MSL15X Model Independent Software

Reference

Publication Number 60469390

Manual History

Revision	Date	Description
A	05–82	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	05–83	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	05–84	Manual revised; includes ECO PD03109. Reflects release level 161. See List of Effective Pages for changed pages. This edition obsoletes all previous editions.
F	10–84	Manual revised; includes ECO CA 46510. Reflects release level 167. See List of Effective Pages for changed pages.
G	04–85	Manual revised; includes FCO CAs 46800 through 46807, and ECO PDs 3416 through 3420. Reflects release level 173. See List of Effective Pages for changed pages.
H	09–85	Manual revised; includes FCO CAs 47360 through 47367, and ECO PDs 03400 through 03402, 03479, and 03480. Reflects release level 179. See List of Effective Pages for changed pages.
J	03–86	Manual revised; includes FCO CAs 47862, 47863, 47864 and ECO PDs 3499, 3500, and 3501. Reflects release level 186. See List of Effective Pages for changed pages.
K	03–87	Manual revised; includes ECO PDs 3674, 3675, and 3676. Reflects release level 678. See List of Effective Pages for changed pages.

©1982, 1983, 1984, 1985, 1986, 1987, 1988, 1989 by Control Data Corporation
All rights reserved.
Printed in the United States of America.

Revision	Date	Description
L	08–87	Manual revised; includes ECOs PD 03885, PD 03886, PD 03887, CA 49091, CA 49092, and CA 49093. Reflects release level 688. See List of Effective Pages for changed pages.
M	03–88	Manual revised; includes ECOs PD 04041 and CA 49563. Reflects release level 700. See List of Effective Pages for changes.
N	12–89	Revision N updates documentation to CIP L739. Because this manual represents a new printing format, this edition obsoletes all previous editions.

About This Manual

This manual describes the tests provided on the Maintenance Software Library (MSL15X) for use on the CDC® CYBER 170 Models 815, 825, 835, 845, and 855 and CYBER 180 Models 810, 830, 835, 840, 845, 850, 855, 860 and 990 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 chapters:

1. Memory Test	CMEM
2. Random Command Test	RCT1
3. Random Command Test	RCT2
4. Fixed Operand Command Tests	FCT1,2,3,5
5. Virtual Mode Instruction Level Test	FCT9
6. Exchange Test	EXCH
7. Trap Test	TRAP
8. Random Fast/Slow Test	RFST
9. Debug Test	DBUG
10. Keypoint Test	KYPT
11. Cache Test	CACH
12. Model Independent General Design Specification Tests	MIGDS

Related Publications

Control Data Publication	Publication Number
MSL15X Off-line Maintenance Software Library Reference Manual	60456530
MSL151 Models 810, 815, 825, and 830 Maintenance Software Reference Manual	60469400
MSL152 Model 835 Maintenance Software Reference Manual	60469410
MSL153 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

Command Buffers

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

Clock Margins

A mask parameter is provided in the CMSE command ER to inhibit changes to specified bits of the register named by ER. The test command buffers allow setting of clock and voltage margin bits in the mask parameters of ER commands which initialize DEC registers.

The operator may use the CMSE command CM at any time to set clock margins (or use ER to set voltage margins). Subsequent execution of any test command buffer does not affect the previously set margin values.

Single/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 a single CP only
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

1. Command buffers FCT11, FCT51, FCT91, TRAP1, and EXCH1 are PP-based tests for single CP systems only.

   Command buffers RCT11, RCT21, CMEM1, FCT21, FCT31, and the eight FT3XXX command buffers execute on a single CP system; they may also be used on either CP of a dual CP system. To select a CP for test on dual CP systems, execute either command buffer CP0 or CP1 before executing one of the above tests.

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:

$$QLT10	$$FT3FP1	$$CP0
$$PMT10	$$FT3BD1	$$CP1
$$EXT10	$$FT3S11	$$CLEAR1
$$PMU10	$$FT3S21	$$ABU
$$CHD10	$$FCT51	$$DEM0TC1
$$CMA10	$$FCT51S	$$DPALL
$$MRA10	$$FCT91	$$SNAP
$$MRT10	$$FCT91S	$$SNAPA
$$DST10	$$EXCH1	$$SNAPB
$$TPM10	$$EXCH1S	$$0FA10
$$CRA10	$$TRAP1	$$EME
$$MUX10	$$TRAP1S	$$VERIFY
$$FII10	$$EXC1	$$IOUS1
$$CSC10	$$EXC1A	$$INSL1
$$CMT10	$$EXC1C	$$INITS
$$CMI10	$$A1701	$$IASL1
$$FDS10	$$A1701A	$$DFVYS
$$FIS10	$$A1701B	$$ECC
$$CMEM1	$$RCT11	$$STC0
$$FCT11	$$RCT21	$$SDPP1
$$FCT11S	$$RCT111C	$$STCP1
$$FCT21	$$RCT221C	$$SCKP1
$$FCT31	$$RCT211C	$$DTCP1
$$FT3G01	$$RCT121C	$$DCKP1
$$FT3G11	$$RFST1	$$DFSQS
$$FT3G21	$$DBUG1	
$$FT3G31	$$C170CP	

Disclaimer

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

Contents

About This Manual 5

Related Publications 5
Command Buffers 6
Disclaimer 7

Part I: Model Independent Test Procedures

Central Memory Test - CMEM 1-1

Introduction 1-1
Requirements 1-1
Operational Procedure 1-3
Section Index 1-7
Operator Communication 1-8

Random Command Test 1 - RCT1 2-1

Introduction 2-1
Requirements 2-1
Operational Procedure 2-3
Operator Communication 2-8

Random Command Test 2 - RCT2 3-1

Introduction 3-1
Requirements 3-1
Operational Procedure 3-3
Operator Communication 3-7

Fixed Operand Command Tests - FCT1,2,3,5 4-1

Introduction 4-1
Requirements 4-2
Operational Procedure 4-4
Operator Communication 4-30
Operator Entries 4-41

Virtual Mode Instruction Level Test - FCT9 5-1

Introduction 5-1
Requirements 5-1
Operational Procedure 5-3
Operator Communication 5-9

Exchange Test - EXCH 6-1

Introduction 6-1
Requirements 6-1
Operational Procedure 6-3
Operator Communication 6-11

Trap Test - TRAP 7-1

Introduction 7-1
Requirements 7-1
Operational Procedure 7-3
Operator Communication 7-9

Random Fast/Slow Test - RFST ... 8-1

Introduction 8-1
Requirements 8-1
Operational Procedures 8-3
Operator Communication 8-8

Debug Test - DBUG 9-1

Introduction 9-1
Requirements 9-1
Characteristics 9-2
Operational Procedure 9-3
Operator Communication 9-7

Keypoint Test - KYPT 10-1

Introduction 10-1
Requirements 10-1
Operational Procedure 10-3
Operator Communication 10-7

Cache Test - CACH 11-1

Introduction 11-1
Requirements 11-1
Operational Procedures 11-3
Operator Communication 11-7

Model Independent General Design Specification Tests - MIGDS 12-1

Introduction 12-1
Requirements 12-2

Operational Procedure 12-4
Operator Communication 12-8
Exceptions – RNGT and CRPT .. 12-19

Part II: Model Independent Test Descriptions

Central Memory Test – CMEM ... 13-1
Program Description 13-1

Random Command Test – RCT1 . 14-1
Program Description 14-1

Random Command Test – RCT2 . 15-1
Program Description 15-1

Fixed Operand Command Tests – FCT1,2,3,5 16-1
FCT1 Test 16-1
FCT2 Test 16-1
FCT3 Test 16-1
FCT5 Test 16-2

Virtual Mode Instruction Level Test – FCT9 17-1
Program Description 17-1

Exchange Test – EXCH 18-1
Program Description 18-1
Section Descriptions 18-6

Trap Test – TRAP 19-1
Program Description 19-1
Section Descriptions 19-4

Random Fast/Slow Test – RFST .. 20-1
Program Description 20-1

Debug Test – DBUG 21-1
Program Description 21-1

Keypoint Test – KYPT 22-1
Program Description 22-1

Cache Test – CACH 23-1
Program Description 23-1

Model Independent General Design Specification Tests – MIGDS 24-1
Program Description 24-1

Glossary A-1

RCT1 and RCT2 Test/Usage Sets .. B-1

Exchange Sequence Diagrams C-1

Cache Addressing and Layout D-1

Tables

4-1. FCT1,2,3,5 Sizes and Times 4-3

Part I: Model Independent Test Procedures

Central Memory Test - CMEM 1-1

Random Command Test 1 - RCT1 2-1

Random Command Test 2 - RCT2 3-1

Fixed Operand Command Tests - FCT1,2,3,5 4-1

Virtual Mode Instruction Level Test - FCT9 5-1

Exchange Test - EXCH 6-1

Trap Test - TRAP 7-1

Random Fast/Slow Test - RFST 8-1

Debug Test - DBUG 9-1

Keypoint Test - KYPT 10-1

Cache Test - CACH 11-1

Model Independent General Design Specification Tests - MIGDS 12-1

Central Memory Test - CMEM 1

Introduction 1-1

Requirements 1-1
- Hardware 1-1
- Software 1-1
- Accessories 1-1
- Characteristics 1-2

Operational Procedure 1-3
- Restrictions and User Cautions 1-3
- Loading Procedure 1-3
- Running Procedures 1-3
- Parameters And Control Words 1-5
  - Parameters 1-5
  - Control Words 1-7

Section Index 1-7

Operator Communication 1-8
- Displays 1-8
  - Running Display 1-8
  - Running Display with Memory Degraded 1-9
  - Set Parameters Display 1-9
  - Help Display 1-10
  - Error Message Display 1-10
- Operator Entries 1-11
  - Parameter Directives 1-11
  - Run Directives 1-12
- Normal Messages 1-12
- Error Messages 1-12
  - Data Error Messages 1-13
    - Double-Bit Errors 1-13
    - Single-Bit Errors 1-14
    - Data Compare Errors 1-15
  - Command Entry Error Messages 1-15
- Applications 1-15
  - Loop-On-Error Condition 1-15
  - Scope Mode 1-16
  - Additional Running Procedures 1-16

Central Memory Test - CMEM 1

Introduction

CMEM is a CPU program that tests central memory to detect errors. The test runs off-line under the Common Maintenance Software Executive (CMSE) and interacts with the user by the system display device and keyboard. Communication needed to control the test is in English language directives. The test also runs on-line under the Diagnostic Virtual System (DVS). Consult the DVS manual for more details.

Requirements

Hardware

This test is intended for models 810, 815, 825, 830, 835, 840, 845, 850, 855, 860, and 990 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 central memory

Software

This test runs under control of CMSE. Interfaces are handled by CMSE and the Virtual Level Executive (VEXC).

Accessories

None required.

Characteristics

Description	Characteristics
Program name	CMEM
Size (source)	5,650 lines
Size (memory required for execution)	500,000 bytes
Code type	CPU code
Run time (default)	36 sec (4MB)
Run time (quick look)	N/A
Run time (all sections)	36 sec (4MB)
Level of isolation	Detection
Off-line test	Yes
Off-line system	CMSE (MSL15X)
On-line system	NOS/VE-DVS
Assembly language	META
Source code maintenance	SCU

Operational Procedure

Restrictions and User Cautions

None.

Loading Procedure

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

Load the microcode before executing the test.

Command buffer CMEMx (where x is 1 for models 810/815/825/830, 2 for 835, 3 for 845/850/855/860 and 5 for 990) exists on the MSL15X tape to facilitate loading and execution of this test. The test executes upon completion of the load. You can change parameters with the CMEM command set or the CMSE commands EC and EB. The parameter base address displays in word format during test initialization or at any time by typing LIST. The CMSE commands plus any CMEM program commands may be saved and re-executed by the CMSE command buffer capability.

Refer to the MSL15X reference manual for descriptions of CMSE commands. To run on-line, consult the DVS On-Line Usage manual for loading procedures.

Running Procedures

During normal execution of pass 1, the test displays the parameter words and the real memory word base address. PARAM16 contains the installed memory size in hex megabytes. For example, 1 = 1, A = 10, 10 = 16. The default is obtained from the memory Options Installed Register (OI Register 12).

When memory is in degrade mode, CMEM tests only the memory available after reconfiguration. In this case, parameter word 18 contains the amount of testable memory. Use the EC command to specify a memory size different from the one calculated by the test. Type LIST to see the new addresses that can now be tested. Always type LIST after using the EC command if you wish to specify a different memory size once test execution has begun.

NOTE

You do not have to supply any parameters to CMEM. The test automatically calculates the addresses available and tests them. However, if you 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 (that is, beginning, repeat, upon entering scope loop).

