CONTROL DATA
CORPORATION

MSL15X MODEL INDEPENDENT TESTS MAINTENANCE SOFTWARE

REFERENCE MANUAL

PART I TEST PROCEDURES & PART II TEST DESCRIPTIONS

FIXED COMMAND TESTS (FCT1,2,3,5)
VIRTUAL MODE INSTRUCTION LEVEL TEST (FCT9)
EXCHANGE TESTS (EXCH)
TRAP TESTS (TRAP)

CDC® COMPUTER SYSTEMS:

CYBER 170 MODELS 815, 825, 835, 845, and 855

CYBER 180 MODELS 810, 830, 835, 845, and 855

CONTROL DATA
CORPORATION

MSL15X MODEL INDEPENDENT TESTS MAINTENANCE SOFTWARE

REFERENCE MANUAL

PART I TEST PROCEDURES & PART II TEST DESCRIPTIONS

FIXED COMMAND TESTS (FCT1,2,3,5)
VIRTUAL MODE INSTRUCTION LEVEL TEST (FCT9)
EXCHANGE TESTS (EXCH)
TRAP TESTS (TRAP)

CDC® COMPUTER SYSTEMS:

CYBER 170 MODELS 815, 825, 835, 845, and 855

CYBER 180 MODELS 810, 830, 835, 845, and 855

REVISION RECORD

REVISION	DESCRIPTION
A 5-82	Manual released. This manual reflects release level 137.
B 11-82	Manual revised; includes ECO PD 02900. This manual reflects release level 143. Model 815 computer system has been added. Due to extensive changes, revision bars are not used. This edition obsoletes previous one.
C 5-83	Manual revised; includes ECO PD 02979. This manual reflects release level 149. The following pages are affected: cover, ii through ix, xii, I-1-1, I-1-2, I-1-8, I-1-11, I-1-12, I-2-2, I-2-3, I-3-2, I-3-3, I-3-4, I-4-3, I-4-6, I-4-7, I-4-9, I-4-38, I-4-40, I-4-41, I-5-2, I-5-3, I-5-5, I-6-2, I-6-3, I-6-5, I-7,2, I-7-3, II-3-9, II-3-10, II-3-11, II-6-30, comment sheet.
D 11-83	Manual revised; includes ECO PD 03063. This manual reflects release level 156. The following pages are revised: cover, title page, ii through vii; in part I: 1-1, 1-2, 2-1, 2-2, 2-4, 3-1, 3-2, 4-3, 4-4, 4-6, 4-7, 4-8, 4-13, 4-15, 4-28, 4-29, 5-2, 5-3, 5-9, 5-11, 6-2 through 6-12, 7-2, 7-3, 7-4, 7-7 through 7-10, 7-13; in part II: 3-2, 3-4 through 3-10, 6-5, 6-6, 6-7, 6-24, 6-25, 6-26, 6-45, 6-46, 7-5, 7-12, 7-19, 7-21, 7-30, 7-31, 7-32, comment sheet.
E 5-84	Manual revised; includes ECO PD03109. Reflects release level 161. see List of Effective Pages for changed pages. This edition obsoletes all previous editions.

Publication no.
60469390

Revision letters I, O, Q, S, X, and Z are not used

Address comments concerning
this manual to:

Control Data Canada
Toronto Publications
1855 Minnesota Court
MISSISSAUGA, Ont.,
Canada L5N 1K7

or use Comment Sheet in
the back of this manual

LIST OF EFFECTIVE PAGES

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

PAGE	REV	PAGE	REV	PAGE	REV	PAGE	REV
Cover	E	I-2-9	A	I-4-27	E	I-6-8	E
Title page	E	I-2-10	A	I-4-28	E	I-6-9	E
2	E	I-2-11	A	I-4-29	E	I-6-10	D
3	E	Divider	A	I-4-30	E	I-6-11	E
4	E	I-3-1	E	I-4-31	E	I-6-12	E
5/6	E	I-3-2	D	I-4-32	E	I-6-13	E
7	E	I-3-3	C	I-4-33	A	I-6-14	E
8	E	I-3-4	C	I-4-34	E	I-6-15	E
9	E	I-3-5	A	I-4-35	E	I-6-16	E
10	E	I-3-6	A	I-4-36	E	I-6-17	E
11/12	E	I-3-7	A	I-4-37	A	Divider	A
Divider	E	I-3-8	A	I-4-38	C	I-7-1	A
1	E	Divider	A	I-4-39	E	I-7-2	E
2	E	I-4-1	A	I-4-40	C	I-7-3	D
3	E	I-4-2	A	I-4-41	C	I-7-4	E
Pt I Div	B	I-4-3	E	I-4-42	A	I-7-5	E
Divider	A	I-4-4	E	Divider	A	I-7-6	E
I-1-1	E	I-4-5	A	I-5-1	A	I-7-7	D
I-1-2	E	I-4-6	E	I-5-2	E	I-7-8	E
I-1-3	E	I-4-7	E	I-5-3	E	I-7-9	E
I-1-4	E	I-4-8	D	I-5-4	E	I-7-10	E
I-1-5	E	I-4-9	E	I-5-5	C	I-7-11	A
I-1-6	A	I-4-10	C	I-5-6	E	I-7-12	E
I-1-7	E	I-4-11	E	I-5-7	E	I-7-13	E
I-1-8	C	I-4-12	E	I-5-8	A	I-7-14	E
I-1-9	E	I-4-13	D	I-5-9	E	I-7-15	E
I-1-10	A	I-4-14	E	I-5-10	E	I-7-16	A
I-1-11	E	I-4-15	D	I-5-11	E	Pt II Div	B
I-1-12	E	I-4-16	A	I-5-12	E	Divider	A
I-1-13	E	I-4-17	A	I-5-13	E	II-1-1	A
Divider	A	I-4-18	A	I-5-14	E	II-1-2	A
I-2-1	E	I-4-19	A	Divider	A	II-1-3	A
I-2-2	D	I-4-20	A	I-6-1	E	II-1-4	A
I-2-3	C	I-4-21	A	I-6-2	E	Divider	A
I-2-4	D	I-4-22	A	I-6-3	E	II-2-1	A
I-2-5	B	I-4-23	A	I-6-4	D	Divider	A
I-2-6	A	I-4-24	A	I-6-5	D	II-3-1	A
I-2-7	A	I-4-25	E	I-6-6	E	II-3-2	D
I-2-8	A	I-4-26	E	I-6-7	E	II-3-3	A

PAGE	REV	PAGE	REV	PAGE	REV	PAGE	REV
II-3-4	D	II-6-17	A	II-7-20	A		
II-3-5	D	II-6-18	A	II-7-21	D		
II-3-6	D	II-6-19	A	II-7-22	A		
II-3-7	D	II-6-20	A	II-7-23	A		
II-3-8	D	II-6-21	A	II-7-24	A		
II-3-9	D	II-6-22	A	II-7-25	A		
II-3-10	D	II-6-23	A	II-7-26	A		
II-3-11	C	II-6-24	D	II-7-27	A		
Divider	A	II-6-25	D	II-7-28	A		
II-4-1	A	II-6-26	D	II-7-29	A		
II-4-2	A	II-6-27	A	II-7-30	D		
Divider	A	II-6-28	A	II-7-31	D		
II-5-1	A	II-6-29	A	II-7-32	D		
II-5-2	A	II-6-30	C	Divider	A		
II-5-3	E	II-6-31	B	A-1	A		
II-5-4	E	II-6-32	A	A-2	A		
II-5-5	E	II-6-33	A	A-3	A		
II-5-6	E	II-6-34	A	A-4	A		
II-5-7	E	II-6-35	A	A-5	A		
II-5-8	E	II-6-36	A	Divider	B		
II-5-9	E	II-6-37	A	B-1	A		
II-5-10	E	II-6-38	A	B-2	A		
II-5-11	E	II-6-39	A	B-3	A		
II-5-12	E	II-6-40	A	Divider	E		
II-5-13	E	II-6-41	A	C-1	E		
II-5-14	E	II-6-42	A	C-2	E		
II-5-15	E	II-6-43	A	C-3	E		
II-5-16	E	II-6-44	A	C-4	E		
II-5-17	E	II-6-45	D	C-5	E		
II-5-18	E	II-6-46	D	C-6	E		
II-5-19	E	Divider	A	C-7	E		
II-5-20	E	II-7-1	A	C-8	E		
II-5-21	E	II-7-2	A	C-9	E		
Divider	A	II-7-3	A	C-10	E		
II-6-1	E	II-7-4	A	C-11	E		
II-6-2	E	II-7-5	D	C-12	E		
II-6-3	A	II-7-6	A	C-13	E		
II-6-4	A	II-7-7	A	C-14	E		
II-6-5	D	II-7-8	A	C-15	E		
II-6-6	D	II-7-9	A	C-16	E		
II-6-7	D	II-7-10	A	C-17	E		
II-6-8	A	II-7-11	E	Comment Sheet	E		
II-6-9	A	II-7-12	D				
II-6-10	A	II-7-13	A				
II-6-11	A	II-7-14	A				
II-6-12	A	II-7-15	A				
II-6-13	A	II-7-16	A				
II-6-14	A	II-7-17	A				
II-6-15	A	II-7-18	A				
II-6-16	A	II-7-19	D				

PREFACE

This manual describes the tests provided on the Maintenance Software Library (MSL15X) for use on the Control Data® CYBER 170 Models 815, 825, 835, 845, and 855 CYBER 180 Models 810, 830, 835, 845, and 855 Computer Systems.

The manual is organized into two parts. Part I defines test procedures and part II describes program organization and content of each test. Within each part, the text of the manual is divided into the following sections:

1.	Memory Tests	CMEM
2.	Random Command Tests 1	RCT1
3.	Random Command Tests 2	RCT2
4.	Fixed Command Tests	FCT1,2,3,5
5.	Virtual Mode Instruction Level Test	FCT9
6.	Exchange Test	EXCH
7.	Trap Test	TRAP

RELATED PUBLICATIONS

Control Data Publication	Publication Number
MSL 15X Off-line Maintenance Software Library Reference Manual	60456530
MSL 151 Models 810, 815, 825, and 830 Maintenance Software Reference Manual	60469400
MSL 152 Models 835 Maintenance Software Reference Manual	60469410
MSL 153 Models 845/855 Maintenance Software Reference Manual	60459140
CYBER 170 Model 815/825 Hardware Reference Manual	60469350
CYBER 170/180 Model 810/830 Hardware Reference Manual	60469420
CYBER 170/180 Model 835/845/855 Hardware Reference Manual	60469290

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 or undefined parameters.

CONTENTS

SECTION 5 - VIRTUAL MODE INSTRUCTION LEVEL TEST - FCT9

SECTION 6 - EXCHANGE TEST - EXCH

SECTION 7 - TRAP TEST

PART II - MODEL INDEPENDENT TEST DESCRIPTIONS

SECTION 1 - MODEL INDEPENDENT CENTRAL MEMORY TESTS - CMEM

SECTION 2 - RANDOM COMMAND TEST 1 - RCT1

SECTION 3 - RANDOM COMMAND TEST 2 - RCT2

COMMAND BUFFERS

Clock Margins

Dual CP Systems

COMMAND BUFFERS

CLOCK MARGINS

There are no more command buffers with a suffix of N or W for running narrow and wide clock margins. Most test command buffers contain an RJ command which executes one of three small command buffers called IOMARG, CMMARG, and CPMARG. These buffers set the wide/narrow margin bit for the respective subsystem using the CMSE CM command. The operator is required to select and run one of three other small command buffers called WIDE, NARROW, and NORMAL. Each of these initiates a PP program which modifies all three previously mentioned command buffers changing the W, N, or blank flag characters of the CM command.

After executing one (only) of the WIDE, NARROW, or NORMAL buffers with the CM command the operator may run one or more test command buffers. The RJ command in the test command buffers sets the subsystem margin bit (one only) appropriate to the test.

After testing with wide or narrow margins the operator must execute the NORMAL command buffer to reset the system and the IOMARG, CMMARG, and CPMARG buffers for normal (no margins) condition.

DUAL CP SYSTEMS

Some tests for models 810, 815, 825, and 830 have more than one version and thus more than one command buffer. The majority of these command buffers are for instruction level tests as opposed to PP and microcode level tests. There are four classes of command buffers for the instruction level tests. They are identified by a suffix added to the end of the buffer name. The meaning of the suffixes is listed below:

No Suffix:	For the only CP, single CP system
Suffix A:	For CP0, dual CP system
Suffix B:	For CP1, dual CP system
Suffix C:	Concurrently for CP0 and CP1, dual CP system
Suffix S:	Sequentially for CP0 and CP1, dual CP system

Buffers for single CP systems should not be run on dual CP systems and vice versa. Additional details are provided in the following paragraphs:

1) Command buffers without a suffix are functionally the same as those on previos releases although in many cases the nomenclature for individual commands has been changed to reflect the changes introduced in CMSE for dual CP operation.

2) Suffixes A and B distingush between two sets of command buffers for instruction level tests on dual CP systems. Command buffers with the suffix A are used to run the named test on CP0 only and command buffers with the suffix B are used to run the named test on CP1 only when you want only one CP to be active. The CMSE commands HK and CF in the buffers are directed to both CPs. This ensures that the selected CP will not be interfered with when the other CP is left active by the execution of the previous command buffer.

3) Suffix C indicates that the test runs concurrently on both CPs of a dual CP system.

4) Suffix S indicates that the test runs sequentially on dual CP systems. These buffers are for tests that operate under the control of the instruction test controller (ITC). The ITC runs each section of the test first in CP0 and then in CP1. At any given time only one CP is active. Commands, listed on the test's parameter stop display, are provided to select or deselect either of the CPs if you want to test only one.

Current command buffer names are listed below.

Command Buffer Names

$$QLT10
$$PMT10
$$EXT10
$$PMU10
$$CHD10
$$CMA10
$$MRA10
$$MRT10
$$DST10
$$TPM10
$$CRA10
$$MUX10
$$FII10
$$CMT10
$$CMI10
$$FDS10
$$FIS10
$$CMEM1
$$CMEM1A
$$CMEM1B
$$FCT11
$$FCT11S
$$FCT21
$$FCT21A
$$FCT21B
$$FCT31
$$FT3G01
$$FT3G11
$$FT3G21
$$FT3G31
$$FT3FP1
$$FT3BD1
$$FT3S11
$$FT3S21
$$FCT31A
$$FT3G01A
$$FT3G11A
$$FT3G21A
$$FT3G31A
$$FT3FP1A
$$FT3BD1A
$$FT3S11A
$$FT3S21A
$$FCT31B
$$FT3G01B
$$FT3G11B
$$FT3G21B
$$FT3G31B
$$FT3FP1B
$$FT3BD1B
$$FT3S11B
$$FT3S21B
$$FCT51
$$FCT51S
$$FCT91
$$FCT91S
$$EXCH1
$$EXCH1S
$$TRAP1
$$TRAP1S
$$EXC1
$$EXC1A
$$EXC1C
$$RCT11
$$RCT11A
$$RCT11B
$$RCT21
$$RCT21A
$$RCT21B
$$RCT111C
$$RCT221C
$$RCT211C
$$RCT121C
$$C170CP
$$IOMARG
$$CMMARG
$$CPMARG
$$NORMAL
$$NARROW
$$WIDE
$$BUILD
$$CLEAR1
$$ABU
$$DEMOTC1
$$DPALL
$$SNAP
$$SNAPA
$$SNAPB
$$OFA10
$$EME

PART I

MODEL INDEPENDENT TEST PROCEDURES

MEMORY TESTS (CMEM)
RANDOM COMMAND TESTS (RCT1)
RANDOM COMMAND TESTS (RCT2)
FIXED OPERAND COMMAND TESTS (FCT1-3,5)
VIRTUAL MODE INSTRUCTION LEVEL TESTS (FCT9)
EXCHANGE TESTS (EXCH)
TRAP TESTS (TRAP)

SECTION I-1

MODEL INDEPENDENT CENTRAL MEMORY TESTS - CMEM

1 INTRODUCTION

CMEM is a CPU program that tests central memory to detect errors. The test runs with CMSE and interacts with the user via the system display device and keyboard. All necessary communication to control the test is via English language directives.

2 REQUIREMENTS

2.1 HARDWARE

This test is intended for Models 810, 815, 830, 835, 845, 850, and 855 Computer Systems.

In addition to CMSE requirements, CMEM requires the following minimum hardware to execute:

- 1 central processor
- 1 magnetic tape or mass storage library device
- 1 display/keyboard device
- 1 megabyte of CM

2.2 SOFTWARE

This test runs under control of the common maintenance software executive (CMSE). Interfaces are handled by CMSE and the virtual level executive (TLEX).

2.3 ACCESSORIES

None required.

2.4 CHARACTERISTICS

1.	Program name	CMEM
2.	Size (source)	3758 lines
3.	Size (memory required for CMEM executable code)	500,000 bytes
4.	Code type	CPU Code
5.	Run time (default)	36 sec (4MB)
6.	Run time (quick look)	N/A
7.	Run time (all sections)	36 sec (4MB)
8.	Level of isolation	Detect Only
9.	Off-line test	Yes
10.	Off-line system	CMSE (MSL 15X)
11.	Assembly language	META
12.	Source code maintenance	UPDATE

3 OPERATIONAL PROCEDURE

3.1 RESTRICTIONS AND USER CAUTIONS

None.

3.2 LOADING PROCEDURE

The program CMEM is assembled using the virtual machine assembler. The binary output is linked to the binary of TLEX to form the complete executable module.

Load the microcode before executing the test.

Command buffer CMEMx (1 for model 810/815/825/830, 2 for 835, 3 for 845/850/855/860) exists on the MSL 15X tape to facilitate loading and execution of this test.

The test executes upon completion of the load. You can change parameters via the CMEM command set or the CMSE commands EC and EB. Parameter addresses are displayed in word format during test initialization or at any time by typing LIST.

Refer to the MSL 15X Reference Manual for descriptions of CMSE commands.

The CMSE commands plus any CMEM program commands may be saved and reexecuted via the CMSE command buffer capability.

3.2.1 Running Procedures

During normal execution of pass 1, the test displays the parameter words and their real memory word addresses. PARAM3 contains the available memory size in megabytes ($1\text{-}10_{16}$). For example, 1 equals 1, A equals 10, 10 equals 16 megabytes. The default is obtained from the memory Options Installed register (OI-Register 12).

Use the EC command to enter available memory in megabytes. It is displayed with the parameter word display.

NOTE

> The operator does not have to supply any parameters to CMEM. The test will automatically calculate the addresses available and test them. However, if the operator does want to select a different testing area, use the ADDR command.

Press space bar twice to start execution. The default is to run all sections.

On a test restart, the test uses addresses from the previous pass unless you change test addresses using the ADDR command. Type LIST at any time for a parameter display. Type ADDR,p1,p2 to change testing addresses.

NOTE

Once section execution has begun, you can enter the ADDR command at any time, but actual changes to testing addresses will be on section entry only (i.e., beginning, repeat, upon entering scope loop).

3.3 PARAMETERS AND CONTROL WORDS

3.3.1 Parameters

Tag	Word/Byte Address Hexadecimal	Bits	Description	Default
PARAM0	F009/78048	0-40	Not used.	200051
		41	Telex monitor default run	
		42	Telex bypass all options	
		43	Telex test mode 1 (BASIC)	
		44	Telex test mode 2 (PIT and SIT)	
		45	Batch mode	
		46	On-line control	
		47	Off-line control	
		48	Operator stop	
		49	Initial stop only for parameters	
		51	Bypass all messages.	(DR)
		52	Quick look flag.	
		53	Scope mode.	(SM)
		54	Repeat condition.	
		55	Repeat subsection.	(RB)
		56	Repeat section.	(RS)
		57	Repeat test (set by default).	(RT)
		58	Log error in dayfile.	(LE)
		59	Stop on error (set by default).	(SE)

		60	Stop on condition.	
		61	Stop at end of subsection.	(SB)
		62	Stop at end of section.	(SS)
		63	Stop at end of test (setby default).	(ST)
PARAM1	F00A/78050	0-63	Repeat counter. Test automatically repeats the number of times contained in this count. Overrides repeat test bit (bit 57 in PARAM0).	0000
PARAM2	F00B/78058	0-63	Section select bits for sections 63-0. Note that only sections 1-11 exist. Section select bits are numbered from right to left.	00FF
PARAM3	F00C/78060	0-63	Memory size available.	0004
PARAM4	F00D/78068	0-63	When set to nonzero (and stop on error set) and single bit error occurs, test displays a message stating either CORRECTED MEMORY ERROR or CORRECTED PROCESSOR ERROR depending on the error. If PARAM4 is left equal to zero, test does not display the message.	0000
INBUFF	E116/70B30	Input buffer		
COMPFLAG	E152/70A90	Data compare flag. Zero equals no compare, nonzero equals compare data.		
PATTERN	E160/70B00	Testing pattern		
FWA	E158/70AC0	First word address to test (RMA)		
LWA	E159/70AC8	Last word address to test (RMA)		
TFPVA	E15C/70AE0	PVA of FWA		
TLPVA	E15D/70AE8	PVA of LWA		
LAST	E157/70AB8	Last possible test address (determined by size of memory available - PARAM3)		
TRAPMASK	E168/70B40	Mask for CMEM trap handler		
HALTMASK	E169/70B48	Mask for VLEX error handler		

3.3.2 Control Words

Tag	Word/Byte Address Hexadecimal	Description
CW0	F000/78000	Program name/type/monitor identification
CW3	F001/78008	Dynamic error counter
CW4	F002/78010	Pass counter
CW5	F003/78018	Current section executing
CW6	F004/78020	Current subsection
CW7	F005/78028	Current condition

3.4 SECTION INDEX

Section Number	Tag Name	Brief Description
1	SECT1	Test selected area of memory with an all zeros pattern.
2	SECT2	Test selected area of memory with an all ones pattern.
3	SECT3	Test selected area of memory with a 5 pattern (example: $0101\text{-}0101_2$).
4	SECT4	Test selected area of memory with an A pattern (example: $1010\text{-}1010_2$).
5	SECT5	Test selected area of memory with a DF pattern (example: $1101\ 1111\text{-}1101\ 1111_2$).
6	SECT6	Test selected area of memory with a marching pattern.
7	SECT7	Write and read address patterns where memory word address contains the address as data (example: address $3FDB_{16}$ = $00003FDB00003FDB_{16}$).
8	SECT8	Write and read address patterns where memory address contains the complement of the address as data (example: address $3FDB_{16}$ = $FFFFC024FFFFC024_{16}$).
9	SECT9	Test selected area of memory with a 5-byte pattern (DFDFDFDFDF).
10	SECTA	Test selected area of memory in 16-word blocks (SMULT,LMULT).
11	SECTB	Test selected area of memory using random generated pattern.

4 OPERATOR COMMUNICATION

4.1 DISPLAYS

Unless otherwise specified, displayed values are in decimal.

4.1.1 Running Display

Normal test execution produces the following message:

```
                                                    Line
┌──────────────────────────────────────────────┐
│              T E S T  C M E M                │    1
│ *ST SS SB SC *SE LE *RT RS RB RC SM QL DR DE │    2
│ CMEM (OP) PCxxx  Sxxxx SBxxxx Cxxxx          │    3
│ FWA = xxxxxxxx   LWA = xxxxxxxx              │    4
│                                              │    5
│                                              │    6
│                                              │    7
└──────────────────────────────────────────────┘    8
```

where:

Line 2 An asterisk indicates PARAM0 bits 50-63 that are active (i.e., *SE means stop on error selected).

Available options are activated or deactivated by the following test commands.

Option	Set/Clear	Option	Set/Clear
ST	SST/CST	RS	SRS/CRS
SS	SSS/CSS	RB	SRB/CRB
SB	SSB/CSB	DR	SDR/CDR
SE	SSE/CSE	SM	SSM
RT	SRT/CRT	LE	SLE/CLE

Line 3

CMEM	Test name	
(OP)	Operation	
	RU	Running
	SB	Stop at end of subsection
	SS	Stop at end of section
	SE	Stop on error
	PCxxxx	Pass count
	Sxxxx	Section
	SBxxxx	Subsection
	Cxxxx	Condition (not implemented)

Line 4

FWA	First word address of testing area (hexadecimal)
LWA	Last word address of testing area (hexadecimal)

4.1.2 Set Parameters Display

The test command LIST produces the following message:

```
                                                    Line
 CMEM   SET PARAMS   PA = xxxxxx                     3
                                                     4
 ADDR = xxxxxx     PARAM0 = xxxxxx                   5
 ADDR = xxxxxx     PARAM1 = xxxxxx                   6
 ADDR = xxxxxx     PARAM2 = xxxxxx                   7
 ADDR = xxxxxx     PARAM3 = xxxxxx                   8
 ADDR = xxxxxx     PARAM4 = xxxxxx                   9
                                                    10
 MEMORY AVAILABLE FOR TESTING                       11
   XXXXXXXX - YYYYYYYY                              12
 ENTER ADDR, FWA, LWA TO CHANGE TESTING AREA        13
```

where:

PA — Real memory word address of parameter area (hexadecimal).

ADDR — Each parameter word value and its real memory word address in hexadecimal.

Memory available for testing depends on size of memory available. Selection of testing addresses outside of displayed addresses will result in an error message being displayed. XXXXXXXX equals FWA available, YYYYYYYY equals LWA available. Both FWA and LWA are hexadecimal.

4.1.3 HELP Display

The test command HELP produces the following display:

```
                                                                Line
ADDR,P1,P2      SET WORD ADDRESS LIMITS OF MEMORY TESTING         3
                AREA.  P1=FWA, P2=LWA P1 AND P2 ARE              4
                HEXADECIMAL ONLY.                                5
                                                                 6
(SPACE)         START/CONTINUE EXECUTION OF TEST                 7
                                                                 8
R               RESTART TEST FROM BEGINNING                      9
                                                                10
S               STOP EXECUTION OF TEST                          11
                                                                12
D               DROP TEST                                       13
                                                                14
HELP            DISPLAY CURRENT TEST DIRECTIVES                 15
                                                                16
DATAON          ENABLE DATA COMPARE PROCESS                     17
                                                                18
DATAOFF         DISABLE DATA COMPARE PROCESS                    19
                                                                20
SSM             SET SCOPE MODE                                  21
                                                                22
LIST            DISPLAY PARAMETER INFORMATION                   23
```

4.1.4 Error Message Display

When CMEM detects a data error, the following standard error message (lines 3 and 4) is displayed. Depending upon the type of error (double bit or single bit) additional error information and messages are displayed in lines 6, 7, and 10. See paragraph titled, Error Messages for definitions of these messages.

```
| CMEM  SE  PCxxxx  Sxxxx  SBxxxx  Cxxxx          |
| EC1 = xxxx  EC2 = xxxx  TE = xxxx               |
```

where:

SE Stopped on error

PCxxxx Pass count

Sxxxx Failing section number

SBxxxx Failing subsection number

Cxxxx Failing condition number (not implemented)

EC1 Not implemented

EC2 Not implemented

TE Total error count

4.2 OPERATOR ENTRIES

CMEM uses the standard CMSE commands as described in the MSL 15X Reference Manual. For convenience, CMEM also provides the following parameter and run directives.

4.2.1 Parameter Directives

The following parameter directives enable program section selection and control. These directives may be dynamically entered via the console keyboard with the test running and in control of the keyboard and display. These directives may also be built into a command buffer.

Directive	Function
LIST	Displays the current test statistics. See 4.1 Displays for definition of the LIST display.
HELP	Displays all currently available directives and a short function description.

DATAON	Enables software data checking of data written versus the data that was read. Does not apply to sections 6 through 11.
DATAOFF	Disables data checking. Does not apply to sections 6 through 11.
ADDR,p1,p2	Sets word address limits of the memory testing area. Parameter p1 is the first word address. Parameter p2 is the last word address. These parameters are hexadecimal only. The directive applies to all sections.
R	Restarts test from the beginning at any point during execution.
D	Drops CMEM at any point during execution.
(space)	Continues a stopped section, subsection or condition at point that it stopped.
S	Stops a test section that is currently running. The (space) directive continues test from where stopped.
SSM	Causes a continuous looping condition on the section being executed; used mainly for scoping purposes.

4.2.2 Run Directives

The following directives are under control of CMEM, not CMSE.

Directive	Function
SST/CST	Set/clear stop at end of test.
SSS/CSS	Set/clear stop at end of section.
SSB/CSB	Set/clear stop at end of subsection.
SSE/CSE	Set/clear stop on error.
SLE/CLE	Set/clear log errors.
SRT/CRT	Set/clear repeat test.
SRS/CRS	Set/clear repeat section.
SRB/CRB	Set/clear repeat subsection.
SDR/CDR	Set/clear bypass all messages.
SSM	Set scope mode.

4.3 NORMAL MESSAGES

Other than the running display, the set parameters display and the HELP display, described previously, CMEM does not display any test messages for normal operation.

4.4 ERROR MESSAGES

Depending upon the type of error detected, data or command entry error, CMEM displays an error message and data on lines 6, 7, and 10 of the error display or on line 15 of any current display.

4.4.1 Data Error Messages

CMEM detects data errors and displays error messages under two separate conditions: double-bit or single-bit.

Double-Bit Errors:

When CMEM detects a double-bit error, it displays the standard error display on lines 3 and 4, informational data on lines 6 and 7, and an error message on line 10. The following figure shows the format of a double-bit error message.

```
                                                        Line
 CMEM  SE  PCxxxx  Sxxxx  SBxxxx  Cxxxx                   3
 EC1 = xxxx  EC2 = xxxx  TE = xxxx                        4
                                                          5
   ADRS          EXP              RCV                     6
 xxxxxxxx   xxxxxxxxxxxxxxxx   DOUBLE BIT ERROR           7
                                                          8
                                                          9
 Error Message                                            10
```

where:

ADRS	Address of word in error (hexadecimal)
EXP	Expected result (hexadecimal)
RCV	DOUBLE-BIT ERROR. This message indicates that a double-bit error occurred. If a double-bit error occurs when the CPU is reading memory, CMEM cannot retrieve and display the data.
Error Message	Depending on the type of double-bit error, one of the following messages appears on line 10.

MESSAGE	EXPLANATION
READ-UNCORRECTED ERROR	Detected double-bit error in memory on a read function.
WRITE-UNCORRECTED ERROR	Detected double-bit error in memory on a write function.
REJECT-UNCORRECTED ERROR	Detected double-bit error in memory on a reject function.
PROCESSOR-UNCORRECTED ERROR	Detected double-bit error in processor.
CM TAG PARITY ERROR	Detected parity error in tag.
CM RESPONSE CODE PARITY ERR	Detected parity error in response code.

When the test is halted for an error, you can display the bad data by entering the CMSE command AH,adrs with adrs equal to the ADRS displayed in the error message. CMSE uses the PP to read memory and displays the bad data.

Single-Bit Errors:

When CMEM detects a single bit error and PARAM4 is nonzero, CMEM displays the standard error message on lines 3 and 4 and one of the following messages on line 10.

MESSAGE	EXPLANATION
CORRECTED PROCESSOR ERROR	Detected single-bit error in processor.
CORRECTED MEMORY ERROR AT ADDRESS XXXXXXXX	Detected single-bit error in memory at address XXXXXXXX (hexadecimal).

4.4.2 Command Entry Error Messages

One of the following messages is displayed on line 18 of the current display when a command is entered improperly.

MESSAGE	EXPLANATION
INVALID COMMAND	The operator entered an invalid command.
PAGE UNAVAILABLE FOR RMA XXXXXXXX CHANGE TESTING ADDRESS LIMITS	Initialization error. The page associated with RMA cannot be assigned.
FWA TOO SMALL	FWA not within accepted range.
FWA TOO LARGE	FWA not within accepted range.
LWA TOO SMALL	LWA not within accepted range.
LWA TOO LARGE	LWA not within accepted range.

INVALID HEXADECIMAL CHARACTER	Address in command is incorrect.
INVALID MEMORY SIZE	Memory size specified is incorrect.

4.5 APPLICATIONS

4.5.1 Loop-On-Error Condition

On any stop condition, loop the current condition (failure) with the following commands via the keyboard console:

Command	Result
SRB	Sets repeat subsection.
(space)	Enters loop of subsection with failure condition.

Loop the current condition (no failure):

If stopped at end of section:

SRS	Sets repeat section.
(space)	Enters loop of section (runs all subsections in section).

If stopped at end of subsection:

SRB	Sets repeat subsection.
(space)	Enters loop of that subsection.

4.5.2 Scope Mode

Use the SSM command to set scope mode on the section being run. The loop will continue until you clear scope mode bit in parameter word 0; you must use a CMSE command since the test does not monitor the keyboard while in scope mode. To clear scope mode, issue the following command:

EC,base,adrs,data

Where:

base	H for hexadecimal mode O for octal
adrs	RMA to be entered
data	Right justified data 1-16 hexadecimal characters

NOTE

Scope mode does not apply to section 11.

4.5.3 Additional Running Procedures

The following are minimum commands needed to enable test execution:

Command	Result
Space bar	Start test initialization. Memory size is automatically calculated.
Space bar	Starts test execution. Default is to run all sections.

SECTION I-2

RANDOM COMMAND TEST 1 - RCT1

1 INTRODUCTION

RCT1 is a set of model independent random command tests which test general and floating point virtual instructions. RCT1 uses random instruction sequences and random operands to isolate to a single failing instruction within a failing random sequence of instructions.

RCT1 resides on the Maintenance Software Library and is loaded by CMSE. The load module includes the object code of the virtual level executive VLEX, which is required to establish the CPU execution and environment and to facilitate communication with CMSE.

The test interfaces with the user via the system display device and keyboard. All necessary communication to control the test is via CMSE commands and English language test directives.

2 REQUIREMENTS

2.1 HARDWARE

This test is intended for Models 810, 815, 825, 830, 835, 845, and 855 Computer Systems.

In addition to hardware required for CMSE, RCT1 requires the following to execute:

1 CPU
1 megabyte of central memory

2.2 SOFTWARE

This test runs under control of the common maintenance software executive (CMSE). Interfaces are handled by CMSE and the virtual level executive (VLEX).

2.3 ACCESSORIES

None required.

2.4 CHARACTERISTICS

1.	Program Name	RCT1
2.	Size (source)	4,509 lines
3.	Size (memory required)	114,688 bytes
4.	Code type	CPU
5.	Run time (default)	*
6.	Run time (quick look)	N/A
7.	Run time (all sections)	*
8.	Level of isolation	detect only
9.	Off-line test	yes
10.	Off-line system	CMSE (MSL 15X)
11.	Assembly language	META
12.	Source code maintenance	UPDATE

3 OPERATIONAL PROCEDURE

3.1 RESTRICTIONS AND USER CAUTIONS

RCT1 is highly dependent on the correct execution of its instruction usage set. Therefore, in a troubleshooting environment FCT2 must run successfully before running RCT1.

3.2 LOADING PROCEDURE

RCT1 is assembled using the virtual machine assembler. The binary output is linked to the assembled binary of VLEX to form the complete executable module. It can then be written to disk using TDX.

Load the microcode before executing the test.

Command buffer RCT1x (where x is 1 for model 810/815/825/830, 2 for 835, 3 for 845/855) exists on the MSL 15X tape to facilitate loading and execution of this test. Before using this command buffer you may have to modify it for your system. Display the command buffer using the CMSE command buffer display commands or print the contents of the command buffer using procedures provided in the Command Buffer Maintenance section of the MSL 15X Reference Manual. Then modify the command buffer as directed by comments embedded in the command buffer. When you are satisfied that the command buffer is set up properly, enter a GO,RCT1x command to execute it, where x is the model dependent number from above.

Modified command buffers can be saved on a back-up tape for future use. Refer to the Utilities section of the MSL 15X Reference Manual for procedures.

The test executes upon completion of the RCT1 load phase. Parameter changes can be made via the RCT1 command set or the CMSE commands, EC and EB. The parameter area address will be displayed in word format during test initialization.

---

*Runs until halted by the operator.

3.3 PARAMETERS AND CONTROL WORDS

3.3.1 Parameters

Parameter bits can be altered by CMSE EC or EB commands or by RCT1 commands. Refer to the MSL 15X Reference Manual for descriptions of CMSE commands. The RCT1 commands are described in detail under Operator Entries.

The following parameters are defined for RCT1. The CMSE or RCT1 command which alters a parameter is indicated in parentheses.

Parameter Word Zero

Bit(s)	Meaning
63	Stop at end of test pass (SST/CST).Set by default.
60-62	Not used.
59	Stop on error (SSE/CSE). Set by default.
58	Not used.
57	Repeat present instruction/operand set (SRT/CRT). Cleared by default.
52-56	Not used.
51	Bypass all messages (CMSE EC command). Cleared by default.
50	Display only error messages (CMSE EC command). Cleared by default.
49	Not used.
48	Operator stop requested (informational only).
47	End of test pass (informational only).
44-46	Not used.
43	An error has occurred (informational only).
42	A-Register error (informational only).
41	X-Register error (informational only).
40	Memory buffer error (informational only).
0-39	Not used.

The CM word address of parameter word zero is displayed in the RCT1 header display. The yyy portion of PA=000yyy gives the CM word address.

As there are no test sections in RCT1, there is no need for section select parameter words one and two.

Parameter Word Three (DS,x)

Bit (60-62) - Display select

0 = Null display

1 = RCT1 header/parameter display (selected by default)

2 = A register result display

3 = X register result display

4 = Memory buffer result display (first half)

5 = Memory buffer result display (second half)

6 = Initial A and X registers

7 = Instruction list

8 = Initial input memory buffer

9 = Initial output memory buffer

Skip Pass Parameter (P,x)

Skips forward to the specified pass count prior to test execution. Default value is zero.

Instruction Select Parameter (NORM/SEL/OMIT)

Allows a single instruction or a subset of instructions to be tested or deleted from the test. The default mode (NORM) is to test all instructions. Select mode (SEL) tests only those instructions listed in the optional test list. Omit mode (OMIT) tests all instructions except those listed in the optional test list. Default selection is NORM.

Optional Instruction List (ADD,x,y,.../DEL,x,y,...)

Before adding or deleting select either SEL or OMIT, above.

A user-selected list of opcodes used in building the random instruction list when not running in default mode. Default is no entries.

Test Length Parameter (L,x)

Selects the number of words of random instructions to be tested. Values may range from 1 to $1F_{16}$. Default value is seven.

Trim Mode Parameter (TRM/NTRM)

Trim mode (TRM) isolates to a single failing instruction when an error is detected. Deselecting trim mode (NTRM) will leave a failing random instruction loop unaltered when an error is detected. Default selection is TRM.

Mask Parameter (MSK/NMSK)

Mask select mode (MSK) sets the floating point condition bits (overflow, underflow, loss of significance, indefinite) in the user condition mask. No mask mode (NMSK) clears the floating point condition bits. RCT1 correctly simulates the results with either condition selected. Default selection is NMSK.

Loop Parameter (LOOP)

Loop mode (LOOP) causes the test to continuously machine execute the present instruction/operand set with no communication with the operator. To regain control, the parameter must be cleared with an EC command. Default is loop mode off.

3.3.2 Other Locations of Interest

The following areas of RCT1 contain various items of interest when an error has been detected. The first location of each area is provided in the form of assembly tag, real memory byte address, and real memory word address. All address values are hexadecimal.

	Real Memory Address		
Tag	Byte	Word	Notes
V_IAR	17208	2E41	DS,6 (DS,x bring up display)
V_IXR	17288	2E51	DS,6
B_SRC	16E08	2DC1	DS,8
B_DST	17008	2E01	DS,9
V_MAR	17408	2E81	DS,2
V_MXR	17488	2E91	DS,3
B_MCH	16F08	2DE1	DS,4 and DS,5
V_SAR	17308	2E61	DS,2
V_SXR	17388	2E71	DS,3
B_SIM	17108	2E21	DS,4 and DS,5
V_SIL	17588	2E61	Instructions that were simulated.
I_LIST	169D0	2D3A	DS,7
P_IL	16B00	2D60	Optional test list of instructions.
P_LOERR	16080	2C10	Scope loop select parameter.

3.4 SECTION INDEX

RCT1 has no selectable test sections.

4 OPERATOR COMMUNICATION

4.1 DISPLAYS

Nine RCT1 displays are defined. A header display presents the test name, pass count, parameter word address, comment field, and current parameter word settings. Four displays present the actual and simulated results for the A registers, X registers, and memory buffer, plus an exclusive OR comparison of each. Four displays present the initial content of the A and X registers, input memory buffer, output memory buffer, and the instruction list being executed. An additional NULL display is defined which, when selected, will not present a display, resulting in faster test execution.

All values in displays are shown in hexadecimal notation.

Header/Parameter Display

When RCT1 begins execution, or when a DS,1 command is entered, RCT1 presents a header/parameter display with the following format:

```
RCT1(op) PC=xxxxxx PA=000yyy (comment field)
EC1=0000 EC2=0000 TE=xxxx RN=0000
PARAM0 = 0000 0000 pppp pppp
PARAM1 = 0000 0000 0000 0000
PARAM2 = 0000 0000 0000 0000
PARAM3 = 0000 0000 0000 00dd
PARAM4 = 0000 0000 0000 0000
PARAM5 = 0000 0000 0000 0000
```

Where:

(op) Operation

- RU Running message. RCT1 is executing random instructions/operands.
- SP Parameter stop, operator stop, or end of test stop.
- SE Error stop.
- RT Repeating set of instructions/operands.
- SM Scope loop mode is selected. Entry of a space will put RCT1 into a scope loop executing the present machine set of instructions and operands.

- PC Pass count.
- PA Parameter word 0 real memory word address.

Comment Field	32-character test message
EC1	Not implemented.
EC2	Not implemented.
TE	Error count.
RN	Not implemented.
pppp(PARAM0)	Bits 40-63 of parameter word 0.
dd(PARAM3)	Display select bits defined for parameter word 3.

All other PARAMs are not used.

Register/Memory Buffer Displays

Four displays are available for presenting error information. Thus the actual and simulated results of a test pass can be displayed. The displays present the A-registers, X-registers, R/W buffer (first half), and R/W buffer (second half). The formats of displays are similar except for the register identifier at the left of each line. All values are in hexadecimal.

	EXPECTED		ACTUAL		DIFFERENCE	
REGID0	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx
REGID1	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx
REGID2	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx
REGID3	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx
REGID4	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx
REGID5	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx
REGID6	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx
REGID7	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx
REGID8	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx
REGID9	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx
REGIDA	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx
REGIDB	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx
REGIDC	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx
REGIDD	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx
REGIDE	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx
REGIDF	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx

Where:

REGID
- Register A for display 2
- Register X for display 3
- Memory 0 for display 4
- Memory 1 for display 5

4.2 OPERATOR ENTRIES

4.2.1 Commands

The following RCT1 commands are provided:

Command	Function
ADD,x,y,...	Add opcodes x,y,... to instruction list. Opcodes 00-FF are accepted.
DEL,x,y,...	Delete opcodes x,y,... from instruction list. Opcodes 00-FF are accepted. An asterisk (*) will delete all opcodes.
P,x	Skip forward to pass x. Values from 0-FFFFFF are accepted.
L,x	Set execution list length to x words. Values from $1\text{-}1F_{16}$ are accepted.

DS,x	Select Display x. Display selections are 0-9. 0 = null display 1 = header/parameter display 2 = A-register display 3 = X-register display 4 = first half of result memory buffer 5 = second half of result memory buffer 6 = initial A and X registers 7 = instruction list 8 = initial input memory buffer 9 = initial output memory buffer
MSK	Select floating point bits in user condition mask.
NMSK	Deselect floating point bits in user condition mask.
TRM	Trim execution list to failing instruction.
NTRM	Deselect execution list trimming.
NORM	Test all instructions.
OMIT	Omit instruction list opcodes from test list.
SEL	Test only instruction list opcodes.
R	Restart with initial start (DK,1A2) parameters.
S	Stop test execution.
(space)	Continue test execution.
LOOP	Continuously machine execute present instruction/operand set (scope loop).
SST/CST	Set/clear stop at end of test parameter bit.
SRT/CRT	Set/clear repeat test parameter bit.
SSE/CSE	Set/clear stop on error parameter bit.

Each command is checked for proper syntax and parameters are compared to limits prior to command execution. Refer to Error Messages for messages which will result from improper entry of these commands.

4.2.2 Running Procedures

Load and Run RCT1

Type in the CMSE commands listed for the loading procedure or make a CMSE command buffer of the commands and call the command buffer with a GO,xxxx; where xxxx is the name of the command buffer.

Change Test Parameters

Test parameters can be changed prior to starting RCT1 execution by using the CMSE EC and/or EB commands, or at any time RCT1 is running or has stopped using the RCT1 commands or the CMSE EC and/or EB commands.

When the stop-at-end-of-test-parameter bit is set, RCT1 stops before the first pass to allow parameter changes. Entry of a space will resume RCT1 execution. Type in CST to clear the stop-at-end-of-test-parameter bit and allow nonstop execution.

Refer to Parameter Words for a list of RCT1 parameters. Refer to Operator Entries for RCT1 keyboard commands required to change the parameters.

The parameter settings at the time RCT1 is initially started (DK,1A2) are saved and will be reinstated whenever RCT1 is restarted by typing an R command.

Stop on Error

Type in SSE to set the stop-on-error parameter bit. RCT will then stop whenever an error is detected.

Type in CSE to clear the stop-on-error parameter bit. RCT1 will not stop when errors are detected, but will increment the error count if errors are detected.

Type in a space to resume RCT1 execution whenever it is stopped.

Loop on Failing Set of Instructions/Operands

After RCT1 has stopped on an error, entry of an SRT command will set the repeat test parameter bit. Entry of a space will cause RCT1 to execute the set of instructions/operands for the failing pass both with simulator execution and machine execution.

Type in CRT to clear the repeat test parameter bit.

Note that during repeat test operation, the pass count will be incremented each pass even though the instructions and operands are not changed. The error count will also be incremented whenever an error is detected.

Scope Loop on a Failing Set of Instructions/Operands

After RCT1 has stopped on an error, type in an SRT command to set the repeat test parameter bit and then type in LOOP to set the scope loop parameter. Press the space key and RCT1 will continually execute the failing set of instructions/ operands.

RCT1 does not communicate with the operator during scope loop mode.

For scope triggering, a scope loop sync instruction is executed just prior to each machine execution of the instructions/ operands.

To remove RCT1 from scope loop mode and reinstate communication with the operator, clear CM word location 2C10 with the CMSE instruction EC,2C10,0.

Stop Test After Executing One Set of Random Instructions/ Operands

Type in the SST command to set the stop-at-end-of-test-parameter bit. RCT1 will stop after one pass of executing one set of random instructions/operands. Type a space to cause RCT1 to execute the next pass and stop.

Type CST to clear the stop-at-end-of-test-parameter bit.

Display Parameter or Error Information

Type in DS,x to direct RCT1 to display one of numerous information messages. Refer to Displays and to Operator Entries, paragraphs 4.1 and 4.2, for display formats and for descriptions of the x parameter.

Additional useful information can be obtained by displaying CM contents via the CMSE AB and AH commands. Refer to Parameter Words, paragraph 3.3.1 for the real memory addresses of locations of interest.

Stop Test While It Is Executing

Type in an S to stop RCT1 execution of random instructions/operands.

Type in a space to resume execution.

4.3 NORMAL MESSAGES

None provided.

4.4 ERROR MESSAGES

RCT1 will display the following messages in the comment field of the header/ parameter display when an RCT1 command is entered incorrectly. Messages will be cleared by entering a valid command.

Message	Explanation
INVALID ENTRY	RCT1 does not recognize the command entered.
INST LIST FULL	The instructions added using the ADD command exceed the instruction buffer.
CANNOT TEST xx	RCT1 cannot simulate the instruction identified by the hexadecimal value xx.

SECTION I-3

RANDOM COMMAND TEST 2 - RCT2

1 INTRODUCTION

RCT2 is a set of model independent random command tests which test BDP virtual instructions. RCT2 uses random instruction sequences and random operands to isolate to a single failing instruction within a failing random sequence of instructions.

RCT2 resides on the Maintenance Software Library and is loaded by CMSE. The load module includes the object code of the virtual level executive VLEX, which is required to establish the CPU execution and environment and to facilitate communication with CMSE.

The test interfaces with the user via the system display device and keyboard. All necessary communication to control the test is via CMSE commands and English language test directives.

2 REQUIREMENTS

2.1 HARDWARE

This test is intended for Models 810, 815, 825, 830, 835, 845, and 855 Computer Systems.

In addition to hardware required for CMSE, RCT2 requires the following to execute:

1	CPU
1	megabyte of central memory

2.2 SOFTWARE

This test runs under control of the common maintenance software executive (CMSE). Interfaces are handled by CMSE and the virtual level executive (VLEX).

2.3 ACCESSORIES

None required.

2.4 CHARACTERISTICS

1.	Program name	RCT2
2.	Size (source)	6,007 lines
3.	Size (memory required)	122,880 bytes
4.	Code type	CPU
5.	Run time (default)	*
6.	Run time (quick look)	N/A
7.	Run time (all sections)	*
8.	Level of isolation	detect only
9.	Off-line test	yes
10.	Off-line system	CMSE (MSL 15X)
11.	Assembly language	CPAS180
12.	Source code maintenance	UPDATE

3 OPERATIONAL PROCEDURE

3.1 RESTRICTIONS AND USER CAUTIONS

RCT2 is highly dependent on the correct execution of its instruction usage set. Therefore, in a troubleshooting environment FCT2 and RCT1 must run successfully before running RCT2.

3.2 LOADING PROCEDURE

RCT2 is assembled using the virtual machine assembler. The binary output is linked to the assembled binary of VLEX to form the complete executable module. It can then be written to disk using TDX.

Load the microcode before executing the test.

Command buffer RCT2x (where x is 1 for model 810/815/825/830, 2 for 835, 3 for 845/855) exists on the MSL 15X tape to facilitate loading and execution of this test. Before using this command buffer you may have to modify it for your system. Display the command buffer using the CMSE command buffer display commands or print the contents of the command buffer using procedures provided in the Command Buffer Maintenance section of the MSL 15X Reference Manual. Then modify the command buffer as directed by comments embedded in the command buffer. When you are satisfied that the command buffer is set up properly, enter a GO,RCT2x command to execute it, where x is the model dependent number from above.

Modified command buffers can be saved on a back-up tape for future use. Refer to the Utilities section of the MSL 15X Reference Manual for procedures.

The test executes upon completion of the RCT2 load phase. Parameter changes can be made via the RCT2 command set or the CMSE commands, EC and EB. The parameter area address will be displayed in word format during test initialization.

*Runs until halted by the operator.

3.3 PARAMETERS AND CONTROL WORDS

3.3.1 Parameters

Parameter bits can be altered by CMSE EC or EB commands or by RCT2 commands. Refer to the MSL 15X Reference Manual for descriptions of CMSE commands. The RCT2 commands are described in detail under Operator Entries.

The following parameters are defined for RCT2. The CMSE or RCT2 command which alters a parameter is indicated in parentheses.

Parameter Word Zero

Bit 63	Stop at end of test pass (SST/CST). Set by default.
Bit 62	Not used.
Bit 61	Stop at end of subsection (SSB/CSB). Cleared by default.
Bit 60	Not used.
Bit 59	Stop on error (SSE/CSE). Set by default.
Bit 58	Not used.
Bit 57	Repeat test (SRT/CRT). Cleared by default.
Bit 56	Repeat section (SRS/CRS). Cleared by default.
Bit 55	Repeat subsection present instruction/operand (SRB/CRB). Cleared by default.
Bits 52-54	Not used.
Bit 51	Bypass all messages (CMSE EC command). Cleared by default.
Bits 47-50	Not used.
Bits 0-46	Not used.

Parameter Word One

Bits 0-63	Repeat counter. Default value is zero. The test will automatically repeat by the number of times contained in this count. The test will repeat until this count equals zero and then unconditionally exit to the end if the repeat bit is not set.

Parameter Word Two

Bits 60-63	Section select bits. Default value is 0007; all sections selected. Only section 0-2 exist and are numbered from right to left for consistency with the IOU section select bits.
Bits 0-59	Not used.

Parameter Word Three

Bits 32-63 Subsection select bits. Default is 073F7F; all subsections selected.
Subsection select bits are numbered from right to left in the following format:

DDCCBBAA

AA = section 0, subsections 0 through 6 select bits.
BB = section 1, subsections 0 through 5 select bits.
CC = section 2, subsections 0 through 2 select bits.

If any section is deselected (section select bit in parameter word two equals 0) the subsection select bits for that section have no meaning.

3.3.2 Other Locations of Interest

Tag	Address (Hex Word)	Description
I_MLST0	2C00	Machine execution list
V_SDF0	2C45	Simulator destination field buffer
V_MDF0	2C13	Machine destination field buffer
P_LOERR	3A44	Flag for scope loop of subsections
V_PASS	2E0E	Test pass count
HALTMASK	2E1B	Mask for test halt conditions

3.4 SECTION INDEX

Section Number	Tag Name	Brief Description
0	SEC0	Test BDP numeric instructions
1	SEC1	Test BDP byte instructions
2	SEC3	Test BDP immediate data instructions

NOTE

Due to test restructuring, section 2 has been tagged SEC3 and S3.

4 OPERATOR COMMUNICATION

4.1 DISPLAYS

All values shown in displays are in hexadecimal.

4.1.1 Running Display

During test execution, RCT2 presents the following display.

```
RCT2  (op)  PCxxxxxxxx  Sxxxx  SBxxxx
EC1=0000  EC2=0000  TE=xxxx
```

where:

(op) Operation

RU Running section/subsection
RB Test is repeating a subsection/test is in a scope loop on subsection

PCxxxxxxxx Current pass

Sxxxx Section executing

SBxxxx Subsection executing

EC1 Not implemented

EC2 Not implemented

TE Total errors

4.1.2 Set Parameters Display

When the CPU begins execution of RCT2, the following parameter display is presented.

```
(date)                RANDOM COMMAND TEST 2
RCT2    SET PARAMS     PA = xxxxxxxx

ADDR  xxxxxxxx       PW0 = xxxaaaaa
ADDR  xxxxxxxx       PW1 = xxxxxxxx
ADDR  xxxxxxxx       PW2 = xxxxxxxb
ADDR  xxxxxxxx       PW3 = cccccccc
```

where:

(date) Last revision date of RCT2.

PA	RMA word address of control parameter word area.
PW0	Parameter word 0. aaaaa are test control bits. See parameter/control words.
PW2	Parameter word 2. b bits are selection select bits.
PW3	Parameter word 3. cccccccc bits are subsection select bits.

4.1.3 Error Message Display

When RCT2 detects an error, the following standard error message is displayed. Depending upon the type of error, an additional error message is displayed below the standard message. See Error Messages for definitions of these messages.

```
RCT2   SE   PCxxxxxxxx  Sxxxx  SBxxxx
EC1=0000  EC2=0000  TE=xxxx
```

where:

SE	Stopped on error
PCxxxxxxxx	Pass count
Sxxxx	Failing section number
SBxxxx	Failing subsection number
EC1	Not implemented
EC2	Not implemented
TE	Total error count

4.2 OPERATOR ENTRIES

4.2.1 Commands

The following RCT2 commands are provided:

Command	Function
P,x	Skip pass count. Allows test to skip forward a specified count prior to test execution. Parameter x equals the specified pass count.
R	Restart test with initial parameter settings.
S	Stop test execution.

(space)	Start/continue test execution.
LOOP	Set scope loop; causes test to continuously execute instructions/operands in subsection. Test does not communicate with operator. To regain control, clear CM word location $3A44_{16}$ with EC command, EC,3A44,0.
SST/CST	Set/clear stop at end of test parameter bit.
SRT/CRT	Set/clear repeat test parameter bit.
SSE/CSE	Set/clear stop on error parameter bit.
SRS/CRS	Set/clear repeat section parameter bit.
SRB/CRB	Set/clear repeat subsection parameter bit.
SSB/CSB	Set/clear stop at end of subsection parameter bit.

Each command is checked for proper format. The test will display the message INVALID ENTRY on line 10 of the display screen if an improper format is entered. The message is cleared by any subsequent valid command.

4.2.2 Running Procedures

Load and Run RCT2

Type the CMSE commands listed to load or make a CMSE command buffer of the commands and call it GO,xxxx; where xxxx is the name of the command buffer.

Change Test Parameters

Test parameters can be changed prior to starting RCT2 execution by using the CMSE EC and/or EB commands, or at any time RCT2 is running or has stopped using the RCT2 commands or the CMSE EC and/or EB commands.

When the stop-at-end-of-test-parameter bit is set, RCT2 stops before the first pass to allow parameter changes. Press space to resume RCT2. Type CST to clear the stop-at-end-of-test-parameter bit and allow nonstop execution.

Refer to Parameter Words for a list of RCT2 parameters. Refer to Operator Entries for RCT2 keyboard commands required to change the parameters. The parameter settings at the time RCT2 is initially started (DK,1A2 in command buffer) are saved and reinstated whenever RCT2 is restarted by typing an R command.

NOTE

The repeat test parameter bit is set by default. The test continuously generates random instructions/operands. If bit is cleared (type in CRT), it runs one complete pass; it can be dropped via a call to VLEX.

Stop on Error

Type in SSE to set the stop on error parameter bit. RCT will then stop whenever an error is detected and wait for an input.

Entry of a space resumes RCT2 execution until another error is detected.

Type in CSE to clear the stop on error parameter bit. RCT2 will not stop when errors are detected, but will increment the error count if errors are detected.

Loop on Error

After RCT2 has stopped on an error, to loop on a failure, type SRB and a space. The test will reexecute the failing set of instruction/operands. The running message will indicate a repeat subsection loop (RB).

Type in CRB to stop the loop and continue to the next selected subsection.

Scope Loop

After RCT2 has stopped on an error, to scope the failure, type in SRB and LOOP. This causes the test to reexecute the failing subsection and begin scope loop. During scope loop, the test does not refresh the display or monitor the keyboard.

To remove RCT2 from scope loop mode and reinstate communication with the operator, clear CM word location $3A44_{16}$ with the CMSE instruction EC,3A44,0.

Stop Test

The test can be stopped at any time during execution by typing in S. The test will stop and wait for input. Entering a space will continue the test.

4.3 NORMAL MESSAGES

None provided.

4.4 ERROR MESSAGES

When RCT2 receives an improperly formatted command or when it detects an error, it displays one of the following messages.

Message	Explanation
INVALID ENTRY	RCT2 does not recognize the command entered. Entry of a valid command clears the message.
SxSy ERROR (mmm)	RCT2 detected an error in the section designated by x and the subsection designated by y. Parameter mm identifies the instruction type. Enter a space to continue the test.

SECTION I-4

FIXED OPERAND COMMAND TESTS - FCT1-3,5

==========

1 INTRODUCTION

This part of the manual describes the Fixed Instruction Command Tests, FCT1,2,3,5. The tests detect failures within the processor at the instruction level.

FCT1 is a IOU based test which briefly tests every instruction to ensure that it will not hang up any of the following CPU based tests. Some testing of the usage set instructions (for FCT3) has been included in FCT1 rather than in FCT2.

FCT2 is a CPU-based test which uses the inverted pyramid form of testing to check the usage set for FCT3 (and for other instruction level tests such as the random command test RCT1). The list of instructions in the usage set exactly matches the list of instructions tested in FCT2. See FCT2 Section Descriptions, later in this document.

FCT3 is a CPU based, fixed operand test. Most of the testing that is done by the FCT, is done in FCT3. The implementation of this test is based on the concept of a usage set (a small group of machine instructions in which the bulk of the test's object code is implemented). These instructions test the entire instruction set.

FCT5 is an IOU based test for those instructions and portions of instructions which cannot be tested by a CPU-based test.

The fixed operands command tests consist of nearly 400 test sections, divided into ten tests (FCT1, FCT2, FCT5, and seven tests that comprise FCT3). FCT is divided this way to allow for flexibility in test sequencing, variations in testing technique, and differences in maintenance requirements.

Within each test there is one section for each instruction that is tested. Note that any one test does not necessarily test all instructions, and that any one test does not always contain a complete test for a given instruction. The testing of each instruction has been partitioned in a manner best suited to that individual instruction.

The test sections are broken down into subsections, each of which tests a given feature of that instruction.

Within the subsection each data pattern used (to test a feature), is assigned to a different condition. In FCT2 and FCT3, where the test uses the elements of a data pattern table as test operands, the index into the table is displayed as the loop index. When more than one data pattern table is used simultaneously by a given condition, all active loop indexes are displayed (one per table, up to three). Sometimes, when the required test data is a simple count, the loop index itself is used as the test data.

To further explain the relationship between a condition and the loop indexes, the following example is given. In FCT3 there is a section to test an ADD instruction. A subsection is used to check carry propagation. A condition is assigned to the case where Xk contains a propagating one's pattern and Xj contains a sliding one's pattern. Two loop indexes are displayed, one giving the index into each pattern table. To assist the test user, the index value generally indicates the bit number being tested. (In the ADD instruction example, if loop 1 equals 16, bits 0 through 16 of the propagating pattern are set. If loop 2 equals 56, bit 56 of the sliding 1 pattern is set.) The user can determine how the loop indexes are being used by either stepping through the test a condition at a time, or by referring to the listings. It must be noted that the user can always fully understand the error messages without understanding how the loops are being used.

2 REQUIREMENTS

This section contains the hardware and software requirements for execution and continuation.

2.1 HARDWARE

This test is intended for Models 810, 815, 825, 830, 835, 845, and 855 computer systems.

Hardware Required to Run Test

In addition to requirements to run CMSE, FCTs require the following:

1 CPU
1 megabyte of central memory
1 PP

2.2 SOFTWARE

This product executes under the control of, and using the facilities of, the Common Maintenance Software Executive (CMSE), for MSL15X. FCT3 also incorporates the virtual level executive (VLEX), and FCT1 and 5 require the instruction level test controller (ITC). Portions of the Diagnostic Executive (DEX) are compiled with, and are a part of the ITC object code.

2.3 ACCESSORIES

Microfiche of listings and test case descriptions.

2.4 CHARACTERISTICS

1.	Program name	FCT1-3, 5
2.	Size (source) (see table 4-1)	250,000
3.	Size (memory required) (see table 4-1)	1MB
4.	Code type	PP, CP code
5.	Run time (default) (see table 4-1)	
6.	Run time (quick look) (see table 4-1)	
7.	Run time (all sections) (see table 4-1)	
8.	Level of isolation	detect only
9.	Off-line test	yes
10.	Off-line system	CMSE (MSL15X)
11.	Resident during execution	PP, CP
12.	Assembly language	PP Compass, META
13.	Source code maintenance	ASCII-MODIFY, UPDATE
14.	Uses maintenance channel	yes

TABLE 4-1. FCT1-3,5 SIZES AND TIMES

TOTALS	Size (Source)	Size Memory Required	Approximate Run Time (in seconds) for Model 810/815/825/830	Approximate Run Time (in seconds) for Model 835/845/855
FCT1 - CP - IOU	42,000 5,000	1MB 4096	120	114
FCT2 - CP	17,000	1MB	23	20
FCT3 - CP	166,000	1MB	300 (quick look)	24 (quick look)
FCT5 - CP - IOU	13,000 5,000	1MB 4096	32	17

3 OPERATIONAL PROCEDURE

FCT tests may be run in one of the following sequences:

FCT1	FCT1	FCT1
FCT2	FCT5	FCT2
FCT3	FCT2	FCT5
FCT5	FCT3	FCT3

The order is specified to minimize the amount of hardware that is used before it is tested. Running the tests out of order may cause error messages which describe the problem incorrectly.

