dm	Function m on channel d	2	-

Timing Notes:

8. 5 major cycles per CM word.

PROGRAMMING INFORMATION

This section contains special programming information about the CP, CM, PPs, display station, real-time clock and maintenance register.

CP PROGRAMMING

EXCHANGE JUMP

The CP uses an exchange jump operation to switch from job mode to monitor mode and back again. The execution of an exchange jump permits the CP to send pertinent information from the operating and control registers to CM and permits CM to send new information to the same registers. The information that flows from and into the operating and control registers during an exchange jump is called an exchange package.

Exchange jump packages may exist in either standard (figure 5-1) or expanded addressing mode (figure 5-2), depending on the state of the expanded addressing select flag. Refer to Extended Memory Addressing Modes later in this section. An exchange jump instruction is 013 in the CP and 2600, 2610, or 2620 in the IOU. The instruction starts or interrupts the CP and provides CM with the first address of a 16-word exchange package. For the CP-initiated exchange, the address is K plus the content of the Bj register or the monitor address. For the IOU-initiated exchange, the address is the content of A plus R (if bit 17 in the A register is set), or the content of the monitor address (MA) register. In the former case, the A-plus-R address is truncated to 18 bits at CMC. The IOU also has the monitor exchange jump to MA (2620) instruction in which the content of MA is used for the exchange address.

The exchange package provides the following information for a program to be executed.

Program address (P) - 18 bits.

Reference address for CM (RAC) - 21 bits.

Field length of program for CM (FLC) - 21 bits.

Exit mode (EM) - 6 bits.

Flag register - 6 bits.

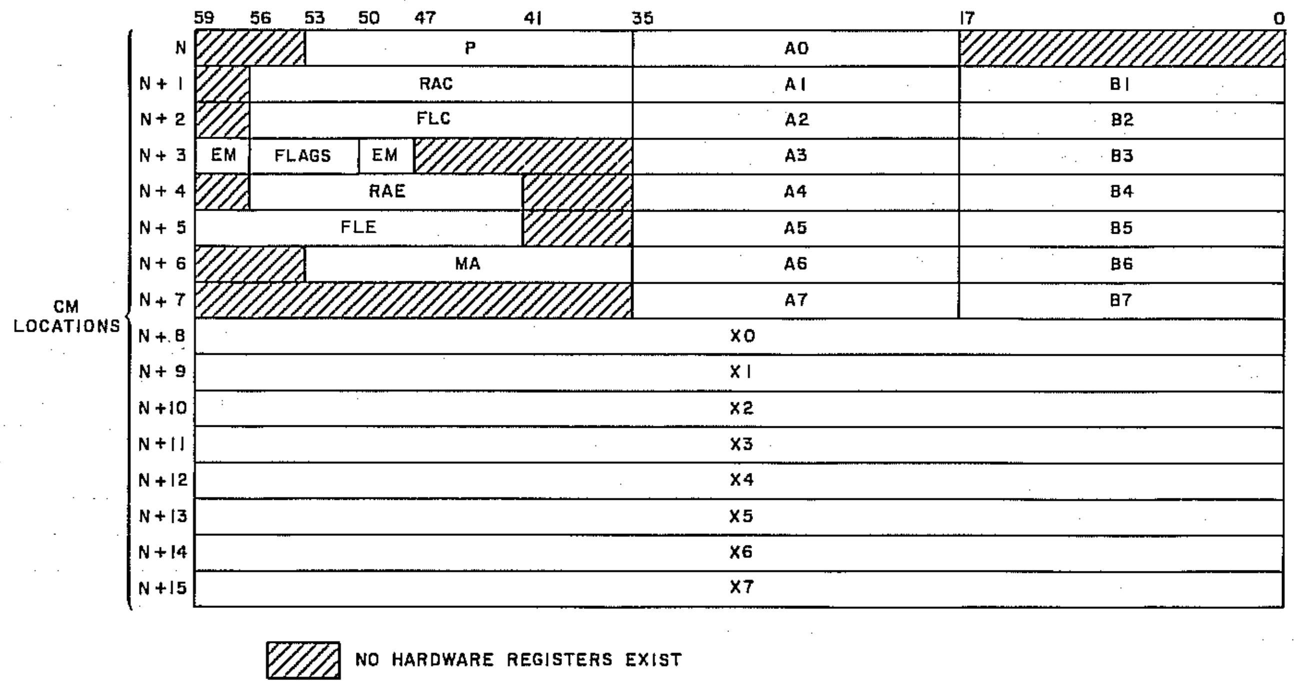


Figure 5-1. Exchange Package - Standard Addressing Mode

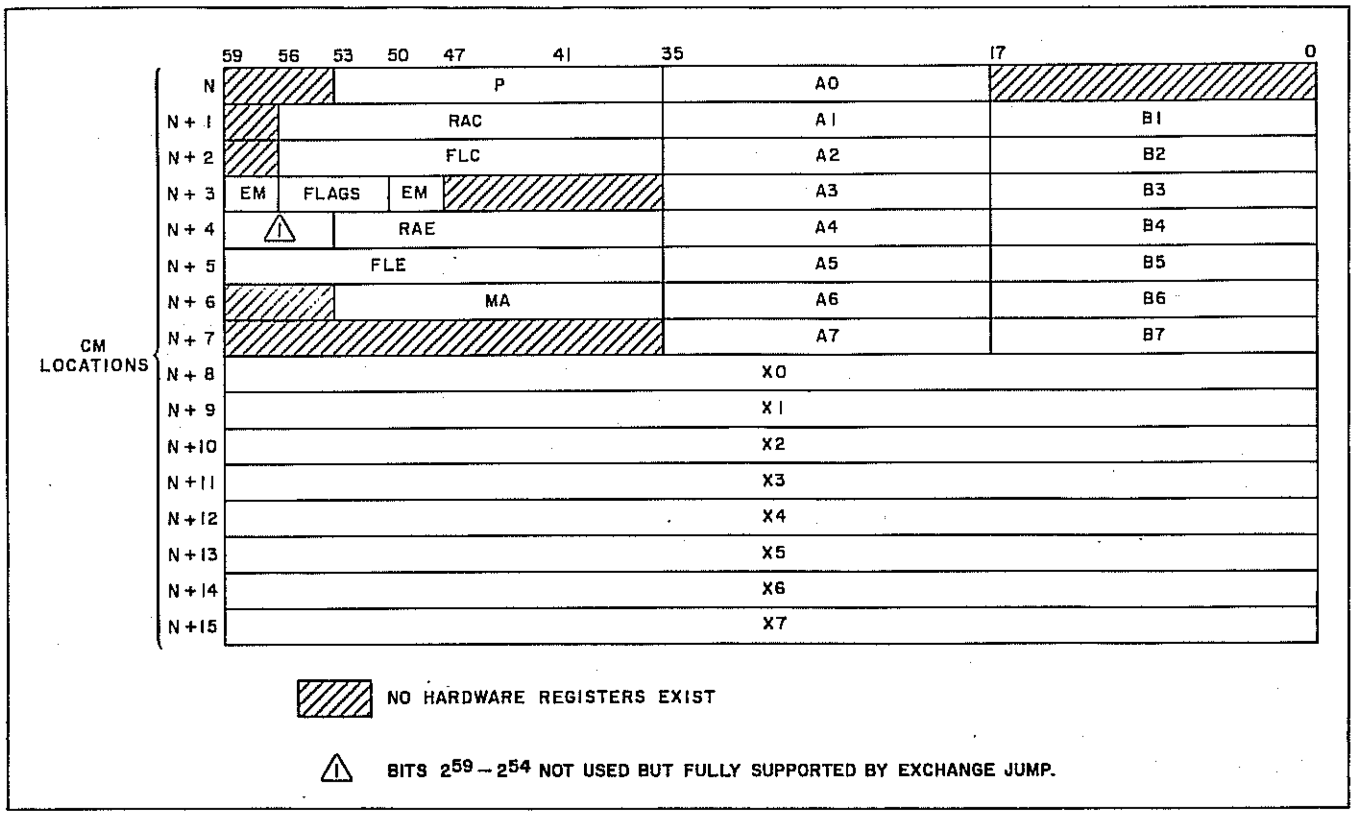


Figure 5-2. Exchange Package – Expanded Addressing Mode

Reference address (RAE) for extended memory – 21 or 24 bits, depending on selection of standard or expanded addressing mode (lower 6 bits are assumed to be zeros).

Field length of block transfer (FLE) for extended memory – 21 or 24 bits, depending on selection of standard or expanded addressing mode (lower 6 bits are assumed to be zeros).

Monitor address (MA) – 18 bits.

Initial contents of eight A registers – 18 bits.

Initial contents of eight X registers – 60 bits.

Initial contents of B1 through B7 registers (B0 contains constant 0) – 18 bits.

The time that a particular exchange package resides in the CP hardware registers is the execution interval. The execution interval begins with an exchange jump that swaps the exchange package information in CM with the information contained in the CP registers. The execution interval ends with the next exchange jump. A hardware flag called a monitor flag (MF) indicates the type of program the CP is executing.

When the flag is set, the CP is in noninterruptible monitor mode. When the flag is clear, the CP is in an interruptible program job mode. A master clear (deadstart) clears the monitor flag.

FLOATING-POINT ARITHMETIC

Format

Floating-point arithmetic expresses a number in the form kB^n.

k	Coefficient.
B	Base number.
n	Exponent or power to which the base number is raised.

B is assumed to be 2 for binary-coded quantities. In the 60-bit floating-point format (figure 5-3), the binary point is considered to be to the right of the coefficient. The lower 48 bits express the integer coefficient, which is the equivalent of 15 decimal digits. The sign of the coefficient is separated from the rest of the coefficient and appears in the highest-order bit of the packed word. Negative numbers are represented in one's complement notation. The exponent is biased by complementing the exponent sign bit.

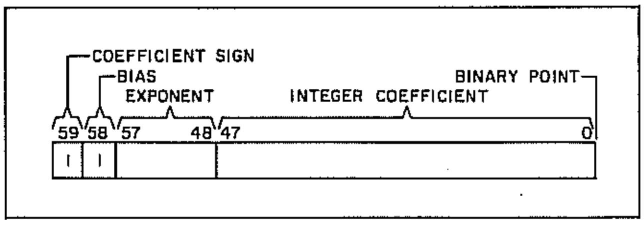


Figure 5-3. Floating-Point Format

Table 5-1 summarizes the configurations of bits 58 and 59 and the implications regarding signs of the possible combinations.

Table 5-1. Bits 58 and 59 Configurations

Bit 59	Bit 58	Coefficient Sign	Exponent Sign
0	1	Positive	Positive
0	0	Positive	Negative
1	0	Negative	Positive
1	1	Negative	Negative

Packing

Packing refers to the conversion of numbers in the form kB^n to floating-point format. A shortcut method of packing exponents can be derived by considering the representation of negative and positive zero exponents. Assuming a positive coefficient, zero exponents are packed as follows:

Positive zero exponent	2000x,...,x
Negative zero exponent	1777x,...,x

Since positive exponents are expressed in true form, begin with a bias of 2000 (positive zero) and add the magnitude of the exponent. The range of positive exponents is 0000 through 1777. In packed form, the range is 2000 through 3777.

When the coefficient is negative, the packed positive exponent is complemented to become 5777 through 4000.

Negative exponents are expressed in complement form by beginning with a bias of 1777 (negative zero) and then subtracting the magnitude of the exponent. The range of negative exponents is negative 0000 through negative 1777. In packed form, the range is 1777 through 0000.

When the coefficient is negative, the packed negative exponent is complemented to become 6000 through 7777.

Examples of packed and unpacked floating-point numbers are shown in octal notation to illustrate the packing process. Examples 1 and 2 are different forms of the integer positive 1. Example 3 is positive 100 (decimal), and example 4 is negative 100 (decimal). Examples 5 and 6 are large and small positive numbers. The unpacked values are shown as they might appear in the X and B registers prior to a pack operation.

The packed negative zero exponent is not used for normal operation. Instead, 1777 is used to indicate the special error condition of indefinite.

1.	Unpacked coefficient	0000	0000	0000	0000	0001
	Unpacked exponent	00	0000			
	Packed format	2000	0000	0000	0000	0001
2.	Unpacked coefficient	0000	4000	0000	0000	0000
	Unpacked exponent	77	7720			
	Packed format	1720	4000	0000	0000	0000
3.	Unpacked coefficient	0000	6200	0000	0000	0000
	Unpacked exponent	77	7726			
	Packed format	1726	6200	0000	0000	0000
4.	Unpacked coefficient	7777	1577	7777	7777	7777
	Unpacked exponent	77	7726			
	Packed format	6051	1577	7777	7777	7777
5.	Unpacked coefficient	0000	4771	3000	0044	7021
	Unpacked exponent	00	1363			
	Packed format	3363	4771	3000	0044	7021
6.	Unpacked coefficient	0000	6301	0277	4315	6033
	Unpacked exponent	77	6210			
	Packed format	0210	6301	0277	4315	6033

Overflow

Overflow of the floating-point range is indicated by an exponent value of positive 1777 (3777 or 4000 in packed form). This is the largest exponent value that can be represented in the floating-point format. This exponent value may result from the calculation in which this exponent value, together with the computed coefficient value, is a correct representation of the result. This situation is called a partial overflow. However, further computation using this result generates an overflow.

A complete overflow occurs whenever a result requires an exponent larger than positive 1777. In this case, a complete overflow value results. This result has a positive 1777 exponent and a zero coefficient. The sign of the coefficient is the same as that which generates if the result had not overflowed the floating-point range.

Underflow

Underflow of the floating-point range is indicated by an exponent value of negative 1777 (0000 or 7777 in packed form). This is the smallest exponent value that can be represented in the floating-point format. This exponent value may result from the calculation in which this exponent value, together with the computed coefficient value, is a correct representation of the result. This situation is called a partial underflow. Further computation using this result may be detected as an underflow.

A complete underflow occurs whenever a result requires an exponent smaller than negative 1777. In this case, a complete underflow value results. This result has a negative 1777 exponent and a zero coefficient. The complete underflow indicator is a word of all zeros, and it is the same as a zero word in integer format.

Indefinite

An indefinite result indicator generates whenever the calculation cannot be resolved. An example is division when the divisor is 0 and the dividend is also 0. Another example is multiplication of an overflow number times an underflow number. The indefinite result indicator is a value that cannot occur in normal floating-point calculations. This indicator corresponds to a negative 0 exponent and a 0 coefficient (177770,...,0 in packed form).

Any indefinite indicator used as an operand generates an indefinite result no matter what the other operand value is. Although indefinite indicators always generate with a positive sign, they may occur as operands with a negative sign.

Nonstandard Operands

In summary, the special operand forms in octal are:

Positive overflow (+ ∞)	3777x,...,x
Negative overflow (− ∞)	4000x,...,x
Positive indefinite (+IND)	1777x,...,x
Negative indefinite (-IND)	6000x,...,x
Positive underflow (+0)	0000x,...,x
Negative underflow (-0)	7777x,...,x

Tables 5-2 through 5-5 indicate the resulting forms when various combinations of underflow, overflow, and indefinite forms are used in floating-point operations. The designations W and N are defined as follows:

W	Any word except + ∞ and + IND
N	Any word except + ∞ , + IND, and + 0

Normalized Numbers

A normalized floating-point number has as large a coefficient and as small an exponent as possible. A floating-point number in packed format is normalized if the coefficient sign bit is different from bit 47. This condition indicates that the coefficient has been left shifted until bit 47 contains the most significant bit in the coefficient; therefore, the floating-point number has no leading sign bits in the coefficient. The normalized instructions perform the coefficient shift. The floating-multiply and floating-divide instructions deliver normalized results when provided with normalized operands. The floating-add instructions may deliver unnormalized results even when both operands are normalized. Therefore, it is necessary to perform the normalize operation after each sequence of floating-add or floating-subtract operations if the result is to be kept in a normalized form.

Rounding

Floating-point instructions round the results in single-precision computation. These instructions execute in the same amount of time as the unrounded versions. The operands are modified to accomplish the rounding function. The amount of bias introduced by the rounding operation varies and is affected by the coefficient value in the operands. The descriptions of the round instructions define the effects of rounding in detail.

Double-Precision Results

The floating-point arithmetic instructions generate double-precision results. Use of unrounded instructions allows separate recovery of upper and lower half results with proper exponents. Rounded instructions allow only upper half results to be obtained. Two instructions, one single-precision and one double-precision, are required to retrieve an entire double-precision result.

To add or subtract two floating-point numbers, the coefficient having the smaller exponent enters the upper half of an accumulator and is right shifted by the difference of the exponents. The other coefficient is then added into the upper half of the accumulator. The result is a double-length register with the format shown in figure 5-4.

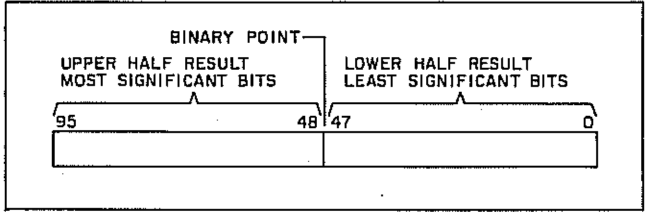


Figure 5-4. Floating-Add Result Format

If single precision is selected, the upper 48 bits of the 96-bit result and the larger exponent are returned as the result. Selecting double precision causes only the lower 48 bits of the 96-bit result and the larger exponent minus 60 (octal) to be returned as the result. The subtraction of 60 (octal) is necessary because the binary point is effectively moved from the right of bit 48 to the right of bit 0.

A 96-bit product generates from two 48-bit coefficients. The result of a multiply is a double-length register with the format shown in figure 5-5.

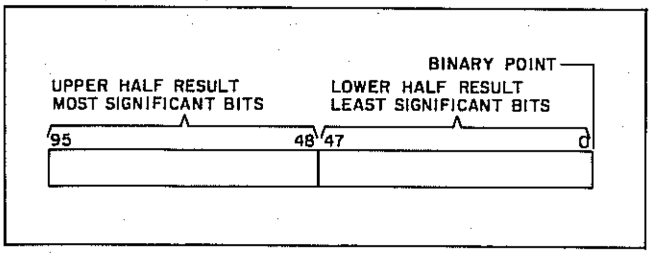


Figure 5-5. Multiply Result Format

If single precision is selected, the upper 48 bits of the product and the sum of the exponents plus 60 (octal) are returned as the result. The addition of 60 (octal) is necessary because the binary point effectively moves from the right of bit 0 to the right of bit 48 when the upper half of the 96-bit result is selected. If double precision is selected, the result is the lower 48 bits of the product and the sum of the exponents.

FIXED-POINT ARITHMETIC

Fixed-point addition and subtraction of 60-bit numbers are handled by the long-add instructions (36 and 37). Negative numbers are represented in one's complement notation, and overflows are ignored. The sign bit is in the high-order bit position (bit 59), and the binary point is to the right of the low-order bit position (bit 0).

Fixed-point addition and subtraction of 18-bit numbers are handled by the increment instructions (50 through 77). Negative numbers are represented in one's complement notation, and overflows are ignored. The sign bit is in the high-order bit position (bit 17), and the binary point is to the right of the low-order position (bit 0).

Table 5-2. Xj Plus Xk (30, 32, 34 Instructions)

		Xk			
		W	+∞	-∞	± IND
Xj	W		+∞	-∞	IND
	+∞	+∞	+∞	IND	IND
	-∞	-∞	IND	-∞	IND
	± IND	IND	IND	IND	IND

Table 5-3. Xj Minus Xk (31, 33, 35 Instructions)

		Xk			
		W	+∞	-∞	± IND
Xj	W		-∞	+∞	IND
	+∞	+∞	IND	+∞	IND
	-∞	-∞	-∞	IND	IND
	± IND	IND	IND	IND	IND

Table 5-4. Xj Multiplied by Xk (40, 41, 42 Instructions)

		Xk						
		+N	-N	+0	-0	+∞	-∞	± IND
Xj	+N			0	0	+∞	-∞	IND
	-N			0	0	-∞	+∞	IND
	+0	0	0	Integer † multiply		IND	IND	IND
	-0	-0	0			IND	IND	IND
	+∞	+∞	-∞	IND	IND	+∞	-∞	IND
	-∞	-∞	+∞	IND	IND	-∞	+∞	IND
	± IND	IND	IND	IND	IND	IND	IND	IND

† If both operands used in the integer multiply are normalized, an underflow results.

Table 5-5. Xj Divided by Xk (44, 45 Instructions)

		Xk						
		+N	-N	+0	-0	+∞	-∞	+ IND
Xj	+N			+∞	-∞	0	0	IND
	-N			-∞	+∞	0	0	IND
	+0	0	0	IND	IND	0	0	IND
	-0	0	0	IND	IND	0	0	IND
	+∞	+∞	-∞	+∞	-∞	IND	IND	IND
	-∞	-∞	+∞	-∞	+∞	IND	IND	IND
	± IND	IND	IND	IND	IND	IND	IND	IND

Integer multiplication is handled as a subset operation of the floating-multiply (42) instruction. The integer multiply requires that both 47-bit integer operands have zero exponents and are not normalized. The result is 48 bits with sign extension. Normalized operands cause underflow results to be reported. If the results exceed 48 bits, overflow is not detected.