Parameters And Control Words

Parameters

Tag	Word ADDR Hexadecimal	Bits	Description	Default
	CPU0/CPU1			
PARAM0	3A020/3A820	0-35	Not used	4005051
		36	Monitor default run	
		37	Monitor bypass all options	
		38	Monitor test mode 1 (BASIC)	
		39	Monitor test mode 2 (PIT and SIT)	
		40	Page file (page map enabled)	
		41	Segment file (segment map enabled)	
		42	Cache (cache enabled)	
		43	Monitor mode	
		44	Remote access active	
		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	3A021/3A821	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	3A022/3A822	0-63	Section select bits for sections 63-0. Only sections 1-11 exist. Section select bits are numbered from right to left.	7FF
PARAM16	3A030/3A830	0-63	Memory size installed	0004

Tag	Word ADDR Hexadecimal	Bits	Description	Default
	CPU0/CPU1			
PARAM17	3A031/3A831	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 PARAM17 is left equal to zero, test does not display the message.	0000
PARAM18	3A032/3A832	0-63	When nonzero contains the amount of usable (testable) memory after degrade. If zero, the memory is not in degrade mode and PARAM 16 contains an amount of usable memory.	0000
COMPFLAG	3A29D/3A69D		Data compare flag. Zero equals no compare, nonzero equals compare data.	
PATTERN	3A2AD/3A6AD		Testing pattern	
FWA	3A2A4/3A6A4		First word address to test (RMA)	
LWA	3A2A5/3A6A5		Last word address to test (RMA)	
TFPVA	3A2A8/3A6A8		PVA of FWA	
TLPVA	3A2A9/3A6A9		PVA of LWA	
TRAPMASK	3A2B5/3A6B5		Mask for CMEM trap handler	
HALTMASK	3A2B6/3A6B6		Mask for VEXC error handler	
FIRST	3A2A2/3A6A2		First possible test address	
LAST	3A2A3/3A6A3		Last possible test address (determined by size of memory installed – PARAM16 or, if in degrade mode PARAM18, by size of usable memory).	
MEMOI#	3A2E8/3A6E8		Contents of Memory OI Register 12	
MAXSWCH	3A2F4/3A6F4		Number of highest memory degrade switch in either up or down position	

Control Words

Tag	Word ADDR Hexadecimal	Description
	CPU0/CPU1	
CWO	3A000/3A800	Program name/type/monitor identification
CW3	3A003/3A803	Dynamic error counter
CW4	3A004/3A804	Pass counter
CW5	3A005/3A805	Current section executing
CW6	3A006/3A806	Current subsection
CW7	3A007/3A807	Current condition

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

Operator Communication

Displays

Unless otherwise specified, displayed values are in decimal.

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 (that is, *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)

Running Display with Memory Degraded

When memory is degraded by setting switches in the Memory OI Register (bits 16-22), the amount of usable memory is less than the amount installed. In this case, the following two lines are appended to the running display beginning on line 6. FWA and LWA are automatically adjusted.

```
                                                                    Line
RUNNING WITH DEGRADED MEMORY                                          6
USABLE MEMORY = xxxxx (HEX MBYTES)                                    7
```

Set Parameters Display

The test command LIST produces the following message:

```
                                                                    Line
CMEM   SET PARAMETERS, WORD ADDR = xxxxxxxx                           3
                                                                      4
PW00 = XXXX XXXX XXXX XXXX      TEST/DIAG CONTROL BITS                5
PW01 = XXXX XXXX XXXX XXXX      REPEAT COUNTER                        6
PW02 = XXXX XXXX XXXX XXXX      SECTION SELECT BITS (63-00)           7
PW03-PW15 - NOT USED                                                  8
PW16 = XXXX XXXX XXXX XXXX      MEMORY SIZE IN HEX MEGABYTES          9
PW17 = XXXX XXXX XXXX XXXX      ERROR HANDLER PARAMETER              10
PW18 = XXXX XXXX XXXX XXXX      USABLE MEMORY IF DEGRADE             11
                                                                     12
MEMORY AVAILABLE FOR TESTING                                         13
    XXXXXXXXX-YYYYYYYYY                                              14
ENTER ADDR,FWA,LWA TO CHANGE TESTING AREA                            15
                                                                     16
EC,XXXXXX,MEM SIZE TO CHANGE MEMORY SIZE                             17
```

where:

WORD ADDR Real memory word address of parameter area (hexadecimal).

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

The memory size can be specified different from that calculated by using the CMSE EC command as directed on the display.

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
```

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 Error Messages later in this chapter.

```
                                                                    Line
CMEM  SE  PCxxxx  Sxxxx  SBxxxx  Cxxxx                              3
TE = xxxx                                ERROR DESCRIPTION          4
```

where:

SE	Stopped on error
PCxxxx	Pass count
Sxxxx	Failing section number
SBxxxx	Failing subsection number
Cxxxx	Failing condition number (not implemented)
TE	Total error count
ERROR DESCRIPTION	Data compare error Single bit error Double bit error

Operator Entries

CMEM uses the standard CMSE commands as described in the MSL15X reference manual. For convenience, CMEM also provides the following parameter and run directives.

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 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.

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.

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.

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 8 of any current display.

Data Error Messages

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

Double-Bit Errors

When CMEM detects a double-bit error, it displays the standard error display on lines 3 and 4, descriptive data on lines 6 and 7, an additional message on line 9, an error message on line 10, and a continuation message on line 11.

The following figure shows the format of a double-bit error message.

```
                                                                    Line
CMEM  SE  PCXXXX  SYYYY  SBZZZZ  CXXXX                               3
TE = XXXX                                ERROR DESCRIPTION          4
                                                                    5
WORD ADDR          EXP                          RCV                 6
          XXXX XXXX XXXX XXXX            XXXX XXXX XXXX XXXX       7
                                                                    8
 Additional Message                                                 9
 Error Message                                                     10
 Next Message                                                      11
```

where:

WORD ADDR	Address of word in error (hexadecimal)
EXP	Expected result (hexadecimal)
RCV	If a double-bit error occurs when the CPU is reading memory, CMEM cannot retrieve and display the data. This field is left blank.
Additional Message	When a single- or double-bit error occurs in section 10, the following message is displayed: `NOTE: ADDR is first of 16 word block in which error occurred`
Error Message	Depending on the type of double-bit error, one of the following messages appears on line 10. Unless otherwise stated, the WORD ADDR field is filled in with a value.

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. The ADRS field has an N/A below it for this type of error.
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 display the bad data.

Next Message As a further aid to the operator, the following message is displayed beginning on line 11:

```
REFER TO THE MAINT REGISTERS FOR
MORE DETAILS (AR,M or AR,P) TYPE CE,M
(CE,P) TO CLEAR MEMORY (PROCESSOR)
ERROR LOGS.
```

Single-Bit Errors

When CMEM detects a single-bit error and PARAM17 is nonzero, CMEM displays the standard error message on lines 3 and 4, descriptive data on lines 6 and 7, and one of the following messages on line 10. The RCV and EXP fields are equal because the error has been corrected.

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).

Data Compare Errors

When CMEM detects a data compare error (data read is not equal to data written), CMEM displays the standard error message on lines 3 and 4, and additional information on lines 6 and 7.

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.

Applications

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.

Scope Mode

Use the SSM command to set scope mode on the section being run. The loop continues 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.

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.

Random Command Test 1 - RCT1 2

Introduction 2-1

Requirements 2-1
- Hardware 2-1
- Software 2-1
- Accessories 2-1
- Characteristics 2-2

Operational Procedure 2-3
- Restrictions and User Cautions 2-3
- Loading Procedure 2-3
- Parameters and Control Words 2-4
  - Parameters 2-4
- Additional Parameter Descriptions 2-6
- Control Words 2-7
- Section Index 2-7

Operator Communication 2-8
- Displays 2-8
  - Header/Parameter Display (DS,1) 2-8
  - Register/Memory Buffer Displays (DS,2-5) 2-10
  - Initial Data Displays (DS,6-9) 2-11
  - General Memory Display (DS,A) 2-12
- Operator Entries 2-12
  - Commands 2-12
- Running Procedures 2-14
    - Load and Run RCT1 2-14
    - Change Test Parameters 2-14
    - Stop on Error 2-15
    - Loop on Failing Set of Instructions/Operands/Pad/Purge 2-15
    - Scope Loop on a Failing Set of Instructions/Operands/Pad/Purge 2-15
    - Stop Test After Executing One Set of Random Instructions/Operands 2-15
    - Stop Test After Executing One Type of Purge/Pad Count 2-15
    - Display Parameter or Error Information 2-16
    - Stop Test While It Is Executing 2-16
- Normal Messages 2-16
- Error Messages 2-16

Random Command Test 1 - RCT1 2

Introduction

RCT1 is a set of model independent random command tests of 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 VEXC, required to establish the CPU execution and environment and to facilitate communication with the Common Maintenance Software Executive (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.

Requirements

Hardware

This test is intended for models 810, 815, 825, 830, 835, 840, 845, 850, 855, 860, and 990 computer systems.

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

- 1 CPU
- 1 Megabyte of central memory

Software

This test runs under control of the CMSE. Interfaces are handled by CMSE and the virtual level executive (VEXC).

Accessories

None required.

Characteristics

Description	Characteristics
Program name	RCT1
Size (source)	5,740 lines
Size (memory required for execution)	261,400 bytes
Code type	CPU
Run time (default)	Note[1]
Run time (quick look)	N/A
Run time (all sections)	Note[1]
Level of isolation	Detection
Off-line test	Yes
Off-line system	CMSE (MSL15X)
On-line system	NOS/VE-DVS
Assembly language	META
Source code maintenance	SCU

1. Runs until halted by the operator.

Operational Procedure

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.

Loading Procedure

RCT1 is assembled using the virtual machine assembler. The binary output is linked to the assembled binary of VEXC 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 models 810/815/825/830, 2 for 835, 3 for 840/845/850/855/860, and 5 for 990) exists on the MSL15X 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 chapter of the MSL15X 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 chapter of the MSL15X 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 is displayed in word format during test initialization.

Parameters and Control Words

Parameters

Parameter bits can be altered using standard CMSE commands such as EC, EB, SLE, CLE, and SRT or using RCT1 commands. Refer to the MSL reference manual for descriptions of CMSE commands. The RCT1 commands are described under Operator Entries.

Tag	Relative Byte Address (HEX)	Bits	Description	Default Setting
PW0	66100	0-63	Test control bits	0000 0000 0801 4010
		0-35	Not used	
		36	Monitor (VEXC) default mode forces running of test with default pit/sit settings, # of pages etc.	1
		37	Monitor (VEXC) bypass	0
		38-45	Not used	0
		46	On-line control. Set if running under DVS	0
		47	Off-line control	1
		48	Operator stop (S - KEY)	0
		49	Initial stop for parameters	1
		50	Display errors only	0
		51	Bypass all messages	0
		52	Not used	0
		53	Not used	0
		54	Repeat condition on current pad/purge count and random number (SRC/CRC)	0
		55	Not used	0
		56	Not used	0
		57	Repeat test/pass-run through all pad/purge counts with same random number (SRT/CRT)s	0
		58	Log errors (SLE/CLE)	0
		59	Stop on error (SSE/CSE)	1
		60	Stop on condition (SSC/CSC)	0
		61	Not used	0
		62	Not used	0
		63	Stop at end of test/pass (SST/CST)	0
PW1	66108	0-63	Repeat counter	-1
PW2			Not used	
PW3			Not used	
PW4			Not used	
PW5			Not used	
PW6	66130	48-63	Display select (set by using DS,X commands)	1

Tag	Relative Byte Address (HEX)	Bits	Description	Default Setting
PW7	66138	0-63	Skip pass count (set by P,XXXX command)	0
PW8	66140	32-63	Process Interval Timer (PIT) value (set by PV,XXX command)	100(16)
PW9	66148	32-63	System Interval Timer (SIT) value (set by SV,XXX command)	100(16)
PW10			Not used	
PW11			Not used	
PW12	66160	0-63	Monitor (VEXC) control bits (see Additional Parameter Descriptions for more detail)	F00000010(16)
PW13	66168	0-63	Scope loop control bit set by LOOP or SSM command Clear by using EC/EB command (see Running Procedures for more detail)	 0
PW14			Not used	
PW15			Not used	
PW16	66180	0-63	Trim instruction list on error (use TRM/NTRM commands to control)	1
PW17	66188	0-63	Pad count (maximum number of pads to insert between instructions)	A(16)
PW18	66190	0-63	Instruction select control (use NORM/OMIT/SEL commands)	0
PW19	66198	0-63	PIT control (use PITON/PITOF commands)	1
PW20	661A0	0-63	SIT control (use SITON/SITOF commands)	1
PW21	661A8	0-63	User mask control (use MSK/NMSK commands)	1
PW22	661B0	0-63	Instruction list length (use L,XX command)	1F(16)
PW23	661B8	0-63	Intermittent count determines how many times to retry on a problem before giving up.	FF(16)
PW24	661C0	62	Pad control Use PAD to enable padding Use NPAD to disable padding	1
		63	Purge control Use PURG to enable purging Use NPURG to disable purging	1
PW25	661C8	0-63	Optional instruction list (16 CM words)	0

Additional Parameter Descriptions

The parameters are described in more detail as follows:

PW7 Skip Pass Parameter (P,X)

PW7 skips forward to the specified pass count prior to test execution. This allows correct random number generation for any pass count selected.

PW6 Display Select Parameter (DS,X)

For DS,A requests (general memory display), bits 00-47 hold the PVA of the area in memory for display. The default is the starting PVA of the main test section. The byte portion can be altered to display higher portions of memory.

PW12 Monitor Control Bits

The parameter and control word segment is shared with RCT1 (job) and VEXC (monitor). PW12 is used by VEXC to set up the running environment. PW12 is set to $F00000010_{16}$ during the compile of RCT1. PW0 bit 36 interacts with PW12. The default is PW0 bit 36 set. When set, VEXC looks at PW12 for environment control bits. These can be changed in the command buffer using CMSE EC or EB commands.

Bit definitions for PW12 are as follows:

Bit	Meaning	Default
28	Run with repetitive page faults	1
29	Run with cache purges	1
30	Run with map purges	1
31	Run with page faults	1
32-63	Number of pages in working set	10_{16}

PW17 Pad Count

When run in default mode, pad instructions (1AFF) are inserted, one at a time, between each test instruction. From one to a maximum of ten pads may be selected. The use of pads creates timing conflicts for more thorough testing. The pad count can be decreased by using CMSE EC or EB commands.

PW23 Intermittent Count

If RCT1 detects an error but loses it during the trimming process, the test reruns the instruction list the number of times specified in PW23 to check for intermittency. If the error is not caught again during this loop, the test reports an intermittent error that was not repeatable. This count can be increased or decreased using CMSE EC or EB commands.

PW24 Pad/Purge Control

PW24 contains two bits, 62 and 63, that control if padding/purging is enabled/disabled. These bits can be controlled by operator entries of PURG/NPURG (enable/disable purging) and PAD/NPAD (enable/disable padding).

Bit 62 = 0	No padding is done.
Bit 62 = 1	Padding enabled.
Bit 63 = 0	No purging is done. Purge count is forced to a 1 which says do not purge.
Bit 63 = 1	Enable purging.

Control Words

The starting CM word address for the parameter block is displayed in the test parameter display (DS,1). The control words reside above the parameter words and are for information only. The address of each word is relative to zero (0).

Tag	Relative Byte Address (HEX)	Bits	Description	Default Setting
CW0	66000		Program Name	ASCII RCT1
CW1	66008		System Type	F800(16)
CW2	Not used			
CW3	66018		Error Count	0
CW4	66020		Pass Count	0
CW5-31	Not used			

Section Index

RCT1 has no selectable test sections.

Operator Communication

Displays

Ten RCT1 displays are defined. A header display presents the test name, pass count, pad/purge information, total error count, and PIT/SIT information. 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. One display presents a general memory display based on the address (PVA) in the left 48 bits of PW6.

Header/Parameter Display (DS,1)

When stop-for-parameters is set, or when a DS,1 command is entered, RCT1 presents a header/parameter display with the following format:

```
RCT1 - SET PARMS, WRD ADR=00CC20 85/09/30 -
PRM0  = 0000 0000 0801 4010 CONTROL/REPEAT/STOP BITS
PRM1  = 0000 7FFF FFFF FFFF REPEAT COUNT
PRM6  = B00C 0000 0000 0001 DISPLAY SELECT (0-A(16)), DS,X
PRM7  = 0000 0000 0000 0000 SKIP PASS COUNT, P,DDDDDD
PRM8  = 0000 0000 0000 0100 PIT VALUE (7FFFFFFF(16) MAX)
PRM9  = 0000 0000 0000 0100 SIT VALUE (7FFFFFFF(16) MAX)
PRM12 = 0000 000F 0000 0010 MONITOR CONTROL BITS
PRM13 = 0000 0000 0000 0000 SCOPE LOOP, LOOP=1, NO LOOP=0
PRM16 = 0000 0000 0000 0001 INSTRCTN LOOP TRM=1, NTRM=0
PRM17 = 0000 0000 0000 000A PAD COUNT (1-10)
PRM18 = 0000 0000 0000 0000 INST SELECT NORM/OMIT/SEL=0/1/2
PRM19 = 0000 0000 0000 0001 PIT CNTRL, ENABL=1, DISABL=0
PRM20 = 0000 0000 0000 0001 SIT CNTRL, ENABL=1, DISABL=0
PRM21 = 0000 0000 0000 0001 MASK UCR FP BITS, MSK=1, NMSK=0
PRM22 = 0000 0000 0000 001F INSTRCTN LIST (2-31 WRDS), L,XX
PRM23 = 0000 0000 0000 00FF INTERMITTENT COUNT (255 MAX)
PRM24 = 0000 0000 0000 0003 PAD/PURG=PAD/NPAD PURG/NPURG
```

where:

WRD ADR = Parameter word address in hex

When RCT1 begins execution, or when DS,0 is entered, RCT1 presents the following running display:

```
RCT1 CC PC=XXXXXX PADS=XX PURGE=AAAAAA TE=0000
PIT - VALU=XXXXXXXX +INTRUPT=XXXXXXXX OCCURNCS=XXX LOST=XXXXX
SIT - VALU=XXXXXXXX LOST=XXXXX
```

where:

cc	RU	Running
	SP	Parameter stop, end of pad/purge type, or end of pass stop
	SE	Error stop
	RT	Repeating test pass
	RC	Repeating pad/purge type (repeat condition)
	SM	Scope looping
	SC	Stop on condition
PC	Pass count (decimal)	
PADS	Pad count (1-10 decimal)	
PURGE	NONE, MAPS, CACHE, or ALL	
TE	Total errors	
PIT VALUE	Value in PIT (decimal)	
+INTRUPT	Value of PIT if positive interrupt	
OCCURNCS	Number of positive interrupts	
LOST	Number of times PIT interrupt didn't occur	
SIT VALUE	Value in SIT (decimal)	
LOST	Number of times SIT interrupt didn't occur	

Register/Memory Buffer Displays (DS,2-5)

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

Initial Data Displays (DS,6-9)

Four displays are available for presenting initial data information. The formats of the displays are similar except for the headings and the identifiers at the left of each line. All values are in hexadecimal. One such display is as follows:

```
                    A0-AF     INITIAL A AND X REGISTERS     X0-XF

REG-00 00000000 00000000                          00000000 00000000
REG-01 00000000 00000000                          00000000 00000000
REG-02 00000000 00000000                          00000000 00000000
REG-03 00000000 00000000                          00000000 00000000
REG-04 00000000 00000000                          00000000 00000000
REG-05 00000000 00000000                          00000000 00000000
REG-06 00000000 00000000                          00000000 00000000
REG-07 00000000 00000000                          00000000 00000000
REG-08 00000000 00000000                          00000000 00000000
REG-09 00000000 00000000                          00000000 00000000
REG-0A 00000000 00000000                          00000000 00000000
REG-0B 00000000 00000000                          00000000 00000000
REG-0C 00000000 00000000                          00000000 00000000
REG-0D 00000000 00000000                          00000000 00000000
REG-0E 00000000 00000000                          00000000 00000000
REG-0F 00000000 00000000                          00000000 00000000
```

Other headings and identifiers:

INSTRUCTION LIST

INSTR0
:
INSTRF

INITIAL INPUT BUFFER

WORD 0
:
WORD F

INITIAL OUTPUT BUFFER

WORD 0
:
WORD F

General Memory Display (DS,A)

The following display presents a requested portion of memory based on the PVA in the left 48 bits of PW6. The default address in PW6 is the starting address of the test section.

```
RNG=B   SEG=00C

00000000 00000000 00000000 00000080 00000000 00000000
00000008 00000000 00000000 00000088 00000000 00000000
00000010 00000000 00000000 00000090 00000000 00000000
00000018 00000000 00000000 00000098 00000000 00000000
00000020 00000000 00000000 000000A0 00000000 00000000
00000028 00000000 00000000 000000A8 00000000 00000000
00000030 00000000 00000000 000000B0 00000000 00000000
00000038 00000000 00000000 000000B8 00000000 00000000
00000040 00000000 00000000 000000C0 00000000 00000000
00000048 00000000 00000000 000000C8 00000000 00000000
00000050 00000000 00000000 000000D0 00000000 00000000
00000058 00000000 00000000 000000D8 00000000 00000000
00000060 00000000 00000000 000000E0 00000000 00000000
00000068 00000000 00000000 000000E8 00000000 00000000
00000070 00000000 00000000 000000F0 00000000 00000000
00000078 00000000 00000000 000000F8 00000000 00000000
```

where:

Each line contains: 32 bits of byte offset, 64 bits of data for that address, 16 bits of byte offset, and 64 bits of data for that address.

Operator Entries

Commands

The following RCT1 commands are provided:

Command	Function
ADD,x,y,...	Add opcodes x,y,... to optional instruction list. Opcodes 00-FF are accepted.
DEL,x,y,...	Delete opcodes x,y,... from optional 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-$1F_{16}$ are accepted. Default=$1F_{16}$.

Command	Function
DS,x	Select Display x. Display selections are 0-9. 0 = Running 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 A = General memory display
MSK	Select floating point bits in user condition mask. Default = set.
NMSK	Deselect floating point bits in user condition mask.
TRM	Trim execution list to failing instruction. Default = set.
NTRM	Deselect execution list trimming.
NORM	Test all instructions. Default = norm.
OMIT	Omit optional instruction list opcodes from test list.
SEL	Test only optional instruction list opcodes.
PAD	Enable padding. After each loop through the purge counts, a pad instruction will be inserted between the original set of instructions.
NPAD	Disable use of pad instructions.
PURG	Each set of random instructions will be run with purge instructions in front of it.
NPURG	Disable the use of purge instructions.
R	Restart with initial load parameters.
S	Stop test execution.
(space)	Continue test execution.
LOOP	Machine executes continuously present instruction/operand set (scope loop).
PITON	Enables running with small value in PIT. Default = on.
PITOF	Disables running with small value in PIT. PIT value is set to $FFFFFFFF_{16}$ for off-line, $7FFFFFFF_{16}$ for on-line and PW8 is ignored.

Command	Function
PV,X	Enter selected value into PIT. Default small PIT value = 256 microseconds (decimal). Input in decimal.
SITON	Enables running with small value in SIT. Default = on. Not valid on-line.
SITOF	Disables running with small value in SIT. SIT value is set to $FFFFFFFF_{16}$ for off-line and PW9 is ignored. Not valid on-line.
SV,X	Enter selected value into SIT. Default small SIT value = 256 microseconds (decimal). Input in decimal.
SSM	Same as LOOP command.
SSC/CST	Set/clear stop-at-end-of-condition parameter bit.
SRC/CRC	Set/clear repeat condition bit.
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.
SLE/CLE	Set/clear log errors in day file.

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

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 for parameter bit is set, RCT1 stops before the first pass to allow parameter changes. Entry of a space will resume RCT1 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 loaded 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/Pad/Purge

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

Type in CRC to clear the repeat test parameter bit.

Note that during repeat test operation, the error count will be incremented whenever an error is detected.

Scope Loop on a Failing Set of Instructions/Operands/Pad/Purge

After RCT1 has stopped on an error, type in an SRC command to set the repeat condition parameter bit and then type in LOOP or SSM to set the scope loop parameter. Press the space key and RCT1 will continually execute the failing set of instructions/operands/pad count/purge type.

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 the CM word displayed in the scope loop message.

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 and all pads/purges. 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.

Stop Test After Executing One Type of Purge/Pad Count

Type in SSC to set stop-at-end-of-condition parameter bit. RCT1 will stop after executing the current random instruction set and the current pad count/purge type.

Type in a space to execute next purge type/pad count set with same instruction set.

CSC will clear bit and RCT1 will execute pads and purges without stopping after each condition.

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 for display formats and for descriptions of the x parameter.

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.

Normal Messages

None provided.

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 the instruction buffer.
CANNOT TEST xx	RCT1 cannot simulate the instruction identified by the hexadecimal value xx.

Random Command Test 2 - RCT2 3

Introduction 3-1

Requirements 3-1
- Hardware 3-1
- Software 3-1
- Accessories 3-1
- Characteristics 3-2

Operational Procedure 3-3
- Restrictions and User Cautions 3-3
  - Loading Procedure 3-3
- Parameters and Control Words 3-4
  - Parameters 3-4
- Control Words 3-6
- Section Index 3-6

Operator Communication 3-7
- Displays 3-7
  - Running Display 3-7
  - Set Parameters Display 3-8
  - Error Running Display 3-8
  - Section List Display 3-9
  - Help Display 3-9
  - Memory Display 3-10
  - Instruction List Display 3-10
  - Results Display 3-10
  - Operand Display 3-11
- Operator Entries 3-12
  - Commands 3-12
- Running Procedures 3-14
    - Load and Execute RCT2 3-14
    - Change Test Parameters 3-14
    - Scope Loop 3-14
- Normal Messages 3-15
- Error Messages 3-15

Random Command Test 2 - RCT2 3

Introduction

RCT2 is a set of model independent random command tests of BDP virtual instructions. RCT2 uses fixed instruction sequences with random operands and descriptors.

RCT2 resides on the Maintenance Software Library and is loaded by the Common Maintenance Software Executive (CMSE). The load module includes the object code of the virtual level executive VEXC, 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.

Requirements

Hardware

This test is intended for models 810, 815, 825, 830, 835, 840, 845, 850, 855, 860, and 990 computer systems.

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

- 1 CPU
- 1 Megabyte of central memory

Software

This test runs under control of CMSE. Interfaces are handled by CMSE and the virtual level executive (VEXC).

Accessories

None required.

Characteristics

Description	Characteristics
Program name	RCT2
Size (source)	8,033(10) lines
Size (memory required for execution)	36,700(16) bytes
Code type	CPU
Run time (default)	Note[1]
Run time (quick look)	N/A
Run time (all sections)	Note[1]
Level of isolation	Detection
Off-line test	Yes
Off-line system	CMSE (MSL15X)
On-line system	NOS/VE-DVS
Assembly language	META
Source code maintenance	SCU

1. Runs until halted by the operator.

Operational Procedure

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.

Loading Procedure

RCT2 is assembled using the virtual machine assembler. The binary output is linked to the assembled binary of VEXC 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 840/845/850/855/860 and 5 for 990) exists on the MSL15X 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 chapter of the MSL15X 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 chapter of the MSL15X 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 is displayed in word format during test initialization.

Parameters and Control Words

Parameters

Parameter bits can be altered by CMSE EC or EB commands or by RCT2 commands. Refer to the MSL15X reference manual for descriptions of CMSE commands. Test parameter words are located starting at central memory word address CC20(16). 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 0

Bit 63	Stop at end of test pass (SST/CST).
Bit 62	Stop at end of section (SSS/CSS). Cleared by default.
Bit 61	Not used.
Bit 60	Not used.
Bit 59	Stop on error (SSE/CSE). Set by default.
Bit 58	Log errors (SLE/CLE). Cleared by default.
Bit 57	Repeat test (SRT/CRT). Cleared by default.
Bit 56	Repeat section (SRS/CRS). Cleared by default.
Bits 54-55	Not used.
Bit 53	Scope mode (LOOP or SSM). Cleared by default.
Bit 52	Not used.
Bit 51	Bypass all messages (CMSE EC command). Cleared by default.
Bit 50	Display errors only (CMSE EC command). Cleared by default.
Bit 49	Initial stop for parameter entry. Set by default.
Bits 47-48	Not used.
Bit 46	On-line control (set only for on-line use, do not alter).
Bits 38-45	Not used.
Bit 37	Monitor bypass. Cleared by default.
Bit 36	Monitor default mode. Set by default.
Bits 0-35	Not used.

Parameter Word 1

Bits 0-63	Repeat Counter. Default value is 7FFF FFFF FFFF FFFF(16). The test will repeat and decrement the count until it reaches zero and will then exit to the end if the repeat parameter bit is not set.

Parameter Word 2

Bits 60-63	Section Select Bits. Default value is FFFF(16); all sections selected. Sections are numbered from right to left.
Bits 0-47	Not used.

Parameter Word 3, 4 and 5

Bits 0-63	Not used.

Parameter Word 6

Bits 0-63 Display Control. This parameter selects displays and their options; it is defaulted to display parameters when the test is initialized. Refer to Operator Entries for further information concerning display selection.

Parameter Word 7

Bits 0-63 Skip Pass Count. When starting the test this parameter may be set to a specified pass count and the conditions normally prevailing for that pass will occur on the first pass.

Parameter Word 8

Bits 32-63 PIT Value. This parameter determines the value entered into the PIT initially and after it times out. The default is 512(10).

Parameter Word 9

Bits 32-63 SIT Value. This parameter determines the value entered into the SIT initially and after it times out.

Parameter Words 10 and 11

Bits 0-63 Not used.

Parameter Word 12

Monitor Control Bits.
Bit 28 Run with repetative page faults, default=1.
Bit 29 Run with cache purges, default=1.
Bit 30 Run with map purges, default=1.
Bit 31 Run with page faults, default=1.
Bits 32-63 Number of pages in working set, default=0D(16).

Parameter Word 13

Bits 0-63 Scope Loop Control. This parameter should be set with the LOOP or SSM commands. Refer to Operator Entries for further information.

Parameter Words 14 through 18

Bits 0-63 Not used.

Parameter Word 19

Bit 63 PIT Control. This parameter should be enabled/disabled with the PITON/PITOF commands. Default is enabled.

Parameter Word 20

SIT Control. This parameter should be enabled/disabled with the SITON/SITOF commands. Default is enabled.

Control Words

Test control words are located starting at central memory word address CC00(16).

Control Word 0 - Program identification
Control Word 1 - System type
Control Word 2 - Not used
Control Word 3 - Error count
Control Word 4 - Pass count
Control Word 5 - Current section
Control Word 6 - Not used

Section Index

Section Number	Brief Description
0	BDP NUMBERIC SUM - OP 70
1	BDP NUMBERIC DIFFERENCE - OP 71
2	BDP NUMBERIC PRODUCT - OP 72
3	BDP NUMBERIC QUOTIENT - OP 73
4	BDP NUMBERIC SCALE - OP E4/E5
5	BDP NUMBERIC COMPARE - OP 74
6	BDP NUMBERIC MOVE - OP 75
7	BDP BYTE COMPARE - OP 77
8	BDP BYTE COMPARE COLLATED - OP E9
9	BDP BYTE SCAN - OP F3
10	BDP BYTE TRANSLATE - OP EB
11	BDP BYTE MOVE - OP 76
12	BDP BYTE EDIT - OP ED
13	BDP IMMEDIATE DATA MOVE - OP F9
14	BDP IMMEDIATE DATA COMPARE - OP FA
15	BDP IMMEDIATE DATA ADD - OP FB

Operator Communication

Displays

All values shown in displays are in hexadecimal.

Running Display

During test execution, RCT2 presents a display similar to the following.

