CONTROL DATA

CYBER 180

810 and 830 Computer Systems

Hardware Reference Manual

810 Computer System
810A Computer System
65810 Computer System
830 Computer System
830A Computer System
65830 Computer System

60469420

REVISION RECORD

REVISION	DESCRIPTION
01	Preliminary manual released.
(12-83)	
A	Manual released. This edition obsoletes all previous editions.
(3-84)	
B	Manual revised; includes ECO PD 03413, 16/32 megabyte memory options.
(4-85)	
C	Manual revised; includes ECO PD 3834.
(05-87)	

Publication No.
60469420

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

Address comments concerning this manual to:

Attention: Publications

Control Data Canada Ltd.
1855 Minnesota Court
MISSISSAUGA, Ontario
CANADA
L5N 1K7

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	PAGE	REV	PAGE	REV	PAGE	REV	PAGE	REV
Cover	C	I-4-6	A	I-5-8	A	Divider	A		
Title	C	I-4-7	A	I-5-9	A	II-4-1	A		
2	C	I-4-8	A	I-5-10	A	II-4-2	A		
3	C	I-4-9	A	I-5-11	A	II-4-3	A		
4	A	I-4-10	A	I-5-12	A	II-4-4	A		
5	C	I-4-11	A	I-5-13	A	Divider	A		
6	C	I-4-12	A	I-5-14	A	II-5-1	A		
7	A	I-4-13	A	I-5-15	A	II-5-2	A		
8	A	I-4-14	A	I-5-16	A	Divider	A		
9	A	I-4-15	A	I-5-17	A	A-1	A		
Dividers	A	I-4-16	A	I-5-18	A	Divider	A		
I-1-1	C	I-4-17	A	I-5-19	A	Index-1	A		
I-1-2	C	I-4-18	A	I-5-20	A	Index-2	A		
I-1-3	A	I-4-19	A	I-5-21	A	Index-3	A		
I-1-4	B	I-4-20	A	I-5-22	A	Index-4	A		
I-1-5	A	I-4-21	A	I-5-23	A	Index-5	A		
Divider	A	I-4-22	A	I-5-24	A	Index-6	A		
I-2-1	A	I-4-23	A	I-5-25	A	Index-7	A		
I-2-2	A	I-4-24	A	I-5-26	A	Comment Sheet	C		
I-2-3	A	I-4-25	A	I-5-27	A	Back Cover	A		
I-2-4	A	I-4-26	A	I-5-28	A				
I-2-5	B	I-4-27	A	I-5-29	A				
I-2-6	B	I-4-28	A	I-5-30	A				
I-2-7	A	I-4-29	A	I-5-31	A				
I-2-8	A	I-4-30	A	I-5-32	A				
I-2-9	A	I-4-31	A	I-5-33	B				
I-2-10	A	I-4-32	A	I-5-34	B				
I-2-11	B	I-4-33	A	I-5-35	A				
Divider	A	I-4-34	A	I-5-36	A				
I-3-1	B	I-4-35	A	Dividers	A				
I-3-2	B	I-4-36	A	II-1-1	B				
I-3-3	A	Divider	A	II-1-2	A				
I-3-4	A	I-5-1	A	Divider	A				
Divider	A	I-5-2	A	II-2-1	A				
I-4-1	A	I-5-3	A	II-2-2	A				
I-4-2	A	I-5-4	A	Divider	A				
I-4-3	A	I-5-5	A	II-3-1	A				
I-4-4	A	I-5-6	A	II-3-2	A				
I-4-5	A	I-5-7	A	II-3-3	A				

This page intentionally left blank.

PREFACE

This manual contains hardware reference information for the CDC CYBER 170 Models 810, 810A, 65810, 830, 830A and 65830 Computer Systems. Since all 810 models have the same technical characteristics, the term "model 810" will stand for models 810, 810A, and 65810. Similarly, "model 830" includes 830, 830A and 65830.

Part I describes the functional, operational, and programming characteristics of the computer system hardware.

Part II describes the differences between models 810 and 830 and members of the CYBER 170 series. Current users of CYBER 170 computer systems should read this part first for a summary of differences. The comparison is based on the model 730. Refer to the appropriate CYBER 170 series hardware reference manual for differences between the model 730 and other CYBER 170 computer systems.

The 800 series of computers has two states of operation:

- o The 170 state, capable of running programs written for earlier CDC computer systems (60-bit word length, in effect real memory).
- o The virtual state, with its own instruction set, capable of performing housekeeping functions for the 170 state or running complete programs written for the virtual state (64-bit word length, virtual memory).

Additional system hardware information is available in the publications listed in the system publication index on the following page.

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 but not listed in the following index are:

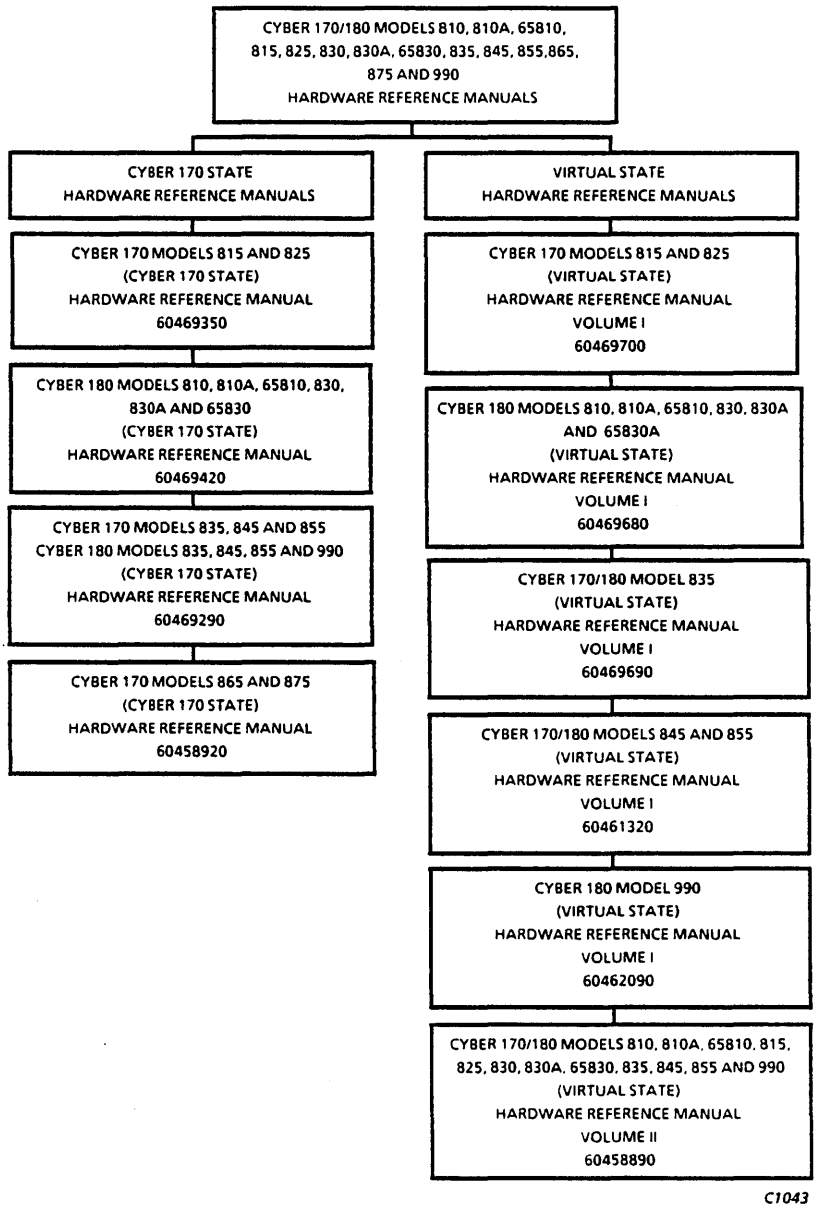
Control Data Publication	Publication Number
NOS Operator's Guide	60457700
NOS System Programmer's Instant	60457790

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. It has been tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of the FCC Rules which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference in which case the user, at his own expense, will be required to take whatever measures may be required to correct the interference.

PUBLICATION INDEX



C1043

CONTENTS

PART I

PART II

APPENDIX

INDEX

FIGURES

TABLES

PART 1

This section introduces the CDC® CYBER 180 Model 810 and 830 Computer Systems, gives physical and functional characteristics, and provides descriptions of major system components.

INTRODUCTION

CYBER 180 Model 810 and 830 (figure I-1-1) are medium scale, high-speed computer systems for both business and scientific applications. The systems include the following components:

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

PHYSICAL CHARACTERISTICS

The mainframe configuration (figure I-1-2) includes a display station and a single cabinet for the CP, CM, and IOU. The cabinet contains a logic chassis with plug-in circuit boards. It also contains fans to cool the logic chassis, an ac/dc control section with voltage margin testing facilities, and dc power supplies.

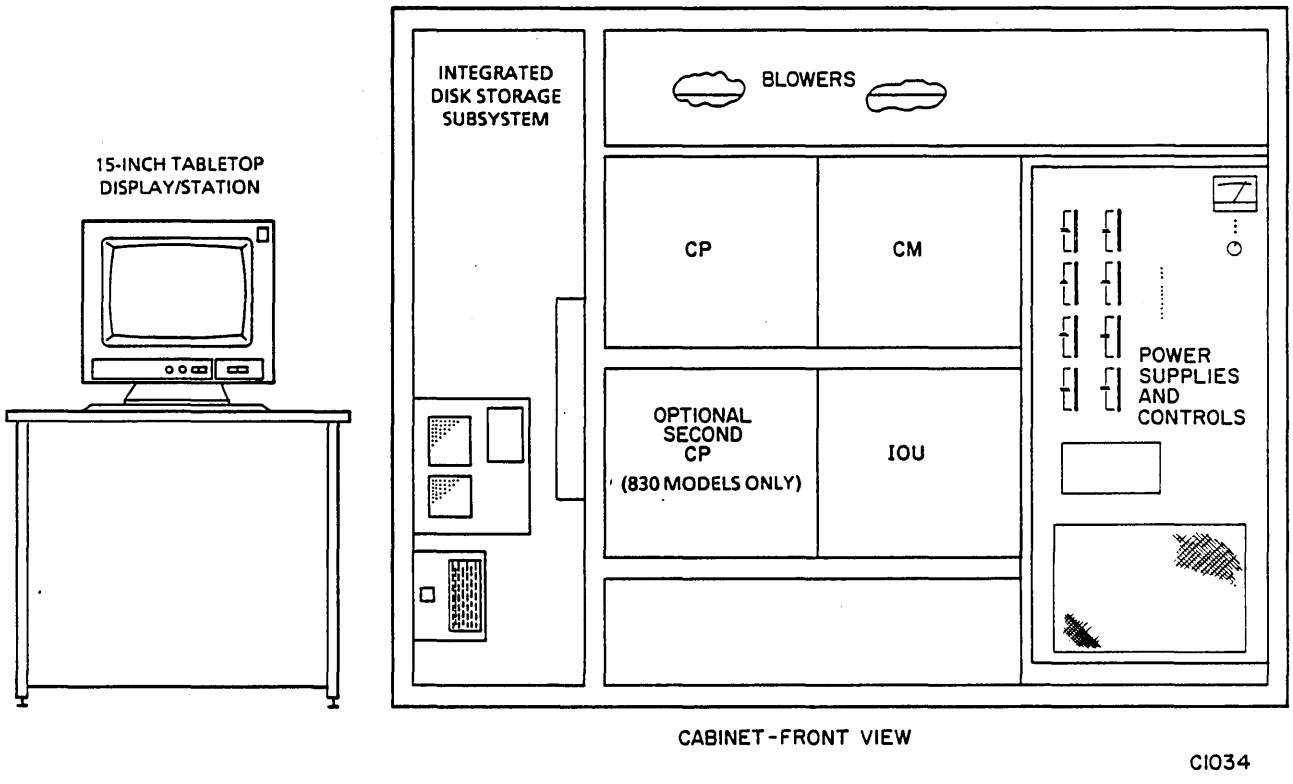


Figure I-1-1. Computer System Components

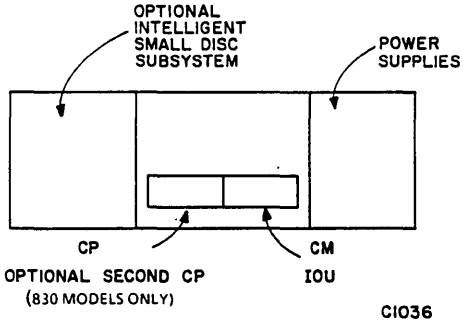


Figure I-1-2. Chassis Configuration (Top Cutaway View)

FUNCTIONAL CHARACTERISTICS

The computers use emitter-coupled logic (ECL) to achieve high computation speeds. Design architecture is also oriented towards this objective. 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 semiconductor central memory is divided into four independent banks. 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.

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 the central memory space of each user (RAC, FLC)
- Two registers that isolate the unified extended memory space of each user (RAE, FLE)
- Register exchanging instructions (exchange jumps) for interrupting programs
- Floating-point arithmetic (11-bit exponent, 48/96-bit coefficient)
- Integer arithmetic (60/18-bit operands)
- Character string compare/move facilities (6-bit characters)
- Packed instructions (15/30/60-bit instructions in 60-bit words)
- Synchronous internal logic
- 50-nanosecond clock period
- Instruction lookahead
- Microcode control
- Parity checking of all major data and address paths
- Maintenance channel to IOU

CENTRAL MEMORY

- 72-bit data word (60 data bits, 8 SECDED bits, and 4 unused bits)
- Dynamic semiconductor memory; options available from 262K words to 2096K words in 170 state; to 4192K words in virtual state.
- Organized into independent banks as follows: 262K memories into two banks, other memories into four banks
- Two memory ports
- Bounds register to limit write access from specified ports
- 50-nanosecond clock period
- Maximum data transfer rate of one word every 100 nanoseconds
- CM cycle time for a read or write is 400 nanoseconds
- Read and write data queuing capability
- Single-error correction double-error detection (SECDED) on stored data
- Parity checking of all major data, address and control paths
- Unified extended memory (UEM)

INPUT/OUTPUT UNIT

- Ten peripheral processors (PPs); 10 additional PPs are available. Each PP has 4K independent memory (PPM) of 16-bit words with the upper 4 bits zero.
- Port to central memory
- Bounds register to limit writes to central memory
- Parity checking on all major data and address paths
- 500 nanosecond major cycle and 50 nanosecond minor cycle

- Each PP connects to 12, 18, or 24 internal I/O channels, depending on the option choosen; 6, 9, or 12 of these can interface customer devices through 12-bit CYBER 170 I/O channel interfaces. The balance interfaces the devices listed next.
- Real-time clock (channel 14_8)
- Display controller (CYBER 170 channel 10_8)
- Two-port multiplexer (channel 15_8)
- Time-of-day/date clock (channel 15_8)
- Telephone dial-out equipment (channel 15_8)
- Integrated controllers (channels 0, 6, 20, 26), controlling small intelligent discs in a bolt-on cabinet

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 I-1-3) consists of the following:

- Control section
- Registers
- Execution section
- Addressing section

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

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 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 contains only zeros.

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

- Program address (P) register, 18 bits
- Reference address for CM (RAC) register, 21 bits
- Field length for CM (FLC) register, 21 bits
- Exit mode (EM) register, 6 bits
- Flag register, 6 bits
- Reference address for UEM (RAE) register, 21 bits
- Field length for UEM (FLE) register, 24 bits
- Monitor address (MA) register, 18 bits

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.

Addressing Section

The addressing section checks memory addresses against the bounds registers to ensure isolation of user memory space.

CENTRAL MEMORY

The CM (figure I-1-3) consists of the following:

- Either two or four memory banks
- Two memory ports
- Distributor

The CM is a refresh type metal oxide semiconductor (MOS) memory, which is organized into four independent banks. Memory read/write requests are stored and distributed at ECL speeds, after which each bank completes the requests presented to it at MOS speeds.

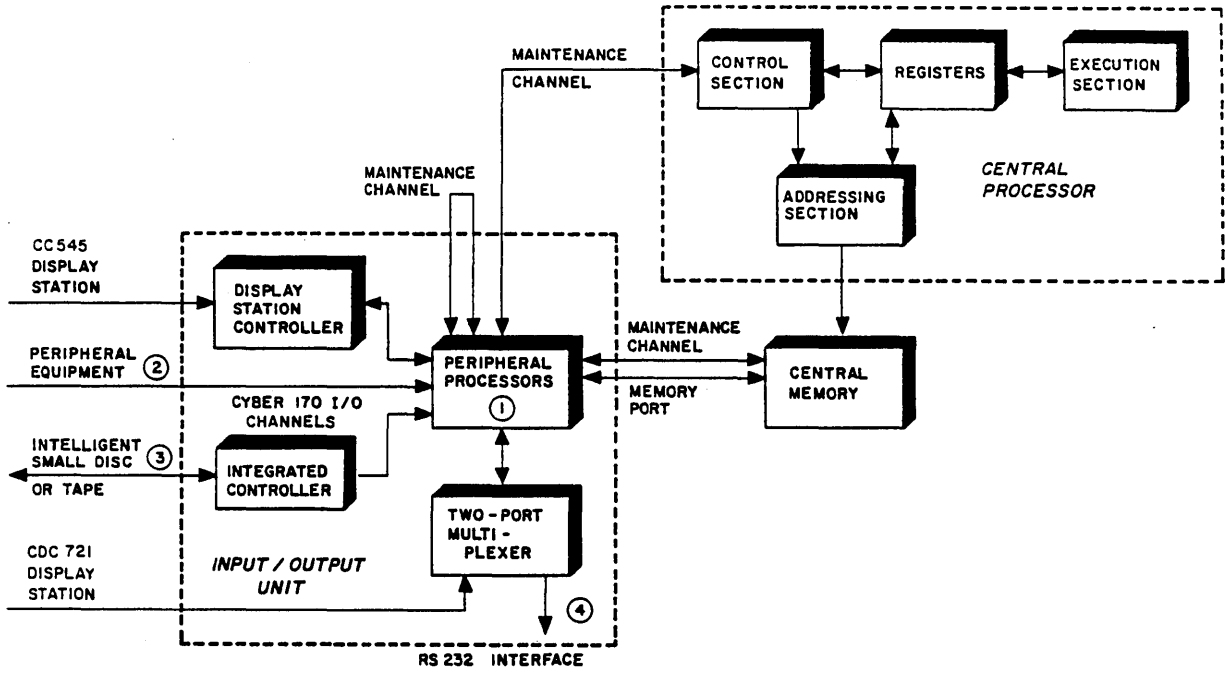


Figure I-1-3. Computer System Block Diagram

INPUT/OUTPUT UNIT

The IOU (figure I-1-3) consists of the following:

- Ten logically independent peripheral processors (PP). An option is available to increase total to 20 PPs.
- Internal interface to 12 I/O channels. An additional 12 channels are available in 6-channel increments.
- External interfaces to I/O channels:
  - 4, 6, 7, 9, 10 or 12 CYBER 170 channel interfaces
  - Display controller interface (CYBER 170 channel 10_8)
  - Real-time clock interface (channel 14_8)
  - Two-port multiplexer interface (channel 15_8)
  - Maintenance channel interface (channel 17_8)
  - Integrated control interfaces 2, 3, 4, 5 or 6
- Interface to central memory.
- Bounds register to limit writes to central memory.

The PPs are 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 STATIONS

The computer operator monitors and controls the system by an operator console. Two types are available as options for the model 810/830 computer system: the CC545 Display Station and the CDC 721 Display Terminal.

The display station has a 21-inch screen, a fixed keyboard, and a stand. It is driven from the display station controller in the IOU via channel 10 octal. It can display the system dayfile (A display) and/or job status (B display). The character set and programming information of this station are in part I, section 5. Refer to CC545 Display Station Manual (62952600) for more information.

The display terminal is a compact desktop unit in a tiltable display module with a 15-inch screen and a detached alphanumeric keyboard. It is microprocessor based, with character/symbol generation memory and circuits. It can receive/transmit at up to 19 200 BPS through the two port mux.

Refer to CDC 721-XO Display Terminal Hardware Reference Manual (62940020) for further hardware information.

This section provides functional descriptions of the CP, CM, and IOU parts shown in the block diagram in section 1 in this part. These parts consist of:

- Central processor (CP)
- Central Memory (CM)
- Input/output unit (IOU)

Functional description for the system display station is in the Display Station Reference Manual.

CENTRAL PROCESSOR

The CP consists of the control section, registers, the execution section, and the addressing section.

CONTROL SECTION

The control section keeps the instruction lookahead buffer registers filled with instruction words. This prefetch consists of two words for the regular program sequence. A program address source supplies the memory address for the regular program sequence. Instruction buffer and program address buffers hold instructions and their addresses during execution.

Instruction Lookahead

The read next instruction (RNI) hardware prefetches instruction words to make the next instruction immediately available when execution of the previous instruction is complete. To accomplish this, RNI reads instructions from CM into a two-word, first-in, first-out stack.

Maintenance Access Control

The maintenance access control performs initialization and maintenance operations in the CP, CM and IOU.

Instruction Control Sequences

The instruction control section performs instruction translation and control sequences. Each control sequence obtains the necessary instruction operands from the operating registers and provides the control signals for execution. Instructions read from CM are 60-bit instruction words that are in four 15-bit groups, two 30-bit groups, or a combination of 15-bit and 30-bit groups. The 15-bit groups are termed parcels with the first parcel (parcel 0) being the highest-order 15 bits of the 60-bit CM word. Second, third, and fourth parcels (parcels 1, 2, and 3) follow in order. The 30-bit groups contain two 15-bit parcels.

The instruction control sequences control the execution of one or more instructions of a common type. These sequences and associated instructions are briefly described in this section. (For further information, refer to CP Instruction Descriptions in section 4 in this part.)

Boolean Sequence

The Boolean sequence controls 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	BXi -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 Sequence

The shift sequence controls 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 sequence 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

The shift sequence also controls the normalize operations. The coefficient portion of the operand is repositioned, and the exponent is adjusted so that the most significant bit of the coefficient is in the highest-order bit position of the coefficient, and the exponent is decreased by the number of bit positions shifted. 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 Sequence

The floating-add sequence controls 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

Floating-Multiply and Floating-Divide Sequence

The floating-multiply and floating-divide sequence controls the operation of floating-multiply, floating-divide, and population-count instructions.

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

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

The population-count instruction counts the number of one bits in a 60-bit operand. The instruction is:

47	Population count of (Xk) to Xi	CXi Xk

Increment Sequence

The increment sequence controls the one's complement addition and subtraction of 18-bit fixed-point operands for increment instructions 50 through 77. The sequence also controls the 60-bit one's complement sum and difference values for long-add instructions 36 and 37.

The increment instructions are:

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

The long-add instructions are:

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

Compare/Move Sequence

The compare/move sequence controls the data manipulation on a character basis. The compare/move instructions (also referred to as CMU instructions) are 60-bit instructions that use six support registers for source and result field CM addresses and character position offsets. The support registers load from the 60-bit instruction word. The compare/move instructions are:

464	Move indirect (Bj) + K	IM
465	Move direct	DM
466	Compare collated	CC
467	Compare uncollated	CU

The support registers are:

- An 18-bit K1 register that specifies which relative CM address word contains the first character of the source data field.
- An 18-bit K2 register that specifies which relative CM address word contains the first character of the result field.
- A 4-bit C1 register that specifies the character position or offset of the first CM word of the source field.
- A 4-bit C2 register that specifies the character position or offset of the first CM word of the result field.
- Two 16-bit L registers (LA and LC) that specify the number of characters in the data field. The LA register is associated with K1, and the LC register is associated with K2. Instruction 464 uses 14 register bits. Instructions 465, 466, and 467 use only the lower eight register bits.

NOTE

For further information refer to Virtual State, section 5 of this part.

CYBER 170 Exchange Sequence

The CYBER 170 exchange sequence generates timed CM reference signals to implement the exchange of data between the CP and CM, as required by the CYBER 170 exchange jump instruction. In addition, the CYBER 170 exchange sequence provides internal control signals to the operating and control registers to systematically enter the contents of a CYBER 170 exchange jump package. (Refer to CYBER 170 exchange jump in this part for further information.)

UEM Block Copy Sequence

The block copy sequence controls the transfer of data between CM and UEM. The number of words to be transferred is determined by the addition of K to the contents of Bj. The locations of the starting addresses are determined by the setting of the block copy flag. The block copy instructions are:

011	Block copy Bj + K words from UEM to CM	REC Bj+K
012	Block copy Bj + K words from CM to UEM	WEC Bj+K

Direct Read/Write Sequence

Instructions 014 and 015 perform single word direct read and write operations for UEM, and instructions 660 and 670 perform single word direct read and write operations for central memory.

014	Read one word from UEM at (Xk + RAE) to Xj	RXj Xk
015	Write one word from Xj to UEM at (Xk + RAE)	WXj Xk
660	Read central memory at (Xk) to Xj	CRXj Xk
670	Write Xj into central memory at (Xk)	CWXj Xk

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. 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 (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 a CYBER 170 exchange package by a single CP instruction (CYBER 170 exchange jump).

The time a CYBER 170 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 CYBER 170 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 register is used in the compare instructions to indicate if two fields of characters are equal. Also, the X0 register provides the relative unified extended memory (UEM) starting address in a block copy operation.

The X1 through X7 registers are primarily data handling registers for computation with X1 through X5 used to input data from CM, and X6 and X7 used to transmit data to CM.

Operands and results transfer between CM and the X registers as a result of placing CM addresses into corresponding A registers.

A Registers

The CP contains eight 18-bit A registers, A0 through A7. The A0 register has special uses. The A0 register is used in the compare collate instruction for the collate table address. Also, the A0 register provides the relative CM starting address in a 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 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 a CYBER 170 exchange sequence. With the exception of the P register, the contents of the support registers cannot be altered during the execution interval of a CYBER 170 exchange package. When the execution interval completes, the data in the support registers is sent back to CM through a CYBER 170 exchange jump.

P Register

The 18-bit program address (P) register loads from CM during the first word of a CYBER 170 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 a CYBER 170 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 a CYBER 170 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 a CYBER 170 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	Hardware error
58	Hardware error
59	Hardware error

Flag Register

The 6-bit flag register loads from CM during the fourth word of a CYBER 170 exchange sequence. The flag register holds 6 bits that function as control flags.

Bits	Condition
51	Hardware error bit
52	Instruction stack (lookahead) purge flag. When set, enables extended purging of instruction lookahead registers to be done.
53	CMU interrupted flag. When set, one of instructions 464 through 467 has been interrupted. The information necessary to resume operation has been saved.
54	Block copy flag. When set, block copy instructions (011, 012) use bits 30 through 50 of X0 (rather than A0) to determine the CM address.
55	Reserved for use in other models. Must be a zero.
56	UEM enable flag. When set, UEM is being used. This flag must be set to allow 011, 012, 014, and 015 instructions to access UEM.

RAE Register

The 21-bit UEM reference address (RAE) register loads from CM during the fifth word of a CYBER 170 exchange sequence. The lower 6 bits of this register are always zero. An absolute UEM address forms by adding RAE to the relative address which is determined by the instruction.

FLE Register

The 24-bit UEM field length (FLE) register loads from CM during the sixth word of a CYBER 170 exchange sequence. The lower 6 bits of this register are always zero. The FLE register defines the size of the field in UEM for the program in execution. Relative UEM addresses are compared with FLE.

MA Register

The 18-bit monitor address (MA) register loads from CM during the seventh word of a CYBER 170 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 CYBER 170 monitor flag set, or when honoring a monitor exchange jump to MA (262x) instruction with the CYBER 170 monitor flag clear. For further information, refer to CYBER 170 Exchange Jump in section 5 of this part.

EXECUTION SECTION

The execution section combines the operands into results, providing additional sequencing control where necessary.

ADDRESSING SECTION

An address adder calculates memory addresses for data and unconditional jump instructions.

Memory management hardware verifies that memory addresses are to access permitted memory areas. If this is the case, this hardware accesses central memory.

CENTRAL MEMORY

The CM performs the following functions.

- In the 170 state, the memory banks store from 262K to 2096K of 64-bit words (the leftmost 4 bits are undefined) and an 8-bit SECDED code.
- The two ports make CM accessible to the CP and every PP.
- A bounds register limits access to CM from either or both ports.
- The SECDED generators generate the SECDED code bits stored with each word. SECDED checks circuits, corrects single-bit errors, and detects double-bit errors.