An integer divide takes several steps. For example, an integer quotient X1 equal to X2/X3 is produced by the following steps.

	Instructions	Remarks
1.	Pack X2 from X2 and B0.	Pack X2.
2.	Pack X3 from X3 and B0.	Pack X3.
3.	Normalize X3 in X0 and B0.	Normalize X3 (divisor).
4.	Normalize X2 in X2 and B0.	Normalize X2 (dividend).
5.	Floating quotient of X2 and X0 to Xi.	Divide.
6.	Unpack X1 to X1 and B7.	Unpack quotient.
7.	Shift X1 nominally left B7 places.	Shift to integer position.

The divide requires that both integer (2^{47} maximum) operands be in floating-point format, and the dividend coefficient must be less than two times the divisor coefficient. The normalize X3 instruction ensures this condition.

The normalize X3 instruction left shifts the divisor n places ($n \geq 0$), providing a divisor exponent of negative n. The quotient exponent is then 0 minus (-n) minus 48 equals n minus $48 < 0$.

After unpacking and left shifting nominally, the negative (or zero) value in B7 right shifts the quotient 48 minus n places, producing an integer quotient in X1. A remainder may be obtained by an integer multiply of X1 and X3 and subtracting the result from X2.

INTEGER ARITHMETIC

Integer divide packs the integers into floating-point format using the pack instruction with a zero-exponent value.

In integer multiplication, a 48-bit product can be formed by using the double-precision multiply instruction. Both operands must have an exponent value of ±0, and the coefficients cannot both be normalized. The result is sign-extended to 60 bits and sent to an X register.

In integer division, the divisor must be normalized but the dividend need not be normalized. The resulting quotient must be unpacked and the coefficient shifted by the amount of the unpacked exponent using the left shift (22) instruction to obtain the integer quotient.

INSTRUCTION LOOKAHEAD PURGE CONTROL

Prefetching of instructions at a branch target address by instruction lookahead hardware can lead to program failures if a program modifies its own code dynamically. Instruction lookahead hardware consists of the instruction word stack (IWS), instruction address stack (IAS), and current instruction word (CIW) registers. Refer to Instruction Lookahead in section 2 for additional instruction control information. Each IAS address register has a full bit contained in a shift register. As addresses are assigned, full bits shift in to indicate that the address register contains a valid address.

The IAS is purged by clearing the IAS full bits. A purged stack causes the next instruction (P + 1) to read from CM into the IWS before execution. After the stack is purged, instructions in remaining parcels of CIW (word P) execute without being reloaded from memory.

Under normal conditions, the lookahead (stack) registers are purged by execution of a return jump instruction (010), UEM/EEM block read instruction (011), exchange jump instruction (XSF), or unconditional jump instruction (02). The lookahead registers are also purged by any jump to an address not in the IAS. These conditions can be extended by selecting extended purge control. When extended purge control is in effect, lookahead registers are also purged by execution of any conditional branch instruction (03 through 07) or any CM store instruction (50 through 57 when i equals 6 or 7). To enable extended purge control, the system sets bit 52 of the stack purge flag register in the exchange package.

When self-modifying code is present, it may be helpful to set extended purge control; however, the additional purging does cause a degradation in execution and does not cover all cases of code modification.

ERROR RESPONSE

When the CP detects or is informed of an error, it records the error. Depending upon the type of error and the exit mode selection bits set, the program in execution may be interrupted. If the error is an illegal instruction or an address-range error on an RNI or branch, the program interruption is unconditional. For other types of errors, the exit mode selection bits determine whether or not the program is interrupted. If the exit mode selection bit is set and the corresponding condition is detected, the program is interrupted. The exit mode selection bits are contained in word N plus 3 of the exchange package. Figure 5-6 shows the format of the exit condition register at RAC. Table 5-6 describes the possible contents of the register. Tables 5-7, 5-8, and 5-9 list CP error responses.

Illegal Instructions

An instruction is illegal when it has an illegal operating code, an illegal operating parameter, or when it is positioned so that it begins in one instruction word and extends into the next instruction word. In job mode, illegal instructions cause an exchange to monitor mode. In monitor mode, they cause a simulated CP halt. CP illegal instructions are:

- 011, 012, 014, 015 if the UEM enable flag is clear, with no version of EEM present with the system.
- 464, 465, 466, 467.
- 013 with MEJ/CEJ disabled.
- 017.
- Any 30-bit instruction which begins at parcel 3.
- 01x in parcel 1, 2 and 3 (excluding 010).

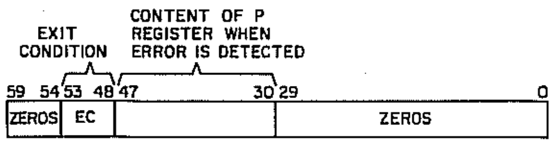


Figure 5-6. Format of Exit Condition Register at (RAC)

Table 5-6. Contents of Exit Condition Register at (RAC)

Field	Description
EC	6-bit exit condition code Code — Condition 00_8: illegal instruction 01_8: address out of range 02_8: infinite operand 04_8: indefinite operand 10_8: EEM flag function or direct parity error 20_8: CMC input error 40_8: CM data error
P	When an error exit occurs, the content of the P register may not correspond to the address of the instruction that caused the error exit. The P register may have been incremented prior to the execution of the instruction.

Hardware Errors

CP/CM hardware errors are: data parity errors, address parity errors, and double bit errors. If the CP is in job mode, a hardware error causes a jump to monitor mode. If the CP is in monitor mode when a hardware error occurs, the CP halts. The instruction being executed when such a fault is detected is not necessarily connected with the fault.

Conditional Software Errors

Conditional software errors are caused by address-range errors, and floating-point infinite/indefinite operands or results. A conditional software error causes action depending on bits set in the EM field in the current exchange package. If the bit reserved for use with the specific type of error is clear, the error is ignored in both job and monitor modes. If the bit is set and the error occurs in job mode, it causes an exchange to monitor mode. If the bit is set and the error occurs in monitor mode, the CPU stores at RAC and halts.

CM PROGRAMMING

All references to CM by the CP for instructions or read/write data are made relative to RAC. The RAC defines the lower limit of the addresses of a program in CM. The upper limit of the program addresses is defined by FLC added to RAC. The field length is a number of 60-bit words established by the operating system prior to program execution. All references to CM for a program must be within the field established for that program. During an exchange jump, a 21-bit RAC and a 21-bit FLC load into respective registers to define the CM limits of the program that is initiated by the exchange jump.

Figure 5-7 shows the absolute and relative memory addresses, RAC, FLC, and P register relationships. For a program to operate within the established limits, the following conditions must exist:

For absolute central memory addresses:

$$\text{RAC} \leq (\text{RAC} + \text{P}) < (\text{RAC} + \text{FLC})$$

For relative central memory addresses:

$$0 \leq \text{P} < \text{FLC}$$

EXTENDED MEMORY TRANSFERS

Extended memory (UEM or EEM) transfers occur with use of instructions 011, 012, 014 and 015. All CP references to extended memory for read/write data are made relative to RAE. The RAE defines the lower limit of the transfer addresses for extended memory. The upper limit of the transfer addresses is defined by FLE added to RAE.

During an exchange jump in standard addressing mode, a 21-bit RAE and a 24-bit FLE load into their respective registers to define the extended memory limits of the transfer. During an exchange jump in expanded addressing mode, a 24-bit RAE and a 30-bit FLE load into their respective registers to define the extended memory limits of the transfer. Refer to Extended Memory Addressing Modes in this section for discussion of standard and expanded addressing modes.

Figure 5-7 shows the absolute and relative memory addresses, RAE, FLE and P register relationships. For a program to operate within the established limits, the following conditions must exist:

For absolute extended memory (UEM/EEM) addresses:

$$\text{RAE} \leq (\text{RAE} + \text{UEM/EEM address}) < (\text{RAE} + \text{FLE})$$

For relative extended memory addresses:

$$0 \leq \text{UEM/EEM address} < \text{FLE}$$

Table 5-10 summarizes the possible UEM and EEM operations for the relative extended memory address field.

EXTENDED MEMORY ADDRESSING MODES

Two addressing modes provide access to UEM and EEM. These are standard addressing mode and expanded addressing mode. The addressing mode is determined by bit 55 of word 3 (expanded addressing select flag) of an exchange package. The addressing modes differ by which bit positions of exchange package words 4 and 5 define RAE and FLE.

Standard Addressing Mode

Standard addressing mode provides addressing up to 21 bits in a 24-bit format, or a maximum of two million 60-bit words. Refer to Legal Address Reference in table 5-10 for restrictions on extended memory transfers in standard addressing mode. A clear conditon of the expanded addressing select flag selects standard addressing mode for data transfer between CM and UEM or between CM and EEM.

Table 5-10, Extended Memory Operations, summarizes the various operations which may occur with an extended memory reference. The extended memory address forms from the addition of X0 plus RAE (block copies) or Xr plus RAE (single-word read/write). The CP instruction which makes the extended memory reference must be in range, such that X0 + (Bj + K) is less than or equal to FLE, and A0 + (Bj + K) is less than or equal to FLC.

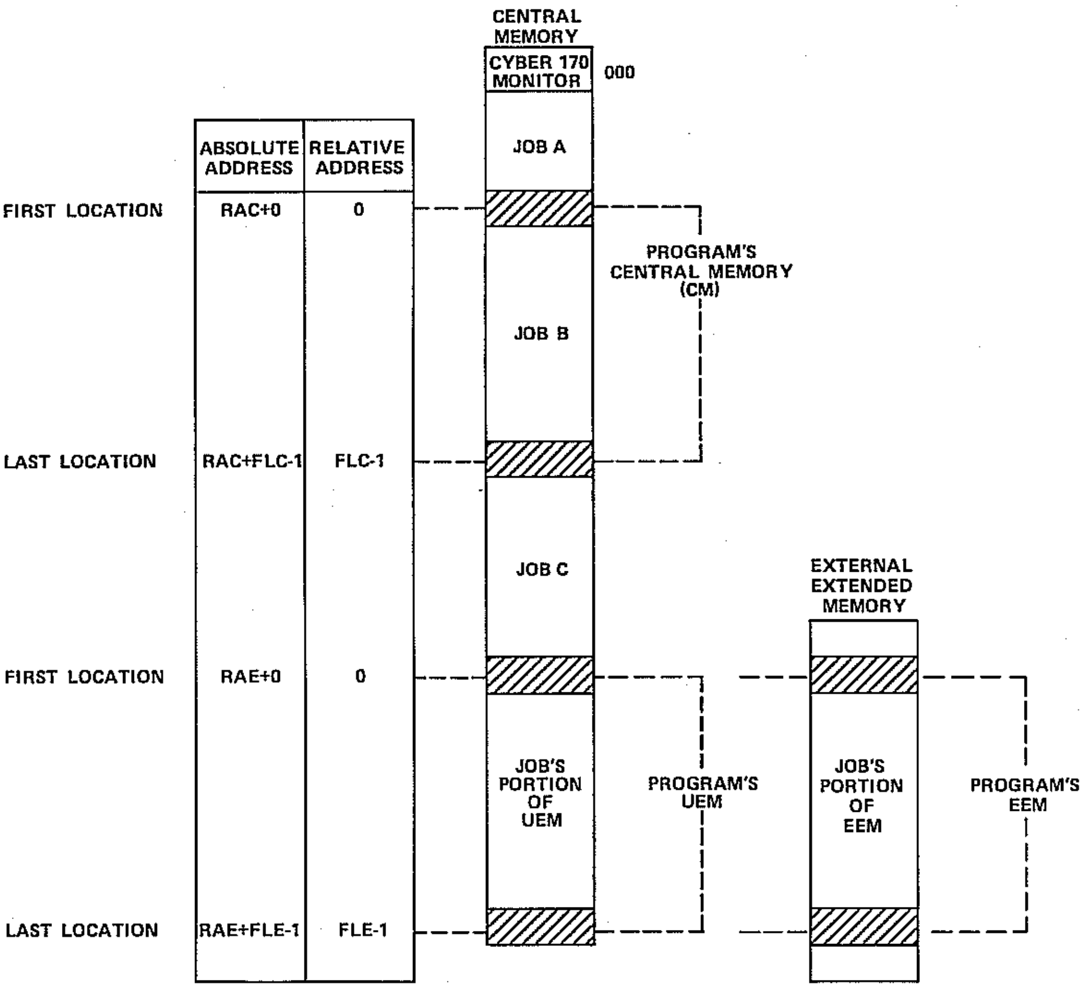


Figure 5-7. Memory Map

Table 5-7. Error Response with MEJ/CEJ Enabled, MF Set (Sheet 1 of 2)

Error Condition	Error Response	
	Exit Mode Selected	Exit Mode Not Selected
Illegal instruction.	1. Execute the illegal instruction as if it were a pass. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P.	1. Execute the illegal instruction as if it were a pass. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P.
Exit condition bit 48 set by an incremental read of CM or a direct extended memory read (014) of UEM of an address out of range.	1. Read all zeros of the selected X register. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P.	1. Inhibit read, X unchanged. 2. Continue execution.
Exit condition bit 48 set by an incremental write or by a direct extended memory write (015) of UEM to an address out of range.	1. Block write operation, contents of CM and UEM are unchanged. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P.	1. Block write operation, contents of CM and UEM are unchanged. 2. Continue execution.
Exit condition bit 48 set on RNI or branch out of range.	1. Stop CPU. 2. Store P and exit condition bits at RAC. 3. Clear P.	1. Stop CPU. 2. Store P and exit condition bits at RAC. 3. Clear P.
Exit condition bit 48 set by an EEM address range check, for direct extended memory instructions (014/015) using EEM or for extended memory block transfers (011/012).	1. Force EEM instruction to execute as a pass instruction. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P.	1. Force EEM instruction to execute as a pass instruction. 2. Exit to next 60-bit word. 3. Continue execution with next 60-bit word.
Infinite condition (bit 49) Indefinite condition (bit 50) EEM flag or direct parity condition (bit 51) CM data error condition (bit 53).	1. Stop CPU. 2. Store P and exit condition bits at RAC. 3. Clear P.	1. Continue execution.

Table 5-7. Error Response with MEJ/CEJ Enabled, MF Set (Sheet 2 of 2)

Error Condition	Error Response	
	Exit Mode Selected	Exit Mode Not Selected
CMC input error condition (bit 52).	1. Block write operation, contents of CM are unchanged. Block read operation, force read data to 777...777. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P.	1. Block write operation, contents of CM are unchanged. Block read operation, force read data to 777...777. 2. Continue execution.
00 Instruction.	1. Stop CPU. 2. Store P and exit condition bits at RAC. 3. Clear P.	1. Stop CPU. 2. Store P and exit condition bits at RAC. 3. Clear P.
Breakpoint signal from CMC. Refer to breakpoint notes.	1. Execute remaining parcels of 60-bit word currently executing. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P.	1. Execute remaining parcels of 60-bit word currently executing. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P.
Illegal instruction.	1. Execute the illegal instruction as if it were a pass. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P. 5. Exchange jump to MA and set MF.	1. Execute the illegal instruction as if it were a pass. 2. Exit to next 60-bit word. 3. Continue execution with next 60-bit word. 4. Clear P. 5. Exchange jump to MA and set MF.
Exit condition bit 48 set by an incremental read of CM or by a direct extended memory read (014) of UEM of an address out of range.	1. Read all zeros to the selected X register. 2. Stop CPU. 3. Store P and exit condition bits at MF. 4. Clear P. 5. Exchange jump to MA and set MF.	1. Read all zeros to the selected X register. 2. Continue execution.

Table 5-8. Error Response with MEJ/CEJ Enabled, MF Clear (Sheet 1 of 2)

Error Condition	Error Response	
	Exit Mode Selected	Exit Mode Not Selected
Exit condition bit 48 set by an incremental write of CM or by a direct extended memory write (015) of UEM to an address out of range.	1. Block write operation, content of CM and UEM are unchanged. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P. 5. Exchange jump to MA and set MF.	1. Block write operation, content of CM and UEM are unchanged. 2. Continue execution.
Exit condition bit 48 set due to an RNI or branch address out of range.	1. Stop CPU. 2. Store P and exit condition bits at RAC. 3. Clear P. 4. Exchange jump to MA and set MF.	1. Stop CPU. 2. Store P and exit condition bits at RAC. 3. Clear P. 4. Exchange jump to MA and set MF.
Exit condition bit 48 set by an EEM address range check, for direct extended memory instructions (014/015) using EEM or for extended memory block transfers (011/012).	1. Force EEM instruction to execute as a pass. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P. 5. Exchange jump to MA and set MF.	1. Force EEM instruction to execute as a pass. 2. Continue execution with next 60-bit word.
Infinite condition (bit 49) Indefinite condition (bit 50) EEM flag or direct parity condition (bit 51) CM data error condition (bit 53).	1. Stop CPU. 2. Store P and exit condition bits at RAC. 3. Clear P. 4. Exchange jump to MA and set MF.	1. Continue execution.
CMC input error condition (bit 52).	1. Block write operation, contents of CM are unchanged. Block read operation, force read data to 777...777. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P. 5. Exchange jump to MA and set MF.	1. Block write operation, contents of CM are unchanged. Block read operation, force read data to 777...777. 2. Continue execution.

Table 5-8. Error Response with MEJ/CEJ Enabled, MF Clear (Sheet 2 of 2)

Error Condition	Error Response	
	Exit Mode Selected	Exit Mode Not Selected
00 Instruction.	1. Stop CPU. 2. Store P and exit condition bits at RAC. 3. Clear P. 4. Exchange jump to MA and set MF.	1. Stop CPU. 2. Store P and exit condition bits at RAC. 3. Clear P.
Breakpoint signal from CMC. Refer to breakpoint notes.	1. Execute remaining parcels of 60-bit word currently executing. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P. 5. Exchange jump to MA and set MF.	1. Execute remaining parcels of 60-bit word currently exe-cuting. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P. 5. Exchange jump to MA and set MF.

Table 5-9. Error Response with MEJ/CEJ Disabled (Sheet 1 of 2)

Error Condition	Error Response	
	Exit Mode Selected	Exit Mode Not Selected
Illegal instruction.	1. Execute the illegal instruction as if it were a pass. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P.	1. Execute the illegal instruction as if it were a pass. 2. Exit to next 60-bit word. 3. Continue execution with next 60-bit word. 4. Clear P.
Exit condition bit 48 set by an incremental read of CM or by a direct extended memory read (014) of UEM of an address out of range.	1. Read all zeros to the selected X register. 2. Stop CPU. 3. Store P and exit condition bits at MF. 4. Clear P.	1. Read all zeros to the selected X register. 2. Continue execution.
Exit condition bit 48 set by an incremental write of CM or by a direct extended memory write (015) of UEM to an address out of range.	1. Block write operation, content of CM and UEM are unchanged. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P.	1. Block write operation, content of CM and UEM are unchanged. 2. Continue execution.
Exit condition bit 48 set due to an RNI or branch address out of range.	1. Stop CPU. 2. Store P and exit condition bits at RAC. 3. Clear P.	1. Stop CPU.
Illegal instruction.	1. Execute the illegal instructions.	1. Execute the illegal instruction.
Exit condition bit 48 set by an EEM address range check, for direct extended memory instructions (014/015) using EEM or for extended memory block transfers (011/012).	1. Force EEM instruction to execute as a pass. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P.	1. Force EEM instruction to execute as a pass. 2. Continue execution with next 60-bit word.

Table 5-9. Error Response with MEJ/CEJ Disabled (Sheet 2 of 2)

Error Condition	Error Response	
	Exit Mode Selected	Exit Mode Not Selected
Infinite condition (bit 49) Indefinite condition (bit 50) EEM flag or direct parity condition (bit 51) CM data error condition (bit 53).	1. Stop CPU. 2. Store P and exit condition bits at RAC. 3. Clear P.	1. Continue execution.
CMC input error condition (bit 52).	1. Block write operation, contents of CM are unchanged. Block read operation, force read data to 777...777. 2. Stop CPU. 3. Store P and exit condition bits at RAC. 4. Clear P.	1. Block write operation, contents of CM are unchanged. Block read operation, force read data to 777...777. 2. Continue execution.
00 Instruction.	1. Stop CPU.	1. Continue execution.