3.1 RESTRICTIONS AND USER CAUTIONS

PSM and PTL Masks

FCT3 is coded with the values for the page size mask (PSM) and for the page table length (PTL) specified at assembly time. These values are used throughout the test and cannot be altered during linking, loading or execution, thereby limiting testing of the TPAGE, LPAGE and PURGE instructions to cases using the given values for these masks.

Real/Virtual Memory Addressing Modes

FCT is a virtual level test. However, it is possible to run some sections in real memory addressing mode as well as in the normal virtual mode.

FCT2 and FCT3 can only run in virtual mode. FCT1 and FCT5 can run most sections in both modes. The sections of FCT which require virtual addressing mode are listed in the table below. All sections or subsections which are not listed can run both addressing modes.

Test	Section	Subsection	Opcode	Ref No.	Mnemonic
FCT1	48	7	94	037	BRXEQ
FCT1	58	all	2E	047	BRREL
FCT1	110	1,2,4	B0	116	CALLREL
FCT1	111	2,4	B5	115	CALLSEG
FCT1	112	all	04	117	RETURN
FCT1	113	19	0E	130	CPYSX
FCT1	114	all	16	126	TPAGE
FCT1	115	all	17	127	LPAGE
FCT2	all	—	—	---	
FCT3	all	—	—	---	
FCT5	1	all	2F	048	BRDIR
FCT5	2	all	05	138	PURGE (map)
FCT5	3	all	05	138	PURGE (cache)
FCT5	8	2,3,4	0E	130	CPYSX

3.2 LOADING PROCEDURES

The first instructions given in the sections below for the loading of the FCTs are general in nature and do not reflect the differences in loading that will be required for different models.

Command buffers exist on the MSL 15X tape to facilitate loading and execution of each test. (Refer to the table with each test for the names of command buffers provided.) Before using a command buffer you may have to modify it for your system. Display the command buffer using the CMSE command buffer display commands or print the contents of the command buffer using procedures provided in the Command Buffer Maintenance section of the MSL 15X Reference Manual. Then modify the command buffer as directed by comments embedded in the command buffer. When you are satisfied that the command buffer is set up properly, enter a GO,xxxx command (where xxxx is the command buffer name) to execute it.

Modified command buffers can be saved on a back-up tape for future use. Refer to the Utilities section of the MSL 15X Reference Manual for procedures.

See sections 3.2 and 3.3 for information on DEC settings.

In general, the command buffers attempt to run the tests when the state of the system is unknown (thus, a great deal of initialization), and assumes that no other tests are running (otherwise, the initialization might interfere with them).

3.2.1 Loading of FCT1

The version level is represented by a modification date for each section. This date is formed in ASCII characters after the subsection table which starts at 800_{16} word address. It is followed by the assembly date and the copyright message. The product resides as a series of overlays on the Maintenance Software Library (MSL).

Model Dependent Command Buffer Names

Test Name	810/815/ 825/830	835	845/ 855
FCT1	FCT11	FCT12	FCT13

Please refer to MSL15X for a description of how to set up and use command buffers. The version of ITC used by FCT1 is loaded from the file FCTITC.

3.2.2 Loading of FCT2

The version number for FCT2 may be found with the copyright message, after the parameter area. It is represented in ASCII characters.

The product resides as a single file on the Maintenance Software Library (MSL).

Model Dependent Command Buffer Names

Test Name	810/815/ 825/830	835	845/ 855
FCT2	FCT21	FCT22	FCT23

Command buffers provided on the MSL for FCT2 execute a DK command which causes test execution to commence without a parameter stop. If parameter changes are required, a modified command buffer without the DK command should be used. This will allow the parameter display to appear on the screen. Parameters may then be modified before the DK command is issued directly from the keyboard.

3.2.3 Loading Of FCT3

The product resides as a seven files on the Maintenance Software Library (MSL). These seven files are: FT3GI0B, FT3GI1B, FT3GI2B, FT3GI3B, FT3FPB, FT3BDPB, and FT3SYSB. The version number for FCT3 may be found with the copyright message, after the control/parameter area which is at 1400_{16} word address. It is represented with ASCII characters.

The sections in FCT3 are numbered in one sequence as if the test was one large test and not seven parts. The sections in each part are listed in the following table. The section enable bits must be set to enable only those sections which exist in the part of FCT3 which is being run.

The file name for each part is different, and is given in the table below.

There are other optional differences related to parameter settings. The test should be run with the processor in different states to ensure complete testing. For example, FCT3 should be run in both job and monitor modes, with and without cache, etc. Refer to MSL 15X Reference Manual for the modes of the processor; and to FCT3 Parameter Words for bits which must be set to correspond to the state of the machine. The command buffers suggested for FCT3 do only a part of this, as listed in the following table.

Since it is important to run the system instructions in both job and monitor modes, two command buffers are suggested to run FT3SYSB. The command sequence given in this document causes the test to be run in job mode, and testing of the external port is turned on. To modify the command sequence for monitor mode, add: MC,A80,A40,34 and set the monitor mode bit in parameter word 0.

The differences in the command buffers provided for running FCT3 are detailed in the following table. The tests have alpha-numeric names of six characters. The sixth is a model dependent number (shown by an asterisk below) with the following meaning: Model 810/815/825/830 = 1, Model 835 = 2, Model 845/855 = 3

Command Buffer	Binary File	Sections Enabled	Mode	Instructions Tested
FT3G0*	FT3GI0B	0- 16	Job	Part of General Instr.
FT3G1*	FT3GI1B	17- 30	Job	Part of General Instr.
FT3G2*	FT3GI2B	31- 55	Job	Part of General Instr.
FT3G3*	FT3GI3B	56- 75	Job	Part of General Instr.
FT3FP*	FT3FPB	128-143	Job	Floating Point Instr.
FT3BD*	FT3BDPB	160-177	Job	BDP Instructions
FT3S1*	FT3SYSB	192-207	Job	System Instructions
FT3S2*	FT3SYSB	192-207	Mon.	System Instructions

3.2.4 Loading Of FCT5

The version level is represented by a modification date for each section. This date is formed in ASCII characters after the subsection table which starts at 800_{16} word address. It is followed by the assembly date and the copyright message. The product resides as a series of overlays on the Maintenance Software Library (MSL).

Model Dependent Command Buffer Names

Test Name	810/815/ 825/830	835	845/ 855
FCT5	FCT51	FCT52	FCT53

Execution of these commands causes default parameters to be used by the Instruction Test Controller (ITC). The version of ITC used by FCT5 is loaded from the file FCTITC. The program is set for FCT5 by storing the ASCII representation of F5 into PP word 57_8.

3.3 PARAMETERS AND CONTROL WORDS

3.3.1 Parameters

FCT1 and FCT5 Parameter Words

Parameter words control the execution of the test. These words are located at PP locations 122_8 to 144_8 (directly following the control words).

Parameter words can be set or cleared manually through the CMSE commands such as SSC or CSC. Refer to the MSL 15X Reference Manual. Also see section 4.2.1 for other commands that affect the parameter settings.

Word Address In Octal	Tag	Bit	Position Octal/Hex	*	Meaning
122	PARAM0	63	0001/0001	*	Stop at end of test
		62	0002/0002		Stop at end of section
		61	0004/0004		Stop at end of subsection
		60	0010/0008		Stop at end of condition
		59	0020/0010	*	Stop on error
		58	0040/0020		Log errors in dayfile
		57	0100/0040	*	Repeat test
		56	0200/0080		Repeat section
		55	0400/0100		Repeat subsection
		54	1000/0200		Repeat condition
		53	2000/0400		Scope mode
		52	4000/0800		Quick look (not used)
		51	10000/1000		Bypass all messages
		50	20000/2000		Display only error messages
		49	40000/4000		Not used
		48	100000/8000	*	Accept CMSE parameter words(must be set)
123	PARAM1	63	0001/0001		Reserved
		62	0002/0002		Reserved
		61	0004/0004		Bypass parameter stop
		60	0020/0010		Reserved
		59	0040/0020		Reserved
		58	0100/0040		Reserved
124	PARAM2	48-63			Repeat test count
125	PARAM3	63	0001/0001	*	Test CPU 0
		62	0002/0002	*	Test CPU 1
		48-61			Reserved
126	PARAM4	48-63			Reserved

* An asterisk in this column indicates that default is nonzero.

Page I-4-10 is blank to compensate for deleted material.

Word Address In Octal	Tag	Bit	Position Octal/Hex	*	Meaning
127	PARAM5	48-63		*	Sections 15-00 select **
130	PARAM6	48-63		*	Sections 31-16 select **
131	PARAM7	48-63		*	Sections 47-32 select **
132	PARAM8	48-63		*	Sections 63-48 select **
133	PARAM9	48-63		*	Sections 79-64 select **
134	PARAM10	48-63		*	Sections 95-80 select **
135	PARAM11	48-63		*	Sections 111-96 select **
136	PARAM12	48-63		*	Sections 127-112 select **
137	PARAM13	48-63		*	Sections 143-128 select **
140	PARAM14	48-63			Reserved
141	PARAM15	48-63			Reserved
142	PARAM16	48-63		*	Delay count for processor hung (default value is 20_8)
143	PARAM17	62	0002/0002	*	Enable PFS register operation
144	PARAM18				Reserved

FCT2 Parameter Words

Parameter words for FCT2 are located in central memory and can be set or cleared manually through the CMSE commands, such as EB or EC. See paragraph 3.2.2 for additional information concerning parameter changes. The default value of all parameter words is zero unless otherwise specified. Section 00 cannot be repeated after advancing beyond that section. This is because it is checking the initial load of the exchange package. Section 00 is omitted when repeat test is selected.

Word Address In Hex	Tag	Bit	Position In Hex	*	Meaning
608	PARAM 0	00-53			Unused, reserved, or unimplemented
		54	200		Repeat condition
		55	100		Repeat subsection
		56	80		Repeat section
		57	40	*	Repeat test
		58	20		Log errors
		59	10	*	Stop on error
		60	8		Stop at end of condition
		61	4		Stop at end of subsection
		62	2		Stop at end of section
		63	1	*	Stop at end of test
609	PARAM 1	00-63			Repeat count (test will repeat if pass count is less than repeat count)
60A	PARAM 2	00-63		*	Section select for sections 63-00. All sections set by default

* An asterisk in this column indicates that default is nonzero.
** All sections set by default.

FCT3 Parameter Words

Parameter words control the execution of the test.

Parameter words can be set/cleared manually with the CMSE commands EB or EC.

NOTE

Unless otherwise specified below, all parameter values are defaulted to 0.

Some parameter bits must be set to correspond to the state that the machine is running in, or else false error reports will be produced. These are external port connected, cache buffer enabled, map segment file enabled, and map page file enabled. One parameter bit (test started in monitor mode) must correspond to the mode that the test was started in, or else false errors may result.

All parameters except quick look omit number, repeat count, stop at end of loop level, and repeat loop level are controlled by the setting/clearing of one bit, . The latter two require that the loop level desired (1, 2, or 3) be entered numerically into the reserved space. For example, entering 0003020000000000_{16} into PARAM0 would specify a stop at the completion of loop level 2, and to repeat loop level 3. The omit number may be in the range $00\text{-}FF_{16}$, with progressively more test cases skipped as the number increases. The repeat count is used as a 64-bit signed integer.

Word Address In Hex	Tag	Bit	Position In Hex	*	Meaning
1409	PARAM 0	00-07	FF00000000000000	*	Quick look omit number (default 20_{16})
		08-15	FF000000000000		Repeat loop level
		16-23	FF0000000000		Stop at end of loop level
		24	8000000000		Omit all result checking,
		25	4000000000	*	Stop for operator request
		26	2000000000		Exchange instruction use allowed
		27	1000000000		External port connected
		28-31			Unused
		32-39			Reserved
		40	800000		Map page buffer enabled
		41	400000		Map segment buffer enabled
		42	200000		Cache buffer enabled
		43	100000		Test started in monitor mode
		44-47			Unimplemented
		48-49			Reserved

* An asterisk in this column indicates that default is nonzero.

		50	2000		Display only error messages
		51			Unimplemented
		52	800		Quick look mode
		53	400		Scope mode
		54	200		Repeat condition
		55	100		Repeat subsection
		56	80		Repeat section
		57	40	*	Repeat test
		58	20		Log errors
		59	10	*	Stop on error
		60	8		Stop at end of condition
		61	4		Stop at end of subsection
		62	2		Stop at end of section
		63	1	*	Stop at end of test
140A	PARAM 1	00-63			Repeat test count (test will repeat if pass count is less than repeat test count)
140B	PARAM 2	00-63		*	Section select 063-000 (default $FFFFFFFFFFFFFFFF_{16}$)
140C	PARAM 3	00-63		*	Section select 127-064 (default $0000000000000FFF_{16}$)
140D	PARAM 4	00-63		*	Section select 191-128 (default $03FFFF000000FFFF_{16}$)
140E	PARAM 5	00-63		*	Section select 255-192 (default $000000000000FFFF_{16}$)
140F	PARAM 6	00-58			Unused
		59	10		Reserved, must be set to zero
		60	8		Executing on Model 845/855, must be set
		61	4		Executing on Model 835, must be set
		62	2		Executing on Model 810/815/825/830, must be set
		63	1		Reserved, must be set to zero

The values for the parameter words 1 though 4 must be altered to account for the separate binaries used to implement FCT3.

The table below gives the values for these parameter words that may be set in the command buffers used to load and execute the different binaries. The significance of the command buffer names used below is explained in section

* An asterisk in this column indicates that default is nonzero.

RECOMMENDED COMMAND BUFFER NAME	PARAM2	PARAM3	PARAM4	PARAM5
FT3G0*	1FFFF	0	0	0
FT3G1*	7FFE0000	0	0	0
FT3G2*	0FFFFFF80000000	0	0	0
FT3G3*	FF00000000000000	0FFF	0	0
FT3FP*	0	0	0FFFF	0
FT3BD*	0	0	3BFFF00000000	0
FT3S1*	0	0	0	0FFFF
FT3S2*	0	0	0	0FFFF

3.3.2 Control Words

FCT1 and FCT5 Control Words

Control words are intended to identify a program and supply information to a higher system or operator. They do not normally affect test execution. The control words for FCT1 and FCT5 are located at PP locations 102_8 to 121_8 and are as follows:

Tag	Meaning
CW0	Program name (first 2 characters)
CW1	Program name (last 2 characters)
CW2	Program type
CW3	Monitor ID word
CW4	Error code number 1
CW5	Error code number 2 (not implemented)
CW6	Pass counter
CW7	Current section counter
CW8	Current subsection counter
CW9	Current condition counter
CW10	Current error counter
CW11	Current CPU

FCT2 Control Words

Control words are intended to identify a program and supply information to a higher system or operator. They do not normally affect test execution. Control words are located beginning at central memory 600_{16} word address.

* An asterisk in this column indicates that default is nonzero.

They are:

Word Address In Hex	Tag	Bit	Position In Hex	*	Meaning
600	CW 0	00-63		*	Test name FCT2 (in ASCII code)
601	CW 1	00-15	FFFF000000000000	*	$2A00_{16}$, models 810, 815, 825, 830, 835, 845, and 855 are supported.
		16-31	FFFF00000000	*	$CCF8_{16}$, indicates configurations and options that the test may execute with.
		32-47	FFFF0000	*	$FFFF_{16}$, indicates that FCT2 may execute with all memory size increments.
		48-63			0000_{16}, reserved bits.
602	CW 2	00-31			Error code 1, which is not implemented.
		32-63			Error code 2, which is not implemented.
603	CW 3	00-63			Error count.
604	CW 4	00-63			Pass count.
605	CW 5	00-63			Section number.
606	CW 6	00-63			Subsection number.
607	CW 7	00-63			Condition number.

FCT3 Control Words

Control words are intended to identify a program and supply information to a higher system or operator. They do not normally affect test execution. Control words are located beginning at 1400_{16} word address. They are:

Word Address In Hex	Tag	Bit	Position In Hex	*	Meaning
1400	CW 0	00-63		*	Test name FCT3 (in ASCII code)
1401	CW 1	00-15	FFFF000000000000	*	$2A00_{16}$, models 810, 815, 825, 830, 835, 845, and 855 are supported.
		16-31	FFFF00000000	*	$CCF8_{16}$, indicates configurations and options that the test may execute with.
		32-47	FFFF0000	*	$FFFF_{16}$, indicates that FCT3 may execute with all memory size increments.
		48-63			0000_{16}, reserved bits.

* An asterisk in this column indicates that default is nonzero.

1402	CW 2	00-31		Error code 1, which is not implemented.
		32-63		Error code 2, which is not implemented.
1403	CW 3	00-63		Error count.
1404	CW 4	00-63		Pass count.
1405	CW 5	00-63		Section number.
1406	CW 6	00-63		Subsection number.
1407	CW 7	00-63		Condition number.
1408	CW 8	00-15	FFFF000000000000	Loop 1 count.
		16-31	FFFF00000000	Loop 2 count.
		32-47	FFFF0000	Loop 3 count.
		48-55		Unused.
		56-63	FF	Currently active loop level.

3.3.3 Data Patterns

In the descriptions of the testing that is done by each test section (see Test Sections) there are references to the data patterns which are being used. These patterns are too large to be included in this document but are listed below for reference. In some cases the name of the table will fully define the contents of the table. The data patterns can be seen in the listings for each test. Each section can contain its own specialized data tables, but in FCT2 and FCT3, commonly used data has been grouped together. For FCT2 see the listing for section zero, S00. For FCT3 see the listing titled FCT3 PAT, Defined Data Patterns.

The intent behind the other names and terms found in the section descriptions, whose meanings are not self evident, can be obtained from the listing for the given section.

Test	Pattern	Contents
FCT2	PROP0	Propagating zeros, right to left
	ZEROS	Word of zeros
	PROP1	Propagating ones, right to left
	ONES	Word with all bits set
	PRMPAT	Primary pattern. See listing
	SLIDE0	Sliding zero, right to left
	SLIDE1	Sliding one, right to left
	STDNOT	Complement of STDPAT
	STDPAT	Standard pattern. See listing
	COUNT	Count from 0 to 40
	PAT_24	Patterns for section 24
	V36Z	Pattern for section 36
	V36F	Pattern for section 36
	V36A	Patterns for sections 36 and 37
	V36B	Patterns for sections 36 and 37
	V37X	Patterns for section 37
	ISOMP1	Patterns for sections 46 and 47
	ISOMP2	Patterns for sections 46 and 47
	DATA3C	Patterns for section 47

FCT3	STDPAT	Standard pattern. See listing
	STDPT_7F	'7F7F...7F' pattern in STDPAT
	COMSTD	Complement of STDPAT
	BRDATA	Data patterns for branch testing
	BITMASK	Masking patterns
	PRMPAT	Primary pattern. See listing
	COUNT	Count pattern from 0 to 40
	PROGAT1A	Propagating ones, left to right
	PROGAT1	Propagating ones, left to right
	LBIT_SET	Word with bits 0 to 31 set
	PROGAT0	Propagating zeros, left to right
	RBIT_SET	Word with bits 32 to 63 set
	ASTDPAT	STDPAT modified for A register use
	SLIDE0	Sliding zero, left to right
	SLIDE1	Sliding one, left to right
	PACKDECI	Packed decimal count, 00 to 99 then 00 to 09
	UNPKDECI	Unpacked decimal count, 00 to 99 then 00 to 08

3.4 TEST AND SECTION INDEX

3.4.1 FCT1 Test Sections

Sect	Tag	Instruction Mnemonic	Op Code	Ref	Remarks
0	F100	HALT	00	121	
1	F13D	ENTP	3D	057	
2	F13E	ENTN	3E	058	
3	F11F	ENTO,ENTZ,ENTS	1F	061	
4	F139	ENTX	39	164	
5	F13F	ENTL	3F	060	
6	F187	ENTC	87	165	
7	F18D	ENTE	8D	059	
8	F1B3	ENTA	B3	168	
9	F11E	MARK	1E	145	
10	F126	MULX	26	024	
11	F1B2	MULXQ	B2	168	
12	F122	MULR	22	032	
13	F18C	MULRQ	8C	033	
14	F127	DIVX	27	025	
15	F123	DIVR	23	034	
16	F109	CPYAA	09	051	
17	F10D	CPYXX	0D	049	
18	F10B	CPYAX	0B	050	
19	F10A	CPYXA	0A	052	
20	F10C	CPYRR	0C	053	
21	F118	IORX	18	065	
22	F119	XORX	19	066	
23	F11A	ANDX	1A	067	
24	F11B	NOTX	1B	068	
25	F11C	INHX	1C	069	
26	F124	ADDX	24	022	
27	F120	ADDR	20	027	

Sect	Tag	Instruction Mnemonic	Op Code	Ref	Remarks
28	F110	INCX	10	166	
29	F18B	ADDXQ	8B	143	
30	F128	INCR	28	029	
31	F125	SUBX	25	023	
32	F121	SUBR	21	030	
33	F111	DECX	11	167	
34	F129	DECR	29	031	
35	F12D	CMPX	2D	035	
36	F12C	CMPR	2C	036	
37	F18A	ADDRQ	8A	028	
38	F1A8	SHFC	A8	062	
39	F18E	ADDAQ	8E	054	
40	F18F	ADDPXQ	8F	055	
41	F12A	ADDAX	2A	056	
42	F1A7	ADDAD	A7	161	
43	F1A9	SHFX	A9	063	
44	F1AA	SHFR	AA	064	
45	F1AC	ISOM	AC	070	
46	F1AD	ISOB	AD	071	
47	F1AE	INSB	AE	072	
48	F194	BRXEQ	94	037	
49	F195	BRXNE	95	038	
50	F196	BRXGT	96	039	
51	F197	BRXGE	97	040	
52	F190	BRREQ	90	041	
53	F191	BRRNE	91	042	
54	F192	BRRGT	92	043	
55	F193	BRRGE	93	044	
56	F19C	BRINC	9C	045	
57	F19D	BRSEG	9D	046	
58	F12E	BRREL	2E	047	
59	F188	LBIT	88	014	
60	F1A2	LXI	A2	005	
61	F182	LX	82	006	
62	F1D0	LBYTS,S	D0-7	001	
63	F1A0	LAI	A0	016	
64	F184	LA	84	017	
65	F180	LMULT	80	020	
66	F186	LBYTP,J	86	013	
67	F1A4	LBYT,X0	A4	009	
68	F183	SX	83	008	
69	F1A1	SAI	A1	018	
70	F1A3	SXI	A3	007	
71	F181	SMULT	81	021	
72	F189	SBIT	89	015	
73	F185	SA	85	019	
74	F1D8	SBYTS,S	D8-F	003	
75	F1A5	SBYT,X0	A5	011	
76	F13A	CNIF	3A	097	
77	F13B	CNFI	3B	098	
78	F130	ADDF	30	099	
79	F131	SUBF	31	100	

Sect	Tag	Instruction Mnemonic	Op Code	Ref	Remarks
80	F134	ADDD	34	105	
81	F135	SUBD	35	106	
82	F132	MULF	32	103	
83	F136	MULD	36	107	
84	F133	DIVF	33	104	
85	F137	DIVD	37	108	
86	F13C	CMPF	3C	114	
87	F198	BRFEQ	98	109	
88	F199	BRFNE	99	110	
89	F19A	BRFGT	9A	111	
90	F19B	BRFGE	9B	112	
91	F19E	BROVR,UND,INF	9E	113	
92	F1F4	CALDF	F4	096	
93	F176	MOVB	76	089	
94	F1EB	TRANS	EB	088	
95	F177	CMPB	77	084	
96	F1E9	CMPC	E9	085	
97	F1ED	EDIT	ED	091	
98	F1F3	SCNB	F3	086	
99	F1F9	MOVI	F9	154	
100	F1FA	CMPI	FA	155	
101	F1FB	ADDI	FB	156	
102	F170	ADDN	70	074	
103	F171	SUBN	71	075	
104	F172	MULN	72	076	
105	F173	DIVN	73	077	
106	F1E4	SCLN	E4	078	
107	F1E5	SCLR	E5	079	
108	F175	MOVN	75	092	
109	F174	CMPN	74	083	
110	F1B0	CALLREL	B0	116	
111	F1B5	CALLSEG	B5	115	
112	F104	RETURN	04	117	
113	F10E	CPYSX	0E	130	
114	F116	TPAGE	16	126	
115	F117	LPAGE	17	127	
116	F10F	CPYXS	0F	131	
117	F12F	BRDIR	2F	048	
118	F102	EXCHANGE	02	120	
119	F106	POP	06	118	
120	F108	CPYTX	08	132	
121	F114	LBSET	14	124	
122	F1B1	KEYPOINT	B1	136	
123	F19F	BRCR	9F	134	
124	F1B4	CMPXA	B4	125	
125	F103	INTRUPT	03	122	
126	F105	PURGE	05	138	
127	F1C0	EXECUTE	C0-7	139	
128	F1BE	RESERVED	BE	170	
129	F1BF	RESERVED	BF	171	
130	F1UI	UNIMPLEM.	**	---	
131	F101	SYNC	01	194	

3.4.2 FCT2 Test Sections

Sect	Tag	Instruction Mnemonic	Op Code	Ref	Remarks
0	S00	BRXEQ	94	037	Partially tests BRXEQ instruction; tests that X registers contain all zeros. Cannot be rerun once X registers equal zero; tested further in section 11.
1	S01	ENTE	8D	059	Tests ENTE instruction and partially tests ability of X registers to hold patterns.
2	S02	ENTS	1F	061	Tests ENTS instruction and partially tests ability of X registers to hold patterns.
3	S03	CPYRR	0C	053	
4	S04	CPYXX	0D	049	
5	S05	ENTL	3F	060	
6	S06	ENTP	3D	057	
7	S07	ENTN	3E	058	
8	S08	SHFC	A8	062	Tests SHFC instruction and partially tests ability of X registers to hold patterns.
9	S09	SHFX	A9	063	Tests SHFX instruction and partially tests ability of X registers to hold patterns.
10	S10	SHFR	AA	064	
11	S11	BRXEQ	94	037	Further tests the BRXEQ instruction begun in section 0.
12	S12	BRXNE	95	038	
13	S13	BRXGT	96	039	
14	S14	BRXGE	97	040	
15	S15	BRREQ	90	041	
16	S16	BRRNE	91	042	
17	S17	BRRGT	92	043	
18	S18	BRRGE	93	044	
19	S19	BRREL	2E	047	
20	S20	BRINC	9C	045	
21	S21	CPYAX, XA	0B, 0A	050, 052	Tests CPYAX and CPYXA instructions. Partially tests ability of A registers to hold patterns.
22	S22	CPYAA	09	051	Tests CPYAA instruction and partially tests ability of A registers to hold patterns.
23	S23	ADDXQ	8B	143	
24	S24	LXI	A2	005	
25	S25	LX	82	006	
26	S26	SXI	A3	007	
27	S27	SX	83	008	
28	S28	IORX	18	065	
29	S29	ANDX	1A	067	
30	S30	NOTX	1B	068	
31	S31	XORX	19	066	

Sect	Tag	Instruction Mnemonic	Op Code	Ref	Remarks
32	S32	INHX	1C	069	
33	S33	ADDX	24	022	
34	S34	SUBX	25	023	
35	S35	ADDAQ	8E	054	
36	S36	LAI	A0	016	
37	S37	SAI	A1	018	
38	S38	ADDPXQ	8F	055	
39	S39	BRDIR	2F	048	
40	S40	LBYTS,S	D0-D7	001	
41	S41	SBYTS,S	D8-DF	003	
42	S42	ADDR	20	027	
43	S43	INCR	28	029	
44	S44	SUBR	21	030	
45	S45	DECR	29	031	
46	S46	ISOM	AC	070	
47	S47	ISOB	AD	071	
48	S48	ENTX	39	164	
49	S49	ENTC	87	165	
50	S50	INCX	10	166	
51	S51	DECX	11	167	
52	S52	ENTA	B3	169	
53	S53	CALLSEG	B5	115	
54	S54	CALLREL	B0	116	
55	S55	RETURN	04	117	
56	S56	CPYSX	0E	130	
57	S57	CPYXS	0F	131	
58	S58	BRCR	9F	134	Tests BRCR instruction. Only alteration of UCR bits and testing of status of UCR bits is checked. Use of this instruction in FCT usage set requires following values of k(4 LE K LE 7) or (C LE k LE F).

3.4.3 FCT3 Test Sections

Sect	Tag	Instruction Mnemonic	Op Code	Ref	Remarks
0	REF001	LBYTS	D0-D7	001	Load bytes to Xk from (Aj) displaced by D and indexed by (Xi) right, length per S.
1	REF003	SBYTS	D8-DF	003	Store bytes from Xk at (Aj) displaced by D and indexed by (Xi) right, length per S.
2	REF005	LXI	A2	005	Load Xk from (Aj) displaced by 8*D and indexed by 8*(Xi) right.
3	REF006	LX	82	006	Load Xk from (Aj) displaced by 8*Q.
4	REF007	SXI	A3	007	Store Xk at (Aj) displaced by 8*D and indexed by 8*(Xi) right.
5	REF008	SX	83	008	Store Xk at (Aj) displaced by 8*Q.

Sect	Tag	Instruction Mnemonic	Op Code	Ref	Remarks
6	REF009	LBYT	A4	009	Load bytes to Xk from (Aj) displaced by D and indexed by (Xi) right, length per X0.
7	REF011	SBYT	A5	011	Store bytes from Xk at (Aj) displaced by D and indexed by (Xi) right, length per X0.
8	REF013	LBYTP	86	013	Load bytes to Xk from (P) displaced by Q, length per j.
9	REF014	LBIT	88	014	Load bit to Xk from (Aj) displaced by Q and bit-indexed by (X0) right.
10	REF015	SBIT	89	015	Store bit from Xk from (Aj) displaced by Q and bit-indexed by (X0) right.
11	REF016	LAI	A0	016	Load Ak from (Aj) displaced by D and indexed by (Xi) right.
12	REF017	LA	84	017	Load Ak from (Aj) displaced by Q.
13	REF018	SAI	A1	018	Store Ak at (Aj) displaced by D and indexed by (Xi) right.
14	REF019	SA	85	019	Store Ak at (Aj) displaced by Q.
15	REF020	LMULT	80	020	Load multiple registers from (Aj) displaced by 8*Q, selectively per (Xk) right.
16	REF021	SMULT	81	021	Store multiple registers to (Aj) displaced by 8*Q, selectively per (Xk) right.
17	REF022	ADDX	24	022	Integer sum, (Xk) replaced by (Xk) plus (Xj).
18	REF023	SUBX	25	023	Integer difference, (Xk) replaced by (Xk) minus (Xj).
19	REF024	MULX	26	024	Integer product, (Xk) replaced by (Xk) times (Xj).
20	REF025	DIVX	27	025	Integer quotient, (Xk) replaced by (Xk) divided by (Xj).
21	REF027	ADDR	20	027	Integer sum, (Xk) right replaced by (Xk) right plus (Xj) right.
22	REF028	ADDRQ	8A	028	Integer sum, (Xk) right replaced by (Xj) right plus Q.
23	REF029	INCR	28	029	Integer sum, (Xk) right replaced by (Xk) right plus j.
24	REF030	SUBR	21	030	Integer difference, (Xk) right replaced by (Xk) right minus (Xj) right.
25	REF031	DECR	29	031	Integer difference, (Xk) right replaced by (Xk) right minus j.
26	REF032	MULR	22	032	Integer product, (Xk) right replaced by (Xk) right times (Xj) right.
27	REF033	MULRQ	8C	033	Integer product, (Xk) right replaced by (Xj) right times Q.
28	REF034	DIVR	23	034	Integer quotient, (Xk) right replaced by (Xk) right divided by (Xj) right.

Sect	Tag	Instruction Mnemonic	Op Code	Ref	Remarks
29	REF035	CMPX	2D	035	Integer compare, (Xj) to (Xk), result to X1 right.
30	REF036	CMPR	2C	036	Integer compare, (Xj) right to (Xk) right, result to X1 right.
31	REF037	BRXEQ	94	037	Branch to (P) displaced by 2*Q if (Xj) equal to (Xk).
32	REF038	BRXNE	95	038	Branch to (P) displaced by 2*Q if (Xj) not equal to (Xk).
33	REF039	BRXGT	96	039	Branch to (P) displaced by 2*Q if (Xj) greater than (Xk).
34	REF040	BRXGE	97	040	Branch to (P) displaced by 2*Q if (Xj) greater than or equal to (Xk).
35	REF041	BRREQ	90	041	Branch to (P) displaced by 2*Q if (Xj) right equal to (Xk) right.
36	REF042	BRRNE	91	042	Branch to (P) displaced by 2*Q if (Xj) right not equal to (Xk) right.
37	REF043	BRRGT	92	043	Branch to (P) displaced by 2*Q if (Xj) right greater than (Xk) right.
38	REF044	BRRGE	93	044	Branch to (P) displaced by 2*Q if (Xj) right greater than or equal to (Xk) right.
39	REF045	BRINC	9C	045	Branch to (P) displaced by 2*Q and increment (Xk) if (Xj) greater than (Xk).
40	REF046	BRSEG	9D	046	Branch to (P) displaced by 2*Q if SEG(Aj) not equal to SEG(Ak); else compare BN(Aj) to BN(Ak), result to X1 right.
41	REF047	BRREL	2E	047	Branch to (P) indexed by 2*(Xk) right.
42	REF048	BRDIR	2F	048	Branch to (Aj) indexed by 2*(Xk) right.
43	REF049	CPYXX	0D	049	Copy to Xk from Xj.
44	REF050	CPYAX	0B	050	Copy to Xk from Aj.
45	REF051	CPYAA	09	051	Copy to Ak from Aj.
46	REF052	CPYXA	0A	052	Copy to Ak from Xj.
47	REF053	CPYRR	0C	053	Copy Xk right from Xj right.
48	REF054	ADDAQ	8E	054	Address (Ak) replaced by (Aj) plus Q.
49	REF055	ADDPXQ	8F	055	Address (Ak) replaced by (P) plus 2*(Xj) right plus 2*Q.
50	REF056	ADDAX	2A	056	Address (Ak) replaced by (Ak) plus (Xj) right.
51	REF057	ENTP	3D	057	Enter Xk with plus j.
52	REF058	ENTN	3E	058	Enter Xk with minus j.
53	REF059	ENTE	8D	059	Enter Xk with sign extended Q.
54	REF060	ENTL	3F	060	Enter X0 with logical jk.
55	REF061	ENTO,ENTZ,ENTS	1F	061	
56	REF062	SHFC	A8	062	Shift (Xj) to Xk circular, direction and count per (Xi) right plus D.

Sect	Tag	Instruction Mnemonic	Op Code	Ref	Remarks
57	REF063	SHFX	A9	063	Shift (Xj) to Xk, direction and count per (Xi) right plus D.
58	REF064	SHFR	AA	064	Shift (Xj) right to Xk right, direction and count per (Xi) right plus D.
59	REF065	IORX	18	065	Logical sum, (Xk) replaced by (Xk) OR (Xj).
60	REF066	XORX	19	066	Logical difference, (Xk) replaced by (Xk) XOR (Xj).
61	REF067	ANDX	1A	067	Logical product, (Xk) replaced by (Xk) AND (Xj).
62	REF068	NOTX	1B	068	Logical complement, (Xk) replaced by NOT (Xj).
63	REF069	INHX	1C	069	Logical inhibit, (Xk) replaced by (Xk) AND NOT (Xj).
64	REF070	ISOM	AC	070	Isolate bit mask into Xk per (Xi) right plus D.
65	REF071	ISOB	AD	071	Isolate into Xk from (Xi) right plus D.
66	REF072	INSB	AE	072	Insert into Xk from Xj per (Xi) right plus D.
67	REF143	ADDXQ	8B	143	Integer sum, (Xk) replaced by (Xj) plus Q.
68	REF145	MARK	1E	145	Set Xk per j and (X1) right.
69	REF161	ADDAD	A7	161	Address (Ak) replaced by (Ai) plus D per j.
70	REF164	ENTX	39	164	Enter X1 with logical jk.
71	REF165	ENTC	87	165	Enter X1 with sign extended jkq.
72	REF166	INCX	10	166	Integer sum, (Xk) replaced by (Xk) plus j.
73	REF167	DECX	11	167	Integer difference, (Xk) replaced by (Xk) minus j.
74	REF168	MULXQ	B2	168	Integer product, (Xk) replaced by (Xj) times Q.
75	REF169	ENTA	B3	169	Enter X0 with sign extended jkq.
76-127		Reserved Sections			
128	REF097	CNIF	3A	097	Floating point convert from integer, floating point (Xk) formed from integer (Xj).
129	REF098	CNFI	3B	098	Floating point convert to integer, integer (Xk) formed from floating point (Xj).
130	REF099	ADDF	30	099	Floating point sum, (Xk) replaced by (Xk) plus (Xj).
131	REF100	SUBF	31	100	Floating point difference, (Xk) replaced by (Xk) minus (Xj).
132	REF103	MULF	32	103	Floating point product, (Xk) replaced by (Xk) times (Xj).
133	REF104	DIVF	33	104	Floating point quotient, (Xk) replaced by (Xk) divided by (Xj).

Sect	Tag	Instruction Mnemonic	Op Code	Ref	Remarks
134	REF105	ADDD	34	105	Floating point DP sum, (Xk, Xk+1) replaced by (Xk, Xk+1) plus (Xj, Xj+1).
135	REF106	SUBD	35	106	Floating point DP difference (Xk, Xk+1) replaced by (Xk, Xk+1) minus (Xj, Xj+1).
136	REF107	MULD	36	107	Floating point DP product, (Xk, Xk+1) replaced by (Xk, Xk+1) times (Xj, Xj+1).
137	REF108	DIVD	37	108	Floating point DP quotient, (Xk, Xk+1) replaced by (Xk, Xk+1) divided by (Xj, Xj+1).
138	REF109	BRFEQ	98	109	Branch to (P) displaced by 2*Q if floating point (Xj) equal to (Xk).
139	REF110	BRFNE	99	110	Branch to (P) displaced by 2*Q if floating point (Xj) not equal to (Xk).
140	REF111	BRFGT	9A	111	Branch to (P) replaced by 2*Q if floating point (Xj) greater than (Xk).
141	REF112	BRFGE	9B	112	Branch to (P) displaced by 2*Q If floating point (Xj) greater than or equal to (Xk).
142	REF113	BROVR,UND,INF	9E	113	Branch to (P) displaced by 2*Q if floating point exception per j contained in Xk.
143	REF114	CMPF	3C	114	Compare floating point (Xj) to (Xk), result to X1 right.
144-159		Reserved Sections			
160	REF074	ADDN	70	074	Decimal sum, D(Ak) replaced by D(Ak) plus D(Aj).
161	REF075	SUBN	71	075	Decimal difference, D(Ak) replaced by D(Ak) minus D(Aj).
162	REF076	MULN	72	076	Decimal product, D(Ak) replaced by D(Ak) times D(Aj).
163	REF077	DIVN	73	077	Decimal quoitent, D(Ak) replaced by D(Ak) divided by D(Aj).
164	REF078	SCLN	E4	078	Decimal scale, D(Ak) replaced by D(Aj) scaled per (Xi) right plus D.
165	REF079	SCLR	E5	079	Decimal scale rounded, D(Ak) replaced by rounded D(Aj) scaled per (Xi) right plus D.
166	REF083	CMPN	74	083	Decimal compare, D(Aj) to D(Ak), result to X1 right.
167	REF084	CMPB	77	084	Byte compare, D(Aj) to D(Ak), result to X1 right, index to X0 right.
168	REF085	CMPC	E9	085	Byte compare collated, D(Aj) to D(Ak), both translated per (Ai) plus D, result to X1 right, index to X0 right.

Sect	Tag	Instruction Mnemonic	Op Code	Ref	Remarks
169	REF086	SCNB	F3	086	Byte scan while nonmember, D(Ak) for presence bit in (Ai)+D, index to X0 right, character to X1 right.
170	REF088	TRANB	EB	088	Byte translate, D(Ak) replaced by D(Aj), translated per (Ai) plus D.
171	REF089	MOVB	76	089	Move bytes, D(AK) replaced by D(Aj).
172	REF091	EDIT	ED	091	Edit, D(Ak) replaced by D(Aj) edited per M((Ai)+D).
173	REF092	MOVN	75	092	Numeric move, D(Ak) replaced by D(Aj) after formatting.
174	REF096	CALDF	F4	096	Calculate subscript and add D(Aj), checked and modified per (Ai) plus D, result added to Xk right.
175	REF154	MOVI	F9	154	Move immediate data, (Xi) right plus D to D(Ak).
176	REF155	CMPI	FA	155	Compare immediate data, (Xi) right plus D compared to D(Ak), result to (X1) right.
177	REF156	ADDI	FB	156	Add immediate data, (Xi) right plus D to D(Ak).
178-191			Reserved Sections		
192	REF115	CALLSEG	B5	115	Call per (Aj) displaced by 8*Q, arguments per (Ak).
193	REF116	CALLREL	B0	116	Call to (P) displaced by 8*Q, binding section pointer per (Aj), arguments per (Ak).
194	REF117	RETURN	04	117	Return.
195	REF118	POP	06	118	Pop.
196	REF120	EXCHANGE	02	120	Exchange.
197	REF122	INTRUPT	03	122	Interrupt processor per (Xk).
198	REF124	LBSET	14	124	Load bit to Xk right from (Aj) bit indexed by (X0) right and set bit in central memory.
199	REF125	CMPXA	B4	125	Compare (Xk) at (Xj); if not equal, load Xk from (Aj); if equal store (X0) at (Aj); however, if (Aj) locked, branch to P plus 2*Q.
200	REF126	TPAGE	16	126	Test page (Aj) and set Xk right. Requires that, for segment containing page table, contiguous PVAs be contiguous (RMAs) also.
201	REF127	LPAGE	17	127	Load page table index per (Xj) to Xk right and set X1 right. Requires that, for segment containing page table, contiguous PVAs be contiguous RMAs also.
202	REF130	CPYSX	0E	130	Copy to Xk per (Xj).
203	REF131	CPYXS	0F	131	Copy from Xk per (Xj).
204	REF132	CPYTX	08	132	Copy free running counter to Xk at (Xj) right.
205	REF134	BRCR	9F	134	Branch to (P) displaced by 2*Q and alter condition register per jk.

Sect	Tag	Instruction Mnemonic	Op Code	Ref	Remarks
206	REF136	KEYPOINT	B1	136	Keypoint, class j, code equal to (Xk) right plus Q.
207	REF138	PURGE	05	138	Purge buffer k of entry per (Xj) (model dependent).

3.4.4 FCT5 Test Sections

Sect	Tag	Instruction Mnemonic	Op Code	Ref	Remarks
208-255		Reserved Sections			
0	F500	EXCHANGE	02	120	
1	F501	BRDIR	2F	048	
2	F502	PURGE (MAP)	05	138	
3	F503	PURGE (CACHE)	05	138	
4	F504	CALLREL	B0	116	
5	F505	CALLSEG	B5	115	
6	F506	RETURN	04	117	
7	F507	POP	06	118	
8	F508	CPYSX	0E	130	
9	F509	CPYXS	0F	131	
10	F510	LBSET	14	124	
11	F511	CMPXA	B4	125	
12	F512	KEYPOINT	B1	136	
13	F513	EXECUTE	C0-7	139	

4 OPERATOR COMMUNICATION

Communication with FCT is via any console that can be driven by CMSE. All communication between the operator and FCT through CMSE.

4.1 DISPLAYS

4.1.1 FCT1 and FCT5 Displays

Initial Display

The initial display presented by FCT1 or FCT5 is shown below. The format of the FCT5 display is identical to that of the FCT1 display with FCT1 changed to FCT5, and the file name, F1xx changed to F5xx. The test name is at the beginning of the line, and the date on the display gives the assembly date of ITC program. Refer to Operator Entries for a detailed description of the keyboard commands.

```
FCT1  PARAMS PA=122B 80/12/11.REV 2.4

            KEYBOARD COMMANDS

ABS/ABB                 ABORT SECT/SUBSEC
ECPX/DCPX               EN/DISABL CPU X=0+1
EPFS/DPFS               EN/DISABLE PFS UTRAPS - P1,P2,P3 ONLY
EP,4,142,XXXX           SET CP HANG COUNT
F1XX YY ZZ  -           TEST SUB-SET OF OP CODES
S/R/D/SPACE             STOP/RESTRT/DROP/CONTINU

CONTROL DATA PROPRIETARY PRODUCT
COPYR.  CONTROL DATA 1980
```

Running Display

The running display presented by FCT1 (and FCT5) is as follows.

```
FCT1  op  PCXXXX CPX SXXXX  SBXXXX CXXXX  F1XX  yy/mm/dd
```

where:

operation (op)

RU	Running message
SC	End of condition
SB	End of subsection
SS	End of section
ST	End of test
SE	Stopped on error
SM	Executing scope loop
HT	Test halted by operator

	RC Repeat condition
	RB Repeat subsection
	RS Repeat section
PCxxxx	Pass count
CPx	Current CPU
Sxxxx	Current section number
SBxxxx	Current subsection number
Cxxxx	Current condition number
F1xx or F5xx	File name of the current section being executed; xx is the command op code
yy/mm/dd	Date of last modification of current section

PARAM3 controls testing for dual CP systems. Either one or both of the CPs may be tested. When two CPs are tested, execution occurs alternately, not concurrently. CP0 is tested first for any given section and, when testing is complete, the section is repeated for CP1.

NOTE

If the repeat section paramater is set, the section is repeated for the currently executing CP. If the section stop bit is set the test stops at the end of section for each processor.

Data Comparison Error Display

NOTE

There may be minor variations in the spacing (by byte or parcel) for some of the displays represented here.

The data comparison error display issued by FCT1 is shown below. The format of the FCT5 display is identical to that of the FCT1 display with FCT1 changed to FCT5, and the file name, F1xx changed to F5xx. Except for the S register and the address index, all addresses displayed are byte addresses.

```
FCT1  SE  PC0000 CP0 S0000  SB0000 C0001  F100  80/10/30
EC1=0000  TE=0001

OP CODE 00  REF 0121
P REG          00 00 B0 00 00 00 40 30
S REG          00 00 04 31 04 31 04 31
MON PACK       004100  PROCESS  0000 B000 0000 4030
JOB PACK       000000  PROCESS  0000 0000 0000 0000
COMPARE NO     0005
MASK           FFFF FFFF FFFF FFFF
XPCTD          0000 0000 0000 0000
RECVD          0000 0000 0000 1000
ADRS OF XPCTD 0044E8
ADRS OF RECVD 000088
ADRS INDEX    000000
```

where:

EC1	Error code 1. EC1 equals $1C20_{16}$ when the MAC channel was unable to transfer data after a function was sent by the driver program. No other EC1 values have been implemented. For other errors EC1 will remain set to 0000.
TE	Total errors since start of test to date.
RN=XXXX	(Not used).
OP CODE - REF	Current instruction op code and reference number.
P REG	Contents of P register.
S REG	Contents of S register.
MON PACK	Byte address of monitor exchange package.
PROCESS	Contents of first word in monitor package (P).
COMPARE NO.	Indicates the comparison, starting with 1, within the current condition that caused the error. The comparison number is used to correlate the display with the listing, by indicating which compare directive within the condition has reported the error.
MASK	Mask used to indicate which bits of the received word are being tested.
XPCTD	Expected data.
RECVD	Received data.
ADRS OF XPCTD	Byte address pointing to the word containing the expected data or the word containing the beginning of the expected data. The 64 bits of an expected data may begin on any parcel boundary. For immediate data, address is the byte address of the word where immediate data started. When expected data is not immediate data, this is the byte address of the word containing the (first) 64 bits of expected data.
ADRS OF RCVD	Byte address pointing to the word containing the received data.
ADRS INDEX	When nonzero, indicates a block comparison occurred. The index is not used when referencing immediate data. When the expected data is not immediate data, the index is a word offset from the expected and received addresses that points to the data that was used in the failing comparison. When multiple errors are detected, only data for the first detected error is displayed.

An error directory for FCT1 and FCT5 is available to give more descriptive information about what is occurring, or not occurring as the case may be. The error directory is found in the listings. Each condition describes its own errors.

System Register Error Display

When the test detects an error while checking any register via the Maintenance Access Control (MAC) Channel then a display similar to that shown below will occur.

```
FCT1  SE  PC0000 CP0 S0000  SB0000 C0001  F100  80/10/30
EC1=0000  TE=0001

OP CODE 00  REF 0121
P REG          00 00 B0 00 00 00 40 30
S REG          00 00 04 31 04 31 04 31
MON PACK       004100  PROCESS  0000 B000 0000 4030
JOB PACK       000000  PROCESS  0000 0000 0000 0000
COMPARE NO     0005
MASK           FF FF FF FF FF FF FF FF
XPCTD          00 00 00 00 00 00 00 00
RECVD          00 00 00 00 00 00 00 00
REG FUNC       0240
REG ADRS       0048
```

Lines 0 through 11 are as defined for the data comparison error display.

REG FUNC — Contains the connect code, op code and type code as transmitted to the MAC channel to read the register contents.

REG ADRS — Address of the register within the processor.

Processor Hung Error Display

When the CP does not halt after testing an instruction, the following message appears. Note that a similar display may appear when the required system microcode has not been loaded.

```
FCT1  SE  PC0000 CP0 S0008  SB0001 C0001  F10E  80/10/30
EC1=0000  TE=0001

OP CODE 0E  REF 0130
P REG      00 00 B0 00 00 00 40 34
S REG      00 00 04 31 04 31 04 31
MON PACK   004100  PROCESS  0000 B000 0000 4030
JOB PACK   004280  PROCESS  0000 B000 0000 4068

               CP HUNG
```

where:

EC1	Error code 1
TE	Total errors
OP CODE - REF	Current instruction op code and reference number
CP HUNG	Processor hung indication

Program Error Display

When the PP driver program detects an error while interpreting control commands, the following display is presented. The error could be caused by bad data being loaded from disk to central memory or by an overwrite of the control tables in central memory as a result of a CP error.

```
FCT1  SE  PC0000 CP0 S0008  SB0001 C0001  F10E  80/10/30
EC1=0000  TE=0001

OP CODE OE REF 0130
P REG          00 00 B0 00 00 00 40 34
S REG          00 00 04 31 04 31 04 31
MON PACK       004100  PROCESS  0000 B000 0000 4030
JOB PACK       004280  PROCESS  0000 B000 0000 4068

               PROGRAM ERR.
```

where:

EC1	Error code 1
TE	Total errors
OP CODE - REF	Current instruction op code and reference number
PROGRAM ERR.	Program error indication

4.1.2 FCT2 Displays

Initial Display

This display will appear after the CN command, when FCT2 is loaded as described in this document. The display will change to the FCT2 Running display after the processor has been deadstarted and the test has begun execution. If the display does not change then the processor is not functioning correctly and FCT1 should be run. This display may also remain unchanged if the correct microcode has not been loaded.

The initial display presented by FCT2 is as follows:

```
FIXED OPERAND COMMAND TEST 2
VERIFIES USAGE SET FOR FCT3, AND RCT1

PARAMETER WORD 0 IS AT WORD ADDRESS 608
   BIT 54  -  REPEAT CONDITION
       55  -  REPEAT SUBSECTION
   BIT 56  -  REPEAT SECTION
       57  -  REPEAT TEST
       58  -  LOG ERRORS
       59  -  STOP ON ERROR
   BIT 60  -  STOP AT END OF CONDITION
       61  -  STOP AT END OF SUBSECTION
       62  -  STOP AT END OF SECTION
       63  -  STOP AT END OF TEST

PARAMETER WORD 2 (60A) CONTAINS SECTION ENABLE BITS
```

Running Display

The running display presented by FCT2 is as follows:

```
FIXED OPERAND COMMAND TEST 2
VERIFIES USAGE SET FOR FCT3, AND RCT1

Sxx    mmmmm       OP = xx      REF  = xxx

SUB-SECT  xx      CONDITON  xx     IDX= xx  IDX= xx
```

where:

Sxx	Current section number
SUB-SECT xx	Subsection number
CONDITON	Condition number
mmmmm	Instruction mnemonic
OP	Op code
REF	Reference number
IDX	Loop index values

Error Display

The standard error display format used by FCT2 is as follows:

```
FIXED OPERAND COMMAND TEST 2
VERIFIES USAGE SET FOR FCT3, AND RCT1

S10    SHFR      OP = AA     REF = 064

SUB-SECT  00    CONDITON  02

STOP. SHFR ZERO IS NOT A CPYRR
```

where:

Line 0 and 1: FCT2 header

Line 2: Not used

Line 3:

S10	Test section number (10 decimal in example)
SHFR	Mnemonic
AA	Operation code
064	Reference number

Line 4: Not used

Line 5:

00	Subsection number
02	Condition number

Line 6: Not used

Line 7: This line contains a one-line error message appropriate to the problem detected. Refer to the FCT2 program listing for a detailed description of the failing section to determine which registers are being compared and for the origin of their contents.

FCT2 issues the message, STOPPED, for end of test, end of section, end of subsection, and end of condition. The type of stop can be determined by observing the section, subsection, and condition numbers. Zero, one, two, or three of these numbers are displayed for the end of test, section, subsection, and condition respectively. An error stop is indicated when the error message line appears with the normal running display.

4.1.3 FCT3 Displays

Initial Display

The initial display presented by FCT3 is as follows:

```
FCT3  **  PC00000  S***  SS**  C**
EC1=     EC2=     TE=     0
```

The initial display is an uninitialized template of the standard running display. The display may flash on the screen during the initialization process done by the FCT3 Common Routines. If this display ever remains on the screen, the Common Routines have not completed the initialization and the CPU has hung up. This is usually caused by a failure of an instruction in the usage set, and indicates that either FCT1 or FCT2 should be run to verify the execution of all usage set instructions.

If FCT3 makes no display at all, then the usage set instructions may not be functioning. This also may be the result if the correct microcode has not been loaded into the processor.

Running Display

The running display presented by FCT3 is as follows:

```
FCT3  op  PCxxxxx  Sxxx  SSxx  Cxx  LP1= xxx  LP2= xxx  LP3= xxx
EC1=     EC2=     TE= xxxxx


ADDD       OP = 34       FP    JK        REF = 105
FP DP SUM,        (XK, XK+1) = (XK, XK+1) + (XJ, XJ+1)
SECTION REVISED 80/05/31,    ASSEMBLED 80/06/10.
```

where:

Line 0:

Operation (op)	RU	Running message
	SO	Stopped for operator action
	SL	Stopped at end of loop
	SC	Stopped at end of condition
	SB	Stopped at end of subsection
	SS	Stopped at end of section
	ST	Stopped at end of test
	SE	Stopped on error
PCxxxxx	Pass count is xxxxx	
Sxxx	Current section number is xxx	
SSxx	Current subsection number is xx	
Cxx	Current condition number is xx	
LP1= xxx	Loop 1 index value is xxx (shown when used)	
LP2= xxx	Loop 2 index value is xxx (shown when used)	
LP3= xxx	Loop 3 index value is xxx (shown when used)	

Line 1:

EC1	Error code 1 (not used)
EC2	Error code 2 (not used)
TE=xxxxx	Total number of errors found since start of test is xxxxx

Line 2: Used by COMMON routines to describe operator action requests or parameter errors.

Line 3: Not used (blank)

Line 4: Description of instruction under test including:
Mnemonic
Operation code
Instruction type and format
Reference number

Line 5: Description of instruction function

Line 6: Date line; the last code revision for section; assembly date for section

Line 7: Not used

Lines 8 through 19: Usually blank; content depends on section being run and parameter settings (See FCT3 Error Messages).

Standard Error Display

The first eight lines (lines 0 to 7) of the FCT3 error display are always in the form described for the FCT3 running display.

The last 12 lines (lines 8 to 19) are test-defined and vary in format from condition to condition. Two kinds of information are displayed on these lines. The first lines describe the initial conditions used for the test and are only displayed when the display only error messages parameter is cleared (zero). The remaining lines describe actual errors that have been detected when the no result checking parameter is cleared (zero). Some lines at the bottom of the display may be unused.

Every effort has been made to include in the test defined display all of the information needed to understand the nature of the failure. This has resulted in the use of execution time generated error messages which are too numerous to list.

If additional information is required to understand an error, refer to the test case descriptions for the failing subsection.
Abbreviated test case descriptions are included later in this document. Full descriptions are in the section listings near the front of the listing and are also available as a separate document. Descriptions are sequenced by reference number. Two examples of these test defined messages are given below.

A sample FCT3 error display is shown below.

```
FCT3  SE  PC00000 S123  SS00  C00  LP1= 000  LP2= 016  LP3= 063
EC1=5 EC2=5 TE= 00001

ADDD       OP = 34       FP    JK        REF = 105
FP DP SUM,         (XK, XK+1) = (XK, XK+1) + (XJ, XJ+1)
SECTION REVISED 80/05/31,    ASSEMBLED 80/06/10.


XJ           4000800000000000 0000000000000000
XK INITIAL   4000000000000001 0000000000000000
XK EXPECTED  4000800000000001 4000000000000000
XK RECEIVED  4000800000000000 4000000000000000
DATA ERROR
```

Note that the indexes for LOOP 2 and LOOP 3 have been used to indicate the bit numbers which are being tested in the Xj and Xk data patterns.

In this example, lines 8 through 12 give the initial conditions for the test and lines 11 and 12 give the error description.

A sample BDP error display sample is shown below:

```
FCT3  SE  PC00000  S170  SS02  C00  LP1= 054  LP2= 000
EC1=     EC2=     TE= 00001


TRANB      OP = EB      BDP   JKID(2)   REF = 088
BYTE TRANSLATE,      (D(AK)) = (D(AJ)), PER (AI) + D
SECTION REVISED 80/05/31,    ASSEMBLED 80/06/10.

INSTRUCTION = EBA9B000 83000000 84000000  ADDRESS = B00F000258A2
AJ = B00B00001FB8  AK = B00B00000F38  AI = B00B00000C20
X0 = 0000000000000036  X1 = 0000000000000000
SOURCE  B00B00001FB8 = 000102030405060708090A0B0C0D0E0F101112....
TABLE   B00B00000C20 = 000102030405060708090A0B0C0D0E0F101112....
INITIAL B00B00000A10 = FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF....
EXPECTED DEST 000B18 = FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF....
RECEIVED DEST 000F38 = 00FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF....
FAILING BYTE LIST  = 8000000000000000 0000000000000000
(ONE BIT PER BYTE)   0000000000000000 0000000000000000
DATA ERROR
```

Note that the indexes for LOOP 1 and LOOP2 indicate only the test case that has failed. In this example, lines 8 through 14 give the initial conditions for the test and lines 15 through 18 give the error description as follows:

Line 8: This line gives the test instruction and descriptors, and the virtual address of the test instruction. This information is required for the BDP instructions because instruction modification is used to alter the descriptors and the Q or D instruction fields.

Line 9: This line gives the contents of the address registers used for Aj, Ak, and Ai. The register numbers used are shown on line 8.

Line 10: This line gives the contents of the (length) registers X0 and X1.

Line 11: This line gives the virtual address of the source field and the data in memory. Only the first 19 bytes of memory are displayed. The ... at the end of the line indicates that the actual source data may extend off the screen. (Look at the instruction itself to determine the exact length used.) The remaining data can be seen by using CMSE commands to display memory at the given address, but the tests have been set up so that this is rarely necessary.

Line 12: This line gives the virtual address of the translation table and the data in memory. The ... indicates continuation.

Line 13: This line gives the virtual address and contents of the field used to initialize the destination field before the test.

Line 14: This line gives the virtual address and contents of the field used to hold the precalculated expected results. The upper bits of the PVA are not shown but will be the same as those displayed on line 13.

Line 15: This line gives the virtual address and (final) contents of the field used as the destination for the test instruction. The upper bits of the PVA are not shown but will be the same as those displayed on line 13.

Lines 16 and 17: These lines indicate which bytes of the received results do not match the expected results. The nth bit will be set if the nth byte did not match. This compressed error indication is required because of the possibility of lengthy expected and received fields. All 256 bytes of expected and received data are always checked, even if the instruction length is (supposed to be) shorter. The bits make it easy to spot failure patterns. 80808080... would indicate a failure in the leftmost byte of each word. The data on line 17 is a continuation of line 16.

Line 18: Error indication.

4.2 OPERATOR ENTRIES

FCT tests use standard CMSE commands as described in the MSL 15X Reference Manual.

4.2.1 FCT1 and FCT5 Commands

The following commands are provided by the PP driver program and DEX for FCT1 and FCT5. Except for the space bar command, all commands are terminated by a carriage return.

COMMAND	FUNCTION
(space)	Continues test execution from point at which it stopped.
S	Stops test execution at next condition boundary (or at any other time when the test gives control to DEX). This stop does not alter the contents of the (op) field of the display. The display continues to show the previous op.
R	Restarts test from first selected section.
D	Stops test execution and idles the test driver PP. The test cannot be restarted.
ABS	Aborts section and skips to the next section (normally the next op code) of the test. The CE enters this command at an error stop which causes a skip to the next section.
ABB	Aborts subsection and skips to next subsection of the test. Entered at an error stop.
ECP0/DCP0 ECP1/DCP1	Enables or disables test execution for CP0 or CP1 on dual CP systems. These commands are ineffective on single CP systems.
EPFS/DPFS	Enables or disables the PFS micro-traps. (Comparable to setting or clearing a bit in PARAM 17.)
F1xx/ F5xx xx	Selects one or more test sections for execution. xx is the op code of the command to be tested. Thus F502 selects op code 02 of FCT5 and F1D0 selects op code D0 of FCT1. The command entry is the file name and the appropriate section. These commands are entered from the keyboard or command buffer when the test stops for parameter entry. They cannot be entered after test execution commences. When used in a command buffer, they should immediately follow the RU command that gives execution control to ITC. Entry of the command F123 85 00 D0 0E would select five different sections of FCT1. More than one command entry can be used to build up a large series of sections to be run. Test sections are run in the sequence selected.

Some sections that have special characters, rather than simple opcodes, are listed below:

F1UI - for all unimplemented opcodes.
F1C0 - for opcodes C0 through C7 (SjkiD format).
F1D0 - for opcodes D0 through D7 (SjkiD format).
F1D8 - for opcodes D8 through DF (SjkiD format).
F5PC - for purge cache (part of 05 opcode).
F5PM - for purge map (part of 05 opcode).