- The maintenance channel interface gives a PP in the IOU access to the CM maintenance registers for system initialization, corrective action, error reporting and diagnostics, and for setting the port bounds register.

CM PORTS AND PRIORITIES

A priority network resolves access conflicts on a rotating basis, preventing long-term lockout of any port. In case of simultaneous requests, the CP has priority. The CM also has a refresh mechanism which may consume a maximum of 6 percent of memory time. Refresh requests have priority over port requests.

ADDRESS FORMAT

Figure I-2-1 illustrates address format.

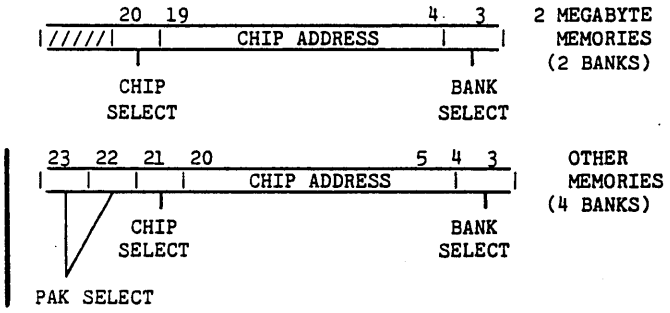


Figure I-2-1. Address Format

The following list defines the address fields.

- Bank select specifies one of two/four banks. Since the bank address is the lowest order two bits of the storage address, sequential addressing results in a phased-bank operation which allows a maximum data transfer rate of one word each clock period. Model 830 read or write bank cycle time is 8 clock periods and a read/modify/write (partial write) bank cycle time is 16 clock periods. Model 810 read or write bank cycle time is 24 clock periods and a read/modify/write (partial write) bank cycle time is 32 clock periods.

- Chip address specifies address of a word in 64K MOS memory chips for the selected bank.

- Chip select selects one of two word rows in a pak.

- Pak select selects one of two paks. It is used only for storage units larger than 524K.

CM ACCESS AND CYCLE TIMES

CM cycle time for a read or write is 400 nanoseconds. The difference between Model 810 and Model 830 is in the access time. Refer to table I-2-1 for maximum request lockout time.

TABLE I-2-1. PORT PRIORITY

Maximum Request Lockout Time in Bank Cycles	
Port	Read or Write Requests
Refresh	1
Port 0	2
Port 1	3
NOTE:	1 bank cycle equals 8 clock periods equals 400 nanoseconds for model 830.
	1 bank cycle equals 24 clock periods equals 1200 nanoseconds for model 810.

SECDED

The SECDED logic corrects single-bit errors during a CM read, permitting unimpeded computer operation. 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 I-2-2 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. A PP in the IOU may then analyze the syndrome bits using the maintenance channel.

UNIFIED EXTENDED MEMORY

Central memory contains an area which is reserved for special software called Virtual State software. Along with the hardware and microcode, this software handles the operations in the Virtual State as described in section 5 of this part. Virtual State software is located at the higher end of memory. The remaining memory is available to the CYBER 170 State and may be allocated as central memory (accessible via RAC and FLC) or as unified extended memory (accessible via RAE, FLE, and the 011, 012, 014, and 015 instructions). Refer to figure I-2-2.

TABLE I-2-2. SECDED SYNDROME CODES/CORRECTED BITS

Code	Bit	Code	Bit	Code	Bit	Code	Bit	Code	Bit	Code	Bit	Code	Bit	Code	Bit
00	(6)	20	66 (2)	40	65(2)	60	(3)	80	64(2)	A0	(3)	C0	(3)	E0	0 (1)
01	71 (2)	21	(3)	41	(3)	61	(4)	81	(3)	A1	(4)	C1	(4)	E1	0 (5)
02	70 (2)	22	(3)	42	(3)	62	(4)	82	(3)	A2	(4)	C2	(4)	E2	16 (5)
03	(3)	23	(4)	43	(4)	63	(3)	83	(4)	A3	(3)	C3	(3)	E3	16 (1)
04	69 (2)	24	(3)	44	(3)	64	(4)	84	(3)	A4	(4)	C4	(4)	E4	2 (5)
05	(3)	25	(4)	45	(4)	65	(3)	85	(4)	A5	(3)	C5	(3)	E5	2 (1)
06	(3)	26	(4)	46	(4)	66	(3)	86	(4)	A6	(3)	C6	(3)	E6	18 (1)
07	44 (1)	27	60 (5)	47	46(5)	67	62 (1)	87	45(5)	A7	61 (1)	C7	47 (1)	E7	18/63(5)
08	68 (2)	28	(3)	48	(3)	68	(4)	88	(3)	A8	(4)	C8	(4)	E8	1 (5)
09	(3)	29	(4)	49	(4)	69	(3)	89	(4)	A9	(3)	C9	(3)	E9	1 (1)
0A	(3)	2A	(4)	4A	(4)	6A	(3)	8A	(4)	AA	(3)	CA	(3)	EA	17 (1)
0B	12 (1)	2B	28 (5)	4B	14(5)	6B	30 (1)	8B	13(5)	AB	29 (1)	CB	15 (1)	EB	17/31(5)
0C	(3)	2C	(4)	4C	(4)	6C	(3)	8C	(4)	AC	(3)	CC	(3)	EC	3 (1)
0D	40 (1)	2D	56 (5)	4D	41(5)	6D	58 (1)	8D	41(5)	AD	57 (1)	CD	43 (1)	ED	3/59(5)
0E	8 (1)	2E	24 (5)	4E	10(5)	6E	26 (1)	8E	9 (5)	AE	25 (1)	CE	11 (1)	EE	19/27(5)
0F	(3)	2F	(4)	4F	(4)	6F	(3)	8F	(4)	AF	(3)	CF	(3)	EF	19 (1)
10	67 (2)	30	(3)	50	(3)	70	36 (1)	90	(3)	B0	4 (1)	D0	32 (1)	F0	(3)
11	(3)	31	(4)	51	(4)	71	36 (5)	91	(4)	B1	4 (5)	D1	32 (5)	F1	(4)
12	(3)	32	(4)	52	(4)	72	52 (5)	92	(4)	B2	20 (5)	D2	48 (5)	F2	(4)
13	(4)	33	(3)	53	(3)	73	52 (1)	93	(3)	B3	20 (1)	D3	48 (1)	F3	(3)
14	(3)	34	(4)	54	(4)	74	38 (5)	94	(4)	B4	6 (5)	D4	34 (5)	F4	(4)
15	(4)	35	(3)	55	(3)	75	38 (1)	95	(3)	B5	6 (1)	D5	34 (1)	F5	(3)
16	(4)	36	(3)	56	(3)	76	54 (1)	96	(3)	B6	22 (1)	D6	50 (1)	F6	(3)
17	44 (5)	37	28 (1)	57	46(1)	77	54/62(5)	97	45(1)	B7	22/61(5)	D7	47/50(5)	F7	63 (1)
18	(3)	38	(4)	58	(4)	78	37 (5)	98	(4)	B8	5 (5)	D8	33 (5)	F8	(4)
19	(4)	39	(3)	59	(3)	79	37 (1)	99	(3)	B9	5 (1)	D9	33 (1)	F9	(3)
1A	(4)	3A	(3)	5A	(3)	7A	53 (1)	9A	(3)	BA	21 (1)	DA	49 (1)	FA	(3)
1B	12 (5)	3B	28 (1)	5B	14(1)	7B	30/53(5)	9B	13(1)	BB	21/29(5)	DB	15/49(5)	FB	31 (1)
1C	(4)	3C	(3)	5C	(3)	7C	39 (1)	9C	(3)	BC	7 (1)	DC	35 (1)	FC	(3)
1D	40 (5)	3D	56 (1)	5D	42(1)	7D	39/58(5)	9D	41(1)	BD	7/57 (5)	DD	35/43(5)	FD	59 (1)
1E	8 (5)	3E	24 (1)	5E	10(1)	7E	26/55(5)	9E	9 (1)	BE	23/25(5)	DE	11/51(5)	FE	27 (1)
1F	(4)	3F	(3)	5F	(3)	7F	55 (1)	9F	(3)	BF	23 (1)	DF	51 (1)	FF	(3)

Notes:
1 Corrected single-bit error.
2 Syndrome code bit failed (single code bit set).
3 Double error or multiple error (even number of code bits set).
4 Multiple error reported as a single error.
5 Double error or multiple error with indicated bit(s) inverted.
6 No error detected.

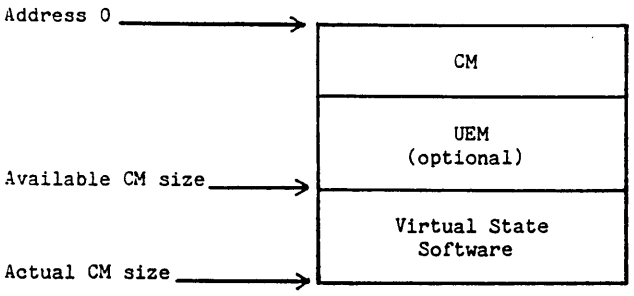


Figure I-2-2. Unified Extended Memory

CM BOUNDS REGISTER

The CM bounds register limits the write access to CM from specified ports. The ports are limited to the area between an upper and lower bound as specified in the CM bounds register. Bits in byte 0 specify the port(s) from which the write access is limited. The CM bounds register is set through the maintenance channel. For further information refer to Maintenance Channel Programming in section 5 of this part.

CENTRAL MEMORY RECONFIGURATION

Central memory reconfiguration is a manually performed function that permits the computer operator to restructure the CM addresses so that a failing part of CM can be quickly locked out to provide a continuous block of usable CM. CM reconfiguration is accomplished by setting switch registers in the memory unit to manipulate the upper address bits by issuing commands through deadstart display.

When a configuration switch register is set to force a CM address bit to a zero/one, the address range corresponding to the original installed memory accesses some parts of the reconfigured memory more than once. Addresses up to the rightmost forced bit, and half the addresses using the rightmost forced bit, cover a contiguous address space from location zero, which is the reconfigured memory. (Refer to section 3 in this part.)

INPUT/OUTPUT UNIT

The input/output unit (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
- Two-port multiplexer
- Maintenance channel
- CM access
- Time-of-day/date clock
- Telephone dial-out equipment

PERIPHERAL PROCESSOR

The basic IOU contains 10 PPs and is expandable to 20 PPs. Each PP is a logically independent computer with its own memory. Each 10-PP group is organized into a multiplexing system which allows the PPs to share common hardware for arithmetic, logical, and I/O operations without losing independence. This multiplexing system comprises a 10-rank first in, first out buffer of registers termed a barrel. Each rank contains information related to the instruction being executed by one PP.

Each PP communicates with the CP through CM using the CYBER 170 exchange jump. The PPs communicate with each other over the I/O channels.

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

Deadstart

A microprocessor controlled deadstart sequence allows the IOU to initialize itself. This sequence is initiated by the DEAD START switch on the display station and continued through commands selecting a deadstart option. When the switch is operated, a deadstart display appears on the display station, with operator selectable options which include the assigning of any PPM to PP0.

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. (Refer to figure I-2-4.) Information in these registers moves conceptually from one rank to the next at a uniform 20-megahertz rate, providing a multiplexed system of 10 PPs, each operating at a 2-megahertz rate. The registers are stationary while the PPs rotate. For example, rank 4 registers contain PP0 through PP11$_8$ in succession, each 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.

PP Registers

R Register

The 28-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 I-2-3.

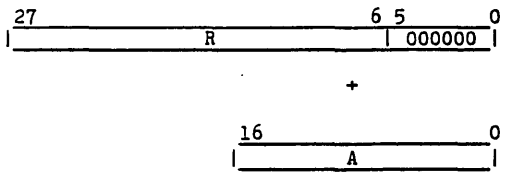


Figure I-2-3. Formation of Absolute CM Address

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 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 previously 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 7-bit K-register is visible to the programmer through the maintenance channel only. This register holds the operation code field of an instruction for display on the IOU deadstart display. When a PP is halted (idled), this register contains all ones.

PP Numbering

Logical PPs are numbered as follows:

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

The deadstart sequence accepts commands that determine barrel numbers and PP numbering within a barrel. According to the commands received, the sequence first assigns barrel numbers. Next, during the first minor cycle after deadstart, it loads a zero into the Q register in barrel 0 when the desired physical PP of this barrel is in the loading position. This defines all the data in that rank of the barrel as belonging to PP0 and since Q is the channel selector, assigns PP0 and channel 0. This defines PP1 and assigns it to channel 1. This process occurs in parallel in both 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 16 data bits with the upper 4 bits set to zero, and 1 parity bit. PP0 reads the deadstart program from the deadstart display 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.

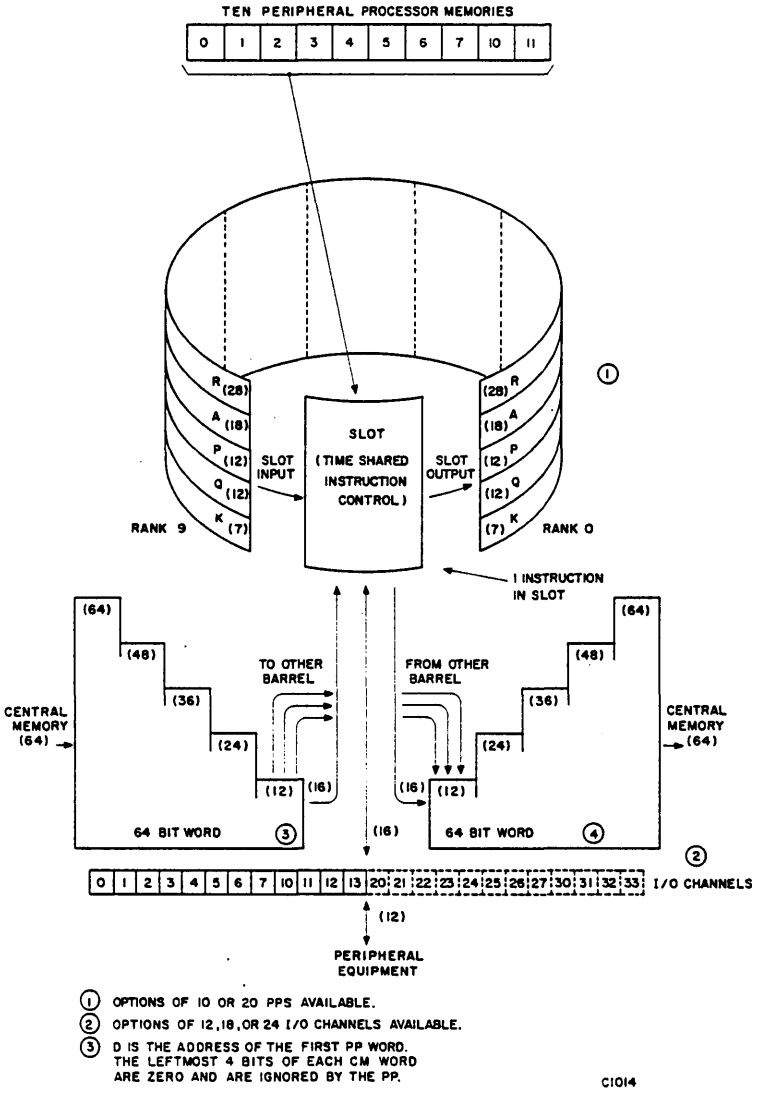


Figure I-2-4. Barrel and Slot

I/O CHANNELS

The I/O channels comprise 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 16-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 1000 nanoseconds.

Any PP can access any of the CYBER 170 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 CYBER 170 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.

TWO-PORT MULTIPLEXER

The two-port multiplexer provides communication capability between a PP and two attached terminals. One port is reserved for maintenance purposes and the other port is reserved for future use. The two-port multiplexer is permanently attached to channel 15_8.

MAINTENANCE CHANNEL

The maintenance channel consists of the maintenance channel interface on channel 17_8, a maintenance access control in each system element, and a set of interconnecting cables.

CENTRAL MEMORY 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 64-bit CM word with the leftmost 4 bits undefined. During a CM read, the IOU disassembles the rightmost 60 bits of the 64-bit CM word into five PP words. A PP forms a 28-bit CM address by adding the 28-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 64-bit word to or from the CM every 50 nanoseconds.

TIME-OF-DAY/DATE CLOCK

The two-port multipleter incorporates a calendar clock which supplies the year, month, day, hour, minute and second of date/time. This clock can be set to the minute and includes a power interruption indicator bit to guarantee the accuracy at the date/time supplied.

TELEPHONE DIAL-OUT EQUIPMENT

The two-port multiplexer includes auto dial-out equipment which can control any auto calling unit designed for operation through a RS366 channel. The equipment is intended for maintenance use; it is, however, accessible to all PPs and can be adopted for other uses. It can dial any telephone number and then transmit an unlimited amount of data to suitable receiving equipment.

INTEGRATED CONTROLLERS

Up to five integrated controllers can be mounted in the IOU panel. Each of these also requires an integrated control interface.

The integrated controllers normally control small intelligent discs or tapes, but can be adapted for other uses. The discs or tapes are housed in a cabinet which bolts to the main cabinet, eliminating the need for external cables.

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

CONTROLS AND INDICATORS

This section describes the IOU deadstart display and CM configuration switches used by the system operator. Other controls used by maintenance personnel are described in the Hardware Operator's Guide and the Display Station Hardware Reference Manual.

DEADSTART PANEL CONTROLS/INDICATORS

When the deadstart pushbutton on the display station is pressed, an initial deadstart display appears on the screen of the display station. It is created by an independent microcomputer in the mainframe, and does not rely on any program being operational in the PPs. The initial deadstart display is used to select a 16-word deadstart program for logical PP0 and to initiate the deadstart sequence for this PP. It is also used to reconfigure PPMs and barrels, and to display error status and maintenance information.

The format of the deadstart display is shown in figure I-3-1 and table I-3-1.

```
                    DEADSTART

XX YYYYYY = CHANGE DS PRO        PPM CONF = 00
XX+YYYYYY = CHANGE DS PRO INC    BRL CONF = 0
    S = SHORT DS                 DLY LOOP = 0
    L = LONG DS                  LDS ADDR = 6000
    H = HELP                     CLF FREQ = NORMAL
                        *) CM RECONF SW3 = C SW4 = C
                                   SW5 = C
    PROGRAM 0

01  001402
02  007303
03  000017
04  007503
05  007703
06  000300
07  007403
10  007103
11  007301
12  000010
13  000000
14  007112
15  000000
16  000000
17  000000
20  007112
```

Figure I-3-1. Initial Deadstart Display

Other deadstart displays are available for maintenance use. Refer to System Initialization, section in part II-3 and to the Hardware Operator's Guide for further information.

TABLE I-3-1. DEADSTART DISPLAY

Display	Description
CHANGE DS PRO	Change the single word in the dead-start program at XX to new value YYYYYY.
CHANGE DS PRO INC	Change a block of sequential words in the deadstart program starting at XX.
S = SHORT DS	S = select short deadstart sequence.
L = LONG DS	L = select long deadstart sequence.
H = HELP	Brings up a display that lists and explains available commands.
PPM CONF	PP memory configuration control.
BRL CONF	Barrel configuration control.
DLY LOOP	For maintenance use, controls delay between repeated dead-starts.
LDS ADDR	Long deadstart start address.
CLF FREQ	Sets clock frequency margin; for maintenance use.
CM RECONF SW3,4,5*)	CM reconfiguration switch registers; force high order CM address bits to zero or one.
PROGRAM 0	Deadstart program.
LDS BRANCH TEST	Appear when a long deadstart sequence is aborted to indicate nature of failure.
LDS DATA/ADDRESS	
ERROR (NOT SHOWN)	

Note: The display line also echoes keyboard entries on the bottom line, and erroneous entries on the next line up. Refer to System Initialization, section 3 in part II and to the Hardware Operator's Guide for further information.

*) For installed memory of 2096K or more, using boards with 256K chips, these switch registers become SW1, SW2, and SW3 respectively.

CENTRAL MEMORY CONTROLS

The CM contains three three-position configuration switch registers. The switch registers can be set through commands before deadstart to eliminate CM sections with malfunctions. Each switch, SW2 and SW3, or SW3, SW4, and SW5, forces one corresponding CM address bit, 24 through 21, either to a zero (switch down, D) or to a one (switch up, U). Refer to table I-3-2. In the table, C means center position.

In case of CM malfunctions, the remaining good memory can be reconfigured so it is accessible by contiguous addresses from zero to the maximum remaining address. This is accomplished by setting configuration switch registers (figure I-3-2) as listed in table I-3-2 (refer to the Hardware Operator's Guide).

TABLE I-3-2. CM RECONFIGURATION

CM Size Words	Failing Address Range	Configuration Switches			CM After Reconfig. Words
		SW3	SW4	SW5	
2000K	000000 - 0FFFFF	U	C	C	1000K
2000K	100000 - 1FFFFF	D	C	C	1000K
1000K	000000 - 07FFFF	C	U	C	500K
1000K	080000 - 0FFFFF	C	D	C	500K
500K	000000 - 03FFFF	C	C	U	250K
500K	04FFFF - 07FFFF	C	C	D	250K
		SW1	SW2	SW3	
4000K*)	000000 - 1FFFFF	C	U	C	2000K
4000K*)	200000 - 3FFFFF	C	D	C	2000K
2000K	000000 - 0FFFFF	C	C	U	1000K
2000K	100000 - 1FFFFF	C	C	D	1000K

*) 4000K is available for use in the virtual or dual state by NOS/VE.

IOU MAINTENANCE PANEL

The IOU contains a maintenance panel which directs IOU microcomputer operations. This panel incorporates the following switches:

Switch	Description
DIAGNOSTIC/NORMAL	When in the diagnostic position, causes the microcomputer to go into a diagnostic routine to verify IOU hardware integrity.
BAUD RATE SELECTION	(One switch for each port of the two-port mux). Selects one of eight baud rates (110 to 19.2K) at which the port operates.
PORT OPTIONS	(One switch for each port of the two-port mux). This switch can be set to enable/disable the following: • All communications • Nonsystem originated communications • Remote power control and deadstart.

Refer to Two-Port Mux, section 5 of this part for further information.

POWER-ON AND POWER-OFF PROCEDURES

In case of an emergency, use the system EMERGENCY OFF switch. Power-on and power-off procedures are described in the Hardware Operator's Guide.

CAUTION

Improper application or removal of power may damage system circuits. 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.

After being operated, the EMERGENCY OFF switch must be reset by a technician.

OPERATING PROCEDURES

Refer to the Hardware Operator's Guide. The system is initialized by setting its deadstart display control parameters, and then by running either a long or short deadstart sequence (defined later in this section). After initialization, the keyboard is used to instruct the system further, under program control.

CONTROL CHECKS

Before activating a long or short deadstart sequence, check the deadstart display parameters against their intended use. The normal settings of these parameters are as follows:

Parameter	Value
PPM CONF	00
BRL CONF	0
LDS ADDR	6000
Error messages	none

Deadstart Sequences

In response to a keyboard command (L or S) to the deadstart display, the IOU performs a deadstart sequence. Depending on the command (L or S), either the long or the short deadstart sequence is performed. The short deadstart sequence is used when hardware integrity verification is not required. The long deadstart sequence performs all the tasks performed by the short deadstart sequence and some additional tasks. The main additional task is the running of a diagnostic program, from a read-only memory (ROM) in the IOU, on logical PP0. The diagnostic program takes approximately 15 seconds to run.

Both deadstart sequences begin with a master clear which sets up all PPs, except logical PP0, for a 4096-word block input starting at PP location 0. The input into each PP is from the channel with the same number as the logical number of the PP concerned. The master clear also resets all external devices and sets maintenance channel connect code bit 52. The individual channels and registers are set as follows:

Channel	Active/ Inactive	Full/ Empty	Channel Flag	Channel Error Flag
0	Inactive	Empty	Clear	CLear
10 (display controller)	Active	Empty	Clear	Clear
14 (real-time clock)	Active	Full	Set	Set
15 (two-port mux)	Active	Empty	Clear	Clear
17 (maintenance)	Active	Empty	Clear	Clear
Other installed channels	Active	Empty	Clear	Clear
Noninstalled channels	Inactive	Empty	Clear	Clear

The flags of channel 14 and of noninstalled channels are fixed by hardware and cannot be changed.

Register	Initialization	Description
K	007100_8	Instruction display
P	007777_8	Causes block input to start from location 0
A	$10,000_8$	Count of 4096 words
Q	0, 1, 2...	I/O channel numbers (PP0: 0, PP1: 1, and so on)

All registers in both barrels are set to these values, except the registers of PP0.

If the long deadstart sequence is being performed, hardware clears location 7777_8 in all PP memories and sets the P register of PP0 to the value indicated by the parameter LDS ADDR = XXXX (normally 6000_8). PP0 starts performing a test program from a read-only memory in IOU. Hardware errors cause the LDS program to hang before completion. In the absence of errors, execution proceeds until the test program reaches location 7776_8. When this happens, the unique part of the long deadstart sequence ends with a master clear.

Next, both deadstart sequences clear PP0 location 0, write the deadstart program on the display into PP0 memory locations 1 to 20_8, and clear PP0 location 21_8. PP0 then starts executing the program entered from the deadstart display, (which is normally a bootstrap program to input more data from an assigned external device).

The short deadstart sequence does not disturb PP memory other than PP0 locations 0 to 21_8. Both deadstart sequences leave all PPs, except PP0, waiting for a block input, or for action through the maintenance channel. After the block input is complete, each PP starts executing the program entered from whatever address was entered into location 0 of that PP.

IOU RECONFIGURATION

The logical PP numbers and hardware are assigned to physical PPs circularly from the settings of IOU deadstart display PPM CONF and BRL CONF parameters, specifying which physical barrel and PPM is PP0. Maximum values for these parameters depend on the number of PPs installed. Illegal values entered in RB X and RP XX commands are rejected by the deadstart display, and cause error messages to appear on the screen (refer to the Hardware Operator's Guide). Reconfiguration is discussed in detail in the Hardware Operator's Guide; allowable values for the PPM CONF and BRL CONF parameters and reconfiguration examples are shown in tables I-3-3 and I-3-4.

TABLE 1-3-3. PP and Barrel Reconfiguration, RP=0

No. of PPs	Physical PPM in each Barrel	Logical PP RB = 0 Barrel 0	Logical PP RB = 0 Barrel 1	Logical PP RB = 1 Barrel 0	Logical PP RB = 1 Barrel 1	Valid RP Range	RB
	00	00					
	01	01					
	02	02				0 ≤ RP ≤ 11	Not valid
	03	03					
10	04	04	X	X	X		
	05	05					
	06	06					
	07	07					
	10	10					
	11	11					
	00	00	20	20	00		
	01	01	21	21	01		
	02	02	22	22	02	0 ≤ RP ≤ 11	Valid
	03	03	23	23	03		
	04	04	24	24	04		
20	05	05	25	25	05		
	06	06	26	26	06		
	07	07	27	27	07		
	10	10	30	30	10		
	11	11	31	31	11		

Notes
RP = PP reconfiguration parameter
RB = Barrel reconfiguration parameter
Physical PP numbers are determined by the PP memory assigned to each PP

TABLE 1-3-4. PP and Barrel Reconfiguration, RP=3

No. of PPs	Physical PPM in each Barrel	Logical PP RB = 0 Barrel 0	Logical PP RB = 0 Barrel 1	Logical PP RB = 1 Barrel 0	Logical PP RB = 1 Barrel 1	Valid RP Range	RB
	00	07					
	01	10					
	02	11				0 ≤ RP ≤ 11	Not valid
	03	00					
10	04	01	X	X	X		
	05	02					
	06	03					
	07	04					
	10	05					
	11	06					
	00	07	27	27	07		
	01	10	30	30	10		
	02	11	31	31	11	0 ≤ RP ≤ 11	Valid
	03	00	20	20	00		
	04	01	21	21	01		
20	05	02	22	22	02		
	06	03	23	23	03		
	07	04	24	24	04		
	10	05	25	25	05		
	11	06	26	26	06		

Notes
RP = PP reconfiguration parameter
RB = Barrel reconfiguration parameter
Physical PP numbers are determined by the PP memory assigned to each PP.

CP INSTRUCTIONS

CP INSTRUCTION FORMATS

NOTE

Cyber 170 CP instructions use the rightmost 60 bits in the 64-bit word. The leftmost 4 bits are undefined. 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 I-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 I-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.