Table 5-10. Extended Memory Operations

EXTENDED MEMORY OPERATION	EXTENDED MEMORY TYPE	ADDRESSING MODE	X0 + RAE (011, 012) OR Xk + RAE (014, 015)																													
			29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LEGAL ADDRESS REFERENCE	EEM①	STANDARD	▨	▨	▨	▨	▨	▨	0	0	0	EEM ADDRESS																				
		EXPANDED	0	0	R	R	R	0	EEM ADDRESS																							
	UEM②	STANDARD	▨	▨	▨	▨	▨	▨	0	0	0	R	UEM ADDRESS																			
		EXPANDED	0	0	R	R	R	R	R	R	R	R	UEM ADDRESS																			
UNCONDITIONAL OVERFLOW (FAKE READ/WRITE AND EXIT)	EEM	STANDARD	▨	▨	▨	▨	▨	▨	0	0	1	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
		EXPANDED	0	▨	R	R	R	1	▨	0	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
		EXPANDED	0	1	R	R	R	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
	UEM	STANDARD	▨	▨	▨	▨	▨	▨	0	▨	1	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
		STANDARD	▨	▨	▨	▨	▨	▨	0	1	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
		EXPANDED	0	1	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
MAINTENANCE FUNCTION	EEM③	STANDARD	▨	▨	▨	▨	▨	▨	0	1	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	MAINTENANCE FUNCTION CODE							
		EXPANDED	0	0	R	R	R	1	▨	1	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	MAINTENANCE FUNCTION CODE							
ADDRESS RANGE ERROR (ATTEMPTED FLAG OPERATION)	UEM	STANDARD	▨	▨	▨	▨	▨	▨	1	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
		EXPANDED	1	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
FLAG REGISTER OPERATION 18-BIT RGTR	EEM	STANDARD	▨	▨	▨	▨	▨	▨	1	F	F	U	U	U	FLAG WORD																	
		EXPANDED	1	▨	R	R	R	▨	▨	F	F	0	U	U	FLAG WORD																	
FLAG REGISTER OPERATION 4-BIT RGTRS	EEM	EXPANDED	1	▨	R	R	R	▨	▨	F	F	1	F	F	FLAG REGISTER ADDRESS														FLAG WORD			
			29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

KEY	
R	RESERVED FOR FUTURE USE AND SET TO ZERO
F	FLAG OPERATION FUNCTION CODE
U	UNDEFINED RESULTS IF NOT SET TO ZERO
▨	BITS MAY BE SET OR CLEAR, WITH NO EFFECT ON EXTENDED MEMORY OPERATION
①	A LEGAL ADDRESS REFERENCE IN STANDARD ADDRESSING MODE ALLOWS ADDRESSING UP TO TWO MILLION WORDS OF EEM (BOTH ECS AND ESM VERSIONS). IF THE SYSTEM CONFIGURATION IS LESS THAN TWO MILLION WORDS, ANY EEM REFERENCE ABOVE THIS LIMIT RESULTS IN AN UNCONDITIONAL OVERFLOW (FAKE READ/WRITE AND EXIT). A LEGAL ADDRESS REFERENCE IN EXPANDED ADDRESSING MODE ALLOWS ADDRESSING UP TO TWO MILLION WORDS OF EEM (ECS VERSION) OR 16 MILLION WORDS OF EEM (ESM VERSION). IF THE SYSTEM CONFIGURATION IS LESS THAN TWO MILLION WORDS (ECS) OR 16 MILLION WORDS (ESM), ANY EEM REFERENCE ABOVE THIS LIMIT RESULTS IN AN UNCONDITIONAL OVERFLOW (FAKE READ/WRITE AND EXIT).
②	A LEGAL ADDRESS REFERENCE IN STANDARD OR EXPANDED ADDRESSING MODE ALLOWS ADDRESSING UP TO THE SOFTWARE-DEFINED LIMIT OF UEM. IN A MAXIMUM CM CONFIGURATION, THIS LIMIT IS ONE MILLION WORDS. ANY UEM REFERENCE ABOVE THE LIMIT RESULTS IN AN UNDEFINED OPERATION.
③	MAINTENANCE FUNCTIONS ARE SUPPORTED ONLY BY THE ESM VERSION OF EEM.

Expanded Addressing Mode

Expanded addressing mode provides addressing up to 24 bits in a 30-bit format, or a maximum of 16 million 60-bit words. Refer to Legal Address Reference in table 5-10 for restrictions on extended memory transfers in expanded addressing mode. A set condition of the expanded addressing select flag selects expanded addressing mode for data transfer between CM and UEM or CM and EEM.

EXCHANGE BREAK-IN CHARACTERISTICS

The exchange break-in characteristics of a direct read/write or block copy instruction and a flag register operation can result in various exit conditions. The conditions leading to or during the operations can result in an error exit, full exit, or half exit.

Refer to Error Response in this section for error exit description.

A full exit causes the CP to exit to parcel 0 of the next instruction word.

A half exit causes the CP or CPU to exit to the instruction in parcel 2 of the 60-bit instruction word being executed. Parcel 2 normally contains a branch instruction to an error routine. Table 5-11 and Figure 5-8 further define conditions that lead to exits from a block copy operation.

An exchange jump breaks in to a CP performing a block copy transfer (011 or 012) providing there are more than 64 CM words remaining to be transferred. On dual CP systems, an exchange jump may be initiated to a CP while the opposite CP is performing a block copy transfer. In this case, the transfer pauses during the exchange jump and resumes when the exchange jump completes.

DIRECT READ/WRITE INSTRUCTIONS (014, 015, 660, 670)

These instructions transfer one 60-bit word between the selected X register and a central or extended memory location, using a 24-bit or 30-bit relative address.

Instructions 660 and 670 execute in standard addressing mode to address CM, using the memory address Xk (21 bits) plus RAC (21 bits).

Instructions 014 and 015 may execute in standard or expanded addressing mode to access UEM or EEM. In standard addressing mode, instructions 014 and 015 use the memory address Xk (21 bits) plus RAE (21 bits) to address UEM or EEM, depending on the condition of the UEM mode flag. A set condition of the UEM mode flag (bit 56 of word 3 in an exchange package) selects a single-word transfer from UEM. A clear condition of the UEM mode flag selects a single-word transfer from EEM. In expanded addressing mode, instructions 014 and 015 use the memory address Xk (24 bits) plus RAE (24 bits) to address UEM or EEM, depending on the condition of the UEM mode flag.

BLOCK COPY INSTRUCTIONS (011, 012)

These instructions transfer up to 131 071 60-bit words from extended memory to CM (011), or from CM to extended memory (012). A block copy instruction may access UEM or EEM, depending on a set or clear condition of the UEM mode flag, respectively. The transfers occur in blocks of up to 64 words, during which exchange jump requests to the CPU performing the transfer are suspended. As a result, with model 865, transfer speeds up to one 60-bit word each 100 nanoseconds with UEM or EEM can be obtained. With model 875, transfer speeds up to one 60-bit word each 100 nanoseconds with EEM or 50 nanoseconds with UEM can be obtained. Figure 5-8 depicts in flowchart form the possible operations which may result from an 011/012 instruction issue.

The block copy transfer may occur in standard or expanded addressing mode, depending on a clear or set condition of the expanded addressing select flag.

In standard addressing mode, a clear condition of bits 22 and 23 of the extended memory address at X0 initiates a block copy transfer. A set condition of bit 23 of the X0 address and bit 23 of the FLE register causes a flag register operation (refer to Flag Register Operation). Flag register operations exist only for EEM references, and result in an address range error for UEM references.

In expanded addressing mode, a clear condition of bits 22 and 29 of the extended memory address at X0 initiates a block copy transfer. A set condition of bit 29 of the X0 address and bit 29 of the FLE register causes a flag register operation.

Table 5-11. 011/012 Block Copy Operation Exit Condition (Sheet 1 of 2)

Function	Transfer Conditions	Transfer Results
Read or write EEM with EEM memory size $> X0 + Bj + K \leq FLE$	Error-free transfer occurs for: • $Bj + K > 0$ • $X0 + Bj + K \leq FLE$ • $A0 + Bj = K \leq FLC$	Entire transfer completes and a full exit occurs.
	EEM bank is not available because: • Computer is in maintenance mode. • EEM loses power. • EEM is not part of system.	Additional data does not transfer, including current record. A half exit occurs.
Read EEM with EEM memory size $> X0 + Bj + K \leq FLE$	Parity error occurs in address from SMU to EEM.	Entire transfer completes with zero data (proper data parity) to CM from point EEM address error.
	Parity error occurs in address from CMC to CM.	Entire transfer completes. Data does not store in CM for words that had an address associated with a parity error. A half exit occurs after the transfer.
	Parity error occurs in data detected by ECS/ESM controller or SMU.	Entire transfer completes, including erroneous data. A half exit occurs after the transfer.†
	Parity error occurs in data from SMU and is detected by CMC.	Entire transfer completes, including erroneous data. A half exit occurs after the transfer.
	Block of data reads from an existing EEM address and continues into a nonexistent memory address.	Data transfer occurs until nonexistent address is reached. Transfer then completes with zero data (proper data parity) to CM. A half exit occurs after the transfer.
	Block of data reads, starting before an existing EEM address.	Entire data transfer completes with zero data (proper data parity) to CM. A half exit occurs after the transfer.
Write EEM with EEM memory size $> X0 + Bj + K \leq FLE$	Parity error occurs in address from SMU to ECS/ESM controller.	No additional data transfers, including current EEM record. A half exit occurs.
	Parity error occurs in address from CMC to CM.	Entire transfer completes with all ones data to EEM for words associated with parity error. A half exit occurs after the transfer.
	Data reads from CM with a corrected error.	Entire transfer completes. A full exit occurs after the transfer.

†This refers specifically to parity errors on data transferred to CM. A parity error beyond the word count is ignored. For example, a single-word read of word 4 from an EEM record with parity errors in words 3 and 5 will not half exit.

Table 5-11. 011/012 Block Copy Operation Exit Condition (Sheet 2 of 2)

Function	Transfer Conditions	Transfer Results
	Data reads from CM with an uncorrectable error.	Entire transfer completes, including erroneous data. A half exit occurs after the transfer.
	Parity error occurs in data from CM and is detected at SMU.	Not tested.
	Parity error occurs in data from SMU and is detected by ECS/ESM controller.	Entire transfer completes including erroneous data. A half exit occurs after the transfer.
	Block of data writes into an existing EEM address and continues into a nonexistent memory address.	Data transfer occurs until nonexistent address is reached. Transfer then stops, and a half exit occurs.
	Block of data writes, starting before an existing EEM address.	Data does not transfer. A half exit occurs.
Read EEM with EEM memory size $\leq$ XO + Bj + K $\leq$ FLE	EEM overflow.	Entire transfer completes with zero data (proper data parity) to CM. A half exit occurs after the transfer.
Write EEM with EEM memory size $\leq$ XO + Bj + K $\leq$ FLE	EEM overflow.	Data does not transfer and a half exit occurs.

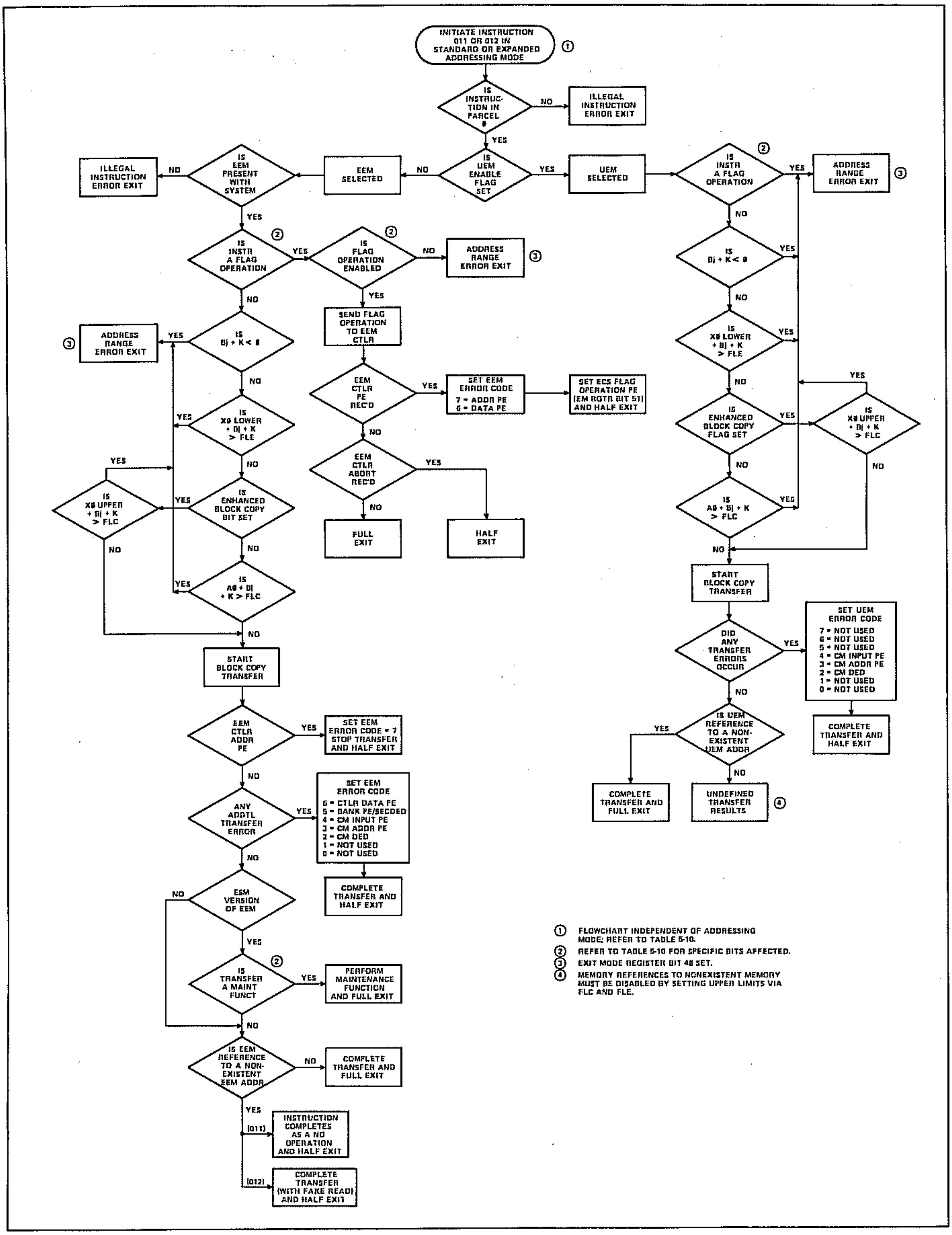

INITIATE INSTRUCTION 011 OR 012 IN STANDARD OR EXPANDED ADDRESSING MODE ①
IS INSTRUCTION IN PARCEL 0
ILLEGAL INSTRUCTION ERROR EXIT
IS UEM ENABLE FLAG SET
EEM SELECTED
UEM SELECTED
IS EEM PRESENT WITH SYSTEM
ILLEGAL INSTRUCTION ERROR EXIT
IS INSTR A FLAG OPERATION ②
IS FLAG OPERATION ENABLED ②
ADDRESS RANGE ERROR EXIT ③
SEND FLAG OPERATION TO EEM CTLR
EEM CTLR PE REC'D
SET EEM ERROR CODE 7 = ADDR PE 6 = DATA PE
SET ECS FLAG OPERATION PE (EM RDTR BIT 51) AND HALF EXIT
EEM CTLR ABORT REC'D
FULL EXIT
HALF EXIT
IS Bj + K < 0
ADDRESS RANGE ERROR EXIT ③
IS X0 LOWER + Bj + K > FLE
IS ENHANCED BLOCK COPY BIT SET
IS X0 UPPER + Bj + K > FLC
IS A0 + Bj + K > FLC
START BLOCK COPY TRANSFER
EEM CTLR ADDR PE
SET EEM ERROR CODE = 7 STOP TRANSFER AND HALF EXIT
ANY ADDTL TRANSFER ERROR
SET EEM ERROR CODE 6 = CTLR DATA PE 5 = BANK PE/SECDED 4 = CM INPUT PE 3 = CM ADDR PE 2 = CM DED 1 = NOT USED 0 = NOT USED
COMPLETE TRANSFER AND HALF EXIT
ESM VERSION OF EEM
IS TRANSFER A MAINT FUNCT ②
PERFORM MAINTENANCE FUNCTION AND FULL EXIT
IS EEM REFERENCE TO A NON-EXISTENT EEM ADDR
COMPLETE TRANSFER AND FULL EXIT
(011) INSTRUCTION COMPLETES AS A NO OPERATION AND HALF EXIT
(012) COMPLETE TRANSFER (WITH FAKE READ) AND HALF EXIT
IS INSTR A FLAG OPERATION ②
ADDRESS RANGE ERROR EXIT ③
IS Bj + K < 0
IS X0 LOWER + Bj + K > FLE
IS ENHANCED BLOCK COPY FLAG SET
IS X0 UPPER + Bj + K > FLC
IS A0 + Bj + K > FLC
START BLOCK COPY TRANSFER
DID ANY TRANSFER ERRORS OCCUR
SET UEM ERROR CODE 7 = NOT USED 6 = NOT USED 5 = NOT USED 4 = CM INPUT PE 3 = CM ADDR PE 2 = CM DED 1 = NOT USED 0 = NOT USED
COMPLETE TRANSFER AND HALF EXIT
IS UEM REFERENCE TO A NON-EXISTENT UEM ADDR
COMPLETE TRANSFER AND FULL EXIT
UNDEFINED TRANSFER RESULTS ④
YES
NO
① FLOWCHART INDEPENDENT OF ADDRESSING MODE; REFER TO TABLE 5-10.
② REFER TO TABLE 5-10 FOR SPECIFIC BITS AFFECTED.
③ EXIT MODE REGISTER BIT 48 SET.
④ MEMORY REFERENCES TO NONEXISTENT MEMORY MUST BE DISABLED BY SETTING UPPER LIMITS VIA FLC AND FLE.

Figure 5-8. Block Copy Operation Flowchart

Enhanced Block Copy Instruction

A modified version of the 011/012 instruction, called the enhanced block copy instruction, permits accessing a block of CM memory words more than 262K words from RAC. This is achieved by using the upper 30 bits of X0 as a CM starting address instead of the 18 bits of A0. X0 bits 30 through 50 provide a 21-bit format for the CM starting address in EEM, as compared to the 18-bit format of A0.

The enhanced block copy mode flag is bit 54 of word 3 of an exchange package. A clear flag selects A0 as the starting CM address; a set flag selects the upper 30-bit parcel of X0 as the starting CM address.

Fake Read/Write and Exit

A fake read/write and exit condition occurs in a 011, 012, 014, or 015 instruction reference to UEM or EEM if an unconditional overflow occurs (refer to table 5-10), or if the EEM address exceeds the amount of EEM available (in both standard and expanded addressing modes).

A fake read and exit enters Xj in CM with zero, and causes a half exit (011, 012 only) to parcel two of the instruction currently being executed. A fake write and exit does not store data and results in a No Operation with unspecified timing and a half exit to parcel two of the instruction currently being executed.

FLAG REGISTER OPERATION

A flag register operation, used only with EEM references, involves the use of a flag register located in the ECS/ESM controller. The register allows programs to provide information about current or previous EEM operations. One use of the register is analogous to a reserved status word that is maintained in EEM. The register is, however, accessible at a far greater speed than the status word because it does not require an EEM reference. The flag register cannot be read directly. Interrogation and/or writing into the register requires an SMU.

In standard addressing mode, selection of a flag register operation occurs when bit 23 sets in the EEM address (table 5-10) and bit 23 sets in the FLE register (figure 5-1) during an EEM read or write operation. In expanded addressing mode, selection of a flag register operation occurs when bit 29 sets in the EEM address (table 5-10) and bit 29 sets in the FLE register (figure 5-1) during an EEM read or write operation.

With an 18-bit flag register operation, the ECS/ESM controller recognizes the flag register operation and translates bits 21 and 22 of the address. The translation determines the function to be performed, which may be to compare or enter the flag word (bits 0 through 17) into the flag register. The controller then sends either an abort or an accept signal back to the SMU.

With a 4-bit flag register operation, the ECS/ESM controller recognizes the flag register operation and translates bits 18 and 19 of the address, in addition to bits 21 and 22. The translation determines the function to be performed, and the controller then sends either an abort or an accept signal back to the SMU.

If the ECS/ESM controller receives an address with bad parity, the controller does not send an accept signal to the SMU. Instead, the controller sends an abort signal and a controller parity signal. The flag register does not change, and an exit to the lower 30 bits of the instruction word occurs immediately.

The flag register operation is the same for either an ECS/ESM read or write and is unaffected by a reduction of ECS to 50-percent capacity.

FLAG FUNCTION CODES

The flag function codes may specify seven flag operations. The flag register format, detailed in table 5-10 in standard and expanded addressing modes, is shown in the following examples to depict the particular function codes and respective flag operations.

Ready/Select

Figure 5-9 illustrates the function code for a ready/select operation. This operation performs a bit-by-bit comparison between the content of the flag register and the flag word. If all the set bits in the flag word are clear in the flag register, a positive comparison occurs, and all the set bits in the flag word enter the flag register. The cleared bits in the flag word have no effect on the flag register.

