CYBER 960/962 Computer Systems

Virtual State Volume 2

Reference

Manual History

New features, as well as changes, deletions, and additions to this manual, are indicated by vertical (change) bars in the margins.

Technical changes and additions are indicated by change bars and are correlated with the revision of the page on which they occur. Other changes, such as editorial and pagination, are not identified by change bars but may be included as part of a revision.

Revision	Change Order	Date	Reason for Change
A	-	09-23-88	Manual released.
B	50760	01-15-90	Manual revised; includes miscellaneous corrections and Engineering Change Order 50760 which adds CYBER 960-xxS information.

Contents

Figures

Tables

About This Manual

This manual contains hardware reference information for the CDC® CYBER 960/962 computer systems in their Virtual State of operation.

This manual provides model-independent instruction descriptions and programming information relative to the computer systems hardware.

Audience

This manual is for use by programming and engineering services personnel who operate, program, and maintain the computer systems.

Organization

Chapter 1 – contains the Virtual State central processor (CP) instruction descriptions and peripheral processor (PP) instruction descriptions.

Chapter 2 – contains programming information.

The appendixes consist of a glossary, edit examples, interface information, instruction index, and typical fast DMA transfers.

Conventions

New features, as well as changes, deletions, and additions to this manual, are indicated by vertical bars in the margins.

FCC Compliance

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 (insert peripheral computing device if appropriate) 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.

Related Manuals

Additional hardware reference information regarding operation of the computer systems in both their CYBER 170 State and Virtual State environments is available in manuals shown in the hardware manuals figure on the following page.

CYBER 960/962 HARDWARE MANUALS

SYSTEM MANUALS

Site Preparation

| General Information Site Preparation 60275100 | Mainframe Complex Data Site Preparation 60000119 | Peripheral Equipment Data Site Preparation 60275300 |

Installation

| Mainframe Installation and Checkout 60000120 |

Operation

| Virtual State Reference Volume 1 60000132 | Virtual State Reference Volume 2 60000133 | CYBER 170 State Reference 60000127 |

| 19003 System Console Operations/ Maintenance 60463610 |

Troubleshooting

| System Troubleshooting 60000122 |

Codes

| System Codes Booklet 60458100 |

EQUIPMENT MANUALS

Central Processing Unit (CPU)

| CPU Maintenance 60000123 | CPU Theory and Diagrams 60000118 |

Mainframe Power and Environmental

| Mainframe Power and Environmental Subsystem Maintenance 60000125 | Mainframe Power and Environmental Subsystem Theory and Diagrams 60000121 |

Input/Output Unit (IOU)

| IOU Maintenance 60000130 |

CYBER 960 IOU Manuals

| IOU Theory and Diagrams Levels 0-3 60463540 | IOU Diagrams Level 4 60462210 | Wire Lists Microfiche See note 1 |

CYBER 962 IOU Manuals

| IOU Theory and Diagrams Levels 0-3 60000235 | IOU Diagrams Level 4 60000236 | Wire Lists Microfiche 60000135 |

Motor Generator (MG) Equipment

| 25-kVA Frequency Converter Hardware Maintenance 60456520 | 40-kVA Control Cabinet and Motor Generator Hardware Maintenance 60454720 | MG Interface Unit Installation and Maintenance 60000124 |

Note:

1. Wire list 60000134 applies to AT478-B Nonconcurrent Input/Output (NIO) Unit.
   Wire list 60000446 applies to AT478-C IOU.
   Wire list 60000553 applies to AT478-D IOU.

M02240-2

Additional Related Manuals

Other manuals applicable are:

Title	Publication Number
NOS Version 2 Operator/Analyst Handbook	60459310
NOS Version 2 Systems Programmer's Instant	60459370
NOS Version 1 Operator's Guide	60457700
NOS Version 1 Systems Programmer's Instant	60457790
NOS/VE Analysis Usage	60463915
NOS/VE Operations Usage	60463914
CYBER Initialization Package (CIP) Reference manual	60457180
CYBER 15x Offline MSL Reference Manual	60456530

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

Ordering Manuals

Control Data manuals are available through Control Data sales offices or through:

CONTROL DATA
Literature and Distribution Services
308 North Dale Street
St. Paul, Minnesota 55103-2495

Submitting Comments

Control Data welcomes your comments about this manual. Your comments may include your opinion of the usefulness of this manual, your suggestions for specific improvements, and the reporting of any errors you have found.

You can submit your comments on the comment sheet on the last page of this manual. If the manual has no comment sheet, mail your comments to:

CONTROL DATA
Technical Publications, ARH219
4201 Lexington Avenue North
St. Paul, Minnesota 55126-6198

Please indicate whether you would like a written response.

Disclaimer

This product is intended for use only as described in this document. Control Data cannot be responsible for the proper functioning of undescribed features and parameters.

Instruction Descriptions   1

Instruction Descriptions   1

This chapter contains the Virtual State central processor (CP) instruction descriptions and peripheral processor (PP) instruction descriptions.

Virtual State CP Instructions

The Virtual State CP instructions comprise the following four groups.

● General

● Business data processing (BDP)

● Floating point (FP)

● System

CP Instruction Formats

The CP instructions are 16 or 32 bits long, and have four basic formats.

- Format jkiD (32 bits)

```
0          78  1112 1516 1920          31
+----------+----+----+----+--------------+
|  OPCODE  | j  | k  | i  |      D       |
+----------+----+----+----+--------------+
      8      4    4    4         12
```

- Format SjkiD (32 bits)

```
0     45  78  1112 1516 1920          31
+------+---+----+----+----+-------------+
| OP-  | s | j  | k  | i  |     D       |
| CODE |   |    |    |    |             |
+------+---+----+----+----+-------------+
   5     3   4    4    4        12
```

Field	Description
Opcode	Operation code
j,k,i	Register designators
D	Signed shift count, positive displacement, or bit string descriptor
S	Suboperation code

Business data processing (BDP) instructions using these formats also have one or two 32-bit data descriptor words which are stored in central memory (CM) immediately after the instruction. (Refer to BDP Instruction Descriptions in this chapter.)

● Format jk (16 bits)

```
0            78  1112 15
  | OPCODE  |  j  |  k  |
      8        4     4
```

Field	Description
Opcode	Operation code
j	Register designator, suboperation code, or immediate operand
k	Register designator or immediate operand

BDP instructions using this format have two data descriptor words which are stored in CM immediately after the instruction. (Refer to BDP Instruction Descriptions in this chapter.)

● Format jkQ (32 bits)

```
0          78  1112 1516                    31
  | OPCODE |  j  |  k  |          Q           |
      8       4     4             16
```
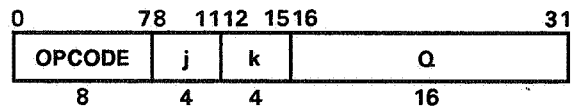
Field	Description
Opcode	Operation code
j,k	Register designators, suboperation codes, or immediate operand value
Q	Signed displacement or immediate operand value

Instruction Description Nomenclature

The instruction descriptions in this chapter use the following address, register, and instruction designators.

Designator	Description
j,k,i,Q,D, or S	Refer to corresponding field in CP Instruction Formats in this chapter
Aj or Ak	One of sixteen 48-bit A registers (A0-AF) specified by j or k field
Xj or Xk	One of sixteen 64-bit X registers (X0-XF) specified by j or k field
XXj or XXk	A double-length X register comprised of Xj and $X(j+1)$ or Xk and $X(k+1)$
XjL or XkL	Bits 0 through 31 of an X register
XjR or XkR	Bits 32 through 63 of an X register
2^n*Xk 2^n*Q 2^n*D	Xk, Q, or D left-shifted n places with zero fill on right, for example, 8*Q shifts Q three places
BN	Byte number (BN) field of a process virtual address (PVA)
SEG	Segment number (SEG) field of a process virtual address
RN	Ring number (RN) field of a process virtual address
( )	Content of memory location (address is the quantity in parentheses)

Additional designators used with the BDP instructions are listed in BDP Nomenclature in this chapter.

Interrupts

Refer to CP Interrupts in chapter 2 of this manual for further information on interrupts. Exceptions caused by instruction execution are noted according to residence in the monitor condition register (MCR) or the user condition register (UCR) and are listed with instruction descriptions. The following exceptions occur independently of instruction execution and, therefore, are not listed.

Bit	Description
MCR 48	Detected uncorrectable error
MCR 50	Short warning
MCR 53	CYBER 170 State exchange request
MCR 56	External interrupt
MCR 59	System interval timer
MCR 62	Soft error log
MCR 63	Trap exception
UCR 51	Process interval timer
UCR 50	Free flag

CP General Instructions

The 84 CP general instructions are divided into 10 subgroups. Tables 1-1 through 1-10 list the instructions in each subgroup. The subgroups are as follows.

● Load and store

● Integer arithmetic

● Branch

● Copy

● Address arithmetic

● Enter

● Shift

● Logical

● Register bit string

● Mark to Boolean

CP Load and Store Instructions

The load and store instructions (table 1-1) transfer a single bit, byte string, 64-bit word, or multiple 64-bit words between one or more registers and one or more CM locations. Store instructions do not alter any register serving as the source of the data transferred to CM.

Table 1-1. CP Load and Store Instructions

Opcode	Format	Instruction	Mnemonic
80	jkQ	Load multiple	LMULT
81	jkQ	Store multiple	SMULT
82	jkQ	Load word	LX
83	jkQ	Store word	SX
A2	jkiD	Load word, indexed	LXI
A3	jkiD	Store word, indexed	SXI
84	jkQ	Load address	LA
85	jkQ	Store address	SA
A0	jkiD	Load address, indexed	LAI
A1	jkiD	Store address, indexed	SAI
A4	jkiD	Load bytes	LBYT
A5	jkiD	Store bytes	SBYT
DS	jkiD	Load bytes, immediate	LBYTS
DS	jkiD	Store bytes, immediate	SBYTS
86	jkQ	Load bytes, relative	LBYTP
88	jkQ	Load bit	LBIT
89	jkQ	Store bit	SBIT

The following interrupt conditions apply to all load and store instructions. Refer to CP Interrupts in chapter 2 of this manual for descriptions of these conditions.

● Address specification error

● Invalid segment/ring number zero

● Access violation

● Page table search without find

● Debug

Load/Store Multiple

Opcode   80jkQ

Mnemonic   LMULT Xk,Aj,Q

Instruction   Load Multiple Registers, from (Aj displaced by 8*Q), selectivity per XkR

Opcode   81jkQ

Mnemonic   SMULT Xk,Aj,Q

Instruction   Store Multiple Registers, at (Aj displaced by 8*Q), selectivity per XkR

Format

0	78	1112	1516	31
80,81	j	k	Q	

Remarks   These instructions transfer data between the A and X registers and contiguous word locations in CM.

The CM starting address forms by left-shifting Q three places with zero insertion on right, and adding the shifted Q to the BN field of the PVA from Aj.

XkR bits 48 through 63 specify which contiguous A and X registers are transferred as follows.

48	5152	5556	59 60	63
A−FIRST	X−FIRST	A−LAST	X−LAST	

XkR Bits	Register Transferred
48-51	First A register
52-55	First X register
56-59	Last A register
60-63	Last X register

The A registers transfer first. When A-first exceeds A-last, no A registers transfer. When X-first exceeds X-last, no X registers transfer.

For example, when:

A-first $= B_{16}$, X-first $= 2_{16}$
A-last $= 4_{16}$, X-last $= C_{16}$

the instruction does not transfer any A registers and transfers X registers 2 through C.

The store multiple instruction clears CM bits 0 through 15 when storing the A registers. The load multiple instruction unconditionally transfers bits 20 through 63 of each CM word to the corresponding bits of the designated A registers. Bits 16 through 19 (ring number field) of each A register are set to the largest of the following.

● Bits 16 through 19 of the CM word

● Bits 16 through 19 of Aj

● Bits 8 through 11 (R1 field) of the segment descriptor associated with the PVA in Aj

During a debug scan operation, the PVA resulting from the addition of Aj and Q is the only data read argument for the load multiple instruction, or the only data write argument for the store multiple registers instruction. Refer to Debug in chapter 2 of this manual.

Load/Store Word

Opcode	**82jkQ**
Mnemonic	LX Xk,Aj,Q
Instruction	Load Xk, from (Aj displaced by 8*Q)
Opcode	**83jkQ**
Mnemonic	SX Xk,Aj,Q
Instruction	Store Xk, at (Aj displaced by 8*Q)

Format

```
0          78   1112 1516                   31
 _____
|  82,83   |  j  |  k  |          Q            |
|_____|_____|_____|_____|
```
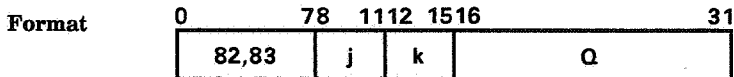
Remarks   These instructions transfer one word between Xj and CM. The CM address
of the word transferred is the sum eight times Q plus the BN field
from Aj.

Load/Store Word, Indexed

Opcode	**A2jkiD**
Mnemonic	LXI Xk,Aj,Xi,D
Instruction	Load Xk, from (Aj displaced by 8*D and indexed by 8*XiR)
Opcode	**A3jkiD**
Mnemonic	SXI Xk,Aj,Xi,D
Instruction	Store Xk, at (Aj displaced by 8*D and indexed by 8*XiR)

Format

```
0          78   1112 1516 1920              31
 _____
|  A2,A3   |  j  |  k  |  i  |       D        |
|_____|_____|_____|_____|_____|
```
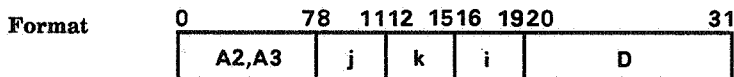
Remarks   These instructions transfer one word between Xk and a word address in
CM. The CM address of the word is the BN field from Aj plus eight times
XiR (index), plus eight times D (displacement). For indexing, these
instructions interpret the X0 contents as zeros. Aj bits 61 through 63 must
be zeros or an address specification error occurs.

Load/Store Address

Opcode	**84jkQ**
Mnemonic	LA Ak,Aj,Q
Instruction	Load Ak, from (Aj displaced by Q)
Opcode	**85jkQ**
Mnemonic	SA Ak,Aj,Q
Instruction	Store Ak, at (Aj displaced by Q)

Format

0	78	1112	1516	31
84,85	j	k	Q	

Remarks These instructions transfer a 6-byte field between Ak and CM. The field's leftmost byte address is the sum of Q (sign-extended to 32 bits) plus the BN field from Aj.

The load Ak instruction transfers the rightmost 44 bits of the 6-byte CM field to Ak bits 20 through 63. The value transferred to Ak bits 16 through 19 is the largest of the following.

- Leftmost four bits of the 6-byte CM field

- Initial Aj bits 16 through 19

- Bits 8 through 11 (R1 field) of the segment descriptor associated with the PVA in Aj

Load/Store Address, Indexed

Opcode   A0jkiD

Mnemonic   LAI Ak,Aj,Xi,D

Instruction Load Ak, at (Aj displaced by D and indexed by XiR)

Opcode   AljkiD

Mnemonic   SAI Ak,Aj,Xi,D

Instruction Store Ak, at (Aj displaced by D and indexed by XiR)

Format

0	78	1112	1516	1920	31
A0,A1	j	k	i	D	

Remarks   These instructions transfer six bytes between Ak and a 6-byte CM field. The starting (leftmost) CM address of the 6-byte field is the sum of the displacement D plus the index value XiR plus the BN field from Aj. For indexing, these instructions interpret the X0 contents as zeros.

The load Ak instruction unconditionally transfers only the rightmost 44 bits of the 6-byte CM field to Ak bit positions 20 through 63. The instruction transfers to Ak bits 16 through 19 a value that is the largest of the following.

● Leftmost four bits of the 6-byte CM field

● Aj bits 16 through 19

● Bits 8 through 11 (R1 field) of the segment descriptor for the PVA in Aj

Load/Store Bytes

Opcode	A4jkiD
Mnemonic	LBYT,X0 Xk,Aj,Xi,D
Instruction	Load Bytes, to Xk from (Aj displaced by D and indexed by XiR), length per X0
Opcode	A5jkiD
Mnemonic	SBYT,X0 Xk,Aj,Xi,D
Instruction	Store Bytes, from Xk at (Aj displaced by D and indexed by XiR), length per X0

Format

0	78	1112	1516	1920	31
A4,A5	j	k	i	D	

Remarks These instructions transfer a field of bytes between Xk and CM. The byte field length equals one plus X0 bits 61 through 63. For lengths less than eight, the load byte instruction right-justifies and zero-extends the bytes loaded into Xk.

The beginning (leftmost) CM address of the byte field is the sum of D (zero-extended to 32 bits) plus XiR, plus the BN field from Aj.

Load/Store Bytes, Immediate

Opcode	DSjkiD
Mnemonic	LBYTS,S Xk,Aj,Xi,D
Instruction	Load Bytes, to Xk from (Aj displaced by D and indexed by XiR), length per S (DS = D0 through D7)
Opcode	DSjkiD
Mnemonic	SBYTS,S Xk,Aj,Xi,D
Instruction	Store Bytes, from Xk at (Aj displaced by D and indexed by XiR), length per S (DS = D8 through DF)

Format

0	78	1112	1516	1920	31
DS	j	k	i	D	

Remarks These instructions transfer a field of bytes between Xk and CM. The field length equals S plus one. For lengths less than eight, the load instruction right-justifies and zero-extends the bytes loaded into Xk.

The beginning (leftmost) CM address of the byte string is the sum of D (displacement) plus XiR (index), plus the BN field of Aj. For indexing, these instructions interpret the X0 contents as zeros.

Load Bytes, Relative

Opcode 86jkQ

Mnemonic LBYTP,j,Xk,Q

Instruction Load Bytes, to Xk at (P displaced by Q), length per j

Format

0	78	1112	1516	31
86	j	k	Q	

Remarks This instruction transfers a field of bytes from CM to Xk. The CM byte field length is the value of the rightmost three bits of j plus one. For lengths less than eight, the byte(s) loaded into Xk are right-justified and zero-extended on the left. The starting (leftmost) CM byte address is the sum of Q (sign-extended to 32 bits) plus the BN field from P.

For this instruction, the CP considers the read operation for the byte field an instruction fetch rather than a data read, and therefore tests the fetch for execute access validity. Refer to Access Protection in chapter 2 of this manual.

Load/Store Bit

Opcode   **88jkQ**

Mnemonic   LBIT Xk,Aj,Q,X0

Instruction   Load Bit, to Xk from (Aj displaced by Q and bit-indexed by X0R)

Opcode   **89jkQ**

Mnemonic   SBIT Xk,Aj,Q,X0

Instruction   Store Bit, from Xk at (Aj displaced by Q and bit-indexed by X0R)

Format

0	78	1112	1516	31
88,89	j	k	Q	

Remarks   These instructions transfer a single bit between XkR bit 63 and a bit position in CM. The load bit instruction also clears Xk bits 0 through 62.

The instructions first generate the CM address of the byte containing the bit loaded or stored as follows.

1. Form byte index by right-shifting X0R three bit positions, end-off, and sign-extend to 32 bits.

2. Form the sum of this 32-bit byte index plus Q (sign-extended to 32 bits) plus BN field from Aj.

These instructions then use the original X0R bits 61 through 63 to select the bit position within the addressed byte. Binary values 0 through 7 for these 3 bits select the corresponding bit position (0 through 7) within the byte.

The store bit (89) instruction executes as follows: the byte containing the bit to be stored is read, modified in the appropriate bit position, and written. No other accesses from any port to the addressed byte are permitted between these read and write accesses. Clearing a synchronization lock with this instruction requires preserialization which can be achieved as follows: a test and set bit (14) instruction (which postserializes) issues immediately before the store bit instruction. This postserialization effectively preserializes the lock clearing.

For the store bit instruction, operand access validation consists of write access validation only.

CP Integer Arithmetic Instructions

The instructions in this subgroup (table 1-2) perform integer arithmetic on signed twos-complement words or half words in Xk or XkR. The sign bit is bit 0 for full-word integers or bit 32 for half-word integers.

Table 1-2. CP Integer Arithmetic Instructions

Opcode	Format	Instruction	Mnemonic
20	jk	Half-word integer sum	ADDR
28	jk	Half-word integer sum, immediate	INCR
8A	jkQ	Half-word integer sum, signed immediate	ADDRQ
24	jk	Integer sum	ADDX
10	jk	Integer sum, immediate	INCX
8B	jkQ	Integer sum, signed immediate	ADDXQ
21	jk	Half-word integer difference	SUBR
29	jk	Half-word integer difference, immediate	DECR
25	jk	Integer difference	SUBX
11	jk	Integer difference, immediate	DECX
22	jk	Half-word integer product	MULR
8C	jkQ	Half-word integer product, signed immediate	MULRQ
26	jk	Integer product	MULX
B2	jkQ	Integer product, signed immediate	MULXQ
23	jk	Half-word integer quotient	DIVR
27	jk	Integer quotient	DIVX
2C	jk	Half-word integer compare	CMPR
2D	jk	Integer compare	CMPX

The integer format is as follows.

```
01                                                                       63
s|          REGISTER Xk                      FULL-WORD INTEGER            |
```

```
0                                        313233                          63
///////////// REGISTER Xk /////////////|s|      HALF-WORD INTEGER         |
```
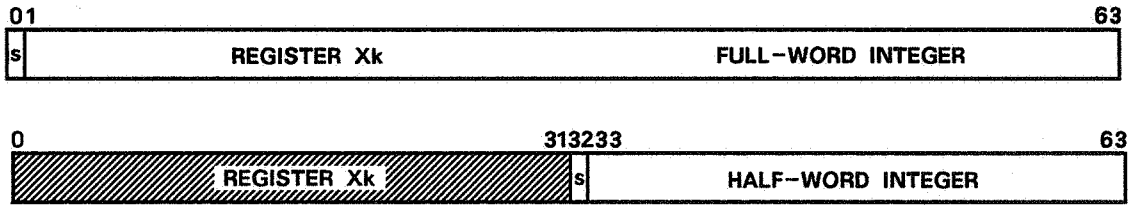
The half-word integer instructions do not alter X register bits 0 through 31.

The arithmetic overflow interrupt condition applies to all integer arithmetic instructions. Individual instruction descriptions list additional interrupt conditions where applicable. (Refer to CP Interrupts in chapter 2 of this manual for descriptions of these conditions.)

Half-Word Integer Sum

Opcode   20jk

Mnemonic   ADDR Xk,Xj

Instruction Half-Word Integer Sum, XkR replaced by XkR plus XjR

Format

0	78	1112	15
20		j	k

Remarks   This instruction forms XkR plus XjR and transfers the 32-bit sum to XkR.

Opcode   28jk

Mnemonic   INCR Xk,j

Instruction Half-Word Integer Sum, Immediate, XkR replaced by XkR plus j

Format

0	78	1112	15
28		j	k

Remarks   This instruction forms XkR plus j zero-extended to 32 bits and transfers the 32-bit sum to XkR.

Opcode   8AjkQ

Mnemonic   ADDRQ Xk,Xj,Q

Instruction Half-Word Integer Sum, Signed Immediate, XkR replaced by XjR plus Q

Format

0	78	1112	1516	31
8A		j	k	Q

Remarks   This instruction forms XjR plus Q sign-extended to 32-bits and transfers the 32-bit sum to XkR.

Integer Sum

Opcode   24jk

Mnemonic   ADDX Xk,Xj

Instruction   Integer Sum, Xk replaced by Xk plus Xj

Format

0	78	1112 15
24	j	k

Remarks   This instruction forms Xk plus Xj and transfers the 64-bit sum to Xk.

Opcode   10jk

Mnemonic   INCX Xk,j

Instruction   Integer Sum Immediate, Xk replaced by Xk plus j

Format

0	78	1112 15
10	j	k

Remarks   This instruction forms Xk plus j zero-extended on the left to 64 bits and transfers the 64-bit sum to Xk.

Opcode   8BjkQ

Mnemonic   ADDXQ Xk,Xj,Q

Instruction   Integer Sum Signed Immediate, Xk replaced by Xj plus Q

Format

0	78	1112	1516	31
8B	j	k	Q	

Remarks   This instruction forms Xj plus Q sign-extended to 64 bits and transfers the 64-bit sum to Xk.

Half-Word Integer Difference

Opcode	21jk
Mnemonic	SUBR Xk,Xj
Instruction	Half-Word Integer Difference, XkR replaced by XkR minus XjR
Opcode	29jk
Mnemonic	DECR Xk,j
Instruction	Half-Word Integer Difference, Immediate, XkR replaced by XkR minus j

Format

```
0          78   1112  15
┌──────────┬────┬──────┐
│  21,29   │ j  │  k   │
└──────────┴────┴──────┘
```

Remarks   These instructions subtract the 32-bit subtrahend in XjR, or in the j field zero-extended to 32 bits on the left, from the 32-bit minuend in XkR and transfer the 32-bit difference to XkR. The instructions treat each half word as a signed twos-complement integer.

Integer Difference

Opcode	25jk
Mnemonic	SUBX Xk,Xj
Instruction	Integer Difference, Xk replaced by Xk minus Xj
Opcode	11jk
Mnemonic	DECX Xk,j
Instruction	Integer Difference, Immediate, Xk replaced by Xk minus j

Format

```
0          78   1112  15
┌──────────┬────┬──────┐
│  11,25   │ j  │  k   │
└──────────┴────┴──────┘
```

Remarks   These instructions subtract Xj, or j zero-extended to 64 bits on the left, from Xk, and transfer the 64-bit difference to Xk. The instructions treat each word as a signed twos-complement integer.

Half-Word Integer Product

Opcode   22jk

Mnemonic   MULR Xk,Xj

Instruction   Half-Word Integer Product, XkR replaced by XkR times XjR

Format

0	78	1112	15
22	j	k	

Remarks   This instruction multiplies XkR by XjR and forms an intermediate 64-bit product. The rightmost 32 bits of this product transfer to XkR as the final product.

Opcode   8CjkQ

Mnemonic   MULRQ Xk,Xj,Q

Instruction   Half-Word Integer Product, Signed Immediate, XkR replaced by XjR times Q

Format

0	78	1112	1516	31
8C	j	k	Q	

Remarks   The first instruction multiplies XkR by XjR. The second instruction multiplies the Q field (sign-extended to 32 bits) by XjR. The multiplication forms an algebraically-signed, 64-bit intermediate product. The rightmost 32 bits of this intermediate product transfer to XkR as the final product.

Integer Product

Opcode   **26jk**

Mnemonic   MULX Xk,Xj

Instruction   Integer Product, Xk replaced by Xk times Xj

Format

0	78	1112	15
26	j	k	

Remarks   This instruction multiplies the signed twos-complement integers in Xk and Xj to form an algebraically-signed, 128-bit intermediate product. The rightmost 64 bits of this intermediate product transfer to Xk as the final product.

Opcode   **B2jkQ**

Mnemonic   MULXQ Xk,Xj,Q

Instruction   Integer Product, Signed Immediate, Xk replaced by Xj times Q

Format

0	78	1112	1516	31
B2	j	k	Q	

Remarks   This instruction multiplies the Q field sign-extended to 64 bits by Xj. An algebraically-signed, 128-bit intermediate product forms. The rightmost 64 bits of this intermediate product transfer to Xk as the final product.

Half-Word Integer Quotient

Opcode   **23jk**

Mnemonic   DIVR Xk,Xj

Instruction   Half-Word Integer Quotient, XkR replaced by XkR divided by XjR

Format

0	78	1112	15
23	j	k	

Remarks   This instruction divides XkR by XjR and transfers the algebraically-signed, 32-bit quotient to XkR. A divide fault (UCR bit 55) interrupt condition occurs if XjR is equal to zero.

Integer Quotient

Opcode   27jk

Mnemonic   DIVX Xk,Xj

Instruction   Integer Quotient, Xk replaced by Xk divided by Xj

Format

```
0           78  1112 15
┌──────────┬────┬────┐
│    27    │ j  │ k  │
└──────────┴────┴────┘
```

Remarks   This instruction divides the word in Xk by the 64-bit word in Xj and transfers the result, consisting of an algebraically-signed, 64-bit quotient to Xk. A divide fault (UCR bit 55) interrupt condition occurs if Xj is equal to zero.

Half-Word Integer/Integer Compare

Opcode   2Cjk

Mnemonic   CMPR X1,Xj,Xk

Instruction   Half-Word Integer Compare, XjR to XkR, result to X1R

Opcode   2Djk

Mnemonic   CMPX X1,Xj,Xk

Instruction   Integer Compare, Xj to Xk, result to X1R

Format

```
0           78  1112 15
┌──────────┬────┬────┐
│   2C,2D  │ j  │ k  │
└──────────┴────┴────┘
```

Remarks   These instructions algebraically compare the twos-complement binary integer in XjR or Xj to the signed twos-complement binary integer in XkR or Xk, respectively. X0 is interpreted as containing all zeros. Based on the comparison result, X1R sets as follows.

Condition	Action Taken
Xj = Xk	Clear X1R bits 32 through 63
Xj ≥ Xk	Clear X1R bits 32 and 34 through 63, set bit 33
Xj ≤ Xk	Clear X1R bits 34 through 63, set bits 32 and 33

CP Branch Instructions

This subgroup (table 1-3) consists of both conditional and unconditional branch instructions. Each conditional branch instruction compares the contents of two general registers to determine whether a normal or a branch exit is taken.

Table 1-3. CP Branch Instructions

Opcode	Format	Instruction	Mnemonic
2E	jk	Branch relative	BRREL
2F	jk	Branch intersegment	BRDIR
90	jkQ	Branch on half word equal	BRREQ
91	jkQ	Branch on half word not equal	BRRNE
92	jkQ	Branch on half word greater than	BRRGT
93	jkQ	Branch on half word greater than or equal	BRRGE
94	jkQ	Branch on equal	BRXEQ
95	jkQ	Branch on not equal	BRXNE
96	jkQ	Branch on greater than	BRXGT
97	jkQ	Branch on greater than or equal	BRXGE
9C	jkQ	Branch and increment	BRINC
9D	jkQ	Branch on segments unequal	BRSEG

The debug interrupt condition applies to all branch instructions. Individual instruction descriptions list additional interrupt conditions where applicable. Refer to CP Interrupts in chapter 2 of this manual for a description of these conditions.

Branch Relative

Opcode   2Ejk

Mnemonic   BRREL Xk

Instruction   Branch to P indexed by 2*XkR

Format

```
0          78   1112 15
┌──────────┬────┬────┐
│    2E    │  j │  k │
└──────────┴────┴────┘
```

Remarks   This instruction causes an unconditional branch to the CM address formed by adding two times XkR to the BN field in P.

Branch Intersegment

Opcode   2Fjk

Mnemonic   BRDIR Aj,Xk

Instruction Branch to Aj indexed by 2*XkR

Format

0	78	1112	15
2F		j	k

Remarks   This instruction causes an unconditional branch by modifying the key (KEY), SEG, and BN fields of the PVA in P, as follows.

1. The key in P copies to the lock of the branched-to segment. The branch is permitted if the key and lock are equal, if the key is a master key, or if the lock is zero or equals no lock.

2. The 12-bit SEG field in Aj (bits 20 through 31) transfers to P bits 20 through 31.

3. A value two times XkR adds to the rightmost 32 bits from Aj. X0 consists of all zeros. This sum transfers to bit positions 32 through 63 of P.

This instruction can cause the following exception conditions.

● Address specification error

● Invalid SEG/ring number (RN) zero

● Access violation

Branch on Half Word

Opcode	**90jkQ**
Mnemonic	BRREQ Xj,Xk,Q
Instruction	Branch to P displaced by 2*Q, if XjR equal to XkR
Opcode	**91jkQ**
Mnemonic	BRRNE Xj,Xk,Q
Instruction	Branch to P displaced by 2*Q, if XjR not equal to XkR
Opcode	**92jkQ**
Mnemonic	BRRGT Xj,Xk,Q
Instruction	Branch to P displaced by 2*Q, if XjR greater than XkR
Opcode	**93jkQ**
Mnemonic	BRRGE Xj,Xk,Q
Instruction	Branch to P displaced by 2*Q, if XjR greater than or equal to XkR

Format

0	78	1112	1516	31
90,91,92,93	j	k	Q	

Remarks   These instructions algebraically compare XjR with XkR, treating each as a signed twos-complement binary integer. X0 is interpreted as containing all zeros. If the comparison between XjR and XkR does not satisfy the branch condition specified, the instruction takes a normal exit by adding four to the BN field in P to generate the next instruction address. If the Xj right (XjR) and Xk right (XkR) comparison satisfies the branch condition, the instruction takes a branch exit by adding two times Q to the BN field in P to form the next instruction address.

Branch

Opcode	**94jkQ**
Mnemonic	BRXEQ Xj,Xk,Q
Instruction	Branch to P displaced by 2*Q, if Xj equal to Xk
Opcode	**95jkQ**
Mnemonic	BRXNE Xj,Xk,Q
Instruction	Branch to P displaced by 2*Q, if Xj not equal to Xk
Opcode	**96jkQ**
Mnemonic	BRXGT Xj,Xk,Q
Instruction	Branch to P displaced by 2*Q, if Xj greater than Xk
Opcode	**97jkQ**
Mnemonic	BRXGE Xj,Xk,Q
Instruction	Branch to P displaced by 2*Q, if Xj greater than or equal to Xk

Format

0 78	1112	1516	31
94,95,96,97	j	k	Q

Remarks   These instructions algebraically compare the Xj word with the Xk word, treating each as a signed twos-complement binary integer. X0 is interpreted as containing all zeros.

If the comparison between Xj and Xk does not satisfy the branch condition specified, the instruction takes a normal exit by adding four to the BN field in P to generate the next instruction address. If the comparison satisfies the branch condition, the instruction causes a branch exit by adding two times Q to the BN field in P to form the the next instruction address.

Branch and Increment

Opcode   **9CjkQ**

Mnemonic   BRINC Xj,Xk,Q

Instruction Branch to P displaced by 2*Q and Increment Xk, if Xj greater than Xk

Format

0	78	1112	1516	31
9C	j	k	Q	

Remarks   This instruction algebraically compares the Xj word with the Xk word, treating each as a signed twos-complement binary integer. For Xj only, the instruction interprets X0 as all zeros.

The comparison results are as follows.

Condition	Action Taken
Xj ≤ Xk	Normal exit; add four to BN field in P to form next instruction address
Xj > Xk	Branch exit; add two times Q to BN field in·P to form next instruction address, and increase word in Xk by one; overflow is ignored

Branch on Segments Unequal

Opcode   9DjkQ

Mnemonic   BRSEG Xi,Aj,Ak,Q

Instruction   Branch to P displaced by 2*Q, if segments unequal, else compare byte numbers, result to X1R.

Format

0		78	1112	1516	31
9D		j	k	Q	

Remarks   This instruction performs a bit-for-bit comparison between the SEG fields in Aj and Ak (bits 20 through 31). If the SEG fields are unequal, the instruction takes a branch exit by adding two times Q to the BN field in P to form the next instruction address.

If the SEG fields are equal, the instruction takes a normal exit by adding four to the BN field in P to form the next instruction address. The instruction also algebraically compares Aj bits 32 through 63 with Ak bits 32 through 63, treating each 32-bit quantity as a signed twos-complement binary integer, and stores the comparison result in X1R, as follows.

Result	Action Taken
AJ = Ak	Clear X1R
Aj > Ak	Clear X1R bits 32 and 34 through 63, set bit 33
Aj < Ak	Clear X1R bits 34 through 63, set bits 32 and 33

CP Copy Instructions

The copy instructions (table 1-4) transfer information between registers.

Table 1-4. CP Copy Instructions

Opcode	Format	Instruction	Mnemonic
09	jk	Copy address, A to A	CPYAA
0A	jk	Copy address, X to A	CPYXA
0B	jk	Copy address, A to X	CPYAX
0C	jk	Copy half word	CPYRR
0D	jk	Copy full word	CPYXX

Copy Address

Opcode   09jk

Mnemonic   CPYAA Ak,Aj

Instruction   Copy, Ak replaced by Aj

Format

0	78	1112	15
09	j	k	

Remarks   This instruction transfers the 48 bits in Aj to Ak.

Opcode   0Ajk

Mnemonic   CPYXA Ak,Xj

Instruction   Copy, Ak replaced by Xj

Format

0	78	1112	15
0A	j	k	

Remarks   This instruction unconditionally transfers Xj bits 20 through 63 to the corresponding bit positions of Ak. The instruction also compares Xj bits 16 through 19 with P bits 16 through 19 and transfers the larger field to Ak bits 16 through 19.

Opcode   0Bjk

Mnemonic   CPYAX Xk,Aj

Instruction   Copy, Xk replaced by Aj

Format

0	78	1112	15
0B	j	k	

Remarks   This instruction transfers the 48 bits in Aj to Xk bit positions 16 through 63 and clears Xk bits 0 through 15.

Copy Half Word

Opcode   0Cjk

Mnemonic   CPYRR Xk,Xj

Instruction Copy, XkR replaced by XjR

Format

```
0        78  1112 15
 ┌──────────┬────┬────┐
 │    0C    │ j  │ k  │
 └──────────┴────┴────┘
```

Remarks   This instruction transfers the half word in XjR to XkR. The XkL content does not change.

Copy Full Word

Opcode   0Djk

Mnemonic   CPYXX Xk,Xj

Instruction Copy, Xk replaced by Xj

Format

```
0        78  1112 15
 ┌──────────┬────┬────┐
 │    0D    │ j  │ k  │
 └──────────┴────┴────┘
```

Remarks   This instruction transfers the Xj word to Xk.

CP Address Arithmetic Instructions

Address arithmetic instructions (table 1-5) perform address arithmetic in twos complement, ignoring overflow.

Table 1-5. CP Address Arithmetic Instructions

Opcode	Format	Instruction	Mnemonic
2A	jk	Address increment, indexed	ADDAX
8E	jkQ	Address increment, signed immediate	ADDAQ
8F	jkQ	Address relative	ADDPXQ
A7	jkiD	Address increment, modulo	ADDAD

Address Increment, Indexed

Opcode   2Ajk

Mnemonic   ADDAX Ak,Xj

Instruction   Address Ak replaced by Ak plus XjR

Format

```
0        78  1112 15
+---------+----+----+
|   2A    | j  | k  |
+---------+----+----+
```

Remarks   This instruction adds XjR and Ak bits 32 through 63 and returns the sum to Ak bits 32 through 63.

Address Increment, Signed Immediate

Opcode   8EjkQ

Mnemonic   ADDAQ Ak,Aj,Q

Instruction   Address Ak replaced by Aj plus Q

Format

```
0        78  1112 1516                    31
+---------+----+----+---------------------+
|   8E    | j  | k  |          Q          |
+---------+----+----+---------------------+
```

Remarks   This instruction transfers Aj bits 16 through 31 to the corresponding Ak bit positions. Also, the instruction adds Q (sign-extended to 32 bits) and Aj bits 32 through 63 and transfers the sum to Ak bits 32 through 63. Overflow is ignored.

Address Relative

Opcode	8FjkQ
Mnemonic	ADDPXQ Ak,Xj,Q
Instruction	Address Ak replaced by P plus 2*XjR plus 2*Q
Format	

```
0         78   1112 1516                    31
┌─────────┬─────┬─────┬──────────────────────┐
│   8F    │  j  │  k  │          Q           │
└─────────┴─────┴─────┴──────────────────────┘
```

Remarks   This instruction transfers P bits 16 through 31 to the corresponding 16 bit positions of Ak. The instruction also adds two times Q (sign-extended to 32 bits) to the rightmost 32 bits of P, adds this 32-bit sum to two times the XjR value, and transfers the final sum to Ak bits 32 through 63. Overflow is ignored. The instruction interprets X0 as all zeros.

Address Increment, Modulo

Opcode	A7jkiD
Mnemonic	ADDAD Ak,Ai,D,j
Instruction	Address Ak replaced by Ai plus D per j
Format	

```
0         78   1112 1516 1920              31
┌─────────┬─────┬─────┬─────┬───────────────┐
│   A7    │  j  │  k  │  i  │       D       │
└─────────┴─────┴─────┴─────┴───────────────┘
```

Remarks   This instruction transfers Ai bits 16 through 31 to the corresponding bit positions of Ak. The instruction also adds D (zero-extended to 32 bits on left) to Ai bits 32 through 63 and transfers bits 32 through 60 of this sum to Ak bits 32 through 60. The instruction performs a logical product (AND) between bits 61 through 63 of this 32-bit sum and the rightmost three bits of j and transfers the 3-bit logical product to Ak bits 61 through 63. Overflow is ignored.

The following is an example of the logical product (AND) operation.

First operand   0011
Second operand   0101
                  ────

Result (AND)   0001

CP Enter Instructions

The instructions in this subgroup (table 1-6) enter immediate operands (consisting of logical quantities or signed twos-complement binary integers) into the X registers.

Table 1-6. CP Enter Instructions

Opcode	Format	Instruction	Mnemonic
1F	jk	Enter zeros	ENTZ
1F	jk	Enter ones	ENT0
1F	jk	Enter signs	ENTS
3D	jk	Enter, immediate positive	ENTP
3E	jk	Enter, immediate negative	ENTN
39	jk	Enter X1, immediate logical	ENTX
3F	jk	Enter X0, immediate logical	ENTL
87	jkQ	Enter X1, signed immediate	ENTC
B3	jkQ	Enter X0, signed immediate	ENTA
8D	jkQ	Enter, signed immediate	ENTE

Enter Zeros/Ones/Signs

Opcode   1Fjk

Mnemonic   ENTZ Xk

Instruction   Enter XkL with zeros

Opcode   1Fjk

Mnemonic   ENT0 Xk

Instruction   Enter XkL with ones

Opcode   1Fjk

Mnemonic   ENTS Xk

Instruction   Enter XkL with signs

Format

0	78	1112	15
1F		j	k

Remarks   This instruction translates the rightmost two bits of j as follows.

j Field	Action Taken
xx00	Clear XkL bits 0 through 31
xx01	Set XkL bits 0 through 31
xx10 xx11	Copy bit 32 (sign) of XkR to bits 0 through 31 of XkL

Enter, Immediate Positive/Negative

Opcode	3Djk
Mnemonic	ENTP Xk,j
Instruction	Enter Xk with plus j
Opcode	3Ejk
Mnemonic	ENTN Xk,j
Instruction	Enter Xk with minus j

Format

0	. 78	1112 15
3D,3E	j	k

Remarks   These instructions zero-extend j to 64 bits and transfer this result, or the twos complement of this result, to Xk.

Enter X1/X0, Immediate Logical

Opcode	39jk
Mnemonic	ENTX X1,jk
Instruction	Enter X1 with logical jk
Opcode	3Fjk
Mnemonic	ENTL X0,jk
Instruction	Enter X0 with logical jk

Format

0	78	1112 15
39,3F	j	k

Remarks   These instructions transfer k to bits 60 through 63 and j to bits 56 through 59 of X0 or X1. The instructions clear bits 0 through 55.

Enter X1/X0, Signed Immediate

Opcode	**87jkQ**
Mnemonic	ENTC X1,jkQ
Instruction	Enter X1 with Sign-Extended jkQ
Opcode	**B3jkQ**
Mnemonic	ENTA X0,jkQ
Instruction	Enter X0 with Sign-Extended jkQ

Format

0	78	1112	1516	31
87,B3	j	k	Q	

Remarks These instructions expand the 24-bit concatenation of j, k, and Q (bit positions 8 through 31) to 64 bits (right-justified) by extending the leftmost bit of j through bits 0 through 39. The 64-bit result transfers to X0 or X1.

Enter, Signed Immediate

Opcode	**8DjkQ**
Mnemonic	ENTE Xk,Q
Instruction	Enter Xk with Sign-Extended Q

Format

0	78	1112	1516	31
8D	j	k	Q	

Remarks This instruction sign-extends Q to 64 bits, and transfers this value to Xk.

CP Shift Instructions

The shift instructions (table 1-7) shift the Xj 64 bits through the number of bit positions determined from a computed shift count. The result transfers to Xk.

Table 1-7. CP Shift Instructions

Opcode	Format	Instruction	Mnemonic
A8	jkiD	Shift word, circular	SHFC
A9	jkiD	Shift word, end-off	SHFX
AA	jkiD	Shift half word, end-off	SHFR

The computed shift count is the sum of the D field rightmost eight bits plus XiR bits 56 through 63. An overflow from this 8-bit sum is ignored. The instructions interpret X0 as all zeros. The computed shift count is determined by the following.

1. The leftmost bit of the 8-bit computed shift count determines the shift direction:

   Positive sign: Left shift

   Negative sign: Right shift

2. For right shifts, the actual shift count is the twos complement of the rightmost five or six bits of the computed shift count for 32- and 64-bit operands, respectively. Thus, half words can be shifted from 0 to 31 places left or from 1 to 32 places right. Similarly, full words can be shifted from 0 to 63 places left or from 1 to 64 places right. The shifts are as follows.

Shift Count	32-Bit Shifts	Shift Count	64-Bit Shifts
0111 1111 ⋮	Left shift 31 (repeating) ⋮	0111 1111 ⋮	Left shift 63 ⋮
0010 0000	Left shift 0	0100 0000	Left shift 0
0001 1111 ⋮	Left shift 31 ⋮	0011 1111 ⋮	Left shift 63 ⋮
0000 0000	Left shift 0	0000 0000	Left shift 0
1111 1111 ⋮	Right shift 1 ⋮	1111 1111 ⋮	Right shift 1 ⋮
1110 0000	Right shift 32	1100 0000	Right shift 64
1101 1111 ⋮	Right shift 1 (repeating) ⋮	1011 1111 ⋮	Right shift 1 ⋮
1000 0000	Right shift 32	1000 0000	Right shift 64

3. If the computed shift count results in an actual shift count of zero, Xj transfers to Xk without shifting.

Shift Word, Circular

Opcode   A8jkiD

Mnemonic   SHFC Xk,Xj,Xi,D

Instruction   Shift Circular, Xk replaced by Xj, direction and count per XiR plus D

Format

0	78	1112	1516	1920	31
A8	j	k	i	D	

Remarks   This instruction shifts the Xj word by the computed shift count and transfers the result to Xk. The shift is circular. Bits that shift out one end of the word transfer into bit positions which become unoccupied at the opposite end of the word.

Shift End-Off, Word/Half Word

Opcode   A9jkiD

Mnemonic   SHFX Xk,Xj,Xi,D

Instruction   Shift Word, End-Off, Xk replaced by Xj, direction and count per XiR plus D

Opcode   AAjkiD

Mnemonic   SHFR Xk,Xj,Xi,D

Instruction   Shift Half Word, End-Off, XkR replaced by XjR, direction and count per XiR plus D

Format

0	78	1112	1516	1920	31
A9,AA	j	k	i	D	

Remarks   These instructions shift the Xj word, or the XjR half word, and transfer the result to Xk or XkR. The computed shift count determines the direction and number of bit positions to be shifted. In a right shift, the instruction right-shifts the word end-off on the right and sign-extends on the left. In a left shift, the instruction left-shifts the word end-off on the left and inserts zeros on the right.

CP Logical Instructions

The instructions in this subgroup (table 1-8) perform logical (Boolean) operations on 64-bit operands in the X registers.

Table 1-8. CP Logical Instructions

Opcode	Format	Instruction	Mnemonic
18	jk	Logical sum	IORX
19	jk	Logical difference	XORX
1A	jk	Logical product	ANDX
1B	jk	Logical complement	NOTX
1C	jk	Logical inhibit	INHX

Logical Sum/Difference/Product

Opcode	18jk
Mnemonic	IORX Xk,Xj
Instruction	Logical Sum, Xk replaced by Xk OR Xj
Opcode	19jk
Mnemonic	XORX Xk,Xj
Instruction	Logical Difference, Xk replaced by Xk XOR Xj
Opcode	1Ajk
Mnemonic	ANDX Xk,Xj
Instruction	Logical Product, Xk replaced by Xk AND Xj

Format

```
0          78   1112 15
| 18,19,1A |  j  |  k  |
```

Remarks   These instructions form the logical sum, difference, or product between the words in Xj and Xk, and return the 64-bit Boolean result to Xk. Examples of these operations are as follows.

	Logical Sum (OR)	Logical Difference (XOR)	Logical Product (AND)
First operand	0011	0011	0011
Second operand	0101	0101	0101
Result	0111	0110	0001

Logical Complement

Opcode	1Bjk
Mnemonic	NOTX Xk,Xj
Instruction	Logical Complement, Xk replaced by Xj NOT
Format	

```
0         78  1112 15
+--------+----+----+
|   1B   |  j |  k |
+--------+----+----+
```

Remarks   This instruction transfers the ones complement of the Xj word to Xk. The ones complement of a number results from subtracting the original number, bit for bit, from a number consisting of all ones. For example:

Ones Complement	
Ones	1111
Xj	0110
Xk	1001

Logical Inhibit

Opcode	1Cjk
Mnemonic	INHX Xk,Xj
Instruction	Logical Inhibit, Xk replaced by Xk AND Xj NOT
Format	

```
0         78  1112 15
+--------+----+----+
|   1C   |  j |  k |
+--------+----+----+
```

Remarks   This instruction forms the logical product (AND) between the ones complement of the Xj number and the Xk number and returns the result to Xk. For example:

Logical Inhibit	
Xj	0011
NOT Xj	1100
Xk	0101
Xk AND Xj NOT	0100

CP Register Bit String Instructions

The instructions in this subgroup (table 1-9) address a contiguous string (field) of bits within a register, beginning and ending at any bit position. The instructions interpret X0 as all zeros.

Table 1-9. CP Register Bit String Instructions

Opcode	Format	Instruction	Mnemonic
AC	jkiD	Isolate bit mask	ISOM
AD	jkiD	Isolate bit string	ISOB
AE	jkiD	Insert bit string	INSB

Bit String Descriptor

The beginning bit position and bit string length are specified by a bit string descriptor. The descriptor is the rightmost 12 bits of the sum of D (sign-extended) plus XiR and has the following format.

```
52      57 58       63
┌─────────┬──────────┐
│Leftmost │ String   │
│Position │Length-1  │
└─────────┴──────────┘
```

XiR Bits	Description
52-57	Beginning (leftmost) bit position
58-63	Length, one less than the number of bits in the string

The instruction specification error interrupt condition applies to all register bit string instructions. Refer to CP Interrupts in chapter 2 of this manual for a description of this condition.

Isolate Bit Mask

Opcode   **ACjkiD**

Mnemonic   ISOM Xk,Xi,D

Instruction Isolate Bit Mask, transfer into Xk per XiR plus D

Format

0	78	1112	1516	1920	31
AC	j	k	i	D	

Remarks   This instruction generates a bit mask consisting of a contiguous field of ones and places this field into Xk. The bit string descriptor defines the leftmost bit position and the length of the Xk field. All Xk bits outside the specified string clear.

Isolate

Opcode   **ADjkiD**

Mnemonic   ISOB Xk,Xj,Xi,D

Instruction Isolate, transfer into Xk from Xj per XiR plus D

Format

0	78	1112	1516	1920	31
AD	j	k	i	D	

Remarks   This instruction clears Xk and transfers a field of contiguous data from Xj into Xk, right-justified. The bit string descriptor defines the leftmost bit position and Xj field length.

Insert

Opcode   **AEjkiD**

Mnemonic   INSB Xk,Xj,Xi,D

Instruction Insert, transfer into Xk from Xj per XiR plus D

Format

0	78	1112	1516	1920	31
AE	j	k	i	D	

Remarks   This instruction transfers a field of contiguous bits from Xj to Xk. The field is obtained from the Xj rightmost bit positions, with the length specified by the bit string descriptor. The field inserts into Xk with the leftmost bit position and the field length also specified by the bit string descriptor. All Xk bit positions outside the specified field remain unchanged.

CP Mark to Boolean Instruction

The following instruction tests X1R bits 32 through 33 for values specified by the j field.

Opcode   1Ejk

Mnemonic   MARK Xk,Xi,j

Instruction   Mark to Boolean, set Xk per j and X1R

Format

0	78	1112	15
1E		j	k

Remarks   This instruction tests X1R bits 32 through 33 for a bit combination by comparing j and X1R bits 32 through 33 for an equal condition (EQ) as shown in table 1-10. If X1R bits 32 through 33 equal a value specified by j, the instruction clears Xk bits 01 through 63 and sets Xk bit 0. The instruction clears Xk if no equality occurs.

From left to right, the four bits of j are individual pointers associated with the four possible values of X1R bits 32 and 33 (00, 01, 10, and 11). When set, the first bit in the j field tests bits 32 and 33 for a value of 00, the second bit for 01, the third bit for 10, and the fourth bit for 11. For example, if j equals 0101, equality occurs when bits 32 and 33 are either 01 or 11.

Table 1-10. Compare j Field and X1 Bits 32 and 33

j Field	00	01	10	11
0000	1	1	1	1
0001				EQ
0010			EQ	
0011			EQ	EQ
0100		EQ		
0101		EQ		EQ
0110		EQ	EQ	
0111		EQ	EQ	EQ
1000	EQ			
1001	EQ			EQ
1010	EQ		EQ	
1011	EQ		EQ	EQ
1100	EQ	EQ		
1101	EQ	EQ		EQ
1110	EQ	EQ	EQ	
1111	2	2	2	2

1. Unconditional inequality.

2. Unconditional equality.

BDP Instruction Descriptions

The BDP instruction group consists of 18 opcodes in three subgroups:

- BDP numeric

- Byte

- Subscript and immediate data

Tables 1-11 through 1-14 list the instructions within each subgroup. For descriptions of source and destination fields, data descriptors, access types, data formats, and data types, refer to BDP Programming in chapter 2 of this manual.

BDP Nomenclature

The BDP instruction descriptions use the following additional terms.

Term	Description
D(Aj)	Source data field addressed by PVA in Aj
D(Ak)	Other source data field, or destination data field, addressed by PVA in Ak
D(Ai+D)	Edit mask addressed by PVA in Ai plus displacement D. The edit (ED) instruction uses this term

BDP Numeric Instructions

The instructions in this subgroup (table 1-11) perform arithmetic, shift, conversion, and comparison operations on byte fields from CM.

Table 1-11. BDP Numeric Instructions

Opcode	Format	Instruction	Mnemonic
70	jk	Decimal sum	ADDN
71	jk	Decimal difference	SUBN
72	jk	Decimal product	MULN
73	jk	Decimal quotient	DIVN
74	jk	Decimal compare	CMPN
75	jk	Numeric move	MOVN
E4	jkiD	Decimal scale	SCLN
E5	jkiD	Decimal scale rounded	SCLR

After completing the required operation, the instructions store the right-justified result in the destination field. These instructions also do the following.

- Zero-fill the high-order destination field positions if the decimal result is shorter than the destination field

- Truncate the result's leftmost bits if the result exceeds the destination field

- Treat a decimal numeric value of minus zero as equal to plus zero

- Do not store minus zero as a result, except when truncation takes place

An instruction specification error occurs if the length and type fields in the source and destination field data descriptors do not conform to the length and type allowed for a particular instruction. This inhibits instruction execution and initiates the corresponding program interrupt.

The following conditions apply to all BDP numeric instructions.

- Instruction specification error

- Address specification error

- Invalid segment/ring number zero

- Access violation

- Page table search without find

- Debug

- Invalid BDP data

A destination BDP operand of length zero transforms the instruction into a no-operation. However, when the source field length is nonzero, exception sensing for the source field occurs. This includes testing for arithmetic loss-of-significance and overflow, but excludes testing for a divide fault.

Individual instruction descriptions list additional interrupt conditions, where applicable. Refer to CP Interrupts in chapter 2 of this manual for descriptions of these conditions.

BDP data descriptor format is shown below.

```
01  34   78        1516              31
┌─┬───┬────┬─────────┬────────────────┐
│F│ D │ T  │    L    │       O        │
└─┴───┴────┴─────────┴────────────────┘
```

Field	Description
F	Function of the L field (1 bit). Length retrieval information, as follows.

 0   Length is obtained from the L field.

 1   Length of the descriptor associated with Aj is obtained from X0R bits 55 through 63. Length of the descriptor associated with Ak is obtained from X1R bits 55 through 63. Other bits in X0R and X1R are not used.

D	Reserved (3 bits).
T	Data type (4 bits) (refer to table 2-5).
L	Length (in bytes) of the source or destination field (8 bits) (refer to table 2-5). The maximum length is restricted according to the operand data type. When the maximum length is exceeded, an instruction specification error occurs, causing an interrupt or halt.
O	Offset (16 bits). PVA of the leftmost byte of source or destination field is obtained by adding the sign-extended O field to the BN field of the base PVA in Aj or Ak, respectively.

Decimal Arithmetic

Opcode	70jk
Mnemonic	ADDN,Aj,X0 Ak,X1
Instruction	Decimal Sum, D(Ak) replaced by D(Ak) plus D(Aj) (2 descriptors)
Opcode	71jk
Mnemonic	SUBN,Aj,X0 Ak,X1
Instruction	Decimal Difference, D(Ak) replaced by D(Ak) minus D(Aj) (2 descriptors)
Opcode	72jk
Mnemonic	MULN,Aj,X0 Ak,X1
Instruction	Decimal Product, D(Ak) replaced by D(Ak) times D(Aj) (2 descriptors)
Opcode	73jk
Mnemonic	DIVN,Aj,X0 Ak,X1
Instruction	Decimal Quotient, D(Ak) replaced by D(Ak) divided by D(Aj) (2 descriptors)

Format

```
0          78   1112 15
┌───────────┬────┬─────┐
│70,71,72,73│  j │  k  │
└───────────┴────┴─────┘
```

Remarks   These instructions perform arithmetic operations on the initial destination field (an augend, minuend, multiplicand, or dividend) and the source field (an addend, subtrahend, multiplier, or divisor). The decimal result (sum, difference, product, or quotient) transfers to the destination field.

The instructions allow packed and unpacked decimal data types 0 through 6. They do not support unpacked decimal leading sign data types 7 and 8. A numeric move (75) instruction must be used to format operands of these types prior to use in arithmetic operations.

The instruction results are algebraically signed. If the results equal zero with no loss-of-significance, a positive sign is entered. The result translates to the preferred codes of the data type specified by the destination field data descriptor.

These instructions can cause the following exception conditions.

● Arithmetic overflow

● Divide fault (instruction 73 only, refer to table 1-12)

Table 1-12. BDP Divide Fault

k Field Length	k Value	j Field Length	j Value	Divide Fault
0	1[1]	0	1[1]	No
0	1[1]	Nonzero	0	No
0	1[1]	Nonzero	Nonzero	No
Nonzero	0	0	1[1]	Yes
Nonzero	0	Nonzero	0	Yes
Nonzero	0	Nonzero	Nonzero	No
Nonzero	Nonzero	0	1[1]	Yes
Nonzero	Nonzero	Nonzero	0	Yes
Nonzero	Nonzero	Nonzero	Nonzero	No

1. Since field length is zero, the data is disregarded.

Decimal Compare

Opcode   74jk

Mnemonic   CMPN,Aj,X0 Ak,X1

Instruction Decimal Compare, D(Aj) to D(Ak), result to X1R (2 descriptors)

Format

```
0        78  1112 15
┌─────────┬────┬────┐
│   74    │ j  │ k  │
└─────────┴────┴────┘
```

Remarks   This instruction algebraically compares the decimal contents of the source and destination fields, and depending on the comparison results, transfers a half word to X1R as follows.

Condition	Action Taken
D(Aj) = D(Ak)	Clear X1R
D(Aj) > D(Ak)	Clear X1R bits 32 and 34 through 63, set bit 33
D(Aj) < D(Ak)	Clear X1R bits 34 through 63, set bits 32 and 33

The instruction allows data types 0 through 6. The maximum operand length is a function of the data type. The instruction accommodates unequal field lengths by using decimal zero fill in the leftmost positions of the shorter-length field.

Numeric Move

Opcode	**75jk**
Mnemonic	MOVN,Aj,X0 Ak,X1
Instruction	Numeric Move, D(Ak) replaced by D(Aj) after formatting (2 descriptors)
Format	

```
0        78  1112 15
|   75   |  j  |  k  |
```

Remarks

This instruction obtains a number from the source field, validates the number according to the T field from its associated data descriptor, reformats it according to the T field in the destination field data descriptor, and transfers the result to the destination field.

The instruction can convert and format any combination of data types 0 through 8 and 10 or 11. If the conversion is from a decimal data type to a binary data type, the decimal data type determines the maximum length for the source as follows.

Source Field Data Type	Maximum Source Field Length (Bytes)
0 through 3	19
4 through 8	38

The maximum destination field length is eight bytes. The instruction truncates the leftmost bytes if the destination field is not long enough to accommodate the entire binary number, or extends the sign bit on the left if the destination field exceeds the conversion result. When truncation places a negative zero into the destination field, it is not changed to positive zero.

The same length restrictions apply if the source is a binary data type and the destination is a decimal data type, except that if the receiving field exceeds the converted number, the instruction adds leading zeros according to the decimal data type [ASCII character zero (30_{16}) or digit zero (0)].

When both operands are decimal, the destination field fills from right to left. If the field lengths are unequal, the instruction either truncates leading digits or inserts leading zeros according to the destination data type.

This instruction can cause the arithmetic loss-of-significance exception condition.

Decimal Scale

Opcode E4jkiD

Mnemonic SCLN,Aj,X0 Ak,X1,Xi

Instruction Decimal Scale, D(Ak) replaced by D(Aj), scaled per XiR plus D (2 descriptors)

Opcode E5jkiD

Mnemonic SCLR,Aj,X0 Ak,X1,Xi

Instruction Decimal Scale Rounded, D(Ak) replaced by rounded D(Aj), scaled per XiR plus D (2 descriptors)

Format

0	78	1112	1516	1920	31
E4,E5	j	k	i	D	

Remarks These shift instructions move data from the source field to the destination field, shifting the data under control of a computed shift count. This count is the 8-bit sum of the twos-complement 32-bit integer from XiR plus the D-field rightmost eight bits of the instruction. Any overflow from the 8-bit sum is ignored. The X0 contents interpret as all zeros. The instruction acts as a move instruction if the shift count equals zero.

With positive shift count (bit 56 = 0), the source data left-shifts as determined by bits 57 through 63 of the computed shift count. A negative shift count (bit 56 = 1) causes a shift to the right. In this case, the number of positions is determined by the twos complement of bits 57 through 63 of the computed shift count. A value of 1000 0000 interprets as a right shift of 128 positions.

A positive shift count effectively multiplies the source data by powers of 10; a negative shift count divides the source data by powers of 10. The shifting occurs as the data moves from the source to the destination field. Shifting is end-off with zero-fill as required to accommodate the length and type specified for the destination field. The source field sign moves the destination field unchanged.

The shift counts are interpreted as follows.

Shift Count	Shifts
01111 1111	Left shift 127
$\vdots$	$\vdots$
0000 0000	Left shift 0
1111 1111	Right shift 1
$\vdots$	$\vdots$
1000 0001	
1000 0000	Right shift 128

The instruction allows data types 0 through 6 for the source and destination fields.

The decimal scale rounded (E5) instruction rounds upward the absolute value of the right-shift result. This occurs by adding 5 to the last digit shifted end-off, and propagating carries through the decimal result.

These instructions may cause the arithmetic loss-of-significance exception condition.

BDP Byte Instructions

The instructions in this subgroup (table 1-13) compare, translate, move, edit, or scan byte fields in CM.

Table 1-13. BDP Byte Instructions

Opcode	Format	Instruction	Mnemonic
77	jk	Byte compare	CMPB
E9	jkiD	Byte compare, collated	CMPC
EB	jkiD	Byte translate	TRANB
76	jk	Move bytes	MOVB
ED	jkiD	Edit	EDIT
F3	jkiD	Byte scan while nonmember	SCNB

The following conditions apply to all byte instructions.

● Instruction specification error

● Address specification error

● Access violation

● Page table search without find

● Debug

Individual instruction descriptions list additional interrupt conditions where applicable. Refer to CP Interrupts in chapter 2 of this manual for descriptions of these conditions.

Byte Compare

Opcode 77jk

Mnemonic CMPB,Aj,X0 Ak,X1

Instruction Byte Compare, D(Aj) to D(Ak), result to X1R, index to X0R (2 descriptors)

Format

0	78	1112	15
77	j	k	

Opcode E9jkiD

Mnemonic CMPC,Aj,X0 Ak,X1,Ai,D

Instruction Byte Compare Collated, D(Aj) to D(Ak), both translated per (Ai plus D), result to X1R, index to X0R (2 descriptors)

Format

0	78	1112	1516	1920	31
E9	j	k	i	D	

Remarks These instructions compare the bytes in the source and destination fields, and set X1R according to the result. The comparison proceeds from left to right. When the field lengths are unequal, trailing space characters (20_{16}) are used for the shorter field. The maximum operand length is 256 bytes. Data types are ignored. The comparison continues until the longer field is exhausted or the instructions detect an inequality, as follows.

Compare (77) The byte comparison ends when the instruction detects an inequality between corresponding bytes in the source and destination field.

Compare Collated (E9) An inequality detected between corresponding bytes from the source and destination fields results in the translation of both bytes, using a translation table in CM. If the translated bytes are unequal, the comparison stops with the results shown in the following list. If the translated bytes are equal, the comparison continues until the longer field is exhausted, or until the instruction detects another inequality. In the latter case, another translation and comparison occurs.

The comparison results are indicated in X1R as follows.

Condition	Action Taken
D(Aj) = D(Ak)	X1R cleared
D(Aj) > D(Ak)	Clear X1R bits 32 and 34 through 63, set bit 33
D(Aj) < D(Ak)	Clear X1R bits 34 through 63, set bits 32 and 33

An unequal comparison places the sequence number of the byte causing the inequality into XOR. The instruction adds each field's leftmost byte address to the sequence number in XOR to determine the byte addresses within the source and destination fields causing the unequal comparison. Register XOR does not change if inequalities do not exist.

The user determines the translation table contents used by the compare collated instruction, and preloads the table into CM. The translation table contains 256 bytes. Its starting address forms by adding the BN field in Ai to the zero-extended D field from the instruction. Each translated byte adds as a positive offset to the translation table starting address, forming the address of the translated byte read from CM.

Byte Translate

Opcode   **EBjkiD**

Mnemonic   TRANB,Aj,X0 Ak,Xi,D

Instruction   Byte Translate, D(Ak) replaced by D(Aj), translated per (Ai plus D)
(2 descriptors)

Format

0	78	1112	1516	1920	31
EB	j	k	i	D	

Remarks   This instruction translates each source field byte according to a user-generated translation table in CM and transfers the results to the destination field. The source and destination field lengths are limited to 256 bytes. Data types are ignored.

The translation proceeds from left to right. The instruction uses each source field byte as a positive offset which it adds to the translation table address to locate the translated byte. Translated bytes transfer to the destination field. The translation terminates after the destination field length has been exhausted.

If the source field exceeds the destination field, the instruction truncates the rightmost bytes of the source field. When the source field is shorter than the destination field, the instruction fills the destination field rightmost byte positions with translated space characters.

The user determines the translation table contents and preloads the table into CM. This table contains 256 bytes; its starting address forms by adding the BN field in Ai to the zero-extended D field from the instruction.

Move Bytes

Opcode   76jk

Mnemonic   MOVB,Aj,X0 Ak,Xi

Instruction   Move Bytes, D(Ak) replaced by D(Aj) (2 descriptors)

Format

0	78	1112	15
76		j	k

Remarks   This instruction moves bytes from the source field to the destination field. The move operation is from left to right; data types are ignored. Maximum field lengths are 256 bytes. Unequal field lengths result in truncating trailing characters from the source field or inserting trailing space characters into the destination field.

Edit

Opcode   **EDjkiD**

Mnemonic   EDIT,Aj,X0 Ak,Xi,Ai,D

Instruction   Edit, D(Ak) replaced by D(Aj) edited per D(Ai+D) (1 descriptor)

Format

0	78	1112	1516	1920	31
ED		j	k	i	D

Remarks   Under control of a CM byte field called an edit mask, this instruction edits digits or characters from the source field and transfers the result to the destination field. It can perform the following editing functions.

● Move source field digits/characters to destination field

● Move characters from the edit mask to destination field

● Specify and insert a string of 0 through 15 characters (symbol) into the destination field

● Specify an 8-byte special character table (SCT) and insert any character from this table

● Insert suppression characters and floating signs to the left of the first significant digit

● Perform insertion of signs, suppression characters, blanks, symbols, or SCT characters based on whether the source field is positive or negative

● Spread suppression character through the destination field

● Write suppression characters if destination field is zero

The source data descriptor type fields are restricted to data types 0 through 9. The instruction ignores the destination data descriptor type fields.

Edit Mask

The edit mask consists of a length-indication byte followed by up to 254 microoperation (MOP) bytes. The length is a binary number indicating the number of bytes in the edit mask (including the length-indication byte). If the length-indicating byte is either zero or one, the associated edit instruction results in a no-operation. After the length indicator, the mask contains a string of 1-byte microinstructions.

The edit mask address is the sum of the BN field from Ai plus the zero-extended D field from the instruction. The edit mask format is as follows.

```
0          70   34   70   34   7
   +----------+----+----+----+----+
   | LENGTH   |MOP | SV |MOP | SV |
   +----------+----+----+----+----+
   First byte        Following bytes
```
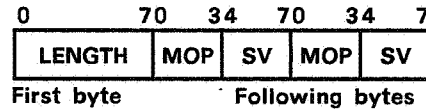
Field	Description
LENGTH	Binary number indicating the total number of bytes in the edit mask (0 to 255_{10})
MOP	Microoperation specifying the editing function
SV	Binary specification value (SV) from 0 through 15. Meaning varies according to the associated MOP

Edit Operation

The edit operation uses the tables and toggles described in the following paragraphs. Edit control proceeds from left to right on the mask, one character at a time. The instruction performs the editing function specified by the MOP and the specification value (SV).

Indexing through the source field is by bytes unless its data type is packed numeric. Packed-numeric data is indexed by half-bytes. Indexing through the destination field is by bytes.

Microoperation Description Nomenclature

The MOP descriptions use the following additional terms.

Term	Description
ES	End suppression (ES) toggle
SCT	Special characters table
SV	Specification value (refer to Edit Mask, preceding)
SM	Symbol
SN	Negative sign (NS) toggle
ZF	Zero field (ZF)

End Suppression Toggle –

The end suppression (ES) toggle controls zero suppression. Hardware sets the ES toggle false at the start of edit. The ES toggle sets true when zero suppression ends, when the first nonzero leading digit is encountered, or by a MOP.

Special Characters Table –

The eight-byte SCT is stored in hardware. Entries are written by the MDP code D. For proper editing, the SCT must be as follows.

Byte	0	1	2	3	4	5	6	7
Character	ƀ	ƀ	+	−	,	.	$	/
Hexadecimal	20	20	2B	2D	2C	2E	24	2F

- Negative sign (byte 3)
- Positive sign (byte 2)
- Suppression character (byte 1)
- Blank fill character (byte 0)

Symbol –

The symbol (SM) is a string of 0 through 15 characters that the edit instruction creates and inserts into the destination field, under edit mask control. Once the SM has been inserted, the instruction must recreate it before reinserting it. The SM has a length of zero when an edit operation begins. The system uses the SM for the floating-sign and floating-currency editing features, and for sign-sensitive and significance-sensitive character string insertion.

Negative Sign Toggle –

The SN toggle provides the source field sign. At start of edit, hardware sets the SN toggle false if the source field is an alphanumeric, an unsigned numeric, or a positive numeric. The SN toggle is initialized true only for a negative numeric source field.

Zero Field Toggle –

The zero field (ZF) toggle depicts a zero or nonzero source field. It is initialized true and sets false after encountering the first nonzero character.

Skipping of Signs

The edit instruction (under edit mask control) automatically skips signs when reading numeric data types. The signs interpret numerically when reading combined signed data types, also under edit mask control.

Microoperation 0

This MOP translates source field characters to ASCII and moves these to the destination field as follows. The translation performs as described in the Edit Function NUMERIC in this chapter.

1. Set ES true if SV is not equal to zero.

2. Translate SV digits from the source field to the equivalent ASCII characters and copy these into the destination field.

Microoperation 1

This MOP moves type 9 characters as follows.

1. Set ES true if SV is not equal to zero.

2. Move SV characters from the source field to the destination field. The source field must be type 9 or an invalid BDP data condition occurs.

Microoperation 2,3

These MOPs are no-operations.

Microoperation 4

This MOP moves the next edit mask SV bytes to the destination field.

Microoperation 5

This MOP sets the SM to a single character from SCT, respresenting the source data field sign as follows.

- Negative source data field. Copy SCT byte 3 to destination field.

- Positive source data field. Copy SCT byte SV into SM field. The SV rightmost three bits provide an index into the SCT.

Microoperation 6

This MOP moves the next edit mask SV bytes to the SM.

Microoperation 7

This MOP conditionally translates source field SV digits to their equivalent ASCII characters and copies them to the destination field. The translation performs as described in the Edit Function NUMERIC in this chapter.

- ES false and zero source field digit. Copy SCT byte 1 to destination field.

- ES false and nonzero source field digit. Set ES true and copy SM to destination field followed by the translated digit.

- ES true. Copy translated digit to destination field.

Microoperation 8

This MOP conditionally copies the SM to the destination field as follows.

- ES true. No operation.

- ES false. Copy SM to destination field and set ES true.

Microoperation 9

This MOP conditionally copies the SM or SCT character to the destination field as follows.

- SV > 7. Copy SM to destination field.

- SV ≤ 7. Copy SCT byte SV into destination field. The SV rightmost three bits provide an index into SCT.

Microoperation A

This MOP conditionally copies the SM or SCT character to the destination field as follows.

- SV > 7 and source field positive. Copy SM to destination field.

- SV > 7 and source field negative. Copy SCT byte 0 to destination field, once for each SM character.

- SV ≤ 7 and source field positive. Copy SCT byte SV into destination field. The SV rightmost three bits provide an index into SCT.

- SV ≤ 7 and source field negative. Copy SCT byte 0 into destination field.

Microoperation B

This MOP is identical to MOP A, but with the action caused by a reversal of the source field sign.

Microoperation C

This MOP conditionally copies the SM or SCT character to the destination field as follows.

- SV > 7 and ES true. Copy SM to destination field.

- SV > 7 and ES false. Copy SCT byte 1 character to destination field, once for each SM character.

- SV ≤ 7 and ES true. Copy SCT byte SV into destination field. The SV rightmost three bits provide an index into SCT.

- SV ≤ 7 and ES false. Copy SCT byte 1 into destination field.

Microoperation D

This MOP copies the next edit mask character into the SCT byte determined by using the SV rightmost three bits as an index into the SCT.

Microoperation E

This MOP copies SCT byte 1 into the destination field, SV times.

Microoperation F

This MOP conditionally copies the SCT character into the destination field as follows.

- No operation when SV = 0.

- ZF false and nonzero source field: terminate the edit instruction.

- ZF true and zero source field: reset to start of destination field and copy SCT byte 1 into destination field SV times.

Edit Function NUMERIC

Microoperations 0 and 7 translate and move a source digit into the destination field as follows.

- Each source digit is checked. Invalid decimal digits cause an Invalid BDP data condition. A program interrupt occurs when enabled.

- When the source field is packed-numeric, appropriate ASCII zone bits are supplied for the destination character.

- A nonzero digit causes the ZF toggle to be set false.

Termination of the Edit Instruction

The edit instruction terminates when the edit mask is exhausted, or when a MOP 15 is read and the ZF toggle is false. The CP detects no exception conditions for either condition, even though the instruction may not have exhausted the source or destination fields. If the instruction terminates with the destination field not full, the remaining portion of the destination field is not altered. If the source field is not exhausted when the instruction terminates, the source field is checked for invalid BDP data, and the sign is examined.

The edit instruction may cause the invalid BDP data exception condition.

Byte Scan While Nonmember

Opcode   F3jkiD

Mnemonic   SCNB,Aj,X0 Ak,Xi

Instruction   Byte Scan While Nonmember, D(Ak) for presence bit in (Ai+D), character
to X1R, index to X0R (1 descriptor)

Format

0	78	1112	1516	1920	31
F3	j	k	i	D	

Remarks   This instruction detects possible unwanted characters in a character string
by inspecting a 256-bit table in CM. The starting byte address of the table
forms by adding the BN field from Ai to the zero-extended D field from
the instruction.

The scan proceeds from left to right, one character at a time. The data
type is ignored. The binary value of each character addresses a bit in the
table. The scan terminates if this bit is a one or if the source field has
been exhausted.

If the scan terminates because the addressed bit is set, the following
occurs.

● The binary value of the sequence number (index) pointing to the byte
causing scan termination is placed right-justified into X0R.

● The binary value of the character causing scan termination is placed
right-justified into X1R.

If the scan terminates from exhaustion of characters in the byte string,
X0R contains the original byte string length, X1R bit 32 sets, and bits 33
through 63 clear.

This instruction can also perform the Byte Scan While Member function. In
this case, the table specifying the nonallowed byte string characters is
logically complemented before the instruction executes.

BDP Subscript and Immediate Data Instructions

The instructions in this subgroup are listed in table 1-14.

Table 1-14. BDP Subscript and Immediate Data Instructions

Opcode	Format	Instruction	Mnemonic
F4	jkiD	Calculate subscript and add	CALDF
F9	jkiD	Move immediate data	MOVI
FA	jkiD	Compare immediate data	CMPI
FB	jkiD	Add immediate data	ADDI

The following conditions apply to all subscript and immediate data instructions.

- Instruction specification error

- Address specification error

- Invalid SEG/RN zero

- Access violation

- Page table search without find

- Debug

- Invalid BDP data

Individual instruction descriptions list additional interrupt conditions where applicable. Refer to CP Interrupts in chapter 2 of this manual.

Calculate Subscript and Add

Opcode F4jkiD

Mnemonic CALDF,Aj,X0 Ak,Xi,Ai,D

Instruction Calculate Subscript and Add, D(Aj) checked and modified per (Ai plus D), result added to XkR (1 descriptor)

Format

0	78	1112	1516	1920	31
F4	j	k	i	D	

Remarks This instruction uses a subscript range table (SRT) contained in CM. The SRT contains one or more 64-bit entries with each entry divided into three binary integer values as follows.

Field	Description
SIZE	Sixteen bits, unsigned; specifies number of elements in one dimension of an array (table)
MIN	Sixteen bits, signed; specifies minimum allowable value of source field
MAX	Thirty-two bits, signed; specifies maximum allowable value of source field

This instruction forms the PVA of the subscript range table entry using: 1) ring number and SEG from Ai, and 2) the BN generated by adding the BN field from Ai to the instruction D field (expanded to 32 bits using zeros in the high-order bit positions). A signed, 32-bit twos-complement binary integer is obtained from the CM source field at location D(Aj). The instruction uses binary source field data unchanged and converts decimal data to its binary equivalent.

The occurrence number is the difference between the binary value of the source field's rightmost 32 bits and the MIN value (sign-extended to 32 bits). The occurrence number is a signed, 32-bit twos-complement integer.

D(Aj) - MIN = OCCURRENCE NUMBER

To calculate the subscript, the instruction multiplies the OCCURRENCE NUMBER by SIZE and adds the product to the index value in the destination register XkR. The CP does not detect overflow during any arithmetic operation associated with this instruction.

The source field is restricted to data types 0 through 6, 10, and 11, with the maximum field lengths determined by the source field data type.

Move Immediate Data

Opcode	**F9jkiD**
Mnemonic	MOVI,Xi,D Ak,Xi,j
Instruction	Move Immediate Data, D(Ak) replaced by XiR plus D per j (1 descriptor)

Format

0	78	1112	1516	1920	31
F9	j	k	i	D	

Remarks   The immediate data byte is the twos-complement sum of XiR bits 56 through 63, plus the rightmost eight bits of the instruction D field. Overflow is ignored on this summation. X0 consists of all zeros.

This instruction moves the immediate data to the destination field after format conversion specified by the destination field data type and the j-field suboperation code. The conversion is encoded in the least significant 2 bits (bits 10 and 11) of the instruction's j field as follows.

j Field Bits 10,11	Operation
00	The positive, unsigned numeric value (type 10) in the immediate data byte moves right-justified to the destination field. The destination field is restricted to data types 10 or 11.
01	The decimal numeric (type 4) immediate data byte moves right-justified to the destination field after reformatting (if necessary). A positive sign is supplied as required. The destination field is restricted to decimal data types 0 through 6.
10	The ASCII character in the immediate data byte repeats left-to-right in the destination field. Destination data type is ignored.
11	The ASCII character in the immediate data byte moves, left-justified, into the destination field; the remainder of the field fills with space characters. The destination data type is ignored.

The slack digit of destination field types 1 and 3 is unchanged by this instruction. The instruction may cause the arithmetic loss-of-significance exception condition.

Compare Immediate Data

Opcode   FAjkiD

Mnemonic   CMPI,Xi,D Ak,Xi,j

Instruction   Compare Immediate Data, XiR plus D to D(Ak) per j, result to XiR
(1 descriptor)

Format

0	78	1112	1516	1920	31
FA	j	k	i	D	

Remarks   The immediate data byte is the twos-complement sum of XiR bits 56
through 63, plus the rightmost eight bits of the instruction D field.
Overflow is ignored on this summation. X0 consists of all zeros.

This instruction performs a format conversion on the immediate data byte
as specified by destination field data type and the j field suboperation code.
The instruction then compares the reformatted immediate data byte to the
contents of D(Ak). The instruction j field encodes the operation as follows.

j Field Bits 10,11	Operation
00	The positive, unsigned numeric value (type 10) in the immediate data byte compares to the contents of D(Ak). The destination field is restricted to data types 10 or 11. If field D(Ak) exceeds one byte, the immediate data byte zero-fills in its high-order positions.
01	The decimal numeric (type 4) immediate data byte compares to the contents of D(Ak) after reformatting (if necessary) to match the data type of field D(Ak). A positive sign is supplied as required. The D(Ak) field is restricted to decimal data types 0 through 6. If D(Ak) exceeds one byte, the immediate data byte zero-fills in its high-order positions.
10	The ASCII character in the immediate data byte compares left-to-right to the D(Ak) field. Then (DAk) field data type is ignored.
11	The ASCII character in the immediate data byte compares to the leftmost byte in field D(Ak). If the comparison is equal and field D(Ak) exceeds one byte, a space character compares left-to-right with each successive byte remaining in the D(Ak) field. The D(Ak) field data type is ignored.