Instructions 013 and 464 through 467 are 60-bit instructions which must start at parcel 0. They ignore any information in parcels 2 and 3; however, these parcels are normally set to all zeros.

INSTRUCTION COMBINATIONS

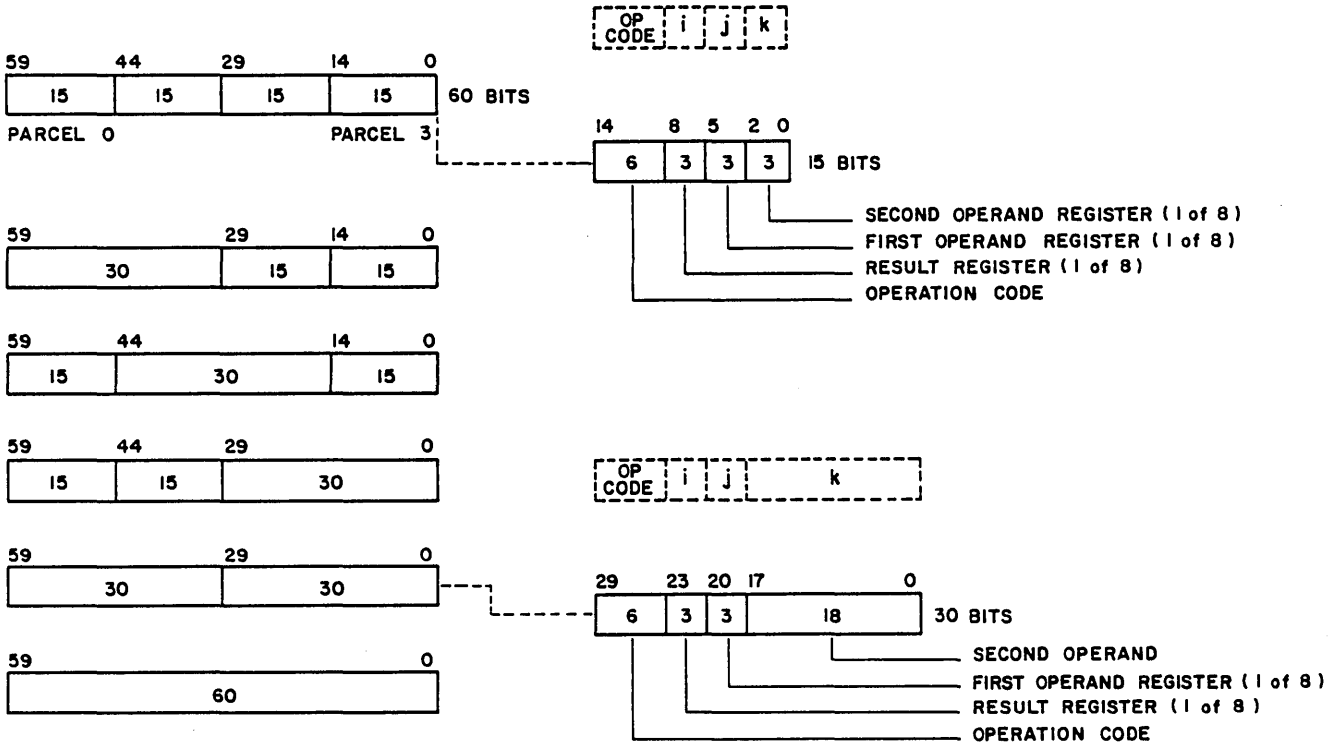


Figure I-4-1. CP Instruction Parcel Arrangement

TABLE I-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
C1*	Offset (character address) of the first character in the first word of the source field.
C2*	Character address of the first character in the first word of the result field.
K1*	18-bit address indicating the CM location of the first (leftmost) character of the source field.
K2*	18-bit address indicating the CM location of the first (leftmost) character of the result field.
LL*	Lower 4 bits of the field length (character count) for a move or compare instruction; used with LU to specify field length.
LU*	Upper 9 bits of the field length; (character count) for indirect move instruction or the upper 3 bits for direct instructions; used with LL to specify field length.

* Applicable to compare/move instructions only.

CP OPERATING MODES

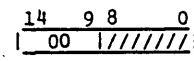
The CP executes instructions in CYBER 170 job mode, in CYBER 170 monitor mode, or in a Virtual State, and changes these through exchange jumps. Such exchanges are caused by CP instruction 013, PP instructions 2600, 2610, and 2620, by certain virtual state instructions, or by hardware-detected error conditions. Hardware caused exchanges are also called error exits; most of these can be enabled/disabled by setting/clearing bits in the CYBER 170 exchange package. Refer to CYBER 170 Exchange Jump, Virtual State, and Error Response, section 5 of this part.

A hardware flag called the CYBER 170 monitor flag (MF) when set/clear indicates that the CP is in the CYBER 170 monitor/job mode. Virtual State provides CYBER 170 environment at initialization, interacts with the central processor error detection logic, and is invoked when a CYBER 170 halt occurs. The Virtual State is invisible to the CYBER 170 state applications programmer.

CP INSTRUCTION DESCRIPTIONS

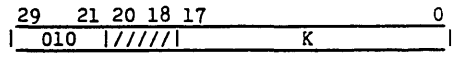
The instruction descriptions are in numerical order. The shaded areas, like those in the following 00xxx and 010xK instruction formats, indicate unused bits. The unused bits are ignored by the CP.

00xxx Error Exit to MA when CYBER 170 MF Clear
Interrupt to Virtual State when CYBER 170 MF Set — PS



This instruction causes an illegal instruction error exit. CYBER 170 MF is the hardware monitor flag. Refer to Illegal Instructions, section 5 of this part.

010xK Return Jump to K — RJ



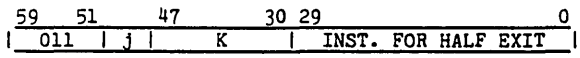
This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction writes a special word into CM at relative address K. The current program sequence then terminates by a jump to address K plus 1. The word stored in memory contains a jump instruction which causes an unconditional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



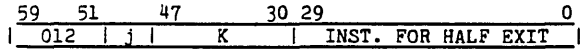
This instruction copies a block of Bj plus K consecutive words from unified extended memory (UEM) to CM, beginning at address A0 plus RAC in CM and address X0 plus RAE in UEM. It leaves Bj, X0 and A0 unchanged. Bj and K are both signed 18-bit one's complement numbers, making it possible to transfer a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

When the block copy flag is set, the destination CM address is calculated using the value in bits 30 through 50 of the X0 register rather than the A0 register. Bits 51 through 59 must be set to zero. Results are undefined if these bits are not zero.

If bit 21 or 22 of the result of X0 plus RAE is a one, zeroes are transferred and the next instruction is taken from parcel 2 of the same instruction word. If this is not the case, the next instruction is taken from parcel 0 of the next instruction word. Also, the execution of the 011jk instruction may be interrupted; if this occurs, execution is restarted from the beginning. Refer to Block Copy Instructions, section 5 of this part.

This instruction is illegal if the instruction does not start in parcel 0 or the UEM enable flag in the CYBER 170 exchange package is clear. If Bj plus K is negative, an address range error exit takes place. If the source field and the destination field overlap in physical memory, the final contents of the destination field are undefined.

012jK Block Copy Bj + K Words from CM to UEM) — WEC Bj+K



This instruction copies a block of Bj plus K consecutive words from CM to UEM, beginning at address X0 plus RAE in UEM and address A0 plus RAC in CM. 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.

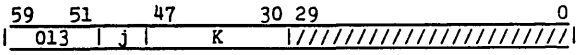
When the block copy flag is set, the source CM address is calculated using the value in bits 30 through 50 of the X0 register rather than the A0 register. Bits 51 through 59 must be set to zero. Results are undefined if these bits are not zero.

If bit 21 or 22 of the result of X0 plus RAE is a one, the next instruction is taken from parcel 2 of the same instruction word. If this is not the case, the next instruction is from parcel 0 of the next instruction word. Also, the execution of the 012jk instruction may be interrupted; if this occurs, execution is restarted from the beginning. Refer to Block Copy Instructions, section 5 of this part.

This instruction is illegal if the it does not start in parcel 0 or the UEM enable flag in the CYBER 170 exchange package is clear. If Bj plus K is negative, an address range error exit takes place. If the source field and the destination field overlap memory, the final contents of the destination field are undefined.

013jK Central Exchange Jump to Bj + K when CYBER 170 MF Set

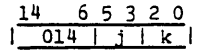
Monitor Exchange Jump to MA when CYBER 170 MF Clear — XJ Bj+K



This instruction must start at parcel 0. Also, a CYBER 170 exchange package must be ready at address Bj plus K or at address MA.

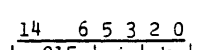
This instruction stores P plus 1 into the outgoing CYBER 170 exchange package in hardware and then exchanges this CYBER 170 exchange package with the CYBER 170 exchange package stored in memory. If, at the beginning of the instruction, CYBER 170 MF is set, then the incoming CYBER 170 exchange package starts at absolute address Bj plus K. If, at the beginning, CYBER 170 MF is clear, then the j and K fields of the instruction are ignored, and the incoming CYBER 170 exchange package starts at absolute address MA which is obtained from the outgoing CYBER 170 exchange package. In either case, CYBER 170 MF is toggled and the outgoing CYBER 170 exchange package is stored beginning at the same CM address from where the incoming CYBER 170 exchange package is obtained. Also, the jump is always to relative address P, parcel 0, from the new CYBER 170 exchange package. Refer to CYBER 170 Exchange Jumps, section 5 of this part.

014jk Read One Word from (Xk + RAE) to Xj — RXj Xk



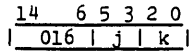
This instruction is illegal if the UEM enable flag in the CYBER 170 exchange package is clear. This instruction reads the 60-bit word from UEM location Xk plus RAE into Xj. Xk is less than FLE.

015jk Write One Word from Xj to (Xk + RAE) — WXj Xk



This instruction is illegal if the UEM enable flag in the CYBER 170 exchange package is clear. This instruction writes the 60-bit word from Xj into the UEM location Xk plus RAE. Xk is less than FLE.

016jk Read Free Running Counter



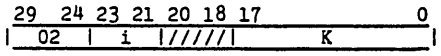
This instruction transfers the current contents of the 48-bit free running counter to the Xj register. The leftmost twelve bits of Xj are set to zero. The k field is ignored.

This instruction is a single parcel instruction that can be located in any parcel.

017jk Illegal Instruction

Refer to Illegal Instructions, section 5 of this part.

02ixK Jump to (Bi) + K JP

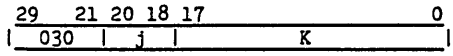


This two-parcel instruction uses the lower-order 18 bits as operand K. The instruction causes the current program sequence to terminate with a jump to address Bi plus K in CM.

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

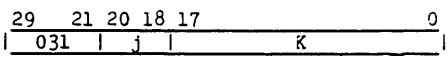


This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending upon the content of Xj. The branch to address K occurs only on the following 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) ≠ 0 NZ

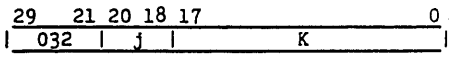


This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending upon the content of Xj. The program sequence continues only on the following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



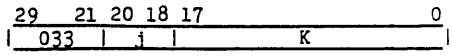
This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending upon the content of Xj. The branch decision for this instruction is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



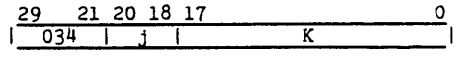
This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending upon the content of Xj. The branch decision for this instruction is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

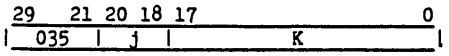


This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending upon the content of Xj. The program sequence continues only on the following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

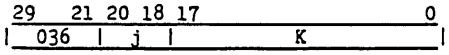
035jK Branch to K if (Xj) is Out of Range OR



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

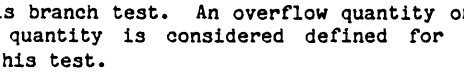


This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending upon the content of Xj. The program sequence continues only on the following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

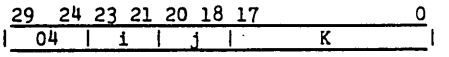


This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending upon the content of the Xj register. The branch to address K occurs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

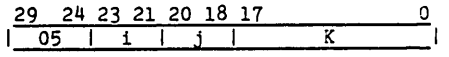
This instruction branches on a floating-point quantity which is not defined. The value of the coefficient is ignored in making this branch test. An overflow quantity or an underflow quantity is considered defined for purposes of this test.

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



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.

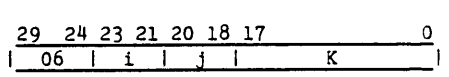
05ijK Branch to K if (Bi) ≠ (Bj) NE



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.

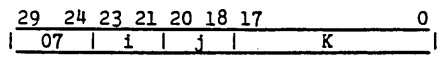
06ijK Branch to K if (Bi) ≥ (Bj) GE



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.

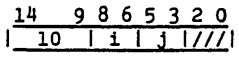
07ijK Branch to K if (Bi) < (Bj) LT



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.

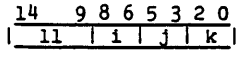
10ijx Transmit (Xj) to Xi BXi Xj



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*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.

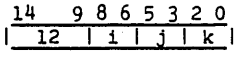
(Xj) = 7777 7000 0123 4567 1010

(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 Sj+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.

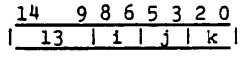
(Xj) = 0000 7777 0123 4567 1010

(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 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.

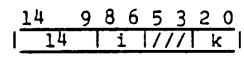
(Xj) = 0123 7777 0123 4567 1010

(Xk) = 0123 4567 7777 3210 1100

(Xi) = 0000 3210 7654 7777 0110

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

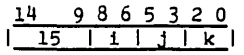
14ixk Transmit Complement of (Xk) to Xi BXi -Xk



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.

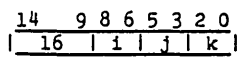
(Xj) = 7777 7000 0123 4567 1010

(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.

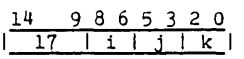
(Xj) = 0000 7777 0123 4567 1010

(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 combinations that may occur.

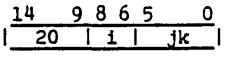
(Xj) = 0123 7777 0123 4567 1010

(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 toward 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.

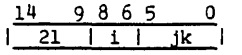
Initial (Xi) = 2323 6600 0000 0000 0111

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

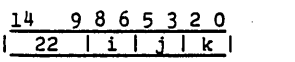
Initial (Xi) = 6000 4420 2222 0000 5643

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 Shifted (Xk) Nominally LXi Bj Xk
(Bj) Places to Xi



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 toward 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 toward 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

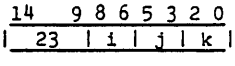
(Bj) = 77 7771

(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 Xk. 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 AXi Bj Xk
(Bj) Places to Xi



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 toward 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 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 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

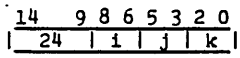
(Bj) = 77 7765

(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



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

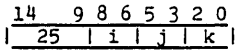
(Xk) = 5743 7730 1277 7777 5515

(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. If the exponent is less than negative 1777 with a zero coefficient, the contents of Xi and Bj are set to zero. For further information, refer to Floating-Point Arithmetic under CP Programming in section 5 of this part.

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



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

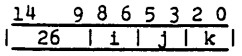
(Xk) = 5743 7730 1277 7777 5515

(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 of this part.

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



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 asigned 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

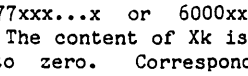
(Xk) = 6034 3277 4444 5777 7700

(Xi) = 7777 3277 4444 5777 7700

(Bj) = 77 7743

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

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



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

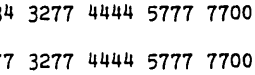
(Xk) = 7777 3277 4444 5777 7700

(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



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 of this part.

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 of this part.

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 of this part.

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

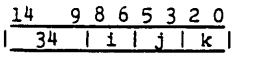
This instruction reads operands from two X registers, operates upon them to form a double-precision floating-point difference, and delivers the lower half of this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are in floating-point format and are not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

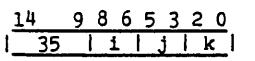
34ijk Round Floating Sum of (Xj) and (Xk) to Xi — RXi Xj+Xk



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 infloating-point format and is not necessarily normalized.

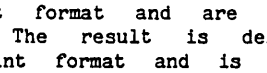
The round floating-point sum is a single-precision floating-point sum with a round bit (or bits) inserted before the add operation takes place. A round bit is always inserted in the coefficient with the larger exponent. If the two exponents are equal, the round bit is inserted in the coefficient for 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 of this part.

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



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 of this part.

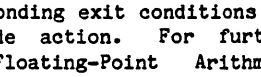
36ijk Integer Sum of (Xj) and (Xk) to Xi — IXi Xj+Xk



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 of this part.

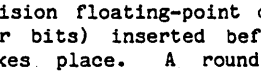
37ijk Integer Difference of (Xj) and (Xk) to Xi — IXi Xj-Xk



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 of this part.

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



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

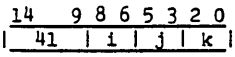
Xj and Xk. These operands are in floating-point format and are not necessarily normalized. The result is delivered to Xi in floating-point format. If both operands are normalized, the result is also normalized. If both operands are not normalized, the result is not normalized.

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 readfrom the double-precision product register. Leading zeros occur in this result coefficient. For further information, refer to Floating-Point Arithmetic under CP Programming in section 5 of this part.

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 of this part.

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



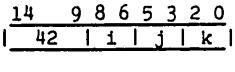
This instruction reads operands from two X registers, operates upon them to form a rounded floating-point product, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are in floating-point format and are not necessarily normalized. The result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

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



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.

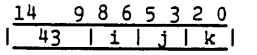
The operands are not rounded in this operation. The two operands are unpacked from floating-point format. The exponents are added to determine the exponent for the result. The result exponent is exactly 48 less than the exponent for a 40 instruction. The coefficients are multiplied as signed integers 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 of this part.

43ijk Form Mask of jk Bits to Xi MXi jk



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.

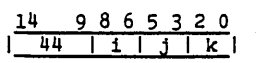
j = 2

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



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.

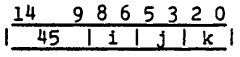
If the exponent subtraction causes an underflow or overflow, an underflow or overflow result is returned even with the occurrence of a divide fault.

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. (Refer to Floating-Point Arithmetic under Central Processor Programming in section 5 of this part.)

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 of this part.

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



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.

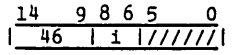
The two operands are unpacked from floating-point format in this operation. The exponents are subtracted with a correction factor to determine the exponent for the result. The coefficient from Xj is positioned in a dividend register. The Xj quantity is modified by inserting a 2525...25 round pattern 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 Central Processor Programming in section 5 of this part.)

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 of this part.

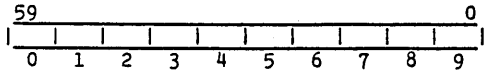
460xx through 463xx Pass NO



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 Compare/Move instructions must appear in parcel 0 or be treated as illegal instructions.

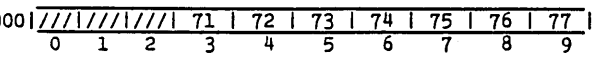
Data fields consisting of 6-bit characters may start or end with any character position (offset) of the 10 6-bit positions in each word. The character positions are designated as follows:



For move instructions, a K1 designator specifies which CM word contains the first character of the source data field, and a C1 designator specifies the character position (offset) of the first character. The K2 designator specifies the CM location in which the first character of the result data field is placed, and the C2 designator specifies the first character position. For compare instructions, both data field addresses specify source fields.

Example:

If the instruction is K1=1000 and C1=3, the first character of the source field is in position 3 of location 1000.



Therefore, the first character of the source field is 71.

An address is out of range if C1 or C2 is greater than 9, K1 plus N1 is greater than the program field length for CM (FLC), or K2 plus N2 is greater than FLC. N1 equals the number of CM references made to the source data field starting at K1, and N2 equals the number of CM references made to the result data field starting at K2. When an address-out-of-range condition occurs, the CMU instruction is not executed.

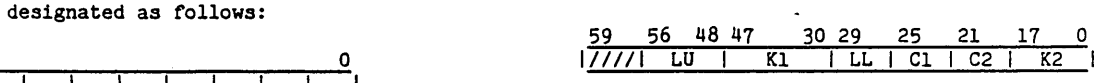
LL is the lower 4 bits, and LU is the upper 9 bits of the field length designator in numbers of characters. The maximum length of the data fields for the move direct and the compare instructions is 127 (177_8) characters. The maximum data field length for the move indirect instruction is 8191 (17777_8) characters. If L (LU and LL combined) is zero, the instruction becomes a pass.

For overlapping move instructions, the address of the source field (specified by K1) must be greater than the address of the result field (specified by K2) to provide proper field overlap. If K1 is less than K2, part of the source field is changed during execution, with the amount of change determined by the number of CM conflicts encountered. Overlapping fields should not contain more than 377 (octal) characters, because an exchange jump interrupts any compare/move operation having a decremented field length greater than 377 (octal).

464jK Move Indirect IM

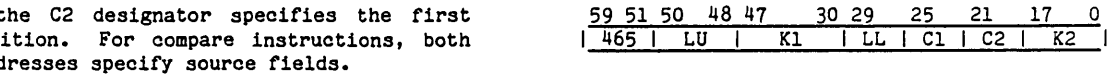
Any instructions located in the lower two parcels of the instruction word do not execute.

Bj plus K specifies a relative address in CM for the following descriptor word.



The descriptor word specifies the movement of the source field to the result field. The movement is from left to right through the field. Register X0 clears at the end of the execution.

465 Move Direct DM



This instruction moves the source field to the result field as specified by the 60-bit instruction word. The field length is limited to a 7-bit count.

466 Compare Collated CC

This instruction compares the field designated by K1,C1 with the field designated by K2,C2 as specified by the 60-bit instruction word.

The compare is from left to right through the fields until two unequal characters are found. These two characters are then collated and referenced in the

collate table beginning at address A0 (table I-4-2). If the table values found for the two unequal characters are equal, the compare continues until another pair of characters is unequal or until the field length is exhausted. If the table values found for the two unequal characters are unequal, X0 is set prior to instruction termination as follows:

If field K1 is greater than field K2, set X0 to 0000 0000 0000 0000 0xxx.

If field K1 is equal to field K2, set X0 to 0000 0000 0000 0000 0000.

If field K1 is less than field K2, set X0 to 7777 7777 7777 7777 7yyy where yyy is the complement of xxx.

The value of the three octal numbers xxx, stored in X0, is determined by the equation L minus N equals xxx (L is the length of the field, and N is the number of pairs of characters that were collated equal prior to instruction termination). In other words, xxx is the number of pairs of characters not yet compared plus one.

The A0 register contains the starting word address of an 8-word, 64-character collate table (table I-4-2). This table must have been previously stored in consecutive CM locations.

The collated value of a character is found by examining the collate table. The upper 3 bits of the character to be collated are added to A0 to obtain the relative address of the word containing the collated value. The lower 3 bits of the character to be collated specify the character address of the collated value.

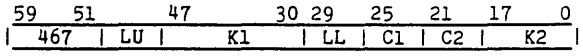
Example:

Suppose the character under examination is an octal 63. The 6 is added to the A0 to form the word address. The 3 is used to pick the correct character from that word. The value of 63 is 63 in the collate table.

TABLE I-4-2. COLLATE TABLE

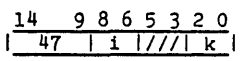
Address	Collating Character Locations									
A0	00	01	02	03	04	05	06	07	xx	xx
A0+1	10	11	12	13	14	15	16	17	xx	xx
A0+2	20	21	22	23	24	25	26	27	xx	xx
A0+3	30	31	32	33	34	35	36	37	xx	xx
A0+4	40	41	42	43	44	45	46	47	xx	xx
A0+5	50	51	52	53	54	55	56	57	xx	xx
A0+6	60	61	62	63	64	65	66	67	xx	xx
A0+7	70	71	72	73	74	75	76	77	xx	xx

467 Compare Uncollated CU



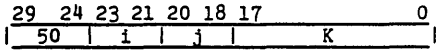
This instruction is similar to the 466 instruction except that the collate table is not used. The X0 register is set when the first pair of unequal characters is encountered or when the field length is exhausted.

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

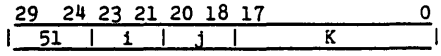


This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction reads an operand from Aj, forms the sum of the operand plus K, and delivers the result to Ai. If the i designator is nonzero, a reference is made to CM using the result as the relative address. The type of reference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

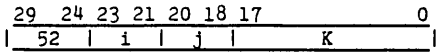


This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction reads an operand from Bj, forms the sum of the operand plus K, and delivers the result to Ai. If the i designator is nonzero, a reference is made to CM using the result as the relative address. The type of reference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

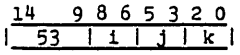


This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction reads an operand from Xj, forms the sum of the operand plus K, and delivers the result to Ai. If the i designator is nonzero, a reference is made to CM using the result as the relative address. The type of reference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

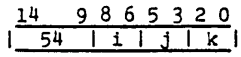


This instruction reads operands from Xj and Bk, forms the sum of the operands, and delivers the result to Ai. If the i designator is nonzero, a reference is made to CM using the result as the relative address. The type of reference is a function of the i designator value.

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

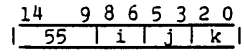


This instruction reads operands from Aj and Bk, forms the sum of the operands, and delivers the result to Ai. If the i designator is nonzero, a reference is made to CM using the result as the relative address. The type of reference is a function of the i designator value.

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

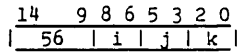


This instruction reads operands from Aj and Bk, subtracts the Bk operand from the Aj operand, and delivers the result to Ai. If the i designator is nonzero, a reference is made to CM using the result as the relative address. The type of reference is a function of the i designator value.

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

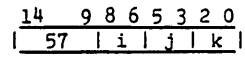


This instruction reads operands from Bj and Bk, forms the sum of the operands, and delivers the result to Ai. If the i designator is nonzero, a reference is made to CM using the result as the relative address. The type of reference is a function of the i designator value.

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

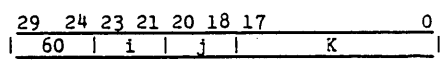


This instruction reads operands from Bj and Bk, subtracts the Bk operand from the Bj operand, and delivers the result to Ai. If the i designator is nonzero, a reference is made to CM using the result as the relative address. The type of reference is a function of the i designator value.

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

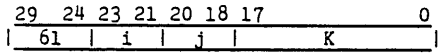
This instruction obtains operands from CM for computation and delivers the result back into CM.

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



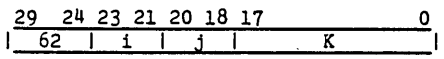
This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction reads an operand from Aj, forms the sum of the operand plus K and delivers the result to Bi. The sum is formed in an 18-bit one's complement mode. This instruction is for address modification in the increment registers.

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



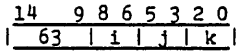
This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction reads an operand from Bj, forms the sum of the operand plus K, and delivers the result to Bi. The sum is formed in an 18-bit one's complement mode.

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



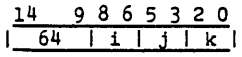
This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction reads an operand from Xj, forms the sum of the operand plus K, and delivers the result to Bi. The sum is formed in an 18-bit one's complement mode.

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



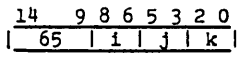
This instruction reads operands from Xj and Bk, adds the operands, and delivers the result to Bi. The sum is formed in an 18-bit one's complement mode.

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



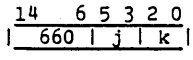
This instruction reads operands from Aj and Bk, adds the operands, and delivers the result to Bi. The sum is formed in an 18-bit one's complement mode.

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



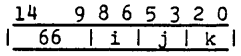
This instruction reads operands from Aj and Bk, subtracts the Bk operand from the Aj operand, and delivers the result to Bi. The difference is formed in an 18-bit one's complement mode. If the i designator is zero, this becomes a pass instruction.

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



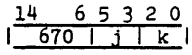
This instruction places into Xj the word at location (Xk), where Xk is a right-justified 21-bit relative word address. Bits 21 through 59 of Xk are ignored. Xk is less than FLC.

66ijk Set Bi to (Bj) + (Bk) SBi Bj+Bk
i not equal 0



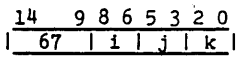
This instruction reads operands from Bj and Bk, adds the operands, and delivers the result to Bi. The sum is formed in an 18-bit one's complement mode.