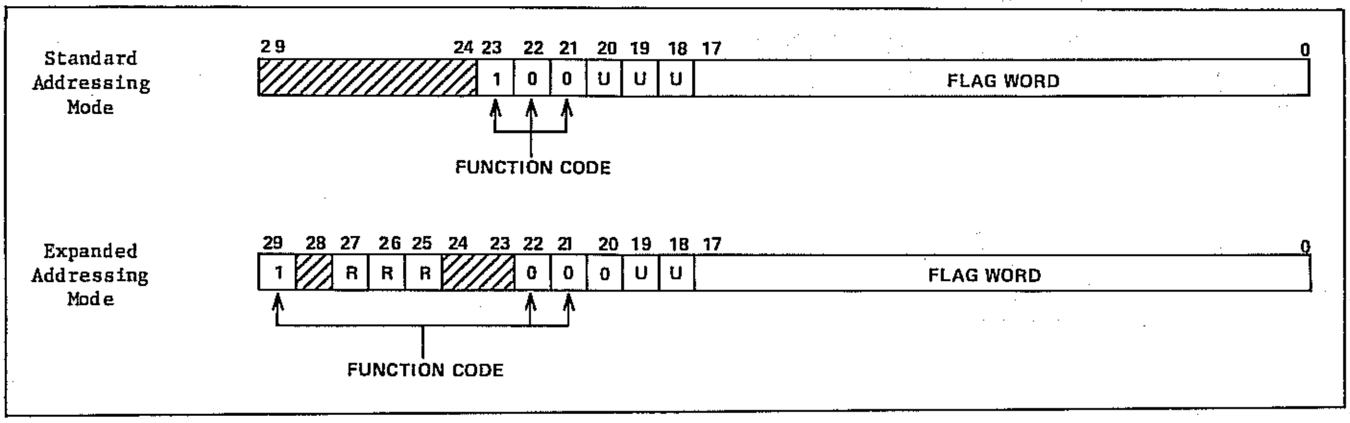


Figure 5-9. Ready/Select Function Code

Example: (Only three bits are shown)

Initial contents of flag register = 010

Flag word = 101

(This is a positive comparison so the flag register is changed, and an accept is transmitted by the ECS/ESM controller to the SMU.)

Final contents of flag register = 111

If a positive comparison is not made, the flag register remains unchanged, and an abort is transmitted to the SMU.

Example: (Only three bits are shown)

Initial contents of flag register = 010

Flag word = 111

(This is a negative comparison so the flag register is unchanged and transmits an abort to the SMU.)

Final contents of flag register = 010

Selective Set

Figure 5-10 illustrates the function code for a selective set operation. No comparison is made. All set bits in the flag word set in the flag register. The only response is an accept.

Example: (Only three bits are shown)

Initial contents of flag register = 010

Flag word = 100

Final contents of flag register = 110

Status

Figure 5-11 illustrates the function code for a status operation. This is the same as a ready/select code, but the flag register is not changed on a positive comparison. The comparison is made in the same manner, and the exit conditions are the same.

Example: (Only three bits are shown)

Initial contents of flag register = 010

Flag word = 101

(This is a positive comparison and an accept is transmitted by the ECS/ESM controller to the SMU. The flag register is unchanged.)

Final contents of flag register = 010

If a positive comparison is not made, an abort is transmitted to the SMU. The flag register remains unchanged.

Example: (Only three bits are shown)

Initial contents of flag register = 010

Flag word = 111

(This is a negative comparison so the flag register is unchanged and transmits an abort to the SMU.)

Final contents of flag register = 010

Selective Clear

Figure 5-12 illustrates the function code for a selective clear operation. No comparison is made. All set bits in the flag word clear in the flag register. The only response is an accept.

Example: (Only three bits are shown)

Initial contents of flag register = 110

Flag word = 101

Final contents of flag register = 010

NOTE

The remaining functions are only operational on the 16K-by-4-bit flag registers.

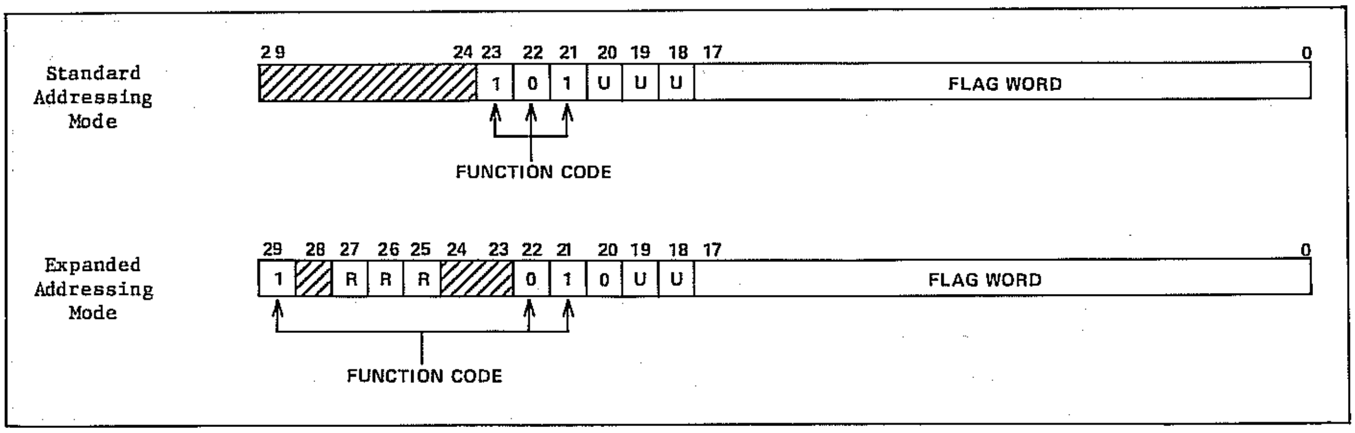


Figure 5-10. Selective Set Function Code

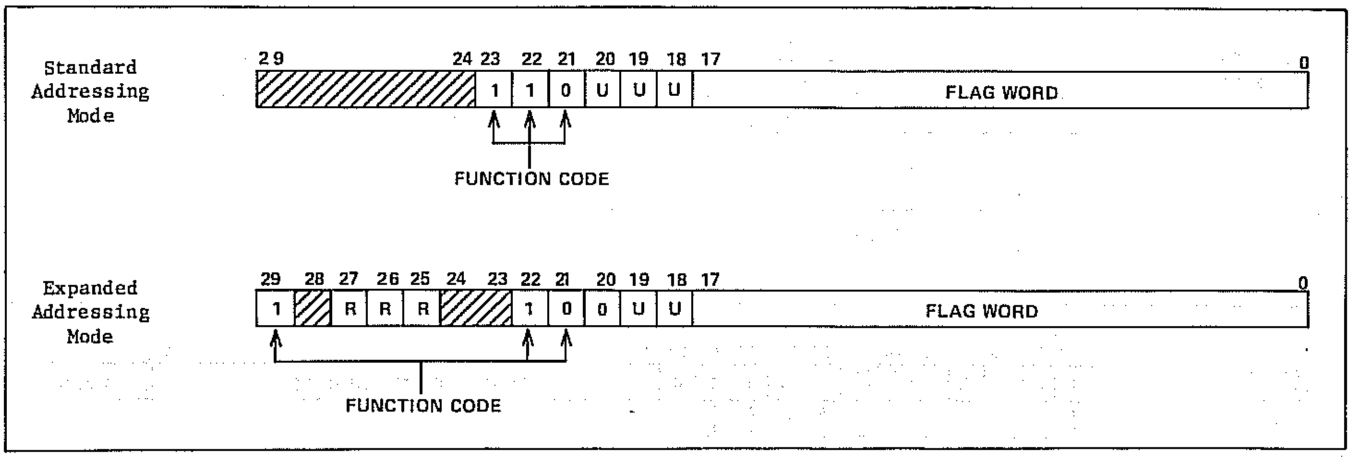


Figure 5-11. Status Function Code

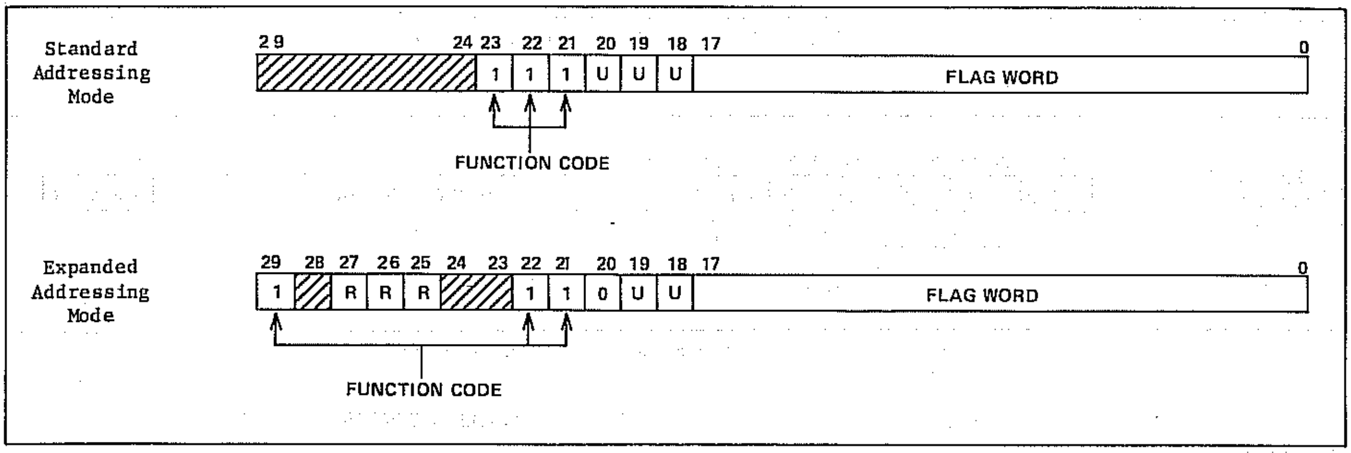


Figure 5-12. Selective Clear Function Code

Zero/Select

Figure 5-13 illustrates the function code for a zero/select operation. This operation checks the flag register contents for all bits being clear. If all flag register bits are clear, the contents of the flag word enters the flag register. If any bit is previously set in the flag register, the ECS/ESM controller responds with an abort and the flag register is unchanged.

Bit 20 must be set in this function, because this does not operate on the 18-bit register, but operates only on the 16K-by-4-bit registers.

Example:

Initial contents of flag register = 0100

Flag word = xxxx

This is a negative comparison because all flag register bits are not clear. An abort is returned.

Final contents of flag register = 0100 (unchanged from initial contents).

Example:

Initial contents of flag register = 0000

Flag word = 1010

This is a positive comparison because the flag register is fully clear. An accept is returned.

Final contents of flag register = 1010 enter the flag word.

Equality Status

Figure 5-14 illustrates the function code for an equality status operation. In this operation, the flag register contents compare with the flag word. If the comparison is positive in all four bits, an accept returns to the accessing port. If the comparison is negative in any bit, an abort returns to the accessing port.

Detected Error Status

Figure 5-15 illustrates the function code for a detected error status operation. This function returns an accept or an abort, dependent on the state of the detected errors in the side door channel. (The side door channel is a low-speed

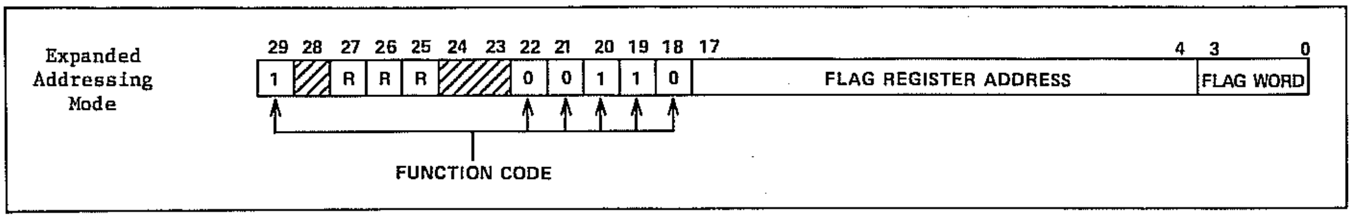


Figure 5-13. Zero/Select Function Code

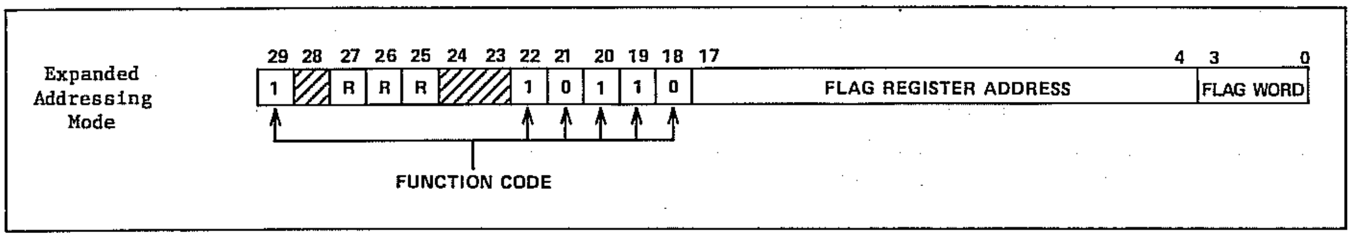


Figure 5-14. Equality Status Function Code

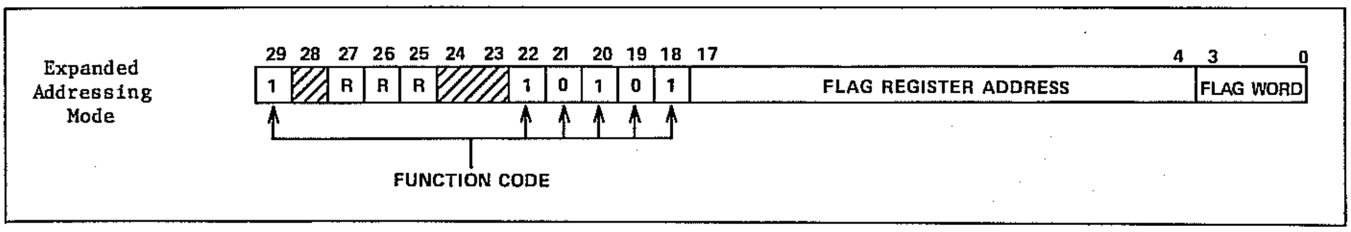


Figure 5-15. Detected Error Status Function Code

port connected to an IOU channel, used for monitoring EEM.) If all of the detected errors are clear, an accept returns. If any of these bits are set, an abort returns.

The side door channel parity is not included for the purpose of this function. This function does not change or check status on any of the flag registers. The address bits 0 through 17 are not used in the flag word.

PP PROGRAMMING

The PPs have access to all CM storage locations. One 60-bit word or a block of 60-bit words can be transferred from a peripheral processor memory (PPM) to CM or from CM to PPM. (Five 12-bit PP words equal one 60-bit CM word.) Data from external devices is read into a PPM, and with additional instructions, is transferred to CM. Conversely, data is transferred from CM to a PPM and is then transferred by additional instructions to external devices. Addresses sent to CM from PPs are absolute or relocation addresses.

CENTRAL MEMORY ADDRESSING BY PPs

PPs address central memory using either absolute or relocation addressing. Every PP can read all central memory locations without restriction. Every PP has write access to central memory.

Instructions 24/25 load/store the relocation (R) register. If bit 17 of the A register is zero, bits 0 through 16 of A specify an absolute central memory address 0 through 377 777_8. If bit 17 of A is one, bits 0 through 16 of A are added to the 21-bit R register to specify an absolute central memory address 0 through 0 007 777 777_8. If bit 17 of A changes during a transfer, the addressing mode also changes accordingly. The leftmost 7 bits of R represent extra addressing capacity which is unused. The rightmost 6 bits of R are appended zeros. Instruction 24 loads R from two consecutive PP memory locations. Instruction 25 stores R into two PP memory locations. Figure 4-4 shows how R is stored in PP memory.

PP MEMORY ADDRESSING BY PPs

PP instructions use 6-bit or 12-bit direct operands, or access PP memory through direct, indirect, or indexed addressing.

Direct 6-Bit Operand

PP instructions in this category are no-address instructions. They have the format OPCODEd. The d field is used as a 6-bit direct operand, zero-extended to 18 bits in calculations.

Direct 18-Bit Operand

PP instructions in this category are constant address instructions. They have the format OPCODEdm. The combined d and m fields are used as an 18-bit operand.

Direct 6-Bit Address

PP instructions in this category are direct address instructions. They have the format OPCODEd. The d field is used as a 6-bit direct address, accessing PP memory locations 0 to 77_8.

Direct 12-Bit Address

PP instructions in this category are indexed direct address instructions, with zero index. They have the format OPCODEdm, d equals 0. The m field is used as a 12-bit direct address, accessing PP memory locations 0 through 7777_8.

Indexed 12-Bit Address

PP instructions in this category are indexed direct address instructions. They have the format OPCODEdm, d equals 0. The m field is used as a 12-bit direct address (base address). The d field specifies a PP memory location from 1 to 77_8, the contents of which is a 12-bit one's complement number index. The indexed direct address is formed by adding the index to the base address as signed one's complement numbers, ignoring overflow. When m plus (d) equals 7777 the result is set to 0000, except as follows: adding 7777 plus 7777 equals 7777. In general, adding 0000 or 7777 leaves the other number unchanged, except when the other number is also 0000 or 7777.

Indirect 6-Bit Address

PP instructions in this category are indirect address instructions. They have the format OPCODEd. The 6-bit d field is used to read a 12-bit number from PP locations 0 through 77_8; this number is used as a 12-bit address to access PP memory locations 0 through 7777_8.

READ/WRITE INSTRUCTIONS

PP Central Memory Read Instructions (60,61)

Instruction 60 transfers one CM word into five 12-bit PP memory words. Instruction 61 transfers a block of 1 through 811 CM words into 5 through 4095 12-bit PP words; it is possible to transfer up to 4096 CM words overwriting PP memory cyclically; location 0, however, has special properties. Refer to instruction 61.

PP Central Memory Write Instructions (62,63)

Instruction 62 transfers five 12-bit PP memory words into one CM word. Instruction 63 transfers 5 through 4095 PP memory words into 1 through 811 CM words. It is possible to transfer up to 20 480 PP memory words, repeating information from PP memory cyclically.

INPUT/OUTPUT CHANNEL COMMUNICATIONS

Data transfers to and from external devices are controlled by PP instructions 64 through 77. The assignment of PPs, transfer priorities and resolution of conflicts are software responsibilities.

Channel parity and reservation must be provided for, using the channel marker flag and/or software interlocks in central memory. After all conflicts have been resolved, proceed as follows:

	Action	Typical Instruction
1.	Verify inactive status.	Jump if active (64).
2.	Verify read status.	
	Prepare for reading summary status.	Function m (77).
	Verify that the device responded.	Jump if active (64).
	Activate channel.	Activate (74).
	Read summary status.	Input to A (70).
	Disconnect channel.	Deactivate (75).
3.	Enter number of words to A.	Load d (14).
4.	Prepare for input/output.	
	Verify inactive status.	Jump if active (64).
	Prepare for read/write.	Function m (77).
	Verify that the device responded.	Jump if active (64).

5. Read/write data.

Activate channel.	Activate (74).
Read/write data.	Input/output A words (71/73).
If write, loop until empty.	Jump if full (66).
Disconnect channel.	Deactivate (75).
Verify inactive status.	Jump if active (64).

6. Verify transfer integrity.

Verify A words were transferred (refer to note 1).	Nonzero jump (05).
Verify inactive status.	Jump if active (64).
Prepare for reading device status.	Function m (77).
Verify that the device responded.	Jump if active (64).
Activate channel.	Activate (74).
Read device status.	Input to A (70).
Disconnect channel.	Deactivate (75).

NOTE

If A equals the original value, no words are transferred.

If A is not equal to 0, the device or another PP ends the transfer.

INTER-PP COMMUNICATIONS

Any PP can communicate with any other PP using any channel (except the real-time clock) by omitting the conditioning of the external devices of that channel for a data transfer. Both single word and block transfers can be used. Either the sending or the receiving PP can activate the channel used, after which the sending PP outputs data into the channel register of the channel concerned and the receiving PP inputs data from the same register. The transfer rate is one word every 500 nanoseconds, except when the transfer is between PPs in different barrels but in the same time slot. In such a case the transfer rate is one word every 1000 nanoseconds. PPs which use the same time slots are as follows:

Slot	PP Number
1	0, 20
2	1, 21
3	2, 22
4	3, 23
5	4, 24
6	5, 25
7	6, 26
8	7, 27
9	10, 30
10	11, 31

Software resolves priority and reservation problems arising in inter-PP communications by interlocks stored in CM or by other means.