```
RCT2 SS PC=000146 SECT=09 TE=0000 OPR SET=NUL
PIT - VALU=00000512 +INTRUPT=00000000 OCCURNCS=000 LOST=00000
SIT - VALU=00001024 LOST=00000
```

where:

Operation (op)	RU	Running section/test
	SE	Stopped on error
	SS	Stopped at end of section
	ST	Stopped at end of test
	SM	Scope mode loop
	LE	Log error
	RS	Repeat section
	RT	Repeat test
PC		Current pass count.
SECT		Section executing.
TE		Total errors (does not include SITs/PITs).
OPR SET		The NUL entry indicates errors have not been detected for any set of commands and their operands.
PIT - VALUE		Value loaded into PIT upon timeout.
PIT - +INTRUPT		Value of the PIT when the last false PIT interrupt occurred.
PIT - OCCURNS		Number of occurences of false PIT interrupts.
PIT - LOST		Number of occasions when the PIT decremented through zero without generating an interrupt.
SIT - VALUE		Value loaded into SIT upon timeout.
SIT - LOST		Number of occasions when the SIT decremented through zero without generating an interrupt.

Set Parameters Display

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

```
RCT2 - SET PARMS, WRD ADR=00CC20 85/07/31 -
PRM0  =  0000 0000 0800 4010 CONTROL/REPEAT/STOP BITS
PRM1  =  7FFF FFFF FFFF FFFF REPEAT COUNT
PRM2  =  0000 0000 0000 FFFF SECTION SELECT BITS
PRM3  =  RESERVED
PRM4  =  RESERVED
PRM5  =  RESERVED
PRM6  =  B00A 0000 0000 0001 DISPLAY SELECT
PRM7  =  0000 0000 0000 0000 SKIP PASS COUNT
PRM8  =  0000 0000 0000 0200 PIT VALUE
PRM9  =  0000 0000 0000 0400 SIT VALUE
```

The first line of this display gives the central memory word address of the first parameter word (PRM0). Also provided is the revision date of the version being used.

Error Running Display

When RCT2 detects an error, a message similar to that shown below is displayed. SIT and PIT faults will neither cause this display nor cause an error stop.

This display is actually the Running Display with the addition of a fourth line which indicates the type of BDP command that failed, its op code, and the section number. The last entry on the first line of the display indicates which set or sets of operands caused the failure. The set numbers may range from 0 through 9 depending upon which test section is involved.

To view the failing operands or results, the operator may employ a display command which requires the use of the appropriate set number. See Operator Entries for further information.

```
RCT2 SE PC=000146 SECT=09 TE=0001 OPR SET=1,2
PIT - VALU=00000512 +INTRUPT=00000000 OCCURNCS=000 LOST=00000
SIT - VALU=00001024 LOST=00000

SECT 9 - BDP BYTE SCAN - OP F3
```

Section List Display

A display of the sections executed by RCT2 is shown below. With the exception of section 4 which tests the BDP Scale commands SCLN (op code E4) and SCLR (op code E5), each section tests a single command type. Depending on the section number, a group of 1, 3, 6, 7, or 10 commands are executed sequentially with their random operands. Each command and its operands are referred to as a set.

The instruction list defines the range of set numbers for each command type or section number.

```
SEC 0  - BDP NUMBERIC SUM - OP 70                    SETS 0-9
SEC 1  - BDP NUMBERIC DIFFERENCE - OP 71             SETS 0-9
SEC 2  - BDP NUMBERIC PRODUCT - OP 72                SETS 0-9
SEC 3  - BDP NUMBERIC QUOTIENT - OP 73               SETS 0-9
SEC 4  - BDP NUMBERIC SCALE - OP E4/E5               SETS 0-9
SEC 5  - BDP NUMBERIC COMPARE - OP 74                SETS 0-9
SEC 6  - BDP NUMBERIC MOVE - OP 75                   SETS 0-9
SEC 7  - BDP BYTE COMPARE - OP 77                    SETS 0-5
SEC 8  - BDP BYTE COMPARE COLLATED - OP E9           SETS 0-2
SEC 9  - BDP BYTE SCAN - OP F3                       SETS 0-2
SEC 10 - BDP BYTE TRANSLATE - OP EB                  SET  0
SEC 11 - BDP BYTE MOVE - OP 76                       SET  0
SEC 12 - BDP BYTE EDIT - OP ED                       SET  0
SEC 13 - BDP IMMEDIATE DATA MOVE - OP F9             SETS 0-9
SEC 14 - BDP IMMEDIATE DATA COMPARE - OP FA          SETS 0-6
SEC 15 - BDP IMMEDIATE DATA ADD - OP FB              SETS 0-9
```

Help Display

The RCT2 Help display provides a complete list of all operator entered commands which are decoded and acted upon by RCT2.

```
                              RCT2 COMMANDS
S - STOP TEST                    P,XXXXXX - SET SKIP PASS COUNT(10)
C OR SPACE - CONTINUE            PV,XXXXXXXX - SET PIT VALUE(10)
C - RESTART TEST                 PV,XXXXXXXX - SET PIT VALUE(10)
PITON - ENABLE PITS                     DISPLAY SELECTIONS
PITOF - DISABLE PITS             DS,RUNNING
SITON - ENABLE SITS              DS,PARAMET
SITOF - DISABLE SITS             DS,SECTION
SST/CST - STOP END TEST          DS,MEMORY,RING,SEGMENT,BYTE
SSS/CSS - STOP END SECTN         DS,RESULTS,SET,HALF       SEE NOTES
SSE/CSE - STOP ON ERROR          DS,OPERAND,SET,HALF       SEE NOTES
SLE/CLE - LOG ERRORS             DS,INSTRUC
LOOP - SET SCOPE MODE            DS,HELP
SRT/CRT - REPEAT TEST
SRS/CRS - REPEAT SECTION                      NOTES
                                 REFER TO THE RUNNING DISPLAY FOR
                                 THE FAILING SET NUMBER.
                                 A VALUE OF 1 OR 2 SELECTS THE FIRST
                                 OR SECOND HALF OF THE DISPLAY.
```

Memory Display

The memory display provides a display of two blocks of 16 CM words set side by side. Each block consists of three columns, the first of which is the BN field of the PVA. The two adjacent columns are the contents of the PVA. A header at the top of the display provides the corresponding ring and segment numbers for all 32 words of data.

If the display is selected without specifying the ring, segment, and byte number, then the machine instruction list is displayed using default values. This pertains only to the first use after the test is initiated.

Instruction List Display

The instruction list display shows the format and content of the fields of the BDP command and its descriptor. One line of data is given for each of up to ten sets of commands and descriptors.

The display below is for section 9 which has sets 0, 1, and 2. If the command uses only one descriptor, one descriptor field is blank.

```
SCNB,XO          AK,X1,AI,D

SET  COMMAND          DESCRIPT J      DESCRIPT K
     OP J K I D       F T L  Q        F T L  Q

0    F3 4 8 3 A10                     0 0 36 0710
1    F3 4 8 3 A30                     0 0 36 0710
2    F3 4 8 3 810                     0 0 36 0710
```

This display is generated from the actual machine list. It omits commands in the list which save result data or perform other housekeeping functions. Generally there are one or two commands before or after each BDP command in the list.

Results Display

The results display provides hex data in three columns with the headings EXPECTED, ACTUAL, and DIFFERENCE. The first line gives the name of the display and the selected operand set number.

The data displayed is for the currently active section. The number of data bytes displayed in each column is dependent upon the section or command being tested; generally 256 bytes are displayed for byte type commands and 40 for others. Also, note that because of the maximum lengths specified by the different data types and the random length specified within these limits, the number of active bytes is usually less than the number displayed. For some commands, such as compares, the expected and actual results are contained in a register and therefore, only eight bytes may be displayed. For those sections where 256 bytes are used, the display is provided in a first and second half selectable by the operator.

Operand Display

The operand display provides hex data in either two or three columns. The first line gives the name of the display and the selected operand set number. The data displayed is for the currently active section.

The headers for each column are dependent upon the type of BDP command being tested. The command mnemonic and headers for each column are specified for each section in the following list.

Sec.	Instr.	Column 1	Column 2	Column 3
0	ADDN	SOURCE	DESTINATN-INIT	DESTINATN-FINAL
1	SUBN	SOURCE	DESTINATN-INIT	DESTINATN-FINAL
2	MULN	SOURCE	DESTINATN-INIT	DESTINATN-FINAL
3	DIVN	SOURCE	DESTINATN-INIT	DESTINATN-FINAL
4	SCLN/R	SOURCE	DESTINATN-FINAL	NUL
5	CMPN	SOURCE	DESTINATION	X1R RESULT
6	MOVN	SOURCE	DESTINATION	NUL
7	CMPB	SOURCE	DESTINATION	X0R/X1R RESULT
8	CMPC	SOURCE	DESTINATION	TRANSLATE TABLE
9	SCNB	BIT ARRAY	DESTINATION	X0R/X1R RESULT
10	TRANB	SOURCE	TRANSLATE TABLE	DESTINATION
11	MOVB	SOURCE	DESTINATION	NUL
12	EDIT	SOURCE	EDIT MASK	DESTINATION
13	MOVI	IMMEDIATE DATA	DESTINATION	NUL
14	CMPI	IMMEDIATE DATA	DESTINATION	X1R RESULT
15	ADDI	IMMEDIATE DATA	DESTINATION-INIT	DESTINATION FIN

Comments concerning the number of bytes displayed for each column in the results display also apply to the operand display. The bit array used by the SCNB command in section 9 is displayed as an array of 16 by 16 bits rather than as 32 hex bytes. The array is complete on the first half of the display and is repeated on the second half.

Operator Entries

Commands

The following RCT2 commands are provided:

Command	Function
P,xxxxxx	Skip pass count. Allows test to skip forward a specified count prior to test execution. Parameter x equals the specified pass count and is entered into PW7.
R	Restart test with initial parameter settings.
S	Stop test execution.
(space)	Start/continue test execution.
LOOP OR SSM	Set scope loop; causes test to continuously execute instructions/operands in the section. Before entering the scope loop, the operator is shown how to regain control from the 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.
SRS/CRS	Set/clear repeat section parameter bit.
SLE/CLE	Set/clear log errors parameter bit.
PITON	Enables operation of the PIT. Parameter word 19 is set non-zero.
PITOF	Disables operation of the PIT. Parameter word 19 is set to zero.
SITON	Enables operation of the SIT. Parameter word 20 is set non-zero.
SITOF	Disables operation of the SIT. Parameter word 20 is set to zero.
PV,xxxxxxxx	Sets the specified value into parameter word 8 to define the required PIT value.
SV,xxxxxxxx	Sets the specified value into parameter word 9 to define the required SIT value.
	NOTE The keyboard entry for any display (DS) command may be simplified by omitting the bracketed characters.
DS,RU[NNING]	Selects the running display.
DS,P[ARAMET]	Selects the parameter display.
DS,S[ECTION]	Selects the section list display.

Command	Function
DS,M[EMORY]	Ring, segment, byte. Selects the memory display. Either the "ring" parameter or both the "ring" and "segment" parameter or the "ring" and "segment" and "byte" parameters may be omitted. Default or previously entered values will be used. Only hex values are accepted. RCT2 has three segments (not including VEXC) which are as follows: PARAMS Ring = B Segment = C Includes parameters, messages and displays. BINDING Ring = B Segment = 9 TESTCODE Ring = B Segment = A Includes data, literals, and all executable code. Refer to microfiche of the RCT2 listing to obtain byte number data for the BYTE parameters.
DS,O[PERAND], set, half	This command selects a display of the operands for the currently active section. The use of "set" and "half" parameters is the same as for the RESULTS display described below.
DS,I[NSTRUC]	This command selects and displays the BDP commands in the instruction list for the currently active section. One BDP command only occurs for each set. Commands other than BDP commands are not displayed. See the description of the Instruction List Display for additional details.
DS,H[ELP]	This command selects a display which shows the format of all keyboard entries which are decoded and acted upon by RCT2.
DS,RE[SULTS], set, half	This command selects the results display for the currently active section. The "set" parameter may have a value of 0 through 9 or may be omitted. When an error has occurred the operator would normally check the results for each failing set which is listed at the end of the first line of the running display. The "set" parameter must be included if the "half" parameter is to be employed. The "half" parameter should only be used for sections 7 through 12 where there is insufficient space for all data on one display. The "half" parameter can be omitted for the first half. A value of 2 must be specified for the second half.

Running Procedures

Load and Execute RCT2

The command buffer for RCT2 on the MSL should be employed to load and execute the test. The required command buffer will have a name of 5 to 7 characters in length. The first 4 characters will be "RCT2". Use the CMSE command AG to examine the list of available command buffers. It is recommended that the CMSE command AE be used to examine the selected buffer for any informative messages that it may contain. Use the command GO,xxxxx (where xxxxx is the command buffer name) to initiate execution.

Change Test Parameters

Parameter word 0 is defaulted to cause a stop prior to commencing execution so that parameter changes can be made. All parameters may be changed while the test is stopped or during execution except for monitor control (parameter word 12). This parameter may be altered only with an EC command entered in the command buffer prior to the point where the processor is deadstarted by the CMSE DK command.

If it is desired to have the test commence execution without a parameter stop, add an entry in the command buffer for parameter word 0. This entry should be set so that bit 49 is clear. Note that such an entry may already exist in the command buffer for some systems.

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 in command buffer) are saved and reinstated whenever RCT2 is restarted by typing an R command.

Scope Loop

A scope loop may be set up for any section of RCT2. If the test is stopped on an error (SE) or stopped at the end of a section (SS), the LOOP or SSM command may be entered to set the scope mode. Use of the space bar allows the test to resume the scope loop after a stopped condition.

Just prior to entering the loop, the scope mode flag (SM) will be set on the running display and an additional line will be displayed:

```
LOOPING - TO REGAIN CONTROL ENTER EC,XXXX,0
```

Execution of this command allows the test to exit the loop and stop. If the error is persistent, and stop-on-error is set, the test will halt with the SE flag set. If the error was not detected on last loop, the test is preconditioned to stop with the SS flag set even though there is no stop set for end of section.

The LOOP or SSM command must be re-issued to re-enter the scope loop. The SRS (set repeat section) command is not required to cause the test to enter the scope loop.

Normal Messages

None provided.

Error Messages

When RCT2 receives an improperly formatted command, 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.
BAD DISPLAY NAME	The display name which is the first parameter of the display command (DS) was not recognized by the test. Refer to the list of commands under Operator Entries.

Fixed Operand Command Tests - FCT1,2,3,5 4

Introduction 4-1
Test/Loop Relationship 4-2

Requirements 4-2
Hardware 4-2
Hardware Required to Run Test 4-2
Software 4-2
Accessories 4-2
Characteristics 4-3

Operational Procedure 4-4
Restrictions and User Cautions 4-4
PSM and PTL Masks 4-4
Real/Virtual Memory Addressing Modes 4-4
Loading Procedures 4-5
Loading of FCT1 4-5
Model Dependent Command Buffer Names 4-5
Loading of FCT2 4-6
Model Dependent Command Buffer Names 4-6
Loading of FCT3 4-6
Loading of FCT5 4-7
Model Dependent Command Buffer Names 4-7
Parameters and Control Words 4-8
Parameters 4-8
FCT1 and FCT5 Parameter Words 4-8
FCT2 Parameter Words 4-10
FCT3 Parameter Words 4-10
Control Words 4-13
FCT1 and FCT5 Control Words 4-13
FCT2 Control Words 4-13
FCT3 Control Words 4-14
Data Patterns 4-15
Test and Section Index 4-16
FCT1 Test Sections 4-16
FCT2 Test Sections 4-19
FCT3 Test Sections 4-21
FCT5 Test Sections 4-29

Operator Communication 4-30
Displays 4-30
FCT1 and FCT5 Displays 4-30
Initial Display 4-30
Running Display 4-30
Data Comparison Error Display 4-31
System Register Error Display 4-33
Processor Hung Error Display 4-33
Program Error Display 4-34
FCT2 Displays 4-35
Initial Display 4-35
Running Display 4-35
Error Display 4-36

FCT3 Displays 4-37
Initial Display 4-37
Running Display 4-37
Standard Error Display 4-38

Operator Entries 4-41
FCT1 and FCT5 Commands 4-41
FCT2 Commands 4-42
FCT3 Commands 4-42
Normal Messages 4-42
Error Messages 4-42
FCT1 and FCT2 Error Messages 4-42
FCT2 Error Messages 4-42
FCT3 Error Messages 4-42
Common Routine Error Messages 4-43
Subsection Omitted Messages 4-43
Applications 4-44
Scope Mode Control of FCT1 and FCT5 4-44

Fixed Operand Command Tests - FCT1,2,3,5 4

Introduction

FCT1,2,3, and 5 tests detect failures within the processor at the instruction level.

FCT1 is an IOU based, brief test of 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 chapter.

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. Any one test does not necessarily test all instructions, and 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.

Test/Loop Relationship

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. The user can always fully understand the error messages without understanding how the loops are being used.

Requirements

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

Hardware

These tests are intended for models 810, 815, 825, 830, 835, 840, 845, 850, 860, and 990 computer systems.

Hardware Required to Run Test

In addition to requirements to run CMSE, FCT tests require the following:

- 1 CPU
- 1 Megabyte of central memory
- 1 PP

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.

Accessories

Microfiche of listings and test case descriptions.

Characteristics

Description	Characteristics
Program name	FCT1,2,3,5
Size (source) (see table 4-1)	250,000
Size (memory required) (see table 4-1)	1MB
Code type	PP, CP code
Run time (default) (see table 4-1)	
Run time (quick look) (see table 4-1)	
Run time (all sections) (see table 4-1)	
Level of isolation	Detect only
Off-line test	Yes
Off-line system	CMSE (MSL15X)
Resident during execution	PP, CP
Assembly language	PP Compass, META
Source code maintenance	ASCII-MODIFY, UPDATE
Uses maintenance channel	Yes

Table 4-1. FCT1,2,3,5 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 835/840/ 845/850/855/860
FCT1 - CP	42,000	1MB	20	114
FCT1 - IOU	5,000	4096		
FCT2 - CP	17,000	1MB	23	20
FCT3 - CP	166,000	1MB	Times Unavailable	Times Unavailable
FCT5 - CP	13,000	1MB	32	17
FCT5 - IOU	5,000	4096		

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.

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

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 MSL15X 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 chapter of the MSL15X 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 chapter of the MSL15X reference manual for procedures.

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).

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 MSL.

Model Dependent Command Buffer Names

Test Name	810/815 825/830	835	840/845 850/855/860
FCT1	FCT11	FCT12	FCT13

Refer to the MSL15X reference manual 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.

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 MSL.

Model Dependent Command Buffer Names

Test Name	810/815 825/830	835	840/845 850/855/860
FCT2	FCT21	FCT22	FCT23

Command buffers provided on 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.

Loading of FCT3

The product resides as seven files on 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 following table.

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, and so on. Refer to MSL15X 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.

You should set up other testing situations by varying DEC and the FCT paramter words. Keep in mind there are several bits in PARAMO which must be kept in agreement with the setting of DEC bits. See FCT3 Parameter Words. Errors are reported if FCT3 tries to test disabled hardware.

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 840/845/850/855/860 = 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

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 MSL.

Model Dependent Command Buffer Names

Test Name	810/815 825/830	835	840/845 850/855/860
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.

Parameters and Control Words

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 MSL15X reference manual. Also see FCT1 and FCT5 Commands later in this chapter for other commands that affect the parameter settings.

Word Address in Octal	Tag	Bit	Position Octal/Hex	*1	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

Note:

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

Word Address in Octal	Tag	Bit	Position Octal/Hex	*[1]	Meaning
127	PARAM5	48-63		*	Sections 15-00 select[2]
130	PARAM6	48-63		*	Sections 31-16 select[2]
131	PARAM7	48-63		*	Sections 47-32 select[2]
132	PARAM8	48-63		*	Sections 63-48 select[2]
133	PARAM9	48-63		*	Sections 79-64 select[2]
134	PARAM10	48-63		*	Sections 95-80 select[2]
135	PARAM11	48-63		*	Sections 111-96 select[2]
136	PARAM12	48-63		*	Sections 127-112 select[2]
137	PARAM13	48-63		*	Sections 143-128 select[2]
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

Notes:

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

2. All sections set by default.

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 Loading of FCT2 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 Octal	Tag	Bit	Position Octal/Hex	*[1]	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[2]

Note:

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

2. 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 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 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 repeat loop of 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 Octal	Tag	Bit	Position Octal/Hex	*[1]	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
		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

Note:

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

Word Address in Octal	Tag	Bit	Position Octal/Hex	*[1]	Meaning
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 840/845/850/855/860, 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

Note:

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

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 Loading of FCT3.

Recommended Command Buffer Name[1]	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

Note:

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

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 following control words for FCT1 and FCT5 are located at PP locations 102_8 to 121_8.

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. The following control words are located beginning at central memory 600_{16} word address.

Word Address in Hex	Tag	Bit	Position in Hex	*1	Meaning
600	CW 0	00-63		*	Test name FCT2 (in ASCII)
601	CW 1	00-15	FFFF000000000000	*	$2A00_{16}$, models 810, 815, 825, 830, 835, 840, 845, 850, 855, and 860 are supported
		16-31	FFFF00000000	*	$CCF8_{16}$, indicates options and configurations 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, not implemented
		32-63			Error code 2, 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

Note:

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

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	*1	Meaning
1400	CW 0	00-63		*	Test name FCT3 (in ASCII)
1401	CW 1	00-15	FFFF000000000000	*	$2A00_{16}$, models 810, 815, 825, 830, 835, 840, 845, 850, 855, and 860 are supported
		16-31	FFFF00000000	*	$CCF8_{16}$, indicates options and configurations 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
1402	CW 2	00-31			Error code 1, not implemented
		32-63			Error code 2, 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

Note:

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

Data Patterns

In the descriptions of the testing that is done by each test section 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

Test and Section Index

FCT1 Test Sections

Section	Tag	Instruction Mnemonic	Instruction Op Code	Reference	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	
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	

Section	Tag	Instruction Mnemonic	Instruction Op Code	Reference	Remarks
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	
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	Note[1]				
93	F176	MOVB	76	089	

Note:

1. Unused section (CALDF deleted).

Section	Tag	Instruction Mnemonic	Instruction Op Code	Reference	Remarks
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	

FCT2 Test Sections

Section	Tag	Instruction Mnemonic	Instruction Op Code	Reference	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	

Section	Tag	Instruction Mnemonic	Instruction Op Code	Reference	Remarks
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	
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).

FCT3 Test Sections

Section	Tag	Instruction Mnemonic	Instruction Op Code	Reference	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.
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.

Section	Tag	Instruction Mnemonic	Instruction Op Code	Reference	Remarks
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.

Section	Tag	Instruction Mnemonic	Instruction Op Code	Reference	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.

Section	Tag	Instruction Mnemonic	Instruction Op Code	Reference	Remarks
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.
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.

Section	Tag	Instruction Mnemonic	Instruction Op Code	Reference	Remarks
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[1]					
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).

Note:

1. Reserved sections.

Section	Tag	Instruction Mnemonic	Instruction Op Code	Reference	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[1]					

Note:

1. Reserved sections.

Section	Tag	Instruction Mnemonic	Instruction Op Code	Reference	Remarks
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.
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).

Section	Tag	Instruction Mnemonic	Instruction Op Code	Reference	Remarks
173	REF092	MOVN	75	092	Numeric move, D(Ak) replaced by D(Aj) after formatting.
174[1]					
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[2]					
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.

Notes:

1. Unused section (CALDF deleted).

2. Reserved sections.

Section	Tag	Instruction Mnemonic	Instruction Op Code	Reference	Remarks
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.
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).

FCT5 Test Sections

Section	Tag	Instruction Mnemonic	Instruction Op Code	Reference	Remarks
208-255[1]		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	

Note:

1. Reserved section.

Operator Communication

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

Displays

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 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

               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

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 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.

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
```