A half word transfers to X1R to indicate the comparison result as follows.

Results of Compare	Action Taken
Source = Destination	Clear X1R
Source > Destination	Clear bits 32 and 34 through 63, set bit 33
Source < Destination	Clear bits 34 through 63, set bits 32 and 33

Add Immediate Data

Opcode	**FBjkiD**
Mnemonic	ADDI,X1,D Ak,Xi,j
Instruction	Add Immediate Data, D(Ak) replaced by D(Ak) plus XiR plus D per j (1 descriptor)

Format

```
0        78  1112 1516 1920            31
|   FB    |  j |  k |  i |      D        |
```

Remarks   The add immediate instruction converts the source field immediate data to match the destination field data type (if required) and adds the immediate data byte to D(Ak). The immediate data byte stores the integer value of the addend. The instruction j field encodes the data type contained in the immediate data byte.

The j field least significant bit (bit 11) decodes as follows.

j Field Bit 11	Data Type Immediate Data Byte
0	Data type = 10; unsigned (positive) binary integer value
1	Data type = 4; one ASCII character representing a decimal digit

If the source field is data type 10, the destination field is restricted to data type 10 or 11.

If source data is type 4, the destination is restricted to types 0 through 6.

This instruction may cause the arithmetic overflow exception condition.

FP Instruction Descriptions

Refer to FP Programming in chapter 2 of this manual for descriptions of floating point (FP) data formats, standard and nonstandard numbers, and normalization. The FP instructions consist of 18 opcodes in four subgroups.

● Conversion

● Arithmetic

● Branch

● Compare

Tables 1-15 through 1-17 list the instructions in the first three subgroups.

Double-Precision Register Designators

The double-precision FP add, subtract, multiply, and divide instructions operate on double-length registers, designated as follows.

XXk or XXj   Two successive registers Xk, X(k+1) or Xj, X(j+1) containing a double-precision FP number. Xk or Xj contains the high order (leftmost) part of this number.

FP Conversion Instructions

The instructions in this subgroup (table 1-15) convert 64-bit words between FP and integer formats.

Table 1-15. FP Conversion Instructions

Opcode	Format	Instruction	Mnemonic
3A	jk	Convert from integer to FP	CNIF
3B	jk	Convert from FP to integer	CNFI

Convert From Integer to FP

Opcode   3Ajk

Mnemonic   CNIF Xk,Xj

Instruction Convert, FP Xk formed from integer Xj

Format
```
0        78  1112 15
┌──────────┬────┬────┐
│    3A    │  j │  k │
└──────────┴────┴────┘
```

Remarks   This instruction converts the signed 64-bit twos-complement binary integer from Xj to its normalized FP representation and transfers the 64-bit result to Xk.

During conversion, the instruction truncates the rightmost bits of integers outside the range -2^{48} through $(2^{48})-1$. When Xj is all zeros, it transfers unchanged to Xk.

Convert From FP to Integer

Opcode   3Bjk

Mnemonic   CNFI Xk,Xj

Instruction Convert, integer Xk formed from FP Xj

Format
```
0        78  1112 15
┌──────────┬────┬────┐
│    3B    │  j │  k │
└──────────┴────┴────┘
```

Remarks   This instruction converts the 64-bit FP number in Xj to a signed twos-complement binary integer and transfers the result to Xk. The fractional part of the binary equivalent truncates. This conversion results in an integer consisting of all zeros if the FP number satisfies any of the following.

● Is indefinite

● Has an exponent equal to zero

● Has a fraction equal to zero

● Is infinite

This instruction may cause the arithmetic loss-of-significance, FP indefinite, and FP infinite exception conditions.

FP Arithmetic Instructions

The instructions in this subgroup (table 1-16) perform arithmetic operations on FP numbers.

Table 1-16. FP Arithmetic Instructions

Opcode	Format	Instruction	Mnemonic
30	jk	FP sum	ADDF
31	jk	FP difference	SUBF .
34	jk	Double-precision FP sum	ADDD
35	jk	Double-precision FP difference	SUBD '
32	jk	FP product	MULF
36	jk	Double-precision FP product	MULD
33	jk	FP quotient	DIVF
37	jk	Double-precision FP quotient	DIVD

The following conditions apply to all FP arithmetic instructions.

● Exponent overflow

● Exponent underflow

● FP loss-of-significance

● FP indefinite

Individual instruction descriptions list additional interrupt conditions where applicable. Refer to CP Interrupts in chapter 2 of this manual.

FP Sum/Difference

Opcode   **30jk**

Mnemonic   ADDF Xk,Xj

Instruction   FP Sum, Xk replaced by Xk plus Xj

Opcode   **31jk**

Mnemonic   SUBF Xk,Xj

Instruction   FP Difference, Xk replaced by Xk minus Xj

Opcode   **34jk**

Mnemonic   ADDD Xk,Xj

Instruction   Double-Precision FP Sum, XXk replaced by XXk plus XXj

Opcode   **35jk**

Mnemonic   SUBD Xk,Xj

Instruction   Double-Precision FP Difference, XXk replaced by XXk minus XXj

Format

0	78	1112	15
30,31,34,35	j	k	

Remarks   The following instruction description applies to either single- or double-precision operations. References to Xk or Xj in the description also apply to XXk or XXj for the double-precision instructions.

These instructions algebraically compare the exponents of the two FP operands in Xk and Xj. If the exponents are equal, no adjustment is necessary. If the exponents are unequal, the instruction aligns the coefficients by right-shifting the coefficient with the smaller exponent the number of bit positions designated by the difference between the exponents. The maximum shift is 48 positions for single-precision instructions or 96 positions for double-precision instructions.

The two aligned coefficients consist of a signed 48-bit single-precision or 96-bit double-precision fraction. The instructions add or subtract the two coefficients as determined by the opcode, using the Xj coefficient as the addend or subtrahend. The algebraic result is a signed coefficient with 48 bits (single precision) or 96 bits (double precision), plus an overflow bit. The overflow bit provides the required allowance for true addition (FP sum of coefficients with like signs or FP difference of coefficients with unlike signs).

If coefficient overflow occurs (overflow bit = 1), the instruction right-shifts the coefficient one place, inserts the overflow bit in the high order bit position (bit 16), increases the exponent by one, and places the result in Xk. If the coefficient overflow bit is zero and the coefficient is not all zeros, the instructions normalize the result before placing the result in Xk.

If either or both of the input operands in Xk and Xj consists of an infinite or indefinite FP number, the result transferred to Xk is a nonstandard FP number. Refer to Standard and Nonstandard FP Numbers in chapter 2 of this manual.

FP Product

Opcode	32jk
Mnemonic	MULF Xk,Xj
Instruction	FP Product, Xk replaced by Xk times Xj
Opcode	36jk
Mnemonic	MULD Xk,Xj
Instruction	Double-Precision FP Product, XXk replaced by XXk times XXj

Format

```
0           78  1112 15
+-----------+----+----+
|   32,36   | j  | k  |
+-----------+----+----+
```

Remarks   The following instruction description applies to either single- or double-precision operations. References to Xk or Xj in the description also apply to XXk and XXj for the double-precision instructions.

The multiply FP instructions algebraically add the signed exponents for the two FP operands in Xk and Xj, using the result as an intermediate exponent. The instructions multiply the coefficient in Xk by the coefficient in Xj to produce an algebraically-signed product consisting of 96 bits (single precision) or 192 bits (double precision). If the products high-order bit (bit 16) is a one, the product is already normalized and the high-order 48 bits (single precision) or 96 bits (double precision) become an intermediate coefficient.

If the high-order bit is a zero, the instructions left-shift the 96-bit or 192-bit product one bit position, decrease the intermediate exponent by one, and use the high-order 48 bits (single precision) or 96 bits (double precision) as the intermediate coefficient. This one-position shift results in a normalized product if both input operands were normalized before executing the multiply instruction. If the intermediate exponent (including the adjustment for normalization) is not equal to an out-of-range value, the intermediate exponent and the intermediate coefficient (with its sign) transfer to Xk to form the final result.

If one or both of the input operands in Xk and Xj consist of an infinite, indefinite, or zero FP number, the result transferred to Xk is a nonstandard FP number. Refer to Standard and Nonstandard FP Numbers in chapter 2 of this manual.

FP Quotient

Opcode   33jk

Mnemonic   DIVF Xk,Xj

Instruction   FP Quotient, Xk replaced by Xk divided by Xj

Opcode   37jk

Mnemonic   DIVD Xk,Xj

Instruction   Double-Precision FP Quotient, XXk replaced by XXk divided by XXj

Format

0	78	1112	15
33,37		j	k

Remarks   The following instruction description applies to either single- or double-precision operations. References to Xk or Xj in the description also apply to XXk or XXj for the double-precision instructions.

The divide FP instructions subtract the Xj exponent (divisor) from the Xk exponent (dividend) and use the signed result as an intermediate exponent.

These instructions divide the Xk signed coefficient by the Xj signed coefficient. If the Xj coefficient is unnormalized before instruction execution, and can be divided into the Xk coefficient by a factor exceeding or equal to two, the CP detects a divide fault.

If the CP does not detect errors, the division results in an algebraically-signed quotient with 48 bits (single precision) or 96 bits (double precision), plus an overflow bit. The overflow bit allows for cases in which the divisor can be divided into the dividend by a factor equal to or exceeding one, but less than two. If the overflow bit is a zero, the sign bit and 48- or 96-bit quotient require no further adjustments. If the overflow bit is a one, the instruction right-shifts the quotient one position, end-off, with the overflow bit inserted into the high-order bit position, and the exponent increased by one. The intermediate exponent and intermediate coefficient (with its sign) transfer to Xk to form the final result. When one or both of the input operands in Xk and Xj consist of an infinite, indefinite, or zero FP number, the result transferred to Xk is a nonstandard FP number. Refer to Standard and Nonstandard FP Numbers in chapter 2 of this manual.

This instruction may cause a divide fault exception condition.

FP Branch Instructions

This subgroup (table 1-17) consists of five conditional branch instructions. Each instruction compares two FP numbers and performs either a normal or branch exit based on the comparison results.

Table 1-17. FP Branch Instructions

Opcode	Format	Instruction	Mnemonic
98	jkQ	FP branch on equal	BRFEQ
99	jkQ	FP branch on not equal	BRFNE
9A	jkQ	FP branch on greater than	BRFGT
9B	jkQ	FP branch on greater than or equal to	BRFGE
9E	jkQ	FP branch on overflow	BROVR
9E	jkQ	FP branch on underflow	BRUND
9E	jkQ	FP branch on indefinite	BRINF
3C	jk	FP compare	CMPF

Normal Exit

The instruction takes a normal exit if the branch condition is not satisfied. The next instruction address forms by adding four to the BN field of the current PVA in P.

Branch Exit

The instruction takes a branch exit if the branch condition is satisified. The next instruction address forms by adding two times the Q field value (from the branch instruction) to the BN field of the current PVA in P.

Group Interrupt Conditions

The following interrupt conditions apply to all FP branch instructions.

● Debug

● FP loss-of-significance

● FP indefinite

Refer to CP Interrupts in chapter 2 of this manual for descriptions of these conditions.

FP Branch on Comparison

Opcode	**98jkQ**
Mnemonic	BRFEQ Xj,Xk,Q
Instruction	Branch to P displaced by 2*Q, if FP Xj equal to Xk
Opcode	**99jkQ**
Mnemonic	BRFNE Xj,Xk,Q
Instruction	Branch to P displaced by 2*Q, if FP Xj not equal to Xk
Opcode	**9AjkQ**
Mnemonic	BRFGT Xj,Xk,Q
Instruction	Branch to P displaced by 2*Q, if FP Xj greater than Xk
Opcode	**9BjkQ**
Mnemonic	BRFGE Xj,Xk,Q
Instruction	Branch to P displaced by 2*Q, if FP Xj greater than or equal to Xk

Format

0   78 1112 1516   31

98,99,9A,9B	j	k	Q

Remarks   Each compare and branch instruction performs an algebraic comparison between the 64-bit words in Xj and Xk. If the branch conditions are satisfied, the instruction takes a branch exit. If the conditions are not satisified, a normal exit results.

These instructions treat the 64-bit words in Xj and Xk as single-precision FP numbers. If Xj or Xk specifies register X0, these instructions interpret X0 as all zeros.

For the results with the various combinations of comparison input data, refer to Standard and Nonstandard FP Numbers in chapter 2 of this manual.

FP Branch on Condition

Opcode   9EjkQ

Mnemonic

j Field	Xk Tested For	
00	Exponent overflow	BROVR Xk,Q
01	Exponent underflow	BRUND Xk,Q
10/11	Exponent indefinite	BRINF Xk,Q

Instruction   Branch to P displaced by 2*Q, if FP Xk is exception per j

Format

0	78	1112	1516	31
9E	j	k	Q	

Remarks   The instruction takes a branch exit if the exception condition designated by bits 10 and 11 of the instruction j field applies to the 64-bit FP number in Xk. A normal exit occurs if the exception condition designated by j field bits 10 and 11 does not apply to the 64-bit FP number in Xk.

FP Compare

Opcode   3Cjk

Mnemonic   CMPF Xi,Xj,Xk

Instruction   Compare FP Xj to Xk, result to X1R

Format

0	78	1112	15
3C		j	k

Remarks   This instruction algebraically compares the Xj 64-bit word to the Xk 64-bit word and indicates the result by setting bits in X1R. The instruction treats the 64-bit words in Xj and Xk as single-precision FP numbers.

If Xj or Xk specifies register X0, the instruction interprets X0 as all zeros. X1R bits are set as follows.

Results of Compare	Action Taken
Xj = Xk	Clear X1R
Xj > Xk	Clear bits 32 and 34 through 63, set bit 33
Xj < Xk	Clear bits 34 through 63, set bits 32 and 33
Indefinite	Clear bits 33 through 63, set bit 32

If the comparison results are indefinite, the CP records an FP indefinite condition and sets register X1R as shown in the preceding table. If the corresponding user mask bit is set and the trap enabled, the corresponding program interrupt occurs.

System Instruction Descriptions

The system instructions consist of 27 opcodes in six classes. The classes are based on the characteristics of the code segment from which the instructions are accessed, or the CP mode in which the instructions may operate. The classes are as follows.

● Nonprivileged

● Local privileged

● Global privileged

● Virtual State

● Virtual State monitor mode

● Mixed mode

Local and global privileged instructions execute only when the executive privilege (XP) field of the associated segment descriptor designates the appropriate privilege (with the CP in any mode). Virtual State monitor mode instructions execute only when the CP is in Virtual State monitor mode. Mixed mode instruction parameters within the instruction determine their privilege/mode requirements. Refer to Access Protection in chapter 2 of this manual for more information.

Nonprivileged System Instructions

The instructions in this subgroup are listed in table 1-18. In some cases, a portion of the instruction word is unused, as indicated in the instruction format. Instruction execution is not affected by these unused bits, but it is recommended these bits be zeros.

Table 1-18. Nonprivileged Instructions

Opcode	Format	Instruction	Mnemonic
00	jk	Program error	HALT
01	jk	Scope loop synchronization	SYNC
02	jk	Exchange	EXCHANGE
04	jk	Return	RETURN
06	jk	Pop	POP
08	jk	Copy free-running counter	CPYTX
14	jk	Test and set bit	LBSET
16	jk	Test and set page	TPAGE
B0	jkQ	Call relative	CALLREL
B4	jkQ	Compare swap	CMPXA
B5	jkQ	Call indirect	CALLSEG
BE,BF	jkQ	Reserved opcodes	–
CS	jkiD	Execute algorithm	EXECUTE,S

Program Error

Opcode   00jk

Mnemonic   HALT

Instruction   Program Error

Format
```
0          78          15
┌──────────┬──────────────┐
│   00     │░░░░░░░░░░░░░░│
└──────────┴──────────────┘
```

Remarks   This instruction causes an instruction specification error with the corresponding program interrupt or halt.

Scope Loop Sync

Opcode   01jk

Mnemonic   SYNC

Instruction Scope Loop Sync

Format

```
0          78      15
 ┌──────────┬──────────┐
 │    01    │░░░░░░░░░░│
 └──────────┴──────────┘
```

Remarks   Set CP breakpoint register 32 to 101_{16} by performing the Common Maintenance Software Executive (CMSE) command ER2,32 = 101_{16}. The instruction triggers at TP 14 at location CPG.

This instruction is a no-operation within the CP. The instruction generates a pulse to a test point for oscilloscope synchronization.

Exchange

Opcode	**02jk**
Mnemonic	EXCHANGE
Instruction	Exchange
Format	

```
0        78      15
┌────────┬────────┐
│   02   │////////│
└────────┴────────┘
```

Remarks This instruction exchanges the current process registers (formatted as an exchange package) with another set stored in CM, and does the following.

- When executed with CP in Virtual State monitor mode, the processor switches from monitor to job mode.

- When executed in Virtual State job mode, the processor switches from job to monitor mode; and the system call bit sets in MCR10.

In either case, the P register stored in the outgoing exchange package points to the next instruction that would have executed if the exchange had not occurred.

This instruction can cause the following exception conditions.

- Environment specification error

- System call

Refer to CP Modes of Operation in chapter 2 of this manual for further information.

Return

Opcode	04jk
Mnemonic	RETURN
Instruction	Return

Format

```
0       78      15
 ┌──────┬────────┐
 │  04  │////////│
 └──────┴────────┘
```

Remarks   This instruction requires the following register assignments.

Register	Description
(A0)	Dynamic space pointer (DSP)
(A1)	Current stack frame (CSF) pointer
(A2)	Previous save area (PSA) pointer
In exchange package	Top of stack (TOS) pointer for current ring of execution.
In exchange package	TOS pointer for previous ring of execution.

This instruction reestablishes the stack frame and environment of the previous procedure (which must be executing in an equal or less privileged ring as the current procedure). This operation does not load MCR or UCR. The instruction executes as follows.

1. Update the TOS pointer by storing the CSF pointer from A1 into the TOS pointer for the current ring of execution. This has the effect of cancelling the CSF.

2. Load the environment from the PSA (as defined by PSA pointer in A2 and the stack frame descriptor in PSA) as follows.

   ● P register (all fields)

   ● Virtual machine identifier (VMID) (CP state switch may take place)

   ● Critical frame flag and on-condition flag (CFF and OCF)

   ● User mask register

   ● A0 through At (per SFSA descriptor)

   ● Xs through Xt (per SFSA descriptor)

3. Set the ring number field of each A register loaded from SFSA equal to the largest of the following.

   ● A(RN) from SFSA

   ● Initial A2(RN)

   ● R1 of segment descriptor table entries (SDE) for initial A2

4. If the final P(RN) does not equal the initial P(RN), set any A(RN) not loaded from PSA in step 2 (and less than the final P(RN)) equal to the final P(RN).

5. Update TOS pointer in the exchange package.

6. Clear trap enable delay.

7. If any A(RN) loaded from PSA in step 2 is zero, set MCR 60 with interrupt or halt. When this happens, the instruction execution completes and the untranslatable pointer (UTP) is unaltered.
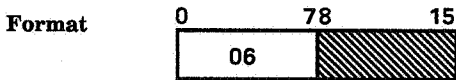
This instruction can cause the following exception conditions.

● Address specification error

● Invalid SEG/RN zero

● Access violation

● Environment specification error

● Page table search without find

● Outward call/inward return

● CFF

● Debug

Pop

Opcode	**06jk**
Mnemonic	POP
Instruction	Pop

Format

```
0        78        15
 ┌──────┬──────────┐
 │  06  │▓▓▓▓▓▓▓▓▓▓│
 └──────┴──────────┘
```

Remarks   This instruction requires the following register assignments.

Register	Description
(A0)	DSP
(A1)	CSF pointer
(A2)	PSA pointer
In exchange package	TOS pointer for current ring of execution

This instruction moves the CSF, PSA, and TOS pointers to eliminate the stack frame without changing the P-counter. This instruction reestablishes the stack frame (but not the environment) of the previous procedure, which must be in the same ring of execution as the current procedure. The stack frame is reestablished as follows.

1. Obtain the stack frame descriptor from the PSA (SFSA for the previous procedure) using the PSA pointer in A2.

2. Update the CSF pointer by loading A1 with word 2 from the PSA. Set A1 ring number equal to P ring number.

3. Update the PSA pointer by loading A2 with word 3 from the PSA. Set A2 RN equal to the largest of: 1) the A2 initial RN, 2) the A2 RN from PSA, or 3) the R1 field of the segment descriptor associated with the PSA.

4. Load the CFF and the OCF from the PSA.

5. Update the TOS pointer by storing the CSF pointer from final A1 into the TOS pointer for the current ring of execution. This has the effect of cancelling the CSF.

6. If any A1(RN) or A2(RN) loaded from PSA in step 2 is zero, set MCR 60 with interrupt or halt. Instruction execution completes and UTP is unaltered.

This instruction may cause the following exception conditions.

- Address specification error

- Invalid SEG/RN zero

- Access violation

- Environment specification error

- Page table search without find

- Inter-ring pop

- CFF

- Debug

Copy Free-Running Counter

Opcode	08jk
Mnemonic	CPYTX Xk,Xj
Instruction	Copy Free-Running Counter to Xk at XjR

Format

0	78	1112	15
08	j	k	

Remarks This instruction copies the free-running counter in CM into Xk (the free-running counter consists of either 64 bits of counter or 48 bits of counter which are right-justified with zero-fill in the leftmost 16 bits). XjR bits 32 and 34 through 63 are zeros. XjR bit 33 specifies which processor port the instruction uses to read the counter as follows.

Bit 33	Port Selected
0	Local processor port to CM
1	External processor port to CM of another system

Test and Set Bit

Opcode   14jk

Mnemonic   LBSET Xk,Aj,X0

Instruction Load Bit to XkR from Aj Bit indexed by X0R and set Bit in CM

Format

0	78	1112	15
14	j	k	

Remarks   This instruction transfers one bit from CM into XkR bit position 63 and clears Xk bits 0 through 62. The instruction also sets that bit in CM without changing any other bits in CM.

The instruction addresses the CM byte containing the bit by adding bits 32 through 60 of X0R (right-shifted three positions, end-off, with sign extension on the left), to bits 32 through 63 of Aj. The instruction uses X0R bits 61 through 63 to locate the bit position within the addressed byte. Values 0 through 7 for these three bits select corresponding bits 0 through 7 from the addressed byte.

No other CM accesses (from any port) to the CM byte containing that bit are permitted from the start of the read access until the end of the write access (when the instruction sets the bit in CM).

The system performs a serialization function before and after instruction execution. The CP delays instruction execution until all previous CM accesses by previous instructions complete, and delays execution of the next instruction until all CM accesses from this instruction complete.

To establish operand access validity, the instruction uses read- and write-type CM accesses. The read access bypasses cache memory. Termination of the write access purges the associated cache entry.

This instruction may cause the following exception conditions.

● Address specification error

● Invalid SEG/RN zero

● Access violation

● Page table search without find

● Debug

Test and Set Page

Opcode   16jk

Mnemonic   TPAGE Xk,Aj

Instruction   Test Page (Aj) and set XkR

Format

0	78	1112	15
16	j	k	

Remarks   This instruction tests CM for the presence of the page (corresponding to the PVA in Aj) in the system page table with its valid bit set in the associated page descriptor. If the tested page is in CM, the used bit in the associated page descriptor sets, and the real memory address translated from the PVA from Aj transfers to XkR. If the tested page is not in CM, the instruction sets XkR bit 32 and clears XkR bits 33 through 63.

This instruction may cause the following exception conditions.

● Address specification error

● Invalid SEG/RN zero

Call Relative

Opcode   B0jkQ

Mnemonic   CALLREL Aj,Ak,Q

Instruction   Call to P displaced by 8*Q, binding section pointer per Aj, arguments per Ak

Format

0	78	1112	1516	31
B0	j	k	Q	

Remarks   Register assignments are as follows.

Register	Description
(A0)	DSP
(A1)	CSF pointer
(A2)	PSA pointer
(A4)	Argument pointer
In exchange package	TOS pointer per RN in P

This instruction saves the current procedure (caller) environment and calls another procedure (callee) within the same segment as the caller. The RN and SEG fields of P remain unaltered.

The caller's environment is saved by storing designated process and processor registers into a stack frame save area (SFSA) generated on top of the CSF. The DSP in A0, rounded to the next available full-word address, is the PVA of the first word in this SFSA. The instruction saves some CP registers in the SFSA unconditionally. These registers are as follows.

- P register

- Stack frame descriptor

- User mask

- VMID

- Register A0

The caller specifies other registers saved. A0 is always the first register saved, and X0R specifies other A and X registers to be saved. X0R has the following format.

```
     52 55 56 59 60  63
    ┌────┬────┬────┬────┐
    │    │    │    │    │
    └────┴────┴────┴────┘
                    └────X-last
                └────────A-last
            └────────────X-first
```
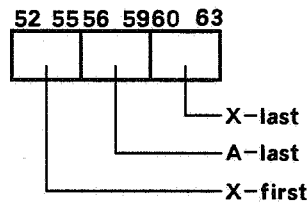
X0 Bits	Registers Saved
52-55	First X register
56-59	Last A register
60-63	Last X register

The call instruction does not store any X register if the value of X-last exceeds X-first.

After storing the registers in the SFSA, the instruction executes as follows.

1. Modify DSP in A0 by adding eight times the number of SFSA words to the BN in A0.

2. Update the TOS pointer in the exchange package by storing the modified DSP into the exchange package TOS entry corresponding to the current ring of execution, as determined by the RN in P. This creates a new stack frame.

3. Form the target address by adding eight times Q to the BN in P. Bits 61 through 63 of P are forced to zero.

4. Establish the stack frame of the callee by loading A0, A1, and A2 from the PSA (SFSA of the callee).

5. Copy Aj to A3 and Ak to A4 to reflect parameter changes required to transfer control to the callee.

This instruction may cause the following exception conditions.

- Instruction specification

- Address specification error

- Invalid SEG/RN zero

- Access violation

- Page table search without find

- Debug

Compare Swap

Opcode B4jkQ

Mnemonic CMPXA Xk,Aj,X0,Q

Instruction Compare Xk to (Aj), if locked, branch to P displaced by 2*Q, if unlocked, load/store (Aj), result to X1R

Format

0	78	1112	1516	31
B4	j	k	Q	

Remarks If the leftmost 32 bits of a 64-bit word in CM location Aj are all ones (Aj locked), the instruction takes a branch exit. The target address forms by adding the value 2 times Q (sign-extended) to the BN field of the PVA in P.

If the above condition is absent, the instruction compares the Xk 64-bit word with the word in CM location Aj (64-bit integer compare). If the two words are equal, the instruction stores X0 in location Aj and clears X1R. If the two words are unequal, the instruction loads the word in CM location Aj into Xk and sets X1R as follows (in either case, the instruction takes a normal exit).

Results of Compare	**Action Taken**
Xk = (Aj)	Store X0 at (Aj), clear X1R
Xk > (Aj)	Load (Aj) into Xk, clear X1R bits 32 and 34 through 63, set X1R bit 33
Xk < (Aj)	Load (Aj) into Xk, clear X1R bits 34 through 63, set X1R bits 32 and 33

Within a given CP, execution of this instruction delays until all previous CM accesses complete. Execution of all subsequent instructions delays until all CM accesses due to this instruction complete. In dual-CP systems, if a second CP executes a compare swap instruction while the other CP is processing one, the second CP reads the 64-bit word in location Aj, finds the leftmost 32 bits all ones (locked), and branch-exits. The hardware, however, does not inhibit other instruction codes issued from the other CP (or any PP instructions) from accessing and altering location Aj.

The read access bypasses cache and the write access purges the associated cache entry.

For the debug scan, Aj is both a read and a write address; whereas P+2Q is a branch target address only when the branch occurs.

This instruction may cause the following exception conditions.

- Instruction specification

- Address specification error

- Invalid SEG/RN zero

- Access violation

- Page table search without find

- Debug

Call Indirect

Opcode   **B5jkQ**

Mnemonic   CALLSEG Aj,Ak,Q

Instruction Call per (Aj displaced by 8*Q), arguments per Ak

Format

0	78	1112	1516	31
B5	j	k	Q	

Remarks   The instruction uses the following assigned registers.

Register	Description
(A0)	DSP
(A1)	CSF pointer
(A2)	PSA pointer
(A3)	Binding section pointer
(A4)	Argument pointer
In exchange package	TOS pointer for the caller's ring of execution
In exchange package	TOS pointer for the callee's ring of execution
In CM	Code base pointer (CBP) addressed by A3+8*Q

This instruction saves the current procedure (caller) environment and calls another procedure (callee) indirectly. The callee must be executing within the same or in a higher privileged ring as the caller. The indirect target address is listed in the CBP addressed by (Aj displaced by eight times sign-extended Q). The instruction saves the environment (as specified by X0R) in the SFSA generated on top of the CSF. For details, refer to the call relative instruction described in this chapter.

The instruction executes as follows.

1. Add eight times Q to the BN field from register Aj to form the PVA of a CBP from a binding section segment (which contains the target PVA).

2. Round DSP upward as follows: Add seven to A0, then force A0 bits 61 through 63 to zero.

3. Store environment in SFSA, per X0R.

4. Copy P bits 0 through 31 to X0R (caller's ID).

5. Modify DSP in A0 by adding eight times the number of SFSA words to A0 bits 32 through 63.

6. Adjust TOS pointer in the exchange package by storing this modified DSP in the TOS entry for the current ring of execution, as determined by the RN field in P.

7. Load P key with segment descriptor lock for callee.

8. If P ring number is less than callee segment descriptor R2 (inter-ring call), go to step 12.

9. Set P ring number equal to callee segment descriptor R2.

10. Load P SEG and BN fields with CBP SEG and BN fields.

11. If CBP VMID = 1 (call is to CYBER 170 State), go to step 17.

12. If internal procedure (CBP EPF = 0), go to step 16.

13. Load A3 with new binding chapter pointer, setting RN equal to the larger of the RN in CBP and the new RN in P register.

14. If trap operation, go to step 17; if call instruction, copy Ak to A4 (pass parameters). When k is 0 through 3, the final contents of A4 is with respect to which A register is copied.

15. Copy (A0) to A2 (DSP from step 2 to PSA pointer).

16. Clear OCF.

17. Load A1 with TOS pointer from exchange package per final P ring number, and clear the CFF.

18. Set DSP in A0 equal to CSF pointer in A1.

19. Copy VMID from CBP to the VMID register.

NOTE

The trap interrupt operation unconditionally includes all the above steps except steps 10, 11, and 14.

This instruction may cause the following exception conditions.

● Instruction specification

● Address specification error

● Invalid SEG/RN number zero

● Access violation

● Environment specification error

● Page table search without find

● Outward call/inward return

● Debug

Reserved Operation Codes

Opcode	**BEjkQ**
Mnemonic	(None)
Instruction	Reserved for user
Opcode	**BFjkQ**
Mnemonic	(None)
Instruction	Reserved for user

Format

```
0          78   1112 1516                  31
  | BE,BF  |  j  |  k  |        Q            |
```

Remarks
These instructions are reserved for the user for software simulation of operations that executive state does not provide through trap interrupts. These opcodes will not be used in future hardware extensions.

When not implemented, these instructions cause the unimplemented instruction exception, with interrupt or halt.

Execute Algorithm

Opcode	**CSjkiD**
Mnemonic	EXECUTE,S j,k,i,D
Instruction	Execute Algorithm per S
Remarks	This instruction is reserved for future expansion.

When not implemented, this instruction causes the unimplemented instruction exception, with interrupt or halt.

Local Privileged System Instruction

Instructions in this subgroup (table 1-19) can execute only from segments which (by the associated segment descriptor) have either local or global privilege. If a local privileged instruction is fetched from a segment without either local or global privilege, the CP detects a privileged instruction fault, inhibits execution, and initiates the corresponding program interrupt or halt.

Table 1-19. Local Privileged Instruction

Opcode	Format	Instruction	Mnemonic
17	jk	Load page table index	LPAGE

Load Page Table Index

Opcode 17jk

Mnemonic LPAGE Xk,Xj,Xi

Instruction Load Page Table Index per Xj to XkR and set X1R

Format

```
0        78  1112 15
 ┌────────┬───┬───┐
 │   17   │ j │ k │
 └────────┴───┴───┘
```

Remarks This instruction searches the page table in CM for the presence of a page, returns the final search index value to XkR, and sets X1R to indicate the search results. The system virtual address (SVA) in Xj defines the required page table entry.

The SVA determines the starting point in the page table search. The search continues until the corresponding page descriptor is found, a continue bit equal to zero is detected, or 32 entries have been searched. The validity bit is ignored.

When the page is found, the page table index associated with that entry transfers to XkR, the number of entries searched transfers to X1R bits 33 through 63 (right-justified with zeros extended), and X1R bit 32 sets to indicate the find.

When a page corresponding to the SVA in Xj is not found, the page table index value of the last entry tested transfers to XkR, the number of entries searched transfers to X1R bits 33 through 63 (right-justified with zeros extended), and X1R bit 32 clears.

If the instruction's k field equals one, register X1R loads with the result-indication bit and the number of entries searched instead of the index value.

This instruction can cause the following exception conditions.

• Privileged instruction fault

• Address specification error

Global Privileged System Instruction

The processor interrupt instruction can execute only from segments which (by the associated segment descriptor) have global privilege. If this instruction is fetched from a segment without global privilege, the CP detects a privileged instruction fault, inhibits execution, and initiates the corresponding program interrupt or halt.

Processor Interrupt

Opcode   03jk

Mnemonic   INTRUPT Xk,j

Instruction Processor Interrupt per Xk

Format

0	78	1112	15
03	j	k	

Remarks   This instruction sends an external interrupt to one or more CPs (including the executing CP) through their CM ports. The interrupting CP sends Xk to CM. CM then sends an external interrupt to the processor(s) connected to the ports whose numbers correspond to the bits set in Xk as follows.

Xk Bit	Port Number
60	3
61	2
62	1
63	0

Bits 0 through 59 are not used to send interrupts and are ignored by the CM, but have correct parity. When two ports of the same memory connect to the interrupting CP, the state of Xk bit 33 selects the port the CP uses to send Xk to CM along with the interrupt. (Xk bit 33 thus overrides RMA bit 33 for memory port selection).

State of Bit 33	Memory Port Used
Clear	0
Set	1

The system delays this instruction's execution until all previous CM accesses by the interrupting CP complete. If a CP sends an interrupt to itself, this instruction completes executing before the interrupt is taken.

This instruction can cause the privileged instruction fault exception condition.

Monitor Mode Instructions

Instructions in this subgroup can execute only with the processor in executive monitor mode. Otherwise, the CP detects an instruction specification error, inhibits instruction execution, and initiates the corresponding program interrupt. Refer to Mixed Mode Instructions in the following text.

Mixed Mode Instructions

The execution of instructions in this subgroup (table 1-20) depends on an instruction parameter. The parameter value determines whether the instruction is executable from nonprivileged, local-privileged, or global-privileged segments, or whether the CP must be in Virtual State monitor mode.

Table 1-20. Mixed Mode Instructions

Opcode	Format	Instruction	Mnemonic
05	jk	Purge buffer	PURGE
0E	jk	Copy from state register	CPYSX
0F	jk	Copy to state register	CPYXS
9F	jk	Branch on condition register	BRCR

Purge Buffer

Opcode   05jk

Mnemonic PURGE Xj,k

Instruction Purge Buffer k of entry per Xj

Format

0	78	1112	15
05	j	k	

Remarks   This instruction invalidates entries in the cache, map, or instruction buffer, selectable as follows.

- All entries in cache, map, or instruction buffer

- All entries for a given segment in cache or map

- All entries for a given page in cache or map

- All entries for a given 512-byte block in cache

Xj contains the required address information as either the SVA or the PVA. The k value determines the buffer to be purged, the range of entries to be purged, and the addressing type used, as follows.

Value of k	Description
0	Purge all cache entries in a 512-byte block defined by SVA in Xj
1	Purge all cache entries in the active segment identifier (ASID) defined by SVA in Xj
2	Purge all cache entries
3	Purge all cache entries in 512-byte block defined by PVA in Xj
4 through 7	Purge all cache entries in SEG defined by PVA in Xj
8	Purge all map entries in page associated with page table entry defined by SVA in Xj (page size is determined from page size mask register)

Value of k	Description
9	Purge all map entries pertaining to page table entries included in segment defined by SVA in Xj
A	Purge all map entries pertaining to page table entry defined by PVA in Xj (page size mask register specifies number of bytes in page)
B	Purge all map entries pertaining to segment table entry defined by PVA in Xj and to all page table entries included within that segment
C through F	Purge all map entries, ignore Xj

If k equal 0, 1, 2, or 8 through F, this instruction is a local-privileged instruction. It is a nonprivileged instruction for all other k values.

The system performs a serialization function before this instruction begins execution, and again when execution completes. The system delays instruction execution until all previous accesses to CM by this processor complete, and delays the fetch or execution of subsequent instructions until all CM accesses for this instruction complete.

This instruction may cause the following exception conditions.

- Privileged instruction fault

- Address specification error (k = 0, 1, 8, or 9)

- Invalid segments

Copy to/from State Register

These instructions copy certain state registers to and from X registers.

If a copy instruction reads a nonexistent register or any register restricted to Maintenance Control Unit (MCU) access only, the system clears all 64 bits of Xk. A copy instruction used to write a nonexistent register, or any register restricted to read only or MCU access only, results in a no-operation.

Opcode	0Ejk
Mnemonic	CPYSX Xk,Xj
Instruction	Copy to Xk from State Register per Xj

Format

```
0           78  1112 15
 ┌──────────┬────┬────┐
 │    OE    │  j │  k │
 └──────────┴────┴────┘
```

Remarks   This instruction copies the state register addressed by Xj into Xk.

Opcode	0Fjk
Mnemonic	CPYXS Xk,Xj
Instruction	Copy to State Register from Xk per Xj

Format

```
0           78  1112 15
 ┌──────────┬────┬────┐
 │    OF    │  j │  k │
 └──────────┴────┴────┘
```

Remarks   This instruction copies Xk into the state register addressed by Xj.

These instructions can cause the following exception conditions.

* Instruction specification error

* Privileged instruction fault (CPYXS only)

Branch on Condition Register

Opcode   9FjkQ

Mnemonic   BRCR j,k,Q

Instruction   Branch to P displaced by 2*Q and Alter Condition Register per jk

Format

0	78	1112	1516	31
9F	j	k	Q	

Remarks   This instruction tests the state of a bit in the MCR or UCR, as selected by the instruction j and k fields. The j field selects the bit within the register; and the k field selects the register, branch condition, and bit alteration. When the branch condition is satisfied, the target address forms by adding two times Q (sign-extended) to the BN in P. The instruction depends on k as follows.

Value of k	Description
0 or 8	If bit j of MCR is set, clear bit and branch
1 or 9	If bit j of MCR is clear, set bit and branch
2 or A	If bit j of MCR is set, branch
3 or B	If bit j of MCR is clear, branch
4 or C	If bit j of UCR is set, clear bit and branch
5 or D	If bit j of UCR is clear, set bit and branch
6 or E	If bit j of UCR is set, branch
7 or F	If bit j of UCR is clear, branch

When the k field is 0, 1, 8, or 9, this instruction executes in Virtual State monitor mode only. If the processor is not in monitor mode with execution restricted to that mode, the CP detects an instruction specification error, inhibits instruction execution, and initiates the corresponding program interrupt or halt.

This instruction can cause the following exception conditions.

● Instruction specification error

● Debug

Peripheral Processor Instruction Descriptions

The PP instruction set comprises the following eight subgroups:

- Load and store

- Arithmetic

- Logical

- Replace

- Branch

- CM access

- Input/Output (I/O)

- Other IOU

The Virtual State PP instruction set includes the CYBER 170 State PP instructions as a subset. The instruction set uses a 7-bit opcode which includes the CYBER 170 State 6-bit opcode. Extensions to the instruction set allow programs to manipulate 16-bit IOU words, 64-bit CM words (as both 12- and 16-bit bytes), and to reference 28-bit CM addresses.

PP Instruction Formats

The following figures show and table 1-21 describes PP instruction formats. PP instructions are 16 or 32 bits long. In instruction descriptions, the opcode is represented by four or five octal digits. The fifth digit, when used, indicates the state of bit 58 (zero or one) in I/O instructions.

4849	5152		5758		63
g	e	f		d	

48	63
m	

4849	5152		575859		63
g	e	f	s	c	

48	63
m	
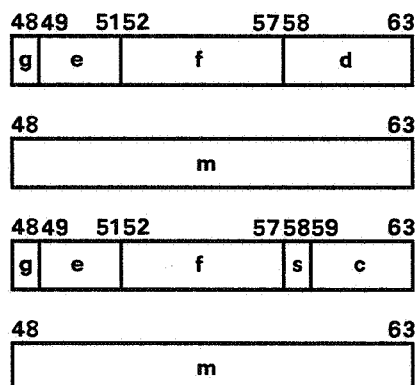
Table 1-21. PP Instruction Format Nomenclature

Designator	Description
f	Instruction opcode, least significant six bits of 7-bit opcode
e	Unused bits for future use; should be set to zero
d	Operand, direct/indirect address, shift count
g	Most significant bit of 7-bit opcode controls width of operand from PP memory (PPM); g=0 is 12 bits. g=1 is 16 bits
m	Operand, direct address, or I/O function code, CYBER 170 State uses 12 bits zero filled on the left; Virtual State uses all 16 bits
s	I/O instruction subcode
c	I/O instruction channel number
A	Eighteen-bit arithmetic register
P	Twelve-bit program address counter
R	Twenty-eight-bit relocation register for CM addressing (bits 58 through 63 of R are appended zeros)
( )	Quantity in brackets is a direct address (used when required for clarity)
(( ))	Quantity in brackets is an indirect address (used when required for clarity)

PP Data Format

Figure 1-1 shows PP data formats, the packing of PP data into CM words, and the unpacking of CM words into PP words.
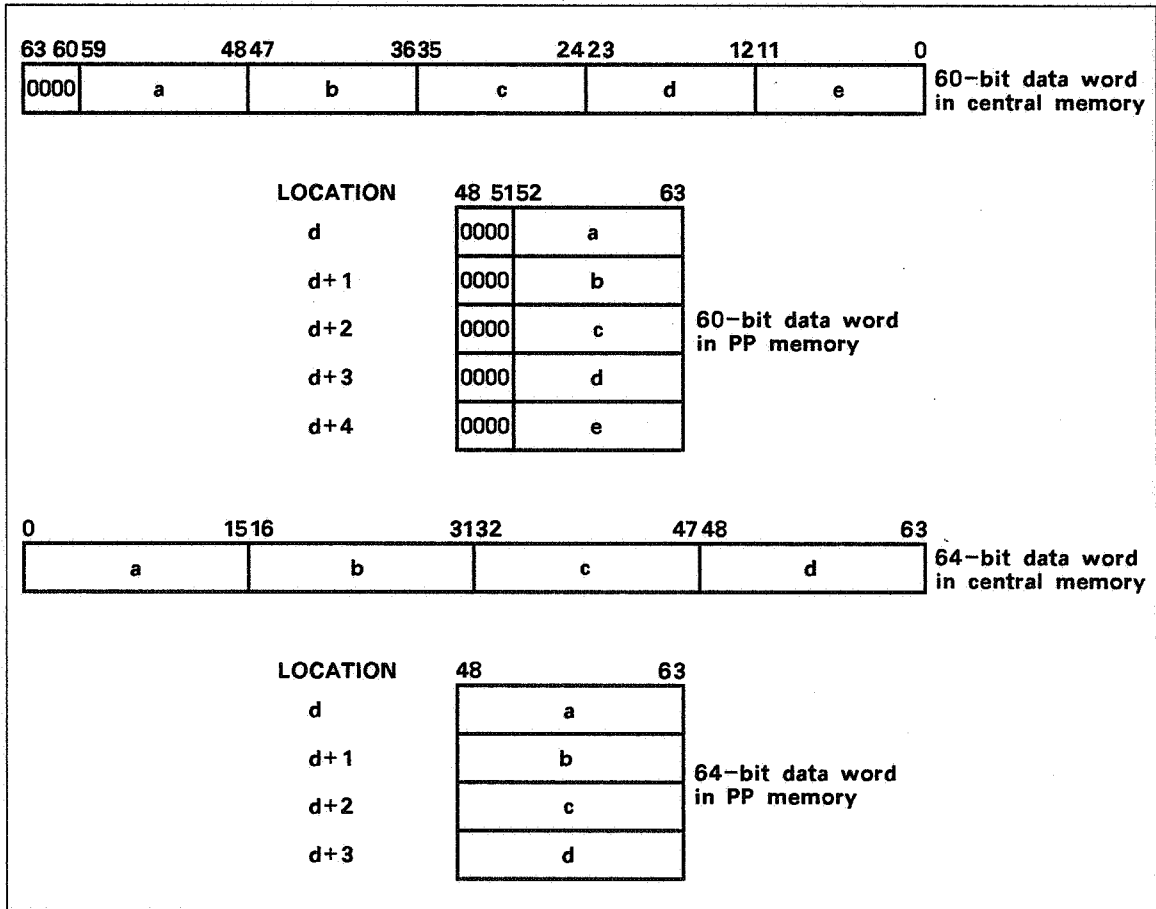
Figure 1-1. PP Data Format

PP Relocation Register Format

Figure 1-2 shows the PP relocation register format. This register is loaded/stored by instructions 24 and 25 (load/store R register).
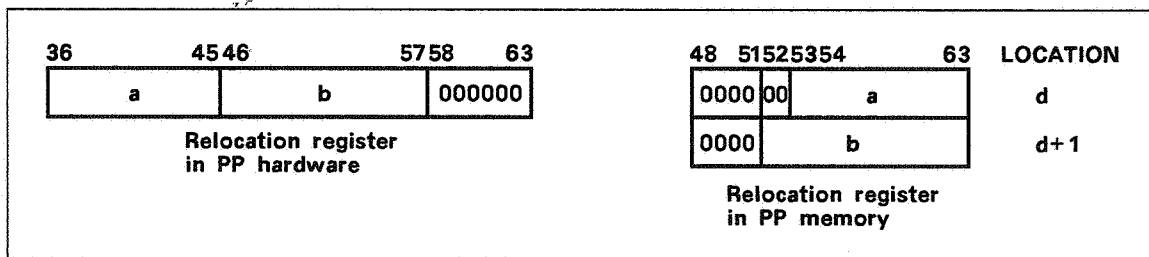
Figure 1-2. PP Relocation Register Format

PP Load and Store Instructions

Load and store instructions (table 1-22) transfer 6-, 12-, 16-, and 18-bit quantities between the PP A register and the PPM.

Table 1-22. PP Load and Store Instructions

Opcode	Format	Instruction	Mnemonic
0014	d	Load d	LDN
0015	d	Load complement d	LCN
0020	dm	Load dm	LDC
0030	d	Load (d)	LDD
1030	d	Load (d) long	LDDL
0040	d	Load ((d))	LDI
1040	d	Load ((d)) long	LDIL
0050	dm	Load (m+(d))	LDM
1050	dm	Load (m+(d)) long	LDML
0034	d	Store (d)	STD
1034	d	Store (d) long	STDL
0044	d	Store ((d))	STI
1044	d	Store ((d)) long	STIL
0054	dm	Store (m+(d))	STM
1054	dm	Store (m+(d)) long	STML

Load

Opcode	**0014d**
Mnemonic	LDN d
Instruction	Load d

Format

```
48         57 58    63
┌──────────┬────────┐
│   0014   │   d    │
└──────────┴────────┘
```

Remarks   This instruction clears the A register and loads the rightmost six bits of A with a copy of the 6-bit positive integer in the d field. The leftmost 12 bits of A are zeros.

Opcode	**0015d**
Mnemonic	LCN d
Instruction	Load Complement d

Format

```
48         57 58    63
┌──────────┬────────┐
│   0015   │   d    │
└──────────┴────────┘
```

Remarks   This instruction clears the A register and loads the rightmost 6 bits of A with a ones-complement copy of the d field. The leftmost 12 bits of A are ones.

Opcode	**0020dm**
Mnemonic	LDC dm
Instruction	Load dm

Format

```
32          41 42    47 48 51 52            63
┌────────────┬───────┬──────┬────────────────┐
│    0020    │   d   │ 0000 │       m        │
└────────────┴───────┴──────┴────────────────┘
      └──────── P ────────┘ └────── P+1 ──────┘
```

Remarks   This instruction clears the A register and loads A with an 18-bit operand consisting of d as the leftmost 6 bits and m as the rightmost 12 bits.

Opcode	**0030d**
Mnemonic	LDD d
Instruction	Load (d)

Format

```
48         57 58    63
┌──────────┬────────┐
│   0030   │   d    │
└──────────┴────────┘
```

Remarks   This instruction clears the A register and loads A with the rightmost twelve bits of the positive integer from PPM location d. The leftmost 6 bits of A are zeros.

Opcode	**1030d**
Mnemonic	LDDL d
Instruction	Load (d) Long
Format	

```
48          5758     63
┌──────────────┬────────┐
│    1030      │   d    │
└──────────────┴────────┘
```

Remarks — This instruction clears the A register and loads A with the 16-bit positive integer from PPM location d. The leftmost two bits of A are zeros.

Opcode	**0040d**
Mnemonic	LDI d
Instruction	Load ((d))
Format	

```
48          5758     63
┌──────────────┬────────┐
│    0040      │   d    │
└──────────────┴────────┘
```

Remarks — This instruction clears the A register and loads A with the rightmost 12 bits of an operand obtained by indirect addressing. The leftmost 6 bits of A are zeros. To perform indirect addressing, the IOU reads a word from PPM location d and uses it as the operand address.

Opcode	**1040d**
Mnemonic	LDIL d
Instruction	Load ((d)) Long
Format	

```
48          5758     63
┌──────────────┬────────┐
│    1040      │   d    │
└──────────────┴────────┘
```

Remarks — This instruction clears the A register and loads A with a 16-bit operand obtained by indirect addressing. The leftmost two bits of A are zeros. To perform indirect addressing, the IOU reads a word from PPM location d and uses it as the operand address.

Opcode	**0050dm**
Mnemonic	LDM m,d
Instruction	Load (m+(d))
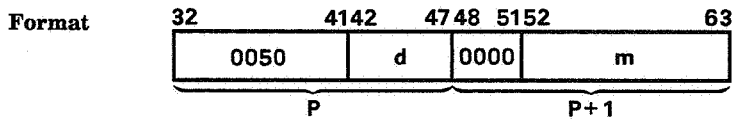
Format

32	4142	4748 5152	63
0050	d	0000	m

P   P+1

Remarks   This instruction clears the A register and loads A with the rightmost 12 bits of an operand read from PPM using indexed direct addressing. The leftmost 6 bits of A are zeros.

To accomplish indexed direct addressing, the IOU adds an index value to a base address to form the operand address. The m field contains the base address and the d field specifies the PPM location containing the index value. If d equals zero, the m field base address is the operand address.

Opcode	**1050dm**
Mnemonic	LDML m,d
Instruction	Load (m+(d)) Long

Format

32	4142	4748 5152	63
1050	d	0000	m

P   P+1
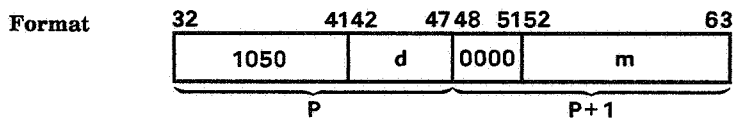
Remarks   This instruction clears the A register and loads A with a 16-bit operand read from PPM using indexed direct addressing. The leftmost two bits of A are zeros.

To accomplish indexed direct addressing, the IOU adds an index value to a base address to form the operand address. The m field contains the base address and the d field specifies the PPM location containing the index value. If d equals zero, the m field base address is the operand address.

Store

Opcode	0034d
Mnemonic	STD d
Instruction	Store (d)

Format

```
48          5758    63
 ┌──────────┬─────────┐
 │   0034   │    d    │
 └──────────┴─────────┘
```

Remarks   This instruction stores the quantity contained in the A register rightmost 12 bits in location d and clears the leftmost 4 bits of location d. The operation does not alter the contents of A.

Opcode	1034d
Mnemonic	STDL d
Instruction	Store (d) Long

Format

```
48          5758    63
 ┌──────────┬─────────┐
 │   1034   │    d    │
 └──────────┴─────────┘
```

Remarks   This instruction stores the A register rightmost 16 bits in location d. The operation does not alter the contents of A.

Opcode	0044d
Mnemonic	STI d
Instruction	Store ((d))

Format

```
48          5758    63
 ┌──────────┬─────────┐
 │   0044   │    d    │
 └──────────┴─────────┘
```

Remarks   This instruction stores the A register rightmost 12 bits in the storage location specified by the location d contents. The leftmost 4 bits of ((d)) are zeros and the A register contents are unaltered.

Opcode	1044d
Mnemonic	STIL d
Instruction	Store ((d)) Long

Format

```
48          5758    63
 ┌──────────┬─────────┐
 │   1044   │    d    │
 └──────────┴─────────┘
```

Remarks   This instruction stores the A register rightmost 16 bits at a location obtained by indirect addressing. The A register contents are unaltered.

Opcode	0054dm
Mnemonic	STM m,d
Instruction	Store (m+(d))

Format

```
32          4142    4748 5152              63
┌───────────┬─────┬─────┬──────────────────┐
│   0054    │  d  │0000 │        m         │
└───────────┴─────┴─────┴──────────────────┘
       P              P+1
```
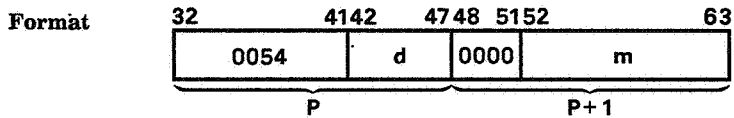
Remarks

This instruction stores the A register rightmost 12 bits in the location determined by indexed direct addressing. Bits 48 through 51 of (m+(d)) clear. The A register contents are unaltered.

To accomplish indexed direct addressing, the IOU adds an index value to a base address to form the operand address. The m field contains the base address and the d field specifies the PPM location containing the index value. If d equals zero, the m field base address is the operand address.

Opcode	1054dm
Mnemonic	STML m,d
Instruction	Store (m+(d)) Long

Format

```
32          4142    4748 5152              63
┌───────────┬─────┬─────┬──────────────────┐
│   1054    │  d  │0000 │        m         │
└───────────┴─────┴─────┴──────────────────┘
       P              P+1
```
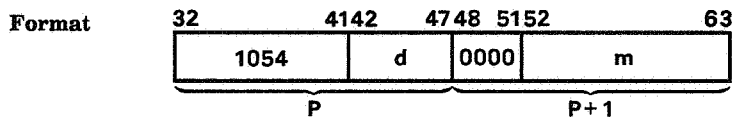
Remarks

This instruction stores the A register rightmost 16 bits in the location determined by indexed direct addressing.

To accomplish indexed direct addressing, the IOU adds an index value to a base address to form the operand address. The m field contains the base address and the d field specifies the PPM location containing the index value. If d equals zero, the m field base address is the operand address.

PP Arithmetic Instructions

The PP arithmetic instructions (table 1-23) perform integer arithmetic using the PP A register contents as one operand, with the other operand specified by the instruction. The result replaces the original contents of A. The IOU considers the operands as ones-complement integers and performs the arithmetic in ones complement.

Table 1-23. PP Arithmetic Instructions

Opcode	Format	Instruction	Mnemonic
0016	d	Add d	ADN
0021	dm	Add dm	ADC
0031	d	Add (d)	ADD
1031	d	Add (d) long	ADDL
0041	d	Add ((d))	ADI
1041	d	Add ((d)) long	ADIL
0051	dm	Add (m+(d))	ADM
1051	dm	Add (m+(d)) long	ADML
0017	d	Subtract d	SBN
0032	d	Subtract (d)	SBD
1032	d	Subtract (d) long	SBDL
0042	d	Subtract ((d))	SBI
1042	d	Subtract ((d)) long	SBIL
0052	dm	Subtract (m+(d))	SBM
1052	dm	Subtract (m+(d)) long	SBML

Arithmetic Add

Opcode   **0016d**

Mnemonic   ADN d

Instruction   Add d

Format

```
48        57 58    63
 ┌──────────┬────────┐
 │   0016   │    d   │
 └──────────┴────────┘
```

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

Opcode   **0021dm**

Mnemonic   ADC dm

Instruction   Add dm

Format

```
32          41 42    47 48 51 52            63
 ┌────────────┬────────┬──────┬──────────────┐
 │    0021    │    d   │ 0000 │      m       │
 └────────────┴────────┴──────┴──────────────┘
 └────────────────────┘ └────────────────────┘
           P                      P+1
```

Remarks   This instruction adds an 18-bit value comprised of the 6-bit d field and the m field rightmost 12 bits to the A register operand. The d field becomes the leftmost 6 bits and m becomes the rightmost 12 bits of the value.

Opcode   **0031d**

Mnemonic   ADD d

Instruction   Add (d)

Format

```
48        57 58    63
 ┌──────────┬────────┐
 │   0031   │    d   │
 └──────────┴────────┘
```

Remarks   This instruction adds the rightmost 12 bits of the operand in location d to the A register contents.

Opcode	**1031d**
Mnemonic	ADDL d
Instruction	Add (d) Long
Format ·	

```
48          5758    63
 _____
|     1031      |   d    |
|_____|_____|
```

Remarks	This instruction adds the 16-bit operand in location d to the A register contents.

Opcode	**0041d**
Mnemonic	ADI d
Instruction	Add ((d))
Format	

```
48          5758    63
 _____
|     0041      |   d    |
|_____|_____|
```

Remarks	This instruction reads a 12-bit operand from PPM using indirect addressing and adds the rightmost 12 bits to the A register contents. To perform indirect addressing, the PP reads a word from PPM location d and uses it as the operand address.

Opcode	**1041d**
Mnemonic	ADIL d
Instruction	Add ((d)) Long
Format	

```
48          5758    63
 _____
|     1041      |   d    |
|_____|_____|
```

Remarks	This instruction reads a 16-bit operand from PPM using indirect addressing and adds the operand to the A register contents. To perform indirect addressing, the PP reads a word from PPM location d and uses it as the operand address.

Opcode   **0051dm**

Mnemonic   ADM m,d

Instruction   Add (m+(d))
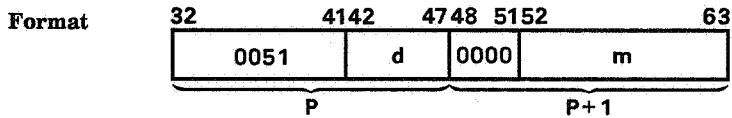
Format

32	4142	4748 5152	63
0051	d	0000	m

P   P+1

Remarks   This instruction reads an operand from PPM using indexed direct addressing and adds the rightmost 12 bits to the A register contents.

To accomplish indexed direct addressing, the PP adds an index value to a base address to form the operand address. The m field contains the base address and the d field specifies the PPM location containing the index value. If d equals zero, the m field base address is the operand address.

Opcode   **1051dm**

Mnemonic   ADML m,d

Instruction   Add (m+(d)) Long
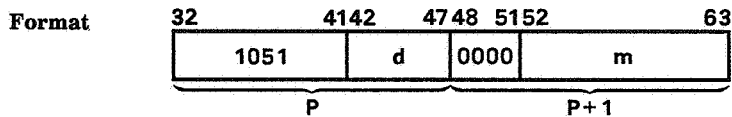
Format

32	4142	4748 5152	63
1051	d	0000	m

P   P+1

Remarks   This instruction reads a 16-bit operand from PPM using indexed direct addressing and adds the operand to the A register contents.

To accomplish indexed direct addressing, the PP adds an index value to a base address to form the operand address. The m field contains the base address and the d field specifies the PPM location containing the index value. If d equals zero, the m field base address is the operand address.

Arithmetic Subtract

Opcode   0017d

Mnemonic   SBN d

Instruction   Subtract d

Format

48	5758	63
0017	d	

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

Opcode   0032d

Mnemonic   SBD d

Instruction   Subtract (d)

Format

48	5758	63
0032	d	

Remarks   This instruction subtracts the 12-bit operand in location d from the A register contents.

Opcode   1032d

Mnemonic   SBDL d

Instruction   Subtract (d) Long

Format

48	5758	63
1032	d	

Remarks   This instruction subtracts the 16-bit operand in location d from the A register contents.

Opcode	**0042d**
Mnemonic	SBI d
Instruction	Subtract ((d))
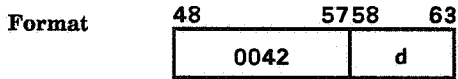
Format

```
48          5758    63
┌──────────────┬──────┐
│     0042     │  d   │
└──────────────┴──────┘
```

Remarks This instruction reads a 12-bit operand from PPM using indirect addressing and subtracts the operand from the A register contents. To perform indirect addressing, the PP reads a word from PPM location d and uses it as the operand address.

Opcode	**1042d**
Mnemonic	SBIL d
Instruction	Subtract ((d)) Long
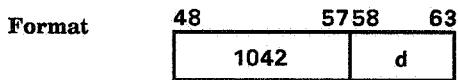
Format

```
48          5758    63
┌──────────────┬──────┐
│     1042     │  d   │
└──────────────┴──────┘
```

Remarks This instruction reads a 16-bit operand from PPM using indirect addressing and subtracts the operand from the A register contents. To perform indirect addressing, the PP reads a word from PPM location d and uses it as the operand address.

Opcode	**0052dm**
Mnemonic	SBM m,d
Instruction	Subtract (m+(d))
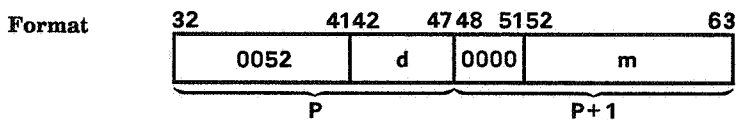
Format

```
32          4142    4748 5152            63
┌──────────────┬──────┬──────┬──────────────┐
│     0052     │  d   │ 0000 │      m       │
└──────────────┴──────┴──────┴──────────────┘
        └────────P───────┘  └──────P+1──────┘
```

Remarks This instruction reads an operand from PPM using indexed direct addressing and subtracts the rightmost 12 bits of the operand from the A register contents.

To accomplish indexed direct addressing, the PP adds an index value to a base address to form the operand address. The m field contains the base address and the d field specifies the PPM location containing the index value. If d equals zero, the m field base address is the operand address.

Opcode   **1052dm**

Mnemonic   SBML m,d

Instruction   Subtract (m+(d)) Long
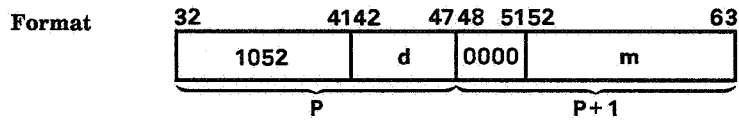
Format

32	41	42	47	48	51	52	63
1052		d		0000		m	

P   P+1

Remarks   This instruction reads a 16-bit operand from PPM using indexed direct addressing and subtracts this operand from the A register contents.

To accomplish indexed direct addressing, the PP adds an index value to a base address to form the operand address. The m field contains the base address and the d field specifies the PPM location containing the index value. If d equals zero, the m field base address is the operand address.

PP Logical Instructions

The logical instructions (table 1-24) perform operations with one operand as the PP A register contents, and the other as specified by the instruction. The result replaces the original contents of A.

Table 1-24. PP Logical Instructions

Opcode	Format	Instruction	Mnemonic
0010	d	Shift d	SHN
0011	d	Logical difference d	LMN
0023	dm	Logical difference dm	LMC
0033	d	Logical difference (d)	LMD
1033	d	Logical difference (d) long	LMDL
0043	d	Logical difference ((d))	LMI
1043	d	Logical difference ((d)) long	LMIL
0053	dm	Logical difference (m+(d))	LMM
1053	dm	Logical difference (m+(d)) long	LMML
0012	d	Logical product d	LPN
0022	dm	Logical product dm	LPC
1022	d	Logical product (d) long	LPDL
1023	d	Logical product ((d)) long	LPIL
1024	dm	Logical product (m+(d)) long	LPML
0013	d	Selective clear d	SCN

Logical Shift

Opcode   **0010d**

Mnemonic   SHN d

Instruction   Shift A by d

Format

48 57	58 63
0010	d

Remarks   This instruction shifts the A register operand in the direction and by the number of places specified by the d field value. If d is in the range 00 through 37 (a positive value), the shift is left-circular, d positions. A circular shift means that a bit shifted out of the highest-order position moves into the lowest-order position.

If d is in the range 40 through 77 (a negative value), the shift is to the right, end-off. Thus, d equal to 06 causes a left shift of six places; d equal to 71 causes a right shift of six places.

Logical Difference

Opcode   **0011d**

Mnemonic   LMN d

Instruction   Logical Difference d

Format

48 57	58 63
0011	d

Remarks   This instruction forms the logical difference between the d field contents and the rightmost 6 bits of the A register operand. The operation does not alter the most significant 12 bits of A.

The logical difference (exclusive OR) results from a bit-for-bit logical comparison of the two binary quantities, as illustrated by the following example.
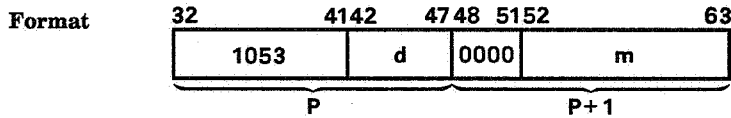
Operand 1	0011
Operand 2	0101

Result	0110

This comparison is equivalent to complementing the first operand bits corresponding to the second operand bits that are ones.

Opcode   0023dm

Mnemonic   LMC dm

Instruction   Logical Difference dm

Format

32	4142	4748 5152	63
0023	d	0000	m

P   P+1
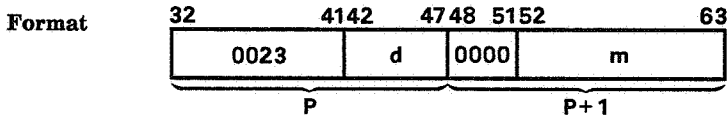
Remarks   This instruction replaces the A register contents with the bit-for-bit logical difference between the A register operand and the 18-bit quantity dm.

The logical difference (exclusive OR) results from a bit-for-bit logical comparison of the two binary quantities, as illustrated by the following example.

Operand 1   0011
Operand 2   0101
——————
Result   0110

This comparison is equivalent to complementing the first operand bits corresponding to the second operand bits that are ones.

Opcode   0033d

Mnemonic   LMD d

Instruction   Logical Difference (d)
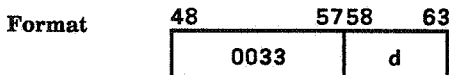
Format

48	5758	63
0033	d	

Remarks   This instruction replaces the A register contents with the logical difference between the A register rightmost 12 bits and the rightmost 12 bits of the operand in the location specified by the d field. The operation does not alter the leftmost 6 bits of A.

Opcode   1033d

Mnemonic   LMDL d
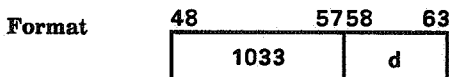
Instruction   Logical Difference (d) Long

Format

48	5758	63
1033	d	

Remarks   This instruction replaces the A register contents with the logical difference between the A register rightmost 16 bits and the operand in the location specified by the d field. The operation does not alter the most significant 2 bits of A.

Opcode	**0043d**
Mnemonic	LMI d
Instruction	Logical Difference ((d))

Format

48	5758	63
0043		d

Remarks   This instruction replaces the A register contents with the logical difference between the A register rightmost 12 bits and the rightmost 12 bits of an operand read by indirect addressing. The operation does not alter the leftmost 6 bits of A. The d field contents specify the PPM location containing the operand address.

The logical difference (exclusive OR) results from a bit-for-bit logical comparison of the two binary quantities, as illustrated by the following example.

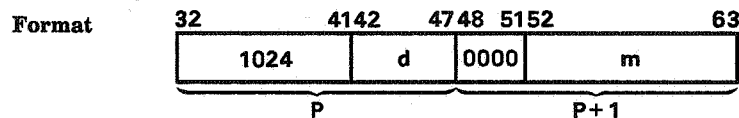| Operand 1	0011
Operand 2	0101
Result	0110

This comparison is equivalent to complementing the first operand bits corresponding to the second operand bits that are ones.

Opcode	**1043d**
Mnemonic	LMIL d
Instruction	Logical Difference ((d)) Long

Format

48	5758	63
1043		d

Remarks   This instruction replaces the A register contents with the logical difference between the A register rightmost 16 bits and an operand read by indirect addressing. The operation does not alter the leftmost 2 bits of A. The d field contents specify the PPM location containing the operand address.

The logical difference (exclusive OR) results from a bit-for-bit logical comparison of the two binary quantities, as illustrated by the following example.

Operand 1	0011
Operand 2	0101
Result	0110

This comparison is equivalent to complementing the first operand bits corresponding to the second operand bits that are ones.

Opcode   **0053dm**

Mnemonic   LMM m,d

Instruction   Logical Difference (m+(d))
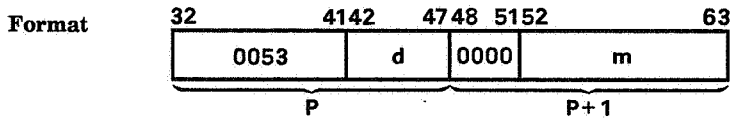
Format

32	4142	4748 5152	63
0053	d	0000	m

          P   P+1

Remarks   This instruction replaces the A register contents with the logical difference between the A register rightmost 12 bits and the rightmost 12 bits of an operand read by indexed direct addressing. The operation does not alter the most significant 6 bits of A. Indexed direct addressing uses the m field contents as a base address. The d field specifies the PPM location containing the index value which the PP adds to the base address to form the operand address.

The logical difference (exclusive OR) results from a bit-for-bit logical comparison of the two binary quantities, as illustrated by the following example.

Operand 1	0011
Operand 2	0101
	————
Result	0110

This comparison is equivalent to complementing the first operand bits corresponding to the second operand bits that are ones.

Opcode	**1053dm**
Mnemonic	LMML m,d
Instruction	Logical Difference (m+(d)) Long

Format

```
32          4142     4748 5152              63
 ┌──────────┬────┬──────┬───────────────────┐
 │   1053   │ d  │ 0000 │        m          │
 └──────────┴────┴──────┴───────────────────┘
       _____P_____/    \_____P+1_____/
```

Remarks This instruction replaces the A register contents with the logical difference between the A register rightmost 16 bits and an operand read by indexed direct addressing. The operation does not alter the leftmost 2 bits of A. Indexed direct addressing uses the m field contents as a base address. The d field specifies the PPM location containing the index value which the PP adds to the base address to form the operand address.

The logical difference (exclusive OR) results from a bit-for-bit logical comparison of the two binary quantities, as illustrated by the following example.

Operand 1	0011
Operand 2	0101
	————
Result	0110

This comparison is equivalent to complementing the first operand bits corresponding to the second operand bits that are ones.

Logical Product

Opcode   0012d

Mnemonic   LPN d

Instruction   Logical Product d

Format

48	5758	63
0012	d	

Remarks   This instruction forms the logical product of the d field contents and the rightmost 6 bits of the A register operand. The leftmost 12 bits of A are zeros.

The logical product results from a bit-for-bit logical comparison of the two binary quantities, as illustrated by the following example.

Operand 1	0011
Operand 2	0101

Result	0001

The individual result bits are ones if the corresponding bits in the first and second operand are ones.

Opcode   0022dm

Mnemonic   LPC dm

Instruction   Logical Product dm

Format

32	4142	4748	5152	63
0022	d	0000	m	

P   P+1

Remarks   This instruction forms the bit-for-bit logical product of the A register operand and an 18-bit operand consisting of d as the leftmost 6 bits and m as the rightmost 12 bits.

The logical product results from a bit-for-bit logical comparison of the two binary quantities, as illustrated by the following example.

Operand 1	0011
Operand 2	0101

Result	0001

Opcode	**1022d**
Mnemonic	LPDL d
Instruction	Logical Product (d) Long
Format	
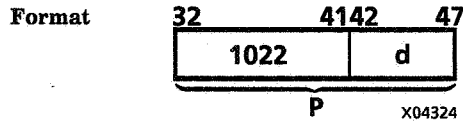
```
32          4142      47
 _____
|   1022    |     d     |
 _____
      P          X04324
```

Remarks This instruction forms the bit-for-bit logical product of the A register operand and the 16-bit quantity from location d. The result replaces the original contents of A; bits 46 and 47 of A clear.

The logical product results from a bit-for-bit logical comparison of the two binary quantities, as illustrated by the following example.

Operand 1   0011
Operand 2   0101
            _____

Result   0001

Opcode	**1023d**
Mnemonic	LPIL d
Instruction	Logical Product ((d)) Long
Format	

```
32          4142       47
 _____
|   1023    |     d     |
 _____
      P          X04325
```
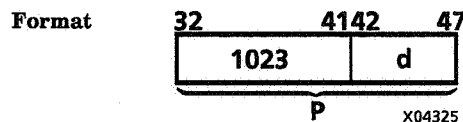
Remarks This instruction forms the logical product of a 16-bit operand read from storage (using indirect addressing) and the original contents of A. Bits 46 and 47 of A clear. The d field contents specify the PPM location containing the operand address.

Opcode	**1024dm**
Mnemonic	LPML m,d
Instruction	Logical Product (m+(d)) Long
Format	

```
32              4142    4748 5152              63
 _____
|   1024    |     d     |0000|        m          |
 _____
        P                       P+1
```

Remarks This instruction replaces the A register contents with the logical product of the A register rightmost 16 bits and a 16-bit operand read by indexed direct addressing. Bits 46 and 47 of A clear. Indexed direct addressing uses the m field contents as a base address. The d field specifies a PPM location containing the index value which adds to the base address to form the operand address.

Selective Clear

Opcode   0013d

Mnemonic   SCN d

Instruction   Selective Clear d

Format

48 57	58 63
0013	d

Remarks   This instruction clears each of the rightmost 6 bits of the A register operand if the corresponding d field bit is a one. The operation does not alter the leftmost 12 bits of A.

PP Replace Instructions

The replace instructions (table 1-25) perform integer arithmetic with one operand as the contents of A and the location of the other as specified by the instruction. The result replaces the original contents of A and the contents of the other operand's location. The result stored in PPM is either the rightmost 12 bits (for the normal instructions) or the rightmost 16 bits (for the long instructions) of the A register. Therefore, since A contains 18 bits, the value remaining in A cannot equal the value stored in PPM.

The PP considers the operands as ones-complement integers and performs ones-complement arithmetic.

Table 1-25. PP Replace Instructions

Opcode	Format	Instruction	Mnemonic
0035	d	Replace add (d)	RAD
1035	d	Replace add (d) long	RADL
0036	d	Replace add one (d)	AOD
1036	d	Replace add one (d) long	AODL
0045	d	Replace add ((d))	RAI
1045	d	Replace add ((d)) long	RAIL
0046	d	Replace add one ((d))	AOI
1046	d	Replace add one ((d)) long	AOIL
0055	dm	Replace add (m+(d))	RAM
1055	dm	Replace add (m+(d)) long	RAML
0056	dm	Replace add one (m+(d))	AOM
1056	dm	Replace add one (m+(d)) long	AOML
0037	d	Replace subtract one (d)	SOD
1037	d	Replace subtract one (d) long	SODL
0047	d	Replace subtract one ((d))	SOI
1047	d	Replace subtract one ((d)) long	SOIL
0057	dm	Replace subtract one (m+(d))	SOM
1057	dm	Replace subtract one (m+(d)) long	SOML

Replace Add

Opcode   0035d

Mnemonic   RAD d

Instruction   Replace Add (d)

Format

48	5758	63
0035	d	

Remarks   This instruction adds the rightmost 12 bits of the location d contents to the A register and stores the rightmost 12 bits of A, zero-extended, at location d. The result also remains in A at the end of the operation, with the original contents purged.

Opcode   1035d

Mnemonic   RADL d

Instruction   Replace Add (d) Long

Format

48	5758	63
1035	d	

Remarks   This instruction replaces the operand in the PPM location d with the sum of the PPM location d operand plus the A register rightmost 16 bits. The result also remains in A at the end of the operation, with the original contents purged.

Opcode   0036d

Mnemonic   AOD d

Instruction   Replace Add One (d)

Format

48	5758	63
0036	d	

Remarks   This instruction clears the A register, loads A with the location d rightmost 12 bits, and adds one to A. The instruction then stores the rightmost 12 bits of A, zero-extended, at location d. The result remains in the A register at the end of the operation, with the original contents purged.

Opcode	**1036d**
Mnemonic	AODL d
Instruction	Replace Add One (d) Long

Format

```
48          57 58    63
┌───────────┬────────┐
│   1036    │   d    │
└───────────┴────────┘
```

Remarks This instruction replaces the operand in PPM location d with the sum of the original operand value plus one. The result remains in the A register at the end of the operation, with the original contents purged.

Opcode	**0045d**
Mnemonic	RAI d
Instruction	Replace Add ((d))

Format

```
48          57 58    63
┌───────────┬────────┐
│   0045    │   d    │
└───────────┴────────┘
```

Remarks This instruction replaces the rightmost 12 bits of the operand at the address specified by the PPM location d contents with the sum of the original operand value plus the A register rightmost 12 bits. The result also remains in A at the end of the operation, with the original contents purged.

Opcode	**1045d**
Mnemonic	RAIL d
Instruction	Replace Add ((d)) Long

Format

```
48          57 58    63
┌───────────┬────────┐
│   1045    │   d    │
└───────────┴────────┘
```

Remarks This instruction replaces the 16-bit operand at the address specified by the PPM location d contents with the sum of the original operand value plus the A register contents. The result also remains in A at the end of the operation, with the original contents purged.

Opcode	**0046d**
Mnemonic	AOI d
Instruction	Replace Add One ((d))

Format

```
48          5758    63
┌──────────────┬──────┐
│    0046      │  d   │
└──────────────┴──────┘
```

Remarks This instruction replaces the rightmost 12 bits of the operand at the address specified by the PPM location d contents with the sum of the original operand value plus one. The result remains in the A register at the end of the operation, with the original contents purged.

Opcode	**1046d**
Mnemonic	AOIL d
Instruction	Replace Add One ((d)) Long

Format

```
48          5758    63
┌──────────────┬──────┐
│    1046      │  d   │
└──────────────┴──────┘
```

Remarks This instruction replaces the operand at the address specified by the PPM location d contents with the sum of the original operand value plus one. The result remains in the A register at the end of the operation, with the original contents purged.

Opcode	**0055dm**
Mnemonic	RAM m,d
Instruction	Replace Add (m+(d))

Format

```
32          4142    4748 5152              63
┌──────────────┬──────┬──────┬───────────────┐
│    0055      │  d   │ 0000 │      m        │
└──────────────┴──────┴──────┴───────────────┘
└──────────────────────┘└──────────────────────┘
           P                      P+1
```

Remarks This instruction reads the rightmost 12 bits of an operand from PPM using indexed direct addressing and adds the operand to the the A register contents. The sum's rightmost 12 bits replace the original PPM operand. The result remains in A at the end of the operation, with the original contents of A purged. Indexed direct addressing uses the m field contents as a base address. The d field specifies the PPM location containing the index value which the PP adds to the base address to form the operand address.

Opcode	**1055dm**
Mnemonic	RAML m,d
Instruction	Replace Add (m+(d)) Long
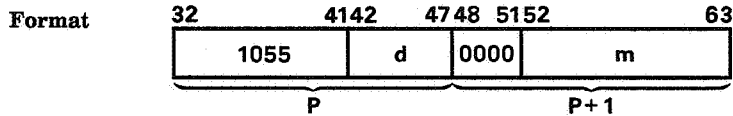
Format

```
32          4142    4748 5152              63
|   1055   |   d   |0000|        m          |
     _____P_____/  _____P+1_____/
```

Remarks This instruction reads a 16-bit operand from PPM using indexed direct addressing, adds the operand to the A register contents, and replaces the original PPM operand with the sum's rightmost 16 bits. The result remains in A at the end of the operation. Indexed direct addressing uses the m field contents as a base address. The d field specifies the PPM location containing the index value which the PP adds to the base address to form the operand address.

Opcode	**0056dm**
Mnemonic	AOM m,d
Instruction	Replace Add One (m+(d))

Format

```
32          4142    4748 5152              63
|   0056   |   d   |0000|        m          |
     _____P_____/  _____P+1_____/
```
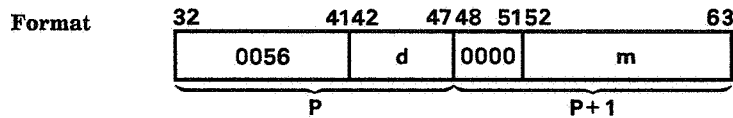
Remarks This instruction adds one to the rightmost 12 bits of a PPM operand read using indexed direct addressing and replaces the original PPM operand with the sum's rightmost 12 bits. Indexed direct addressing uses the m field contents as a base address. The d field specifies the PPM location containing the index value which the PP adds to the base address to form the operand address.

Opcode	**1056dm**
Mnemonic	AOML m,d
Instruction	Replace Add One (m+(d)) Long

Format

```
32          4142    4748 5152              63
|   1056   |   d   |0000|        m          |
     _____P_____/  _____P+1_____/
```
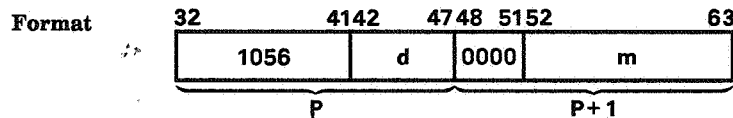
Remarks This instruction adds one to a 16-bit PPM operand read using indexed direct addressing and replaces the original PPM operand with the sum's rightmost 16 bits. Indexed direct addressing uses the m field contents as a base address. The d field specifies the PPM location containing the index value which the PP adds to the base address to form the operand address.

Replace Subtract

Opcode   0037d

Mnemonic   SOD d

Instruction   Replace Subtract One (d)

Format

48	5758	63
0037		d

Remarks   This instruction replaces the rightmost 12 bits of the operand in PPM location d with the original operand value minus one. The result remains in the A register at the end of the operation, with the original contents purged.