PP PROGRAM TIMING CONSIDERATIONS

Some external equipment may require timing considerations in issuing function, activate, and input instructions. Refer to the applicable external equipment reference manual. Such timing considerations may, for example, be required to ensure that the equipment attains a proper speed before data is sent (required by some magnetic tape equipment). Also, equipment which terminates a data transfer by resetting the active flag to inactive often requires timing considerations in issuing the next function instruction.

CHANNEL OPERATION

Channel Control Flags

Channel operation is affected by the channel active/inactive and full/empty flags and, depending on the status of these two flags, the channel is said to be active, inactive, full, or empty. Each channel also has a marker flag for software use, and an error flag for indicating transmission parity errors.

Channel Active/Inactive Flag

A channel is normally activated by a function (76 or 77) instruction or by an activate channel (74) instruction. The channel can also be activated by an external device.

A function instruction conditions the external device for a coming data or status information transfer. The instruction places a 12-bit function word plus parity in the channel register and sets the active and full flags. The function word and a function signal are sent to the external device. No active or full signals are sent during a function instruction. The external device accepts the function word and sends an inactive signal which clears the channel active and full flags, clearing the channel register.

An activate channel instruction prepares a channel for data transfer and sends an active signal to the external device. Subsequent input or output instructions transfer data. A disconnect channel (75) instruction after a data transfer returns the channel to an inactive state, and an inactive signal is sent to the external device.

Register Full/Empty Flag

A register is full when it contains a function or data word for an external device or contains a word received from the external device. The register is empty when the flag clears. The flag is turned on or off as the register changes state. A channel can only be full when it is active.

On data output, the processor places a word in the channel register (the channel should be active and empty) and sets a full flag. The data word plus parity and a full signal are sent to the external device. The external device accepts the word and sends an empty signal to the channel which clears the full flag, clearing the channel register. The active and empty status of the channel signals the PP to send the next word to the register.

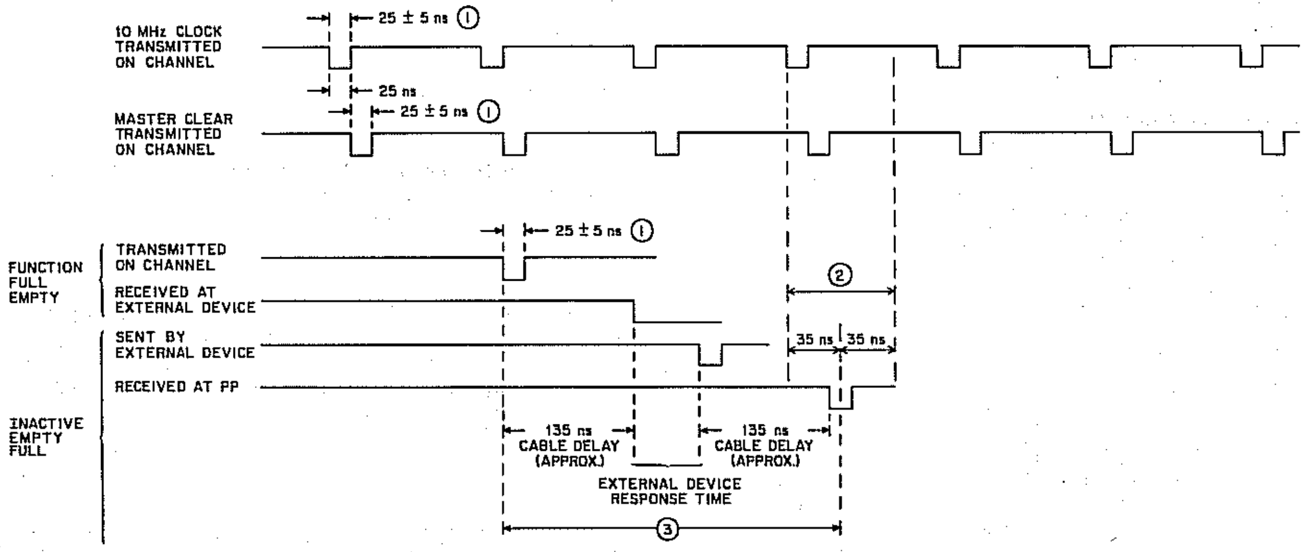
On data input, the external device sends a word and a full signal to the data channel. The word is placed in the channel register, and the full flag sets. The PP stores the word and clears the full flag, clearing the data register. An empty signal is sent to the external device signaling it to send the next data word.

Channel Transfer Timing

Figure 5-16 shows channel transfer timing. All signal pulses are 25±5 nanoseconds in width and occur 25±5 nanoseconds following the 10-megahertz clock.

To maintain the fastest possible cycle time (500 nanoseconds), a function/full/empty pulse from the PP must be answered with an inactive/empty/full pulse, respectively, within 310±35 nanoseconds. If the maximum speed is not required, this response time may be increased by multiples of 100 nanoseconds.

The PP master clock frequency can be varied by ±4 percent. The peripheral devices used must tolerate this frequency variation.



10 MHz CLOCK TRANSMITTED ON CHANNEL
25 ± 5 ns ①
25 ns
MASTER CLEAR TRANSMITTED ON CHANNEL
25 ± 5 ns ①
FUNCTION FULL EMPTY
TRANSMITTED ON CHANNEL
25 ± 5 ns ①
RECEIVED AT EXTERNAL DEVICE
②
INACTIVE EMPTY FULL
SENT BY EXTERNAL DEVICE
35 ns
35 ns
RECEIVED AT PP
135 ns CABLE DELAY (APPROX.)
135 ns CABLE DELAY (APPROX.)
EXTERNAL DEVICE RESPONSE TIME
③

NOTES:

① ALL TRANSMISSION PULSE WIDTHS (INCLUDING DATA, FULL, EMPTY, ETC.) ARE 25±5 ns.

② TO AVOID LOST DATA, ALL INPUTS FROM THE CHANNEL TO THE PP MUST ARRIVE WITHIN THE 70 ns. INPUTS MAY BE EARLIER OR LATER BY 100 ns MULTIPLES.

③ TOTAL TURNAROUND TIME BETWEEN FUNCTION AND INACTIVE IS MEASURED AT PP. THIS TIME VARIES DUE TO EXTERNAL DEVICE RESPONSE TIME BUT MUST BE WITHIN 310±35 ns TO MAINTAIN THE 500 ns CYCLE TIME.

Figure 5-16. Channel Transfer Timing

INPUT/OUTPUT TRANSFERS

Data Input Sequence

The external device sends data (figure 5-17) to the PP via the controller as follows:

1. The PP places a function word in the channel register and sets the full flag and the channel active flag. At the same time, the PP sends the first of a group of words and function signals to all controllers. The function signals cause all controllers to sample the words and identify the words as function codes rather than data words. Connect codes select controllers and modes of operation and clear nonselected controllers. Only selected controllers are connected.

2. The controller sends an inactive signal to the PP, indicating acceptance of the function code. The signal drops the channel active flag, which in turn drops the full flag and clears the channel register.

3. The PP sets the channel active flag and sends an active signal to the controller which signals the input equipment to start sending data.

4. The input equipment reads a 12-bit data word plus one parity bit and then sends the word with parity to the channel register with a full signal which sets the channel full flag (10 to 15 nanoseconds after the data arrives).

5. The PP stores the word, drops the full flag, and returns an empty signal, indicating acceptance of the word. The input equipment clears its data register and prepares to send the next word.

6. Steps 4 and 5 repeat for each word transferred.

7. At the end of the transfer, the controller clears its active condition and sends an inactive signal to the PP to indicate the end of the data. The signal clears the channel active flag to disconnect the controller and the PP from the channel.

8. As an alternative, the PP may choose to disconnect from the channel before the input equipment has sent all its data. The PP does this by dropping the active flag and sending an inactive signal to the controller which immediately clears its active condition and sends no more data, although the input equipment may continue to the end of its record or cycle (for example, a magnetic tape unit would continue to end-of-record and stop in the record gap).

Data Output Sequence

The PP sends data (figure 5-18) to the external device as follows:

1. The PP places a function word in the channel register and sets the full flag and the channel active flag. The function signal causes all controllers to sample the word and identify the word as a function code rather than a data word. Connect codes select controllers and modes of operation and clear nonselected controllers. Only selected controllers are connected.

2. The controller sends an inactive signal to the PP, indicating acceptance of the function code. The signal drops the channel active flag, which in turn drops the full flag and clears the channel register.

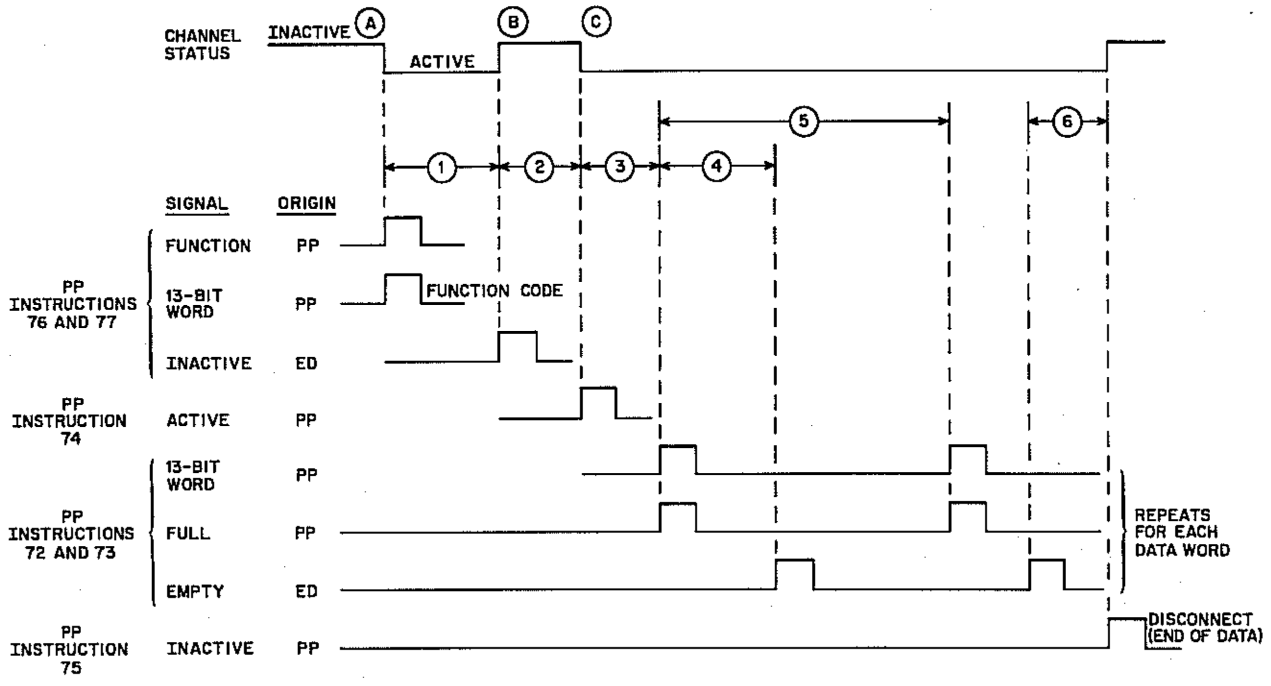
3. The PP sets the channel active flag and sends an active signal to the controller which signals the output equipment that data flow is starting.

4. The PP places a 12-bit data word plus one parity bit in the channel register and sets the full flag. Coincidently, the PP sends a word with parity and a full signal to the controller.

5. The controller accepts the word and sends an empty signal to the PP where the signal clears the channel register and drops the full flag.

6. Steps 4 and 5 repeat for each PP word.

7. After the last word is transferred and acknowledged by the controller empty signal, the PP drops the channel active flag and turns off the controller with an inactive signal.



NOTES:

(1) TIME IS A FUNCTION OF EXTERNAL DEVICE (ED). PP RECOGNIZES INACTIVE 1 MAJOR CYCLE (OR A MULTIPLE OF MAJOR CYCLES) AFTER FUNCTION. THE PP MUST PREVIOUSLY RECEIVE INACTIVE.

(2) TIME IS A FUNCTION OF PERIPHERAL PROCESSOR (PP). MINIMUM TIME IS 1 MINOR CYCLE. ACTUAL TIME IS A FUNCTION OF THE PP PROGRAM.

(3) TIME IS A FUNCTION OF ED.

(4) TIME IS A FUNCTION OF PP. MINIMUM TIME IS 1 MINOR CYCLE. MAXIMUM TIME IS UP TO 4 MINOR CYCLES TO ALLOW OPERATION WITHIN 1 MAJOR CYCLE.

(5) TIME IS A FUNCTION OF PP. MINIMUM TIME IS 2 MAJOR CYCLES. MAXIMUM TIME IS AN INTEGRAL MULTIPLE OF MAJOR CYCLES.

(6) TIME IS A FUNCTION OF ED.

7. MAJOR CYCLE TIME IS 500 NS.

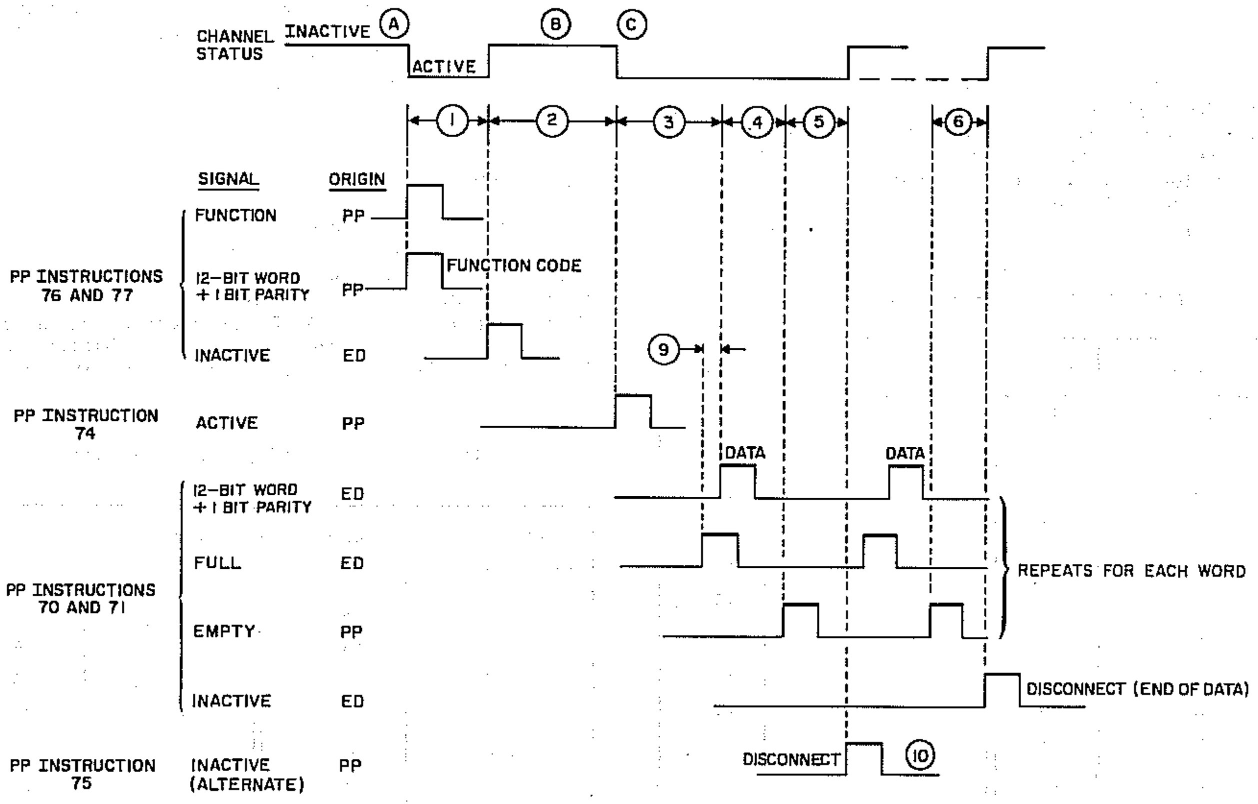
8. MINOR CYCLE IS 50 NS.

(A) CHANNEL MUST BE PREVIOUSLY INACTIVE.

(B) CHANNEL REMAINS ACTIVE UNTIL ED SENDS INACTIVE.

(C) CHANNEL MUST BE PREVIOUSLY INACTIVE.

Figure 5-17. Data Input Sequence Timing



NOTES:

(1) TIME IS A FUNCTION OF EXTERNAL DEVICE (ED). PP RECOGNIZES INACTIVE 1 MAJOR CYCLE (OR A MULTIPLE OF MAJOR CYCLES) AFTER FUNCTION. THE PP MUST PREVIOUSLY RECEIVE INACTIVE.

(2) TIME IS A FUNCTION OF PERIPHERAL PROCESSOR (PP). MINIMUM TIME IS 1 MINOR CYCLE. ACTUAL TIME IS A FUNCTION OF THE PP PROGRAM.

(3) TIME IS A FUNCTION OF ED.

(4) TIME IS A FUNCTION OF PP. MINIMUM TIME IS 1 MINOR CYCLE. MAXIMUM TIME IS UP TO 4 MINOR CYCLES TO ALLOW OPERATION WITHIN 1 MAJOR CYCLE.

(5) TIME IS A FUNCTION OF PP. MINIMUM TIME IS 2 MAJOR CYCLES. MAXIMUM TIME IS AN INTEGRAL MULTIPLE OF MAJOR CYCLES.

(6) TIME IS A FUNCTION OF ED.

7. MAJOR CYCLE TIME IS 500 NS.

8. MINOR CYCLE IS 50 NS.

(9) TIME IS A FUNCTION OF ED. FULL SHOULD PROCEED THE DATA BY A MINIMUM OF 5 NS (15 NS MAXIMUM) TO REMOVE THE CLEAR ON THE INPUT DATA RECEIVERS.

(10) PP MAY DISCONNECT AFTER EMPTY SIGNAL OF ANY ED WORD. STATUS REQUEST DISCONNECTS IN THIS MANNER.

(A) CHANNEL MUST BE PREVIOUSLY INACTIVE.

(B) CHANNEL REMAINS ACTIVE UNTIL ED SENDS INACTIVE.

(C) CHANNEL MUST BE PREVIOUSLY INACTIVE.

Figure 5-18. Data Output Sequence Timing

DISPLAY STATION PROGRAMMING

KEYBOARD

A PP transmits function code 7020_8 to request data from the keyboard of the display station. The PP then activates the input channel and receives one character from the keyboard. This character enters as the lower 6 bits of the word. The upper bits clear. There is no status report by the keyboard. Table 5-12 lists the keyboard character codes.

Table 5-12. Keyboard Character Codes

Character	Code	Character	Code
No data	00	0	33
A	01	1	34
B	02	2	35
C	03	3	36
D	04	4	37
E	05	5	40
F	06	6	41
G	07	7	42
H	10	8	43
I	11	9	44
J	12	+	45
K	13	-	46
L	14	*	47
M	15	/	50
N	16	(	51
O	17	)	52
P	20	Left blank key	53
Q	21	=	54
R	22	Right blank key	55
S	23	,	56
T	24	.	57
U	25	Carriage return	60
V	26	Backspace	61
W	27	Space	62
X	30		
Y	31		
Z	32		

DATA DISPLAY

Data is displayed within an 8- by 8-inch area of a cathode-ray tube (CRT). The display can be in character mode (alphanumeric) or dot mode (graphic). There are 262 144 dot locations arranged in a 512- by 512-dot format. Each dot position is determined by the intersection of X and Y coordinates. The lower left corner dot is octal address X=6000 and Y=7000, and the upper right corner dot is octal address X=6777 and Y=7777.

Character Mode

In character mode, three sizes are provided. Large characters are arranged in a 32- by 32-dot format with 16 characters per line. Medium characters are arranged in a 16- by 16-dot format with 32 characters per line. Small characters are arranged in an 8- by 8-dot format with 64 characters per line. Table 5-13 lists the display character codes.

Table 5-13. Display Character Codes

Character	Code	Character	Code
Space	00	0	33
A	01	1	34
B	02	2	35
C	03	3	36
D	04	4	37
E	05	5	40
F	06	6	41
G	07	7	42
H	10	8	43
I	11	9	44
J	12	+	45
K	13	-	46
L	14	*	47
M	15	/	50
N	16	(	51
O	17	)	52
P	20	Space	53
Q	21	=	54
R	22	Space	55
S	23	,	56
T	24	.	57
U	25		
V	26		
W	27		
X	30		
Y	31		
Z	32		

Dot Mode

In dot mode, display dots are positioned by the X and Y coordinates. The X coordinates position the dots horizontally. The Y coordinates position the dots vertically and unblank the CRT for each dot. Horizontal lines are formed by a series of X and Y coordinates. Vertical lines are formed by a single X coordinate and a series of Y coordinates.