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 = FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF....
EXPECTED DEST 000B18 = FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF....
RECEIVED DEST 000F38 = 00FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF....
FAILING BYTE LIST  = 8000000000000000 0000000000000000
(ONE BIT PER BYTE)   0000000000000000 0000000000000000
DATA ERROR
```

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.

Operator Entries

FCT tests use standard CMSE commands as described in the MSL15X reference manual.

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 as follows:

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).

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 MSL15X reference manual.

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 MSL15X reference manual.

Normal Messages

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

Error Messages

FCT1 and FCT2 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.

FCT2 Error Messages

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

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

Line 2 of the display is also used for subsection omitted messages and for VLEX error messages. Refer to the MSL15X 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
OMITTED Ssss SSbb - REDUNDANT SUB-SECTION

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.

Line 2 is also used for common routine error messages and for VLEX error messages. Refer to the MSL15X reference manual.

Applications

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 behavior 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.

Virtual Mode Instruction Level Test - FCT9 5

Introduction 5-1

Requirements 5-1
- Hardware 5-1
- Software 5-2
- Accessories 5-2
- Characteristics 5-2

Operational Procedure 5-3
- Restrictions and User Cautions 5-3
- Loading Procedure 5-3
  - Load product set microcode. 5-3
  - Loading Processor Dependent Overlays 5-4
  - Loading of Slave PPs 5-4
- Parameters and Control Words 5-6
  - Parameters 5-6
  - Control Words 5-8
- Section Index 5-8

Operator Communication 5-9
- Displays 5-9
  - Initial Display 5-9
  - Running Display 5-9
- FCT9 Error Messages 5-10
  - Data Comparison Error Message 5-10
  - Processor Hung Error Message 5-11
  - Program Error Message 5-11
  - System Register Error Display 5-12
  - Miscellaneous Error Messages 5-12
- Operator Entries 5-13

Virtual Mode Instruction Level Test - FCT9 5

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.

Requirements

Hardware

Equipment for which test is intended:

Models 810, 815, 825, 830, 835, 840, 845, 850, 855, 860, and 990 computer systems.

The test requires:

- 1 CPU
- Product set microcode
- 131K word of central memory
- IOU hardware required by CMSE
- 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.)

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.

Accessories

None required.

Characteristics

Description	Characteristics
Test name	FCT9
Test/diagnostic/utility/system	Test
Size (source)	6000 lines (for PP code) 10000 lines (for CP code)
Size (memory required for execution)	4096 PP words (for PP code) 131000 CM words minimum (for CP code)
Virtual code/microcode/PP code/other	Virtual and PP code
Run Time	Approx. 30 sec
Level of isolation	Detection
On-line/off-line/both	Off-line
Off-line system	MSL15X
Resident during execution	Yes
Language	CP: CYBER VM Assembler PP: Compass
Source code maintenance	CP: MODIFY (ASCII) PP: UPDATE
Uses maintenance channel	Yes

Operational Procedure

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, 840, 845, 850, 855, and 860 computer systems.

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 840/845/850/855/860) exists on the MSL15X 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 chapter of the MSL15X 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 chapter of the MSL15X reference manual for procedures.

The command buffer performs the following steps:

- Initialize system (that is, 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.

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.

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:

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

2. If PARAM 12 is zero, skip to step 3; otherwise, assign PPs as slaves according to parameters 11 thru 14 and skip to step 6.

3. 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.

4. 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 3.

5. 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.

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

7. 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.

8. 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.

Parameters and Control Words

Parameters

Parameter words control the execution of the test. Some of these words are defined in the MSL15X reference manual. 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 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 MSL15X reference manual.

The use of PARAM7 through PARAM18 are peculiar to FCT9. For further details, see Loading the Slave PPs.

PARAM 7 - Not used
PARAM 8 - Not used
PARAM 9 - Not used
PARAM 10 - Not used
PARAM 11 - First slave PP
PARAM 12 - Second slave PP
PARAM 13 - Third slave PP
PARAM 14 - Fourth slave PP
PARAM 15 - EXCHITC to slave Comm. channel
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

Octal Address	Parameter	Octal/Hex Bit	Meaning
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. Some processors implement these commands via microcode and therefore section 12 is not selected.

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

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)

Operator Communication

Displays

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
```

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.

FCT9 Error Messages

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.

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.

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 overwrite of the control tables in central memory as a result of a CP error.

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.

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.
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 Loading the Slave PPs.
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.

Message	Meaning
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 Loading the Slave PPs.
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.

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.
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.

Exchange Test - EXCH 6

Introduction 6-1

Requirements 6-1
Hardware 6-1
Software 6-1
Accessories 6-2
Characteristics 6-2

Operational Procedure 6-3
Restrictions and User Cautions 6-3
Loading Procedure 6-3
Loading Processor Dependent Overlays 6-3
Loading of Slave PPs 6-4
Load Map 6-5
Parameters and Control Words 6-6
Parameters 6-6
Control Words 6-8
Central Memory Control Flags and Data 6-8
Section Index 6-10

Operator Communication 6-11
Displays 6-11
Initial Display 6-11
Running Display 6-11
Error Message Display 6-12
Operator Entries 6-13
Normal Messages 6-14
Error Messages 6-14
Data Comparison Error Message 6-14
Processor Hung Error Message 6-15
Program Error Message 6-15
System Register Error Display 6-16
Miscellaneous Error Messages 6-16
Applications 6-17
Scope Mode Control and Triggering 6-17

Exchange Test - EXCH 6

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 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.

Requirements

Hardware

Equipment for which test is intended:

Models 810, 815, 825, 830, 835, 840, 845, 850, 855, 860, and 990 computer systems.

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

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

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.

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.

Accessories

None required.

Characteristics

Description	Characteristics
Test name	EXCH
Test/diagnostic/utility/system	Test
Size (source)	6000 lines (for PP code) 7000 lines (for CP code)
Size (memory required for execution)	4096 PP words (for PP code) 1 million bytes minimum (for central memory)
Virtual code/microcode/PP code/other	Virtual and PP code
Run time	1 minute
Level of isolation	Detection
On-line/off-line/both	Off-line
Off-line system	MSL15X
Resident during execution	Yes
Language	CP: CYBER VM Assembler PP: COMPASS
Source code maintenance	CP: MODIFY (ASCII) PP: UPDATE
Uses maintenance channel	Yes

Operational Procedure

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.

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 840/845/850/855/860) exists on the MSL15X 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 chapter of the MSL15X 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 chapter of the MSL15X 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.

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.

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:

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

2. If PARAM12 is zero, skip to step 3; otherwise, assign PPs as slaves according to parameters 11 through 14 and skip to step 6.

3. 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.

4. 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 3.

5. 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.

6. 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.

7. 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.

8. 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.

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 Central Memory Control Flags and Data for more information.

Parameters and Control Words

Parameters

Parameter words control the execution of the test. These words are defined in the MSL15X 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 chapter 18 of this manual.

Parameters are as follows:

Octal Address	Parameter	Octal/Hex 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

Octal Address	Parameter	Octal/Hex Bit	Meaning
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 = 000016_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

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

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 18). 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 chapter 18. 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 chapter 18. 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 chapter 18.

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
19	SECT19	UEM read/write and block copy commands

Operator Communication

Displays

Initial Display

The initial display presented by EXCH is 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
```

Running Display

The normal running display for EXCH is shown below.

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

where:

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 Operator Entries) 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.

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.

```
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

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.
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 microtraps	These commands provide a more convenient method of enabling or disabling the PFS microtraps as opposed to setting or clearing a bit in PARAM word 17.

Normal Messages

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

Error Messages

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

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 chapter 18 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.

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.

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.

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.

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 Loading of Slave PPs.
- 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 Loading of Slave PPs.
- 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.

Applications

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 chapter 18 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.

Trap Test - TRAP 7

Introduction 7-1

Requirements 7-1
- Hardware 7-1
- Software 7-1
- Accessories 7-1
- Characteristics 7-2

Operational Procedure 7-3
- Restrictions and User Cautions 7-3
- Loading Procedure 7-3
- Parameters and Control Words 7-4
  - Parameters 7-4
  - Control Words 7-6
  - Central Memory Flags, Pointers, and Exchange Packages 7-6
- Section Index 7-8

Operator Communication 7-9
- Displays 7-9
  - Trap Initial Display 7-9
  - Running Display 7-10
  - Error Message Display 7-11
- Operator Entries 7-11
- Normal Messages 7-13
- Error Messages 7-13
  - Data Comparison Error Message 7-13
  - Processor Hung Error Message 7-15
  - Program Error Message 7-15
  - System Register Error Display 7-16
  - Miscellaneous Error Messages 7-16
- Scope Mode Control and Triggering 7-17

Trap Test - TRAP 7

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 Monitor Condition Register (MCR) and User Condition Register (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 maintenance access control (MAC) channel 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.

Requirements

Hardware

Equipment for which test is intended: models 810, 815, 825, 830, 835, 840, 845, 850, 855, 860, and 990 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).

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.

Accessories

None required.

Characteristics

Description	Characteristics
Test name	TRAP
Test/diagnostic/utility/system	Test
Size (source)	6000 lines (for PP code) 11000 lines (for CP code)
Size (memory required for execution)	4096 PP words 1 million bytes minimum (for central memory)
Virtual code/microcode/PP code/other	Virtual and PP code
Run time	1 minute
Level of isolation	Detection
On-line/off-line/both	Off-line
Off-line system	MSL15X
Resident during execution	Yes
Language	CP: CYBER VM Assembler PP: COMPASS
Source code maintenance	CP: MODIFY· (ASCII) PP: UPDATE
Uses maintenance channel	Yes

Operational Procedure

Restrictions and User Cautions

None.

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, FCT0P1, FCT0P2, FCT0P3, FCT0P4, 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 840/845/850/855/860) exists on the MSL15X 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 chapter of the MSL15X 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 chapter of the MSL15X 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.

Parameters and Control Words

Parameters

Parameter words control the execution of the test. They are defined in the MSL15X 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 – chapter 19

PARAM 17 – Enable/Disable PFS microtraps – Operator Entires in this chapter

PARAM 18 – CM port select – chapter 19

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

Parameter	Octal Address	Octal/Hex Bit	Meaning
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 = 000000	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= 000000	Reserved

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

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 Operator Entries. 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 (sections 0, 1, and 34).

Item	CM Byte Hex Address	Description
MX0	5F80	An executive state monitor exchange package used to initiate job mode operation during job mode MCR/UCR bit tests (sections 0, 1, and 34).
MX1	6120	An executive state monitor exchange package used in monitor mode MCR/UCR bit tests (sections 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.

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 - PRIVILEGED 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 - NOT USED
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		990 PROCESSOR MODEL DEPENDENT
41		KEYPOINT FUNCTION TEST

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.

Displays

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 840/845/850/855/860 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
```

Running Display

The normal running display for TRAP is shown below.

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

where:

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 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.

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

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.

Action	Command	Description
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 data according to requirements of EPFS/DPFS command and copies it to DEC register.
TIxx	Execute Subset of Sections	This command selects one or more sections of the test directly. The xx is the number of the section to be selected. Thus TI02 selects section 2 of the test. An entry of TI02 03 28 29 30 would run 5 different sections of the test. More than one command entry can be used to build up a series of sections to be run. 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.

Normal Messages

None.

Error Messages

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

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 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
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 chapter 19 regarding Control Command Tables.

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
```

where:

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.

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.