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



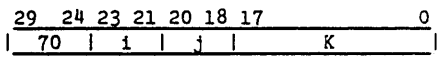
This instruction places Xj into location (Xk) where Xk is a 21-bit relative word address. Bits 21 through 59 of Xk are ignored. Xk is less than FLC.

67ijk Set Bi to (Bj) - (Bk) SBi Bj-Bk
i not equal 0



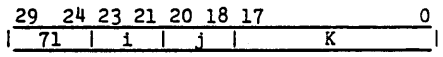
This instruction reads operands from Bj and Bk, subtracts the Bk operand from the Bj operand, and delivers the result to Bi. The difference is formed in an 18-bit one's complement mode.

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



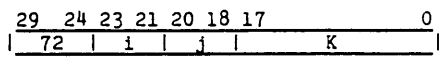
This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction reads an operand from Aj, forms the sum of the operand plus K, and delivers the result to Xi. The sum is formed in an 18-bit one's complement mode. The 18-bit result is sign-extended by copying the highest-order bit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

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



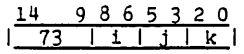
This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction reads an operand from Bj, forms the sum of the operand plus K, and delivers the result to Xi. The sum is formed in an 18-bit one's complement mode. The 18-bit result is sign-extended by copying the highest-order bit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

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



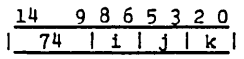
This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction reads an operand from Xj, forms the sum of the operand plus K, and delivers the result to Xi. The sum is formed in an 18-bit one's complement mode. The 18-bit result is sign-extended by copying the highest-order bit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

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



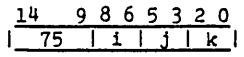
This instruction reads operands from Xj and Bk, adds the operands, and delivers the result to Xi. The sum is formed in an 18-bit one's complement mode. The 18-bit result is sign-extended by copying the highest-order bit of the result into the upper 42 bit positions in Xi.

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



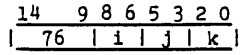
This instruction reads operands from Aj and Bk, adds the operands, and delivers the result to Xi. The sum is formed in an 18-bit one's complement mode. The 18-bit result is sign-extended by copying the highest-order bit of the result into the upper 42 bit positions in Xi.

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



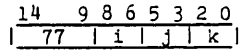
This instruction reads operands from Aj and Bk, subtracts the Bk operand from the Aj operand, and delivers the result to Xi. The difference is formed in an 18-bit one's complement mode. The 18-bit result is sign-extended by copying the highest-order bit of the result into the upper 42 bit positions in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

Approximate execution times for models 810 and 830 CP instructions are listed in tables 1-4-3 and 1-4-4 respectively. These 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.

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 I-4-3. MODEL 810 CP INSTRUCTION TIMING (1 OF 2)

Instruction Code	Description	Execution Time (Clocks)	Timing Notes
010	Return Jump	25-30	
011	Block copy (X to A)	4/word	
012	Block copy (A to X)	4/word	
013	Exchange	200-220	
014	Read UEM	30-40	
015	Write to UEM	15-20	
02	Branch to Bit K	26-30	
030	Branch Xj = 0	5-8, 26-30	1
031	Branch Xj ≠ 0	5-8, 26-30	1
032	Branch Xj = positive	5-8, 26-30	1
033	Branch Xj = negative	5-8, 26-30	1
034	Branch Xj in range	5-8, 26-30	1
035	Branch Xj out range	5-8, 26-30	1
036	Branch Xj def	5-8, 26-30	1
037	Branch Xj indef	5-8, 26-30	1
04	Branch Bi = Bj	5-8, 26-30	1
05	Branch Bi ≠ Bj	5-8, 26-30	1
06	Branch Bi ≥ Bi	5-8, 26-30	1
07	Branch Bi < Bj	5-8, 26-30	1
10	Copy Xj to Xi	2	
11	Logical Product	2	
12	Logical Sum	2	
13	Logical Difference	2	
14	Complement	2	
15	Logical Product, Comp	2	
16	Logical Sum, Comp	4	
17	Logical Diff, Comp	4	
20	Shift Left Circ	4	
21	Shift Right	4	
22	Shift Xk by Bi	5-8, 10	2
23	Shift Xk by Bi Comp	5-8, 10	2
24	Normalize	10-15	
25	Round & Norm	11-20	
26	Unpack	5-8	
27	Pack	5-8	
30	Floating Sum	45-55	
31	Floating Diff	45-55	
32	Floating DP Sum	45-55	
33	Floating DP Diff	45-55	
34	Floating Sum Round	45-55	
35	Floating Diff Round	45-55	
36	Integer Sum	5-8	
37	Integer Diff	5-8	
40	Floating Product	120-130	
41	Floating Product Round	120-130	
42	Floating DP Product	120-130	
43	Form Mask	4	
44	Floating Divide	150	
45	Floating Divide Round	150	
46	No Operation	2	
464	Move Indirect	120 + 29 (LW-1)	3
465	Move Direct	122 + 29 (LW-1)	3
466	Compare Collated	100 + 46 (LW) + MISS	3, 4
467	Compare Uncollated	100 + 46 (LW) + MISS	3, 4
47	Population Count	14 + 8 (pop-1)	
50	Ai = Aj + K	3	i = 0
50	Ai = Aj + K	34	i = 1-5
50	Ai = Aj + K	19	i = 6,7
51	Ai = Bj + K	3	i = 0
51	Ai = Bj + K	34	i = 1-5
51	Ai = Bj + K	19	i = 6,7
52	Ai = Xj + K	3	i = 0
52	Ai = Xj + K	34	i = 1-5
52	Ai = Xj + K	19	i = 6,7
53	Ai = Xj + Bk	3	i = 0
53	Ai = Xj + Bk	34	i = 1-5
53	Ai = Xj + Bk	19	i = 6,7

TABLE I-4-3. MODEL 810 CP INSTRUCTION TIMING (2 OF 2)

Instruction Code	Description	Execution Time (Clocks)	Timing Notes
54	Ai = Aj + Bk	3	i = 0
54	Ai = Aj + Bk	34	i = 1-5
54	Ai = Aj + Bk	19	i = 6,7
55	Ai = Aj - Bk	3	i = 0
55	Ai = Aj - Bk	34	i = 1-5
55	Ai = Aj - Bk	19	i = 6,7
56	Ai = Bj + Bk	3	i = 0
56	Ai = Bj + Bk	34	i = 1-5
56	Ai = Bj + Bk	19	i = 6,7
57	Ai = Bj - Bk	3	i = 0
57	Ai = Bj - Bk	35	i = 1-5
57	Ai = Bj - Bk	19	i = 6,7
60	Bi = Aj + K	3	
61	Bi = Bj + K	3	
62	Bi = Xj + K	3	
63	Bi = Xj + Bk	3	
64	Bi = Aj + Bk	3	
65	Bi = Aj = Bk	3	
66	Bi = Bj + Bk	3	
660	Read CM at (Xk)	27-36	
67	Bi = Bj - Bk	3	
670	Write Xj into CM	5-8	
70	Xi = Aj + K	3	
71	Xi = Bj + K	3	
72	Xi = Xj + K	3	
73	Xi = Xj + Bk	3	
74	Xi = Aj + Bk	3	
75	Xi = Aj - Bk	3	
76	Xi = Bj + Bk	3	
77	Xi = Bj - Bk	3	

Timing Notes:

1. First time shown if branch not taken;
   second time shown if branch was taken.

2. First time shown if left shift;
   second time shown if right shift.
   Type of shift depends on the sign.

3. LW = Number of destination fields involved.

4. MISS = 14 for no miscompare
   MISS = 103 + 66 (number of miscompare -1)

TABLE I-4-4. MODEL 830 CP INSTRUCTION TIMING (1 OF 2)

Instruction Code	Description	Execution Time (Clocks)	Timing Notes
010	Return Jump	15-20	
011	Block copy (X to A)	4/word	
012	Block copy (A to X)	4/word	
013	Exchange	200-220	
014	Read UEM	20-30	
015	Write to UEM	15-20	
02	Branch to Bit K	26-30	
030	Branch Xj = 0	5-8, 26-30	1
031	Branch Xj ≠ 0	5-8, 26-30	1
032	Branch Xj = positive	5-8, 26-30	1
033	Branch Xj = negative	5-8, 26-30	1
034	Branch Xj in range	5-8, 26-30	1
035	Branch Xj out range	5-8, 26-30	1
036	Branch Xj def	5-8, 26-30	1
037	Branch Xj indef	5-8, 26-30	1
04	Branch Bi = Bj	5-8, 26-30	1
05	Branch Bi ≠ Bj	5-8, 26-30	1
06	Branch Bi ≥ Bi	5-8, 26-30	1
07	Branch Bi < Bj	5-8, 26-30	1
10	Copy Xj to Xi	2	
11	Logical Product	2	
12	Logical Sum	2	
13	Logical Difference	2	
14	Complement	2	
15	Logical Product, Comp	2	
16	Logical Sum, Comp	4	
17	Logical Diff, Comp	4	
20	Shift Left Circ	4	
21	Shift Right	4	
22	Shift Xk by Bi	5-8, 10	2
23	Shift Xk by Bi Comp	5-8, 10	2
24	Normalize	10-15	
25	Round & Norm	11-20	
26	Unpack	5-8	
27	Pack	5-8	
30	Floating Sum	26-30	
31	Floating Diff	26-30	
32	Floating DP Sum	26-30	
33	Floating DP Diff	26-30	
34	Floating Sum Round	26-30	
35	Floating Diff Round	26-30	
36	Integer Sum	5-8	
37	Integer Diff	5-8	
40	Floating Product	100-110	
41	Floating Product Round	100-110	
42	Floating DP Product	100-110	
43	Form Mask	4	
44	Floating Divide	130	
45	Floating Divide Round	130	
46	No Operation	2	
464	Move Indirect	60 + 14 (LW-1)	3
465	Move Direct	62 + 14 (LW-1)	3
466	Compare Collated	70 + 16 (LW) + MISS	3, 4
467	Compare Uncollated	70 + 16 (LW) + MISS	3, 4
47	Population Count	14 + 8 (pop-1)	
50	Ai = Aj + K	3	i = 0
50	Ai = Aj + K	15	i = 1-7
51	Ai = Bj + K	3	i = 0
51	Ai = Bj + K	15	i = 1-7
52	Ai = Xj + K	3	i = 0
52	Ai = Xj + K	15	i = 1-7
53	Ai = Xj + Bk	3	i = 0
53	Ai = Xj + Bk	15	i = 1-7
54	Ai = Aj + Bk	3	i = 0
54	Ai = Aj + Bk	15	i = 1-7
55	Ai = Aj - Bk	3	i = 0
55	Ai = Aj - Bk	15	i = 1-7
56	Ai = Bj + Bk	3	i = 0
56	Ai = Bj + Bk	15	i = 1-7

TABLE I-4-4. MODEL 830 CP INSTRUCTION TIMING (2 OF 2)

Instruction Code	Description	Execution Time (Clocks)	Timing Notes
57	Ai = Bj - Bk	3	i = 0
57	Ai = Bj - Bk	15	i = 1-7
60	Bi = Aj + K	3	
61	Bi = Bj + K	3	
62	Bi = Xj + K	3	
63	Bi = Xj + Bk	3	
64	Bi = Aj + Bk	3	
65	Bi = Aj = Bk	3	
66	Bi = Bj + Bk	3	
660	Read CM at (Xk)	11-20	
67	Bi = Bj - Bk	3	
670	Write Xj into CM	5-8	
70	Xi = Aj + K	3	
71	Xi = Bj + K	3	
72	Xi = Xj + K	3	
73	Xi = Xj + Bk	3	
74	Xi = Aj + Bk	3	
75	Xi = Aj - Bk	3	
76	Xi = Bj + Bk	3	
77	Xi = Bj - Bk	3	

Timing Notes:

1. First time shown if branch not taken;
   second time shown if branch was taken.

2. First time shown if left shift;
   second time shown if right shift.
   Type of shift depends on the sign.

3. LW = Number of destination fields involved.

4. MISS = 14 for no miscompare
   MISS = 88 + 66 (number of miscompare -1)

PP INSTRUCTIONS

PP INSTRUCTION FORMATS

Figure I-4-2 shows PP instruction formats. PP instructions are 16 or 32 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 instructions. (Refer to table I-4-5.)

The upper 4 bits of the PP instructions must be zero to ensure that the instructions operate as defined in this section.

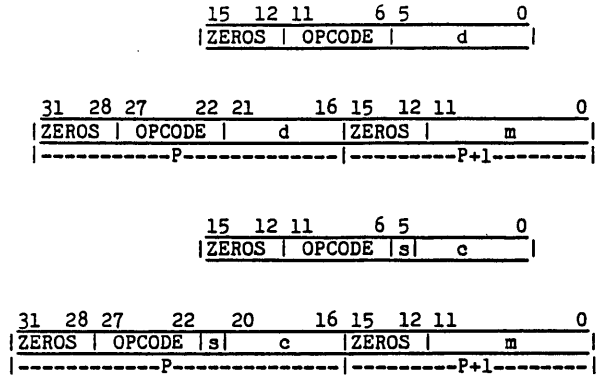


Figure I-4-2. PP Instruction Formats

TABLE I-4-5. PP NOMENCLATURE

Term	Description
Opcode	Specifies instruction operation code
s	Specifies I/O instruction sub-code
c	Specifies channel number
A	Refers to the A register (arithmetic register) or the content of the A register
(A)	Refers to the content of the word at the CM address specified by the A register
P	Refers to the P register or to the content of the P register (program address register)
R	Refers to the R register or to the content of the R register (relocation register)
(d)	Refers to the content of the word at the PP memory address specified by the d field (direct mode)
((d))	Refers to the content of the word at the PP memory address specified by the d field (indirect mode)
m+(d)	Refers to the PP memory address specified by the m field indexed by the content of the word at the PP memory addressed specified by the d field
(m+(d))	Refers to the content of the word at the PP memory address specified by the m field indexed by the content of the word at the PP memory address specified by the d field (memory mode)

PP DATA FORMAT

Figure I-4-3 shows PP data format and how 12-bit data is packed into 64-bit CM words or unpacked from 64-bit CM words.

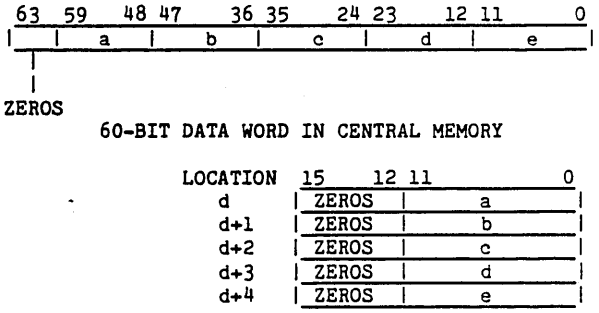


Figure I-4-3. PP Data Format

PP RELOCATION REGISTER FORMAT

Figure I-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).

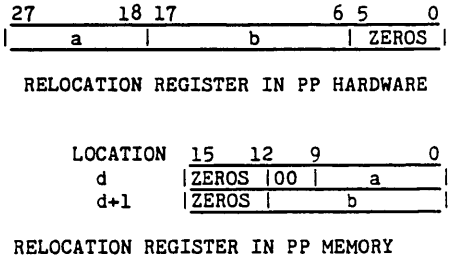
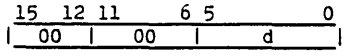


Figure 1-4-4. PP Relocation (R) Register Format

PP INSTRUCTION DESCRIPTIONS

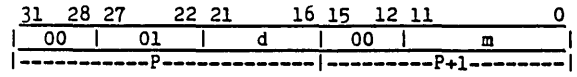
PP instruction descriptions are in numerical order. Refer to section 5, Programming Information.

00xx Pass PSN



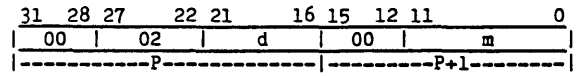
This instruction specifies that no operation is to be performed. The instruction provides a means of padding out a program.

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



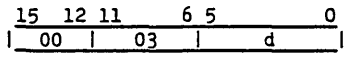
This instruction jumps to the address given by m plus the content of location d. If d equals zero, m is not modified.

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



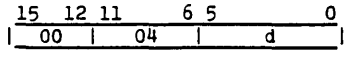
This instruction jumps to the address given by m plus the content of location d. If d equals zero, m is not modified. The current program address (P) plus 2 is stored at the jump address. The next instruction starts at the jump address plus 1. The subprogram exits with a long jump or normal sequencing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



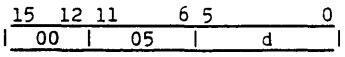
This instruction provides an unconditional jump to any address up to 31 (decimal) locations forward or backward from the current program address. The value of d is added to the current program address. If d is positive (01 through 37), 0001 through 0037 is added, and the jump is forward. If d is negative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

04d Zero Jump d ZJN d



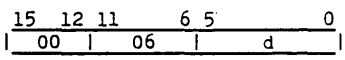
This instruction provides a conditional jump to any address up to 31 (decimal) locations forward or backward from the current program address. If the content of the A register is zero, the jump is taken. If the content of A is nonzero, the next instruction executes from P plus 1. Negative zero (777777)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

05d Nonzero Jump d NJN d



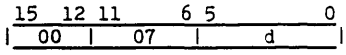
This instruction provides a conditional jump to any address up to 31 (decimal) locations forward or backward from the current program address. If the content of the A register is nonzero, the jump is taken. If the content of A is zero, the next instruction executes from P plus 1. Negative zero (777777)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

06d Plus Jump d PJN d



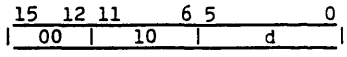
This instruction provides a conditional jump to any address up to 31 (decimal) locations forward or backward from the current program address. If the sign of the A register is positive, the jump is taken. If the sign of A is negative, the next instruction executes from P plus 1. Positive zero is treated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



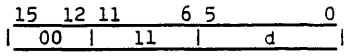
This instruction provides a conditional jump to any address up to 31 (decimal) locations forward or backward from the current program address. If the content of the A register is negative, the jump is taken. If the content of A is positive, the next instruction execute from P plus 1. Positive zero is 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



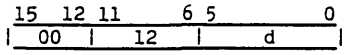
This instruction shifts the content of the A register right or left d places. If d is positive (00 through 37), the shift is left circular. If d is negative (40 through 77), the shift is right (end-off with no sign extension). Thus, d equal to 06 requires a left shift of six places; d equal to 71 requires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

11d Logical Difference d LMN d



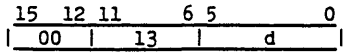
This instruction forms the bit-by-bit logical difference of d and the lower 6 bits of A in the register in A. This is equivalent to complementing individual bits of A that correspond to bits of d that are one. The upper 12 bits of A are not altered.

12d Logical Product d LPN d



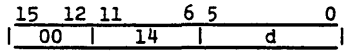
This instruction forms the bit-by-bit logical product of d and the lower 6 bits of the A register and leaves this quantity in the lower 6 bits of A. The upper 12 bits of A are zero.

13d Selective Clear d SCN d



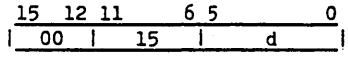
This instruction clears any of the lower 6 bits of the A register where corresponding bits of d are one. The upper 12 bits of A are not altered.

14d Load d LDN d



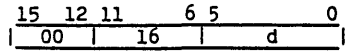
This instruction clears the A register and loads d. The upper 12 bits of A are zero.

15d Load Complement d LCN d



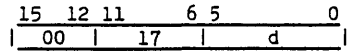
This instruction clears the A register and loads the complement of d. The upper 12 bits of A are one.

16d Add d ADN d



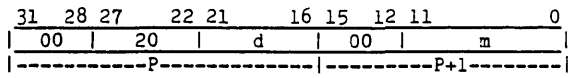
This instruction adds d (treated as a 6-bit positive quantity) to the content of the A register.

17d Subtract d SBN d



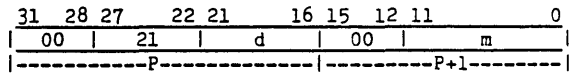
This instruction subtracts d (treated as a 6-bit positive quantity) from the content of the A register.

20dm Load dm LDC dm



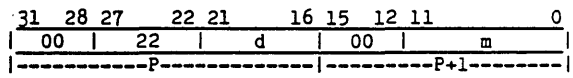
This instruction clears the A register and loads an 18-bit quantity consisting of d as the upper 6 bits and m as the lower 12 bits. The content of the location (P plus 1) which follows the present program address (P) is read to provide m.

21dm Add dm ADC dm



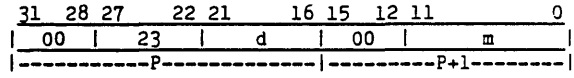
This instruction adds to the A register the 18-bit quantity consisting of d as the upper 6 bits and m as the lower 12 bits. The content of the location (P plus 1) which follows the present program address (P) is read to provide m.

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

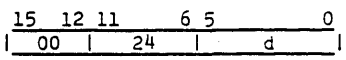


Figure I-4-4 shows R register format. If d is not equal to 0, this instruction loads the R register from bits 9 through 0 of PP memory locations d (this becomes the rightmost 10 bits of R) and from bits 11 through 0 of d plus 1 (this becomes the next 12 bitsof R). If d equals 0, the instruction is a pass.

25d Store R Register SRD d

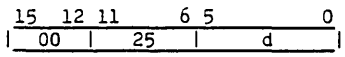
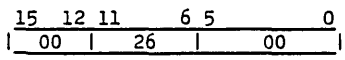


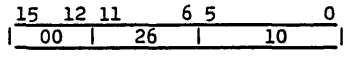
Figure I-4-4 shows R register format. If d is not equal to 0, this instruction stores the R register into PP locations d (rightmost 10 bits) and d plus 1 (next 12 bits). If d equals 0, the instruction is a pass.

2600 Exchange Jump EXN



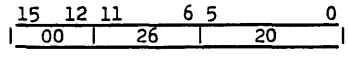
This instruction causes an unconditional exchange jump in the CP, leaving the CP CYBER 170 monitor flag unaltered. The new CYBER 170 exchange package begins at central memory location R plus A when the leftmost bit in A is set. When this bit is clear, A specifies the address. The PP waits until the exchange has been completed before proceeding with the next instruction.

2610 Monitor Exchange Jump MXN



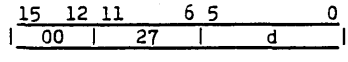
If the CP is in the CYBER 170 monitor mode, this instruction is a pass. If the CP is in the CYBER 170 job mode, it causes a CYBER 170 exchange jump in the CP, switching the CP to the CYBER 170 monitor mode (MF equals 1). The new CYBER 170 exchange package begins at central memory location R plus A when 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.

2620 Monitor Exchange Jump to MA MAN



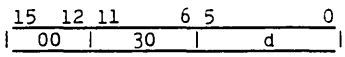
If the CP is in CYBER 170 monitor mode, this instruction is a pass. If the CP is in CYBER 170 job mode, it causes a CYBER 170 exchange jump in the CP, switching the CP to CYBER 170 monitor mode (MF equals 1). The new CYBER 170 exchange package begins at the absolute address given in the MA field of the outgoing CYBER 170 exchange package. The PP waits until the exchange has been completed before proceeding with the next instruction.

27d Pass KPT d



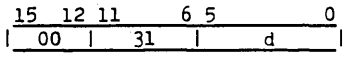
This instruction is no operation. However, it generates a pulse to a testpoint (keypoint) for optional monitoring by external equipment.

30d Load (d) LDD d



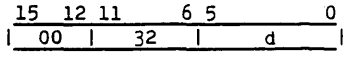
This instruction clears the A register and loads the content at location d. The upper 6 bits of A are zero.

31d Add (d) ADD d



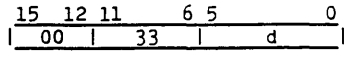
This instruction adds the content at location d (treated as a 12-bit positive quantity) to the A register.

32d Subtract (d) SBD d



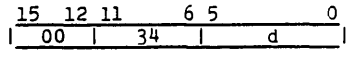
This instruction subtracts the content at location d (treated as a 12-bit positive quantity) from the A register.

33d Logical Difference (d) LMD d



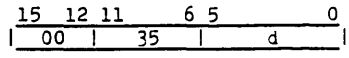
This instruction forms in the A register, the bit-by-bit logical difference of the lower 12 bits of the A register and the content at location d. This is equivalent to complementing individual bits of A which correspond to bits in location d that are ones. The upper 6 bits are not altered.

34d Store (d) STD d



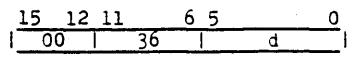
This instruction stores the lower 12 bits of the A register at location d.

35d Replace Add (d) RAD d



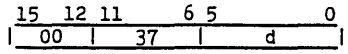
This instruction adds the quantity at location d to the content of the A register and stores the lower 12 bits of the result at location d. The result remains in A at the end of the operation and the original content of A is destroyed.

36d Replace Add One (d) AOD d



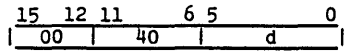
This instruction replaces the quantity at location d with its original value plus 1. The result remains in the A register at the end of the operation, and the original content of A is destroyed.

37d Replace Subtract One (d) SOD d



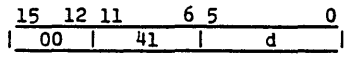
This instruction replaces the quantity at location d with its original value minus 1. The result remains in the A register at the end of the operation, and the original content of A is destroyed.

40d Load ((d)) LDI d



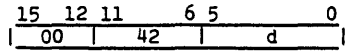
This instruction clears the A register and loads a 12-bit quantity that is obtained by indirect addressing. The upper 6 bits of A are zero. Location d is read from PPM, and the word read is used as the operand address.

41d Add ((d)) ADI d



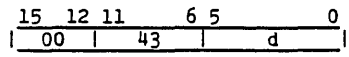
This instruction adds to the content of the A register a 12-bit operand (treated as a positive quantity) obtained by indirect addressing. Location d is read from PPM, and the word read is used as the operand address.

42d Subtract ((d)) SBI d



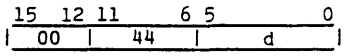
This instruction subtracts from the A register a 12-bit operand (treated as a positive quantity) obtained by indirect addressing. Location d is read from PPM, and the word read is used as the operand address.

43d Logical Difference ((d)) LMI d



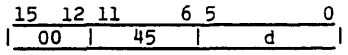
This instruction forms in the A register the bit-by-bit logical difference of the lower 12 bits of the A register and the 12-bit operand read by indirect addressing. Location d is read from PPM, and the word read is used as the operand address. The upper 6 bits of A are not altered.

44d Store ((d)) STI d



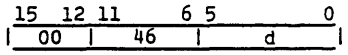
This instruction stores the lower 12 bits of the A register at the location specified by the content of location d.

45d Replace Add ((d)) RAI d



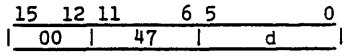
This instruction adds the operand, which is obtained from the location specified by the content at location d, to the content of the A register. The lower 12 bits of the sum replace the original operand. The result remains in A at the end of the operation.

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



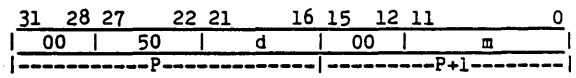
This instruction replaces the operand, which is obtained from the location specified by the content at location d, by its original value plus 1. The result remains in the A register at the end of the operation, and the original content of A is destroyed.

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



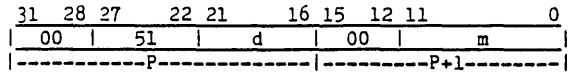
This instruction replaces the operand, which is obtained from the location specified by the content at location d, by its original value minus 1. The result remains in the A register at the end of the operation, and the original content of A is destroyed.

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



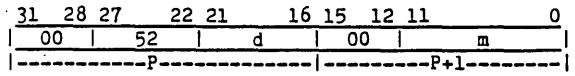
This instruction clears the A register and loads a 12-bit quantity. The upper 6 bits of A are zeros. The 12-bit operand is obtained by indexed direct addressing. The quantity m, read from PPM location P plus 1, serves as the base operand address to which the content of d is added. If d equals 0, the 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



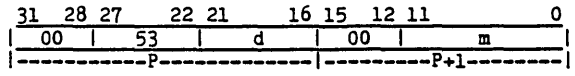
This instruction adds the 12-bit operand (treated as a positive quantity) read by indexed direct addressing (refer to 50 instruction) to the A register.