Codes

A single function word is transmitted to select the presentation, mode, and character size (character mode only). Figure 5-19 illustrates the function word format. The word following the function word specifies the starting coordinates for the display (for either mode). Figure 5-20 illustrates the coordinate data word. In character mode, the words that follow are display character codes. Figure 5-21 illustrates the character word.

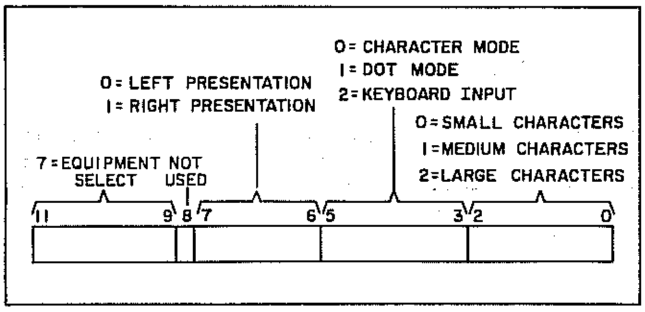


Figure 5-19. Display Station Output Function Code

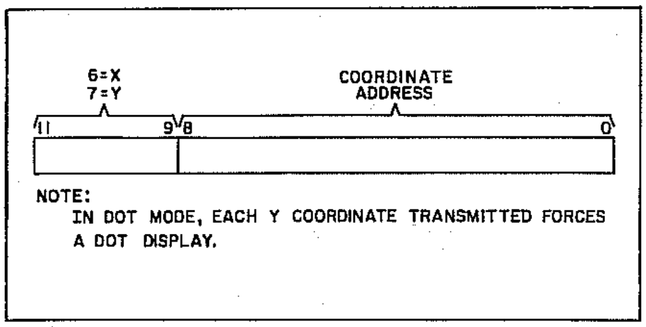

6=X
7=Y
COORDINATE ADDRESS
11
9
8
0
NOTE:
IN DOT MODE, EACH Y COORDINATE TRANSMITTED FORCES A DOT DISPLAY.

Figure 5-20. Coordinate Data Word

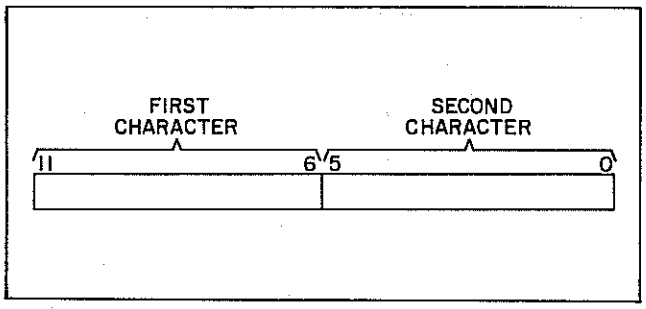


Figure 5-21. Character Data Word

When the display operation has started, the controller regulates character spacing on the line. A new coordinate data word must be sent to start each line. If new coordinates are not specified, data is written on the line specified by the active coordinate word, and information already on that line is overwritten. Character sizes can be mixed by sending a new function word and coordinate word for each size change. Spacing on a line can be varied by sending a coordinate word for the character which is to be spaced differently.

PROGRAMMING EXAMPLE

The following programming example (figure 5-22) requests an input of one line of data from the display station and displays this data on the CRT as it is being typed.

PROGRAMMING TIMING CONSIDERATIONS

When performing an output operation, the computer must wait at the end of the output for a channel empty condition to prevent a loss of coordinates or data. A full jump at the end of the output ensures that the channel is empty and the display controller accepts the last word of the output before disconnecting from the channel.

REAL-TIME CLOCK PROGRAMMING

Channel 14_8 is reserved for the real-time clock. This channel is always active and full and may be read at any time. The real-time clock is a 12-bit free-running counter incrementing at a 1-megahertz rate from 0 through 4095_{10}.

COMMUNICATIONS INTERFACE PROGRAMMING

Channel 15_8 is reserved for communications with one or two external devices through the communications interface. The communcations interface can communicate with all external devices which use EIA standard RS-232 serial interface. The interface can accommodate data with odd/even parity, 5 to 8 bits per character and 1 or 2 stop bits. The format is set by issuing appropriate function codes. The rate is switch selectable for each channel for operation between 110 and 9600 baud. These switches are located internally on the communications interface.

NOTE

Bit numbering is 0 through 63 from left to right.

COMMUNICATIONS INTERFACE OPERATION

The communications interface uses the rightmost 12 bits on channel 15_8. A 12-bit (octal) function word from the PP is translated to specify the following operating conditions.

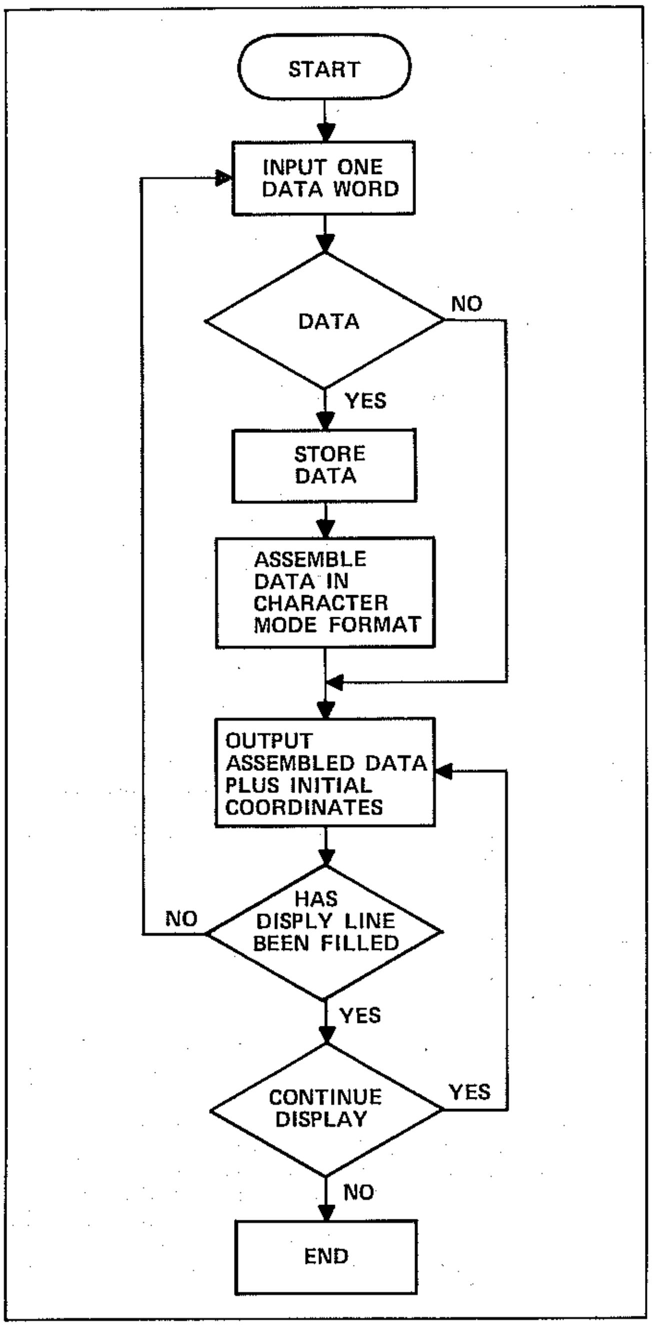

START
INPUT ONE DATA WORD
DATA
NO
YES
STORE DATA
ASSEMBLE DATA IN CHARACTER MODE FORMAT
OUTPUT ASSEMBLED DATA PLUS INITIAL COORDINATES
NO
HAS DISPLY LINE BEEN FILLED
YES
CONTINUE DISPLAY
YES
NO
END

Figure 5-22. Receive and Display Program Flowchart

Code	Function
7XXX	Terminal select.
6XXX	Terminal deselect.
00XX	Read status summary.
01XX	Read terminal data.
02XX	Output to FIFO.
03XX	Set operation mode to terminal.
04XX	Set/clear terminal control signal, data terminal ready (DTR).
05XX	Set/clear terminal control signal, request to send (RTS).
06XX	Not used.
07XX	Master clear selected port.

Terminal Select (7XXX)

The PP sends this select code to specify the terminal to which the function codes and data transmissions apply. Code 7000 selects port 0 (for future use) and code 7001 selects port 1 (maintenance console).

Terminal Deselect (6XXX)

The PP sends this code which deselects the communications interface from channel 15_8 so the 16-bit channel is available for inter-PP communications.

Read Status Summary (00XX)

This code permits the PP to input status from the selected terminal. One word input must follow to read the status response. The response is 12 bits.

Bit	Status
52-58	Not used.
59	Output buffer not full.
60	Input ready.
61	Data carrier detect or carrier on.
62	Data set ready.
63	Ring indication.

PP Read Terminal Data (01XX)

This code permits the PP to input the terminal data from the selected terminal. Channel 15_8 must be activated and a one-word input must follow to read in the terminal data. The data word is 12 bits.

Bit	Status
52	Data set ready.
53	Data set ready and data carrier detector.
54	Over run.
55	Framing or parity error.
56-63	8-bit data.

Data Character (Bits 56 through 63)

The lower 8 bits of the input word form the data character. The communications interface forms this character directly from the Universal Asynchronous Receiver and Transmitter (UART).

Framing or Parity Error (Bit 55)

When the received character does not have a valid stop bit (framing error), or when this bit sets, the received character parity does not agree with the select parity (parity error).

Over Run (Bit 54)

When the previously received character is not read by the PP before the present character is transferred to the data holding register, the over run bit sets.

Data Set Ready (DSR) and Data Carrier Detector (DCD) (Bit 53)

When both data set ready and data carrier detector signals are active, this bit sets.

Data Set Ready (Bit 52)

When the data set ready signal is active, this bit sets.

PP Write Output Buffer (02XX)

This code prepares the communications interface for an output operation to the 64-character FIFO memory. Before an output operation can proceed, channel 15_8 must be activated. The output operation is terminated when the multiplexer receives an inactive signal from the PP, or when no more locations are available in the output buffer. In the latter case, an inactive (instead of empty) signal is sent back to the channel, which in turn terminates the output operations.

Set Operation Mode to the Terminal (03XX)

This code permits the PP to set the terminal operation mode register. A 12-bit function code word from the PP specifies the operation of the terminal. This word is decoded in the function register. Segments of the word define the mode as follows:

Bit	Status
58	Not used.
59	No parity. When this bit is set, it eliminates the parity bit from the transmitted and received characters. The stop bit(s) immediately follow the last data bit.
60	Number of stop bits. This bit selects the number of stop bits, 1 or 2, to be appended immediately after the parity bit. When this bit is clear, it inserts 1 stop bit and when set, it inserts 2 stop bits.
61-62	Number of bits per character. These 2 bits are internally decoded to select 5, 6, 7, or 8 data bits per character.

Bit 61	Bit 62	Bits per Character
0	0	5
0	1	6
1	0	7
1	1	8

Bit	Status
63	Odd/even parity select This bit selects the type of parity which will be appended immediately after the data bits. It also determines the parity that will be checked on read data.

Set/Clear Data Terminal Ready (04XX)

This code permits the PP to set or clear the terminal control signal, data terminal ready (DTR). When bit 63 is set, DTR is active, and when bit 63 is clear, DTR is inactive.

Set/Clear Request to Send (05XX)

This code permits the PP to set or clear the terminal control signal, request to send (RTS). When bit 63 is set, RTS is active, and when bit 63 is clear, RTS is inactive.

Master Clear (07XX)

This code permits the PP to master clear the selected port including its FIFO memory and UART. The terminal operation mode register and terminal control signals are not cleared.

PROGRAMMING CONSIDERATIONS

Channel 15_8 communicates with the terminals connected to the external interface, one at a time. Toestablish communications between a PP and the terminal, the PP issues a function for select. The function word for select is formed by the least significant 12 bits, sent to channel 15_8, and specifies the following information.

- A select code to select the communications interface (7XXX).
- The terminal with which the PP would like to establish communication (7XXX).

When the connect is established, the communications interface routes all data to the terminal designated by the select code. The interface responds with the inactive signal to acknowledge the receipt of the function code of 7XXX for select, 6XXX for deselect, and 0XXX for operation. Otherwise, the function is ignored by the interface.

Output Data

The communications interface accepts a maximum data block length of 64 characters per terminal. During the block data transfer, the interface terminates the output operation either when it receives an inactive signal from the channel or when the output buffer is full. When the output buffer is full, the interface sends back an inactive signal instead of an empty signal to the channel on the last output word. The signal indicates the output buffer accepts the last output word and it cannot receive anymore data from the PP. Output to a full buffer is not allowed by the interface. The multiplexer sends back an inactive signal to deactivate channel 15_8 after the interface decodes the previous function code which is 02XX (PP write output buffer), and receives an activate signal from the PP.

Input Data

The interface does not store the input data from the terminal. A lost data condition exists if the PP does not input the previous data before the new data arrives from the terminal. The multiplexer allows input from an empty input buffer.

Request to Send and Data Terminal Ready

Request to send and data terminal ready are brought up automatically by the hardware under the following conditions regardless of the software RTS and DTR bits.

- Data in the UART output register.
- Data in the FIFO output register.

When no data is in the FIFO or UART, the software bit determines RTS and DTR.

MAINTENANCE REGISTER

Table 5-14 provides a summary of the maintenance register bit information for IOU-0 and IOU-1.

The following list explains table 5-14.

Column	Information
Word No.	Register word listed in octal (8).
Bit No.	Register bit listed in decimal (10) and octal (8).

Description	Name of bit
S/C	Status (S) bits have inputs from various sources in the computer. Control (C) bits have outputs which enable various conditions in the computer.
PRGM FCTN	Indicates which programming functions are applicable to the maintenance register bits and which of the bits are cleared at deadstart. The programming functions are indicated by abbreviations in four categories.

TE	Read, test, clear, test/clear, set, test/set, clear all, and test error. This status bit is included in test error.
R	Read. No other operations can be performed.
D	Read, test, clear, test/ clear, set, test/set, and clear all. This control bit clears at deadstart.
No	Read, test, clear, set, abbreviation, test/set, and clear all.
Channel 36	X indicates the bit is also used in the abbreviated maintenance register of the optional IOU-1.
Display	X indicates that a light-emitting diode displays the bit on a module in IOU-0. When there is an adjacent X in the channel 36 column, the bit is similarly displayed in the optional IOU-1.

In the following maintenance register bit descriptions, the bit names are preceded by their decimal/octal bit numbers. The decimal numbers are for reference only. The octal numbers are for use in programming, setting, clearing, and testing the bits. The bit functions, status or control, follow each bit name.

Table 5-14. Maintenance Register Bit Assignments (Sheet 1 of 9)

Word No. (8)	Bit No. (10)	Bit No. (8)	Description	S/C	PRGM FCTN	Channel 36	Display	Remarks
0	0	0	CM data parity error at IOU.	S	TE	X	X	
	1	1	CM address parity error.	S	TE		X	
	2	2	CP-1 SECDED error.	S	TE		X	
	3	3	SECDED error.	S	TE		X	Loads and locks bits 40 through 52 and 190.
	4	4	CP-1 parity error.	S	TE		X	For future enhancement.
	5	5	CMC parity error.	S	TE		X	Loads and locks bits 054, 55, and 139.
	6	6	Not used.					
	7	7	Not used.					For future enhancement.
	8	10	Not used.					
	9	11	Not used.					
	10	12	Any error bit equals one.	TE		X		Tests 0 through 39 of IOU-1.
	11	13	ECS transfer error.	TE		X		Loads and locks bits 136 through 138.
1	12	14	CP-0 parity error.	S	TE	X	X	
	13	15	CP-1 parity error.	S	TE	X	X	Used only in dual-CP models.
	14	16	PP-0 parity error.	S	TE	X	X	
	15	17	PP-1 parity error.	S	TE	X	X	
	16	20	PP-2 parity error.	S	TE	X	X	
	17	21	PP-3 parity error.	S	TE	X	X	
	18	22	PP-4 parity error.	S	TE	X	X	
	19	23	PP-5 parity error.	S	TE	X	X	
	20	24	PP-6 parity error.	S	TE	X	X	
	21	25	PP-7 parity error.	S	TE	X	X	
	22	26	PP-8 parity error.	S	TE	X	X	
	23	27	PP-9 parity error.	S	TE	X	X	

Table 5-14. Maintenance Register Bit Assignments (Sheet 2 of 9)

Word No. (8)	Bit No. (10)	Bit No. (8)	Description	S/C	PRGM FCTN	Channel 36	Display	Remarks
2	24	30	Channel 0 parity error.	S	TE	X	X	For channel 36, channel numbers 20 through 33 (octal) apply.
	25	31	Channel 1 parity error.	S	TE	X	X	
	26	32	Channel 2 parity error.	S	TE	X	X	
	27	33	Channel 3 parity error.	S	TE	X	X	
	28	34	Channel 4 parity error.	S	TE	X	X	
	29	35	Channel 5 parity error.	S	TE	X	X	
	30	36	Channel 6 parity error.	S	TE	X	X	
	31	37	Channel 7 parity error.	S	TE	X	X	
	32	40	Channel 10 parity error.	S	TE	X	X	
	33	41	Channel 11 parity error.	S	TE	X	X	
	34	42	Channel 12 parity error.	S	TE	X	X	
	35	43	Channel 13 parity error.	S	TE	X	X	
3	36	44	Mains power failure.	S	TE		X	Power/environmental abnormal condition.
	37	45	Shutdown imminent.	S	TE		X	
	38	46	Not used.		TE			For future enhancement.
	39	47	ESM environment failure.	S	TE		X	Environmental abnormal condition.
	40	50	Syndrome bit 0.	S	R		X	Loaded and locked by bit 3 (memory SECDED error).
	41	51	Syndrome bit 1.	S	R		X	
	42	52	Syndrome bit 2.	S	R		X	
	43	53	Syndrome bit 3.	S	R		X	
	44	54	Syndrome bit 4.	S	R		X	
	45	55	Syndrome bit 5.	S	R		X	
	46	56	Syndrome bit 6.	S	R		X	
	47	57	Syndrome bit 7.	S	R		X	

Table 5-14. Maintenance Register Bit Assignments (Sheet 3 of 9)

Word No. (8)	Bit No. (10)	Bit No. (8)	Description	S/C	PRGM FCTN	Channel 36	Display	Remarks
	48	60	Syndrome address bit 0.	S	R		X	
	49	61	Syndrome address bit 1.	S	R		X	
	50	62	Syndrome address bit 2.	S	R		X	Loaded and locked by bit 3.
	51	63	Syndrome address bit 3.	S	R		X	
	52	64	Syndrome address bit 4.	S	R		X	
4	53	65	Syndrome address bit 5.	S	R		X	For future enhancement.
	54	66	Parity error port code bit 0.	S	R		X	From CMC; identifies port; loaded and locked by bit 5.
	55	67	Parity error port code bit 1.	S	R		X	
	56	70	Breakpoint port code bit 0.	S	R		X	
	57	71	Breakpoint port code bit 1.	S	R		X	Loaded and locked by bit 77.
	58	72	Breakpoint function code bit 0.	S	R		X	
	59	73	Breakpoint function code bit 1.	S	R		X	
	60	74	P input bit 0.	S	R	X	X	If bit 83 clears, bits 60 through 71 display P of the PP selected by bits 120 through 123, and bits 72 through 75 display selected PP.
	61	75	P input bit 1.	S	R	X	X	
	62	76	P input bit 2.	S	R	X	X	
	63	77	P input bit 3.	S	R	X	X	
	64	100	P input bit 4.	S	R	X	X	If bit 83 sets, the content of P register is latched and retained on every CM breakpoint bit.
5	65	101	P input bit 5.	S	R	X	X	
	66	102	P input bit 6.	S	R	X	X	If bit 76 sets when bit 83 sets, bits 60 through 75 hold until bit 76 clears. Refer to text for more detailed information.
	67	103	P input bit 7.	S	R	X	X	
	68	104	P input bit 8.	S	R	X	X	
	69	105	P input bit 9.	S	R	X	X	
	70	106	P input bit 10.	S	R	X	X	
	71	107	P input bit 11.	S	R	X	X	
								If bit 76 sets when bit 83 sets, bits 60 through 75 hold until bit 76 clears. Refer to text for more detailed information.