4.2.2 FCT2 Commands

When FCT2 is stopped, entry of RUN followed by a carriage return will continue the test. To alter parameters, use the EC and EB CMSE commands described in the MSL 15X Reference Manual.

4.2.3 FCT3 Commands

When FCT3 is stopped, entry of RUN followed by a carriage return will continue the test. To alter parameters, use the EC and EB CMSE commands described in the MSL 15X Reference Manual.

4.3 NORMAL MESSAGES

Other than the initial and running displays described previously, FCT tests do not display any test messages for normal operation.

4.4 ERROR MESSAGES

4.4.1 FCT1 and FCT5 Error Messages

Information regarding error messages displayed by FCT1 and FCT5 is provided in error directories preceding each test listing. These error directories are organized by section numbers. Each error condition is described.

4.4.2 FCT2 Error Messages

Information regarding error messages displayed by FCT2 is provided in the listing of FCT2 in each section description.

4.4.3 FCT3 Error Messages

Information regarding data error messages displayed by FCT3 is provided in test case descriptions preceding the listings for each test section.

These descriptions are also presented in a separate listing where they are organized by reference number.

Common Routine Error Messages

In addition to the test error messages, the error messages issued by FCT3 common routines always display the following messages on line 2 of the display:

MESSAGE	EXPLANATION
SECTION NUMBER OUT OF VALID RANGE	Invalid parameter entry
A NONEXISTENT SECTION IS ENABLED	Invalid parameter entry
SUB-SECTION NUMBER OUT OF VALID RANGE	Invalid parameter entry
THIS SUB-SECTION IS NOT LINKED	Invalid parameter entry

Note that line 2 of the display is also used for subsection omitted messages and for VLEX error messages. Refer to the MSL 15X Reference Manual.

Subsection Omitted Messages

The subsection omitted messages issued by an FCT3 test section appear on line 2 of the display. A number of sections are run only for specific settings of the parameter bits. For example, some subsections run only in monitor mode and others run only in job mode. Thus, there are always some subsections omitted when FCT3 is run. Parameter requirements are described as part of the test case descriptions.

The following subsection omitted messages may be displayed:

```
OMITTED Ssss SSbb DUE TO MONITOR MODE PARAMETER
OMITTED Ssss SSbb PER MTR.MODE PARAMETER
OMITTED Ssss SSbb PER CACHE PARAMETER
OMITTED Ssss SSbb PER MTR.MODE/EXCHANGE PARAMETERS
OMITTED Ssss SSbb PER MODEL PARAMETER
OMITTED Ssss SSbb PER MTR.MODE/EXCH/MODEL PARAMETERS
OMITTED Ssss SSbb DUE TO PORT 1 PARAMETER.
```

where:

sss	section number
bb	subsection that was omitted

The message lists the parameter bits examined by the subsection. This line is rewritten each time a subsection is omitted, so only the most recent message will appear.

Note that line 2 is also used for common routine error messages and for VLEX error messages. Refer to the MSL 15X Reference Manual.

SCOPE MODE CONTROL of FCT1 and FCT5

Use CMSE command SSM to set scope mode while the test is at a stop. When execution is restarted with a space bar, ITC sets the repeat condition bit to force the looping of the current condition. If a number of errors have occurred during the condition, more than one space bar entry may be required to reach end of condition and enter loop. Entry to the loop is confirmed by observing activity of MAC channel flag on the display, and the appearance of SM code appears on the running display line.

The test may be pulled out of the scope loop with the CMSE command UP,x where x is the PP in which ITC resides. ITC stops at end of condition regardless of state of Stop at End of Condition bit. It also stops on an error when Stop on Error bit is set. Reenter the scope loop by pressing space bar. The CSM command to clear scope mode must be entered to allow continuation to the next condition or end of subsection.

Some conditions execute an exchange instruction altering the P register in the exchange package. These conditions cannot enter the normal scope loop. They appear to work correctly, but do not loop the test code. An effective loop is created by repeating condition with display off. Conditions that exhibit this behaviour are:

FCT1, Section 118, all conditions except
Subsection 0, condition 0,1
FCT5, Section 0, all conditions except
Subsection 0, each condition,
Subsection 1, condition 0,1,
Subsection 2, condition 0,1,3,5,7.

Some conditions initialize central memory locations that are destroyed by the test code. If put into scope mode, they will report errors on exiting the scope loop. These conditions may be looped by repeating the condition:

FCT5, Section 8, subsection 21, condition 21,
Section 9, subsection 2, condition 1,
Section 10, subsection 1, conditions 65-128.

In all conditions of the test where the CP is required to execute an instruction level code sequence, the operation is initiated with a deadstart which is used as a scope trigger.

Some conditions in FCT1 and FCT5 have no deadstart. If scope mode is entered within one of these conditions it behaves as though the repeat condition and stop at end of condition bits are set. When the scope mode bit is cleared, the test is controlled by the explicit setting of the parameter bits.

SECTION I-5

VIRTUAL MODE INSTRUCTION LEVEL TEST - FCT9

1 INTRODUCTION

FCT9 is a PP-based instruction level test of redefined features of the A170 system. The features tested are:

1) The positive operation of the redefined A170 instructions (CRXj, CWXj, RXj, WXj, REC, and WEC). Instructions with ECS capability are tested in UEM (ECS mode) only.

2) The end case exit conditions of the redefined A170 instructions.

3) Error exits in a system environment, where a system environment consists of an A170 processor, the Environment Interface and a page table.

4) The Environment Interface's processing of the RT (017_8) and CMU instructions.

Included in the above testing is the interruptability of the block copy and the CMU instructions.

The test uses the PP controller, EXCHITC, which has facilities to generate PP interrupts (exchanges) and scope loops.

2 REQUIREMENTS

2.1 HARDWARE

Equipment for which test is intended:

Models 810, 815, 825, 830, 835, 845, and 855 Computer Systems.

The test requires a central processor, the product set microcode, 131K word of central memory, IOU hardware required by CMSE, and IOU hardware required by the EXCHITC controller. EXCHITC requires one PP as the master, one channel for master to slaves communication, and requests four other PPs and their channel as slaves.

2.2 SOFTWARE

The test controller, EXCHITC, uses the facilities of the Diagnostic Executive, and the Common Maintenance Software Executive.

FCT9 uses the Environment Interface to interface to the processor and also as part of the A170 system being tested.

2.3 ACCESSORIES

None required.

2.4 CHARACTERISTICS

1.	Test name	FCT9
2.	Test/diagnostic/utility/system	Test
3.	Size (source) PP code	6000 lines
	CP code	10000 lines
4.	Size (memory required for execution)	
	PP code	4096 PP words
	CP code	131000 CM words minimum
5.	Virtual code/microcode/PP code/other	Virtual & PP code
6.	Run Time	Approx. 30 sec
7.	Level of isolation	detection
8.	On-line/off-line/both	off-line
9.	Off-line system	MSL15X
10.	Resident during execution	Yes
11.	Language	CP: CYBER VM Assembler
		PP: Compass
12.	Source code maintenance	CP: MODIFY (ASCII)
		PP: UPDATE
13.	Uses maintenance channel	Yes

3 OPERATIONAL PROCEDURE

3.1 RESTRICTIONS AND USER CAUTIONS

The Environment Interface is used as an interface between the processor and the test. It is modified in order to create the appropriate environment for each condition of the test. Therefore, any change in the format or function of the EI can affect the results of the test.

FCT9 is model independent. The test controller EXCHITC runs on Models 810, 815, 825, 830, 835, 845, and 855 Computer Systems.

3.2 LOADING PROCEDURE

The object code of EXCHITC and its associated overlays plus the FCT9 object code must reside on the Maintenance Software Library (MSL) device. The file names on the MSL for these binaries are EXCHITC, EXCHP1, EXCHP2, EXCHP3, EXCHP4, EXCHOV, EXCHSLV, EI, and FCT9.

Load product set microcode.

Command buffer FCT91, FCT92, or FCT93 (1 for model 810/815/825/830, 2 for 835, 3 for 845/855) exists on the MSL 15X tape to facilitate loading and execution of this test. Before using this command buffer you may have to modify it for your system. Display the command buffer using the CMSE command buffer display commands or print the contents of the command buffer using procedures provided in the Command Buffer Maintenance section of the MSL 15X Reference Manual. Then modify the command buffer as directed by comments embedded in the command buffer. When you are satisfied that the command buffer is set up properly, enter a GO,FCT9x command to execute it, where x is the model dependent number from above.

Modified command buffers can be saved on a back-up tape for future use. Refer to the Utilities section of the MSL 15X Reference Manual for procedures.

The command buffer performs the following steps:

- Initialize system (ie. master clear, clear errors, and initialize registers of individual units).
- Load Environment Interface.
- Load FCT9.
- Deadstart PP, load EXCHITC to PP, assign display to PP, and set PP running.

Some of the parameters provided by EXCHITC do not apply to FCT9 and are indicated as not used.

The following paragraphs describe the loading of EXCHITC, its messages, parameters, and operator entries.

3.2.1 Loading Processor Dependent Overlays

After the command buffer initiates the execution of EXCHITC, it accesses the element ID register of the subsystem on each radial interface to determine the type of IOU, processor, and memory which are a part of the system. Based on this information, the appropriate connect and type codes are set up for later use. Determination of the type of central processor allows EXCHITC to adapt to different machines. An overlay specific to the processor type (EXCHP1, EXCHP2, EXCHP3, EXCHP4) is loaded into the EXCHITC PP to account for most of the differences. In the few cases where differences cannot be conveniently handled by the overlays, code modification is employed. The initializing code modifies constants and command operands in the permanent code of EXCHITC.

3.2.2 Loading of Slave PPs

The loading of slave PPs commences after the initial parameter stop. The sequence for doing this is described below. The number of the PP which will be selected as the the first possible slave PP is defined by PARAM11; default is PP0. If EXCHITC is already residing in PP number 4, the slaves would normally be loaded into PPs 5,6,7, and 11_8.

Parameters 11, 12, 13, and 14 are the PPs to be used as slaves. The search for available PPs proceeds sequentially from PP0. If PARAM11 is nonzero, the search commences from that nonzero value. Only PPs for which the corresponding channel number is available will be selected. PPs and channels used by CMSE will be skipped automatically.

If desired a specific set of four slaves may be selected by entering the PP numbers in parameter words 11 thru 14. A nonzero value in PARAM12 is recognized by EXCHITC as a signal to use the four parameter values as the required PPs.

NOTE

In this mode, EXCHITC does not check to determine if the PPs are actually available and provides no warning if they are not.

Parameter word 15 is used to control the selection of the channel used by EXCHITC to communicate with the four slaves. This parameter is defaulted to 0 and, as such, either channel 12 or 13 will be selected depending upon which of these is not in use by CMSE to communicate with the MSL device. If neither of these is available, the number of some other suitable channel must be entered in the parameter word.

The sequence employed by EXCHITC for setting up the channels and loading the slave PPs is as follows:

a) Select and request use of the channel which is to be used for EXCHITC to slave PP communication.

b) If PARAM 12 is zero, skip to step c); otherwise, assign PPs as slaves according to parameters 11 thru 14 and then skip to step f).

c) Starting with the PP defined (by PARAM 11) to be the first slave PP, scan to determine if that PP is: PP 0 the CMSE I/O driver PP; the EXCHITC PP; the CMSE monitor PP; the CMSE display PP; the same number as the CMSE MSL device channel; the same number as the CMSE PP communication channel; or, if the PP number is the same as the display channel number 10_8. If none of these is true, assign the PP to be the first (or next) slave PP.

d) If four slave PPs have not been assigned and if all PPs in the system have not been scanned, skip to the next higher PP and return to step c).

e) If no PPs were assigned, display the message TOO FEW PPS and exit to the CMSE idler. The operator should check the entries for parameters 11 through 14.

f) If only one, two, or three slave PPs are assigned, the test continues with only the assigned number of PPs.

g) Sequentially load each of the assigned slave PPs with its object code (EXCHSLV), initiate its program execution and check that it responds by clearing a central memory flag.

h) Check the response of all slave PPs by commanding them to execute dummy operations which will require their simutaneous access of central memory and their assigned channel. If any slave PP hangs up, display the message ABORTED - CHAN OR CM ERR and exit to the CMSE idler.

3.3 PARAMETERS AND CONTROL WORDS

3.3.1 Parameters

Parameter words control the execution of the test. Some of these words are defined in the Maintenance Software Library Reference Manual (MSL15X). These words are located at PP locations 122_8 to 144_8 (directly following the control words).

Parameter words can be set/cleared manually through CMSE commands. Individual bits of parameter word 0 can be set/cleared with a set of CMSE command which references the bits by mnemonics. For example, CSM clears scope mode while SRC sets repeat condition. For more information on CMSE commands see the Maintenance Software Library Reference Manual (MSL15X).

The use of PARAM7 through PARAM18 are peculiar to FCT9. For further details, see the paragraphs referenced below:

PARAM 7 - Not Used
PARAM 8 - Not Used
PARAM 9 - Not Used
PARAM 10 - Not Used
PARAM 11 - First Slave PP - 3.2.2
PARAM 12 - Second Slave PP - 3.2.2
PARAM 13 - Third Slave PP - 3.2.2
PARAM 14 - Fourth Slave PP - 3.2.2
PARAM 15 - EXCHITC to Slave Comm. Channel - 3.2.2
PARAM 16 - Processor hang count
PARAM 17 - Enable/Disable PFS micro-traps
PARAM 18 - Not Used

Parameters are as follows:

Octal Address	Parameter	Octal/Hex Bit	Meaning
122	PARAM0	0001/0001	Stop at end of test (ST)
		0002/0002	Stop at end of section (SS)
		0004/0004	Stop at end of subsection (SB
		0010/0008	Stop at end of condition (SC)
		0020/0010	Stop on error (SE)
		0040/0020	Log errors in dayfile (LE)
		0100/0040	Repeat test (RT)
		0200/0080	Repeat section (RS)
		0400/0100	Repeat subsection (RB)
		1000/0200	Repeat condition (RC)
		2000/0400	Scope mode (SM)
		4000/0800	Quick look (QL),not used
		10000/1000	Bypass all messages (DR)
		20000/2000	Display only err messages (DE)
		40000/4000	Reserved
		100000/8000	Accept CMSE param commands
123	PARAM1	0001/0001	Reserved
		0002/0002	Reserved
		0004/0004	Bypass parameter stop
		0020/0010	Reserved
		0040/0020	Reserved
		0100/0040	Reserved
124	PARAM2		Repeat test count
125	PARAM3	0001/0001	Test CPU 0
		0002/0002	Test CPU 1
126	PARAM4		Reserved
127	PARAM5		Sections 00-12 select
130	PARAM6		Not used
131	PARAM7		Not used
132	PARAM8		Not used
133	PARAM9		Not used
134	PARAM10		Not used

135	PARAM11		First slave PP
136	PARAM12		Second slave PP
137	PARAM13		Third slave PP
140	PARAM14		Fourth slave PP
141	PARAM15		EXCHITC to slave comm. channel
142	PARAM16		Delay count for processor hung
143	PARAM17	0002/0002	Enable PFS register operation
144	PARAM18		Not used

Parameter defaults in octal are as follows:

PARAM0 = 100121	The most significant bit must be set
PARAM1 = 000000	The parameter stop is not by-passed
PARAM2 = 000010	The repeat test count will have effect only if the repeat test bit is clear
PARAM3 = 000003	Test both CPU0 and CPU1 if available
PARAM5 = 017777	All sections selected (007777 if model 810/815/825/830)
PARAM11= 000000	Select slaves sequentially starting with first available
PARAM12= 000000	
PARAM13= 000000	
PARAM14= 000000	
PARAM15= 000000	Use either channel 12 or 13 for for controller to slave communication
PARAM16= 000777	Hang count of 511
PARAM17= 000002	Enable PFS register operation

Section 12 checks the emulation of CYBER 170 compare move commands by the error interface (EI). Some processors implement these commands via microcode and therefore section 12 is not selected.

3.3.2 CONTROL WORDS

Control words are intended to identify a program and supply information to a higher system or operator. They do not normally affect test execution. The control words are located at PP locations 102_8 to 121_8 and are as follows:

CW0	Program name (first 2 characters)
CW1	Program name (last 2 characters)
CW2	Program type
CW3	Monitor ID word
CW4	Error code #1
CW5	Error code #2 (not used)
CW6	Pass counter
CW7	Current section counter
CW8	Current subsection counter
CW9	Current condition counter
CW10	Current error counter

3.4 SECTION INDEX

Due to the structure of FCT9 all the sections are implemented in program FCT9.

Section Number	Brief Description
1	Test positive operation of CRXj(660) instruction.
2	Test positive operation of CWXj(670) instruction.
3	Test positive operation of the RXj(014) instruction in UEM (ECS mode).
4	Test positive operation of WXj(015) instruction in UEM (ECS mode).
5	Test positive operation of the REC (block read ECS,011) instruction in UEM (ECS mode).
6	Test positive operation of the WEC (block read ECS,012) instruction in UEM (ECS mode).
7	Test error exits of CRXj and CWXj instructions.
8	Test error exits of RXj and WXj instructions.
9	Test error exits and half exits for block copy instructions in UEM (ECS mode).
10	Test illegal instruction, address out of range, indefinite, and infinite error exits.
11	Test operation of RT (017_8) instruction.
12	Test simulation of the CMU instruction by the Environment Interface (EI).

4 OPERATOR COMMUNICATION

4.1 DISPLAYS

4.1.1 Initial Display

The FCT9 initial display is shown below. When the display appears, the test stops to allow the operator to enter any parameter changes that may be required. The address of the first parameter word is given in the display. The entry EP,4,142,XXXX is a parameter entry which can be made into the EXCHITC PP memory. If the command buffer is altered such that EXCHITC is not loaded into PP 4 then this entry must be altered accordingly.

```
FCT9   PARAMS PA=122B  80/11/30.REV 2.4

         KEYBOARD COMMANDS

ABS/ABB              ABORT SECT/SUBSEC
ECPX/DCPX            EN/DISABL CPU X=0+1
EPFS/DPFS            EN/DISABL PFS UTRAPS - P1,P2,P3 ONLY
EP,4,142,XXXX        SET CP HANG COUNT
S/R/D/SPACE          STOP/RESTRT/DROP/CONTINU

CONTROL DATA PROPRIETORY PRODUCT
COPYR. CONTROL DATA 1980
```

4.1.2 Running Display

The normal running display for FCT9 is shown below. In addition, a date indicating when the test was last modified will be appended to the extreme right of all the displays.

```
FCT9 (op) PCxxxx  CPx  Sxxxx  SBxxxx  Cxxxx
```

OPERATION (op)	RU	Running Message
	RC	Repeating Condition
	RB	Repeating Subsection
	RS	Repeating Section
	SC	End of Condition
	SB	End of Subsection
	SS	End of Section
	ST	End of Test
	SE	Stopped on Error
	SM	Executing in scope mode
	HT	Test halted by operator

PC	Pass Count
CP	Current CPU
S	Current Section Number
SB	Current Subsection Number
C	Current Condition Number

Normally the RB and RS codes are not seen when a subsection or section is being repeated. Instead, a dynamic display of the current subsection and section is provided by the RU code. The display is updated at every subsection boundary.

PARAM3 controls testing for dual CP systems. Either one or both of the CPs may be tested. If two CPs are tested, CP0 is tested first for any given section and, when testing is complete, the section is repeated for CP1.

NOTE

If the repeat section paramater is set, the section is repeated for the currently executing CP. If the section stop bit is set the test stops at the end of section for each processor.

4.2 FCT9 ERROR MESSAGES

4.2.1 Data Comparison Error Message

A standard FCT9 error message for a data comparison error is shown below.

```
FCT9  SE  PC0000 CP0 S0008  SB0002 C0021          YY/MM/DD
EC1=0000 TE=0001

ACTIVE SLAVES   PP 05 06 07
P REG          00 00 B0 00 00 00 40 34
S REG          00 00 04 31 04 31 04 31
MON PACK       004100  PROCESS  0000 B000 0000 4030
JOB PACK       004280  PROCESS  0000 B000 0000 4068
COMPARE NO     0003
MASK           FFFF FFFF FFFF FFFF
XPCTD          0000 0000 0000 0000
RECVD          0000 0000 0000 1000
ADRS OF XPCTD 0044E8
ADRS OF RECVD 000088
ADRS INDEX     0000
```

The first line is similiar to that for a running display; the second line provides an error code and the total errors as follows:

EC1	Error Code 1
TE	Total Errors since test first start

The EC1 error code is set with a value of $1C20_{16}$ when the MAC channel is found unable to transfer data after a function was sent by EXCHITC.

If the address index is nonzero, a block comparison occured and the index is a word offset from the expected and received addresses pointing to the data in error within the block. In the event of multiple errors in a block comparison, only the data for the first detected error is displayed.

The addresses of the monitor and job exchange packages are provided along with the value of the P register word contained in the packages. It is not possible to provide equivalent information pertaining to the CYBER 170 exchange packs.

Aside from the S register and the address index, all displayed addresses are byte addresses. There are minor variations in the spacing (by byte or parcel) for some of the data which is displayed. These variations are a result of differences between the various model processors.

4.2.2 Processor Hung Error Message

When a central processor operation is initiated and does not subsequently halt, a display similiar to the following may appear.

```
FCT9  SE  PC0000 CP0 S0008  SB0000 C0001          YY/MM/DD
EC1=0000 TE=0001

ACTIVE SLAVES   PP 05 06
P REG           00 00 B0 00 00 00 40 34
S REG           00 00 04 31 04 31 04 31
MON PACK        004100  PROCESS  0000 B000 0000 4030
JOB PACK        004280  PROCESS  0000 B000 0000 4068

PROCS HUNG      CP  PP 06
```

The last line indicates the central processor and slave PP 6 are hung up. Typically, if the CP hangs in the sequence of commands it is executing, any PP which has sent an exchange request may also hang up because it will fail to receive an exchange accept from the CP.

When a processor hangs up, the S and P registers are read immediately for the purpose of displaying their values. EXCHITC will subsequently send a special deadstart command to the processor which forces exchange accepts to be sent to the PP system. This action will release all slave PPs that may have hung and will allow EXCHITC to regain control of them. If on such an occasion the operator observes the value of S or P via the CMSE AR display, they may not correspond.

4.2.3 Program Error Message

An error display similiar to that shown above may occur except the last line reads PROGRAM ERR. This indicates EXCHITC detected an error in the process of interpreting the control commands which it reads from central memory. The error could be caused by bad data loaded from disc to central memory, or by an over-write of the control tables in central memory as a result of a CP error.

4.2.4 System Register Error Display

In the event an error is detected while checking either a processor hardware register or a processor soft register (dedicated register file register accessed by microcode) a display similiar to that shown below occurs.

```
FCT9  SE  PC0000 S0000  SB0000 C0003                YY/MM/DD
EC1=0000 TE=0001

ACTIVE SLAVES   PP 00
P REG           00 00 B0 00 00 00 40 30
S REG           00 00 04 31 04 31 04 31
MON PACK        004100  PROCESS  0000 B000 0000 4030
JOB PACK        000000  PROCESS  0000 0000 0000 0000
COMPARE NO      0005
MASK            FFFF FFFF FFFF FFFF
XPCTD           0000 0000 0000 0000
RECVD           0000 0000 0000 1000
REG ADRS        0048
```

The register address is the address of the register within the central processor or central memory subsystem.

4.2.5 Miscellaneous Error Messages

The following single line error messages may also occur. In all cases, EXCHITC exits to the CMSE idler since recovery is not possible.

Message	Meaning
ABORT - MAC ERR	An error occured during initialization when ITC was reading the EID registers on each radial interface.
ABORT - ELEMENT ID ERR	During initialization while reading EID registers, too few responses were received. Apparently a system element was not connected.

Message	Meaning
ABORT - TOO FEW PP S	During the loading of the slave PPs, EXCHITC was unable to find any slave PPs available for use. The entry for PARAM 11 should be checked. See also section 3.2.2.
CPX NOT A170 CAPABLE	Prior to executing the first condition of the test, the C170 bit of the VMCL register and the SOLO C170 bit of the options installed register are checked to ensure that they are set. This error is more likely to occur in dual CP systems where the second CP may not have A170 capability. If this is so, deselect the parameter bit for the second CP.
ABORT - CHAN OR CM ERR	After the slave PPs are loaded, EXCHITC checks to determine if it can communicate with them. The error may be a result of a channel problem or a PP to CM interface problem possibly caused by 4 slaves accessing CM simultaneously. See also section 3.2.2.
ABORT - SLAVE PP HUNG	During normal execution of the test, an exchange accept problem may arise which causes a slave PP to hang up on a 26XX command. When this occurs, EXCHITC initiates a sequence which should force the CP to send exchange accepts to the IOU system. If this operation fails to release the slave or slaves from the 26XX command this message should appear.

4.3 OPERATOR ENTRIES

Aside from the operator entered commands which are provided by CMSE, EXCHITC provides additional commands as follows:

Action	Command	Description
SPACE BAR	Continue Execution	Entered after any stop to cause execution of the test to continue.
S	Stop	Entered while the test is executing to cause a halt at the next condition boundary.
R	Restart	Restarts the test from the first selected section.

Action	Command	Description
D	Drop	Halts test execution and requests CMSE to deadstart all slave PPs and the EXCHITC PP. The test cannot subsequently be restarted.
ABS	Abort Section	Entered by the operator after an error stop when it is desired to skip to the next section of the test.
ABB	Abort Subsection	Entered by the operator after an error stop when it is desired to skip to the next subsection of the test. This command is ignored if the test is already at an end of subsection stop.
ECP0/DCP0 ECP1/DCP1		Enables or disables test execution for CP0 or CP1 on dual CP systems. These commands are ineffective on single CP systems.
EPFS/DPFS	Enable/Disable PFS micro-traps	These commands provide a more convenient method of enabling or disabling the PFS micro-traps as opposed to setting or clearing a bit in PARAM word 17.

SECTION I-6

EXCH - EXCHANGE TEST

1 INTRODUCTION

This Exchange test (EXCH) includes, and operates in conjunction with, an Instruction Test controller (EXCHITC) which provides various interface and test facilities including the control of slave PPs which are used to initiate CYBER 170 type exchanges.

The CYBER 170 PP exchange commands EXN, MXN, and MAN are tested while the processor is operating in the executive state and CYBER 170 job and monitor states. The test checks the operation of the exchange and trap interrupts which result from these PP exchanges and also checks that the processor can subsequently be moved into the CYBER 170 mode to provide the proper exchange accept response to the PP system. A number of sections of the test employ multiple PPs performing EXN, MXN, or MAN commands. These sections check that all exchanges are responded to and that an interrupted process can be resumed and properly completed. In some sections, other activities such as starting and stopping the processor (test section 9) or System Interval Timer (SIT) and Process Interval Timer (PIT) interrupts (test section 11) or external interrupts (test section 10) are caused to occur while the PP exchange activity is also present. Sections 12 through 15 check CYBER 170 error exit conditions.

Appendix C contains a set of diagrams that illustrate the sequence of exchanges that occur in the test sections.

ENGINEERING CHANGE ORDERS

In some cases provision can be made in EXCH to alter the tests performed on the basis of ECOs that may or may not have been incorporated on the processor being tested. Where this is possible, the setting of a specific bit in the ECO FLAGS word (see paragraph 3.3.3) will cause the test to assume that the related ECO has been incorporated. If the bit is reset or if an old version of the test is employed the test operates in the normal way. The bits will be assigned from least significant (bit 63) to most significant (bit 48). Flag bits are assigned as follows:

Bit 63 The change related to this bit affects Section 15, Subsections 1,2,3,4,5, and 7. Comparison number 2 in several conditions of these subsections will fail if the flag bit is not set/reset in concert with the hardware change. The expected status of the CYBER 170 infinite and indefinite error exits bit are altered as a result of the change.

2 REQUIREMENTS

2.1 HARDWARE

Equipment for which test is intended:

Models 810, 815, 825, 830, 835, 845, and 855 Computer Systems.

In addition to requirements for CMSE, EXCH requires the following:

1	CPU	
5	PPs	(4 of the channels must have the
5	PP channels	same number as 4 of the PPs)
	A minimum of one megabyte of CM	

In sections 2 and 16, up to 16 megabytes of memory may be selected in order to check memory addressing by the PP exchange command.

With 5-PP systems the test bypasses some sections which require multiple slave PPs. EXCH may run all sections with 10 or more PPs.

2.2 SOFTWARE

This product uses the facilities of the Common Maintenance Software Executive (CMSE); additionally, portions of the Diagnostic Executive (DEX) are compiled with, and are a part of the EXCHITC object code.

2.3 ACCESSORIES

None required.

2.4 CHARACTERISTICS

1.	Test name	EXCH
2.	Test/diagnostic/utility/system	Test
3.	Size (source) PP code	6000 lines
	CP code	7000 lines
4.	Size (memory required for execution)	
	PP code	4096 PP words
	Central memory	1 million bytes minimum
5.	Virtual code/microcode/PP code/other	Virtual and PP code
6.	Run time	1 minute
7.	Level of isolation	detection
8.	On-line/off-line/both	off-line
9.	Off-line system	MSL 15X
10.	Resident during execution	Yes
11.	Language	CP:CYBER VM Assembler PP: COMPASS
12.	Source code maintenance	CP: MODIFY (ASCII) PP:UPDATE
13.	Uses maintenance channel	Yes

3 OPERATIONAL PROCEDURE

3.1 RESTRICTIONS AND USER CAUTIONS

With 5 PP systems the test bypasses some sections which require multiple slave PPs. EXCH may run all sections with 10 or more PPs.

3.2 LOADING PROCEDURE

The object code of EXCHITC and its associated overlays plus the EXCH object code must reside on the Maintenance Software Library (MSL) device. The file names on the MSL for these binaries are EXCHITC, EXCHP1, EXCHP2, EXCHP3, EXCHP4, EXCHSLV, and EXCH.

Command buffers do not load the microcode. The standard product set microcode must be loaded prior to executing the command buffer.

Command buffer EXCHx (where x is 1 for model 810/815/825/830, 2 for 835, 3 for 845/855) exists on the MSL 15X tape to facilitate loading and execution of this test. Before using this command buffer you may have to modify it for your system. Display the command buffer using the CMSE command buffer display commands or print the contents of the command buffer using procedures provided in the Command Buffer Maintenance section of the MSL 15X Reference Manual. Then modify the command buffer as directed by comments embedded in the command buffer. When you are satisfied that the command buffer is set up properly, enter a GO,EXCHx command to execute it, where x is the model dependent number from above.

Modified command buffers can be saved on a back-up tape for future use. Refer to the Utilities section of the MSL 15X Reference Manual for procedures.

Execution of the command buffer causes default parameters to be used by EXCHITC. If desired, commands to alter the default parameters may be included in the command buffer immediately prior to the RU command. Inclusion of an SP command after the RU command will automatically provide the space bar command, which is required to cause execution to proceed after the default parameter entry stop.

3.2.1 Loading Processor Dependent Overlays

After the command buffer initiates the execution of EXCHITC, it accesses the element ID register of the subsystem on each radial interface to determine the type of IOU, processor, and memory which are a part of the system. Based on this information, the appropriate connect and type codes are set up for later use. Determination of the type of central processor allows EXCHITC to adapt to different machines. An overlay specific to the processor type (EXCHP1, EXCHP2, EXCHP3, EXCHP4) is loaded into the EXCHITC PP to account for most of the differences. In the few cases where differences cannot be conveniently handled by the overlays, code modification is employed. The initializing code modifies constants and command operands in the permanent code of EXCHITC. When all changes are complete, EXCHITC makes the initial parameter stop and display.

3.2.2 Loading of Slave PPs

The loading of slave PPs commences after the initial parameter stop. The sequence for loading is described below. The number of the PP which will be selected as the the first possible slave PP is defined by PARAM 11; default is PP number 0. If EXCHITC is already residing in PP number 4 then the slaves would normally be PPs 5,6,7, and 11_8. This assumes CMSE is using channels or PPs 0 through 3.

NOTE

After execution of EXCH is complete, the operator should execute a D (drop) command to make all PPs available to CMSE. If the EXCH command buffer is executed a second time (without the drop command and assuming auto PP assignment is in effect), a much different and unexpected set of PPs will be caused to execute the test.

Parameters 11, 12, 13, and 14 select the PPs to be used as slaves. The search for available PPs proceeds sequentially from PP 0. If PARAM 11 is nonzero, the search commences from that nonzero value. Only PPs for which the corresponding channel number is available will be selected. PPs and channels used by CMSE will be skipped automatically.

If desired, a specific set of four slaves may be selected by entering the PP numbers in parameter words 11 through 14. A nonzero value in PARAM 12 is recognized by EXCHITC as a signal to use the four parameter values as the required PPs.

NOTE

In this mode, EXCHITC does not check to determine if the PPs are actually available. It provides no warning if they are not.

Parameter word 15 is used to control the selection of the channel used by EXCHITC to communicate with the four slaves. Parameter default is 0 and, as such, either channel 12 or 13 will be selected, depending upon which of these is not in use by CMSE to communicate with the MSL device. If neither of these is available, the number of some other suitable channel must be entered in the parameter word. The sequence employed by EXCHITC for setting up the channels and loading the slave PPs is as follows:

a) Select and request use of the channel which is to be used for EXCHITC to slave PP communication.

b) If PARAM12 is zero, skip to step c); otherwise, assign PPs as slaves according to parameters 11 through 14 and then skip to step f).

c) Starting with the PP defined (by PARAM11) to be the first slave PP, scan to determine if that PP is: PP 0, the CMSE I/O driver PP; the EXCHITC PP; the CMSE monitor PP; the CMSE display PP; the same number as the CMSE MSL device channel; the same number as the CMSE PP communication channel; or if the PP number is the same as the display channel number 10_8. If none of these is true then assign the PP to be the first (or next) slave PP.

d) If four slave PPs have not been assigned and if all PPs in the system have not been scanned, skip to the next higher PP and return to step c).

e) If no PPs were assigned, display the message TOO FEW PP'S and exit to the CMSE idler. The operator should check the entries for parameters 11 through 14.

f) If only one, two, or three slave PPs are assigned, then modify the section select parameter words so that only sections which employ one slave PP will be executed.

g) In sequence, load each of the assigned slave PPs with its object code (EXCHSLV), initiate its program execution and check that it responds by clearing a central memory flag.

h) Check the response of all slave PPs by commanding them to execute dummy operations which will require their simultaneous access of central memory and their assigned channel. If any slave PP hangs up, display the message ABORTED - CHAN OR CM ERR and exit to the CMSE idler.

3.2.3 Load Map

The above sections describe the loading of the PP portions of the test. The CP portion of EXCH is loaded via command buffer into central memory as a single record at byte address 4000_{16}. The following table provides information as to the byte address of the various blocks of data and code. The segment and page tables are generated by the test after it commences execution.

Hex Address	
0000	Register file dump and deadstart flag
1000	Segment table
2000	Page table
4000	Section address table
4100	Control flags and data
4400	CYBER 170 Process data and code
4A00	CYBER 170 Exchange packages
4F00	Executive state Instruction level code
5600	Executive state Exchange packages and stacks
6200	Control command tables

Refer to section 3.3.3 for the location of various control flags and data.

3.3 PARAMETERS AND CONTROL WORDS

3.3.1 Parameters

Parameter words control the execution of the test. These words are defined in the MSL15X Off-Line Maintenance Software Library Reference Manual listed in the preface. They are located at PP locations 122_8 to 144_8 (directly following the control words).

Parameter words can be set/cleared manually through CMSE commands. Individual bits of parameter word 0 can be set/cleared with a set of CMSE commands which references the bits by mnemonics. For example, CSM clears scope mode while SRC sets repeat condition.

The use of PARAM7 through PARAM18 are peculiar to EXCHITC and EXCH. For further details, refer to section II-6, this manual.

Parameters are as follows:

Octal Address	Parameter	Bit	Meaning
122	PARAM0	$0001_8/0001_{16}$	Stop at end of test (ST)
		$0002_8/0002_{16}$	Stop at end of section (SS)
		$0004_8/0004_{16}$	Stop at end of subsection (SB)
		$0010_8/0008_{16}$	Stop at end of condition (SC)
		$0020_8/0010_{16}$	Stop on error (SE)
		$0040_8/0020_{16}$	Log effort in dayfile (LE)
		$0100_8/0040_{16}$	Repeat test (RT)
		$0200_8/0080_{16}$	Repeat section (RS)
		$0400_8/0100_{16}$	Repeat subsection (RB)
		$1000_8/0200_{16}$	Repeat condition (RC)
		$2000_8/0400_{16}$	Scope mode (SM)
		$4000_8/0800_{16}$	Quick look (QL), not used
		$10000_8/1000_{16}$	Bypass all messages (DR)
		$20000_8/2000_{16}$	Display only error messages (DE)
		$40000_8/4000_{16}$	Reserved
		$100000_8/8000_{16}$	Accept CMSE Param commands
123	PARAM1	$0001_8/0001_{16}$	Reserved
		$0002_8/0002_{16}$	Reserved
		$0004_8/0004_{16}$	Bypass parameter stop
		$0020_8/0010_{16}$	Reserved
		$0040_8/0020_{16}$	Reserved
		$0100_8/0040_{16}$	Reserved
124	PARAM2		Repeat test count
125	PARAM3	$0001_8/0001_{16}$	Test CPU 0
		$0002_8/0002_{16}$	Test CPU 1
126	PARAM4		Reserved
127	PARAM5		Sections 15-00 SELECT
130	PARAM6		Sections 31-16 SELECT
131	PARAM7		Exchange address, byte, 16 MSBs
132	PARAM8		Exchange address, byte, 16 LSBs
133	PARAM9		Exchange address increment, byte
134	PARAM10		Number of exchanges
135	PARAM11		First slave PP

136	PARAM12		Second slave PP
137	PARAM13		Third slave PP
140	PARAM14		Fourth slave PP
141	PARAM15		EXCHITC to slave communication channel
142	PARAM16		Delay count for processor hung
143	PARAM17	$0002_8/0002_{16}$	Enable PFS register operation
144	PARAM18		Memory segment select, one Bit/megabyte

Parameter defaults are as follows:

Octal Address	Setting	Meaning
122	PARAM0 = 100121_8	The most significant bit must be set.
123	PARAM1 = 000000_8	The parameter stop is not bypassed.
124	PARAM2 = 000000_8	The repeat test count will have effect only if the repeat test bit is clear.
125	PARAM3 = 000003_8	Test both CPU0 and CPU1 if available.
127	PARAM5 = 177777_8	All sections selected.
130	PARAM6 = 177777_8	
131	PARAM7 = 000001_8	First exchange at 10000_{16}.
132	PARAM8 = 000000_8	
133	PARAM9 = 000200_8	Address increment of 16 CM words.
134	PARAM10= 017000_8	Number of exchanges go up to but not over 100000_{16}.
135	PARAM11= 000000_8	Select slaves sequentially starting with first available PP.
136	PARAM12= 000000_8	
137	PARAM13= 000000_8	
140	PARAM14= 000000_8	
141	PARAM15= 000000_8	Use either channel 12_8 or 13_8 for controller to slave communication.
142	PARAM16= 000777_8	Hang count of 511.
143	PARAM17= 000002_8	Enable PFS register operation.
144	PARAM18= 000001_8	Test only first megabyte of CM.

3.3.2 Control Words

Control words are intended to identify a program and supply information to a higher system or operator. They do not normally affect test execution. The control words are located at PP locations 102_8 to 121_8 and are as follows:

Octal Address	Word	Meaning
102	CW0	Program name (first 2 characters)
103	CW1	Program name (last 2 characters)
104	CW2	Program type
105	CW3	Monitor ID word
106	CW4	Error code number 1
107	CW5	Error code number 2 (not used)

110	CW6	Pass counter
111	CW7	Current section counter
112	CW8	Current subsection counter
113	CW9	Current condition counter
114	CW10	Current error counter

3.3.3 Central Memory Control Flags and Data

The central memory is used as a means of communication between EXCHITC, the central processor, and the slave PPs. Certain flags and data may be of interest to the operator when a failure occurs. These are itemized below by their central memory byte address:

0000_{16} DEADSTART FLAG - A hand-shake flag used in all sections by EXCHITC and the CP (see section II-6). Note that when a register file dump occurs, this flag will be overwritten.

4108_{16} SLAVE CHANNELS - Each of the four parcels shows the channel number (and PP number) used by EXCHITC to communicate with the four slave PPs.

4110_{16} EXCHANGE COMMANDS - Each of the four parcels contains a special command to be executed by each slave PP. Most often a 26XX command.

4118_{16} SLAVE DELAYS AND EXCHANGE COUNTS - A word of data set up by EXCHITC and used by the slaves to control the number of exchanges to be performed and the delay to be executed prior to each exchange. The word is divided into four parcels, 0 through 3 which are referenced by slaves 0 through 3, respectively. The upper eight bits of a parcel define the delay and the lower eight bits the exchange count. Used in sections 1 through 11.

4120_{16} SLAVE FUNCTION CONTROL - This word contains data passed by EXCHITC to the slave PPs to provide functional control.

4128_{16} SLAVE PP0 EXCHANGE ADDRESS - The current exchange address set up by EXCHITC and used by slave PP0.

4130_{16} SLAVE PP1 EXCHANGE ADDRESS - Same as for slave PP0.

4138_{16} SLAVE PP2 EXCHANGE ADDRESS - Same as for slave PP0.

4140_{16} SLAVE PP3 EXCHANGE ADDRESS - Same as for slave PP0.

4148_{16} EXCHANGE SEGMENT ADDRESS - The upper four bits of an address which will determine which one megabyte segment of memory will be checked in sections 2 and 16.

4150_{16} SLAVE PP0 RESPONSE DATA - A word written into central memory by the slave PP upon completion of a test condition. The four parcels serve the following purpose:

- Parcel 0 - Slave exchange count; indicates the number of exchanges for which an exchange accept was not received.

Should be decremented to zero upon completion of a condition; see section II-6. Checked in test sections 7 through 11.

- Parcel 1 - B0 miss count, indicates the number of times an exchange request did not result in a CYBER 170 exchange package swap; see section II-6. Checked only in section 7.

- Parcel 2 - The PP op code of the command executed by the slave. Usually a 26XX. May be a 2400 for an inactive slave. May also be a 63XX (CWM), 1063XX (CWML), or a 102601 (INPN) for sections 8 or 10.

- Parcel 3 - Not applicable.

4158_{16} SLAVE PP1 RESPONSE DATA - Same as for 4150_{16}

4160_{16} SLAVE PP2 RESPONSE DATA - Same as for 4150_{16}

4168_{16} SLAVE PP3 RESPONSE DATA - Same as for 4150_{16}

4170_{16} SYSTEM INTERVAL TIMER (SIT) INTERRUPT COUNT - Written to central memory and decremented to zero by the CP during a condition of section 11. Nonzero value indicates that all SIT interrupts were not performed. Comparison checked by EXCHITC.

4178_{16} SIT INTERVAL VALUE AND COUNT - Written to central memory by EXCHITC for use by the CP in section 11 only. Parcels 0 and 1 are the interval and parcel 3 is the interrupt count.

4180_{16} PROCESS INTERVAL TIMER (PIT) INTERRUPT COUNT - Written to central memory and decremented to zero by the CP during a condition of section 11. Nonzero value indicates that not all PIT interrupts were performed. Comparison checked by EXCHITC.

4188_{16} PIT INTERVAL VALUE AND COUNT - Written to central memory by EXCHITC for use by the CP in section 11 only. Parcels 0 and 1 are the interval and parcel 3 is the interrupt count.

4220_{16} ECO FLAGS - The least significant 16 bits of this word may be used to select or control test options related to the incorporation of ECOs. See paragraph 1 for additional details.

4230_{16} 170 MODE INTERRUPT COUNT - Count of the number of external interrupts received in a single condition of section 10 while the CP was executing in the 170 state job mode.

4238_{16} VIRTUAL MODE INTERRUPT COUNT - Same as 4238_{16} above except for the virtual monitor mode.

4400_{16} REFERENCE ADDRESS - In the CYBER 170 mode the error exit location and condition bits are stored in RA by the processor.

4408_{16} SAVED REFERENCE ADDRESS - The test's CYBER 170 monitor mode process saves the contents of RA (from a previous CYBER 170 job mode error exit) at this location.

4410_{16} EXCHANGE TO A FLAG - In some test sections a CYBER 170 exchange to A results in execution of code which causes a value of $AAAA_{16}$ to be stored in this location. The flag is checked to see that the particular condition sequenced properly.

4418_{16} EXCHANGE TO MA FLAG - A CYBER 170 exchange to monitor address causes a value of $DEAD_{16}$ to be stored in this location (like exchange to A flag).

4420_{16} SEQUENCE COMPLETE FLAG - A hand-shake control flag used only in sections 9 and 11 by EXCHITC and the CP. See section II-6.

3.4 SECTION INDEX

Section	Tag	Brief Description
0	SECT0	Test initialization
1	SECT1	CEJ/MEJ commands
2	SECT2	Exchange addressing
3	SECT3	26xx execution in CYBER 170 job and monitor mode
4	SECT4	26xx sets MCR5 in executive state job mode
5	SECT5	26xx execution in executive state job mode
6	SECT6	26xx execution in executive state monitor mode
7	SECT7	Multiple 2600 exchanges and B0 checking
8	SECT8	26xx exchanges and block writes in CYBER 170 mode
9	SECT9	2600 exchanges and CP stop/start in CYBER 170 mode
10	SECT10	2600 exchanges and external interrupts
11	SECT11	2600 exchanges and SIT/PIT interrupts
12	SECT12	Error exit, illegal instructions
13	SECT13	Error exit, address out of range
14	SECT14	Error exit, floating point infinite
15	SECT15	Error exit, floating point, indefinite
16	SECT16	Exchange addressing, selectable addresses
17	SECT17	OS Bounds Register test
18	SECT18	State switch with XO sign bit

4 OPERATOR COMMUNICATION

4.1 DISPLAYS

4.1.1 Initial Display

The initial display presented by EXCH is as shown below. The display is made and the test stops to allow the operator to enter any parameter changes that may be required. The address of the first parameter word is given in the first line of the display. The revision level of EXCHITC is also specified. The entry, EP,4,142,xxxx is a parameter entry which can be made into the EXCHITC PP memory. If the command buffer is altered such that EXCHITC is not loaded into PP 4 then this entry must be altered accordingly.

```
EXCH   PARAMS PA=122B  80/11/30 REV 2.4

         KEYBOARD COMMANDS

ABS/ABB              ABORT SECT/SUBSEC
ECPX/DCPX            EN/DISABL CPU X=0+1
EPFS/DPFS            EN/DISABL PFS UTRAPS - P1,P2,P3 ONLY
EP,4,142,XXXX        SET CP HANG COUNT
S/R/D/SPACE          STOP/RESTRT/DROP/CONTINU

CONTROL DATA PROPRIETARY PRODUCT
COPYR. CONTROL DATA 1980
```

4.1.2 Running Display

The normal running display for EXCH is shown below.

```
EXCH (op) PCxxxx  CPx  Sxxxx  SBxxxx  Cxxxx        yy/mm/dd
```

OPERATION (op):

RU	Running Message
RC	Repeating Condition
RB	Repeating Subsection
RS	Repeating Section
SC	End of Condition
SB	End of Subsection
SS	End of Section

	ST End of Test
	SE Stopped on Error
	SM Executing in scope mode
	HT Test halted by operator
PCxxxx	Pass Count is xxxx
CPx	Current CPU
Sxxxx	Current Section Number is xxxx
SBxxxx	Current Subsection Number is xxxx
Cxxxx	Current Condition Number is xxxx
yy/mm/dd	Year, month, and day of revision of the EXCH test

Normally the RB and RS codes would not be seen when a subsection or section is being repeated. This is due to a running message (RU) being issued at every subsection boundary which erases the previous message and provides a dynamic display of the current subsection and section.

Also note that there is no operation code for an operator stop. If an S (stop) command (see paragraph 4.2) is entered while the test is running, the test will stop and the HT code will be displayed.

PARAM3 controls testing for dual CP systems. Either one or both of the CPs may be tested. When two CPs are tested, execution occurs alternately, not concurrently. CP0 is tested first for any given section and, when testing is complete, the section is repeated for CP1.

NOTE

If the repeat section paramater is set, the section is repeated for the currently executing CP. If the section stop bit is set the test stops at the end of section for each processor.

4.1.3 Error Message Display

When EXCH detects an error, the following standard error message is displayed. Depending on the type of error, additional error messages are displayed beneath this display. See Error Messages, section 4.4, for definitions of these messages.

```
EXCH   SE    PCxxxx CPx Sxxxx     SBxxxx     Cxxxx
EC1 = xxxx   TE = xxxx
```

Where:

SE	Stopped on error
PCxxxx	Pass count
CPx	Failing CPU
Sxxxx	Failing section number
SBxxxx	Failing subsection number
Cxxxx	Failing condition number
EC1	Error Code 1
TE	Total error count

4.2 OPERATOR ENTRIES

In addition to the operator entered commands which are provided by CMSE, EXCHITC provides additional commands as follows:

Action	Command	Description
SPACE BAR	Continue execution	Entered after any stop to cause execution of the test to continue.
S	Stop	Entered while the test is executing to cause a halt at the next condition boundary.
R	Restart	Restarts the test from the first selected section.
D	Drop	Halts test execution and requests CMSE to deadstart all slave PPs and the EXCHITC PP. The test cannot subsequently be restarted.
ABS	Abort section	Entered by the operator after an error stop when it is desired to skip to the next section of the test.
ABR	Abort subsection	Entered by the operator after an error stop when it is desired to skip to the next subsection of the test. This command is ignored if the test is already at an end of subsection stop.
ECP0/DCP0 ECP1/DCP1		Enables or disables test execution for CP0 or CP1 on dual CP systems. These commands are ineffective on single CP systems.
EPFS/DPFS	Enable/disable PFS microtraps	These commands provide a more convenient method of enabling or disabling the PFS micro- traps as opposed to setting or clearing a bit in PARAM word 17.

4.3 NORMAL MESSAGES

Other than the running display and initial display, described previously, EXCH does not display any test messages for normal operation.

4.4 ERROR MESSAGES

Depending on the type of error detected, EXCH displays one of the following error messages.

4.4.1 Data Comparison Error Message

An example of a standard data comparison error message is shown below.

```
EXCH  SE  PC0000 CP0 S0008  SB0002 C0021      yy/mm/dd
EC1=0000 TE=0001

ACTIVE SLAVES   PP 05 06 07
P REG           00 00 B0 00 00 00 40 34
S REG           00 00 04 31 04 31 04 31
MON PACK        004100  PROCESS  0000 B000 0000 4030
JOB PACK        004280  PROCESS  0000 B000 0000 4068
COMPARE NO      0003
MASK            FFFF FFFF FFFF FFFF
XPCTD           0000 0000 0000 0000
RECVD           0000 0000 0000 1000
ADRS OF XPCTD 0044E8
ADRS OF RECVD 000088
ADRS INDEX      0000
```

The first line is similar to that for a running display; the second line provides an error code and the total errors as follows:

EC1	Error Code 1
TE	Total Errors since test first started

The EC1 error code is set with a value of $1C20_{16}$ when the MAC channel is found unable to transfer data after a function was sent by EXCHITC.

The compare number indicates which comparison within the the current condition caused the error. Numbering of comparisons commences with number 1. Given this number and the condition and section, a description of the failure may be found in section II-6 of this manual.

If the address index is nonzero, then a block comparison occurred and the index is a word offset from the expected and received addresses pointing to the data in error within the block. In the event of multiple errors in a block comparison, only the data for the first detected error is displayed.

The addresses of the executive state monitor and job exchange packages are provided along with the value of the P register word contained in the packages. It is not possible to provide equivalent information pertaining to the CYBER 170 exchange packages.

Except for the S register and the address index, all displayed addresses are byte addresses.

4.4.2 Processor Hung Error Message

In the event that an operation of the central processor is initiated and does not subsequently halt, a display similar to the following appears.

```
EXCH  SE  PC0000 PC0 S0008  SB0000 C0001      yy/mm/dd
EC1=0000 TE=0001

ACTIVE SLAVES   PP 05 06
P REG           00 00 B0 00 00 00 40 34
S REG           00 00 04 31 04 31 04 31
MON PACK        004100  PROCESS  0000 B000 0000 4030
JOB PACK        004280  PROCESS  0000 B000 0000 4068

PROCS HUNG      CP  PP 06
```

In this example, the last line indicates that the central processor and slave PP 6 are hung up. Typically, if the CP hangs in the sequence of commands which it is executing, then any PP which has sent an exchange request may also hang up because it will fail to receive an exchange accept from the CP.

NOTE

When a processor hangs up, the S and P registers are read immediately for the purpose of displaying their values. EXCHITC then sends a special deadstart command to the processor to force exchange accepts to be sent to the PP system to release any slave PPs that may have hung and to allow EXCHITC to regain control of them. If, on such an occasion, the operator observes the value of S or P via the CMSE AR display, one or both may not correspond.

4.4.3 Program Error Message

A display similar to that shown above, except that the last line reads PROGRAM ERR indicates that EXCHITC detected an error in the process of interpreting the control commands which it reads from central memory. The error could be caused by bad data loaded from disc to central memory or by an overwrite of the control tables in central memory as a result of a central processor error.

4.4.4 System Register Error Display

If an error is detected while checking either a processor hardware register or a processor soft register (dedicated register file register accessed by microcode) then a display similar to that shown below will occur.

```
EXCH  SE  PC0000 PC0 S0000  SB0000 C0003      yy/mm/dd
EC1=0000 TE=0001

ACTIVE SLAVES   PP 00
P REG           00 00 B0 00 00 00 40 30
S REG           00 00 04 31 04 31 04 31
MON PACK        004100  PROCESS  0000 B000 0000 4030
JOB PACK        000000  PROCESS  0000 0000 0000 0000
COMPARE NO      0005
MASK            FFFF FFFF FFFF FFFF
XPCTD           0000 0000 0000 0000
RECVD           0000 0000 0000 1000
REG ADRS        0048
```

The register address is the address of the register within the central processor or central memory subsystem.

4.4.5 Miscellaneous Error Messages

The following error messages may also occur. In all cases, EXCHITC exits to the CMSE idler since recovery is not possible.

- ABORT - MAC ERR - An error occurred during initialization when ITC was reading the EID registers on each radial interface.
- ABORT - ELEMENT ID ERR - During initialization while reading EID registers, too few responses were received. Apparently a system element was not connected.
- ABORT - TOO FEW PP S - During the loading of the slave PPs , EXCHITC was unable to find any slave PPs available for use. The entry for PARAM 11 should be checked. See section 3.2.2.
- ABORT - CHAN OR CM ERR - After the slave PPs are loaded, EXCHITC checks to determine if it can communicate with them. The error may indicate a channel problem or a PP to CM interface problem caused by four slaves accessing CM simultaneously. See section 3.2.2.
- ABORT - SLAVE PP HUNG - During normal execution of the test, an exchange accept problem may arise causing a slave PP to hang up on a 26XX command. When this occurs, EXCHITC initiates a sequence which should force the CP to send exchange accepts to the IOU system. If this operation fails to release the slave or slaves from the 26XX command this message should appear.

- CPX NOT A170 CAPABLE - Prior to executing the first condition of the test, the C170 bit of the VMCL register and the SOLO C170 bit of the options installed register are checked to ensure that they are set. This error is more likely to occur in dual CP systems where the second CP may not have A170 capability. If this is so, deselect the parameter bit for the second CP.

4.5 APPLICATIONS

4.5.1 SCOPE MODE CONTROL AND TRIGGERING

The CMSE command, SSM is used to set the scope mode. The command is entered when the test is at a stop. When execution is restarted with a space bar, EXCHITC forces the repeat of the current condition and the entry to the scope loop. If a number of errors have occurred during the condition then more than one space bar entry may be required to reach the end of the condition and enter the loop. Entry to the loop may be confirmed by observing activity of the MAC channel flag on the display. The SM code will also appear on the running display line.

The test may be pulled out of the scope loop with the CMSE command UP,x where x is the PP in which EXCHITC resides.

In all sections of the test, a scope may be triggered using the pulse generated by the deadstart function which performs the half-exchange-in of the monitor process.

In most sections of the test, a scope may be triggered using the deadstart which performs the half-exchange-in of the monitor process. In all sections of the test, the scope may also be triggered from the test point for the model independent refresh resync command (executive state mode op code 01_{16}). This command occurs in the CP code sequence for each section, immediately prior to the point where any failures are likely to occur (usually just before the executive state EXCHANGE which switches the processor to executive state job mode). In section 9, the refresh resync command is the only trigger that should be used (see section II-6 for further details).

The processor hang count (PARAM16) is defaulted to a value of 500_8. This value has been used to provide for the case where the processor is running without cache and map enabled. In cases where the cache and map are in use and where a scope loop is required, this value can be changed to 140_8 in order to obtain a loop with a higher repetition rate. If the scope is observed to have a long period where no activity is occurring then values even lower than 140_8 could be employed.

SECTION I-7

TRAP TEST

==========

1 INTRODUCTION

This trap interrupt test (TRAP) includes and operates in conjunction with an Instruction Test controller (FCTITC) which provides various interface and test facilities.

The test executes MCR and UCR bit tests in which individual bits of these registers are set:

1) during the load of an exchange pack from memory,

2) via the MAC or,

3) by a branch on condition register command.

The purpose is to ensure that the appropriate response (trap, exchange, halt, stack) occurs for each bit prior to commencing with individual function type tests. These tests are conducted with traps enabled and disabled and with the processor operating in job mode and monitor mode.

Subsequent sections of the test execute command sequences to generate the functions which would activate the MCR and UCR bits in normal operation. One test section is devoted to a function test of this type for each bit of the MCR and UCR register. These tests are also performed with traps enabled and disabled in the monitor and job modes. The TRAP test does not execute a test case for every particular case that can cause an interrupt. A reasonable subset is checked.

2 REQUIREMENTS

2.1 HARDWARE

Equipment for which test is intended: Models 810, 815, 825, 830, 835, 845, and 855 Computer Systems.

In addition to requirements for CMSE, TRAP requires the following to execute:

1	CPU
1	PP and the MAC channel (17_8)
$21,000_{16}$	bytes of central memory (TRAP occupies less than the first $14,000_{16}$ bytes. The first $21,000_{16}$ bytes are used during execution).

2.2 SOFTWARE

This product uses the facilities of the Common Maintenance Software Executive (CMSE). Additionally, portions of the Diagnostic Executive (DEX) are compiled with, and are a part of the FCTITC object code.

2.3 ACCESSORIES

None required.

2.4 CHARACTERISTICS

1.	Test name	TRAP
2.	Test/diagnostic/utility/system	Test
3.	Size (source) PP code	6000 lines
	CP code	11000 lines
4.	Size (memory required for execution)	
	PP code	4096 PP words
	Central memory	1 million bytes minimum
5.	Virtual code/microcode/PP code/other	Virtual and PP code
6.	Run time	1 minute
7.	Level of isolation	detection
8.	On-line/off-line/both	off-line
9.	Off-line system	MSL 15X
10.	Resident during execution	Yes
11.	Language	CP: CYBER VM Assembler PP: COMPASS
12.	Source code maintenance	CP: MODIFY (ASCII) PP: UPDATE
13.	Uses maintenance channel	Yes

3 OPERATIONAL PROCEDURE

3.1 RESTRICTIONS AND USER CAUTIONS

None.

3.2 LOADING PROCEDURE

The object code of FCTITC and its associated overlays plus the TRAP object code must reside on the Maintenance Software Library (MSL) device. The file names on the MSL for these binaries are FCTITC, FCTOP1, FCTOP2, FCTOP3, FCTOP4, FCTOTI, and TRAP.

Command buffers do not load the microcode. The standard product set microcode must be loaded prior to executing this command buffer.

Command buffer TRAPx (where x is 1 for model 810/815/825/830, 2 for 835, 3 for 845/855) exists on the MSL 15X tape to facilitate loading and execution of this test. Before using this command buffer you may have to modify it for your system. Display the command buffer using the CMSE command buffer display commands or print the contents of the command buffer using procedures provided in the Command Buffer Maintenance section of the MSL 15X Reference Manual. Then modify the command buffer as directed by comments embedded in the command buffer. When you are satisfied that the command buffer is set up properly, enter a GO,TRAPx command to execute it, where x is the model dependent number from above.

Modified command buffers can be saved on a back-up tape for future use. Refer to the Utilities section of the MSL 15X Reference Manual for procedures.

Default parameters are automatically set when this command buffer is executed. Default parameters may be reset by entering commands immediately prior to the RU command. Inclusion of a SP command after the RU command will automatically provide the space bar which is required to cause execution to proceed after the parameter entry stop.

3.3 PARAMETERS AND CONTROL WORDS

3.3.1 Parameter Words

Parameter words control the execution of the test. They are defined in the MSL15X Off-Line Maintenance Software Library Reference Manual. These words are located at PP locations 122_8 to 144_8 directly following the control words.

Parameter words are set/cleared manually through CMSE commands. Individual bits of parameter word 0 can be set/cleared with a set of CMSE commands which reference the bits by mnemonics. For example, CSM clears scope mode while SRC sets repeat condition.

Further details concerning the specified parameters are provided as follows:

PARAM 16 - Processor hang count - section II-7
PARAM 17 - Enable/Disable PFS microtraps - paragraph 4.2 (this section)
PARAM 18 - CM port select - section II-7

The use of PARAM18 is peculiar to TRAP.