Opcode   1037d

Mnemonic   SODL d

Instruction   Replace Subtract One (d) Long

Format

48	5758	63
1037		d

Remarks   This instruction replaces the operand in PPM location d with the difference of the original operand value minus one. The result remains in the A register at the end of the operation, with the original contents purged.

Opcode   0047d

Mnemonic   SOI d

Instruction   Replace Subtract One ((d))

Format

48	5758	63
0047		d

Remarks   This instruction replaces the rightmost 12 bits of the operand at the address specified by the PPM location d contents with the original operand value minus one. The result remains in the A register at the end of the operation, with the original contents purged.

Opcode	1047d
Mnemonic	SOIL d
Instruction	Replace Subtract One ((d)) Long

Format

```
48          57 58      63
|   1047    |    d    |
```

Remarks This instruction replaces the operand at the address specified by the PPM location d contents with the original operand value minus one. The result remains in the A register at the end of the operation, with the original contents purged.

Opcode	0057dm
Mnemonic	SOM m,d
Instruction	Replace Subtract One (m+(d))

Format

```
32          41 42    47 48 51 52              63
|   0057    |    d   | 0000 |      m          |
      P                        P+1
```

Remarks This instruction subtracts one from the rightmost 12 bits of a PPM operand read using indexed direct addressing, and replaces the original PPM operand with the rightmost 12 bits of the difference. The result remains in the A register at the end of the operation, with the original contents purged. Indexed direct addressing uses the m field contents as a base address. The d field specifies the PPM location containing the index value which the PP adds to the base address to form the operand address.

Opcode	1057dm
Mnemonic	SOML m,d
Instruction	Replace Subtract One (m+(d)) Long

Format

```
32          41 42    47 48 51 52              63
|   1057    |    d   | 0000 |      m          |
      P                        P+1
```

Remarks This instruction subtracts one from a 16-bit PPM operand read using indexed direct addressing and replaces the original PPM operand with the rightmost 16 bits of the difference. The result remains in the A register at the end of the operation, with the original contents purged. Indexed direct addressing uses the m field contents as a base address. The d field specifies the PPM location containing the index value which the PP adds to the base address to form the operand address.

PP Branch Instructions

The branch instructions (table 1-26) allow departure from sequential instruction execution.

Table 1-26. PP Branch Instructions

Opcode	Format	Instruction	Mnemonic
0001	dm	Long jump to m+(d)	LJM
0002	dm	Return jump to m+(d)	RJM
0003	d	Unconditional jump d	UJN
0004	d	Zero jump d	ZJN
0005	d	Nonzero jump d	NJN
0006	d	Plus jump d	PJN
0007	d	Minus jump d	MJN

Long Jump

Opcode   0001dm

Mnemonic   LJM m,d

Instruction Long Jump to m+(d)

Format

32	41 42	47 48 51	52	63
0001	d	0000	m	

P   P+1
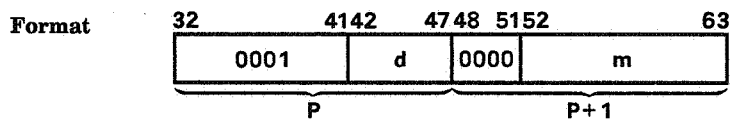
Remarks   The long jump instruction branches to the address formed by adding the m field rightmost 12 bits to the location d rightmost 12 bits. The result is the first word's address in the new instruction sequence. If d equals zero, the m field contents is the jump address.

Return Jump

Opcode   0002dm

Mnemonic   RJM m,d

Instruction Return Jump to m+(d)

Format

```
32            4142      4748 5152              63
┌──────────────┬───────┬──────┬──────────────────┐
│     0002     │   d   │ 0000 │        m         │
└──────────────┴───────┴──────┴──────────────────┘
        P                          P+1
```
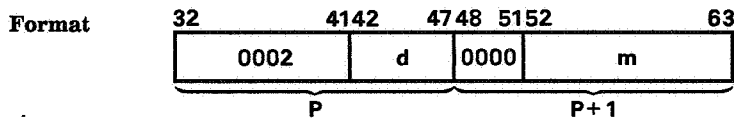
Remarks   This instruction stores the current program address plus two (P+2) in the address formed from m+(d). (If the d field is zero, the m field contents is the address.) The instruction then branches to location m+(d)+1.

This instruction interrupts the current program sequence and jumps to a subroutine while providing the means for a return to the original program. The entry point address to the subroutine forms from m+(d). The called subroutine must have a common exit point in the form of a long-jump-to-m instruction (opcode 0001) preceding the entry point (location m+(d)-1). The return jump instruction stores the current program address plus two (P+2) in the first word of the subroutine (location m+(d)). This word is the second half of the long jump instruction (its m field) and is the return address to the original program sequence. The instruction then branches to m+(d)+1. When the subroutine completes, the long jump instruction at location m+(d)-1 executes and performs a branch to the return address in the original program sequence.

Unconditional Jump

Opcode   0003d

Mnemonic   UJN d

Instruction Unconditional Jump d
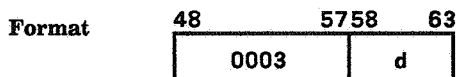
Format

```
48            5758      63
┌──────────────┬──────────┐
│     0003     │    d     │
└──────────────┴──────────┘
```

Remarks   This instruction causes an unconditional branch to any address up to 31_{10} locations forward or backward from the current program address. If d is positive (1_8 through 37_8), the jump is forward. If d is negative (40_8 through 76_8), the jump is backward.

Zero/Nonzero Jump

Opcode   **0004d**

Mnemonic ZJN d

Instruction Zero Jump d

Format

48 57	58 63
0004	d

Remarks   If the contents of A is positive zero, this instruction causes an unconditional branch to any address up to 31_{10} locations forward or backward from the current program address. The instruction adds the d value to the current program address. If d is positive (1_8 through 37_8), the jump is forward. If d is negative (40_8 through 76_8), the jump is backward.

Opcode   **0005d**

Mnemonic NJN d

Instruction Nonzero Jump d

Format

48 57	58 63
0005	d

Remarks   If the contents of A is nonzero, this instruction causes an unconditional branch to any address up to 31_{10} locations forward or backward from the current program address. If d is positive (1_8 through 37_8), the jump is forward. If d is negative (40_8 through 76_8) the jump is backward.

Plus/Minus Jump

Opcode   **0006d**

Mnemonic   PJN d

Instruction Plus Jump d

Format

48 57	58 63
0006	d

Remarks   If the contents of A is positive, this instruction causes an unconditional branch to any address up to 31_{10} locations forward or backward from the current program address. The instruction adds the d value to the current program address. If d is positive (1_8 through 37_8, the jump is forward. If d is negative (40_8 through 76_8), the jump is backward.

Opcode   **0007d**

Mnemonic   MJN d

Instruction Minus Jump d

Format

48 57	58 63
0007	d

Remarks   If the contents of A is negative, this instruction causes an unconditional branch to any address up to 31_{10} locations forward or backward from the current program address. The instruction adds the d value to the current program address. If d is positive (1_8 through 37_8), the jump is forward. If d is negative (40_8 through 76_8), the jump is backward.

PP Central Memory Access Instructions

The CM access instructions (table 1-27) provide the capability to read and write CM words to and from PPM. The PPs have read access to all CM storage locations, while the OS bounds register controls write and exchange accesses. (Refer to IOU Maintenance Registers in chapter 2 of volume 1.) The IOU performs CM addressing with real memory word addresses. To address all locations in the larger CM sizes available, the IOU uses address relocation to modify the CM address in the A register of the PP. Refer to figure 1-3. If bit 46 in A is a one during a PP CM read or write instruction, the IOU adds the R register contents to A register bits 47 through 63 to produce the CM address. If bit 46 of A is zero, the IOU does not perform address relocation but uses the A address. The R register contains an absolute 64-word starting boundary within CM. When relocation is desired, an absolute CM address forms by concatenating six zeros to the rightmost end of the R contents, and adding bits 47 through 63 of A.

Table 1-27. PP Central Memory Access Instructions

Opcode	Format	Instruction	Mnemonic
0024	d	Load R register	LRD
0025	d	Store R register	SRD
0060	d	Central read from (A) to d	CRD
1060	d	Central read from (A) to d long	CRDL
0061	dm	Central read (d) words from (A) to m	CRM
1061	dm	Central read (d) words from (A) to m long	CRML
1000	d	Central read and set lock from d to (A)	RDSL
1001	d	Central read and clear lock from d to (A)	RDCL
0062	d	Central write to (A) from d	CWD
1062	d	Central write to (A) from d long	CWDL
0063	dm	Central write (d) words to (A) from m	CWM
1063	dm	Central write (d) words to (A) from m long	CWML

NOTE

Although there is no restriction as to which PPs execute any of these instructions, those opcodes beginning with a 1 are intended for 64-bit word formats and those beginning with 0 are compatible with the C170 60-bit word.
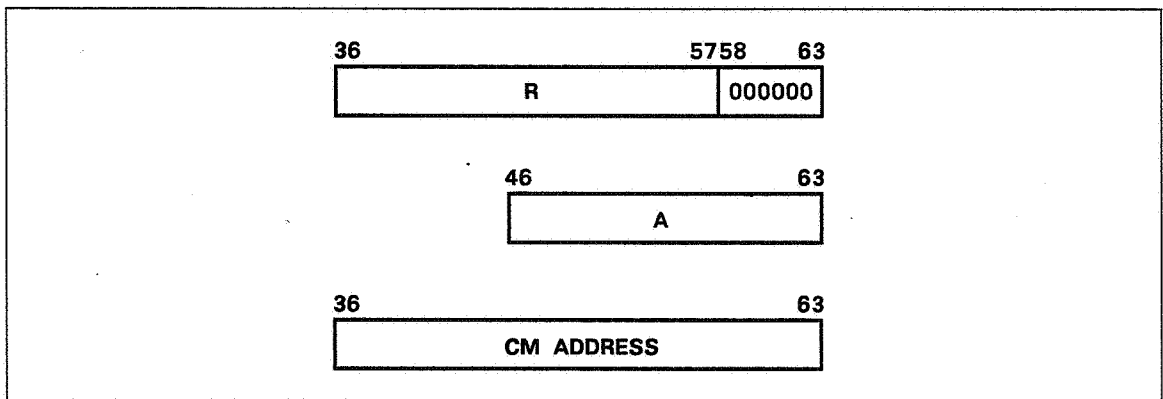
Figure 1-3. Relocation Address Formation

Central Load/Store

Opcode	**0024d**
Mnemonic	LRD d
Instruction	Load R Register

Format

Remarks   This instruction loads the 22-bit R register from PPM locations d and d+1. If the instruction d field is not zero, the instruction loads bits 47 through 57 of R from bits 52 through 63 of location (d)+1. Bits 36 through 45 of R are loaded from bits 54 through 63 of (d). If the d field is zero, this instruction is a pass.

Opcode	**0025d**
Mnemonic	SRD d
Instruction	Store R Register

Format

Remarks   This instruction stores the 22-bit R register contents into PPM locations d and d+1. If d is nonzero, the instruction stores bits 46 through 57 of R in bit positions 52 through 63 of (d)+1, and bits 36 through 45 of R store in bit positions 54 through 63 of (d). Bits 48 through 51 of (d)+1 and 48 through 53 of (d) clear. If the d field is zero, this instruction is a pass.

Central Read

Opcode   **0060d**

Mnemonic   CRD d

Instruction   Central Read from (A) to d

Format

48	57	58	63
0060		d	

Remarks   This instruction transfers the rightmost 60 bits of one CM word to the rightmost 12 bits of five consecutive PPM words. The IOU discards the leftmost 4 bits of the CM word and disassembles the remaining 60 bits from left to right into five 12-bit bytes. The following illustrations show this unpacking. $R+A$ specifies the CM word address (refer to figure 1-3); d specifies the first PPM word address.

The CM word is as follows.

0	3 4	15 16	27 28	39 40	51 52	63
(4)	a (12)	b (12)	c (12)	d (12)	e (12)	

The PPM words formed by unpacking one CM word are as follows.

48	51 52	63
0 (4)	a (12)	
0 (4)	b (12)	
0 (4)	c (12)	
0 (4)	d (12)	
0 (4)	e (12)	
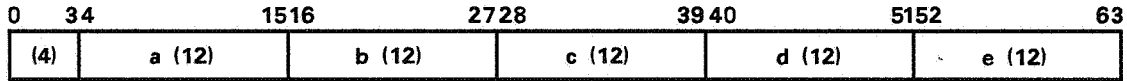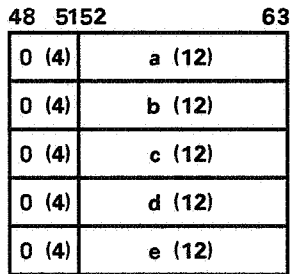
Opcode   **1060d**

Mnemonic   CRDL d

Instruction   Central Read from (A) to d Long

Format

48	57	58	63
1060		d	

Remarks   This instruction transfers one CM word to four consecutive PPM words. The IOU disassembles the CM word from the left. The following illustrations show this unpacking. R+A specifies the CM word address (refer to figure 1-3); d specifies the first PPM word address. The CM word is as follows.

0	1516	3132	4748	63
a (16)	b (16)	c (16)	d (16)	

The PPM words formed by unpacking one CM word are as follows.

48	63
a (16)	
b (16)	
c (16)	
d (16)	
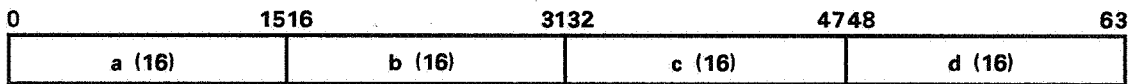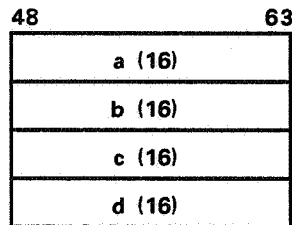
Opcode   **0061dm**

Mnemonic   CRM m,d

Instruction Central Read (d) Words from (A) to m

Format

32	4142	4748 5152	63
0061	d	0000	m

P   P + 1
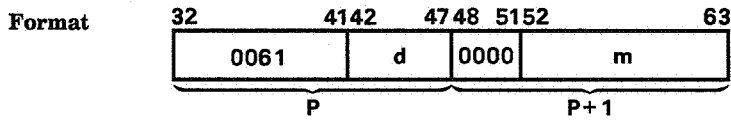
Remarks   This instruction transfers the rightmost 60 bits of consecutive CM words to the rightmost 12 bits of consecutive PPM words. The PP discards the leftmost 4 bits of each CM word and disassembles the remaining 60 bits from the left into five 12-bit bytes. Refer to instruction 0060 for an illustration of unpacking. R+A specifies the first CM word address (refer to figure 1-3), m specifies the first PPM word address, and location d specifies the number of CM words transferred. Upon completion, A contains the nonrelocated portion of the CM address (plus one) for the last word transferred.

NOTE

If the value of the rightmost 17 bits of A exceeds (2^{17})-1, the leftmost bit toggles, switching the operation from direct address to relocation address mode. If the last word transferred is from a relative address of 377776_8 and relocation is in effect, the PP clears the A register and the value returned to A may not point to the last word transferred plus one. Also, when bit 17 switches to zero, addressing switches to the direct addressing mode.

Opcode **1061dm**

Mnemonic CRML m,d

Instruction Central Read (d) Words from (A) to m Long

Format

32		41	42	47	48	51	52		63
	1061		d		0000			m	

        P   P+1
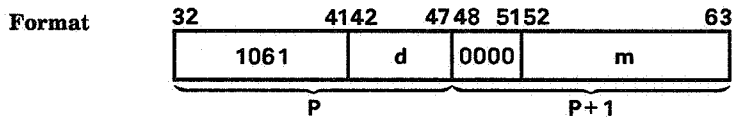
Remarks This instruction transfers consecutive CM words to consecutive PPM words. The IOU disassembles each CM word from the left. Refer to the 1060 instruction for an illustration of this unpacking. R+A specifies the first CM word address (refer to figure 1-3), m specifies the first PPM word address, and location d specifies the number of CM words transferred. On completion, A contains the nonrelocated portion of the CM address (plus one) for the last word transferred.
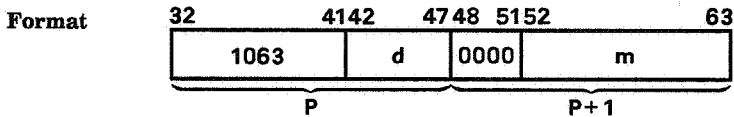
NOTE

If the value of the rightmost 17 bits of A exceeds (2^{17})-1, the leftmost bit toggles, switching the operation from direct address to relocation address mode. If the last word transferred is from a relative address of 377776_8 and relocation is in effect, the PP clears the A register and the value returned to A may not point to the last word transferred plus one. Also, when bit 17 switches to zero, addressing switches to the direct addressing mode.

Opcode **1000d**

Mnemonic RDSL d

Instruction Central Read and Set Lock from d to (A)

Format

48		57	58		63
	1000			d	
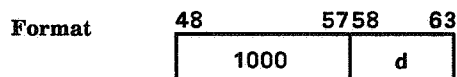
Remarks This instruction performs a logical OR function between four consecutive PPM words and one CM word. The original CM word contents replace the four PPM words; the logical OR result replaces the original CM word. Refer to the instructions 1060 and 1062 for packing and unpacking of the words. R+A specifies the CM word address (see figure 1-3); d is the first PPM word address. The IOU verifies the CM address against the OS bounds address.

The instruction does not start execution until the IOU completes all previous CM accesses. The CM does not permit any other port to access the CM word from the start of the read until the end of the write. The instruction delays subsequent instruction execution by the IOU until all CM accesses for the instruction complete.

Opcode	**1001d**
Mnemonic	RDCL d
Instruction	Central Read and Clear Lock from d to (A)

Format

48	5758	63
1001	d	

Remarks   This instruction performs a logical AND function between four consecutive
PPM words and one CM word. The original CM word contents replace the
four PPM words; the logical AND result replaces the original CM word.
See the 1060 and 1062 instructions for packing and unpacking of the
words. R+A specifies the CM word address (see figure 1-3); d is the first
PPM word address. The IOU verifies the CM address against the OS
bounds address. The instruction does not start execution until all previous
CM accesses by the IOU complete. The CM does not permit any other port
to access the CM word from the start of the read until the end of the
write. The instruction delays execution of subsequent instructions by the
IOU until all CM accesses for the instruction complete.

Central Write

Opcode   **0062d**

Mnemonic   CWD d

Instruction   Central Write to (A) from d

Format

48 57	58 63
0062	d

Remarks   This instruction transfers the rightmost 12 bits of five consecutive PPM words to the rightmost 60 bits of one CM word. (The IOU ignores the leftmost 4 bits of the words.) These words are assembled from left to right, as shown in the following illustration. R+A specifies the CM word address (see figure 1-3); d specifies the first PPM word address. The IOU verifies the CM address against the OS bounds address. PPM words are as follows.

48 51	52 63
0 (4)	a (12)
0 (4)	b (12)
0 (4)	c (12)
0 (4)	d (12)
0 (4)	e (12)

The CM word formed by packing five 12-bit PPM words is as follows.

0 3	4 15	16 27	28 39	40 51	52 63
(4)	a (12)	b (12)	c (12)	d (12)	e (12)

Opcode	**1062d**
Mnemonic	CWDL d
Instruction	Central Write to (A) from d Long

Format

```
48          5758    63
┌──────────────┬────────┐
│     1062     │   d    │
└──────────────┴────────┘
```

Remarks This instruction transfers four consecutive PPM words to one CM word, as shown in the following illustration. R+A specifies the CM word address (see figure 1-3); d specifies the first PPM word address. The IOU verifies the CM address against the OS bounds address. The PPM words are as follows.

```
48                    63
┌────────────────────────┐
│         a (16)         │
├────────────────────────┤
│         b (16)         │
├────────────────────────┤
│         c (16)         │
├────────────────────────┤
│         d (16)         │
└────────────────────────┘
```
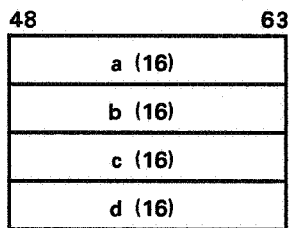
The CM word formed by packing four 16-bit PPM words is as follows.

```
0            1516        3132        4748          63
┌────────────┬────────────┬────────────┬────────────┐
│   a (16)   │   b (16)   │   c (16)   │   d (16)   │
└────────────┴────────────┴────────────┴────────────┘
```
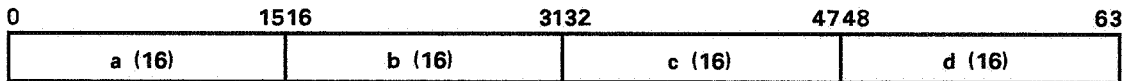
PP Input/Output Instructions

The 26 instructions (table 1-28) direct activity on the I/O channels. They select an external device and transfer data to or from that device. The instructions also determine whether a channel or external device is available and ready to transfer data. The preparatory steps ensure that the channels carry out an orderly data transfer. Each external device has a set of external function codes the PP uses to establish operation modes, and to start and stop data transfer. The devices can also detect certain errors which are indicated to the controlling PP.

Table 1-28. PP Input/Output Instructions

Opcode	Format	Instruction	Mnemonic
00640	cm	Jump to m if channel c active	AJM
1064x[1]	cm	Jump to m if channel c flag set	FSJM
00650	cm	Jump to m if channel c inactive	IJM
1065x[1]	scm	Jump to m if channel c flag clear	FCJM
00660	cm	Jump to m if channel c full	FJM
00670	cm	Jump to m if channel c empty	EJM
00641	cm	Test and set channel c flag	SCF
00651	cm	Clear channel c flag	CCF
00661	cm	Test and clear channel c error flag set	SFM
00671	cm	Test and clear channel c error flag clear	CFM
00700	c	Input to A from channel c when channel active and full	IAN
00701	c	Input to A from channel c if channel active and full	IAN
0071x[1]	cm	Input A words to m from channel c	IAM
1071x[1]	cm	Input A words to m from channel c packed	IAPM

1. x = don't care, but will be an assumed 0 in C170 mode.

(Continued)

Table 1-28.   PP Input/Output Instructions *(Continued)*

Opcode	Format	Instruction	Mnemonic
00720	c	Output from A to channel c when active	OAN
00721	c	Output from A on channel c if active	OAN
0073x[1]	cm	Output A words from m to channel c	OAM
1073x[1]	cm	Output A words from m to channel c packed	OAPM
00740	c	Activate channel c when inactive	ACN
00741	c	Unconditionally activate channel c	ACN
00750	c	Deactivate channel c when active	DCN
00751	c	Unconditionally deactivate channel c	DCN
00760	c	Function A on channel c when inactive	FAN
00761	c	Function A on channel c if inactive	FAN
00770	cm	Function m on channel c when inactive	FNC
00771	cm	Function m on channel c if inactive	FNC

1. x = don't care, but will be an assumed 0 in C170 mode.

Opcode	**0063dm**
Mnemonic	CWM m,d
Instruction	Central Write (d) Words to (A) from m
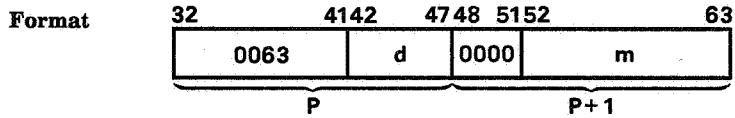
Format

32	4142	4748	5152	63
0063	d	0000	m	

P   P+1

Remarks   This instruction transfers the rightmost 12 bits of consecutive PPM words to the rightmost 60 bits of consecutive CM words. Refer to the instruction 0062 for an illustration of this packing. R+A specifies the first CM word address (see figure 1-3), m specifies the first PPM word address, and d specifies the number of CM words transferred. The IOU verifies each CM address against the OS bounds address. On completion, A contains the nonrelocated portion of the CM address (plus 1) for the last word transferred.

NOTE

If the value of the rightmost 17 bits of A exceeds (2^{17})-1, the leftmost bit toggles, switching the operation from direct address to relocation address mode. If the last word transferred is from a relative address of 377776_8 and relocation is in effect, the PP clears the A register and the value returned to A may not point to the last word transferred plus one. Also, when bit 17 switches to zero, addressing switches to the direct addressing mode.

Opcode	**1063dm**
Mnemonic	CWML m,d
Instruction	Central Write (d) Words to (A) from m Long
Format	

```
32          4142    4748 5152           63
|    1063    |   d  |0000|      m        |
      P              P+1
```

Remarks This instruction transfers consecutive PPM words to consecutive CM words. The IOU packs four PPM words (from the left) into each CM word. R+A specifies the first CM word address (see figure 1-3), m specifies the first PPM word address, and d specifies the number of CM words transferred. The IOU verifies each CM address against the OS bounds address. On completion, A contains the nonrelocated portion of the CM address (plus 1), for the last word transferred.

NOTE

If the value of the rightmost 17 bits of A exceeds (2^{17})-1, the leftmost bit toggles, switching the operation from direct address to relocation address mode. If the last word transferred is from a relative address of 377776_8 and relocation is in effect, the PP clears the A register and the value returned to A may not point to the last word transferred plus 1. Also, when bit 17 switches to zero, addressing switches to the direct addressing mode.

Jump

Opcode	**00640cm**
Mnemonic	AJM m,c
Instruction	Jump to m if Channel c Active

Format

```
32          414243 4748 5152              63
 _____
|        |  |    |     |                    |
|  0064  |0|  c  |0000 |         m          |
|_____|__|____|_____|_____|
     _____/_____/
            P                   P+1
```

Remarks   This instruction branches to the location specified by m if channel c is active.

Opcode	**1064xcm**
Mnemonic	FSJM m,c
Instruction	Jump to m if Channel c Flag Set

Format

```
32          414243  4748 5152             63
 _____
|        |  |    |     |                    |
|  1064  |x|  c  |0000 |         m          |
|_____|__|____|_____|_____|
     _____/_____/
            P                   P+1
```

Remarks   This instruction branches to the location specified by m if the channel c flag is set.

Opcode	**00650cm**
Mnemonic	IJM m,c
Instruction	Jump to m if Channel c Inactive

Format

```
32           414243  4748 5152            63
 _____
|        |  |    |     |                    |
|  0065  |0|  c  |0000 |         m          |
|_____|__|____|_____|_____|
     _____/_____/
            P                   P+1
```

Remarks   This instruction branches to the location specified by m if channel c is inactive.

Opcode   **1065xcm**

Mnemonic   FCJM m,c

Instruction Jump to m if Channel c Flag Clear

Format

32		4142	43	47	48	5152		63
1065		x	c		0000	m		

  P   P+1
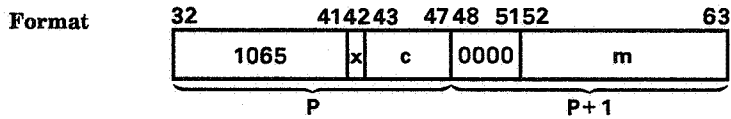
Remarks   This instruction branches to the location specified by m if the channel c
flag is clear.

Opcode   **00660cm**

Mnemonic   FJM m,c

Instruction Jump to m if Channel c Full

Format

32		4142	43	47	48	5152		63
0066		0	c		0000	m		

  P   P+1
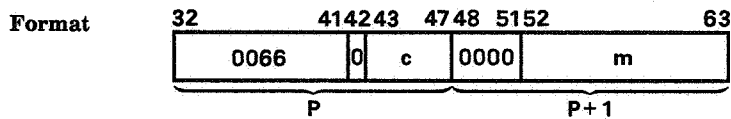
Remarks   This instruction branches to the location specified by m if channel c is full.

Opcode   **00670cm**

Mnemonic   EJM m,c

Instruction Jump to m if Channel c Empty

Format

32		4142	43	47	48	5152		63
0067		0	c		0000	m		

  P   P+1
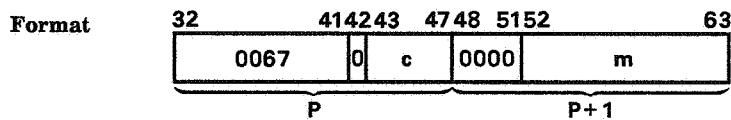
Remarks   This instruction branches to the location specified by m if channel c is
empty.

Test/Clear

Opcode	**00641cm**
Mnemonic	SCF m,40B+c
Instruction	Test and set Channel c Flag
Format	

```
32          414243 4748 5152              63
 |   0064    |1| c |0000|       m         |
  _____/    _____/
      P                  P+1
```

Remarks   This instruction branches to the location specified by m if the channel c flag is set. Otherwise, it sets the channel flag and exits. Setting m to P+2 unconditionally sets the channel flag.

NOTE

A conflict condition may occur when two or more PPs simultaneously issue this instruction for the same channel. If that channel is channel 17_8, this condition is resolved by letting the PP in the lowest physical barrel see the true status of the flag. The flag appears set to the other PPs in conflict and those PPs take the branch. For all other channels, the PPs sense the true state of the flag.

Opcode	**00651cm**
Mnemonic	CCF c
Instruction	Clear Channel c Flag
Format	

```
32          414243 4748 5152              63
 |   0065    |1| c |0000|       m         |
  _____/    _____/
      P                  P+1
```

Remarks   This instruction clears the channel c flag and requires, but does not use, the m field.

Opcode	**00661cm**
Mnemonic	SFM m,40B+c
Instruction	Test and clear Channel c Error Flag Set
Format	

```
32          414243 4748 5152              63
 |   0066    |1| c |0000|       m         |
  _____/    _____/
      P                  P+1
```

Remarks   If the channel c error flag is set, this instruction branches to the location specified by m and clears the error flag.

Opcode	00671cm

Mnemonic	CFM m,40B+c

Instruction	Test and clear Channel c Error Flag Clear

Format

32		41	42	43	47	48	51	52	63
	0067		1	c		0000		m	

P   P+1

Remarks	If the channel c error flag is clear, this instruction branches to the location specified by m. If the error flag is set, this instruction clears it.

Input

Opcode	00700c

Mnemonic	IAN c

Instruction	Input to A from Channel c when Active and Full

Format

48		57	58	59	63
	0070		0	c	

Remarks	This instruction transfers one word from channel c to the low-order 16 bits of A. The high-order 2 bits of A are zeros. The instruction waits for the channel to become active and full therefore executing.

NOTE

If the channel uses a 12-bit external interface, the high-order 6 bits of A are zeros. If it uses an 8-bit external interface, the high-order 10 bits of A are zeros.

Opcode	00701c

Mnemonic	IAN 40B+c

Instruction	Input to A from Channel c if Active and Full

Format

48		57	58	59	63
	0070		1	c	

Remarks	This instruction transfers one channel c word to the low-order 16 bits of A. The high-order 2 bits of A are zeros. If the channel is inactive or becomes inactive before becoming full, no transfer occurs and the instruction exits with A = 0.

NOTE

If the channel uses a 12-bit external interface, the high-order 6 bits of A are zeros. If it uses an 8-bit external interface, the high-order 10 bits of A are zeros. If the addressed channel is not connected, the instruction exits with A = 177777_8.

Opcode	**0071xcm**
Mnemonic	IAM m,c
Instruction	Input A Words to m from Channel c
Format	

```
32          4142 43   47 48 51 52           63
  ┌──────────┬─┬────┬──────┬────────────────┐
  │  0071    │x│ c  │0000  │      m         │
  └──────────┴─┴────┴──────┴────────────────┘
        P                      P+1
```
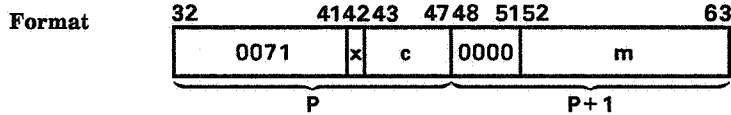
Remarks   This instruction transfers successive words from channel c to PPM. The m field specifies the first PPM word address; A specifies the number of words transferred.

The transfer completes when either A = 0 or the channel becomes inactive. If an inactive channel caused termination, the instruction clears the next PPM word, and A contains the difference between its initial value and the number of channel words actually transferred.

No transfer takes place if the instruction executes with the channel initially inactive. The instruction exits with A unchanged and the PPM word specified by m sets to zero. However, if the addressed channel is not connected, the instruction exits with A = 177777_8.

NOTE

If the channel uses a 12-bit external interface, the high-order 4 bits of the PPM word are zeros. If it uses an 8-bit external interface, the high-order 8 bits of the PPM word are zeros.

Opcode	**1071xcm**
Mnemonic	IAPM m,c
Instruction	Input A Words to m from Channel c Packed
Format	

```
32            41424 3   4748 5152          63
┌──────────────┬─┬────┬────┬──────────────┐
│   1071       │x│ c  │0000│     m        │
└──────────────┴─┴────┴────┴──────────────┘
        P                    P+1
```
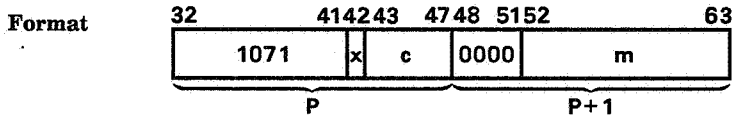
Remarks This instruction transfers the low-order 12 bits of successive channel c words to consecutive PPM words. During this transfer, the IOU packs four 12-bit words (48 bits) into three PPM words. (Refer to the following paragraphs.) The high-order 4 bits of the channel words are ignored. The m field specifies the first PPM word address; A specifies the number of channel words transferred.

A complete transfer depends on the channel word count being a multiple of 4. If the channel word count is not a multiple of 4, the IOU fills the bits left over when A is counted to zero (these bits copy the corresponding bits on the channel). When the channel is inactive or empty, these bits are zeros and, hence, the fill is with zeros. When, however, the external device and the PP have different word counts, or for some other reason the channel bits are nonzero, the fill is not zero-fill.

The instruction exits when A is zero or when the channel becomes inactive. If an inactive channel causes termination, the leftover bits from the previous channel word will fill up to the next four-channel-word boundary as described above.

No transfer takes place if the instruction executes with the channel initially inactive. The instruction exits with A unchanged, and the next three PPM words specified by m, m+1, and m+2 fill as described above.

This instruction allows 16-bit PPM words to be loaded from 12-bit external devices. The channel words are as follows.

```
48  5152              63
┌────┬─────────────────┐
│(4) │    a (12)        │
├────┼──────┬──────────┤
│(4) │b (4) │  c (8)    │
├────┼──────┴─────┬────┤
│(4) │   d (8)    │e (4)│
├────┼────────────┴────┤
│(4) │    f (12)        │
└────┴─────────────────┘
```

The PPM words are as follows.

```
48  5152 5556 5960 63
┌─────────────┬──────┐
│   a (12)    │ b (4)│
├──────┬──────┴──────┤
│ c (8)│   d (8)     │
├──────┼─────────────┤
│e (4) │   f (12)    │
└──────┴─────────────┘
```
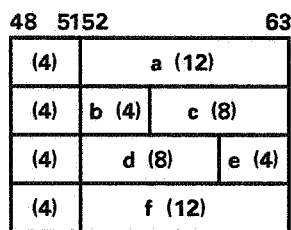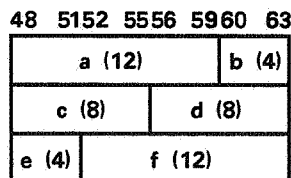
Output

Opcode	00720c
Mnemonic	OAN c
Instruction	Output from A on Channel c when Active
Format	

```
48          575859  63
┌─────────────┬─┬─────┐
│    0072     │0│  c  │
└─────────────┴─┴─────┘
```

Remarks   This instruction transfers one word from the A register low-order 12 bits to channel c. The instruction waits for an active and empty channel before executing. The A register contents are unaltered.

NOTE

If the channel uses a 12-bit interface, it does not transmit the channel word high-order 4 bits to the external device. Similarly, if it uses an 8-bit external interface, it does not transmit the channel word high-order 8 bits.

Opcode	00721c
Mnemonic	OAN 40B+c
Instruction	Output from A on Channel c if Active
Format	

```
48          575859  63
┌─────────────┬─┬─────┐
│    0072     │1│  c  │
└─────────────┴─┴─────┘
```

Remarks   This instruction transfers the A register low-order 16 bits to channel c. If the channel is inactive, no transfer occurs and the instruction exits. The operation does not alter the contents of A.

NOTE

If the channel uses a 12-bit interface, it does not transmit the channel word high-order 4 bits to the external device. Similarly, if it uses an 8-bit external interface, it does not transmit the channel word high-order 8 bits.

Opcode   0073xcm

Mnemonic   OAPM m,c

Instruction Output A Words from m on Channel c

Format

```
32           414243  4748 5152              63
┌───────────┬──┬────┬──────┬──────────────────┐
│   0073    │x │ c  │ 0000 │        m         │
└───────────┴──┴────┴──────┴──────────────────┘
└─────────────────┘ └──────────────────────────┘
         P                     P+1
```
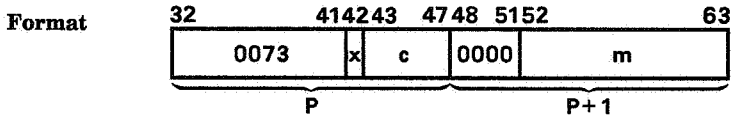
Remarks   This instruction transfers the contents of successive PPM words as
successive words on channel c. The m field specifies the first PPM word
address; A specifies the number of words to be transferred. The transfer
completes when either A = 0 or the channel becomes inactive. If an
inactive channel caused termination, A contains the difference between its
initial value and the number of words transferred on the channel. If the
instruction executes with the channel initially inactive, no transfer occurs
and the instruction exits with A unchanged.

NOTE

If the channel uses a 12-bit interface, it does not transmit the channel
word high-order 4 bits to the external device. Similarly, if it uses an 8-bit
external interface, it does not transmit the channel word high-order 8 bits.

Opcode   1073xcm

Mnemonic   OAPM m,c

Instruction Output A Words from m on Channel c Packed

Format

```
32           414243  4748 5152              63
┌───────────┬──┬────┬──────┬──────────────────┐
│   1073    │x │ c  │ 0000 │        m         │
└───────────┴──┴────┴──────┴──────────────────┘
└─────────────────┘ └──────────────────────────┘
         P                     P+1
```
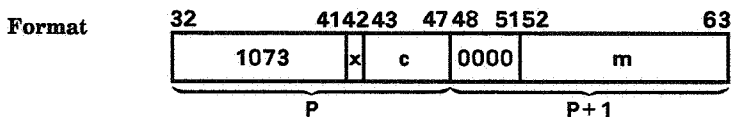
Remarks   This instruction transfers consecutive PPM words as the low-order 12 bits
of successive words on channel c. During the transfer, processing occurs
such that the contents of three PPM words result in four channel words.
The high-order 4 bits of the 16-bit channel words set to zeros. This
packing is illustrated in the 1071 instruction. The m field specifies the
first PP word address; A specifies the number of channel words to be
transferred. The transfer completes when either A = 0 or the channel
becomes inactive. If an inactive channel caused termination, A contains the
difference between its initial value and the number of words actually
transferred on the channel.

If the instruction executes with the channel initially inactive, no transfer
occurs and the instruction exits with A unchanged.

Activate/Deactivate

Opcode   00740c

Mnemonic   ACN c

Instruction Activate Channel c

Format

48	57	58	59	63
0074		0	c	

Remarks   This instruction sets channel c active to prepare it for I/O transfer operations. If the channel is initially active, the instruction waits for the channel to become inactive before executing.

Opcode   00741c

Mnemonic   ACN 40B+c

Instruction Unconditionally Activate Channel c

Format

48	57	58	59	63
0074		1	c	

Remarks   This instruction sets channel c active to prepare it for I/O transfer operations. The instruction executes regardless of the channel's active/inactive status. The active out pulse is sent to the external device only if the channel is inactive at the time of execution.

Opcode	**00750c**
Mnemonic	DCN c
Instruction	Deactivate Channel c
Format	

```
48          575859  63
+---------+--+-----+
|  0075   |0|   c  |
+---------+--+-----+
```

Remarks   This instruction sets channel c inactive to terminate I/O operations on the channel. If the channel is initially inactive, the instruction waits for the channel to become active before executing.

If this instruction executes after an output instruction without waiting for the channel to become empty, the last channel word transferred may be lost.

Opcode	**00751c**
Mnemonic	DCN 40B+c
Instruction	Unconditionally Deactivate Channel c
Format	

```
48          575859  63
+---------+--+-----+
|  0075   |1|   c  |
+---------+--+-----+
```

Remarks   This instruction sets channel c inactive to terminate I/O operations on the channel. If the channel is initially inactive, the instruction executes regardless of the channel's active/inactive state. If this instruction executes after an output instruction without waiting for the channel to become empty, the last channel word transferred may be lost. The deactive pulse is sent to the external device only if the channel is active at the time of execution.

Function

Opcode	**00760c**
Mnemonic	FAN c
Instruction	Function A on Channel c when Inactive
Format	

```
48            575859   63
┌──────────┬─┬────────┐
│  0076    │0│   c    │
└──────────┴─┴────────┘
```

Remarks   This instruction transfers the low-order 16 bits of A to channel c as a function code. If the channel is initially active, the instruction waits for the channel to become inactive before executing. The operation does not alter the contents of A.

NOTE

If the channel uses a 12-bit interface, it does not transmit the channel word high-order 4 bits to the external device. Similarly, if it uses an 8-bit external interface, it does not transmit the channel word high-order 8 bits. Parity, however, is always calculated on the rightmost 16 bits of A when outputting a function word from A.

Opcode	**00761c**
Mnemonic	FAN 40B+c
Instruction	Function A on Channel c if Inactive
Format	

```
48            575859   63
┌──────────┬─┬────────┐
│  0076    │1│   c    │
└──────────┴─┴────────┘
```

Remarks   This instruction transfers the low-order 16 bits of A to channel c as a function code. If the channel is initially active, the IOU does not transfer the function on the channel, and the instruction exits. The status of the channel is not changed.

NOTE

If the channel uses a 12-bit interface, it does not transmit the channel word high-order 4 bits to the external device. Similarly, if it uses an 8-bit external interface, it does not transmit the channel word high-order 8 bits.

Opcode	**00770cm**
Mnemonic	FNC m,c
Instruction	Function m on Channel c when Inactive
Format	
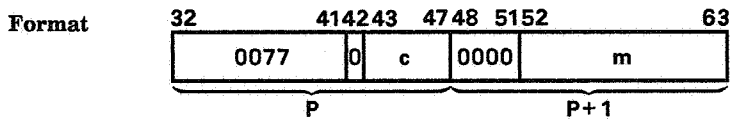
```
 32          414243  4748 5152              63
 ┌──────────────┬─┬────┬────┬──────────────────┐
 │    0077      │0│ c  │0000│       m          │
 └──────────────┴─┴────┴────┴──────────────────┘
        └──────── P ────────┘└───── P+1 ─────┘
```

Remarks   This instruction transfers the m field contents to channel c as a function code. If the channel is initially active, the instruction waits for the channel to become inactive before executing.

NOTE

If the channel uses a 12-bit interface, it does not transmit the channel word high-order 4 bits to the external device. Similarly, if it uses an 8-bit external interface, it does not transmit the channel word high-order 8 bits.

Opcode	**00771cm**
Mnemonic	FNC m,40B+c
Instruction	Function m on Channel c if Inactive
Format	
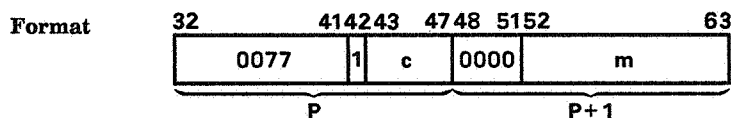
```
 32          414243  4748 5152              63
 ┌──────────────┬─┬────┬────┬──────────────────┐
 │    0077      │1│ c  │0000│       m          │
 └──────────────┴─┴────┴────┴──────────────────┘
        └──────── P ────────┘└───── P+1 ─────┘
```

Remarks   This instruction transfers the m field contents to channel c as a function code. If the channel is initially active, the IOU does not transfer the function on the channel and the instruction exits. The status of the channel is not changed.

NOTE

If the channel uses a 12-bit interface, it does not transmit the channel word high-order 4 bits to the external device. Similarly, if it uses an 8-bit external interface, it does not transmit the channel word high-order 8 bits.

Other IOU Instructions

Pass

The following opcodes are implemented as pass instructions and perform no operation. The mnemonic for a pass is PSN.

0000	1066
1002	1067
:	1070
:	1072
:	1074
1021	1075
1025	1076
1027	1077

The following instructions are also considered pass instructions due to the value of d.

002400

002500

Opcode	**0027**
Mnemonic	KPT d
Instruction	PP Keypoint
Format	

```
48          57 58      63
 _____
|    0027    |//////////|
 ------------------------
```

Remarks	This instruction executes as a pass instruction, but allows test-point sensing of its execution by way of external monitoring equipment.

Exchange Jumps

The exchange jump instructions allow PP programs to request CYBER 170 State exchanges in the CP. The IOU transmits the exchange request to the CP designated for executing CYBER 170 State instructions. Bit 05 of the MCR in the NIO sets to indicate an outstanding IOU exchange request. Bit 53 in the CIO sets to indicate an outstanding IOU exchange request.

If an exchange request for any PP is outstanding, another exchange request from any other PP causes the second PP to wait until the outstanding exchange request completes.

The d field value controls the action taken to process the exchange request in CYBER 170 State, as described in the following paragraphs.

Opcode   **002600**

Mnemonic   EXN d

Instruction Exchange Jump

Format

48	57	58	63
0026		0x	

Remarks   This is an unconditional exchange jump performed with the exchange package at address R+A (see table 1-22). The IOU verifies this address against the OS bounds address and, if in the prohibited region, the exchange does not occur. In this case, the OS bounds fault sets and, if the enable error stop is set in the environment control register, the PP is idled.

Opcode   **002610**

Mnemonic   MXN d

Instruction Monitor Exchange Jump

Format

48	57	58	63
0026		10	

Remarks   This is a conditional exchange jump performed with the exchange package at address R+A (see figure 1-3). The IOU verifies this address against the OS bounds address and, if in the prohibited region, the exchange does not occur. The OS bounds fault sets and, if the enable error stop is set in the environment control register, the PP is idled. If monitor flag is clear, the exchange jump occurs and the monitor flag sets. If the monitor flag is set before this instruction begins to execute, the exchange jump is not performed.

Opcode	002620
Mnemonic	MAN d
Instruction	Monitor Exchange Jump to MA

Format

48	57	58	63
0026		20	

Remarks   This is a conditional exchange jump performed with the exchange package at the address in the CP monitor address register. If the monitor flag is clear, the exchange jump occurs and the monitor flag sets. If the monitor flag is set before this instruction begins to execute, the exchange jump is not performed.

Opcode	00263x
Mnemonic	(None)
Instruction	Executes as if d = 2x.
Opcode	1026d
Mnemonic	INPN d
Instruction	Interrupt Processor

Format

48	57	58	63
1026		d	

Remarks   This instruction causes the IOU to transmit an interrupt for a CP on the memory port specified by d. This interrupt signal causes the external interrupt bit to set in the MCR. Refer to CP Interrupts in chapter 2 of this manual.

Programming Information   2

Programming Information   2

This chapter contains programming information for:

● The central processor (CP) and the central memory (CM), both of which are contained in the central processing unit (CPU)

● The input/output unit (IOU)

● The system console

● The two-port multiplexer (MUX)

CP Exchange Operations

Figure 2-1 shows CP modes of operation. Exchange operations switch the CP between monitor and job modes in both Virtual State and CYBER 170 State. Exchange operations may also switch states while switching modes. Refer to Interstate Programming Information in this chapter for a description of state-switching operations.

An exchange operation exchanges the process running in CP with another process and switches CP modes. The exchange stores the CP registers of the outgoing process in CM as an exchange package (refer to Exchange Packages, figure 2-2), and reads the registers of the incoming process from another exchange package in CM into the CP registers. Exchange operations are caused by the following.

- Execution of a Virtual State exchange instruction in the CP

- Execution of the interrupt processor (0026) instruction in any peripheral processor (PP)

- Hardware-detected fault or exception with the CP in Virtual State job mode; such exchange operations are called exchange interrupts
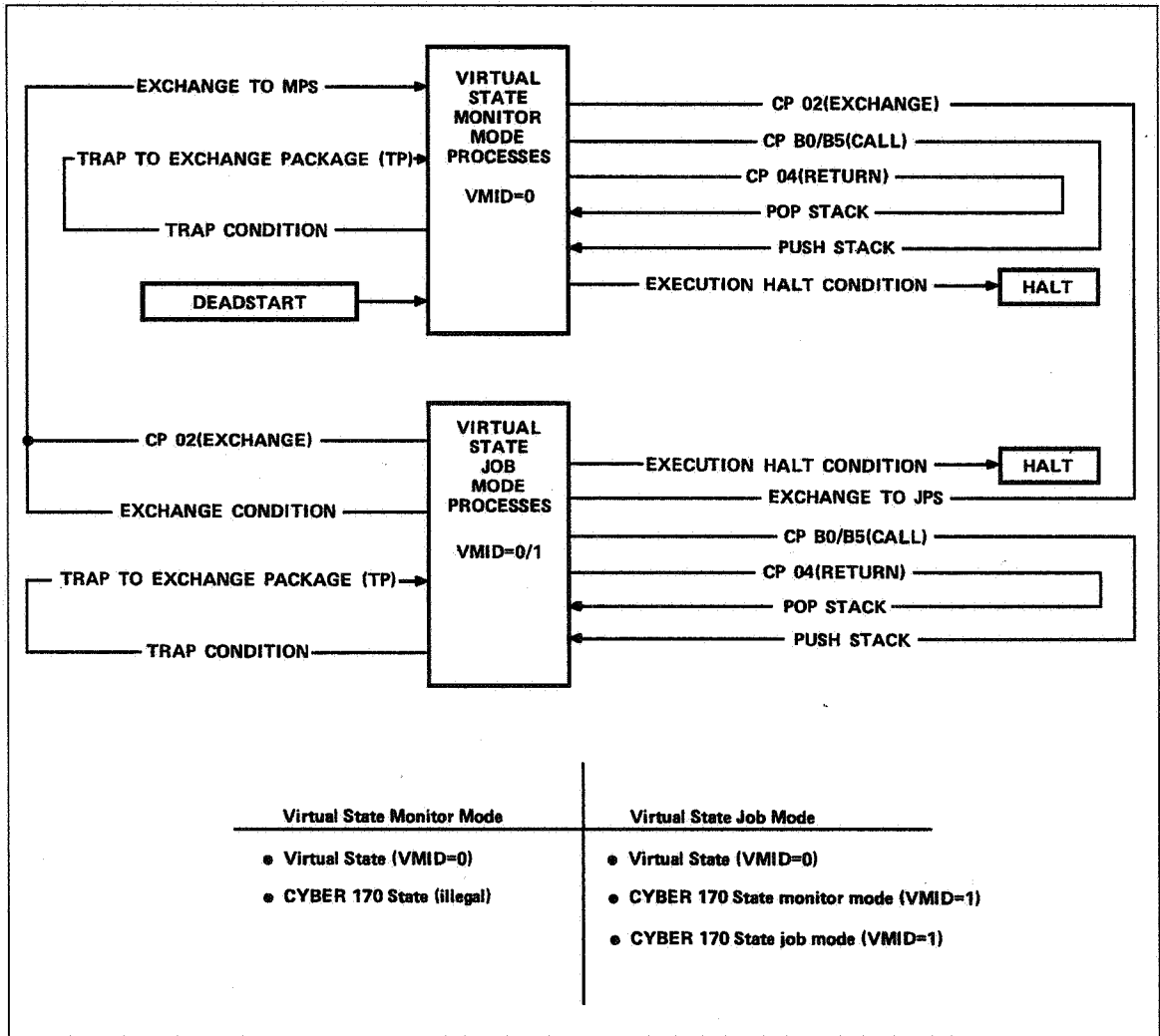
Figure 2-1. CP Calls, Returns, and Interrupts

Virtual State Job-to-Monitor Exchange Operations

The CP performs an exchange from Virtual State job mode to Virtual State monitor mode as follows.

1. It stores the outgoing job process registers as an exchange package starting at the CM address in the job process state (JPS) pointer register.

2. It disables exchange interrupts.

3. It loads the incoming monitor process registers (from another exchange package in CM) into the CP registers, starting at the CM address in the monitor process state (MPS) pointer register.

Exchange interrupt conditions occurring with the CP in monitor mode do not cause an exchange interrupt, but may cause a trap interrupt. Refer to CP Interrupts in this chapter.

Virtual State Monitor-to-Job Exchange Operations

The CP performs an exchange from Virtual State monitor mode to Virtual State job mode as follows.

1. It stores the outgoing monitor process registers as an exchange package starting at the CM address in the MPS pointer register.

2. It enables exchange interrupts.

3. It loads the incoming job process registers (stored in CM as another exchange package into the CP registers) starting at the CM address in the JPS pointer register.

When a Virtual State monitor-to-job mode exchange operation sets MCR bit 55 (environment specification error), the CP completes the exchange and initiates a job-to-monitor mode exchange in response to the error. Refer to CP Interrupts in this chapter.

Exchange Packages

Before initiating a Virtual State monitor-to-job exchange, the operating system specifies the process environment by composing in CM an exchange package for that process. When the process is suspended, hardware records system conditions into the same exchange package in CM, permitting process reactivation (in the absence of uncorrectable errors) without violating process integrity.

Each suspended process (including the monitor program) has one exchange package stored in CM. Each exchange package contains the process registers in fifty-two 64-bit words as shown in figure 2-2. Refer to Process State Registers in this chapter for descriptions of the exchange package entries.

VIRTUAL STATE EXCHANGE PACKAGE

BYTE(HEX)　　　　　　　　　　　　　　　　　　　　　　WORD(DEC)

BYTE(HEX)	00	07,08	15,16		63	WORD(DEC)
0			P			0
8	VMID	UVMID	A0			1
10	Flags	Trap Enables	A1			2
18	User Mask		A2			3
20	Monitor Mask		A3			4
28	User Condition		A4			5
30	Monitor Condition		A5			6
38	Reserved	LPID	A6			7
40	Reserved		A7			8
48	Reserved		A8			9
50			A9			10
58	Process Int. Timer		AA			11
60			AB			12
68	Base Constant		AC			13
70			AD			14
78	Model Dependent Flags		AE			15
80	Segment Table Length		AF			16
88			X0			17
90			X1			18
C0						24
C8			X8			25
D0			X9			26
D8			XA			27
E0			XB			28
E8			XC			29
F0			XD			30
F8			XE			31
100			XF			32
108			Model Dependent Word			33
110	Segment Table Address		Untranslatable Pointer			34
118			Trap Pointer			35
120	Debug Index	Debug Mask	Debug List Pointer			36
128	Largest Ring Number		Top of Stack Ring Number 1			37
198			Top of Stack Ring Number 15			51

00　07 08　15 16　　　　　　63
r

Figure 2-2. Virtual State Exchange Package

The CP uses the following types of exchange packages.

● For exchanges between Virtual State monitor and job modes (exclusively within Virtual State), the exchange package format is shown in figure 2-2 would hold the detail of the selected words shown in figure 2-3.

● For exchanges between Virtual State monitor and job modes (including switching between Virtual State and CYBER 170 State), the exchange package format is shown in figure 2-4.

● For exchanges with CYBER 170 State monitor and job modes, the exchange package format is shown in figure 2-5.

● For exchanges exclusively within CYBER 170 State, the exchange package format is shown in the CYBER 170 State Hardware Reference Manual shown in About This Manual.
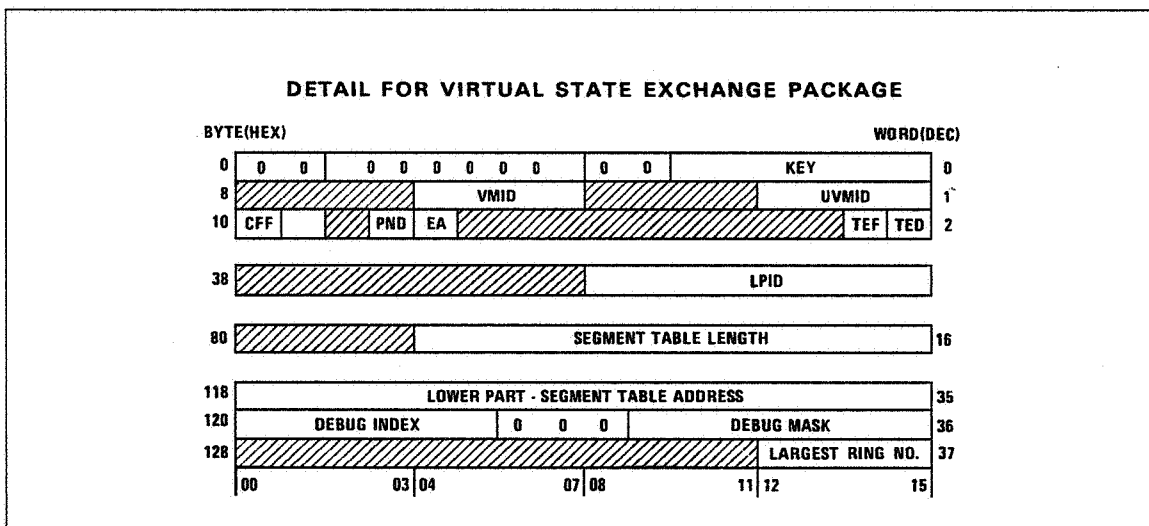
Figure 2-3. Virtual State Exchange Package Detail

BYTE(HEX)   WORD(DEC)

00   03 04 07 08 11 12 15 16 19 20 22 23   28 29 31 32   45 46   57 58   63

BYTE(HEX)					WORD(DEC)
0				C170 P + RAC	0
8	VMID	UVMID		Virtual State	A0	1
10	Flags	Trap Enables		Virtual State	A1	2
18	User Mask			Virtual State	A2	3
20	Monitor Mask		EM	Flags*	EM*	RAC	4
28	User Condition		*	FLC	5
30	Monitor Condition	R	*	MA	6
38	Reserved	LPID	I N	RAE	00←→00	7
40	Reserved	G	FLE	00←→00	8
48	Reserved	N U M B E R S	A0	9
50		A1	10
58	Process Int. Timer	A2	11
60		A3	12
68	Base Constant	A4	13
70		A5	14
78	Model Dependent Flags	A6	15
80	Segment Table Length	A7	16
88		17
90		B1	18
C0		B7	24
C8	s i g n e x t e n s i o n	X0	25
D0		X1	26
D8		X2	27
E0		X3	28
E8		X4	29
F0		X5	30
F8		X6	31
100		X7	32
108	Model Dependent Word	33
110	Segment Table Address	Untranslatable Pointer	34
118		Trap Pointer	35
120	Debug Index	Debug Mask	Debug List Pointer	36
128	LRN	Top of Stack Ring Number 1	37
198		Top of Stack Ring Number 15	51

00   03 04 07 08   11 12 15 16   45 46   63

* Flags:

Bit		
Bit 23, Word 4	UEM Enable Flag
Bit 24, Word 4	Expanded Addressing Select Flag (ESM Mode Flag)
Bit 25, Word 4	Extended Block Copy Flag
Bit 26, Word 4	Software Flag
Bit 27, Word 4	Instruction Stack Purge Flag
Bit 28, Word 4	Software Flag
Bit 29, Word 4	EM-Indefinite Operand
Bit 30, Word 4	EM-Infinite Operand
Bit 31, Word 4	EM-Address out of Range
Bit 31, Word 5	C170 Monitor Flag
Bit 31, Word 6	Exit Mode Halt

Figure 2-4. Interstate Exchange Package

C170 BIT:				WORD

Figure showing CYBER 170 State Exchange Package layout:

- Word 0: P, A0
- Word 1: RAC, A1, B1
- Word 2: FLC, A2, B2
- Word 3: EM Flags* EM, A3, B3
- Word 4: RAE, A4, B4
- Word 5: FLE, A5, B5
- Word 6: MA, A6, B6
- Word 7: A7, B7
- Word 8: X0
- Word 9: X1
- Word 10: X2
- Word 11: X3
- Word 12: X4
- Word 13: X5
- Word 14: X6
- Word 15: X7

Bit positions: 59 56 53 50 47 36 35 18 17 0

* REFER TO FLAG BIT DESCRIPTIONS ON PRECEDING PAGES.
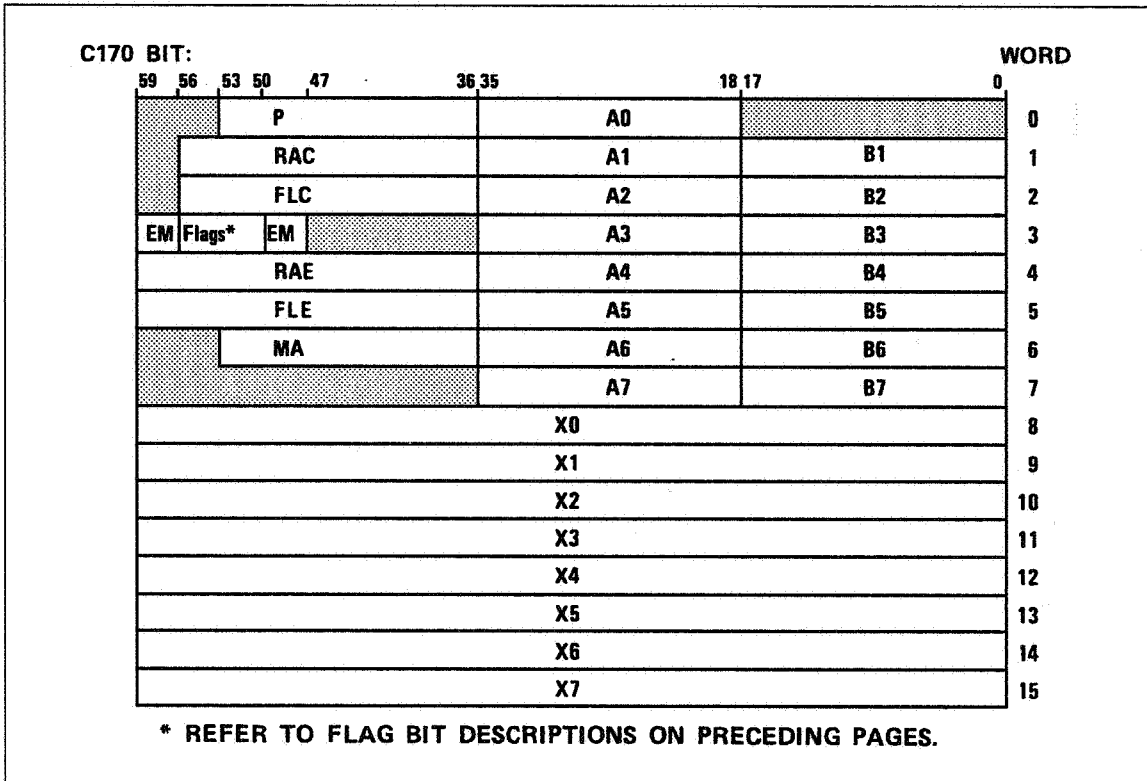
Figure 2-5. CYBER 170 State Exchange Package

Exchange Package Addresses

Exchange package addresses, used by hardware to locate exchange packages during exchange operations, are real memory addresses (RMAs) in hardware registers designated as follows.

● For exchanges to Virtual State monitor mode, the RMA is in the MPS pointer register.

● For exchanges to Virtual State job mode, the RMA is in the JPS pointer register.

Exchange operations do not copy an exchange package into cache memory.

CP Registers

The CP registers comprise two classes: process state registers and processor state registers. This distinction arises because the state of the process, and the state of the processor, characterize CP operation. Both classes of registers are accessible to the CP and PPs. .

CP Process State Registers

The process state registers relate to a specific Virtual State process executing in the CP. Various process state registers also support CYBER 170 State operation. The contents of the process state registers can be written into memory as a Virtual State exchange package for either a Virtual State process or a CYBER 170 State process (refer to figures 2-4 and 2-5, respectively).

The exchange package for each process contains the step-by-step operating register contents as directed by process execution. In addition, the exchange package holds other detailed process state information such that the CP may dynamically switch between exchange packages (processes) while preserving process integrity. When a process executes in the CP, its exchange package resides in the process state registers. When a process awaits execution, its exchange package resides in CM.

Table 2-1 lists the processor state registers and permissible access types for CP Copy-to/from-State-Register instructions and for maintenance channel (MCH) access.

Table 2-1. Process State Registers

Register or Flag Name	No. of Bits	Address	Access Type Copy	Access Type MCH
Address (A0 through AF) (16 registers)	48	–	–	–
Base constant (BC)	32	47	R	R/W
Debug index (DI)	6	E4	R/W	R/W
Debug list pointer (DLP)	48	C5	R/L	R/W
Debug mask (DM)	7	E5	R/W	R/W
Critical-frame flag (CFF)	1	E0,E1	R/W	R/W
On-condition flag (OCF)	1	E2,E3	R/W	R/W
Process-not-damaged (PND) flag	1	–	–	–
Largest ring number (LRN)	4	–	–	CM
Last processor identification (LPID)	8	–	–	–
Monitor condition (MCR)	16	42	R	R/W
Monitor mask (MMR)	16	60	R/W	R/W
Operand (X0 through XF) (16 registers)	64	–	–	–
Process interval timer (PIT)	32	C9	R/L	R/W
Program address (P)	64	40	R	R/W
Segment table address (STA)	32	46	R	R/W
Segment table length (STL)	12	45	R	R/W
Top-of-stack (TOS) pointer (15 registers)	48	–	–	CM
Trap enable (TE)	2	C0-C3	R/L	R/W
Trap pointer (TP)	48	C4	R/L	R/W
Untranslatable pointer (UTP)	48	44	R	R/W
Untranslatable VMID (UVMID)	4	–	–	–
User condition (UCR)	16	43	R	R/W
User mask (UMR)	16	E6	R/W	R/W
Virtual machine identifier (VMID)	4	–	–	–

Notes:

    R: Unprivileged read
    W: Unprivileged write
    L: Locally-privileged write
    CM: In CP (indirectly accessible)

CP Base Constant (BC) Register

The 32-bit BC register provides a means for communication with the operating system. The contents of this register do not directly affect hardware operation.

CP Debug Index (DI) Register

The 6-bit DI register is added to the debug list pointer (DLP) register to form the process virtual address (PVA) of each word read from the debug list (DL). DI increments after each word is read. When a DL match occurs, DI + DLP points to the second word of the matched DL entry.

CP Debug List Pointer (DLP) Register

The 48-bit DLP register contains the PVA of the first DL entry. DLP bits 61 through 63 must be zeros or an address specification error (MCR 52) occurs.

CP Debug Mask (DM) Register

The 7-bit DM register contains two flag bits and five mask bits. The flag bits control debug initiation and termination. A mask bit, when set, enables the corresponding debug interrupt. DM has the following bit assignments.

DM Bit	Description
9	End-of-list-seen flag
10	Debug scan-in-progress flag
11	Data-read mask
12	Data-write mask
13	Instruction-fetch mask
14	Branch target instruction-fetch mask
15	Call target instruction-fetch mask

CP Flag Register

The 4-bit flag register contains the flag bits described in the following paragraphs.

Critical Frame Flag (CFF)

Software sets this flag to indicate that the stack frame in use (when the flag is set) requires special attention before this frame may be abandoned. Executing return/pop instructions with CFF set causes an interrupt (other enables-permitting). Call instructions and trap interrupts record CFF in the stack frame save area (SFSA) and proceed to clear CFF; return instructions restore the previous CFF condition.

On-Condition Flag (OCF)

Software sets this flag to assist the operating system in the handling of certain trap interrupts. Call instructions and trap interrupts record OCF in the SFSA and proceed to clear OCF; return instructions restore the previous OCF condition.

Process-Not-Damaged (PND) Flag

Hardware sets this flag in the outgoing exchange package if no uncorrectable error (MCR 48) occurs. PND indicates that the interrupted process is undamaged and may be restarted. Hardware ignores PND in an incoming exchange package.

CP Largest Ring Number (LRN) Register

The 4-bit LRN register contains the largest ring number for which there is a top-of-stack entry in the associated TOS register.

CP Last Processor Identification (LPID) Register

In dual-CP systems, the 8-bit LPID register identifies the last CP which executed the process defined by the exchange package.

CP Monitor Condition Register (MCR)

The 16-bit MCR records system exception conditions which the operating system must resolve (for example, hardware errors, instruction specification errors, and access violations). Refer to CP Interrupts in this chapter for further information.

CP Monitor Mask Register (MMR)

The 16-bit MMR enables or masks certain software-specified conditions directly associated with the MCR. An interrupt occurs when an MCR bit is set with the corresponding MMR bit set (other enables-permitting). Refer to CP Interrupts in this chapter for further information.

CP Operand (X) Registers

The sixteen 64-bit X registers, X0 through XF, supply operands for arithmetic operations and data manipulation. Depending on the operation, the registers contain a logical quantity, a signed binary integer, or a signed FP number. CP instructions which only require 32 data bits access the X registers as X-left (bits 0 through 31) or X-right (bits 32 through 63). The X-register formats are as follows (the S-field is the sign bit).

```
0                                                              63
 _____
|S|                    X Register                           |
|_|_____|

0                               3132                          63
 _____ _____
|S|     X Register Left           |s|    X Register Right     |
|_|_____|_|_____|
```
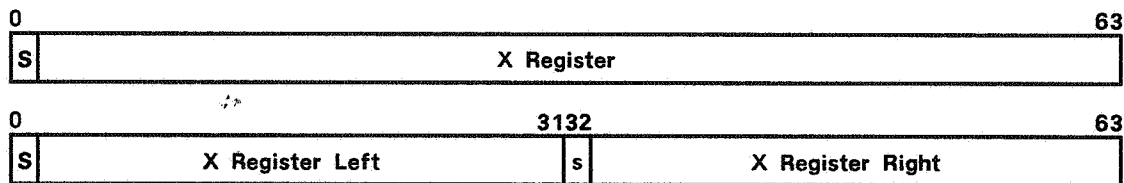
Store operations to X-left (XkL) do not alter X-right (XkR). Store operations to XkR do not alter XkL.

CP Process Interval Timer (PIT)

The 32-bit PIT register allows each process to track its own execution time. PIT is set either by a Copy-to-State-Register instruction with local privilege or from an incoming exchange package. The CP records PIT in an outgoing exchange package. PIT can be read via a Copy-from-State-Register instruction.

PIT decrements at a 1-µs rate without stopping. A trap interrupt occurs whenever the count equals zero (other enables-permitting).

CP Program Address (P) Register
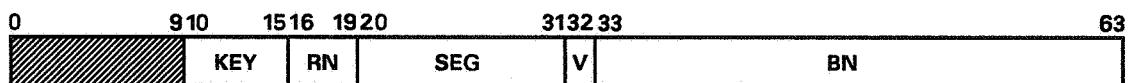
The 64-bit P register contains the PVA of an instruction during the time the CP interprets and executes it. The P register also contains bits which define memory access protection. The P register format is as follows.

```
0          9 10    15 16  19 20         31 32 33                        63
[/////////]  KEY  | RN  |    SEG       | V |            BN              |
```

Field	Name	Description
KEY	Key	Access permission attribute (refer to Access Protection under Virtual and CM Programming in this chapter).
RN	Ring Number	Access privilege indicator (refer to Access Protection under Virtual and CM Programming in this chapter).
SEG	Segment Number	Process segment number (refer to Access Protection under Virtual and CM Programming in this chapter).
V	Valid bit	Validity indicator (refer to Process Virtual Memory under Virtual and CM Programming in this chapter).
BN	Byte Number	Byte displacement within the 2^{31} bytes in a segment. Bit 32 in the final PVA used as a validity indicator and must be a zero or the PVA is rejected. Bit 32 in an A register may be a one, provided indexing or displacement changes this bit to a zero in the final PVA.

CP Segment Table Address (STA) Register

The 32-bit STA register contains the RMA of the first segment descriptor table entry (interpreted as a byte address). Hardware ignores the rightmost three bits of STA. Refer to Virtual and CM Programming in this chapter for further information.

CP Segment Table Length (STL) Register

The 12-bit STL register contains a count equal to one less than the number of 64-bit entries in the associated segment descriptor table. The virtual addressing mechanism uses this count to verify that segment table references are to an address within the segment table boundaries. Refer to Virtual and CM Programming in this chapter for further information.

CP Top-of-Stack (TOS) Pointer Registers

The operating system has for each process fifteen 48-bit TOS pointers to guarantee access protection. The TOS pointers are located in words 37 through 51 of an exchange package in CM. Each TOS is associated with one of the 15 rings of access protection; hardware uses the pointers to switch stacks during call/return instructions. Each TOS is a PVA pointing to the top of its associated stack when this stack is not in active use. Refer to Stack Manipulating Operations in this chapter, and Access Protection under Virtual and CM Programming in this chapter, for further information.

CP Trap Enable (TE) Register

The 2-bit TE register contains information that determines how traps are enabled. The TE register bits are represented as follows.

- Trap-enable flip-flop (TEF)

  When set, TEF is one of the conditions which enable trap interrupts. TEF is normally set by a Copy-to-State-Register instruction and cleared by hardware when a trap interrupt occurs. TEF can also be cleared by another Copy-to-State-Register instruction.

- Trap-enable delay (TED) flip-flop

  The TED flip-flop disables a trap interrupt until after the next return instruction completes execution. TED is normally set by a Copy-to-State-Register instruction and cleared by a return instruction. TED can also be cleared by another Copy-to-State-Register instruction.

CP Trap Pointer (TP) Register

The 48-bit TP register contains a PVA which is the indirect address of the entry point into the trap interrupt target procedure (refer to CP Interrupts in this chapter). The code base pointer (CBP) is the direct address to which TP points. An incoming exchange package loads the TP register. TP can also be written by a Copy-to-State-Register instruction with local privilege, and read by a Copy-from-State-Register instruction.

CP Untranslatable Pointer (UTP) Register

When an interrupt occurs because the CP cannot translate a PVA or system virtual address (SVA) to an RMA, the 48-bit UTP register contains the untranslatable PVA or SVA. UTP is set to PVA when the following MCR conditions occur.

● MCR 52 sets (except during load-page or purge instructions).

● MCR 54 or 57 sets.

● MCR 60 sets due to invalid SDE or exceeded STL.

● MCR 60 sets due to CBP RN-field of zero during a call-indirect instruction.

In the following cases, UTP may be set to either PVA or SVA.

● MCR 52 sets during load-page instruction.

● MCR 52 sets during purge instruction with k=0/1/8/9.

No other interrupt alters the UTP register.

CP Untranslatable Virtual Machine Identifier (UVMID) Register

Hardware sets the 4-bit UVMID register when an exchange operation, call-indirect instruction, or return instruction is interrupted due to an attempt to switch the CP to a state for which there is no set bit in the virtual machine capability list. In such a case, hardware sets the UVMID code to indicate which virtual machine capability list (VMCL) bit is missing. Values of 0 through 15 of UVMID correspond to VMCL bits 48 through 63 as follows.

UVMID	Missing VMCL Bit
0	48 (Virtual State)
1	49 (CYBER 170 State)
2 through FF	50 through 63 (reserved)

CP User Condition Register (UCR)

The 16-bit UCR register records conditions that normally do not require an exchange to monitor mode for corrective action. Each bit indicates detection of a particular error or exception condition in the CP (for example, divide fault, arithmetic overflow and underflow, or invalid BDP data). A trap interrupt occurs when a UCR bit sets with the trap enable flip-flop set, the corresponding user mask register (UMR) bit set, and trap-enable delay flip-flop clear. Refer to CP Interrupts in this chapter for further information.

CP User Mask Register (UMR)

The 16-bit UMR register enables or masks certain software-specified conditions directly associated with the UCR. An interrupt occurs when a UCR bit is set with the corresponding UMR bit set (other enables-permitting). Refer to CP Interrupts in this chapter for further information.

CP Virtual Machine Identifier (VMID) Register

The 4-bit binary code in the VMID register identifies the virtual machine capability being used, as follows.

VMID	Virtual Machine Capability
0	Virtual State
1	CYBER 170 State
2 through FF	Reserved

CP Processor State Registers

The processor state registers contain information about the state of the CP hardware, rather than a unique process. Included among this category of registers are the CP maintenance registers, which provide additional information about the condition of the CP hardware for diagnostic purposes. Other processor state registers contain such information as parameters of tables and pointers to exchange packages in CM.

Table 2-2 lists the processor state registers and permissible access types for CP Copy-to/from-State-Register instructions and for MCH access.

Table 2-2. Processor State Registers

Register Name	No. of Bits	Address	Access Type Copy	Access Type MCH
Dependent environment control (DEC)	32	30	–	R/W
Element identifier (EID)	32	10	R	R
Job process state (JPS)	32	61	R/M	R/W
Model-dependent word (MDW)	64	51	R	R/W
Monitor process state (MPS)	32	41	R	R/W
Options installed (OI)	64	12	R	R
Page size mask (PSM)	7	4A	R	R/W
Page table address (PTA)	32	48	R	R/W
Page table length (PTL)	14	49	R	R/W
Processor fault status (PFS)	64	80-89	–	R/W
Processor identifier (PID)	8	11	R	R
Processor test mode (PTM)	48	A0	–	R/W
Status summary (SS)	6	00	–	R
System interval timer (SIT)	32	62	R/M	R/W
Virtual machine capability list (VMCL)	16	13	R	R

Notes:

    R: Unprivileged read
    W: Unprivileged write
    G: Globally-privileged write
    M: Virtual State monitor mode write

CP Dependent Environment Control (DEC) Register

The 32-bit DEC register is a maintenance register which controls CP operating conditions.

Bit	Description
35	Disable corrected error to status summary. When set, disables the setting of corrected error (bit 62) in the CP status summary register.

Refer to the System Codes Booklet, listed under Related Manuals, for model-dependent bit descriptions.

CP Element Identifier (EID) Register

The 32-bit EID register is a backpanel-wired register identifying each system hardware element. The EID bits are represented as follows.

EID Bits	Description
32 through 39	Element type
40 through 47	Model number
48 through 63	Serial number (hexadecimal)

Refer to the System Codes Booklet, listed under Related Manuals, for model-dependent information contained in the EID register.

CP Job Process State (JPS) Register

The 32-bit JPS register contains the RMA of the first word of a Virtual State job mode exchange package. Hardware aligns the JPS address with bits 32 through 63 of RMAs and interprets the JPS address as 0, modulo 16. The CP ignores bit 32 and interprets bits 60 through 63 as zeros. System deadstart procedures load the initial JPS.

CP Model-Dependent Word (MDW) Register

The MDW register contains the PVA of the next location in CP for writing keypoint data. Bits 0 through 15 are unused. Bits 16 through 63 contain the PVA.

CP Monitor Process State (MPS) Register

The 32-bit MPS register contains the RMA of the first word of a Virtual State monitor mode exchange package. Hardware aligns the MPS address with bits 32 through 63 of RMAs, and interprets the MPS address as 0, modulo 16. The CP ignores bit 32 and interprets bits 60 through 63 as zeros. System deadstart procedures load the initial MPS.

CP Options Installed (OI) Register

The 64-bit OI register is a hardwired register identifying the options installed in the CP. Refer to the System Codes Booklet, listed under Related Manuals, for model-dependent information contained in the OI register.

CP Page Size Mask (PSM) Register

The 7-bit PSM register specifies the page size used in allocating real memory. Page sizes are selectable at system initialization time and range from 2K to 16K bytes. Refer to Virtual and CM Programming in this chapter for further information.

CP Page Table Address (PTA) Register

The 32-bit PTA register is a real-memory byte address pointing to the first page table entry. Refer to Virtual and CM Programming in this chapter for further information.

CP Page Table Length (PTL) Register

The 14-bit PTL register is a mask specifying the page table length. The page table ranges from 512K to 8192K words in 512-word increments. Refer to Virtual and CM Programming in this chapter for further information.

CP Processor Fault Status (PFS) Registers

The PFS registers record hardware-detected errors occurring within the CP. Refer to the System Codes Booklet, listed under Related Manuals, for model-dependent information contained in the PFS registers.

CP Processor Identifier (PID) Register

The 8-bit PID register is a hardwired register identifying each processor in the system. Refer to the System Codes Booklet, listed under Related Manuals, for model-dependent information contained in the PID register.

CP Processor Test Mode (PTM) Register

The 48-bit PTM register provides a maintenance capability which forces faults for the purpose of testing hardware-fault sensing within the CP. Refer to the System Codes Booklet, listed under Related Manuals, for model-dependent information contained in the PTM register.

CP Status Summary (SS) Register

The 6-bit SS register indicates CP status (similar registers exist in the CM and IOU). Aside from the Virtual State monitor mode bit, if any SS bit is set the SS bit also sets in the IOU status summary register. Refer to the System Codes Booklet, listed under Related Manuals, for further information.

The SS bits are represented as follows:

Bit	Description
58	Virtual State monitor mode.
59	Short warning. Sets to warn the system of an imminent environmental failure (for example, system power failure or local 50-Hz/60-Hz power failure).
60	Processor halted.
61	Uncorrectable error. Sets whenever the detected uncorrectable error (DUE) bit in MCR sets.
62	Corrected error. Sets after the CP corrects an error. The dependent environment control (DEC) register can be set to disable the recording of corrected errors.
63	Long environment warning. Sets to indicate an imminent failure condition (for example, high-temperature warning, blower fault, or low-temperature fault).

CP System Interval Timer (SIT) Register

The 32-bit SIT register is a timer which establishes maximum time intervals for process execution. The operating system first sets the timer to the desired value. The timer then decrements at a 1-µs rate until the count equals zero (the timer does not stop counting at zero; it decrements to all ones and continues decrementing). When enabled, the zero count causes an interrupt.

CP Virtual Machine Capability List (VMCL)

The 16-bit VMCL register is a backpanel-wired register indicating CP capabilities. The VMCL format is as follows.

VMCL Bit	Capability
48	Virtual State
49	CYBER 170 State
50 through 63	Reserved

CM Registers

The CM contains maintenance registers which hold memory status and error information (table 2-3). CM maintenance registers are accessible through the MCH (register B0 is also accessible through the memory ports).