Table 5-14. Maintenance Register Bit Assignments (Sheet 4 of 9)

Word No. (8)	Bit No. (10)	Bit No. (8)	Description	S/C	PRGM FCTN	Channel 36	Display	Remarks
6	72	110	PP identification bit 0.	S	R	X	X	Refer to remarks for bits 60 through 71.
	73	111	PP identification bit 1.	S	R	X	X	
	74	112	PP identification bit 2.	S	R	X	X	
	75	113	PP identification bit 3.	S	R	X	X	
	76	114	IOU breakpoint bit.	S		X	X	
	77	115	CMC breakpoint match.	S		X	X	Loads and locks bits 56 through 59.
	78	116	Clear CM busy.					
	79	117	CP-1 breakpoint.	S				
	80	120	Force zero parity on channels.	C	D	X		
	81	121	Force zero parity on PPM.	C	D	X		
	82	122	Not used.					For future enhancement.
	83	123	IOU breakpoint mode select.	C	D	X		Refer to remarks for bits 60 through 75.
7	84	124	All PPs 500-nano-second major cycle.	S				
	85	125	Inhibit IOU request to CMC.	C	D	X	X	
	86	126	Narrow clock width margin.	C				
	87	127	Wide clock width margin.	C				
	88	130	Diagnostic aid.	S			X	
	89	131	Diagnostic aid.	S			X	
	90	132	Diagnostic aid.	S			X	
	91	133	Diagnostic aid.	S			X	
	92	134	Diagnostic aid.	S			X	
	93	135	Micro record counter maintenance mode.	C				For future enhancement.
	94	136	Stop on error.	C	D	X	X	
	95	137	Stop on PPM parity error.	C	D	X	X	Applies to all PPs.

Table 5-14. Maintenance Register Bit Assignments (Sheet 5 of 9)

Word No. (8)	Bit No. (10)	Bit No. (8)	Description	S/C	PRGM FCTN	Channel 36	Display	Remarks
10	96	140	Breakpoint address bit 0.	C				Absolute 18-bit address bits 96 through 113 are sent to CMC to establish breakpoint address when bits 116 and/ or 117 are set.
	97	141	Breakpoint address bit 1.	C				
	98	142	Breakpoint address bit 2.	C				
	99	143	Breakpoint address bit 3.	C				
	100	144	Breakpoint address bit 4.	C				
	101	145	Breakpoint address bit 5.	C				
	102	146	Breakpoint address bit 6.	C				
	103	147	Breakpoint address bit 7.	C				
	104	150	Breakpoint address bit 8.	C				
	105	151	Breakpoint address bit 9.	C				
	106	152	Breakpoint address bit 10.	C				
	107	153	Breakpoint address bit 11.	C				
11	108	154	Breakpoint address bit 12.	C				
	109	155	Breakpoint address bit 13.	C				
	110	156	Breakpoint address bit 14.	C				
	111	157	Breakpoint address bit 15.	C				
	112	160	Breakpoint address bit 16.	C				
	113	161	Breakpoint address bit 17.	C				
	114	162	Breakpoint condition code bit 18.	C				Select function RD/WT/ RNI or all three to CMC for port selection.
	115	163	Breakpoint condition code bit 19.	C				
	116	164	Breakpoint condition code bit 20.	C				
	117	165	Breakpoint condition code bit 21.	C				
	118	166	Inhibit single-error report.	C			X	Single errors are not recorded in maintenance register when set.
	119	167	CM read double error.	S	D	X		

Table 5-14. Maintenance Register Bit Assignments (Sheet 6 of 9)

Word No. (8)	Bit No. (10)	Bit No. (8)	Description	S/C	PRGM FCTN	Channel 36	Display	Remarks
12	120	170	PP select code bit 0.	C	D	X	X	Select 1 of 10 PPs for forced exit, deadstart, or display.
	121	171	PP select code bit 1.	C	D	X	X	
	122	172	PP select code bit 1.	C	D	X	X	
	123	173	PP select code bit 3.	C	D	X	X	
	124	174	PP select auto/manual mode.	C	D	X	X	Clear equals manual.
	125	175	Force exit on selected PP.	C	D	X		
	126	176	Force deadstart on selected PP.	C	D	X		Set forces deadstart. PP remains in deadstart condition until bit clears.
	127	177	Master clear.	C	D			
	128	200	Force zero SECDED code and parity CMC to CM.	C				
	129	201	Force zero address parity CMC to CM.	C				
	130	202	Disable address parity error.	C				Model 865 CM only.
	131	203	Not used.					For future enhancement.
13	132	204	Force zero parity code 0.	C				Storage move unit.
	133	205	Force zero parity code 1.	C				
	134	206	Not used.					Loaded and locked by bit 11.
	135	207	Not used.					
	136	210	ECS transfer error code bit 0.	S	R			
	137	211	ECS transfer error code bit 1.	S	R			
	138	212	ECS transfer error code bit 2.	S	R			
	139	213	CMC address/data parity error.	S	R		X	Loaded and locked by bit 5. Clear equals data error.
	140	214	Not used.					
	141	215	Clock frequency margin 0.	C	D			Bits 141 through 143 are code bits for selecting clock margins. For bit 143, clear equals slow.
	142	216	Clock frequency margin 1.	C	D			
	143	217	Clock frequency slow/ fast.	C	D			

Table 5-14. Maintenance Register Bit Assignments (Sheet 7 of 9)

Word No. (8)	Bit No. (10)	Bit No. (8)	Description	S/C	PRGM FCTN	Channel 36	Display	Remarks
14	144	220	Not used.	S			X	
	145	221	Not used.	S			X	
	146	222	Not used.	S			X	
	147	223	Not used.	S			X	
	148	224	Not used.	S			X	
	149	225	Not used.	S			X	
	150	226	Not used.	S			X	Clear equals low.
	151	227	Not used.	S			X	Clear equals one.
	152	230	Clock margin width narrow.	C			X	
	153	231	Clock margin width wide.	C			X	
	154	232	Select hi/lo RVM.	C				Clear equals low.
	155	233	Select all/one RVM.	C				Clear equals one (refer to text).
15	156	234	RVM quadrant 0 select.	C				Used with bits 154 and 155.
	157	235	RVM quadrant 1 select.	C				
	158	236	RVM quadrant 2 select.	C				
	159	237	RVM quadrant 3 select.	C				
	160	240	RVM quadrant 4 select.	C				
	161	241	RVM quadrant 5 select.	C				
	162	242	RVM quadrant 6 select.	C				
	163	243	RVM quadrant 7 select.	C				
	164	244	RVM quadrant 8 select.	C				
	165	245	RVM quadrant 9 select.	C				
	166	246	RVM quadrant 10 select.	C				
	167	247	RVM quadrant 11 select.	C				

Table 5-14. Maintenance Register Bit Assignments (Sheet 8 of 9)

Word No. (8)	Bit No. (10)	Bit No. (8)	Description	S/C	PRGM FCTN	Channel 36	Display	Remarks
16	168	250	RVM module address bit 0.	C				
	169	251	RVM module address bit 1.	C				
	170	252	RVM module address bit 2.	C				
	171	253	RVM module address bit 3.	C				
	172	254	RVM module address bit 4.	C				
	173	255	RVM module address bit 5.	C				
	174	256	IOU to CMC zero address parity.	C		X		
	175	257	IOU to CMC zero data parity.	C		X		
	176	260	CM configuration.	S				
	176	260	CM configuration 0.	S				For future enhancement.
	177	261	CM configuration 1.	S				For future enhancement.
	178	262	CM configuration 2.	S				For future enhancement.
	179	263	CM configuration 3.	S				For future enhancement.
17	180	264	Not used.					For future enhancement.
	181	265	Not used.					For future enhancement.
	182	266	Not used.					For future enhancement.
	183	267	Double error.	S			X	
	184	270	Not used.					For future enhancement.
	185	271	Not used.	C				
	186	272	Not used.					For future enhancement.
	187	273	Not used.	C				
	188	274	Software flag 0.	C		X		Diagnostic aids.
	189	275	Software flag 1.	C		X		
	190	276	Syndrome address bit 18.				X	
	191	277	Syndrome address bit 19.					For future enhancement.

Table 5-14. Maintenance Register Bit Assignments (Sheet 9 of 9)

Word No. (8)	Bit No. (10)	Bit No. (8)	Description	S/C	PRGM FCTN	Channel 36	Display	Remarks
	192	300	CP-0 stopped.	S	R		X	
	193	301	CP-1 stopped.	S	R		X	Used only in dual-CP models.
	194	302	ECS in progress flag.	S	R		X	
	195	303	Monitor flag CP-0.	S	R		X	
	196	304	Monitor flag CP-1.	S	R		X	Used only in dual-CP models.
20	197	305	PPM select bit 0.	S	R	X		
	198	306	PPM select bit 1.	S	R	X		
	199	307	PPM select bit 2.	S	R	X		
	200	310	PPM select bit 3.	S	R	X		
	201	311	External channel select.	S	R			IOU select.
	202	312	CP-0 standard addressing mode.					
	203	313	CP-1 standard addressing mode.					

BIT 0/0 CM DATA PARITY ERROR AT IOU — STATUS

This bit indicates a parity error on data transmitted from CMC to the IOU. The bit also indicates a CM parity error on data to the IOU during the parity mode operation. The bit may set at the same time a SECDED double error is indicated (bit 3).

BIT 1/1_8 CM ADDRESS PARITY ERROR — STATUS

This bit indicates a parity error on the address transmitted from CMC to CM.

BIT 2/2_8 CPU SECDED ERROR — STATUS

This bit indicates that a single-error correction or a double-error detection has occurred. Bit 2 indicates that the SECDED error was detected by the CMC in CP-1.

BIT 3/3_8 SECDED ERROR — STATUS

This bit indicates that a single-error correction or a double-error detection has occurred. The bit also locks bits 40 through 53, 190, and 191. These bits are unlocked but not reset by software to detect further SECDED status. Bit 183 identifies the SECDED error as a single error when cleared or a double error when set. Bit 118, if set, inhibits bit 3 for single errors but not for double errors.

BIT 4/4_8 CP-1 PARITY ERROR

This bit indicates that an address or data transmission parity error was detected by the CMC in CP-1.

BIT 5/5_8 CMC PARITY ERROR — STATUS

This bit indicates that an address or data transmission parity error has been received by CMC. The bit is used in conjunction with bits 54, 55, and 139. The bit locks bits 54, 55, and 139 so that their status cannot be modified until bit 5 clears. Bit 5 must be reset by software to detect further CMC parity errors.

BIT 6/6_8 THROUGH 9/11_8 — NOT USED

BIT 10/12_8 ANY ERROR BIT EQUALS ONE — STATUS

This bit indicates that one or more status and control register bits 0 through 39 in IOU-1 are set.

BIT 11/13_8 ECS TRANSFER ERROR — STATUS

This bit indicates that an error occurred on an ECS transfer. The type of error is indicated by the status locked in bits 136, 137, and 138 by bit 11. Bit 11 must be reset to detect further errors.

BIT 12/14_8 CP-0 P-REGISTER PARITY ERROR — STATUS

This bit indicates that the IOU detected a parity error on a read of the P register for CP-0.

BIT 13/15_8 CP-1 P-REGISTER PARITY ERROR — STATUS

This bit applies only to models with two CPs and is unused in models with one CP. The bit indicates that the IOU detected a parity error on a read of the P register for CP-1.

BITS 14/16_8 THROUGH 23/27_8 PP-0 THROUGH PP-9 PARITY ERROR — STATUS

These bits indicate the occurrence of a PP error condition. The bits prevent a PP from executing instructions following the detection of the error condition. On a one-to-one basis, the bits indicate the status of each PP. The bits indicate the logical PP numbers and are not affected by a reconfiguration of the PPMs that results from resetting the IOU and PP MEMORY SELECT switches on the deadstart panel.

The error conditions which can stop the PPs and their associated status and control register bits are:

Error Condition	Bit 0	Bit 119
PPM parity error	0	0
Read pyramid parity error on a CM read	1	0
Double SECDED error on a CM read	0	1

The PP associated with the error stops only if the appropriate enable bits are set in the status and control register. If the enable bits are not set, the error condition is reported but the PP is not stopped.

BITS 24/30_8 THROUGH 35/43_8 CHANNELS 0 THROUGH 13 (PPS-0) [20 THROUGH 33 (PPS-1)] PARITY ERROR — STATUS

These bits indicate the occurrence of a parity error in the corresponding I/O channel. Each bit indicates the status of one channel as listed in table 5-14. The checking of these bits may be disabled on any or all of the I/O channels with the PARITY switches on the I/O connector panel.

BIT 36/44_8 MAINS POWER FAILURE — STATUS

This bit indicates that the primary power mains feeding the computer system are deenergized and have remained so for more than one-half cycle (8.3 milliseconds for 60-Hz power and 10.0 milliseconds for 50-Hz power) of the mains frequency. If power returns within one cycle of the mains frequency, the line feeding the bit automatically goes false.

BIT 37/45$_8$ SHUTDOWN IMMINENT — STATUS

This bit indicates one of the following conditions.

- The primary power mains feeding the system are deenergized and have remained so for at least 100 milliseconds. Power probably will not return to normal within the regulation range of the system secondary power supply, normally a motor-generator set.
- An environmental condition (including dewpoint warning and chassis temperature) is abnormal and approaching an emergency power shutdown.
- An environmental condition is changing at an abnormally high rate.
- An environmental condition is about to execute a controlled power shutdown.
- A critical system device is down because of environmental conditions. (This indication exists only if the system has monitoring provisions for the device.)

If power and environmental conditions return to normal, except in the case of an emergency shutdown limit, the line feeding the bit automatically goes false within one cycle of the mains frequency. The bit must be cleared by software.

When both the mains power failure and power shutdown imminent bits are set, one of the following coincident conditions exists.

- A power mains failure has occurred for longer than 100 milliseconds. Power will probably not return within the regulation range of the system secondary power supplies. The kernel system (CP, all PPs, all channels, store, all first-level controllers, and all system disk units) remains available for processing for the balance of the motor-generator ride-through after the shutdown imminent bit sets. In this case, the mains power failure bit sets at least 50 milliseconds before the shutdown imminent bit sets. However, all peripheral equipment powered directly from the mains has probably failed.
- A controlled shutdown limit has been reached. The limit sensor has disconnected the primary power mains from the system secondary power supply, and the kernel system processing remains available for the balance of the motor-generator ride-through after the shutdown imminent bit sets. In this case, the mains power failure and the shutdown imminent bits set at approximately the same time.

Examples of possible conditions are:

	Condition	Explanation
1.	Mains power failure only; power returns.	Indicates that all peripheral equipment powered directly from the mains has probably failed. The system is not down, but user intervention is necessary to restore power to affected peripherals.
2.	Mains power failure and shutdown imminent.	Indicates the system will probably terminate and require restart.
3.	Mains power failure and shutdown imminent, mains power failure bit clears, or the mains power failure and shutdown imminent bits clear.	The explanation for condition 1 applies. This is a rare occurrence and may not be a stable condition.
4.	Shutdown imminent, no mains power failure.	Either a shutdown timeout (1 to 2 minutes) is in progress because of an environmental problem, or a warning level has been reached which ultimately requires user intervention. Sufficient time may exist for the user or software, if software capability exists, to initiate and complete system checkpoints. If the mains power failure bit 36 sets later, a timeout has been completed and the system behaves as though an emergency shutdown limit was reached.

BITS 38/46$_8$ — NOT USED

BIT 39/47$_8$ ESM ENVIRONMENT FAILURE WARNING — STATUS

This bit indicates that loss of the ESM system is imminent due to a malfunction in one of the environmental conditions monitored by the ESM controller.

BITS 40/50$_8$ THROUGH 47/57$_8$ SYNDROME BITS 0 THROUGH 7 — STATUS

These bits and bits 48 through 52, 190, and 191 are provided by CMC upon detection of a SECDED error. Bits 40 through 47 provide the information needed to isolate a single-error failure to a particular memory module. Setting bit 3 locks bits 40 through 47. Clearing bit 3 unlocks bits 40 through 47 but does not clear them. Software functions cannot clear or set the read-only syndrome bits.

BITS 48/60_8 THROUGH 53/65_8 SYNDROME ADDRESS BITS 0, 1, 2, 16, AND 17 — STATUS

These bits and bits 40 through 47 and 190 are provided by CMC upon detection of a SECDED error. Bits 48, 49, and 50 indicate the CM bank number (model 865 only). Bits 51 and 52 indicate the CM quadrant. Bits 48 through 52 are locked by the setting of bit 3. Clearing bit 3 unlocks bits 48 through 52 but does not clear them. Bits 48 through 52 are read-only bits that cannot be cleared or set by software functions.

BIT 53/65_8 — NOT USED

BITS 54/66_8 AND 55/67_8 PARITY ERROR PORT CODE BITS 0, 1 — STATUS

These bits indicate which CMC port of CP-0 had a parity error. The bits are locked by the setting of bit 5 and cannot be modified until bit 5 clears.

Bit 55	Bit 54	Port
1	0	IOU-1
1	1	IOU-0

BITS 56/70_8 AND 57/71_8 BREAKPOINT PORT CODE BITS 0, 1 — STATUS

These bits indicate the CMC in either CP-0 or CP-1 has sensed a breakpoint condition. The bits are locked by the setting of bit 77 and cannot be cleared until bit 77 clears.

Bit 57	Bit 56	Port
0	0	CP-1 (models with two CPs)
0	1	CP-0
1	0	IOU-1
1	1	IOU-0

These bits and bits 40 through 47, 190 and 191 are provided by CMC upon detection of SECDED error. The bits are defined as follows:

Model 865	Bit 48, 49, 50	CM bank, address 0,1,2
	Bit 5	Chip select, address 3
	Bit 52,53	CM quadrant, address 4,5
	Bit 190,191	Address 18,19
Model 875	Bit 48,49,50,51	CM bank address 0,1,2,3
	Bit 52,50	CM port address 4,5
	Bit 190, 191	Chip select, address 18,19

BITS 58/72_8 AND 59/73_8 BREAKPOINT FUNCTION CODE BITS 0, 1 — STATUS

These bits indicate what type of instruction caused the breakpoint condition to be satisfied. The bits are locked by the setting of bit 77 and cannot be modified until bit 77 clears.

Bit 59	Bit 58	Type
0	0	Read
0	1	Write
1	0	RNI
1	1	This condition does not occur.

BITS 60/74_8 THROUGH 71/107_8 P INPUT BITS 0 THROUGH 11 — STATUS

These bits indicate the content of the PP P-register. The content can be the program address or data buffer address for the PP that satisfied the breakpoint condition when bits 76 and 83 are set. When bit 83 is not set, the bits display the P register of the selected PP. The PP selection can be made manually by switches on the IOU module located at J40 or through software selection of control bits 120 through 124. Bits 60 through 71 are locked by the setting of bit 76 and cannot be modified until bit 76 clears.

BITS 72/110_8 THROUGH 75/113_8 SCANNER CHANNEL SELECT BITS 0 THROUGH 3 — STATUS

These bits indicate which PP stored the content of its P register in bit positions 60 through 71. Bits 72 through 75 are locked by the setting of bit 76 and cannot be modified until bit 76 clears. Bit 83 is associated with bits 72 through 75.

BIT 76/114_8 PPS BREAKPOINT BIT — STATUS

This bit, with bit 83 set, indicates that the breakpoint address was referenced by IOU. The content of the P register of the referencing PP is locked into bit positions 60 through 71. The referencing PP code is also locked into bit positions 72 through 75. These bits are locked so that their status cannot be modified until bit 76 is cleared. Bit 76 must be reset by software to detect further PPS breakpoint addresses. With bit 83 clear, the content of the P register of the PP selected by bits 120 through 124 is made available for monitoring by bits 60 through 71. The P register status is not locked but continually tracks the program address of the selected PP.

BIT 77/115$_8$ CMC BREAKPOINT MATCH — STATUS

This bit indicates that the breakpoint condition occurred in either CP-0 or CP-1. The breakpoint condition is defined by the absolute address (located in bits 96 through 113) and the breakpoint condition code (located in bits 114 through 117). It also locks bits 56 through 59 so that their status cannot be modified until bit 77 clears. Bit 77 must be reset by software to detect further breakpoint conditions.

BIT 78/116$_8$ CLEAR CENTRAL MEMORY BUSY — CONTROL

This bit clears CM busy and unhangs a PP on an unanswered CM request. The bit causes a one-shot operation. The bit must be cleared by software and set again to execute its function a second time.

BIT 79/117$_8$ CP-1 BREAKPOINT — STATUS

This bit indicates that the breakpoint condition described by bit 77 and bits 56 through 59 apply to CP-1.

BIT 80/120$_8$ FORCE ZERO PARITY ON CHANNELS — CONTROL

This bit forces the data parity bits in the I/O channels to zero. A deadstart clears the bit.

BITS 81/121$_8$ FORCE ZERO PARITY ON PP MEMORIES — CONTROL

This bit forces the PPM parity bits to zero. A deadstart clears the bit.

BIT 82/122$_8$ — NOT USED

BIT 83/123$_8$ IOU BREAKPOINT MODE SELECT — CONTROL

This bit, when set, forces the P register field (bits 60 through 75) into breakpoint mode. When clear, it forces the P register field into program address display mode. The breakpoint field is locked by the setting of bit 76. A deadstart clears the bit.

BIT 84/124$_8$ ALL PPS 500 – NANOSECOND MAJOR CYCLE — STATUS

This bit is constantly set to indicate a major cycle time of 500 nanoseconds for all PPs in IOU-0 and IOU-1. The bit cannot be cleared.