Parameters are as follows:

Parameter	Octal Address	Octal/Hex Bit	Meaning
PARAM0	122	0001/0001	Stop end test (ST)
	122	0002/0002	Stop end section (SS)
	122	0004/0004	Stop end subsection (SB)
	122	0010/0008	Stop end condition (SC)
	122	0020/0010	Stop on error (SE)
	122	0040/0020	Log errors (LE)
	122	0100/0040	Repeat test (RT)
	122	0200/0080	Repeat section (RS)
	122	0400/0100	Repeat subsection (RB)
	122	1000/0200	Repeat condition (RC)
	122	2000/0400	Scope mode (SM)
	122	4000/0800	Quick look (QL),Not used
	122	10000/1000	Bypass all msgs (DR)
	122	20000/2000	Display only err msgs (DE)
	122	40000/4000	reserved
	122	100000/8000	Accept CMSE param cmds
PARAM1	123	0001/0001	reserved
	123	0002/0002	reserved
	123	0004/0004	Bypass param stop
	123	0020/0010	reserved
	123	0040/0020	reserved
	123	0100/0040	reserved
PARAM2	124		Repeat test count
PARAM3	125	0001/0001	Test CPU 0
		0002/0002	Test CPU 1
PARAM4	126		reserved
PARAM5	127		Sections 15-00 Select
PARAM6	130		Sections 31-16 Select
PARAM7	131		Sections 47-32 Select
PARAM8	132		reserved
PARAM9	133		reserved
PARAM10	134		reserved
PARAM11	135		reserved
PARAM12	136		reserved
PARAM13	137		reserved
PARAM14	140		reserved
PARAM15	141		reserved
PARAM16	142		Delay count for processor hung
PARAM17	143	0002/0002	Enable PFS operation
PARAM18	144		CM port select (by bit position)

Parameter defaults are as follows:

Octal Address	Setting (Octal)	Meaning
122	PARAM0 = 100121	Test control (most significant bit must be set)
123	PARAM1 = 000000	Parameter stop is not bypassed
124	PARAM2 = 000010	Repeat test count will have affect only if repeat test bit is clear
125	PARAM3 = 000003	Test both CPU0 and CPU1 if available
127	PARAM5 = 177757	All sections (except 4) selected
130	PARAM6 = 177777	
131	PARAM7 = 177777	
132	PARAM8 = 177777	reserved
133	PARAM9 = 177777	reserved
135	PARAM11= 177777	reserved
136	PARAM12= 177777	reserved
137	PARAM13= 177777	reserved
140	PARAM14= 000000	reserved
141	PARAM15= 000000	reserved
142	PARAM16= 000020	Hang count of 16
143	PARAM17= 000002	Enable PFS register operation
144	PARAM18= 000001	Use CM port 0

3.3.2 Control Words

Control words are intended to identify a program and supply information to a higher system or operator. They do not normally affect test execution. The control words are located at PP locations 102_8 to 121_8 and are as follows:

CW0	Program name (first 2 characters)
CW1	Program name (last 2 characters)
CW2	Program type
CW3	Monitor ID word
CW4	Error code #1
CW5	Error code #2 (not used)
CW6	Pass counter
CW7	Current section counter
CW8	Current subsection counter
CW9	Current condition counter
CW10	Current error counter

3.3.3 Central Memory Flags, Pointers, and Exchange Packages

Most flags, pointers, and exchange packages of interest to an operator in event of a failure are itemized below by their central memory byte address:

Item	CM Byte Hex Address	Description
DEADSTART FLAG	0000	A hand shake flag used in all sections by FCTITC and the CP. See paragraph 4.2. Note that when a register file dump occurs, this flag will be overwritten.
REFERENCE ADDRESS	4700	In the CYBER 170 mode the error exit location and condition bits are stored in RA by the processor.
C7XPXMA	4880	A CYBER 170 exchange package used only in the event of an error which would cause an unexpected CYBER 170 exchange to MA (monitor address).
CALLSTAK	5800	A 33-word stack referenced whenever a CALLSEG command is executed.
TRAPSTAK	5908	A 33-word stack referenced whenever a trap interrupt occurs.
CBP	5B20	The code base pointer for trap interrupt operations.
CALCBP	5B38	The code base pointer referenced when a CALLSEG command is executed.
JX0	5DE0	An executive state job exchange package used in job mode MCR/UCR bit tests (sect 0,1, and 34).
MX0	5F80	An executive state monitor exchange package used to initiate job mode operation during job mode MCR/UCR bit tests (sect 0,1, and 34).
MX1	6120	An executive state monitor exchange package used in monitor mode MCR/UCR bit tests (sect 0,1, and 34).
JX2	6460	The primary executive state job exchange package. Never used in sections 0,1, and 34; nor in those conditions of the remaining sections where the function test is performed in executive state monitor mode or CYBER 170 mode.
MX2	6600	The primary executive state monitor exchange package used in all sections except 0,1, and 34.
JX3	6940	An executive state job exchange package used in any condition which performs a test in CYBER 170 mode. This package establishes a CYBER 170 environment with the CYBER 170 monitor flag set when the executive state monitor to job exchange is executed.

3.4 SECTION INDEX

Section Number	Tag Name	Brief Description
00		MCR BIT TESTS
01		UCR BIT TESTS
		MCR/UCR FUNCTION TESTS
02		MCR 00 - DETECTED UNCORRECTABLE ERROR
03		MCR 01 - NOT USED
04		MCR 02 - SHORT WARNING
05		MCR 03 - INSTRUCTION SPEC. ERROR
06		MCR 04 - ADDRESS SPEC. ERROR
07		MCR 05 - EXCHANGE REQUEST
08		MCR 06 - ACCESS VIOLATION
09		MCR 07 - ENVIRONMENT SPEC. ERROR
10		MCR 08 - EXTERNAL INTERRUPT
11		MCR 09 - PAGE TABLE SEARCH
12		MCR 10 - SYSTEM CALL
13		MCR 11 - SYSTEM INTERVAL TIMER
14		MCR 12 - INVALID SEGMENT/RING=0
15		MCR 13 - OUTWARD CALL/INWARD RETURN
16		MCR 14 - SOFT ERROR LOG
17		MCR 15 - TRAP EXCEPTION
18		UCR 00 - PRIVILEDGED INSTRUC. FAULT
19		UCR 01 - UNIMPLEMENTED INSTRUCTION
20		UCR 02 - FREE FLAG
21		UCR 03 - PROCESS INTERVAL TIMER
22		UCR 04 - INTER RING POP
23		UCR 05 - CRITICAL FRAME FLAG
24		UCR 06 - KEYPOINT
25		UCR 07 - DIVIDE FAULT
26		UCR 08 - DEBUG
27		UCR 09 - ARITHMETIC OVERFLOW
28		UCR 10 - EXPONENT OVERFLOW
29		UCR 11 - EXPONENT UNDERFLOW
30		UCR 12 - FP LOSS OF SIGNIFICANCE
31		UCR 13 - FP INDEFINITE
32		UCR 14 - ARITHMETIC LOSS OF SIGNIFICANCE
33		UCR 15 - INVALID BDP DATA
34		MULTIPLE INTERRUPT/PRIORITY CHECKS
35		MCR BIT LEFT IN MEMORY
36		RESERVED
37		810/815/825/830 PROCESSOR MODEL DEPENDENT TESTS
38		835 PROCESSOR MODEL DEPENDENT TESTS
39		RESERVED
40		RESERVED

4 OPERATOR COMMUNICATION

Communication with TRAP is via any console that can be driven by CMSE. All communication between the operator and TRAP goes via CMSE and FCTITC. TRAP communicates with FCTITC through tables in central memory. A command buffer initiates the loading into central memory of a single binary file for the complete test. FCTITC fetches a segment of the file from central memory for each condition of the test. The data in the segment is interpreted as a sequence of special commands, and is used by the PP to control the condition. FCTITC performs all comparisons and generates the display data.

4.1 DISPLAYS

4.1.1 Trap Initial Display

The TRAP initial display is shown below. The test stops and issues the display to allow the operator to enter parameter changes. The address of the first parameter word is given in the first line of the display. The date of assembly and the revision level of FCTITC are also specified. The entry EP,4,142,xxxx is a parameter entry which can be made into the FCTITC PP memory. If the command buffer is altered such that FCTITC is not loaded into PP 4 then this entry must be altered accordingly.

In the line beginning EPFS/DPFS, P1,P2,P3 only means model 810/815/825/830/835, and 845/855 processors respectively.

```
TRAP   PARAMS PA=122B  81/10/30 REV 2.4

         KEYBOARD COMMANDS

ABS/ABB              ABORT SECT/SUBSEC
ECPX/DCPX            EN/DISABL CPU X=0+1
EPFS/DPFS            EN/DISABL PFS UTRAPS - P1,P2,P3 ONLY
EP,4,142,XXXX        SET CP HANG COUNT
TIXX YY ZZ           TEST SUBSET OF OP CODES/SECTIONS
S/R/D/SPACE          STOP/RESTRT/DROP/CONTINU

CONTROL DATA PROPRIETARY PRODUCT
COPYR. CONTROL DATA 1980
```

4.1.2 Running Display

The normal running display for TRAP is shown below.

```
TRAP (op) PCxxxx CPx Sxxxx  SBxxxx  Cxxxx  TIxx  yy/mm/dd
```

OPERATION (op)	RU	Running Message
	RC	Repeating Condition
	RB	Repeating Subsection
	RS	Repeating Section
	SC	End of Condition
	SB	End of Subsection
	SS	End of Section
	ST	End of Test
	SE	Stopped on Error
	SM	Executing in scope mode
	HT	Test halted by operator
PCxxxx		Pass Count is xxxx
CPx		Current CPU
Sxxxx		Current Section Number is xxxx
SBxxxx		Current Subsection Number is xxxx
Cxxxx		Current Condition Number is xxxx
TIxx		Current Section Number is xx
yy/mm/dd		Year, Month and Day of revision of the TRAP test

The RB and RS codes are not normally seen when a subsection or section is being repeated. Instead, a running message (RU) is issued at every subsection boundary that erases the previous message and provides a dynamic display of the current subsection and section. There is no operation code for an operator stop. If an S (stop) command (see paragraph 4.2, Operator Entries) is entered while the test is running, the test stops but the RU code remains.

In other tests such as FCT1, the TIxx entry would be F1xx and would indicate the MSL file name or the op code being tested. In the TRAP test all sections of the test reside in CM simultaneously; each section does not pertain to a specific op code.

PARAM3 controls testing for dual CP systems. Either one or both of the CPs may be tested. When two CPs are tested, execution occurs alternately, not concurrently. CP0 is tested first for any given section and, when testing is complete, the section is repeated for CP1.

NOTE

If the repeat section paramater is set, the section is repeated for the currently executing CP. If the section stop bit is set the test stops at the end of section for each processor.

4.1.3 Error Message Display

When EXCH detects an error, the following standard error message is displayed. Depending on the type of error, additional error messages are displayed beneath this display. See Error Messages for definitions of these messages.

```
TRAP    SE    PCxxxx     Sxxxx     SBxxxx     Cxxxx  yy/mm/dd
EC1  = xxxx    TE = xxxx
```

Where:

SE	Stopped on error
PCxxxx	Pass count
Sxxxx	Failing section number
SBxxxx	Failing subsection number
Cxxxx	Failing condition number
EC1	Error code 1
TE	Total error count

4.2 OPERATOR ENTRIES

EXCHITC provides the following commands in addition to those provided by CMSE:

Action	Command	Description
SPACE BAR	Continue Execution	Continues test execution following a stop.
S	Stop	Stops test at the next condition boundary.
R	Restart	Restarts test from first selected section.
D	Drop	Halts test execution and requests CMSE to deadstart FCTITC PP. Test cannot be restarted.
ABS	Abort Section	Skips to next section after an error stop.
ABB	Abort Subsection	Skips to the next subsection of the test after an error stop; ignored if test is at end of subsection stop.
ECP0/DCP0 ECP1/DCP1		Enable or disable test execution for CP0 or CP1 on dual CP systems. These commands are ineffective on single CP systems.
EPFS/DPFS	Enable/Disable PFS microtraps	Provide a more convenient method of enabling or disabling PFS microtraps than setting or clearing a bit in PARAM word 17. Value entered into DEC by initializing command buffer is kept in FCTITC PP memory. During subsequent execution of test, FCTITC modifies this

Action	Command	Description
		Note that when the test is running, the current section number corresponds to the sequence in which the section numbers were entered. For the example above the displayed section numbers would be 00 through 04. Note however that on the first line of the running display the xx in the TIxx entry will give the normal section number. This command can be entered from the keyboard or a command buffer after FCTITC stops for parameter entries. It cannot be entered after test execution commences. In a command buffer it should immediately follow the RU command that gives execution control to FCTITC.

4.3 NORMAL MESSAGES

None.

4.4 ERROR MESSAGES

Depending on the type of error detected, TRAP displays one of the following error messages.

4.4.1 Data Comparison Error Message

An example of a standard error message issued by FCTITC for a data comparison error is shown below.

```
TRAP  SE   PC0000 S0008  SB0002 C0021  TI08  81/10/30
EC1=0000 EC2=0000 TE=0001 RN=XXXX


P REG          00 00 B0 00 00 00 40 34
S REG          00 00 04 31 04 31 04 31
MON PACK       004100  PROCESS  0000 B000 0000 4030
JOB PACK       004280  PROCESS  0000 B000 0000 4068
COMPARE NO     0003
MASK           FFFF FFFF FFFF FFFF
XPCTD          0000 0000 0000 0000
RECVD          0000 0000 0000 1000
ADRS OF XPCTD  0044E8
ADRS OF RECVD  000088
ADRS INDEX     0000
```

```
TRAP  SE  PC0000 CP0 S0008  SB0002 C0021  TI08  81/10/30
EC1=0000 TE=0001


P REG          00 00 B0 00 00 00 40 34
S REG          00 00 04 31 04 31 04 31
MON PACK       004100  PROCESS  0000 B000 0000 4030
JOB PACK       004280  PROCESS  0000 B000 0000 4068
COMPARE NO     0003
MASK           FFFF FFFF FFFF FFFF
XPCTD          0000 0000 0000 0000
RECVD          0000 0000 0000 1000
ADRS OF XPCTD  0044E8
ADRS OF RECVD  000088
ADRS INDEX     0000
```

where:

LINE 0

TRAP header	See Running Display, paragraph 4.1.2

LINE 1

EC1	Error Code 1 (see text).
TE	Total Errors since start of test.

LINE 2

	Not used (blank)

LINE 3

	Not used (blank)

LINE 4

P REG	Contents of P register.

LINE 5

S REG	Contents of S register.

LINE 6

MON PACK	Byte address of Monitor Exchange Package
PROCESS	Contents of first word in package (P).

LINE 7

JOB PACK	Byte address of Job Exchange Package.
PROCESS	Contents of first word in package (P).

LINE 8

COMPARE NO	Compare Number (see following text).

LINE 9

MASK	Mask used to indicate which bits of the received data word are being tested.

LINE 10

XPCTD	Expected data.

LINE 11

RECVD	Received data.

LINE 12

ADRS OF XPCTD	Byte address of the expected result data area (see following text).

LINE 13

ADRS OF RECVD	Byte address of the received result data area (see following text).

LINE 14

ADRS INDEX	Address index for block compares (see following text).

Aside from the S register and the address index, all displayed addresses are byte addresses. There are minor variations in the spacing (by byte or parcel) for some of the data which is displayed. These variations are a result of the differences between the processors.

The EC1 error code is set with a value of $1C20_{16}$ when the MAC channel is found unable to transfer data after a function was sent by FCTITC. No other EC1 values have been implemented. For other errors EC1 will remain set to 0000_{16}.

The compare number indicates which comparison within the current condition caused the error. Numbering of comparisons commences with number 1. The comparison number correlates the display with the listing by indicating which compare directive within the condition has reported the error.

The address of the expected data is a byte address pointing to the word containing or the word beginning the expected data.

If the expected data is immediate data, the 64-bits of expected data may begin on any parcel boundary. The address of the expected data is shown as the byte address of the word where the data started. The address index is not used when referencing immediate data.

If the expected data is not immediate data, the expected address is the byte address of the word containing the first 64-bits of expected data. The address index is a word offset from the address of the expected data and points to the data that was used in the failing comparison.

The address of the received data is a byte address pointing to the word containing the received data. The address index is a word offset from the address of the received data and points to the data in error.

If the address index is nonzero, a block comparison occurred. If multiple errors occur in a block comparison, only the data for the first detected error is displayed. The address index may be used for both the expected and received data at the same time under the conditions described above. For more details see section II-7 regarding Control Command Tables.

4.4.2 Processor Hung Error Message

If an operation of the central processor is initiated and does not subsequently halt, a display similar to the following may occur.

```
TRAP  SE  PC0000 S0008 CP0 SB0000 C0001  TI08  81/10/30
EC1=0000 TE=0001


P REG          00 00 B0 00 00 00 40 34
S REG          00 00 04 31 04 31 04 31
MON PACK       004100  PROCESS  0000 B000 0000 4030
JOB PACK       004280  PROCESS  0000 B000 0000 4068

               CP  HUNG
```

LINES 0-7
See description in Data Comparison Error Message (above).

LINE 8
Not used (blank)

LINE 9
CP HUNG Processor hung indication.

When the processor halts or hangs up, the S and P registers are read immediately for the purpose of displaying their values. FCTITC then sends a deadstart function to dump the processor register file to central memory. This may affect the S and P registers. If you display the S and P registers using the CMSE AR display a difference in values may be noted.

4.4.3 Program Error Message

The program error message is similar to the processor hung display above except the last line reads PROGRAM ERR. This indicates that FCTITC detected an error while interpreting control commands from central memory. The error could be caused by bad data loaded from disc to central memory or by an overwrite of the control tables in central memory due to a CP error.

4.4.4 System Register Error Display

If an error is detected while checking either a processor hardware register or a processor soft register (dedicated register file register accessed by microcode), a display similar to that shown below occurs.

```
TRAP  SE  PC0000 S0000 CP0 SB0000 C0003  TI08  81/10/30
EC1=0000 TE=0001


P REG          00 00 B0 00 00 00 40 30
S REG          00 00 04 31 04 31 04 31
MON PACK       004100  PROCESS  0000 B000 0000 4030
JOB PACK       000000  PROCESS  0000 0000 0000 0000
COMPARE NO     0005
MASK           FFFF FFFF FFFF FFFF
XPCTD          0000 0000 0000 0000
RECVD          0000 0000 0000 1000
REG ADRS       0048
```

LINES 0-11
See description in Data Comparison Error Message.

LINE 12
REG ADRS Register address

4.4.5 Miscellaneous Error Messages

The following single line error messages may also occur. In all cases, FCTITC exits to the CMSE idler since recovery is not possible.

ABORT - MAC ERR	An error occurred during initialization when FCTITC was reading the EID registers on each radial interface.
ABORT - ELEMENT ID ERR	During initialization while reading EID registers, an incorrect or too few responses were received. Apparently a required system element was not connected.

4.5 SCOPE MODE CONTROL AND TRIGGERING

A scope loop can be used in any test where CP instruction level execution is initiated by FCTITC. This is true for all conditions except for those in subsection 0 of each section. See section II-7 for further information concerning the scoping of any condition in any subsection 0 of TRAP.

The CMSE command SSM sets the scope mode when the test is at a stop. When execution is restarted with a space bar, FCTITC forces the repeat of the current condition and it enters the scope loop. If a number of errors have occurred during the condition, more than one space bar entry may be required to reach the end of the condition and enter the loop.

Activity of the MAC channel flag on display and appearance of the SM code on the running display line indicates the loop is running. Type UP,x (CMSE command; x is the PP in which FCTITC resides) to exit the scope loop. When the test exits the loop, it stops on an error if an error is present and the stop on error bit is set (default). If no error is present, it stops at the end of the current condition. This occurs upon exit from the scope loop even though the end of condition stop may not be set. If the same scope loop is to be reentered, press the space bar until this occurs; otherwise, clear the scope mode bit (CSM) and press the space bar to continue. A scope may be triggered using the pulse generated by the deadstart function which performs the half-exchange-in of the monitor process.

PART II

TEST DESCRIPTIONS

MEMORY TESTS (CMEM)
RANDOM COMMAND TESTS (RCT1)
RANDOM COMMAND TESTS (RCT2)
FIXED OPERAND COMMAND TESTS (FCT1-3,5)
VIRTUAL MODE INSTRUCTION LEVEL TESTS (FCT9)
EXCHANGE TESTS (EXCH)
TRAP TESTS (TRAP)

SECTION II-1

MODEL INDEPENDENT CENTRAL MEMORY TESTS - CMEM

==========

PROGRAM DESCRIPTION

GENERAL

CMEM is organized into eleven sections. Sections 1-5 test central memory with five unique data patterns. Section 6 uses a marching pattern. Section 7 uses an addressing pattern where the word address of the word being accessed is used as the data word. Section 8 is the complement of section 7 word addresses.

Data compare operations can be switched on or off for sections 1-5. Section 9 tests memory using a 5-byte data pattern. Section 10 tests memory in 16-word blocks. Section 11 tests memory with a randomly-generated data pattern.

SECTION DESCRIPTIONS

Section 1 (SECT1)

Test memory with zero data pattern. Memory is written from FWA to LWA with zero data pattern, then one word at a time is read and compared with word written.

Pattern is 00 - 00_{16}.

Subsection 0 writes memory with pattern.
Subsection 1 reads and compares data.

The memory test area may be restricted by using the ADDR,p1,p2 command. Parameter p1 is FWA and p2 is LWA. Compare may be switched off.

Section 2 (SECT2)

Test memory with ones data pattern. Memory is written from FWA to LWA with ones data pattern, then one word at a time is read and compared with word written.

Pattern is FF - FF_{16}.

Subsection 0 writes memory with ones pattern.
Subsection 1 reads and compares data.

The memory test area may be restricted by using the ADDR,p1,p2 command. Parameter p1 is FWA and p2 is LWA. Compare may be switched off.

Section 3 (SECT3)

Test memory using a 5 pattern from FWA to LWA. Memory is written from FWA to LWA with data pattern, then one word at a time is read and compared with word written.

Example: $01010101\text{-}01010101_2$
5555555555555555_{16}

Subsection 0 writes memory with pattern.
Subsection 1 reads and compares data.

The memory test area may be restricted by using the ADDR,p1,p2 command. Parameter p1 is FWA and p2 is LWA. Compare may be switched off.

Section 4 (SECT4)

Test memory using an A data pattern. Memory is written from FWA to LWA with data pattern, then one word at a time is read and compared with word written.

Example: $01010101\text{-}01010101_2$
$AAAAAAAAAAAAAAAA_{16}$

Subsection 0 writes memory with pattern.
Subsection 1 reads and compares data.

The memory test area may be restricted by using the ADDR,p1,p2 command. Parameter p1 is FWA and p2 is LWA. Compare may be switched off.

Section 5 (SECT5)

Test memory using a DF pattern. Memory is written from FWA to LWA with data pattern, then one word at a time is read and compared with word written.

Example: $11011111\text{-}11011111_2$
$DFDFDFDFDFDFDFDF_{16}$

Subsection 0 writes memory with pattern.
Subsection 1 reads and compares data.

The memory test area may be restricted by using the ADDR,p1,p2 command. Parameter p1 is FWA and p2 is LWA. Compare may be switched off.

Section 6 (SECT 6)

Test memory using a marching pattern. Starting at FWA, zeros are written throughout memory testing area to LWA. Then, starting at FWA, word 0 is read and checked for zero (00-00). After compare, ones are written into word 0. Next, word 1 is read and checked for zero. After compare, ones are written into word 1. Process is repeated for each word through LWA. Then starting at FWA each word is compared with ones pattern ($FF\text{-}FF_{16}$).

The memory test area may be restricted by using the ADDR, p1, p2 command. p1 is FWA and p2 is LWA. Note that data compare cannot be switched off for this section except for scope loop operation.

Section 7 (SECT7)

Test memory with word address pattern. The address of the word being accessed is used as the data word.

Example: address $3FDB_{16}$ equals 00003FDB-00003FDB$_{16}$

Address is in both the upper four bytes and lower four bytes of data word. The memory testing area may be restricted by using the ADDR,p1,p2 command where p1 is FWA and p2 is LWA. Note that
data compare cannot be switched off for this section except for scope loop operation.

Section 8 (SECT8)

Test memory with the complement of the address of the word being accessed as the data word.

Example: address $3FDB_{16}$ equals FFFFC024-FFFFC024$_{16}$

The complemented address is in both the upper four bytes and lower four bytes of the data word. The memory testing area may be restricted by using the ADDR,p1,p2 command. p1 is FWA and p2 is LWA. Note that data compare cannot be switched off for this section except for scope loop operation.

Section 9 (SECT9)

Test memory using the load and store bytes. Memory is written from FWA to LWA five bytes at a time. Then memory is read from FWA to LWA five bytes at a time and compared to bytes written.

Subsection 0 writes memory in five byte increments.

Subsection 1 reads memory in five-byte increments and compares.

Section 10 (SECTA)

Test memory using multiple register instructions (LMULT, SMULT). Memory is written from FWA to LWA in increments of 16-word blocks. Then memory is read in 16-word blocks and compared to blocks written.

Pattern is DFDFDFDFDFDFF00F$_{16}$.

Subsection 0 writes memory with SMULT instruction.

Subsection 1 reads memory with LMULT instruction and compares.

The memory test area may be restricted by using the ADDR,p1,p2 command. p1 is FWA and p2 is LWA. Note that data compare cannot be switched off for this section.

Section 11 (SECTB)

Test memory using a random data pattern. Memory is written from FWA to LWA with generated random data pattern. Pattern is then regenerated, memory is read and compared to regenerated pattern.

Subsection 0 writes memory one word at a time.

Subsection 2 reads memory one word at a time and compares.

Note that scope loop does not apply to this section.

SECTION II-2

RANDOM COMMAND TEST 1

==========

PROGRAM DESCRIPTION

RCT1 generates a random set of instructions and operands each pass. The random set of instructions are then executed two different ways. The instructions are simulated using a basic set of instructions and the results of the simulation saved. The instructions are then executed by the machine hardware and the results of the machine execution are saved. The simulator and machine results are then compared and any difference is flagged as an error.

GENERAL

The basic operation performed by RCT1 each pass is as follows:

1) Display required messages.

2) Process any keyboard operator requests.

3) Generate random instructions, random A and X register values and random data in memory buffers.

4) Simulate the random set of instructions using the A and X register values and memory buffer data. Save A and X register contents after simulation, as well as the final contents of the memory buffer data.

5) Execute the random set of instructions using the actual machine hardware. Machine execution will use the same A and X register values and random data in the memory buffer. After machine execution, save A and X register contents and final memory buffer data.

6) Compare simulator and machine A and X register results and memory buffer final results. Any differences will result in an error being recorded and reported.

7) If trim parameter is selected and a miscompare occurs, RCT1 will reduce the failing set of instructions to the minimum number of instructions needed to cause the failure, usually one. Reduction technique is to attempt to replace each instruction, one at a time, with a pad instruction of $1AFF_{16}$ and still have the error occur.

SECTION II-3

RANDOM COMMAND TEST 2

PROGRAM DESCRIPTION

GENERAL

The test is divided into sections with each section testing a subgroup of BDP instructions. Each section is divided into subsections with each subsection testing an individual BDP instruction type.

The general method of testing is to pack the instruction being tested into a machine execution list beginning at location I_MLST0. This list contains from 1 to 10 copies of the instruction depending on the instruction type. If the instruction alters an X register (i.e., compare), additional instructions are packed into the list to save intermediate results. In addition, randomly formed descriptors, which reference operands that have also been randomly formed, are packed into the list.

Once established, the machine execution list is simulated by the subsection and its subroutines. The results of the simulation are stored in the simulator destination field buffer beginning at location V_SDF0. The machine instruction list is then executed with the results being stored in the machine destination field buffer beginning at location V_MDF0. The result format is determined by the instruction being tested. The machine results are then compared with the simulator results.

SECTION DESCRIPTIONS

Section 0 (SEC0)

This section (SEC0) tests the BDP numeric instructions. The section is divided into subsection, each of which tests an individual BDP numeric instruction type. The section does the following.

- Generates five random BDP decimal operands (types 0 through 6, 12, 13) and stores them in 40-byte fields, V_OPA to V_OPE. The operands are various section 0 subsections.
- Generates descriptor table entries for the five operands generated and stores them in the T_DES descriptor table as follows:

  Word 0 = type/length for operand A, field V_OPA
  Word 1 = type/length for operand B, field V_OPB
  Word 2 = type/length for operand C, field V_OPC
  Word 3 = type/length for operand D, field V_OPD
  Word 4 = type/length for operand E, field V_OPE

- Uses ten 40-byte fields, V_MDF0 to V_MDF9 for BDP numeric destination operands and five 40-byte fields, V_OPA to V_OPE for source fields. Initializes the destination fields with operands selected from the operands V_OPA to V_OPE and copies the descriptors for these operands into the machine instruction list with addressing offsets added to reference their memory locations. The results of executing the machine list are as follows:

Source Operand	Destination Operand	Results
V_OPB	V_OPA	V_MDF0
V_OPC	V_OPA	V_MDF1
V_OPC	V_OPB	V_MDF2
V_OPD	V_OPB	V_MDF3
V_OPD	V_OPC	V_MDF4
V_OPE	V_OPC	V_MDF5
V_OPA	V_OPD	V_MDF6
V_OPE	V_OPD	V_MDF7
V_OPA	V_OPE	V_MDF8
V_OPB	V_OPE	V_MDF9

  The length of the result operand is defined by the destination operand descriptor.

- Halts on error and displays an error message. In addition, for subsections 0 through 4, and 6, bits 2^{32} through 2^{41} of the X2 register (job process) indicate which machine destination fields are in error. (2^{32} equals field 0, 2^{33} equals field 1, etc.)

Subsection 0 (SOS0)

This subsection (SOS0) tests the BDP numeric sum (70) instruction.

- Sets up machine destination operand fields, V_MDF0 to V_MDF9, by moving operands (A through E) into the appropriate destination field described by the descriptors in the machine instruction list.

- Simulates the BDP numeric sum result and saves the simulated results in 40-byte fields V_SDF0 to V_SDF9.

- Executes a fixed sequence of BDP numeric sum instructions to add source operand fields to destination operand fields.

- Compares simulated results, V_SDF0 to V_SDF9, with machine results, V_MDF0 to V_MDF9. If results compare, branches to next subsection; else halts and displays:

  SOS0 ERROR (BDP NUMERIC SUM)

Subsection 1 (S0S1)

This subsection (S0S1) tests the BDP numeric difference (71) instruction. The section is similar to S0S0 except that BDP numeric differences are simulated and executed. The following error message is displayed on error:

S0S1 ERROR (BDP NUMERIC DIFFERENCE)

Subsection 2 (S0S2)

This subsection (S0S2) tests the BDP numeric product (72) instruction. The section is similar to S0S0 except that BDP numeric products are simulated and executed. The following error message is displayed on error:

S0S2 ERROR (BDP NUMERIC PRODUCT)

Subsection 3 (S0S3)

This subsection (S0S3) is designed to test the BDP numeric quotient (73) instruction. The section is similar to S0S0 except that BDP numeric quotients are simulated and executed. The following error message is displayed on error:

S0S3 ERROR (BDP NUMERIC QUOTIENT)

Subsection 4 (S0S4)

This subsection (S0S4) tests the BDP numeric scale (E4) and scale rounded (E5) instructions.

- Generates ten random shift count values. The shift count values are stored in V_SCT (shift count table) in consecutive 4-byte fields. Shift count values are copied from V_SCT to the D field of the appropriate instruction in the machine instruction list.

- Simulates the scale/scale-rounded result and saves the simulated results in V_SDF0 to V_SDF9.

- Executes a fixed sequence of BDP numeric scale/scale- round instructions to scale source operand fields to destination operand fields as follows:

  Scale operand B (V_OPB) to type A (V_MDF0)
  Scale operand C (V_OPC) to type A (V_MDF1)
  Scale operand C (V_OPC) to type B (V_MDF2)
  Scale operand D (V_OPD) to type B (V_MDF3)
  Scale operand D (V_OPD) to type C (V_MDF4)
  Scale round operand E (V_OPE) to type C (V_MDF5)
  Scale round operand A (V_OPA) to type D (V_MDF6)
  Scale round operand E (V_OPE) to type D (V_MDF7)
  Scale round operand A (V_OPA) to type E (V_MDF8)
  Scale round operand B (V_OPB) to type E (V_MDF9)

- Compares simulated results, V_SDF0 to V_SDF9, with machine results, V_MDF0 to V_MDF9. If results compare, branches to next subsection; else halts and displays:

  SOS4 ERROR (BDP NUMERIC SCALE/SCALE ROUNDED)

Subsection 5 (SOS5)

This subsection (SOS5) tests the BDP numeric compare (74) instruction.

- Sets up machine destination operand fields, V_MDF0 to V_MDF9, by moving operands (A through E) into the appropriate destination fields described by the descriptors in the machine instruction list.

- Simulates the BDP numeric compare result and saves each simulated result (lower X1 value) in one of ten four byte fields, V_SCDR0 to V_SCDR9.

- Executes a fixed sequence of BDP numeric compare (74) instructions to compare source operand fields to destination operand fields as follows. After each instruction execution, the result (X1 lower) is saved in consecutive four byte fields, V_MCDR0 to V_MCDR9).

  B (V_OPB) to A (V_MDF0), result to V_MCDR0
  C (V_OPC) to A (V_MDF1), result to V_MCDR1
  C (V_OPC) to B (V_MDF2), result to V_MCDR2
  D (V_OPD) to B (V_MDF3), result to V_MCDR3
  D (V_OPD) to C (V_MDF4), result to V_MCDR4
  E (V_OPE) to C (V_MDF5), result to V_MCDR5
  A (V_OPA) to D (V_MDF6), result to V_MCDR6
  E (V_OPE) to D (V_MDF7), result to V_MCDR7
  A (V_OPA) to E (V_MDF8), result to V_MCDR8
  B (V_OPB) to E (V_MDF9), result to V_MCDR9

- Compares simulated results, V_SCDR0 to V_SCDR9, with machine results, V_MCDR0 to V_MCDR9. If results compare branches to next subsection; else halts and displays:

  SOS5 ERROR (BDP NUMERIC COMPARE)

Subsection 6 (SOS6)

This subsection (SOS6) tests the BDP numeric move (75) instruction.

- Generates five random BDP (type 7 through 11, 14, 15) operands. The operands generated (F through J) are based on existing (A through E) operands and are stored in 40-byte fields, V_OPF to V_OPJ. Descriptor word entries for the (F through J) operands are stored in words 5 through 9 of the T_DES descritpor table.

- Generates ten random descriptor types 0 through 15 and stores them in words 10 through 19 of table T_DES. They are also packed into the destination descriptor fields of the machine instruction list.

- Simulates the BDP numeric move instructions and saves the simulated results in fields V_SDF0 to V_SDF9.

- Executes a fixed sequence of BDP numeric move instructions to move source operand fields to destination operand fields.

```
operand A (V_OPA) to V_MDF0
operand B (V_OPB) to V_MDF1
operand C (V_OPC) to V_MDF2
operand D (V_OPD) to V_MDF3
operand E (V_OPE) to V_MDF4
operand F (V_OPF) to V_MDF5
operand G (V_OPG) to V_MDF6
operand H (V_OPH) to V_MDF7
operand I (V_OPI) to V_MDF8
operand J (V_OPJ) to V_MDF9
```

- Compares simulated results, V_SDF0 to V_SDF9, with machine results, V_MDF0 to V_MDF9. If results compare, branches to next section; else halts and displays:

  S0S6 ERROR (BDP NUMERIC MOVE)

Section 1 (SEC1)

This section (SEC1) tests the BDP byte instructions. The section is divided into subsections with each subsection designed to test an individual BDP byte instruction type. The section does the following.

- Fills a 256-byte field V_BUFA with random data from V_RNB.

- Executes selected subsections and returns to main RUN routine.

Subsection 0 (S1S0)

This subsection (S1S0) tests the BDP byte compare (77) instruction.

- Generates two random lengths (0 through 256) called L1 and L2. L2 is greater than or equal to L1.

- Generates descriptor words for V_BUFA, V_BUFB, and V_BUFC fields and packs them into the machine instruction list.

- Initializes V_BUFB by moving bytes from V_BUFA per length L1 and filling with a random fill byte to length L2.

- Initializes V_BUFC by moving bytes from V_BUFA per length L1 and filling with ASCII 20 codes to length L2.

- Simulates the BDP byte compare result (S0, S1 lower value). Packs the results into words (X0 equals upper, X1 equals lower) and saves them in V_SDF0 to V_SDF0+5.

- Executes a fixed sequence of BDP byte compare instructions to compare the following fields and saves the packed results (X0, X1 lower values) in V_MDF0 to V_MDF0+5.

  V_BUFA, L1 and V_BUFB, L1 to V_MDF0
  V_BUFC, L2 and V_BUFA, L1 to V_MDF0+1
  V_BUFA, L1 and V_BUFC, L2 to V_MDF0+2
  V_BUFA, L2 and V_BUFB, L2 to V_MDF0+3
  V_BUFA, L2 and V_BUFB, L1 to V_MDF0+4
  V_BUFC, L1 and V_BUFA, L2 to V_MDF0+5

- Compares simulated results, V_SDF0 to V_SDF0+5 with machine results, V_MDF0 to V_MDF0+5. If results compare, branches to next subsection; else halts and displays:

  S1S0 ERROR (BDP BYTE COMPARE)

Subsection 1 (S1S1)

This subroutine (S1S1) tests the BDP byte compare collated (E9) instruction.

- Generates two random lengths (0 through 256) called L1 and L2. L2 is greater than or equal to L1.

- Generates descriptor words for V_BUFA and V_BUFB fields and packs them into the machine instruction list.

- Initializes V_BUFC as translation table by filling with random lengths of random bytes.

- Simulates the BDP byte compare collated result (X0, X1 lower values) for the following combinations using V_BUFC as a translation table. The results are packed into words (X0 equals upper, X1 equals lower) and saved in V_SDF0 to V_SDF0+2.

  V_BUFA, L2 and V_BUFB, L2 to V_SDF0
  V_BUFA, L2 and V_BUFB, L1 to V_SDF0+1
  V_BUFB, L1 and V_BUFA, L2 to V_SDF0+2

- Executes a fixed sequence of BDP byte compare collated instructions to compare the following fields and saves packed results (X0, X1 lower values) in V_MDF0 to V_MDF0+2. V_BUFC is used as the translation table.

  V_BUFA, L2 and V_BUFB, L2 to V_MDF0
  V_BUFA, L2 and V_BUFB, L1 to V_MDF0+1
  V_BUFB, L1 and V_BUFA, L2 to V_MDF0+2

- Compares simulated results, V_SDF0 to V_SDF0+2, with machine results, V_MDF0 to V_MDF0+2. If results compare, branches to next subsection, else halts and displays:

  S1S1 ERROR (BDP BYTE COMPARE COLLATED)

Subsection 2 (S1S2)

This subsection (S1S2) tests the BDP byte scan (EA) instruction.

- Generates a destination descriptor word for V_BUFA, with a random length (0 through 256 bytes), and packs it into the machine instruction list.

- Generates three source descriptor words and packs them into the machine instruction list. The source operands are stored in buffers with the following data (instruction assumes 32-byte length).

  V_BUFD (fixed 32 bytes, zeros)
  V_BUFE (fixed 32 bytes, ones)
  V_BUFB (random bytes)

- Moves 32 bytes of random data to V_BUFB.

- Simulates the BDP byte scan result (X0, X1 lower values) using V_BUFA as the destination field. Packs the results into words (X0 equals upper, X1 equals lower) and saves them in V_SDF0 to V_SDF0+2.

- Executes a fixed sequence of BDP byte scan instructions for the following source fields using V_BUFA as the destination field. Saves packed results in V_MDF0 to V_MDF0+2.

  V_BUFD result to V_MDF0
  V_BUFE result to V_MDF0+1
  V_BUFB result to V_MDF0+2

- Compares simulated results, V_SDF0 to V_SDF0+2, with machine results, V_MDF0 to V_MDF0+2. If results compare, branches to next subsection, else halts and displays:

  S1S2 ERROR (BDP BYTE SCAN)

Subsection 3 (S1S3)

This subsection (S1S3) tests the BDP byte translate (EB) instruction.

- Generates two random lengths (0 through 256 byte values) for the source and destination fields. Descriptor words for the fields are packed into the machine instruction list referencing V_BUFB as the source operand and V_BUFC as the destination operand.

- Generates random data for source field V_BUFB.

- Simulates BDP byte translate instruction to translate source field V_BUFB using V_BUFA as the translation table. The simulated result (destination field) is stored in field V_SBUF.

- Executes a BDP byte translate instruction to translate source field V_BUFB to destination field V_BUFC. V_BUFA is used as the translation table.

- Compares simulated result V_SBUF with machine result V_BUFC. If results compare, branches to next subsection, else halts and displays:

  S1S3 ERROR (BDP BYTE TRANSLATE)

Subsection 4 (S1S4)

This subsection (S1S4) tests the BDP byte move (EC) instruction.

- Generates two random lengths (0 through 256 byte values) for the source and destination fields. Descriptor words for the fields are packed into the machine instruction list referencing V_BUFA as the source field and V_BUFB as the destination field.

- Clears V_BUFB and V_SBUF.

- Simulates BDP byte move result using V_BUFA as the source field and V_SBUF as the destination field.

- Executes a BDP byte move instruction using V_BUFA as the source field and V_BUFB as the destination field.

- Compares simulated result V_SBUF with machine result V_BUFB. If results compare, branches to next subsection, else halts and displays:

  S1S4 ERROR (BDP BYTE MOVE)

Subsection 5 (S1S5)

This subsection (S1S5) tests the BDP byte edit (ED) instruction.

- Generates a random BDP operand (type 0 through 9, 12, 13) for source field V_BUFB.

- Packs descriptor words into the machine instruction list referencing V_BUFB as the source field and V_BUFC as the destination field.

- Clears destination field V_BUFC and simulated destination field V_SBUF.

- Simulated BDP byte edit result using V_BUFB as the source field and V_BUFA as the edit mask. The simulated result (destination field) is stored in field V_SBUF. Subroutine EDIT may modify the random mask data in V_BUFA as it simulates.

- Executes a BDP byte edit instruction to edit source field V_BUFB to destination field V_BUFC using V_BUFA as the edit mask.

- Compares simulated destination field V_SBUF with machine destination field V_BUFC. If results compare, branches to next section; else halts and displays:

  S1S5 ERROR (BDP BYTE EDIT)

Section 2 (SEC3)

This section (SEC3) tests the BDP immediate data instructions. The section is divided into subsections, each of which tests an individual BDP immediate data instruction type. The section does the following:

- Generates five random BDP operands (types 0 through 6, 10 through 15) and stores the operands in 40-byte fields, V_OPA to V_OPE.
- Generates descriptor table entries for the five operands and stores them in the T_DES descriptor table as follows:

  Word 0 = type/length for operand A, field V_OPA
  Word 1 = type/length for operand B, field V_OPB
  Word 2 = type/length for operand C, field V_OPC
  Word 3 = type/length for operand D, field V_OPD
  Word 4 = type/length for operand E, field V_OPE

- Executes selected subsections and returns to main RUN routine.

NOTE

Due to test restructuring, section 2 has been tagged SEC 3 and S3.

Subsection 0 (S3S0)

This subsection (S3S0) tests the BDP immediate data move (F9) instruction.

- Generates ten BDP immediate data move instructions using random legal values for the j, k, and D fields. The instructions are packed into the machine instruction list.
- Generates ten random operand descriptor words referencing destination fields, V_MDF0 to V_MDF9, and packs them into the machine instruction list.
- Simulates the BDP immediate data move results for the ten instructions generated. The results (destination fields) are saved in 40-byte fields V_SDF0 to V_SDF9.
- Executes the BDP immediate data move instruction list.
- Compares simulated results, V_SDF0 to V_SDF9, with machine results, V_MDF0 to V_MDF9. If results compare, branches to next subsection, else halts and displays:

  S3S0 ERROR (BDP IMMEDIATE DATA MOVE)

  On error, bits 2^{32} to 2^{41} of the X2 register (job process) indicate which machine destination fields are in error. (2^{32} equals field 0, 2^{33} equals field 1, etc.)

Subsection 1 (S3S1)

This subsection (S3S1) tests the BDP immediate data compare (FA) instruction.

- Generates five BDP immediate data compare (numeric) instructions and stores simulated compare results (X1 lower values) in four byte fields, V_SCDR0 to V_SCDR4. Each instruction generated is followed by a store byte (DB) instruction which saves the machine compare result (lower X1) values in four byte fields, V_MCDR0 to V_MCDR4. The instructions generated are packed into the machine execution list. The five compare instructions will compare operand fields V_OPA to V_OPE with random legal values generated for the D fields.

- Generates two BDP immediate data compare (ASCII) instructions and stores simulated compare results (X1 lower values) in four byte fields, V_SCDR5 and V_SCDR6. Each of the instructions is followed by a store byte (DB) instruction which saves the machine compare result (lower X1) values in four byte fields, V_MCDR5 and V_MCDR6. The instructions generated are added to the machine instruction list. The two instructions generated will compare operand fields, V_OPF, V_OPG, with random legal values generated for the instruction D fields. Subroutine GCIA also generates operand fields V_OPF, V_OPG, and their associated descriptor words which are packed into the machine instruction list.

- Executes the BDP immediate data compare instruction list.

- Compares simulated results, V_SCDR0 to V_SCDR6, with machine results, V_MCDR0 to V_MCDR6. If results compare, branches to next subsection; else halts and displays:

  S3S1 ERROR (BDP IMMEDIATE DATA COMPARE)

Subsection 2 (S3S2)

This subsection (S3S2) tests the BDP immediate data add (FB) instruction.

- Sets up descriptor words for the instruction destination fields and packs them into the machine instruction list. Initializes fields V_MDF0 to V_MDF9 by moving operands V_OPA to V_OPE into the appropriate destination field.

- Generates ten BDP immediate data add instructions and simulates results. The instructions are generated using random legal values for the D field and are packed into the machine instruction list. The destination operand fields referenced are V_MDF0 to V_MDF9. The simulated results are saved in fields V_SDF0 to V_SDF9.

- Executes the BDP immediate data add instruction list.

- Compares simulated results, V_SDF0 to V_SDF9, with machine results, V_MDF0 to V_MDF9. If results compare, branches to next subsection, else halts and displays:

S3S2 ERROR (BDP IMMEDIATE DATA ADD)

On error, bits 2^{32} to 2^{41} of the X2 register (job process) indicate which machine destination fields are in error. (2^{32} equals field 0, 2^{33} equals field 1, etc.)

SECTION II-4

FIXED OPERAND COMMAND TESTS - FCT1-3,5

==========

FCT1 TEST

This test is an inverted pyramid style of test. Each instruction test lists the instruction which should be checked before this test is run. The test is set up to automatically run each instruction test in the correct sequence for meaningful testing.

FCT1 does some preliminary testing of the usage set for FCT2 and makes sure that no op code will hang the machine. The test does not generate error messages other than expected and received results.

A summary of section activity appears with FCT1 Test and Index, section I-4. For description of section and subsection makeup, refer to the listings.

FCT2 TEST

This test is an inverted pyramid style of test, with minimal operator control, and minimal ability to generate error messages. FCT2 tests the usage set for FCT3 and RCT1. FCT2 is a decision-point test. If it runs correctly, then further meaningful testing can be done at the instruction level. If it reports errors, or if it will not run at all, then there is little point in running any CPU-based test. Further diagnosis should be done using FCT1, FCT5, or the microcode tests.

If it becomes necessary to find out exactly what is causing FCT2 to fail, these steps can be followed:

- Determine the failing section, subsection and condition from the display.
- Find this condition in the listings.
- Find the instruction which does the error check. Usually directly in front of call to the procedure named FAILURE.
- Determine from the listing which registers are being used in the error check.
- Obtain (from the actual hardware) the contents of these two registers.

A summary of section activity appears with FCT2 Test and Section Index. For description of section and subsection contents, refer to the listings.

FCT3 TEST

This is a usage set style of test, with full operator control and detailed error messages. The instructions used for housekeeping purposes are limited to those checked by FCT2 (the usage set). FCT3 tests all CPU-testable instructions using simple data patterns with emphasis on end cases.

A summary of section activity appears with FCT3 Test and Section Index, section II-4. For description of section and subsection makeup, refer to the listings.

FCT5 TEST

This test requires that FCT1 has been completed successfully. FCT5 tests each instruction that cannot be tested meaningfully from the CPU.

The test does not generate error messages other than expected and received results.

A summary of section activity appears with FCT5 Test and Section Index. For description of section and subsection makeup, refer to the listings.

SECTION II-5

VIRTUAL MODE INSTRUCTION LEVEL TEST - FCT9

==========

PROGRAM DESCRIPTION

GENERAL

This section describes the operation of the test and the testing done within each section. This section will be of particular interest if the test detects an error.

The test is divided into basic units known as conditions. Within a condition, there is a control table containing commands for the PP controller to set up the environment for the test, and CPU code which essentially define the operation being tested.

Controller Operation

The controller commands in each condition will set up data in central memory and processor registers, start CPU execution, and when the CPU halts, compare the results in central memory.

The data in central memory is used to control the operation of the slave PPs, to set up the various exchange packages, and to set up input data for the test instructions. Processor registers are loaded with the locations of exchange packages, and page table parameters.

To start CPU execution, the controller first triggers the slave PPs to read their control information and prepares them for their operation. The CPU is then deadstarted. This deadstart generates a triggering signal for the scope loop and starts the CPU running. The CPU, during its execution sequence, clears the 'deadstart flag' to trigger the slave PPs to begin their operation, and then executes the test instruction. The PP controller monitors the slave PPs and the CPU, waiting for them to finish their operation. The period of this wait is set by the hang count parameter (parameter 16). If any of the processors do not finish by the end of this period, a processors hung error message will be displayed, and all the processors are forced to finish. The results are then compared.

In scope mode, the above process is repeated indefinitely except that the results are not compared until the scope loop is exited. The scope loop is exited by entering the CMSE command, UP,x, where x is the number of the PP in which EXCHITC resides.

CPU Operation

The operation of the CPU can be divide into two categories depending on whether the Environment Interface is used or not.

Environment Interface (EI)

The EI is an executive state monitor program which implements some of the functions of the system.

The EI contains an executive state monitor and an executive state job exchange package. By virtue of the virtual machine identifier, the executive state job exchange package is interpreted by the machine as an A170 job package in executive state format. Therefore, an exchange to this executive state job package establishes the A170 state. Besides initializing the A170 state, the EI handles all executive state interrupt conditions and all A170 monitor mode error exits. Also, the EI implements the simulation of the CMU instructions.

Whenever the above mentioned functions are tested, the EI is included as part of the A170 system being tested by setting the Monitor Process State Pointer (MPS) and the Job Process State Pointer (JPS) to point to the EI's executive state monitor and job exchange packages; otherwise the MPS and JPS point to a different set of executive state monitor and job exchange packages and processes, which exclude the EI code. Excluding the EI shortens the amount of CP code executed and allows a much tighter scope loop.

Testing Without EI

If the EI is not used, the executive state monitor code consists of the following steps:

1) clearing the deadstart flag, which tells EXCHITC to trigger the slaves;

2) an exchange to the A170 job where the CPU is halted;

3) on a redeadstart, the code branches back to the beginning of the executive state monitor code.

This looping allows the CPU code to be executed repeatedly when the CP is deadstarted repeatedly in scope mode. The A170 state should never return to the executive state monitor because it would start an infinite loop. (Looping indefinitely will never occur in practice because EXCHITC will break in after a certain period). To avoid returning to the executive state monitor, the 170 job runs with traps enabled and all monitor mask bits cleared, therefore all executive state interrupt conditions will either stack, trap, or halt. The trap routine will halt the CPU. If there should be an A170 error exit from the A170 job, the A170 monitor will alter the P-register in the A170 error exchange package to point at a halt routine and exchange to it, halting the CPU. The exchange back to the error package restores the A170 monitor package to MA, and allows the error results to be dumped to the dump area.

Once the executive state monitor has exchanged to the A170 job, the A170 job will execute the test instruction sequence which will eventually end in a page fault instruction. The page fault will cause a real halt of the processor.

Therefore, in all cases, the processor will be halted in job mode and the result exchange package will be dumped to the dumped area by EXCHITC. The fact that the executive state job exchange package is always dumped to the dump area and never updates the job exchange package in central memory, allows the test instruction sequence to be executed repeatedly with no reinitialization, when in scope mode.

the test instruction sequence to be executed repeatedly with no reinitialization, when in scope mode.

Testing With EI

When the EI is used, it is altered slightly to conform to the testing format of FCT9 and EXCHITC. The modifications to EI are kept to a minimum to maintain the integrity of the EI program and to minimize the impact to the test of any future changes to EI.

The EIs executive state job exchange package is set up with data so that an exchange from executive state monitor mode will immediately start the execution of the A170 test instruction sequence. The A170 test instruction sequence must clear the deadstart flag to trigger the slave PPs before executing the test instruction. The test instruction will generally cause an exchange from A170 state to executive state monitor. If not, a following instruction will. In all cases, other than testing of CMU instruction, the exchange to EI will end in a loop in EI used to simulate a CP halt. This loop is changed to a branch to additional code. This additional code will reset the executive state monitor and job exchange packages' P-registers to their initial value at the beginning of the condition, and then halt the processor. Resetting the P-registers allow the test instruction sequence to be scoped. Note that scoping requires that the processor is operational enough that this code is reached and executed correctly.

SECTION DESCRIPTIONS

SECTION 0

Section 0 has a single subsection of four conditions used for initialization. They perform functions as follows:

Condition	Description
0	Load and check the PTA, PSM, and PTL register and set up a segment table containing two descriptors. This table is used with subsections of the test which perform tests without the EI.
1	Sets up registers and pointers for the trap mechanism. This is also associated with tests performed without the EI.
2	This condition is concerned with initiating EI which, with some assisstance, initializes itself. A temporay page table is created and a temporary monitor package in the EI code is modified to point to its segment table. The simulated halt command in EI is modified to return control to FCT9 where an actual HALT command will be executed. An EI initialization function request is set up and the CP is deadstarted to excute the EI initialization code.

The test (EXCHITC) control PP waits until EI generates a response to the function request indicating completion of initialization. Subsequently, the control PP issues an invalid function request which causes EI to branch to the actual HALT command. At this point EI has set up page and segment tables, stack frames, permanent monitor, and job exchange pachages, etc. and is ready for normal operation.

3 To complete initialization the control PP modifies the page table created by EI with a few additional entries required for FCT9. The test may select operation with or without EI by setting the MPS and JPS registers to point respectively to EIs or the test's monitor and job exchange packages.

SECTIONS 1 and 2

Sections 1 and 2 test the positive operation of the CRXj (660) and CWXj (670) instructions respectively. The testing within these two sections is similar. The subsections and conditions within these two sections are described below. Subsection 0 sets up processor registers and common data.

Subsection	Description
1	Tests the CRXj/CWXj instruction with different values in the J and K fields.

Conditions 0 to 6 use values of 1 to 7 in the K field using registers X1 through X7 to address central memory.

Condition	
0	CRX0/CWX0 X1
1	CRX0/CWX0 X2
2	CRX0/CWX0 X3
3	CRX0/CWX0 X4
4	CRX0/CWX0 X5
5	CRX0/CWX0 X6
6	CRX0/CWX0 X7

Conditions 7 to 13 use values of 1 to 7 in the J field loading registers X1 through X7.

Condition	
7	CRX1/CWX1 X0
8	CRX2/CWX2 X0
9	CRX3/CWX3 X0
10	CRX4/CWX4 X0
11	CRX5/CWX5 X0
12	CRX6/CWX6 X0
13	CRX7/CWX7 X0

Subsection	Description
2	Tests the addressing of central memory by the CRXj/CWXj instructions. A pattern of ten different addresses are used to test the instruction.

Condition

0	(Xk)= 17,777
1	(Xk)= 20,000
2	(Xk)= 37,777
3	(Xk)= 40,000
4	(Xk)= 77,777
5	(Xk)= 100,000
6	(Xk)= 177,777
7	(Xk)= 200,000
8	(Xk)= 375,777
9	(Xk)=7,777,776

The pattern was chosen to test a wide range of addresses, especially end cases. The last address, 7,777,776, is translated via the page table to 375,776.

Note that the reference address (RAc) is equal to zero. If there should be an error exit, the content of RA will be overwritten with the dumped register file. To avoid this, set scope mode and the contents of RAc can be displayed until the scope loop is exited, at which time the register file is dumped.

3 Tests the transmission of data patterns by the CRXj/CWXj instruction. The patterns used are:

Condition

0	data = octal 25252525252525252525
1	data = octal 52525252525252525252
2	data = octal 70707070707070707070
3	data = octal 07070707070707070707

At the end of each condition, the following comparisons are made:

1) Compare the dumped P register to the address of the instruction halting the processor.

2) Check that the dumped MCR is clear except for the page fault bit (this interrupt condition is used to halt the processor in A170 mode, see Testing Without EI, above).

3) Compare the dumped Xk register or the central memory location written to the expected data.

SECTION 3-4

Sections 3 and 4 test the positive operation of the RXj (014) and WXj (015) instructions in UEM (ECS mode). Section 3 tests the RXj instruction, and section 4 tests the WXj instruction. The subsections and conditions within these two sections are described below. Subsection 0 sets up processor registers and common data.

Subsection	Description
1	Tests the RXj/WXj instruction with different values in the J and K fields.

Condition 0 to 6 uses values of 1 to 7 in the K field, therefore, using registers X1 through X7 to address central memory.

Condition	
0	RX0/WX0 X1
1	RX0/WX0 X2
2	RX0/WX0 X3
3	RX0/WX0 X4
4	RX0/WX0 X5
5	RX0/WX0 X6
6	RX0/WX0 X7

Conditions 7 to 13 uses values of 1 to 7 in the J field, therefore, loading registers X1 through X7.

Condition	
7	RX1/WX1 X0
8	RX2/WX2 X0
9	RX3/WX3 X0
10	RX4/WX4 X0
11	RX5/WX5 X0
12	RX6/WX6 X0
13	RX7/WX7 X0

2	Tests the addressing of central memory by the RXj/WXj instruction. A pattern of ten different addresses are used to test the instruction.

Subsection	Description

Condition

0	(Xk)= 17,777
1	(Xk)= 20,000
2	(Xk)= 37,777
3	(Xk)= 40,000
4	(Xk)= 77,777
5	(Xk)= 100,000
6	(Xk)= 177,777
7	(Xk)= 200,000
8	(Xk)= 375,777
9	(Xk)=7,777,677

The pattern was chosen to test a wide range of addresses, especially end cases. The last address, 7,777,677, is translated via the page table to 375,677.

Note that the reference address (RAc) is equal to zero. If there should be an error exit, the content of RA will be overwritten with the dumped register file. To avoid this, set scope mode and the contents of RAc can be displayed until the scope loop is exited, at which time the register file is dumped.

3 Tests the transmission of data patterns by the RXj/WXj instruction. The patterns used are:

Condition

0	data = octal 2525252525252525252
1	data = octal 5252525252525252525
2	data = octal 7070707070707070707
3	data = octal 0707070707070707070

At the end of each condition, the following comparisons are made:

1) Compare the dumped P register to the address of the instruction halting the processor.

2) Check that the dumped MCR is clear except for the page fault bit (this interrupt condition is used to halt the processor in A170 mode, see Testing Without EI, above).

3) Compare the dumped Xk register or the central memory location written to the expected data.

SECTION 5

This section tests the positive operation of the REC (block read ECS, 011) instruction in UEM (ECS mode). Subsection 0 sets up processor registers and common data.

Subsection	Description
1	Tests the REC instruction with different values in the J field (ie. B1 through B7 are used).

Condition

0	REC B1+k
1	REC B2+k
2	REC B3+k
3	REC B4+k
4	REC B5+k
5	REC B6+k
6	REC B7+k

2 Tests the addressing of central memory and unified extended memory (ECS mode) by the REC instruction. A pattern of ten addresses is used. Each of the addresses is used in combination with the other addresses including itself. Therefore, each address in the pattern is used to read UEM to every address in the pattern, now as addresses in CM.

Condition

0-9	REC (B1)+k words from (adrs1-adrs10) to adrs1
10-19	REC (B1)+k words from (adrs1-adrs10) to adrs2
20-29	REC (B1)+k words from (adrs1-adrs10) to adrs3
30-39	REC (B1)+k words from (adrs1-adrs10) to adrs4
40-49	REC (B1)+k words from (adrs1-adrs10) to adrs5
50-59	REC (B1)+k words from (adrs1-adrs10) to adrs6
60-69	REC (B1)+k words from (adrs1-adrs10) to adrs7
70-79	REC (B1)+k words from (adrs1-adrs10) to adrs8
80-89	REC (B1)+k words from (adrs1-adrs10) to adrs9
90-99	REC (B1)+k words from (adrs1-adrs10) to adrs10

The reference addresses, RAc and RAe, are both equal to zero. Therefore, adrs1 through adrs10 are equal to the addresses given in the description of Section 4, Subsection 2.

3 Tests the transmission of data patterns by the REC instruction. The patterns used are:

Condition

0	data = octal 2525252525252525252
1	data = octal 5252525252525252525
2	data = octal 7070707070707070707
3	data = octal 0707070707070707070

Subsection	Description

4 Tests the summation of the B-register with the k-field by the REC instruction. The various combinations used are:

Condition	(Bj)	k	(Bj)+k	
0	2	-1	1	
	1	0	1	*this condition
1	0	1	1	tested in previous
2	-1	2	1	subsection

5 Tests the ability of the REC instruction to copy blocks of varying sizes. Since transfers are divided into blocks with boundaries of (0 mod 8) the transfer sizes and addresses were chosen so as to require a maximum number of blocks for a minimum transfer size. The block sizes in octal are:

Condition	
0	12
1	22
2	42
3	102
4	202
5	1,002
6	2,002
7	4,002
8	10,002
9	20,002

6 Tests the interruptible of the REC instruction. It checks that transfers greater than two blocks can be interrupted, and that the interruption occurs at an ECS record boundary, but not the last one. It also checks that transfer sizes of two ECS records in length cannot be interrupted (ie. cannot interrupt between the last two records).

Condition	Transfer Sizes	Expected Result
0	256 words 32 ECS records minimum of 4 blocks	Transfer is interrupted at an ECS record boundary
1	58 words 9 ECS records minimum of 2 blocks	Transfer is interrupted at an ECS record boundary, but not the last one.
2	15 words 2 ECS records	Transfer is not interrupted

3	2 words 2 ECS records	Transfer is not interrupted

For each condition, the test checks that all the data is transferred correctly, and whether the transfer is interrupted.

7 Tests that the REC instruction can be restarted after a page table search without find. The test checks that the page fault does not interrupt the transfer at other than UEM ECS record boundaries or between the last two records.

Condition	Transfer Sizes	Expected Result
0	271 words	Transfer is interrupted by invalid page in UEM.
1	15 words 2 ECS records	Transfer never starts because second and last UEM ECS record not in page table.
2	271 words	Transfer never starts because the number of words in CM are not enough to complete first UEM ECS record.
3	15 words 2 ECS records	Transfer never starts because the words needed to form the second and last UEM ECS record are not in CM.

For each condition, the test checks that the transfer is completed correctly when restarted and the appropriate page validated.

8-14 Subsections 8-14 are identical to subsections 1-7 with the exception that the Block Copy Flag is set. Therefore, X0 upper instead of A0 is used for the block transfer to CM. At the end of each condition, the following comparisons are made:

1) Compare the dumped P-register to the address of the instruction halting the processor.

2) Check that the dumped MCR is clear except for the page fault bit.

3) Check that the source and destination data locations are correct.

4) For subsection four, check that the location next to the destination data locations is not changed.

SECTION 6

This section tests the positive operation of the WEC (block write ECS, 012) instruction in UEM (ECS mode). Subsection 0 sets up processor registers and common data.

Subsection	Description
1	Tests the WEC instruction with different values in the J field (ie. B1 through B7 are used).