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
```

where:

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

Line 12 REG ADRS Register address

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.

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 chapter 19 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.

Random Fast/Slow Test - RFST 8

Introduction 8-1

Requirements 8-1
- Hardware 8-1
- Software 8-1
- Accessories 8-1
- Characteristics 8-2

Operational Procedures 8-3
- Restrictions and User Cautions 8-3
- Loading Procedure 8-3
  - Running Procedure 8-3
- Parameters and Control Words 8-4
  - Parameters 8-4
  - Installation Parameters 8-6
  - Control Words 8-6
- Section Index 8-7

Operator Communication 8-8
- Displays 8-8
  - Initial Display 8-8
  - Running Display 8-9
  - Initial A/X Register Display 8-9
  - Instruction List Display 8-10
  - Error Displays 8-11
- Operator Entries 8-13
  - Test Commands 8-13
  - Using Test Commands 8-16
    - Change Test Parameters 8-16
    - Stop on Error 8-16
    - Repeat Failing Set of Instruction/Operands 8-16
    - Scope Loop a Failing Set of Instructions/Operands 8-16
    - Stop After Executing One Set of Random Instructions/Operands/Pad Count . 8-17
    - Stop After Executing One Set of Random Instructions/Operands 8-17
    - Repeat Test Pass 8-17
    - Stop Test While Executing 8-17
    - Display Error Information 8-17
    - Trim Control 8-17
- Normal Messages 8-18
- Error Messages 8-18
  - Recovery 8-18
- Applications 8-18
  - Scope Mode 8-18
  - Tracking Exchanges and Traps 8-18

Random Fast/Slow Test - RFST 8

Introduction

Random Fast/Slow Test (RFST) is a hardware maintenance test for all CYBER 180 computer systems. RFST is a model independent test designed to detect errors at the virtual instruction level. This test runs off-line under Common Maintenance Software Executive (CMSE) and on-line under the Diagnostic Virtual System (DVS).

RFST tests the computer's 180 instruction set by using a random set of instructions and random data buffers. This is called the fast loop. Pad instructions are inserted between instructions and the stream is executed again. This is the slow loop. Loops are continued, using one additional pad instruction between each instruction until 16 pads have been added. Results from each slow loop are compared with the results from the fast loop for any differences.

Requirements

Hardware

This test is intended for CDC® CYBER 170/180 Models 810, 815, 825, 830, 835, 840, 845, 850, 855, 860, and 990 computer systems.

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

Target Configuration:

- 1 C180 CPU with 8 MB of memory
- 1 Disk unit supported by CMSE for test residence
- 1 System console
- Cache/page maps/segment maps
- 20 PPs

Minimum Configuration:

- 1 C180 CPU with 1 MB of memory
- 1 Disk unit supported by CMSE for test residence
- 1 System console
- 5 PPs

Software

RFST runs under the CMSE.

Accessories

None required.

Characteristics

Description	Characteristics
Program name	RFST
Test/Diag/Utility	Test
Size of source lines including comments	6048
Size of memory required for test execution	1MB
Size of total assembled binary	123063
Run time	N/A[1]
Level of isolation	Detect Only
Off-line system	MSL15X
On-line system	DVS - NOS/VE
Test control resides in	CPU
Source language	META
Source code maintenance	SCU

1. RFST runs until halted by the operator with a target of 5 passes per minute.

Operational Procedures

Restrictions and User Cautions

1. All instructions must be independently operational for RFST to test instruction streaming. The following tests must run: FCT1, FCT2, FCT3, FCT5, RCT1, and RCT2.

2. The virtual addressing mechanism must be operational. RFST cannot run in RMA mode.

3. RFST does not test A170 instructions.

RFST does not test some instructions, and therefore does not test some hardware. The RETURN instruction is used only in conjunction with a CALL instruction. The EXCHANGE instruction is not tested but is used to go from job to monitor when required. The SYNC instruction is used at the end of the instruction stream for scoping purposes. INTERRUPT, CPYXS, EXECUTE, and HALT are not tested.

Loading Procedure

RFST is assembled using the C180 assembler. The binary output is linked to the binary of VEXC using the C180 LINKER/LOADER to form the complete executable module. It can then be written to disk using TDX. To run off-line, the executable binary is loaded and executed using the CMSE command buffer capability. Consult the MSL15X reference manual for a description of the command buffer structure. To run on-line, consult the DVS On-line Usage manual for loading procedures.

Running Procedure

During normal execution, the test runs with default environment set-up values. If bit 36 of Parameter Word 0 (PWO) is clear, the operator can select values. In this case, VEXC displays the first messages and deals with test environment selection. For example, the first messages displayed could be, RUN WITH PAGE FAULTS? or RUN WITH LIMITED WORKING SET SIZE? Operator input is in the form of (Y)es/(N)o answers. If bit 36 of PWO is set, VEXC uses the information in PW12 to set up the test environment.

The first display from RFST is a parameter stop display. Refer to Test Commands for a list of available commands. Press the space bar to start random instruction testing.

Parameters and Control Words

Parameters

Tag	RMA Byte Address (Hex)	Bits	Description	Default Setting
	CPU0/CPU1			
PW0	A6100/A7100	0-63		00000000 08014010_{16}
		0-35	Not used	0
		36	Monitor-default run (CMSE EC command)	1
	(1)	37	Monitor bypass	0
	(1)	38	Test mode one	0
	(1)	39	Test mode two	0
		40-44	Not used	
	(2)	45	Batch mode	0
	(2)	46	On-line control	0
	(2)	47	Off-line control	1
		48	Operator stop	0
		49	Initial stop for parameters	1
		50-53	Not used	0
		54	Repeat present instruction/operand/pad count set (SRC/CRC)	0
		55-56	Not used	
		57	Repeat present instruction/operand set with pads 1-10 (SRT/CRT)	0
		58	Log errors (SLE/CLE)	0
		59	Stop on error (SSE/CLE)	1
		60	Stop end of condition	0
		61-62	Not used	0
		63	Stop at end of test (SST/CST)	0
			NOTE (1) Used in test development only. (2) Used by DVS – not for off-line use.	
PW1	A6108/A7108	0-63	Repeat counter	1000_{10}
PW2-P-W5			Not used	
PW6	A6130/A7130	0-63	Display select control (set by DS,x command)	0
PW7	A6138/A7138	0-63	Skip pass count (set by P,x command)	1

Tag	RMA Byte Address (Hex)	Bits	Description	Default Setting
PW8	A6140/A7140	0-63	PIT value (set by PV,X command)	250_{10}
PW9	A6148/A7148	0-63	SIT value (set by SV,X command)	200_{10}
PW10--PW11			Not used	
PW12	A6160/A7160	0-63	Monitor control bits See Installation Parameters for bit definitions	$F00000040_{16}$
PW13	A6168/A7168	0-63	Scope loop control (set by LOOP1/LOOP2 commands)	0
PW14--PW15			Not used	
PW16	A6180/A7180	0-63	Trim inst loop control (set by NTRIM/TRIM/TRIMR/ TRIMB/TRIMI default=TRIM)	1
PW17	A6188/A7188	0-63	Selected pad count (set by PAD,x command)	0
PW18	A6190/A7190	0-63	Instr selection control (set by NORM/OMIT/SEL –default = NORM)	0
PW19	A6198/A7198	0-6	PIT control (set by PITON/ PITOF command)	1
PW20	A61A0/A71A0	0-63	SIT control (set by SITON/ SITOF command)	1
PW21	A61A8/A71A8	0-63	User mask control	1
PW22	A61B0/A71B0	0-63	Instr list length500_{10}	
PW23	A61B8/A71B8	0-63	Optional instruction list (16 words)	0
PW24	A6238/14C47	0-63	PVA modification control	0

Locations of Additional Information

Additional information is obtained by displaying CM contents through the AB and AH commands of the CMSE. Addresses for instruction lists, and input/ output memory buffers are indicated in error displays.

Installation Parameters

The Parameter and Control word segment is a segment shared by RFST (job) and VEXC (monitor). VEXC uses PW12 to set up the running environment.

PW12 is set to $F00000040_{16}$ during the compile of RFST. PW0, bit 36 interacts with PW12. The default is PW0, bit 36 set. When set, VEXC looks at PW12 for environment control bits. These can be changed in the command buffer using the CMSE EC or EB command.

Bit definitions for PW12 are as follows:

Bit	Meaning	Default
28	Run with repetitive page faults	1
29	Run with cache purges	1
30	Run with map purges	1
31	Run with page faults	1
32-63	Number of pages in working set	40_{16}

Control Words

The starting RMA for the parameter word block is given in the initial parameter stop display and in the running message. Control words reside above parameter words.

Tag	RMA Byte Address (Hex)	Description	Default Setting
	CPU0/CPU1		
CW0	A6000/A7000	Program name	ASCII 'RFST'
CW1	A6008/A7008	System type (Not used)	0
CW2	A6010/A7010	Error code	0
CW3	A6018/A7018	Error count	0
CW4	A6020/A7020	Pass count	0
CW5-7		Not used	
CW8	A6040/A7040	Pad count	0
CW9-31		Not used	

Section Index

RFST is organized on a test pass structure rather than into sections or subsections. Each test pass consists of the following operations:

1. Create random instruction stream of up to 500 instructions long.
2. Create random A/X register values.
3. Create random input/output memory buffers.
4. Execute instructions and save all results (regs/mem/ucr).
5. Insert X pad instructions between instructions.
6. Restore input value to registers and memory.
7. Execute instructions with pads and save results.
8. Compare results from step 7 with results from step 4.
9. Do steps 5, 6, 7 and 8 using 0 through 16 pad instructions.

Each pad count is considered a condition and the operator can stop RFST at the end of each padded pass which is a stop at end of condition.

The instruction stream is trimmed to the shortest failing stream upon detection of an error in compare. Compare is done in step 8 above.

Operator Communication

Displays

Initial Display

```
RANDOM FAST/SLOW TEST       CPU X xxxx PAGES x.x  mm/dd/yy
RFST   CC   PC=xxxxxxxx   EC=xxxx   TE=xxxx PADS=xx   PA=xxxxx
(COMMENT FIELD)                     PV=xxxxxxxx, SV=xxxxxxxx

   PW0  - XXXX XXXX XXXX XXXX  CONTROL BITS
   PW1  - XXXX XXXX XXXX XXXX  REPEAT COUNTER
   PW6  - XXXX XXXX XXXX XXXX  DISPLAY SELECT
   PW7  - XXXX XXXX XXXX XXXX  STARTING PASS COUNT
   PW8  - XXXX XXXX XXXX XXXX  PIT VALUE
   PW9  - XXXX XXXX XXXX XXXX  SIT VALUE
   PW12 - XXXX XXXX XXXX XXXX  MONITOR CONTROL
   PW13 - XXXX XXXX XXXX XXXX  SCOPE LOOP CONTROL
   PW16 - XXXX XXXX XXXX XXXX  TRIM CONTROL
   PW17 - XXXX XXXX XXXX XXXX  SELECTED PAD COUNT
   PW18 - XXXX XXXX XXXX XXXX  INSTR SELECT CONTROL
   PW19 - XXXX XXXX XXXX XXXX  PIT CONTROL
   PW20 - XXXX XXXX XXXX XXXX  SIT CONTROL
   PW21 - XXXX XXXX XXXX XXXX  USER MASK CONTROL
   PW22 - XXXX XXXX XXXX XXXX  INSTR LIST LENGTH
```

where:

CPU X	CPU identification (0 or 1)
xxxx PAGES	Number of pages in use if option selected
x.x	Current version of VEXC monitor
CC	One of the following: RU Running SP Stop for parameters SE Stop on error ST Stop at end of test pass RT Repeat test pass RC Repeat inst/operands/pad count
PC	Pass count (decimal)
PADS	Pad count (decimal)
EC	Error code (hex) 0001 X reg error 0002 A reg error 0004 Memory error 0008 UCR error 0010 Trim mode 0030 Lost error during trim
TE	Total number of errors (decimal)
PA	Address of parameter words (hex-byte address)
PV	PIT value running with
SV	SIT value running with

Comment Field = Keyboard input error message (see Instruction List Display). If running with small PIT/SIT value, line 3 displays the following information:

```
(Comment Field)  SL=xxxx  PL=xxxx  GZ=xxxx  PV=xxxxxxxx,  SV=xxxxxxxx
```

where:

SL Number of SIT losses
PL Number of PIT losses
GZ Number of PIT interrupts with PIT value greater than zero

Now the operator can enter CSME commands or RFST test commands. See Test Commands for RFST commands.

Running Display

```
RANDOM FAST/SLOW TEST     CPU X xxxx PAGES x.x  mm/dd/yy
RFST  RU  PC=xxxxxxxx  EC=xxxx   TE= xxxx PADS=xx    PA=xxxxx
(   Comment Line             )    PV= xxxxxxxx    SV=xxxxxxxx
```

Initial A/X Register Display

```
              A0-AF        INITIAL A AND X REGISTERS       X0-XF

Reg-0   XXXXXXXXXXXX                              XXXXXXXXXXXXXXXX
    1   XXXXXXXXXXXX                              XXXXXXXXXXXXXXXX
    .       .                                         .
    .       .                                         .
    .       .                                         .
Reg-F   XXXXXXXXXXXX                              XXXXXXXXXXXXXXXX
```

Instruction List Display

```
RANDOM FAST/SLOW TEST     CPU X xxxx PAGES x.x  mm/dd/yy
RFST SE  PC=xxxxxxxx  EC=xxxx  TE=xxxx  PADS=xx      PA=xxxxx
                                        PV=xxxxxxxx   SV=xxxxxxxx

72000    0BBB        CPYAX       XB,AB
72002    17B0        LPAGE       X0,XB,X1
72004    0DF0        CPYXX       X0,XF
72006    3940        ENTX        X1,40
72008    83AE  057F  SX          XE,AA,057F
7200C    A3A7  0752  SXI         X7,AA,X0,752
72010    71CC        SUBN,AC,X0  AC,X1  04060000  0D060006
7201A    9CEE  0004  BRINC       XE,XE,0004
7201E    DFA6  0476  SBYTS,8     X6,AA,X0,476
72022    3D05        ENTP        X5,0
72024    8D01  625B  ENTE        X1,625B
72028    55A8  5800  GTHV        A8,XA,X5,800
7202C    3D70        ENTP        X0,7
7202E    88AB  0309  LBIT        XB,AA,0309,X0
72032    1997        XORX        X7,X9
72034    D2AF  0563  LBYTS,3     XF,AA,X0,563
ENTER (F)IRST, (L)AST, (N)EXT, (P)REVIOUS, /RMA, OR (E)XIT
```

The above display converts the instruction list to a format equivalent to the assembly listing output. In this example, the RMA 72000 is used for descrip- tive purposes. The RMA of each instruction is displayed in column 1. The next two columns display the machine code for each instruction. Instructions are one or two parcels in length. The mnemonic is displayed next, and the address fields are calculated in the last column. Each screen can display 16 instructions.

The operator may enter:

F – for the first 16 instructions in the list
L – for the last 16 instructions in the list
N – to page forward
P – to page backward

Any RMA between the first address displayed on the first screen of instructions and the last RMA displayed on the last screen of instructions.

Example:

If this instruction list started at 72000 and ended at 72814, entering /72560 would display 16 instructions starting at RMA 72560. RMAs entered should be 0 mod 2 (parcel aligned).

E – to exit the display

Error Displays

1. If an error occurs when stop-on-error is set, RFST will display as shown below. The addresses shown are RMA byte addresses.