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



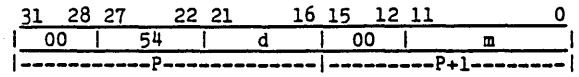
This instruction subtracts the 12-bit operand (treated as a positive quantity) read by indexed direct addressing (refer to the 50 instruction) from the A register.

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



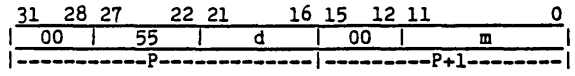
This instruction forms the bit-by-bit logical difference of the lower 12 bits of the A register and a 12-bit operand obtained by indexed direct addressing (refer to the 50 instruction) in A. The upper 6 bits of A are not altered.

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



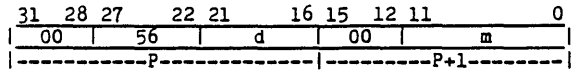
This instruction stores the lower 12 bits of the A register in the location determined by indexed direct addressing (refer to 50 instruction).

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



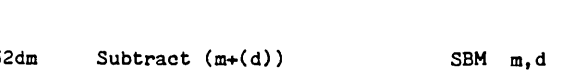
This instruction adds the operand, which is obtained from the location determined by indexed direct addressing, (refer to the 50 instruction) to the A register. The lower 12 bits of the sum replace the original operand in PPM. The result remains in A at the end of the operation, and the original content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

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



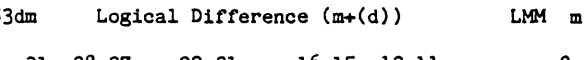
This instruction replaces the operand, which is obtained from the location determined by indexed direct addressing, (refer to 50 instruction) by its original value plus 1. The result remains in the A register at the end of the operation, and the original content of A is destroyed.

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 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 of this part). 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:

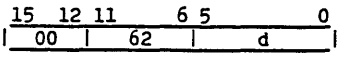
If bit 17 of A is zero, A bits 0 through 16 contain the absolute address of the first 60-bit word transferred. If bit 17 of A is one, hardware adds relocation register R to zero-extended A bits 0 through 16 to obtain the absolute address of the first 60-bit word transferred. PP location d must contain the number of 60-bit words transferred (for further information, refer to R Register under Input/Output Unit in section 2 of this part). 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.

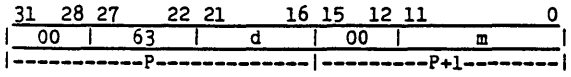
62d Central Write to (A) from d CWD d



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 (for further information, refer to R Register under Input/Output Unit in section 2 of this part). Field d gives the PP location of the first 12-bit word transferred. 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 leftmost 12 bits of the 60-bit word. The parameters of the transfer are as follows:

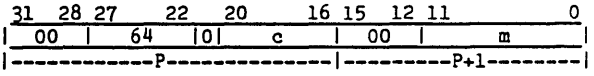
If bit 17 of A is zero, A bits 0 through 16 contain the absolute address of the first 60-bit word transferred. If bit 17 of A is one, hardware adds relocation register R to zero-extended A bits 0 through 16 to obtain the absolute address of the first 60-bit word transferred. PP location d must contain 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 of this part. 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:

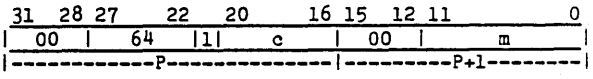
If the last word transferred is from a relative address 377776_8 and relocation is in effect, then the A register is cleared, and the value returned in A may not point to the last word transferred plus one.

640cm Jump to m if Channel c Active AJM m,c



If channel c is active this instruction causes a jump to m. Otherwise, it is a pass.

641cm Test and Set Channel c Flag SCF m,c

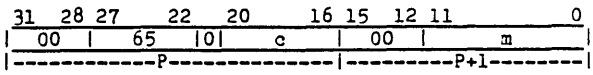


If the channel c flag is set, this instruction causes a jump to m. If the channel c flag is clear, it sets this flag and continues with the next instruction. When m is set to P plus 2, the channel flag is unconditionally set.

If two or more PPs simultaneously issue this instruction for the same channel, the conflict is resolved as follows:

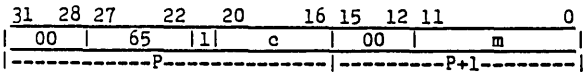
If one of the competing channels is channel 17 (maintenance channel), the PP in the lowest physical level sees the true condition of the flag; the other conflicting PPs see the flag set (and hence take a jump). If the competing channel is any other channel, software must resolve the conflict. Any five consecutively numbered PPs issue instructions at different times.

650cm Jump to m if Channel c Inactive IJM m,c



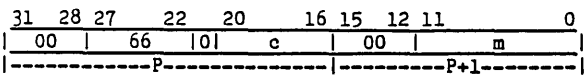
This instruction provides a conditional jump to a new address specified by m. The jump is taken if the channel specified by c is inactive. The next instruction is at P plus 2 if the channel is active.

651cm Clear Channel c Flag CCF m,c



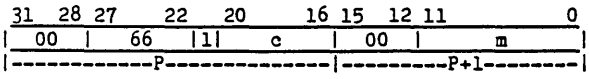
This instruction clears the channel c flag. The m field is required but not used.

660cm Jump to m if Channel c Full FJM m,c



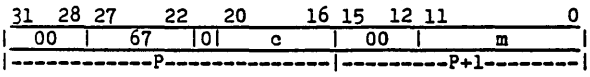
This instruction provides a conditional jump to a new address specified by m. The jump is taken if the channel designated by c is full. The next instruction is at P plus 2 if the channel is empty. An input channel is full when the input equipment places a word in the channel and that word has not been accepted by a PP. The channel is empty when a word has been accepted. An output channel is full when a PP places a word on the channel. The channel is empty when the output equipment accepts the word.

661cm Jump to m if Channel c Error Flag Set SFM m,c



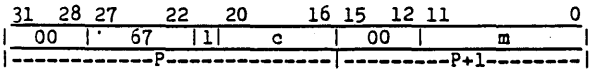
If the channel c error flag is set, this instruction clears the error flag and causes a jump to m. If this error flag is clear, the instruction is a pass. When m is set to P plus 2, the channel error flag is unconditionally cleared.

670cm Jump to m if Channel c Empty EJM m,c



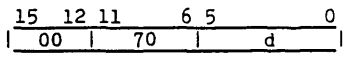
This instruction provides a conditional jump to a new address specified by m. The jump is taken if the channel specified by c is empty. The next instruction is at P plus 2 if the channel is full. (Refer to 660 instruction for explanation of full and empty.)

671cm Jump to m if Channel c Error Flag Clear CFM m,c



If the channel c error flag is clear, this instruction causes a jump to m. If this error flag is set, the instruction clears the error flag and proceeds with the next instruction. When m is set to P plus 2, the channel error flag is unconditionally cleared.

70d Input to A from Channel d IAN d

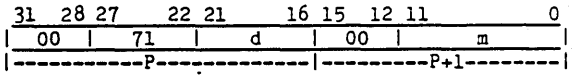


This instruction transfers a word from input channel d to the lower 12 bits of the A register. The upper 6 bits of A are cleared to zero.

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.

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



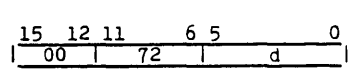
This instruction transfers a block of 12-bit words from input channel d to PPM. The first word goes to the PPM address specified by m. The A register holds the block length. A reduces by one as each word is read. The input operation completes when A equals zero or the data channel becomes inactive. If 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

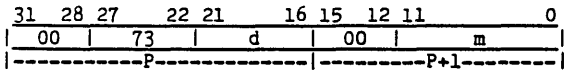


This instruction transfers a word from the A register (lower 12 bits) to output channel d.

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

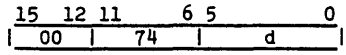


This instruction transfers a block of words from PPM to channel d. The first word is read from the address specified by m. The A register holds the number of words to be sent: A reduces by one as each word is read. The output operation completes when A equals zero or the channel becomes inactive. During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

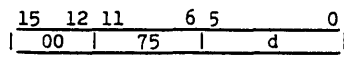


This instruction activates the channel specified by d and sends the active signal on the channel to equipment connected to the channel. Activating a channel, which must precede a 70 through 73 instruction, prepares I/O equipment for the exchange of data.

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.

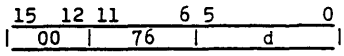
NOTE

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

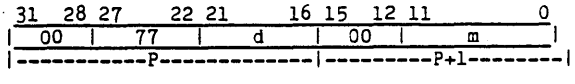


This instruction sends the external function code in the lower 12 bits of the A register on channel d.

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

Execution times for the PP instructions are listed in table I-4-6. The times listed in the execution time column assume that no conflicts occur. The timing notes refer to the notes at the end of the table. Execution times are given in 500-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 I-4-6. PP INSTRUCTION TIMING (1 of 3)

Instruction Code	Description	Execution Time (Major Cycles)	Timing Notes
00xx	Pass	1	-
01dm	Long jump to m + (d)	3	-
02dm	Return jump to m + (d)	4	-
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	-
25d	Store R register at (d) and (d) + 1	4	-
260x	Exchange jump	2	1
261x	Monitor exchange jump	2	1
262x	Monitor exchange jump to MA	2	1
27d	Pass	1	-
30d	Load (d)	2	-
31d	Add (d)	2	-
32d	Subtract (d)	2	-
33d	Logical difference (d)	2	-
34d	Store (d)	2	-
35d	Replace add (d)	4	-

TABLE I-4-6. PP INSTRUCTION TIMING (2 of 3)

Instruction Code	Description	Execution Time (Major Cycles)	Timing Notes
36d	Replace add one (d)	5	-
37d	Replace subtract one (d)	5	-
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))	5	-
46d	Replace add one ((d))	6	-
47d	Replace subtract one ((d))	6	-
50dm	Load (m + (d))	4	-
51dm	Add (m + (d))	4	-
52dm	Subtract (m + (d))	4	-
53dm	Logical difference (m + (d))	4	-
54dm	Store (m + (d))	4	-
55dm	Replace add (m + (d))	6	-
56dm	Replace add one (m + (d))	7	-
57dm	Replace subtract one (m + (d))	7	-
60d	Central read from (A) to d	12	2
61dm	Central read (d) words from (A) to m	-	2,3
62d	Central write to (A) from d	6	2
63dm	Central write (d) words to (A) from m	-	2,4
640cm	Jump to m if channel c active	2	-
641cm	Test and set channel c flag	2	-
650cm	Jump to m if channel c inactive	2	-
651cm	Clear channel c flag	2	-
660cm	Jump to m if channel c full	2	-
661cm	Jump to m if channel c error flag set	2	-
670cm	Jump to m if channel c empty	2	-
671cm	Jump to m if channel c error flag clear	2	-
70d	Input to A from channel d	2	-
71dm	Input A words to m from channel d	-	5

TABLE I-4-6. PP INSTRUCTION TIMING (3 of 3)

Instruction Code	Description	Execution Time (Major Cycles)	Timing Notes
72d	Output from A on channel d	2	-
73dm	Output (A) words from m on channel d	-	5
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:

1. No assembly-disassembly unit (ADU) conflicts and no outstanding CYBER 170 exchange jump request in the ADU.

2. No ADU conflicts. No central memory conflicts. Add a possible trip due to resynchronization (CM read instructions only).

3. 7 major cycles for instruction set up and instruction exit.
   5 major cycles for every CM word.

4. 6 major cycles for instruction set up and instruction exit.
   5 major cycles for every CM word.

5. 5 major cycles for instruction set up and exit
   1 major cycle per word (non-conflict case)
   or 2 major cycles per word (conflict case).

   Nonconflict case is when two PPs communicating to each other are not in the slot at the same time.

   Conflict case is when two PPs communicating to each other are in the slot at the same time.

This section contains special programming information such as CYBER 170 exchange jump, instruction execution, floating and fixed-point arithmetic, address formats, and data formats. This section also lists central processor error responses.

CP PROGRAMMING

CYBER 170 EXCHANGE JUMP

The CP uses a CYBER 170 exchange jump operation to switch from CYBER 170 job mode to CYBER 170 monitor mode and back again. The execution of a CYBER 170 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 a CYBER 170 exchange jump is called a CYBER 170 exchange package (figure I-5-1).

A CYBER 170 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. The address is K plus the content of the Bj register, or the monitor address, for the CP-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 A only, if bit 17 of the A register is clear, in the exchange initiated by IOU instructions 2600 or 2610. 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 CYBER 170 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

Reference address for UEM (RAE) - 21 bits (lower 6 bits are assumed to be zeros)

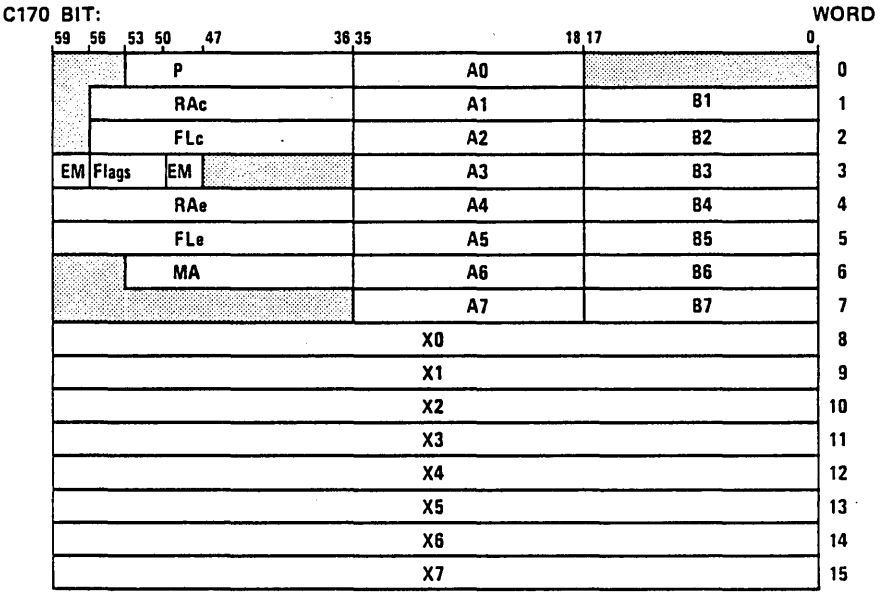


Figure I-5-1. CYBER 170 Exchange Package

Field length of block transfer for UEM (FLE) - 24 bits (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

A particular CYBER 170 exchange package resides in the CP hardware registers for the length of the execution interval. The execution interval begins with a CYBER 170 exchange jump that swaps the CYBER 170 exchange package information in CM with the information contained in the CP registers. The execution interval ends with the next CYBER 170 exchange jump. A hardware flag called a CYBER 170 monitor flag (MF) indicates the type of program the CP is executing.

With the flag set, the CP is in noninterruptible CYBER 170 monitor mode. With the flag clear, the CP is in an interruptible program (job) mode. A master clear (deadstart) clears the CYBER 170 monitor flag.

VIRTUAL STATE

The Virtual State is implemented by a combination of hardware, software, and microcode. This state handles the following:

- Processor detected hardware errors.
- Hardware integrity verification (diagnostics).
- CP software errors and exceptions if they occur with the CP in CYBER 170 monitor mode.

The following CP conditions cause programs to interrupt to Virtual State:

- Unimplemented instruction (illegal, 00) in CYBER 170 monitor mode.
- CP detected hardware faults.
- Software errors in CYBER 170 monitor mode.

In general, Virtual State determines the cause of an interrupt and decides whether to return the CP to the interrupted mode, to halt the CP, or to simulate a CYBER 170 exchange and return control to CYBER 170 monitor mode. Refer to Error Response, this section.

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 I-5-2), 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.

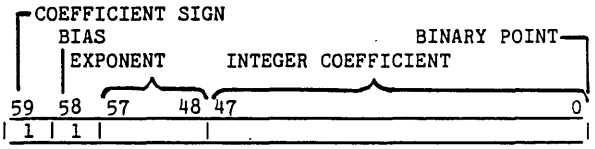


Figure I-5-2. Floating-Point Format

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

TABLE I-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 I-5-2 through I-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 I-5-3.

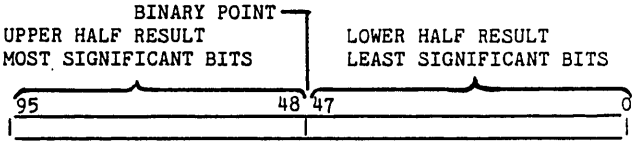


Figure I-5-3. Floating-Add Result Format

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 I-5-4.

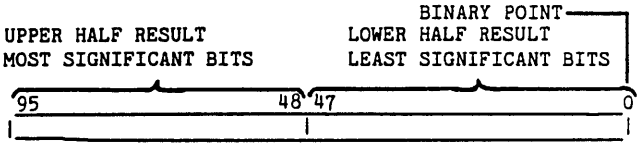


Figure I-5-4. Multiply Result Format

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 I-5-2. Xj PLUS Xk
(30, 32, 34 INSTRUCTIONS)

		Xk			
		W	+INF	-INF	+IND
Xj	W	---	+INF	-INF	IND
	+INF	+INF	+INF	IND	IND
	-INF	-INF	IND	-INF	IND
	±IND	IND	IND	IND	IND

TABLE I-5-3. Xj MINUS Xk
(31, 33, 35 INSTRUCTIONS)

		Xk			
		W	+INF	-INF	+IND
Xj	W	---	-INF	+INF	IND
	+INF	+INF	IND	+INF	IND
	-INF	-INF	-INF	IND	IND
	±IND	IND	IND	IND	IND

TABLE I-5-4. Xj MULTIPLIED BY Xk (40, 41, 42 INSTRUCTIONS)

		Xk						
		+N	-N	+0	-0	+INF	-INF	+IND
Xj	+N	-----	-----	0	0	+INF	-INF	IND
	-N	-----	-----	0	0	-INF	+INF	IND
	+0	0	0	Integer* Multiply		IND	IND	IND
	-0	-0	0			IND	IND	IND
	+INF	+INF	-INF	IND	IND	+INF	-INF	IND
	-INF	-INF	+INF	IND	IND	-INF	+INF	IND
	±IND	IND	IND	IND	IND	IND	IND	IND

* If both operands used in the interger multiply are normalised, an underflow results.

TABLE I-5-5. Xj DIVIDED BY Xk (44,45 INSTRUCTIONS)

		Xk						
		+N	-N	+0	-0	+INF	-INF	+IND
Xj	+N	-----	-----	+INF	-INF	0	0	IND
	-N	-----	-----	-INF	+INF	0	0	IND
	+0	0	0	IND	IND	0	0	IND
	-0	0	0	IND	IND	0	0	IND
	+INF	+INF	-INF	+INF	-INF	IND	IND	IND
	-INF	-INF	+INF	-INF	+INF	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 $\pm$ 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.

COMPARE/MOVE ARITHMETIC

The compare/move arithmetic provides multiple character manipulation. The characters are 6 bits long. Characters can be moved from one CM location to another, and fields of characters can be compared either directly or through a collate table.

The move direct instruction moves a field of up to 127 characters from one location to another location as specified in the instruction. The move indirect instruction performs the same kind of move, but a CM reference is used to obtain the parameters. The move indirect instruction moves a field of up to 8181 characters.

The compare collated instruction compares two fields of up to 127 characters. When two characters are unequal, the characters are referenced in a collate table, and the values are compared. If those values are unequal, the field with the larger character is indicated. The compare uncollated instruction compares two fields of up to 127 characters and indicates the larger of the first character pair that is found to be unequal.

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 in the EM register, 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. Refer to figure I-5-5 and table I-5-6.

The CP has the following error conditions: illegal instructions, hardware errors, and conditional software errors.

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 the CYBER 170 job mode, illegal instructions cause an exchange to the CYBER 170 monitor mode. In the CYBER 170 monitor mode, they cause a simulated CP halt. CP illegal instructions are:

- 017
- 011, 012, 013, 464, 465, 466, 467 if they do not begin at parcel 0
- 011, 012, 014, 015 if the UEM enable flag in the flag register of the CYBER 170 exchange package is clear
- 011, 012 if both X0 bit 23 and FLE bit 23 of the CYBER 170 exchange package are set
- Any 30-bit instruction which begins at parcel 3.

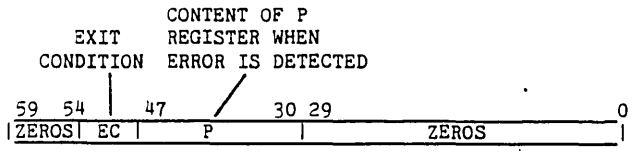


Figure I-5-5. Format of Exit Condition Register at (RAC)

TABLE I-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-range error (bit 48) 02_8 — Floating-point infinite (bit 49) 04_8 — Floating-point indefinite (bit 50) 20_8 — Processor-detected malfunction 67_8 — Hardware malfunction
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.
	NOTE: Nonzero information in bits 0 through 29 is error status for customer engineering and maintenance.

Hardware Errors

CP/CM hardware errors are: data parity errors, address parity errors, and double bit errors. If the CP is in CYBER 170 job mode, a hardware error causes a jump to CYBER 170 monitor mode. If the CP is in CYBER 170 monitor mode when a hardware error occurs, the CP performs a simulated halt. 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 that depends on bits set in the EM field in the current CYBER 170 exchange package. If the bit associated with the type of error is clear, the error is ignored in both CYBER 170 job and CYBER 170 monitor modes. With the bit set and the error in CYBER 170 monitor mode, an interrupt to Virtual State results. With the bit set and the error in CYBER 170 job mode, an exchange to CYBER 170 monitor mode results.

CM PROGRAMMING

All CM to CP references for instructions or read/write data are made relative to RAC. RAC defines the lower limit and FLC added to RAC defines the upper limit of the CM program addresses. 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 a CYBER 170 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 CYBER 170 exchange jump.

Figure I-5-6 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 memory addresses:

$$RAC \leq (RAC + P) < (RAC + FLC)$$

For relative memory addresses:

$$0 \leq P < FLC$$

DIRECT READ/WRITE (INSTRUCTIONS 014,015,660,670)

These instructions transfer one 60-bit word between the selected X register and a memory location, using a 21-bit relative address. Instructions 660 and 670 use the memory address Xk (21 bits) plus RAC (21 bits) to address CM. Instructions 014 and 015 use the memory address Xk (21 bits) plus RAE (21 bits) to address UEM.

BLOCK COPY (INSTRUCTIONS 011, 012)

These instructions transfer up to 131,071 60-bit words between fields in memory. The UEM address is obtained from X0 plus RAE. The CM address is obtained from A0 plus RAE (if the block copy flag is clear) or X0 bits 30 through 50 plus RAC (if the block copy flag is set).

The transfers occur in blocks of up to 64 words, during which other CP activities are suspended; therefore, transfer speeds up to one 60-bit word each 300 nanoseconds can be obtained for model 830, and one word every 1250 nanoseconds for model 810.

These instructions are 30-bit instructions which must start at parcel 0. If the UEM address used has bit 21 or bit 22 set, the next instruction is taken from parcel 2, the same instruction word. In case of a block read, a transfer of all zeros can be made to central memory by setting bit 21 or 22 of the address (X0 + RAE) provided FLE is sufficiently large. Tables I-5-7 and I-5-8 list CP error response.

TABLE I-5-7. ERROR EXITS IN CYBER 170 MONITOR MODE (MF=1) (1 of 2)

Error Condition	Error Response	
	Exit Mode Selected	Exit Mode Not Selected
Illegal instruction or 00	1. The instruction is not executed. 2. Store P and exit condition bits (00) at location RAC. P equals address of illegal instruction. 3. Interrupt to Virtual State. 4. CP stops in Virtual State.	1. N/A (exit mode is always selected).
Exit condition bit 48 set by an incremental read with an address out of range (AOR).	1. The X register is unchanged. 2. The A register contains the AOR address. 3. Store P and exit condition bits (01) at location RAC. P equals address of increment instruction or address of instruction following the increment. 4. Interrupt to Virtual State. 5. CP stops in Virtual State.	1. Inhibit read, X unchanged. 2. Continue execution.
Exit condition bit 48 set by an incremental write with an address out of range (AOR).	1. Block write operation; content of CM is unchanged. 2. The A register contains the AOR address. 3. Store P and exit condition bits (01) at location RAC. P equals address of instruction or address of instruction following the increment. 4. Interrupt to Virtual State. 5. CP stops in Virtual State.	1. Inhibit write, CM unchanged. 2. Continue execution.
Exit condition bit 48 set by an RNI or branch address out of range.	1. Inhibit execution. 2. Store P and exit condition bits (01) at location RAC. P equals address of instruction required by RNI or address of branch destination instruction. 3. Interrupt to Virtual State. 4. CP stops in Virtual State.	1. N/A (exit mode is always selected regardless of status of EM Register bit 48).

TABLE I-5-7. ERROR EXITS, CYBER 170 MONITOR MODE (MF=1) (2 of 2)

Error Condition	Error Response	
	Exit Mode Selected	Exit Mode Not Selected
Exit condition bit 48 set on CMU instruction 1. C1 or C2 greater than 9. 2. K1 or K2 address out of range.	1. Condition 1 omits reading/writing; CM is unchanged. Condition 2 causes the instruction to go unexecuted. 2. Stores P and exit bits (01) at RAC. 3. Interrupt to Virtual State 4. CP stops in Virtual State.	1. Condition 1 omits reading/writing; CM is unchanged. Condition 2 causes the instruction to go unexecuted. 2. Continue with next instruction.
Exit condition bit 48 set by a UEM address range check for instructions 011 and 012.	1. Execute instruction as a pass. 2. Store P and exit bits (01) at RAC. 3. Interrupt to Virtual State. bits (01) at RAC. 4. CP stops in Virtual State.	1. Execute instruction as a pass. 2. Interrupt to Virtual State. 3. Exit to next 60-bit word and continue execution.
Exit condition bit 48 set by a UEM address range check for instructions 014 and 015.	1. Execute instruction as a pass. 2. Store P and exit condition bits (01) at RAC. P equals address of following instruction. 3. Interrupt to Virtual State. 4. CP stops in Virtual State.	1. Execute instruction as a pass. 2. Exit to next parcel and continue execution.
Exit condition bit 49 set by infinite condition, or bit 50 set by indefinite conditions.	1. Store P and exit condition bits (02 for infinite or 04 for indefinite). P equals address of arithmetic instruction or address of instruction following. 2. Interrupt to Virtual State. 3. CP stops in Virtual State.	1. Continue execution.
Any hardware parity error or double SECDED error.	1. Interrupt to Virtual State. 2. Virtual State stores P and exit condition bits (20) at RAC. 3. CP stops in Virtual State.	1. Interrupt to Virtual State. 2. Virtual State stores P and exit condition bits (20) at RAC. 3. CP stops in Virtual State.

TABLE I-5-8. ERROR EXITS IN CYBER 170 JOB MODE (MF=0) (1 of 2)

Error Condition	Error Response	
	Exit Mode Selected	Exit Mode Not Selected
Illegal instruction or 00 instructions.	1. The instruction is not executed. 2. Store P and exit condition bits (00) at location RAC. P equals address of illegal instruction. 3. Exchange jump to MA and set CYBER 170 MF.	1. N/A (exit mode is always selected).
Exit condition bit 48 set by an incremental read with an address out of range (AOR).	1. The X register is unchanged. 2. The A register contains the AOR address. 3. Store P and exit condition bits (01) at location RAC. P equals address of increment instruction or address of instruction following the increment. 4. Exchange jump to MA and set CYBER 170 MF.	1. Inhibit read, X unchanged. 2. Continue execution.
Exit condition bit 48 set by an incremental write with an address out of range (AOR).	1. Block write operation; content of CM is unchanged. 2. The A register contains the AOR address. 3. Store P and exit condition bits (01) at location RAC. P equals address of instruction or address of instruction following the increment. 4. Exchange jump to MA and set CYBER 170 MF.	1. Inhibit write, CM unchanged. 2. Continue execution.
Exit condition bit 48 set by an RNI or branch address out of range.	1. Inhibit execution. 2. Store P and exit condition bits (01) at location RAC. P equals address of instruction required by RNI or address of branch destination instruction. 3. Exchange jump to MA and set CYBER 170 MF.	1. N/A (exit mode is always selected regardless of status of EM register bit 48).