Condition

0	WEC B1+k
1	WEC B2+k
2	WEC B3+k
3	WEC B4+k
4	WEC B5+k
5	WEC B6+k
6	WEC B7+k

2 Tests the addressing of central memory and unified extended memory (ECS mode) by the WEC instruction. A pattern of ten addresses is used. Each of the addresses is used in combination with the other addresses including itself. Therefore, each address in the pattern is used to write UEM from every address in the pattern, now as addresses in CM.

Condition

0-9	WEC (B1)+k words from (adrs1-adrs10) to adrs1
10-19	WEC (B1)+k words from (adrs1-adrs10) to adrs2
20-29	WEC (B1)+k words from (adrs1-adrs10) to adrs3
30-39	WEC (B1)+k words from (adrs1-adrs10) to adrs4
40-49	WEC (B1)+k words from (adrs1-adrs10) to adrs5
50-59	WEC (B1)+k words from (adrs1-adrs10) to adrs6
60-69	WEC (B1)+k words from (adrs1-adrs10) to adrs7
70-79	WEC (B1)+k words from (adrs1-adrs10) to adrs8
80-89	WEC (B1)+k words from (adrs1-adrs10) to adrs9
90-99	WEC (B1)+k words from (adrs1-adrs10) to adrs10

The reference addresses, RAc and RAe, are both equal to zero. Therefore, adrs1 through adrs10 are equal to the addresses given in the description of Section 4, Subsection 2.

3 Tests the transmission of data patterns by the WEC instruction. The patterns used are:

Subsection	Description

Condition

0	data = octal 2525252525252525
1	data = octal 5252525252525252
2	data = octal 7070707070707070
3	data = octal 0707070707070707

4 Tests the summation of the B-register with the k-field by the WEC instruction. The various combinations used are:

Condition

	(Bj)	k	(Bj)+k	
0	2	-1	1	
	1	0	1	*this condition
1	0	1	1	tested in previous
2	-1	2	1	subsection

5 Tests the ability of the WEC instruction to copy blocks of varying sizes. Since transfers are divided into blocks with boundaries of (0 mod 8) the transfer sizes and addresses were chosen so as to require a maximum number of blocks for a minimum transfer size. The block sizes in octal are:

Condition

0	12
1	22
2	42
3	102
4	202
5	1,002
6	2,002
7	4,002
8	10,002
9	20,002

6 Tests the interruptible of the WEC instruction. It checks that transfers greater than two blocks can be interrupted, and that the interruption occurs at an ECS record boundary, but not the last one. It also checks that transfer sizes of two ECS records in length cannot be interrupted (ie. cannot interrupt between the last two records).

Condition	Transfer Sizes	Expected Result
0	256 words 32 ECS records minimum of 4 blocks	Transfer is interrupted at an ECS record boundary

Subsection		Description	
	1	58 words 9 ECS records minimum of 2 blocks	Transfer is interrupted at an ECS record boundary, but not the last one.
	2	15 words 2 ECS records	Transfer is not interrupted
	3	2 words 2 ECS records	Transfer is not interrupted

For each condition, the test checks that all the data is transferred correctly, and whether the transfer is interrupted.

7 Tests that the WEC instruction can be restarted after a page table search without find. The test checks that the page fault does not interrupt the transfer at other than UEM ECS record boundaries or between the last two records.

Condition	Transfer Sizes	Expected Result
0	271 words	Transfer is interrupted by invalid page in UEM.
1	15 words 2 ECS records	Transfer never starts because second and last UEM ECS record not in page table.
2	271 words	Transfer never starts because the number of words in CM are not enough to complete first UEM ECS record.
3	15 words 2 ECS records	Transfer never starts because the words needed to form the second and last UEM ECS record are not in CM.

For each condition, the test checks that the transfer is completed correctly when restarted and the appropriate page validated.

8-14 Subsections 8-14 are identical to subsections 1-7 with the exception that the Block Copy Flag is set. Therefore, XO upper instead of AO is used for the block transfer from CM. At the end of each condition, the following comparisons are made:

Subsection | Description

1) Compare the dumped P-register to the address of the instruction halting the processor.

2) Check that the dumped MCR is clear except for the page fault bit.

3) Check that the source and destination data locations are correct.

4) For subsection four, check that the location next to the destination data locations is not changed.

SECTION 7

This section tests the error exits of the CRXj and CWXj instructions. Subsection 0 sets up processor registers and common data.

1 Tests the address out of range exit on a CM read/write in 170 job mode. The error exit is handled by the 170 monitor, therefore, the EI is not used. The following table indicate the values set up in the various variables that affect the execution of the instruction.

Condition	Instr	(Xk)<FLc	AOR EM	Result
0	CRXj	no	set	AOR Exit to MA
1	CRXj	no	clear	Pass
2	CWXj	no	set	AOR Exit to MA
3	CWXj	no	clear	Pass

2 Tests the address out of range exit on a CM read/write in 170 monitor mode. The error exit is handled by the executive state monitor, therefore, the EI is used. The following table indicate the values set up in the various variables that affect the execution of the instruction.

Condition	Instr	(Xk)<FLc	AOR EM	Result
0	CRXj	no	set	AOR Exit to EI
1	CRXj	no	clear	Pass
2	CWXj	no	set	AOR Exit to EI
3	CWXj	no	clear	Pass

At the end of each condition, the following comparisons are made:

1) Compare the dumped P register to the address of the instruction halting the processor.

Subsection	Description

2) Check that the dumped MCR is clear except for the page fault bit or instruction specification error bit depending on what is used to halt the processor.

3) Check that the destination register or memory location is correct.

4) Check that the contents of RAC is correct.

5) If the EI is used, check that the job's P-register saved by EI is correct.

SECTION 8

This section tests the error exits of the RXj and WXj instructions. Subsection 0 sets up processor registers and common data.

1 Tests the address out of range exit on a UEM (ECS mode) read/write in 170 job mode. The error exit is handled by the 170 monitor, therefore, the EI is not used. The following table indicate the values set up in the various variables that affect the execution of the instruction.

Condition	Instr	UEM Set	ECS Pres	ESM Set	(Xk)<FLe	AOR EM	Expected Result
0	RXj	yes	no	no	no	set	AOR
1	RXj	yes	no	no	no	clear	PASS
2	WXj	yes	no	no	no	set	AOR
3	WXj	yes	no	no	no	clear	PASS

The expected results given in the table above are: AOR for address out of range error exit, and PASS for when the instruction is executed as a no-op.

2 Tests the address out of range exit on a UEM (ECS mode) read/write in 170 monitor mode. The error exit is handled by the executive state monitor, therefore, the EI is used. The following table indicate the values set up in the various variables that affect the execution of the instruction.

Condition	Instr	UEM Set	ECS Pres	ESM Set	(Xk)<FLe	AOR EM	Expected Result
0	RXj	yes	no	no	no	set	AOR
1	RXj	yes	no	no	no	clear	PASS
2	WXj	yes	no	no	no	set	AOR
3	WXj	yes	no	no	no	clear	PASS

Subsection	Description

At the end of each condition, the following comparisons are made:

1) Compare the dumped P register to the address of the instruction halting the processor.

2) Check that the dumped MCR is clear except for the page fault bit or instruction specification error bit depending on what is used to halt the processor.

3) Check that the destination register or memory location is correct.

4) Check that the contents of RAc is correct.

5) If the EI is used, check that the job's P-register saved by EI is correct.

SECTION 9

This section tests the error exits and half exits for the block copy instructions in UEM (ECS mode). Full exits, where execution continues normally, are tested in section 5 and 6.

1-4 Tests the error exits and half exits of the REC and WEC instruction in 170 job and monitor mode. Subsection 1 tests the REC instruction in 170 job mode, and subsection 2 tests the WEC instruction in 170 job mode. These error exits are handled by the 170 monitor, therefore, the EI is not used. Subsection 3 tests the REC instruction in 170 monitor mode, and subsection 4 tests the WEC instruction in 170 monitor mode. These error exits are handled by the executive state monitor, therefore, EI is used. The following table indicate the values set up in the various variables that affect the execution of the block copy instructions. For each condition, the variables are chosen such that there are a number of possible errors and exits. However, due to the order in which these variables are tested, there is only one correct exit and result. The underlined value indicates the particular variable test which causes the expected exit.

For example, in condition 01, the expected result is an address out of range error exit due to A0+Bj+k being greater than FLc. However, the variables are set up such that the instruction may pass due to a zero word data transfer if the variables are tested out of order.

Condition	Instr Parcel 0	UEM SET	ESM SET	ECS Pres	ECS Auth	A0+Bj+k <=FLc	Bj+k	X0+Bj+k <=FLe	AOR EM	X0 Bit 21, 22	Expected Result
0	no	yes	no	no	no	no	zero	no	set	clear	ILL
1	yes	yes	no	no	no	no	zero	yes	set	clear	AOR
2	yes	yes	no	no	no	no	zero	yes	clear	clear	PASS
3	yes	yes	no	no	no	yes	neg	yes	set	clear	AOR
4	yes	yes	no	no	no	yes	neg	yes	clear	clear	PASS
5	yes	yes	no	no	no	yes	zero	no	set	clear	AOR
6	yes	yes	no	no	no	yes	zero	no	set	clear	AOR
7	yes	yes	no	no	no	yes	zero	no	clear	clear	PASS
8	yes	yes	no	no	no	yes	zer	yes	set	set	PASS
9	yes	yes	no	no	no	yes	pos	yes	set	set	HALF

The expected results given in the table above are: ILL for illegal instruction error exit, AOR for address out of range error exit, PASS for when the instruction is executed as a no-op, and HALF for half exit.

Subsection	Description
5-8	Tests the REC and WEC instruction in 170 job and monitor mode with the Block Copy Flag set. With the exception of the Block Copy Flag, and the usage of X0 upper instead of A0 to address CM, these four subsections are identical to subsections 1-4.

At the end of each condition, the following comparisons are made:

1) Compare the dumped P register to the address of the instruction halting the processor.

2) Check that the dumped MCR is clear except for the page fault bit or instruction specification error bit depending on what is used to halt the processor.

3) Check that the destination register or memory location is correct.

4) Check that the contents of RAc is correct.

5) If the EI is used, check that the job's P-register saved by EI is correct.

SECTION 10

This section tests the illegal instruction, address out of range, indefinite, and infinite error exits. The processing of these exits by the hardware has been tested by the executive state exchange test. Therefore, only particular conditions that require processing by the EI are tested here. Those conditions are the various error exits in 170 monitor mode and the particular exit mode selected. Subsection 0 sets up processor registers and common data.

Subsection	Description
1	Tests the PS (00) instruction in parcel 0 to 3.

Condition

0	PS in parcel 0
1	PS in parcel 1
2	PS in parcel 2
3	PS in parcel 3

2 This subsection uses the SXi Xj+K instruction in parcel 3 to test for illegal instruction.

3 This subsection tests address out of range error exits.

Condition

0	RNI AOR
1	Branch AOR
2	AOR on a read CM
3	AOR on a write CM

4 Tests the infinite condition error exit.

Condition

0	FXi Xj+Xk	Xj is infinite
1	FXi Xj-Xk	Xj is infinite
2	FXi Xj*Xk	Xj is infinite
3	FXi Xj/Xk	Xj is infinite

5 Tests the indefinite condition error exit.

Condition

0	FXi Xj+Xk	Xj is indefinite
1	FXi Xj-Xk	Xj is indefinite
2	FXi Xj*Xk	Xj is indefinite
3	FXi Xj/Xk	Xj is indefinite

These error conditions cause an exchange to executive state mode, where the Environment Interface will set up the exit condition and P in location RAc, and halt the processor.

The following comparisons are then made:

1) Compare the dumped P-register to the address of the halt instruction in EI.

Subsection	Description

2) Compare the job's P register saved by EI to address of error instruction.

3) Compare the contents of RAc to the expected exit condition and P-register value.

4) Check that the dumped MCR is clear except for the instruction specification error bit due to the halt instruction.

SECTION 11

This section tests the operation of the RT (017_8) instruction. The test is done with and without the EI. Subsection 0 sets up processor registers and initializes common data.

1 Tests for a Privileged Instruction Fault interrupt when a RT instruction is encountered.

Condition	OPERATION TESTED	TRAPS	EXP RESULT
0	017 in A170 Job Mode	Traps Enabled	Trap
1	017 in A170 Monitor Mode	Traps Enabled	Trap
2	017 in A170 Job Mode	Traps Disabled	Exchange
3	017 in A170 Monitor Mode	Traps Disabled	Exchange

2 Tests the operation of the EI on an RT instruction. For an RT instruction in A170 job mode, the EI simulates an exchange jump to MA, and for an RT instruction in A170 Monitor Mode, the EI simulates a halt.

Condition	
0	017 in A170 job mode causes an exchange jump to MA.
1	017 in A170 monitor mode causes a halt.

For each condition, the test checks that the expected result occurs.

SECTION 12

This section tests the simulation of the CMU instruction by the EI. The positive operation of the CMU instructions are tested by the diagnostic, CT8. Only the interruptible and the error exits of the CMU instructions are tested here. Subsection 0 sets up processor registers and common data.

Subsection	Description
1-4	Tests the interruptible of the 464, 465, 466, and 467 instructions respectively. Each subsection tests that the CMU instruction is simulated correctly with and without PP interruptions. A PP exchange interrupts the CMU simulation at a certain point, where a second job checks that the CMU simulation has started but not finished. After the processor has returned to and finished the CMU simulation, the processor halts and the results are compared.

Condition	Job/Monitor Mode	PP Interrupt	Interrupt Taken
0	JOB	NONE	NO
1	MONITOR	NONE	NO
2	MONITOR	MAN	NO
3	JOB	MXN	YES
4	MONITOR	EXN	YES

5-8 Checks the error exits of the 464, 465, 466, and 467 instructions respectively. The error exits tested are described in the table below. Conditions 20-23 applies only to Subsection 5 and 7, where the move descriptor for the Indirect Move instruction and the collate table for the Compare Collate instruction is out of range.

Condition	Instr. Parcel 0	C1>9	C2>9	K1+L >=FLc	K2+L >=FLc	Bj+K>=FLc A0+7>=FLc	Len=0	Mode	AOR EM	Expected Result
0	no	yes	yes	yes	yes	no	yes	job	clear	ILL
1	no	yes	yes	yes	yes	no	yes	mon	clear	ILL
2	yes	yes	no	no	no	no	no	job	set	AOR
3	yes	yes	no	no	no	no	no	job	clear	PASS
4	yes	yes	no	no	no	no	no	mon	set	AOR
5	yes	yes	no	no	no	no	no	mon	clear	PASS
6	yes	no	yes	no	no	no	no	job	set	AOR
7	yes	no	yes	no	no	no	no	job	clear	PASS
8	yes	no	yes	no	no	no	no	mon	set	AOR
9	yes	no	yes	no	no	no	no	mon	clear	PASS
10	yes	no	no	yes	no	no	no	job	set	AOR
11	yes	no	no	yes	no	no	no	job	clear	PASS
12	yes	no	no	yes	no	no	no	mon	set	AOR
13	yes	no	no	yes	no	no	no	mon	clear	PASS
14	yes	no	no	no	yes	no	no	job	set	AOR
15	yes	no	no	no	yes	no	no	job	clear	PASS
16	yes	no	no	no	yes	no	no	mon	set	AOR
17	yes	no	no	no	yes	no	no	mon	clear	PASS
18	yes	no	no	no	no	no	yes	job	clear	PASS
19	yes	no	no	no	no	no	yes	mon	clear	PASS
20	yes	no	no	no	no	yes	no	job	set	AOR
21	yes	no	no	no	no	yes	no	job	clear	PASS
22	yes	no	no	no	no	yes	no	mon	set	AOR
23	yes	no	no	no	no	yes	no	mon	clear	PASS

At the end of each condition, the following comparisons are made:

1) If data is moved, check that the destination locations are correct.

2) Compare the dumped P register to the address of the instruction halting the processor.

3) Check that the dumped MCR is clear except for the page fault bit.

4) Check that the Job's P-register is correct.

5) Check that the Job's X0 register is correct.

6) Check that the contents of RAc is correct.

SECTION II-6

EXCH - EXCHANGE TEST

PROGRAM DESCRIPTION

GENERAL

This section describes the mechanism by which EXCHITC maintains control over EXCH. The test is assembled on the executive state virtual machine assembler and, during execution, resides in CM. EXCHITC is assembled on the 16-bit Compass assembler and resides in a PP.

All the information required by EXCHITC to control EXCH is contained within the object code of EXCH and is read from CM by EXCHITC. This information contains a series of control commands that cause the PP to execute memory writes and compares and other types of operations which affect the CP. The source code of EXCH contains a set of procedures which provides a convenient method of generating the control commands.

EXCHITC contains a number of subroutines, one for each different control command, which perform the task required by a command according to its specific address and data operands.

Structure

The data contained within the object code of EXCH is composed primarily of the following:

CP Executable Code

This is the virtual machine code executed by the CP. It consists of both executive state and CYBER 170 commands. Generally the code sequences are short and are followed by a HALT or PS (CYBER 170 program stop) command. EXCHITC causes CP execution to commence at the first command of the test sequence. When the HALT command is detected by EXCHITC, it causes the contents of the machine registers to be dumped to CM. EXCHITC will subsequently perform all comparison checks required by EXCH using the data dumped to CM or other data in CM.

Exchange Packages

Appendix C contains a set of diagrams that illustrate the sequence of exchanges that occur in the test sections.

A single executive state monitor exchange package is used for all sections of the test. Three different executive state job exchange packages and eight CYBER 170 exchange packages are shared between various sections. In most cases, the executive state monitor process will generate a working copy of a job exchange package by copying from an original. In the first subsection of

a section, control commands executed by EXCHITC initialize the original package with values specific to the section. Control commands are provided so EXCHITC may set up the MPS exchange package pointer register. Executive state monitor process sets up the JPS pointer register using a CPYXS command.

Section and Subsection Address Tables

These tables are compiled into EXCH for use by EXCHITC. The tables consist of a series of addresses, one for each section/ subsection of EXCH. Each table entry is an executive state word containing a right justified real memory byte address pointing to a section/subsection of the test. The table entries are ordered according to the section/subsection. The last entry in the table is a word of all ones which indicates the end of the section/test. The section table must reside in CM commencing at byte address 4000_{16}. The entries in the section table point to the subsection address table for the section while the subsection address table entries point to the control command tables for the subsection.

Control Command Tables

Control command tables constitute the bulk of the object code of EXCH. They contain the control commands which are interpreted and acted upon by EXCHITC. The tables are divided into subsections commencing at the CM addresses pointed to by the entries of the previously described subsection address table. The subsections are in turn divided into segments; a single segment consists of the control commands for a single condition of the test. The first segment (condition) of a subsection commences at the beginning of a subsection control table. The first 16-bit word of every segment contains the segment length. This length is a nonzero count of the number of executive state words in the segment, including the segment length word itself.

Each segment commences on a executive state word boundary. There should normally be only 0, 1, 2, or 3 unused 16-bit words between segments. EXCHITC uses the segment length to determine the end of a condition and the CM address of the segment for the next condition. The segments (conditions) of EXCH are not individually addressable (as is the case for the sections) and therefore they must be executed sequentially.

Execution Sequence

To execute a single condition of EXCH, EXCHITC references the section and subsection address tables to obtain a pointer to the first condition of a section. EXCHITC then transfers the complete control table for a condition from CM into its own PP memory. Only when this condition is complete will the segment for the next condition be obtained. The segment is not transferred repeatedly for repeat condition or scope mode operation.

EXCHITC saves the segment length for future use and then proceeds to interpret each of the control commands in the current segment. After each control command is executed a check is performed using the segment length to determine if all the commands in the segment have been executed.

Setting Up The Test Condition

In a typical condition of EXCH, control commands are used to write data in various locations in memory. For example, to enter a data pattern in an X register of an exchange package, or to zero out a memory cell in which the function to be tested will subsequently enter data. One or several WRITE_CM commands may be executed in order to set up the required test conditions.

A number of WRITE_CM commands are used to define the operation to be performed by each individual slave PP. One of these selects the command for each slave: EXN, MXN, MAN, INPN, CWM, CWML, or PSN. From 1 to 4 WRITE_CM command specifies the exchange address; in most cases, the same address is used for all four slaves. Finally a single WRITE_CM specifies both the number of times the selected command is to be executed and also a delay count. The delay count causes a small variable delay prior to the execution of the selected command. A different delay may be specified for each slave.

Exchange Sequence

The exchange sequence usually consists of two EXCHITC commands, XENTRY and XEXIT. XENTRY's initial task is to trigger the slaves to read the previously defined control data and prepare them for the required operation. Subsequently, XENTRY sets a deadstart flag, reloads the MPS, JPS, PSM, PTL, PTA, and SIT with their previously defined data, and deadstarts the CP. The CP then enters an executive state monitor process which usually includes some form of initialization affecting the job process. When this initialization is complete, the CP clears the deadstart flag. EXCHITC senses this and then allows sufficient time for the CP to switch to executive state job mode after which it triggers the slaves to commence their exchange commands.

Normally, EXCHITC then executes the XEXIT command to scan all the slave PPs and the CP to determine if they have completed their operation. If each slave channel is active, it has completed its operation and is ready. The status summary of the CP is checked to determine if it is halted. EXCHITC repeats the scan loop approximately 300 times (determined by PARAM 16 - Processor Hang Count) before declaring that one or more of the the processors is hung. If all processors are ready, EXCHITC exits the XEXIT command and data checking commences. Otherwise, EXCHITC repeatedly deadstarts the CP at a special microcode address which forces an exchange accept to be sent to the PP system. The deadstart is executed once for each active PP (one or more of which could be hung).

During the XEXIT sequence the S and P registers of the CP are read for later display. These registers are read before the force exchange-accept deadstart (if any) and the deadstart which causes the register file dump.

Data Checking

One or more compare (COMP or COMP_IMM) control commands will be employed to check results. Comparisons of data in exchange packages or varaious memory locations may be performed. Frequently, the exchange count read back to central memory from the slave PPs will also be checked. An error display will occur for each comparison error.

Repeat Condition and Scope Loops

If the operator calls for a condition to be repeated, all the control commands within the segment (writes, exchange, and compares) are repeated. If a scope loop is set up by the operator, only the exchange entry/exit sequence is repeated on each pass of the scope loop. This is necessary to provide a high repetition rate scope display. Critical data is automatically refreshed on each loop. During a scope loop where a processor is hung, EXCHITC will cause the exchange operation to be repeated whenever the processor hang count becomes zero. The counter is initialized by loading it from PARAM16. If desired, PARAM16 can be altered.

Slave PP Control

EXCHITC employs the slave PPs principally for the purpose of executing 26XX PP exchange commands. A separate PP must be employed for this purpose because the PP which executes the 26XX command cannot free itself from a hung condition if the CP fails to respond with an exchange accept. As well as executing the 2600(EXN), 2610(MXN), and 2620(MAN) type exchange commands, section 10 of the test requires that a slave execute a 102601(INPN) command which sends an external interrupt to the CP via its central memory port. Section 8 uses the CWM and CWML commands to write data to CM while other PPs are executing exchanges. For any condition where a slave is not required to be active, it executes its normal loop of commands. However, a 2400(PSN) command is substituted for the 2600 type.

Slave Triggering Via PP Channels

The slaves are controlled via PP channels and by central memory. The channels are used as interlocking control flags to trigger an operation at the request of EXCHITC. A communication channel selected under the control of PARAM 15 is used by EXCHITC to trigger all slaves simultaneously. Each slave uses its own channel to signal a response to EXCHITC.

All five channels are normally active. When EXCHITC wishes to trigger an operation, it checks that all slave PP channels are active; it then disconnects its communication channel and checks that all slaves have responded by disconnecting their channel. When this response is received, EXCHITC reactivates its communication channel.

When EXCHITC's communication channel is again active, the slaves immediately proceed with their specified task. When this is complete, they activate their channel to complete the cycle and wait for EXCHITC to trigger another operation by disconnecting its communication channel.

If either the CP or a slave PP hangs up (slave PP channel inactive), EXCHITC outputs a word on the communication channel (sets the channel full) and then forces the CP to send exchange accepts. This allows the slave (or slaves) to exit the 26XX command. When the communication channel is full, the slaves immediately return to their idle routine and bypass decrementation of their exchange count. EXCHITC retrieves the decremented exchange count from the slaves on each condition of the test. The count should normally be decremented to zero from some preset nonzero value.

Communication Via Central Memory

During each condition of a test, EXCHITC triggers the slave PPs causing them to read control data from central memory. EXCHITC will have set up this data prior to triggering the slaves. The data defines the type of operation to be performed, the exchange address, etc. After the slaves have read the data and prepared themselves to execute the condition, they wait to receive another trigger from EXCHITC. Simultaneously, EXCHITC is deadstarting the CP. When its initialization is complete and time has been allowed for a possible exchange to executive state job mode, EXCHITC triggers the slaves to proceed with their exchange sequence.

Assuming that an exchange command is accepted by a slave, it will decrement its exchange count and, if specified by the previously received control data, it will check that the exchange caused B0 in the CYBER 170 exchange package to be zeroed out. The slave will then wait for the next trigger action in which EXCHITC will request the slave to copy its exchange count and B0 check data into central memory where it can be checked.

With the exception of checking B0 in the CYBER 170 exchange package in sections 7 and 9, the slave PPs do not communicate with CM for the purpose of reading or writing control information while the CP is active (executing instruction level code). Similarly the EXCHITC PP obtains its control information prior to deadstarting the processor and only accesses CM to sense the status of the deadstart flag when processor execution commences and in sections 9 and 11 to access a sequence complete flag. This organization minimizes the possibility that conflicts which arise during the actual test sequence will cause a loss of control.

Synchronized Versus Free-Running Control

In some sections of the test, the slaves are each required to execute more than one 26XX exchange command for each condition. The synchronized mode allows a slave to execute only a single 26XX command per trigger operation. EXCHITC checks that each slave completes the operation and is waiting to be triggered before the next trigger is transmitted.

In free-running mode, the slaves execute their allotment of exchange commands at their own speed having received only a single trigger from EXCHITC. In many conditions of the test, the slaves are programmed to execute a small delay between exchanges. This delay can vary for each slave. The free-run mode is used in the later conditions of test sections 7, 8, and 11.

SECTION DESCRIPTIONS

The following portions of this document provide a description of each of the sections of the test. There are 19 sections altogether. Excepting section 0, each section of the test contains two or more subsections. The first (subsection 0), always performs initialization peculiar to the section, and in most cases, it performs no comparison checks.

Section 0 of EXCH is used solely for the purpose of performing initialization related to virtual addressing. Section 1 checks the operation of the CYBER 170 PS (program stop) and the CEJ/MEJ commands. Sections 2 through 6, and 16 employ only a single slave PP. Sections 2 and 16 check the ability of the 26XX commands to properly address central memory. Sections 3 through 6 are related to checking the CP's response when any one of three different 26XX commands are received when the processor is in either the 60-bit or 64-bit job or monitor modes.

Sections 7 through 11 use multiple slave PPs. Their purpose is to detect failures which result from either simultaneous exchanges or the sensitivity of the system to the asynchronous nature of PP exchanges or other events.

Sections 12 through 15 are employed to check CYBER 170 error exits resulting from illegal commands, address out of range, floating point infinite, and floating point indefinite errors. Section 18 checks the state switching mechanism using the CYBER 170 X0 register sign bit. PP exchanges are not used in sections 12 through 15 and 18.

The data for each section includes the purpose and overall operation of the section and a discussion of the differences between the subsections and conditions of the section. Each condition of a subsection may make one or more comparisons. In most cases, the corresponding comparisons in all conditions of a subsection are checking the same function or data item. A comparison number, CMP.XX, and a brief description is given for each comparison that is made. The number corresponds to the COMPARE NO. on the test error display. Numbering of comparisons commences with number 1 in each new condition.

CYBER 170 Program Stop Sequence

The CYBER 170 instruction level command sequences in this test usually execute a PS (program stop) command upon completion. This, however, does not stop the processor. Instead , the processor returns to the executive state monitor mode where it will execute a executive state halt command to halt instruction execution. A CYBER 170 process will not always return directly to the executive state monitor; the action taken is dependent upon the state of the CYBER 170 monitor flag when the PS command is executed.

If the monitor flag is set (monitor mode), the executive state exchange interrupt to executive state monitor mode will occur directly with the CYBER 170 environment left in the executive state job package in memory.

If the CYBER 170 monitor flag is clear (job mode), the processor will first perform a CYBER 170 exchange to MA using the CYBER 170 exchange package referenced by the MA register. This exchange will cause the processor to revert to the CYBER 170 monitor mode with the flag set. In this latter case, the new CYBER 170 process will also execute a PS command. Since the flag is now set, the sequence will continue as for the former case: the executive state exchange interrupt will occur and the processor will halt in the executive state monitor mode.

In both cases, the action taken is a result of executing the PS command which is an illegal CYBER 170 command. The action which has been described is a CYBER 170 error exit sequence. A part of this error exit sequence includes storing the CYBER 170 P register in RA. It should be noted that for the second case described above, the storage of P into RA will occur twice. Both CYBER 170 processes use the same RA. The CYBER 170 process which is executed as a result of the second PS command saves the previous entry in RA for possible later examination.

CYBER 170 X Register Shift Sequence

Sections 8 through 11 of the test require that the processor be prepared to accept PP 2600 exchanges and other interrupts. Rather than simply placing the processor in an idle loop , it executes a CYBER 170 process which can be checked for proper operation at the completion of a condition. Registers X1 through X7 start with a true bit in the most significant bit position (X0 starts with a false bit). A program loop is executed 59 times which shifts each register right one place during each loop. The registers fill with ones (X0 fills with zeroes).

Within the loop there are six jump commands which occur on various parcel boundaries. There are also two CYBER 170 CP exchange jumps (CEJ/MEJ, op code 013_8) in the loop. When one of these commands is executed, the current CYBER 170 process swaps to the identical CYBER 170 code sequence. Thus, there are two identical CYBER 170 processes running; the CEJ/MEJ commands cause a continual swap back and forth allowing each the opportunity to shift its X registers.

On each excursion of the loop, both processes increment and test their B2 register. When the current process has completed 59 loops it exits the loop and checks the B2 value from the other process. If that process is not complete, it gives control to the other process via another CEJ/MEJ exchange jump. If the other process was complete, then the current process executes a PS command (program stop, see above) or sets a flag to notify EXCHITC.

Note that, in sections 8 through 11, whenever a CEJ or an MEJ command is executed, it always causes a swap to the same CYBER 170 exchange package. Additionally, whenever a slave PP executes a 26XX exchange command its A register or MA also points to the same CYBER 170 exchange package.

Upon completion of the condition, the test checks the X registers in both processes to ensure they are all ones. Since control is returned to the executive state monitor process, one set of X registers should be seen in the executive state job exchange package while the other set is in the CYBER 170 exchange pack. The CYBER 170 process seen in the executive state job package would be the last of the two CYBER 170 processes to execute and the CYBER 170 monitor flag should be set.

Section 0 - Test Initialization (SECT 0)

This section performs initialization of the processor virtual environment for all subsequent sections of the test. This initialization includes the loading and checking of the PSM, PTL, and PTA registers and the writing into CM of the page and segment tables at their appropriate locations.

The page table maps virtual addresses one to one into the real memory address space for a complete 16 megabyte memory. The page size is 64K bytes. The page table is located at 2000_{16} (byte address) and is 8K (2000_{16}) bytes long.

If a comparison error occurs, check to ensure that the microcode has been loaded. If it has been loaded, other tests should be performed to check for proper operation of the PSM, PTL, and PTA before proceeding further with this test.

CMP.01 Checks that the PSM register was loaded properly (see 4.4.4)

CMP.02 Checks that the PTL register was loaded properly (see 4.4.4)

CMP.03 Checks that the PTA register was loaded properly (see 4.4.4)

Section 1 CEJ/MEJ Commands (SECT 1)

This section checks the program swap mechanism employed by the CEJ/MEJ commands and also used during an error exit caused a PS command. The 26XX PP exchange commands are not used.

Subsection 0 is employed for initialization. Subsection 1 has four conditions which are defined as follows:

Cond 0 - Execute Program Stop in CYBER 170 monitor mode
Cond 1 - Execute Program Stop in CYBER 170 job mode
Cond 2 - Execute CEJ command in CYBER 170 monitor mode
Cond 3 - Execute MEJ command in CYBER 170 job mode

After the initial deadstart, the executive state monitor process sets up two CYBER 170 exchange packages: the "exchange to A" package and the "exchange to MA" package. The processor then executes a executive state exchange to job mode. The job package has a VMID equal to one, and therefore the processor will be in CYBER 170 mode.

Condition 0: CYBER 170 monitor flag is set (MF is equal to one), and the processor is pointed to a command sequence which enters $AAAA_{16}$ into the "exchange to A" flag in CM and then executes a PS command. This illegal command should immediately cause an error exit resulting in a executive state exchange back to executive state monitor mode where the processor will stop on a HALT command.

In condition 1, the processor initially enters CYBER 170 job (MF equal to 0) mode instead of monitor mode; it sets the value $AAAA_{16}$ as for condition 0, but the error exit due to the PS command executed in job mode causes an exchange to MA and the setting of CYBER 170 monitor flag. This exchange to MA points to a sequence which sets the exchange to MA flag to $DEAD_{16}$ and then executes another PS command. This causes another error exit via an exchange to executive state monitor mode.

In conditions 2 and 3, the process executes a CEJ/MEJ after entering the CYBER 170 mode. If MF is equal to 1 (condition 2) the commmand is interpreted as a CEJ command where K points to the exchange to A package. If MF is equal to 0

(condition 3) it is interpreted as an MEJ where the current MA points to the exchange to MA package.

In either case, an exchange swap occurs and the CYBER 170 monitor flag changes state. For condition 3, the resulting process sets up the exchange to MA flag, executes the PS command, and, because MF is now set, the error exit causes the executive state exchange to executive state monitor mode.

For condition 2, the CEJ command first swaps the exchange to A package and sets MF equal to 0. The new process sets up the exchange to A flag, sets MF equal to 0, and then executes the PS command. With MF equal to 0, the PS causes an exchange to MA. This exchange to MA points to a sequence which sets the exchange to MA flag to $DEAD_{16}$ and then executes another PS command. This causes another error exit via an exchange to executive state monitor mode.

CMP.01 The P value in the exchange to A package is checked. A swap of this package, which only occurs condition 2, alters the value.

CMP.02 The P value in the exchange to MA package is checked. It is swapped in conditions 1, 2, and 3. In conditions 1 and 2 the execution of a PS command clears P to 0 before the swap out occurs.

CMP.03 The value of P entered into RA is checked. It shows the location of the last PS command to be executed before the processor returns to the executive state monitor mode.

CMP.04 The P value in the executive state job package is checked. It should point to the same address as for comparison 3 above.

CMP.05 The P value in the register file dump is checked. It should point to the executive state HALT command where the processor stops after return to the executive state monitor mode.

CMP.06 The CYBER 170 monitor flag in the executive state job package is checked. It should always be set after the exit from CYBER 170 mode.

CMP.07 The exit mode halt flag in the executive state job package is checked; it should also be set on completion of all 4 conditions.

CMP.08 The exchange to A flag should be $AAAA_{16}$ for conditions 0, 1 and, 2.

CMP.09 The exchange to MA flag should be $DEAD_{16}$ for conditions 1, 2 and, 3.

Section 2 - Exchange Addressing (SECT 2)

This section checks the operation of a PP 2600 command and its addressing of CM by referencing the command to exchange packages located in many different locations in CM. Each condition of the test uses a different CM address for the location of the exchange package. These exchange or test addresses are essentially patterns of sliding ones or sliding zeroes. Address bits 23 through 2^{19} are tested. The highest address is less than 1 megabyte and the lowest is greater than 64K bytes. The test is repeated for each selected one megabyte segment of memory commencing at the lowest selected segment. The individual bits of PARAM 18 determine which of 16 one megabyte segments will be tested. This parameter is defaulted to test the first megabyte only.

Specifications for executive state indicate that the exchange address passed to the CP by the PP may be limited to 18 bits. Some processors may therefore fail to pass this test if PARAM18 is set for the 3rd megabyte or higher.

After the initial deadstart, the executive state monitor process sets up the P, RA, FL, and MA of 2 CYBER 170 exchange packages. One package is the "exchange to A" package which is set up at the test address pointed to by the 2600 command. The second package is the "exchange to MA" package.

After the the CYBER 170 packages are set up, the processor performs a executive state EXCHANGE command. The executive state job process which is then loaded has a VMID equal to one and therefore the processor should now also be in the CYBER 170 mode. Additionally, the CYBER 170 monitor flag is clear, and MA points to the "exchange to MA" package. The processor enters a countdown loop to wait for the PP to execute the 2600 command. If a swap does not occur a PS (program stop) command will be executed when the countdown is complete.

The P values set up in both of the CYBER 170 packages each point to a different CYBER 170 command sequence. Regardless of which package is selected for the swap, a unique flag is set up in memory and then a program stop occurs at a unique address.

Subsection 0 is for initialization only. The successive conditions of subsection 1 perform the following checks for each test address.

CMP.01 Checks the P value in the exchange package at the test address. Since the 2600 command causes a package swap, this P should point to the countdown loop where the processor was awaiting the arrival of the 2600.

CMP.02 Checks that the exchange to A flag contains $AAAA_{16}$ as a result of the execution of the process swapped in by the PP 2600 command.

CMP.03 Checks that the exchange to MA flag contains $DEAD_{16}$ as a result of executing a program stop in CYBER 170 job mode and then swapping to a CYBER 170 monitor process.

CMP.04 Checks that the P in the RF dump is at the HALT command in the executive state monitor process. A program stop (PS) in CYBER 170 monitor mode causes a executive state exchange to executive state monitor.

CMP.05 Checks that the CYBER 170 monitor flag in the executive state job package is set, thereby confirming the swap to monitor mode caused by the program stop in the CYBER 170 job mode.

CMP.06 This is a block comparison that checks the CYBER 170 A, B, and X registers in the CYBER 170 package at the test address. The values which are there are as a result of the swap-out caused by the 2600 command. These values were loaded into the processor registers when it initially entered the CYBER 170 mode after the executive state EXCHANGE command.

Section 3 - 26XX Execution In CYBER 170 Job And Monitor Mode (SECT 3)

This section checks the operation of the 2600(EXN), 2610 (MXN), and 2620 (MAN) commands while the processor is in the CYBER 170 mode with the CYBER 170 monitor flag clear and set. There are 6 conditions in subsection 1; these are defined by the following table:

COND 0 - 2600, CYBER 170 Job mode, Swap Per A, MF No Change
COND 1 - 2610, CYBER 170 Job mode, Swap Per A, MF Is Set
COND 2 - 2620, CYBER 170 Job mode, Swap Per MA, MF Is Set
COND 3 - 2600, CYBER 170 Mon mode, Swap Per A, MF No Change
COND 4 - 2610, CYBER 170 Mon mode, No Swap, MF No Change
COND 5 - 2620, CYBER 170 Mon mode, No Swap, MF No Change

Each condition of this section is set up and operates in a manner similar to that of section 2, except that the CM address for the "exchange to A package" is within the test's address space rather than being determined by a sliding pattern.

After the initial deadstart, the executive state monitor process sets up the P, RA, FL, and MA of 2 CYBER 170 exchange packages. One package is the "exchange to A" package which is pointed to by the PPs A register during the 26XX command. The second package is the "exchange to MA" package.

After the CYBER 170 packages are set up, the processor performs a executive state EXCHANGE command. The executive state job process which is then loaded has a VMID equal to 1 and therefore the processor should now also be in the CYBER 170 mode. The CYBER 170 monitor flag will already be set or clear according to the current condition and MA points to the "exchange to MA" package. The processor enters a countdown loop to wait for the PP to execute the 26XX command. If a swap does not occur a PS (program stop) command will be executed when the countdown is complete.

In conditions 0 and 1 a swap of the "exchange to A" package should occur. If, in condition 2, the processor correctly interprets the 2620 command and executes an exchange to MA, the address on the cache invalidation bus (from the PPs A register) is ignored and the current contents of MA (which point to the "exchange to MA package") are used instead. In condition 3 the 2600 command should again cause a swap with the "exchange to A package". For conditions 4 and 5 (the 2610 and 2620 respectively), no swap should occur and the CP should complete the countdown and execute the PS command.

For those conditions where the 26XX command causes a swap (conditions 0 through 3), the processor will then execute code which will set up a flag in memory for later checking (see comparisons 8 and 9 below). In all cases, a PS command will be executed which will cause the processor to return to the executive state monitor mode and HALT.

CMP.01 The P value in the CYBER 170 "exchange to A" package is checked. If the package was swapped, the P will point to the count-down loop where the processor awaited the execution of the 26XX command. If the swap does not occur, the original value (pointing to the "exchange to A" process) should still be there. The "exchange to A" swap should always occur for the 2600 command and also for the 2610 when the response is from the CYBER 170 job mode (conditions 0, 1, and 3).

CMP.02 This compare is similar to CMP.03 above, except that the CYBER 170 "exchange to MA" package is checked. The swap should only occur to this package for the 2620 command when the response is from the CYBER 170 job mode (condition 2). In this case the P value should point to the countdown loop. For conditions 1, 3, 4, and 5 it should point to the "exchange to MA" process. For condition 0 it should be 0. The first program stop which causes the exchange to MA, clears P before the swap occurs.

CMP.03 Checks the value entered in RA as a result of the last error exit. It should point to a PS command at one of 3 different locations, dependent upon the current condition.

CMP.04 Checks the P value in the executive state job exchange package. They should be identical to the addresses entered in RA.

CMP.05 Checks the value of P in the RF dump. It should be at the HALT in the executive state monitor process to which the processor returns after the CYBER 170 process is complete.

CMP.06 Checks the CYBER 170 monitor flag in the executive state job package. It would be set at the time of last PS command which causes the exchange to executive state monitor.

CMP.07 Checks that the CYBER 170 exit mode halt flag in the executive state job package is set.

CMP.08 The status of the exchange to A flag is checked. It should be $AAAA_{16}$ for conditions 0, 1, and 3 only.

CMP.09 The status of the exchange to MA flag is checked. It should be $DEAD_{16}$ for conditions 0 and 2 only.

Section 4 - 26XX Sets MCR 5 In Executive State Job Mode (SECT 4)

This section checks the operation of the 2600(EXN), 2610 (MXN), and 2620 (MAN) commands while the processor is in the executive state job mode with VMID equal to 0. There are 3 conditions in the section, one for each of the commands (sequenced as above). The section operates with the MCR mask clear and the traps disabled. The purpose is to ensure that each one of the 26XX commands will cause the exchange request bit of the MCR register to set.

After the initial deadstart, the executive state monitor process does a executive state EXCHANGE to a executive state job process (VMID equal to 0). In this process, the CP continuously tests the status of the exchange request bit in the MCR using a BRCR command. When the 26XX command occurs the CP executes halt command.

CMP.01 Checks the value of P in the RF dump. It should point to the halt command which is executed after the exchange request MCR bit is set.

CMP.02 Checks the exchange request bit in the MCR register field of the RF dump. The bit should be set.

Section 5 - 26XX Execution In Executive State Job Mode (SECT 5)

This section checks the capability of the system to respond to an exchange interrupt resulting from a 26XX command executed while the processor is in the executive state Job mode. The section consists of six conditions; the first three switch to the CYBER 170 job mode to allow an exchange accept response to the slave PP. The last three switch to the CYBER 170 monitor mode. The conditions are as follows:

COND 0 - 2600, Response in CYBER 170 Job, Swap Per A, MF No Change
COND 1 - 2610, Response in CYBER 170 Job, Swap Per A, MF Is Set
COND 2 - 2620, Response in CYBER 170 Job, Swap Per MA, MF Is Set
COND 3 - 2600, Response in CYBER 170 Mon, Swap Per A, MF No Change
COND 4 - 2610, Response in CYBER 170 Mon, No Swap, MF No Change
COND 5 - 2620, Response in CYBER 170 Mon, No Swap, MF No Change

The sequence of events for all conditions is very similar and is described in the following steps:

a) After the initial deadstart, the executive state monitor process performs some initialization and then switches to the executive state job mode via an EXCHANGE.

b) In the executive state job mode, the VMID is 0, the exchange request MCR mask bit is set and the traps are disabled. The P points to executive state BRXEQ command which performs an unconditional jump to itself. The processor is now prepared to receive the exchange interrupt.

c) When the 26XX command is executed by the slave PP, the processor interrupts back to the executive state monitor mode where the P value in 2 CYBER 170 exchange packages are set up. These are an "exchange to A" and an "exchange to MA" package. Additionally, a CPYXS command is executed which sets the JPS pointer for a new executive state job process exchange package.

d) Subsequent execution of a executive state EXCHANGE now loads this new executive state job process which has a VMID of 1 and thus the mode is simultaneously also CYBER 170. This process also loads the CYBER 170 monitor flag which will be set (for job or monitor mode) according to the list of conditions above. The P of this process points to a string of NO (no operation) commands followed by a PS (program stop).

e) Once in the CYBER 170 mode, the processor should (for condition 0 through 3) immediately respond to the 26XX command by performing the exchange swap and sending an exchange accept to the PP. Conditions 4 and 5 should fall through the NO commands and execute the PS.

f) Whether or not a swap occurs and which of the 2 CYBER 170 packages are selected for the swap is dependent upon whether the process is now in the CYBER 170 monitor or job mode and which of the 26XX commands is executed. Whether the CYBER 170 monitor flag changes state is dependent upon both of these factors.

g) For conditions 4 and 5 the program stop referred to in step e) above will cause an immediate executive state exchange to executive state monitor mode.

h) For conditions 0 through 3, either the CYBER 170 "exchange to A", or the "exchange to MA" package is selected and will determine which of 2 short CYBER 170 programs are executed. In both cases, a unique flag is set up in CM ($AAAA_{16}$ in the exchange to A flag or $DEAD_{16}$ in the exchange to MA flag) and a program stop (CYBER 170 PS command) occurs at a unique address.

i) For conditions 1 through 3 the PS command will now cause a return to executive state monitor mode.

j) For condition 0, the swap caused by the 2600 command is an exchange to A and leaves the CYBER 170 monitor flag clear. The program stop in the resulting exchange to A process causes a second swap (an exchange to MA) which sets the CYBER 170 monitor flag. Finally a second program stop occurs causing the return to executive state monitor mode. Note that in this case, both the exchange to A and exchange to MA flags will be set up.

k) For all conditions, the return to executive state monitor mode results in the execution of a HALT command. The HALT causes a register file dump. If scope mode operation is selected, the process is deadstarted again and the loop is repeated starting at step a) above.

CMP.01 The P value in the executive state monitor package is checked. The P should point to the command following the second EXCHANGE which was executed by the executive state monitor process (see step d) above)

CMP.02 The P value in the executive state job package is checked. It should point to the BRXEQ command where the executive state job process awaits the arrival of the 26XX command (see step b) above).

CMP.03 The P value in the CYBER 170 "exchange to A" package is checked. If the package was swapped, the P will point to the NO command where the CYBER 170 process commences (see steps d) and e) above). If the swap does not occur, the original value (pointing to the "exchange to A" process) should still be there. The swap would occur for the 2600 command and also for the 2610 when the response is from the CYBER 170 job mode (conditions 0, 1, and 3).

CMP.04 This compare is similar to CMP.03 above except that the CYBER 170 "exchange to MA" package is checked. The swap should only occur to this package for the 2620 command when the response is from the CYBER 170 job mode (condition 2). In this case the P value should point to the NO command (see steps d) and e) above). For conditions 1, 3, 4, and 5 it should point to the "exchange to MA" process. For condition 0 it should be 0. The first program stop which causes the exchange to MA (see step j) above) clears P before the swap occurs.

CMP.05 This comparison checks the P value in the register file dump. In all cases (conditions 0 through 5), it should point to the HALT in the executive state monitor process.

CMP.06 The status of the exchange to A flag is checked. It should be $AAAA_{16}$ for conditions 0, 1, and 3 only.

CMP.07 The status of the exchange to MA flag is checked. It should be $DEAD_{16}$ for conditions 0 and 2 only.

Section 6 - 26XX Execution In Executive State Monitor Mode (SECT 6)

This section is similar in many details to section 5; it checks the capability of the system to respond to an exchange interrupt resulting from a 26XX command executed while the processor is in the executive state monitor mode. The section consists of 6 conditions; the first 3 switch to the CYBER 170 job mode to allow an exchange accept response to the slave PP. The last 3 switch to the CYBER 170 monitor mode. The conditions are as follows:

COND 0 - 2600, Response in CYBER 170 Job, Swap Per A, MF No Change
COND 1 - 2610, Response in CYBER 170 Job, Swap Per A, MF Is Set
COND 2 - 2620, Response in CYBER 170 Job, Swap Per MA, MF Is Set
COND 3 - 2600, Response in CYBER 170 Mon, Swap Per A, MF No Change
COND 4 - 2610, Response in CYBER 170 Mon, No Swap, MF No Change
COND 5 - 2620, Response in CYBER 170 Mon, No Swap, MF No Change

The sequence of events for all conditions is very similar, and is described in the following steps:

a) After the initial deadstart, the executive state monitor process initializes its own A0 through A4 registers in preparation for a trap interrupt. The P values of the CYBER 170 "exchange to A" and "exchange to MA" packages are initialized and traps are enabled. The process clears the deadstart flag in CM (to signal EXCHITC that initialization is complete) and then executes a executive state BRXEQ command which performs an unconditional jump to itself. The processor is now prepared to receive the trap interrupt.

b) When the 26XX command is executed by the slave PP, a trap interrupt occurs and the current process registers are copied into a stack frame save area. A code base pointer is picked up which points to a new process (still executive state monitor mode). One of the effects of this trap is to update the value in register A0 which is the Dynamic Space Pointer. The new process will save the contents of A0 for a later comparison.

c) The new process will also perform a CPYXS to set the JPS pointer for the executive state job process and subsequently will switch to this mode by executing a executive state EXCHANGE command.

d) The new executive state job process has a VMID of 1 and therefore the machine is also in the CYBER 170 mode. The CYBER 170 monitor flag is loaded and will be set (for job or monitor mode) according to the list of conditions above. The P of this process points to a string of NO (no operation) commands followed by a PS (program stop).

e) Once in the CYBER 170 mode, the processor should (for conditions 0 through 3) immediately respond to the 26XX command by performing the exchange swap and sending an exchange accept to the PP. Conditions 4 and 5 should fall through the NO commands and execute the PS.

f) Whether or not a swap occurs, and which of the 2 CYBER 170 packages are selected for the swap, is dependent respectively upon whether the process is now in the CYBER 170 monitor or job mode and which of the 26XX commands is executed. Whether the CYBER 170 monitor flag changes state is dependent upon both of these factors.

g) For conditions 4 and 5 the program stop referred to in step e) above will cause an immediate executive state exchange to executive state monitor mode.

h) For conditions 0 through 3, either the CYBER 170 "exchange to A", or the "exchange to MA" package is selected and will determine which of 2 short CYBER 170 programs are executed. In both cases, a unique flag is set up in CM ($AAAA_{16}$ in the exchange to A flag or $DEAD_{16}$ in the exchange to MA flag) and a program stop (CYBER 170 PS command) occurs at a unique address.

i) For conditions 1 through 3 the PS command will now cause a return to executive state monitor mode.

j) For condition 0, the swap caused by the 2600 command is an exchange to A and leaves the CYBER 170 monitor flag clear. The program stop in the resulting exchange to A process causes a second swap (an exchange to MA) which sets the CYBER 170 monitor flag. Finally a second program stop occurs causing the return to executive state monitor mode. Note that in this case, both the exchange to A and exchange to MA flags will be set up.

k) For all conditions, the return to executive state monitor mode results in the execution of a HALT command. The HALT causes a register file dump. If scope mode operation is selected, the process is deadstarted again and the loop is repeated starting at step a) above.

CMP.01 The P value in the executive state monitor package is checked. The P should point to the command following the EXCHANGE which was executed by the executive state monitor process (see step c) above).

CMP.02 The P value in the CYBER 170 "exchange to A" package is checked. If the package was swapped, the P will point to the NO command where the CYBER 170 process commences (see steps d) and e) above). If the swap does not occur, the original value (pointing to the "exchange to A" process) should still be there. The swap should always occur for the 2600 command and also for the 2610 when the response is from the CYBER 170 job mode (conditions 0, 1, and 3).

CMP.03 This compare is similar to CMP.02 above except that the CYBER 170 exchange to MA package is checked. The swap should only occur to this package for the 2620 command when the response is from the CYBER 170 job mode (condition 2). In this case the P value should point to the NO command (see steps d) and e) above). For conditions 1, 3, 4, and 5 it should point to the "exchange to MA" process. For condition 0 it should be 0. The first program stop which causes the exchange to MA (see step j) above) clears P before the swap occurs.

CMP.04 The status of the exchange to A flag is checked. It should be $AAAA_{16}$ for conditions 0, 1, and 3 only.

CMP.05 The status of the exchange to MA flag is checked. It should be $DEAD_{16}$ for conditions 0 and 2 only.

CMP.06 The value for P pushed onto the stack frame as a result of the trap is checked. It should point to the BRXEQ command where the executive state monitor process awaited the arrival of the trap interrupt. Note that the trap occurs for all conditions of this section, regardless of the type of 26XX command and whether CYBER 170 job or monitor mode is selected for the response.

CMP.07 The exchange request bit in the MCR register pushed onto the stack frame is checked to ensure that it is set.

CMP.08 This comparison checks that the Top of Stack pointer for the ring of execution of the monitor exchange package is properly advanced in response to the trap interrupt.

CMP.09 The Dynamic Space Pointer (DSP) in A0 of the monitor exchange package is checked to ensure that it was advanced to the next stack frame. Note that the contents of A0 were saved in X5 after the trap (step b) above).

CMP.10 This is a block comparison to check that the proper values were set up in registers A1 through A4 of the monitor exchange package when the trap interrupt occurred. These registers are for the Current Stack Frame Pointer, the Previous Save Area Pointer, the Binding Section Pointer, and the Argument Pointer.

Section 7 - Multiple 2600 Exchanges and B0 Checking (SECT 7)

In this section, from one to four slave PPs perform 2600 exchange commands and check the processor's response by ensuring that B0 of the CYBER 170 exchange package is zeroed out as a result of the package swap (register B0 in a CYBER 170 environment is always zero). Each of the 4 slave PPs causes the swap of a package reserved for that PP alone. While the slave PPs are performing their operation, the CP is continually executing a CEJ/MEJ command which specifys the use of another (5th) CYBER 170 package reserved for CP use only.

The CYBER 170 code executed by the CP consists of a CEJ/MEJ followed by two no operation commands (NO - OP code 46000_8) which are in turn followed by an unconditional jump (JP - op code 02_8) back to the CEJ/MEJ. There are five identical code sequences like this; the P value in each of the 5 CYBER 170 exchange packages points to one of these code sequences.

In every case, the value which is loaded into MA when a package swap occurs points to the fifth CYBER 170 exchange package reserved for the CP. Note that whenever a CEJ/MEJ command is executed, the CYBER 170 monitor flag (MF) changes state. If the flag is clear (job mode) a CEJ/MEJ uses MA to select the next exchange package to be used. If MF is set (monitor mode), the operand of the CEJ/MEJ determines the package to be used. In all of the 5 code sequences, this operand points to the package reserved for the CP.

To summarize, every CEJ/MEJ command will cause a swap using only the package in central memory reserved for the CP. On the other hand every slave PPs A register will select its own specific package when a 2600 command is executed. This provides the basis for verifying the occurance of every exchange swap for which a PP sends a request and receives an exchange accept

from the CP. If for some reason, any given slave PPs exchange request is missed or the exchange request address is misinterpreted, the check of B0 by each slave will detect the failure.

There are 98 conditions in the section; the first 49 operate in the synchronized mode while the last 49 are identical except that the PPs operate in the free-run mode (see section 5.1.3.3) Both sets of 49 conditions are controlled by 2 tables, each occupying 49 words (one for each condition) of central memory. The first table determines how many of the 4 PPs will be active and executing 2600 (EXN) commands. The various conditions proceed from having 1 slave active to 2, 3, and 4. The second table determines how many 2600 commands will be executed by each slave and also the length of the delay prior to execution of each command. For any given condition, the number of 2600 commands executed by each slave is identical; however the delays are variable for each slave.

It should be observed that as a result of variations in timing and thus the sequencing of the CEJ/MEJ and 2600 commands, the packages will move around in CM in an apparent random manner. The CYBER 170 B4 register in each package is used to uniquely identify each package as follows:

Exchange package for slave PP 0 - B4 = 7070_{16}
Exchange package for slave PP 1 - B4 = 7171_{16}
Exchange package for slave PP 2 - B4 = 7272_{16}
Exchange package for slave PP 3 - B4 = 7373_{16}
Exchange package for CP (CEJ/MEJ) - B4 = 7777_{16}
Initial register contents - B4 = 8888_{16}

The package with B4 equal to 8888_{16} is loaded into the CP registers when the processor first enters the CYBER 170 mode after the EXCHANGE from executive state monitor to job mode. There are, of course, only 5 physical CYBER 170 packages in CM for this section (the first 5 listed above). In some cases it may be possible to determine the sequence of events leading to a failure by examining the location of the packages after the failure. The value of B4 in each of the packages is restored on each new condition and on repeat condition. The sequence of events for all conditions is as follows:

a) After the initial deadstart, the executive state monitor process sets up the 5 CYBER 170 exchange packages. The P, RA, FL, and B4 of the package are restored. The CP clears a flag in CM to signal EXCHITC that initialization is complete and then it executes a executive state EXCHANGE command.

b) The EXCHANGE switches the processor to the executive state job mode with a VMID of 1. The processor will then begin to execute the CEJ/MEJ exchange sequence.

c) After the EXCHANGE to job, EXCHITC will trigger the slave PP or PPs to commence execution of their 2600 commands. Note that the processor does not exit the executive state job process to perform the swap in response to a PP exchange request.

d) Each slave commanded to execute 2600 commands will perform the following sequence:

1. Executes the delay defined for itself in the control table entry for the current condition.

2. Loads into its A register the address of its specific CYBER 170 exchange package.

3. Performs a CRDL command to read the first word of the exchange package which contains P, A0, and B0.

4. Modifies the B0 portion of the word and writes it back into the package in CM. The B0 portion is modified to contain the address of the exchange package itself.

5. Executes a keypoint command and then immediately afterward the 2600 command.

6. After the exchange accept is received, the exchange count is decremented and then the first word of the exchange package is again read from memory.

7. The B0 portion of the word is examined. Only if the value is zero is the incrementing of a "B0 miss count" skipped.

8. Each active slave PP returns to step 1 and repeats the sequence until the required number of 2600 commands have been executed.

d) EXCHITC deadstarts the CP after sufficient time has been allowed for all slaves to execute their required number of 2600 commands. This causes a CP halt and a register file dump.

CMP.01 Upon completion of each condition, each slave writes two values into central memory. First, a value which indicates the number of exchanges for which it did not receive an exchange accept and second, the "B0 miss count". The two values are written into the first and second parcels respectively of one central memory word. Both values should be zero. The 4 slaves write into sequential memory words; this is a block compare which checks all 4 words at once.

Section 8 - 26XX Exchanges and Block Writes In CYBER 170 Mode (SECT 8)

In this section the processor is placed in the CYBER 170 mode executing the "X register shift sequence" while from 1 to 4 slave PPs are commanded to execute PP 26XX exchanges or 60-and-64 bit block writes. The purpose is to determine if these asynchronously occurring PP operations will conflict with the successful completion of the shift sequence.

SUBSECTION 0 - This subsection is used only to perform initialization for the section.

SUBSECTION 1 - There are 23 conditions in this subsection; they all operate in the synchronized mode. The 23 conditions are controlled by two tables, each occupying 23 words (one for each condition) of central memory. The first table determines how many of the 4 PPs will be active and executing 2600 (EXN) commands. The various conditions proceed from having 1 slave active to having 2 and 4 salves active. The second table determines how many 2600 commands

will be executed by each slave and also the length of the delay prior to execution of each 2600 command. For any given condition, the number of 2600 commands executed by each slave is identical; however, the delays are variable for each slave. In most conditions, each slave performs only one 2600 command; in later conditions they perform 2 or more.

SUBSECTION 2 - This subsection is identical to subsection 1 except that the PPs operate in the free-run mode.

SUBSECTION 3 - This subsection operates in the free-run mode and is similar to subsection 2 except that one or more of the slave PPs will execute 2610 or 2620 exchange commands. These commands operate as pass commands if executed while the processor is in CYBER 170 mode. Since the CYBER 170 process contains CEJ/MEJ commands which switch the processor back and forth between job and monitor mode, the 2610s and 2620s will randomly pass or execute, depending upon the PP/CP timing. In this section all 26XX commands will always address the same CYBER 170 package.

SUBSECTION 4 - This subsection operates in the free-run mode and is similar to subsection 2 with the following differences. Two of the four slave PPs are assigned to perform 60- or executive state block writes to central memory while the remaining 2 perform 2600 exchanges. Each of the slaves performing the write operations are assigned their own block (3040 words) of CM. The timing is such that these PPs will be writing CM throughout the time that the CP is performing its X register shift sequence.

The data written by each of the two PPs consists of eight identical blocks (380 words each) of sequentially numbered words imprinted with the identification of the PP involved. This arrangement allows the detection of any shuffled or missing words. EXCHITC compares the first set of four blocks with the second set of 4 in separate comparison checks for each PP. It is unlikely that a conflict will generate an identical error in both sets of one PPs data.

The executive state job exchange package is defined with a VMID of 1 for CYBER 170 mode, all MCR mask bits are zeroes and traps are disabled. The processor should never return to the executive state monitor mode to service a PP exchange request since the response occurs immediately from the job mode. The sequence of events for all conditions is as follows:

a) After the initial deadstart, the executive state monitor process performs a complete setup of this section's CYBER 170 exchange package. It clears the deadstart flag in CM to signal EXCHITC that initialization is complete and then it executes a executive state EXCHANGE command.

b) The EXCHANGE switches the processor to the executive state job mode with a VMID of 1. The processor will then begin to execute the CYBER 170 X register shift sequence (see section 5.2).

c) After EXCHANGE to job, EXCHITC will trigger the slave PP or PPs to commence execution of their commands. These will occur during the shift sequence. Since the CEJ/MEJ commands in the shift sequence cause a swap between two identical CYBER 170 processes, and, since the PP exchanges point to the same CYBER 170 package as the CEJ/MEJs, the processes should not be affected except to the extent that the swapping will occur more often.

d) When the shift sequence for both CYBER 170 processes is complete, the processor will execute a CYBER 170 PS command. This will result in a return to the executive state monitor mode as described in section 5.2 above. If the scope mode is selected, EXCHITC will deadstart the CP and loop back to step a) above; otherwise, comparisons will commence as defined below.