Table 2-3. CM Maintenance Registers

Register Name	No. of Bits	Address	Access Type Copy	Access Type MCH
Corrected error log (CEL)	64	A0	–	R/W
Element identifier (EID)	32	10	–	R
Environment control (EC)	64	20	–	R/W
Free-running counter[1]	64	B0	R	W
Options installed (OI)	32	12	–	R
Port bounds register	64	21	–	R/W
Status summary	6	00	–	R
Uncorrectable error log (UEL) 1	64	A4	–	R/W
Uncorrectable error log (UEL) 2	64	A8	–	R/W

1. The free-running counter can be read from the CP by the Copy-Free-Running-Counter (08) instruction.

CM Corrected Error Log (CEL) Register

The 64-bit CEL register contains details concerning the first corrected error since this register was last reset. The model-independent bits are represented as follows.

Bit	Description
0	Valid bit. Indicates that the CEL contains a valid entry. When this bit is set, further correctable errors are discarded.
1	Unlogged corrected error. Indicates that a correctable error occurred but could not be logged because the CEL already contained an entry.

The CEL register contains model-dependent information regarding the address, parity, and encoded number of the memory port associated with the error. Refer to the System Codes Booklet, listed under Related Manuals, for model-dependent information contained in the CEL register.

CM Element Identifier (EID) Register

The 32-bit EID register is set by switches on logic panels. EID identifies each system hardware element according to the following bit assignments.

Bits	Description
32 through 39	Element type
40 through 47	Model number
48 through 63	Serial number (hexadecimal)

Refer to the System Codes Booklet, listed under Related Manuals, for model-dependent information contained in the EID register.

CM Environment Control (EC) Register

The 64-bit EC register controls CM error-checking and system clock timing. Refer to the System Codes Booklet, listed under Related Manuals, for information contained in the EC register.

CM Free-Running Counter Register

This 64-bit counter register consists of 48 counter bits right-justified and zero-filled in the leftmost 16 bits. The counter increments at a 1-μs rate. Successive reads of the free-running counter guarantee different values.

The free-running counter can be written at any time through the MCH. The CP can read the counter using the Copy-Free-Running-Counter (08) instruction.

CM Options Installed (OI) Register

The 64-bit OI register identifies the memory configuration and is set by field-modifiable switches on logic panels. Refer to the System Codes Booklet, listed under Related Manuals, for model-dependent information contained in the OI register.

CM Port Bounds Register

This 16-bit register controls the range of addresses accessible during a write operation through specified ports. For ports specified by DEC bits, write access is prevented either above or below the bounds RMA contained in the bounds register as determined by a bit in the DEC register. Refer to Maintenance Channel Programming in this chapter for further information.

CM Status Summary Register

This 64-bit register (6 bits are used) provides information about the CM error status and physical environment condition. Refer to the Maintenance Register Codes Booklet as shown in About This Manual for specific information contained in the CM status summary register.

CM Uncorrected Error Log (UEL) Registers

The two 64-bit UEL registers contain details of the first two uncorrected CM errors which occurred since the registers were last reset. The model-independent bits are represented as follows.

Bit	Description
0	Valid bit. Indicates that the UEL contains a valid entry. When this bit is set, further uncorrected errors are discarded.
1	Unlogged uncorrected error. Indicates that an uncorrected error occurred but could not be logged because the UEL already contained an entry.

The UEL registers also contain information about the source of the error. This information includes the illegal function, memory bounds fault, and multiple-bit memory error. Refer to the Maintenance Register Codes Booklet as shown in About This Manual for model-dependent information contained in the UEL registers.

IOU Registers

The IOU contains maintenance registers which hold IOU status and error information (table 2-4). IOU registers are accessible through the MCH.

Table 2-4. IOU Maintenance Registers

Register Name	No. of Bits	Address	Access Type Copy	Access Type MCH
Element identifier (EID)	32	10	–	R
Environment control (EC)	32	30	–	R/W
Fault status (FS) 1	64	80	–	R/W
Fault status (FS) 2	64	81	–	R/W
Fault status mask	64	18	–	R/W
Options installed (OI)	64	12	–	R
OS bounds	64	21	–	R/W
Status summary	6	00	–	R
Test mode (TM)	16	A0	–	R/W

IOU Element Identifier (EID) Register

The 32-bit EID register is a backpanel-wired register identifying each system hardware element. The EID bits are represented as follows.

Bits	Description
32 through 39	Element type
40 through 47	Model number
48 through 63	Serial number (hexadecimal)

IOU Environment Control (EC) Register

The 64-bit EC register controls timing margins, test mode and deadstart, PP memory dumps, reconfiguration, and stop-on-error conditions for the IOU. It also selects PP internal registers for reading. Refer to the Maintenance Register Codes Booklet as shown in About This Manual for further information.

IOU Fault Status (FS) Registers

The 64-bit FS registers indicate the presence of uncorrected faults in the IOU, PP memories, I/O channels, or PP hardware. Refer to the Maintenance Register Codes Booklet as shown in About This Manual for further information.

IOU Fault Status Mask Register

This 64-bit register controls IOU fault reporting to the IOU fault status (FS) registers. Refer to the Maintenance Register Codes Booklet as shown in About This Manual for further information.

IOU Options Installed (OI) Register

The 64-bit OI register identifies the options installed in the IOU. Refer to the Maintenance Register Codes Booklet as shown in About This Manual for further information.

IOU OS Bounds Register

The 64-bit operating system (OS) bounds register divides the CM into an upper and a lower region for system protection. The OS bounds register contains a bit for each PP which indicates the region in CM into which the specified PP may initiate exchange operations or writes. Refer to the Maintenance Register Codes Booklet as shown in About This Manual for further information.

IOU Status Summary (SS) Register

The status summary register indicates errors in the CP, CM, and IOU. It also provides information about the PP halt, error status, and physical environment conditions. Refer to the Maintenance Register Codes Booklet as shown in About This Manual for further information.

IOU Test Mode (TM) Register

The 64-bit TM register (16 bits are used) forces faults in the IOU for testing of the fault sensing logic. Bits 48 through 63 of this register serve a dual role. With the Enable Test Mode Register bit set in the EC register, these bits are used to force test conditions (refer to the Maintenance Register Codes Booklet as shown in About This Manual for further information). When the Enable Test Mode Register bit is clear, these read/write bits can be used by software as interlock/flag status bits.

CP Condition and Mask Registers

The CP contains a MCR and a UCR, each of which records interrupt causes and displays flag conditions. Each condition register has a corresponding mask register controlling the action taken when a condition register bit sets. Table 2-5 lists the bit assignments in the monitor condition and mask registers. Table 2-6 lists the bit assignments in the user condition and mask registers. The significance of the individual bits is further described in Interrupt Conditions in this chapter. Refer to the Virtual State, volume 1, hardware reference manual for a description of all CP registers.

In general, MCR and UCR bits may be altered by the following.

- CP hardware indicating a processor condition or external event.

- Branch-on-condition register (9F) instruction.

- PP communication over the MCH.

- Software with the condition register stored in an exchange package in CM.

- Trap interrupts which clear any condition register bits for which the corresponding mask register bit is set.

Monitor condition and user mask register bits may be altered by the following.

- Copy-to-State-Register (0F) instruction.

- PP communication over the MCH.

- Software with the mask register stored in an exchange package (EP) in CM.

Table 2-5. Monitor Condition/Mask Register Bit Assignments

MCR Bit	EP Bit	Type	Description	Job *	Mon *	Job **	Mon **	Any ***
48	0	X X	Detected uncorrectable error	Exch	Trap	Exch	Halt	Halt
49	1	X X	Not assigned	Exch	Trap	Exch	Halt	Halt
50	2	E N	Short warning	Exch	Trap	Exch	Stack	Stack
51	3	S T	Instruction specification error	Exch	Trap	Exch	Halt	Halt
52	4	S T	Address specification error	Exch	Trap	Exch	Halt	Halt
53	5	E N	CYBER 170 State exchange request	Exch	Trap	Exch	Stack	Stack
54	6	S T	Access violation	Exch	Trap	Exch	Halt	Halt
55	7	S 1	Environment specification error	Exch	Trap	Exch	Halt	Halt
56	8	E N	External interrupt	Exch	Trap	Exch	Stack	Stack
57	9	S T	Page table search without find	Exch	Trap	Exch	Halt	Halt
58	10	E N	System call (status bit)	None	None	None	None	None
59	11	E N	System interval timer	Exch	Trap	Exch	Stack	Stack
60	12	S 2	Invalid SEG/RN zero	Exch	Trap	Exch	Halt	Halt
61	13	S T	Outward call/Inward return	Exch	Trap	Exch	Halt	Halt
62	14	E N	Soft error	Exch	Trap	Exch	Stack	Stack
63	15	S T	Trap exception (status bit)	None	None	None	None	None

Notes:

Refer to Interrupt Conditions in this chapter for bit descriptions.

Stack: Test for opportunity to trap or exchange at each instruction fetch.
X: Either condition may happen.
E: Execution completed.
S: Execution suppressed.
N: P = PVA of next instruction.
T: P = PVA of this instruction.
1: For RN = 0 on Load A, Return or Pop, 1 = N; for RN = 0 on Call or Trap, 1 = T; for invalid segment, 1 = T.
2: For exchange operations, 2 = N; Call, Return or Pop, 2 = T.

*CP mode of operation – Trap enabled; associated MCR bit set.
**CP mode of operation – Trap disabled; associated MCR bit set.
***CP mode of operation – Not affected by trap condition; associated MCR bit clear.

Table 2-6. User Condition/Mask Register Bit Assignments

UCR Bit	EP Bit	Type	Description	Job *	Mon *	Job **	Mon **	Any ***
48	0	S T	Privileged instruction fault	Trap	Trap	Exch	Halt	–
49	1	S T	Unimplemented instruction	Trap	Trap	Exch	Halt	–
50	2	E N	Free flag	Trap	Trap	Stack	Stack	–
51	3	T	Process interval timer	Trap	Trap	Stack	Stack	–
52	4	S T	Inter-ring pop	Trap	Trap	Exch	Halt	–
53	5	T	Critical frame flag	Trap	Trap	Exch	Halt	–
54	6		Reserved	Trap	Trap	Stack	Stack	–
55	7	S N	Divide fault	Trap	Trap	Stack	Stack	Stack
56	8	S T	Debug	Trap	Trap	Stack	Stack	Stack
57	9	S T	Arithmetic overflow	Trap	Trap	Stack	Stack	Stack
58	10	E T	Exponent overflow	Trap	Trap	Stack	Stack	Stack
59	11	E T	Exponent underflow	Trap	Trap	Stack	Stack	Stack
60	12	E T	FP loss of significance	Trap	Trap	Stack	Stack	Stack
61	13	S T	FP indefinite	Trap	Trap	Stack	Stack	Stack
62	14	S T	Arithmetic loss of significance	Trap	Trap	Stack	Stack	Stack
63	15	E T	Invalid BDP data	Trap	Trap	Stack	Stack	Stack

Notes:

Refer to Interrupt Conditions in this chapter for bit descriptions.

Stack: Test for opportunity to trap or exchange at each instruction fetch.
E: Execution of instruction completed.
S: Execution of instruction suppressed.
N: P = PVA of next instruction.
T: P = PVA of this instruction.

*CP Mode of Operation – Trap enabled; associated MCR bit set.
**CP Mode of Operation – Trap disabled; associated MCR bit set.
***CP Mode of Operation – Not affected by trap condition; associated MCR bit clear.

CP Condition Register Bit Grouping

Refer to Interrupt Conditions in this chapter for bit descriptions. The four groups of condition register bits shown in table 2-7 are a function of the characteristics of the event detected, and of the P register PVA at time of interrupt.

The PVA from the P register stored in the exchange package during exchange interrupts [or in the SFSA during trap interrupts] points to an instruction address dependent on the condition register bit group as follows.

Group	PVA in P Stored During Interrupt
1	Points to the instruction executing when the malfunction was detected. This instruction did not necessarily initiate the activity resulting in the malfunction.
2a,2b	Points to the instruction that would have been executed if the interrupt had not occurred. After executing an exchange or return to the interrupted procedure, processing continues (from the PVA stored in the exchange package) as though the interrupt had not occurred.
3	Points to the instruction causing the interrupt.

Table 2-7. Interrupt Condition Groups

Group	Interrupt Condition	MCR/UCR Bit	Type	CY170 Occur- rence	Virtual Occur- rence
Malfunctions Not Necessarily Related to Current Instruction					
1	Detected uncorrectable error	MCR 48	Mon	Yes	Yes
Tested Between Instructions, Not Generated by Instructions					
2a	Short warning	MCR 50	Sys	Yes	Yes
	System interval timer	MCR 51	Sys	Yes	Yes
	Soft error	MCR 52	Sys	Yes	Yes
	External interrupt	MCR 59	Sys	Yes	Yes
	Free flag	UCR 50	User	Yes	Yes
	Process interval timer	UCR 51	User	Yes	Yes
	CYBER 170 State exchange request	MCR 53	Sys	Yes	Yes
Tested Between Instructions, Generated by Instructions					
2b	System call	MCR 58	Status	No	Yes
	Exponent overflow	UCR 58	User	No	Yes
	Exponent underflow	UCR 59	User	No	Yes
	FP loss-of-significance	UCR 60	User	No	Yes
	Invalid SEG/RN zero[1]	MCR 60	Mon	Yes	Yes
	Environment specification error[2]	MCR 55	Mon	Yes	Yes
Tested Before Execution, Generated by Instruction					
3	Instruction specification error	MCR 62	Mon	No	Yes
	Address specification error	MCR 52	Mon	Yes	Yes
	Invalid SEG/RN zero[1]	MCR 60	Mon	Yes	Yes
	Access violation	MCR 54	Mon	Yes	Yes
	Environment specification error[2]	MCR 55	Mon	Yes	Yes
	Page table search without find	MCR 57	Mon	Yes	Yes
	Outward call/inward return	MCR 61	Mon	Yes	Yes
	Trap exception	MCR 63	Status	Yes	Yes
	Privileged instruction fault	UCR 48	Mon	Yes	Yes
	Unimplemented instruction	UCR 49	Mon	Yes	Yes
	Inter-ring pop	UCR 52	Mon	No	Yes
	Critical frame flag	UCR 53	Mon	Yes	Yes
	Divide fault	UCR 55	User	No	Yes
	Debug	UCR 56	User	No	Yes
	Arithmetic overflow	UCR 57	User	No	Yes
	FP indefinite	UCR 61	User	No	Yes
	Arithmetic loss-of-significance	UCR 62	User	No	Yes
	Invalid BDP data	UCR 63	User	No	Yes

1. MCR 60 set by load address, return, or pop instruction when RN = 0 is in group 2b. MCR 60 set by call or trap instruction when RN = 0, or set by an invalid segment, is in group 3.

2. MCR 55 set by exchange operations is in group 2b. MCR 55 set by call, return, or pop instruction is in group 3.

CP Interrupts

Exchange interrupts and trap interrupts comprise the CP interrupt structure. The following paragraphs describe the characteristics of the two interrupt types.

Exchange Interrupts

An exchange interrupt causes an exchange operation as described in Exchange Operations in this chapter. Exchange interrupts switch the system from Virtual State job mode (Virtual State or CYBER 170 State environment) to Virtual State monitor mode. Exchange interrupts are disabled in Virtual State monitor mode.

Exchange interrupts initiate from conditions that set a bit in the MCR (and in some cases in the UCR). Exchange interrupts can only occur when the CP is in Virtual State job mode and the corresponding mask register bit is set. Refer to tables 2-5 and 2-6.

Trap Interrupts

A trap interrupt acts as an implicit call indirect (B0) instruction to an interrupt-handling procedure. Trap interrupts save the current SFSA environment, push the stack, and switch control to a software procedure for trap handling. Trap interrupts occur in response to the setting of UCR or MCR bits (as shown in tables 2-5 and 2-6) in the following environments.

- Within Virtual State monitor mode

- Within Virtual State job mode

- Upon switching from CYBER 170 State to Virtual State job mode (refer to State Switching Operations in this chapter)

The trap interrupt creates the maximum (33 word) SFSA descriptor and preserves the contents of the following in the associated SFSA: the P register, A and X registers, VMID, SFSA descriptor, monitor mask and condition registers, and user mask and condition registers. After the trap interrupt stores these registers in the SFSA, the CP clears the UCR or MCR bits causing the trap interrupt.

The trap interrupt target address is obtained by the CP using the PVA in the trap pointer register to access a CBP in a system binding chapter. This CBP contains the PVA of the next instruction to be executed. The external procedure flag must be set in the CBP. Refer to Stack Manipulating Operations in this chapter.

A trap interrupt disables further trap interrupts. Software may reenable traps by either setting the trap enable delay flip-flop and executing the return (04) instruction, or by setting the trap enable flip-flop. The return instruction reestablishes the suspended environment but does not load the monitor/UCRs from the SFSA into the CP.

If an exception condition arises during execution of a trap operation, the trap interrupt aborts and the following actions occur.

1. The trap exception bit (MCR 63) sets.

2. The appropriate UCR/MCR bit sets for the condition causing the trap to abort.

3. The trap enable flip-flop condition (set) is recorded in the exchange package stored for the interrupted procedure.

4. The exchange or halt performs as indicated in tables 2-5 and 2-6.

Virtual State job mode processes can control trap interrupts by setting bits in the user mask register. Bits set in the user mask register permit the trap interrupt when the corresponding UCR bit sets. Trap conditions occurring when traps are disabled have effect as listed in tables 2-5 and 2-6.

Interrupt Conditions

The following paragraphs present the various conditions causing system interrupts. Each condition description includes a reference to the corresponding MCR or UCR bit that sets upon condition detection. Refer to CP Condition and Mask Registers in this chapter.

Access Violation (MCR 54)

This bit sets when the CP attempts a CM access without the required access permission. The CM access is blocked. The following conditions result in a CM access violation.

● Attempt to read a nonreadable segment

● Attempt to read outside the read ring limit

● Attempt to write into a nonwritable segment

● Attempt to write outside the write ring limit

● Attempt to execute from a nonexecutable segment

● Attempt to execute from outside the execute ring bracket

● Attempt to call indirect when the CBP is not in a binding chapter segment

● Attempt to call indirect from a process outside the CBP call ring limit

● Key/lock violation

The PVA in P points to the instruction attempting the illegal access.

Address Specification Error (MCR 52)

This bit sets when the CP attempts to use an improper address as in the following cases.

● Data address with nonzero bits 61 through 63 generated by the following instructions:

Opcode	Instruction
04	Return
B5	Call indirect
80	Load multiple
81	Store multiple
82	Load word
83	Store word
B0	Call relative
F4	Calculate subscript

● Instruction address with nonzero bits 62 and 63 generated by unconditional branch (2F) instruction

● Any PVA with nonzero bit 32

The following instructions may also detect an address specification error.

● Decimal arithmetic (70 to 75, E4, and E5)

● Move immediate data (F9)

● Convert from FP to integer (3B)

● Test and set page (16)

The PVA in P points to the instruction specifying the incorrect address.

Arithmetic Loss-of-Significance (UCR 62)

This bit sets when significant digit(s) in the result are truncated or not stored in CM during execution of the following instructions.

● Decimal arithmetic (70 to 75, E4, and E5)

● Move immediate data (F9)

● Convert from FP to integer (3B)

The PVA in P points to the instruction causing this condition.

Arithmetic Overflow (UCR 57)

This bit sets as a result of one of the following conditions.

● Integer sum instructions (10, 20, 24, 28, 8A, and 8B) when augend and addend have same signs but sum has opposite sign

● Integer difference instructions (11, 21, 25, and 29) when minuend and subtrahend have opposite signs but difference sign is opposite of minuend sign

● Half-word integer product instructions (22 and 8C) when most significant 32 bits of intermediate product are not equal to sign bit

● Integer product instructions (26 and B2) when leftmost 64 bits of intermediate product are not equal to sign bit

● Half-word integer quotient (23) instruction when -2^{31} is divided by -2^0

● Integer quotient (27) instruction when -2^{64} is divided by -2^0

● Decimal arithmetic instructions (70 to 73) when result length exceeds destination field length

● Add immediate data (FB) instruction when source or destination field data descriptors specify invalid data type

The PVA in P points to the instruction causing the arithmetic overflow condition.

Critical Frame Flag (UCR 53)

This bit sets when the CP attempts to execute a pop or return instruction from a critical stack frame. The PVA in P points to the pop or return instruction causing this interrupt.

Debug (UCR 56)

This bit sets when a debug match occurs. Refer to Debug in this chapter for a description of this condition. The PVA in P points to the instruction causing the debug interrupt.

Divide Fault (UCR 55)

This bit sets when the CP detects a divisor equal to zero during execution of one of the integer quotient instructions (23, 27, 33, 37, and 73). Also, for the floating-point quotient instructions (33 and 37), the CP detects a divide fault if the divisor coefficient is a nonstandard value of zero, or is unnormalized and divisible into the dividend coefficient by a factor exceeding or equal to 2. The PVA in P points to the instruction causing the divide fault condition.

Environment Specification Error (MCR 55)

This bit sets when the CP detects an error in the environment during a call, return, or pop instruction, or during an exchange or trap operation. The PVA in P at the time of interrupt points to the instruction causing the error when the error results from any of the following conditions.

- A mismatch between VMCL and the VMID obtained from the CBP on a call indirect (B5) instruction

- A mismatch between VMCL and the VMID obtained from the SFSA on a return (04) instruction

- Initial A2 (previous save area pointer) not equal to A0 (dynamic space pointer) in the SFSA on a return (04) or pop (06) instruction

- In the previous stack frame descriptor, the field value designating the last A register to be loaded is less than 2 on a return (04) instruction

When this error results from the following condition, the PVA in P at the time of interrupt points to the next instruction which would have executed.

- A mismatch between VMCL and the VMID obtained from the exchange package during an exchange operation

When this error results from the following conditions, the PVA in P at the time of interrupt points to the instruction as defined under the condition causing the trap operation.

- A mismatch between VMCL and the VMID obtained from the CBP during a trap interrupt

- External procedure flag not set in the CBP during a trap interrupt

- The VMID from the CBP not equal to zero when executing a trap interrupt

Exponent Overflow (UCR 58)

Exponent overflow occurs when an FP comparison or arithmetic instruction produces an exponent with an actual value between 2[4096] and 2[12187]. The PVA in P points to the next instruction that would have executed.

NOTE
_____

Quantity in brackets indicates power factor.
_____

Exponent Underflow (UCR 59)

Exponent underflow occurs when an FP arithmetic instruction produces an intermediate exponent value between -2[4096] and -2[12287]. The PVA in P points to the next instruction that would have executed.

NOTE
_____

Quantity in brackets indicates power factor.
_____

External Interrupt (MCR 56)

This bit sets as a result of a processor interrupt (03) instruction in the interrupted CP (or in another CP in a multiprocessor system), through the CM port the instruction specifies. At the time of interrupt, the PVA in P points to the next instruction that would have executed.

FP Indefinite (UCR 61)

This bit sets when an FP arithmetic instruction produces a final nonstandard indefinite result. The PVA in P points to the instruction causing the FP indefinite condition.

FP Loss-of-Significance (UCR 60)

This bit sets when an FP arithmetic instruction produces an intermediate result with an overflow bit and coefficient of all zeros. The PVA in P points to the next instruction that would have executed.

Free Flag (UCR 50)

The free flag is normally set by software in an exchange package in CM: this bit causes an immediate trap interrupt after an exchange operation loads this exchange package into the CP. Software conventions dictate the use of this flag; hardware does not set this bit. The PVA in P points to the next instruction that would have executed.

Instruction Specification Error (MCR 51)

This bit sets:

- During isolate/insert instructions (AC, AD, and AE) when the sum of the leftmost position designator plus the length designator exceeds 63

- During business data processing (BDP) instructions when the length specified by the data descriptor L field exceeds the maximum length for applicable data type

- If data type fields in source and/or destination data descriptors are invalid during the following BDP instructions

Opcode	Instruction
70	Decimal sum
71	Decimal difference
72	Decimal product
73	Decimal quotient
74	Decimal compare
75	Numeric move
E4	Decimal scale
E5	Decimal scale rounded
ED	Edit
F4	Calculate subscript and add
F9	Move immediate data
FA	Compare immediate data
FB	Add immediate data

- During execution of BDP calculate subscript and add (FA) instruction when PVA bits 61 through 63 [used to access the subscript range table (SRT)] do not equal zero

- During execution of the program error (00) instruction

- During execution of the Copy-to-State-Register (0F) instruction, or the Branch-on-Condition-Register (9F) instruction, when execute access is restricted to Virtual State monitor mode with the CP not in this mode

- During execution of a call (B0/B5) instruction when the number of the last A register field [(At) in X0R bits 56 through 59] is less than two

The P register contains the PVA of the instruction with the error.

Inter-Ring Pop (UCR 52)

This bit sets from an attempt to pop a stack frame in one ring with a pop (06) instruction executing in a different ring. The pop instruction moves the current stack frame (CSF), previous save area (PSA), and TOS pointers to eliminate the stack frame without changing the P-counter. The PVA in P points to the pop instruction attempting the inter-ring pop.

Invalid BDP Data (UCR 63)

This bit sets when the CP detects an invalid decimal digit during execution of the following instructions: BDP decimal numeric, calculate subscript and add, compare immediate data, move immediate data, edit, and convert FP to integer. The PVA in P points to the instruction causing this condition.

Invalid Segment/Ring Number Zero (MCR 60)

This bit sets for the following reasons.

- A PVA was untranslatable into an RMA because the segment table length was exceeded or the segment descriptor was invalid. The PVA in P points to the instruction attempting the CM access.

- A call (B0 or B5) instruction attempted to execute with a CBP ring number (RN) equal to zero. The PVA in P points to the instruction attempting the CM access.

- An A register was loaded with a PVA with RN equal to zero during a load A (80, 84, A0), return (04), or pop (06) instruction. The PVA in P points to the next instruction that would have executed.

Not Assigned (MCR 49)

When set explicitly by software, this bit causes an interrupt identical to the detected uncorrectable error (MCR 48).

Outward Call/Inward Return (MCR 61)

This bit sets when the CP attempts an outward call or an inward return. The PVA in P points to the instruction attempting the outward call or inward return.

Page Table Search Without Find (MCR 57)

This bit sets when a page table search does not locate the requested page table entry. The PVA in P points to the instruction attempting the CM access that resulted in this condition, except when this exception is caused by an instruction fetch directly after a branch exit. In this case, the PVA in P points to the branched-to instruction.

Privileged Instruction Fault (UCR 48)

This bit sets when:

● An attempt is made to execute a local privileged instruction from other than a locally-privileged or globally-privileged segment.

● An attempt is made to execute a globally-privileged instruction from other than a globally-privileged segment.

● A trap (017) instruction executes in CYBER 170 State.

The PVA in P points to the instruction causing the privileged instruction fault interrupt.

Process Interval Timer (UCR 51)

This bit sets when the process interval timer decrements to zero. The PVA in P points to the next instruction that would have executed.

Detected Uncorrectable Error (MCR 48)

This bit sets when the CP detects an uncorrectable error condition in the processor, or on a processor-initiated memory reference. Typical examples are a parity error in data from memory that cannot be retried, an uncorrectable error in control storage, and CP errors that cannot be corrected or retried. A CM bounds violation also causes this exception.

The PVA in P does not necessarily point to the instruction causing the malfunction.

CYBER 170 State Exchange Request (MCR 53)

This bit sets when the CP receives the CYBER 170 State exchange request signal from the IOU, indicating that a PP in the IOU has executed one of the following instructions: exchange jump (00260x), monitor exchange jump (00261x), or monitor exchange jump MA (00262x). When the CP is in Virtual State, the operating system must switch the CP to CYBER 170 State job mode before the exchange can occur. The PVA in P points to the next instruction that would have executed.

Short Warning (MCR 50)

This bit sets when certain power distribution and warning system faults occur (refer to the appropriate power system manual listed under Related Manuals). The PVA in P points to the next instruction that would have executed. This bit remains set until the condition returns to normal, at which time it clears.

Soft Error Log (MCR 62)

This bit sets to indicate error detection and correction by the hardware regarding the following.

- A corrected error in CM for the port used by this CP (also recorded in the CM corrected-error register)

- A corrected hardware malfunction in the CP

The PVA in P points to the next instruction that would have executed.

System Interval Timer (MCR 59)

This bit sets when the system interval timer decrements to zero. The PVA in P points to the next instruction that would have executed.

Trap Exception (MCR 63)

This bit sets when the system detects a fault during a trap interrupt operation. In such a case, at least one other MCR bit indicates the cause of the trap exception. The PVA in P points to the PVA that would have been stored in the stack frame, word 0, if the trap had completed without any exception conditions.

Unimplemented Instruction (UCR 49)

This bit sets when an instruction not implemented in the CP attempts to execute. The instruction descriptions in chapter 1 specify which instructions are model-dependent. The PVA in P points to the instruction causing the interrupt.

The CYBER 170 State compare/move instructions (464 through 467) also cause this interrupt.

Multiple Interrupt Conditions

Tables 2-5 and 2-6 list the interrupt action taken in various operating modes. When more than one bit sets in the MCR/UCR, the interrupts are processed with the following priority.

1. Halt (any halt condition present)

2. Exchange (no halt condition present, and any exchange condition present)

3. Trap (no halt or exchange condition present, and any trap condition present)

4. Stack (none of the above conditions present)

Figure 2-6 is a flowchart showing the CP detecting an exception condition and taking action on it.
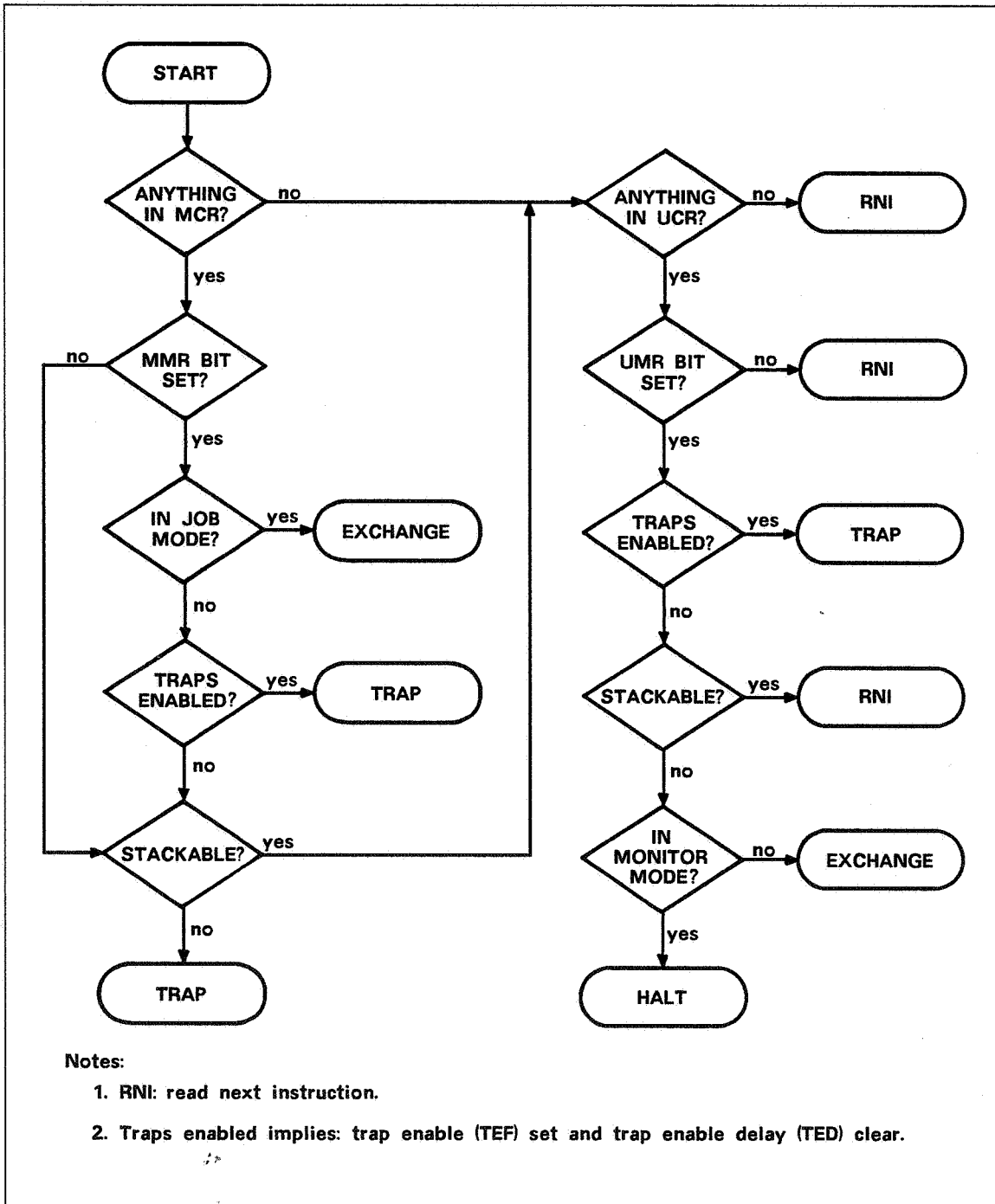
Figure 2-6. Interrupt Flowchart

Notes:

1. RNI: read next instruction.

2. Traps enabled implies: trap enable (TEF) set and trap enable delay (TED) clear.

Flags

Table 2-8 indicates the state of the CFF, OCF, trap enable flip-flop, and trap enable delay flip-flop following the execution of the Virtual State call, return, pop, exchange, and trap operations.

Table 2-8. Condition of Flags Following Call, Return, Pop, Exchange, and Trap Operations

Operations	CFF Flag	OCF Flag	TEF Flag	TED Flag
Call	C	C	A	A
Return	PS	PS	A	C
Pop	PS	PS	A	A
Exchange	XP	XP	XP	XP
Trap	C	C	C	A

Notes:

C: Cleared by operation.
A: As is (unchanged by operation).
PS: Loaded by operation from previous SFSA.
XP: Loaded by operation from exchange package.

Stack Manipulating Operations

Each process has up to 15 stacks: one for each ring of execution privilege as defined by the RN field (bits 16 through 19) of the hardware P register. Hardware accesses these stacks to save/restore the process registers and operating conditions during trap interrupts, and during the following Virtual State instructions.

- Call indirect

- Call relative

- Return

- Pop

- Trap

These 15 stacks are used as parts of a single stack divided solely to guarantee access protection. The buildup and reduction of the 15 stacks always occurs through the same locations (in opposite directions), switching from stack to stack only when the P register RN changes.

The operating system allocates stack space to each process. One use of the CFF is to mark the first frame in each stack to indicate the beginning of the stack. The operating system may also check for a maximum allowable stack length when assigning a new page to the stack through the virtual memory demand paging mechanism.

The 15 stacks operate in conjunction with assigned registers A0 through A4 and 15 TOS pointers for the specific process. An exchange operation switches stacks by providing new A0 through A4 and new TOS pointers.

Stack Frames and Save Areas

A procedure may use its stack for storing its dynamic variables. At times it may call another procedure, which in turn may call another procedure, and so on. Also, at any time, a trap interrupt condition may initiate a call-type operation. Each time a call occurs, hardware saves the registers of the suspended part of the process (the caller) in the currently active stack, together with some status information. This leaves these registers free for use by the branched-to software (the callee). The area in which the registers are stored is the SFSA. The SFSA combines with the previously stored dynamic variables (if any) to comprise a stack frame.

The CP hardware design provides that the string of successively-called procedures may include previously called procedures (recursive calls), provided code modification is not used.

Stack Frame Save Area Format

For call instructions, the programmer specifies the number of registers stored in SFSA (from 4 to 33) by way of a descriptor placed into X0, as shown in figure 2-7. Trap interrupts always generate the maximum save area of 33 words. Figure 2-8 shows the format of SFSA.
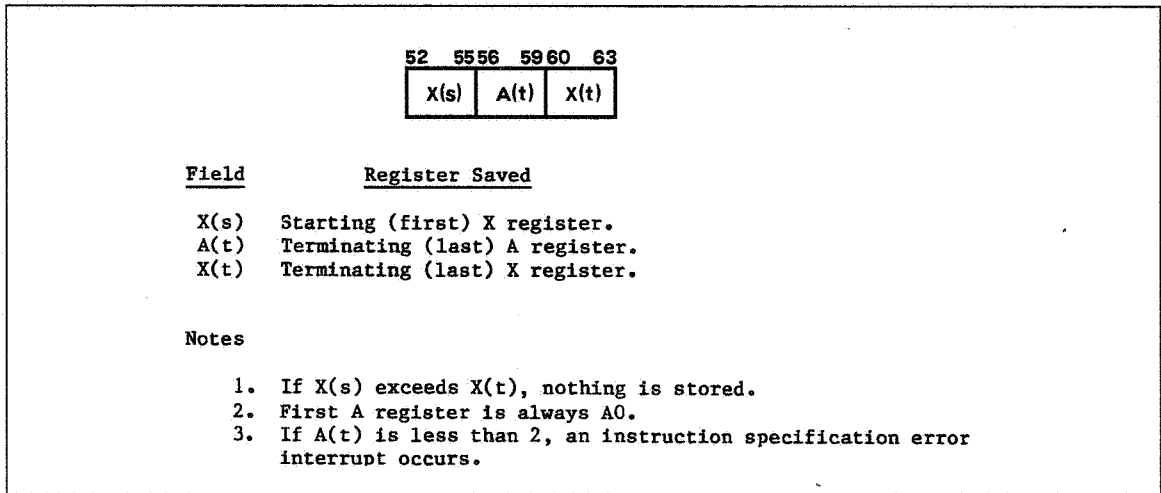
```
       52   55 56   59 60   63
      ┌──────┬──────┬──────┐
      │ X(s) │ A(t) │ X(t) │
      └──────┴──────┴──────┘
```

Field   Register Saved

X(s)   Starting (first) X register.
A(t)   Terminating (last) A register.
X(t)   Terminating (last) X register.

Notes

1. If X(s) exceeds X(t), nothing is stored.
2. First A register is always A0.
3. If A(t) is less than 2, an instruction specification error
    interrupt occurs.

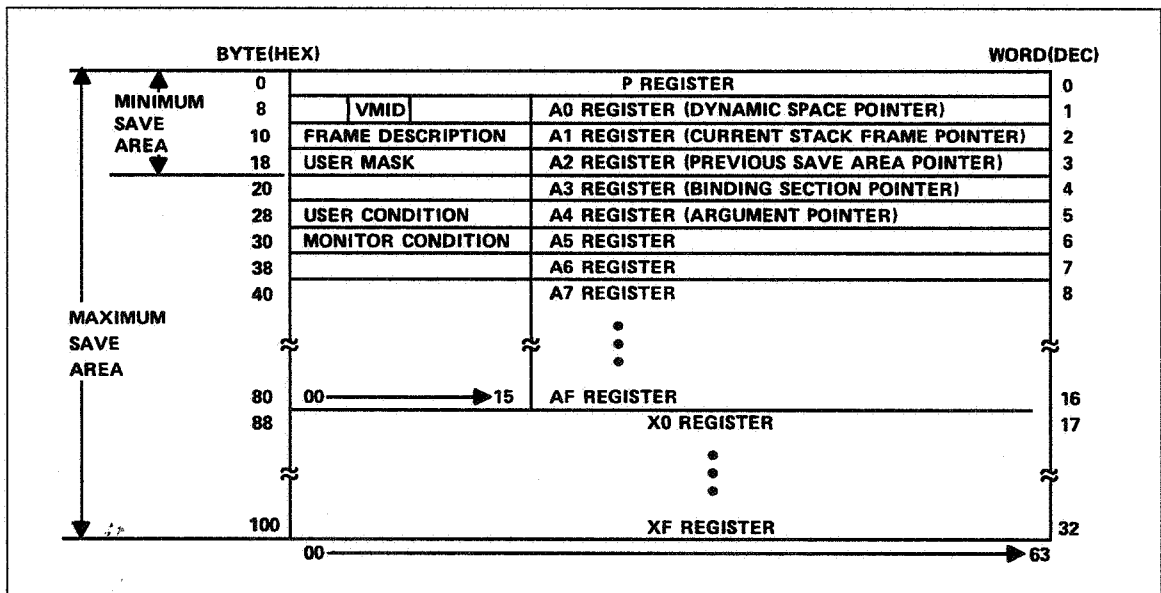Figure 2-7. Format of X0 for Call Instructions

Figure 2-8. Virtual State Stack Frame Save Area

Stack Frame Save Area Descriptor Field

The SFSA descriptor (figure 2-9) is in word 2, bits 0 through 15 of the SFSA. It records the number of X and A registers saved in the SFSA, and also the state of the CFF, the OCF, and the process-not-damaged (PND) flag when the SFSA is generated.

The CFF and the OCF are hardware register flags set/cleared by Copy-to/from-State-Register instructions, and which may also be prerecorded in an exchange package or the SFSA. CFF set for the current stack frame inhibits instruction execution and causes an interrupt when encountered during a return or pop instruction.

Executing a call instruction or a trap interrupt stores the CFF and OCF in the SFSA descriptor generated for the current stack frame and clears these flags. Executing a return instruction loads these flags from the previous SFSA.

The PND flag indicates whether or not a process being executed was damaged and whether the process may be restarted. This flag is intended to allow recovery of monitor mode processes where possible.

The PND flag is valid when set during a Virtual State monitor mode trap operation caused by an uncorrected error. In this case, the flag indicates that the executing process was undamaged and that it may be restarted. The PVA in P of the stack frame is the restart address for the process, but is not necessarily the address of the instruction which initiated the malfunction.

The default (clear) state of the PND flag interprets the process as damaged. The hardware ignores the flag when loading a stack frame.
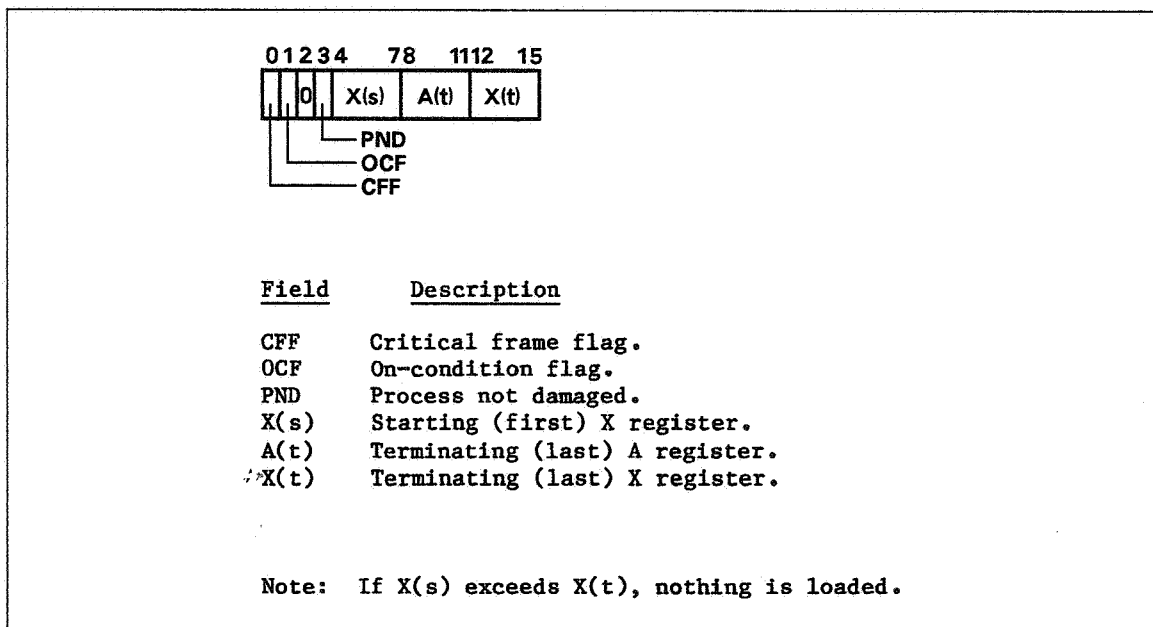

Figure 2-9. Stack Frame Save Area Descriptor

Virtual Machine Identifier (VMID) Field

A call instruction or a trap interrupt stores the VMID in SFSA word 1, bits 4 through 7. A return instruction loads the VMID from the previous SFSA into the CP, with the exception that an attempt to load a VMID = 1 requires global privilege.

User Mask/Condition and Monitor Condition Fields

A trap interrupt (but not a call instruction) stores the UCR and MCR in words 5 and 6, respectively (bits 0 through 15), in the SFSA. The CP clears the condition register bit(s) causing the trap interrupt. The return instruction does not restore the condition registers from the SFSA to the CP.

Assigned Registers During Stack Operation

Stack manipulating operations change registers A0 through A4 and TOS pointers from the exchange package. For proper operation, the programmer must use registers A0 through A4 as designated.

Top of Stack Pointers

An exchange package contains 15 TOS fields which initially point to the next available vacant word in each stack, as set by the operating system. In subsequent use, TOS for the active stack points to the first word in the current stack frame. Hardware updates TOS when any stack frame is pushed or popped. Hardware uses TOS only when stacks switch.

The TOS pointers remain in the exchange package stored in CM and are accessed from there by hardware.

Dynamic Space Pointer (A0)

Register A0 has the role of dynamic space pointer (DSP), pointing to the first available vacant byte number in the active stack. Hardware updates DSP when a stack frame is pushed or popped. Software must update DSP when storing/removing process dynamic variables in the stack.

Current Stack Frame Pointer (A1)

Register A1 has the role of CSF pointer, pointing to the first word in the current stack frame. CSF updates when a stack frame is pushed or popped. The process must not reduce a stack frame below CSF, and must not change CSF.

Previous Save Area Pointer (A2)

Register A2 has the role of PSA pointer, pointing to the first word of the PSA (not necessarily in the currently active stack). PSA updates when a stack frame is pushed or popped. The process must not change PSA.

Binding Section Pointer (A3)

A binding chapter pointer (BSP) points to the first word in a list of indirect addresses (CBPs) for use by call indirect instructions or other information determined by software conventions. During call indirect instructions to an external procedure (which has its own binding chapter), register A3 always provides the BSP of this external procedure. The call indirect instruction first uses the BSP in Aj to access the CBP containing the target address. When this CBP has its external procedure flag set, the word stored in CM immediately after this CBP loads into A3.

Argument Pointer (A4)

Register A4 has the role of argument pointer, used through software conventions. It copies from Ak during call indirect instructions. The process may use A4 as permitted through software conventions.

Exceptions During Stack Operations

When an exception causes a call instruction or trap interrupt to abort, the following may precede the abort.

● DSP (A0) may be rounded up.

● Portions of the environment may be stored into the save area on top of the current stack frame.

BDP Programming

BDP instructions operate on CM data fields which may be 1 through 8, 19, 38, or 256 bytes in length. BDP instructions utilize two forms of data fields in CM: 1) the source field, and 2) the destination field. The former modifies, replaces, or compares to the latter. These fields are independently designated by BDP data descriptors, described in the following paragraphs. The CP accommodates nine types of packed and unpacked binary-coded decimal (BCD) data, plus alphanumeric, binary-unsigned, and binary-signed data types. In many cases the data types may be freely mixed as the hardware performs the necessary type translations. The CP also manipulates alphanumeric data fields.

BDP Data Descriptors

The source and destination field data is described by one or two data descriptors obtained from the CM at locations immediately following a BDP instruction. The instructions using the format jk have two descriptors. The instructions using the format jkQ have either one or two descriptors.

As shown in figure 2-10, each BDP data descriptor is a 32-bit half word describing the source or destination field data type, number of bytes, and relative memory location.
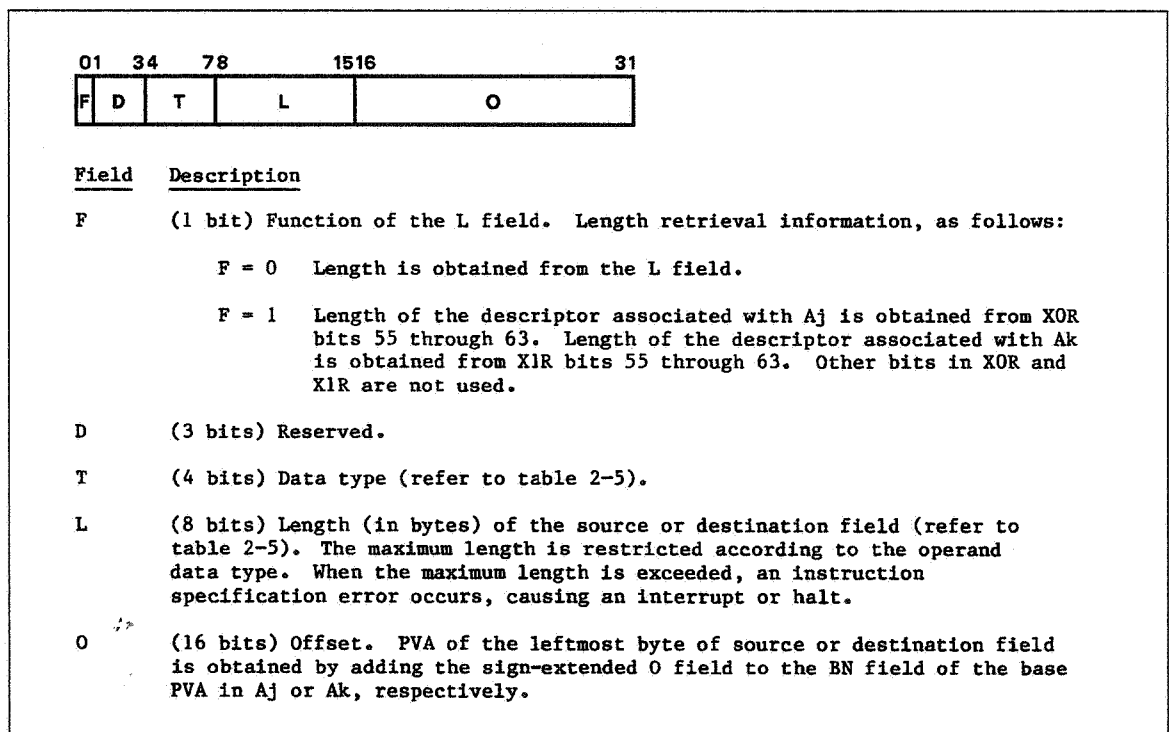
```
01   34   78        1516              31
┌─┬───┬────┬─────────┬────────────────┐
│F│ D │ T  │    L    │       O        │
└─┴───┴────┴─────────┴────────────────┘
```

Field	Description
F	(1 bit) Function of the L field. Length retrieval information, as follows:

        F = 0   Length is obtained from the L field.

        F = 1   Length of the descriptor associated with Aj is obtained from XOR bits 55 through 63. Length of the descriptor associated with Ak is obtained from X1R bits 55 through 63. Other bits in XOR and X1R are not used.

Field	Description
D	(3 bits) Reserved.
T	(4 bits) Data type (refer to table 2-5).
L	(8 bits) Length (in bytes) of the source or destination field (refer to table 2-5). The maximum length is restricted according to the operand data type. When the maximum length is exceeded, an instruction specification error occurs, causing an interrupt or halt.
O	(16 bits) Offset. PVA of the leftmost byte of source or destination field is obtained by adding the sign-extended O field to the BN field of the base PVA in Aj or Ak, respectively.

Figure 2-10. BDP Data Descriptor Format

BDP Data Types

The 12 data types listed in table 2-9 are described in this subsection, including the permitted range of values for each data type with respect to digits (D), characters (C), signs (S), and maximum length (L).

Table 2-9. BDP Operand Types and Field Lengths

Data Type	Description	Maximum Length (Bytes)
0	Packed decimal, no sign.	
1	Packed decimal, no sign, leading slack digit.	19
2	Packed decimal, signed.	
3	Packed decimal, signed, leading slack digit.	
4	Unpacked decimal, unsigned.	
5	Unpacked decimal, trailing sign combined Hollerith.	
6	Unpacked decimal, trailing sign separate.	38
7	Unpacked decimal, leading sign combined Hollerith.	
8	Unpacked decimal, leading sign separate.	
9	Alphanumeric.	256
10	Binary, unsigned.	8
11	Binary, signed.	

Data Type 0: Packed Decimal, Unsigned

```
          ┌────┬────┬────┬─)) ┬────┬────┬────┐
          │ DD │ DD │ DD │    │ DD │ DD │ DD │
          │    │    │    │─)) │    │    │    │
Byte      │ 0  │ 1  │ 2  │    │ 16 │ 17 │ 18 │
          └────┴────┴────┴────┴────┴────┴────┘

          D:  Hex 0 through 9.
          L:  19 bytes maximum.
```

This format corresponds to an even number of digits in the decimal number.

Data Type 1: Packed Decimal, Unsigned, Slack Digit

```
          ┌────┬────┬────┬─)) ┬────┬────┬────┐
          │ 0D │ DD │ DD │    │ DD │ DD │ DD │
          │    │    │    │─)) │    │    │    │
Byte      │ 0  │ 1  │ 2  │    │ 16 │ 17 │ 18 │
          └────┴────┴────┴────┴────┴────┴────┘

          0:  Hex 0.
          D:  Hex 0 through 9.
          L:  19 bytes maximum.
```

This format corresponds to an odd number of digits in the decimal number.

Data Type 2: Packed Decimal, Signed

DD	DD	DD	⟩⟩	DD	DD	DS
0	1	2	⟩⟩	16	17	18

Byte

D: Hex 0 through 9.
S: (Positive sign) hex A, B, C, E, or F (C preferred);
   (Negative sign) hex D.
L: 19 bytes maximum.

This format corresponds to an odd number of digits in the decimal number.

Data Type 3: Packed Decimal, Signed, Slack Digit

OD	DD	DD	⟩⟩	DD	DD	DS
0	1	2	⟩⟩	16	17	18

Byte

O: Hex 0.
D: Hex 0 through 9.
S: (Positive sign) hex A, B, C, E, or F (C preferred);
   (Negative sign) hex D.
L: 19 bytes maximum.

This format corresponds to an even number of digits in the decimal number.

Data Type 4: Unpacked Decimal, Unsigned

D	D	D	D	⟩⟩	D	D	D	D
00	01	02	03	⟩⟩	34	35	36	37

Byte

D: ASCII characters 0 through 9 (represented by hex 30 through 39).
L: 38 bytes maximum.

Data Type 5: Unpacked Decimal, Trailing Sign Combined Hollerith

D	D	D	D	⟩⟩	D	D	D	C
00	01	02	03	⟩⟩	34	35	36	37

Byte

In the following, the preferred characters and codes are underlined.

D: ASCII character 0 to 9 (represented by hex 30 through 39).
C: ASCII character decoded as follows:

      ASCII 1 to 9 (hex 31 through 39) represents +1 through +9, or
      ASCII A through I (hex 41 through 49) represents +1 through +9.
      ASCII J through R (hex 4A through 4F and hex 50 through 52)
      represents −1 through −9.
      ASCII {, [, 0, 8 (hex 7B, 3C, 30, 26) represents +0.
      ASCII },], − (hex 7D, 21, 2D) represents −0.

L: 38 bytes maximum.

Data Type 6: Unpacked Decimal, Trailing Sign Separate

D	D	D		D	D	D	D	S
0	1	2		33	34	35	36	37

Byte

D: ASCII character 0 through 9 (hex 30 through 39).
S: ASCII character + (hex 2B), positive sign;
    ASCII character - (hex 2D), negative sign.
L: 38 bytes maximum.

Data Type 7: Unpacked Decimal, Leading Sign Combined Hollerith

C	D	D	D		D	D	D	D
0	1	2	3		34	35	36	37

Byte

D: Same as data type 5.
C: Same as data type 5.
L: 38 bytes maximum.

Data Type 8: Unpacked Decimal, Leading Sign Separate

S	D	D	D		D	D	D	D
0	1	2	3		34	35	36	37

Byte

S: Same as data type 6.
D: Same as data type 6.
L: 38 bytes maximum.

Data Type 9: Alphanumeric

C	C	C	C		C	C	C	C
0	1	2	3		252	253	254	255

Byte

C: Any ASCII character code.
L: 256 bytes maximum.

Data Type 10: Binary, Unsigned

L: 8 bytes maximum.

The L bytes of the type 10 data field contain the positive binary operand value.
Negatively-signed data moved to a type 10 destination field is considered positive.

Data Type 11: Binary, Signed

L: 8 bytes maximum.

The L bytes of the type 11 data field contain the signed binary operand value.
Negative values are represented in twos complement form.

Slack Digit

With data types 1 and 3, the slack digit value as read from CM is ignored and treated as zero. The slack digit value as written into CM is forced to zero and is not affected by any arithmetic overflow or arithmetic loss-of-significance that may occur.

Undefined Results

Overlap

BDP instruction execution produces undefined results whenever the source and destination fields overlap and the leftmost and rightmost byte positions do not coincide.

Invalid Data

As a rule, invalid BDP data causes undefined results to be stored in the destination field in CM only when the corresponding mask bit is clear or traps are disabled. An exception: the decimal-compare and numeric-move instructions always store undefined results in X1R when invalid BDP data is detected.

FP Programming

FP arithmetic automatically maintains binary point placement during computations involving large numeric values or values within a widely varying range. This occurs by separating a number's significant digits from the number size to express the number as a fraction multiplied by a power of 2. Thus, each FP number contains two values as follows.

● Coefficient (fraction) represents the number's significant digits. The binary point of the coefficient is always directly left of its most significant bit.

● Exponent (characteristic) is a power of 2 by which the coefficient must be multiplied to obtain the whole FP number value.

FP Data Formats

In Virtual State, FP data exists in 64- and 128-bit fixed-length formats (single precision and double precision), as shown in figure 2-11.

0 1	1516	63
S	BIASED EXPONENT	COEFFICIENT (leftmost 48 bits)

Format of single precision FP number and of the leftmost part of double precision FP number

6465	7980	127
S	BIASED EXPONENT	COEFFICIENT (rightmost 48 bits)

Format of rightmost part of double precision FP number

Bits	Description

0,64	Coefficient sign
	  0 Coefficient positive.
	  1 Coefficient negative.
	Double-precision FP sign bit 64 is set equal to bit 0 in results.

16-63 80-127	Coefficient magnitude (leftmost 48 bits). Coefficient magnitude (rightmost 48 bits). The coefficient without the sign bit is an unsigned, exclusively positive fraction with the binary point directly left of bit 16. Double-precision FP number bits 64 through 79 are set equal to bits 0 through 15 in results.
1 4-15, 65-79	Exponent sign, biased. Exponent value, two's complement. The exponent is a biased-signed two's complement integer. A bias of 16,384 (4000_{16}) adds to the exponent to allow encoding of exponent values from -4096 to +4095 inclusive (within the 15 exponent bits) as shown in table 2-10.

1-3	Coded to indicate the following FP numbers
	  00X FP zero.
	  0X0 FP zero.
	  011 Standard FP number.
	  100 Standard FP number.
	  101 FP infinity.
	  110 FP infinity.
	  111 FP indefinite.

Figure 2-11. FP Data Formats

Table 2-10. FP Representation

Exponent With Coefficient Sign (Hexadecimal)	Actual Exponent (Binary)	Input Arguments	Terms Used for Numbers in This Range
Positive Numbers (Coefficient sign = 0)			
7XXX		Indefinite	+IND
6FFF	12278	Infinite	+∞
5000	4096		
4FFF	4095	Standard	+N
4000	0		
3FFF	−1		+Z3
3000	−4096		
2FFF	−4097	Zero	+Z2
1000	−12288		
0XXX		Zero	+Z1
Negative Numbers (Coefficient sign = 1)			
8XXX		Zero	−Z1
9000	−12288	Zero	−Z2
AFFF	−4096		
B000	−4096	Standard	−Z3
BFFF	−1		
C000	0		−N
CFFF	4095		
D000	4096	Infinite	+∞
EFFF	12287		
FXXX		Indefinite	−IND

Standard and Nonstandard FP Numbers

Nonstandard FP numbers are FP numbers outside the capacity of standard FP numbers. Special exponent field codes identify the three nonstandard FP numbers: zero ($\pm$Z1,$\pm$Z2), infinity ($\pm\infty$), and indefinite ($\pm$INDEF). Table 2-10 lists hexadecimal exponent codes for nonstandard and standard FP numbers.

FP Zero

Nonstandard FP operands with bits 01 and 02, or 01 and 03 clear, are treated as if consisting of all zeros.

The nonstandard zero FP numbers are represented as $\pm$Z1 or $\pm$Z2 as shown in table 2-11. The specific number in the +Z1 range which consists of all (64) zeros is termed +0. Thus, wherever $\pm$Z1 is indicated, the +0 is also included since it is a member of +Z1.

The standard zero FP numbers are represented as $\pm$Z3 as shown in table 2-10.

FP Nonzero

Standard FP operands which have nonzero coefficients are represented as $\pm$N in table 2-10.

FP Infinite

Nonstandard FP operands with bit 01 set, and bit 02 not equal to bit 03, are treated as infinite values.

The nonstandard FP numbers in the Infinite range are represented as $\pm\infty$ as shown in table 2-10. The specific number in the $+\infty$ range consists of 5000∞000. The specific number in the $-\infty$ range consists of D000∞000.

FP Indefinite

Nonstandard FP numbers with bits 01, 02, and 03 set are treated as indefinite values.

The nonstandard FP numbers in the Indefinite range are represented as $\pm$IND as shown in table 2-10. The specific number in the +IND range consists of 7000∞000. The specific number in the –IND range consists of F000∞000.

Double-Precision Nonstandard FP Numbers

When nonstandard results are generated, the rightmost part of a double-precision FP result is made identical to the leftmost part.

Exponent Arithmetic

When the operand exponent fields are added (as in FP multiplication) or subtracted (as in FP division), the exponent arithmetic performs algebraically in twos-complement mode. Such operations take place as if the bias were removed.

Exponent underflow and overflow conditions are detected for all single-precision results, but only for the leftmost part of double-precision results.

Normalization

Normalized operands ensure the highest possible precision in the result. An FP number is normalized when the coefficient bit 16 is a one.

Normalization takes place when intermediate results become final results. It occurs by left-shifting the 48-bit fraction until bit 16 is a one, and by reducing the exponent value by the number of positions shifted. Numbers with zero fractions cannot be normalized and remain equal to zero.

When normalizing a double-precision FP number, the entire 96-bit fraction left-shifts until bit 16 is a one, with a corresponding exponent value reduction.

If the coefficient of an intermediate result overflows, the fraction right-shifts one bit position and the exponent increases by one. If the input operands for the FP multiply (32 and 36) and the FP divide (33 and 37) instructions are unnormalized, the result may be unnormalized.

FP Sum and Difference

When two FP operands with unequal exponents are added or subtracted, the hardware aligns copies of these operands before the addition or subtraction performs, as follows: the fraction with the smaller exponent is right-shifted, end-off, by the number of bit positions equal to the difference between the exponents. The maximum shift is 48 positions (single precision) or 96 positions (double precision). After copies of the fractions have been aligned in this manner, they are added or subtracted. The result generated is 48 bits (single precision) or 96 bits (double precision).

When summing coefficients with like signs or subtracting coefficients with unlike signs, the result may overflow/underflow by one bit; this bit is saved. In such a case, the 48- or 96-bit intermediate result fraction shifts right one position, end-off. The overflown bit inserts into the high-order position. The result's exponent increases by one to adjust for the right shift of the coefficient. The adjusted exponent and the 48-bit or 96-bit fraction and its sign bit are the final result.

FP Multiply

The signed exponents for the two input operands to be multiplied are algebraically added, with the result used as an intermediate exponent.

The multiplied fractions generate an intermediate product with 96 bits (single precision) or 192 bits (double precision). The correct sign bit is algebraically determined. If the high-order bit in the product is a one, the product is already normalized and remains unchanged. If the high-order bit in the product is a zero, the entire 96- or 192-bit product left-shifts one bit position and the intermediate exponent decreases by one. This one-position shift normalizes the product if the original input operands were normalized. The high-order 48 or 96 bits of the product or shifted product become an intermediate coefficient. If the final intermediate exponent indicates a standard FP number, the intermediate exponent and the intermediate coefficient with its sign bit are the final result.

FP Divide

During execution of an FP divide instruction, the divisor (Xj, single precision or XXj, double precision) exponent is subtracted from the dividend (Xk, single precision or XXk, double precision) exponent, and the signed result provides an intermediate exponent.

The dividend (Xk or XXk) fraction is divided by the divisor (Xj or XXj) fraction. The result's sign is determined algebraically from the operand signs.

If the fraction in Xj or XXj is initially unnormalized and can be divided into the fraction in Xk or XXk by a factor equal to or exceeding two, a divide fault occurs, setting UCR bit 55, with an interrupt (when enabled).

If no error occurs, the intermediate quotient generated is 48 or 96 bits. When the divisor can be divided into the dividend by a factor equal to or exceeding one, but less than two, an overflow bit also generates. If the overflow bit is a zero, the sign bit and the 48- or 96-bit fraction require no further adjustment. When the overflow bit is a one, the 48- or 96-bit fraction right-shifts one position, end-off, and the overflow bit inserts into the high-order bit position. In such a case, the exponent increases by one to compensate for the shift.

The intermediate exponent and the intermediate fraction (with its sign) then transfer as the final result to the Xk register.

FP End Cases

Tables 2-11 through 2-19 list FP end cases. The nomenclature used is as follows.

N   Standard FP number: $(3000)_{16} \leq$ exponent $< (5000)_{16}$ nonzero, coefficient normalized or unnormalized

0   Zero: sign bit followed by 63 zero bits

Z1   Zero: FP numbers with exponents in the range $0000_{16} \leq$ exponent $< 1000_{16}$

Z2   Underflow, zero: FP numbers with exponents in the range $1000_{16} \leq$ exponent $< 3000_{16}$

Z3   Zero: an unnormalized FP number with a zero coefficient and a standard exponent, that is, $3000_{16} \leq$ exponent $< 5000_{16}$

INF   FP numbers with exponents in the range $5000_{16} \leq$ exponent $< 7000_{16}$

∞   Infinite: the nonstandard FP number sign, $5000\ 0000\ 0000\ 0000_{16}$

INDEF   FP numbers with exponents in the range $7000_{16} \leq$ exponent $\leq 7FFF_{16}$

+IND   Indefinite: the nonstandard FP number $7000\ 0000\ 0000\ 0000_{16}$

INDC   A result of indefinite generated by the FP compare instruction, that is, a value for X1R $= 8000\ 0000_{16}$

S   Algebraic sum of two FP numbers (excluding Z3)

D   Algebraic difference of two FP numbers (excluding Z3)

P   Algebraic product of two FP numbers (excluding Z3)

Q   Algebraic quotient of two FP numbers (excluding Z3)

DVF   Divide fault condition

OVL   Exponent overflow, FP

UNL   Exponent underflow, FP

LOS   Loss of significance, FP

IND   Indefinite, FP

Table 2-11. FP Compare Results (Xj is compared to Xk)

Xj \ Xk	Standard Numbers				Nonstandard Numbers			
	+N	-N	+Z3	-Z3	±Z1 ±Z2	+INF	-INF	±INDEF
+N	+D, +Z2< -D, -Z2> +Z3 =	<	+D, +Z2< +Z3 =	<	<	>	<	Note 1
-N	>	+D, +Z2< -D, -Z2> +Z3 =	>	-D, -Z2> +Z3 =	>	>	<	
+Z3	-D, -Z2> +Z3 =	<	+Z3 =	<	<	>	<	
-Z3	>	+D, +Z2< +Z3 =	>	+Z3 =	>	>	<	
±Z1 ±Z2	>	<	>	<	=	>	<	
+INF	<	<	<	<	<	Note 1	<	
-INF	>	>	>	>	>	>	Note 1	
±INDEF	Note 1							

Note: 1. FP branch instructions perform normal exit and record FP indefinite (UCR 61). FP compare instructions set X1 to INDC and record FP indefinite (UCR 61), except when UMR 61 is set traps are enabled, in which case X1 is unaltered.

Table 2-12. FP Sum Results, UM Clear

Xj / Xk	Standard Numbers				Nonstandard Numbers			
	+N	-N	+Z3	-Z3	±Z1 ±Z2	+INF	-INF	±INDEF
+N	+S +∞ OVL +0 UNL	+S +0 UNL +0 LOS	+S +0 UNL +0 LOS	+S +0 UNL +0 LOS	+N +0 UNL	+∞ OVL	+∞ OVL	+IND IND
-N		-S - OVL +0 UNL	-S +0 UNL +0 LOS	-S +0 UNL +0 LOS	-N +0 UNL	+∞ OVL	+∞ OVL	+IND IND
+Z3			+0 LOS	+0 LOS	+0 LOS	+∞ OVL	+∞ OVL	+IND IND
-Z3				+0 LOS	+0 LOS	+∞ OVL	+∞ OVL	+IND IND
±Z1 ±Z2			+0 LOS	+0 LOS	+0	+∞ OVL	+∞ OVL	+IND IND
+INF						+∞ OVL	+IND IND	+IND IND
+INF							+∞ OVL	+IND IND
±INDEF								+IND IND

Table 2-13. FP Sum Results, UM Set

Xk \ Xj	Standard Numbers				Nonstandard Numbers			
	+N	-N	+Z3	-Z3	±Z1 ±Z2	+INF	-INF	±INDEF
+N	+S +∞ OVL +Z2 UNL	+S ±Z2 UNL ±Z3 LOS	+S +Z2 UNL +Z3 LOS	+S +Z2 UNL +Z3 LOS	+N +Z2 UNL	+∞ OVL	-∞ OVL	+IND IND
-N		-S -∞ OVL +Z2 UNL	-S -Z2 UNL +Z3 LOS	-S -Z2 UNL ±Z3 LOS	-N -Z2 UNL	+∞ OVL +∞ OVL	-∞ OVL +∞ OVL	+IND IND
+Z3			+Z3 LOS	+Z3 LOS	+Z3 LOS	+∞ OVL	+∞ OVL	+IND IND
-Z3				+Z3 LOS	+Z3 LOS	+∞ OVL	+∞ OVL	+IND IND
±Z1 ±Z2			+Z3 LOS	+Z3 LOS	+0	+∞ OVL	+∞ OVL	+IND IND
+INF						+∞ OVL	+IND IND	+IND IND
+INF							-∞ OVL	+IND IND
±INDEF								+IND IND

Note: This chart is for traps disabled. For traps enabled, replace +IND with Xk.

Table 2-14. FP Difference Results, UM Clear

Xk \ Xj	Standard Numbers				Nonstandard Numbers			
	+N	-N	+Z3	-Z3	±Z1 ±Z2	+INF	-INF	±INDEF
+N	-D +0 UNL +0 LOS	+D +∞ OVL +0 UNL	+D +0 UNL +0 LOS	+D +0 UNL +0 LOS	+N +0 UNL	-∞ OVL	+∞ OVL	+IND IND
-N	-D -∞ OVL +0 UNL	+D +0 UNL +0 LOS	-D +0 UNL +0 LOS	-D +0 UNL +0 LOS	-N +0 UNL	-∞ OVL	+∞ OVL	+IND IND
+Z3	-D +0 UNL +0 LOS	+D +0 UNL +0 LOS	+0 LOS	+0 LOS	+0 LOS	+∞ OVL	+∞ OVL	+IND IND
-Z3	-D +0 UNL +0 LOS	+D +0 UNL +0 LOS	+0 LOS	+0 LOS	+0 LOS	+∞ OVL	+∞ OVL	+IND IND
±Z1 ±Z2	-N +0 UNL	+N +0 UNL	+0 LOS	+0 LOS	+0	+∞ OVL	+∞ OVL	+IND IND
+INF	+∞ OVL	+∞ OVL	+∞ OVL	+∞ OVL	+∞ OVL	+IND IND	+∞ OVL	+IND IND
-INF	-∞ OVL	-∞ OVL	-∞ OVL	-∞ OVL	-∞ OVL	-∞ OVL	+IND IND	+IND IND
±INDEF	+IND IND	+IND IND	+IND IND	+IND IND	+IND IND	+IND IND	+IND IND	+IND IND

Table 2-15. FP Difference Results, UM Set

Xk \ Xj	Standard Numbers				Nonstandard Numbers			
	+N	-N	+Z3	-Z3	±Z1 ±Z2	+INF	-INF	±INDEF
+N	+D ±Z2 UNL +Z3 LOS	+D +∞ OVL +Z2 UNL	+D +Z2 UNL +Z3 LOS	+D +Z2 UNL +Z3 LOS	+N +Z2 UNL	-∞ OVL	+∞ OVL	+IND IND
-N	-D -∞ OVL -Z2 UNL	+D ±Z2 UNL ±Z3 LOS	-D -Z2 UNL +Z3 LOS	-D -Z2 UNL +Z3 LOS	-N -Z2 UNL	-∞ OVL	+∞ OVL	+IND IND
+Z3	-D -Z2 UNL +Z3 LOS	+D +Z2 UNL +Z3 LOS	+Z3 LOS	+Z3 LOS	+Z3 LOS	+∞ OVL	+∞ OVL	+IND IND
-Z3	-D +Z2 UNL +Z3 LOS	+D +Z3 UNL +Z3 LOS	+Z3 LOS	+Z3 LOS	+Z3 LOS	+∞ OVL	+∞ OVL	+IND IND
±Z1 ±Z2	-N -Z2 UNL	+N +Z2 UNL	+Z3 LOS	+Z3 LOS	+0	+∞ OVL	+∞ OVL	+IND IND
+INF	+∞ OVL	+∞ OVL	+∞ OVL	+∞ OVL	+∞ OVL	+IND IND	+∞ OVL	+IND IND
-INF	-∞ OVL	-∞ OVL	-∞ OVL	-∞ OVL	-∞ OVL	-∞ OVL	+IND IND	+IND IND
±INDEF	+IND IND	+IND IND	+IND IND	+IND IND	+IND IND	+IND IND	+IND IND	+IND IND

Note: This chart is for traps disabled. For traps enabled, replace +IND with Xk.

Table 2-16. FP Product Results, UM Clear

Xj / Xk	Standard Numbers				Nonstandard Numbers			
	+N	−N	+Z3	−Z3	±Z1 ±Z2	+INF	−INF	±INDEF
+N	+P +∞ OVL +0 UNL +Z3	−P − OVL +0 UNL +Z3	+∞ OVL +0 UNL +Z3	+∞ OVL +0 UNL +Z3	+0	−∞ OVL	+∞ OVL	+IND IND
−N		+P +∞ OVL +0 UNL +Z3	−∞ OVL +0 UNL +Z3	+∞ OVL +0 UNL +Z3	+0	−∞ OVL	+∞ OVL	+IND IND
+Z3			+∞ OVL +0 UNL +Z3	−∞ OVL +0 UNL +Z3	+0	+∞ OVL	+∞ OVL	+IND IND
−Z3				+∞ OVL +0 UNL +Z3	+0	−∞ OVL	+∞ OVL	+IND IND
±Z1 ±Z2			+0	+0	+0	+IND IND	+IND IND	+IND IND
+INF						+∞ OVL	−∞ OVL	+IND IND
−INF							+∞ OVL	+IND IND
±INDEF								+IND IND

s, UM Set

3	-Z3	±Z1 ±Z2	+INF	-INF	+INDEF
		Nonstandard Numbers			
DVF	Xk DVF	Xk DVF	+0	+0	+IND IND
DVF	Xk DVF	Xk DVF	+0	+0	+IND IND
DVF	Xk DVF	Xk DVF	+0	+0	+IND IND
DVF	Xk DVF	Xk DVF	+0	+0	+IND IND
	+0	Xk DVF	+0	+0	+IND IND
OVF	-∞ OVF	+∞ OVF	+IND IND	+IND IND	+IND IND
OVF	+∞ OVF	-∞ OVF	+IND IND	+IND IND	+IND IND
IND	+IND IND	+IND IND	+IND IND	+IND IND	+IND IND

isabled. For traps enabled, replace +IND with Xk.

Table 2-17. FP Product Results, UM S

Xj / Xk	+N	-N	+Z3	-Z
	Standard Numbers			
+N	+P / +∞ OVL / +Z2 UNL / +Z3	-P / -∞ OVL / +Z2 UNL / +Z3	+P OVL / +Z2 UNL / +Z3	-P / +Z2 / +Z3
-N		+P / +∞ OVL / +Z2 UNL / +Z3	-P OVL / -Z2 UNL / +Z3	+P / +Z2 / +Z3
+Z3			+P OVL / +Z2 UNL / +Z3	-P / -Z2 / +Z3
-Z3				+P / +Z2 / +Z3
±Z1 ±Z2			+0	+0
+INF				
-INF				
+INDEF				

Note: This chart is for traps disabled.

60000133 B

r

	Nonstandard Numbers			
	±Z1 ±Z2	+INF	-INF	+INDEF
F	Xk DVF	+0	+0	+IND IND
L F	Xk DVF	+0	+0	+IND IND
F	Xk DVF	+0	+0	+IND IND
F	Xk DVF	+0	+0	+IND IND
	Xk DVF	+0	+0	+IND IND
F	Xk DVF	+IND IND	+IND IND	+IND IND
F	Xk DVF	+IND IND	+IND IND	+IND IND
ID	Xk DVF	+IND IND	+IND IND	+IND IND

Program Monitoring

The CP provides a debug feature to aid the debug

The debug feature causes a debug trap interrupt (
a given type into a given PVA range occurs. The
ranges and 5 access types for simultaneous debugg

Debug

When enabled, the debug feature tests all executiv
a list of up to 32 entries of selected access type an
match is found, UCR bit 56 sets and, if enabled, a

A debug address range may span an entire process
byte field within the segment. Any or all of the fo

- Data read

- Data write

- Instruction fetch (excluding target instruction fe

- Branch or return target instruction fetch (exclu

- Call target instruction fetch

Debug is enabled by setting UMR bit 56 and the t
controlled by the following.

- Debug list – lists access types and PVA ranges

- Debug list pointer register – PVA of first entry

- Debug mask register – enables/disables any or

- Debug index registers – record number of debu
  of a single instruction

Debug List

The debug list has up to 32 double-word entries (r
at a word boundary. The end of list (debug code bi
bits in the same debug code have been interpreted
types can be selected for debug by the debug code.
type of access privilege required (read, write, or ex

Debug List Pointer Register

This register is a process register containing a PV/
entry.

```
0         78              1920            3132            ──────                      63
┌─────────┬────────────────────┬────────────────┬──────────────────────────────────┐
│   DC    │////////////////////│      SEG       │            BN (Low)              │
└─────────┴────────────────────┴────────────────┴──────────────────────────────────┘

0                                            3132                                    63
┌──────────────────────────────────────────────┬──────────────────────────────────┐
│////////////////////////////////////////////// │            BN (High)             │
└──────────────────────────────────────────────┴──────────────────────────────────┘
```

Field	Description
BN	Byte numbers of the first and last bytes in the contiguous byte field in memory to which the debug code applies.
SEG	Process segment number to which the debug code applies.
DC	Debug code (listed below).

DC Bit	Operations Triggering Scan
0	Data read. Activates debug scan on all CM read accesses.
1	Data write. Activates debug scan on all CM write accesses.
2	Instruction fetch. Activates debug scan on all CM execute accesses, after instruction fetch.
3	Branch or return target instruction fetch, if branch occurs. PVA bracket applies to the PVA branched to. Call (B5/BO) target addresses are excluded. Branch address generated during compare and swap instruction (B4) is excluded.
4	Call (B5/BO) target instruction fetch. Address bracket applies to address of the called procedure. For the call indirect (B5) instruction, this is the code base pointer address; for the call relative (BO) instruction, this is the address in the P register plus displacement.
5	End of list. Denotes the last debug list entry.

Figure 2-12. Debug List Entry

Debug Index Register

This register is a process register which increments as the debug scan proceeds. It contains a 6-bit word index that is added to the DLP register contents. This generates the debug list entry addresses during a debug scan initiated by each instruction accessing virtual memory. The debug index register format is as follows.

```
58                    63
┌──────────────────────┐
│   6-BIT WORD INDEX    │
└──────────────────────┘
```
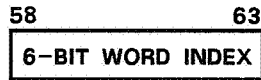
Debug Mask Register

This register (figure 2-13) is a process register which activates the access types selected in the debug code field of a debug list entry. Each access type selected in the debug code is activated for debugging only if the corresponding debug mask bit is also set, as shown in figure 2-13.
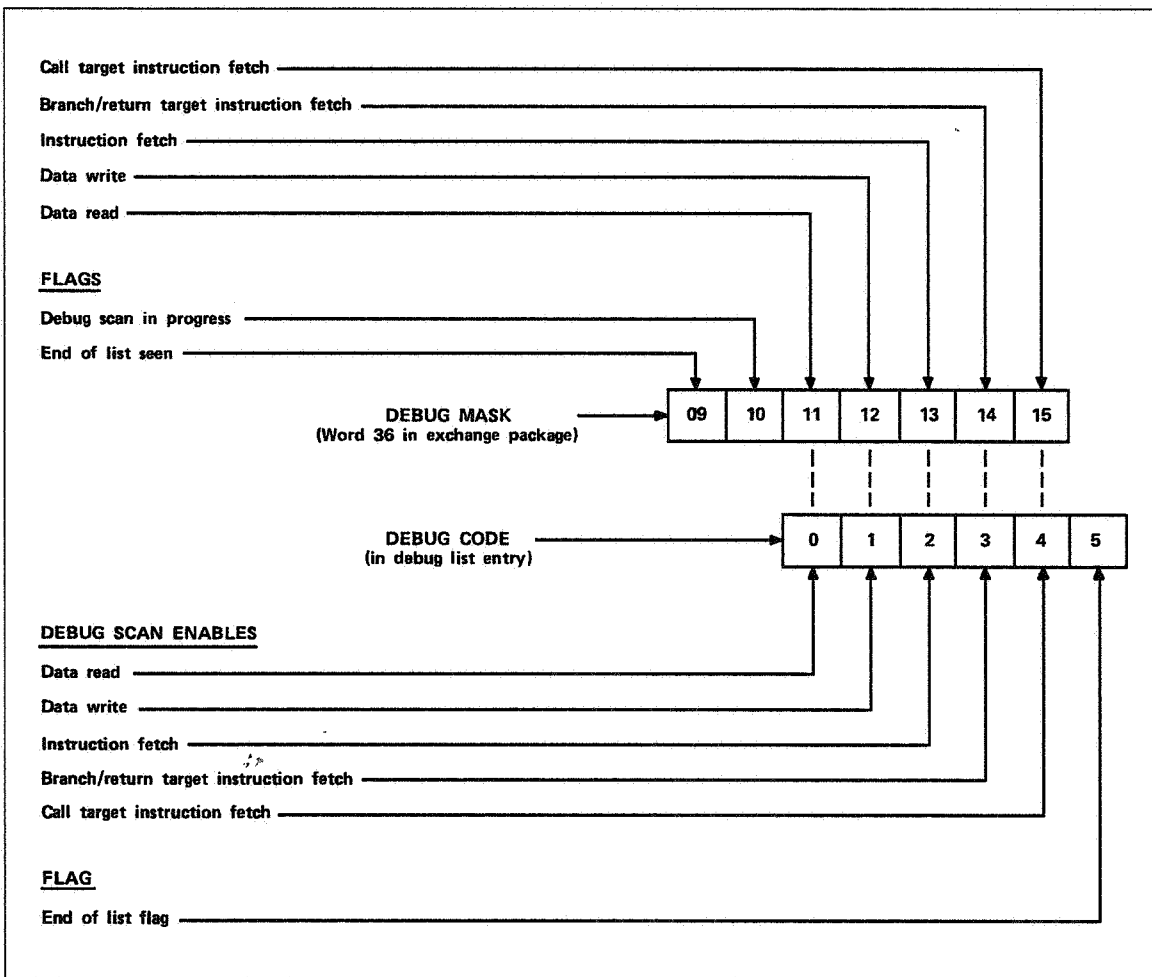
Figure 2-13. Debug Condition Select

Enabling Debug

The debug operation may be enabled by any of the following.