TABLE I-5-8. ERROR EXITS IN CYBER 170 JOB MODE (MF=0) (2 of 2)

Error Condition	Error Response	
	Exit Mode Selected	Exit Mode Not Selected
Exit condition bit 48 set on CMU instruction. 1. C1 or C2 greater than 9. 2. K1 or K2 address out of range.	1. Condition 1 omits reading/writing; CM is unchanged. Conditon 2 causes the instruction to go unexecuted. 2. Stores P and exit bits (01) at RAC. 3. Exchange jump to MA and set CYBER 170 MF.	1. Condition 1 omits reading/writing; CM is unchanged. Condition 2 causes the instruction to go unexecuted. 2. Continue with next instruction.
Exit condition bit 48 set by a UEM address range check for instructions 011 and 012.	1. Execute instruction as a pass. 2. Store P and exit bits (01) at RAC. 3. Exchange jump to MA and set CYBER 170 MF.	1. Execute instruction as a pass. 2. Exit to next 60-bit word and continue execution.
Exit condition bit 48 set by a UEM address range check for instructions 014 and 015.	1. Execute instruction as a pass. 2. Stop CP. 3. Store P and exit condition bits (01) at location RAC. 4. Exchange jump to MA and set CYBER 170 MF.	1. Execute instruction as a pass. 2. Exit to next parcel and continue execution.
Exit condition bit 49 set by infinite condition, or bit 50 set by indefinite condition.	1. Store P and exit condition bits (02 for infinite or 04 for indefinite). P equals address of arithmetic instruction or address of instruction following. 2. Exchange jump to MA and set CYBER 170 MF.	1. Continue execution.
Any hardware parity error or double SECDED error.	1. Interrupt to Virtual State. 2. Virtual State stores P and exit condition bits (20) at RAC. 3. Exchange jump to MA and set CYBER 170 MF.	1. Interrupt to Virtual State. 2. Virtual State stores P and exit condition bits (20) at RAC. 3. Exchange jump to MA and set CYBER 170 MF.

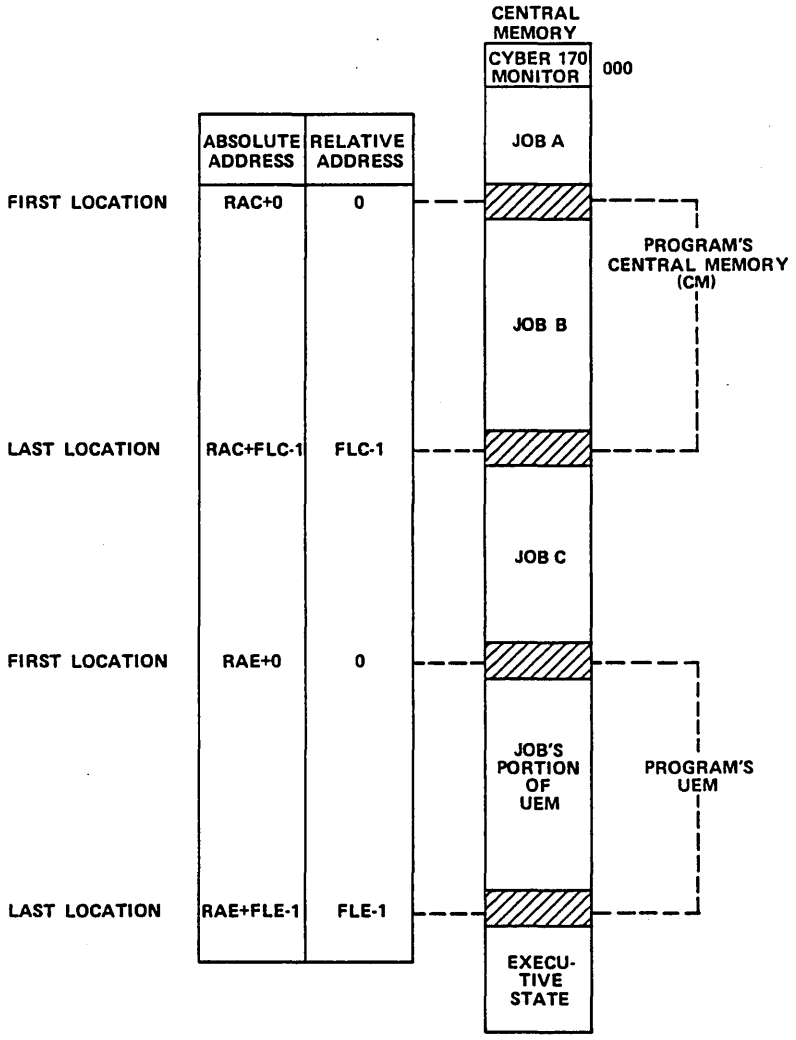


Figure I-5-6. Memory Map

PP PROGRAMMING

The PPs have access to all CM storage locations. One 64-bit word or a block of 64-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 64-bit CM word, with the leftmost 4 bits equal to zero.) 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, described next.

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 but the bounds register in central memory may be set to limit write access from IOU.

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 28-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 I-4-4 shows how R is stored in PP memory.

PP MEMORY ADDRESSING BY PPs

PP instructions use 6-bit or 18-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 any conflicts have been resolved, proceed as follows:

	Action	Typical Instruction
1.	Clear error flag.	Jump if error flag set, and clear flag (661).
2.	Verify inactive status.	Jump if active (640).
3.	Verify read status.	
	Prepare for reading summary status.	Function m (77).
	Verify that the device responded.	Jump if active (640).
	Activate channel.	Activate (74).
	Read summary status.	Input to A (70).
	Verify error flag clear.	Jump if error flag set (661).
	Analyze summary status.	Logical product (12). Zero jump (04).
4.	Enter number of words to A.	Load d (14).
5.	Prepare for input/ output.	
	Verify inactive status.	Jump if active (640).
	Prepare for read/write.	Function m (77).
	Verify that the device responded.	Jump if active (640).
6.	Read/write data.	
	Activate channel.	Activate (74).
	Read/write data.	Input/output A words (71/73).
	If write, loop until empty.	Jump if full (660).
	Disconnect channel.	Deactivate (75).
	Verify inactive status.	Jump if active (640).
7.	Verify transfer integrity.	
	Verify A words were transferred (see note 1).	Nonzero jump (05).
	Verify error flag clear.	Jump if error flag set (661).
	Verify inactive status.	Jump if active (640).
	Prepare for reading device status.	Function m (77).
	Verify that the device responded.	Jump if active (640).
	Activate channel.	Activate (74).
	Read device status.	Input to A (70).
	Verify error flag clear.	Jump if error flag set (661).
	Analyze device status.	Logical product (12). Nonzero jump (05).
	Disconnect channel.	Deactivate (75).

NOTE

If A equals the original value, no words were transferred.

If A is not equal to 0, the device or another PP ended 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	1	2	3	4	5
PPs	00,20	01,21	02,22	03,23	04,24

Slot	6	7	10	11	12
PPs	05,25	06,26	07,27	10,30	11,31

Note Numbers in octal

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 OPERATIONS

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 (Marker) Flag Instructions (641, 651)

This flag is used by software as a marker and does not affect hardware operation. When PPs in the same time slot use this flag, priority conflicts exist. For channel 17_8 (maintenance channel) marker flag, priority problems are resolved by hardware. For other channels, such conflicts must be resolved by software. Refer to Inter PP Communications, this section, for a list of PPs in each time slot.

Error Flag Instructions (661, 671)

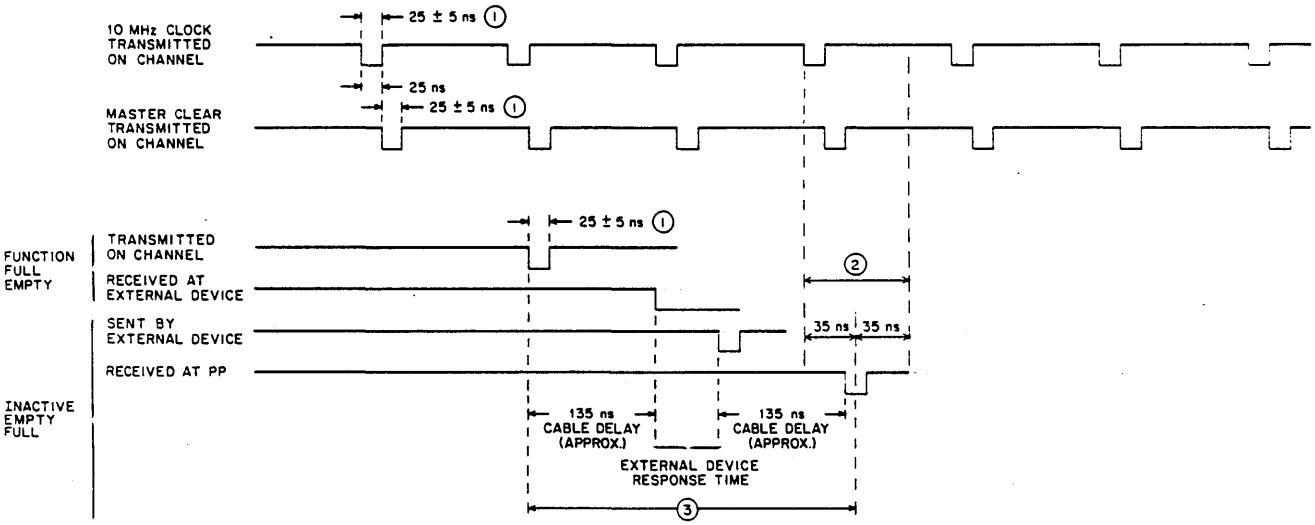
This flag indicates an input data parity error on the specific channel being tested. It also indicates an output data parity error on channels which have the capability of sending an error signal to the IOU in case of such an error. The status register of the device concerned must be read to verify output data integrity.

Channel Transfer Timing

Figure I-5-7 shows channel transfer timing. All signal pulses are 25 $\pm$ 5 nanoseconds in width and occur 25 $\pm$ 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 $\pm$ 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 $\pm$2 percent. The peripheral devices used must tolerate this frequency variation.



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 I-5-7. Channel Transfer Timing

INPUT/OUTPUT TRANSFERS

Data Input Sequence

The external device sends data (figure I-5-8) 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 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 I-5-9) 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.

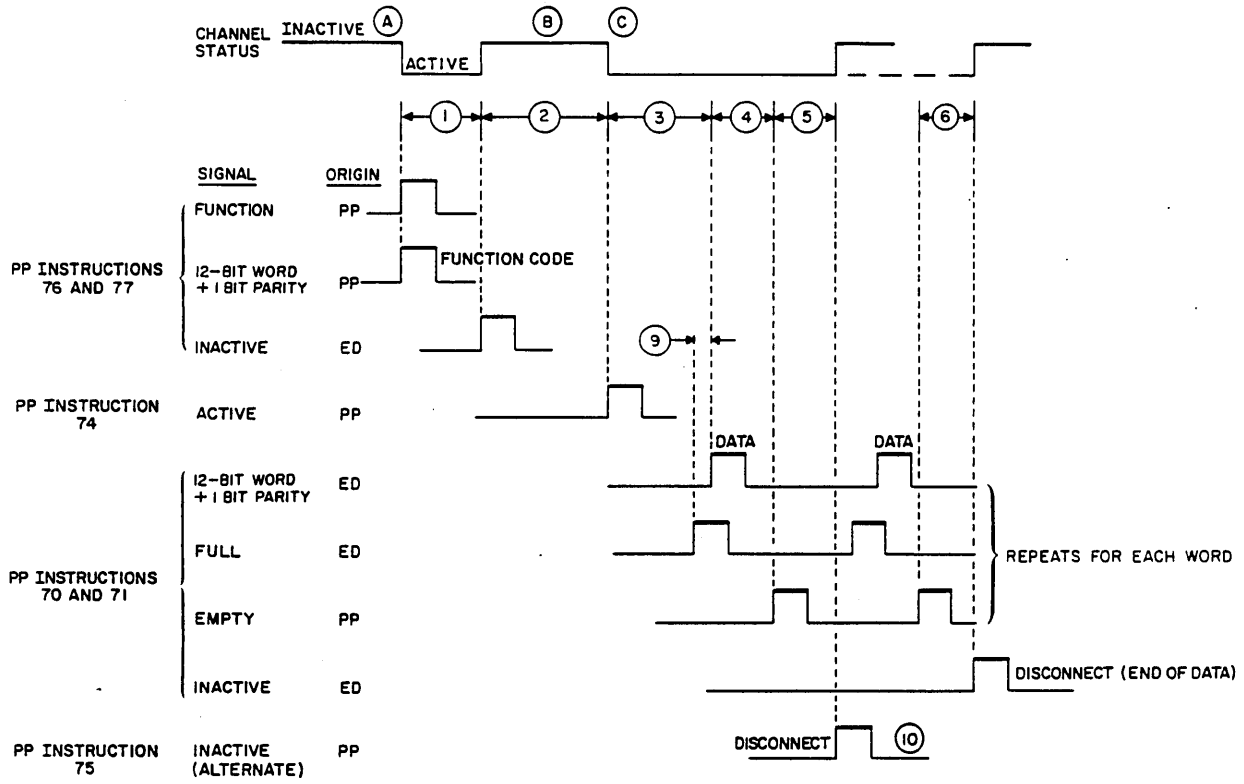
3. The PP sets the channel active flag and sends an active signal to the controller which signals the output equipment that data flow is starting.

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:

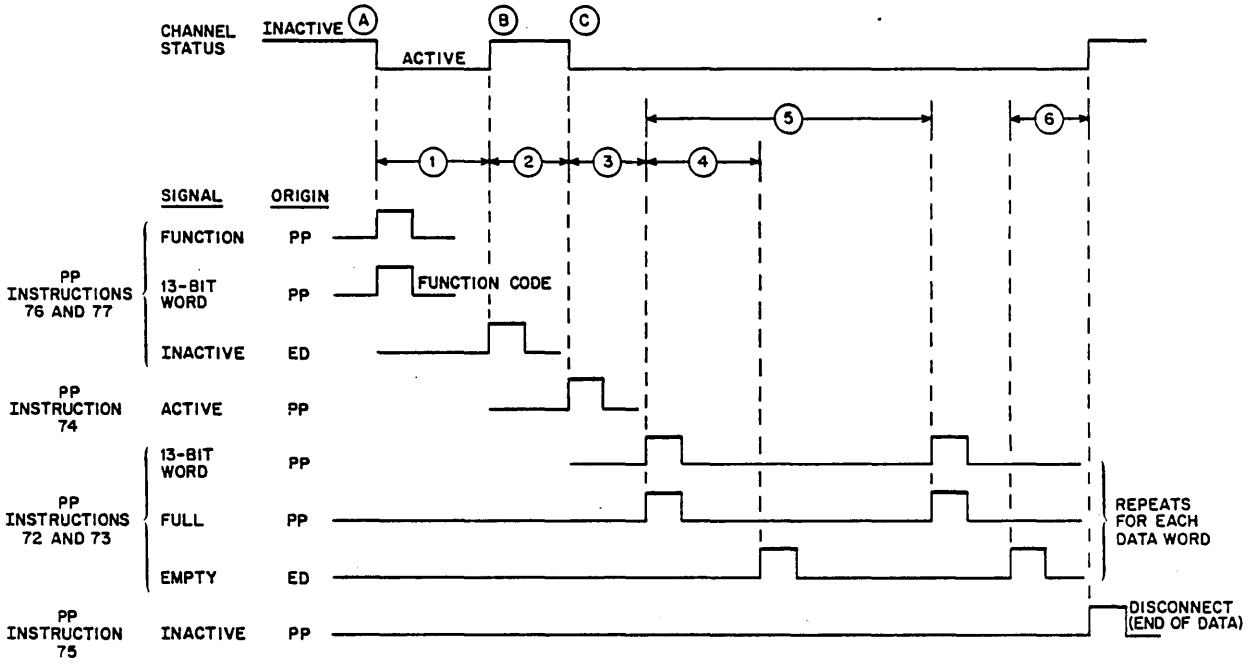
1. TIME IS A FUNCTION OF EXTERNAL DEVICE (ED). PP RECOGNIZES INACTIVE 1 MAJOR CYCLE (OR A MULTIPLE OF MAJOR CYCLES) AFTER FUNCTION. THE PP MUST PREVIOUSLY RECEIVE INACTIVE.
2. TIME IS A FUNCTION OF PERIPHERAL PROCESSOR (PP). MINIMUM TIME IS 1 MINOR CYCLE. ACTUAL TIME IS A FUNCTION OF THE PP PROGRAM.
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.
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 I-5-8. 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.

A. CHANNEL MUST BE PREVIOUSLY INACTIVE.

B. CHANNEL REMAINS ACTIVE UNTIL ED SENDS INACTIVE.

C. CHANNEL MUST BE PREVIOUSLY INACTIVE.

Figure I-5-9. Data Output Sequence Timing

DISPLAY STATION PROGRAMMING (CC545)

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 inputs one character from the keyboard. This character enters as the lower 6 bits of the word with the upper bits cleared. There is no status report by the keyboard. Table I-5-9 lists the keyboard character codes.

TABLE I-5-9. 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 11 inch area of a cathode-ray tube (CRT). The display can be in character mode (alphanumeric) and/or dot mode (graphic). There are two presentation areas (left and right) displayed. Each is made up of 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 I-5-10 lists the display character codes.

TABLE I-5-10. 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 I-5-10 illustrates the function word format. The word following the function word specifies the starting coordinates for the display (for either mode). Figure I-5-11 illustrates coordinate data word. In character mode, the following words are display character codes. Figure I-5-12 illustrates the character data word.

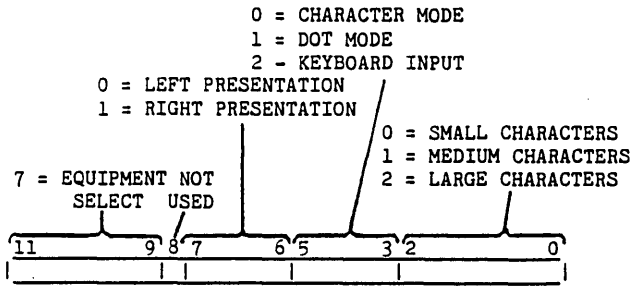
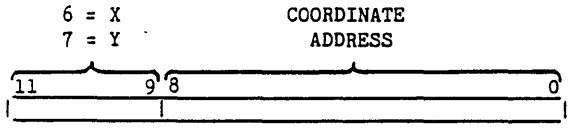


Figure I-5-10. Display Station Output Function Code



NOTE: IN DOT MODE, EACH Y COORDINATE TRANSMITTED FORCES A DOT DISPLAY.

Figure I-5-11. Coordinate Data Word

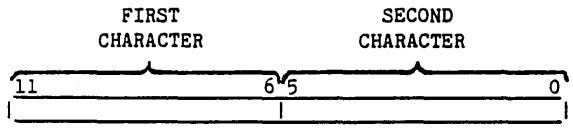


Figure I-5-12. Character Data Word

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 I-5-13) requests an input of one line of data from the display station and displays this data on the CRT as it is being typed.

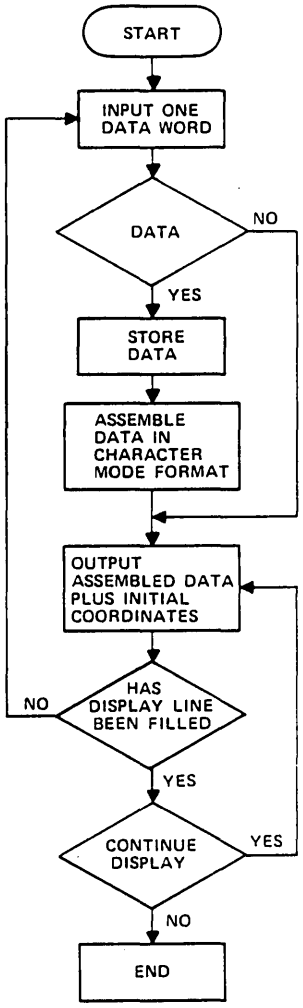


Figure I-5-13. Receive and Display Program Flowchart

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}.

TWO-PORT MULTIPLEXER PROGRAMMING

Channel 15_8 is reserved for communications with one or two external devices through the two-port multiplexer. One port is reserved for maintenance purposes and the other is reserved for future use. The two-port multiplexer can communicate with all external devices which use EIA standard RS-232C serial interface. It can also communicate with telephone dial-out equipment through a RS-366A interface (port 1 only), and accommodate auto answer equipment (both ports). It also has a remote deadstart and power on/off capability, controlled by a four-position switch.

The two-port multiplexer supports data communication using ASCII code only. It 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 19200 baud. Each port has a 64-character output buffer and a 16-character input buffer.

NOTE

Bit numbering is 0 through 63 from left to right.

TWO-PORT MULTIPLEXER OPERATION

The two-port multiplexer 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.

Code	Functions
7XXX	Terminal select
6XXX	Terminal deselect
1X04	Read calendar clock
1X05	Write calendar clock
1X06	Write auto dial-out data
1X07	Read auto dial-out status
1X10	Abandon call
00XX	Read status summary
01XX	Read terminal data
02XX	Write output buffer
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 two-port multiplexer from channel 15_8 so the 16-bit channel is available for inter-PP communications.

Deselecting a port does not affect any operations going on between the two-port mux and external equipment. For example, if an output operation (02XX function) has just been performed and the 64 character output buffer is full, the two-port mux continues to empty the buffer after the port is deselected.

Read Calendar Clock (1X04)

A select port 0 or port 1 function code must be issued before the calendar clock can be read. Function 1X04 initiates an eight-word data input with format as follows:

Status Information

	56	57	58	59	60	61	62	63
WORD 0	0	0	0	0	0	0	0	S

	Tens of Years				Units of Years			
WORD 1	Y	Y	Y	Y	y	y	y	y

	Tens of Months			Units of Months				
WORD 2	0	0	0	M	m	m	m	m

	Tens of Days				Units of Days			
WORD 3	0	0	D	D	d	d	d	d

	Tens of Hours				Units of Hours			
WORD 4	0	0	H	H	h	h	h	h

	Tens of Minutes				Units of Minutes			
WORD 5	0	M	M	M	m	m	m	m

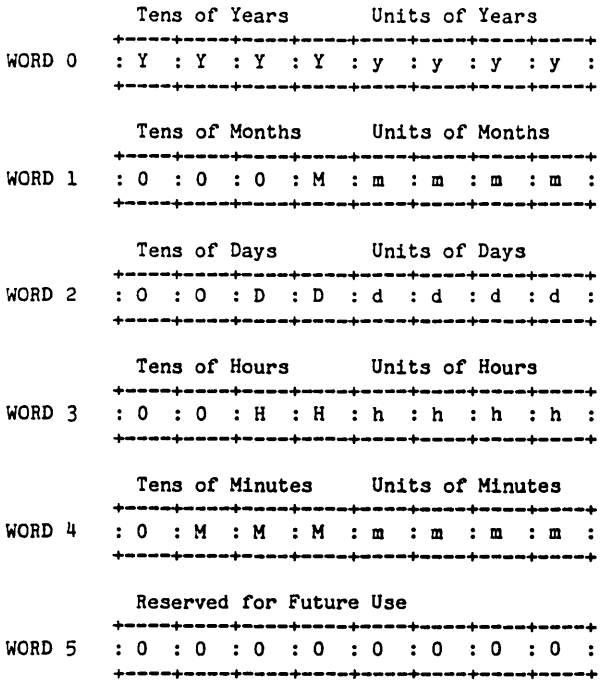
	Tens of Seconds				Units of Seconds			
WORD 6	0	S	S	S	s	s	s	s

	Reserved for Future Use							
WORD 7	0	0	0	0	0	0	0	0

Bit 63, when set, indicates that there has been a power failure and the clock data is no longer valid.

Write Calendar Clock (1X05)

A select port 0 or port 1 function code must be issued before the calendar clock can be written. The format of data sent by the 1X05 function is as follows:



The smallest time unit that can be set is minutes. Tenths, units, and tens of seconds are set to zero. Writing the wall clock automatically resets bit 63 of the status word to zero to indicate that the wall clock data is valid.

Write Auto Dial-Out Data (1X06)

Port 1 must be selected before this function can be issued. Function 1X06 conditions the two-port mux to pass sequential 8-bit output characters to the auto dial-out equipment as 4-bit numbers/codes as follows:

Channel Bits 56/60	57/61	58/62	59/63	Number/Code
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	*
1	0	1	1	#
1	1	0	0	End of number
1	1	0	1	Wait for 2nd tone
1	1	1	0	Unassigned
1	1	1	1	Unassigned

Bits 56 - 59 represent the first number/code output, and bits 60 - 63 the second. An End of Number code must follow the last telephone number in the data. If there is an even number of digits in the telephone number, this code is in the four most significant bits of the last word and the four least significant bits are ignored.

The function, 1X06 may be issued only when the auto dial-out status bit 62 is a zero and bit 63 is a one (line not occupied and power on). While the two-port mux is dialing, the PPs may monitor the status of the call by looking at the Abandon Call and Call Origination Status status bits, described next.

Read Auto Dial-Out Status (1X07)

Port 1 must be selected before this function is issued. The function, 1X07 causes a one-word input with the following format.

Bit	Description
52-59	Not Used
60	Abandon call
61	Call origination status
62	Data line occupied
63	Power on

Abandon Call (1X10)

This function aborts the current call.

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 multiplexer 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).

Overrun (Bit 54)

The overrun bit sets, when the earliest previously received character is not read by the PP before the fourth character arrives.

Data Set Ready (DSR) and Data Carrier Detector (DCD) (Bit 53)

This bit sets when both data set ready and data carrier detector signals are active.

Data Set Ready (Bit 52)

This bit sets when the data set ready signal is active.

PP Write Output Buffer (02XX)

This code prepares the multiplexer for an output operation to the 64-character output buffer 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 in- active (instead of empty) signal is sent back to the channel, which in turn will terminate 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	Enable loop-back This bit, when set, enables a round trip data path from channel 15_8 to the selected RS232C port (UART chip) and back to channel 15_8 When in loop back mode, the UART chip does not transmit data externally.
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.
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.

Bit 61	Bit 62	Bits per Character
0	0	5
0	1	6
1	0	7
1	1	8

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 output buffer memory and UART. The terminal operation mode register and terminal control signals are not cleared.

Device Initiated Functions

The two-port mux has the capability to react to a signal on a separate line from suitable external devices, and to special ASCII code sequences input over the data line. In either case, two-port mux status bit 63 (Ring Indicator) sets. As the two-port mux operates from 60-Hz power, this capability exists even when the computer (400-Hz) power is off and includes the capability to turn the computer power on.

Status bit 63 is monitored by a microcomputer in the two-port mux. The action taken depends on the position of the port option switch for each port as follows:

Switch Position	Description
Disabled	The port is disabled for all input and output operations. No data can be sent out and all incoming signals are ignored.
MSG Only	The port is enabled for system originated functions only. These functions may be output or input operations. Remote power control and remote deadstart are disabled.
DS Enabled	The port is enabled for all functions except remote power control.
DS & PWR Enabled	The port is enabled for all functions (system originated, remote power control and remote deadstart).

The port option switches and the baud rate selection switches are located on a panel in the back of the mainframe. Since the switches are key operated, they can be changed only by an authorized maintenance person.

Device-initiated functions are also safeguarded by passwords.

PROGRAMMING CONSIDERATIONS

Channel 15_8 communicates with the terminals connected to the external interface, one at a time. To establish 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 multiplexer (7XXX).
- The terminal with which the PP would like to establish communication (7XXX).

When a connect is established, the two-port multiplexer routes all data to the terminal designated by the select code. The multiplexer responds with the inactive signal to acknowledge receipt of the function code of 7XXX for select, 6XXX for deselect, and 0XXX for operation. Otherwise, the function is ignored by the multiplexer.

Output Data

The multiplexer accepts a maximum data block length of 64 characters per terminal. During the block data transfer, the multiplexer 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 multiplexer sends back an inactive signal instead of an empty signal after receiving the last output word. This signal indicates that the output buffer has accepted the last output word and cannot receive any more data from the PP. Output to a full buffer is not allowed by the multiplexer.

Input Data

The multiplexer stores 16 words of input data from the terminal. A lost data condition exists if the PP does not input the previous data before the fourth word 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 independent of 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 CHANNEL PROGRAMMING

MAINTENANCE CHANNEL

A PP in the IOU can perform any or all of the following operations through the maintenance channel (MCH) to each system element (CP, IOU, or CM).

- Initializing registers, controls, and memories.
- Monitoring and recording error information.
- Verifying error detection and correction hardware.