CMP.01 Upon completion of each condition, each slave writes into central memory a value which indicates the number of exchanges for which it did not receive an exchange accept from the CP. The value should be zero and is written into the first 16-bit parcel of a CM word. The 4 slaves write into sequential memory words; this is a block compare which checks all 4 words at once.

CMP.02 This compare checks the P value in the RF dump. It should point to the HALT command in the executive state monitor process which is executed upon completion of the shift sequence.

CMP.03 A block compare is used to check that the CYBER 170 X registers (X0 through X7) in the executive state job package contain all ones.

CMP.04 A block compare is used to check that the CYBER 170 X registers (X0 through X7) in the CYBER 170 Exchange package in central memory contain all ones.

CMP.05 This comparison only occurs for subsection 4. The block of data written by the first slave PP performing a block write operation is checked. The first half of the block is compared with the second half.

CMP.06 Same as for compare 06 above except for the second slave PP performing a write opertion.

Section 9 - 2600 Exchanges and CP Stop/Start In CYBER 170 Mode (SECT 9)

This section is very similar to section 8 with the addition that while a CYBER 170 process is being executed along with 2600 exchanges from the slave PPs, the EXCHITC PP is also continuously stopping and starting instruction execution. The purpose of the test is to check that a process can be stopped and saved; that various registers can be read and written, and that the process can then be reloaded and restarted without disturbing the successful completion of the process.

Subsection 0 has 1 condition only and this is employed to perform initialization. During any condition of subsection 1, the processor is normally in CYBER 170 mode executing the X register shift sequence (see section 5.2). The executive state job and monitor processes both operate with the UCR and MCR mask registers cleared and with the traps disabled. When a 2600 exchange occurs, the CYBER 170 exchange package swap occurs directly during the executive state job mode.

There are 14 conditions in subsection 1, all of which operate with the PPs in the free-run mode (see section 5.1.3). The 14 conditions are controlled by 2 tables, each occupying 14 words (one for each condition) of central memory. The first table determines how many of the 4 PPs will be active and executing

2600 (EXN) commands. The various conditions proceed from having no slave active to having 1, 2, 3, and 4 active.

The second table determines how many 2600 commands will be executed by each slave and also the length of the delay prior to execution of each command. For any given condition, the number of 2600 commands executed by each slave is identical; however, the delays are variable for each slave.

There are approximately 100 stop/start operations during the execution of a single condition. Deadstart operations occur during each of these stop/starts. If a scope trigger is required, it should be taken from the test point for the model independent refresh resync command. This trigger will occur only once during a condition or a scope loop. The sequence of events for all conditions in subsection 1 is as follows:

a) After the initial deadstart, the executive state monitor process sets up a CYBER 170 exchange package and an executive state job exchange package. This is done by 2 copy sequences. A sequence complete flag in central memory is cleared at this time. At the end of the condition sequence, this flag will be set by the CP and tested by EXCHITC. The CP also clears the deadstart flag in CM to signal EXCHITC that initialization is complete; it executes the refresh resync command and then a executive state EXCHANGE command.

b) The EXCHANGE switches the processor to the executive state job mode with a VMID of 1 AND THE C170 MF=0. The processor will then begin to execute the CYBER 170 X register shift sequence (see paragraph titled, CYBER 170 X Register Shift Sequence, above).

c) After the EXCHANGE to job, EXCHITC will trigger the slave PP or PPs to commence execution of their 2600 commands. These will occur during the shift sequence. Since the CEJ/MEJ commands in the shift sequence cause a swap between two identical CYBER 170 processes, and since the PPs point to the same CYBER 170 package as the CEJ/MEJs, the processes should not be affected except to the extent that the swapping will occur more often.

d) EXCHANGE OUT - While the X register shift sequence and PP exchange activity are occuring, the EXCHITC PP will commence an exchange out sequence which is described in the following steps:

    1. A stop function is sent to the CP on the MAC channel.

    2. EXCHITC checks the halt bit in the CP status summary register; if it is set, it continues to step 3. If it is not set, a check for the halt condition will continue for roughly 2 milliseconds before the exchange out/exchange in sequence is abandoned.

    3. EXCHITC sets the preserve exchange bit in the DEC register. This ensures that a PP exchange request which occurs during the stopped condition will be maintained in the the MCR register until the processor returns to the running state. A subsequent exchange request by another PP is effectively held in the IOU until an accept is sent in response to the first request in the MCR.

    4. EXCHITC initiates a deadstart to cause a half exchange out which writes the process registers to a executive state exchange package in

central memory. EXCHITC tests the executive state monitor flag in the status summary to select the proper deadstart address. Since the process is stopped in the job mode, the address selected should always be to exchange out to the job package.

5. EXCHITC initiates another deadstart which starts microcode execution and allows EXCHITC to read into PP memory the contents of the following processor registers: SIT,PSM,PTL,PTA,MPS,JPS.

e) RELOAD - EXCHITC now commences a reload operation which consists of the following operations:

1. A deadstart is initiated which causes microcode initialization including setting default values into various egisters including some or all of those just saved in PP memory.

2. A clear errors function is issued to the CP.

3. A reload sequence is initiated in which the register contents saved in PP memory in step d)5 above are now written back into the same registers.

f) EXCHANGE IN - The EXCHITC PP completes the stop/start sequence by executing the exchange in sequence which is as follows:

1. A master clear function and then a clear errors function are sent to the CP.

2. The processor's S (CMA) register is loaded with the appropriate address to allow instruction execution to commence. This is performed by a write operation on the MAC channel.

3. The preserve exchange bit in the DEC register is cleared.

4. The B0 word in the CYBER 170 exchange package in CM is set to a nonzero value.

5. The start function is issued to the processor. This causes a half exchange in sequence which loads processor registers from the executive state monitor package in CM. The monitor process immediately executes a executive state EXCHANGE and picks up the job process. The process (CYBER 170 X register shift sequence) should be in exactly the same state as it was when the stop function was issued in step d)1 above, and it should continue execution.

6. EXCHITC checks to see if the CYBER 170 process running in the CP has now set the sequence complete flag in CM indicating that all X registers have finished sequencing. If it is set, EXCHITC exits to step g) below.

7. If the sequence is not complete, EXCHITC checks to determine if B0 in the CYBER 170 package in CM has been cleared. This would indicate that a CYBER 170 exchange package swap has occurred and that the process has successfully continued operation subsequent to the start function (step f)5 above). If this is the case, the sequence returns

to step d) above and the commencement of another stop/start operation. If B0 does not clear, step f)6 above and this step are repeated several times before EXCHITC exits to step g). A failure of the process to restart properly will be detected by errors during comparison checks.

g) The CYBER 170 process, having set the sequence complete flag, waits for EXCHITC to respond by clearing the flag. The processor will detect the clearing of the flag and will immediately execute a CYBER 170 PS command. This will result in a return to the executive state monitor mode and a HALT as described in paragraph titled, CYBER 170 Program Stop Sequence, above. If the scope mode is selected, EXCHITC will deadstart the CP and loop back to step a) above; otherwise, comparisons will commence as defined below.

CMP.01 Upon completion of each condition, each slave writes into central memory a value which indicates the number of exchanges for which it did not receive an exchange accept from the CP. The value should be zero and is written into the first 16 bit parcel of a CM word. The 4 slaves write into sequential memory words. This is a block compare which checks all 4 words at once.

CMP.02 This compare checks the P value in the RF dump. It should point to a HALT command in the executive state process (step g) above).

CMP.03 A block compare is used to check that the CYBER 170 X registers (X2 through X7) in the executive state job exchange package contain all ones.

CMP.04 A block compare is used to check that the CYBER 170 X registers (X2 through X7) in the CYBER 170 Exchange package in central memory contain all ones.

Section 10 - 2600 Exchanges and External Interrupts (SECT 10)

This section is similar to section 8 exceptthat a single slave PP is assigned to execute an INPN command (op code 102601) instead of a 2600 command. The 102601 command causes an external interrupt (MCR bit 8) to be sent to the processor via CM port 1, to which the processor is connected.

During any condition of this test, the processor is usually in CYBER 170 mode executing the X register shift sequence. The executive state job and monitor processes both operate with the mask bit for the external interrupt set and with the traps enabled. Normally when a 102601 is executed, the processor will perform an exchange interrupt back to the executive state monitor process. This process increments a count of the number of external interrupts received in the 170 mode and returns to the 170 process by executing a executive state EXCHANGE.

If another external interrupt arrives while the processor is in the executive state monitor process servicing the previous external interrupt, a trap will occur. The resulting trap process increments another counter for the external interrupts received in the executive state monitor mode and then RETURNs to the interrupted executive state monitor process. For either interrupt case, the CP and the slave PP which is sending the external interrupts perform a

handshake to ensure that the interrupts occur one at a time and that a response is received.

Subsection 0 has only 1 condition; it performs initialization. Subsection 1 has 15 conditions and operates in the free-run mode. Both sets of 15 conditions are controlled by two tables, each occupying 15 words (one for each condition) of central memory. The first table determines how many of the four PPs will be active and executing; either the 102601 (INPN) or the 2600 (EXN) commands. The various conditions proceed from having 1 slave active to 2, 3, and 4. The second table determines how many 2600 commands will be executed by each of three slaves and also the delay between the execution of each 2600 command. Only one slave is assigned to execute the 102601 interrupt command and the number (approximately 100) of these depends upon the speed of interaction between the slave and the CP. This slave executes a small random delay before executing each 102601 command. The sequence of events for all conditions is as follows:

a) After the initial deadstart, the executive state monitor process sets up a CYBER 170 exchange pack and executive state job exchange pack. This is done by two copy sequences. The CP clears the deadstart flag in CM to signal EXCHITC that initialization is complete. The CP then clears an interrupt disable flag and sets an interrupt request flag; this triggers the first external interrupt from the selected slave PP. Immediately thereafter the CP executes a executive state EXCHANGE command.

b) The EXCHANGE switches the processor to the executive state job mode with a VMID of 1. The processor will then begin to execute the CYBER 170 X register shift sequence.

c) After the EXCHANGE to job, EXCHITC will trigger the slave PP or PPs to commence execution of their 102601 or 2600 commands. These will occur during the shift sequence. Since the CEJ/MEJ commands in the shift sequence cause a swap between 2 identical CYBER 170 processes, and since the PPs point to the same CYBER 170 package as the CEJ/MEJs, the processes should not be affected except to the extent that the swapping will occur more often. Note that the processor does not exit the executive state job process to perform the swap in response to a PP exchange request.

d) If an external interrupt occurs, the processor exchange interrupts to the executive state monitor process. The external interrupt bit in the MCR of the executive state job package is cleared and the 170 mode external interrupt count in CM is incremented. The CP waits for the slave PP to clear the interrupt request flag and then checks the interrupts disabled flag. If the disable is not set, the CP requests another interrupt before a executive state EXCHANGE is executed to return to the 170 state X register shift operation.

e) This step only occurs if a trap interrupt occurs as a result of an external interrupt arriving during step d) above. The executive state monitor process is stacked and the processor is pointed to a trap routine where the MCR bits from the stack are checked for an external interrupt. The executive state mode external interrupt count in CM is incremented by one and then, as described in step d) above, another interrupt request may be issued before a RETURN is made to the point of interruption in step d) above.

f) When the shift sequence for both CYBER 170 processes is complete, the processor will set the interrupt disable flag and check the interrupt request flag. If a request is outstanding it will wait until the request is cleared before it executes a 170 state PS command. This will result in a return to the executive state monitor mode as described in paragraph titled, CYBER 170 Program Stop Sequence under Section Descriptions, above. If scope mode is selected, EXCHITC will deadstart the CP and loop back to step a) above; otherwise, comparisons begin as defined below.

CMP.01 Upon completion of each condition, each slave writes into central memory a value which indicates the number of exchanges for which it did not receive an exchange accept from the CP. The value should be zero and is written into the first 16-bit parcel of a CM word. The four slaves write into sequential memory words; this is a block compare which checks all four words at once.

CMP.02 Checks the P value in the RF dump. It should point to the HALT command in the executive state monitor process to which control is returned as a result of the program stop sequence. See paragraph titled, CYBER 170 Program Stop Sequence, under Section Descriptions, above.

CMP.03 A block compare is used to check that the CYBER 170 X registers (X1 through X7) in the executive state job package contain all ones.

CMP.04 A block compare is used to check that the CYBER 170 X registers (X1 through X7) in the 170 Exchange pack in central memory contain all ones.

Section 11 - 2600 Exchanges and SIT/PIT Interrupts (SECT 11)

This section is similar to section 8 with the addition that The executive state CP code contains routines that will cause SIT (system interval timer) and PIT (process interval timer) exchange and trap interrupts to occur at relatively high rates. Also included is code to handle these interrupts and to check that a specified number of them occur. The purpose is to verify the operation of the SIT and PIT trap mechanism and to ensure that failures due to conflicts among SIT and PIT interrupts and PP and CP exchanges do not occur.

During any condition of this test, the processor is normally in CYBER 170 mode executing the X register shift sequence. The executive state job and monitor process both operate with the mask bits for the PIT (UCR mask) and SIT (MCR mask) set and with traps enabled. When either of these interrupts occur, the processor exchanges or traps to a routine where the PIT or SIT will be restored with a value which will cause another interrupt of the same type after a short interval. In the case of a SIT interrupt or a PIT interrupt from the job process, an interrupt count is decremented; when the count reaches zero, the SIT or PIT will be restored with a high value which inhibits further interrupts of the corresponding type.

PIT interrupts from the monitor process are not counted at all. If the timing were such that only PIT interrupts from the monitor process were left to occur, they would not occur since the machine is normally executing in the executive state job state.

The MCR mask bit for the exchange requests is turned off in both the monitor and job exchange packages. The processor responds to a PP exchange with a CYBER 170 package swap immediately from the executive state job mode.

Subsection 0 is used for initialization. There are 100 conditions in each of subsections 1 and 2. Subsection 1 operates in the synchronized mode while subsection 2 is identical except that the PPs operate in the free-run mode. Both sets of 100 conditions are controlled by 4 tables, each occupying 10 words of central memory. Each set of 100 conditions is controlled by 2 loops. Whenever the least significant digit of the condition number changes, new entries are selected from 2 of the tables. One entry controls the PIT interrupts and the other the SIT interrupts. Each entry specifys the interval between interrupts and the number of interrupts. When no interrupt is specified, the interval is set for maximum.

Whenever the tens digit of the condition number changes, new entries are selected from the 2 other tables. One of these selects the number of slaves executing 2600 commands (from 0 to 4). The remaining table determines the number of 2600 commands executed per slave and the delay prior to the excution of each of the 2600 commands. For any given condition, the number of 2600 commands executed by each slave is identical; however the delays are variable for each slave.

The tables are set up such that no 2600 commands are executed during the first ten (0 to 9) conditions. Additionally, condition 0 has only a single SIT interrupt, condition 1, a single PIT interrupt, and condition 2 a single one of each. The sequence of events for all conditions is as follows:

a) After the initial deadstart, the executive state monitor process picks up the table entries which control the SIT and PIT interrupts. EXCHITC will have selected the appropriate entries for the current condition and placed them in central memory for the processor. The CP masks the SIT and PIT interrupt counts and saves them in central memory. The SIT interval value is retained in the monitor process while the PIT interval value is written into the executive state job package in CM. It is written into an X register to be used to restore the PIT after a job trap and it is also written into the PIT of the job package to force the first PIT job trap to occur after the job process commences. The monitor process also sets up the CYBER 170 exchange package and the executive state job exchange package. This is done by two copy sequences. The CP clears the deadstart flag in CM to signal EXCHITC that initialization is complete; it then executes two CPYXS commands to load the SIT and PIT with the interval values. Finally, the CP executes a executive state EXCHANGE command.

b) The EXCHANGE switches the processor to the executive state job mode with a VMID of 1. The processor will then begin to execute the CYBER 170 X register shift sequence.

c) After the EXCHANGE to job, EXCHITC will trigger the slave PP or PPs to commence execution of their 2600 commands. These will occur during the shift sequence. Since the CEJ/MEJ commands in the shift sequence cause a swap between two identical CYBER 170 processes, and since the PPs point to the same CYBER 170 package as the CEJ/MEJs, the processes should not be affected except to the extent that the swapping will occur more often.

d) JOB TRAP - If or when a job trap interrupt occurs, a job trap routine performs the following operations:

1. The MCR bits from the job trap stack frame are checked for a PIT interrupt.

2. The job process PIT interrupt count in central memory is decremented by one.

3. If the resulting count is zero, a CPYXS command sets the PIT to maximum value (to inhibit further job interrupts) and then skips to step d)5.

4. If the count was nonzero a CPYXS command sets the PIT to the test interval to generate a subsequent job process PIT interrupt.

5. A RETURN command switches the processor back to the CYBER 170 X register shift sequence.

e) EXCHANGE INTERRUPT - When an exchange interrupt occurs, the processor returns to the executive state monitor mode where the following steps occur:

1. The MCR bits in the executive state job exchange package are checked for a SIT interrupt.

2. The SIT interrupt count in central memory is decremented by one.

3. The SIT bit in the MCR of the job package is cleared.

4. If the SIT interrupt count is now zero, a CPYXS command sets the SIT to maximum value (to inhibit further SIT interrupts) and then the process skips to step e)6.

5. A CPYXS command sets the SIT to the test interval to cause a subsequent SIT interrupt.

6. The processor performs an EXCHANGE to return to the job mode and the CYBER 170 shift sequence.

f) MONITOR TRAP - This step would not normally occur and should only occur if a SIT or PIT trap interrupt occurs while the processor is in the monitor mode servicing a SIT exchange interrupt as described in step e) above. A monitor process trap routine would perform the following operations:

1. The MCR bits from the monitor trap stack frame are checked. If all bits are false skip to step f)5.

2. If the SIT bit is set, the SIT interrupt count in central memory is decremented by one.

3. If the resulting SIT interrupt count is zero, the SIT is set to maximum value with a CPYXS to inhibit further SIT interrupts and the process skips to step f)6.

4. The SIT is set up with the SIT interval value to cause a subsequent SIT interrupt. Skip to step f)6.

5. The UCR bits from the monitor stack frame are checked. If a PIT interrupt has occurred the PIT is restored with the PIT interval value to permit a possible subsequent interrupt.

6. The processor executes a RETURN command to switch back to the monitor process described in step e) above.

g) When the shift sequence for both CYBER 170 processes is complete, the processor will execute a CYBER 170 PS command. The PS command will cause the processor to return to the executive state monitor mode as described in section 5.2 above. If the scope mode is selected, EXCHITC will deadstart the CP and loop back to step a) above; otherwise, comparisons will commence as defined below.

CMP.01 Upon completion of each condition, each slave writes into central memory a value which indicates the number of exchanges for which it did not receive an exchange accept from the CP. The value should be zero and is written into the first 16-bit parcel of a CM word. The four slaves write into sequential memory words; this is a block compare which checks all 4 words at once.

CMP.02 This compare checks the P value in the RF dump. It should point to the HALT command in the executive state monitor process to which the processor returns after completion of the X register shift sequence.

CMP.03 This compare checks the SIT interrupt count in central memory; it should have been decremented to zero.

CMP.04 This compare checks the job process PIT interrupt count in central memory; it should have been decremented to zero.

CMP.05 A block compare is used to check that the CYBER 170 X registers (X2 through X7) in the executive state job package contain all ones.

CMP.06 A block compare is used to check that the CYBER 170 X registers (X2 through X7) in the CYBER 170 Exchange package in central memory contain all ones.

Section 12 - Error Exit, Illegal Instructions (SECT 12)

This section tests illegal CYBER 170 instruction error exits. Subsection 0 is used primarily for initialization.

Subsections 1 and 2 perform checks in the CYBER 170 job and monitor modes respectively. There is no error exit select bit for illegal instructions. The set of conditions listed in the following table are tested in both subsection 1 and 2.

In this table short forms and symbols having the following meanings are employed:

parcl	Refers to one of the four parcels in a CYBER 170 60 bit word
^	Used to denote logical AND
'	Used to denote logical NOT
UEM_EN	Refers to the flag in bit 23 of word 4 of the executive state exchange package which enables unified extended memory operation.
ESM_MOD	Refers to the flag in bit 24 of word 4 of the executive state exchange package which specifies extended semiconductor memory mode.
EA	Refers to the flag in bit 4 of word 2 of the executive state exchange package which authorizes ECS operation.

Cond No.	Command	Violation
00	PS	30 bit command in parcl 3
01	RJ	30 bit command in parcl 3
02	JP	30 bit command in parcl 3
03	ZR	30 bit command in parcl 3
04	NZ	30 bit command in parcl 3
05	PL	30 bit command in parcl 3
06	NG	30 bit command in parcl 3
07	IR	30 bit command in parcl 3
08	OR	30 bit command in parcl 3
09	DF	30 bit command in parcl 3
10	ID	30 bit command in parcl 3
11	EQ	30 bit command in parcl 3
12	NE	30 bit command in parcl 3
13	GE	30 bit command in parcl 3
14	LT	30 bit command in parcl 3
15	SA5 A0+K	30 bit command in parcl 3
16	SA5 B0+K	30 bit command in parcl 3
17	SA5 X0+K	30 bit command in parcl 3
18	SB5 A0+K	30 bit command in parcl 3
19	SB5 B0+K	30 bit command in parcl 3
20	SB5 X0+K	30 bit command in parcl 3
21	SX5 A0+K	30 bit command in parcl 3
22	SX5 B0+K	30 bit command in parcl 3
23	SX5 X0+K	30 bit command in parcl 3
24	CEJ/MEJ	30 bit command in parcel 1
25	CEJ/MEJ	30 bit command in parcel 2
26	CEJ/MEJ	30 bit command in parcel 3
27	REC	parcl 0 ^ UEM_EN ^ ESM_MOD'

```
28          REC          parcl 1 ^ UEM_EN ^ ESM_MOD'
29          REC          parcl 2 ^ UEM_EN ^ ESM_MOD'
30          WEC          parcl 1 ^ UEM_EN ^ ESM_MOD'
31          WEC          parcl 2 ^ UEM_EN ^ ESM_MOD'
32          WEC          parcl 3 ^ UEM_EN ^ ESM_MOD'
```

After the initial deadstart in any condition of subsection 1 or 2, the processor enters the executive state monitor mode where it sets up a executive state job exchange package and a CYBER 170 exchange to MA package. The CYBER 170 exchange package will be swapped on any occasion when an error exit occurs in the CYBER 170 job mode. The processor subsequently performs a executive state exchange to executive state job mode which is defined by the executive state job package VMID to be CYBER 170 mode. The CYBER 170 monitor flag will also be set or clear as specified for the particular subsection.

Once in the CYBER 170 mode, the processor will initialize its registers and central memory as required to provide the proper conditions for any one of the illegal commands. Prior to the deadstart, ITCEXCH will have selected a word of data containing the illegal command for the current condition. It will write this word into the sequence of commands executed by the CYBER 170 process.

If the processor fails to detect the illegal command set up by ITCEXCH and continues execution it should sequence its way to a subsequent PS command which will cause an illegal command error exit sequence.

Normally, when the illegal condition is detected, the processor records the current value of the P register in RA and proceeds to execute the error exit sequence. This sequence will differ in detail depending upon the subsection (job or monitor mode) and the current condition. The descriptive data provided for each comparison check provides additional detail.

CMP.01 The P value in the CYBER 170 exchange to MA package is checked. For subsection 1 (CYBER 170 job mode), the error exit which occurs sets the P to zero after which this package is swapped. The P in this package will therefore be 0. For subsection 2 (CYBER 170 monitor mode), the error exit immediately causes a return to executive state monitor mode and a swap of this package does not occur. The value in P should point to the CYBER 170 exchange to MA process to which it was initialized by the executive state monitor process.

CMP.02 The P value and error exit condition bits at the time of the error exit are saved in RA by the CP and are checked by this comparison. Any illegal CYBER 170 command should always cause an error exit. There is neither an exit select bit nor an exit condition bit for illegal commands. In this comparison, no exit condition bits should be set at all. For subsection 1 (CYBER 170 job mode) this comparison is performed on a memory cell in which RA was previously saved for later checking.

CMP.03 The P value in the executive state job exchange package is checked. For subsection 1 (CYBER 170 job mode), it should point to the PS command in the process which is entered after CYBER 170 exchange to MA package is swapped. Since the return to executive state monitor occurs directly after an exit in CYBER 170 monitor mode, the value for subsection 2 will point directly to the illegal command which caused the error. The illegal commands occur on various parcel boundaries as specified by the tables above. This comparison checks the P value accordingly.

CMP.04 The value of P in the register file dump is checked. The processor should always return to the executive state monitor process via a executive state exchange operation after the error exit caused by the infinite error or the PS command. The P should point to HALT which is executed after reentry to the executive state process.

CMP.05 This comparison checks that the CYBER 170 A register (and B register for job mode) addressed by the illegal command as the destination register is not altered. This pertains to A5 in conditions 15, 16, and 17 and B5 in conditions 18, 19, and 20 for job mode). For monitor mode (subsection 2), comparison 11 below, checks the B registers.

CMP.06 This is similar to comparison 5 above except that X5 is checked for conditions 21, 22, and 23.

CMP.07 This comparison checks to ensure that the REC, WEC, RX, and WX commands (see table of conditions above) did not execute and thereby alter central memory. Only that portion of central memory indicated by their appropriate operands is checked.

CMP.08 The CYBER 170 monitor flag in the executive state job exchange package is checked. It should always be set.

CMP.09 The exit mode halt flag in the executive state job exchange package is checked. It should always be set.

CMP.10 The value in the exchange to MA flag is checked. In subsection 1, the error exit causes a swap to a CYBER 170 process that sets this flag to $DEAD_{16}$. In subsection 2 the exit is directly to executive state mode and this flag will remain all zeroes.

CMP.11 This comparison occurs only in subsection 2 (CYBER 170 monitor mode) and checks that the destination B register (B5 in conditions 18, 19, and 20) are not altered. The extra comparison is required because the destination registers are found in the executive state exchange package in monitor mode subsections and in the CYBER 170 exchange to MA package in job mode subsections.

Section 13 - Error Exit, Address Out Of Range (SECT 13)

This section performs address out of range (AOR) error exit checks. Subsection 0 is used for initialization only. Subsections 1 through 4 perform the AOR checks for read and write commands with the CYBER 170 monitor flag and AOR exit select bit (248) set as follows:

Subsection 1 - Read/write AOR, CYBER 170 job mode, exit selected
Subsection 2 - Read/write AOR, CYBER 170 job mode, exit not selected
Subsection 3 - Read/write AOR, CYBER 170 monitor mode, exit selected
Subsection 4 - Read/write AOR, CYBER 170 monitor mode, exit not selected

Subsections 5 through 8 perform AOR error exit checks on branch and RNI operations with CYBER 170 monitor flag and the AOR exit select bit (248) set as follows:

Subsection 5 - Branch/RNI to AOR, CYBER 170 job mode, exit selected
Subsection 6 - Branch/RNI to AOR, CYBER 170 job mode, exit not selected
Subsection 7 - Branch/RNI to AOR, CYBER 170 monitor mode, exit selected
Subsection 8 - Branch/RNI to AOR, CYBER 170 monitor mode, exit not selected

In the following tables, the operand OUTRANG refers to the address which is the first word beyond the field length, while the operand INRANG is the adjacent address which is at the end of the field length.

The following set of conditions is repeated in subsections 1 through 4:

Cond No.	Command		
00	SA5 A5+K	where	A5=INRANG, K=1
01	SA5 B5+K	where	B5=INRANG, K=1
02	SA5 X5+K	where	X5=INRANG, K=1
03	SA5 X5+B1	where	X5=INRANG, B1=1
04	SA5 A5+B1	where	A5=INRANG, B1=1
05	SA5 A4-B4	where	A4=0, B4= -OUTRANG
06	SA5 B5+B1	where	B5=INRANG, B1=1
07	SA5 B6-B1	where	B6=OUTRANG+1, B1=1
08	SA6 A5+K	where	A5=INRANG, K=1
09	SA6 B5+K	where	B5=INRANG, K=1
10	SA6 X5+K	where	X5=INRANG, K=1
11	SA6 X5+B1	where	X5=INRANG, B1=1
12	SA6 A5+B1	where	A5=INRANG, B1=1
13	SA6 A4-B4	where	A4=0, B4= -OUTRANG
14	SA6 B5+B1	where	B5=INRANG, B1=1
15	SA6 B6-B1	where	B6=OUTRANG+1, B1=1

The following set of conditions is repeated in subsections 5 through 8:

Cond No.	Command
00	JP B0,OUTRANG
01	ZR X0,OUTRANG
02	NZ X5,OUTRANG
03	PL X5,OUTRANG
04	MI X4,OUTRANG
05	EQ B1,B1,OUTRANG
06	NE B0,B1,OUTRANG
07	GE B1,B0,OUTRANG
08	LT B0,B1,OUTRANG
09	JP B0,INRANG (RNI to OUTRANG)
10	RJ OUTRANG

After the initial deadstart in any condition of subsection 1 through 8, the processor enters the executive state monitor mode where it sets up a executive state job exchange package and a CYBER 170 exchange to MA package. The CYBER 170 exchange package will be swapped on any occasion when an error exit occurs in the CYBER 170 job mode. The processor subsequently performs a executive state exchange to executive state job mode which is defined by the executive state job package VMID to be CYBER 170 mode. The CYBER 170 monitor flag will also be set or clear as specified for the particular subsection.

Once in the CYBER 170 mode the processor will initialize its registers as required to provide the proper conditions for any one of the AOR commands. Prior to the deadstart, ITCEXCH will have selected a word of data containing the AOR command for the current condition. It will write this word into the sequence of commands executed by the CYBER 170 process.

If the processor fails to detect the AOR error set up by ITCEXCH and continues execution it should sequence its way to a subsequent PS command which will cause an illegal command error exit sequence.

Normally, when the AOR error condition is detected, the processor records the current value of the P register and the exit condition bits in RA and proceeds to execute the error exit sequence. This sequence will differ in detail depending upon the subsection (job or monitor mode) and the current condition. The descriptive data provided for each comparison check provides additional detail.

CMP.01 The P value in the CYBER 170 exchange to MA package is checked. For subsections 1, 2, 5, and 6 (CYBER 170 job mode), the error exit which occurs sets the P to zero after which this package is swapped. The P in this package will therefore be 0. For subsections 3, 4, 7, and 8 (CYBER 170 monitor mode), the error exit immediately causes a return to executive state monitor mode and a swap of this package does not occur. The value in P should point to the CYBER 170 exchange to MA process to which it was initialized by the executive state monitor process.

CMP.02 The P value and error exit condition bits at the time of the error exit are saved in RA by the CP and are checked by this comparison. When the exit select bit is set (subsections 1, 3, 5, and 7), the exit condition bit (248) should be set and the P value should point to the command which caused the AOR error. When the exit is not selected (subsections 2 and 4 for read/write AOR), an error exit should occur as a result of a PS command which follows shortly after the command which normally causes the AOR error. The PS command is an illegal command and although it causes an exit, there is no associated exit condition bit. For branch or RNI operations an exit should always occur, even if the AOR select bit is clear (subsections 6 and 8). In addition, the value recorded in RA should point to the branched to command, or to the command accessed via the RNI operation. For subsections 1, 2, 5, and 6 (CYBER 170 job mode) these values are in a memory cell in which RA was previously saved for later checking.

CMP.03 The value of P in the register file dump is checked. The processor should always return to the executive state monitor process via a executive state exchange operation after the error exit caused by the AOR error or the PS command. The P should point to HALT which is executed after reentry to the executive state process.

CMP.04 The CYBER 170 monitor flag in the executive state job exchange package is checked. It should always be set.

CMP.05 The exit mode halt flag in the executive state job exchange package is checked. It should always be set.

CMP.06 The value in the exchange to MA flag is checked. In subsections 1, 2, 5, and 6, the error exit causes a swap to a CYBER 170 process that sets this flag to $DEAD_{16}$. In subsections 3, 4, 7, and 8 the exit is directly to executive state mode and this flag will remain all zeroes.

CMP.07 This comparison only occurs in subsections 1 through 4. It checks to ensure the the A register referenced by the read/write AOR command is set to the out of range address.

CMP.08 This comparison also occurs only for subsections 1 through 4. It checks to ensure that the destination X register (for read AOR, or the addressed memory cell (for write AOR) have not been altered by the AOR command.

CMP.09 The P value in the executive state job exchange package is checked. For subsections 1 and 2 (CYBER 170 job mode), it should point to the PS command in the process which is entered after CYBER 170 exchange to MA package is swapped. Since the return to executive state monitor occurs directly after an exit in CYBER 170 monitor mode, the value for subsection 3 will point directly to the command which caused the read/write AOR error, and, for subsection 4 with no exit selected, it will point to the following PS command. This check is not performed in subsections 5 through 8.

Section 14 - Error Exit, Floating-point Infinite (SECT 14)

This section checks infinite error exit conditions (exit select bit 249) on all of the floating point add, subtract, multiply, and divide commands.

Subsection 0 is used for initialization only. Subsections 1 through 7 perform the checks with the CYBER 170 monitor flag and the exit select bit set up for the combinations described in the following tables:

Subsections 1 to 4 - Infinite error, CYBER 170 job mode, exit selected

Subsection 1 conditions

Command:	Xj +	Xk to	Xi	Op Code	30	32	34
ADD 30, 32, 34	W	+INF	+INF	Cond No.	00	06	12
ADD 30, 32, 34	W	-INF	-INF	Cond No.	01	07	13
ADD 30, 32, 34	+INF	W	+INF	Cond No.	02	08	14
ADD 30, 32, 34	+INF	+INF	+INF	Cond No.	03	09	15
ADD 30, 32, 34	-INF	W	-INF	Cond No.	04	10	16
ADD 30, 32, 34	-INF	-INF	-INF	Cond No.	05	11	17

Subsection 2 conditions

Command:	Xj -	Xk to	Xi	Op Code	31	33	35
SUB 31, 33, 35	W	+INF	-INF	Cond No.	00	06	12
SUB 31, 33, 35	W	-INF	+INF	Cond No.	01	07	13
SUB 31, 33, 35	+INF	W	+INF	Cond No.	02	08	14
SUB 31, 33, 35	+INF	-INF	+INF	Cond No.	03	09	15
SUB 31, 33, 35	-INF	W	-INF	Cond No.	04	10	16
SUB 31, 33, 35	-INF	+INF	-INF	Cond No.	05	11	17

Subsection 3 conditions

Command:	Xj *	Xk to	Xi	Op Code	40	41	42
MPY 40, 41, 42	+N	+INF	+INF	Cond No.	00	12	24
MPY 40, 41, 42	+N	-INF	-INF	Cond No.	01	13	25
MPY 40, 41, 42	-N	+INF	-INF	Cond No.	02	14	26
MPY 40, 41, 42	-N	-INF	+INF	Cond No.	03	15	32
MPY 40, 41, 42	+INF	+N	+INF	Cond No.	04	16	28
MPY 40, 41, 42	+INF	-N	-INF	Cond No.	05	17	29
MPY 40, 41, 42	+INF	+INF	+INF	Cond No.	06	18	30
MPY 40, 41, 42	+INF	-INF	-INF	Cond No.	07	19	31
MPY 40, 41, 42	-INF	+N	-INF	Cond No.	08	20	32
MPY 40, 41, 42	-INF	-N	+INF	Cond No.	09	21	33
MPY 40, 41, 42	-INF	+INF	-INF	Cond No.	10	22	34
MPY 40, 41, 42	-INF	-INF	+INF	Cond No.	11	23	35

Subsection 4 conditions

Command:	Xj /	Xk to	Xi	Op Code 44	45
DVD 44, 45	+INF	+N	+INF	Cond No. 00	08
DVD 44, 45	+INF	-N	-INF	Cond No. 01	09
DVD 44, 45	+INF	+0	+INF	Cond No. 02	10
DVD 44, 45	+INF	-0	-INF	Cond No. 03	11
DVD 44, 45	-INF	+N	-INF	Cond No. 04	12
DVD 44, 45	-INF	-N	+INF	Cond No. 05	13
DVD 44, 45	-INF	+0	-INF	Cond No. 06	14
DVD 44, 45	-INF	-0	+INF	Cond No. 07	15

Subsection 5 - Infinite error, CYBER 170 job mode, exit not selected; conditions same as for subsection 2

Subsection 6 - Infinite error, CYBER 170 monitor mode, exit selected; conditions same as for subsection 3

Subsection 7 - Infinite error, CYBER 170 monitor mode, exit not selected; conditions same as for subsection 4

After the initial deadstart in any condition of subsection 1 through 7, the processor enters the executive state monitor mode where it sets up a executive state job exchange package and a CYBER 170 exchange to MA package. The CYBER 170 exchange package will be swapped on any occasion when an error exit occurs in the CYBER 170 job mode. The processor subsequently performs a executive state exchange to executive state job mode which is defined by the executive state job package VMID to be CYBER 170 mode. The CYBER 170 monitor flag will also be set or clear as specified for the particular subsection.

Prior to the deadstart, ITCEXCH will have selected a word of data containing the floating-point command for the current condition. It will write this word into the sequence of commands executed by the CYBER 170 process. ITCEXCH also sets up (in the executive state job exchange package) the operands which will cause the infinite error when operated upon by the floating-point command.

If the processor fails to detect the infinite error set up by ITCEXCH and continues execution it should sequence its way to a subsequent PS command which will cause an illegal command error exit sequence.

Normally, when the infinite error is detected, the processor records the current value of the P register and the exit condition bits in RA and proceeds to execute the error exit sequence. This sequence will differ in detail depending upon the subsection (job or monitor mode) and the current condition. The descriptive data provided for each comparison check provides additional detail.

CMP.01 The P value in the CYBER 170 exchange to MA package is checked. For subsections 1 through 5 (CYBER 170 job mode), the error exit which occurs sets the P to zero after which this package is swapped. The P in this package will therefore be 0. For subsections 6 and 7 (CYBER 170 monitor mode), the error exit immediately causes a return to executive state monitor mode and a swap of this package does not occur. The value in P should point to the CYBER 170 exchange to MA process to which it was initialized by the executive state monitor process.

CMP.02 The P value and error exit condition bits at the time of the error exit are saved in RA by the CP and are checked by this comparison. When the exit select bit is set (subsections 1 through 4, and 6),the exit condition bit (249) should be set and the P value should point to the command which caused the infinite error. When the exit is not selected (subsections 5 and 7), an error exit should occur as a result of a PS command which follows shortly after the command that normally causes the infinite error. The PS command is an illegal command, and although it causes an exit there is no associated exit condition bit. For subsections 1 through 5 (CYBER 170 job mode) these values are in a memory cell in which RA was previously saved for later checking (see section 5.2 for further details).

CMP.03 The P value in the executive state job exchange package is checked. For subsections 1 through 5 (CYBER 170 job mode), it should point to the PS command in the process which is entered after CYBER 170 exchange to MA package is swapped. Since the return to executive state monitor occurs directly after an exit in CYBER 170 monitor mode, the value for subsection 6 will point directly to the command which caused the infinite error. For subsection 7 with no exit selected, it will point to the following PS command.

CMP.04 The value of P in the register file dump is checked. The processor should always return to the executive state monitor process via a executive state exchange operation after the error exit caused by the infinite error or the PS command. The P should point to HALT which is executed after reentry to the executive state process.

CMP.05 The value in the CYBER 170 destination register Xi (X2) is checked for the result from the floating point operation. The result should always be either the standard plus or minus infinite operand. Refer to the table of conditions above for the appropriate subsection. For subsections 1 through 5 (CYBER 170 job mode) this result is found in the CYBER 170 exchange to MA package. For subsections 6 and 7 where the exit is directly to executive state monitor mode the result should be found in the executive state job exchange package.

CMP.06 The CYBER 170 monitor flag in the executive state job exchange package is checked. It should always be set.

CMP.07 The exit mode halt flag in the executive state job exchange package is checked. It should always be set.

CMP.08 The value in the exchange to MA flag is checked. In subsections 1 through 5, the error exit causes a swap to a CYBER 170 process that sets this flag to $DEAD_{16}$. In subsection 6 and 7 the exit is to executive state mode and this flag will remain all zeroes.

Section 15 - Error Exit, Floating-point Indefinite (SECT 15)

This section checks indefinite error exit conditions on all of the floating point add, subtract, multiply and divide commands. The errors which are generated are refered to as indefinite errors since in each test case the destination register should be set up with the standard positive indefinite result (+IND). An indefinite or infinite error exit will only occur if at least one of the input operands is either infinite or indefinite. Specifically, in this section, cases are tested where combinations of 1 or 2 of the input operands may be + or - indefinite or + or - infinite. Either one or both of the error exit bits should be set, depending upon the input, but in all cases the destination register should be +IND. Either the indefinite or infinite error exit bit should be set if there is an input operand of the corresponding type.

Sub-section 0 is used for initialization only. Sub-sections 1 through 8 perform the checks with the Cyber 170 monitor flag and the exit select bit set up for the combinations described in the following tables:

Subsections 1 to 5 - Indefinite error, CYBER 170 job mode exit selected

Subsection 1 conditions

Commands:	Xj +,-	Xk to Xi		Cond Nos.
ADD,30,32,34 SUB,31,33,35	W	+IND	+IND	00 to 05
ADD,30,32,34 SUB,31,33,35	+INF	+IND	+IND	06 to 11
ADD,30,32,34 SUB,31,33,35	-INF	+IND	+IND	12 to 17
ADD,30,32,34 SUB,31,33,35	+IND	+IND	+IND	18 to 23
ADD,30,32,34 SUB,31,33,35	-IND	+IND	+IND	24 to 29
ADD,30,32,34 SUB,31,33,35	W	-IND	+IND	30 to 35
ADD,30,32,34 SUB,31,33,35	+INF	-IND	+IND	36 to 41
ADD,30,32,34 SUB,31,33,35	-INF	-IND	+IND	42 to 47
ADD,30,32,34 SUB,31,33,35	+IND	-IND	+IND	48 to 53
ADD,30,32,34 SUB,31,33,35	-IND	-IND	+IND	54 to 59
ADD,30,32,34 SUB,31,33,35	+IND	W	+IND	60 to 65
ADD,30,32,34 SUB,31,33,35	-IND	W	+IND	66 to 71
ADD,30,32,34 SUB,31,33,35	+IND	+INF	+IND	72 to 77
ADD,30,32,34 SUB,31,33,35	-IND	+INF	+IND	78 to 83
ADD,30,32,34 SUB,31,33,35	+IND	-INF	+IND	84 to 89
ADD,30,32,34 SUB,31,33,35	-INF	-INF	+IND	90 to 95
ADD,30,32,34 SUB,31,33,35	+IND	+IND	+IND	96 to 101
ADD,30,32,34 SUB,31,33,35	-IND	+IND	+IND	102 to 107
ADD,30,32,34 SUB,31,33,35	+IND	-IND	+IND	108 to 113
ADD,30,32,34 SUB,31,33,35	-IND	-IND	+IND	114 to 119

Subsection 2 conditions

Command:	Xj +,-	Xk to	Xi	Cond No.
ADD 30	+INF	-IND	+IND	0
ADD 30	-INF	+IND	+IND	1
ADD 32	+INF	-IND	+IND	2
ADD 32	-INF	+IND	+IND	3
ADD 34	+INF	-IND	+IND	4
ADD 34	-INF	+IND	+IND	5
SUB 31	+INF	+IND	+IND	6
SUB 31	-INF	-IND	+IND	7
SUB 33	+INF	+IND	+IND	8
SUB 33	-INF	-IND	+IND	9
SUB 35	+INF	+IND	+IND	10
SUB 35	-INF	-IND	+IND	11

Subsection 3 conditions

Commands:	Xj *,/	Xk to	Xi	Cond Nos.
MPY,40,41,42 DVD,44,45	+N	+IND	+IND	00 to 04
MPY,40,41,42 DVD,44,45	-N	+IND	+IND	05 to 09
MPY,40,41,42 DVD,44,45	+0	+IND	+IND	10 to 14
MPY,40,41,42 DVD,44,45	-0	+IND	+IND	15 to 19
MPY,40,41,42 DVD,44,45	+INF	+IND	+IND	20 to 24
MPY,40,41,42 DVD,44,45	-INF	+IND	+IND	25 to 29
MPY,40,41,42 DVD,44,45	+IND	+IND	+IND	30 to 34
MPY,40,41,42 DVD,44,45	-IND	+IND	+IND	35 to 39
MPY,40,41,42 DVD,44,45	+N	-IND	+IND	40 to 44
MPY,40,41,42 DVD,44,45	-N	-IND	+IND	45 to 49
MPY,40,41,42 DVD,44,45	+0	-IND	+IND	50 to 54
MPY,40,41,42 DVD,44,45	-0	-IND	+IND	55 to 59
MPY,40,41,42 DVD,44,45	+INF	-IND	+IND	60 to 64
MPY,40,41,42 DVD,44,45	-INF	-IND	+IND	65 to 69
MPY,40,41,42 DVD,44,45	+IND	-IND	+IND	70 to 74
MPY,40,41,42 DVD,44,45	-IND	-IND	+IND	75 to 79
MPY,40,41,42 DVD,44,45	+IND	+N	+IND	80 to 84
MPY,40,41,42 DVD,44,45	-IND	+N	+IND	85 to 89
MPY,40,41,42 DVD,44,45	+IND	-N	+IND	90 to 94
MPY,40,41,42 DVD,44,45	-IND	-N	+IND	95 to 99
MPY,40,41,42 DVD,44,45	+IND	+0	+IND	100 to 104
MPY,40,41,42 DVD,44,45	-IND	+0	+IND	105 to 109
MPY,40,41,42 DVD,44,45	+IND	-0	+IND	110 to 114
MPY,40,41,42 DVD,44,45	-IND	-0	+IND	115 to 119
MPY,40,41,42 DVD,44,45	+IND	+INF	+IND	120 to 124
MPY,40,41,42 DVD,44,45	-IND	+INF	+IND	125 to 129
MPY,40,41,42 DVD,44,45	+IND	-INF	+IND	130 to 134
MPY,40,41,42 DVD,44,45	-IND	-INF	+IND	135 to 139

Subsection 4 conditions

Commands:	Xj *	Xk	to Xi	Cond Nos.
MPY 40,41,42	+0	+INF	+IND	00 to 02
MPY 40,41,42	+0	-INF	+IND	03 to 05
MPY 40,41,42	-0	+INF	+IND	06 to 08
MPY 40,41,42	-0	-INF	+IND	09 to 11
MPY 40,41,42	+INF	+0	+IND	12 to 14
MPY 40,41,42	+INF	-0	+IND	15 to 17
MPY 40,41,42	-INF	+0	+IND	18 to 20
MPY 40,41,42	-INF	-0	+IND	21 to 23

Subsection 5

Commands:	Xj /	Xk	to Xi	Cond Nos.
DVD 44,45	+INF	+INF	+IND	00 to 01
DVD 44,45	+INF	-INF	+IND	02 to 03
DVD 44,45	-INF	+INF	+IND	04 to 05
DVD 44,45	-INF	-INF	+IND	06 to 07

Subsection 6 - Indefinite error, CYBER 170 job mode, exit not selected; conditions same as for subsection 3

Subsection 7 - Indefinite error, CYBER 170 monitor mode, exit selected; conditions same as for subsection 3

Subsection 8 - Indefinite error, CYBER 170 monitor mode, exit not selected; conditions same as for subsection 1

After the initial deadstart in any condition of subsections 1 through 8, the processor enters the executive state monitor mode where it sets up a executive state job exchange package and a CYBER 170 exchange to MA package. The CYBER 170 exchange package will be swapped on any occasion when an error exit occurs in the CYBER 170 job mode. The processor subsequently performs a executive state exchange to executive state job mode which is defined by the executive state job package VMID to be CYBER 170 mode. The CYBER 170 monitor flag will also be set or clear as specified for the particular subsection.

Prior to the deadstart, ITCEXCH will have selected a word of data containing the floating-point command for the current condition. It will write this word into the sequence of commands executed by the CYBER 170 process. ITCEXCH also sets up (in the executive state job exchange package) the operands which will cause the indefinite error when operated upon by the floating-point command.

If the processor fails to detect the indefinite error set up by ITCEXCH and continues execution it should sequence its way to a subsequent PS command which will cause an illegal command error exit sequence.

Normally, when the infinite error is detected, the processor records the current value of the P register in RA and proceeds to execute the error exit sequence. This sequence will differ in detail depending upon the subsection (job or monitor mode) and the current condition. The descriptive data provided for each comparison check provides additional detail. The reader should also refer to the description of the program stop sequence.

CMP.01 The P value in the CYBER 170 exchange to MA package is checked. For subsections 1 through 6 (CYBER 170 job mode), the error exit which occurs sets the P to zero after which this package is swapped. The P in this package will therefore be 0. For subsections 7 and 8 (CYBER 170 monitor mode), the error exit immediately causes a return to executive state monitor mode and a swap of this package does not occur. The value in P should point to the CYBER 170 exchange to MA process to which it was initialized by the executive state monitor process.

CMP.02 The P value and error exit condition bits at the time of the error exit are saved in RA by the CP and are checked by this comparison. When the exit select bit is set (subsections 1 through 5, and 7),the exit condition bit (250) should be set and the P value should point to the command which caused the indefinite error. When the exit is not selected (subsections 6 and 8), an error exit should occur as a result of a PS command which follows shortly after the command which normally causes the indefinite error. The PS command is an illegal command and although it causes an exit there is no associated exit condition bit. For subsections 1 through 6 (CYBER 170 job mode) these values are in a memory cell in which RA was previously saved for later checking.

CMP.03 The P value in the executive state job exchange package is checked. For subsections 1 through 6 (CYBER 170 job mode), it should point to the PS command in the process which is entered after CYBER 170 exchange to MA package is swapped. Since the return to executive state monitor occurs directly after an exit in CYBER 170 monitor mode, the value for subsection 7 will point directly to the command which caused the infinite error and for subsection 8 with no exit selected, it will point to the following PS command.

CMP.04 The value of P in the register file dump is checked. The processor should always return to the executive state monitor process via a executive state exchange operation after the error exit caused by the infinite error or the PS command. The P should point to HALT which is executed after reentry to the executive state process.

CMP.05 The value in the CYBER 170 destination register Xi (X2) is checked for the result from the floating point operation. The result should always be the standard plus indefinite operand. For subsections 1 through 6 (CYBER 170 job mode) this result is found in the CYBER 170 exchange to MA package. For subsections 7 and 8, where the exit is directly to executive state monitor mode, the result should be found in the executive state job exchange package.

CMP.06 The CYBER 170 monitor flag in the executive state job exchange package is checked. It should always be set.

CMP.07 The exit mode halt flag in the executive state job exchange package is checked. It should always be set.

CMP.08 The value in the exchange to MA flag is checked. In subsections 1 through 6, the error exit causes a swap to a CYBER 170 process that sets this flag to $DEAD_{16}$. In subsection 7 and 8 the exit is to executive state mode and this flag will remain all zeroes.

Section 16 - Exchange Addressing, Selectable Addresses (SECT 16)

The operation of this section of the test is very similar to section 2 except that the CM addresses pointed to by the 2600 command are controlled by selectable parameters instead of a fixed table of addresses. This permits the exchange to be checked at any address in memory above that area in which the test resides.

The parameters which control this operation are PARAMS 7 through 10 and 18. Refer to section I-6, paragraph 3.3.1. The exchange address (starting address), the address increment, and the number of exchanges are determined by the first 4 parameters. The set of exchanges which are selected by these parameters will be repeated within each one megabyte segment of memory.

The starting address (PARAM 7 and 8) must be greater than 10000_{16} if the first megabyte (bit 20 in PARAM 18) is enabled. EXCH resides below 10000_{16}. The starting address, the increment (PARAM 9) and the number of exchanges (PARAM 10) should be such that all exchanges will commence at addresses within a one megabyte segment. The condition count will advance by one after each exchange.

A CYBER 170 exchange involves the swap of 16 words to and from memory. It should be noted that although the first word of an exchange might be within a given one megabyte segment, one or more of the remaining 15 words may be in a nonexistent portion of memory for systems having less than 16 megabytes.

The parameters mentioned above are defaulted to test only the first megabyte of memory from 10000_{16} to 100000_{16} (byte address) while incrementing the exchange address by 16 words after each exchange.

Specifications for executive state indicate that the exchange address passed to the CP by the PP may be limited to 18 bits. Some processors may therefore fail to pass this test if PARAM18 is set for the third megabyte or higher.

Refer to the section description for test section 2 for other information.

Section 17 - OS Bounds Register Test (SECT 17)

This section tests the OS bounds register of the IOU by use of PP 2610 (MAN) exchange requests. The exchange address is set up so that on even conditions the exchange address is at the first word address immediately below the out of bounds region. On the following odd condition, the address is set for the next word and should cause the out of bounds condition.

On the even condition the test checks to ensure that the Cyber 170 exchange package swap occurs in the normal manner. On the odd conditions the exchange should be inhibited and the test checks to ensure that the exchange package at the specified address remains unchanged.

Each pair of conditions sets up the same address in the bounds register but alters the exchange address and the location of the exchange package by one word. A series of pairs of conditions sets up the bounds register addresses so that they occur in a sliding ones pattern. A subseqent series uses a sliding zeros pattern. The addresses cover the range from byte address

10000_{16} up to 16 megabytes using 28 pairs or 56 conditions. Specific conditions may be skipped if PARAM18 indicates that certain one-megabyte segments of memory are not to be tested. Refer to section 3.3.1 for additional data on the use of PARAM18. Note that in some conditions, the first word of the Cyber 170 exchange package may reside in one particular one-megabyte segment of memory while the remainder of the package may reside in the following one-megabyte segment. These particular conditions will be skipped unless the bits of PARAM18 enable the testing of both segements.

Specifications for executive state indicate that the exchange address passed to the CP by the PP may be limited to 18 bits. Some processors may therefore fail to pass this test if PARAM18 is set for the third megabyte or higher.

In all conditions of this section, bit 60 of the IOU DEC register is set to enable OS bounds detection. The enable error stop bit (bit 63) which causes the PP to be idled in the event of an error is not set. Under these conditions, the bounds error will cause the PP making the exchange request to either hang waiting for an exchange accept (model 825 type PPs) or simply exit the 2600 command (model 835 and 855 type PP's). In even numbered conditions, PP operation should continue normally; in odd numbered conditions the control PP will automatically prompt the CP to issue an exchange accept when a PP hangs (no hang message occurs). This will allow recovery of an model 825 type PP.

The reader may refer to the description of test section 2 for additional details since sections 2 and 17 are similiar in many respects.

SUBSECTION 1 - COMPARISON CHECKS, EVEN NUMBERED CONDITIONS

CMP.01 Checks that the P in the RF dump is at the HALT command in the executive state monitor process. A program stop (PS) in 170 monitor mode causes a executive state exchange to executive state monitor.

CMP.02 Checks the P value in the exchange pack at the test address. Since the 2600 command causes a package swap, this P should point to the count-down loop where the processor was awaiting the arrival of the 2600. This is a block comparison which also checks the Cyber 170 A, B and X registers in the 170 pack at the test address. The values which are there are as a result of the swap-out caused by the 2600 command. These values were loaded into the processor registers when it initially entered the 170 mode after the executive state EXCHANGE command.

SUBSECTION 1 - COMPARISON CHECKS, ODD NUMBERED CONDITIONS

CMP.01 The Cyber 170 exchange package at the test address in CM is checked to ensure that it has not been accessed. This package is set up with P, RAC, FLC and MA, all other registers in the package should be zeroed out.

SUBSECTION 2 -

This subsection is similiar to the previous except that a 2600 (EXN) command is employed; the hardware should react in the same manner. In order to speed exection, all even numbered conditions are skipped (ie: not executed).

SUBSECTION 3 -

This subsection uses the 2620 (MAN) command and skips all even number conditions. Both the exchange address used by the PP and the address in the MA register of the CP are the same out of bounds addresses. Since the CP should use MA instead of the PP exchange address and since the OS bounds register has no effect upon a CP address the swap of the 170 exchange package should occur in the normal way. The comparison checks performed in this subsection are therefore the same as those specified for the even numbered conditions of subsection 1.

Section 18 - State Switch With X0 Sign Bit (SECT 18)

Subsection 0 is used for initialization only. Subsection 1 has two conditions which test a state switching mechanism which employs the CYBER 170 X0 register sign bit. In both conditions the X0 sign bit is set and a CEJ/MEJ (op code 013_8) command is executed. In condition 0 where the CP is operating in the 170 monitor mode, the state switch to executive state monitor mode should occur. In conditon 1 (170 job mode), no switch should occur. However, a 170 exchange package swap is the normal result of executing the 013 command and the processor should commence execution in 170 monitor mode.

In condition 0, although a 170 package swap occurs, the processor should not start execution of the new 170 process. Instead, the 170 job environment is immediately exchanged out to the executive state job package in memory and the processor initiates the executive state monitor process. This executive state type exchange is the direct result of the X0 sign bit being set while an 013 command is executed in 170 monitor mode.

CMP.01 This compare checks P in the 170 exchange to A pack. In condition 0 it should point to the command following the 013 command.

CMP.02 This compare checks the P in the exchange to MA pack. In condition 1 where the processor is in 170 job mode the swap initiated by the 013 is directed to this package and the P should point to the command following the 013 command.

CMP.03 This compare checks RA. In condition 0, no error exit occurs and RA should remain 0. In condition 1, a 170 PS command is executed in the 170 process swapped in by the 013 command and RA should point to this command.

CMP.04 The P contained in the executive state job package is checked. In condition 0, the 013 swapped in the 170 job process which was then immediately exchanged out to the executive state job package. The P should therefore point to the start of the 170 job process.

CMP.05 The P in register file dump is checked. In both conditions, the processor finally halts at the same location in the executive state monitor process.

CMP.06 This compare checks the 170 monitor flag in the executive state job exchange package in memory. It should be clear for condition 0 and set for condition 1.

CMP.07 The system call bit (MCR bit 10) in the executive state job exchange package should be set as a result of the state switch in condition 0. This is a specific function of the state switch function. The bit should be clear in condition 1.

SECTION II-7

TRAP TEST

PROGRAM DESCRIPTION

GENERAL

This paragraph describes the FCTITC mechanism which controls TRAP. The test is normally assembled by the processor and resides in central memory during execution. FCTITC is assembled on the 16-bit Compass assembler and resides in a PP.

All information required by FCTITC to control TRAP is contained within the object code of TRAP and is read from central memory by FCTITC. This information is primarily a series of control commands which are interpreted by the PP to cause memory writes and compares and various other types of operations which affect the central processor. The TRAP source code contains a set of procedures which provides a convenient method of generating the control commands.

FCTITC contains a number of subroutines, one for each different control command, which perform the task required by a command according to it's specific address and data operands.

Structure

The data contained within the object code of TRAP is composed primarily of the following:

Central Processor Executable Code

This is the virtual machine code executed by the central processor. It consists of both executive state and Cyber 170 commands. Generally the code sequences are short and are followed by a HALT or PS (Cyber 170 program stop) command. FCTITC causes central processor execution to commence at the first command with machine registers containing predetermined data. When FCTITC detects the HALT command, it causes the contents of the machine registers to be dumped to central memory. FCTITC then performs all comparison checks required by TRAP using the data dumped to central memory along with data already in central memory.

Exchange Packages

A number of job and monitor exchange packages are used in the TRAP test. They are set up at assembly time to suit requirements of specific sections of the test. Each subsection of the test has control commands executed by FCTITC modify the original package with values specific to the subsection.

When an error occurs, the location of the job and monitor packages are displayed. In those conditions where the test sequence operates in executive state monitor mode, the Bn field of P in the job package is zeroed out. If the job process is entered in error, the processor should halt immediately.

Control commands are provided so that FCTITC can set up the MPS exchange package pointer register. The executive state monitor process usually sets up the JPS pointer register using a CPYXS command.

Section and Subsection Address Tables

These tables are compiled into TRAP for use by FCTITC. They consist of a series of addresses, one for each section/subsection of TRAP. Each table entry is a 64-bit word containing a right justified real memory byte address pointing to a section/subsection of the test. The table entries are ordered according to the section/subsection number. The last entry in the table is a word of FFF---F_{16} which indicates the end of the section/test. The section table must reside in central memory commencing at hex byte address 4000. The entries in the section table point to the subsection address table for the section while the subsection address table entries point to the control command tables for the subsection.

Control Command Tables

Control command tables constitute the bulk of the object code of TRAP. They contain control commands which are interpreted and acted upon by FCTITC. The tables are divided into subsections commencing at the central memory addresses pointed to by the entries of the previously described subsection address table. The subsections are in turn divided into segments called conditions, where a single segment consists of the control commands for a single condition of the test.

The first segment (condition) of a subsection commences at the beginning of a subsection control table. The first 16-bit word of every segment contains the segment length. This length is a nonzero count of the number of 64-bit words in the segment, including the segment length word itself.

Each segment commences on a 64-bit word boundary. There should normally only be 0, 1, 2, or 3 unused 16-bit words between segments. FCTITC uses the segment length to determine the end of a condition and the central memory address of the segment for the next condition. The segments (conditions) of TRAP are not individually addressable (as is the case for the sections) and therefore they must be executed sequentially.

Execution Sequence

To execute a single condition of TRAP, FCTITC references the section and subsection address tables to obtain a pointer to the first condition of a section. FCTITC then transfers the complete control table for a condition from central memory into its own PP memory. Only when this condition is complete will the segment for the next condition be obtained. The segment is not transferred repeatedly for repeat condition or scope mode operation.

FCTITC saves the segment length and then interprets each of the control commands in the current segment. The segment length is used to determine if all commands in the segment have been executed.

In a typical condition of a test, control commands are used to write data in various locations in memory. For example, to enter a data pattern in an X register of an exchange package or to zero out a memory cell in which the command to be tested will subsequently enter data. One or several write commands may be executed in order to set up the required test conditions.

Subsequently an INITIATE control command will be used to deadstart the CP and cause the execution of the test sequence. Prior to the deadstart, the control PP sets a deadstart flag at CM location 0 and loads the MPS, JPS, PSM, PTL, PTA, and SIT with their previously defined data. After the deadstart, the CP will usually enter the monitor process and execute initializing code. When this is complete, the CP clears the deadstart flag which advises the PP that the CP is about to execute the test sequence. The PP then continuously scans the status summary halt flag waiting for the CP to complete the sequence. When the halt flag sets or when the PP determines that the CP is hung up, the processor S and P registers are read for later display and a deadstart is transmitted which will cause the CP register file to be dumped to central memory. This completes the functions performed by the INITIATE control command.

The INITIATE command will be followed by one or more COMPARE control commands which will check the results. Comparisons of data in exchange packages or various memory locations may be performed. An error display will occur for each comparison error. (Note that one block comparison can cause only one error display.)