● Exchange to a Virtual State process where traps are enabled and UMR 56 is set.

● Return (04) to a Virtual State process where the return operation enables traps and UMR 56 is set in the user mask register being loaded.

● Set UMR 56 by way of Copy-to-State-Register instruction when traps are enabled. The debug flags and index must be zero prior to execution of the Copy-To-State-Register instruction or an undefined initial debug scan follows the instruction.

● Enable traps by way of Copy-to-State-Register instruction when UMR 56 is set. The debug flags and index must be zero prior to execution of the Copy-to-State-Register instruction or an undefined initial debug scan follows the Copy-to-State-Register instruction.

Debug Scan Operation

Debug conditions apply to specific instructions as described in table 2-20. BN(low) and BN(high) are matched against the address of the leftmost byte of a piece of information only, whether it is a word, half word, byte string, or 16-bit instruction. The match is as follows.

$$BN(low) \leq Address \leq BN(high)$$

If BN(low) exceeds BN(high) in any debug list entry, the scan proceeds to the next double-word entry. If either BN(low) or BN(high) bit 32 is set, the comparison results are undefined.

The CP starts the debug list scan (after instruction fetch but before instruction execution) if all of the following conditions exist.

● Traps are enabled.

● Debug mask bit in user mask register is set (UMR 56).

● One or more bits in debug mask register are applicable to the type of access. Refer to table 2-18.

● End-of-list flag in debug mask register is clear.

When the debug scan initiates, the debug scan-in-progress flag is clear and the debug index register is zero. The debug scan locates a debug list entry by adding the debug list pointer and debug list index registers. The debug scan proceeds (not necessarily in the exact order given) as follows.

1. Set debug scan-in-process flag in debug mask register.

2. Read first half of debug list entry. If end-of-list code, set end-of-list-seen flag in debug mask register.

3. Add one to debug index register and read second half of debug list entry.

4. Set UCR 56 (triggering a trap interrupt) if a debug list match is found as follows.

    ●   Accessed PVA is within address bracket of debug list entry.

    ●   One or more debug code bits of debug list entry match the type of access, with the corresponding bit set in the debug mask register.

5. If step 4 triggered a debug interrupt, proceed to step 8.

6. If end-of-list flag is set, or 32 entries have been read, proceed to step 8.

7. Add one to debug index register and repeat from step 2.

8. Clear debug index register, debug scan-in-process flag, and end-of-list-seen flag to complete the debug scan.

9. Execute the instruction triggering the scan.

Debug list scanning prior to instruction execution includes all instruction results except the following (which may occur before the debug scan completes).

1. Setting the page-used bit, either explicitly as in test and set page (16) instruction, or implicitly as with any instruction.

2. Setting of condition register bits.

3. Rounding of A0 on call instructions.

4. Storing the environment into SFSA on call instructions (a debug trap also stores the environment into SFSA).

The exception testing and debug scan are not constrained to occur in any given sequence relative to each other. Two or more matches within the same entry produce only one trap. The traps due to execution testing may occur concurrently with a debug trap (several bits set in MCR and/or UCR) or separately, either before or after the debug scan.

Interrupts During Debug Scan

The debug index and flags provide the means for properly initiating, resuming, and terminating debug scan operations, particularly when an instruction's execution has been inhibited by one or more interrupts. These interrupts may either be trap or exchange interrupts.

Exchange interrupts cause the flags and debug index register to be stored in the exchange package, for example, to allow resumption of a partially completed debug entry list scan.

On trap interrupts, the processor retains the flags and index register to allow proper completion of the debug scan upon return from the trap interrupt.

When enabling traps during the processing of a debug trap interrupt, software must not reenable debug to prevent loss of integrity of the interrupted debug scan.

Debug-Software Interaction, Debug Enabled

The following items describe interactions with the debug facility that are available with debug enabled.

- Debug mask bits 11 through 15 of debug mask register may be set or cleared by way of a Copy-to-State-Register instruction; the new bits will be in effect for the debug scan on the instruction following the copy instruction.

- Any copy to the debug flags or index must clear both flags and the index or the following debug scan is undefined.

- UMR 56 may be cleared or traps disabled by way of a Copy-to-State-Register instruction with no scan performed on the instruction following the copy instruction.

- A return instruction disables debug by loading a user mask register with bit 56 clear, or by entering CYBER 170 State. In such a case, no scan is performed on the next instruction.

Debug-Software Interaction, Debug Disabled

When debug is disabled after a debug match (after which an exchange or trap interrupt occurs), the scan-in-progress flag sets and, if applicable, the end-of-list flag sets. In this case, the following software action may be taken through a Copy-to-State-Register instruction after a trap interrupt, or through altering the exchange package in CM after an exchange interrupt.

- Any of the debug mask register bits 11 through 15 may be set or cleared. The new mask bits affect the first debug scan when debug is reenabled for this process.

- Debug flags and index may be cleared to reinitiate the debug scan from the beginning when debug is reenabled.

- Debug index may be modified by multiples of 2 as the final value is greater than or equal to 1 and less than or equal to 61.

- End-of-list-seen flag may be set to terminate the current debug scan when debug is reenabled. The scan-in-progress flag may (but need not) be altered when setting this flag.

- The end-of-list-seen flag may be cleared and the scan-in-progress flag set to continue a scan that terminated. The debug index may also be modified by multiples of 2 as long as the final value is greater than or equal to 1 and less than or equal to 61.

When a debug match is absent, the debug may also be disabled by any of the following.

- Trap interrupt

- Exchange interrupt

- Copy-to-State-Register instruction which clears UMR 56 or disables traps

- Call to CYBER 170 State

- Return which clears UMR 56

- Return to CYBER 170 State

When a debug match is absent, and the debug is disabled by any of the six methods described in the preceding paragraph, only the following software actions may be taken.

- Any of the debug mask register bits 11 through 15 may be set or cleared. The new mask bits affect the first debug scan when debug is enabled for this process.

- The debug flags and index may be cleared to reinitiate the full debug scan when debug is enabled.

Table 2-20. Debug Conditions

Opcode	Mnemonic	Instruction	Debug Condition	
Bit 0: Data Read				
DS	LBYTS	Load bytes, immediate	$LO \leq Aj + Xi + D$	$\leq HI$
A2	LXI	Load word, indexed	$LO \leq Aj + 8*Xi + 8*D$	$\leq HI$
82	LX	Load word	$LO \leq Aj + 8*Q$	$\leq HI$
A4	LBYT	Load bytes	$LO \leq Aj + Xi + D$	$\leq HI$
88	LBIT	Load bit	$LO \leq Aj + Q + X0/8$	$\leq HI$
A0	LAI	Load address, indexed	$LO < Aj + Xi = D$	$\leq HI$
84	LA	Load address	$LO \leq Aj + Q$	$\leq HI$
80	LMULT	Load multiple	$LO \leq Aj + 8*Q$	$\leq HI$
70	ADDN	Decimal sum	$LO \leq Aj + 01$ $LO \leq Ak + 02$	$\leq HI$ $\leq HI$
71	SUBN	Decimal difference	$LO \leq Aj + 01$ $LO \leq Ak + 02$	$\leq HI$ $\leq HI$
72	MULN	Decimal product	$LO \leq Aj + 01$ $LO \leq Ak + 02$	$\leq HI$ $\leq HI$
73	DIVN	Decimal quotient	$LO \leq Aj + 01$ $LO \leq Ak + 02$	$\leq HI$ $\leq HI$
E4	SCIN	Decimal scale	$LO \leq Aj + 01$	$\leq HI$
E5	SCLR	Decimal scale, rounded	$LO \leq Aj + 01$	$\leq HI$
74	CMPN	Decimal compare	$LO \leq Aj + 01$ $LO \leq Ak + 02$	$\leq HI$ $\leq HI$
77	CMPB	Byte compare	$LO \leq Aj + 01$ $LO \leq Ak + 02$	$\leq HI$ $\leq HI$
E9	CMPC	Byte compare, collated	$LO \leq Aj + 01$ $LO \leq Ak + 02$ $LO \leq Ai + D$	$\leq HI$ $\leq HI$ $\leq HI$
F3	SCNB	Byte scan while nonmember	$LO \leq Ak + 01$ $LO \leq Ai + D$	$\leq HI$ $\leq HI$
E8	TRANB	Byte translate	$LO \leq Aj + 01$ $LO \leq Ai + D$	$\leq HI$ $\leq HI$
76	MOVB	Move bytes	$LO \leq Aj + 01$	$\leq HI$

(Continued)

Table 2-20. Debug Conditions *(Continued)*

Opcode	Mnemonic	Instruction	Debug Condition	
Bit 0: Data Read (Contd)				
ED	EDIT	Edit	LO ≤ Aj+01	≤ HI
			LO ≤ Ai+D	≤ HI
75	MOVN	Numeric move	LO ≤ Aj+01	≤ HI
F4	CALDF	Calculate subscript and add	LO ≤ Aj+01	≤ HI
			LO ≤ Ai+D	≤ HI
B5	CALLSEG	Call indirect	LO ≤ Aj+8*Q	≤ HI
04	RETURN	Return	LO ≤ A2	≤ HI
06	POP	Pop	LO ≤ A2	≤ HI
14	LBSET	Test and set bit	LO ≤ Aj+X0/8	≤ HI
B4	CMPXA	Compare swap	LO ≤ Aj	≤ HI
FA	CMPI	Compare immediate data	LO ≤ Ak+01	≤ HI
FB	ADDI	Add immediate data	LO ≤ Ak+01	
Bit 1: Data Write				
DS	SBYTS	Store bytes, immediate	LO ≤ Aj+Xi+D	≤ HI
A3	SXI	Store word, indexed	LO ≤ Aj+8*Xi+8*D	≤ HI
83	SX	Store word	LO ≤ Aj+8*Q	≤ HI
A5	SBYT	Store bytes	LO ≤ Aj+Xi+D	≤ HI
89	SBIT	Store bit	LO ≤ Aj+Q+X0/8	≤ HI
A1	SAI	Store address, indexed	LO ≤ Aj+Xi+D	≤ HI
85	SA	Store address	LO ≤ Aj+Q	≤ HI
81	SMULT	Store multiple	LO ≤ Aj+8*Q	≤ HI
70	ADDN	Decimal sum	LO ≤ Ak+02	≤ HI
71	SUBN	Decimal difference	LO ≤ Ak+02	≤ HI
72	MULN	Decimal product	LO ≤ Ak+02	≤ HI
73	DIVN	Decimal quotient	LO ≤ Ak+02	≤ HI
E4	SCLN	Decimal scale	LO ≤ Ak+02	≤ HI

(Continued)

Table 2-20. Debug Conditions *(Continued)*

Opcode	Mnemonic	Instruction	Debug Condition	
Bit 1: Data Write (Contd)				
E5	SCLR	Decimal scale, rounded	LO ≤ Ak+02	≤ HI
EB	TRANB	Byte translate	LO ≤ Ak+02	≤ HI
76	MOVB	Move bytes	LO ≤ Ak+02	≤ HI
ED	EDIT	Edit	LO ≤ Ak+02	≤ HI
75	MOVN	Numeric move	LO ≤ Ak+02	≤ HI
B5	CALLSEG	Call indirect	LO ≤ AQ+7,mod 8	≤ HI
B0	CALLREL	Call relative	LO ≤ AQ+7,mod 8	≤ HI
14	LBSET	Test and set bit	LO ≤ Aj+X0/8	≤ HI
B4	CMPXA	Compare swap	LO ≤ Aj	≤ HI
F9	MOVI	Move immediate data	LO ≤ Ak+01	≤ HI
FB	ADDI	Add immediate data	LO ≤ Ak+01	≤ HI

Bit 2: Instruction Fetch

All instruction fetches initiate a debug trap interrupt when the PVA accessed is
within an address bracket on the debug list. However: 1) The load bytes relative
(DS) reference to P+Q is not detected. 2) Unimplemented instruction, program
error, and execute algorithm is not necessarily detected. 3) The descriptors for BDP
instructions are not detected. 4) The test is applied for each instruction rather than
for each instruction word.

Bit 3: Branch Target Instruction

94	BRXEQ	Branch on equal	LO ≤ P+2*Q	≤ HI
95	BRXNE	Branch on not equal	LO ≤ P+2*Q	≤ HI
96	BRXGT	Branch on greater than	LO ≤ P+2*Q	≤ HI
97	BRXGE	Branch on greater than or equal	LO ≤ P+2*Q	≤ HI

(Continued)

Table 2-20. Debug Conditions *(Continued)*

Opcode	Mnemonic	Instruction	Debug Condition		
Bit 3: Branch Target Instruction					
90	BRREQ	Branch on half-word equal	LO $\leq$ P+2*Q	$\leq$ HI
91	BRRNE	Branch on half-word not equal	LO $\leq$ P+2*Q	$\leq$ HI
92	BRRGT	Branch on half-word greater than	LO $\leq$ P+2*Q	$\leq$ HI
93	BRRGE	Branch on half-word greater than or equal	LO $\leq$ P+2*Q	$\leq$ HI
9C	BRINC	Branch and increment	LO $\leq$ P+2*Q	$\leq$ HI
9D	BRSEG	Branch on segments unequal	LO $\leq$ P+2*Q	$\leq$ HI
2E	BRREL	Branch relative	LO $\leq$ P+2*Xk	$\leq$ HI
2F	BRDIR	Branch intersegment	LO $\leq$ Aj+2*Xk	$\leq$ HI
98	BRFEQ	FP branch on equal	LO $\leq$ P+2*Q	$\leq$ HI
99	BRFNE	FP branch on not equal	LO $\leq$ P+2*Q	$\leq$ HI
9A	BRFGT	FP branch on greater than	LO $\leq$ P+2*Q	$\leq$ HI
9B	BRFGE	FP branch on greater than or equal	LO $\leq$ P+2*Q	$\leq$ HI
9E	BROVR	FP branch on overflow	LO $\leq$ P+2*Q	$\leq$ HI
9E	BRUND	FP branch on underflow	LO $\leq$ P+2*Q	$\leq$ HI
9E	BRINF	FP branch on indefinite	LO $\leq$ P+2*Q	$\leq$ HI
04	RETURN	Return	LO $\leq$ FINAL P	$\leq$ HI
9F	BRCR	Branch on condition register	LO $\leq$ P+2*Q	$\leq$ HI
Bit 4: Call Target Instruction Fetch					
B5	CALLSEG	Call indirect	LO $\leq$ CBP	$\leq$ HI
B0	CALLREL	Call relative	LO $\leq$ P+8*Q,mod 8	$\leq$ HI

Virtual and CM Programming

Figure 2-14 shows how a PVA converts to a SVA, and then to a RMA. The operating system provides a segment descriptor table and a system page table to make the conversion possible.
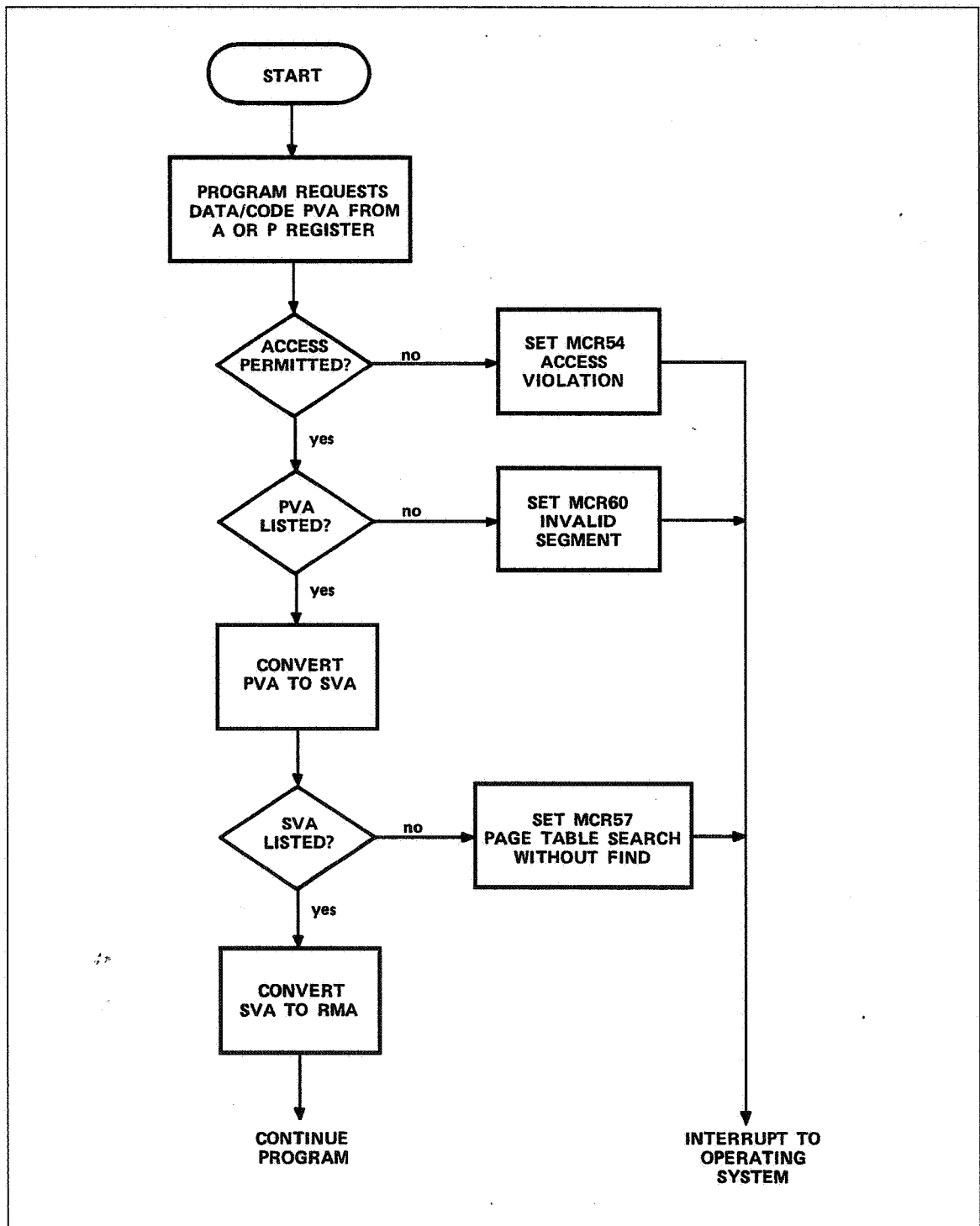
Figure 2-14. CM Addressing from CP

Process Virtual Memory

To the user, memory is a set of segments in process virtual memory space. Each segment is a contiguous byte string of 2^{31}-1 bytes. A maximum of 4096 segments may exist for each process at any one time. During process execution, the CP presents PVAs for hardware translation, first to SVAs, and then to RMAs where the requested data resides in CM. The PVA is the address seen by the user and used by executing code. The PVA may identify a P register address or a CM operand address. The PVA format is shown in figure 2-15.

```
16   19 20           31 32 33                                    63
 ┌─────┬──────────────┬─┬────────────────────────────────────────┐
 │ RN  │     SEG      │0│                   BN                     │
 └─────┴──────────────┴─┴────────────────────────────────────────┘
```

Field	Name	Description
RN	Ring Number	Access privilege indicator that selectes 1 of 16 rings. Refer to Access Protection in this section for further information.
SEG	Segment	Process segment number. The same shared segment may be addressed by different segment numbers.
BN	Byte number	Byte location within the 2^{31}-1 bytes in a segment. Bit 32 in the final PVA used is a validity indicator and must be a zero or the PVA is rejected. Bit 32 in an A register may be a one provided indexing or displacement changes this bit to a zero in the final PVA.

Figure 2-15. Process Virtual Address (PVA) Format

System Virtual Memory

The operating system sees virtual memory as a set of system-wide active segments that totals all of the process segments. An active segment is the virtual memory division for uniquely identifying system data. SVAs address the active segments. Figure 2-16 shows how hardware translates a PVA to an SVA.

In the translation, an active segment identifier (ASID) replaces the RN and SEG fields of the PVA. All segments active in the system are assigned unique ASIDs to ensure unique system data. The 16-bit ASID field in the SVA defines a total of 65 536 active segments that may simultaneously exist in the system. The ASID resides in the segment descriptor table that the operating system maintains for each process. Any number of ASIDs can be listed in various process segment descriptor tables against the same segment. This forms the basis for code sharing.

During the translation, hardware also verifies permission to access the segment containing the SVA. This occurs by way of the access protection attributes listed in the segment descriptor table for that segment.

```
16    1920            313233                          63
| RN |    SEG      |0|              BN                 |  PVA

16                  313233                          63
|        ASID       |0|              BN                 |  SVA
```

Field	Name	Description
ASID	Active segment identifier	Global name that uniquely identifies any active segment in the system.
BN	Byte number	Byte location within a segment.
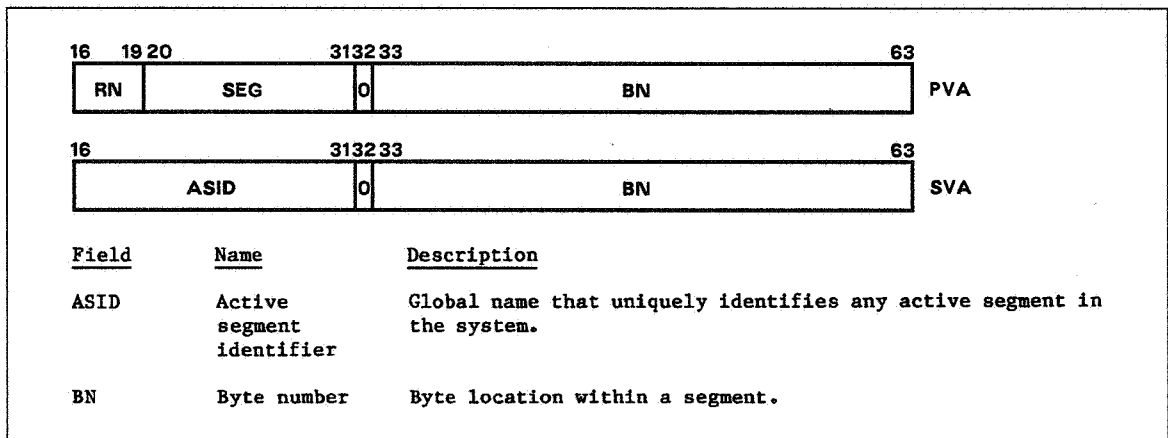
Figure 2-16. System Virtual Address (SVA) Format

Real Memory

The operating system sees real memory as pages in CM or external mass storage. It maintains the necessary tables to identify, address, and retrieve the stored information when an executing process requests it. A page ranges in size from 2K to 16K bytes, as selected at system initialization via the page size mask register.

CM corresponds with virtual memory by page frames. Page frames are the same size as pages, and provide the means for swapping pages between CM and external mass storage. A page frame may be empty, contain new information being created, or contain a copy of a page from external mass storage that is being examined, modified, or executed.

The operating system manages CM by way of demand paging. In demand paging, the CP retrieves the requested page(s) from external mass storage to CM as needed for process execution. This frees the operating system from having to collect in CM all the pages necessary to complete process execution. In operation, the CP executes a process until a page fault occurs, whereby the CP switches execution to another process while it retrieves the page from external mass storage.

The CP always retrieves a copy of a page, and not the page itself, from external mass storage. Pages in such storage remain unaltered unless or until the CP overwrites the new page back to its location in external mass storage. (Hardware keeps track of pages in CM which are still true copies of pages in external mass storage.) The CP writes pages back to external mass storage to make room for new pages. The operating system identifies candidate pages for transfer by way of an algorithm which determines the least-used and least-recently-used pages.

Hardware uses the system page table in CM (described under Address Tables later in this chapter) to convert SVAs to RMAs and to complete address translation. In the SVA-to-RMA conversion, hardware uses a hashing algorithm to replace the ASID and PN fields of the SVA with a system/page identifier (SPID) (refer to figure 2-17). The SPID resides in the single system page table that the operating system maintains for all processes.

Once obtained, the SPID provides an index into a list of coincident hashed entries in the system page table. The CP linearly searches up to 32 coincident entries in comparing the true SPID (maintained by the operating system) to the system page table entries. The search continues until a valid page with the requested entry is found, or until 32 entries have been searched. If the requested entry is not found among the 32 entries, a page fault occurs and the CP retrieves the page and accompanying entry from external mass storage.

A successful search of the system page table results in identifying the desired page and the corresponding page frame address listed with the SPID. The page frame address lists the destination of the page in real memory. The page offset (carried directly from the SVA) identifies the memory word containing the requested byte. The rightmost three bits select the requested byte within the word. The RMA, formed by the page frame address and the page offset, has a format as shown in figure 2-18. The conversion of the virtual BN to the page number and page offset is shown in figure 2-19.

Field	Name	Description
ASID	Active segment identifier	Global name that uniquely identifies any segment active in the system.
BN	Byte number	Byte location within a segment.
PN	Page number	Part of BN which is hashed with the ASID to form the SPID. Its length varies from 15 to 22 bits, depending on system page size.
PO	Page offset	Byte location within a page. Length varies from 9 to 16 bits, depending on system page size.
SPID	Segment/ page identifier	Identifies page in a particular global segment for conversion to RMA.

Figure 2-17. Segment/Page Identifier (SPID) Format

Bits	Description
(a)42-63	Page frame address from system page table. Where the page offset has significant bits, the page frame address rightmost bits must be zeros in the system page table.
(b)48-63	Page offset from SVA, unchanged from PVA. Where the page frame address has corresponding significant bits, the page offset leftmost bits are set to zero by the page size mask.
(c)32	Must be zero.
33	Selects the port when CP has two memory ports.
34,35	Reserved.
36,37	Not used.
38-60	CM word address.
61-63	Byte address within a CM word.

Figure 2-18. Real Memory Address (RMA) Format
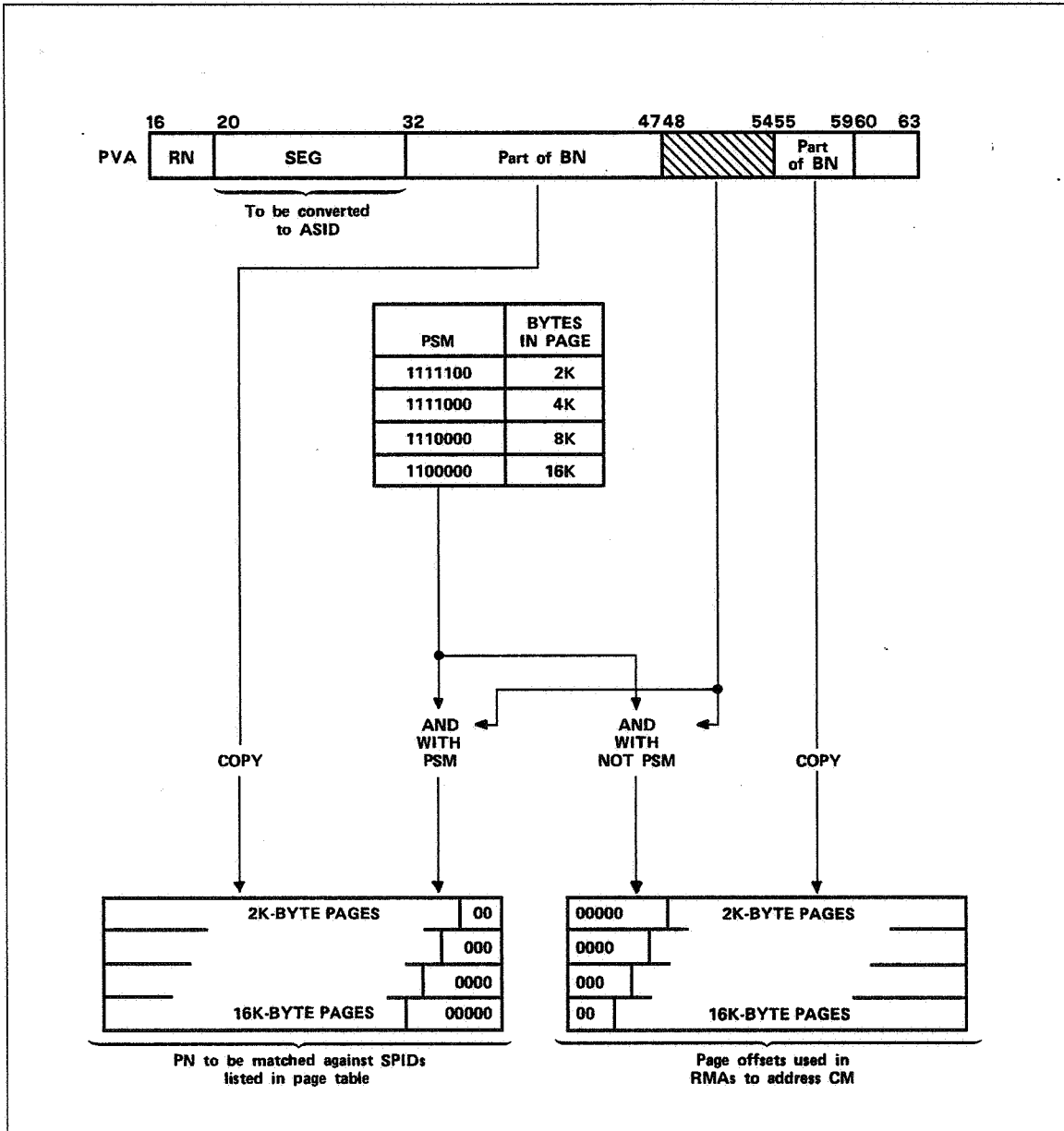
Figure 2-19. Virtual BN-to-Page Number/Page Offset Conversion

```
31  34
  32 33     3940                                        63
   01  LOCK
```
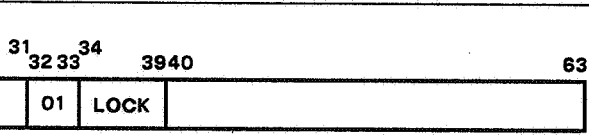
ator

ular segment.
ne bypass segment.

ype indicator
gment.
ecutable segment.
ed executable segment.
ged executable segment.

ype indicator
ent.
by key/lock.
led by key/lock.
segment, read not controlled by key/lock
if RP = 10).

ype indicator
ent.
 by key/lock.
lled by key/lock.

lcator
access, RN of PVA used may not be less than
cessed segment's SDE.
cess, RN of PVA used may not exceed R1 of
 segment's SDE.

lcator
access, RN of PVA used may not exceed R2 of
 segment's SDE.
ess, RN of PVA used may not exceed R2 of the
ment's SDE.

 which uniquely identifies each active
a

zero value indicates a no-lock condition.

iptor Table (SDT) Entry Format

Address Tables

The PVA-to-SVA-to-RMA translation
system keeping the following tables

- A segment descriptor table (SDT
  converting PVAs to SVAs and fo

- A system page table (SPT) comm
  for converting SVAs to RMAs ar

The PVA-to-RMA conversion with re
figure 2-20.

Each process obtaining call-indirect
chapter segments. Hardware accesse
instructions and trap interrupts to
access protection information.
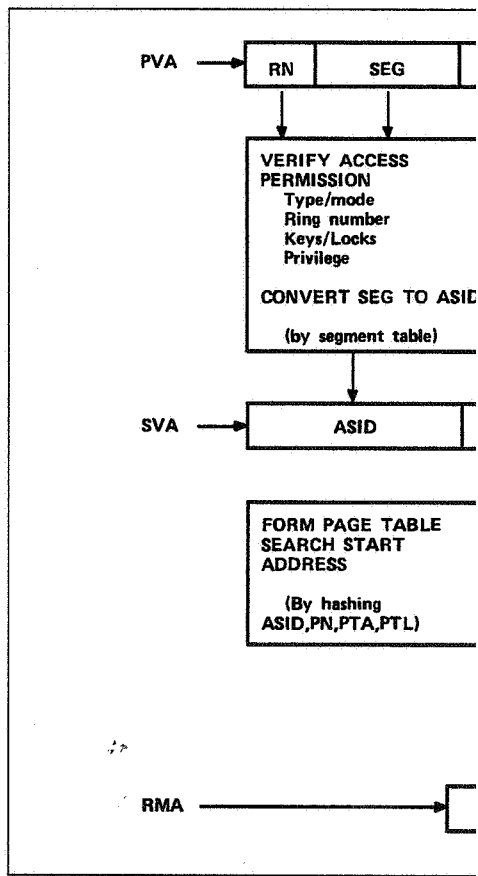
Figure 2-20

against the active system segments
'T starts at a word boundary and is
lange package.

; to the first SDT entry.

T entries, minus 1.

:ntry listing the corresponding ASID
1ent is a cache bypass segment.
: CP reads cache bypass segments
irmation to cache memory. The CP
:he bypass segments. The CP does
CM using an RMA stored in

; to each segment number. The
:h the STA to provide the required
:; thus, process segment numbers
ften the case).

or which there is no valid SDE.

rhich exceeds the segment table

attributes as established by software
pter for further information.)

without cache update or purge must
irtual State exchange operations to
:date cache data because cache is
erations treat target addresses as

System Page Table

The system has a system page table which list:
resident in CM against the physical address of
defined by the following.

- Page table address (PTA). This is an RMA
  which must be zero, modulo the page table

- Page table length (PTL). The page table ler

PTL	Page Table Entries
00 0000 0000 0000	512
00 0000 0000 0001	1 024
00 0000 0000 0011	2 048
00 0000 0000 0111	4 096
00 0000 0000 1111	8 192
⋮	⋮
00 1111 0011 1111	2 097 152
00 1111 0111 1111	4 174 304
11 1111 1111 1111	8 338 608

- Page size mask (PSM). The page size in m

PSM	Words in Page	Bytes
111 1100	256	
111 1000	512	
111 0000	1 024	
110 0000	2 048	

Page Table Search

Hardware converts SVAs to RMAs by searchir
matching the SVA (ASID-PN) being converted.
obtains a starting address for the 32-entry lin
the ASID, page table length, and virtual page
The page table search is necessary because m
search starting address.
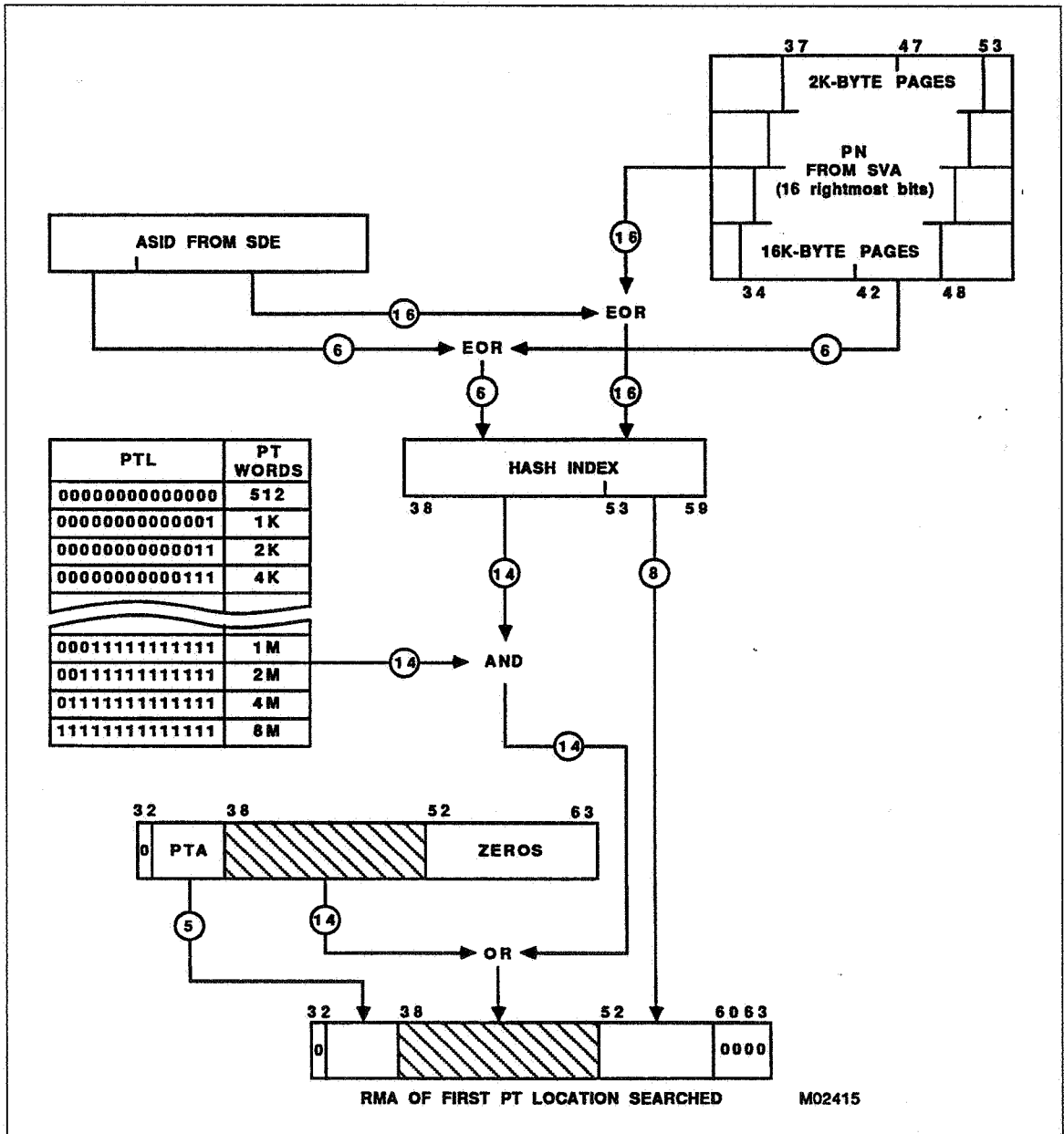
Figure 2-22. Page Table Search, Start RMA Formation

Page Table Entries

Page table entries (refer to PTE in figure 2-23) list all pages residing in CM by listing the SPID against the allocated CM page frame (physical) address. The operating system ensures that only one copy of any active page exists in CM at a time. The PTEs described also contain four control bits.

PTE Control Fields

The PTE contains four control bits: valid (V), continue (C), used (U), and modified (M). Hardware decodes and translates these fields during the page table search as follows.

- The valid (V) bit, when set, causes the PTE under examination to be tested for a SVA-RMA match. When clear, that PTE is ignored.

- The continue (C) bit, when set, causes the hardware search to continue with the next PTE. When clear, the hardware search may be halted after testing the current PTE.

- The used (U) bit is set by hardware whenever a PTE is used for address translation. This bit is cleared by software only.

- The modified (M) bit is set by hardware whenever a PTE is used for a write access to indicate that the page has been modified.

PTE Segment/Page Identifier Field

The segment/page identifier (SPID) is the PTE field tested for a match against the 38-bit ASID-PN combination from the SVA. When the page size exceeds 512 bytes, the PN is less than 22 bits. In this case, the unused rightmost bits are zeros to obtain proper alignment.

PTE Page Frame RMA Field

This 22-bit field is the page frame physical starting address. When the page size exceeds 512 bytes, the rightmost address bits must be zeros to obtain proper alignment.

Listing of Pages in Page Table

To continuously guarantee the same data for all processes, the operating system ensures that only one copy of a page resides in CM at a time. Therefore, each page has the same page frame address and is listed against a unique ASID in the page table. Also, the operating system lists pages in the page table so that a page table search always results in finding the requested page if it is in CM.

Due to the vast number of possible virtual pages, special techniques are used to list pages in the page table. The system page table length is, typically, 2 to 4 times the number of available page frames, and the 16-bit ASID numbers are assigned nonsequentially to facilitate page listing.

If a vacant spot is not found when attempting to list a page, the operating system takes further action such as cancelling a page to make room for the new page, changing the ASID number, or rearranging CM and enlarging the page table size.

```
    16              3132              4748        54
→  |    (SVA)ASID    |     SVA(PN)     |▨▨▨▨▨▨▨|


                                                    62
    0 1 2 3 4                          4142        61 63
   |V|C|U|M|         SPID              | PAGE FRAME ADDRESS |0|0|
```

Field	Name	Description
(SVA)ASID		Active segment identifies part of SVA.
SVA(PN)		Page number part of SVA.
V	Valid	(bit 00) Coded as follows: V = 1 PTE tested. V = 0 PTE ignored.
C	Continue	(bit 01) Coded as follows: C = 1 Search continued. C = 0 Search aborted.
U	Used	(bit 02) Cleared by software only. Indicates whether a page was accessed by the CPU: U = 1 An SVA-PTE match occurred. U = 0 No SVA-PTE match occurred.
M	Modified	(bit 03) Cleared by software only and coded as follows: M = 1 This PTE has been used for a write access since it was entered. M = 0 This PTE has not been used for a write access since it was entered.
SPID	Segment/page Identifier	(38-bits) Identical to the ASID-PN field combination from SVA. Refer to figure 2-14.
PAGE FRAME ADDRESS		Real memory starting address of the page frame. When the page size exceeds 2K bytes, the rightmost bits of this field are zeros (corresponding to the unused rightmost bits in SPID and SVA-PN).

X04326

Figure 2-23. Page Table Entry Format

Process Binding Section

Binding chapters bind different segments into one process. Each process has at least one binding section for use by hardware during the call indirect (CALLSEG,B5) instruction and during trap interrupts.

A binding section used for such purposes resides in a segment for which the SDE(RP) = 11; such a segment represents a binding segment and lists entry points (CBPs) into called code segments. The 64-bit CBPs (figure 2-24) reside at a word boundary and are addressed by PVAs as follows.

- For use by CALLSEG: The relevant CBP is addressed by $(Aj+8*Q)$ from the instruction.

- For use by trap interrupts: The relevant CBP is addressed by the trap pointer (TP) field from the exchange package.

A called procedure may have, and a trap interrupt target procedure always has, its own binding section. In such a case, the external procedure flag (EPF) sets in the relevant CBP. After a CBP with its EPF set is accessed, the word stored following that CBP loads into A3 as the new binding section pointer.

```
0      34      789  1112   1516   1920              3132                                           63
     ▨▨ VMID  ▨▨  R3  RN          SEG                              BN
              └─EPF
```

Field	Name	Description
VMID	Virtual machine identifier	Specifies the state of the CP after a call indirect instruction or after a trap interrupt, as follows: VMID = 0000 Virtual State. VMID = 0001 CYBER 170 State.
EPF	External procedure flag	When EPF = 1, hardware loads the word stored following the CBP as a new binding section pointer into register A3. When the CBP is used for trap interrupts, this bit must be a 1.
R3	Call limit ring number	Highest ring number from which a call indirect instruction to the listed target PVA may be issued. Initial P(RN) and Aj(RN) must both not exceed R3.
RN	CBP ring number	When zero, causes a ring-number zero-exception condition, setting MCR bit 60. When nonzero, prevents this exception with no other effects.
SEG, BN	Segment Number, Byte number	Part of PVA of entry point into branched-to code.

Figure 2-24. Code Base Pointer Format

Access Protection

Access protection is by way of segments, based on the following elements.

● Process segment table: The process segment table defines the process address space. A process may access only the segments listed in its segment descriptor table. Refer to Segment Descriptor Table in this chapter.

● Ring structure: All PVAs used have a 4-bit RN field specifying an access privilege from 1 to 15. (A lower RN in a PVA indicates a higher PVA access privilege.) All SDEs and CBPs list access privilege requirements. (A lower RN listed indicates a higher privilege requirement.)

● Type of access: SDE control fields permit the type of access attempted (read, write, or execute). Refer to figure 2-21.

● Execute access privilege: Some system instructions require special codes in SDE execute permission control fields. (A local/global execution privilege requirement.)

● Execute access mode: Some system instructions may execute only in Virtual State monitor mode.

● Keys and locks: SDE control fields may further specify that a 6-bit key number in the P register must equal a 6-bit lock number in the SDE.

● Code base pointers: Call indirect instructions and trap interrupts use additional access protection attributes in CBPs, which must be listed in binding section segments.

Figures 2-25 and 2-26 illustrate access protection as it pertains to the PVA-to-SVA conversion for the read/write and execute cases, respectively.

General access requirements:
    PVA(SEG) ≤ STL, SDE(VL = 10/11) SDE ADRS BIT 32 = 0

Read access requirements:
    PVA(RN) ≤ SDE(R2)
    SDE(RP) = 10/11
    SDE(RP) = 01, KEY - LOCK test met

Write access requirements:
    PVA(RN) ≤ SDE(R1)
    SDE(WP) = 10
    SDE(WP) = 01, KEY - LOCK test met
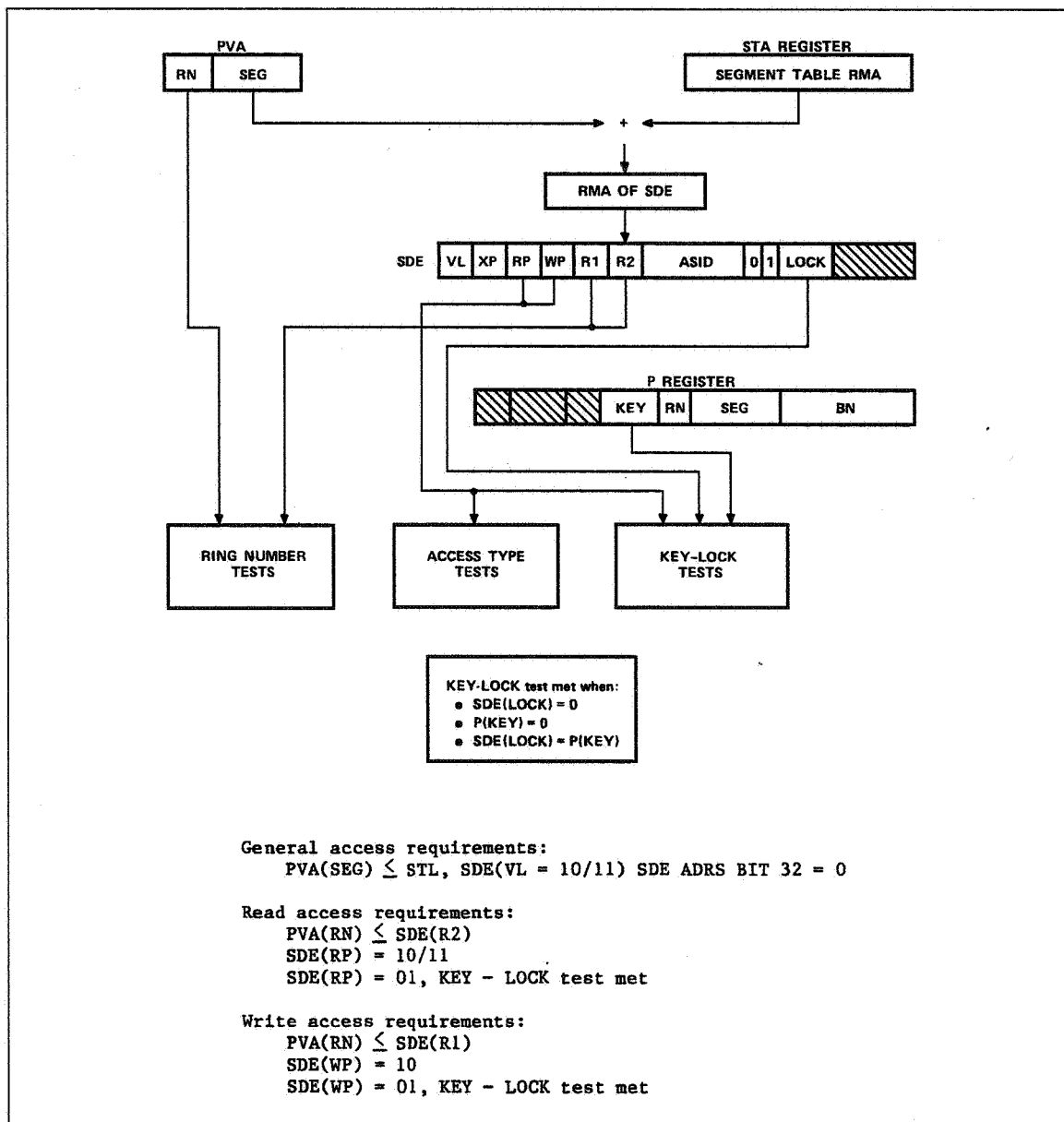
Figure 2-25. PVA-to-SVA Conversion, Read/Write

General access requirements:
PVA(SEG) $\leq$ STL, SDE(VL) = 10/11 SDE ADRS BIT 32 = 0.

Execute access requirements:

● SDE(R1) $\leq$ P(RN) $\leq$ SDE(R2).

● When segment supplies unprivileged instructions only, SDE(XP) = 01.

● When segment supplies unprivileged and local privileged instructions, SDE(XP) = 10.

● When segment supplies unprivileged, local privileged and global privileged instructions, SDE(XP) = 11.
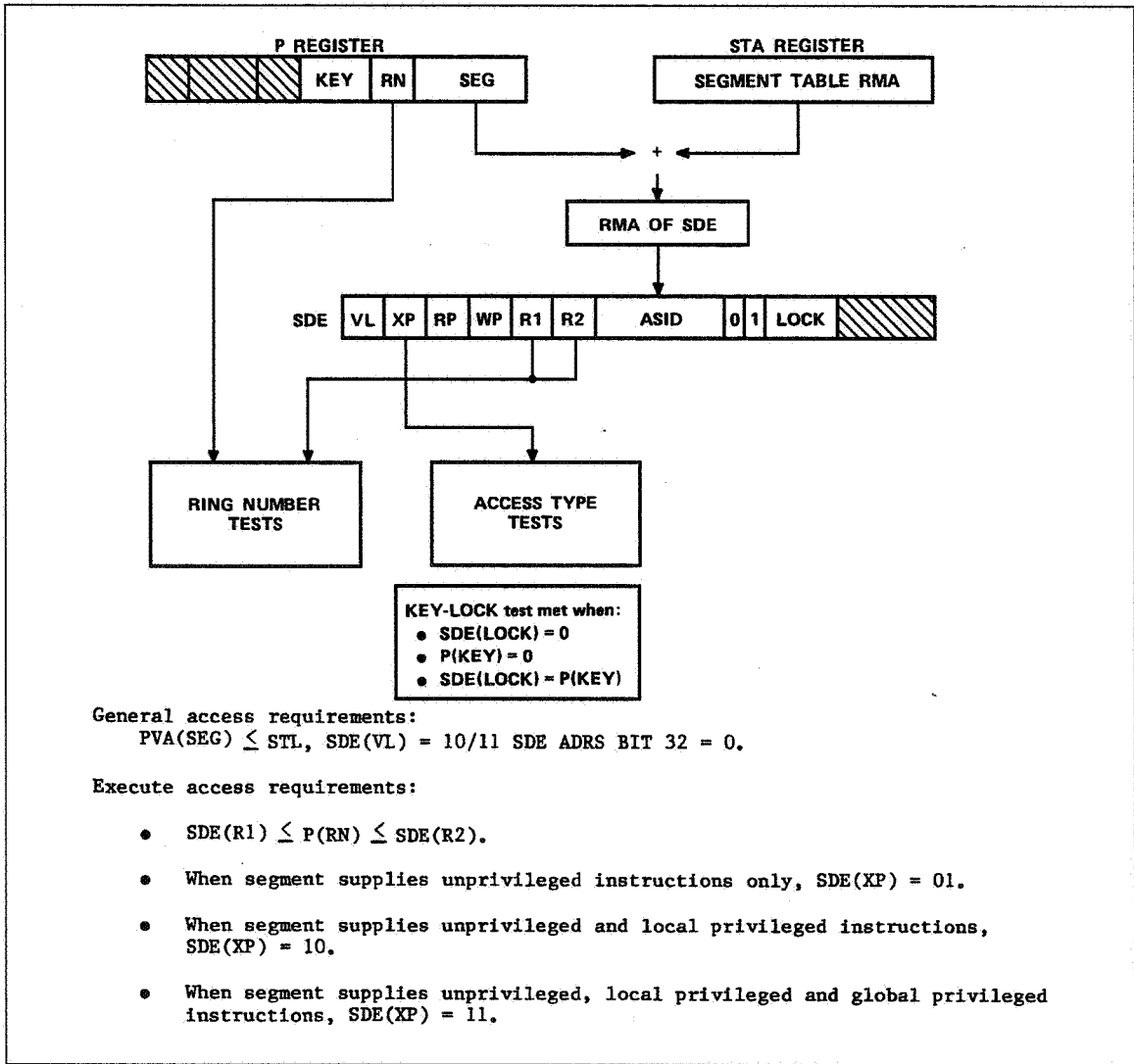
Figure 2-26. PVA-to-SVA Conversion, Execute

Ring Structure

The ring hierarchy controls read, write, and execute accesses to a segment, as follows.

- All PVAs have a 4-bit RN field specifying an access privilege from 1 to 15.

- R1 and R2 fields in SDEs, and the R3 field in CBPs, specify (PVA)RN requirements for access.

Ring Voting

During certain conditions, the RN loaded into an A register is the largest of the following.

- The RN initially in the A register.

- The RN accessed in memory.

- The R1 of the SDE used to access the RN in memory.

Such conditions and the resulting RN loaded are as follows.

- Load AK (A0, 84) instructions load a new Ak(RN) which is the largest of the following.

  - Initial Aj(RN)

  - SDE(R1) addressed by initial Aj(SEG)

  - Aj(RN) from memory

- Load multiple (80) instruction loads new A(RN)s which are the largest of the following.

  - Initial Aj(RN)

  - SDE(RN) addressed by initial Aj(SEG)

  - The relevant A(RN) from memory

- Call indirect (B5) instruction, when the caller's CBP(EPF) = 1, loads a new A3(RN) which is the largest of the following.

  - New CBP(RN) from (Aj+8*Q+8)

  - New P(RN)

- Return (04) instruction loads new A(RN)s which are the largest of the following.

    - Initial A2(RN)

    - SDE(RN) addressed by initial A2(SEG)

    - A(RN) from SFSA in memory

    When the return instruction loads new P(RN), any A(RN) not specified for loading by the SFSA descriptor are set to the largest of the following.

    - Initial A(RN) of the relevant A register

    - New P(RN)

Effect of RN = 0

RN = 0 can serve as a flag to the operating system to link segments on a demand basis. RN = 0 causes an interrupt condition (setting MCR bit 60), when detected as follows.

- Call indirect (B5) instruction and CBP(RN) is 0.

- Return (04) instruction and an A(RN) in memory (specified for loading by SFSA descriptor) is zero.

- Pop (06) instruction and A1(RN) or A2(RN) in SFSA is zero.

- Load Ak (80, 84) instructions and new Ak(RN) in memory are zero.

- Load multiple (80) instruction and any A(RN) accessed in memory is zero.

- Trap interrupt and CBP(RN) is zero.

No test occurs for RN zero when the A registers are loaded by an exchange operation, or when an A register serves as an address source to access memory.

RN for Read/Write Access

PVAs used for read/write are in A registers and must have RN as follows.

- For a write access, the A(RN) used may not exceed the accessed segment's SDE(R1):

    $$A(RN) \leq SDE(R1) \text{ (write limit test)}.$$

- For a read access, the A(RN) used may not exceed the accessed segment's SDE(R2):

    $$A(RN) \leq SDE(R2) \text{ (read limit test)}.$$

RN for Execute Access

The RN for execute access are verified only for operations having the capability to switch segments. The PVA used may originate from an A register, from a CBP, or from the P register. PVA(RN) requirements are as follows.

- For accessing a branch intersegment target instruction, initial P(RN) and Aj(RN) must fall within the execute bracket given by target segment's SDE(R1) and SDE(R2).

    $SDE(R1) \leq P(RN) \leq SDE(R2)$ (prevents branches to outside the execute ring bracket of the target segment).

    $SDE(R1) \leq Aj(RN) \leq SDE(R2)$ (prevents branches to outside the execute ring bracket of the target segment).

- For access to a call indirect target instruction, initial P(RN) must not exceed R3 of CBP used, and also not less than R1 of SDE for target instruction. Refer to figure 2-27. Also, the initial Aj(RN) must not exceed R3 of CBP used.

    $P(RN) \leq CBP(R3)$ (a limit set by the operating system).

    $P(RN) \geq SDE(R1)$ (prevents outward calls).

    $Aj(RN) \leq CBP(R3)$ (a limit set by the operating system).

- For accessing a trap interrupt target instruction, initial P(RN) must not exceed R3 of CBP used, and also not less than R1 of SDE for target instruction. Also, the trap pointer (TP) RN must not exceed R3 of CBP used.

    $P(RN) \leq CBP(R3)$ (a limit set by operating system).

    $P(RN) \geq SDE(R1)$ (prevents outward trap interrupts).

    $TP(RN) \leq CBP(R3)$ (a limit set by operating system).

- For access to a return target instruction, P(RN) loaded from SFSA may not be less than initial PSA pointer A2(RN).

    Initial $A2(RN) \leq$ final $P(RN)$ [prevents user from setting P(RN) in SFSA for an inward return as user cannot diminish A2(RN)].

RN Effect on Pop Instruction

A pop instruction loads A1, A2, CFF, and OCF from SFSA, and updates TOS pointer. It does not alter P or A0, and must not alter the RN in which the stack resides. The entire initial A2 must equal A0 in SFSA, and the initial A2(RN) must equal P(RN).

Initial $A2 = A0$ in SFSA

Initial $A2(RN) = P(RN)$ [which equals A0(RN) by the previous test].

Effect of RN Violations

Ring number violations and the detection of RN = 0 have effect as follows.

Operation	Violation	Effect
Write	Aj(RN) > SDE(R1)	Access violation (MCR 54)
Read	Aj(RN) > SDE(R2)	Access violation (MCR 54)
Branch Intersegment	SDE(R1) > P(RN) > SDE(R2)	Access violation (MCR 54)
Call Indirect	P(RN) > CBP(R3)	Access violation (MCR 54)
	Aj(RN) > CBP(R3)	Access violation (MCR 54)
	P(RN) < SDE(R1)	Outward call (MCR 61)
	CBP(RN) = 0	RN zero (MCR 60)
Return	A2(RN) > P(RN) in SFSA	Inward return (MCR 61)
	Any A(RN) from SFSA = 0	RN zero (MCR 60)
Pop	Initial A2 ≠ A0 in SFSA	Environment specification error (MCR 55)
	Initial A2(RN) ≠ P(RN)	Interring pop (UCR 52)
	A1(RN) from SFSA = 0	RN zero (MCR 60)
	A2(RN) from SFSA = 0	RN zero (MCR 60)
Load A	Accessed A(RN) = 0	RN zero (MCR 60)
Trap interrupt	P(RN) > CBP(R3)	Access violation (MCR 54)
	TP(RN) > CBP(R3)	Access violation (MCR 54)
	P(RN) < SDE(R1)	Outward call (MCR 61)
	CBP(RN) = 0	RN zero (MCR 60)

INITIAL P REGISTER

	KEY	RN	SEG	BN

STA		A0 (DYNAMIC SPACE POINTER)		
RMA		RN	SEG	BN

$+ \longrightarrow$ RMA of SDE

SDE FOR SFSA

VL	XP	RP	WP	R1	R2	ASID	01	LOCK	

STA		Aj+8*Q			
RMA		RN	SEG	(PN) BN (PO)	

$+ \longrightarrow$ RMA of SDE

SDE FOR CBP

VL	XP	RP	WP	R1	R2	ASID	01	LOCK	

PTE FOR CBP

VL	UM	SPID	PAGE FRAME RMA

RMA of CBP

CODE BASE POINTER (CBP)

	VMID	EP		R3	R4	SEG	BN

STA			
RMA	$\longrightarrow + \longrightarrow$	RMA of SDE	

SDE FOR TARGET INSTRUCTION

VL	XP	RP	WP	R1	R2	ASID	01	LOCK	

	KEY	RN	SEG	BN

FINAL P

FINAL P REGISTER
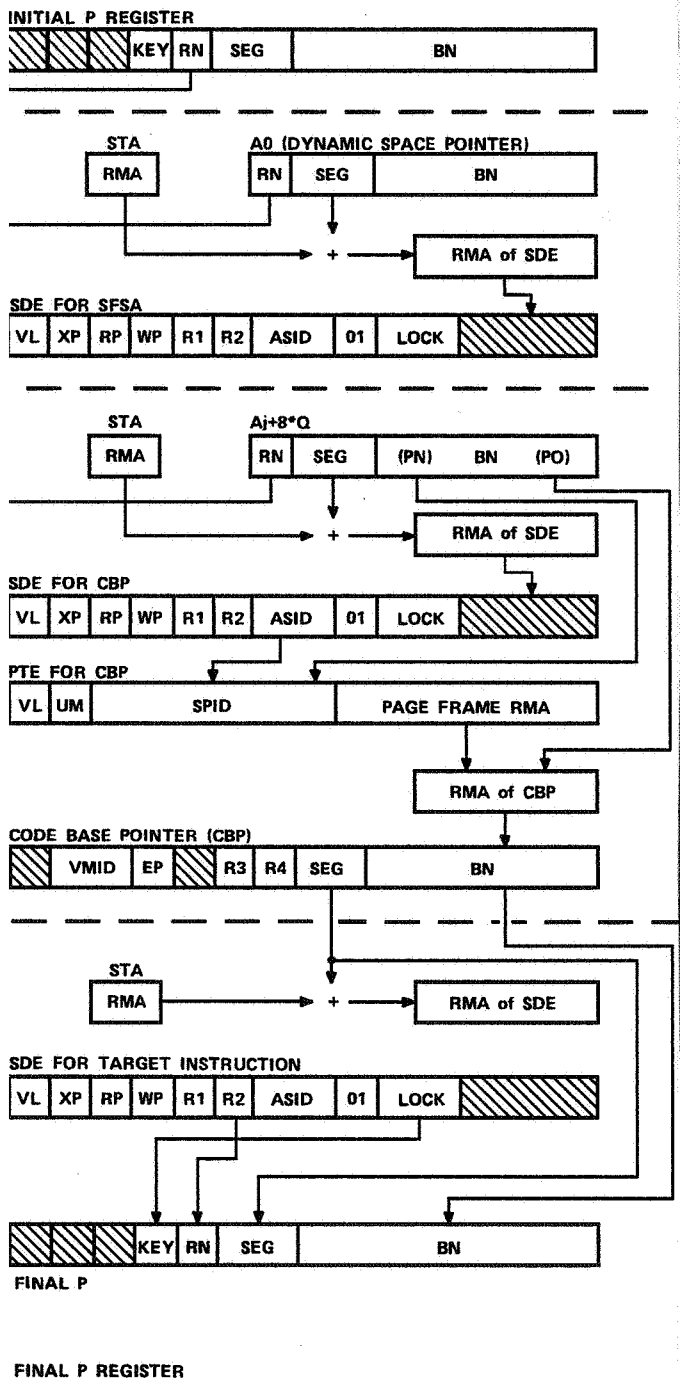
:K)
allee's
:DE(R2).

Call Indirect Access Requirements

Execute Access Privilege/Mode

Execution of some system instructions may c
process, or require global/local privilege. An
fetched from a segment for which the SDE(X
privileged when it is fetched from a segment
instruction which has global privilege also h
requirements are dependent on an instructior
listed in table 2-22.

Table 2-21. System Instruction Privilege

Instruction

Interrupt processor (03)

Copy-To-State-Register:

60-7F
80-BF
C0-DF

Return (04) with SFSA(VMID) = 0

Load page table index (17)

Branch, alter condition register (9F):
k = 0/1/8/9

Purge buffer k (05):
k = 0/1/2/8-F

Notes:

UCR 48: Privileged instruction fault.
MCR 55: Environment specification error.
MCR 51: Instruction specification error.

independent of other access protection

lock.

g a key.

:chanism is not hierarchical. For access
:t match with the key tested, except as

ible by any key.

locks.

:onditions.

ant's SDE(RP) = 01.

ient's SDE(WP) = 01.

d during call indirect access, can be locked

:ed.

ted only when the P register providing the
; has a KEY as follows.

segment's SDE(LOCK).

-lock.

the final P(KEY) equal to the accessed
ess requirements are shown in figure 2-27.

Interstate Programming

The CYBER 170 State environment is a substate of
which the CP uses CYBER 170 State compatible inst
formats.

This subsection contains some general aspects of the
description of interaction between the CYBER 170 St
170 State is described in detail in the CYBER 170 S
listed under Related Manuals.

The general characteristics of the CYBER 170 State

● The CYBER 170 State user sees the P register a:
  Hardware, however, treats the P register the sam
  the PVA-to-SVA-to-RMA conversion with access p

● The CYBER 170 State user sees CM addresses as
  Hardware, however, treats these as PVAs and fill
  fields from P register data.

● The entire CYBER 170 State memory exists in o
  ASID of FFFF$_{16}$.

● Virtual pages into which PPs write 60-bit words :

● Virtual State procedures execute the CYBER 170
  467) instructions by way of trap interrupts.

● Virtual State procedures handle some CYBER 17(
  as system or job timer interrupts or hardware er
  halt a CYBER 170 CP.

Operation in CYBER 170 State

Virtual State programs establish the CYBER 170 State environment and provide recovery facilities for all hardware and some software errors occurring in CYBER 170 State. The general operation is as follows.

● Virtual State programs build the page table, segment table, and exchange packages for the CYBER 170 State environment.

● A Virtual State program switches the CP to CYBER 170 State through one of the following.

  – An exchange from Virtual State monitor mode to Virtual State job mode, CYBER 170 State monitor/job mode.

  – A call/return within Virtual State job mode, from Virtual State to CYBER 170 State monitor/job mode.

● The CP runs in CYBER 170 State, with exchanges between CYBER 170 State monitor and job modes, until one of the following occurs.

  – An exchange interrupt to Virtual State monitor mode.

  – A trap interrupt within Virtual State job mode, from CYBER 170 State to Virtual State.

● Virtual State programs determine the cause of the exchange or trap interrupt and take appropriate action.

State-Switching Operations

Figure 2-28 depicts the state-switching operations. The CP switches states when its VMID changes states through the following.

● Exchanges between Virtual State monitor mode and CYBER 170 State. The new VMID loads from the exchange package.

● Calls within Virtual State job mode, from Virtual State to CYBER 170 State. The new VMID loads from the CBP.

● Returns within Virtual State job mode, from Virtual State to CYBER 170 State. The new VMID loads from the SFSA.

● Trap interrupts within Virtual State job mode, from CYBER 170 State to Virtual State. The new VMID loads from the CBP addressed by the trap pointer.
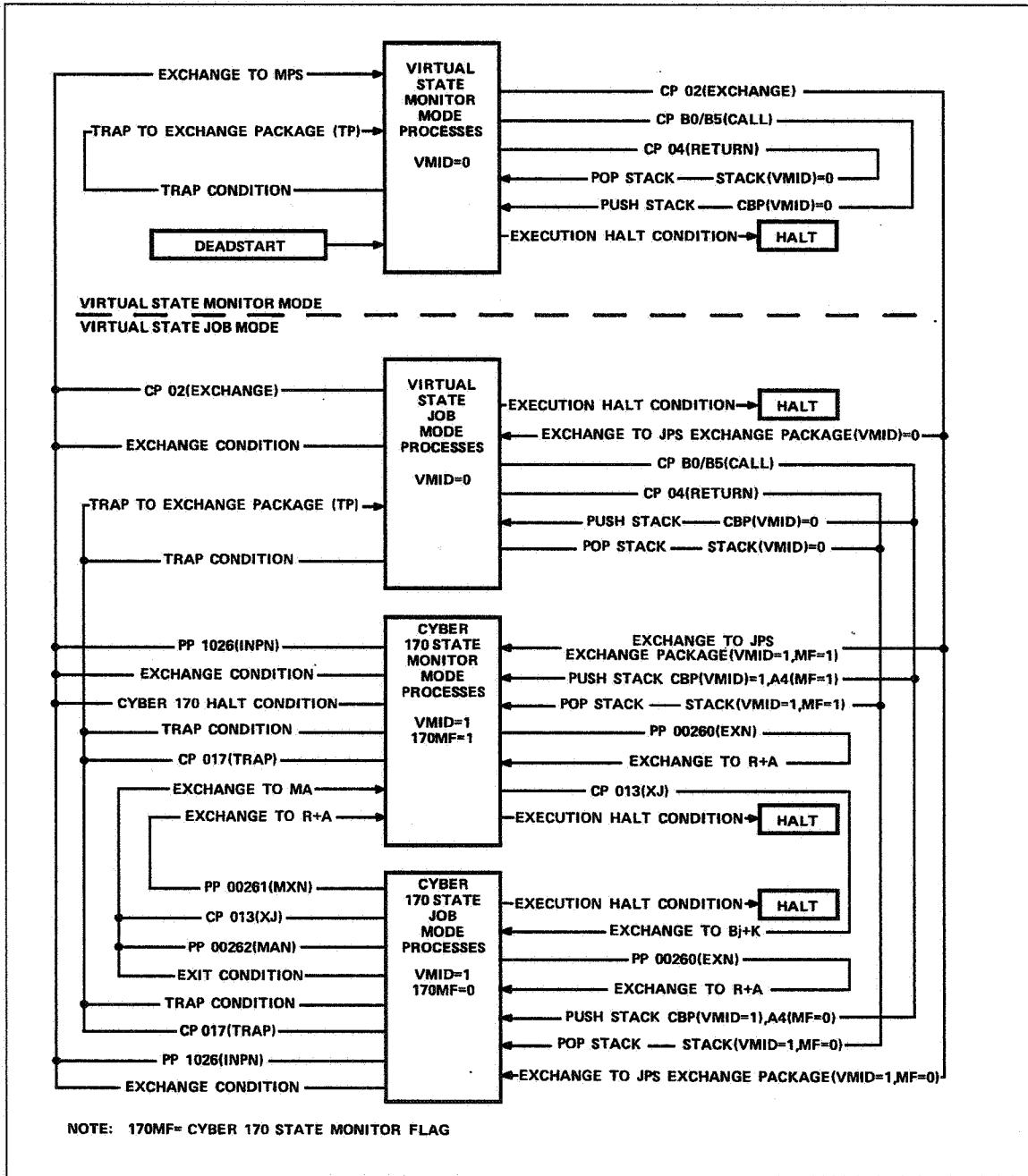
Figure 2-28. Interstate Calls, Returns, and Interrupts

Memory Addressing in CYBER 170 State

The CM space allocated to CYBER 170 State is addressed as a single segment. An ASID of $FFFF_{16}$ is globally reserved for this segment. The hardware uses this ASID value for cache invalidation, described under Cache Invalidation in CYBER 170 State in this chapter.

Hardware treats all CM addresses supplied by CYBER 170 State programs as PVAs and converts these first to SVAs, and then to RMAs. The P register provides the key, RN, and segment number for all CM accesses. The PVA-to-RMA conversion includes instruction fetch, load/store, extended memory transfers, and CYBER 170 State exchanges to MA, Bj+K, or R+A.

The operating system supplies the P register KEY, RN, and SEG fields to make the PVA-to-RMA conversion possible; these cannot be changed by CYBER 170 State programs. The entire P register, including the KEY, RN, and SEG fields, loads as follows.

● During Virtual State monitor mode-to-CYBER 170 State monitor/job mode exchanges, from the exchange package at JPS. The exchange package used is specially formatted for such an exchange.

● During calls from Virtual State job mode to CYBER 170 State job/monitor mode, from the CBP.

● During returns from Virtual State job mode to CYBER 170 State job/monitor mode, from the SFSA.

Cache Invalidation in CYBER 170 State

When a PP writes a 60-bit word into CM, cache memory is updated by cancelling any copy of that word in cache memory. The RMA used to write CM is also supplied to cache memory. Hardware treats this RMA as a SVA with an ASID of FFFF(hex).

To fulfill the requirements of cache invalidation addressing, the operating system assigns an ASID of $FFFF_{16}$ for the virtual memory segment used as CYBER 170 State memory, and must also map the virtual pages into which PPs write 60-bit words, 1:1, into real memory. In such a case, the RMA supplied is numerically equal to the SVA information necessary to locate the addressed word in cache memory. The operating system maps CYBER 170 State address 0 into BN 0 of system virtual segment FFFF(hex).

Cache is not purged when PPs write 64-bit words into CM. Therefore, it is possible to have other than 1:1 mapping of CM for CYBER 170 State PP-write-to-CM instructions in conjunction with software cache invalidation.

Virtual State Monitor Mode-to-CYBER 170 State Exchange

The interstate exchange (02) instruction executes the same as a Virtual State exchange instruction, except the incoming exchange package is formatted as shown in figure 2-29.
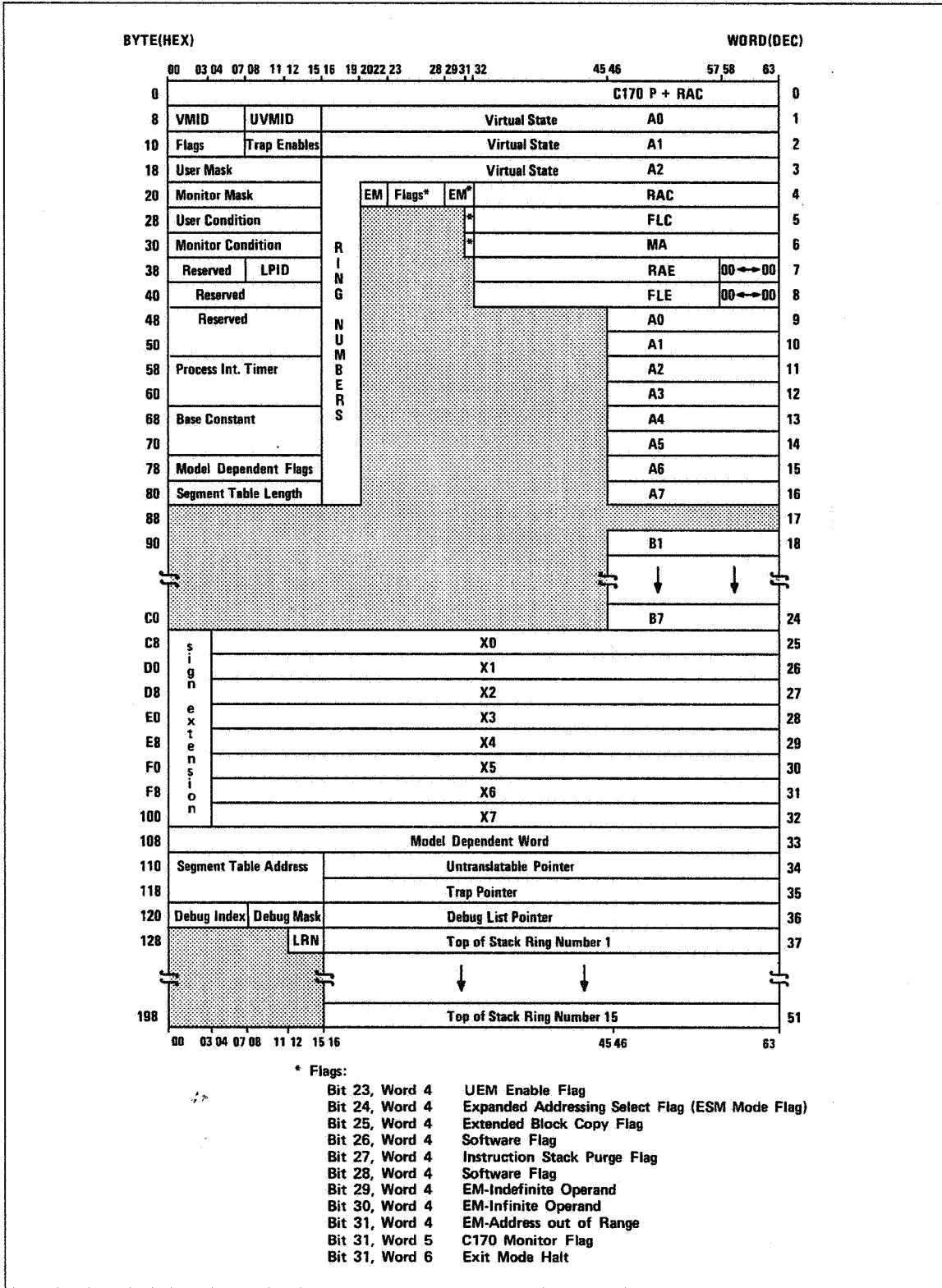
Figure 2-29. Interstate Exchange Package

Interstate Stack Frame Save Area

Interstate calls/returns and trap interrupts use a SFSA with the format shown in figure 2-30. For the description of modified fields, refer to the corresponding fields in the Interstate Exchange Package in this chapter, except as follows.

- The user and MCR are stored only during trap interrupts.

- The user and MCR are not loaded by a return instruction to CYBER 170 State.

- X register field bits 0 through 3 are undefined in the SFSA after a call, return, or trap interrupt. Hardware does not sign-extend these fields.
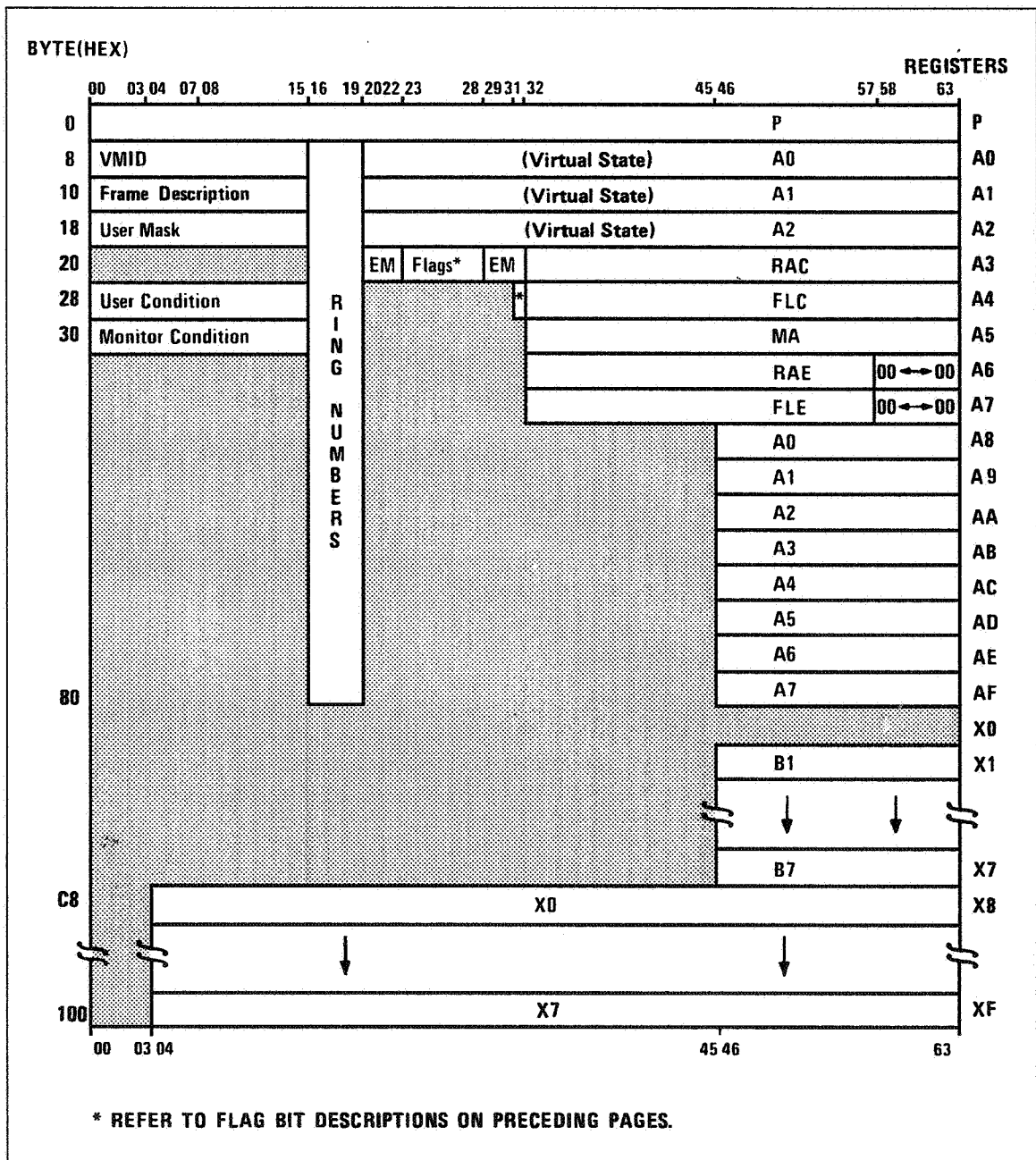
Figure 2-30. Interstate Stack Frame Save Area

Exchange Packages Used in CYBER 170 State

The CP uses two types of exchange packages with CYBER 170 State operations.