The maintenance channel consists of the maintenance channel interface on channel 17_8, a maintenance channel interface in each system element, and a set of interconnecting cables.

A selector on the IOU maintenance channel interface connects to one of up to three system elements. The IOU, CM, and CP are combined together as element 0, and its maintenance access control is internally connected to the selector. Any other system elements are assigned arbitrary element numbers. Each maintenance access control connects to the selector by a single cable. This arrangement results in a radial connection that allows any system element to be shut down or removed without affecting communication with the other elements.

NOTE

Bits numbering is 0 through 63 from left to right.

MCH FUNCTION WORDS

The MCH function word consists of the connect, opcode, and type fields which are used as described in the next three paragraphs and table I-5-11.

The connect field specifies the element to which the MCH is connected, controlling selection within the IOU only. The element remains connected until another connect code selects a different unit. Connect codes 10_8 to 17_8 leave the MCH unconnected; in this state the interface can be used for PP to PP communications.

The OPCODE field controls the unit selected by the connect code, preparing the unit for a coming read/write/echo operation, or causing the unit to halt, start, clear, or deadstart.

The type field selects the group of registers to be operated on (CP, IOU, CM, Control Store, Register File, MAP).

MCH CONTROL WORDS

Some function words must be followed by a 16-bit control word which specifies the internal address of the register to be connected. Refer to table I-5-12 for IOU internal address assignments, table I-5-13 for CM internal address assignments, and table I-5-14 for CP internal address assignments. The control word is issued as two 8-bit data words (sometimes called address bytes). This is accomplished by transmitting two PP words where the lower 8 bits in each word are used as follows:

- Function words to CP with opcodes 4/5 (read/write) and op code F (deadstart).
- Function words to CM and IOU with opcodes 4/5.
- Function words to CP, CM, and IOU with opcode 8 (echo).

MCH Programming for Halt/Start (Opcode 0/1)

These operations consist of the output of a function word. A halt opcode halts the processor without damaging the process being executed, including the integrity of the interunit communication of the halted processor such as CDC CYBER 170 exchange request communication, central memory communications, and the process state. If the process is subsequently restarted without performing any other MCH operations, or after performing read/write with certain precautions as described in Operating Systems Manual, the process continues without damage.

MCH Programming for Read/Write (Opcode 4/5)

Refer to Programming for PP Data Input/Output, this section, for a more complete procedure. In general terms, proceed as follows:

1. Issue function with opcode 4/5.
2. Output data word (leftmost half of control word).
3. Verify error flag clear.
4. Output data word (rightmost half of control word).
5. Verify error flag clear.
6. Input/output required number of data words.
7. Verify error flag clear.

Reading a nonexistent register returns all zeros. Writing to a read-only register, or to a nonexistent register, does not alter any register. Most registers are read/write as 64-bit (eight-byte) registers, requiring the input/output of eight MCH data words. Most registers which are physically smaller than eight bytes are right-justified with zero-fill. Exceptions are as follows:

- Reading a status summary register repeats the status information in each byte.
- IOU may disconnect MCH without affecting subsequent MCH operations.
  - After reading one to eight bytes from any register which does not require microcode assistance.
  - After writing one byte to a corrected error log register.
  - After writing one byte to an uncorrected error log register.

The following MCH operations on CP registers can be performed with the CP running or halted.

- Read CP status summary register.
- Read CP fault status register.
- Read CP corrected error log registers.
- Read CP options installed registers.
- Read CP element identifier register.
- Read/write CP dependent environmental control register.
- Read/write test mode control registers.
- Clear errors.

To read/write other CP registers, the CP must be running since these registers are accessed by microcode. Refer to table I-5-15 for IOU register bit assignments, table I-5-16 for CM register bit assignments, and table I-5-17 for CP register bit assignments.

TABLE I-5-11. MCH FUNCTION WORD BIT ASSIGNMENTS

Field			Description
			MCH Function Word to IOU
CONNECT (bits 8-11)	Code	0_{16} =	Connect IOU maintenance registers
OPCODE (bits 4-7)	Code	4_{16} =	Prepare for read (control word required)
		5_{16} =	Prepare for write (control word required)
		6_{16} =	Master clear
		7_{16} =	Clear fault status registers
		C_{16} =	Read IOU status summary (reads one byte, control word not required)
TYPE (bits 0-3)	Code	0 =	IOU select
			MCH Function Word to CM
CONNECT (bits 8-11)	Code	1_{16} =	Connect CM maintenance registers
OPCODE (bits 4-7)	Code	4_{16} =	Prepare for read (control word required)
		5_{16} =	Prepare for write (control word required)
		6_{16} =	Master clear
		7_{16} =	Clear error logs
TYPE (bits 0-3)	Code	E_{16} =	Select CM
			MCH Function Word to CP
CONNECT (bits 8-11)	Code	2_{16} =	Connect CP maintenance registers
OPCODE (bits 4-7)	Code	0_{16} =	Halt processor
		1_{16} =	Start processor
		4_{16} =	Prepare for read (control word required)
		5_{16} =	Prepare for write (control word required)
		6_{16} =	Master clear
		7_{16} =	Clear errors
TYPE (bits 0-3)	Code	F_{16} =	CP select

MCH Programming for Master Clear/Clear Errors (Opcode 6/7)

These operations consist of the output of a single function word. The master clear immediately and arbitrarily clears the connected unit, without regard to possible information loss. Clear errors clears the error indicators in the connected unit; to avoid loss of error information while the errors are cleared, the unit concerned should be halted.

MCH Programming for Read IOU Status Summary (Opcode C, IOU Only)

This operation is an alternative, faster means of reading the IOU status summary register.

1. Issue function with opcode c.
2. Input status summary byte.

TABLE I-5-12. IOU INTERNAL ADDRESS ASSIGNMENTS

Internal Address Hex	Internal Address Octal	Type (2)	Description
00	000	R	Status summary register
10	020	R	Element identifier register
12	022	R	Options installed register
18	030	R/W	Fault status mask register
40	100	R	Status register
80	200	R/W	Fault status 1 register
81	201	R/W	Fault status 2 register

NOTES

(1) The internal address is the second byte of a 16-bit control word which must be issued after a function word output with OPCODE = 4/5. The first byte is discarded.

(2) R = read, W = write

TABLE I-5-13. CM INTERNAL ADDRESS ASSIGNMENTS

Internal Address Hex	Internal Address Octal	Type (2)	Description
00	000	R	Status summary register
10	020	R	Element identifier register
12	022	R	Options installed register
A0	240	R/W	Corrected error log register
A4	244	R/W	Uncorrected error log 1 register
A8	250	R/W	Uncorrected error log 2 register

NOTES

(1) The internal address is the second byte of a 16-bit control word which must be issued after a function word with OPCODE = 4/5. The first byte is discarded.

(2) R = read, W = write

TABLE I-5-14. CP INTERNAL ADDRESS ASSIGNMENTS

Internal Address Hex	Internal Address Octal	Type (2)	Type (3)	Description
00	000	R	A	Status summary register
10	020	R	A	Element identifier register
30	060	R	A	Dependent environment control register
42	082	R	M	Monitor condition register
80	200	R/W	A	Processor fault status register
90	220	R/W	A	Retry corrected error log register
93	223	R/W	A	Map corrected error log register

NOTES

(1) The internal address is the second byte of a 16-bit control word which must be supplied after a function word output with OPCODE = 4/5. The first byte is discarded.

(2) R = read, W = write

(3) A = always accessible, M = microcode accessible

TABLE I-5-15. IOU REGISTER BIT ASSIGNMENTS (1 of 4)

Channel Bit	Description
	IOU Status Summary Register (MCH Address 00_{16})
59	Summary status
60	Processor halt
61	Valid uncorrected error
62	(not used)
63	Physical environment monitor

NOTE

When more than one byte is read, bits 59-60 are duplicated in each byte.

Channel Bit	Description
	IOU Element Identifier Register (MCH Address 10_{16})
32-39	Element type (02_{16} = IOU)
40-47	Element model (13_{16} = 830, 14_{16} = 810)
48-63	Element serial number (in packed decimal)

TABLE I-5-15. IOU REGISTER BIT ASSIGNMENTS (2 of 4)

Channel Bit	Description
	IOU Options Installed Register (MCH Address 12_{16})
20	PP25-PP31
21	PP20-PP24
22	PP5-PP11
23	PP0-PP4
24	Channel 7
25	Channel 6
26	Channel 5
27	Channel 4
28	Channel 3
29	Channel 2
30	Channel 1
31	Channel 0
32	Channel 17
34	Channel 15
36	Channel 13
37	Channel 12
38	Channel 11
39	Channel 10
40	Channel 27
41	Channel 26
42	Channel 25
43	Channel 24
44	Channel 23
45	Channel 22
46	Channel 21
47	Channel 20
52	Channel 33
53	Channel 32
54	Channel 31
55	Channel 30
60	Radial interface 2/3
61	Radial interface 1/2
62	Two-port mux
63	Display controller (CC545)
	IOU Fault Status Mask Register (MCH Address 18_{16})
03	Barrel 0, PP 4
04	Barrel 0, PP 3
05	Barrel 0, PP 2
06	Barrel 0, PP 1
07	Barrel 0, PP 0
11	Barrel 0, PP 11
12	Barrel 0, PP 10
13	Barrel 0, PP 7
14	Barrel 0, PP 6
15	Barrel 0, PP 5
19	Barrel 1, PP 4
20	Barrel 1, PP 3
21	Barrel 1, PP 2
22	Barrel 1, PP 1
23	Barrel 1, PP 0
27	Barrel 1, PP 11
28	Barrel 1, PP 10
29	Barrel 1, PP 7
30	Barrel 1, PP 6
31	Barrel 1, PP 5
32	Channel 7
33	Channel 6

TABLE I-5-15. IOU REGISTER BIT ASSIGNMENTS (3 of 4)

Channel Bit	Description
	IOU Fault Status Mask Register (MCH Address 18_{16}) (contd)
34	Channel 5
35	Channel 4
36	Channel 3
37	Channel 2
38	Channel 1
39	Channel 0
40	Channel 17
42	Channel 15
44	Channel 13
45	Channel 12
46	Channel 11
47	Channel 10
48	Channel 27
49	Channel 26
50	Channel 25
51	Channel 24
52	Channel 23
53	Channel 22
54	Channel 21
55	Channel 20
59	Radial interface 2/3
60	Channel 33
61	Channel 32
62	Channel 31
63	Channel 30
	IOU Status Register (MCH Address 40_{16})
38-55	Internal Register (A, P, Q, or K as per EC bits)
56	Long deadstart
59	Barrel reconfiguration switches
60-63	PP reconfiguration switches
	IOU Fault Status 1 Register (MCH Address 80_{16})
03	Barrel 0, PP 4 error
04	Barrel 0, PP 3 error
05	Barrel 0, PP 2 error
06	Barrel 0, PP 1 error
07	Barrel 0, PP 0 error
11	Barrel 0, PP 11 error
12	Barrel 0, PP 10 error
13	Barrel 0, PP 7 error
14	Barrel 0, PP 6 error
15	Barrel 0, PP 5 error
19	Barrel 1, PP 4 error
20	Barrel 1, PP 3 error
21	Barrel 1, PP 2 error
22	Barrel 1, PP 1 error
23	Barrel 1, PP 0 error
27	Barrel 1, PP 11 error
28	Barrel 1, PP 10 error
29	Barrel 1, PP 7 error
30	Barrel 1, PP 6 error
31	Barrel 1, PP 5 error
32	CM address formation error
33	IOU internal error
34	Firmware error
35	PP memory data error 7XF0
36	Firmware or channel data error
37	Conversion error
39	PP memory data-in error

TABLE I-5-15. IOU REGISTER BIT ASSIGNMENTS (4 of 4)

Channel Bit	Description
	IOU Fault Status 1 Register (MCH Address 80_{16}) (Contd)
45	OS bounds violation
46	OS boundary address PE
49	Uncorrected CM read error
50	Uncorrected CM write error
51	CM reject
52	Input CM tag error
53	CM response code error
	IOU Fault Status 2 Register (MCH Address 81_{16})
32	Channel 7 error
33	Channel 6 error
34	Channel 5 error
35	Channel 4 error
36	Channel 3 error
37	Channel 2 error
38	Channel 1 error
39	Channel 0 error
40	Channel 17 error
42	Channel 15 error
44	Channel 13 error
45	Channel 12 error
46	Channel 11 error
47	Channel 10 error
48	Channel 27 error
49	Channel 26 error
50	Channel 25 error
51	Channel 24 error
52	Channel 23 error
53	Channel 22 error
54	Channel 21 error
55	Channel 20 error
59	Radial interface 2/3 error
60	Channel 33 error
61	Channel 32 error
62	Channel 31 error
63	Channel 30 error

TABLE I-5-16. CM REGISTER BIT ASSIGNMENTS (1 of 2)

Channel Bit	Description
	CM Status Summary Register (MCH Address 00_{16})
56-57	Oscillator code (00 = normal, 01 = -2%, 10 = +2%)
61	Valid uncorrectable error
62	Corrected error
63	Long warning (low/high temperature, blower 1/blower 2 fault)
NOTE The byte is duplicated in each byte.	
	CM Element Identifier Register (MCH Address 10_{16})
32-39	Element type (01_{16}=CM)
40-47	Element model (13_{16} = 830, 14_{16} = 810)
48-63	Element serial number
	CM Options Installed Register (MCH Address 12_{16}) (Refer to table I-3-2)
01 = 1, 12 = 0	262K memory installed, 64K chips
03 = 1, 12 = 0	524K memory installed, 64K chips
07 = 1, 12 = 0	1049K memory installed, 64K chips
09 = 1, 12 = 0	1573K memory installed, 64K chips
11 = 1, 12 = 0	2098K memory installed, 64K chips
06 = 1, 12 = 1	4196K memory installed, 252K chips
07 = 1, 12 = 1	2098K memory installed, 252K chips
08 = 1, 12 = 1	1049K memory installed, 252K chips
16 = 1	A memory configuration switch is in the up position
17 = 1, 12 = 1	SW1 is not in the centre position
18 = 1, 12 = 1	SW2 is not in the centre position
19 = 1, 12 = 1	SW3 is not in the centre position
19 = 1, 12 = 0	SW3 is not in the centre position
20 = 1, 12 = 0	SW4 is not in the centre position
21 = 1, 12 = 0	SW5 is not in the centre position
	CM Corrected Error Log Register (MCH Address $A0_{16}$)
00	This log contains valid information
01	Unlogged single-bit error
05-06	Port code 01 Port to IOU 10 Port to CP 11 Refresh or second pass
08-31	Address
32-39	Syndrome bits
	CM Uncorrected Error Log 1 Register (MCH Address $A4_{16}$)
00	This log contains valid information
01	Unlogged uncorrected error
02	Illegal function
03	Bounds violation
04	Second pass
5-6	Port number code 01 Port to IOU 10 Port to CP 11 Refresh or second pass
7	Refresh

TABLE I-5-16. CM REGISTER BIT ASSIGNMENTS (2 of 2)

Channel Bit	Description
	CM Uncorrected Error Log 1 Register (MCH Address $A4_{16}$)
8-31	Word address
29	Address parity P5
30	Address parity P6
31	Address parity P7
32-35	Write data parity error
43	Tag in error
44	Function error
45	Mark error
46	Address error 4
47	Address error 5
48	Address error 6
49	Address error 7
50-53	Function bits associated with error word
54	Function parity
55	Mark parity
56-63	Mark bits associated with error word
	CM Uncorrected Error Log 2 Register (MCH Address $A8_{16}$)
00	This log contains valid information
01	Unlogged uncorrected error-2
02	Data read parity error
03	SECDED double bit error
04	Tag out parity error
5-6	Port number code as for A0, A4
8-31	Word address
32	Byte 0 read parity error (1)
33	Byte 1 read parity error (1)
34	Byte 2 read parity error (1)
35	Byte 3 read parity error (1)
36	Byte 4 read parity error (1)
37	Byte 5 read parity error (1)
38	Byte 6 read parity error (1)
39	Byte 7 read parity error (1)

NOTE

If bits 2 and 3 are set and SECDED is disabled, these bits point to the bytes with errors for a read data parity error.

TABLE I-5-17. CP REGISTER BIT ASSIGNMENTS (1 of 2)

Channel Bit	Description
	CP Status Summary Register (MCH Address 00_{16})
58	Excutive Monitor mode
59	Short warning, imminent power failure
60	CP halted
61	Uncorrected error
62	Corrected error
63	Long warning
	CP Elememt Identifier Register (MCH Address 10_{16})
32-39	Element type (00_{16} = CP)
40-47	Element model (13_{16} = 830, 14_{16} = 810)
48-63	Element serial number
	CP Dependent Environment Control Register (MCH Address 30_{16})
05	Micro step
06	Enable processor fault status trap
24	Processor fault status enabled
25	MAP real memory address mode enabled
26	MAP file 0 enabled
27	MAP file 1 enabled
28	MAP file 2 enabled
29	MAP file 3 enabled
30	Instruction retry enabled
31	Instruction step enabled
32	Maintenance scan halt enabled
33	Test mode enabled
35	Disable corrected error to CP status summary
36	Control store breakpoint enabled
37	Control store sweep enabled
	CP Processor Fault Status Register (MCH Address 80_{16}) CP Retry Corrected Error Log Register (MCH Address 90_{16})
32	ARVI parity error bits 0-7, 32-39
33	ARVI parity error bits 8-15, 40-47
34	ARVI parity error bits 16-23, 48-55
35	ARVI parity error bits 24-31, 56-63
36	Uncorrected memory write error
37	Memory reject
38	Memory tag parity error
39	Response code parity error
40	FP exception trap index ROM parity error
41	AD or BD bits 0-15 parity error
42	LD box ROM parity error
43	ADS or BDS or XBD sel ROM parity error
44	Shift type ROM parity error or shifter input
45	Uncorrected memory read error
46	AD or BD bits 16-31 parity error
47	AD-MAC parity error, MAC write parity error
48	Memory response timeout
49	CYBER ROM parity error
50	Instruction parity error
51	XBD ROM parity error
52	AD or BD bits 32-47 parity error

NOTE

Registers 80_{16} and 90_{16} indicate errors in CP hardware.

TABLE I-5-17. CP REGISTER BIT ASSIGNMENTS (2 of 2)

Channel Bit	Description
	CP Processor Fault Status Registers (MCH Address 80, 81_{16}) CP Retry Corrected Error Log Register (MCH Address 90_{16}) (Contd)
53	BDP adder, data ROM, parity error
54	Immediate ROM
55	AD or BD bits 48-63 parity error
56	MAP parity error
57	MAP parity error
58	MAP parity error
59	MAP parity error
60	MAP multiple hit fault
61	(Not used)
62	Maintenance access control error
63	Any control store parity error
	CP MAP Corrected Error Log (MCH Address 93_{16})
56	File 0 parity error
57	File 0 parity error
58	File 1 parity error
59	File 1 parity error
60	File 2 parity error
61	File 2 parity error
62	File 3 parity error
63	File 3 parity error

PART II

HARDWARE

INTRODUCTION

The computer consist of a central processor (CP), central memory (CM), and an input/output unit (IOU). Each of these elements is described below as compared to a model 730. Unless specifically said to be different, models 810 and 830 operate like a model 730.

CENTRAL PROCESSOR

The CP is microprogrammable. The microcode is contained in a special memory in the CP called control store. Microcode is loaded during initialization (refer to section 3 of this part).

The following CP instructions are new or perform differently.

- Block copy instructions (011 and 012) and direct read/write instructions (014 and 015) for extended memory.
- Direct read/write instructions (660 and 670) for CM.

CEJ/MEJ is always enabled; it cannot be disabled. Refer to section 4 in this part for further information on CP instructions.

Instruction execution times differ from those for the model 730. Execution times are given in section 4 of part I.

The operating registers (X0 through X7, A0 through A7, and B0 through B7) and support registers (P, EM, RAE, FLE, and MA) are the same in models 810, 830, and 730. The RAC and FLC registers have increased to 21 bits to support larger memory sizes. A new 6-bit register, the flag register, has been added; it contains new control bits which are explained in greater detail under CYBER 170 Exchange Package in section 5 of this part.

In addition to the instruction control and arithmetic/logic sections, the CP contains a high-speed buffer, called map, used in memory addressing. This buffer minimizes the time used to access central memory. It does not affect programming.

The map is part of the memory addressing section. This section uses tables in central memory during its operation. To save time, the addresses used most recently are saved in the CP a in high-speed buffer called the MAP. If the desired address is among those used most recently, no access to the central memory tables is needed.

The MAP can also be degraded during initialization using the hardware reconfiguration (*H*) display. (For further information on displays, refer to the CIP User Guide.) The MAP is divided into four units, which can be turned on or off. The *H* display shows which of units 0 through 3 of the MAP are logically on.

CENTRAL MEMORY

In the 170 state, central memory is available in sizes from 262 to 2096K words organized in two banks (262K memories) or four banks (other memories). To accommodate the larger addresses, the RAC and FLC registers have increased to 21 bits. There is also a PP relocation register (R) which is used in conjunction with the A register to form absolute CM addresses for CM read/ write instructions. Refer to Addressing by PPs in section 4 of this part. A CM word consists of 64 bits of which only the rightmost 60 bits are used.

CM contains a register, the CM bounds register, which limits the write access to CM from specified ports. The ports are limited to the area between an upper and lower bound as specified in the CM bounds register. This restriction prevents PPs from writing into the area of CM permanently occupied by the Virtual State software. The CM bounds register is set through the maintenance channel described later in this section. Memory addressing (address out of range, and so on) is described in section 4 of this part.

Extended memory is actually a part of central memory and is called unified extended memory (UEM). In general, the instructions which deal with ECS/ESM on the model 730 (and LCME on the model 176) are the same as those used for unified extended memory on models 810 and 830. Specifically, these are the block copy instructions (011 and 012) and direct read/write instructions (014 and 015). For further information, refer to section 4 of this part.

INPUT/OUTPUT UNIT

The IOU consists of 10 or 20 peripheral processors (PPs) with 10 PPs per barrel. Each PP has 4K words of memory. A PP memory word consists of 16 bits of which only the rightmost 12 bits are used; the leftmost 4 bits must be set to zero. Either 12 or 24 I/O channels can be used. The PPs work at a speed which is the same as the CYBER 170 PPs 2xPP speed.

The IOU contains a deadstart microcomputer which creates displays for initialization and maintenance, in addition to displays for PP registers and error indicators (refer to sections 2 and 3 of this part). Channel status and assignments are also given in section 3 of this part.

A two-port multiplexer is permanently attached to channel 15_8. This provides for communication between a PP and two terminals. One port is reserved for maintenance purposes; the other is reserved for future use.

Each PP in the IOU has a 28-bit R (relocation) register. It is used with the A register to form absolute CM addresses for CM read/write instructions. It is loaded and stored with the 24 and 25 PP instructions. Refer to Addressing by PPs in section 4 of this part.

PP instruction 27 (RPN), which reads the P register on the model 730, is now a pass instruction. The P register can be read, however, through the maintenance channel (see Maintenance Registers and Maintenance Channel later in this section). Other differences in PP instructions (24, 25, 641, 651, 661, and 671) are described in section 4 of this part. PP instruction execution times are given in section 4 of part I.

EXTENDED MEMORY

Extended memory for models 810 and 830 is called unified extended memory (UEM) and it exists as a part of central memory. (See preceding discussion of central memory.) Neither ECS nor ECM is available.

MAINTENANCE REGISTERS AND MAINTENANCE CHANNEL

Maintenance registers are present in the CP, CM, and IOU. The maintenance registers, like model 730 status/control registers, reflect system status and errors, and additionally can be used to initialize certain conditions (like the CM bounds described previously). Maintenance registers are on channel 17_8, unlike model 730 status/control registers which are on channel 16_8. Channel 17_8 is called the maintenance channel (MCH).

The maintenance channel has a software rather than hardware interlock using the channel flag. Before maintenance channel I/O is permitted, the channel must be reserved using the test and set channel flag instruction (641cm); it is released by the clear flag instruction (651cm). Refer to Set and Clear Channel Flag Instructions in section 4 of this part.

One PP can be designated to access the maintenance registers in the CP, CM, and IOU through the maintenance channel for the purpose of reading and/or initializing the registers. Section 5 of part I describes maintenance channel programming and several of the maintenance registers.

SOFTWARE

Although the computer hardware differs from the model 730, it operates much the same using microcode and special software. The microcode executes the model 730 instruction set. The software, referred to as Virtual State, simulates certain model 730 error exits. Both the microcode and Virtual State software must be loaded during initialization by the Common Test and Initialization (CTI) software routines. These three items (microcode, Virtual State software, and CTI) are integral parts of the systems and are required to make them function as a model 730 would.

The actions taken by CTI to set up the proper environment are discussed in more detail in section 3 of this part; Virtual State is discussed in sections 3, 4, and 5 of this part.

INTRODUCTION

This section discusses the controls on the central memory that differ from the model 730. It also discusses the deadstart display which differs from the model 730 deadstart panel. New controls are discussed first; controls which appear on a model 730 but do not have corresponding controls on models 810 and 830 are discussed last.

Other controls used by maintenance personnel are described in the hardware manual for the display console listed in the System Publication Index. Power-on and power-off procedures are described in the Hardware Operator's Guide, also listed in the publication index.

DEADSTART DISPLAY

The deadstart display is shown in figure II-2-1. It is a display on the screen of the display station, created by an independent microcomputer in the mainframe. It does not rely on any program being operational in the PPs. The deadstart display is used to select a 16-word deadstart program for PP0 and to initiate the deadstart sequence for PP0.

It is also used to reconfigure PPMs and barrels, and to display error status and maintenance information. In the model 730, the same functions are performed by switches on the deadstart panel. For example on the model 730, pressing the deadstart button loads the contents of the switch panel into logical PP0 and initiates a deadstart of PP0. On models 810 and 830, pressing the deadstart button activates the deadstart microcomputer which brings up the initial deadstart display to prepare for actual deadstart; The further step of deadstarting logical PP0 is accomplished by a command (S or L) to the microcomputer.

The initial deadstart display contains a 16-word program in which each word comprises six octal digits. Only the rightmost four digits may be used for the deadstart load or bootstrap program; the leftmost two digits are reserved for maintenance use and must be set to zero. Other information on the deadstart display is:

- A list of some basic commands (XX YYYYYY, XX+YYYYYY, L, S, and H).
- Reconfiguration status for PPMs and barrels.

  The line PPM CONF=XX indicates the current reconfiguration status of the PP memories. The number XX is an octal number in the range 0-11, and is the number of the physical PP memory which is being used as logical PP0.

- Similarly, the line BRL CONF=X, where X may be 0 or 1, indicates which physical barrel is being used as logical barrel 0.

  If no errors occur, the deadstart program is read into PP0. (More detailed descriptions of the short and long deadstart sequences are given in section 3 of part I.)

- Reconfiguration status for CM

  The line CM RECONF SW indicates the latest reconfiguration commands entered through the keyboard. Each switch can be set to C (center), U (up), or D (down), corresponding to mechanical switch positions.

- Maintenance information

  The lines DLY LOOP and CLF FREQ indicate the latest commands entered during maintenance procedures. These lines should normally be as shown in figure II-2-1.

- A keyboard entry line: the bottom line of the display echos back the entries made by the operator on the keyboard of the console.
- An error line: the line immediately above the keyboard entry line indicates that the keyboard entry has an error in it.
- LDS status: if a long deadstart sequence aborts because of a hardware malfunction, the type of error is displayed in the lower right-hand side of the screen. These errors are:

  LDS BRANCH TEST FAILURE
  LDS DATA/ADDRESS ERROR
  STATUS SUMMARY ERRORS

- PP Registers: the P, A, Q, and K registers for PPs 0 through 11 may be displayed on the right-hand side of the screen. This display is brought up automatically during long deadstart; at other times it may be brought up manually by the maintenance command PR.

Other displays and commands are available for maintenance, or for software debugging. For further information on these displays, or on editing deadstart programs, refer to the Hardware Operator's Guide.

The long/short deadstart selection determines whether IOU diagnostic testing is done or omitted. When the long deadstart program is selected (command L), PP0 runs a diagnostic program from a read-only memory (ROM) in the IOU to test the first five PPs. The detection of a hardware error during this test

suspends the deadstart sequence with the LDS error status display described previously in this section. If no errors occur, the deadstart program is read into PP0. (More detailed descriptions of the short and long deadstart sequences are given in section 3 of part I.)