If the operator calls for a condition to be repeated, all of the control commands within the segment (writes, initiate, and compares) are repeated. If a scope loop is set up by the operator, only the INITIATE control command is repeated on each pass of the scope loop. This is necessary in order to provide a high repetition rate scope display. The test itself is designed in such a manner that critical data is automatically refreshed on each loop. In some cases this occurs as a result of loading an exchange package via the deadstart operation and subsequently dumping the registers rather than exchanging back to the previously loaded package location.

During each execution of the INITIATE command, FCTITC initializes a counter and subsequently decrements the count while it is simultaneously testing to determine if the processor has halted. If the counter reaches zero an error display occurs indicating that the processor is hung. Some processor hardware conditions (example: MAP turned off) may cause a hang condition to appear since the processor is taking longer to execute the instruction sequence. In this case, PARAM16 may be increased to eliminate this problem. During a scope loop where a processor is hung, the counter will cause the INITIATE operation to be repeated whenever the count becomes zero. The counter is initialized by loading it from PARAM16. Setting it to an excessively high count may increase execution time and cause a less intense scope loop trace.

SECTION DESCRIPTIONS

This portion of the manual provides a description of each section of the test.

Test sections are organized as follows:

Section	Type of Test
00 and 01	MCR and UCR bit tests.
02 through 17	MCR function tests
18 through 33	UCR function tests
34 and 35	Multiple interrupt tests
37 through 39	Model dependent processor tests
36	Reserved for future use

Section descriptions give purpose and operation and may include comments on the differences between subsections and conditions.

NOTE

The first subsection of every section is used exclusively for initialization. A description of this subsection (subsection 0) is given below (Section Initialization) and is not repeated for each section.

Each condition of a subsection makes one or more comparisons. In most cases, the corresponding comparisons in all conditions of a subsection are checking the same function or data item. A comparison number, CMP.XX, and a brief description is given for most comparisons. The number corresponds to the COMPARE NO. on the test error display. Numbering of comparisons commences with number 1 in each new condition.

Section Initialization

The first entry in every section's subsection address table points to the same subsection code. This code is therefore repeated as each subsection commences. The tasks performed include initialization related to the first execution of the test as well as cleaning up conditions related to the execution of the previous subsection. Examples of the latter include clearing the processor test mode (PTM) and the CM bounds registers, setting the SIT and PIT registers for maximum count and clearing SECDED errors which have been set up in any memory cells.

This subsection also performs initialization of the processor virtual environment. This initialization includes the loading andchecking of the PSM, PTL, and PTA registers and the writing into CM of the page and segment tables at their appropriate locations.

The page table maps virtual addresses one to one into the real memory address space for 292K bytes of memory. The page size is 64K bytes. The page table is located at 2000_{16} (byte address) and is 8K (2000_{16}) bytes long. Only a few entries are actually made in the page table.

Some system registers are loaded with required values as well as being checked after loading. In these cases, the loading and checking of a specific register is confined to its own condition. If a comparison error occurs, check to ensure that the microcode has been loaded. If it has been loaded, then other tests should be performed to check for proper operation of the register. The error display will provide the register address. A scope loop can be set up by repeating the condition with the display disabled (SRC,CSE,SDR). A scope loop cannot be achieved using the SSM command as in other conditions of the test.

SECT 00 - MCR Bit Tests

This section performs various tests by setting MCR bits and checking the processor's response. In most cases, the bits are already set in the MCR register when the monitor or job exchange package is initially loaded. The required mask bits in the MMR are also set in this manner. Tests are performed with traps enabled and disabled, in job and monitor modes, and with the mask bits set and clear.

Subsections 1,4,5, and 9 test all 16 MCR bits (1 bit per condition). In the remaining subsections, tests for MCR bit 10 (system call) and MCR bit 15 (trap exception) are omitted. Within all subsections, the condition number corresponds to MCR bits 00 through 15. The bits are tested in sequence with an extra condition counter increment and a skip of the test for those cases where a bit is not tested. Note that for model 845 and 855 processors only, the testing of MCR bit 00 is not performed in any subsection of section 0.

Comparisons performed in this subsection will include either four or five of those described below. Comparisons are performed in the order specified.

CMP.01 The P in the register file dump is checked to ensure that the processor halted at the expected location.

CMP.02 The status of the MCR bits are checked in the job exchange pack (if the test is expecting an exchange interrupt) or on the stack (if the test is expecting a trap interrupt).

CMP.03 This comparison checks to determine whether or not the process (job or monitor) completed execution. Usually the CP sets up a flag in a register to provide this indication.

CMP.04 The expected value of P is checked in the job exchange pack (for job mode tests) or on the stack (for monitor mode tests).

CMP.05 The status of the MCR register in the processor after a trap interrupt is checked. After the trap the processor should execute a HALT command in the trap routine. As a result, the control PP causes a register file dump. It is the MCR bits from this dump which are checked.

The following table summarizes characteristics of the individual subsections.

Subsec	Mode	Traps	Mask Bits	MCR Bit Set By	Action
00_0					Initialization
00_1	Job	Enabled	All set	Exchange	Exchange
00_2	Job	Enabled	All set	MAC	Exchange
00_3	Job	Disabled	All but 1 set	Exchange	Halt or stack
00_4	Job	Disabled	Same bit set	Exchange	Exchange
00_5	Mon	Enabled	Same bit set	Exchange	Trap
00_6	Mon	Enabled	Same bit set	BRCR cmd	Trap
00_7	Mon	Enabled	All but 1 set	Exch,all bits	Trap
00_8	Mon	Enaabled	All but 1 set	Exchange	Halt or Stack
00_9	Mon	Disabled	Same bit set	Exchange	Exchange

SUB00_1 Job mode, traps enabled. MCR bit set from the exchange pack. All mask bits set.

The processor should exchange to the monitor mode immediately without executing any part of the job process.

SUB00_2 Job mode, traps enabled. MCR bit set via MAC channel. All mask bits set.

The processor enters and waits in the job process until the MCR bit is set by the MAC after which it should immediately exchange to the monitor mode.

SUB00_3 Job mode, traps disabled. MCR bit set from the exchange pack. All mask bits except the corresponding bit are set.

No interrupt should occur in this test. Bits which are defined to cause a halt for the specified condition should cause a halt immediately upon entry to the job mode. Bits which stack should execute the job process to completion.

SUB00_4 Job mode, traps disabled. MCR bit set from the exchange pack. Corresponding mask bit set.

In this test an exchange to the monitor mode should occur immediately upon entry to the job process. No job process commands should be executed.

SUB00_5 Monitor mode, traps enabled. MCR bit set from the exchange pack. Corresponding mask bit set.

In this test a trap should occur immediately upon entry to the monitor mode without the execution of any commands in the process pointed to by the P of the monitor pack.

SUB00_6 Monitor mode, traps enabled. MCR bit set by BRCR command. Corresponding mask bit set.

In this subsection, the processor enters the monitor mode and subsequently executes a BRCR command. This sets the MCR bit and thereby causes the trap to occur. The resulting p on the stack should point to the branch address of the BRCR command.

SUB00_7 Monitor mode, traps enabled. All MCR bits set from exchange pack. Only the mask bit for the bit being tested is clear. MCR bits 0,10 and 15 are not tested in this subsection.

In this test the MCR bits laid down in the stack (as a result of a trap) are checked. Since all MCR bits and all but one mask bit are set initially, then all but one of the MCR bits should be clear on the stack when a trap occurs.

SUB00_8 Monitor mode, traps enabled. MCR bit set from the exchange pack. All mask bits except the corresponding mask bit set.

Bits which are defined to halt for the specified condition should halt immediately upon entry to the monitor mode. For MCR bits which stack, a subsequent CPYXS command sets all mask bits and a trap interrupt should occur. The P value pushed onto the stack should point to the command following the CPYXS command.

SUB00_9 Monitor mode, traps disabled. MCR bit set from the exchange pack. Corresponding mask bit set.

Bits which are defined to halt for the specified condition should halt immediately upon entry to the monitor mode. For MCR bits which stack, a subsequent CPYXS command enables traps and a trap interrupt should occur. The P value pushed onto the stack should point to the command following the CPYXS command.

SECT 01 - UCR Bit Tests

This section performs various tests by setting UCR bits and checking the processor's response. In most cases, the bits are already set in the UCR register when the monitor or job exchange package is initially loaded. The required mask bits in the UMR are also set in this manner. Tests are performed with traps enabled and disabled, in the job and monitor mode and with the mask bits set and clear.

Each of the 16 UCR bits are tested, one bit per condition, with the condition number in each subsection corresponding to the UCR bit number (bits 00 through 15).

Comparisons performed in this subsection will include either four or five of those described below. The comparisons are performed in the order specified.

CMP.01 The P in the register file dump is checked to ensure that the processor halted at the expected location.

CMP.02 The status of the UCR bits entered on the stack by the trap interrupt are checked.

CMP.03 This comparison checks to determine whether or not the process (job or monitor) completed execution. Usually the CP sets up a flag in a register to provide this indication.

CMP.04 The value of P entered on the stack by the trap interrupt is checked.

CMP.05 The status of the UCR register in the processor after a trap interrupt is checked. After the trap, the processor should execute a HALT command in the trap routine. As a result, the control PP causes a register file dump. It is the UCR bits from this dump which are checked.

The following table summarizes characteristics of the individual subsections.

Subsec	Mode	Traps	Mask Bits	UCR Bit Set By	Action
01_0					Initialization
01_1	Job	Enabled	All but 1 set	Exchange	Stack
01_2	Job	Enabled	Same bit set	Exchange	Trap
01_3	Job	Enabled	Same bit set	Exch,all bits	Trap
01_4	Job	Enabled	All clear	Exchange	Trap
01_5	Job	Disabled	Same bit set	Exchange	Exch. or stack
01_6	Mon	Enabled	Same bit set	Exchange	Trap
01_7	Mon	Enabled	All set	MAC	Trap
01_8	Mon	Enabled	Same bit set	BRCR cmd	Trap
01_9	Mon	Disabled	Same bit set	Exchange	Halt or stack

SUB01_1 Job mode, traps enabled. UCR bit set from the exchange pack. All mask bits except the corresponding bit are set.

Since UCR bits 00 through 06, are permanently set, these particular bits should trap immediately after the exchange to the job mode and before any job process command is executed. Other bits should stack and therefore should execute the job process to completion.

SUB01_2 Job mode, traps enabled. UCR bit set from the exchange pack. Corresponding mask bit set.

In this subsection a trap should occur for all bits immediately after the exchange to the job mode and before any job process command is executed.

SUB01_3 Job mode, traps enabled. All UCR bits are set from the exchange pack. Only the mask bit for the bit being tested is set.

In this test the bits in the UCR register are checked after the trap operation has occurred. All bits should be set except the bit corresponding to the mask bit which was set.

SUB01_4 Job mode, traps enabled. UCR bit set from the exchange pack. All mask bits clear.

UCR bits 00 through 06 for which the mask bits are permanently set, should trap immediately upon entry to the job mode. Bits which are defined to stack for the specified conditions should subsequently execute a CPYXS command which sets all mask bits. When this occurs the processor should trap. The P value entered on the stack should point to the command following the CPYXS command.

SUB01_5 Job mode, traps disabled. UCR bit set from the exchange pack. Corresponding mask bit set.

For bits which are defined to exchange for the specified condition, an exchange interrupt should occur immediately upon entry to the job mode without any command in the job process being executed. For UCR bits which stack, a subsequent CPYXS command enables traps with the result that a trap should then occur. The P value entered on the stack should point to the command following the CPYXS command.

SUB01_6 Monitor mode, traps enabled. UCR bit set from the exchange pack. Corresponding mask bit set.

In this subsection a trap should occur immediately upon entry to the monitor mode and without execution of any command in the monitor process.

SUB01_7 Monitor mode, traps enabled. UCR bit set via the MAC channel. All mask bits set.

In this test the processor enters the monitor mode and waits for a trap to occur as a result of the UCR bit being set by the MAC. The P value entered on the stack should point to the command where the processor was waiting for the trap to occur.

SUB01_8 Monitor mode, traps enabled. UCR bit set by a BRCR command. Corresponding mask bit set.

In this subsection, the processor enters the monitor mode and subsequently executes a BRCR command. This sets the UCR bit and thereby causes the trap to occur. The resulting p on the stack should point to the branch address of the BRCR command.

SUB01_9 Monitor mode, traps disabled. UCR bit set from the exchange pack. Corresponding mask bit set.

Bits which are defined to halt for the specified condition should halt immediately upon entry to the monitor mode.

For UCR bits which stack, a subsequent CPYXS command enables traps and a trap interrupt should occur. The P value pushed onto the stack should point to the command following the CPYXS command.

MCR/UCR FUNCTION TESTS

Most UCR/MCR function tests have four conditions in each subsection. Exceptions are noted. In all four conditions the mask bit which corresponds to the MCR/UCR bit being tested will be set. The four conditions are:

Condition	Function
0	Job Mode, Traps Enabled
1	Job Mode, Traps Disabled
2	Monitor Mode, Traps Enabled
3	Monitor Mode, Traps Disabled

After initial deadstart in each condition, the CP monitor process will perform various initialization tasks. When this is complete, the CP sets up the trap enable and trap enable delay flip-flops using data entered in a monitor process X register by the PP prior to the deadstart. A bit in the same register allows the CP to determine if the test sequence is to be executed in the monitor or job mode. The actual test sequence is the same for either of these cases. A separate test sequence employed for each MCR/UCR bit consists of approximately 10 to 20 commands.

If the sequence is to be executed in the monitor mode, the CP will branch directly to the sequence. If job mode is required, the JPS register and the job package will already have been set up and the CP will perform an EXCHANGE to branch to the test routine.

Depending the condition being executed and any hardware fault that may be present, the processor may trap or exchange out of the test sequence as a result of the interrupt. A common routine is used for all MCR/UCR bits when a trap or exchange occurs. If the normal response of the CP is to halt as a result of the interrupt condition, the CP will halt in the test sequence. In cases where the response to the interrupt condition is stack, the test sequence contains code which will subsequently enable traps. The fact that the CP does in fact stack can be readily determined by checking the value of P on the stack frame when the trap occurs. The trap interrupt and exchange routines all execute a HALT command advising the control PP that the sequence is complete.

Some conditions in sections 2, 4, 16, 18, and 19 operate in the CYBER 170 mode. Sections 2, 4, and 16 check that hardware errors related to the specific MCR bits are handled correctly in CYBER 170 mode. Section 18 checks the operation of the CYBER 170 command referred to as RT (op code 17_8). Section 19 checks that the unimplemented CYBER 170 compare/move commands are correctly recognized as illegal. The test sequences for these commands operate in executive state job mode with the VMID=1 and the CYBER 170 monitor flag set. When the appropriate interrupt occurs, the CP should execute an exchange interrupt back to executive state monitor mode.

If these conditions operate correctly, no CYBER 170 exchange should occur and no exchange package should be required. However, in the CYBER 170 process, the MA register is set up for the unexpected case. If a CYBER 170 swap does occur, the new process will execute a program stop command (PS, CYBER 170 op code 00_8).This should return the processor to executive state monitor mode (also via an exchange interrupt).

Note that all conditions which execute in CYBER 170 mode will be skipped automatically if the processor being tested does not have 170 capability.

One or more of the following types of comparison checks are performed. Except where noted otherwise, the checks performed in any given condition are similar for all conditions within the subsection.

(a) In all cases the untranslatable pointer (UTP) is checked to ensure it is set to correct value.

(b) A check is performed to ensure the test sequence has been executed correctly. This usually consists of checking value of P in:

1) the register file dump,

2) the job or monitor exchange pack or

3) the stack frame.

This establishes that PVA in P at the time of the interrupt is according to specifications. In some subsections, flags set or cleared by the processor are also checked to ensure that the test sequenced correctly.

(c) The register file dump or Job exchange package is checked to ensure that MCR or UCR bits have been set correctly (whichever applies).

(d) Checks specific to individual subsections are performed to ensure that commands which generate interrupts either do, or do not complete execution according to specification.

The MCR and UCR function tests perform, in order, the comparisons specified below. One or more comparisons concerned with checking execution as mentioned in (d) above may be performed in addition to those listed.

MCR FUNCTION TEST COMPARISONS - SECTION 2 to 17

CMP.01 Checks the untranslatable pointer.

CMP.02 Checks P in register file dump (where processor halts).

CMP.03 Checks value of P on stack. Any entry indicates a trap occurred.

CMP.04 Checks P in job exchange package.

CMP.05 Checks status of MCR bits after interrupt.

UCR FUNCTION TEST COMPARISONS - SECTIONS 18 to 33

CMP.01 Checks untranslatable pointer.

CMP.02 Checks P in register file dump (where processor halted).

CMP.03 Checks value of P on stack or in job pack depending upon whether trap or exchange interrupt is expected.

CMP.04 Checks status of UCR bits; may also be found either on stack or in job exchange pack.

SECT 02 - MCR 00 - Detected Uncorrectable Error

This section employs a model independent method of generating the uncorrectable error interrupt. The interrupt is caused by the processor writing to a central memory cell which has been defined as out of bounds by the CM bounds register.

One or more of sections 36 and following may also be concerned with uncorrectable errors. These sections are for model dependent tests for individual central processors.

SUB02_1

Before initiating program execution by the CP, the control PP sets up the CM bounds register so that memory beyond a certain address is out of range. This occurs in the first condition (condition 0). In each of the four subsequent conditions, the CP first writes a word of data at the location immediately below the boundary; subsequently it attempts to write data at the next higher location. The second write operation should generate the interrupt. The four conditions executed are: job and monitor mode with traps enabled and disabled. In all cases, a check is performed to ensure that the in bounds write executes and that the out of bounds write does not execute.

SUB02_2

This subsection is similar to the previous. Condition 0 sets up the bounds register. Conditions 1 and 2 operate in the job mode with traps enabled and disabled respectively and with executive state operation set up via the VMID. The uncorrectable error interrupt should cause an executive state exchange to monitor mode. The Cyber 170 monitor flag will be set; if the interrupt fails to occur, the CP will finally execute a Cyber 170 program stop command. This should cause P to be written to RA and a Cyber 170 error exit sequence. This sequence will also result in an executive state exchange to monitor mode.

SECT 03 - MCR 01 - Not Used

This section is reserved for possible future use in the event that an error or warning function is assigned to MCR bit 01.

SECT 04 - MCR 02 - Short Warning

This section runs only if the operator sets the section select bit for section 4. The default value for this bit is zero.

Execution of the test requires you to physically initiate the generation of the signal which causes the short warning interrupt. The signal must be artificially created without actually removing power from the machine. Reference to documentation concerning the Configuration Environment Module (CEM) may be helpful in this respect.

SUB04_1

This subsection has only one condition which operates in executive state monitor mode with traps enabled. When the subsection commences execution, the operator is forced to step through a series of single line messages as follows:

```
SHORT WARNING INTRUPT TEST - SPACE FOR NXT MSG
DO NOT POWER DOWN MACHINE - SPACE AGAIN
FIRST TEST IS IN 64 BIT MODE - SPACE
YOU MUST INDUCE WARNING SIGNAL - SPACE
ABS COMMAND TO SKIP SECTION - ELSE SPACE
SPACE THEN INDUCE SIGNAL
WAITING FOR INTERRUPT
```

If a short warning interrupt does not occur, the processor will continue in the waiting state and it will be necessary for the operator to dead-start the control PP and restart the test using the command buffer. If the required trap interrupt does occur, the processor will enter a trap routine and halt. Errors, if any, will then be displayed. If the test is successful it should proceed to the next subsection.

SUB04_2

This subsection is similar to the previous except that it operates in Cyber 170 mode (within executive state job mode). Note that the warning signal must have reset automatically to the normal level after the previous operation or it must be reset manually before proceeding with this second test. The following series of messages should occur:

```
LAST TEST IS IN CYBER 170 MODE - SPACE
ABS COMMAND TO SKIP SECTION - ELSE SPACE
SPACE THEN INDUCE SIGNAL
WAITING FOR INTERRUPT
```

Traps are enabled and an exchange from Cyber 170 mode to executive state monitor mode should occur when the warning signal is induced a second time.

SECT 05 - MCR 03 - Instruction Specification Error

SUB05_1

This subsection generates a length type instruction specification error using the ISOM (op code AC_{16}) command. The bit position designator is set to 63 and the length designator to 1 in order to cause the error.

SUB05_2

This subsection generates a length type instruction specification error using the ADDN (op code 70_{16}) BDP command. The F field of one of the data descriptors is set to 1 and the corresponding X register is set for a length of 39. An error occurs for type T (unpacked decimal) since the length should not exceed 38.

SUB05_3

This subsection generates a type instruction specification error using the ADDN (op code 70_{16}) BDP command. The T field of one of the descriptors is set to 9 (alphanumeric) which is an illegal type for BDP arithmetic commands.

SUB05_4

This subsection generates a monitor mode type instruction specification error using the BRCR (op code $9F_{16}$) command. The command attempts to set MCR bit 15 while operating in the job mode with traps enabled (condition 0) and traps disabled (condition 1). There are only 2 conditions in the subsection.

SUB05_5

This subsection generates a call instruction type instruction specification error using the CALLSEG (op code $B5_{16}$) command. The At field of the stack frame save area descriptor referenced by the command is set to a 1 to cause the error. This specifies the stacking of only the first two A registers. The At field should be at least 2 in order to stack 3 or more A registers.

SECT 06 - MCR 04 - Address Specification Error

SUB06_1

This subsection employs an SX (op code 83_{16}) command to generate an address specification error. The Bn field of the A register referenced by the command will contain an address which is not 0 modulo 8.

SUB06_2

This subsection employs a BRDIR (op code $2F_{16}$) command to generate an address specification error interrupt. The A register referenced by the command will contain the correct PVA; however, the X register will contain a 1 thereby specifying a branch to a memory address which is not 0 modulo 2.

SUB06_3

This subsection utilizes a CALLSEG (op code $B5_{16}$) command to generate an address specification error. The code base pointer referenced by the command contains a negative Bn field.

SECT 07 - MCR 05 - Exchange Request

The exchange request function is not tested by the TRAP test. This section is automatically skipped.

SECT 08 - MCR 06 - Access Violation

SUB08_1

A read access violation is generated in this subsection by use of an LX (op code 82_{16}) command. The A register of this command references a segment whose segment descriptor RP field defines the segment as nonreadable.

SUB08_2

A write access violation is generated in this subsection by use of an SX (op code 83_{16}) command. The A register of this command references a segment whose segment descriptor WP field defines the segment as nonwritable.

SUB08_3

This subsection generates an access violation using the BRDIR (op code $2F_{16}$) command. The A register of this command references a segment whose segment descriptor XP field defines the segment as nonexecutable.

SUB08_4

This subsection uses a CALLSEG (op code $B5_{16}$) command to generate an access violation. In this particular case, the ring number in the command's Aj register is greater than the ring number (CBP R3 field) in the code base pointer referenced by the command.

SECT 09 - MCR 07 - Environment Specification Error

SUB09_1

This subsection generates environment specification errors by setting up all unsupported combinations of the VMID field in the executive state exchange package. The unsupported combinations are 2 through 15_{10}. Checks are performed in both the job and monitor mode. The interrupt occurs immediately before the first command in the new process commences execution. There are 17 conditions in the subsection.

SUB09_2

This subsection generates an environment specification error by using a CALLSEG (op code $B5_{16}$) command which references a code base pointer having a VMID field set to one of the unsupported virtual machine IDs.

SUB09_3

This subsection is similar to the previous except that the VMID in error is contained in a stack frame which is loaded during execution of a RETURN (op code 04_{16}) command.

SUB09_4

This subsection generates an environment specification error by using the CALLSEG and RETURN commands. After execution of the CALLSEG command, the called procedure alters the contents of the A0 register in the stack frame. The subsequent RETURN command should generate the error. The RETURN's initial A2 (previous save area pointer) should equal the final A0 thereby verifying that the stack is being returned to the same state as prior to the CALLSEG.

SECT 10 - MCR 08 - External Interrupt

SUB10_1

This subsection checks the external interrupt function by use of the INTRUPT (op code 03_{16}) command which transmits interrupts to a processor via a central memory port. The interrupt is received by the CP connected to the specified port.

The subsection executes 4 conditions (4 through 7) where the interrupt is directed to the port to which the CP is connected. These conditions check the job and monitor modes with the traps enabled and disabled. Four other conditions (0 through 3) check the cases where no port is specified by the INTRUPT command or where the specified port is not the port to which the CP being tested is connected.

Normally the test will assume that the CP is connected to CM port 0. Another port may be specified by setting PARAM18 to the required value (1,2,4 or 8_{16} for ports 0 to 3 respectively).

SECT 11 - MCR 09 - Page Table Search

In this section single word reads and writes are attempted on the first word in page 2 (byte address 20000_{16} of central memory. The PP sets up a string of 32 or 33 page table entries at the corresponding hash address in the page table. These entries are such that the processor should search through 32 entries. In the first 2 subsections, the page table search interrupt should be generated, since none of the first 32 entries are satisfactory. In the last 2 subsections page table entry number 32 is good and no interrupt should occur. The first two subsections each have four conditions which try the combinations of job and monitor mode and traps enabled and disabled. The last two subsections use only one condition each and operate in the job mode with traps enabled.

SUB11_1

This subsection uses an LX (op code 82_{16}) command to generate a page fault. This command initially references a page descriptor where the V/C bits are marked valid/continue and the segment/page ID is not equal to the ASID/PN. The following 31 descriptors are all marked invalid/continue and the segment/page ID is equal to the ASID/PN. Descriptor number 33 is marked valid/stop and the segement/page ID is equal to the ASID/PN. The search of this string of descriptors should result in a page table search interrupt because the only good descriptor is immediately beyond the linear search range.

SUB11_2

This subsection uses an SX command to generate the interrupt. The first descriptor is the same as the set of 31 in the previous subsection and the following 31 are the same as the first of the previous section. Again in this case, the seach should fail because the only good descriptor is beyond the linear search range.

SUB11_3

This subsection is identical to subsection 11_1 except one of the block of 31 identical descriptors is omitted. The processor should now find the good descriptor on the search of the last entry of the 32 and no interrupt should occur. Comparison checks are performed .

SUB11_4

This subsection is similar on 11_2 with one of the block of 31 identical omitted. It should react in the same manner as subsection 11_3.

SECT 12 - MCR 10 - System Call

A function test is not provided for this MCR bit because its set or reset status does not cause any hardware action. Tests performed in section 00 (subsections 1,4,5 and 9) and section 34 (subsection 1 and 2) involve checks on this bit.

SECT 13 - MCR 11 - System Interval Timer

SUB13_1

In this subsection, the CP monitor process sets the SIT register to a relatively low value. Subsequently the CP enters a waiting loop while the SIT counter decrements and finally causes an interrupt. Four conditions are executed, two in the monitor mode and two in the job mode. For the job mode conditions, the processor sets up the SIT prior to the exchange to the job mode where it waits for the interrupt.

SECT 14 - MCR 12 - Invalid Segment/Ring=0

SUB14_1

This subsection generates an invalid segment interrupt by employing an SX (op code 83_{16}) command to access a segment whose segment descriptor is invalid (VL field=0). The invalid descriptor is loaded into the segment table by the PP prior to execution of the SX command.

SUB14_2

This subsection generates an invalid segment interrupt by employing an SX command which references an A register whose segment number field is greater than the length of the segment table minus one.

SUB14_3

This subsection will generate a ring=0 interrupt using the RETURN (op code 04_{16}) command. A CALLSEG command is used to set up a stack frame. The called procedure will contain code which will zero out the ring number field of one of the A registers in the stack. The interrupt should occur upon execution of the return command.

SUB14_4

This subsection will generate a ring=0 interrupt using the LA (op code 84_{16}) command. The four high order bits (ring #) of the 6 byte address loaded from central memory in to register Ak will all be zeroes which should cause the interrupt.

SECT 15 - MCR 13 - Outward Call/Inward Return

SUB15_1

This subsection will generate an interrupt using a CALLSEG (op code $B5_{16}$) command making an outward call. Specfically, the ring number of the initial P of the CALLSEG command is less than the R1 field of the segment descriptor of the called procedure. The test makes a call to a procedure in another segment whose segment descriptor R1 is larger than the P ring number normally used through out the test.

SUB15_2

This subsection will generate an interrupt using a RETURN (op code 04_{16}) command making an inward return. This is defined as the case where the final P ring number which results from execution of the command being less than the ring number in A2 before execution of the command. The test first calls a procedure (via a CALLSEG command) which establishes a stack frame. The procedure then modifies the ring number of the P in the stack frame so that it is lower than the ring number normally found in A2. The interrupt occurs during the subsequent execution of the RETURN command.

SECT 16 - MCR 14 - Soft Error Log

This section employs a method of generating a soft error log interrupt which is the same for all processors.

SUB16_1

Before initiating program execution, the control PP sets up a soft (correctable) error in a specific memory cell as follows: the CM environment control register is set for the write check bits mode, a single memory write copies data into the 8 error check code (ECC) bits as well as the memory word, the environment control register is then returned to the normal mode. The bits written into the ECC are such that a single bit error will be detected whenever that specific cell is read by the central processor code sequence. The subsection operates in the executive state mode. Four conditions are executed, job and monitor mode with traps enabled and disabled.

SUB16_2

This subsection is somewhat similar to the previous. It has two conditions (job mode with traps enabled and disabled) and operates in CYBER 170 mode with VMID=1. The soft error interrupt should cause an executive state exchange to monitor mode. The CYBER 170 monitor flag will be set; if the interrupt fails to occur, the CP will finally execute a CYBER 170 program stop command. This should cause P to be written to RA and a CYBER 170 error exit sequence which will also result in an executive state exchange to monitor mode.

SECT 17 - MCR 15 - Trap Exception

SUB17_1

In this subsection, the KEYPOINT (op code $B1_{16}$) command is employed to generate a trap exception condition. Two conditions are used to perform the test with traps enabled in the job and monitor modes. When the KEYPOINT trap occurs, the code base pointer which is accessed (to obtain the vector to the trap procedure) will have an invalid VMID field. This will generate an environment specification error and thereby also the trap exception condition.

The trap exception should inhibit the trap operation. In condition 0 (job mode), the exception condition should cause an exchange back to monitor mode. In condition 1 (monitor mode), an immediate halt of the processor should occur. The control PP checks that UCR 06 (keypoint) and MCR 07 (environment specification error) are set.

SECT 18 - UCR 00 - Priviledged Instruction Fault

SUB18_1

This subsection generates a privileged instruction fault using the INTRUPT (op code 03_{16}) command. Segment 3 is set up as a local privileged segment and a command sequence including the INTRUPT command is executed. After CP operation commences, the CP branches to the INTRUPT command via a BRDIR (op code $2F_{16}$) command which references an A register pointing to segment 3. The fault should be generated because the INTRUPT command requires global privilege and it is executed from a local privileged segment.

SUB18_2

This subsection generates a privileged instruction fault using the LPAGE (opcode 17_{16}) command. It's operation is similar to the previous subsection except that segment 3 will be defined as a nonprivileged executable segment. The fault should occur because the LPAGE command must be executed from a local or globally priviledged segment.

SUB18_3

This section generates a priviledged instruction fault using the Cyber 170 command RT. This command is executed In the Cyber 170 mode with traps enabled (conditions 0 through 3) and traps disabled (conditions 4 through 7). In the first 4 cases a trap interrupt occurs; in the second set of 4 an exchange interrupt occurs. Each set of the successive 4 conditions executes the command from parcels 0,1,2 and 3 of the 60-bit CP word. The RT command should always cause an interrupt; it is a normal method of entering executive state mode from a Cyber 170 process.

SECT 19 - UCR 01 - Unimplemented Instruction

SUB19_1

In this subsection, each unimplemented op code is tested to ensure that an interrupt occurs. For each condition, the control PP selects an illegal op code from a table and copies it into the CP's instruction sequence; the PP then initiates instruction execution by the CP. The table of unimplemented op codes is sequenced 4 times to check the 4 combinations of job/monitor mode with traps enabled/disabled.

The table of op codes includes every illegal op code for all processor types. The fourth parcel of each table entry defines the processors for which the associated op code is not illegal. Bits 0 (least significant or rightmost bit) through n of each entry are tested for processor types for each model. If the bit is set, then the test of the corresponding op code is skipped.

SUB19_2

This subsection checks that the CYBER 170 compare/move op codes 464_8 through 467_8 will generate an unimplemented instruction interrupt. The tests are performed in the executive state job mode with traps enabled (conditions 0 through 15) and disabled (conditions 16 through 31). Each set of 16 conditions checks the 4 different op codes in each of the 4 parcels of a 60 bit CP word. The executive state job process will have a VMID=1 and MF=1 thereby providing a CYBER 170 monitor process environment.

SECT 20 - UCR 02 - Free Flag

The free flag bit of the UCR is not activated by functional logic as is the case for most other MCR/UCR bits. Instead, it is employed by a monitor process as a signal to a job process. The bit (if required) would be set in the job exchange package in memory prior to an exchange to job mode. Since all UCR bits are tested in exactly this manner in section 1, a function test is not provided for the free flag.

SECT 21 - UCR 03 - Process Interval Timer

SUB21_1

In this subsection, the processor loads the PIT counter with a relatively small value by using the CPYXS (op code $0F_{16}$) command. The processor then enters a loop and waits for the timer to count down to zero and cause the interrupt.

SECT 22 - UCR 04 - Inter-Ring POP

SUB22_1

In this subsection, the POP (op code 06_{16}) command is used to generate the inter ring pop interrupt. A CALLSEG command makes a call to a procedure which operates in the current ring (ring B_{16}). When this procedure is entered, the ring number of the then current A2 register is modified from B_{16} to A_{16}. Subsequently the POP command is executed and should generate the interrupt as a result of the rings of the initial P and A2 registers not being equal.

SECT 23 - UCR 05 - Critical Frame Flag

SUB23_1

The critical frame flag is a single bit process register that is stored in and retrieved from a stack frame by the CALL and RETURN commands. The flag is normally set in the current process by the software using the CPYXS command. Regardless of the means by which the CFF is set, a trap interrupt is generated if a RETURN or POP command is executed when the flag is set in the currentprocess.

In this subsection a CALLSEG command calls a procedure in which a CPYXS command sets the CFF. This procedure subsequently attempts to execute a RETURN (op code 04_{16}) command which should generate the required interrupt.

SUB23_2

This subsection is identical to SUB23_1 except that rather than attempting the execution of a RETURN command, a POP (op code 06_{16}) command is employed.

SECT 24 - UCR 06 - Keypoint

SUB24_1

This section checks the capability of KEYPOINT (op code $B1_{16}$) to generate a trap interrupt. The performance monitoring facility is not employed.

Upon entry to the process, the keypoint enable flag (KEF) is loaded to the set condition as a result of an entry made in the exchange package by the control PP. The CP sets the keypoint mask register (KMR) via a CPYXS command to specify a specific class. Subsequently, the KEYPOINT command is executed with its j operand specifying the same keypoint class; this should result in the generation of the interrupt. The control PP checks that UCR bit 06 is set and that the correct keypoint class and code numbers are set up by the processor.

SECT 25 - UCR 07 - Divide Fault

SUB25_1

In this subsection, the divide fault interrupt is generated by the use of the integer quotient command DIVX (op code 27_{16}). Register Xk is loaded with a test pattern which is a positive nonzero value. Register Xk is loaded with zeroes and the DIVX command is executed causing the interrupt. Register Xj should subsequently contain the original nonzero value.

SUB25_2

This subsection checks the divide fault interrupt using the BDP decimal quotient command DIVN (op code 73_{16}). The k and j descriptor field lengths are nonzero values and the operands themselves are both type 4 (unpacked decimal unsigned). The k operand is a test pattern and the j operand is all zeroes. After execution of the DIVN command, the destination field (k operand) should not be altered regardless of whether the traps are enabled or disabled.

SUB25_3

A divide fault interrupt is generated in this subsection using the floating point quotient command DIVF (op code 33_{16}). The Xk operand has a floating point value of plus 100_{10} and the Xj operand is a nonstandard value of zero (all zeroes). The CP instruction sequence will load the operands and then execute the DIVF command to generate the divide fault condition. The destination register should never be altered when this condition occurs.

SECT 26 - UCR 08 - Debug

SUB26_1

Since the debug feature is only active when traps are enabled and the mask register bit for UCR 08 is set, this subsection has only two conditions. One for a debug trap in executive state job mode (condition 0) and the other for monitor mode (condition 1).

The debug interrupt is generated by a data write operation sensed by the debug hardware when an SX (op code 83_{16}) command is executed. Prior to execution, the PP test controller will set up the debug list pointer in the executive state exchange packages (job and monitor). The PP also sets up the debug list which contains two double word entries which each bracket 2 central memory words. The second entry has the debug code bit set for end of list. Both entries will otherwise only have the data write code bit set.

When CP execution commences, the proper mode (job or monitor) will be selected. The CP will then clear the debug index register, set the debug mask to detect reads or writes and enable traps. These three operations are performed by the use of CPYXS commands. Immediately thereafter, the processor will execute an LX and then an SX command to transfer data from one memory address to another. Both memory addresses are within the bracketed area. The LX command should not interrupt since the debug code bits are set for data write. The SX command should interrupt. The PP will subsequently check the debug index and also ensure that the SX has not altered memory.

SECT 27 - UCR 09 - Arithmetic Overflow

SUB27_1

In this subsection, the arithmetic overflow interrupt is generated by use of the ADDX (op code 24_{16}) integer add command. The operands used are both the same; a large positive number with a single bit set at bit 1 (262). The CP loads the operands from memory and immediately executes the the ADDX command which should cause the interrupt. For conditions 0 and 2 where the mask bit is set and the traps are enabled, instruction execution should be inhibit The destination register is checked to ensure that it contains the original operand. For conditions 1 and 3 with the traps not enabled, the destination register should receive the overflow result.

SUB27_2

In this subsection, the arithmetic overflow interrupt is generated by use of the ADDN (op code 70_{16}) BDP numeric add command. Both operands are of equal length and are type 4 (unpacked decimal unsigned). One operand is a string 9s and the other is a string of 1s. The ADDN command should generate an overflow result of 0s in the destination field for conditions 1 and 3 where traps are not enabled. In conditions 0 and 2 traps are enabled and the destination field should not be altered.

SECT 28 - UCR 10 - Exponent Overflow

SUB28_1

In this subsection, the floating point sum command ADDF (op code 30_{16}) is employed to generate an exponent overflow interrupt. Both operands ($4FFF8000---0_{16}$ are the same and have a normalized coefficient and the maximum possible positive exponent. The overflow result ($50008000---0_{16}$ transferred to the destination register is the same regardless of whether traps are enabled or disabled so long as the user mask bit is set. This is the case for all conditions. The trap interrupt will occur in conditions 0 and 2 where traps are enabled.

SECT 29 - UCR 11 - Exponent Underflow

SUB29_1

In this subsection, the floating point difference command SUBF (op code 31_{16}) is employed to generate an exponent underflow interrupt. The exponents of both operands are the same and have the smallest possible positive values. When the operand for Xj ($30004000---0_{16}$ is subtracted from Xk ($3000800---0_{16}$, the coefficient diminishes and is subsequently normalized with the result that the exponent underflows. The underflow result ($2FFF8000---0_{16}$ transferred to the destination register is the same regardless of whether traps are enabled or disabled so long as the user mask bit is set. This is the case for all conditions. The trap interrupt will occur in conditions 0 and 2 where traps are enabled.

SECT 30 - UCR 12 - FP Loss of Significance

SUB30_1

In this subsection, the floating point difference command SUBF (op code 31_{16}) is employed to generate an FP loss of significance interrupt. Both operands ($30008000---0_{16}$ are the same and have a normalized coefficient and the smallest possible positive exponent. The loss of significance result ($30000000---0_{16}$ transferred to the destination register is the same regardless of whether traps are enabled or disabled so long as the user mask bit is set. This is the case for all conditions. The trap interrupt will occur in conditions 0 and 2 where traps are enabled.

SECT 31 - UCR 13 - FP Indefinite

SUB31_1

In this subsection, the int sum command ADDF (op code 30_{16}) is employed to FP indefinite interrupt. Operand Xj is $+0_{16}$) and operand Xk is ($70000---00123_{16}$). The result of the addition transferred to Xk with traps enabled 0 and 2) should be the value in Xk. For traps onditions 1 and 3), the nonstandard FP number $+IND_{16}$) should be placed in Xk. The trap interrupt occur in conditions 0 and 2 where traps are enabled.

SECT 32 - UCR 14 - Arithmetic Loss of Significance

SUB32_1

In this subsection, the convert to integer command CNFI (op code $3B_{16}$) is employed to generate an arithmetic loss of significance interrupt. The floating point number to be converted (operand Xk) is a very large positive number ($4FFF8000---0_{16}$). The nonzero digit of the coefficient is well beyond the range of the most significant digit in the integer form. Thus, when traps are not enabled, the result transferred to Xk should consist entirely of zeroes. With traps enabled, the loss of significance condition inhibits instruction execution. The original operand in Xk ($0123---CDEF_{16}$) should be left unaltered and a trap interrupt should occur.

SUB32_2

The SCLN (op code $E4_{16}$) BDP decimal scale command is used in this subsection to generate an arithmetic loss of significance interrupt. The command descriptors define both operands as unpacked decimal unsigned (type 4) with a length of 8 digits. The source operand is a string of nonzero digits (1 through 8). The D field of the SCLN command specifies a shift of one place left and when executed should cause the interrupt condition since the destination operand would be truncated on the left. If traps are enabled (conditions 0 and 2), a trap interrupt should occur and the destination field is checked to ensure that it is unaltered. With traps disabled (conditions 1 and 3), the destination field should receive the truncated operand.

SECT 33 - UCR 15 - Invalid BDP Data

SUB33_1

This subsection uses the BDP numeric sum command ADDN (op code 70_{16}) to generate an invalid BDP data interrupt. Both operands are unpacked decimal unsigned (type 4) and are 8 digits long. The trap interrupt should occur for conditions 0 and 2 and the destination field should be unaltered. For conditions 1 and 3 where traps are not enabled, the result in the destination field is not checked since the result is not defined.

SECT 34 - Multiple Interrupt/Priority Checks

In this section, the processor's response to multiple interrupts is checked. In all cases, multiple bits are simultaneouly loaded into the MCR/UCR registers. This is accomplished by setting up the corresponding bits in the job or monitor exchange package in memory prior to initiating instruction execution.

SUB34_1

This subsection checks the priority of an exchange interrupt over a trap interrupt. The test is performed in the job mode with traps enabled and all mask bits set. In this mode all MCR bits should cause an exchange interrupt and all UCR bits would normally cause a trap interrupt. In all conditions, all UCR bits will be set. In any one condition, only a single MCR bit will be set. In 14 of the 16 conditions, an exchange interrupt should occur rather than a trap. After the exchange occurs, the P of the job pack in memory should point to the first command in the job process. Conditions 10 and 15 (MCR bits 10 and 15) should trap (as a result of the UCR bits) because these MCR bits do not cause any hardware action.

CMP.01 Checks P in the register file dump.

CMP.02 Checks P in the job exchange package.

CMP.03 Checks the status of the MCR bits in the job exchange package.

CMP.04 Checks the status of the MCR bits on the stack.

CMP.05 Checks the status of the UCR bits on the stack.

CMP.06 Checks that execution of the job process did not occur.

CMP.07 Checks P on the stack. Any entry indicates that a trap occurred.

SUB34_2

This subsection checks the priority of an exchange interrupt over a stack condition. The test is performed in the job mode with traps disabled and all mask bits set. In this mode, all MCR bits and 4 of the UCR bits should cause an exchange interrupt. The subsection has 20 conditions. In all conditions the other 12 UCR bits will be set. In any one condition, only a single one of the MCR/UCR bits which should cause an exchange will be set. The exchange interrupt should occur immediately after the initial exchange to job mode and without execution of any commands in the job process. Conditions 10 and 15 (MCR bits 10 and 15) exchange to the monitor mode as a result of an instruction specification error on a HALT command rather than as a result of MCR bits 10 or 15 being set.

CMP.01 Checks P in the register file dump.

CMP.02 Checks P in the job exchange package.

CMP.03 Checks the status of the MCR bits in the job exchange package.

CMP.04 Checks the status of the UCR bits in the job exchange package.

CMP.05 Checks whether or not job execution occurred.

CMP.06 Checks that no trap occurred.

SUB34_3

This subsection checks the priority of a halt condition over a stack condition. The test is performed in the monitor mode with traps disabled and all mask bits set. In this mode various MCR/UCR bits should either halt or stack. The subsection has 13 conditions. In all conditions, all bits which stack will be initially set. In any one condition only a single bit which halts will be set. In every condition, the processor should halt immediately upon initiation of instruction execution by the deadstart and before any instruction is actually executed. The P in the register file should remain pointing at the first command in the monitor process.

CMP.01 Checks P in the register file dump.

CMP.02 Checks a flag (in register Xe in the register file dump) to ensure that no monitor process execution occurred.

CMP.03 Checks P in the stack to ensure that no trap occurred (there should be no entry).

SECT 35 - MCR Bit Left in Memory

When performing a normal executive state exchange from monitor to job mode (with traps enabled in monitor mode), the hardware must not return the monitor package to central memory with an MCR bit set for which the corresponding mask bit is also set. Either the trap should occur, or the same MCR bit should be set up in the job process before the EXCHANGE completes execution. This section attempts to provide situations where this type of fault might occur. The test sets up asynchronous type interrupts so that they will occur shortly before or during the exchange from monitor to job mode.

SUB35_1

In this subsection, the system interval timer (SIT) interrupt (MCR 11) is employed for the test. A CPYXS (op code $0F_{16}$) command is used to load the timer with a relatively small count just prior to the execution of the EXCHANGE command. On each successive condition of the 50 conditions of the test, the count which is loaded will be incremented by one. The count which is used for the first 10 to 20 conditions will be such that the SIT interrupt will occur before the EXCHANGE to job. Perhaps 5 to 10 subsequent conditions will cause the timer to reach zero during the execution of the EXCHANGE while on the remaining 10 to 20 conditions, the timer will complete after entry to the job process.

The CP will check to determine if the MCR bit has been set in the monitor pack in memory and it will save the result for a PP comparison. Checks are performed to ensure that the SIT interrupt is received either before or after the exchange.

CMP.01 Checks P in the job exchange package.

CMP.02 Checks that no MCR bits were left in memory.

CMP.03 Checks that the SIT interrupt was received.

SUB35_2

This subsection is similar to the previous. However, instead of using a SIT interrupt, an external interrupt (MCR 08) is employed. The external interrupt is generated by using the INTRUPT (op code 03_{16}) command. The time of receipt of the interrupt via the central memory port is dependent upon the turn around time and any current memory activity. On each successive condition of the 16 conditons of the test, the INTRUPT command is moved closer to the EXCHANGE command until on the last condition it will immediately precede the EXCHANGE. In a few cases where the the two commands are close together, the actual interrupt will be received while the EXCHANGE is executing. Similiarly to the previous subsection, the CP code contains job process and monitor trap routines for processing the interrupt and checking if the MCR bit has been stored in memory.

Normally the test will assume that the CP is connected to CM port 0. Another port may be specified by setting PARAM18 to the required value (0,1,2 or 3 for ports 0 to 3 respectively).

CMP.01 Checks P in the job exchange package.

CMP.02 Checks that no MCR bits were left in memory.

CMP.03 Checks that the external interrupt was received.

SUB35_3

This subsection is similar to subsection 35_1. Rather than using a SIT interrupt, a soft error (MCR 14) generated by CM is employed. initially, a soft (single bit) error is set up in a specific memory cell by writing bad data during a write check bits operation (see also section 16). Subsequently, any read of the same cell using an LXI (op code $A2_{16}$) command will generate the soft error interrupt.

The LXI command is executed shortly before the EXCHANGE command. The error response from memory could arrive after execution of the EXCHANGE command has commenced and may possibly generate the situation where the MCR bit is stored in memory. Successive conditions (176 total) of the subsection move the LXI command closer to the EXCHANGE command until the later condition where it immediately precedes the EXCHANGE.

Since the cache (on some machines) loads data from adjacent memory addresses, additional timing variations are obtained by having the LXI command load data from addresses preceding the address which is actually in error. The previously described set of conditions are repeated with the Xi index of the LXI command modified to access the words 1, 2 and 3 words previous to the error.

CMP.01 Checks P in the job exchange package.

CMP.02 Checks that no MCR bits were left in memory.

CMP.03 Checks that the soft error interrupt was received.

SUB35_4

This subsection is similar to subsection 35_3 except that an uncorrectable error interrupt (MCR 00) is employed. A memory cell addressed by an SXI (op code $A3_{16}$) command is set out of bounds to generate the interrupt. The CM bounds register is set up so that the test address is at the first out of bounds location. Note that the bounds register must be set up for a specific memory port. Normally the test will assume that the CP is connected to CM port 0. Another port may be specified by setting PARAM18 to the required value (0,1,2 or 3 for ports 0 to 3 respectively). Condition number 1 has a single comparison only which checks for the proper set up of the bounds register. The comparisons in conditions 2 through 177 are similar to those of conditions 1 through 176 in subsection 35_3 and are as follows:

CMP.01 Checks P in the job exchange package.

CMP.02 Checks that no MCR bits were left in memory.

CMP.03 Checks that the uncorrectable interrupt was received.

SECT 36 - RESERVED

This section is reserved for use at a later date.

SECT 37 - Model 810/815/825/830 Processor Model Dependent Tests

This section performs interrupt tests utilizing hardware peculiar to these processors. The tests are related to the uncorrected error interrupt (MCR 00) and the soft error log (MCR 14). Subsections 1 and 2 are automatically skipped if the test is not executing on one of these processors.

The processors contain hardware which can inject an error in a specific circuit at a specific time. The hardware is controlled by two instruction level commands which are available when the processor is operating in executive state job mode with the DEC register test mode bit set. One command (op code $C9_{16}$) loads the processor test mode register and determines where the fault will be injected. This command also loads a delay counter which is decremented on each clock (50 nanoseconds). The fault is injected for one clock period when the count reaches zero. The second command (opcode CA_{16}) clears the PTM and the counter for normal operations. A command which transmits data via a selected circuit is inserted in the instruction sequence following the op code $C9_{16}$ command. The counter is loaded with a count which will become zero at the critical time.

NOTE

There is an uncertainty of one clock period regarding the injection of the error. To overcome this problem, the CP may repeat the test sequence using a modified delay (plus or minus one count) to ensure that an interrupt occurs. The processor will stop at a HALT command if this action fails to cause the interrupt.

The comparisons for subsections 37_1 and 37_2 are similar and are as follows:

CMP.01 Checks P in register file dump.

CMP.02 Checks P on stack; should be no entry indicating a trap did not occur.

CMP.03 Checks P in job exchange package.

CMP.04 Checks P in monitor exchange package.

CMP.05 Checks that correct MCR bit was set (bits 0 and 14 for subsections 37_1 and 37_2 respectively).

CMP.06 Checks that all commands in test sequence were executed.

CMP.07 Checks that uncorrected error bit (subsection 37_1) or corrected error bit (subsection 37_2) was set in CP status summary register.

SUB37_1

This subsection tests the uncorrected error interrupt. The instruction retry bit in the DEC register (bit 30) is set by the test. Some other bits of DEC are set by the test and include the test mode bit (bit 33), the PFS enable bit (bit 24) and the PFS trap bit (bit 06). The four conditions of the subsection operate in the job mode with traps enabled and force parity errors in 4 different data paths. The uncorrected error bit (bit 61) in the status summary register is checked.

SUB37_2

This subsection tests the soft error log interrupt. The test checks that the corrected error bit (bit 62) in the CP status summary register is set.

SECT 38 - Model 835 Processor Model Dependent Tests

This section performs interrupt tests utilizing hardware peculiar to this processor. The tests are related to the uncorrected error interrupt (MCR 00). Subsection 1 is automatically skipped if the test is not executing on this processor.

SUB38_1

In this subsection, an uncorrected error from the CP is generated by use of the processor test mode register (PTM). In this processor, the PTM register has a parity check network. Prior to initiating CP instruction execution, the control PP first sets the most significant bit in the PTM and then on a second operation, the PTM is written with a string of zeroes. The PTM will then contain a word with bad parity.

The test sequence executed by the processor generates the interrupt condition simply by executing a CPYSX command (op code $0E_{16}$) which copies the contents of the the PTM register to an X register. The interrupt should occur immediately with PVA in P pointing to the CPYXS command.

This subsection has four conditions which operate in the job and monitor mode with traps enabled and disabled. Comparisons for all conditions are as follows:

CMP.01	Checks for no entry in untranslatable pointer
CMP.02	Checks P in register file dump (where processor halts).
CMP.03	Checks P on stack.
CMP.04	Checks P in job exchange package.
CMP.05	Checks status of MCR bits after interrupt.

SECT 39 -

This section is reserved.

SECT 40 -

This section is reserved.

APPENDIX A

GLOSSARY

A register	Address register
ADU	Assembly/disassembly unit
ASCII	American standard code for information interchange
ASID	Active segment identifier
BASICTC	Basic test control code
BC	Base constant
BCO	Branch on condition
BCR	Branch on condition register
BDP	Business data processing
BN	Byte number
BR	Bounds Register
BS	Binding section
BMUX	Branch multiplexer
CBP	Code base pointer
CEJ/MEJ	Central exchange jump/monitor exchange jump
CEL	Corrected error log
CEM	Configuration environment monitor
CF	Critical frame
CFF	Critical frame flag
CFR	Conflict register
CH	Channel
CHD	Channel test
CM	Central memory
CMA	Central memory access
CMA1	Central memory access test
CMCEL	Cache memory corrected error log
CMU	Central memory access
CMSE	Common maintenance software executive
condition	A test within a subsection that uses a particular set of operands to test the common hardware element.
CPU	Central processing unit
CS	Control store
CSF	Current stack frame pointer
CTI/IPL	Common test initial/initial program load
DAP	Design action paper
DC	Debug code
DCI	Data control information
DCD	Data carrier detector
DDLT	Diagnostic decision logic table
DDP	Distributive data path
DEC	Model-dependent environment control (see EC)
DEX	Diagnostic executive (interface)
DI	Debug index
DLP	Debug list pointer
DM	Debug mask
DMR	Debug mask register
DPPD	Double PP driver
DSC	Display controller
DSP	Dynamic space pointer

DSR	Data set ready
DTR	Data terminal ready
EBCDIC	Extended binary-coded decimal interchange code
EC	Environment control
ECC	Error-correction code
EC1	Error code 1
EC2	Error code 2
ECL	Emitter-coupled logic
ECM	Extended central memory
ECS	Extended core storage
EDS1	Extended deadstart sequence
EIA	Electronic Industries Association
EID	Element identifier
EM	Exit mode
EPS	External procedure flag
ES	End suppression toggle (BDP edit instruction)
EXT1	Execution unit test
FCN	Function
FIFO	First-in, first-out
FL	Field length
FLC	Central memory field length register
FLE	Extended core storage field length register
FP	Floating-point
FSR	Fault Status Register in IOU
FTN	FORTRAN
FUD1	FIS1 utility program
FWA	First word address
GE	Greater than or equal
GK	Global key
GL	Global lock
GT	Greater than
IC	Integrated circuit
II	Illegal instruction
ILH	Instruction look-ahead
IMUX	Instruction multiplexer
I/O	Input/output
IOCP	Input/output unit control program
IOU	Input/output unit
JEP	Job mode exchange package
JPS	Job process state pointer
KC	Keypoint code
KCN	Keypoint class number
KEF	Keypoint enable flag
KM	Keypoint mask
LDS	Long deadstart sequence
LED	Light-emitting diode
LK	Local key
LPID	Last processor identification
LRN	Largest ring number
LRU	Least recently used
LSI	Large-scale integration
LT	Less than
LWA	Last word address

MA	Monitor address
MAC	Maintenance access control
MALADY	Microcode assembler for the CYBER 170 Model 825 processor
MAS	Micrand address select
MBE	Multiple bit error
MC	Master clear
MCH	Maintenance channel
MCI	Maintenance channel interface
MCR	Monitor condition register
MCU	Maintenance control unit
MDF	Model-dependent flags
MDW	Model-dependent word
MEC	CM environment control register
MEP	Monitor mode exchange package
MF	Monitor flag
MHR	Micrand holding register
MID	Maintenance identifier
MM	Monitor mask
MMR	Monitor mask register
MOP	Micro-operator (BDP edit instruction)
MOS	Metal-oxide semiconductor
MPPD	Main PP test driver
MPS	Monitor process state pointer
MR	Maintenance register
MRA1	Maintenance register access test
MRT1	Maintenance register test
MSL	Maintenance software library
NE	Not equal
network	A hardware element wholly contained on a pack and sharing no common circuits with any other network.
NOS	Network Operating System
NS	Negative sign toggle
OCF	On-condition flag
OI	Options installed
ON	Occurrence number
OP	Operation code
OS	Operating system
P register	Program address register
pak	A replaceable hardware module
PE	Parity error
PFA	Page frame address
PFS	Processor fault status
PID	Processor identifier
PIT	Process interval timer
PMF	Performance monitoring flag
PMT1	PP memory test 1
PMU1	PP memory test 2
PN	Page number
PO	Page offset
PP	Peripheral processor
PPM	Peripheral processor memory
PPS	Peripheral processor subsystem
PPU	Peripheral processor unit

PSA	Previous save area
PSF	Previous stack frame
PSM	Page size mask
PTA	Page table address
PTE	Page table entry
PTL	Page table length
PTM	Processor test mode
PVA	Process virtual address
PW	Partial write
QLT1	Quick look test
RA	Reference address
RA/FL	Reference address/field length
RAC	Central memory reference address register
RAE	Extended core storage reference address register
RAM	Random-access memory
RAM	Reliability, availablity, maintainability
RCL	Read and Clear Lock
RCT	Random command test
RMA	Real memory address
RMS	Rotating mass storage
RN	Ring number
RNI	Read next instruction
ROM	Read only memory
RP	Read permission (segment descriptor field)
RSB	Read Syndrome Bits
RTS	Request to send
SBE	Single bit error
SCT	Special characters table (BDP edit instruction)
SDE	Segment descriptor table entries
SDT	Segment descriptor table
SECDED	Single error correction, double error detection
SEG	Process segment number
SFSA	Stack frame save area
SET	Subsection error table
SIT	System interval timer
SM	The symbol (BDP edit instruction)
SPID	Segment page identifier
SPPD	Single PP driver
SPT	System page table
SRT	Subscript range table
SS	Status summary
STA	Segment table address
STL	Segment table length
subsection	A series of tests that check out a specific hardware element.
SV	Specification value
SVA	System virtual address
SWL	Software writer's language
TE	Trap enable
test	A general term which can refer to conditions, subsections, sections or units.
TED	Trap-enable delay
TEF	Trap-enable flip-flop
TOS	Top of stack

TP	Trap pointer
TPM	Two port multiplexer
UART	Universal asynchronous receiver-transmitter
UCEL1	Uncorrected error log 1
UCEL2	Uncorrected error log 2
UCR	User condition register
UEL1...UELn	Uncorrected error log 1...n
UEL2	Uncorrected error Log 2
UM	User mask
unit	An arbitrary functional area within the processor.
UTP	Untranslatable pointer
UVMID	Untranslatable virtual machine identifier
VC	Search control code (page descriptor field)
VL	Segment validation (segment descriptor field)
VLEX	Virtual level executive
VMCL	Virtual machine capability list
VMID	Virtual machine identifier
WAR	Word assembly register
WCB	Write Check Bits
WDR	Word disassembly register
WP	Write access control (segment descriptor field)
XP	Execute access control (segment descriptor field)
ZF	Zero field toggle (BDP edit instruction)

APPENDIX B

RCT1 AND RCT2 TEST/USAGE SETS

Op Code	Ref. No.	Short Name	Usage	Test
00	121	Program error		
01	162	Wait		
02	120	Exchange	1/2	
03	122	Interrupt processor		
04	117	Return	1	
05	138	Purge		
06	118	Pop		
07	133	Copy to central memory maintenance register		
08	132	Copy from central memory maintenance register		
09	051	Copy to Ak from Aj	2	1
0A	052	Copy to Ak from Xj	2	1
0B	053	Copy to Xk from Aj	1/2	1
0C	049	Copy to Xk from Xj, halfword	1/2	1
0D	130	Copy to Xk from Xj	1/2	1
0E	130	Copy from state register		
0F	131	Copy to state register	1	
10-13		Unimplemented		
14	124	Test and set, bit		
15		Unimplemented		
16	126	Test page		
17	127	Load page table index		
18	065	Logical sum	1/2	1
19	066	Logical difference	1/2	1
1A	067	Logical product	1/2	1
1B	068	Logical complement	1/2	1
1C	069	Logical inhibit		1
1D		Unimplemented		
1E	145	Convert mark to boolean		1
1F	061	Enter Xk left, sign per j	1/2	1
20	027	Add integer, halfword	1/2	1
21	030	Subtract integer, halfword	1/2	1
22	032	Multiply integer, halfword		1
23	034	Divide integer, halfword		1
24	022	Add integer	1/2	1
25	023	Subtract integer	1/2	1
26	024	Multiply integer		1
27	025	Divide integer	2	1
28	029	Increase halfword by j	1/2	1
29	031	Decrease halfword by j	1/2	1
2A	056	Add address, halfword	2	1
2B		Unimplemented		
2C	036	Compare integer, halfword		1
2D	035	Compare integer		1
2E	047	Unconditional branch, intra-segment	1/2	
2F	048	Unconditional branch, inter-segment	1	

Op Code	Ref. No.	Short Name	Usage	Test
30	099	Add floating point		1
31	100	Subtract floating point		1
32	103	Multiply floating point		1
33	104	Divide floating point		1
34	105	Add floating point, double precision		1
35	106	Subtract floating point, double precision		1
36	107	Multiply floating point, double precision		1
37	108	Divide floating point, double precision		1
38		Umimplemented		
39	164	Enter X1 with logical jk		1
3A	097	Convert integer to floating point		1
3B	098	Convert floating point to integer		1
3C	114	Compare floating point		1
3D	057	Enter Xk with plus j	1/2	1
3E	058	Enter Xk with minus j	2	1
3F	060	Enter X0 with logical jk		1
40-6F		Unimplemented		
70	074	Decimal sum		2
71	075	Decimal difference		2
72	076	Decimal product		2
73	077	Decimal quotient		2
74	083	Decimal compare		2
75	092	Numeric move		2
76	089	Move bytes		2
77	084	Byte compare		2
78-7F		Unimplemented		
80	020	Load multiple		
81	021	Store multiple		
82	006	Load X word, displaced	1/2	1
83	008	Store X word, displaced	1/2	1
84	017	Load A bytes, displaced		
85	019	Store A bytes, displaced		
86	013	Load X bytes, displaced relative		1
87	165	Enter X1 with sign extended, jkQ		1
88	014	Load X, bit	2	1
89	015	Store X, bit		1
8A	028	Add integer, halfword plus Q	2	1
8B	143	Add integer, word plus Q	2	1
8C	033	Mulitply integer, halfword plus Q		1
8D	059	Enter Xk with Q	1/2	1
8E	054	Add address, A plus Q	2	1
8F	055	Add address, X plus P plus Q		1
90	041	Branch =, halfword integer	1/2	1
91	042	Branch ≠, halfword integer	1/2	1
92	043	Branch >, halfword integer	1/2	1
93	044	Branch ≥, halfword integer	1/2	1
94	037	Branch =, integer	1/2	1
95	038	Branch ≠, integer	1/2	1
96	039	Branch >, integer	1/2	1
97	040	Branch ≥, integer	1/2	1
98	109	Branch =, floating point		1
99	110	Branch ≠, floating point		1
9A	111	Branch >, floating point		1

Op Code	Ref. No.	Short Name	Usage	Test
9B	112	Branch ≥, floating point		1
9C	045	Branch and increment <, integer		1
9D	046	Branch ≠, segment, else compare BN		
9E	113	Branch =, floating point exception		1
9F	134	Branch/alter, condition register		
A0	016	Load A bytes, indexed/ displaced		1
A1	018	Store A bytes, indexed/displaced		1
A2	005	Load X words, indexed/displaced	1/2	1
A3	007	Store X words, indexed/displaced	1/2	1
A4	009	Load X bytes, indexed/displaced	2	1
A5	011	Store X bytes, indexed/displaced	2	1
A6	115	Call, displaced		
A7	161	Copy A with displacement, module		1
A8	062	Shift word, circular	1	1
A9	063	Shift word, end-off	1/2	1
AA	064	Shift halfword, end-off	1/2	1
AB		Unimplemented		
AC	070	Isolate bit mask	1/2	1
AD	071	Isolate bit string		1
AE	072	Insert bit string		1
AF		Unimplemented		
B0	116	Call, displaced relative		
B1	136	Keypoint		
B2-B3		Unimplemented		
B4	125	Compare and swap		
B5-BF		Unimplemented		
C0-C7	139	Execute algorithm 0-7		
C8-CF		Unimplemented		
D0-D7	001	Load X bytes, indexed/displaced, 1-8	1/2	1
D8-DF	003	Store X bytes, indexed/displaced, 1-8	1/2	1
E0-E3		Unimplemented		
E4	078	Decimal scale		2
E5	079	Decimal scale rounded		2
E6-E8		Unimplemented		
E9	085	Byte compare collated		2
EA		Unimplemented		
EB	088	Byte translate		2
EC		Unimplemented		
ED	091	Edit		2
EE-F2		Unimplemented		
F3	086	Byte scan while non-member		2
F4	096	Calculate subscript		2
F5-F8		Unimplemented		
F9	154	Move immediate data		2
FA	155	Compare immediate data		2
FB	156	Add immediate data		2
FC-FF		Unimplemented		

APPENDIX C

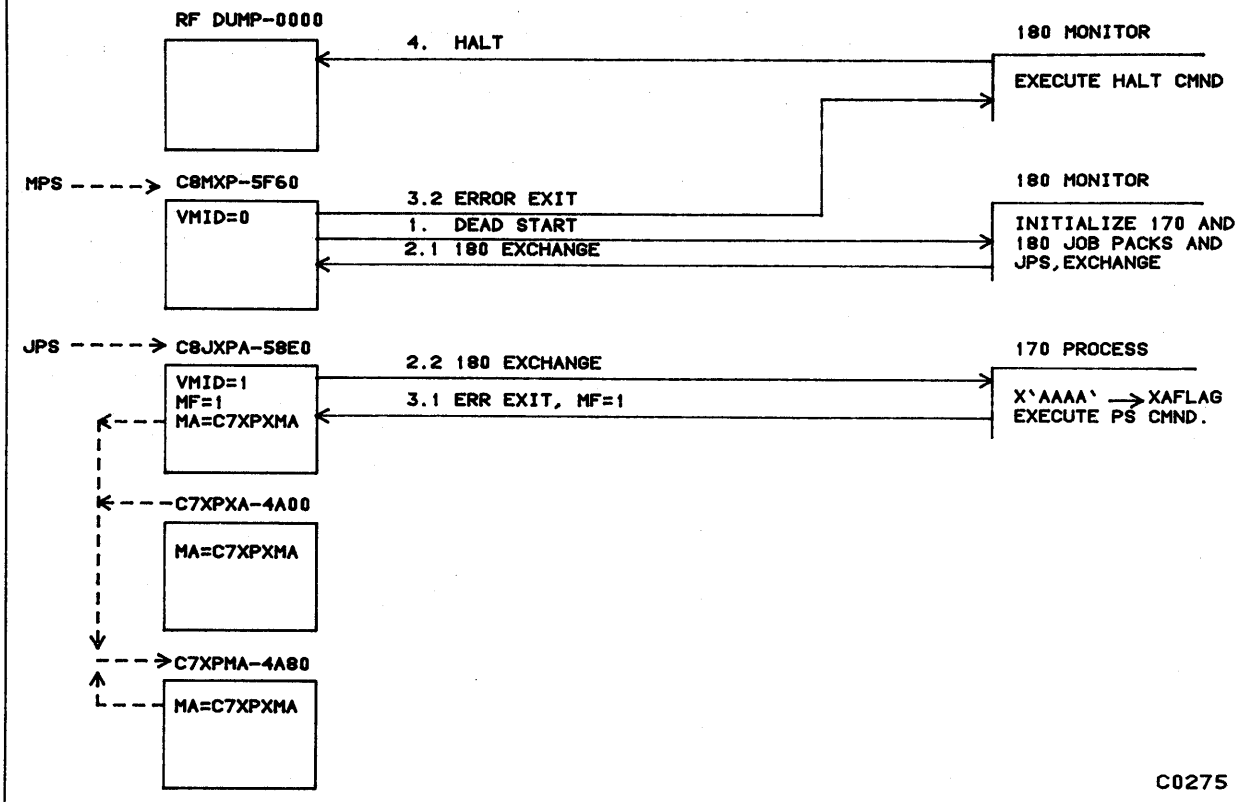
EXCHANGE SEQUENCE DIAGRAMS

The following pages provide diagrams illustrating the sequence of exchanges in the various sections of the test. No diagram is provided for section 4. Some diagrams serve more than one section.