● Exchange package for interstate exchanges

● Exchange package for exchanges within CYBER 170 State

Before Virtual State monitor mode initiates an exchange to CYBER 170 State monitor/job mode, the following exchange packages must be ready in CM.

● An interstate exchange package for exchanging from Virtual State monitor mode to CYBER 170 State monitor/job mode

● An interstate exchange package for exchanging back to Virtual State monitor mode

● A CYBER 170 State exchange package for a possible immediate exchange to CYBER 170 State monitor mode (when exchanging from Virtual State monitor mode to CYBER 170 State job mode)

CYBER 170 State-to-Virtual State Monitor Mode Exchange

An exchange from a CYBER 170 State process to Virtual State monitor mode is through an interstate exchange interrupt, which executes the same as a Virtual State exchange interrupt, except the outgoing interstate exchange package is formatted as shown in figure 2-31. Such an interrupt initiates as follows.

● By conditions which set a bit in the monitor or UCR (when enabled)

● By a central exchange jump (013) instruction when the CYBER 170 State monitor flag is set

This exchange does not update the CYBER 170 State exchange package. Virtual State monitor mode software may, however, perform the update. Refer to the specific operating systems manual for further information.

Exchanges Within CYBER 170 State

Within the CYBER 170 State, CYBER 170 State-compatible exchange jumps and interrupts occur between the CYBER 170 State monitor and job modes. Refer to CYBER 170 State Hardware Reference manual. Such exchanges use the exchange package format shown in figure 2-27. The interstate exchange package is not altered by this exchange.

CYBER 170 State Exchange Package

The CYBER 170 State exchange package (figure 2-31) resides within the CYBER 170 State process segment in memory. During exchange operations between CYBER 170 State monitor and job modes, the current CP CYBER 170 State registers store into an outgoing CYBER 170 State exchange package and reload from the incoming CYBER 170 State exchange package. These two exchange packages store into the same CM locations and, thus, the outgoing exchange package replaces the incoming exchange package in CM. Refer to the CYBER 170 State Hardware Reference manual for further information.
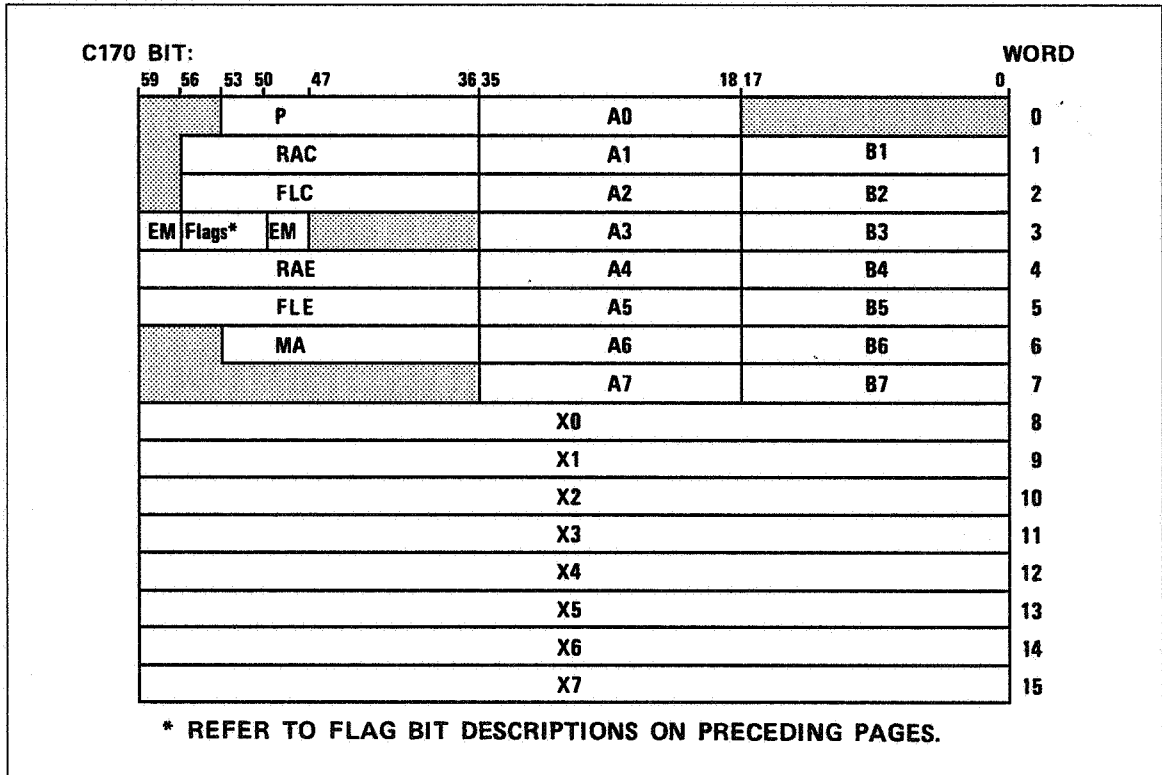
Figure 2-31. CYBER 170 State Exchange Package

Call from Virtual State to CYBER 170 State

The interstate call indirect (B5) instruction executes identical to a Virtual State call indirect instruction, except the SFSA has the format shown in figure 2-30. The operating system arranges the Virtual State A and X registers so the required parameters pass to the CYBER 170 State procedure (the call instruction leaves these registers unchanged in hardware). Figure 2-31 also shows the register formats in hardware. The call instruction does not store the user and MCR in SFSA.

The BN of P supplied by the CBP is the P + RAC of the called procedure target address; BN bits 32 through 39 and 61 through 63 must be zeros.

Return from Virtual State to CYBER 170 State

The interstate return (04) instruction executes the same as the Virtual State return instruction, except the SFSA from which the CYBER 170 State registers are restored is formatted as shown in figure 2-30. The return instruction with VMID = 1 in SFSA is a global-privileged instruction.

The BN of P loaded from SFSA must be P+RAC into the CYBER 170 State process, where BN bits 32 through 39 and 63 are zeros, and bits 61 and 62 denote the parcel where execution starts.

Trap Interrupt from CYBER 170 State to Virtual State

The interstate trap interrupt causes the same action as the Virtual State trap interrupt described in this chapter, except the SFSA has the format shown in figure 2-30. This trap interrupt occurs when traps are enabled and a bit sets in the UCR.

Bits 0 through 3 of the X register fields in SFSA are undefined. The VMID and the P register RN load from the CBP addressed by the trap pointer.

The BN of P stored in SFSA is the P + RAC of the interrupted procedure; BN bits 32 through 39 and 63 are zeros, and bits 61 and 62 denote the parcel address within the instruction word.

Interstate Exchange Package

The JPS pointer locates the first word in this exchange package. The exchange package format is shown in figure 2-26. The format is a modified format of the exchange package used within Virtual State. The CYBER 170 State A, B, X, and other registers reside in locations occupied by the Virtual State A and X registers in the Virtual State exchange package. Therefore, after a trap interrupt, the Virtual State program entered has access to these CYBER 170 State registers by accessing the registers as if they were Virtual State registers.

The following paragraphs describe the modified fields of the exchange package. For other fields, refer to Exchange Package in this chapter.

Program Address (P) Register

The P field (word 1, bits 0 through 63) contains the Virtual State P register, which supplies the PVA for the CYBER 170 State where instruction execution is to begin or resume. The KEY field contains the process key for access protection. The BN field of the PVA interprets as the CYBER 170 State P register plus RAC, appended with the parcel address. The P register part of the parcel address must not exceed 18 bits (because exchanges within CYBER 170 State truncate the P register to 18 bits). The user must also ensure that the CYBER 170 State P register does not count past 18 bits, which is possible. The format is as follows.

Bits	Description
00-01	Not used
02-07	Not used
08-09	Not used
10-15	Key (KEY)
16-19	Ring number (RN)
20-31	Process segment (SEG)
32-43	Must be zero
43-60	P + RAC (word address)
61-62	Parcel address
63	Must be zero

CP operation is out of range or undefined when CYBER 170 State P + RAC > 7777777_8, or when P > 777777_8.

Stack Pointers

The 48-bit A0, A1, and A2 fields (words 1 through 3, bits 16 through 63) are Virtual State A registers. These registers function as stack pointers after a trap interrupt from CYBER 170 State to Virtual State.

EM Register

The EM register contains the exit mode selection bits for use in CYBER 170 State.

Interstate Exchange Package	Description	CYBER 170 State Exchange Package
Word 4 Bit 20	Hardware error (not used).	Word 3 Bit 59
Word 4 Bit 21	Hardware error (not used).	Word 3 Bit 58
Word 4 Bit 22	Hardware error (not used).	Word 3 Bit 57
Word 4 Bit 29	Indefinite operand.	Word 3 Bit 50
Word 4 Bit 30	Infinite operand.	Word 3 Bit 49
Word 4 Bit 31	Address out of range.	Word 3 Bit 48

NOTE

Hardware errors in CYBER 170 State cause an exchange to Virtual State monitor mode. Bits 20 through 22 function as software flags preserved during exchanges.

Flags

The flag formats are as follows.

Virtual State Exchange Package	Description	CYBER 170 State Exchange Package
Word 4 Bit 23	UEM enable flag	Word 4 Bit 56
Word 4 Bit 24	Expanded addressing select flag	Word 4 Bit 55
Word 4 Bit 25	Enhanced block copy flag	Word 4 Bit 54
Word 4 Bit 26	Software flag	Word 4 Bit 53
Word 4 Bit 27	Instruction stack purge flag	Word 4 Bit 52
Word 4 Bit 28	Software flag	Word 4 Bit 51
Word 5 Bit 31	CYBER 170 State monitor flag	–
Word 6 Bit 31	Exit mode halt flag	–

Unified Extended Memory (UEM) Enable Flag –

If set, this flag enables the CM block copy and single-word copy (011, 012, 014, and 015) instructions to access CM.

Expanded Addressing Select Flag –

If set, selects expanded addressing mode, which provides addressing up to 24 bits in a 30-bit format for data transfer between CM and UEM. If clear, selects standard addressing mode, which provides addressing up to 21 bits in a 24-bit format for data transfer between CM and UEM.

Enhanced Block Copy Flag –

If set, CYBER 170 State block copy (011, 012) instructions use X0 bits 30 through 50 rather than A0 to determine the CM address.

Software Flag (Word 4, Bit 26) –

This is a reserved flag described in software documentation.

Instruction Stack Purge Flag –

If set, this flag causes instruction stack purges as described under Code Modification in CYBER 170 State in this chapter.

Software Flag (Word 4, Bit 28) –

This is a reserved flag described in software documentation.

CYBER 170 State Monitor Flag –

If set in an incoming exchange package, this flag indicates that the CYBER 170 State process is to start (or resume) execution in CYBER 170 State monitor mode. If set in an outgoing exchange package, this flag indicates that the interrupted process was executing in CYBER 170 State monitor mode.

Exit Mode Halt Flag –

If set, this flag indicates a software error that would halt a CYBER 170 CP. This bit does not set for hardware errors that would have halted a CYBER 170 CP.

RAC Register

The 32-bit RAC field (word 4, bits 31 through 63) contains the 21-bit CYBER 170 State reference address for CM addressing. CP operation is undefined when RAC $> 7777777_8$ or RAC + FLC $> 7777777_8$.

FLC Register

The 32-bit FLC field (word 5, bits 31 through 63) contains the 21-bit CYBER 170 State field length for CM addressing. CP operation is undefined when FLC $> 7777777_8$ or RAC + FLC $> 7777777_8$.

Monitor Address (MA) Register

The 32-bit MA field (word 6, bits 31 through 63) contains the 18-bit CYBER 170 State MA register pointing to the CYBER 170 State exchange package starting address used when executing the following instructions with the CP in CYBER 170 State job mode.

● CYBER 170 State central exchange jump (013)

● PP monitor exchange jump to MA (262x)

CP operation is undefined when MA $> 777777_8$.

Address (A) Registers

The eight 18-bit A fields (words 9 through 16, bits 46 through 63) are CYBER 170 State CM address (A) registers.

RAE Register

The 32-bit RAE field (word 7, bits 31 through 63) contains the 21-bit CYBER 170 State RAE register supplying the reference address for UEM addressing (instructions 011, 012, 014, and 015). CP operation is undefined when RAE $> 7777777_8$ or when the RAE rightmost six bits are nonzero.

FLE Register

The 32-bit FLE field (word 8, bits 31 through 63) contains the 24-bit CYBER 170 State FLE register supplying the field size for UEM addressing (instructions 011, 012, 014, and 015). CP operation is undefined when FLE > 77777777_8 or when the FLE rightmost six bits are nonzero.

Virtual State Ring Numbers

The operating system supplies the A register Virtual State RNs (words 4 through 16, bits 16 through 19) for use after an interrupt to Virtual State. These must not be altered in CYBER 170 State.

Index (B) Registers

The 18-bit CYBER 170 State B registers 1 through 7 (words 18 through 24, bits 46 through 63) are used primarily as indexing registers. Register B0 is not included in the exchange package; this register always contains all zeros.

CP Operand (X) Registers

Words 25 through 32, bits 4 through 63 contain the CYBER 170 State operand (X) registers. Hardware sign-extends the X registers in an outgoing exchange package to 64 bits. The operating system sign-extends the X registers in an incoming exchange package to 64 bits.

Code Modification in CYBER 170 State

The CYBER 170 State return jump (010), extended memory read or write (011, 012), exchange jump (013), or long jump (02) instruction purges the instruction buffer.

Additionally, when the operating system sets the CYBER 170 State instruction stack purge flag in the interstate exchange package or SFSA, the conditional jump (03 through 07) instructions and store (50 through 57 with $i = 6$ or 7) instructions also purge the instruction stack. In such case, the modified code always executes, even with the previous code in the same instruction word as the modifying code.

With the instruction stack purge flag clear, execution of unmodified code in the instruction buffer may occur but can never be guaranteed, since an exchange interrupt may clear all instructions in the buffer at any time, including instructions in the same instruction word as the code-modifying instruction.

Debug/Performance Monitoring

The CYBER 170 State environment does not support the Virtual State debug feature. Other information related to performance monitoring may be collected through the MCH.

Exception Handling in CYBER 170 State

During execution of CYBER 170 State programs, certain hardware or software errors cause an exchange to Virtual State monitor mode, or a trap interrupt to Virtual State, as described in the following paragraphs. Such exceptions do not set exit condition bits or store the P register at location RAC. Instead, RAC+P stores in the outgoing exchange package or in SFSA.

Software Exception Conditions

Table 2-22 lists CYBER 170 State exception conditions. In general, software errors occurring in CYBER 170 State job mode with the corresponding exit mode selected cause an exchange to CYBER 170 State monitor mode. Corresponding software errors occurring in CYBER 170 State monitor mode which result in an exchange from CYBER 170 State to Virtual State monitor mode with no MCR bits set and the CYBER 170 State halt flag in the outgoing exchange package set are as follows.

- Illegal instruction in CYBER 170 State monitor mode

- Read or write address out of range (with exit mode selected for this error) in CYBER 170 State monitor mode

- Instruction fetch address or branch target address out of range in CYBER 170 State monitor mode

- Infinite or indefinite value detected (with exit mode selected for this error) in CYBER 170 State monitor mode

- 00 instruction in CYBER 170 State monitor mode

Exceptions that may only occur immediately after entering or leaving the CYBER 170 State environment, and which cause an exchange from CYBER 170 State to Virtual State monitor mode, with an MCR bit set and the CYBER 170 State halt flag in the outgoing exchange package clear, are as follows.

Bit	Description
MCR 55	Environment specification error
MCR 60	Invalid SEG/RN zero
MCR 52	Address specification error
MCR 54	Access violation
MCR 61	Outward call/inward return
MCR 63	Trap exception

The following exception causes an exchange from CYBER 170 State to Virtual State monitor mode with an MCR bit set and the CYBER 170 State halt flag in the outgoing exchange package clear.

Bit	Description
MCR 57	Page table search without find

Exceptions that may occur only immediately after entering or leaving the CYBER 170 State environment, and which cause a trap interrupt from CYBER 170 State to Virtual State, with a UCR bit set, are as follows.

Bit	Description
UCR 50	Free flag set in incoming exchange package
UCR 53	Return with CFF set

Exceptions causing a trap interrupt from CYBER 170 State to Virtual State, with a UCR bit set, are as follows.

Bit	Description
UCR 48	Trap to Virtual State (017) instruction in the CYBER 170 State instruction set
UCR 49	Unimplemented instructions

A trap interrupt attempted with trap interrupts disabled causes an exchange or stack operation.

Address Errors

An address error does not change the destination CM location or register. Read/write address errors occurring in CYBER 170 State monitor mode initiate an exchange to Virtual State monitor mode only with the corresponding exit mode selected. Instruction fetch address errors and target instruction fetch address errors in CYBER 170 State monitor mode always initiate the exchange.

A page-table-search-without-find fault (page fault) in CYBER 170 State (MCR bit 57) causes an exchange to Virtual State monitor mode only in the absence of a simultaneous FLC or FLE violation. When the exchange does occur, the interrupted process is restartable only when the page fault occurred during execution of a CYBER 170 State UEM transfer (011, 012, 014, or 015) instruction.

When the page fault causes the exchange, the CP places the PVA causing the page fault into the untranslatable pointer register, including the SEG and RN fields of the PVA.

The CP tests each CM reference for an address specification error, invalid segment, or access violation. When an interrupt occurs because of these conditions, the interrupted CYBER 170 State process may not be restartable.

Illegal Instructions

The following CYBER 170 State instructions, when executed in CYBER 170 State monitor mode, cause an exchange to Virtual State monitor mode with the CYBER 170 State halt flag set in the exchange package.

- Any 30-bit instruction at parcel 3

- Instructions 011 through 013 at parcel 1, 2, or 3

- Instruction 016

- Instructions 011, 012, 014, and 015 with certain parameters as described in the following text

Extended Memory Transfer Exceptions

Instructions 011 and 012 in CYBER 170 State monitor mode, with exit mode selected, cause exceptions with the following priority.

Exception	Response
1. Not in parcel 0	Illegal instruction
2. UEM enable flag clear (exchange package)	Illegal instruction
3. FLC violation	Address range error
4. Negative block length	Address range error
5. FLE violation	Address range error
6. Zero block length	Fetch next instruction

Instructions 014 and 015 in CYBER 170 State monitor mode, with exit mode selected, cause exceptions with the following priority.

Exception	Response
1. UEM enable flag clear (exchange package)	Illegal instruction
2. FLE violation	Address range error
3. Xk (CYBER 170 State) bit 21/22/23 set	Address range error

Hardware Exceptions in CYBER 170 State

In general, hardware exceptions cause an exchange to Virtual State monitor mode or a trap interrupt to Virtual State job mode. Table 2-22 lists CYBER 170 State exception conditions.

Hardware exceptions causing an exchange from CYBER 170 State to Virtual State monitor mode are as follows.

Bit	Description
MCR 48	Detected uncorrectable error
MCR 50	Short warning
MCR 56	External interrupt
MCR 59	System interval timer
MCR 62	Soft error

Hardware exceptions causing a trap interrupt to Virtual State job mode are as follows.

Bit	Description
UCR 53	Process interval timer

Table 2-22. CYBER 170 State Exceptions

MCR/ UCR Bit	JPS Exch Pkg Bit	Error	C170 Mode	C170 Exit Mode Rgtr	Response
MCR 48	–	Hard errors: Parity Double SECDED CM port bounds	Mon/Job	NA	When address error, inhibit write to that address
					Complete current instruction; exchange to (MPS)[1]; P = RAC + next fetch address (not necessarily related to error)
MCR 50	–	Short warning	Mon/Job	NA	Complete current instruction; exchange to (MPS); P = RAC + next fetch address (not necessarily related to condition)
MCR 53	–	CYBER 170 State exchange request (PP)	Mon/Job	NA	See MCR 50 response
MCR 56	–	External interrupt (CP)	Mon/Job	NA	See MCR 50 response
MCR 59	–	System interval timer	Mon/Job	NA	See MCR 50 response
MCR 62	–	Soft error	Mon/Job	NA	See MCR 50 response
MCR 55	–	No CYBER 170 State bit in VMCL on entering CYBER 170 State	Mon/Job	NA	Inhibit execution of target instruction; exchange to (MPS); P = RAC + target fetch address
MCR 60	–	RN = 0 on return to CYBER 170 State	Mon/Job	NA	See MCR 55 response
MCR 52	–	Address specification error when CYBER 170 State entered	Mon/Job	NA	See MCR 55 response
MCR 54	–	Access violation when CYBER 170 State entered	Mon/Job	NA	See MCR 55 response

1. MPS – Monitor process state pointer register.

(Continued)

Exception Handling in CYBER 170 State

Table 2-22. CYBER 170 State Exceptions *(Continued)*

MCR/ UCR Bit	JPS Exch Pkg Bit	Error	C170 Mode	C170 Exit Mode Rgtr	Response
MCR 60	–	Invalid segment when CYBER 170 State entered	Mon/Job	NA	See MCR 55 response
MCR 60	–	RN = 0 on call to CYBER 170 State or trap from CYBER 170 State	Mon/Job	NA	See MCR 55 response
MCR 61	–	Outward call/inward return to CYBER 170 State	Mon/Job	NA	See MCR 55 response
MCR 63	–	Trap exception on trap from CYBER 170 State	Mon/Job	NA	See MCR 55 response
MCR 57	–	Page fault without FLC or FLE violation	Mon/Job	NA	Interrupt execution; exchange to (MPS)[1]; P = RAC + this/next fetch address
UCR 48	–	Trap to executing instruction (017)	Mon/Job	NA	Trap to TP[2] in exchange package; P = RAC + next fetch address
UCR 49	–	Unimplemented instruction fetched	Mon/Job	NA	Inhibit execution; trap to TP in exchange package; P = RAC + this fetch address
UCR 50	–	Free flag in incoming exchange package	Mon/Job	NA	Inhibit next instruction; trap to TP in exchange package; P = RAC + next fetch address
UCR 51	–	Process interval timer	Mon/Job	NA	Complete current instruction: Trap to TP in exchange package; P = RAC + next fetch address

1. MPS = Monitor process state pointer register.

2. TP = Trap pointer register.

(Continued)

60000133 B

Programming Information   2-123

Table 2-22. CYBER 170 State Exceptions *(Continued)*

MCR/ UCR Bit	JPS Exch Pkg Bit	Error	C170 Mode	C170 Exit Mode Rgtr	Response
UCR 53	–	Critical frame on return to CYBER 170 State	Mon/Job	Any	Inhibit execution of target instruction; trap to TP[1] in exchange package; P = RAC + target fetch address
	29	FP indefinite	Mon	Bit 50	Exchange to (MPS)[2]; P = RAC + this/next fetch address
	30	FP infinite	Mon	Bit 49	Exchange to (MPS); P = RAC + this/next fetch address
–	31	FLC violation, incremental read or write	Mon	Bit 48	Interrupt execution; X or CM = unchanged; A = read address less RAC; exchange to (MPS); P = RAC + this/next fetch address
–	31	FLE violation, block transfer instruction 011, 012	Mon	Bit 48	Execute as pass; exchange to (MPS); P = RAC + next fetch address
–	31	FLE violation, single-word transfer instruction 014, 015	Mon	Bit 48	Execute as pass; exchange to (MPS); P = RAC + this/next fetch address

1. TP = Trap pointer register.

2. MPS = Monitor process state pointer register.

IOU Peripheral Processor Programming

Refer to volume 1 of this manual set for IOU functional characteristics and a description of nonconcurrent input/output (NIO) and concurrent input/output (CIO) PPs. The NIO and CIO PPs use the same instruction set and operate identically when operating in the CYBER 170 compatible mode.

The PPs may transfer one CM word or a block of CM words from a peripheral processor memory (PPM) to CM or from CM to a PPM. With NIO PPs, data from external devices is read into a PPM, and with additional instructions, transfers to CM. Conversely, data is transferred from CM to a PPM and then transfers by way of additional instructions to external devices. CIO PPs bypass PPM and transfer data directly to/from CM.

Central Memory Addressing by PPs

Addresses sent to CM from PPs are real-memory addresses. PPs address CM using either absolute or relocation addressing. Every PP can read all CM locations without restriction. Every PP has write access to CM as determined by the OS bounds register in the IOU. The port bounds register in CM may also be set to limit write access from IOU.

Absolute and Relocation Addressing

If A register bit 46 is a zero, bits 47 through 63 of A specify an absolute CM address 0 through 377777_8. If bit 46 of A is a one, bits 47 through 63 of A are added to the 28-bit relocation register R to specify an absolute CM address 0 through 1777777777_8. If bit 46 of A changes during a transfer, the addressing mode changes accordingly.

Instructions 0024/0025 load/store the relocation register. The leftmost seven bits of R represent (unused) extra addressing capacity. The rightmost six bits of R are appended zeros.

OS Bounds Test

The OS bounds test restricts write access from selected PPs to an upper or a lower region in CM. The PP instructions for which the OS bounds test performs are as follows.

Exchange jumps (00260, 00261, 00262)

Central write (0062, 1062)

Central write (d) words (0063, 1063)

Central read and set/clear lock (1000/1001)

If the OS Bounds is enabled, all addresses used by the exchange jumps and central writes will automatically be invalidated in the CP cache. The read and set/clear lock does not automatically invalidate cache.

PP Central Memory Read

Instructions which read CM data into PPM are as follows.

- 60-bit CM words to five 12-bit PP words:

  - Central read from A to d (0060)

  - Central read (d) words from (A) to m (0061)

- 64-bit CM words to four 16-bit PP words:

  - Central read from A to d long (1060)

  - Central read (d) words from (A) to m long (1061)

  - Central read and set lock from d to (A) (1000)

  - Central read and clear lock from d to (A) (1000)

It is possible, by way of block read (0061, 1061), to read up to 4095 CM words, over-writing PP memory cyclically. Hardware, however, uses PPM location 0 to hold the program counter during block transfers. Refer to instructions 0061 and 1061 in chapter 1.

PP Central Memory Write

Instructions which write PPM data from into CM are as follows.

- Five 12-bit PP words to 60-bit CM words:

  - Central write to (A) from d (0062)

  - Central write (d) words to (A) from m (0063)

- Four 16-bit PP words to 64-bit CM words:

  - Central write to (A) from d long (1062)

  - Central write (d) words to (A) from m long (1063)

It is possible, by way of block write (0063, 1063), to write up to 4095 CM words, repeating PP memory cyclically. Hardware, however, uses PPM location 0 to hold the program counter during block transfers. Refer to instructions 1062 and 1063 in chapter 1.

PP Memory Addressing by PPs

PP instructions use 6-bit/18-bit direct operands or obtain the operand from PP memory using direct, indirect, or indexed addressing.

Direct 6-Bit Operand

PP instructions of this type are no-address instructions. They have the format OPCODEd. The d-field provides a 6-bit direct operand zero-extended to 18 bits in calculations.

Direct 18-Bit Operand

PP instructions of this type are constant address instructions. They have the format OPCODEdm. The combined d and m field provides an 18-bit operand.

Direct 6-Bit Address

PP instructions of this type are direct address instructions. They have the format OPCODEd. The d field provides a 6-bit direct address, accessing PPM locations 0 to 77_8.

Direct 12-Bit Address

PP instructions of this type are indexed direct address instructions, with zero index. They have the format OPCODEdm, d = 0. The m field provides a 12-bit direct address, accessing PP memory locations 0 through 7777_8.

Indexed 12-Bit Address

PP instructions of this type are indexed direct address instructions. They have the format OPCODEdm, d = 0. The m field provides a 12-bit direct address (base address). The d field specifies a PP memory location from 0 to 77_8, the contents of which is a 12-bit index. The indexed direct address forms by adding the index to the base address as signed ones-complement numbers, ignoring overflow. When m + (d) = 7777 the result sets to 0000, except in the addition 7777 + 7777 = 7777.

Indirect 6-Bit Address

PP instructions of this type are indirect address instructions. They have the format OPCODEd. The 6-bit d field addresses PP locations 0 through 77_8. The 12 rightmost bits of the addressed location provide an address to access PP memory locations 0 through 7777_8.

Channel Input/Output Operations

All PPs may access all external devices through internal and external interfaces. Each internal interface contains a data register and channel control flags. The internal interfaces connect to external interfaces communicating with the external devices.

Channel Flags

Channel operation is controlled by the channel flags, which are set/reset by PP instructions and by signals from the external devices. The channel flags are as follows.

- Channel active/inactive flag

- Register full/empty flag

- Channel (marker) flag

- Error flag

The active flag and the full flag control the channel input/output transfers. The status of these two flags determines the channel active, inactive, full, or empty. The marker flag is for software use, and the error flag indicates transmission parity errors.

Channel Active Flag

A PP sets the active flag to indicate a reserved channel (channel active). The PP or the external device clears the active flag to indicate a free channel (channel inactive). Devices connected through the CYBER 170 State 12-bit channels may also set this flag to request attention.

A PP sets the active flag by the activate (0074) instruction or function (0076 and 0077) instructions. A PP clears the active flag with the deactivate (0075) instruction. Normally, external devices clear the active flag in response to a function instruction, or when they have no more data to send. A PP senses the active flag state using the jump on active/inactive (00640 and 00650) instructions.

Register-Full Flag

A register is full when it contains a function or data word for an external device, or when it contains a similar word received from the external device. The register is empty after the word is read. The flag turns on or off as the register changes states. A channel can only be full when it is active.

When set, the register-full flag signals the destination that data is available, and signals the sender that no more immediate data can be sent. When clear, full flag signals the destination to wait for the next data word, and signals the sender to send another word. During block transfers, the register-full flag sets once for each word written into the register.

The register-full flag also clears when the channel goes inactive for any reason. The PPs can sense the flag state using the jump-on-full/empty (00660 and 00670) instructions.

Channel (Marker) Flag

This flag is used by software as a marker and does not affect hardware operation. The flag provides dual-PP I/O driver programs with a synchronization mechanism. The flag is inaccessible to external devices.

The marker flag is set/cleared by the channel flag (00641 and 00651) instructions. A PP can sense the marker flag state using the jump-on-set/clear (1064 and 1065) instructions.

Priority conflicts exist when PPs in the same time slot use this flag. Hardware resolves the marker flag priority conflicts for the MCH 17_8. For other channels, the problem is resolved by software interlocks kept in CM, or by not assigning PPs in the same time slot to the same channel. Any five consecutively-numbered PPs are not in the same time slot.

Error Flag

This flag indicates a data parity error on a channel transfer. The IOU interface sets the error flag when it detects a data parity error on input data. External devices connected through 16-bit channels can also set this flag when detecting an output data parity error. When this flag sets in any internal interface, the channel parity error bit also sets in the IOU fault status register. PP instructions 00661 and 00671 clear and sense the error flag.

Programming for Channel Input/Output

Data transfers to/from external devices are controlled by PP instructions 0064 through 0077. The same instruction set services 8-, 12-, and 16-bit channels. The assignment of PPs, transfer priorities, and resolution of conflicts is a software responsibility.

The channel marker flag and/or software interlocks in CM provide for channel priority and reservation. Proceed as follows after resolving conflicts.

1. Clear error flag. Typical instruction: Jump if error flag set, and clear flag (0066).

2. Verify channel availability. Typical instruction: Jump if active (0064).

3. Verify device availability:
   a. Request device to send status. Typical instruction: Function m (0077).
   b. Wait until device responds. Typical instruction: Jump if active (0064).
   c. Activate channel. Typical instruction: Activate (0074).
   d. Read device status. Typical instruction: Input to A (0070).
   e. Verify error status. Typical instruction: Jump if error flag set (0066).
   f. Analyze device status. Typical instruction: Logical product (0012), zero jump (0004).

4. Prepare for input/output:
   a. Enter number of words to A. Typical instruction: Load d (0014).
   b. Verify channel inactive. Typical instruction: Jump if active (0064).
   c. Prepare device for read/write. Typical instruction: Function m (0077).
   d. Wait until device responds. Typical instruction: Jump if active (0064).

5. Read/write data:
   a. Activate channel. Typical instruction: Activate (0074).
   b. Read/write data. Typical instruction: Input/output A words (0071/0073).
   c. If write, loop until empty. Typical instruction: Jump if full (0066).
   d. Disconnect Channel. Typical instruction: Deactivate (0075).
   e. Verify inactive status. Typical instruction: Jump if active (0064).

6. Verify transfer integrity:
   a. Verify A words were transferred.[1] Typical instruction: Nonzero jump (0005).
   b. Verify error status. Typical instruction: Jump if error flag set (0066).
   c. Verify inactive status. Typical instruction: Jump if active (0064).
   d. Request device. Typical instruction: Function m (0077).
   e. Wait until the device responds. Typical instruction: Jump if active (0064).
   f. Activate channel. Typical instruction: Activate (0074).
   g. Read device status. Typical instruction: Input to A (0070).
   h. Verify error status. Typical instruction: Jump if error flag set (0066).
   i. Analyze device status. Typical instruction: Logical product (0012), nonzero jump (0005).
   j. Verify inactive status. Typical instruction: Jump if active (0064).

---

1. If A is equal to the original value, no words are transferred. If A is not equal to zero, device or another PP ended transfer.

Inter-PP Communications

An NIO PP can communicate with any other NIO PP using any NIO channel. CIO PPs can only communicate with other CIO PPs on channels accessible by the given CIO. Communication between NIO and CIO PPs is limited to channels 15_8 and 17_8.

Either the sending PP or the receiving PP can activate the channel used, after which the sending PP outputs data into the data register and the receiving PP inputs data from the same register.

The transfer rate is one word every 250 ns, except when the transfer is between PPs in different barrels but the same time slot. In such a case, the transfer rate is one word every 500 ns. PPs using the same time slots are as follows.

Slot	NIO PP Number	CIO PP Number
1	0, 5, 20, 25	0, 5
2	1, 6, 21, 26	1, 6
3	2, 7, 22, 27	2, 7
4	3, 10, 23, 30	3, 10
5	4, 11, 24, 31	4, 11

Software must resolve priority and reservation problems arising in inter-PP communications.

PP Program Timing Consideration

Some external equipment requires timing considerations in issuing a function, activating a transfer, or inputting an instruction. Refer to the applicable external equipment reference manual. Such timing considerations may be required, for example, to ensure that the equipment attains a proper speed before data is sent (required by some magnetic tape equipment). Also, equipment terminating a data transfer by resetting the active flag often requires timing considerations in issuing the next function instruction.

Cache Invalidation

When a PP executes 60-bit central write instructions and OS Bounds is enabled, the IOU sends cache memory invalidation requests to the CP. The CP responds by purging the cache memory of any former copies of the words stored in CM. Such invalidation requests are sent during central write instructions, except Read and Set Lock and Read and Set Clear.

NOTE

Cache is not invalidated during the execution of instructions 1001, 1002, 1062, and 1063.

Error Detection and Recovery

The IOU and each PP have fault detection and reporting hardware. The IOU generates and checks parity on all data transferred between PP and PP memory, CM and IOU, and IOU and external devices.

PP Hardware Errors

When a PP hardware error occurs with the enable error stop bit set in the IOU environment control register, the PP with the fault halts (idles). In this case, another PP may perform error detection and logging. When one PP halts from error detection, the remaining PPs are affected only when a PP is waiting for the halted PP to perform some action.

Error reporting from any PP with a fault can be disabled by setting the relevant bit in the IOU fault status mask register. This is normally done when removing a PP from service, and restores normal error reporting from other PPs through the summary status byte.

Channel Parity Errors

The output register 16 bits are checked for parity whenever the register is full. When a parity error is detected, the following takes place.

- Channel error flag in the channel concerned sets.

- Fault status register bit for this channel sets.

- Uncorrected error bit in IOU status summary register sets.

Error reporting from any channel with a fault can be disabled by setting the relevant bit in the IOU fault status mask register. This is normally done when removing a channel from service, and restores normal error reporting from other PPs through the summary status byte.

Parity Errors on Output Data

The IOU sends a data or function word to the channel with parity calculated on all 16 bits of the channel output register. In case of 8- and 12-bit channels, software must ensure that the missing bits in the output register are zeros. This ensures correct channel parity after the unused bits discard.

Software must verify the integrity of a 12-bit channel output data transfer by requesting a status word from the device concerned. When a device detects a parity error on a 12-bit function word output, it does not send any response and the channel remains active and full.

Devices connected through 16-bit channels or the MCH respond to a data word parity error detected at the device by resetting the channel full flag and setting the error flag. The channel remains active and execution of the current output instruction continues. These devices respond to a function word parity error by resetting the active flag and setting the error flag.

The 12-bit channel contains a switch to disable parity checking.

Parity Errors on Input Data

For all channels, the IOU sets the channel error flag whenever it detects a parity error on input data. The IOU regenerates correct parity before storing the data into PP memory.

Timeout

The MCH interface provides a 100-μs timeout counter to ensure that the PP dealing with that channel continues operations when the MCH does not respond to a data transfer command. The timeout interval starts when the MCH goes active or full, and resets when the channel goes inactive or empty. If the IOU receives no response by the end of the timeout interval, it clears the channel active flag.

Function word output does not activate the timeout counter. This allows software to recover from a maintenance access control malfunction.

To allow inter-PP communications without timeout, the timeout is disabled when the MCH interface is deselected from channel 17_8 using connect codes 8 through F.

PC System Console

The CYBER 960/962 uses a 19003-1 or 19003-2 PC system console for interface with the computer operator and/or maintenance personnel. Both the 19003-1 and 19003-2 consoles contain the following.

- PC AT processor with 640K-byte memory

- Alphanumeric keyboard

- Color monitor

- 1.2M-byte flexible disk drive

- Nine RS-232-C communication ports

- One parallel printer port

- Hard disk drive

The 19003-1 console contains a 41M-byte hard disk and is required by CYBER 960 systems using the optional standalone IOU. The 19003-2 console uses a 91M-byte hard disk with a SCSI interface as a dedicated load device (DLD). It is required by all CYBER 962 systems. Both consoles operate under the MS-DOS operating system.

The keyboard and monitor provide the operators and local maintenance personnel with the ability to control, adjust, and monitor the system characteristics. Remote maintenance is available through phonelines and the RS-232-C interface. The System Console Operations/Maintenance guide details PC system console information (listed under Related Manuals).

Initialization

System initialization begins with the IOU, which requires no external hardware or software aid to initialize itself. After the operator initiates the deadstart sequence, a storage device in the IOU provides initialization programs and data for further action.

After the IOU has self-initialized, any or all of the following operations may be performed by way of the system console and deadstart options.

- Load CP control memory

- Initialize CM

- Dump CM

- Run CP quicklook test

- Begin maintenance system load

- Begin operating system load

The following displays (figures 2-32 and 2-33) provide the operator with the initialization options. The console main menu appears when the system is energized and the CTRL-F2 key is pressed. Tables 2-23 and 2-24 explain the options and selections presented in the displays. Refer to CYBER Instruction Package (CIP) Reference Manual for a description of additional options.

```
              CONSOLE MAIN MENU

       S - SYSTEM LOAD OPTIONS
       C - CONSOLE UTILITIES
       M - MAINTENANCE OPTIONS

       CR - SYSTEM LOAD OPTIONS




       CIP PARTITION n SELECTED
       LEVEL nnnnn-nnn

                                        CIP37
```
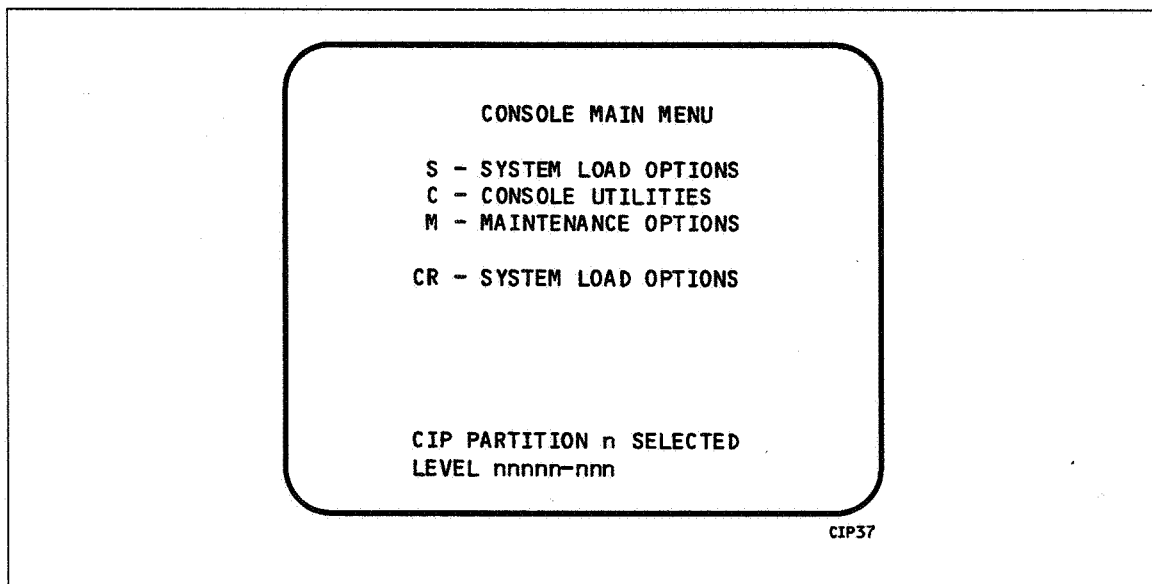
Figure 2-32. Console Main Menu

Table 2-23. Console Main Menu

Option	Description
S	Selects a short deadstart sequence using the deadstart program identified at the bottom of the display. Upon completion of the deadstart sequence, a display for loading system software appears.
C	Allows selection of remote link and communication parameters.
M	Allows selection of maintenance routines and maintenance software updates or installations.
CR	Same as S.

```
                    DEADSTART-REV. OI
  XX YYYYYY-CHANGE DS PRG              PPM CONF = O
  XX+YYYYYY-CHANGE DS PRG INC     NIO BRL CONF = I
           S-SHORT DS                  DLY LOOP = O
           L-LONG DS                   LDS ADDR = 6000
           H-HELP                      CLK FREQ = NORMAL
     PROGRAM O              NIO     PP MAINT REG
     OI OOI4O2         I2    OO  OO  O3  FF  AF  OO  IO  OF
     O2 OO73OI         IB    OO  OO  OO  OO  OO  OO  OO  OO
     O3 OOOOIT         2I    IE  IC  OO  OO  OO  77  O9  F3
     O4 OO75OI         3O    OO  OO  OO  OO  O4  O4  O2  OO
     O5 OO77OI         4O    OO  OO  OO  OO  OO  BO  OO  OB
     O6 OOO34I         8O    OO  IO  OO  OO  OB  OO  OO  3F
     O7 OO74OI         8I    OO  OO  OO  OO  OO  OO  OO  AO
     IO OO7IOI         AO    OO  OO  OO  OO  OO  OO  22  2F
     II OO73OI         CIO
     I2 OOO75O
     I3 OOOOOO
     I4 OOOOOO
     I5 OOOOOO
     I6 OOOOOO
     I7 OOOOOO
     2O OO7II2            BARREL NOT INSTALLED
                                               CIP38C
```

Figure 2-33. Initial Deadstart Display

Table 2-24. Deadstart Display Operator Entries and Functions

Operator Entry	Function
xx yyyyyy	Enters a single word in the deadstart program at xx to a new value yyyyyy (octal).
xx+yyyyyy	Changes words in the deadstart program in sequence starting at xx.
S	Selects a short deadstat sequence.
L	Selects a long deadstat sequence.
H	Brings up a display that lists and explains all available commands. Refer to the hardware operator's guide for detailed information about these commands.

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-MHz rate from 0 to 4095_{10}.

IOU Dedicated Channels

Figure 2-34 illustrates the IOU dedicated channels
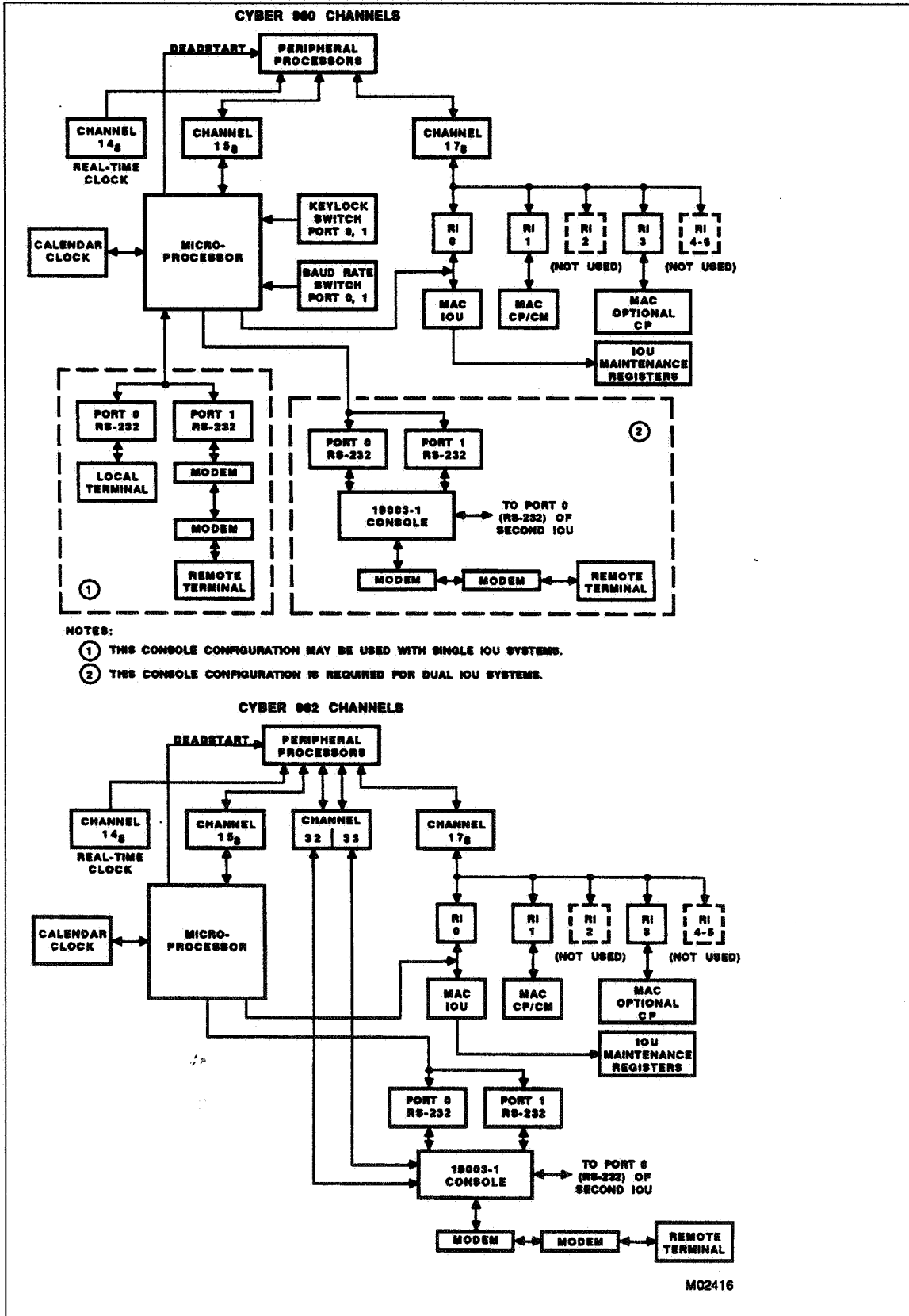
Figure 2-34. IOU Dedicated Channels

Two-Port Multiplexer Programming

Channel 15_8 provides serial communications capability with two external devices through the two-port multiplexer. One port is reserved for maintenance use. With both ports deselected, this channel can also be used for PP-to-PP communications. The arrangement is as shown in figure 2-35.
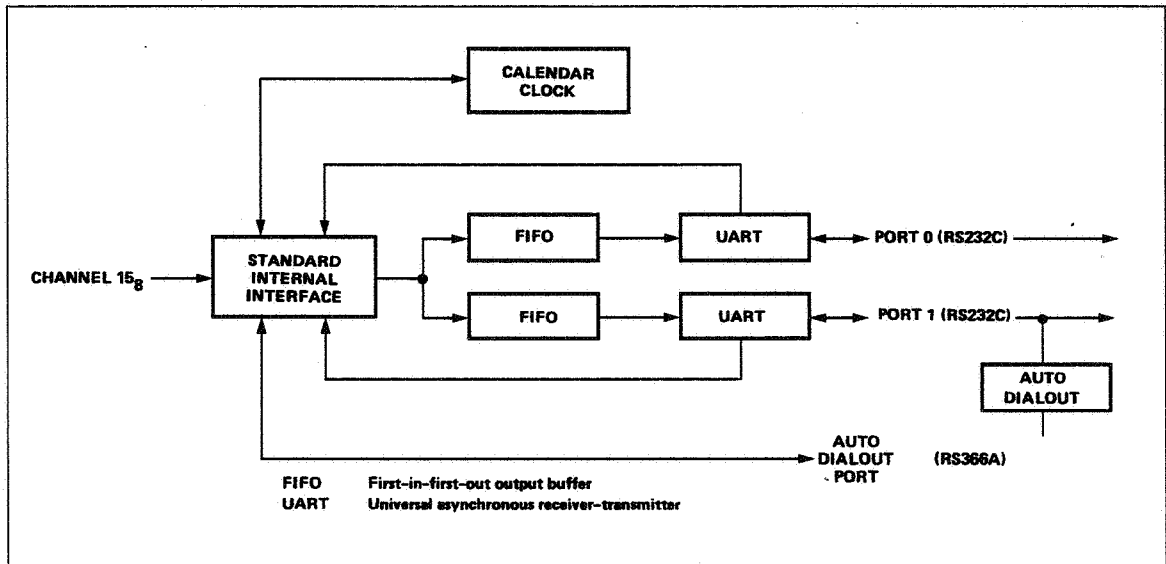
Figure 2-35. Two-Port Multiplexing

The two-port multiplexer can communicate with all devices which use EIA standard RS-232 serial asynchronous interface at baud rates of 110, 300, 600, 1200, 2400, 4800, 9600, or 19 200. The baud rate for each port is independently selectable by switches on the two-port multiplexer logic module.

The multiplexer ports can accommodate data with odd/even parity, five to eight bits per character, and one or two stop bits. The format is set by issuing appropriate function codes.

The two-port multiplexer displays the system deadstart settings on the system console.

The IOU multiplexer supports the following special features.

● Remote deadstart

● Calendar clock

● Internal port-baud-rate selection

Function Words

The two-port multiplexer uses the channel 15_8 rightmost 12 bits as a function word from the PP. The function word specifies the following.

Octal Code	Description
7xxx	Terminal select
6xxx	Terminal deselect
1xxx	Calendar clock/auto dial-out operations
00xx	Read summary status
01xx	Read terminal data
02xx	Output to first in, first out (FIFO) 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)

This code selects the terminal to which the function codes and data transmissions apply:

Code	Description
7000	Select port 0 (future use)
7001	Select port 1 (maintenance use)

Terminal Deselect (6xxx)

This code deselects the two-port multiplexer from channel 15_8. When deselected, channel 15_8 can be used for 16-bit PP-to-PP communications. Inter-PP communications over channel 15_8 should be used with caution since the transfer rate is variable (5 µs/word through 1 ms/word, with 100 µs/word typical).

Calendar Clock (1x02-05)

This code selects several functions which pertain mostly to the calendar clock and auto dial-out functions. These particular functions involve a data transfer from the microprocessor memory to the PPs.

Code	Description
1x02	Read Deadstart Port/Terminal Type. Identifies the port which initiated the last deadstart operation. The multiplexer stores the terminal type and port number of the logged-in deadstart device.
1x04	Read Calendar Clock. Reads the calendar clock after the multiplexer selects either port 0 or port 1.
1x05	Write Calendar Clock. Writes the calendar clock after the multiplexer selects either port 0 or port 1.

Read Pre-DS Copies of P, Q, K, and A Registers (1x20-26)

These codes read the values of the P, Q, K, and A registers that are stored in the two-port multiplexer immediately prior to either a short or long deadstart. Either port 0 or port 1 must be selected. The logical barrel is selected as shown below.

Code	Selection
1x20	NIO Barrel 0
1x21	NIO Barrel 1
1x22	NIO Barrel 2
1x23	NIO Barrel 3
1x24	CIO Barrel 0
1x25	CIO Barrel 1
1x26	All Barrels

Read Pre-DS Copies of Channel Status (1x27)

This code is used to read the copies of NIO and CIO channel status stored in the two-port multiplexer immediately prior to either a short or long deadstart. Either port 0 or port 1 must be selected.

Read Summary Status (00xx)

This code prepares the channel for status input from the selected terminal. A one-word input must follow to read the 12-bit status response, which is as follows.

Bit	Description
52 through 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 prepares the channel for data input from the selected terminal. Channel 15_8 must be activated and one word data input instructions must follow to read in the terminal data. The 12-bit data word has the following format.

Bit	Description
52	Data set ready. Indicates that the Data Set Ready (DSR) signal is active.
53	Data set ready and data carrier detector. Indicates that both Data Set Ready (DSR) and Data Carrier Detector (DCD) signals are active.
54	Overrun. Indicates that the previously received character was not read by the PP before the present character over-wrote the previous character.
55	Framing or parity error. Indicates that the received character does not have a valid stop bit (framing error) or that the received character parity does not agree with the selected parity (parity error).
56 through 63	Data character.

PP Write Output Buffer (02xx)

This code prepares the multiplexer for an output operation to the 64-character output buffer memory. The channel 15_8 active flag must be set before an output operation can proceed.

When an output operation fills the buffer completely and no more locations are available, the multiplexer arbitrarily resets the channel active flag.

Set Operation Mode to Terminal (03xx)

This code sets data terminal operation mode as follows.

Bit	Description
58	Enable loop back. When set, this bit enables a round-trip data path from channel 15_8 to the selected RS-232 port and back to channel 15_8. The RS-232 interface does not transmit data externally in this mode.
59	No parity. When set, this bit eliminates parity bit from transmitted and received character. In such a case, stop bit(s) immediately follow the last data bit.
60	Number of stop bits. Selects number of stop bits (1 or 2) which follow immediately after parity bit.

Bit 60	Description
Clear	1 stop bit
Set	2 stop bits

Bit	Description
61 through 62	Number of bits per character. Select 5, 6, 7, or 8 bits per character.

Code	Bits/Character
00	5
01	6
10	7
11	8

Bit	Description
63	Odd/even parity select. Selects type of parity appended immediately after data bits. Also determines the parity checked on input. When set, selects even parity.

Set/Clear Data Terminal Ready (DTR) (04xx)

This code conditions the data terminal to send or discontinue the Data Terminal Ready (DTR) control signal as follows.

Code Bit 63	Action Taken
Set	DTR set active
Clear	DTR set inactive

Set/Clear Request to Send (RTS) (05xx)

This code sets or clears the terminal control signal Request to Send (RTS) as follows.

Code Bit 63	Action Taken
Set	RTS set active
Clear	RTS set inactive

Master Clear (07xx)

This code master clears the selected port, including any buffer-stored data. The DTR and RTS terminal control signals are not affected.

Programming Considerations

Channel 15_8 communicates one at a time with the terminals connected to the external interface. To establish communications between a PP and the terminal.

1. The PP issues a coded function word to select the terminal.

2. The multiplexer responds by resetting the channel active flag to acknowledge receipt of the function code.

The multiplexer now routes all data to the selected terminal; other function words and data input/output follow.

Data Output

The multiplexer can buffer-store a maximum of 64 characters per port. After 64 characters are stored in the buffer, the multiplexer resets the channel active flag on the last output word. The multiplexer terminates an output transfer when it receives an inactive signal from the channel.

The multiplexer does not permit output to a full buffer. Whenever the output buffer is full and the multiplexer decodes a function code 02XX (PP write output buffer), the multiplexer resets the channel active flag.

Data Input

The multiplexer does not buffer-store input data from the terminal. When the PP does not input previous data before new data arrives, a lost data condition (overrun) exists.

Request to Send and Data Terminal Ready

Request to Send and Data Terminal Ready signals are automatically brought up by the hardware under the following conditions (regardless of the software RTS and DTR bits).

● Data in the universal asynchronous receiver-transmitter (UART) output register

● Data in the FIFO output buffer register

When no data is in the FIFO or UART, the software bit determines RTS and DTR.

Maintenance Channel Programming

Any PP in the IOU can be programmed to perform any or all of the following operations on the CP, CM, and IOU through the 8-bit MCH.

- Initializing registers, controls, and memories

- Monitoring and recording error information

- Verifying error detection and correction hardware

The PP performing such operations is often called the maintenance control unit. The MCH consists of the MCH interface on channel 17_8, a maintenance access control in CP, CM, and IOU, and two sets of interconnecting cables.

The MCH interface contains a selector that connects the MCH to one of up to seven isolated sets of cables. The IOU is element 0 and its maintenance access control connects internally to the selector. The CM shares a common cable and element number with the CP.

MCH Function Words

The MCH function word consists of the connect, opcode, and type fields used as described in the following paragraphs. Table 2-25 describes the MCH function word bit assignments.

The connect field specifies the unit to which the MCH is connected (CP/CM or IOU), controlling selection within the IOU only. The unit 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 without timeout restrictions.

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 use of the Type field depends on the connected unit. With the CP the connected unit, type codes 1 through 7 specify the CP register connected. Also, for the CP, type code 0 specifies that the internal address of the CP register to be connected is specified in a control word sent as two data words immediately following the function word. With the IOU connected unit, the Type field must be set to all zeros. Type code A selects access to CM.

Table 2-25. MCH Function Word Bit Assignments

Field	Code (Hex)	Description
MCH Function Word to IOU		
CONNECT (bits 52-55)	0	Connect IOU maintenance registers
	8-F	PP-to-PP communications
OPCODE (bits 56-59)	3	Clear LED
	4	Prepare for read (control word required)
	5	Prepare for write (control word required)
	6	Master clear ADU and R barrel
	7	Clear fault status registers
	8	Echo
	C	Read IOU summary status (reads one byte, control word not required)
TYPE (bits 60-63)	0	Must be zero
MCH Function Word to CP		
CONNECT (bits 52-55)	1	Connect CP-0 and CMC maintenance registers
	3	Connect CP-1 maintenance registers
OPCODE (bits 56-59)	0	Halt processor
	1	Start processor
	4	Prepare for read (control word required)
	5	Prepare for write (control word required)
	6	Master clear
	7	Clear errors
	8	Echo
TYPE (bits 60-63)	0	CP process state register
	1	Control store micrand data
	2	Clock tuning data
	3	Maintenance access control reference (ROM)
	4	Soft control memories
	5	BDP control memories
	6	Instruction fetch decode memories
	7	Register file
	A	CMC maintenance registers

MCH Control Words

Some function words must be followed by a 16-bit control word specifying the internal address of the register to be connected. The control word must issue as two 8-bit data words (sometimes called address bytes). This is accomplished by outputting two 16-bit words from PP memory where each word's rightmost eight bits comprise the 16-bit control word. Such control words are required after the following:

● Function words to CP with opcodes 4/5

● 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 function word output. A halt opcode halts the processor without damaging the executing process, including the integrity of the halted processor's interunit communications such as CYBER 170 State exchange request communication, CM communications, and the process state. If the process is restarted without performing any other MCH operations, or after performing read/write with precautions as described in the operating systems manual, the process continues undamaged.

MCH Clear LED (Opcode 3)

This operation clears all LEDs associated with pak errors and is intended, however not required, for use at system initialization. For maintenance reasons, this operation can also clear LEDs without initializing and master clearing.

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

Refer to Programming for PP Data Input/Output in this chapter 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/written as 64-bit (8-byte) registers, requiring the input/output of eight MCH data words. Most registers physically smaller than eight bytes are right-justified with zero-fill. Reading a status summary register is an exception in that the status information repeats in each byte.

The IOU may disconnect the MCH without affecting subsequent MCH operations after the following.

● Reading one to eight bytes from any register

● Writing one byte to a CEL register

● Writing one byte to a uncorrected error log register

The following MCH operations on CP registers can be performed with the CP running or halted (when reading or writing registers which may change while being accessed, the CP should be halted to avoid erroneous results).

● Read CP status summary register

● Read CP fault status register

● Read CP CEL registers

● Read CP options installed

● Read CP equipment ID 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 microcode accesses these registers. Refer to table 2-25. When reading or writing registers which may change while being accessed, precautions must be taken as described in the applicable operating systems manual.

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

These operations consist of a single function word output. The master clear immediately and arbitrarily clears the connected unit without regard to possible information loss. The clear-errors operation clears the connected unit error indicators. The unit concerned should be halted to avoid loss of possible (next) error reporting while the errors are cleared.

MCH Echo (Opcode 8)

This operation checks the data path between the MCH and the IOU MAC. Following the operation, MCH is activated and two bytes are sent to IOU MAC. IOU ignores the first byte and latches the second byte in the address holding register, in any data pattern. MCH is deactivated after the second byte is accepted in IOU MAC and the channel is activated followed by an input sequence. IOU MAC sends data (contents of address holding register) upon receiving the active signal and subsequent Empty signals. There is no restriction on the number of data words read.

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

This operation is an alternative, faster means of reading the IOU summary status register. In general terms, proceed as follows.

1. Issue function with opcode c.

2. Input summary status byte.

CIO PP Programming

The CIO PPs are capable of transferring data directly to and from CM without using the PP memory. Three different types of Direct Memory Access (DMA) channel adapters are available for connecting external devices to the CIO PPs. They are Intelligent Subsystem Interface (ISI), Intelligent Peripheral Interface (IPI), and CYBER 170 channel adapter. The following are explained in detail.

- Definition of ISI terms

- Definition of ISI channel adapter signals

- Definition of idle bus mode bits

- Description of bus unit protocol procedures

- Description of ISI adapter function codes

- Explanation of unique ISI adapter registers

- ISI programming examples

- Sequences using the ISI built-in test facilities

- Description of DMA-enhanced CYBER 170 channel adapter

- Description of 170 adapter function codes

- Explanation of unique 170 adapter registers

- 170 adapter programming examples

- Sequences using the 170 adapter built-in test facilities

- Definition of the IPI adapter signals

- Description of IPI protocol procedures

- Description of IPI function codes

- IPI programming examples

Definition of ISI Terms

A working knowledge of the terms described below is desirable before attempting to program the CIO PPs.

Intelligent Subsystem Interface (ISI)

This refers to the electrical interface channel between an ISI adapter and control module(s) and the protocol used to communicate between them. This interface is defined as an ISI-DIFF interface containing differential transmitters/receivers with interlocked and noninterlocked data transfer capabilities.

Control Module

An intelligent peripheral controller that has an ISI interface and accomplishes the functions and definitions of the ISI specification. A control module normally has peripheral devices (that is, disk or tape drives) attached and act as the subsystem controller for these devices.

Master Control Module

A control module which is designed to operate as a bus master on the ISI channel and function as a host or as a bus slave. It can select a bus unit to become a bus slave and issue command blocks to it.

Slave Control Module

A control module which only can act as a slave on behalf of either a master control module or an ISI adapter.

ISI Channel Adapter

The DMA-enhanced ISI channel adapter (ISI adapter) is the interface between the CYBER 180 PP I/O bus and an ISI channel. This adapter contains control and status registers and manages the data transfer between a CYBER PP and an ISI peripheral. The PP addresses the adapter by its channel number. A CYBER adapter is always the bus master.

An ISI adapter with the added capability of directly transferring data between CYBER 180 CM and an ISI channel. This data transfer is initiated by a CYBER PP but runs independently, of and concurrently with, CYBER PP instruction execution.

Bus Master

That ISI adapter which is currently controlling the ISI channel by controlling the Select Hold, Command Sequence, and Sync Out signal lines. It can thus select other bus units and control their operation. One and only one bus master must always be assigned to an ISI channel. This assignment is controlled by the ISI channel bus director.

Bus Slave

That control module which has been selected to respond to but not control the Select Hold, Command Sequence, and Sync Out signal lines on the ISI channel. Only one bus slave can be selected by a bus master at any one time. It is this unit with which the bus master communicates when the bus is in the transaction mode.

Bus Unit

Any control module attached to a ISI channel which may be selected to be either a bus slave or a bus master. A maximum of eight bus units can be attached to a single ISI channel if multiple bus masters are allowed, otherwise a single bus master unit and up to eight bus slave units can be attached.

Port

This refers to the interfacing hardware that comprises one ISI-compatible interface on a bus unit. This consists primarily of drivers and receivers only and does not include the main control logic of the bus unit itself. All bus units will have port A while multichannel units will also have a port B. When two channels interface a single CYBER adapter or control module they are referred to as channel A and channel B and attach to port A and port B respectively.

Transaction Mode

The mode of the ISI data bus when the bus master has activated the Command Sequence line. Either bus master or bus slave data (function, status, or data) may be enabled onto the bus, depending upon the direction of the information Transfer in Progress. Two way communication between a bus master and a bus slave occurs in this mode.

Idle Mode

The mode of the ISI data bus when the Command Sequence line is inactive. Each bus unit continuously transmits its attention and busy state on the attention and busy data lines that correspond to its bus unit address to all ISI bus units on its channel(s). This mode allows any bus unit to ascertain the current state of every other bus unit by reading the bus during idle mode.

ISI Channel Adapter

Programming information for the DMA-enhanced ISI adapter is described in the following paragraphs.

ISI Signal Definition

The ISI channel consists of 16 bidirectional bus lines, two bidirectional bus parity lines, and seven unidirectional control signals. The bus master provides three of these control signals to the bus slave and the bus slave responds by generating three control signals for use by the bus master. The bus director provides one signal to the potential bus masters. Except for special conditions described later, these control signals are said to be interlocked since a given bus master signal will lead to the generation of a corresponding response by the bus slave, which in turn allows the bus master to continue with the next operation. The communication is interlocked to allow easy control of signal timing over longer cable length regardless of internal clock rates. This also allows the implementation of bus units that operate at varying transfer rates.

Data (Bidirectional)

These 16 bidirectional lines transfer information (functions, status, and data) between the bus master and the bus slave. The data bus operates in either the idle or transaction mode.

Attention

A Data Bus signal line which is defined during bus idle mode that, when active, indicates the bus unit assigned to this bit address is requesting the bus master to read its status to determine the reason for the attention request. The bus master intelligence will normally poll the bus units for their attention signals and act upon their requests for action.

Busy

A Data Bus signal which is defined during bus idle mode that, when active, indicates that the bus unit assigned to this bit address is not able to currently respond to any transaction dialog. This signal is usually activated by a bus unit when it is busy performing an operation requested by the bus master.

Bus Parity (Bidirectional)

A given Bus Parity line activates or deactivates as needed to ensure that the associated nine lines (eight data plus parity) have an odd number of lines activated when valid data is being transferred. Correct parity is not present on the bus during the idle bus mode.

During the bit significant response (BSR) transfer, if more than one bus unit responds (error condition), the parity may be incorrect.

Select Hold (Bus Master)

The bus master activates this line at the beginning of all transfers to a bus unit and this line remains active until the transfer is complete. When this line goes active, all bus units expect either a bus slave selection sequence or a broadcast master reset sequence. The bus unit selected is the bus slave. When this line is deactivated, the selected bus slave deselects by deactivating its Select Active. The channel then has no bus slave selected.

Select Active (Bus Slave)

A bus unit activates this signal when it recognizes its own address during unit selection and maintains the signal until the bus master deactivates Select Hold or until an error beyond the bus slave's error handling capabilities occurs causing the bus slave to terminate the command being executed by the bus slave. The bus unit which activated this signal is called the bus slave. The Select Active line may not be valid during execution of the diagnostics command.

Command Sequence (Bus Master)

This line is used by the current bus master to control the I/O protocol and bus utilization. Whenever Command Sequence is inactive (idle mode), each bus unit places its respective attention and busy status onto the two bus lines assigned to each bus unit for this purpose. During a bus unit select sequence, the Command Sequence line activiates so that all of the bus units relinquish the bus. Once a bus unit is selected to be the current bus slave, the Command Sequence line is used to control the protocol of the I/O transfer sequence.

Each leading edge of Command Sequence causes the bus units to deactivate their attention and busy signals and also causes the selected bus slave to monitor the bus for the pending function word transfer. Between function word sequences (after command sequence is deactivated for one function but prior to its activation for the next function), the ISI data bus assumes the idle mode (each bus unit enables its attention and busy status onto the data bus) so that the bus master may sample activity in all bus units while remaining selected to a particular bus unit.

Sync Out (Bus Master)

The bus master uses this signal to validate the information and control signals transmitted by it on the ISI channel. Except for the broadcast master reset function, this signal must have a corresponding Sync In signal from the selected bus slave. The interchange of Sync Out and Sync In signals varies, depending on whether interlocked or noninterlocked mode of operation is being used.

In noninterlocked mode, information exchanges are initiated by the Sync Out signal from the bus master. The bus master may send a train of up to 16 Sync Out pulses without waiting for a Sync In pulse from the bus slave, and may activate a Sync Out pulse whether a Sync In pulse is active or not.

When operating in noninterlocked mode, the bus master must maintain a Sync Out/Sync In difference counter to ensure that a Sync In is received for every Sync Out pulse sent by it. If the difference count reaches 16, the bus master must stop sending Sync Outs until it has received a Sync In to reduce the count below 16. (Note that an interlocked mode of operation may thus be achieved, using pulsed Sync

Outs and Sync Ins, with the bus slave being in control of the transfer rate). In interlocked mode, the Sync Out and Sync In pulses are exchanged with a communication link being maintained.

Sync In (Bus Slave)

This signal is used to validate information and control signals transmitted on the ISI channel by the bus slave. Except for the broadcast master reset function, it must have a one-to-one correspondence with Sync Out signals sent by the bus master. The bus slave responds to a legally issued (that is, Busy and Pause signals are inactive) Sync Out with a Sync In signal within 1 ms maximum, including cable delay. If the bus slave does not respond within this time, the bus master considers it to be down and that a time-out has occurred.

In the CYBER Adapter, if this occurs the Transfer in Progress is terminated and the time-out status flag is set.

During the interlocked mode of operation, the Sync In signal is sent in response to a Sync Out signal from the adapter, and must stay active until Sync Out is deactivated.

During the noninterlocked mode of operation, Sync In is a pulse of defined duration and period. A Sync In pulse is sent by the bus Slave In response to each Sync Out pulse received, delayed in time only by the internal logic of the bus slave. The internal delay may be several Sync Out pulses in time, however. To ensure that all Sync Out pulses are responded to, the bus master maintains a difference count of Sync Outs received and responded to.

The pulse width and period of the Sync Out pulse that the bus slave will transmit is a function of its maximum transfer rate. This may be determined from the bus slave characteristic word.

The pulse width is one-half the minimum period plus or minus 10 percent. The minimum period is defined by the maximum frequency that the bus slave being addressed can accept. The maximum period is undefined.

(Example: A control module has a maximum transfer rate of 6M bytes/s. Because the data bus is two bytes wide, the exchange rate is three megawords, or 3 MHz Sync Out/Sync In pulses. The minimum period is, therefore, 333 ns and the maximum pulse width is 166 ±16 ns).

A Sync In pulse may be activated at any time after the trailing edge of the Sync Out pulse, whether there is another Sync Out active or not.

Pause (Bus Slave)

This signal is used by the bus slave to tell the bus master that it is unable to respond to the bus master Sync Out at this time. The Pause line is most commonly used to indicate buffer full or buffer empty conditions or other similar conditions which are temporary in nature. The Pause signal is active only when the current bus slave is selected, and in many cases is not activated until the bus master attempts a particular function which the bus slave cannot perform at present. The bus master may then decide to perform a different function or wait until pause is deactivated.

The bus slave guarantees the timing relationship of the Pause line during data transfers so that the bus master has adequate time to inhibit the next Sync Out. If the Pause line is used for applications other than data buffer control, the bus slave

must guarantee that the Pause line goes active at a point that will allow the host to react accordingly (see individual product specifications for exact usage of the Pause line). The Pause signal differs from the bus slave Busy signal in that the bus slave is still available for communication with the bus master but on a limited basis. When the Busy line is active, no communication can be performed with the bus slave.

Idle Bus Mode Bit Definitions

When the bus is in the idle mode, that is, when the Command Sequence is inactive, the bus will be defined as eight Attention lines (one from each bus unit) and eight Busy lines (one from each bus unit) as shown in figure 2-36, Idle Mode Bus Busy/Attention Bit Format.

Bus Unit Attention

When one of these eight signal lines goes active, the bus unit assigned to this line is requesting the current bus master to read the bus unit status to determine the reason for the Attention request. The bus master either polls the signals or treats them as interrupts to determine when a bus unit requires service. Refer to figure 2-36 for bit format.

Bus Unit Busy

When one of these eight signal lines goes active, the bus unit assigned to this line is Busy and is informing the bus master that no transaction dialog will be recognized by this bus unit. The active state of the Busy signal signifies that this bus unit will not be able to respond to the bus master, but when this line is active, the bus master may request a transaction dialog with the bus unit. This line is usually activated in response to an operation requested by the bus master. The Busy signal may not be valid during execution of the diagnostic command. Refer to figure 2-36 for bit format.

```
                        Busy BITS            |       Attention BITS
        BUS                                  |
        BIT     15  14  13  12  11  10   9   8 | 7   6   5   4   3   2   1   0
                +-------------------------------+-------------------------------+
                | 0   1   2   3   4   5   6   7 | 0   1   2   3   4   5   6   7 |
                +-------------------------------+-------------------------------+
                        Bus Unit Address               Bus Unit Address
```
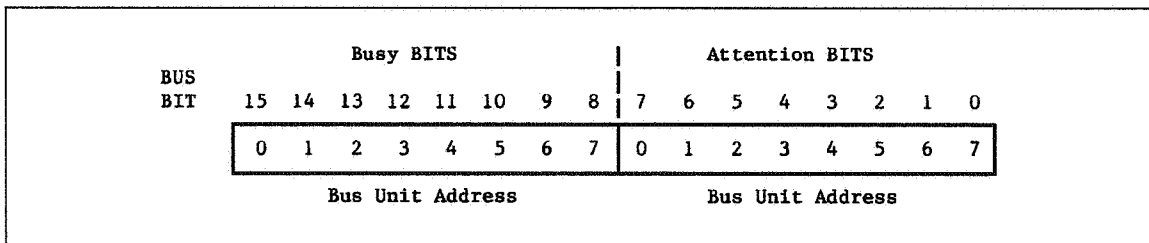
Figure 2-36. Idle Mode Bus Busy/Attention Bit Format

Bus Unit Protocol Procedures

The bus unit protocol is grouped into five categories, four of which use the bus in the transaction mode and one which uses the bus in the idle mode. The five categories are:

1. Bus Unit Selection Operation – Selects one of the eight bus units on the bus for further bus dialog.

2. Function Word Transfer – Defines the utilization of the bus to transfer a one-word function which defines the bus direction and the generalized contents of the subsequent information exchange blocks to be transferred.

3. Information Exchange – The information following the function word transfer. The specific type of information is defined by the function word transferred between the bus master and the bus slave.

4. Broadcast Master Reset – A single word transmitted by the bus master simultaneously to all bus units on the bus, which requests all available bus units to simultaneously perform a master reset operation.

5. Bus Idle Mode – Defines the bus as containing each bus unit's Attention and Busy status signals.

Bus Unit Selection Operation

The bus unit select operation, which is always executed in interlocked mode, allows the bus master to select one of the bus units for further bus dialogs.

The parallel interface is designed to allow multiple bus units to be daisy chained. Up to eight bus units can be connected to one ISI channel, each with its own unique address. The bus master can only communicate directly with one bus unit at a time but can clear all eight bus units simultaneously using a broadcast master reset function. Bus unit selection is only required once per transfer sequence. Once connected, a bus master can transfer multiple information exchange blocks without repeating the bus unit select sequence. Bus unit selection is only necessary in two instances:

1. After the bus master has terminated a transfer sequence and has deselected the previously selected bus slave by deactivating Select Hold, or

2. The bus slave has terminated the transfer by deactivating Select Active (this only occurs if any error conditions occurs which is beyond the bus slave's ability to recover).

The bus unit select process begins when the bus master activiates the Select Hold
signal. Select Hold remains active throughout the entire transfer sequence. The bus
master next activates the Command Sequence line to terminate the bus idle mode and
define the start of the bus unit select sequence. Command Sequence remains active
throughout the selection operation. The bus master places the bus unit select word
(refer to figure 2-37) on the bus and activates the Sync Out line to validate the bus
contents.

If the bus unit select bit in the select word is set and the broadcast master reset bit is
clear, each bus unit samples the three bit bus unit address portion of the word. If the
bits match the bus unit address, the unit responds with Select Active and Sync In. The
Select Active signal is held active as long as the bus unit is communicating with the
bus master and does not drop until Select Hold goes inactive (except in the case of an
error). Once Sync In goes active, the bus master may deactivate Sync Out, wait for
Sync In to be deactivated, and continue with the selected bus slave.

Following the select sequence, the bus master has the option of requesting a bit
significant response (BSR) from the selected bus unit or concluding the selection
operation (refer to figure 2-38). The BSR is significant for daisy-chained bus units. The
BSR is used by the bus master to verify that only the correct bus unit responded to
the select word. In multiple bus unit systems, it is possible for hardware failures to
cause more than one bus unit to respond to an address code. It is also possible that
the wrong bus unit select code could be received so that the wrong bus unit might
respond.

The BSR can, therefore, be useful for maintaining system integrity. If the BSR is
desired, the bus master must generate a Sync Out signal to request the BSR prior to
deactivating Command Sequence. Once the Sync In goes active, the bus master must
wait a delay time to ensure that all bus units have had a chance to respond either
correctly or incorrectly to the address code, then it deactivates Sync Out. When the
Sync In signal is deactivated by the bus unit, the bus master may deactivate Command
Sequence which will conclude the bus unit select sequence. If the BSR is not desired,
Command Sequence must be deactivated without activating the Sync Out for the BSR.

From a systems standpoint, it is important to note that the amount of time the bus
master must delay to ensure that all bus units have had time to respond to the BSR
request is dependent on the type of bus units present on the daisy chain.

1	1	1	1	1	1										
5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0

(Reserved)							BSS	PO	R	BM RS	Not		Bus Slave Address		
0	0	0	0	0	0	0					Used		2	1	0

BSS   Bus Slave Select. Must be set for IMS select format. (Not to be used for broadcast master reset operations). (Also called Intelligent Module Select.)

PO   Priority Override (Dual-channel option only).

R   Reserve (Dual-channel option only). This bit, when set, forces deselection of the alternate ISI interface port and selection via the requesting port.

BMRS   Broadcast Master Reset. This bit, when set, causes all bus units to perform a Master Reset Function. This bit takes precedence over all other bits in the Select word.

Figure 2-37. Bus Unit Select Word

1	1	1	1	1	1										
5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0

(Unused) Must be zero								0*	1*	2*	3*	4*	5*	6*	7*

*Selected Bus Unit Addresses
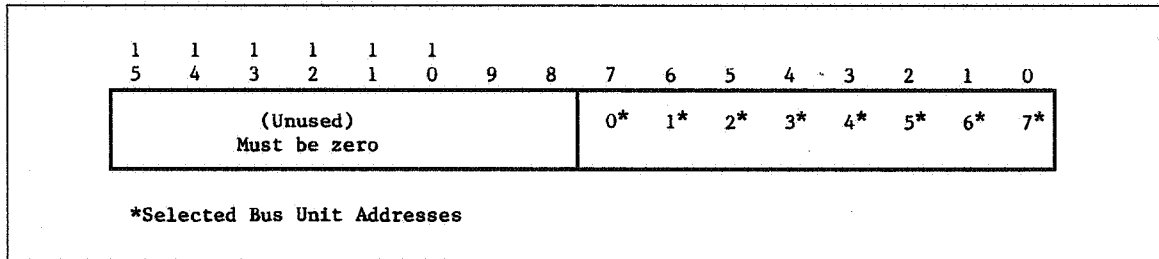
Figure 2-38. Bit Significant Response

Function Transfer

After an individual bus unit has been selected and becomes the bus slave, function word transfers may be made by the bus master. Function word transfers must precede every information exchange operation and are always executed in interlocked mode. A function word transfer is defined as the first word transferred to a selected bus unit (bus slave) after the activation of the Command Sequence line.

To transfer a function word, the bus master activates Command Sequence, places the function word on the data bus, and activates Sync Out. The selected bus slave activates Sync In as a response. The bus master may then deactivate Sync Out and continue with the defined information exhange (if required).

The format of the function word transferred to the bus slave is shown in figure 2-39. This word contains bits which are used for control of the electrical interface as well as bits to define the contents of the following information exchanges at a higher level. The bits for higher level interpretation are: the function code bits, terminate, zero-fill inhibit, data/function and selective reset. The bits applicable for the interface control are: write/read, selective reset, clear attention, noninterlocked, and bus unit select.

1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0
W/R	SR	D/F	Clr Att	ZFI	Ter	Non Int Lk	BSS	Function Buffer Address
								7	6	5	4	3	2	1	0
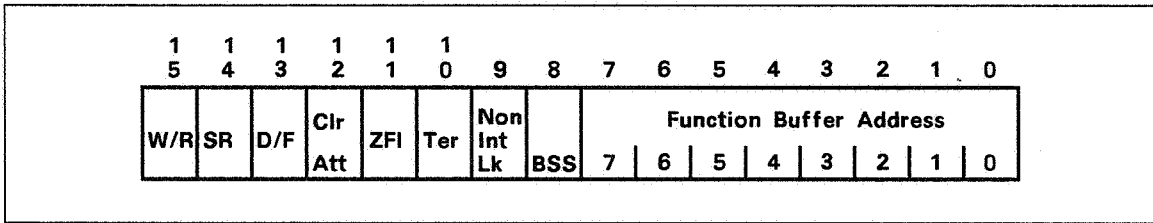
Figure 2-39. Bus Slave Transferred Function Word

Bit	Description
W/R	Write/Read Select. This bit, when set, defines the bus direction for the following information exchange is to be from the bus master to the bus slave (the bus slave receives information). When clear, the bus direction for the following information exhange is defined to be from the bus slave to the bus master (the bus slave transmits information).
SR	Select Reset. This bit, when set, causes the bus slave to terminate all ongoing processes and perform a master reset operation. An information exchange does not occur following the function word.
D/F	Data/Function. This bit differentiates between two general types of information exchanges. When set, the information exchange content is defined as data. When clear, the information exchange information content is defined as function (command, status).
CLR ATT	Clear Attention. When set, this bit deactivates the Attention line of the selected bus unit (if any) which is defined during the bus idle mode of operation. The clear is accomplished at the deactivation of Command Sequence.
ZFI	Zero Fill Inhibit. This bit applies only for data information exchanges to the bus slave. When clear, the zero-fill function is performed. When set, the zero-fill function is inhibited. The zero-fill operation does not occur until the transfer is terminated using the terminate bit.
TER	Terminate. This bit applies only for data information exchanges. When set, this bit allows a data information exchange to be terminated prematurely. Termination of a data information exchange does not occur until Command Sequence is deactivated and both the terminate bit and the data/function bit are set in the function word.
NON INT LK	Noninterlocked. When set, information transfers between the bus master and the bus slave will be in the noninterlocked mode. When clear, information interchanges are in the interlocked mode. Bus units not capable of operating in a noninterlocked mode does not recognize bit 9. The bus master must provide a time-out for receipt of the first Sync In pulse to determine if a malfunction has occurred.
BSS	Bus Slave Select. Must be clear for a function word.
Function Code	This 8-bit code further defines the type of information exchange. This address is a pointer into the 256-word function buffer.