The initialization routines provide for additional testing of the first five PPs if a particular bit of the deadstart program on the panel is set, performing a deadstart called an extended deadstart (EDS). (For further information, refer to the operating system operator's guide or the Hardware Operator's Guide.)

Neither the SWEEP/LOAD/DUMP switch nor the CEJ/MEJ switch on a model 730 deadstart panel are present on models 810 and 830. The central exchange jump instruction for the CP and monitor exchange jump instructions for the PPs are permanently enabled. There is no facility for disabling them.

MEMORY RECONFIGURATION

Central memory has three configuration switch registers, accessible through the deadstart display. If a portion of CM malfunctions, that portion can be disabled using the configuration switch registers. Remaining CM is reconfigured so that it is accessible by contiguous addresses starting at address 0 and going up to the maximum remaining address. Refer to section 3 of part I for further information on reconfiguring memory.

Addressing of reconfigured central memory is discussed in section 4 of this part.

DEADSTART

```
XXX+  YYYYYY=CHANGE DS PRG          PPM CONF = 00
XXX+  YYYYYY=CHANGE DS PRG INC      BRL CONF = 0
      S=SHORT=SHORT DS              DLY LOOP = 0
      L=LONG DS                     LDS ADDR = 6000
      H=HELP                        CLF FREQ = NORMAL
                                CM RECONF SW3=C SW4=C
                                          SW5=C


      PROGRAM 0

01    001402
02    007303
03    000013
04    007503
05    007703
06    000300
07    007403
10    007103
11    007301
12    000010
13    000000
14    007112
15    000000
16    000000
17    000000
20    000000
```

Figure II-2-1. Initial Deadstart Display.

OVERVIEW

System initialization puts the system hardware in a known state where it is ready for the loading of an operating system or off-line maintenance tests. Initialization is an automatic process initiated by pressing the deadstart switch or button and entering S (for short deadstart) or L (for long deadstart) on the keyboard.

The procedures for deadstart described in section 3 of part I initialize the IOU. The last step of the deadstart is to read the deadstart program from the microcomputer into PP0. This program reads or provides for the eventual reading of the first record off an external device. This record is the beginning of the Common Test and Initialization (CTI) software routines. It is up to CTI to initialize the remainder of the system elements and various maintenance registers, and set up the operating environment such that the computer will operate like a model 730. (If maintenance is chosen on the first CTI display, very little initialization is done to preserve the contents of the elements for testing.)

CTI OPERATIONS

One of the first actions CTI takes is to test channel 17_8 (the maintenance channel); if the channel is active, CTI recognizes the machine as a model 8XX. At this point, CTI takes some different actions than it would for models of the CYBER 170 series. Where before it had to run a series of programs to determine the proper machine attributes, it has now only to access the maintenance registers of all the connected elements to find out the size of memory, the number of PPs, and so on. These attributes are then noted as before in the hardware descriptor table.

On CYBER 170 series machines, CTI would test memory after setting up the hardware descriptor table, but on models 810 and 830 microcode may have to be loaded to the CP control store and the Virtual State software must be loaded to memory first. CTI loads the microcode and Virtual State software from the Hardware Initialization and Verification Software (HIVS).

CTI reads the actual CM size (or the adjusted size if memory was reconfigured or degraded by the operator earlier in the process) from the hardware descriptor table and loads the Virtual State software at the highest part of memory (refer to figure II-3-1).

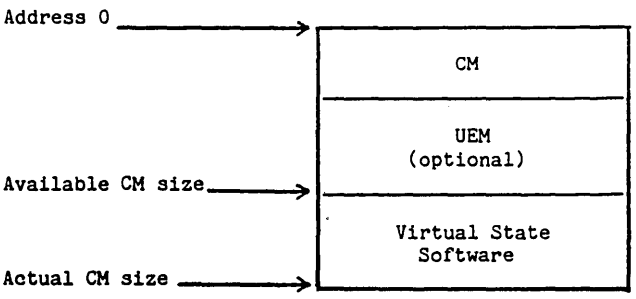


Figure II-3-1. Central Memory Format

CTI then adjusts the size of memory available taking into consideration the space used by the Virtual State software. This new size is the CM available for the operating system or maintenance system and its jobs/tests. The section of memory containing Virtual State software is protected from the operating/maintenance system and its jobs by the CM bounds register discussed in section 1 of this part.

CTI also initializes maintenance registers in the CP, CM, and IOU. After loading microcode and Virtual State software and initializing of applicable registers, the environment necessary to support the computer is complete. At this point, the CP is not running. If the deadstart is a recovery deadstart, control passes immediately to the operating or maintenance system with no further CTI initialization. If it is not a recovery deadstart, the CP is started. The remainder of CTI processing is the same as it is for models of the CYBER 170 computer systems. CTI initializes memory and completes any other settings necessary before handing off the hardware to an operating or maintenance system. When handing off to an operating system, the CP is stopped.

CTI HANDOFF STATE

Before an operating system or maintenance system can be loaded and executed, the elements in the hardware system must be initialized to a known state. This is done by CTI before it passes control to the waiting system. The remainder of this section describes the initial state of the system elements, channels, and registers. Any hardware component not defined is assumed to be in an undefined state.

CENTRAL PROCESSOR

The CP is master cleared on all deadstart levels.

CENTRAL MEMORY

All available memory (that is, reconfigured or degraded memory less the space used by the Virtual State software) is initialized with the exception of the area which contains the initialization code. The rightmost 60 bits of each 64-bit CM word are set to all ones; the leftmost 4 bits are set to zeros. The Virtual State software resides in the upper end of memory as described under CTI Operations.

PP MEMORY

All PPs that are logically on are initialized with the exception of the area that contains the initialization code. The rightmost 12 bits of each 16-bit PP word are set to all ones; the leftmost 4 bits are set to zeros.

The R register of each PP that is logically on is initialized to 4000_8.

PPs AND I/O CHANNELS

All PPs that are logically on, with the exception of PP0, are in the deadstart state. PP0 is executing the operating system or maintenance system bootstrap program. If there was a PP on the deadstart channel, it is executing an IAM instruction (input A words to m from channel d) on channel 12_8.

All PPs that are logically off are idled via the maintenance register(s).

All I/O channels associated with PPs that are logically on, with the exception of channel 0, are active and empty. Channel 0 is inactive. All I/O channels not associated with PPs that are logically on are inactive, except as noted above for the deadstart channel. The deadstart device is logically connected on the deadstart channel.

The status of all non-I/O channels is as follows:

Channel (in octal)	Status
0	Inactive and empty
10 (display controller)	Active and empty
14 (real-time clock)	Active and full
15 (two-port multiplexer)	Active and empty
16	Inactive
17 (maintenance)	Active and empty

At least 10 PPs, two of which are PP0 and PP10, are on. All channel flags and channel error flags are cleared.

MAINTENANCE REGISTERS

The contents of several maintenance registers determine the environment in which the computer operates. At the time of handoff to an operating system or maintenance system, the conditions listed below are in effect. (The register number in hexadecimal is given in parentheses following the register name.)

CP Environment

- Status summary register (00)* shows the processor halted and in Virtual State, and no errors set.
- Element identifier register (10)* shows the element type, element model, and element serial number.
- Options installed register (12)* shows CP options that are installed.
- Dependent environment control register (30) is set so that the CP is not in test mode; corrected error reporting is enabled; instruction retry is enabled; enabling of MAP by quarters is set as specified; clock margins are set to normal.
- Processor fault status register (80) shows no errors.
- Map corrected error log register (93) shows no errors.

CM Environment

- Status summary register (00)* shows no errors;
- Element identifier register (10)* shows the element type, element model, and element serial number.
- Options installed register (12)* shows the memory installed and reconfiguration, if any, at deadstart.
- Environment control register (20) is set so that parity checking is enabled; SECDED is enabled; noninterleaved mode is disabled; check/syndrome bits are normal; corrected

*This register cannot be written; it is for status only.

error logging is enabled; suppression of corrected error reporting via ports is enabled; timing margins are not enabled; the port disable bit vector is set to 0.

- Bounds register (21) is set so that IOU port writes are inhibited above the first word address of the Virtual State software in CM.
- Corrected error log register (A0) shows no errors.
- Uncorrected error log registers 1 and 2 (A4 and A8) show no errors.

IOU Environment

- Status summary register (00)* shows no errors; however, status summary is set.
- Element identifier register (10)* shows the element type, element model, and element serial number.
- Options installed register (12)* shows the barrels, channels, and other options that are installed.
- Status register (40) is set so that fast and slow timing margins are not enabled; barrel and PP reconfigurations are set as required.
- Fault status registers 1 and 2 (80 and 81) show no errors.

In addition to the handoff state set by CTI, certain conditions are set by the Virtual State software. These settings indicate that the CP does not interrupt processing on short warning or correctable hardware errors; it is interrupted, but not necessarily stopped, on all uncorrected hardware errors. Refer to section 5 of this part for further information on error handling.

*This register cannot be written; it is for status only.

CP OPERATING MODES

Three modes of operation are mentioned in part I: CYBER 170 job, CYBER 170 monitor, and Virtual State. CYBER 170 job mode and CYBER 170 monitor mode function similarly to the program and monitor modes of the model 730. The CYBER 170 monitor flag still exists and indicates whether the program currently running is in CYBER 170 monitor mode (flag is set) or CYBER 170 job mode (flag is clear).

Virtual State oversees the CYBER 170 job and monitor modes and ensures that they duplicate the actions of a model 730. This includes the simulation of CMU instructions and processing of certain error conditions (refer to section 5 of this part). Virtual State is controlled by hardware, microcode, and software.

CP INSTRUCTIONS

The CP instruction format is the same on models 810, 830 and 730. The same restrictions on instructions with word boundaries apply. The following instructions are new or differ from the way they performed on the model 730.

- 011, 012 (block copy of UEM)
- 014, 015 (direct read/write UEM)
- 660, 670 (direct read/write CM)

UNIFIED EXTENDED MEMORY INSTRUCTIONS (011, 012, 014, 015)

The UEM enable flag (bit 56 in the flag register) must be set in the CYBER 170 exchange package for these instructions to execute legally.

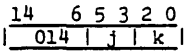
Block Copy Instructions (011, 012)

These instructions differ from the block copy instructions on a model 730 primarily in that the copy is to or from UEM rather than ECS. On models 810 and 830 an exchange breakin is allowed after a read or write of data blocks containing not more than 64 words each. Flag register operations (caused when bit 23 of XO and bit 23 of FLE are both set) are illegal. If XO plus RAE bit 21 or 22 is set, the next instruction is taken from parcel 2 of the same instruction word. If this is not the case, the next instruction is taken from parcel 0 of the next instruction word. A negative Bj plus K result causes an address range error exit rather than a pass instruction.

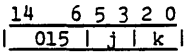
Direct Read/Write UEM Instructions (014, 015)

On a model 730, the 014 and 015 instructions are illegal. On models 810 and 830 these instructions transfer single words between UEM and a specified X register. (For those familiar with the CYBER 170 model 176, these instructions are similar to that machine's 014 and 015 instructions except the transfer is to or from UEM rather than LCME.)

The format of the 014 instruction [read one word from (Xk plus RAE) into Xj] is as follows:



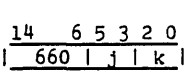
The format of the 015 instruction [write one word from Xj to (Xk plus RAE)] is as follows:



DIRECT READ/WRITE CM INSTRUCTIONS (660, 670)

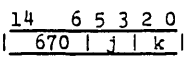
The 660 and 670 instructions are pass instructions on a model 730. On a model 810 or 830, they read and write central memory. When i is not zero, the 66ijk and 67ijk instructions perform the same as on a model 730.

660jk Read Central Memory at (Xk) to Xj



This instruction reads the word at location (Xk) in central memory and places it in Xj. Xk is a right-justified 21-bit relative address. Bits 59 through 21 of Xk are ignored.

670jk Write Xj into Central Memory at (Xk)



This instruction writes Xj into location (Xk) in central memory. Xk is a 21-bit relative address. Bits 59 through 21 of Xk are ignored.

INSTRUCTION LOOKAHEAD

The computers have instruction lookahead hardware that prefetches up to two words for the regular program sequence, into two registers (a two-word first-in, first-out buffer). [The model 730 instruction lookahead hardware prefetches the next instruction (at address P plus 1) after the current instruction.]

The computers allow the purging of instruction lookahead registers to be controlled. Under normal conditions, the lookahead registers are purged by execution of a return jump instruction (operation code 010), UEM read instruction (011), exchange jump instruction (013), or unconditional branch instruction (02).

When extended purge control is selected, the registers are also purged by execution of any conditional jump instruction (operation codes 03 through 07) or any CM store instruction (operation codes 50 through 57 when i equals 6 or 7). To enable extended purge control, the system sets bit 52 of the flag register in the CYBER 170 exchange package. It may be helpful to set extended purge control when self-modifying code is present; however, the additional purging does cause a degradation in execution and does not cover all cases of code modification. Modification of an instruction on the word currently in execution (or any word in the processor's stack) can result in execution of the modified instruction instead of the original instruction.

If normal purge conditions are in effect, a store instruction that modifies a sequential instruction must modify at least P plus 4 words ahead to ensure execution of the modified code.

If the extended purge option is selected, a store instruction can modify the next sequential instruction and be assured of executing the modified instruction. A store instruction followed by a branch to a modified instruction always executes the modified code.

PP INSTRUCTIONS

In general, the PP instruction format is the same on models 810, 830, and 730 although they are shown as 16-bit instructions in this manual. The leftmost 4 bits must be set to zero for the instructions to operate as defined in this manual. If unexpected results occur and any of the leftmost 4 bits are set, contact a Control Data support analyst or customer engineer.

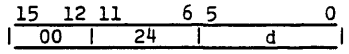
The following instructions are new or differ from the model 730.

- 24, 25 (load/store R register)
- 27 (pass)
- 641, 651 (set/clear channel flag)
- 661, 671 (set/clear channel error flag)

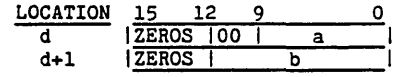
LOAD/STORE R REGISTER INSTRUCTIONS (24, 25)

On a model 730, the 24d and 25d instructions are pass instructions. On models 810 and 830 these are pass instructions only if d is zero; if d is nonzero, the 24d and 25d instructions load and store the relocation (R) register.

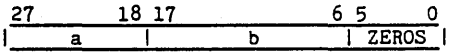
24d Load R Register



If d is not 0, this instruction loads the R register from bits 9 through 0 of PP memory location d (this becomes the leftmost 10 bits of R) and from bits 11 through 0 of location d plus 1 (this becomes the next 12 bits of R). Refer to figure II-4-1. If d is 0, it is a pass like the model 730 2400 instruction.



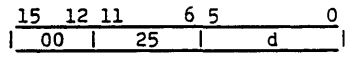
RELOCATION REGISTER IN PP MEMORY



RELOCATION REGISTER IN PP HARDWARE

Figure II-4-1. R Register Formation

25d Store R Register

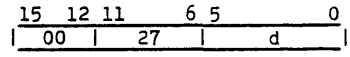


If d is not 0, this instruction stores the R register in PP locations d (this contains the leftmost 10 bits of R) and d + 1 this contains the next 12 bits of R) If d is 0, it is a pass like the model 730 2500 instruction.

PASS INSTRUCTION (27)

On a model 730, the 27d instruction reads the program address. On models 810 and 830 the program address is read through the maintenance channel and 27d instruction is a pass instruction.

27d Pass Instruction



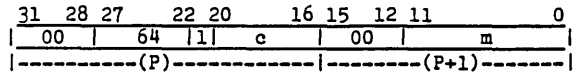
This instruction allows sensing of its execution by generating a pulse to a testpoint where it can be monitored by external equipment; otherwise, it does no other operations.

CHANNEL FLAG INSTRUCTIONS (641, 651, 661, 671)

On a model 730, instructions 64dm, 65dm, 66dm, and 67dm jump to a new address depending on the status of channel c. These instructions are still available on models 810 and 830 using the instructions 640cm, 650cm, 660cm, and 670cm. New model 810 and 830 instructions, 641cm, 651cm, 661cm, and 671cm, set and clear the channel c flag, clear the channel c error flag, and jump to address m depending on the flag setting.

Set and Clear Channel Flag (641, 651)

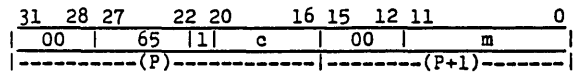
641cm Test and Set Channel c Flag



If the channel c flag is set, this instruction causes a jump to m. If the channel c flag is clear, it sets the flag and continues with the next instruction. If m is P plus 2, it unconditionally sets the channel c flag.

If more than one PP simultaneously issues this instruction on channel 17 the conflict is resolved as follows. The PP in the lowest physical level sees the flag as it actually is; all other PPs in conflict see the flag set and, therefore, jump to m. For other channels, the conflict is not resolved but can be avoided. Any ten consecutively numbered PPs issue instructions at different times. For code compatibility within the 800 series, conflict on accessing the channel flag does not exist when any five consecutively numbered PPs access any channel flag.

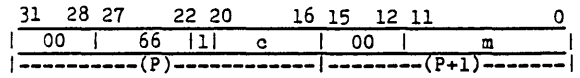
651cm Clear Channel c Flag



This instruction clears the channel c flag. The m field is required but not used.

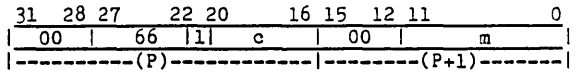
Clear Channel Error Flag and Jump (661, 671)

661cm Test and Clear Channel c Error Flag Set



If the channel c error flag is set, this instruction clears the error flag and causes a jump to m. If the error flag is clear, the instruction is a pass. If m is P plus 2, it unconditionally clears the channel error flag.

671cm Test and Clear c Error Flag Clear



If the channel c error flag is clear, this instruction causes a jump to m. If this error flag is set, the instruction clears the error flag and proceeds with the next instruction.

PP COMMUNICATIONS

On models 810 and 830, data transfers between PPs are performed using 16 bits rather than 12 as is true on a model 730. When transferring 12-bit instructions, it is possible that one or more of the leftmost 4 bits of the 16-bit word being transferred could be set; this could cause unexpected results to occur.

It is also possible to generate incorrect parity when transferring data between PPs on a CYBER 170 channel. The parity bit sent reflects the parity of the 16-bit word in PP memory while the peripheral controller verifies parity of the 12 bits received. Programmer must ensure that the leftmost 6 bits of the 18-bit A register are zero when outputting A to any channel.

CENTRAL MEMORY

MEMORY ADDRESSING

The largest CM address possible is 21 bits (7 777 777) as opposed to 18 bits in the model 730. If an address is larger than the maximum memory size (7 777 777), it is treated as an undefined operation if allowed by FLC; otherwise, it is treated as an address out of range. If an address is larger than the amount of memory installed in the system but less than 7 777 777, a nonexistent address is referenced; for a write operation, all data is lost and for a read operation, all ones are read. (This situation on a model 730 would cause the address to wrap around installed memory.)

If memory has been reconfigured and the address is in a part of memory no longer available, the address is wrapped around installed memory. (This is the same action taken on a model 730.)

ADDRESSING BY PPs

The PP R (relocation) register is a 28-bit register used in conjunction with the A register to form absolute CM addresses for CM read/write instructions. If bit 17 of the A register is clear, bits 16 through 0 of that register specify an absolute CM address in the range 0 through 377 777. If bit 17 of A is set, bits 16 through 0 are added to the R register (aligned as shown in figure II-4-2) to form an absolute CM address in the range 0 through 0 0007 777 777. The rightmost 6 bits of the R register are always zeros. The leftmost 7 bits of the R register represent extra addressing capacity that is unused.

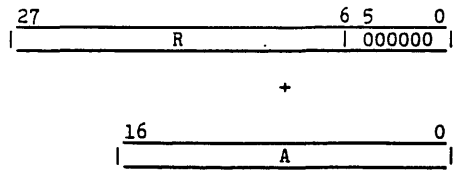


Figure II-4-2. Absolute Address Formation

PP instruction 24 loads the R register and instruction 25 stores the R register. Refer to PP Instructions in this section.

MEMORY MAP

The following discussion assumes that a site is using UEM. If not, the memory map is the same as for a model 730.

A user program can use two areas of memory: CM in locations RAC through RAC plus (FLC-1), and UEM in locations RAE through RAE plus (FLE-1). These two areas never overlap. The user addresses these areas using CM relative addresses 0 through FLC-1 and UEM relative addresses 0 through FLE-1. Refer to figure II-4-3.

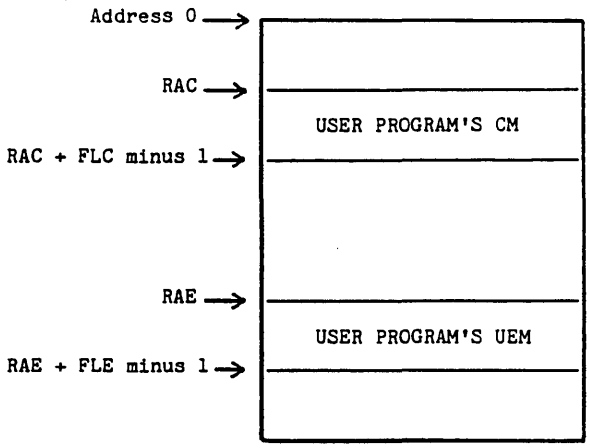


Figure II-4-3. CM/UEM Memory Map

OVERVIEW

Certain errors and conditions occurring in the computer are handled differently than in a model 730 because of the different hardware. Virtual State software is responsible for resolving these errors and special conditions so that the same or similar results occur on both machines.

Virtual State software handles

- All uncorrected processor-detected hardware errors
- Software errors that occur in CYBER 170 monitor mode
- 017 instruction

Software errors that occur in CYBER 170 job mode (with the exception of the 017 instruction) are taken care of by the CP microcode. Results are the same as on a model 730. (The 017 instruction is handled by Virtual State software but, again, results are the same as on a model 730.)

HARDWARE ERRORS

Hardware error exit modes are always enabled. All uncorrected processor-related hardware errors cause Virtual State to take control. For hardware errors in CYBER 170 job mode, control returns to CYBER 170 monitor mode. For hardware errors in CYBER 170 monitor mode, Virtual State simulates a CP halt.

When a hardware error occurs in CYBER 170 monitor mode and the CP is in Virtual State, a PP cannot issue a CYBER 170 exchange request to get out; the machine must be redeadstarted.

SOFTWARE ERRORS

All software errors that cause a CP to halt in a model 730 cause an exchange to Virtual State in models 810 and 830. These instructions are:

- Illegal instruction in CYBER 170 monitor mode
- Read/write address out of range error in CYBER 170 monitor mode with the corresponding exit mode bits set
- Request next instruction (RNI) or branch address out of range in CYBER 170 monitor mode
- Infinite or indefinite value detected in CYBER 170 monitor mode with the corresponding exit mode bits set
- 00 instruction in CYBER 170 monitor mode

Illegal instructions on models 730, 810, and 830 are:

- Instruction 017
- Instructions 011, 012, 013, and 464 through 467 in parcels 1, 2, or 3
- Any 30-bit instruction in parcel 3

On a model 730, instructions 011 and 012 are illegal if ECS is not present; on models 810 and 830, they are illegal if the UEM enable flag is not set in the CYBER 170 exchange package or if bit 23 of X0 and bit 23 of FLE are both set (indicates a flag register operation). Instructions 014 and 015 are always illegal on a model 730; they are illegal on models 810 and 830 only if the UEM enable flag is not set.

If the CP detects an address out of range condition during execution of instructions 50 through 57, the contents of the destination X register is not cleared; it remains unchanged. On a model 730, zeros are written to the X register.

If a K1 or K2 address out of range condition is detected during execution of CMU instructions and exit mode is not selected, no reading or writing is done; CM is unchanged. On a model 730, reading or writing would occur up to the point of the address out of range.

An RNI or branch address out of range that would cause memory to wrap around on a model 730 causes an address out of range error and exchanges to Virtual State on models 810 and 830 if it occurred in CYBER 170 monitor mode.

RAC

For those error conditions that Virtual State software processes, Virtual State writes into RAC the program address (P), the error exit condition code (in the manner described below), and possibly MCR bits in the rightmost 16 bits of RAC.

When an exit condition occurs for which the corresponding mode selection bit is set, the exit condition code is written into RAC. If the corresponding mode selection bit is not set (and therefore, execution continues), the exit condition bit is saved until either an exit condition occurs, which causes an error exit, or an exchange to Virtual State is made. If an error exit is taken, the bit that was saved is written to RAC along with the bit that actually caused the error exit. When an exchange is made to Virtual State, the bit is discarded; because exchanges to Virtual State are transparent to the user, it should not be assumed that the bit will be present.

Error exit condition codes are given under Error Response in section 5 of part I. For software errors, Virtual State writes the code 00 in all cases. For hardware errors, Virtual State writes exit condition codes 20 and 67 only.

Virtual State may set MCR bits in the right- most 16 bits of RAC; if so, these bits enable a Control Data support analyst or customer engineer to further isolate the cause of the error.

CYBER 170 EXCHANGE PACKAGE

The CYBER 170 exchange package differs slightly from that for the model 730. The RAC and FLC registers use 21 bits rather than 18. A new 6-bit register, the flag register, exists in the fourth word. The remaining registers are the same.

The flag register consists of bits 56 through 51. These bits are defined as follows:

Bit	Description
56	UEM enable flag
55	Reserved for hardware; this bit must be set to zero
54	Block copy flag
53	CMU interrupted flag
52	Instruction stack (lookahead) purge flag
51	Hardware error bit

The UEM enable flag must be set to allow the 011, 012, 014, and 015 instructions to access UEM; otherwise, these instructions are illegal. If UEM is not used, this bit is clear.

The block copy flag is set when the block copy instructions (011 and 012) are to use bits 30 through 50 of XO rather than AO to determine the CM address.

The CMU interrupted flag is set when a CMU instruction in progress is interrupted. The information necessary to resume the operation is saved. When the instruction is read again and control switched to Virtual State, the set bit will be noted and the move or compare resumed at the point of interruption.

The instruction stack purge flag is set when extended purging is to be performed. (Refer to Instruction Lookahead in section 4 of this part.)

APPENDIX A
GLOSSARY

GLOSSARY A

ADU	Assembly-disassembly unit
AOR	Address out of range
CEL	Corrected error log
CIF	CMU interrupted flag
CM	Central memory
CMU	Compare/move unit
CP	Central processor
CRT	Cathode-ray tube
CS	Control store
CTI	Common test and initialization
DSC	Display station
DTR	Data terminal ready
ECC	Error correction code
ECL	Emitter-coupled logic
EDS	Extended deadstart
EIA	Electronic Industries Association
EM, EMS	Exit mode selection
EMC	Exit mode condition field at RAC
FIFO	First in, first out
FLC	Field length, central
FLE	Field length, extended
HIVS	Hardware Initialization and Verification
I/O	Input/output
ILH	Instruction lookahead hardware
IOU	Input/output unit
MA	Monitor address
MCH	Maintenance channel
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
RAM	Random access (read-write) memory
RNI	Read next instruction
ROM	Read-only memory
RTS	Request to send
SECDED	Single error correction double error detection
UART	Universal Asynchronous Receiver and Transmitter
UEM	Unified extended memory

INDEX

Comments (continued from other side)

Please fold on dotted line;
seal edges with tape only.

FOLD

FOLD

FOLD

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
First-Class Mail Permit No. 8241 Minneapolis, MN

POSTAGE WILL BE PAID BY ADDRESSEE

CONTROL DATA
Technical Publications
ARH219
4201 N. Lexington Avenue
Arden Hills, MN 55126-9983

COMMENT SHEET

MANUAL TITLE: CYBER 180 Models 810 and 830 Computer Systems Hardware Operator's Guide

PUBLICATION NO.: 60469420 **REVISION:** C

NAME: _____

COMPANY: _____

STREET ADDRESS: _____

CITY: _____ **STATE:** _____ **ZIP CODE:** _____

This form is not intended to be used as an order blank. Control Data Corporation welcomes your evaluation of this manual. Please indicate any errors, suggested additions or deletions, or general comments below (please include page number references).

CUT ALONG LINE

FOLD ON DOTTED LINES

CORPORATE HEADQUARTERS, P.O. BOX O, MINNEAPOLIS, MINN 55440
SALES OFFICES AND SERVICE CENTERS IN MAJOR CITIES THROUGHOUT THE WORLD

LITHO IN U.S.

CONTROL DATA