BIT 85/125$_8$ INHIBIT IOU REQUEST TO CMC — CONTROL

This bit prevents any PP from making a read/write/exchange request. This bit should be used with the CP master clear bit 127 to ensure that the master clear does not hang any PP that is accessing CM. A deadstart clears the bit.

BITS 86/126$_8$ AND 87/127$_8$ NARROW AND WIDE CLOCK WIDTH MARGINS — CONTROL

These bits control the clock pulse width in the PPS chassis according to the following bit translations.

Bit 87	Bit 86	Clock Pulse
0	0	Normal
0	1	Narrow
1	0	Wide
1	1	Normal

BITS 88/138$_8$ THROUGH 93/135$_8$ DIAGNOSTIC AIDS — STATUS

Refer to diagnostic in use.

BIT 93/135$_8$ MICROSECOND COUNTER MAINTENANCE — CONTROL

This bit (when set) causes the CPU microsecond counter to count at a 50-nanosecond rate instead of a microsecond rate. The bit also causes the counter to operate as two 16-bit counters running in parallel instead of a 32-bit counter to allow testing of the upper 16 bits.

BIT 94/136$_8$ STOP ON ERROR — CONTROL

This bit (when set) enables the stop on a CM read error which may be either a double error or a pyramid parity error. The PP associated with the error stops after disassembling the word received from CM. The data with the error is written into the PPM. In case of a block read, the instruction terminates. Bits 14 through 23 are used to identify the PP with the error condition(s). Bit 94 provides control only in its respective status and control register which is for IOU-0 or IOU-1.

BIT 95/137$_8$ STOP ON PPM PARITY ERROR — CONTROL

This bit (when set) enables the stop on PPM parity error network. When a parity error is detected, any instruction except a CM read or write, exchange jump, or channel executes. Following the completion of the instruction, the PP stops.

When a parity error is detected on a channel instruction, the channel select control disables, preventing the instruction from performing any channel operation. The instruction may or may not exit.

When a parity error is detected in a 20-PP system and a PP in one chassis is making a request for a channel in the other chassis, the request to the other chassis is blocked. The PP with the parity error hangs in the instruction it is trying to execute.

When a parity error is detected on a CM write or exchange jump instruction, requests to the control of the CM are blocked. A single-word write and exchange exits, and the block write instruction terminates. In all cases, the PP stops prior to writing incorrect data in CM or executing an exchange jump. The instruction is allowed to exit, preventing write pyramid hang-ups.

When a parity error is detected on a CM read instruction, the request is sent and the PP writes the data into PPM. In the case of a block read, the instruction terminates and the PP always stops.

BITS 96/140$_8$ THROUGH 113/161$_8$ BREAKPOINT ADDRESS BITS 0 THROUGH 17 — CONTROL

These bits, along with bits 134 and 135, define the absolute address to be used for the breakpoint condition, defined by bits 114 through 117. Bits 96 through 113 are sent to CMC and compared with all addresses being accessed. These address bits apply to both CP-0 and optional CP-1.

BITS 114/162$_8$ THROUGH 117/165$_8$ BREAKPOINT CONDITION CODE BITS 18 THROUGH 21 — CONTROL

These bits define the breakpoint conditions, and apply to both CP-0 and optional CP-1.

Bit 117	Bit 116	Bit 115	Bit 114	Condition
X	X	0	0	Read.
X	X	0	1	Write.
X	X	1	0	RNI.
X	X	1	1	Any of the above.
0	0	X	X	Disabled.
0	1	X	X	Enabled for IOU.
1	0	X	X	Enabled for CP.
1	1	X	X	Enabled for IOU or CP.

BIT 118/166$_8$ INHIBIT SINGLE-ERROR REPORT — CONTROL

This bit, when set, stops the recording of single-error status information and blocks setting bit 3 of the status and control register if a single error occurs. Double errors continue to set bit 3 and be reported by bit 183.

BIT 119/167$_8$ CM READ DOUBLE ERROR — STATUS

This bit indicates a double SECDED error on a CM read within the PP chassis. A PP does not force exit (bit 125) if the hung condition resulted from a read pyramid parity error, a PPM parity error, or double SECDED error. A mainframe deadstart or a forced deadstart (bit 126) to the hung PP is the only way to clear a PP that was hung by one of these conditions. Bit 119 functions in conjunction with bits 14 through 23, 94, and 95.

BITS 120/170$_8$ THROUGH 123/173$_8$ PP SELECT CODE BITS 0 THROUGH 3 — CONTROL

These bits determine which PP is forced to exit (bit 125), deadstart (bit 126), or display (when bit 83 is clear) its P register. A deadstart clears bits 120 through 123.

BIT 124/174$_8$ PP SELECT AUTO/MANUAL MODE — CONTROL

This bit selects the mode of PP selection. When set, PP selection is under program control. PP selection is then made by bits 120 through 123. When clear, selection is manual, and the PP selection is made by switches on the PP chassis at locations 2D33 (IOU-0) and 2P34 (IOU-1). A deadstart clears the bit.

BIT 125/175$_8$ FORCE EXIT ON SELECTED PP — CONTROL

This bit clears a selected hung PP (selected by bits 120 through 124) by forcing an instruction exit except in the manual mode. The PP resumes operation at its next slot time at P plus 1. A forced instruction exit occurs once each time bit 125 sets. The bit causes a one-shot operation. The bit must be cleared by software and set again to cause a second exit. A deadstart clears the bit.

BIT 126/176$_8$ FORCE DEADSTART ON SELECTED PP—CONTROL

This bit, along with control bits 120 through 124, provides a programmable capability to make individual PP deadstarts. Bits 120 through 124 select the PP, and bit 126 forces the selected PP into a deadstart input condition. The selected PP then goes through the same deadstart sequence as would occur under a hardware-controlled deadstart. The PP is set up for a 71XX instruction, where XX is the selected PP number. This instruction causes the PP to attempt an input on its own channel. The software must first ensure that the selected channel is empty and active at the time of the deadstart. No other I/O operation can be in process on the channel. The master clear signal to the channel is inhibited. The selected PP remains in the deadstart condition until bit 126 clears. A system deadstart clears bit 126.

Bit 126 causes the IOU to hang when the selected PP is performing a CM read or write operation at the time of deadstart.

BIT 127/177$_8$ MASTER CLEAR — CONTROL

This bit, when set, sends a master clear to the CM, CMC, and CP chassis (two CP chassis for some models). The bit is ORed with the deadstart signal. The bit or the deadstart signal causes a master clear. The bit holds the CMC and CP chassis in a cleared state as long as it is set. The PP chassis is not affected by this bit, unless a PP is making a CM reference. To avoid hanging any PP, bit 85 should be set before bit 127. A deadstart clears bit 127.

BIT 128/200$_8$ FORCE ZERO SECDED CODE AND PARITY CMC TO CM — CONTROL

This bit forces the CMC to put a zero check code and parity bit on data being sent to CM. It also forces CMC to put a zero parity bit on data transmitted to a requesting unit such as ECS/EEM.

BIT 129/201$_8$ FORCE ZERO ADDRESS PARITY CMC TO CM — CONTROL

This bit forces CMC to put a zero parity bit on the address being sent to CM.

BIT 130/202$_8$ DISABLE ADDRESS PARITY ERROR — CONTROL

This bit disables address parity error detection at CM. This prevents a condition where reads or writes are inhibited during the presence of any address parity error.

BIT 131/203$_8$ — NOT USED

BITS 132/204$_8$ AND 133/205$_8$ FORCE ZERO PARITY CODE 0 AND CODE 1 — CONTROL

These bits force a zero parity bit on the following transmission paths.

Bit 133	Bit 132	Transmission Path
0	0	Normal parity.
0	1	Not used.
1	0	Address from SMU to ECS controller.
1	1	Data from ECS to CMC.

Parity does not exist from CP-1 to ECS.

BIT 134/206$_8$ AND 135/207$_8$ — NOT USED

BIT 136/210$_8$ THROUGH 138/212$_8$ ECS TRANSFER ERROR CODE BITS 0 THROUGH 2 — STATUS

These bits indicate errors that occur during an ECS transfer. The following list gives the status-bit code that states where the error occurred. The bits are locked by the setting of bit 11.

Bit 138	Bit 137	Bit 136	Status
0	0	0	Not used.
0	0	1	Not used.
0	1	0	CMC double error.
0	1	1	CMC to CM address parity error.
1	0	0	CMC data input parity error.
1	0	1	ECS bank parity error.
1	1	0	ECS controller data parity error.
1	1	1	ECS controller address parity error (this indicates no error when bit 11 is clear).

BIT 139/213$_8$ CMC ADDRESS/DATA PARITY ERROR — STATUS

This bit indicates an address parity error in CMC. The bit is used with bits 5, 54, and 55. If the bit clears and bit 5 sets, the CMC parity error is a data error. Bit 139 is locked by the setting of bit 5 and cannot be modified until bit 5 clears.

BIT 140/214$_8$ — NOT USED

BIT 141/215$_8$ THROUGH 143/217$_8$ CLOCK FREQUENCY MARGINS 0, 1, AND SLOW/FAST — CONTROL

These bits are used in maintenance operations. The bits form a 3-bit code that sets the frequency margins of the basic 40-MHz clock. A 20-MHz clock and a 10-MHz clock originate from the basic clock and change frequency margins by the same percentage as the basic clock. A deadstart clears these bits.

The following 3-bit code translations are for programming use. The codes result in the margin conditions listed after the codes. For example, code 000 results in the margin condition normal, code 001 results in condition slow 1, and so on.

Bit 143	Bit 142	Bit 141
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Margin Condition	40-MHz Clock	20-MHz Clock	10-MHz Clock
Normal	40.000	20.000	10.000
Slow 1	39.375	19.688	9.844
Slow 2	38.750	19.375	9.688
Slow 3	38.125	19.063	9.531
Normal	40.000	20.000	10.000
Fast 1	40.625	20.313	10.156
Fast 2	41.250	20.625	10.313
Fast 3	41.875	20.938	10.469

BITS 144/220$_8$ THROUGH 149/225$_8$ — NOT USED

BITS 150/226$_8$ THROUGH 151/227$_8$ — NOT USED

BITS 152/230$_8$ AND 153/231$_8$ CLOCK MARGIN WIDTH NARROW AND WIDE — CONTROL

These bits control the clock pulse width in the CP according to the following bit translations.

Bit 153	Bit 152	Clock Pulse
0	0	Normal
0	1	Narrow
1	0	Wide
1	1	Wide

The 152 and 153 bit outputs are in parallel with the CLOCK PULSE switch on CP chassis 5. The CLOCK PULSE switch must be in the normal position (middle) to permit clock pulse margin control from the status and control register. In the narrow or wide position, the CLOCK PULSE switch overrides the status and control register clock pulse width bits.

BIT 154/232$_8$ SELECT HI/LO REFERENCE VOLTAGE MARGINS — CONTROL

When set, this bit selects the high RVM for the CP modules selected by bits 155 through 173. When clear, the bit selects the low RVM for the selected modules. If bits 156 through 167 do not reference a CP chassis quadrant, bit 154 has no effect (figure 2-7).

BIT 155/233$_8$ SELECT ALL/ONE REFERENCE VOLTAGE MARGINS — CONTROL

When this bit sets and bits 163 through 173 set, the RVM for all CP modules within the quadrants selected by bits 156 through 167 are simultaneously selected. When clear, bit 155 permits RVM to be applied to individual modules within the quadrants selected by bits 156 through 173. If bits 156 through 167 do not reference a CP chassis quadrant, bit 155 has no effect (figure 2-7).

BIT 156/234$_8$ THROUGH 167/247$_8$ REFERENCE VOLTAGE MARGINS QUADRANT 0 THROUGH 11 SELECT — CONTROL

These bits determine, on a one-to-one basis, which quadrant(s) of a CP chassis receives an RVM (figure 5-23). For example, select 3 selects quadrant 3, and select 8 selects quadrant 8. Bits 156 through 167 are associated with bits 154, 155, and 168 through 173.

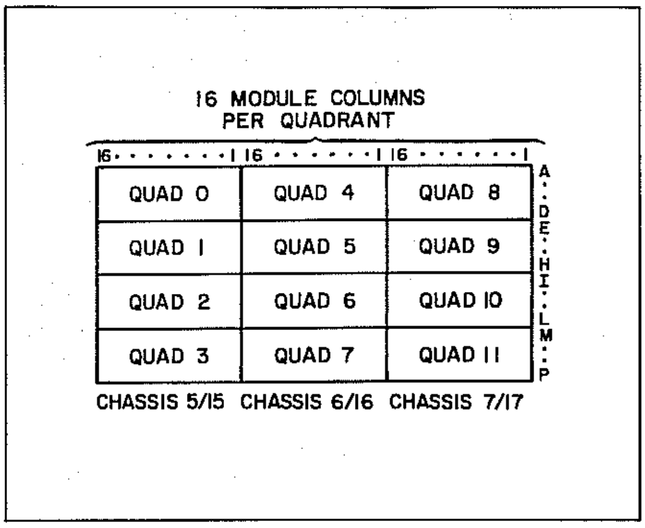


Figure 5-23. CP Chassis Quadrants (Viewed from Module Side)

BITS 168/250$_8$ THROUGH 173/255$_8$ REFERENCE VOLTAGE MARGINS MODULE ADDRESS BITS 0 THROUGH 5 — CONTROL

These bits select one of 64 modules in a CP chassis quadrant (figure 2-7). Address bits 0 through 3 select 1 of 16 module columns. Address bits 4 and 5 select one of four module rows. The addresses increase by module location within a row and by rows within a quadrant.

BIT 174/256$_8$ PPS TO CMC ZERO ADDRESS PARITY — CONTROL

This bit forces the IOU to put a zero parity bit on the address sent to CMC.

BIT 175/257$_8$ PPS TO CMC ZERO DATA PARITY — CONTROL

This bit forces the IOU to put a zero parity bit on the data sent to CMC.

BITS 176/260$_8$ THROUGH 182/266$_8$ CM CONFIGURATION — STATUS

These bits indicate which 262K quadrants/ports are not available.

BIT 183/267$_8$ DOUBLE ERROR — STATUS

This bit, when set, indicates that a double error occurred. Software must reset (clear) the bit. When the bit clears and bit 3 sets, it indicates that a single error set bit 3. When a SECDED error occurs, one of the following conditions describes the error.

- A single-bit error occurred.

  Bit 2 clears if the SECDED error was detected by the CMC in CP-0; bit 2 sets if the SECDED error was detected by the CMC in CP-1.

  Bit 3 sets, indicating a SECDED error.

  Bit 183 clears, indicating a single error.

  Bits 40 through 47 contain the syndrome code (odd number of bits), indicating the failing bit.

  Bits 48 through 53 indicate the failing CM bank, quadrant/port and model 865 chip select.

  Bits 190 and 191 indicate the failing chip enable of the failing CM pak (model 875 only).

  Bit 2 clears if the SECDED error was detected by the CMC in CP-0; bit 2 sets if the error was detected by the CMC in CP-1.

- A double-bit error occurred.

  Bit 2 clears if the SECDED error was detected by the CMC in CP-0; bit 2 sets if the SECDED error was detected by the CMC in CP-1.

  Bit 3 sets, indicating a SECDED error.

  Bit 183 sets, indicating a double error.

  Bits 40 through 47 contain a syndrome code (even number of bits) that does not indicate the failing bits.

  Bits 48 through 53 indicate the failing CM bank, quadrant/port, and model 865 chip select.

  Bits 190 and 191 indicate the failing chip enable of the failing CM pak (model 875 only).

  Bit 2 clears if the SECDED error was detected by the CMC in CP-0; bit 2 sets is the error was detected by the CMC in CP-1.

- A single-bit error occurred. Before software could clear it, a double-bit error occurred.

  Bit 2 clears if the SECDED error was detected by the CMC in CP-0; bit 2 sets if the SECDED error was detected by the CMC in CP-1.

  Bit 3 sets, indicating a SECDED error.

  Bit 183 sets, indicating a double error.

  Bits 40 through 47 contain a syndrome code (odd number of bits) for the single-bit error.

  Bits 48 through 53 indicate the failing CM bank, quadrant/port, and model 865 chip select.

  Bits 190 and 191 indicate the failing chip enable of the failing CM pak (model 875 only).

  Bit 2 clears if the SECDED error was detected by the CMC in CP-0; bit 2 sets is the error was detected by the CMC in CP-1.

BIT 184/270$_8$ — NOT USED

BIT 185/271$_8$ — NOT USED

BIT 186/272$_8$ — NOT USED

BIT 187/273$_8$ SMU-1 IN PROGRESS — STATUS

This bit indicates an SMU-1 transfer is taking place. The bit clears when the transfer completes.

BITS 188/274$_8$ AND 189/275$_8$ SOFTWARE FLAG 0 AND FLAG 1 — CONTROL

These bits are used by diagnostic software for communication between PPs.

BIT 190/276$_8$ AND 191/277$_8$ SYNDROME ADDRESS BIT 18, 19 — STATUS

These bits and bits 40 through 53 are provided upon detection of a SECDED error. Bits 190 and 191 indicate which chip enable failed on a memory module (model 875 only). Setting bit 3 locks bits 190 and 191. Clearing bit 3 unlocks bits 190 and 191 but does not clear it. Software functions cannot clear or set the read-only syndrome address bits.

BIT 192/300$_8$ CP-0 STOPPED — STATUS

This bit, when set, indicates that the CP has stopped. When the CP resumes operation, the bit clears.

BIT 193/301$_8$ CP-1 STOPPED — STATUS

This bit applies only to models with two CPs and is unused in models with one CP. When set, the bit indicates that the CP-1 has stopped. When the CP resumes operation, the bit clears.

BIT 194/302$_8$ SMU-0 IN PROGRESS FLAG — STATUS

This bit indicates SMU-0 transfer is currently in progress. When the transfer completes or terminates, the bit clears.

BIT 195/303$_8$ MONITOR FLAG CP-0 — STATUS

This bit indicates the condition of the monitor flag in CP-0.

BIT 196/304$_8$ MONITOR FLAG CP-1 (BIT 196) — STATUS

This bit applies only to models with two CPs and is unused in models with one CP. The bit indicates the condition of the monitor flag in CP-1.

BITS 197/305$_8$ THROUGH 200/310$_8$ PPM SELECT BITS 0 THROUGH 3 — STATUS

These bits indicate the positions of the IOU-0/IOU-1 and PP MEMORY SELECT switches on the deadstart panel. The switches select which physical PPM is logical PPM-0. The PP associated with the selected PPM is the controlling PP-0.

Bits 197 through 200 indicate the PP selection as follows:

Bit 200	Bit 199	Bit 198	Bit 197	Selection
0	0	0	0	PP-0
0	0	0	1	PP-1
0	0	1	0	PP-2
0	0	1	1	PP-3
0	1	0	0	PP-4
0	1	0	1	PP-5
0	1	1	0	PP-6
0	1	1	1	PP-7
1	0	0	0	PP-8
1	0	0	1	PP-9

BIT 201/311$_8$ EXTERNAL CHANNEL SELECT — STATUS

This bit indicates that IOU-0 is selected when the bit is a 0 and that IOU-1 is selected when the bit is a 1. A PPM reconfiguration is not effective in IOU-1 unless all 10 PPs are installed.

BITS 202/312$_8$ AND 203/313$_8$ EXPANDED MODE ADDRESSING

Bit 202 indicates that CP-0 exchange package has selected expanded mode addressing. Bit 203 indicates that CP-1 exchange package has selected expanded mode addressing.

GLOSSARY A

AOR	Address out of range
CEJ	Central exchange jump
CIW	Current instruction word register
CM	Central memory
CMC	Central memory control
CP	Central processor
CPU	Central processing unit
CRT	Cathode-ray tube
DCC	Data channel converter
DSC	Display station controller
DTR	Data terminal ready
EC	Exit condition code field at RAC
ECC	Error condition code
ECL	Emitter-coupled logic
ECS	Extended core storage
EEM	External extended memory
EIA	Electronic Industries Association
EM, EMS	Exit mode selection
ESM	Extended semiconductor memory
FIFO	First in, first out
FLC	Field length, central memory
FLE	Field length, extended memory
IAS	Instruction address stack
ILH	Instruction lookahead hardware
I/O	Input/output
IOU	Input/output unit
IWS	Instruction word stack
MA	Monitor address
MC	Master clear
MEJ	Monitor exchange jump
MF	Monitor flag
MOS	Metal oxide semiconductor
MUX	Multiplexer, selector
OS	Operating system
PE	Parity error
PP	Peripheral processor
PPM	Peripheral processor memory
RAC	Reference address, central
RAE	Reference address, extended
RNI	Read next instruction
RTS	Request to send
SAS	Storage adress stack
SECDED	Single-error correction double-error detection
SMU	Storage move unit
UART	Universal Asynchronous Receiver and Transmitter
UEM	Unified extended memory

INDEX