Information Exchange

Information exchanges are defined as information (data, status, command blocks) transferred between the bus master and the selected bus unit via the bus after the function word transfer and until the Command Sequence line is deactivated. Note that multiple information exchanges may occur during the time the bus unit is selected.

Information exchanges may take place in two ways, interlocked mode and noninterlocked mode. All selection and function word transfers must be handled in interlocked mode. Data, status, and command block transfers may be handled in either mode.

For information exchanges to the bus slave (write), the data bus contains the information to be transferred and the Sync Out informs the bus slave when to sample the bus. For information exchanges from the bus slave (read), the data bus contains the information to be transferred and Sync In informs the bus master when to sample the bus. The handshaking of Sync Out and Sync In continues until either all of the information is transferred or an error condition is detected.

An information exchange can be terminated with or without deselection of the selected bus unit. If an information exchange is terminated without deselecting the bus unit, then a subsequent function word may be sent to the same bus unit to specify another type of information exchange without reselcting the bus unit. If an information exchange is terminated and the bus slave is to be deselected (Select Hold deactivated), then a new bus unit select operation must occur before another function word can be transmitted to define another information exchange.

Broadcast Master Reset

The broadcast master reset function enables the bus master to master reset all daisy-chained bus units connected to the bus with a single-word bus transfer. The interface control of the Select Hold and Command Sequence signals which specify a broadcast master reset function is the same as for the beginning of the bus unit select operation, that is, activation of both lines. Bit 5 of the bus unit select word is set for a reset operation and reset for a bus unit select operation.

This operation is unique as no Sync In responses are generated by any of the bus units in response to the adapter Sync Out pulse that initiates the reset. This eliminates bus conflicts and ensures that all units on the bus have a guaranteed time interval in which to recognize the reset. Once a master reset operation has been initiated, it cannot be stopped by the bus master. Each bus unit indicates completion of its reset operation by activating its Attention bus line. The adapter must wait for the Attention response before attempting to communicate with any available bus unit.

Function Codes

The CYBER DMA-enhanced ISI adapter is an interface between the CYBER PP I/O channel bus and an ISI channel. This adapter is totally controlled by a PP through the issuance of functions and data transfers to it using standard PP I/O instructions. An adapter is addressed by a PP through the I/O channel it occupies on the PP I/O bus.

The ISI channel adapter uses the least significant 12 bits of data from a PP as the function code. The function word from the PP is translated to specify the operating condition of the adapter as shown in the following table. The adapter responds with an inactive status only when a legal function code is translated.

Hex	Octal	Function
0000	000000	Master Clear
0100	000400	Read Control Register
0101	000401	Write Control Register
0200	001000	Read Mask Register
0201	001001	Write Mask Register
0300	001400	Read Error Status
0400	002000	Read Operational Status
0500	002400	Request ISI Idle Status
0600	003000	Read T Registers
0601	003001	Write T Registers
0700	003400	Read Test Seed
0701	003401	Write Test Seed
0800	004000	Clear Select Hold
0801	004001	Set Select Hold
0900	004400	Clear Command Sequence
0901	004401	Set Command Sequence
0A00	005000	Force Sync Out
0B00	005400	Set PP Mode
0C00	006000	Clear DMA Mode
0C01	006001	Set DMA Mode
0D00	006400	Clear Echo Mode
0D01	006401	Set Echo Mode
0E00	007000	Clear T Registers
0F00	007400	Illegal

Master Clear Channel (0000)

This function master clears the ISI adapter and places it in its power-on state. All registers and internal data buffers are cleared. The channel deselects (Command Sequence and Select Hold clear) and the adapter reverts to a non-bus-master state. The channel is set inactive and empty with the error and channel flag clear.

Read Control Register 16 Bits (0100)

This function is used to read the adapter control register. After the PP sends this function and an activate signal, the ISI adapter responds with a Full signal accompanied by the contents of the control register.

Write Control Register (0101)

This function writes the adapter control register. After the PP sends this function and an activate, the ISI adapter accepts the following output data as control register write data.

Read Flag Mask Register (0200)

This function reads the adapter flag mask register. After the PP sends this function and an activate, the adapter responds with a Full signal accompanied by a copy of the flag mask register contents.

Write Flag Mask Register (0201)

This function writes the adapter flag mask register. After the PP sends this function and an activate, the adapter accepts the following data as flag mask register write data.

Read Error Status Register (0300)

This function reads the adapter error status register. After the PP sends this function and an activate, the adapter reponds with a Full signal and the contents of the error status register. The error status register is cleared when read.

Read Operational Status Register (0400)

This function reads the adapter operational status register. After the PP sends this function and an activate, the adapter responds with a Full signal and the contents of the operational status register.

Request ISI Idle Status (0500)

This function reads the idle bus status word. After the PP sends this function and an activate, the adapter responds with a Full signal and a copy of the current contents of the ISI data bus. If the command sequence line is set, the adapter clears it and waits the appropriate amount of time before sampling the ISI bus.

Read T Registers (0600)

This function reads the contents of the adapter's T registers. After the PP sends this function and an activate, the adapter sends three 16-bit channel words which contain the current contents of the T register. The T registers are read in the following order.

1. First word
2. Second word
3. Third word
4. Byte count
5. Most significant CM address bits
6. Least significant CM address bits

Write T Registers (0601)

This function initiates writing the adapter T registers. After the PP sends this function and an activate, the adapter accepts the next three channel words as T register write data. If the PP sends more than three words to the adapter, the fourth word remains in the adapter channel data register with the Full flag set. The T registers are written in the following order.

1. First word
2. Second word
3. Third word
4. Byte count
5. Most significant CM address bits
6. Least significant CM address bits

Read Test Seed (0700)

This function reads the contents of the test mode operand generator (test seed). After the PP sends this function and an activate, the adapter responds with a Full signal and a copy of the current contents of the 8-bit operand generator. The upper eight bits of the read data are always a copy of the lower eight bits.

NOTE
_____

Data and parity errors may occur if the operand generator is read during a data transfer with test mode enabled.

_____

Write Test Seed (0701)

This function writes to the operand generator (test seed). After the PP sends this function and an activate, the adapter accepts the following data as operand generator write data. Only the least significant eight bits of write data are used.

Clear Select Hold (0800)

This function deactivates the Select Hold and Command Sequence lines on the ISI channel. Deactivating this line deselects any selected bus slaves.

Set Select Hold (0801)

This function activates the Select Hold line on the ISI channel. This line remains active until one of the following conditions occur.

● The Clear Select Hold function is sent.

● The Clear Echo Mode function is sent.

● The Master Clear function is sent.

● The IOU is deadstarted (either SDS or LDS).

● The channel is deadstarted via the maintenance register.

● The adapter is powered down.

Clear Command Sequence (0900)

This function deactivates the Command Sequence line on the ISI channel. Deactivating this line puts the ISI channel into the idle mode.

Set Command Sequence (0901)

This function activates the Command Sequence and Select Hold lines of the ISI channel. This function also puts the adapter into PP mode, thus enabling PP-to-ISI communication. The Command Sequence line remains active until one of the following is sent.

● The Clear Command Sequence function

● The Request Idle Status function

● The Clear Select Hold function

Force Sync Out (0A00)

This function forces a Sync Out on the ISI channel. This is used during the select process to request the Bit Significant Response or it can be used during a PP input to start the data transfer between a bus slave and the adapter. This function also puts the adapter into PP and input mode.

NOTE

The Select Hold and Command Sequence lines must be active prior to sending the Force Sync Out function or the Sync Out is not issued.

Set PP Mode (0B00)

This function puts the adapter into PP mode. In PP mode the adapter transfer registers are connected to the channel data register. Once connected, any PP output to the adapter causes the output data to be transmitted on the ISI channel. If data is available on the ISI channel, the adapter channel data register is set full and the PP can input the contents. The adapter remains in PP mode until one of the following functions is sent.

● Set PP Mode

● Set Command Sequence

● Set Echo Mode

● Force Sync Out

If the adapter is in DMA mode when the Set PP Mode function is received, the function word is held in the channel data register (the channel remains active and full) until the Transfer in Progress signal clears. At this time the channel goes inactive and the channel data register is connected to the transfer register.

Clear DMA Mode (0C00)

This function clears DMA mode. After the PP sends this function, the DMA mode bit of the Operation Status register clears. The adapter inhibits issuing any further request to either CM or the ISI channel. The Transfer in Progress bit in the operation status register clears when all outstanding requests are honored.

NOTE

A DMA transfer taking place at the time the clear DMA mode function is issued cannot be restarted.

Set DMA Mode (0C01)

This function sets the adapter into DMA mode. This function is only valid after the Command Sequence is set for the DMA transfer. If this function precedes the transfer function word, then DMA mode is not set until the function word Sync In signal has been received. This sequence is required because the transfer function word must originate from a PP, and it also contains the direction of the transfer. While DMA mode is enabled:

1. DMA mode bit is set in the operational status register.

2. The assembly/disassembly unit (ADU) registers are connected to the ISI transfer registers.

3. DMA transfers occur provided the T register is not equal to zero and the bus slave is in the proper state.

4. All other registers on the adapter can be accessed by the PP without affecting the transfer.

DMA remains set until one of the following conditions occur.

● A Clear DMA Mode function is sent.

● A Set PP Mode function is sent and Transfer in Progress is clear.

● A Master Clear function is sent.

● A Clear T Register function is sent.

● The Command Sequence line drops (refer to the earlier description on Set Command Sequence for conditions that drop the command sequence line).

NOTE

If the Command Sequence line drops while Transfer in Progress is set, the state of the adapter is undefined and a master clear function must be issued.

Clear Echo Mode (0D00)

This function clears echo mode and returns the adapter's receivers and transmitters to their normal state. This function also clears the Select Hold and Command Sequence lines.

Set Echo Mode (0D01)

This function sets echo mode. Echo mode enables data to flow from the output buffer through the transmitters and receivers and into the input buffer. When this function is issued, it also:

● Sets Command Sequence.

● Sets Select Hold.

● Inhibits the Sync Out transmitter.

● Connects Sync Out to Sync In internally on the adapter.

Clear T Registers (0E00)

This function clears the T register and the T-prime (T') register as well as any internal data buffers.

Illegal Function (0F00)

This is the only illegal adapter function and is ignored. An inactive is not returned upon the issuance of this function.

Registers

The registers used in the CIO subsystem of the I/O unit are described in the following text.

Control Register

The control register selects the different operating modes of the ISI adapter. This register can be read or written with the $010X_{16}$ functions. Power-on master clear or the master clear channel function (0000) also clears this register.

Bit Descriptions

Control register bit assignments and descriptions are as follows.

Bit	Description
48	Enable Cache Invalidate
49	Port B Enable
50	Disable ISI Timeout
51	Enable Test Mode
52	Inhibit Test Mode Increment
53	Inhibit Sync Out
54	Inhibit Outstanding Request Counter
55	Enable Idle Test
56	Enable Force Error Codes
57 and 58	Not Used
59 through 63	Force Error Codes (0-4)

Enable Cache Invalidate (Bit 48)

This bit, when set, generates a Cache Invalidate signal with every fourth and last CM write of a transfer.

Port B Enable (Bit 49)

This bit, when set, enables the adapter to connect to the ISI channel on port B. When the bit is clear, the adapter is connected to the ISI channel on port A.

NOTE

The user should ensure that all bus slaves on the selected port are deselcted prior to enabling the other port. If this is not done, the ISI channel could hang in an undetermined state, and a master clear function must be issued.

Disable ISI Timeout (Bit 50)

This bit, when set, prevents the ISI deadman timer from timing out.

Enable Test Mode (Bit 51)

This bit, when set, enables the test mode feature of the adapter. In test mode, the receivers and tranmitters are disabled and the operand generator (test seed) provides the necessary data and communication signals to simulate high-speed I/O transfers. The feature does not test the actual receivers and transmitters.

Inhibit Test Mode Increment (Bit 52)

This bit, when set, prevents the test mode operand generator (test seed) from incrementing. This bit is also used to enable solid data patterns during test mode execution.

Inhibit Sync Out (Bit 53)

This bit, when set, prevents the sending of Sync Out. If this bit is set during a DMA output, the output requests backup in the adapter, filling the output buffer and the disassembly registers. This provides a good method of checking all the logic responsible for controlling the output data flow.

Inhibit Outstanding Request Counter (Bit 54)

This bit is used in conjunction with echo mode to test the input buffer overflow detection network. With this bit set and echo mode enabled, the adapter is allowed to issue more than 16 requests without getting a Sync In. This causes an input buffer overflow error.

Enable Idle Test (Bit 55)

This bit is used in conjunction with echo mode to test the setting of the channel flag. With this bit set and echo mode enabled, the adapter translates output data as ISI channel idle data. This makes it possible to test the flag mask network. The sequence for testing is:

1. Set echo mode.

2. Set enable idle test.

3. Output idle word. (The flag sets at this time if enabled.)

4. Clear enable idle test.

5. Input idle word.

6. Clear echo mode.

Enable Force Error Codes (Bit 56)

This bit, when set, enables the decoding of the force error code bits (bits 59 through 63 of the control register). These code bits are used to test the various parity networks in the adapter.

Force Error Codes (Bits 59-63)

These bits are enabled by the Enable Force Error Code bit of the control register. The bit decodes are as follows.

Octal	Description
00	Not used
01	Invert Function Decode PROM Parity

This condition inverts the parity bit for the function decode PROMs. The JX error sets in the error status register, along with the JX board LED.

| 02 | Invert Mask Register Parity |

This condition inverts the parity bit at the parity checker for the mask register. The JX error sets in the error status register, along with the JX board LED.

| 03 | Invert T Register Parity |

This condition inverts the parity at the parity checker for the T register byte count. An error condition is forced only when a T"=T condition occurs. The JX error sets in the error status register, along with the JX board LED.

| 04 | Force Invalid Response |

This condition forces an illegal translation of the CM response code. This error condition is forced only during an actual CM response. Therefore a CM reference must be made in order to force this error. The Invalid Response bit sets in the error status register.

| 05 | Invert Response Code Parity |

This condition inverts the parity of the CM response code at the parity checker. An error is detected only if a CM response occurs. The JX error sets in the error status register, along with the JX board LED.

| 06 | Invert Control Register Parity |

This condition inverts the parity for the control register at the parity checker. The JX error sets in the error status register, along with the JX board LED.

| 07 | Not used |
| 10 | Force Zero Input Buffer Parity |

This condition forces the parity bits for the input data to zero. If an even number of bits in a byte is being input, a parity error occurs. The JZ error sets in the error status register, along with the JZ board LED.

Octal	Description
11	Force Zero A Port A Input Parity

This condition forces the parity bit for port A input data to zero. If an even number of bits in a byte is being input on port A, an error occurs. The JZ and ISI input errors set in the error status register, along with the JZ board LED.

12	Force Zero Port B Input Parity

This condition forces the parity bit for port B input data to zero. If an even number of bits in a byte is being input on port B, an error occurs. The JZ and ISI input errors set in the error status register, along with the JZ board LED.

13	Invert Test Mode Operand Generator Parity

This condition inverts the parity bit associated with the test mode operand generator (test seed). An error occurs whenever this condition is enabled and the operand generator is used. The following table describes the error conditions and their results.

Condition	Error Result
Reading Operand Generator	JX error JX board LED lights
Test Mode and DMA Write	ISI input error JZ error JZ board LED
Test Mode and DMA Read	Test mode compare error

14	Invert Output Byte 1 Parity

This condition inverts the parity bit associated with output data bits 56 through 63. Any output done with this condition enabled generates an error. The JZ error sets in the error status register, along with the JZ board LED. Bad parity is also transmitted to the external device.

15	Invert Output Byte 0 Parity

This condition inverts the parity bit associated with output data bits 48 through 55. Any output done with this condition enabled generates an error. The JZ error sets in the error status register along with the JZ board LED. Bad parity is also transmitted to the external device.

16, 17	Not used

Octal	Description
20	Force Channel Input Parity Bit 0 Low

This condition forces the parity bit low for bits 48 through 55 as the data is written into the assembly buffer. When the 64-bit CM word is read from the assembly buffer, a parity error is detected. This sets bits 59 and 62 of the error status register.

21   Force Channel Input Parity Bit 1 Low

This condition forces the parity bit low for bits 56 through 63 as the data is written into the assembly buffer. When the 64-bit CM word is read from the assembly buffer, a parity error is detected. This sets bits 59 and 62 of the error status register.

22   Force T Data Parity Bit Low

This condition forces the parity bit low for the T register data as it enters the T' register. When the T' register is transferred to the T register, a parity error is detected, setting bit 62 of the error status register. This error condition inhibits CM requests and remains present until the channel is master cleared or a clear T register function is performed on the channel.

23   Invert Upper Channel Output Parity Bit

This condition inverts the parity bit for bits 48 through 55 as the data is read from the disassembly buffer, setting bits 59 and 62 of the error status register.

24   Invert Lower Channel Output Parity Bit

This condition inverts the parity bit for bits 56 through 63 as the data is read from the disassembly buffer, setting bits 59 and 62 of the error status register.

25   Force Address Parity Prediction Error

This condition forces the RMA parity predictor to predict that the parity bit will change state on every reference to CM. An error condition sets bit 62 of the error status register. This error inhibits CM requests and remains present until the channel is master cleared or a clear T register function is sent to the channel.

26   Force Byte Count Equal to Zero on JY Board

This condition forces the CM request counter to report an error after 16 CM requests have been issued. This condition is used to stimulate the constant CM request checker. This error sets bit 62 of the error status register and inhibit CM requests until a master clear or clear T register function is sent to the channel.

27   Not used

Operational Status Register (Read Only)

The operational status register is a read-only register which provides information regarding the state of the ISI adapter and channel. This status is used by the PP to control the operation of the adapter. The register can be read with the 2000_8 function.

Bit Descriptions

Operational status register bit assignments and descriptions are as follows.

Bit	Description
48	(Set to 0)
49	Input Buffer Full
50	Pause
51	Sync In
52	Sync Out
53	Command Sequence
54	Select Active
55	Select Hold
56	(Set to 0)
57	Echo Mode
58	Output Mode
59	PP Mode
60	DMA Mode
61	Noninterlocked Mode
62	T' Register Empty
63	Transfer in Progress

Input Buffer Full (Bit 49)

This bit, when set, indicates that the input buffer contains 16 channel words. If a Sync In is received with the input buffer full, an overflow occurs and data is lost. Under normal operating conditions, the adapter's outstanding request counter prevents more than 16 requests from being issued, thus preventing the possibility of an overflow.

Pause (Bit 50)

This bit reflects the current state of the ISI channel Pause signal line. It changes state as the Pause signal line goes active and inactive and follows the line directly.

Sync In (Bit 51)

This bit reflects the current state of the ISI channel Sync In signal and changes state as the Sync In line goes active and inactive and follows the line directly.

Sync Out (Bit 52)

This bit reflects the current state of the ISI channel Sync Out signal line. This bit follows the adapter's Sync Out flip-flop and not the actual channel line.

Command Sequence (Bit 53)

This bit reflects the current state of the ISI channel Command Sequence line. This bit follows the adapter's command sequence flip-flop and not the actual channel line.

Select Active (Bit 54)

This bit reflects the current state of the ISI channel Select Active signal and directly follows this signal.

Select Hold (Bit 55)

This bit reflects the current state of the ISI channel select hold signal. This bit follows the adapter's select hold flip-flop and not the actual channel line.

Echo Mode (Bit 57)

This bit, when set, indicates that the adapter is in echo mode. This mode is used during testing the adapter's receivers and transmitters. This bit is set by a $0D01_{16}$ function and remains set until one of the following functions is sent.

- Clear Echo Mode

- Master Clear Adapter

- Clear Command Sequence

- Clear Select Hold

Output Mode (Bit 58)

This bit, when set, indicates that the adapter is in output mode. In output mode the data flow is from either PP or CM to the external device. The following conditions set this bit.

- Master Clear Adapter function

- Set Command Sequence function

- Set Echo Mode function

- Echo mode set and PP is in output mode

The following conditions clear the Output Mode bit.

- Force Sync Out function and the Sync Out signal is active

- ISI function word has bit 48 clear (reset) and a Sync In signal is active

- Echo mode and PP activated

PP Mode (Bit 59)

This bit, when set, indicates that the adapter is in PP mode. In PP mode the adapter's input and output buffers are connected to the channel data registers and all data flow is between a PP and an external device. Only the following conditions set this bit. All other functions clear the bit.

● Set Command Sequence function

● Set PP Mode function and Transfer in Progress clear

● Set Echo Mode function

● Force Sync Out function

DMA Mode (Bit 60)

This bit, when set, indicates that the adapter is in DMA mode. In DMA mode the adapter's input and output buffers are connected to the ADU buffers, and a DMA transfer occurs whenever the T register byte count is not zero. The bit sets when the Set DMA Mode function is performed and the Sync In signal has been received from the ISI function word. The following conditions clear the bit.

● Set PP Mode function and Transfer in Progress clear

● Clear DMA Mode function

● Master Clear Adapter function or power-on master clear

● Clear Command Sequence function

● Clear Select Hold function

● Set Command Sequence function

Noninterlocked Mode (Bit 61)

This bit indicates the mode of communication between the adapter and a bus slave. When set it indicates noninterlocked mode and when clear indicates interlocked mode. Noninterlocked mode is supported only if the adapter is in the DMA mode. Noninterlocked mode sets if the ISI function word bit 54 is set and the bus slave has returned the Sync In. The following conditions clear the bit.

● Set Command Sequence function

● Master Clear Adapter function or power-on master clear

● Set PP Mode function and Transfer in Progress clear

T-Prime (T') Register Empty (Bit 62)

This bit, when set, indicates that the T' register is empty and can be written into with a write T registers function 0601_{16}.

Transfer in Progress (Bit 63)

This bit, when set, indicates that the adapter has issued a request (CM or ISI) and has not received a response. In DMA mode this bit is set during the entire transfer provided the T register is nonzero and all outstanding requests have not been honored.

Error Status Register (Read Only)

The error status register is a read-only register which provides information in the event of a hardware detectable error. This information is used by the PP for isolation of a fault and in determining the validity of a transfer. All error bits are ORed into the channel error flag and also sent to the CIO fault status 2 register. The error status register is cleared when read by a read error status function or a master clear function.

NOTE

The read error status function clears the error status register but does not clear the channel error flag. The flag is cleared by the appropriate PP instruction.

Bit Descriptions

Error Status Register bit assignments and descriptions are as follows.

Bit	Description
48 and 49	(Set to 0)
50	Uncorrected CM Error
51	CM Reject
52	Invalid CM Response
53	Response Code Parity Error
54	CMI Read Data Parity Error
55	Test Mode Compare Error
56	Overflow Error
57	ISI Input Error
58	ISI Timeout
59	JY Data Error
60	BAS Parity Error
61	JZ Error
62	JY Error
63	JX Error

Uncorrected CM Error (Bit 50)

This bit, when set, indicates that an uncorrected error response was received from CM on a write or read request.

CM Reject (Bit 51)

This bit, when set, indicates that a reject response was received from CM.

Invalid Response Code (Bit 52)

This bit, when set, indicates that the response code received from CM decoded into an illegal value.

Response Code Parity Error (Bit 53)

This bit, when set, indicates that a parity error was detected on the CM response code. If the JX error bit is set, the parity error (PE) was detected on the 1JXH pak. If not set, the error was detected in the CMI logic.

CMI Read Data Parity Error (Bit 54)

This bit, when set, indicates that the CMI logic has detected a read data parity error on a DMA transfer from this adapter.

Test Mode Error (Bit 55)

This bit, when set, indicates that the output data did not compare to the data generated by the operand generator. This check is done only during an output (DMA or PP) and when test mode is enabled.

Overflow Error (Bit 56)

This bit, when set, indicates that a Sync In was received after the input buffer was full. The first word written into the input buffer is overwritten with the last word received.

ISI Input Error (Bit 57)

This bit, when set, indicates that a parity error was received on the input data. This input data could be from a bus slave (normal I/O transfer), the operand generator (test mode), or the adapter output data (echo mode).

ISI Timeout (Bit 58)

This bit, when set, indicates that a channel timeout has occurred. A channel timeout occurs when the adapter sends a Sync Out and does not receive a Sync In within 1 s.

JY Data Error (Bit 59)

This bit, when set, indicates that the JY board has detected a data parity error. When this bit is set, bit 62 of the error status register should also be set.

BAS Parity Error (Bit 60)

This bit, when set, indicates that the JX has detected a parity error on the data received from a PP. This data is either an output word or a function word. When this bit is set, bit 63 of the error status register should also be set.

JZ Error (Bit 61)

This bit, when set, indicates that the JZ board has detected an error. The following error conditions set this bit.

● ISI input data error

● Input buffer data error

● Overflow error

● Output buffer data error

JY Error (Bit 62)

This bit, when set, indicates that the JY board has detected an error. When bit 59 is also set, this indicates a data error and the transfer will continue. When bit 59 is clear, the error is either a byte count parity error, RMA parity error, or a constant CM request error. These conditions inhibit CM requests until the channel is master cleared or a Clear T Register function is performed on the channel.

JX Error (Bit 63)

This bit, when set, indicates that the JX board has detected an error. The following conditions set this error.

● Control register parity error

● Mask register parity error

● T register byte count parity error

● Function decode PROM parity error

● Channel data register parity error

● JN response code parity error

Flag Mask Register

This register is used to enable the selective setting of the channel flag on certain defined conditions. This capability allows the PP to quickly test for these conditions using the jump-if-channel-flag-set or clear instructions. If all bits in this register are clear, then the channel flag acts the same as the channel flag on the NIO channels. The register is cleared by:

● Writing zeros with the Write Mask Register function

● Master Clear Adapter function

● Power-on Master Clear

Bit Descriptions

The register bit assignments and their descriptions are as follows.

Bit	Description
48 – 53	Not used
54	Transfer Complete
55	T' Register Empty
56 – 63	ISI Attention Bits 0 through 7

Transfer Complete (Bit 54)

When this bit is set, the channel flag sets when Transfer in Progress clears. This indicates the completion of a DMA transfer.

T-Prime (T') Register Empty (Bit 55)

When this bit is set, the channel flag sets when the T' register empties. This indicates that the current transfer has completed and the T' register transferred its contents to the T register. The PP can then set up for the next transfer.

ISI Attention Bits 0-7 (Bits 56-63)

If these conditions are enabled, the channel flag sets when the adapter senses an active attention line. This test is enabled only when the ISI channel is in idle mode.

DMA Channel Registers

Each ISI DMA adapter contains two transfer registers, the T register and the T'
register which hold control information for a DMA I/O transfer operation. The registers
are treated as a pair, with the T' register acting as a temporary holding register for
the T register. Data can only be written into the T' register, and can only be read
from the T register.

T Register

The T register is comprised of a 16-bit byte count register and a 32-bit real-memory-
address (RMA) register. The most significant bit and the three least significant bits of
the RMA are not used and are always read as being clear. The T register must be
loaded by the PP before any DMA operation may proceed. The byte count register is
decremented by two for every channel word as it is written into or read from the ADU
buffer. The RMA register is incremented by eight after every CM reference.

The T register is 48 bits long, and may be read at any time by issuing a read-T-
registers function, activating the channel, inputing three PP words, and then
deactivating the channel. The T register can be loaded only through the T' register.

T-Prime (T') Register

The T' register is comprised of a 16-bit byte count register and a 32-bit RMA register.
These registers may be written whenever the T' empty status bit is set. The T' register
is 48 bits long, and may be written by issuing a write-T-registers function, activating
the channel, inputing three PP words, and then deactivating the channel.

The contents of the T' register transfer to the T register when the T' register is full
and:

● DMA Transfer Not in Progress, T register byte count equal to zero, and ADU
  buffers are empty. This transfers all 48 bits to the T register and sets the T'
  register empty status bit.

● DMA Transfer in Progress and T register byte count equal to zero. This transfers
  only the 16-bit byte count. During output operations, this also sets the T' register
  empty status bit.

● DMA Transfer in Progress and the last CM reference of transfer transfers the
  32-bit RMA. During input operations, this also sets the T' register empty status bit.

ISI Programming Examples

The following coding sequences show the level of coding effort needed to control the CYBER ISI adapter. All programs are written in the following format.

Mnemonic	Hex Coding	Comments
(ex.: FNC)	(ex.: 0901,c)	(ex.: Set select hold and command sequence)

The Hex Coding may include the following indicators.

c = channel number
m = PPM address for data transfer
wc = word count

Refer to the Instruction Descriptions section of this manual for further information on programming.

Broadcast Master Reset

FNC	0901,c	Set select hold and command sequence
ACN	c	
LDC	0040	Broadcast master reset select word
OAN	c	Output select word
DCN	c	
FNC	0900,c	Clear command sequence

An I/O timeout occurs if the clear-command-sequence function is not issued within one second after outputing the broadcast master reset select word.

Select Bus Slave with BSR

FNC	0901,c	Set select hold and command sequence
ACN	c	
LDC		Bus unit select word
OAN	c	Output select word
DCN	c	
FNC	0A00,c	Force Sync Out
ACN	c	
IAN	c	Input BSR
DCN	c	
FNC	0900,c	Clear command sequence

Output PP Data

(Assume bus slave previously selected and is not busy.)

FNC	0901,c	Set command sequence
ACN	c	
LDC		Transaction function word
OAN	c	
LDC	wc	Data block word count
OAM	m,c	Output data block from PP memory
DCN	c	
FNC	0900,c	Clear command sequence

Input ISI Data or Status

(Assume bus slave previously selected and is not busy.)

```
        FNC     0901,c      Set command sequence
        ACN     c
        LDC                 Transaction function word
        OAN     c
        FJM                 Delay
        LDC     wc          Load word count
        IAM     m,c         Input data block to PP memory
        DCN     c
        FNC     0900,c      Clear command sequence
```

DMA Output with Control Word

(Assume bus slave is selected and is not busy.)

```
        FNC     0601,c      Write T' register
        LDN     3
        ACN     c
        OAM     m,c         Output length/address pair of first CM page to T'
        DCN     c
        FNC     0901,c      Set command sequence
        ACN     c
        LDC                 Transaction function code
        OAN     c
        LDN     2
        OAM     m,c         Output disk control words from PPM
        DCN     c
        FNC     0C01,c      Enable DMA mode
        FNC     0601,c      Write T' register
        LDN     3
        ACN     c
        OAM     m,c         Output length/address pair of second CM page to T'
Loop    DCN     c
        FNC     0400        Read operational status
        ACN     c
        IAN     c           Input operational status
        LPN                 Mask off all bits except Transfer in Progress
        NJN     Loop        Jump if transfer is not complete
        DCN     c
        FNC     0300        Read error status
        ACN     c
        IAN     c           Input error status
                            Test for error
        DCN     c
        FNC     0900,c      Clear command sequence
```

Built-in Test Facilities

The ISI channel adapter provides two different methods for testing the adapter hardware and interfaces. These features complement each other and enable the PP to test all the data paths and almost all of the control logic. The features are echo mode and test mode.

Echo Mode

Echo mode tests the PP-to-channel interface, the ISI channel receivers and transmitters, the ISI Attention bits to the mask network, and the input buffer overflow detection network. The following sequence of events should be followed in order to ensure valid results.

Checking Receivers and Transmitters

1. Select port to be tested.

2. Deselect any connected bus slave.

3. Set echo mode (function $0D01_{16}$). This function:
   a. Activates select hold
   b. Activates command sequence
   c. Inhibits Sync Out transmitter
   The ISI channel is now in transaction mode with no bus slaves selected.

4. Activate channel.

5. Output up to 16 data words.

6. Deactivate channel.

7. Activate channel.

8. Input data words (same quantity as output in step 5).

9. Compare input and output data.

10. Check for adapter errors.

11. Loop on steps 5 through 10 until all test patterns have been completed.

12. Deactivate channel.

13. Clear echo mode (function 6400_8). This function:
    a. Deactivates select hold
    b. Deactivates command sequence

Checking Flag Mask Network

1. Select port to be tested.

2. Deselect any connected bus slave.

3. Set enable idle test (control register bit 55).

4. Set echo mode.

5. Activate the channel.

6. Output test word. This word is translated as an ISI idle word.

7. Deactivate channel.

8. Verify the channel flag for the correct state.

9. Clear enable idle test.

10. Activate the channel.

11. Input the test word.

12. Compare output test word with input test word.

13. Check adapter for errors.

14. Loop on steps 6 through 12 until all test patterns have been completed.

15. Clear echo mode.

Forcing Input Buffer Overflow

1. Select port to be tested.

2. Deselect any connected bus slave.

3. Set inhibit oustanding request counter (control register bit 54).

4. Set echo mode.

5. Activate channel.

6. Output 17 words. Word 17 generates an overflow.

7. Deactivate channel.

8. Activate channel.

9. Input 16 words. The first word output in step 6 is lost. The input data consists of words 2 through 17.

10. Check adapter for overflow error.

11. Clear echo mode.

Test Mode

Test mode tests all the adapter internal data paths, the paths to and from CM, the T and T' registers, and all of the control logic associated with DMA transfers. In test mode, the adapter is exercised at the maximum transfer rates. This allows testing the CMI logic under worst traffic conditions.

Special hardware has been added to the adapter to provide these features. This hardware includes an operand generator (test seed) and a compare network. The operand generator is an 8-bit incrementer that can be written by a 0701_{16} function and read by a 0700_{16} function. The 8-bit output of the generator is replicated to form a 16-bit operand, making the upper byte a duplicate of the lower byte. The operand generator increments by one for every I/O transfer.

If test mode is enabled, and the adapter is conditioned for a PP input or DMA write, the operand generator supplies the input data. The generator continues to supply data until the DMA transfer completes or the PP terminates the input.

If test mode is enabled and the adapter is conditioned for a PP output or a DMA read, the operand generator supplies an operand to use for comparison with the output data. If the data doesn't compare equally, a bit is set in the error status register.

Testing DMA Paths

The following sequence is a hypothetical use of the test mode to test the DMA logic. In this example, the data generated during the write cycle is used as the read data. Therefore, if the operand generator is initiated with the same value at the start of the write and read cycles, the read data should compare with the operands generated during the read cycle.

Write Cycle –

1. Write test seed (function 0701_{16}).

2. Enable test mode (set control register bit).

3. Select operand generator.
   a. Set select hold (function 0801_{16})
   b. Set command sequence (function 0901_{16})
   c. Output select word

4. If BSR is requested, the operand generator provides the first operand as the BSR and increments to the second operand.

5. Clear command sequence (function 0900_{16}). (Exit select sequence.)

6. Set command sequence (function 0901_{16}). (Start transmission mode.)

7. Output transaction function word (bit 48 must be zero indicating an I/O input).

8. Write T' register (function 0601_{16}), starting address and byte count of DMA write.

9. Enable DMA mode. The operand generator supplies the data for the transfer.

10. At the end of the transfer, clear command sequence (end of write cycle).

Read Cycle –

1. Write test seed with original value written in write cycle step 1.

2. Write T' register with same starting address and byte count used in step 8 of write cycle.

3. Set command sequence.

4. Output transaction function word (bit 48 must be set indicating an I/O output).

5. Enable DMA mode. The comparator compares the operand generator date to the read data and flags the errors in the error status register.

6. At the end of the transfer, clear command sequence.

7. Check for error status.

Testing Operand Generator

The operand generator can be tested by the following sequence.

1. Write test seed.

2. Enable test mode.

3. Select operand generator.
   a. Set select hold
   b. Set command sequence
   c. Output select word
   d. Operand generator returns the Select Active line

4. Clear command sequence. (Exit select sequence.)

5. Set command sequence.

6. Output transaction function code (bit 48 must be clear indicating an I/O input).

7. Loop waiting for channel to go empty.

8. PP inputs from the channel and compares the data with a preset result.

9. Clear command sequence.

Testing Comparator

The comparator can be checked using the following sequence.

1. Write test seed.

2. Enable test mode.

3. Select operand generator.
   a. Set select hold
   b. Set command sequence
   c. Output select word
   d. Operand generator returns the Select Active line

4. Clear command sequence. (Exit select sequence.)

5. Set command sequence.

6. Output transaction function code (bit 48 must be one indicating an I/O output).

7. Output preset data patterns that correspond to the expected operand. Use data patterns that compare to test for no error, or use patterns that do not compare to test for error detection.

8. Monitor the adapter error status to determine pass/fail.

9. Clear command sequence.

DMA-Enhanced CYBER 170 Channel Adapter

The DMA-Enhanced CYBER 170 Channel Adapter (170 adapter) can be installed in any of ten channel locations in the CIO cabinet. This option provides an interface between a CIO PP and a standard 170 channel. In addition to the standard 170 protocol, this adapter supports the fast-transfer capabilities of extended semiconductor memory (ESM) II low-speed port. A PP controls the adapter by issuing functions and sending data using standard PP I/O instructions.

The adapter enables the PP to transfer data between the external device and PP memory using standard I/O instructions. In addition, the adapter may transfer data between the external device and CM. This data flow is called direct-memory access (DMA). The PP operates concurrently with the adapter during DMA data transfers. This allows the PP more time to process I/O requests from the CPU. The primary purpose of the adapter is to allow DMA transfers between a buffer residing in scattered pages of CM and an external device.

An I/O request consists of a list of commands. This list contains control information for the external device and data transfer length/address pairs for the adapter. These length/address pairs define the real-memory address (word boundary) and length (bytes) in the CM buffer. Due to the physical sectoring of some external devices, the PP separates single length/address pairs into multiple pairs. This forces a transfer to end on a physical boundary. For disk I/O devices, the PP adds control words to the beginning of each sector.

Programming information for the 170 channel adapter is described in the following text.

Left column (text cut off at right edge)

Block Full Out

This bit preven ¦endi¦
during a DMA ¦ ¦
allows checking

Enable Overflo¦

When in test n¦
detection netwo¦

Disable Error I¦

This bit disabl¦
channel c error¦
error status reg¦
clear function, ¦

Enable Force E¦

This bit enable¦

Force Error Co¦

When bit 56 is¦
Octal codes 01,¦
the following p¦ ¦rap.

Octal	Desc
01	Inver
	This
	error
02	Inver
	With
	regis
	goes
	boar¦
	sets
04	Forc¦
	This
	actu¦
	regis
05	Inver
	This
	chec¦
	statu

Right column

Function Register and Decode PROMs

The adapter monitors the most significant bit of th¦
If bit 48 is a one, the function word is an adapter¦
of the function word go to the external device. The¦
word to determine the specified operation. Adapter¦
listing and described in the following paragraphs.

Code (Hex)	Function
8000	Master Clear
8200	Clear T
84xx	Start DMA Input
86xx	Start DMA Output
8800	Clear DMA Mode
8C00	Disable Test Mode
8E00	Enable Test Mode
9000	Read Control Register
9200	Write Control Register
9400	Read Error Status Register
9800	Read Operational Status Register
9C00	Read T
9E00	Write T-Prime (T')

NOTE

xx specifies PP word count.

Function codes 8Axx, 96xx, and 9Axx are not used¦
adapter remains active and full. The adapter respo¦
when a used function is received.

Master Clear (8000)

This function clears the control, error status, and ¦
inactive and empty with the channel and error fla¦
and a 1-μs master clear pulse goes to the external¦

Clear T (8200)

This function clears the T registers, DMA mode, a¦
registers. An Inactive Out signal is not sent to th¦
clears.

Operational Stat

This read-only registe
channel during DMA
This status allows th

Bit Descriptions

Operational status re

Bit	Descrij
48 – 51	(Set to
52	Output
53	Input B
54	Data A
55	Fast Tr
56	Externa
57	Test M
58	PP Wor
59	DMA O
60	DMA Ir
61	DMA H
62	T Regis
63	Transfe

Output Buffer Full (B

This bit indicates the

Input Buffer Full (Bit

This bit indicates the

Data Available to Ch

This bit indicates the
word.

Fast Transfer (Bit 55

This bit indicates the

External Clock Presen

This bit indicates the
This clock must be p

Test Mode (Bit 57)

This bit indicates the
paths and associated
8E00 function. The a
Mode function (8C00)
Master Clear.

DMA Halted (Bit 61)

This bit indicates the external device sent an inactivate sig
DMA transfer.

If this condition occurs during a DMA output transfer, the
device is terminated. If the transfer is incomplete (byte cou
issue a Clear T function (8200) or a Master Clear function
transfer.

If this condition occurs during a DMA input transfer, the d
external device is terminated. All data in the Input Resyncl
into CM. If the transfer is incomplete (byte count nonzero),
T function (8200) or a Master Clear function (8000) before s

T Register Empty (Bit 62)

This bit indicates the T-prime register is empty. The PP ca
T function (9E00) which writes the T' register.

Transfer in Progress (Bit 63)

This bit indicates the adapter issued a request to CM or th
not received a response. In DMA mode, this bit is set durin
provided the T' register is nonzero and all outstanding requ

Error Status Register

This read-only register monitors the adapter's error checking circuits to determine if any hardware errors have occurred. The PP uses this error status for fault isolation and to determine the validity of a transfer. All error bits are ORed to form the adapter's error flag.

The error status register and error flag are cleared differently in native 170 mode and enhanced mode. In native 170 mode with control register bit 55 clear, the error status register and error flag are cleared when the PP executes a test and clear channel c error flag set (SFM) or clear (CFM) instruction. This allows the error flag to operate the same as for non-DMA 170 channels. In the enhanced mode with Control Register bit 55 set, the error status register and error flag are not cleared by the SFM and CFM instructions. This allows the PP to use the SFM ad CFM instructions to determine if an error exists. Then the PP uses the read error status register function (9400) to determine the specific error and isolation information. In both modes, the error status register and error flag are cleared by:

● Read Error Status Register function (9400)

● Master Clear function (8000)

● Master Clear from IOU MAC

Bit Descriptions

Error status register bit assignments and descriptions are as follows.

Bit	Description
48 – 49	(Set to zero)
50	Uncorrectable CM Error
51	CM Reject
52	Invalid Response
53	Any Response Code Parity Error
54	CMI Read Data Parity Error
55	Clock Fault
56	Overflow Error
57	Input Data Error
58	12/16 Conversion Error
59	A/D Data Error
60	BAS Parity Error
61	KZ Error
62	JY Error
63	KX Error

Uncorrectable CM Error (Bit 50)

This bit indicates an uncorrectable error response was received from CMI.

CM Reject (Bit 51)

This bit indicates a reject response was received from CMI.

Invalid Response (Bit 52)

This bit indicates the response code received from CMI decoded into an illegal value.

Any Response Code Parity Error (Bit 53)

This bit indicates a parity error was detected on the CM response code. If KX error (bit 63) is set, the parity error was detected on the KX board; if not set, the error was detected in CMI.

CMI Read Data Parity Error (Bit 54)

This bit indicates the CMI detected a read data parity error during a DMA output transfer.

Clock Fault (Bit 55)

This bit indicates a change was detected in the external clock. If this occurs during a DMA transfer, the adapter must be master cleared.

Overflow Error (Bit 56)

This bit indicates data was received after the Input Resynchronize Buffer was full. The first word written into the buffer is overwritten by the last word received.

Input Data Error (Bit 57)

This bit indicates a parity error was detected on the input data. The data is from the external device (normal DMA input transfer) or the Adapter Output Register Rank I (test mode).

12/16 Conversion Error (Bit 58)

This bit indicates a parity error was detected in the conversion network. The KZ error (bit 61) also sets.

A/D Data Error (Bit 59)

This bit indicates a parity error was detected during assembly/disassembly of CM data. The JY error (bit 62) also sets.

BAS Parity Error (Bit 60)

This bit indicates a parity error was detected in the output or function word received from the PP. The KX error (bit 63) also sets.

KZ Error (Bit 61)

This bit indicates the KZ board detected one of the following errors.

● Input data parity error

● Output data parity error

● 12/16 Conversion Shifter error

● Input Resynchronize Buffer overflow error

JY Error (Bit 62)

This bit indicates the JY board detected an error. If bit 59 is also set, the error was during assembly/disassembly and the transfer continues. If bit 59 is not set, one of the following errors occurred.

● RMA parity predictor error

● T Register byte count parity error

● Constant CM request error

These errors inhibit CM requests until the PP issues a clear T function or the adapter is master cleared.

KX Error (Bit 63)

This body indicates the KX board detected one of the following errors.

● Control Register parity error

● Adapter Output Register Rank I parity error

● Adapter Input Register Rank II parity error

● Function Decode PROM parity error

● KX response code parity error

MAC Interface

Any PP can read certain registers in the adapter by doing a maintenance register read of the appropriate channel status register. For example, if the adapter is installed in CIO channel location 0, the PP does a maintenance register read on register B0. If installed in location 1, the read is on register B1. This continues through channel location 11 (octal) with the read on register B9. The format for the adapter's channel status is in the following listing.

Byte	Description
0	Error Status Register Bits 50 through 55

Bit	Description
0	Zero
1	Zero
2	Uncorrected CM Error
3	CM Reject
4	Invalid Response
5	Any Response Code Parity Error
6	CMI Read Data Parity Error
7	Clock Fault

Byte	Description
1	Error Status Register Bits 56 through 63

Bit	Description
8	Overflow Error
9	Input Data Error
10	12/16 Conversion Error
11	A/D Data Error
12	BAS Parity Error
13	KZ Board Error
14	JY Board Error
15	KX Board Error

Byte	Description
2	PP Word Counter Bits 56 through 63 (Bits 16 through 23)
3	Operational Status Register Bits 52 through 55

Bit	Description
24	Zero
25	Zero
26	Zero
27	Zero
28	Output Buffer Full
29	Input Buffer Full
30	Data Available to Channel
31	Fast Transfer

Byte	Description
4	Operational Status Register Bits 56 through 63

Bit	Description
32	External Clock Present
33	Test Mode
34	PP Word Count = 0
35	DMA Output
36	DMA Input
37	DMA Halted
38	T Register Empty
39	Transfer in Progress

Byte	Description
5	Channel Flags

Bit	Description
40	Zero
41	Zero
42	Full I
43	Full II
44	Active
45	Full I or Full II
46	Error Flag
47	Channel Flag

Byte	Description
6	Control Register Bits 48 through 55

Bit	Description
48	Enable Cache Invalidate
49	Zero
50	60 through Bit Mode
51	Enable Test Clock
52	Disable External Clock
53	Block Full Out
54	Enable Overflow
55	Disable Error Register Clear

Byte	Description
7	Control Register Bits 56 through 63

Bit	Description
56	Enable Force Error Codes
57	Zero
58	Zero
59	Force Error Code Bit 0
60	Force Error Code Bit 1
61	Force Error Code Bit 2
62	Force Error Code Bit 3
63	Force Error Code Bit 4

DMA Transfers

The PP specifies a DMA transfer by sending the starting CM address and the byte count to the adapter. All DMA transfers start on a CM address and end on a 16-bit CM word parcel. If the transfer does not end on a CM word boundary, the word is filled with ones.

The adapter holds the address and byte count in the T and T' registers. DMA transfers execute from the T register. T' is a shadow register which maintains the I/O transfer rate across discontinuous pages in CM. If the byte count in T decrements to zero and T' is empty, the DMA transfer is suspended until a new T value is available or the PP issues a clear DMA mode function.

The PP initiates a DMA transfer by issuing a start DMA input or output function and ends the transfer with a clear DMA mode function. Before starting a DMA transfer, the PP prepares the external device. For example, the PP issues a read function to the external device before issuing a start DMA input function to the adapter. After receiving the start DMA function, the adapter sends an activate out signal to the external device to indicate the start of a transfer. After receiving the clear DMA mode function, the adapter sends an inactivate out signal to the external device to indicate the end of a transfer.

By specifying the number of words to be transferred in the PP word count field of a start DMA function, the PP writes or reads data on the front end of a transfer. Each word transferred decrements the PP word count. When the count reaches zero, the DMA transfer starts. Uses of this feature include supplying header information for disk systems and addresses for ESM-II.

Programming Examples

The following coding sequences are examples to show the level of coding effort needed to control the 170 adapter.

Mnemonic	Hex Coding	Comments
(ex.: FNC)	(ex.: 0901,c)	(ex.: Set select hold and command sequence)

The Hex Coding may include the following indicators.

c = channel number
m = PPM address for data transfer
wc = word count

Refer to the Instruction Descriptions section of this manual for further information on programming.

PP to/from an External Device

These sequences are the same as the NIO 170 channel with the exception that the adapter should be master cleared before starting or after an error. This can be done with a master clear function or a master clear channel through the IOU MAC.

DMA Output Without PWC

```
        FNC     xxxx,c      External device write
        FNC     9E00,c      Write T' Register
        LDN     3
        ACN     c
        OAM     m,c         Output length/address pair of first CM page to T'
        DCN     c
        FNC     8600,c      Start DMA output
        FNC     9E00,c      Write T' Register
        LDN     3
        ACN     c
        OAM     m,c         Output length/address pair of second CM page to T'
Loop    DCN     c
        FNC     9800,c      Read operational status
        ACN     c
        IAN     c           Input operational status
        LPN                 Mask off all bits except Transfer in Progress
        NJN     Loop        Jump if not set
        DCN     c
        FNC     8800,c      Clear DMA mode
```

DMA Input With PWC

```
          FNC     xxxx,c      External device read
          FNC     9E00,c      Write T' Register
          LDN     3
          ACN     c
          OAM     m,c         Output length/address pair of first CM page to T'
          DCN     c
          FNC     8420,c      Start DMA input with PWC = 20₁₆
          ACN     c
          IAM     m,c         Input 20₁₆ words to PP memory
Loop      DCN     c
          FNC     9800,c      Read operational statusl
          ACN     c
          IAN     c           Input operational status
          LPN                 Mask off all bits except Transfer in Progress
          NJN     Loop        Jump if not set
          DCN     c
          FNC     8800,c      Clear DMA mode
```

Built-in Test Facilities

NOTE

The following procedures are intended for engineering services personnel only.

Test Mode

The following paragraphs describe data paths and typical programming sequences for the adapter in test mode.

PP Transfers

This test substitutes the PP for the external device during PP input and output transfers. Up to ten 12-bit words pass from the PP back to the PP through Adapter Output Register Rank I, Transmit Register, Deskew Register, Input Resynchronize Buffer, and Adapter Input Register Rank II. All ten words are held in the adapter before they return to the PP for compare testing. The eighth word in the Input Resynchronize Buffer is a duplicate of the Transmit Register contents. The programming sequence is:

1. Function adapter to set test mode.

2. Activate adapter.

3. Output up to ten 12-bit words.

4. Input data.

5. Check adapter for errors.

6. Compare data.

7. Repeat output/input until all patterns are checked.

8. Function adapter to clear test mode.

DMA Transfers

This test substitutes the PP for the external device during the DMA input and output transfers. For DMA output transfers, data passes from CM to the PP through the Disassembly Buffer, Conversion Network, Output Resynchronize Buffer, Transmit Register, Deskew Register, Input Resynchronize Buffer, and Adapter Input Register Rank II. If fast transfer is enabled, data backs up in the buffers because the PP is slower than the fast transfer. For DMA input transfers, data passes from the PP to CM through Adapter Output Register Rank I, Transmit Register, Deskew Register, Input Resynchronize Buffer, Conversion Network, and Assembly Buffer. If fast transfer is enabled (Function Register bit 55 set), the enable test clock bit (Control Register bit 51) must be set. If the PP Word Count is zero, the programming sequence is:

1. Function adapter to set test mode.

2. Function adapter to write T' register.

3. Activate adapter.

4. Output T values.

5. Deactivate adapter.

6. Function adapter to start DMA output or input.

7. Activate adapter.

8. Output or input test data.

9. Deactivate adapter.

10. Function adapter to clear DMA mode.

11. Function adapter to clear test mode.

12. Check error registers.

13. Check data.

If the PP word count is nonzero, the adapter goes into the outstanding DMA mode. Meanwhile, the PP simulates a PP input or output transfer of up to ten 12-bit words. When the PP transfer completes the PP word count decrements to zero. The adapter goes into DMA mode and starts the DMA transfer. The programming sequence is:

1. Function adapter to set test mode.

2. Function adapter to write T' register

3. Activate adapter.

4. Output T values.

5. Deactivate adapter.

6. Function adapter to start DMA output or input. PP word count field contains quantity of 12-bit words to be echoed back to PP before DMA transfer starts.

7. Activate adapter.

8. Output up to ten 12-bit words.

9. Input words from step 8.

10. Check error register for PP transfers errors.

11. Repeat steps 8 through 10 until PP word count is zero.

12. Output or input test data.

13. Deactivate adapter.

14. Function adapter to clear DMA mode.

15. Function adapter to clear test mode.

16. Check error register for DMA transfer errors.

17. Check data.

Intelligent Peripheral Interface (IPI) Channel Adapter

The IPI channel adapter (IPI adapter) is the interface between the CYBER 180 PP I/O bus and an IPI channel. The IPI adapter contains control and status registers and manages the data transfer between a CYBER PP and an IPI peripheral. The PP addresses the IPI adapter by its channel number. The IPI adapter is always the bus master.

The IPI adapter has the added capability of directly transferring data between CM and an IPI channel. This DMA data transfer is initiated by a PP but runs independently of, and concurrently with, PP instruction execution.

Programming information for the DMA-enhanced IPI adapter is described in the following paragraphs.

IPI Term Definitions

Bus Master

The IPI adapter that currently controls the IPI channel with the select out, master out, and Sync Out signal lines. It can select bus slaves and control their operation.

Bus Slave

The control module selected to respond to but not control the select out, master out, and Sync Out signal lines on the IPI channel. Only one bus slave can be selected by a bus master at any one time. The bus master communicates with this unit. A maximum of eight bus slaves are attached to one IPI port.

Port

The hardware that comprises one IPI-compatible interface. This consists primarily of drivers and receivers and does not include the main control logic. The IPI adapter contains two ports that are designated A and B. Only one port may be active at a time.

Bus Exchange

The bus control sequence (initiated by the bus master) and the ending status sequence (initiated by the bus slave) that are used to frame an actual or attempted information transfer. For every bus control sequence, there must be an ending status sequence.

Bus Control

The 8-bit byte (octet) placed on bus A by the bus master during the bus control sequence. It defines the bus configuration for the subsequent information transfer.

Information Transfer

The mode of the IPI data bus when the bus master activates the master out line. Either bus master or bus slave data, commands, or responses may be placed onto the bus, depending on the directon of the information transfer in progress. This mode allows two-way communication between a bus master and a bus slave.

Ending Status

The status octets provided by the bus slave immediately following an information transfer.

Idle

The state of the IPI when all the control signals are inactive. Abnormal entries to the idle state occur whenever the bus master and bus slave recognize an undefined state or state transition. The buses are released prior to entering the idle state except during the request interrupts and master reset sequences.

IPI Signal Definitions

The IPI channel consists of buses A and B. Each bus has nine bidirectional lines (eight data and one parity) and six unidirectional control signals. The bus master provides three of these control signals to the bus slave and the bus slave responds by generating three control signals for use by the bus master. Except for special conditions described in Data Streaming Mode, these control signals are in interlocked mode. For instance, a given bus master signal leads to a corresponding response by the bus slave, which in turn allows the bus master to continue with the next operation. The exchanged signals are interlocked to allow easy control of signal timing over longer cable lengths regardless of internal clock rates. This also allows use of bus slaves that operate at varying transfer rates. The IPI channel signals are shown in figure 2-40.
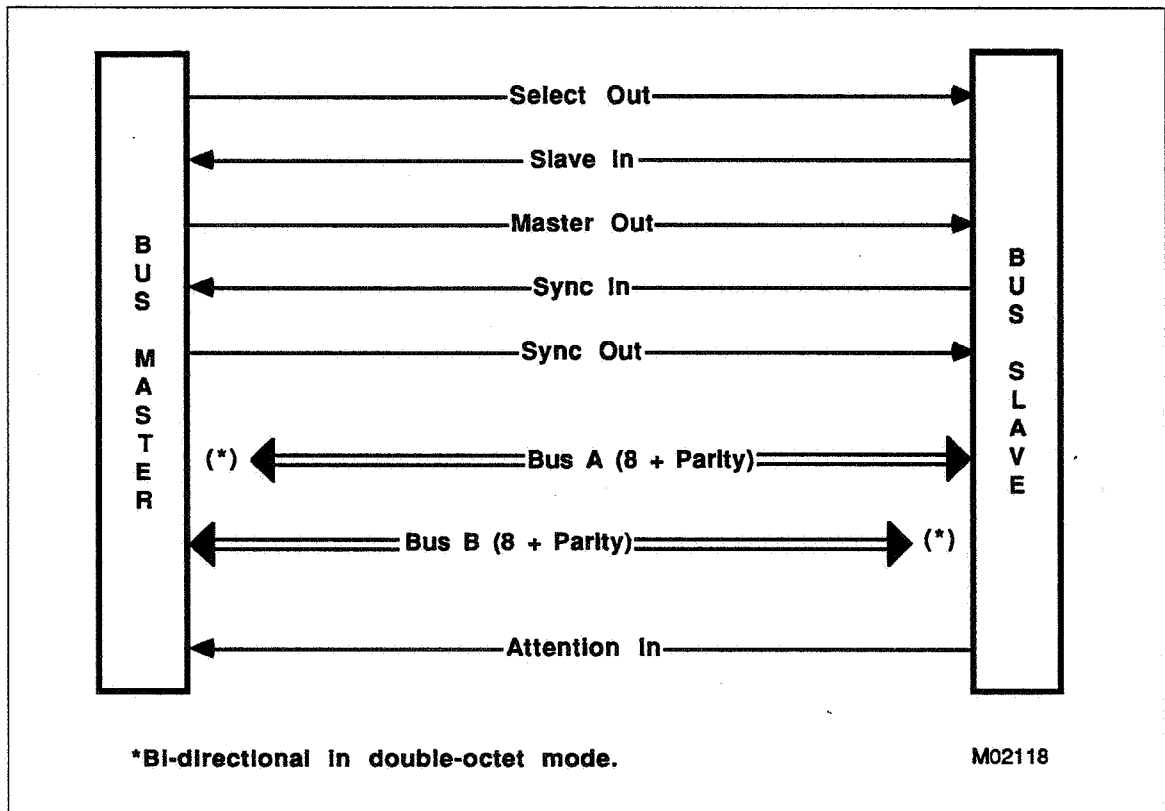
Figure 2-40. IPI Channel Signals

Bus A

This bus has nine lines consisting of data bits 0 through 7 and a parity bit. Bit 7 is the most significant bit. Parity is odd.

The bus master uses Bus A for all control sequences. In single-octet mode, all information passes from the bus master to the bus slave on Bus A. In double-octet mode, information passes either from the bus master to the bus slave or from the bus slave to the bus master on Bus A. Bus A is the first octet of double-octet information and bus B is the second octet. Bus A is released by all slaves when Select Out goes inactive.

Bus B

This bus has nine lines consisting of data bits 0 through 7 and a parity bit. Bit 7 is the most significant bit. Parity is odd.

The bus slave uses Bus B for all control sequences. In single-octet mode, all information passes from the bus slave to the bus master on Bus B. In double-octet mode, information passes either from the bus slave to the bus master or from the bus master to the bus slave on Bus B. Bus B is the second octet of double-octet information and bus A is the first octet.

Select Out

This signal is sent from the bus master to the bus slaves to select a bus slave and to maintain selection. When select out goes inactive, all bus slaves release Bus A.

Slave In

This signal is used by the bus slave to acknowledge control sequences initiated by the bus master, or to terminate information transfers.

Master Out

This signal is used by the bus master to initiate or terminate information transfers, request interrupts, request transfer mode, or reset bus slaves.

Attention In

This signal is a wired-OR gate for all bus slaves. It informs the bus master that service is requested. The bus master services the interrupts as required. This signal does not contribute to the state of the interface. It can be activated regardless of whether or not a bus slave is selected. It is driven active only by an unselected bus slave and cannot be driven inactive.

Sync Out

This signal is activated during transfers in (bus slave to bus master) to indicate that the bus master has accepted information. It is activated during transfers out (bus master to bus slave) to indicate that valid information is on the buses. The information out is present on the bus a minimum time before Sync Out is activated. Sync Out is activated to initiate the bus control sequence. During reset, Sync Out is activated without response for a minimum time.

Sync In

This signal is activated during transfers in to indicate that valid information is on the buses. It is activated during transfers out to indicate that the bus slave is ready to accept information. The information is placed on the buses a minimum time before Sync In is activated. Sync in is activated to acknowledge the bus control octet during the bus control sequence.

Data Transfer Modes

Data is transferred between the bus master and bus Slave In either the interlocked mode or data streaming mode.

Interlocked Mode

In this mode, the bus master sends Sync Out in response to a Sync In from the bus slave. Sync Out stays active until Sync In is deactivated.

Data Streaming Mode

This mode allows high transfer rates over long cable lengths. This is accomplished by not interlocking Sync In and Sync Out to eliminate a round-trip cable delay. Therefore, cable delay is not considered in determining transfer rate. Data streaming is used only during information transfers. All control and status sequences use interlocked mode. Normal IPI state sequences do not apply during data streaming since fully interlocked operation is not required.

The request transfer settings sequence returns transfer mode information to the bus master. Transfers on the bus slave may be interlocked only, data streaming only, or both interlocked and data streaming. In the latter case, the bus master selects the mode to be used during bus slave selection.

The transfer begins with the bus slave activating Sync In and then inactivating it to generate a pulse. The period between successive pulses is determined by the transfer rate. The bus master generates a complementary Sync Out pulse when it recognizes the Sync In pulse. The bus master must answer every Sync In pulse with a Sync Out pulse.

The bus master generates the Sync Out pulse by using a clock value to create a pulse which is greater than 40 percent of the cable configuration dependent (CDD) value. This is the value that a bus slave can recognize at the IPI which is equal to or faster than its own transfer rate.

The transmitter must ensure proper setup and hold times with respect to the active edge of its sync pulse. If the hold time is greater than the one-way cable delay plus setup times, the transfers appear interlocked.

A bus slave transferring from a buffer may be able to permit the bus master to stop and start the transfer stream.

If the bus slave has transmitted eight unanswered Sync In pulses, it waits a minimum of 25 ms for a complementary Sync Out pulse. If a Sync Out pulse is received, operations continue normally. If a Sync Out pulse is not received, the bus slave ends the transfer sequence.

After the first eight Sync In pulses are transmitted by the slave, all successive Sync In pulses are generated only upon receipt of a Sync Out pulse. This allows the bus master to dictate the period between Sync In pulses and throttle the speed at which the bus slave can transfer information. The number of Sync In pulses may differ from eight if vendor dependent or otherwise specified by the bus master.

The bus master must ensure that the Sync Out pulses sent to the bus slave exceed 40 percent of the CDD value.

Bus Slave Termination of Data Streaming

The bus slave first stops transmitting Sync In pulses to terminate a data streaming transfer. It then waits for an equal number of Sync Out pulses from the bus master or until a minimum of 25 ms has expired without receiving Sync Out pulses. It then terminates the transfer by inactivating Slave In and following the normal interlocked ending status sequence.

Bus Master Termination of Data Streaming

The bus master substitutes an inactive pulse on the master out line for the Sync Out pulse to terminate a data streaming transfer. This pulse has the same pulse width and period requirements as the Sync Out pulse. The bus master then continues to answer every Sync In pulse with complementary Sync Out pulses.

For transfers out, the bus master must not transmit information with the master out pulse or subsequent Sync Out pulses. For transfers in, the bus master must accept information with up to eight Sync In pulses following generation of the master out pulse. This allows all information transmitted by the bus slave, before its acceptance of the master out pulse, to be received by the bus master in order to maintain data integrity.

If bus master termination of data streaming is used when the bus master does not require a precise match between the number of octets transferred by the bus master and bus slave, the bus master does not need to accept information after generation of the master out pulse.

When the bus slave senses the master out pulse, it stops transmitting Sync In pulses and waits until it receives an equal number of Sync Out pulses including the master out pulse. For transfers out, the bus slave does not latch any information on the master out pulse and subsequent Sync Out pulses.

After the bus slave detects that the number of Sync In pulses equals the number of Sync Out pulses, including the master out pulse, or a minimum of 25 ms has expired without receiving Sync Out pulses, it inactivates Slave In, and follows the normal interlocked ending status sequence.

IPI Protocol Procedures

Protocol procedures for the bus slave and bus master are described in the following paragraphs.

Bus Slave Selection Operation

The bus slave select operation, which is always executed in interlocked mode, allows the bus master to select one of the bus slaves for further bus dialogs.

The parallel interface is designed to allow multiple bus units to be daisy-chained. Up to eight bus units can be connected to one IPI channel, each with its own unique address.

The bus master can only communicate directly with one bus unit at a time.

Bus unit selection is required only once per transfer sequence. Once connected, a bus master can transfer multiple information exchange blocks without repeating the bus unit select sequence.

Bus unit selection is necessary only after the bus master has terminated a transfer sequence and has deselected the previously selected bus slave by deactivating select out.

The selection sequence occurs as follows when the bus master addresses the bus slave:

1. The bus master places the select octet containing the slave address on Bus A. Then, it activates select out to enter the select state.

2. If the bus slave can process bus exchanges or information transfers, it places the bus slave's bit significant address (BSA) in the address octet on Bus B. Then, it activates Slave In to enter the slave acknowledge (SLAVACK) state.

3. If the bus slave cannot process bus exchanges or information transfers, but is otherwise functioning normally, it activates only Slave In to report the busy condition by entering the SLAVACK state. The busy condition reflects only the current condition of the bus slave.

4. If there is a parity error on Bus A, none of the bus slaves is selected and Slave In is not activated.

5. The bus slave remains selected as long as select out remains active. When select out is inactivated, the addressed bus slave is deselected.

Request Interrupts Sequence

This sequence allows the bus master to interrogate the bus slaves to determine the service or class of service desired.

The bus master initiates the sequence by setting the bus monitor on Bus A and activating master out. Bus slaves with interrupts meeting the request monitor conditions place their BSA in the address octet on Bus B and activate Slave In. Appropriate latching may be required at the bus master because interrupts from the bus slaves can dynamically change. Parity on Bus B is not checked by the bus master. The bus master inactivates master out to return to the idle state.

The response of the bus slaves is not synchronous, and the bus master must wait a time equal to that of the slowest bus slave to respond before latching or sampling Bus B. In addition, the bus master must wait for a time equal to that of the slowest slave to detect the idle state before it releases the BSA on Bus B and starts another sequence.

Request Transfer Settings Sequence

This sequence allows the bus master to interrogate the specified bus slave about its information transfer characteristics.

The bus master initiates the sequence by placing the request transfer settings octet on Bus A and activating master out. The addressed bus slave responds by setting the transfer settings response octet on Bus B and activating Slave In. The bus master then inactivates master out to reach the deselection state. Slave In is inactivated and a return is made to the idle state.

Bus Control Transfer

The bus master may make bus control transfers after a bus slave is selected. Bus control transfers must precede every information exchange operation and are always executed in interlocked mode.

The bus control transfer allows the bus master to establish the bus configuration for the subsequent information transfer.

The bus control transfer is initiated by the bus master after either select status (following bus slave selection) or slave status (following an information transfer) is accepted. The bus master sets the bus control octet on Bus A and activates Sync Out. The bus slave responds by setting the bus acknowledge octet on Bus B and activating Sync In. The bus master ends the sequence by inactivating Sync Out.

Ending Status Sequence

This sequence allows the bus slave or (optionally) the bus master and bus slave to present status, if any, of the previous information transfer.

The bus slave terminates the information transfer by releasing Bus B (transfers in, double-octet mode) and inactivating Slave In. The bus master responds by releasing Bus B (transfers out, double-octet mode), setting the master status octet on Bus A, and inactivating master out. The bus slave then sets the slave status octet on Bus B and re-activates Slave In to enter the slave acknowledge state.

Information Transfer

Information transfers are data or status transferred between the bus master and selected bus slave on the bus after the bus control word transfer. Multiple information exchanges may occur during the time the bus slave is selected.

Information transfers occur in interlocked mode and data streaming mode. Interlocked and streaming modes are methods used to handle Sync Out and Sync In signals and associated data bus information.

All selection words, bus control words, and status transfers must be handled in interlocked mode. Data transfers must be handled in either interlocked or data streaming mode.

For information transfers to the bus slave (write), the data bus contains the information and Sync Out informs the bus slave when to sample the bus. For information transfers to the bus master (read), the data bus contains the information and Sync In informs the bus master when to sample the bus. The signal exchange of Sync Out and Sync In continues until all information is transferred or an error condition is detected.

An information transfer can be terminated with or without deselection of the bus slave. If termination occurs without deselection of the bus slave, another bus control word may be sent to the bus slave specifying another information transfer without reselecting the bus slave. If termination occurs with deselection of the bus slave, select out is inactivated. Then, a new select operation must occur before another bus control word is sent to the bus slave specifying another information transfer.

Information Transfer Sequence

The following paragraphs give the sequence for the different types of information transfers. Each transfer has a bus control sequence, an information transfer sequence, and an ending status sequence.

The bus master initiates the end of an information transfer, and the bus slave terminates the transfer with the ending status sequence. The bus slave may also terminate the transfer without initiation from the bus master.

An information transfer sequence differs when the ending is initiated by the bus master or bus slave. The bus slave may terminate a sequence without transfer of information.

Operation Command Transfer

1. Condition bus for transfers out (bus control sequence).

2. Request transfer out.

3. Transfer out (information transfer sequence).

4. Bus master initiated termination, if any.

5. Ending status sequence.

Operation Response Transfer

1. Condition bus for transfers in (bus control sequence).

2. First transfer in (information transfer sequence).

3. Additional transfers in.

4. Ending status sequence.

Data Transfer Out

1. Condition bus for transfers out (bus control sequence).

2. Request transfer out.

3. Transfer out (information transfer sequence).

4. Bus master initiated termination, if any.

5. Ending status sequence.

Data Transfer In

1. Condition bus for transfers in (bus control sequence).

2. Request transfer in.

3. Transfer in (information transfer sequence).

4. Master initiated termination, if any.

5. Ending status sequence.

Master Reset Sequence

This sequence allows the bus master to initiate the maintenance mode. The bus master ensures that the select out and master out are inactive, and then activates Sync Out for a minimum of 10 µs.

Recognition of the maintenance (MAINT) state is independent of normal state processing logic. The bus slave must not enter maintenance mode until MAINT state is active for a minimum of 2 µs.

Maintenance Mode

This mode provides a communication path for error recovery and fault isolation when a failure exists in the IPI or attached bus slave.

The bus master uses the master reset sequence to force the IPI into maintenance mode and block normal use of the IPI.

Failure modes such as IPI signal lines continually stuck open or closed or a malfunctioning bus slave may prevent IPI bus communication necessary to collect fault isolation information or to implement real-time error recovery procedures.

The maintenance mode permits using the IPI for some basic functions during most predicted failure modes, except those that block the bus master from activating Sync Out and inactivating select out and master out when entering the MAINT state.

Selective Reset Sequence

This sequence allows the bus master to reset a single bus slave and terminate the maintenance mode.

The bus master initiates the sequence by placing the selective reset control octet on Bus A and activating master out. The bus master then allows the bus slave time to respond with Slave In, but disregards Slave In and Bus B contents if activated by the bus slave. The bus master then activates Sync Out for a minimum of 10 µs before inactivating it.

The bus slave neither initiates its reset action nor releases its IPI bus lines until the selective reset sequence is active for a minimum of 2 µs. The bus master then allows the bus slave time to respond with Slave In. Whether or not the bus slave responds, the bus master completes the sequence by inactivating master out and monitoring the IPI to ensure that the bus slave inactivates Slave In.

Function Codes

The function word from the PP is decoded by IPI adapter to determine the adapter's specified operation. The adapter responds with an inactive status only when a legal function is decoded. The following types of functions are used by the adapter.

• Functions for control of internal operations.

• Functions for control of IPI channel.

These functions are listed in the following paragraphs.

Internal Functions

The following internal functions are used by the IPI.

Code (Hex)	Function
0000	Master Clear
0004	Read Random Data Generator
0014	Write Random Data Generator
0022	Clear IPI Error
0x22	Force Error
xx42	Set Sync Period
0x62	External Clock/Port Select
0100	Clear Error
0200	Read Control Register
0300	Write Control Register
0400	Select PP from DMA
0500	Select PP to DMA
0600	Read Error Status Register
0700	Read Operational Status Register
0800	DMA Terminate
0900	Illegal Function
0A00	Read T Register
0B00	Write T Prime Register
0C00	Select IPI to DMA
0D00	Select DMA to IPI
0E00	Clear T Registers
xF00	Illegal Function

Channel Functions

The following channel functions are used by the IPI.

Code	Function
00A1	Master Terminate
00C1	Attention Present
00E1	Read IPI Status Register
00F1	Read IPI Error Register
0X81	Data Transfer Functions

Status

The status register bits are listed in the following paragraphs.

IPI Status Register

This read-only register provides information regarding the state of the IPI channel during DMA transfers. Bit assignments are:

Bit	Status
48	Error
49	Attention
50	Buffer Not Empty
51	Select Out
52	Slave In
53	Master Out
54	Sync In
55	Sync Out
56 through 63	Not used

DMA Operational Status Register

This read-only register provides information regarding the state of the adapter and IPI channel during DMA transfers. The PP reads this register using the 0700 function. This status allows the PP to control the operation of the adapter. Bit assignments are:

Bit	Description
48	Function Register Parity Error
49	Control Register Parity Error
50	Data Register Input Parity Error
51	Data Register Output Parity Error
52	T Register Counter Parity Error
53	Deadman Timer Counter Parity Error
54 and 55	Not used
56	DMA Transfer in Progress
57	IPI Transfer in Progress
58	Output Mode
59	Not used
60	DMA/IPI Mode
61	PP/IPI Mode
62	T Prime Register Empty
63	Transfer in Progress

DMA Error Status Register

This read-only register monitors the adapter's error checking circuits to determine if any hardware errors have occurred. The PP uses this error status for fault isolation and to determine the validity of a transfer. All error bits are ORed to form the adapter's error flag. This register is cleared by a master clear (0000) function or a clear error (0100) function. Bit assignments are:

Bit	Description
48	Not Used
49	Illegal Function or Sequence
50	Uncorrected CM Error
51	CM Reject
52	Invalid Response Code
53	CM Response Code Parity Error
54	CMI Read Data Parity Error
55	IPI Error
56	DMA Register Parity Error
57	MAC Status Parity Error
58	Timeout
59	JY Data Error
60	BAS Parity Error
61	LZ Error
62	JY Error
63	LX Error

IPI Programming Examples

The following coding sequences show the level of coding effort needed to control the IPI adapter. All programs are written in the following format.

Mnemonic	Hex Coding	Comments
(ex.: FNC)	(ex.: 0901,c)	(ex.: Set select hold and command sequence)

The Hex Coding may include the following indicators.

c = channel number
m = PPM address for data transfer
wc = word count

Refer to the Instruction Descriptions chapter of this manual for further information on programming.

Selective Reset

FNC	xx29,c	Set Select Out (xx = bus slave address)
ACN	c	
IAN	c	Input Status when Slave In is received
FCN	0008,c	Set Sync Out
FCN	0001,c	Clear Select Out and Sync Out

Select Bus Slave

FNC	xx29,c	Set Select Out (xx = bus slave address)
ACN	c	
IAN	c	Input Status when Slave In is received

Output PP Data to IPI

(Assume bus slave previously selected and not busy.)

FNC	0x81,c	Output IPI transfer function
ACN	c	
LDC	wc	Data block word count
OAM	m,c	Output data block from PP memory
DCN	c	

Input IPI Data to PP

(Assume bus slave previously selected and not busy.)

FNC	0x81,c	Output IPI transfer function
ACN	c	
LDC	wc	Load word count
IAM	m,c	Input data block to PP memory
DCN	c	

DMA Transfer

(Assume bus slave previously selected and not busy.)

```
        FNC     0x81,c      Output IPI Transfer Function
        FNC     0x00,c      DMA Output or Input Function (0C00=read,
                            0D00=write)
        LDN     3
        ACN     c
        OAM     m,c         Output length/address pair of first CM page to T'
        DCN     c
        FCN     0B00,c      Write T' Register
        LDN     3
        ACN     c
        OAM     m,c         Output length/address pair of second CM page to T'
Loop    DCN     c
        FNC     0700        Read Operational Status
        ACN     c
        IAN     c           Input Operational Status
        LPN                 Mask off all bits except Transfer in Progress
Loop    NJN                 Jump if transfer is not complete
        FNC     0600,c      Read Error Status
        ACN     c
        IAN     c           Input Error Status
                            Test for Error
```

Appendixes

Glossary   A

A

A Register
Address register.

AC
Address control.

Address
A sequence of bits, a character, or a group of characters that identifies a network station, user, or application.

ADU
Assembly/disassembly unit in the input/output unit.

ALN
Arithmetic/logical network.

Alphabetic character
One of the following letters: A through Z, a through z.

ALU
Arithmetic logic unit.

American Standard Code for Information Interchange (ASCII)
The standard code, using a coded character set consisting of 7-bit coded characters (8-bits including parity check), used for information interchange among data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters.

AOR
Address out-of-range.

ASCII
See American Standard Code for Information Interchange.

ASID
Active segment identifier.

AUX
Auxiliary.

B

B Register
Index register.

BAS
Barrel and slot in the input/output unit.

BC
Base constant.

BCD
Binary-coded decimal.

BDP
Business data processor.

Binary Synchronous Communications (BISYNC)
A communications term that means running in two directions at alternate times. A bisync line is one that allows a device to both send and receive data.

BISYNC
See Binary Synchronous Communications.

BMRS
Broadcast master reset.

BN
Byte number.

Board
See Logic Module.

BS
Binding section.

BSA
Bit significant address.

BSP
Binding section pointer.

BSR
Bit significant response.

BSS
Bus slave select.

C

Cache

A high-speed memory, duplicating a portion of central memory, used by the central processor to speed memory access.

Carrier

A continuous frequency capable of being modulated or impressed with a signal.

CB

Circuit breaker.

CBP

Code base pointer.

CCEL

Cache corrected error log.

CDC

Control Data Corporation.

CDD

Cable configuration dependent.

CE

Customer engineer.

CEJ

Central exchange jump.

CEL

Corrected error log.

CEM

Configuration environment monitor.

Central Memory (CM)

The main memory of the CYBER 960/962 within the central processing unit cabinet. CM stores between 64 and 256 megabytes of data on 4 to 16 memory modules. CM hardware includes memory modules, a dedicated logic cage, a memory interface module, and voltage regulator modules.

Central Memory Control (CMC)

The logic element in the central processing unit that controls the movement of data between central memory and the central processor. The central memory control circuits reside on two logic modules (CMCA and CMCB) located in the CP-0 logic cage, but accomplish memory control for both CP-0 and CP-1.

Central Processing Unit (CPU)

The main processing cabinet in the CYBER 960 Series mainframe, which includes the central processor (CP-0 and CP-1), central memory control, and central memory. The CPU cabinet has up to two central processors and between 64 and 256 megabytes of resident memory.

Central Processor (CP)

The functional processing logic within the CYBER 960/962 CPU. The first (or standard) central processor is referred to as CP-0. If a second central processor is provided with a system, it is referred to as CP-1. Each central processor resides in a separate logic cage and consists of eleven logic modules.

CF

Critical frame pointer.

CFF

Critical frame flag.

Chan

Channel.

Character

1. Any alphabetic, numeric, or special symbol that can be encoded. This term applies to the graphic characters for a input or output device and to the encoded control characters used by the terminal. Within Control Data hardware, a character is a coded byte of data, such as a 6-bit display code (NOS only) or 7-bit ASCII code. 2. (ISO) A member of a set of elements upon which agreement has been reached, and that is used for the organization, control, or representation of information. Characters may be letters, digits, punctuation marks, or other symbols. A character can be a graphic character or a control character.

CIO

See Concurrent Input/Output.

CIP

CYBER Initialization Package.

CLK

See Clock.

Clock (CLK)

1. (ISO) A device that generates periodic signals used for synchronization. 2. (ISO) Equipment that provides a time base used in a transmission system to control the timing of certain functions such as sampling and the duration of signal elements. See also Real Time.

CM

See Central Memory.

CMC

See Central Memory Control.

CMCA

See Central Memory Control.

CMCB

See Central Memory Control.

CMI

Control memory interface in the input/output unit.

CML

Concurrent maintenance library.

CML/VE

Concurrent maintenance library/virtual environment.

CMM

Central memory multiplexer in the input/output unit.

CMSE

Common maintenance software executive.

Coded Character Set

(ISO) A set of unambiguous rules that establish a character set and the one-to-one
relationships between the characters of the set and their coded representation.

Computer Room

A room that has a controlled environment that is maintained to meet the requirements
of the system equipment.

Concurrent Input/Output (CIO)

An input/output unit architecture that functions with NOS/VE.

CP

See Central Processor. See also Central Processing Unit.

CP-0

See Central Processor. See also Central Processing Unit.

CP-1

See Central Processor. See also Central Processing Unit.

CPU

See Central Processing Unit.

CRT

Cathode-ray tube.

CSF

Current stack frame.

CSSC

Customer services support center.

CST

Control store.

CST/MAC

Control store/maintenance access control.

CTI

Common test and initialization.