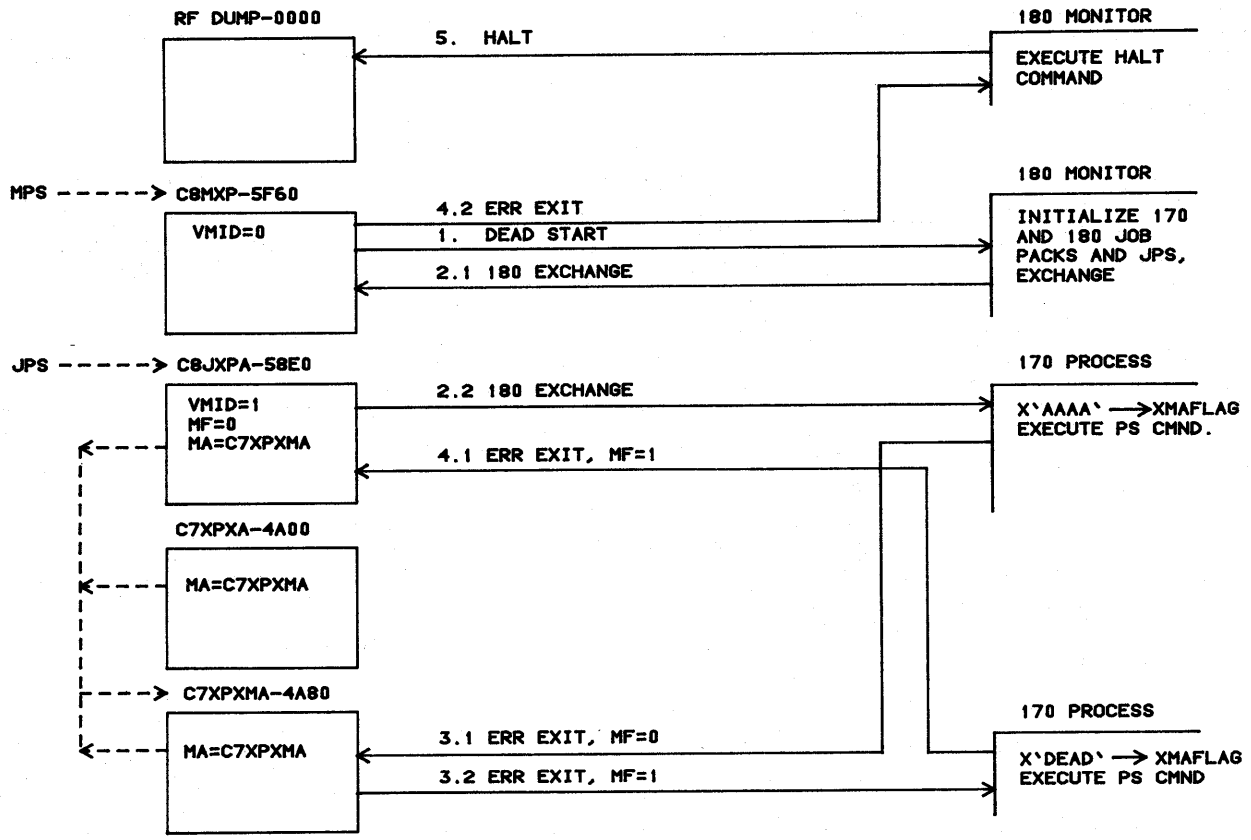
In each diagram, the exchange packages are on the left. The blocks on the right briefly describe the function of an associated process. The connecting lines indicate the movement of a process' registers from an exchange package in memory into the central processor's registers or vice versa. Each line bears a number showing the sequence step and note indicating the event which triggers the action. Dashed lines indicate a pointing action.

Each exchange package shows the assembly listing name of the package and its location (hex byte address) in central memory. A package name (label) which begins with C8 refers to a CYBER 180 package; C7 refers to a CYBER 170 package.

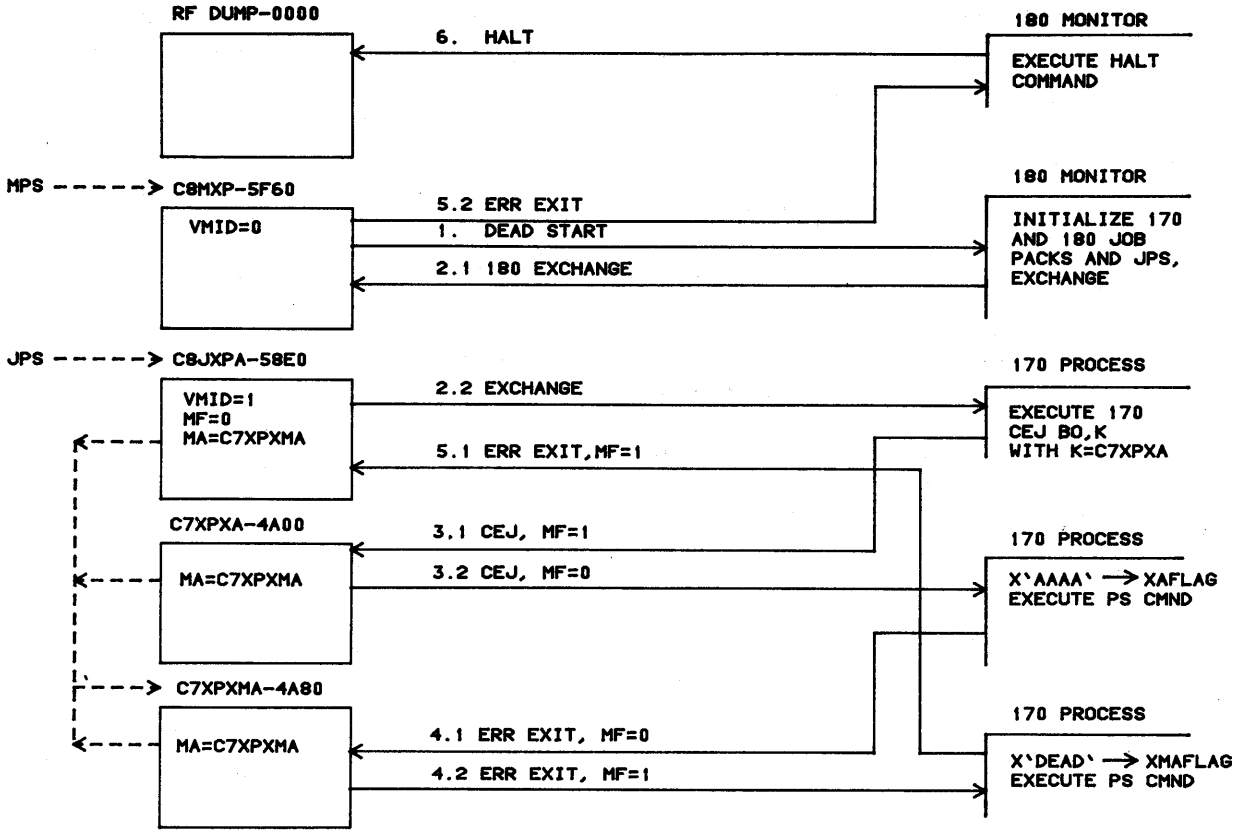
Some sequence steps or a series of steps may be repeated many times during the execution of a given condition. The diagrams only provide an overall view; reference should be made to the descriptions with each test section in part II-6 of this manual for additional details.



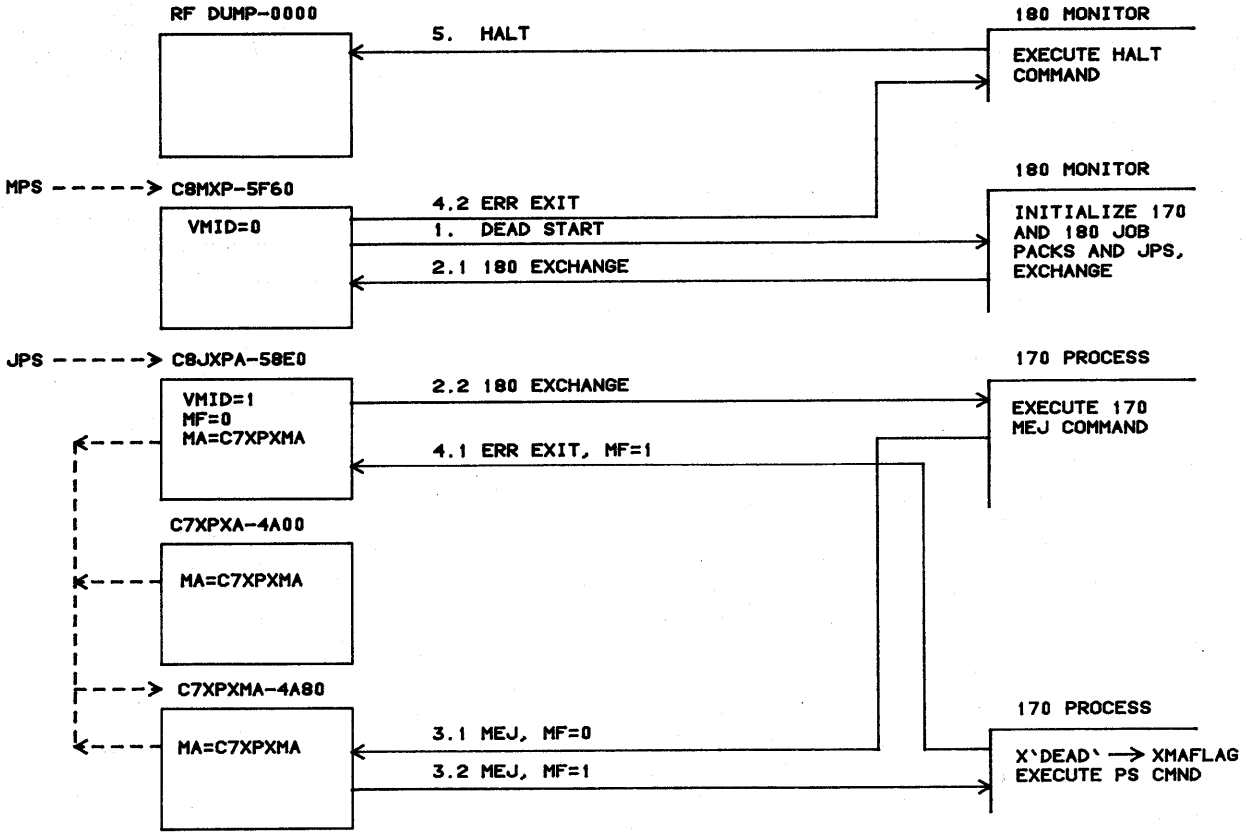
C0275
SECT 1, COND 0 - PS COMMAND, 170 MON



SECT 1,COND 1 - PS COMMAND,170 JOB

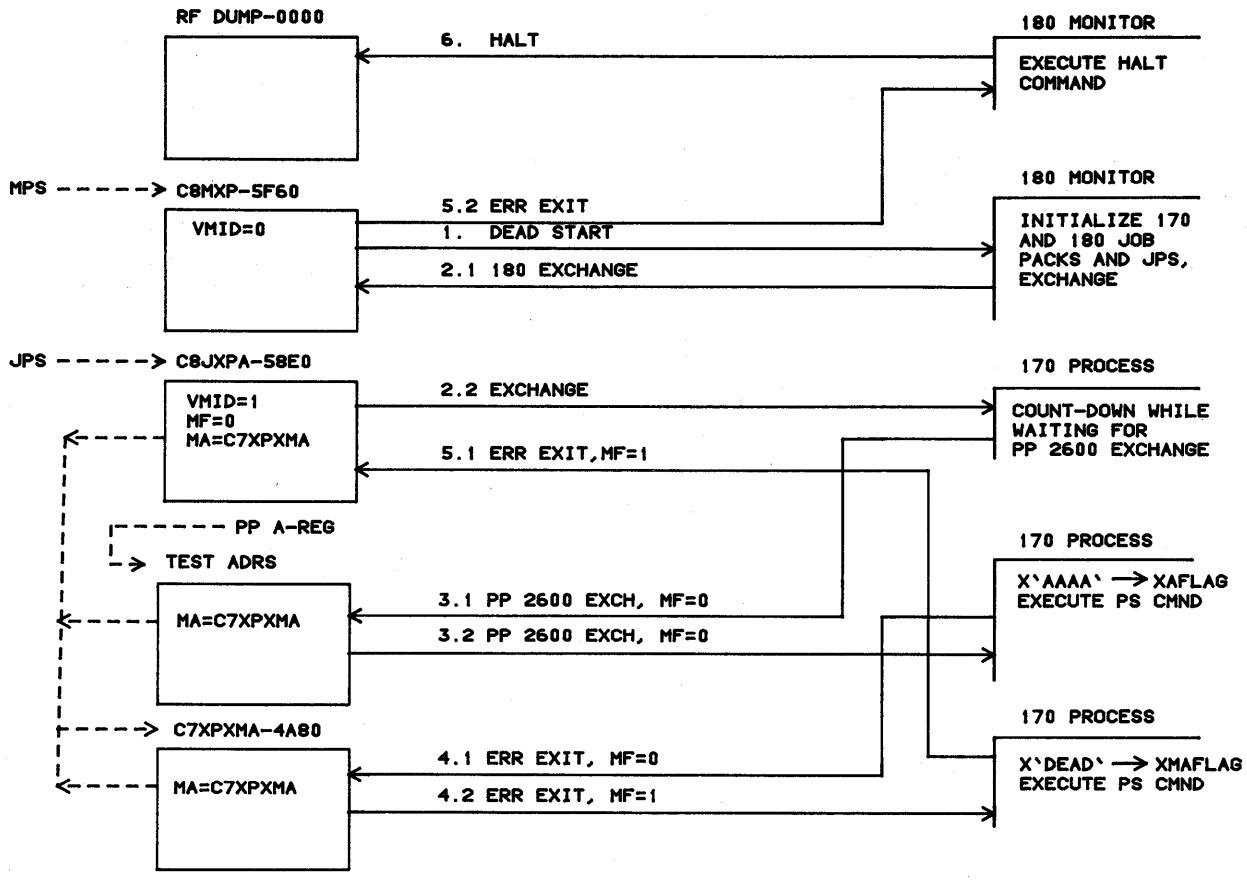


SECT 1, COND 2 - CEJ COMMAND, 170 MON



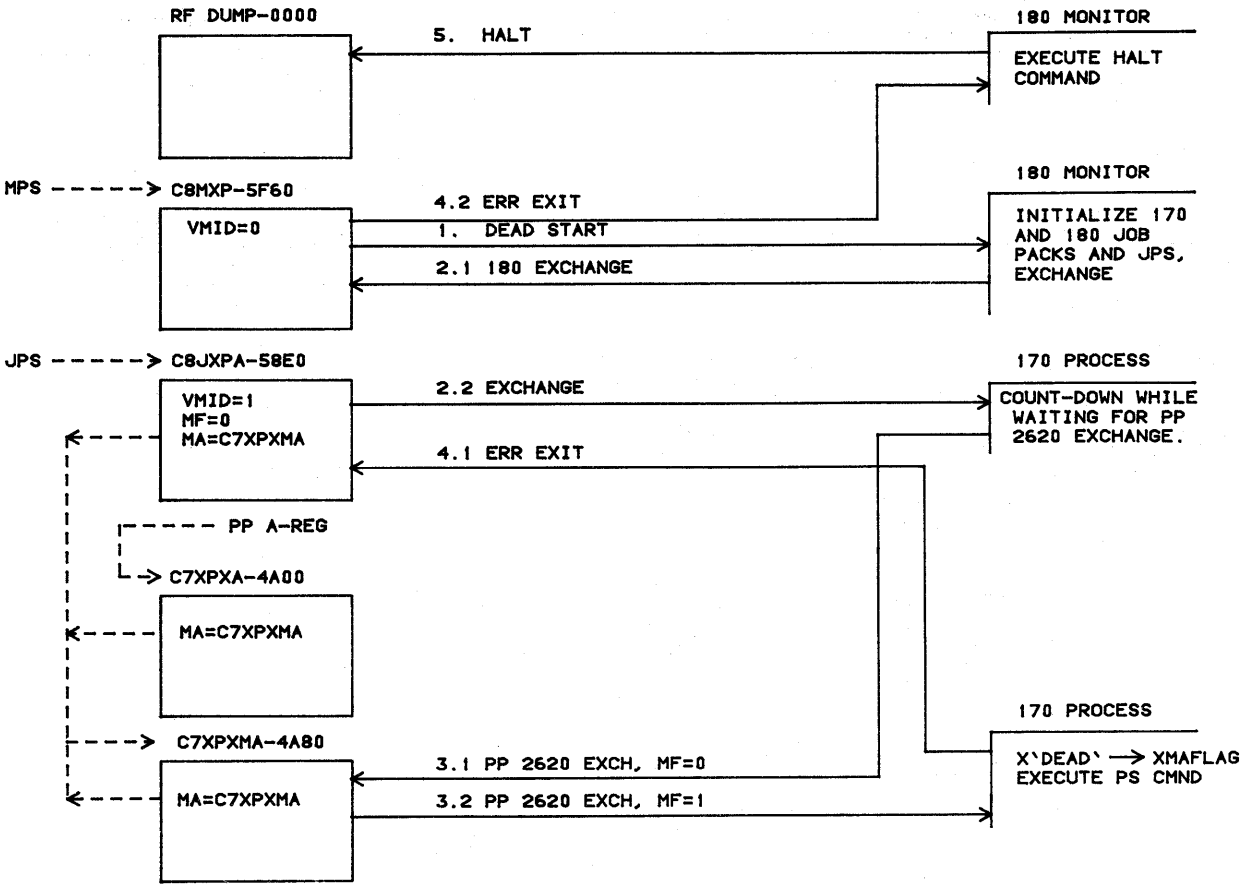
C0278

SECT 1, COND 3 - MEJ COMMAND, 170 JOB



C0279

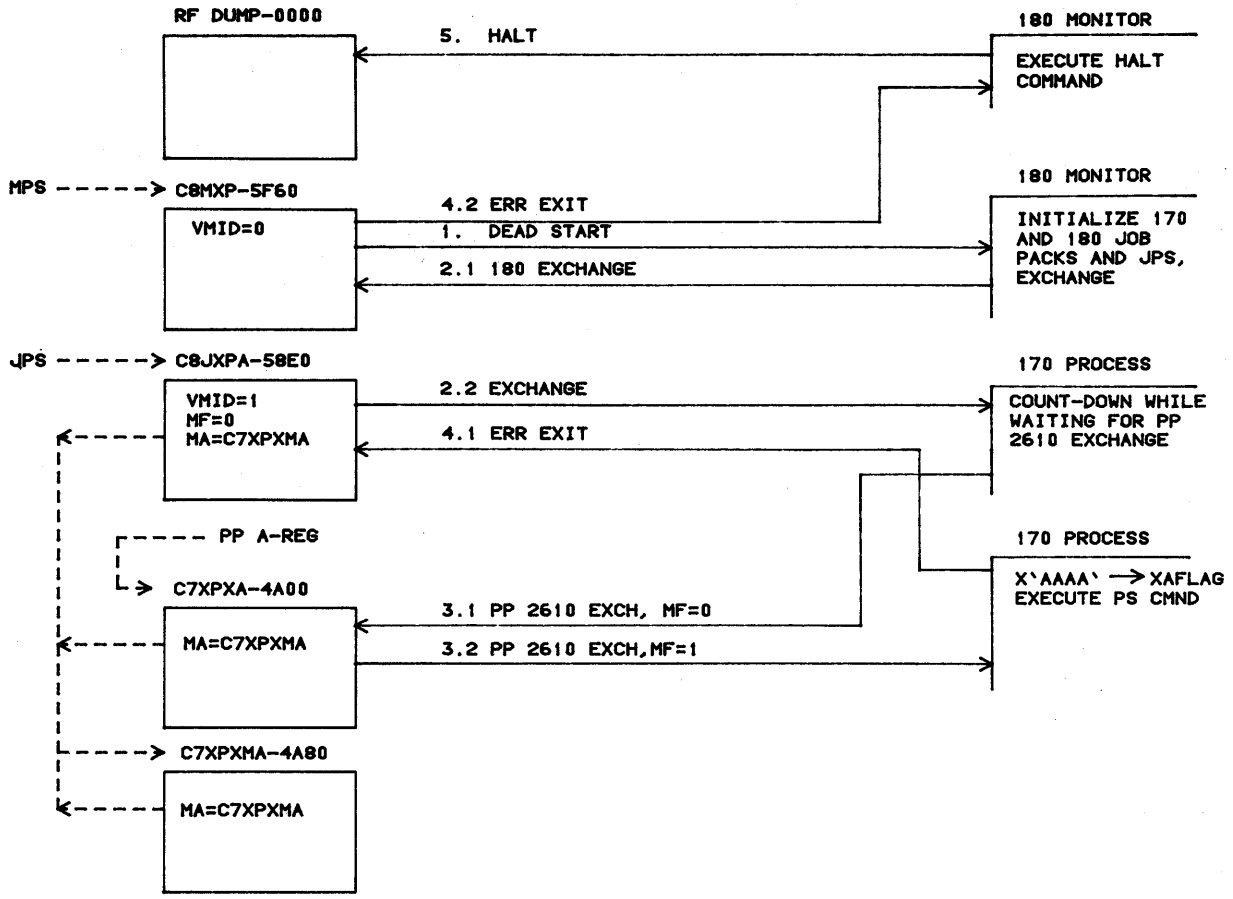
SECT 2 AND 16, EXCHANGE ADDRESSING
SECT 3, COND 0 - 2600, 170 JOB



C0280

C0280

SECT 3, COND 2 - 2620, 170 JOB



NOTE:

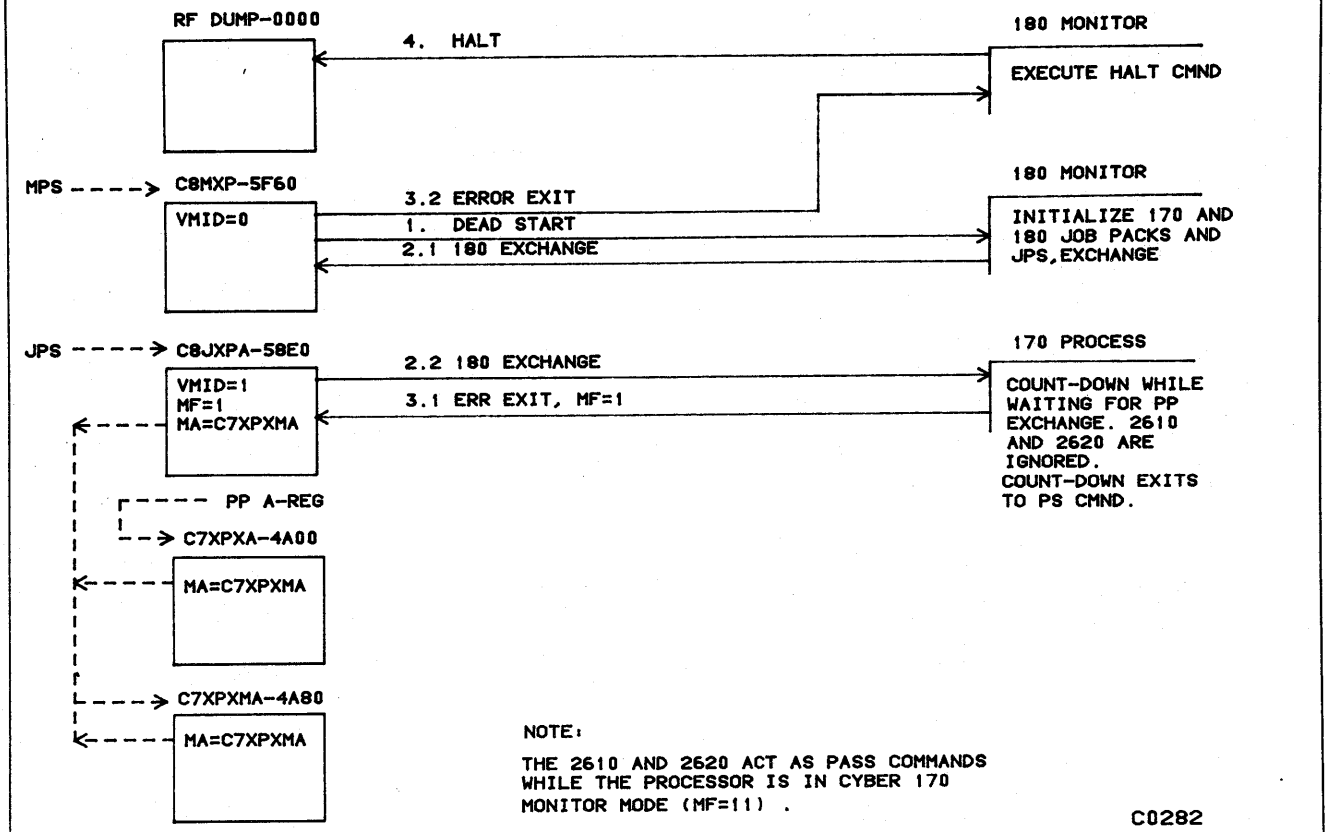
FOR COND #3, A PP 2600 COMMAND IS EXECUTED.

THE CYBER 170 MF=1 BEFORE
AND AFTER THE SWAP OF C7XPXA.

C0281

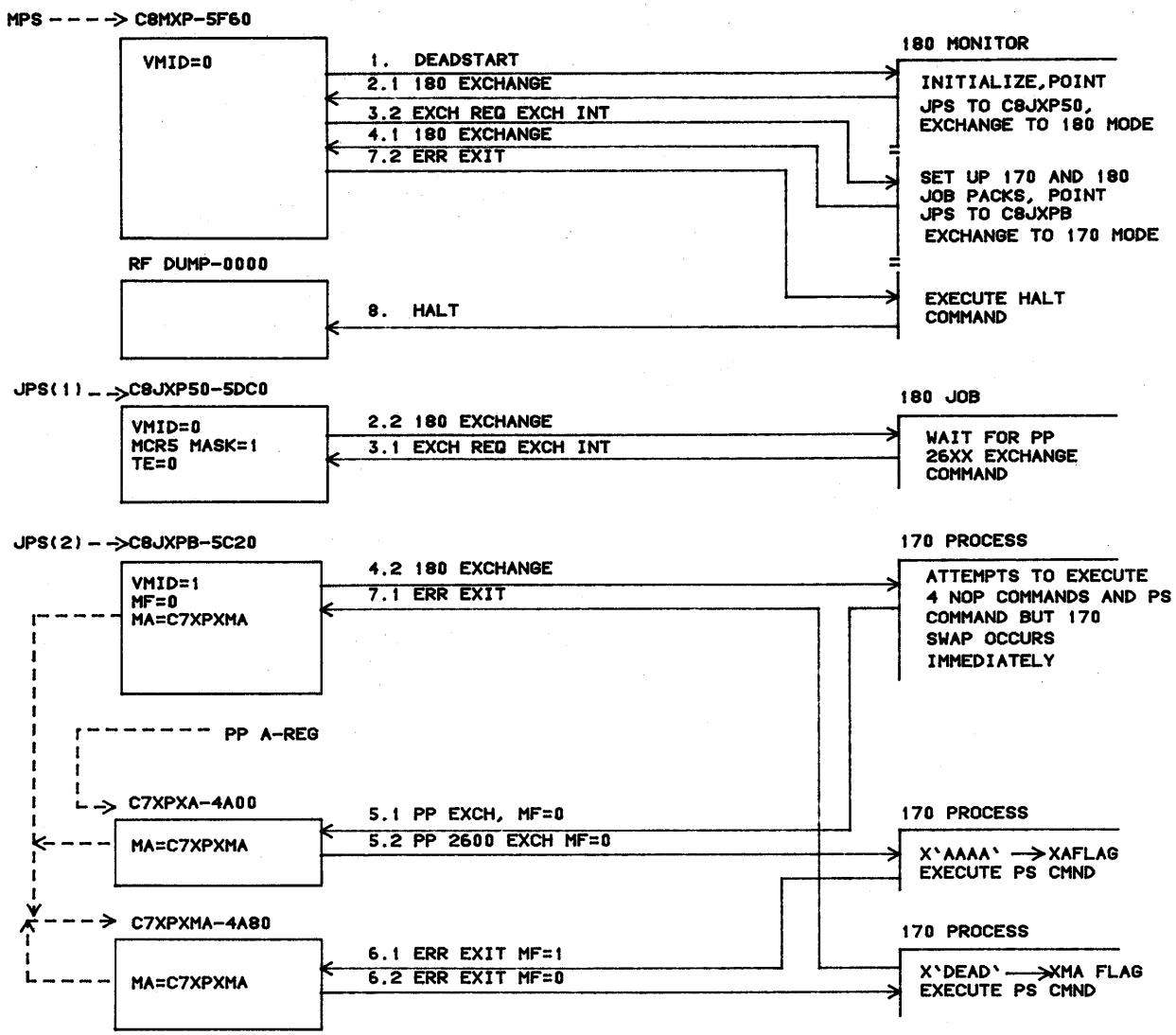
C0281

SECT 3, COND 1, 2610, 170 JOB
SECT 3, COND 3 - 2600, 170 MON



C0282

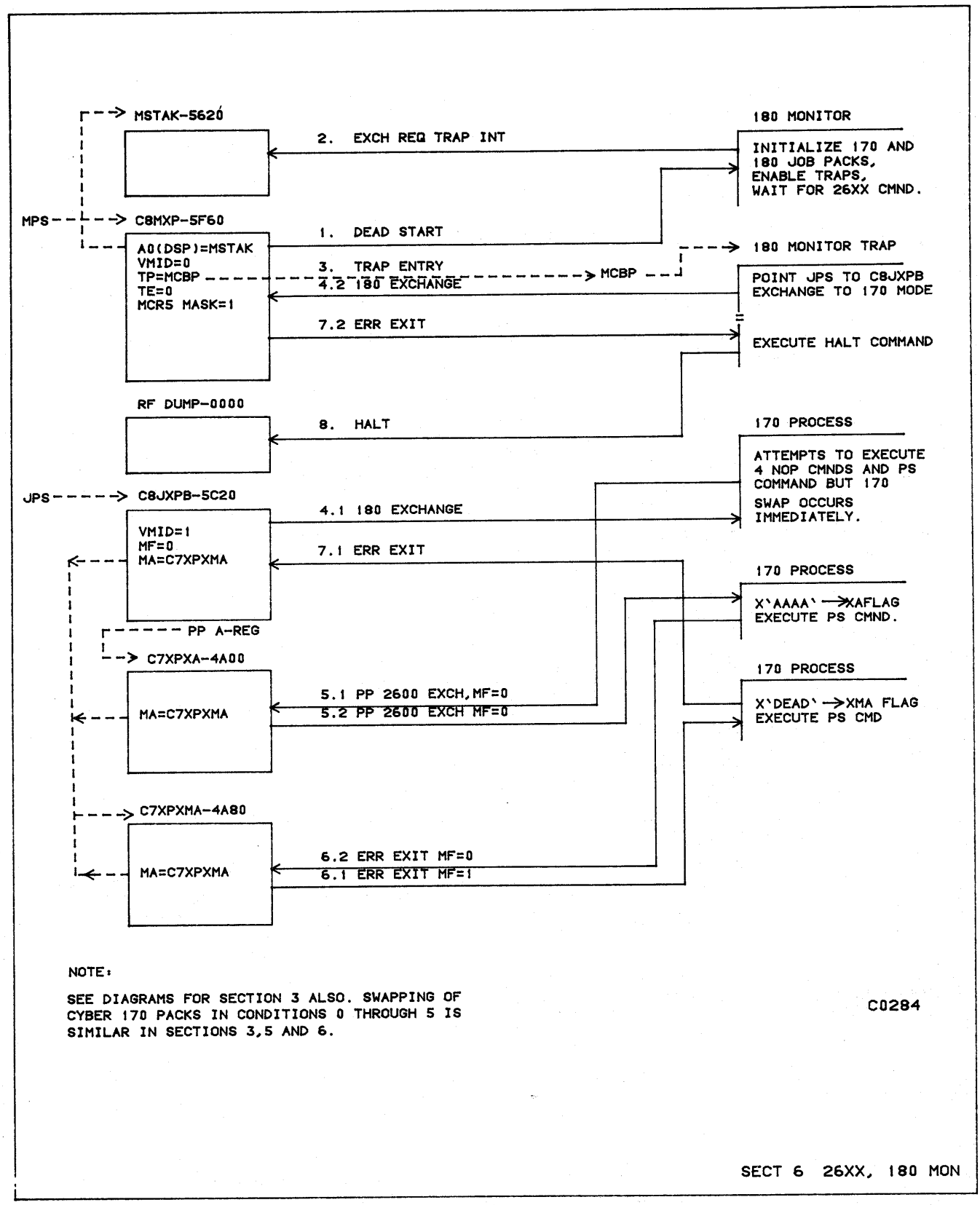
SECT 3, COND 4 - 2610, 170 MON
SECT 3, COND 5 - 2620, 170 MON



NOTE:
SEE DIAGRAMS FOR SECTION 3 ALSO.SWAPPING OF CYBER 170 PAKS IN CONDITIONS 0 THRU 5 IS SIMILAR IN SECTIONS 3,5 AND 6.

C0283

SECT 5 26XX, 180 JOB

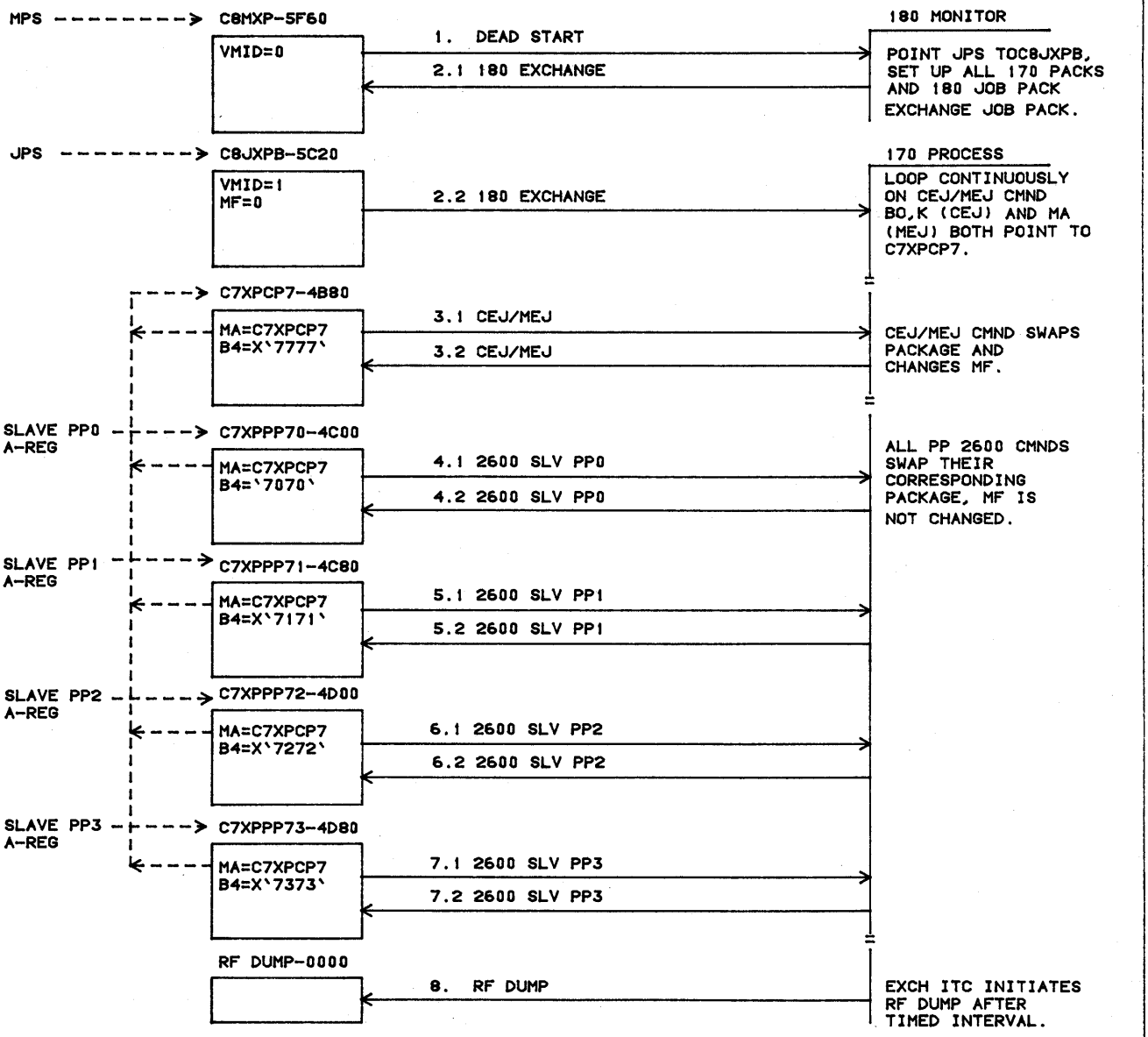


NOTE:

SEE DIAGRAMS FOR SECTION 3 ALSO. SWAPPING OF CYBER 170 PACKS IN CONDITIONS 0 THROUGH 5 IS SIMILAR IN SECTIONS 3,5 AND 6.

C0284

SECT 6 26XX, 180 MON

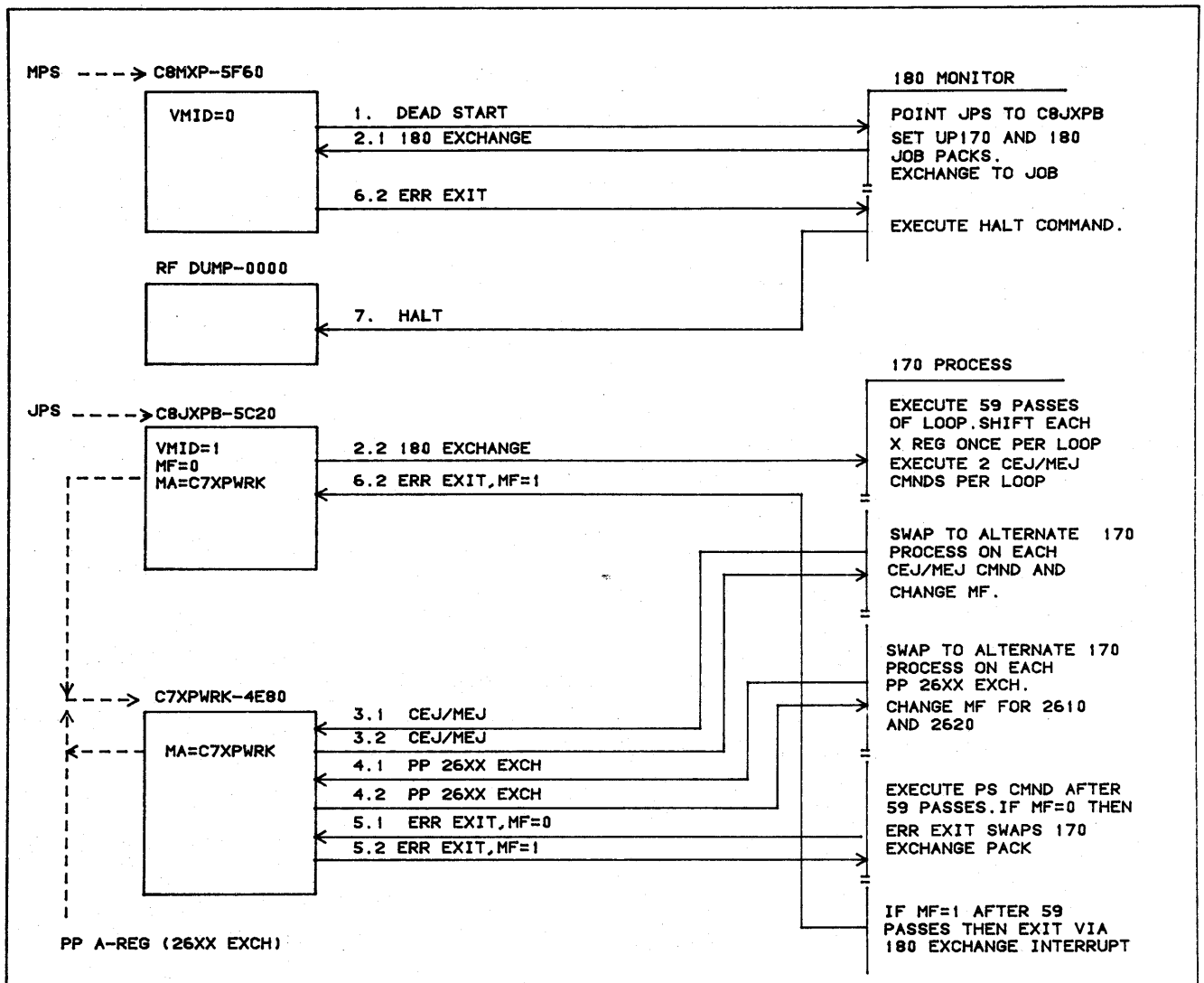


NOTE:

STEPS 3 THRU 7 MAY OCCUR RANDOMLY IN ANY ORDER.

C0285

SECT 7 MULTIPLE 2600 AND BO CHECKS

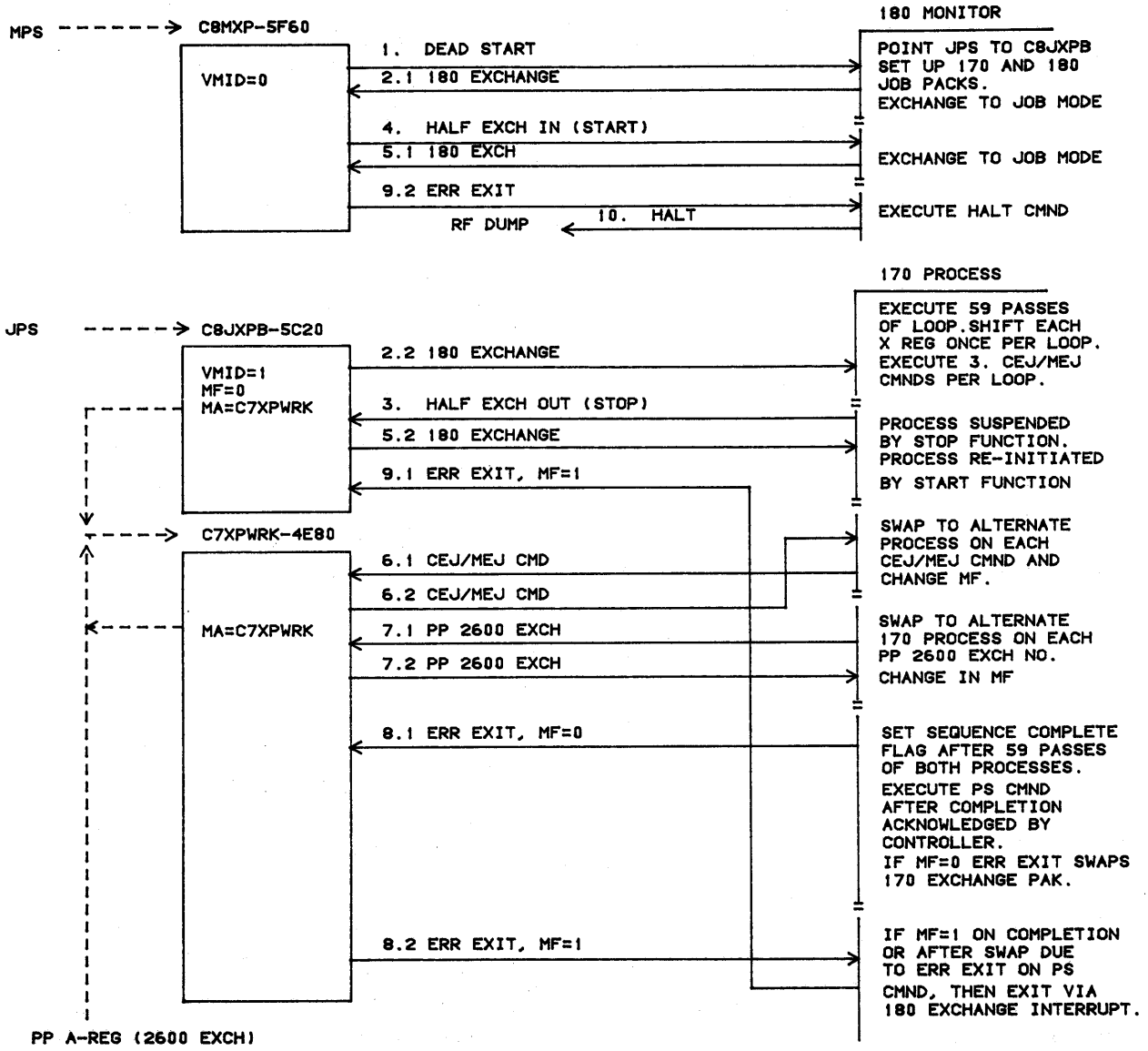


NOTE:

SEQUENCE STEPS 3. AND 4. MAY
OCCUR RANDOMLY IN ANY ORDER.

C0286

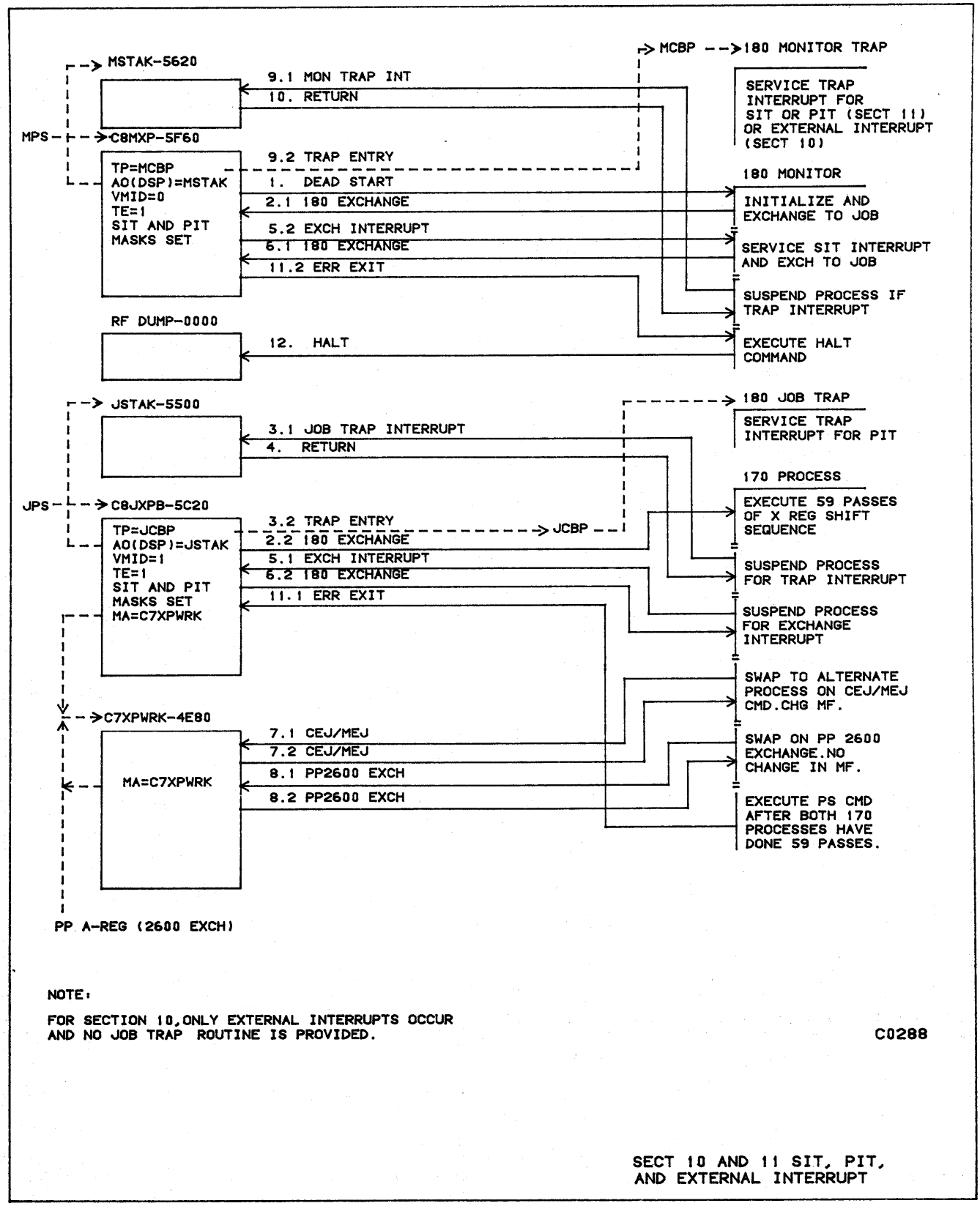
SECT 8 26XX BLOCK READ AND WRITE



NOTE:

SEQUENCE STEPS 3,5,6 AND 7 MAY OCCUR RANDOMLY IN ANY ORDER.

C0287

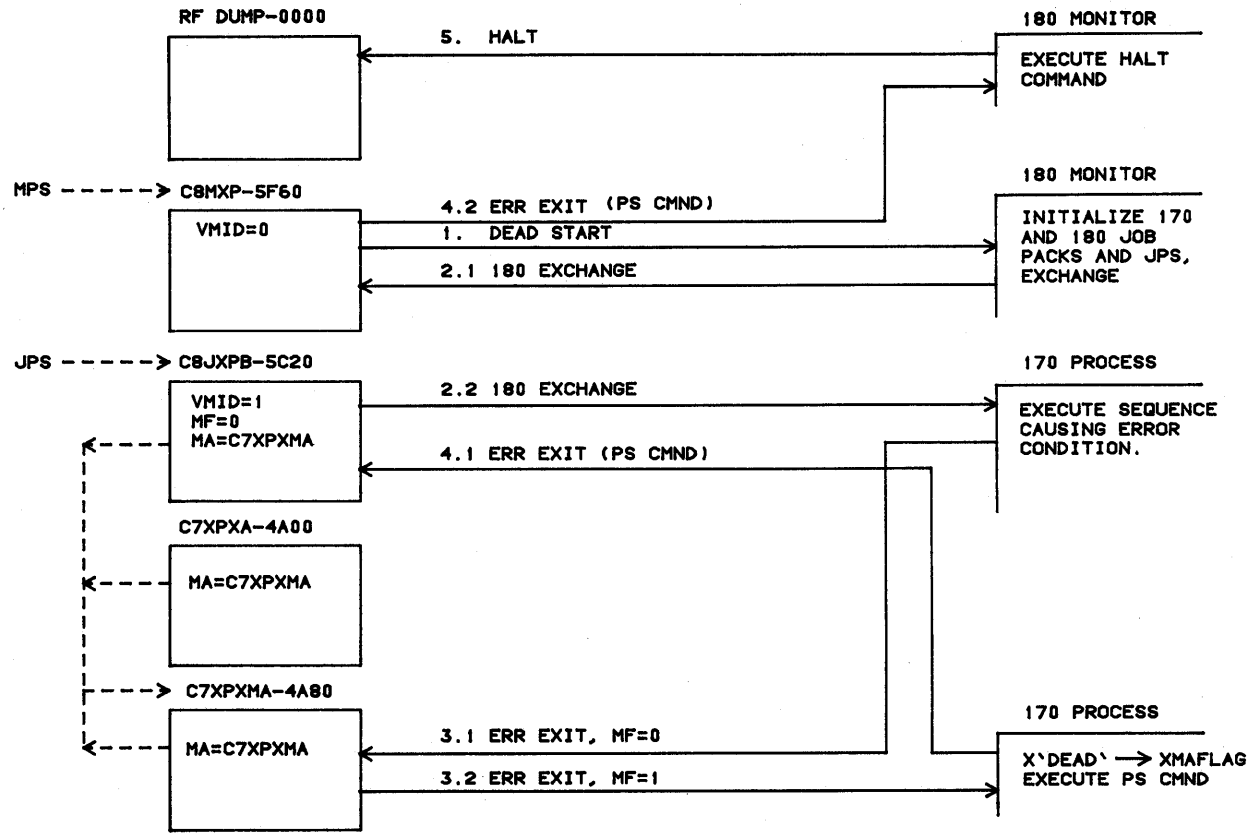


NOTE:

FOR SECTION 10, ONLY EXTERNAL INTERRUPTS OCCUR AND NO JOB TRAP ROUTINE IS PROVIDED.

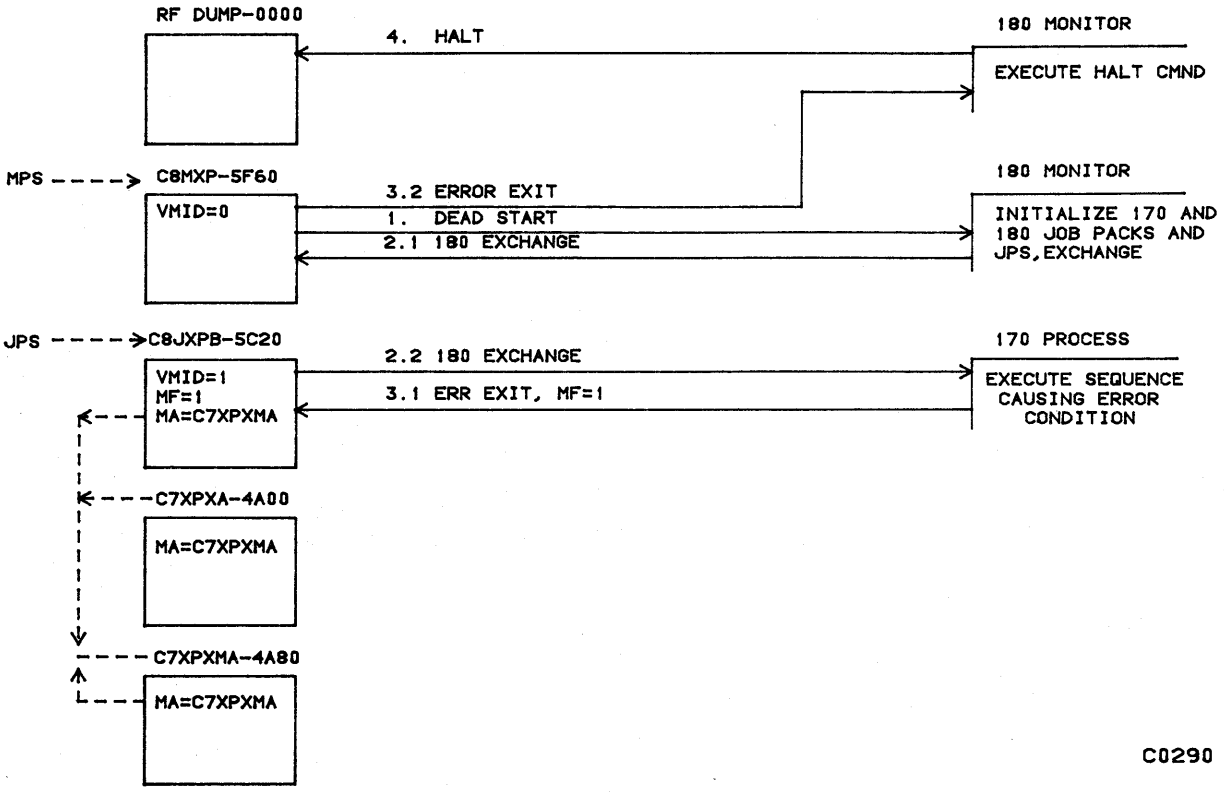
C0288

SECT 10 AND 11 SIT, PIT, AND EXTERNAL INTERRUPT



C0289

SECT 12 TO 15 - ERROR EXIT, 170 JOB



C0290

SECT 12 TO 15 - ERROR EXIT, 170 MON

COMMENT SHEET

MANUAL TITLE: MSL15X Model Independent Tests Maintenance Software Reference Manual

PUBLICATION NO.: 60469390 REVISION: E

NAME: _____

COMPANY: _____

STREET ADDRESS: _____

CITY: _____ STATE/PROV.: _____ POSTAL CODE_____

This form is not an order blank.

Control Data Corporation welcomes your evaluation of this manual. Please indicate any errors, suggested additions or deletions, or general comments below (please include page number references).

This comment sheet becomes a self-mailer when the binding edge is cut off and it is folded along the dashed lines on the reverse side.

fold fold

Place correct
postage here.

CONTROL DATA CANADA
Toronto Publications (210)
1855 Minnesota Court
MISSISSAUGA, Ontario, CANADA
L5N 1K7

fold fold

CORPORATE HEADQUARTERS, P.O. BOX 0, MINNEAPOLIS, MINN. 55440
SALES OFFICES AND SERVICE CENTERS IN MAJOR CITIES THROUGHOUT THE WORLD

LITHO IN U.S.A.

CONTROL DATA CORPORATION