CYBER 960 Series (960/962) Computer Systems

Includes CYBER 960 and CYBER 962 computer systems. The CYBER 960 Series is Control Data's state-of-the-art middle- to high-range system, flanked by the 930 departmental computer on the low end and the 990 computer on the high end of the scalar performance spectrum. A full 960 system includes the mainframe (CPU, IOU, and power unit), MG set, system console, operating software, and a complement of peripherals.

CY170

CYBER 170.

C170

CYBER 170.

C180

CYBER 180 family of computers.

D

D/F

Data/function bit.

DC

1. Direct current. 2. Debug code.

DCD

Data carrier detector.

Deadstart

The process of initializing the system by loading the operating system library programs and any of the product set from magnetic tape or disk. Deadstart recovery is reinitialization after system failure.

DEC

Dependent environment control.

Demodulation

The process of retrieving an original data signal from a modulated carrier wave.

Device Interface (DI)

The communcations processor that Control Data offers as its CDCNET hardware product. Also called a CDCNET device interface.

DI

1. Debug index. 2. See Device Interface.

Digit

One of the following characters: 0 1 2 3 4 5 6 7 8 9.

DLP

Debug list pointer.

DM

Debug mask.

DMA

Direct-memory access.

DMR

Debug mask register.

Down

A status of suspended service.

DS

See Deadstart.

DSC

Display station controller.

DSP

Dynamic space pointer.

DSR

Data set ready.

DTR

Data terminal ready.

Dual IOU

An installation containing a primary (attached) IOU and an optional (standalone) IOU.

DUE

Dependent environment control.

DVS

Diagnostic virtual system.

E

Early Environmental Warning (EEW)

The power unit generates an EEW for temperature conditions that do not warrant an emergency shutdown. The EEW indicates that a temperature sensed is above normal, but not critically so.

EBCDIC

See Expanded Binary Coded Decimal Interchange Code.

EC

Environment control.

ECL
Emitter-coupled logic.

ECM
Extended central memory.

ECS
Extended core storage.

EEW
See Early Environmental Warning.

EI
Environmental interface.

EIA
Electronics Industries Association.

EID
Element identifier.

Ellipsis
Three (or more) consecutive periods at the end of a physical line to indicate the
continuation of a command.

EM
Error mode.

EMI
Electromagnetic interference. Synonomous with Radio Frequency Interference.

EPF
External procedure flag.

EQ
Equal.

ES
End suppression toggle (BDP edit instruction).

ESD
Electrostatic discharge.

ESM
Extended semiconductor memory.

ESM-II
Extended semiconductor memory II.

Expanded Binary Coded Decimal Interchange Code (EBCDIC)
The set of 256 characters, each presented by eight bits, that is used with the 3270
Binary Synchronous Communications protocol.

EXT

External.

F

FCC

Federal Communication Compliance.

FCO

Field change order.

FCTN

Function.

Federal Communication Compliance (FCC)

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 (insert peripheral computing device if appropriate) 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.

FF

See Flip-Flop.

Field-Replaceable Unit (FRU)

Equipment parts that are replaceable at the customer site are referred to as FRUs. Equipment manuals generally recommend a list of FRUs that should be stocked on site for that equipment. FRUs are identified with 8-digit part numbers that can be ordered from Control Data's World Distribution Center (WDC). See also World Distribution Center.

FIFO

See First-In/First-Out.

First-In/First-Out (FIFO)

1. This term applies to data processing services in which requests are serviced in the same order they are received. 2.(ANDIPS) A queuing technique in which the item that has been in the queue for the longest time is the next to be processed.

FL

Field length.

FLC

Central memory field length register.

FLE

Extended core storage field length register.

Flip-Flop

(ANDIPS) A circuit or device containing active elements, capable of assuming either one of two stable states at a given time. Synonymous with bistable circuit, toggle (l).

Floating-Point Base

(ISO) In a floating-point representation system, the implicit fixed positive integer base, greater than unity, that is raised to the power explicitly denoted by the exponent in the floating-point representation or represented by the characteristic in the floating-point representation and then multiplied by the fixed-point part to determine the real number represented, e.g., in the floating-point representation of the number 0.0001234, namely 0.1234-3, the implicit floating-point base is 10. Synonymous with floating-point radix.

FP

Floating-point. See Floating-Point Base.

FRC

Free-running counter.

Frequency Converter

A motor generator mounted within its control cabinet.

FRU

See Field-Replaceable Unit.

FS

Fault status.

FU

Functional unit.

G

G/L

Global/local.

Graphic Character

(ISO) A character other than a control character, that is normally represented by a graphic.

H

Hardware

1.(ISO) Physical equipment as opposed to programs, procedures, rules, and associated documentation. 2. Electronic circuits and its housing, including cabinet, power hook-up, and cooling system.

Hdr

Header.

Hertz (Hz)

A measure of frequency or band width the same as cycles per second.

HEX

See Hexadecimal.

Hexadecimal

(ISO) Synonym for sexadecimal. See also Sexadecimal.

HPA/VE

Hardware performance analyzer/virtual environment.

Hz

See Hertz.

I

I/F

See Interface.

I/O

Input/output.

IC

Integrated circuit.

ICC

Instruction completion control.

ICP

Instruction control pipe.

ID

1. Identification. 2. Identifier.

IDX

Index.

IF

Instruction fetch.

IF/IC

Instruction fetch/instruction control.

ILH

Instruction look-ahead.

Input/Output Unit (IOU)

IOU contains the peripheral processors and channels that enable operator interaction with, and peripherals access to the central processing unit. The IOU has either NOS and NOS/VE capability (CYBER 960) or is NOS/VE only (CYBER 962). The IOU has the interface port for the system console.

INSTR
Instruction.

Interface
1. A shared boundary. An interface might be a hardware component to link two devices or it might be a portion of storage or registers accessed by two or more computer programs. 2. See data transmission interface.

International Standards Organization (ISO)
A worldwide standards group similar in function to the American National Standards Institute (ANSI). ANSI is a member of the International Standards Organization.

IOU
See Input/Output Unit.

IOU Expansion
An add-on IOU product that physically attaches to the CYBER 960 IOU. The IOU expansion is a concurrent input/output (CIO) architecture and allows for addition of up to ten peripheral processors and ten channels. The IOU expansion is the first IOU option to be added to a CYBER 960 mainframe, increasing the number of IOU cabinets to two. (Does not include CYBER 960-xxS.)

IPI
See Intelligent Peripheral Interface.

ISI
See Intelligent Standard Interface.

ISO
See International Standards Organization.

J

JPS
Job process state.

K

K Register
Operation code register.

KEY
Key.

L

Large-Scale Integrated Circuit
A chip that contains around 100 to 1000 logical gates.

LDS
Literature Distribution Services.

LED
Light-emitting diode.

LM
See Local Memory.

LOC
Local.

Local Memory (LM)
The storage area with accompanying control that provides a high-speed interface between the central processor and central memory. LM is contained by two logic modules, LMA and LMB. See also Cache.

LOCK
Lock.

Logic Module
A printed-circuit board with microcircuit chips.

Long Warning (LW)
The power unit generates an LW when an emergency power shutdown is to occur in approximately 1 minute if the fault is in an IOU or in approximately 2 minutes if the fault is in the power unit, CPU cabinet, or with ambient room temperature (assuming the detected fault continues).

LPID
Last processor identification.

LRN
Largest ring number.

LSB
Least significant bit.

LSI
See Large-Scale Integrated Circuit.

LW
See Long Warning.

MA   MDW

M

MA

Monitor address.

MAC

Maintenance access control.

MAF

Maintenance action form.

Mainframe

Includes the central processing unit, input/output unit, and power unit portions of the mainframe complex. The mainframe does not include the motor-generator set or the system console.

Mainframe Complex

The hardware products that make up a central processing computer system are referred to collectively as the mainframe complex. Includes the central processing unit, input/output unit, power unit, system console, and the motor-generator set. Peripherals are not included in the mainframe complex.

MAINT

Maintenance.

MALET

Maintenance application language for equipment testing.

MCEL

Map corrected error log.

MCH

Maintenance channel.

MCR

Monitor condition register.

MCU

Maintenance control unit.

MD

Multiply/divide.

MDD

Monitor display driver.

MDF

Model-dependent flags.

MDW

Model-dependent word.

A-14 Virtual State Volume 2 Hardware Reference   60000133 B

MEJ

Monitor exchange jump.

Memory Interface Module

The logical assembly interfacing central memory and central memory control. The memory interface module is physically attached to the rear of the central memory logic cage and includes circuits for data, addressing, and control.

Memory Module

The logical assembly upon which data is stored in central memory. Four memory modules constitute one memory increment or 64 megabytes of data storage. The central processing unit's central memory contains a minimum of four memory modules and a maximum of 16.

Meter

A unit of measure in the metric system that is equal to 39.3 inches.

MF

Monitor flag.

MG

See Motor-Generator Set.

MMR

Monitor mask register.

Modem

(ISO) A functional unit that modulates and demodulates signals. One of the functions of a modem is to enable digital data to be transmitted over analog transmission facilities. Modem is a contraction of modulator-demodulator.

Modulation

A message signal that is impressed on a carrier signal and transmitted at another signal frequency.

Module Assembly

The circuit boards in the CP. There are 13 different modules in the CP.

MOP

Micro-operator (BDP edit instruction).

MOS

Metal-oxide-semiconductor.

Motor-Generator (MG) Interface Unit

The electrical box that interfaces with the motor-generator set. Also see Motor-Generator (MG) Set.

Motor-Generator (MG) Set

A motor generator and a separate motor-generator control cabinet that converts 3-phase site utility power to 3-phase, 400-Hz power suitable for the mainframe electrical requirements. Smaller MG sets may include the MG and its MG control functions within one cabinet and be referred to as a frequency converter.

MPS

Monitor process state pointer.

MR

Maintenance register.

MSB

Most significant bit.

MSG

Message.

MSL

Maintenance software library.

Multiplexer (MUX)

Equipment that enables a site to concentrate data transmission between multiple slower-speed devices (such as, terminals and workstations) and a higher-speed channel. For example a multiplexer can concentrate data being transmitted between multiple terminals and an host computer by using a local area network.

Must

A mandatory requirement.

MUX

See Multiplexer.

N

Network Operating System (NOS)

An operating system for the host computer. It has network capabilities for time-sharing and transmission processing in addition to local and remote batch processing. NOS controls the computation of programs submitted through remote terminals and maintains normal batch processing operations for jobs submitted locally.

Network Operating System/Virtual Environment (NOS/VE)

An operating system for the host computer. It has network capabilities for time-sharing and transmission processing in addition to local and remote batch processing. NOS/VE operates in Virtual State and controls the computation of programs submitted through remote terminals and maintains normal batch processing operations for jobs submitted locally.

NIO

See Nonconcurrent Input/Output.

Nonconcurrent Input/Output (NIO)

An input/output unit architecture that functions with NOS.

NOS

See Network Operating System.

NOS/VE
See Network Operating System/Virtual Environment.

NPA
Network performance analyzer.

NS
Negative sign toggle.

O

OCF
On-condition flag.

OI
Options installed.

ON
Occurrence number.

Opcode
Operation code.

Operand Issue (OPI)
That portion of the central processor responsible for storage and distribution of the process state register values while a process is executing. OPI is contained by logic modules OPIA, OPIB, and OP/SM.

OPI
See Operand Issue.

OPI/SM
Operand issue/segment map.

Optl
Optional.

OS
Operating system.

P

P/N
Part number.

P Register
Program address register.

PAR
Parity.

PCB

Printed-circuit board; often called logic board or module.

PDM

Processor Detected Malfunction. Any error detected by the central processor during instruction execution.

PE

Parity error.

PFA

Page frame address.

PFS

Processor fault status.

PID

Processor identifier.

PIT

Process interval timer.

PM

See Preventive Maintenance.

PMF

Performance monitoring flag.

PN

Page number.

PND

Process-not-damaged flag.

PO

Page offset.

PONR

Point of no return.

Port

The physical connection on the device interface through which data is transferred to/from the device interface. Each port is numbered (labeled) and supports a single communication line.

Power unit

Provides power to support the electrical systems (logic, environmental, and so on) of the CPU. (The IOU has its own power supply.) The power unit occupies a cabinet that attaches to the CPU.

PP

Peripheral processor.

PPM

Peripheral processor memory.

Preventive maintenance

1. (ISO) Maintenance performed specifically to prevent faults from occurring. 2. Contrast with corrective maintenance.

Primary IOU

The IOU cabinet(s) bolted to the CPU; the IOU and, if present, the IOU nonstandalone-expansion.

PROM

Programmable read-only memory.

PSA

Previous save area pointer.

PSF

Previous stack frame.

PSM

Page size mask.

PSWF

Page search without find.

PTA

Page table address.

PTE

Page table entry.

PTL

Page table length.

PTM

Processor test mode.

PVA

Process virtual address.

PWR

Power.

Q

Q Register

Operand address register.

R

RAC
Central memory reference address register.

RAE
Extended core storage reference address register.

RAM
Random-access memory.

RDS
Register/data select.

Real Time
An operation pertaining to the processing of data by a computer in connection with another process outside the computer according to time requirements imposed by the outside process. This term is also used to describe systems operating in conversational mode and processes that can be influenced by human intervention while they are in progress.

Real-time clock
See Real Time and Clock.

REM
Remote.

RESP
Response.

Resync
Resynchronize.

RF

RGTR
Register.

RI
Radial interface.

RMA
Real-memory address.

RN
Ring number.

ROM
Read-only memory.

RP
Read permission.

RS-232-C

An Electrical and Electronic Industries Association standard that describes the interface between terminals or other data terminal equipment and modems or other data communications equipment employing a serial binary interchange.

RTA

Remote technical assistance.

RTA Modem

(ISO) A functional unit used specifically by a person involved in providing remote technical assistance. See also modem.

RTC

See Real-Time and Clock.

RTS

Request to send.

S

SCD

System console driver.

SCR

Silicon-controlled rectifier.

SCSI

See Small Computer Standard Interface.

SCT

Special characters table.

SDE

Segment descriptor table entries.

SDT

Segment descriptor table.

SECDED

Single error correction/double error detection.

Secondary (Standalone) IOU

The standalone IOU is an option that provides the CYBER mainframe with a dual-IOU configuration. The standalone IOU is installed on an island separate from the mainframe, but is linked to the CYBER with cables.

SEG

Segment.

Sexadecimal

1. (ISO) Characterized by a selection, choice or condition that has sixteen possible different values or states. 2. (ISO) Of a fixed-radix numeration system, having a radix of sixteen. 3. Synonymous with hexadecimal.

SFSA

Stack frame save area.

Short Warning (SW)

The power unit generates on SW when an emergency power shutdown is to occur in 2.5 seconds. Also see Emergency Power Shutdown.

Should

A recommendation that is advised but not required.

SIB

See Side Interconnect Board.

Side Interconnect Board (SIB)

SIBs are the printed-circuit boards that interconnect the central processor logic modules. The multilayered SIBs actually constitute the sides of the CP logic cages. CP-0 logic modules are interconnected with SIB A and SIB B, while CP-1 logic modules are interconnected with SIB C and SIB D.

Single processor

See Central Processor.

SIT

System interval timer.

Site

The computer room and other building locations that may include one or more motor-generator sets and data media storage.

SLAVACK

Slave acknowledge.

SM

1. Segment map. 2. Symbol.

Small Computer Standard Interface (SCSI)

The system console interfaces with the input/output unit through small computer standard interface.

SMAQR

Standardized maintenance approach quick reference.

SN

Negative sign.

SPID

Segment page identifier.

SPM

System power monitor.

SPT

System page table.

SR

Select reset.

SRT

Subscript range table.

SS

Status summary.

STA

Segment table address.

Stack

An area in memory used as temporary storage for chaining calls during task or interrupt service routine execution. Task calls are chained on a user stack. Interrupt service routine calls are chained on a supervisor stack.

Stack Frame

The area within a stack that accommodates a single call.

Standalone IOU

The standalone IOU is an option that provides the CYBER 960/962 mainframe with a dual IOU configuration. The standalone IOU is installed on an island separate from the mainframe, but is linked to the CYBER 960/962 CPU with 50-ft cables. Up to ten peripheral processors (PPs) and ten channels reside in the standalone IOU cabinet.

Standalone IOU Expansion

A standalone IOU expansion product can be added to the standalone IOU, doubling the PP and channel capability of the standalone IOU configuration. The standalone IOU expansion and the IOU expansion differ only in their physical location relative to the mainframe.

STL

Segment table length.

SV

Specification value.

SVA

System virtual address.

SW

See Short Warning.

System Console

The keyboard and display screen used to monitor and control the operating system.

T

T"
T-prime register.

TE
Trap enable.

TED
Trap-enable delay.

TEF
Trap-enable flip-flop.

TER
Terminate.

TM
Test mode.

TOS
Top of stack.

TP
Trap pointer or test point.

TPM
Two-port multiplexer.

U

UART
See Universal Asynchronous Receiver/Transmitter.

UCR
User condition register.

UEL
Uncorrected error log.

UEM
Unified extended memory.

UMID
Untranslatable virtual machine identifier.

UMR
User mask register.

Universal Asynchronous Receiver/Transmitter (UART)
An LSI circuit for start/stop serial data transfer.

User Cond
User condition.

UTC
Utility channels.

UTP
Untranslatable pointer.

UVMID
Untranslatable virtual machine identifier.

V

V
1. Valid bit. 2. Also see Volt.

VC
Search control code.

VL
Segment validation.

VMCL
Virtual machine capability list.

VMID
Virtual machine identifier.

Volt (V)
A measure of electromotive force needed to move an electric charge. It is equivalent to the force required to produce a current of 1 ampere through a resistance of 1 ohm.

W

WDC
See World Distribution Center.

World Distribution Center (WDC)
Control Data's ordering and distribution center for spare hardware parts, software revision packages, and the documentation produced to support its product lines.

WP
Write access control (segment descriptor field).

WR
Write/read.

X

X Register
Operand register.

XP
Execute access control.

Z

ZF
Zero field toggle.

ZFI
Zero-fill inhibit.

Edit Examples B

This appendix contains edit examples for the BDP edit (ED) instruction.

NOTE

For examples in this appendix, the destination field is assumed to have the same length and decimal point position as the source field, except for the differences necessitated by insertion characters.

Edit Masks 1 through 25

These edit masks are used in the examples given in the following pages.

Mask Number	COBOL Picture	Edit Mask (Hexadecimal with insertion characters *, $, 0, /, b, C and R shown as alphanumerics)
1	$ZZ,ZZ9.99	08 96 72 C4 72 01 95 02
2	$ZZ,ZZZ.99	07 96 72 C4 73 95 02
3	$ZZ,ZZZ.ZZ	08 96 72 C4 73 95 02 FA
4	–ZZZZ9.99	06 B3 74 01 95 02
5	ZZZZ9.99+	07 74 01 95 02 52 98
6	ZZ.999,99	06 72 C5 03 94 02 (Decimal point is comma)
7	$$$$.99CR	0B 61 $ 73 80 95 02 62 C R B8
8	$$$,$$$.$$	0A 61 $ 72 C4 73 80 95 02 FA
9	$$$$99,99CR	0C 61 $ 73 80 02 94 02 62 C R B8
10	$$$,$$9.99	DA 61 $ 72 C4 72 80 01 95 02
11	$99.99	05 96 02 95 02
12	$**,**9.99	0A 96 D1 * 72 C4 72 01 95 02
13	$**,***.**BCR	11 96 D1 * 72 C4 73 95 02 63 b C R B8 F7 95 E5
14	$**,***.**	0C 96 D1 * 72 C4 73 95 02 F7 95 E2
15	**,***,**+	0D D1 * 72 C4 73 95 02 52 98 F6 95 E3
16	--99999,99	07 50 71 80 05 94 02
17	----.99	06 50 73 80 95 02
18	++++99	05 52 73 80 02
19	00999.00	09 42 0 0 03 95 42 0 0
20	99,999	05 02 C4 03 F6 (blank when zero)
21	XX/XX/XX	08 12 41 / 12 41 / 12
22	BBB99.99–	09 43 b b b 02 95 02 B3
23	999.00	06 03 95 42 0 0
24	999.BB	06 03 95 42 b b
25	9B9B9	06 01 91 01 91 01 or 08 01 41 b 01 41 b 01

Edit Examples Using Edit Masks 1 through 8

Example	Source Field	Mask Used	Destination Field
1	00000.00	1	$bbbbb0.00
2	00000.01	1	$bbbbb0.01
3	000000.10	1	$bbbbb0.10
4	00001.00	1	$bbbbb1.00
5	00010.00	1	$bbbb10.00
6	00100.00	1	$bbb100.00
7	01000.00	1	$b1,000.00
8	10000.00	1	$10,000.00
9	00000.00	2	$bbbbbb.00
10	00000.00	3	bbbbbbbbbb
11	00000.01	3	$bbbbbb.01
12	00001.00	3	$bbbbb1.00
13	10000.00	3	$10,000.00
14	–00000.00	4	–bbbb0.00
15	+00000.00	4	bbbbb0.00
16	–12345.67	5	12345.67–
17	+00012.34	5	bbb12.34+
18	00000.00	6	bbb000,00
19	01000.00	6	b1.000,00
20	–123.45	7	$123.45CR
21	–023.45	7	b$23.45CR
22	003.45	7	bb$3.45bb
23	000.45	7	bbb$.45bb
24	00000.00	8	bbbbbbbbbb
25	00000.01	8	bbbbbb$.01

Edit Examples Using Edit Masks 9 through 16

Example	Source Field	Mask Used	Destination Field
26	00001.00	8	bbbbbb$.10
27	00001.00	8	bbbbb$1.00
28	00010.00	8	bbbb$10.00
29	00100.00	8	bbb$100.00
30	01000.00	8	b$1,000.00
31	10000.00	8	$10,000.00
32	–0000000	9	bbb$00,00CR
33	0010000	9	bb$100,00bb
34	0100000	9	b$1000,00bb
35	–1000000	9	$10000,00CR
36	00000.00	10	bbbbb$0.00
37	10000.00	10	$10,000.00
38	00.00	11	$00.00
39	12.34	11	$12.34
40	00000.00	12	$*****0.00
41	00000.01	12	$*****0.01
42	00000.10	12	$*****0.10
43	00001.00	12	$*****1.00
44	00010.00	12	$****10.00
45	00100.00	12	$***100.00
46	01000.00	12	$*1,000.00
47	10000.00	12	$10.000.00
48	00000.00	13	*******.*****
49	–00000.01	13	$**,***.01bCR
50	00000.01	13	$**,***.01bbb
51	–00000.00	13	*******.*****

Example	Source Field	Mask Used	Destination Field
52	00000.00	14	*******.**
53	–00000.01	14	$******.01
54	00000.00	15	******.***
55	–00000.00	15	******.***
56	12345.67	15	12,345.67+
57	–12345.67	15	12,345.67–
58	–00000000	16	b–00000,00
59	–12345678	16	–123456,78
60	00000000	16	bb00000,00
61	12345678	16	b123456,78

Edit Examples Using Edit Masks 17 through 25

Example	Source Field	Mask Used	Destination Field
62	–000.00	17	bbb–.00
63	000.00	17	bbbb.00
64	–001.00	17	bb–1.00
65	010.00	17	bb10.00
66	–100.00	17	–100.00
67	00000	18	bbb+00
68	–00000	18	bbb–00
69	00012	18	bbb+12
70	–00123	18	bb–123
71	01234	18	b+1234
72	–12345	18	–12345
73	000	19	00000.00
74	–123	19	00123.00
75	123	19	00123.00
76	00000	20	bbbbbb
77	00001	20	00,001
78	HHMMSS	21	HH/MM/SS
79	–00.00	22	bbb00.00–
80	00.00	22	bbb00.00b
81	12.34	22	bbb12.34b
82	000	23	000.00
83	–123	23	123.00
84	123	23	123.00
85	000	24	000.bb
86	–123	24	123.bb
87	123	24	123.bb
88	000	25	0b0b0
89	123	25	1b2b3

Edit Mask 26

COBOL Picture: $ $ $ $, $ $ $, $ $ $, $ $ $, $ $ $, $ $ $
Edit Mask: 11 61 $ 73 C4 73 C4 73 C4 73 80 95 03 94 03 FF E9

Example Number 90 Using Edit Mask Number 26.

Source Field: 0 0 0 0 0 0 0 0 0 0 0 0 . 0 0 0 0 0 0
Destination Field: b b b b b b b b b b b b b b b b b b b b b b b b

Example Number 91 Using Edit Mask Number 26.

Source Field: 1 2 3 4 5 6 7 8 9 0 1 2 . 6 5 4 3 2 1
Destination Field: $ 1 2 3 , 4 5 6 , 7 8 9 , 0 1 2 . 6 5 4 , 3 2 1

Interface Information   C

Interfaces

This appendix contains signal description and sequencing information for the CYBER 960 and 962 input/output channel interfaces. The following interfaces are available.

- External interface, 12-bit

- Maintenance channel (MCH) interface

- Two-port multiplexer interface

Twelve-Bit External Interface

The 12-bit external interface uses bidirectional, synchronous communication to transmit data between bits 52 through 63 of the channel data register and a number of CDC CYBER 170 external devices. The transmission is over separate input and output coaxial cables using an AC transmission scheme. In addition to 13 data signals (12 data, 1 parity), the cables also transmit eight control signals from a PP to an external device, and four from an external device to a PP. Maximum cable length between repeaters is 21 m (70 ft).

Maintenance Channel Interface

The MCH interface (channel 17_8) uses unidirectional, asynchronous communication. It transmits only 9 data bits (8 data, 1 parity) in each direction. The data transfers between bits 56 through 63 of the channel data register and the external device.

Two-Port Multiplexer Interface

The two-port multiplexer interface is an EIA standard RS-232 serial interface. Refer to this standard for more information.

Signals

The following signals are described below.

Twelve-Bit Channel Control Signals

The 12-bit channel uses the following control signals:

Signal	Description
Active Pulse	Sent by a data sending device to a data receiving device to begin a data transmission. Normally sent by a PP to an external device. An external device can send this signal to a PP only on the 12-bit channel.
Inactive Pulse	Sent from either a data sending or data receiving device to signify end of a data transmission; clears active and full flags.
Full Pulse	Sent from a data sending device to a data receiving device with transmitted data. The full pulse directs the receiving device to sample the data signals.
Empty Pulse	Sent by a data receiving device to a data sending device to acknowledge receipt of a full pulse and associated data. This pulse signals the sender to transmit more data.
Function Pulse	Sent by IOU to an external device to indicate that the associated data signals are control signals.
Master Clear	Sent by IOU to all external devices on the I/O channel. It indicates to those devices that all activity is to cease and initial conditions are to be restored.
10-Megahertz Clock	Consists of a pulse sent every 100 ns. This clock synchronizes all external devices to IOU.
1-Megahertz Clock	A pulse every 1 µs; sent by the IOU to an external device.

Maintenance Channel Signals

The MCH uses the following control signals.

Control Signals

Signal	Description
Active	Sent from IOU to external device to indicate start of data transmission.
Inactive	Sent from either data sending or data receiving device to indicate end of transmission.
Ready	Sent from data sending device to data receiving device with transmitted data; instructs receiving device to sample the data lines. The receiving device then sends a ready pulse back to acknowledge receipt of the data and to indicate it is ready for more data.
Function	Sent only to an external device to indicate that signals on data lines are control signals.
Error	Sent by external device to IOU to indicate that the device detected an error.

Signals and Cables

Table C-1 lists the signals used by the MCH interface.

Data Signals

A data sending device transmits the data signals to a data receiving device along with the associated full pulses. The IOU also transmits a function code over the data lines to an external device, with a function pulse.

Table C-1. Maintenance Channel Signals

Signal Name	Connector Pins
Data out bit 2^0 (unidirectional)	A1/A2
Data out bit 2^1 (unidirectional)	A3/A4
Data out bit 2^2 (unidirectional)	A5/A6
Data out bit 2^3 (unidirectional)	A7/A8
Data out bit 2^4 (unidirectional)	A9/A10
Data out bit 2^5 (unidirectional)	B1/B2
Data out bit 2^6 (unidirectional)	B3/B4
Data out bit 2^7 (unidirectional)	B5/B6
Data out parity (unidirectional)	B7/B8
Data in bit 2^0 (unidirectional)	C1/C2
Data in bit 2^1 (unidirectional)	C3/C4
Data in bit 2^2 (unidirectional)	C5/C6
Data in bit 2^3 (unidirectional)	C7/C8
Data in bit 2^4 (unidirectional)	C9/C10
Data in bit 2^5 (unidirectional)	D1/D2
Data in bit 2^6 (unidirectional)	D3/D4
Data in bit 2^7 (unidirectional)	D5/D6
Data in parity (unidirectional)	D7/D8
Function out	E1/E2
Ready out	E3/E4
Spare	E5/E6
Active out	E7/E8
Inactive out	E9/E10
Ready in	F1/F2
Spare	F3/F4
Inactive in	F5/F6
Summary status in	F7/F8
Exchange accept in	D9/D10
Error in	B9/B10

PP and Channel Interaction

Channel transmissions are controlled by the active and full flags. When a PP executes an I/O instruction, the state of these flags is altered and control signal(s) sent to the external devices. When the devices send control signals to the IOU, the state of these flags is again altered.

Active Flag

When the PP sets the active bit with a 00740 or 00741 instruction, the IOU sends an active pulse on the external channel. The IOU sends an inactive pulse when the PP clears the active bit. An active pulse sent by an external device sets the active bit; an inactive pulse sent by an external device clears the active bit.

Full Flag

When a PP sets the full bit using a 00720, 00721, 0073, or 1073 instruction, the IOU sends a full pulse and data pulses for the data contained in the channel data register to an external device. When a PP clears the full bit with a 00700, 00701, or 1071 instruction, the system sends an empty pulse.

When an external device sends a full pulse, the associated data pulses set the channel data register, and the system sets the full bit. When an external device sends an empty pulse, the IOU clears the full bit.

Function Instructions

When a function instruction (00760, 00761, 00770, 00771) executes, the IOU sets active and full bits, writes a word into the channel data register, and transmits the word from the data register to an external device. The IOU transmits a function pulse to the external device to indicate that the word is a control signal rather than data. The external device sends an inactive pulse to acknowledge the receipt of the function, thereby setting the active bit and the full bit to zero.

External Channel Input/Output Sequences

Figures C-1 (Data Input Sequence) and C-2 (Data Output Sequence) show the sequences followed by the channels during data input and output over an external interface.

Similarily, figures C-3 (MCH Input Sequence) and C-4 (MCH Output Sequence) show input and output sequences for the maintenance channel (17_8).

Data Sequences Timing is shown in figure C-5.
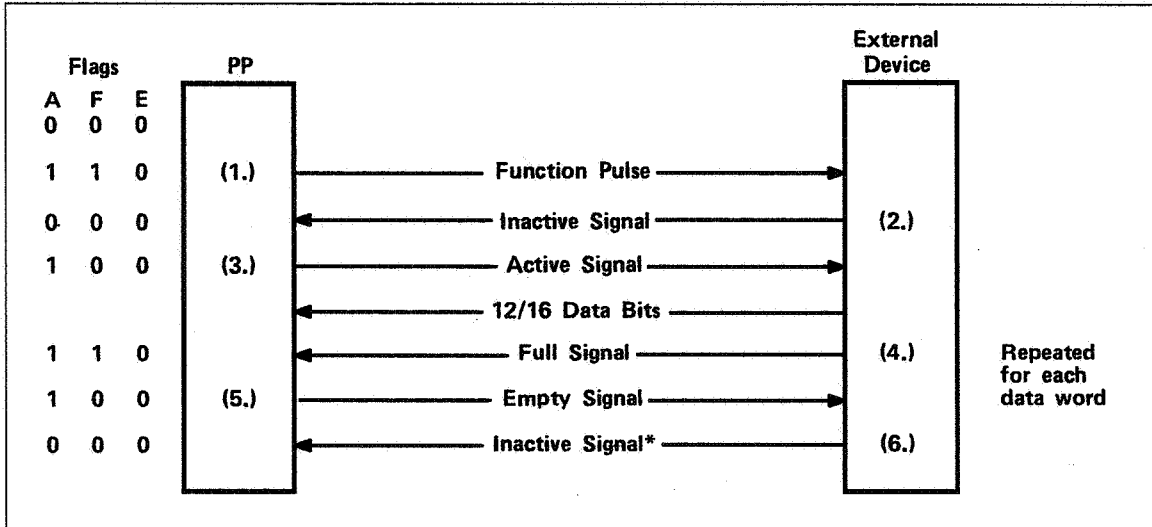
Figure C-1. Data Input Sequence

NOTE
_____

*The inactive signal is normally sent from the external device to the IOU. However, in certain cases, the IOU may deactivate the channel. This is determined by the external device and the function being executed.

_____

Key to figure C-1:

A, F, E are the active, full, and error flags.

1. PP executes a function instruction which sets the active and full flags in the internal interface, places a word in the channel register, and sends a function pulse.

2. The external device acknowledges the acceptance of the function by sending an inactive signal which clears the active flag, the full flag, and the channel register.

3. PP sets the active flag to indicate that data flow may start.

4. The external device sends a 12-bit word (plus parity) to the channel register, with a full signal which sets the full flag.

5. PP stores the data word in PPM and clears the full flag which, in turn, sends an empty signal to the external device.

6. Steps 4 and 5 repeat until the device completes the data transfer. Then the external device clears its active condition and sends an inactive signal to the PP, which clears the channel active flag.
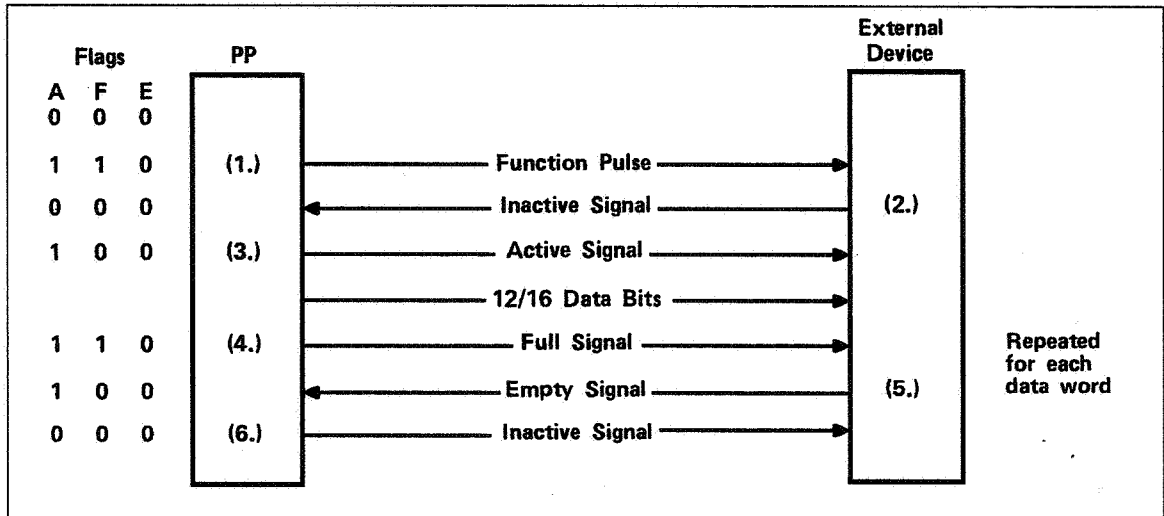
Figure C-2. Data Output Sequence

Key to figure C-2:

A, F, E are the active, full, and error flags.

1. PP executes a function instruction which sets the active and full bits in the internal interface, places a word into the channel register, and sends the function pulse.

2. The external device acknowledges the acceptance of the function by sending an inactive signal. This, in turn, clears the active flag, the full flag, and the channel register.

3. PP sets the active flag to indicate that data flow is about to start.

4. PP places a 12-bit data word (plus parity) into the channel register, which sets the full flag and sends the full signal.

5. The external device accepts the data word and sends an empty signal which clears the channel register and the full flag.

6. Steps 4 and 5 repeat until the PP has sent all the data to complete the data transfer. Then the PP clears the channel active flag, which turns off the external device with an inactive signal.
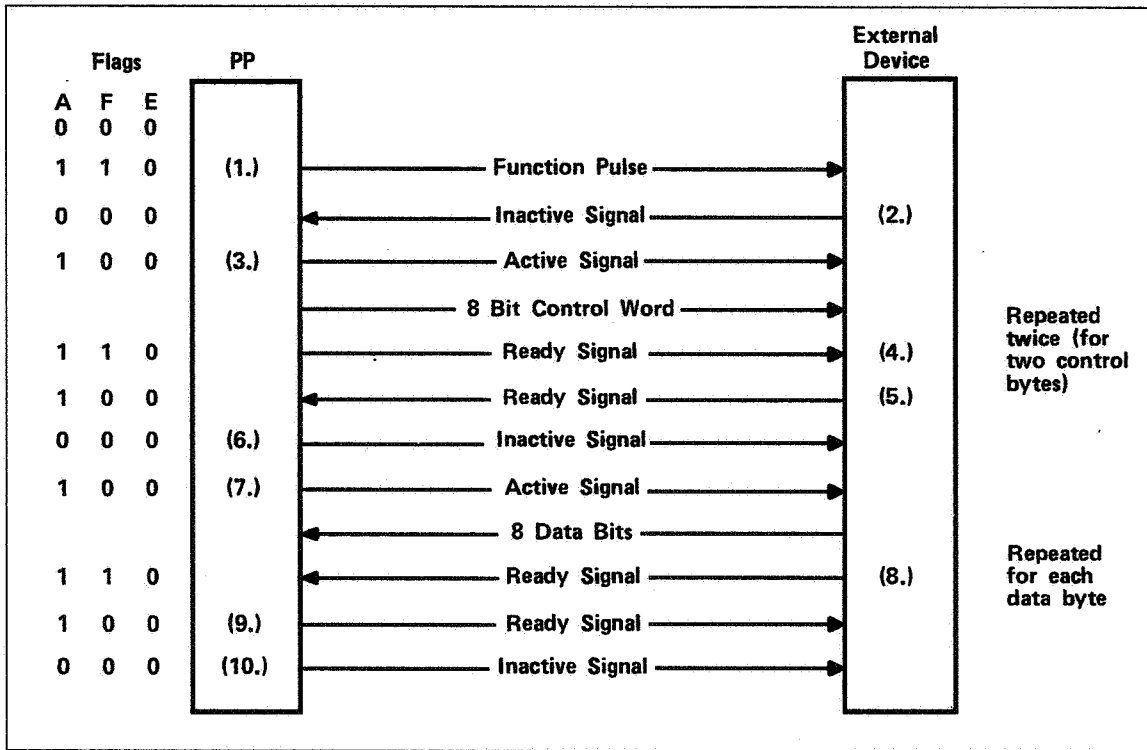
Figure C-3. MCH Input Sequence

Key to figure C-3:

A, F, E are the active, full, and error flags.

1. PP executes a function instruction which sets the active and full flags in the internal interface, places a word in the channel register and sends a function pulse.

2. The external device acknowledges the acceptance of the function by sending an inactive signal. This, in turn, clears the active flag, the full flag, and the channel register.

3. The PP sets the active flag to indicate that control word data flow is about to start.

4. The PP places a control byte into the channel register, which sets the full flag and sends the ready signal.

5. The external device accepts the control byte and sends the ready signal, which clears the channel register and the full flag.

6. Steps 4 and 5 are repeated for a second control byte. The two control bytes contain the upper and lower portions of the address of the data to be read.

7. The PP ensures that the channel is empty and then deactivates the channel, which clears the active flag.

8. The PP sets the active flag to indicate that data flow may start.

9. The external device sends an 8-bit byte to the channel register with a ready signal which, in turn, sets the full flag.

10. The PP stores the data word and clears the full flag which, in turn, sends the ready signal to the external device.

11. Steps 8 and 9 repeat until the data transfer is complete. The PP deactivates the channel, which turns off the external device with an inactive signal.
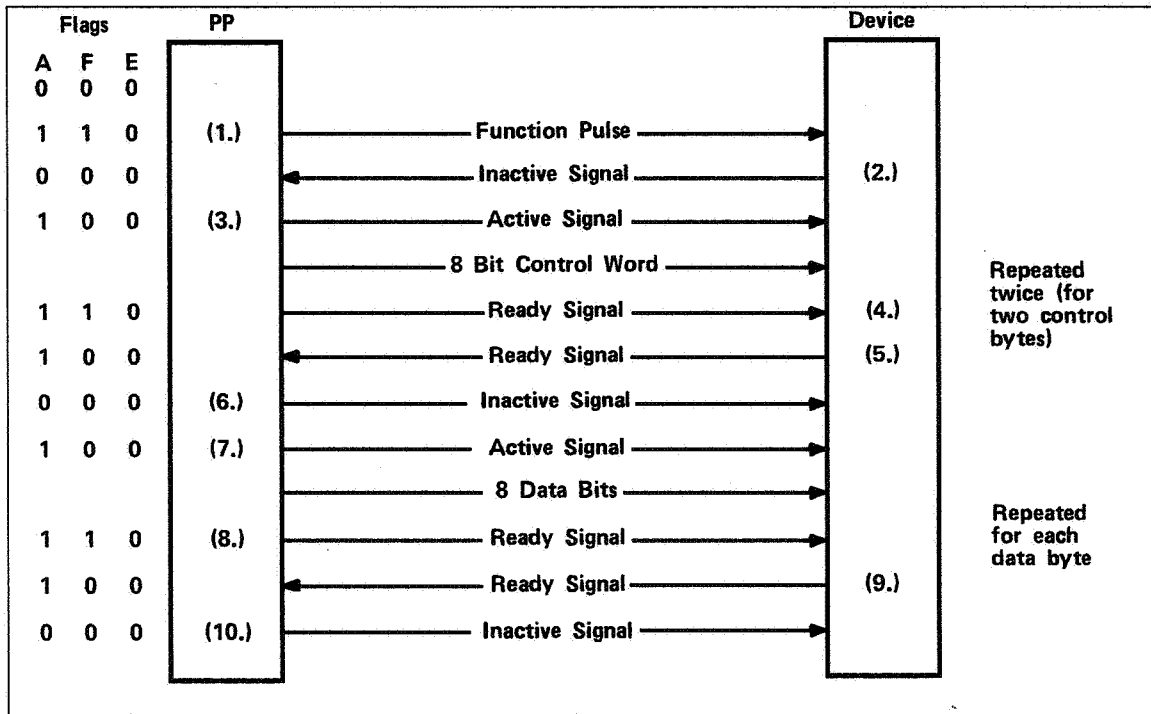
Figure C-4. MCH Output Sequence

Key to figure C-4:

A, F, E are the active, full, and error flags.

1. PP executes a function instruction which sets the active and full bits in the internal interface, places a word in the channel register and sends the function pulse.

2. The external device acknowledges the acceptance of the function by sending an inactive signal. This, in turn, clears the active flag, the full flag, and the channel register.

3. The PP sets the active flag to indicate that control word data flow is about to start.

4. The PP places a control byte into the channel register, which sets the ready flag and sends the full signal.

5. The external device accepts the control byte and sends the ready signal which clears the channel register and the full flag.

6. The PP ensures that the channel is empty and then deactivates the channel, which clears the active flag.

7. The PP sets the active flag to indicate that data flow is about to start.

8. The PP places an 8-bit byte into the channel register, which sets the full flag and sends the ready signal.

9. The external device accepts the data byte and sends the ready signal, which clears the channel register and the full flag.

10. Steps 8 and 9 repeat a sufficient number of times to complete the data transfers. The PP deactivates the channel, which turns off the external device with an inactive signal.
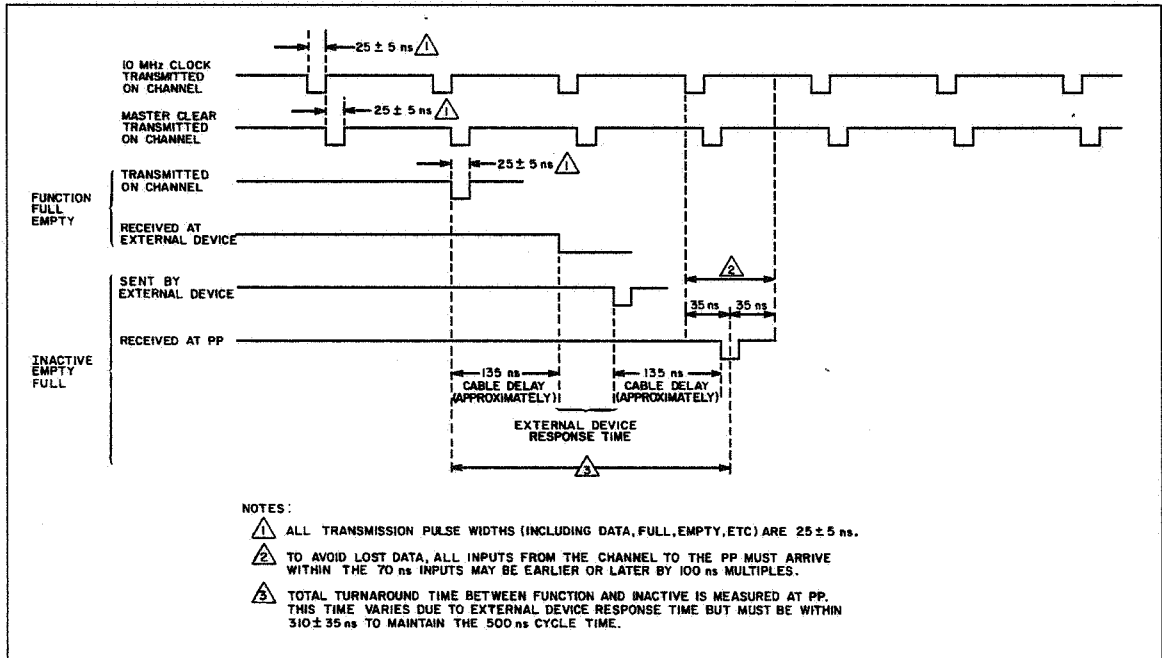
Figure C-5. Data Sequences Timing

Instruction Index

D

This appendix lists the central processor (CP) and peripheral processor (PP) instructions, in both opcode and mnemonic sequences.

Table D-1. CP Instructions - Opcode Sequence

Opcode	Mnemonic	Instruction	Page
00	HALT	Program error	1-83
01	SYNC	Scope loop synchronization	1-84
02	EXCHANGE	Exchange	1-85
03	INTRUPT	Processor interrupt	1-100
04	RETURN	Return	1-86
05	PURGE	Purge buffer	1-102
06	POP	Pop	1-88
08	CPYTX	Copy free running counter	1-89
09	CPYAA	Copy address, A to A	1-31
0A	CPYXA	Copy address, X to A	1-31
0B	CPYAX	Copy address, A to X	1-31
0C	CPYRR	Copy half-word	1-32
0D	CPYXX	Copy full-word	1-32
0E	CPYSX	Copy from state register	1-104
0F	CPYXS	Copy to state register	1-104
10	INCX	Integer sum, immediate	1-19
11	DECX	Integer difference, immediate	1-20
14	LBSET	Test and set bit	1-90
16	TPAGE	Test and set page	1-91
17	LPAGE	Load page table index	1-99
18	IORX	Logical sum	1-41
19	XORX	Logical difference	1-41
1A	ANDX	Logical product	1-41
1B	NOTX	Logical complement	1-42
1C	INHX	Logical inhibit	1-42

(Continued)

Table D-1. CP Instructions - Opcode Sequence *(Continued)*

Opcode	Mnemonic	Instruction	Page
1E	MARK	Mark to Boolean	1-45
1F	ENTZ	Enter zeros	1-36
1F	ENTO	Enter ones	1-36
1F	ENTS	Enter signs	1-36
20	ADDR	Half-word integer sum	1-18
21	SUBR	Half-word integer difference	1-20
22	MULR	Half-word integer product	1-21
23	DIVR	Half-word integer quotient	1-22
24	ADDX	Integer sum	1-19
25	SUBX	Integer difference	1-20
26	MULX	Integer product	1-22
27	DIVX	Integer quotient	1-23
28	INCR	Half-word integer sum, immediate	1-18
29	DECR	Half-word integer difference, immediate	1-20
2A	ADDAX	Address increment, indexed	1-33
2C	CMPR	Half-word integer compare	1-23
2D	CMPX	Integer compare	1-23
2E	BRREL	Branch relative	1-24
2F	BRDIR	Intersegment branch	1-25
30	ADDF	FP sum	1-74
31	SUBF	FP difference	1-74
32	MULF	FP product	1-75
33	DIVF	FP quotient	1-76
34	ADDD	Double-precision FP sum	1-74
35	SUBD	Double-precision FP difference	1-74
36	MULD	Double-precision FP product	1-75
37	DIVD	Double-precision FP quotient	1-76
39	ENTX	Enter X1, immediate logical	1-37

(Continued)

Table D-1. CP Instructions - Opcode Sequence *(Continued)*

Opcode	Mnemonic	Instruction	Page
3A	CNIF	Convert from integer to FP	1-72
3B	CNFI	Convert from FP to integer	1-72
3C	CMPF	FP compare	1-80
3D	ENTP	Enter immediate, positive	1-37
3E	ENTN	Enter immediate, negative	1-37
3F	ENTL	Enter immediate, logical	1-37
70	ADDN	Decimal sum	1-50
71	SUBN	Decimal difference	1-50
72	MULN	Decimal product	1-50
73	DIVN	Decimal quotient	1-50
74	CMPN	Decimal compare	1-52
75	MOVN	Numeric move	1-53
76	MOVB	Move bytes	1-59
77	CMPB	Byte compare	1-57
80	LMULT	Load multiple	1-8
81	SMULT	Store multiple	1-8
82	LX	Load word	1-10
83	SX	Store word	1-10
84	LA	Load address	1-11
85	SA	Store address	1-11
86	LBYTP	Load bytes, relative	1-14
87	ENTC	Enter X1, signed immediate	1-38
88	LBIT	Load bit	1-15
89	SBIT	Store bit	1-15
8A	ADDRQ	Half-word integer sum, signed immediate	1-18
8B	ADDXQ	Integer sum, signed immediate	1-19
8C	MULRQ	Half-word integer product, signed immediate	1-21
8D	ENTE	Enter, signed immediate	1-38

(Continued)

Table D-1. CP Instructions - Opcode Sequence *(Continued)*

Opcode	Mnemonic	Instruction	Page
8E	ADDAQ	Address increment, signed immediate	1-33
8F	ADDPXQ	Address relative	1-34
90	BRREQ	Branch on half-word equal	1-26
91	BRRNE	Branch on half-word not equal	1-26
92	BRRGT	Branch on half-word greater than	1-26
93	BRRGE	Branch on half-word greater than or equal	1-26
94	BRXEQ	Branch on equal	1-27
95	BRXNE	Branch on not equal	1-27
96	BRXGT	Branch on greater than	1-27
97	BRXGE	Branch on greater than or equal	1-27
98	BRFEQ	FP branch on equal	1-78
99	BRFNE	FP branch on not equal	1-78
9A	BRFGT	FP branch on greater than	1-78
9B	BRFGE	FP branch on greater than or equal	1-78
9C	BRINC	Branch and increment	1-28
9D	BRSEG	Branch on segments unequal	1-29
9E	BROVR	FP branch on overflow	1-79
9E	BRUND	FP branch on underflow	1-79
9E	BRINF	FP branch on indefinite	1-79
9F	BRCR	Branch on condition register	1-105
A0	LAI	Load address, indexed	1-12
A1	SAI	Store address, indexed	1-12
A2	LXI	Load word, indexed	1-10
A3	SXI	Store word, indexed	1-10
A4	LBYT	Load bytes	1-13
A5	SBYT	Store bytes	1-13

(Continued)

Table D-1. CP Instructions - Opcode Sequence *(Continued)*

Opcode	Mnemonic	Instruction	Page
A7	ADDAD	Address increment, modulo	1-34
A8	SHFC	Shift word, circular	1-40
A9	SHFX	Shift word, end-off	1-40
AA	SHFR	Shift half-word, end-off	1-40
AC	ISOM	Isolate bit mask	1-44
AD	ISOB	Isolate	1-44
AE	INSB	Insert	1-44
B0	CALLREL	Call relative	1-92
B2	MULXQ	Integer product, signed immediate	1-22
B3	ENTA	Enter X0, signed immediate	1-38
B4	CMPXA	Compare swap	1-94
B5	CALLSEG	Call indirect	1-96
BE	(None)	Reserved for user	1-98
BF	(None)	Reserved for user	1-98
CS	EXECUTE,S	Execute algorithm per S	1-98
DS	LBYTS	Load bytes, immediate	1-13
DS	SBYTS	Store bytes, immediate	1-13
E4	SCLN	Decimal scale	1-54
E5	SCLR	Decimal scale, rounded	1-54
E9	CMPC	Byte compare, collated	1-57
EB	TRANB	Byte translate	1-58
ED	EDIT	Edit	1-59
F3	SCNB	Byte scan while nonmember	1-65
F4	CALDF	Calculate subscript and add	1-67
F9	MCVI	Move immediate data	1-68
FA	CMPI	Compare immediate data	1-69
FB	ADDI	Add immediate data	1-70

To present the information in this chapter in a structured format, this page has been left blank.

Table D-2. CP Instructions - Mnemonic Sequence

Mnemonic	Opcode	Instruction	Page
ADDAD	A7	Address increment, modulo	1-34
ADDAQ	8E	Address increment, signed immediate	1-33
ADDAX	2A	Address increment, indexed	1-33
ADDD	34	Double-precision FP sum	1-74
ADDF	30	FP sum	1-74
ADDI	FB	Add immediate data	1-70
ADDN	70	Decimal sum	1-50
ADDPXQ	8F	Address relative	1-34
ADDR	20	Half-word integer sum	1-18
ADDRQ	8A	Half-word integer sum, signed immediate	1-18
ADDX	24	Integer sum	1-19
ADDXQ	8B	Integer sum, signed immediate	1-19
ANDX	1A	Logical product	1-41
BRCR	9F	Branch on condition register	1-105
BRDIR	2F	Intersegment branch	1-25
BRFEQ	98	FP branch on equal	1-78
BRFGE	9B	FP branch on greater than or equal	1-78
BRFGT	9A	FP branch on greater than	1-78
BRFNE	99	FP branch on not equal	1-78
BRINC	9C	Branch and increment	1-28
BRINF	9E	FP branch on indefinite	1-79
BROVR	9E	FP branch on overflow	1-79
BRREL	2E	Branch relative	1-24
BRREQ	90	Branch on half-word equal	1-26
BRRGE	93	Branch on half-word greater than or equal	1-26

(Continued)

Table D-2. CP Instructions - Mnemonic Sequence *(Continued)*

Mnemonic	Opcode	Instruction	Page
BRRGT	92	Branch on half-word greater than	1-26
BRRNE	91	Branch on half-word not equal	1-26
BRSEG	9D	Branch on segments unequal	1-29
BRUND	9E	FP branch on underflow	1-79
BRXEQ	94	Branch on equal	1-27
BRXGE	97	Branch on greater than or equal	1-27
BRXGT	96	Branch on greater than	1-27
BRXNE	95	Branch on not equal	1-27
CALDF	F4	Calculate subscript and add	1-67
CALLREL	B0	Call relative	1-92
CALLSEG	B5	Call indirect	1-96
CMPB	77	Byte compare	1-57
CMPC	E9	Byte compare, collated	1-57
CMPF	3C	FP compare	1-80
CMPI	FA	Compare immediate data	1-69
CMPN	74	Decimal compare	1-52
CMPR	2C	Half-word integer compare	1-23
CMPX	2D	Integer compare	1-23
CMPXA	B4	Compare swap	1-94
CNFI	3B	Convert from FP to integer	1-72
CNIF	3A	Convert from integer to FP	1-72
CPYAA	09	Copy address, A to A	1-31
CPYAX	0B	Copy address, A to X	1-31
CPYTX	08	Copy free running counter	1-89
CPYRR	0C	Copy half-word	1-32
CPYSX	0E	Copy from state register	1-104
CPYXA	0A	Copy address, X to A	1-31

(Continued)

Table D-2. CP Instructions - Mnemonic Sequence *(Continued)*

Mnemonic	Opcode	Instruction	Page
CPYXS	0F	Copy to state register	1-104
CPYXX	0D	Copy full-word	1-32
DECR	29	Half-word integer difference, immediate	1-20
DECX	11	Integer difference, immediate	1-20
DIVD	37	Double-precision FP quotient	1-76
DIVF	33	FP quotient	1-76
DIVN	73	Decimal quotient	1-50
DIVR	23	Half-word integer quotient	1-22
DIVX	27	Integer quotient	1-23
EDIT	ED	Edit	1-59
ENTA	B3	Enter X0, signed immediate	1-38
ENTC	87	Enter X1, signed immediate	1-38
ENTE	8D	Enter, signed immediate	1-38
ENTL	3F	Enter immediate, logical	1-37
ENTN	3E	Enter immediate, negative	1-37
ENTO	1F	Enter ones	1-36
ENTP	3D	Enter immediate, positive	1-37
ENTS	1F	Enter signs	1-36
ENTX	39	Enter X1, immediate logical	1-37
ENTZ	1F	Enter zeros	1-36
EXCHANGE	02	Exchange	1-85
EXECUTE,S	CS	Execute algorithm per S	1-98
HALT	00	Program error	1-83
INCR	28	Half-word integer sum, immediate	1-18
INCX	10	Integer sum, immediate	1-19
INHX	1C	Logical inhibit	1-42
INSB	AE	Insert	1-44

(Continued)

Table D-2. CP Instructions - Mnemonic Sequence *(Continued)*

Mnemonic	Opcode	Instruction	Page
INTRUPT	03	Processor interrupt	1-100
IORX	18	Logical sum	1-41
ISOB	AD	Isolate	1-44
ISOM	AC	Isolate bit mask	1-44
LA	84	Load address	1-11
LAI	A0	Load address, indexed	1-12
LBIT	88	Load bit	1-15
LBSET	14	Test and set bit	1-90
LBYT	A4	Load bytes	1-13
LBYTP	86	Load bytes, relative	1-14
LBYTS	DS	Load bytes, immediate	1-13
LMULT	80	Load multiple	1-8
LPAGE	17	Load page table index	1-99
LX	82	Load word	1-10
LXI	A2	Load word, indexed	1-10
MARK	1E	Mark to Boolean	1-45
MCVI	F9	Move immediate data	1-68
MOVB	76	Move bytes	1-59
MOVN	75	Numeric move	1-53
MULD	36	Double-precision FP product	1-75
MULF	32	FP product	1-75
MULN	72	Decimal product	1-50
MULR	22	Half-word integer product	1-21
MULRQ	8C	Half-word integer product, signed immediate	1-21
MULX	26	Integer product	1-22
MULXQ	B2	Integer product, signed immediate	1-22
NOTX	1B	Logical complement	1-42

(Continued)

Table D-2. CP Instructions - Mnemonic Sequence *(Continued)*

Mnemonic	Opcode	Instruction	Page
POP	06	Pop	1-88
PURGE	05	Purge buffer	1-102
RETURN	04	Return	1-86
SA	85	Store address	1-11
SAI	A1	Store address, indexed	1-12
SBIT	89	Store bit	1-15
SBYT	A5	Store bytes	1-13
SBYTS	DS	Store bytes, immediate	1-13
SCLN	E4	Decimal scale	1-54
SCLR	E5	Decimal scale, rounded	1-54
SCNB	F3	Byte scan while nonmember	1-65
SHFC	A8	Shift word, circular	1-40
SHFR	AA	Shift half-word, end-off	1-40
SHFX	A9	Shift word, end-off	1-40
SMULT	81	Store multiple	1-8
SUBD	35	Double-precision FP difference	1-74
SUBF	31	FP difference	1-74
SUBN	71	Decimal difference	1-50
SUBR	21	Half-word integer difference	1-20
SUBX	25	Integer difference	1-20
SX	83	Store word	1-10
SXI	A3	Store word, indexed	1-10
SYNC	01	Scope loop synchronization	1-84
TPAGE	16	Test and set page	1-91
TRANB	EB	Byte translate	1-58
XORX	19	Logical difference	1-41
(None)	BE	Reserved for user	1-98
(None)	BF	Reserved for user	1-98

To present the information in this chapter in a structured format, this page has been left blank.

Table D-3. PP Instructions - Opcode Sequence

Opcode	Mnemonic	Instruction	Page
0000	PSN	Pass	1-167
0001dm	LJM	Long jump to m+(d)	1-138
0002dm	RJM	Return jump to m+(d)	1-139
0003d	UJN	Unconditional jump d	1-139
0004d	ZJN	Zero jump d	1-140
0005d	NJN	Nonzero jump d	1-140
0006d	PJN	Plus jump d	1-141
0007d	MJN	Minus jump d	1-141
0010d	SHN	Shift A by d	1-123
0011d	LMN	Logical difference d	1-123
0012d	LPN	Logical product d	1-128
0013d	SCN	Selective clear d	1-130
0014d	LDN	Load d	1-110
0015d	LCH	Load complement d	1-110
0016d	ADN	Add d	1-116
0017d	SBN	Subtract d	1-119
0020dm	LDC	Load dm	1-110
0021dm	ADC	Add dm	1-116
0022dm	LPC	Logical product dm	1-128
0023dm	LMC	Logical difference dm	1-124
002400	PSN	Pass	1-167
0024d	LRD	Load R	1-143
002500	PSN	Pass	1-167
0025d	SRD	Store R	1-143
002600	EXN	Exchange jump	1-168

(Continued)

Table D-3. PP Instructions - Opcode Sequence *(Continued)*

Opcode	Mnemonic	Instruction	Page
002610	MXN	Monitor exchange jump	1-168
002620	MAN	Monitor exchange jump to MA	1-169
00263x	(None)	Executes as if d = 2x	1-169
0027	KPT	Keypoint	1-167
0030d	LDD	Load (d)	1-110
0031d	ADD	Add (d)	1-116
0032d	SBD	Subtract (d)	1-119
0033d	LMD	Logical difference (d)	1-124
0034d	STD	Store (d)	1-113
0035d	RAD	Replace add (d)	1-132
0036d	AOD	Replace add one (d)	1-132
0037d	SOD	Replace subtract one (d)	1-136
0040d	LDI	Load ((d))	1-111
0041d	ADI	Add ((d))	1-117
0042d	SBI	Subtract ((d))	1-120
0043d	LMI	Logical difference ((d))	1-125
0044d	STI	Store ((d))	1-113
0045d	RAI	Replace add ((d))	1-133
0046d	AOI	Replace add one ((d))	1-134
0047d	SOI	Replace subtract one ((d))	1-136
0050dm	LDM	Load (m+(d))	1-112
0051dm	ADM	Add (m+(d))	1-118
0052dm	SBM	Subtract (m+(d))	1-120
0053dm	LMM	Logical difference (m+(d))	1-126
0054dm	STM	Store (m+(d))	1-114
0055dm	RAM	Replace add (m+(d))	1-134
0056dm	AOM	Replace add one (m+(d))	1-135

(Continued)

Table D-3. PP Instructions - Opcode Sequence (Continued)

Opcode	Mnemonic	Instruction	Page
0057dm	SOM	Replace subtract one (m+(d))	1-137
0060d	CRD	Central read from (A) to d	1-144
0061dm	CRM	Central read (d) words from (A) to m	1-146
0062d	CWD	Central write to (A) from d	1-149
0063dm	CWM	Central write (d) words to (A) from m	1-151
00640cm	AJM	Jump to m if channel C active	1-155
00641cm	SCF	Test and set channel C flag	1-157
00650cm	IJM	Jump to m if channel C inactive	1-155
00651cm	CCF	Clear channel C flag	1-157
00660cm	FJM	Jump to m if channel C full	1-156
00661cm	SFM	Test and clear channel C error flag set	1-157
00670cm	EJM	Jump to m if channel C empty	1-156
00671cm	CFM	Test and clear channel C error flag clear	1-158
00700c	IAN	Input to A from channel C when active and full	1-158
00701c	IAN	Input to A from channel C if active and full	1-158
0071xcm	IAM	Input A words to m from channel C	1-159
00720c	OAN	Output from A on channel C when active	1-161
00721c	OAN	Output from A on channel C if active	1-161
0073xcm	OAPM	Output A words from m on channel C	1-162
00740c	ACN	Activate channel C	1-163
00741c	ACN	Unconditionally activate channel C	1-163
00750c	DCN	Deactivate channel C	1-164
00751c	DCN	Unconditionally deactivate channel C	1-164
00760c	FAN	Function a on channel C when inactive	1-165
00761c	FAN	Function a on channel C if inactive	1-165
00770cm	FNC	Function m on channel C when inactive	1-166
00771cm	FNC	Function m on channel C if inactive	1-166

(Continued)

Table D-3. PP Instructions - Opcode Sequence *(Continued)*

Opcode	Mnemonic	Instruction	Page
1000d	RDSL	Central read and set lock from d to (A)	1-147
1001d	RDCL	Central read and clear lock from d to (A)	1-148
1002	PSN	Pass	1-167
1003	PSN	Pass	1-167
1004	PSN	Pass	1-167
1005	PSN	Pass	1-167
1006	PSN	Pass	1-167
1007	PSN	Pass	1-167
1010	PSN	Pass	1-167
1011	PSN	Pass	1-167
1012	PSN	Pass	1-167
1013	PSN	Pass	1-167
1014	PSN	Pass	1-167
1015	PSN	Pass	1-167
1016	PSN	Pass	1-167
1017	PSN	Pass	1-167
1020	PSN	Pass	1-167
1021	PSN	Pass	1-167
1022d	LPDL	Logical product (d) long	1-129
1023d	LPIL	Logical product ((d)) long	1-129
1024dm	LPML	Logical product (m+(d)) long	1-129
1025	PSN	Pass	1-167
1026d	INPN	Interrupt processor	1-169
1027	PSN	Pass	1-167
1030d	LDDL	Load (d) long	1-111
1031d	ADDL	Add (d) long	1-117
1032d	SBDL	Subtract (d) long	1-119

(Continued)

Table D-3. PP Instructions - Opcode Sequence *(Continued)*

Opcode	Mnemonic	Instruction	Page
1033d	LMDL	Logical difference (d) long	1-124
1034d	STDL	Store (d) long	1-113
1035d	RADL	Replace add (d) long	1-132
1036d	AODL	Replace add one (d) long	1-133
1037d	SODL	Replace subtract one (d) long	1-136
1040d	LDIL	Load ((d)) long	1-111
1041d	ADIL	Add ((d)) long	1-117
1042d	SBIL	Subtract ((d)) long	1-120
1043d	LMIL	Logical difference ((d)) long	1-125
1044d	STIL	Store ((d)) long	1-113
1045d	RAIL	Replace add ((d)) long	1-133
1046d	AOIL	Replace add one ((d)) long	1-134
1047d	SOIL	Replace subtract one ((d)) long	1-137
1050dm	LDML	Load (m+(d)) long	1-112
1051dm	ADML	Add (m+(d)) long	1-118
1052dm	SBML	Subtract (m+(d)) long	1-121
1053dm	LMML	Logical difference (m+(d)) long	1-127
1054dm	STML	Store (m+(d)) long	1-114
1055dm	RAML	Replace add (m+(d)) long	1-135
1056dm	AOML	Replace add one (m+(d)) long	1-135
1057dm	SOML	Replace subtract one (m+(d)) long	1-137
1060d	CRDL	Central read from (A) to d long	1-145
1061dm	CRML	Central read (d) words from (A) to m long	1-147
1062d	CWDL	Central write to (A) from d long	1-150
1063dm	CWML	Central write (d) words to (A) from m long	1-152
1064xcm	FSJM	Jump to m if channel C flag set	1-155
1065xcm	FCJM	Jump to m if channel C flag clear	1-156

(Continued)

Table D-3. PP Instructions - Opcode Sequence *(Continued)*

Opcode	Mnemonic	Instruction	Page
1066	PSN	Pass	1-167
1067	PSN	Pass	1-167
1070	PSN	Pass	1-167
1071xcm	IAPM	Input A words to m from channel C packed	1-160
1072	PSN	Pass	1-167
1073xcm	OAPM	Output A words from m on channel C packed	1-162
1074	PSN	Pass	1-167
1075	PSN	Pass	1-167
1076	PSN	Pass	1-167
1077	PSN	Pass	1-167

Table D-4. PP Instructions - Mnemonic Sequence

Mnemonic	Opcode	Instruction	Page
(None)	00263x	Executes as if d = 2x	1-169
ACN	00740c	Activate channel C	1-163
ACN	00741c	Unconditionally activate channel C	1-163
ADC	0021dm	Add dm	1-116
ADD	0031d	Add (d)	1-116
ADDL	1031d	Add (d) long	1-117
ADI	0041d	Add ((d))	1-117
ADIL	1041d	Add ((d)) long	1-117
ADM	0051dm	Add (m+(d))	1-118
ADML	1051dm	Add (m+(d)) long	1-118
ADN	0016d	Add d	1-116
AJM	00640cm	Jump to m if channel C active	1-155
AOD	0036d	Replace add one (d)	1-132
AODL	1036d	Replace add one (d) long	1-133
AOI	0046d	Replace add one ((d))	1-134
AOIL	1046d	Replace add one ((d)) long	1-134
AOM	0056dm	Replace add one (m+(d))	1-135
AOML	1056dm	Replace add one (m+(d)) long	1-135
CCF	00651cm	Clear channel C flag	1-157
CFM	00671cm	Test and clear channel C error flag set	1-158
CRD	0060d	Central read from (A) to d	1-144
CRDL	1060d	Central read from (A) to d long	1-145
CRM	0061dm	Central read (d) words from (A) to m	1-146
CRML	1061dm	Central read (d) words from (A) to m long	1-147
CWD	0062d	Central write to (A) from d	1-149

(Continued)

Table D-4. PP Instructions - Mnemonic Sequence *(Continued)*

Mnemonic	Opcode	Instruction	Page
CWDL	1062d	Central write to (A) from d long	1-150
CWM	0063dm	Central write (d) words to (A) from m	1-151
CWML	1063dm	Central write (d) words to (A) from m long	1-152
DCN	00750c	Deactivate channel C	1-164
DCN	00751c	Unconditionally deactivate channel C	1-164
EJM	00670cm	Jump to m if channel C empty	1-156
EXN	002600	Exchange jump	1-168
FAN	00760c	Function a on channel C when inactive	1-165
FAN	00761c	Function a on channel C if inactive	1-165
FCJM	1065xcm	Jump to m if channel C flag clear	1-156
FJM	00660cm	Jump to m if channel C full	1-156
FNC	00770cm	Function m on channel C when inactive	1-166
FNC	00771cm	Function m on channel C if inactive	1-166
FSJM	1064xcm	Jump to m if channel C flag set	1-155
IAM	0071xcm	Input A words to m from channel C	1-159
IAN	00700c	Input to A from channel C when active and full	1-158
IAN	00701c	Input to A from channel C if active and full	1-158
IAPM	1071xcm	Input A words to m from channel C packed	1-160
IJM	00650cm	Jump to m if channel C inactive	1-155
INPN	1026d	Interrupt processor	1-169
KPT	0027	Keypoint	1-167
LCH	0015d	Load complement d	1-110
LDC	0020dm	Load dm	1-110
LDD	0030d	Load (d)	1-110
LDDL	1030d	Load (d) long	1-111
LDI	0040d	Load ((d))	1-111
LDIL	1040d	Load ((d)) long	1-111

(Continued)

Table D-4. PP Instructions - Mnemonic Sequence *(Continued)*

Mnemonic	Opcode	Instruction	Page
LDM	0050dm	Load (m+(d))	1-112
LDML	1050dm	Load (m+(d)) long	1-112
LDN	0014d	Load d	1-110
LJM	0001dm	Long jump to m+(d)	1-138
LMC	0023dm	Logical difference dm	1-124
LMD	0033d	Logical difference (d)	1-124
LMDL	1033d	Logical difference (d) long	1-124
LMI	0043d	Logical difference ((d))	1-125
LMIL	1043d	Logical difference ((d)) long	1-125
LMM	0053dm	Logical difference (m+(d))	1-126
LMML	1053dm	Logical difference (m+(d)) long	1-127
LMN	0011d	Logical difference d	1-123
LPC	0022dm	Logical product dm	1-128
LPDL	1022d	Logical product (d) long	1-129
LPIL	1023d	Logical product ((d)) long	1-129
LPML	1024dm	Logical product (m+(d)) long	1-129
LPN	0012d	Logical product d	1-128
LRD	0024d	Load R	1-143
MAN	002620	Monitor exchange jump to MA	1-169
MJN	0007d	Minus jump d	1-141
MXN	002610	Monitor exchange jump	1-168
NJN	0005d	Nonzero jump d	1-140
OAPM	0073xcm	Output A words from m on channel C	1-162
OAN	00720c	Output from A on channel C when active	1-161
OAN	00721c	Output from A on channel C if active	1-161
OAPM	1073xcm	Output A words from m on channel C	1-162
PJN	0006d	Plus jump d	1-141

(Continued)

Table D-4. PP Instructions - Mnemonic Sequence *(Continued)*

Mnemonic	Opcode	Instruction	Page
PSN	0000	Pass	1-167
PSN	002400	Pass	1-167
PSN	002500	Pass	1-167
PSN	1002	Pass	1-167
PSN	1003	Pass	1-167
PSN	1004	Pass	1-167
PSN	1005	Pass	1-167
PSN	1006	Pass	1-167
PSN	1007	Pass	1-167
PSN	1010	Pass	1-167
PSN	1011	Pass	1-167
PSN	1012	Pass	1-167
PSN	1013	Pass	1-167
PSN	1014	Pass	1-167
PSN	1015	Pass	1-167
PSN	1016	Pass	1-167
PSN	1017	Pass	1-167
PSN	1020	Pass	1-167
PSN	1021	Pass	1-167
PSN	1025	Pass	1-167
PSN	1027	Pass	1-167
PSN	1066	Pass	1-167
PSN	1067	Pass	1-167
PSN	1070	Pass	1-167
PSN	1072	Pass	1-167
PSN	1074	Pass	1-167
PSN	1075	Pass	1-167

(Continued)

Table D-4. PP Instructions - Mnemonic Sequence *(Continued)*

Mnemonic	Opcode	Instruction	Page
PSN	1076	Pass	1-167
PSN	1077	Pass	1-167
RAD	0035d	Replace add (d)	1-132
RADL	1035d	Replace add (d) long	1-132
RAI	0045d	Replace add ((d))	1-133
RAIL	1045d	Replace add ((d)) long	1-133
RAM	0055dm	Replace add (m+(d))	1-134
RAML	1055dm	Replace add (m+(d)) long	1-135
RDCL	1001d	Central read and clear lock from d to (A)	1-148
RDSL	1000d	Central read and set lock from d to (A)	1-147
RJM	0002dm	Return jump to m+(d)	1-139
SBD	0032d	Subtract (d)	1-119
SBDL	1032d	Subtract (d) long	1-119
SBI	0042d	Subtract ((d))	1-120
SBIL	1042d	Subtract ((d)) long	1-120
SBM	0052dm	Subtract (m+(d))	1-120
SBML	1052dm	Subtract (m+(d)) long	1-121
SBN	0017d	Subtract d	1-119
SCF	00641cm	Test and set channel C flag	1-157
SCN	0013d	Selective clear d	1-130
SFM	00661cm	Test and clear channel C error flag set	1-157
SHN	0010d	Shift A by d	1-123
SOD	0037d	Replace subtract one (d)	1-136
SODL	1037d	Replace subtract one (d) long	1-136
SOI	0047d	Replace subtract one ((d))	1-136
SOIL	1047d	Replace subtract one ((d)) long	1-137
SOM	0057dm	Replace subtract one (m+(d))	1-137

(Continued)

Table D-4. PP Instructions - Mnemonic Sequence *(Continued)*

Mnemonic	Opcode	Instruction	Page
SOML	1057dm	Replace subtract one (m+(d)) long	1-137
SRD	0025d	Store R	1-143
STD	0034d	Store (d)	1-113
STDL	1034d	Store (d) long	1-113
STI	0044d	Store ((d))	1-113
STIL	1044d	Store ((d)) long	1-113
STM	0054dm	Store (m+(d))	1-114
STML	1054dm	Store (m+(d)) long	1-114
UJN	0003d	Unconditional jump d	1-139
ZJN	0004d	Zero jump d	1-140

Fast DMA Transfers   E

The adapter supports the fast-transfer mode of the ESM-II low-speed port. Fast
transfers are in DMA mode and have a transfer rate of one 12-bit word every 100
nanoseconds (120 million bits per second). An external clock from ESM-II enters the
adapter to enable this fast transfer rate. Bit 55 of the start DMA input/output function
enables fast transfers.

External interface signals are the same as standard 170 channels except active in,
empty in, and empty out. Active in receives the asynchronous 10-MHz clock from
ESM-II. All input/output transmissions are synchronized to this clock. A switch in the
ESM-II low-speed port enables this clock. Empty out controls inputs from ESM-II. If the
adapter issues an empty out, ESM-II inhibits the next full in. Empty in controls
outputs from the adapter. If ESM-II issues an empty in, the adapter inhibits the next
full out.

A typical sequence for executing a fast transfer is:

1. Master clear adapter.

2. Master clear ESM-II low-speed port.

3. Function adapter to read Operational Status Register. Check for external clock
    present.

4. Function adapter to write T-Prime Register.

5. Function ESM-II low-speed port to enable fast transfers.

6. Function ESM-II low-speed port for a write operation.

7. Function adapter to start DMA output bit 55 set, and PP Word Count equals two.

8. Output two words from PP. This is ESM-II address. DMA output transfer starts
    immediately after last word of address.

9. Function adapter to read Operational Status Register. Wait for transfer to complete.

10. Function adapter to clear DMA mode.

11. Function ESM-II low-speed port to read status.

12. Verify that transfer completed without errors.

Channel/CM Data Mapping

For 60-bit CM, five 12-bit channel words map evenly into a CM word. The four most-significant bits (0 through 3) in the CM word are not used for reads and zeros for writes. Fifteen channel words map into parcels 0 through 3 at three CM locations as shown below.

0		15		31		47		63
PARCEL 0		PARCEL 1		PARCEL 2		PARCEL 3	
1		2		3		4		5
6		7		8		9		10
11		12		13		14		15

For 64-bit CM, the 12-bit channel words do not map evenly into a CM word. Three CM locations are required for 16 channel words as shown below.

0		15		31		47		63
PARCEL 0		PARCEL 1		PARCEL 2		PARCEL 3	
1	2	3	4	5	6			
6	7	8	9	10	11			
11	12	13	14	15	16			

Data Paths

The following paragraphs describe the adapter data paths for each of the transfers.

PP to T-Prime Register

The PP writes three words into the T Prime Register through Adapter Output Register Rank I and T Register Data In and Out.

PP from T Register

The PP reads three words from the T Register through Adapter Input Register Rank II.

PP to Function Register, Control Register, PP Word Counter

The PP writes these registers and counter through Adapter Output Register Rank I.

PP from Status and Control Registers

The PP reads these registers through Adapter Input Register Rank II.

Inter-PP Transfer

Data passes from one PP to another PP through Adapter Output Register Rank I and Adapter Input Register Rank II.

PP Output Transfer

Data passes from the PP to the external device through Adapter Output Register Rank I and the Transmit Register. The data also enters rank II but is not used. Data remains in the Transmit Register until the external device sends an empty signal.

PP Input Transfer

Data passes from the external device to the PP through the Deskew Register, Input Resynchronize Buffer, and Adapter Input Register Rank II. Up to eight words enter the buffer before input data stops. The adapter sends empty status to the external device until the buffer is full. When rank II accepts a word, the buffer accepts another word from the external device.

DMA Output Transfer

Data passes from CM to the external device as follows:

1. CM sends a 64-bit word to the Disassembly Buffer.

2. Disassembly Buffer separates the 64-bit word into four 16-bit parcels.

3. Conversion Network converts the 16-bit CM parcels to 12-bit channel words.
   Residue of the four bits from the first parcel is held until the second parcel arrives.
   Residue from the first parcel and eight bits from the second parcel form the second
   channel word. This process continues until all CM data is exhausted.

4. Conversion Network sends 12-bit channel words to the external device through the
   Output Resynchronize Buffer and Transmit Register.

DMA Input Transfer

Data passes from the external device to CM as follows:

1. External device sends 12-bit channel words to the Conversion Network through the
   Deskew Register and Input Resynchronize Buffer.

2. Conversion Network converts 12-bit channel words to 16-bit CM parcels. The first
   channel word is held until the second word arrives to form the first CM parcel.
   Residue of the eight bits from the second word is held until the third word arrives
   to form the second CM parcel. This process continues until all channel data is
   exhausted.

3. Conversion Network sends 16-bit CM parcels to the Assembly Buffer.

4. Assembly Buffer combines four 16-bit CM parcels into a 64-bit word and sends it to
   CM.

Index

Comments (continued from other side)

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

FOLD

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

We would like your comments on this manual to help us improve it. Please take a few minutes to fill out this form.

Who are you?	How do you use this manual?
☐ Student	☐ To learn the product or system
☐ Instructor	☐ As a training document
☐ Customer Engineer (CE)	☐ To maintain the product or system
☐ Engineer-in-Charge (EIC)	☐ To troubleshoot the product or system
☐ Technical support	☐ For installation and checkout
☐ Other _____	☐ For quick look-up
	☐ Other _____

How do you like this manual? Answer the questions that apply.

Yes Somewhat No
☐   ☐   ☐   Does it tell you what you need to know about the topic?
☐   ☐   ☐   Is the technical information accurate?
☐   ☐   ☐   Is it easy to understand?
☐   ☐   ☐   Is the order of topics logical?
☐   ☐   ☐   Can you easily find what you want?
☐   ☐   ☐   Are there enough examples?
☐   ☐   ☐   Are the examples helpful? (☐ Too simple?   ☐ Too complex?)
☐   ☐   ☐   Do the illustrations help you?
☐   ☐   ☐   Is the manual easy to read (print size, page layout, and so on)?
☐   ☐   ☐   Do you use this manual frequently?

Comments? If applicable, note page and paragraph. Use other side if needed.

Check here if you want a reply: ☐ _____

Name _____   Company _____

Address _____   Date _____

_____   Phone _____

_____

Please send program listing and output if applicable to your comment.