```
RANDOM FAST/SLOW TEST     CPU X xxxx PAGES x.x  mm/dd/yy
RFST SE  PC=xxxxxxxx  EC=xxxx  TE=xxxx  PADS=xx      PA=xxxxx
                                        PV=xxxxxxxx   SV=xxxxxxxx

     ADDRESSES OF BUFFERS -

        INSTRUCTION STREAM  =  XXXXXX
        X REGISTER RESULTS  =  XXXXXX
        A REGISTER RESULTS  =  XXXXXX
        MEMORY RESULTS (1)  =  XXXXXX
        MEMORY RESULTS (2)  =  XXXXXX
        INITIAL X REGISTERS =  XXXXXX
        INITIAL A REGISTERS =  XXXXXX
        INITIAL MEMORY      =  XXXXXX
        INITIAL BDP MEMORY  =  XXXXXX
        UCR RESULTS         =  XXXXXX
```

2. Register Error

```
                        NO PADS          PADS          DIFFERENCE

TYPE 0                  X------X         X------X      X------X
TYPE 1                  X------X         X------X      X------X
     .                     .                .             .
     .                     .                .             .
     .                     .                .             .
TYPE F                  X------X         X------X      X------X
```

where:

TYPE = REG-A for A register error
REG-X for X register error

3. Memory Error

```
STARTING BUFFER ADRS = NO PADS=XXXXX    PADS=XXXXX

   ADDRS          NO PADS          PADS          DIFFERENCE

   XXXXX          X------X         X------X      X------X
   XXXXX          X------X         X------X      X------X
   XXXXX          X------X         X------X      X------X
     .              .                .             .
     .              .                .             .
     .              .                .             .
   XXXXX          X------X         X------X      X------X
```

The display above shows the first 15 memory errors detected at the address displayed. The address displayed for each error is the byte offset relative to the starting buffer addresses. The NO PADS and PADS starting buffer addresses are RMA byte addresses.

RFST displays the following error messages in the comment field in line 2 when a command is incorrectly entered. Enter a valid command to clear the error message.

Message	Explanation
INVALID ENTRY	RFST does not recognize the command or the command parameters are incorrect. Refer to Test Commands for acceptable ranges.
INST LIST FULL	The instructions added using the ADD command exceed the instruction buffer capacity.
CANNOT TEST XX	RFST cannot execute the instruction opcode XX.

These opcodes are not tested:

OF - CPYXS
00 - HALT
03 - INTERRUPT
C0-C7 - EXECUTE ALGORITHM
02 - EXCHANGE (Used but not tested)
04 - RETURN - Return is tested in conjunction with a CALL instruction.
01 - SYNC - Sync op is used at the end of the instruction list for scoping purposes.
06 - POP - not tested during on-line execution only.

When running on-line, RFST runs as an unprivileged job. Therefore, opcode 17 (LPAGE) is tested, and in opcode 05 (PURGE), only K values of 3 through 7 are used. When running off-line, RFST tests opcode 17 and all K values for opcode 05. In both off-line and on-line testing, opcode 9F (BRCR), is only tested with K values of 2, 3, 5, 6, 7, A, B, D, E, and F.

Operator Entries

Test Commands

Refer to Parameters for bit descriptions and default values of associated parameter words.

Command	Response
S	Stop test execution.
R	Restart RFST from initial VEXC entry.
(Space bar)	Continue test execution.
C	Continue test execution.
P,xxxxxxxx	Skips forward to pass count xxxxxxxx prior to test execution. Default = 1. Input in decimal.
PAD,xx	Sets pad count with xx number of pads. Pads then run only xx pad count. Values range from O through 16_{10}. Default value = O.
	NOTE Entering pad,0 sets the test back to a default mode of running 0 through 16 pads each pass.
L,xxx	Set instruction list to length of X instructions. Values 1 through 500_{10} are acceptable. Default = 500_{10}.
TRIM	Instruction list will be trimmed (by removing instructions) upon detecting an error. Internal trim loop count = 1. Default = true.
TRIMR	Instruction list will be trimmed (by replacing opcodes with $1ADD_{16}$ and $1ACC_{16}$ codes) upon detecting an error. Internal trim loop count = 1. Default = false.
TRIMB	Both TRIM and TRIMR will be executed upon detecting an error. TRIM will execute first to shorten the list and TRIMR will then attempt to replace unnecessary instructions. Default = false.
TRIMI	Intermittent trim will be in effect based on the type of trim mode selected (TRIM,TRIMR or TRIMB). Internal trim loop count = 256. This is helpful for isolating intermittent errors.
NTRIM	No trimming of instruction stream upon error detection.

Command	Response
DS,x	Select display X. Display selections are 0 through 8. Default = 0. 0 = Running message 1 = Buffer addresses display (displays if error) 2 = A register result display 3 = X register result display 4 = Memory error display 5 = Initial A/X registers 6 = Generalized memory display (On-Line use only) 7 = Initial test display (Parameter words) 8 = Instruction list display (including pads) 9 = Instruction list display (original list before pads are inserted)
PITON	Enables running with small value in process interval timer (PIT). Default = on.
PITOF	Disables running with small value in PIT. PIT value is set to $FFFFFFFF_{16}$ for off-line, $7FFFFFFF_{16}$ for on-line and PW8 is ignored.
PV,X	Enter selected value into PIT. Default small PIT value = 250 microseconds (decimal). Input in decimal.
SITON	Enables running with small value in system interval timer (SIT). Default = on. Not valid on-line.
SITOF	Disables running with small value in SIT. SIT value is set to $FFFFFFFF_{16}$ for off-line and PW9 is ignored. Not valid on-line.
SV,X	Enter selected value into SIT. Default small SIT value = 200 microseconds (decimal). Input in decimal.
LOOP1	Sets scope mode and continuously executes present instruction/operand set without pads.
LOOP2	Sets scope mode and continuously executes present instruction/operand set with pads.
NORM	Tests all instructions. Default = true.
OMIT	Omits instruction listed in selected list from test list. Default = false. PW23 is selected list.
SEL	Tests instructions listed in selected list only. Default = false.
ADD,x,y,...	Adds opcodes x,y,... to selected list. Opcodes 00-FF are accepted.
DEL,x,y,...	Deletes opcodes x,y,... from selected list. Opcodes 00-FF are accepted. An asterisk (*) deletes all opcodes.
ADDV	Adds all vector opcodes to selected list.
ADDB	Adds all BDP opcodes to selected list.
ADDS	Adds all system instruction opcodes to selected list.

Command	Response
ADDF	Adds all single precision floating point opcodes to the selected list.
ADDD	Adds all double precision floating point opcodes to the selected list.
ADDL	Adds all load and store opcodes to the selected list.
ADDG	Adds all general instruction opcodes to the selected list.
ADDJ	Adds all branch instruction opcodes to the selected list.
DELV	Deletes all vector opcodes from the selected list.
DELB	Deletes all BDP opcodes from the selected list.
DELS	Deletes all system instruction opcodes from selected list.
DELD	Deletes all double precision floating point opcodes from the selected list.
DELF	Deletes all single precision floating point opcodes from the selected list.
DELL	Deletes all load and store opcodes from selected list.
DELG	Deletes all general instruction opcodes from selected list.
DELJ	Deletes all branch instruction opcodes from selected list.
MSK	Runs with user mask set (FF7F). Default = set.
NMSK	Run with user mask clear (FF00).
SSE/CSE	Set/Clear stop on error parameter bit.
SRT/CRT	Set/Clear repeat test pass parameter bit.
SST/CST	Set/Clear stop at end of test pass parameter bit.
SLE/CLE	Set/Clear log errors parameter bit.
SRC/CRC	Set/Clear repeat condition parameter bit.
SSC/CSC	Set/Clear stop at end of conditions.

Using Test Commands

Change Test Parameters

Test parameters can be changed prior to starting RFST execution or at any time RFST is running or has stopped. Use RFST commands or CMSE EC, and/or EB an imperative commands for off-line.

NOTE

User mask can only be changed by using MSK and NMSK commands at the beginning of a test, end of a test pass, or test restart. This avoids mis-compare errors due to trap/no trap differences. When running with user mask clear, BDP invalid data results are undefined and, on some machines, not repeatable. On these machines, when NMSK is selected, all BDP opcodes should be disabled using the following sequence:

ADDB	Adds BDP opcodes to selected list
OMIT	Omits opcodes in selected list from test list

Bit 49 of PWO is set by default, which allows initial parameter stop. If cleared by CMSE EC or EB command, RFST does not stop for initial parameter changes. The test uses default values for all available parameter settings. Parameter settings at RFST's initial start are saved and reinstated when RFST is restarted by the R command.

Stop on Error

Type SSE to set the stop-on-error parameter bit. This causes RFST to stop when an error is detected. Type CSE to clear the bit; RFST does not stop on error, but increments the error count if errors are detected. Press the space bar to resume RFST execution whenever it stops.

Repeat Failing Set of Instruction/Operands

When RFST stops on an error, the SRC command sets the repeat-condition parameter bit. Press the space bar to cause RFST to execute the same set of operands and instructions with and without pad instructions. The number of pad instructions used are the same as used by the failing set. Type CRC to clear the repeat-condition bit.

In order to return to a failing situation after test initialization, enter failing pass count with P,xxx command, enter failing pad count with PAD,xx command and enter SRC. RFST generates random numbers to selected pass count, begins execution with selected pad count, and repeats execution until CRC is entered. The error count is incremented during repeat condition execution.

Scope Loop a Failing Set of Instructions/Operands

After RFST has stopped on an error, type LOOP1 and press the space bar to loop the instruction set withouts pads. Type LOOP2 and press the space bar to execute the instruction set with pads.

RFST does not communicate with the operator during scope mode. For scope triggering, execute a sync 0100 instruction prior to the execution of the instruction stream.

To stop scope loop, clear PW13 with the CMSE instruction EC. The address of this word is displayed in the scope loop message.

Stop After Executing One Set of Random Instructions/Operands/Pad Count

Type SSC to set the stop-at-end-of-condition parameter bit. RFST stops after execution of the instruction set and the current pad count. Press the space bar to execute the next pad count and stop. These are the same instructions but with an additional pad added. Type CSC to clear the parameter bit. RFST then executes successive pad counts continuously.

Stop After Executing One Set of Random Instructions/Operands

Type SST to set stop-at-end-of-test parameter bit. RFST stops after one test pass which includes execution of the instruction stream with no pads, then with one pad per instruction, then two pads...through 16 pads per instruction. Press the space bar to execute next pass with new random instructions and operands. Type CST to clear the parameter bit. RFST then executes continuously.

Repeat Test Pass

Type SRT to set repeat-test parameter bit. When this bit is set, RFST repeats the test pass without updating instruction or operands. Pad counts of 0 through 16 are repeated. Type CRT to clear the parameter bit. RFST continues to next pass. Note that the error count increments during repeat test pass executions.

Stop Test While Executing

Type S to stop RFST execution. Press the space bar to resume execution.

Display Error Information

Type in DS,x to direct RFST to print a selected display. Refer to Test Commands for a description of the x parameter.

Trim Control

Default mode (TRIM) re-executes the failing instruction list once after trim- ming each instruction. If the test reports that the error was lost during TRIM, then the TRIMI command can set the re-execution loop to a count of 256. This way the failing condition is repeated, and the test re-executes the instruction stream 256 times before it reports that the error was lost. This process is helpful for intermittent errors. In addition, TRIMR or TRIMB can be selected. Using TRIMB will enable both modes of trimming. The instruction list will be shortened as much as possible (TRIM) and unnecessary instructions will be replaced by $1ADD_{16}$ and $1ACC_{16}$ instructions (TRIMR). TRIMI can be selected in conjunction with TRIM, TRIMR or TRIMB.

Normal Messages

See Running Display and A/X Register.

Error Messages

See Error Displays.

Recovery

An unidentified interrupt causes RFST to exchange to monitor (VEXC) and display a message describing the error. The operator can either scope loop this interrupt with a loop offered at the time, or re-start the test by declining the scope loop message. RFST can also be restarted by rerunning the command buffer.

The system will not abort RFST.

Applications

Scope Mode

Scope mode instructions are in Using Test Commands.

Tracking Exchanges and Traps

VEXC monitor keeps a number of information buffers that are useful in tracking exchanges and traps. The addresses and format of these buffers are displayed by VEXC when not running in a default mode with bit 36 of PWO clear. This display can be viewed by clearing bit 36 with a CMSE EC or EB command before the test begins or by clearing it in the command buffer and restarting the test using the command buffer. These tables begin at byte address 25000 hexadecimal for CPU0 and 26000 for CPU1.

Debug Test - DBUG 9

Introduction 9-1

Requirements 9-1
Hardware 9-1
Software 9-1
Accessories 9-2

Characteristics 9-2

Operational Procedure 9-3
Restrictions and User Cautions 9-3
Loading Procedure 9-3
Running Procedure 9-3
Parameters and Control Words 9-4
Parameters 9-4
Other Locations Of Interest 9-5
Installation Parameters 9-5
Control Words 9-6
Section Index 9-6

Operator Communication 9-7
Displays 9-7
Initial Display 9-7
Running Display 9-8
DBUG Help Display 9-8
Initial A/X Register Display 9-9
Instruction List Display 9-9
Error Displays 9-10
Operator Entries 9-13
Test Commands 9-13
Using Test Commands 9-16
Change Test Parameters 9-16
Stop on Error 9-16
Repeat Failing Set of Instruction/Operands 9-16
Scope Loop a Failing Set of Instructions/Operands 9-17
Stop After Executing One Set of Random Instructions/Operands/Debug Code 9-17
Stop After Executing One Set of Random Instructions/Operands 9-17
Repeat Test Pass 9-17
Stop Test While Executing 9-17
Display Error Information 9-17
Normal Messages 9-18
Error Messages 9-18
Recovery 9-18
Applications 9-18
Scope Mode 9-18
Tracking Exchanges and Traps 9-18

Debug Test - DBUG 9

Introduction

The Debug Test (DBUG) tests the debug feature of the CYBER 180 systems. DBUG is a model independent test designed to run on all 180 mainframes, and to detect errors at the virtual instruction level. It runs off-line under the Common Maintenance System Executive (CMSE) and on-line under the Diagnostic Virtual System (DVS).

DBUG tests the debug feature of the 180 instruction set using a random set of instructions and random data buffers. The random instructions run with debug disabled and again with debug enabled. Results from each run are compared for differences. Also, results are checked to ensure that debug occurred when expected, based on the debug mask and debug code.

Requirements

Hardware

This test is intended for CDC® CYBER 170/180 Models 810, 815, 825, 830, 835, 840, 845, 850, 855, 860, and 990 computer systems.

In addition to CMSE requirements, Random Fast/Slow Test (RFST) requires the following hardware to execute:

Target Configuration:

- 1 C180 CPU with 8 MB of memory
- 1 Disk unit supported by CMSE for test residence
- 1 System console
- Cache/page maps/segment maps
- 20 PPs

Minimum Configuration:

- 1 C180 CPU with 1 MB of memory
- 1 Disk unit supported by CMSE for test residence
- 1 System console
- 5 PPs

Software

DBUG runs off-line under the Common Maintenance Software Executive (CMSE) and on-line under the Diagnostic Virtual System (DVS).

Accessories

None required.

Characteristics

Description	Characteristics
Program name	DBUG
Test/Diag/Utility	Test
Size of source lines including comments	7233
Size of memory required for execution	1MB
Size of total assembled binary	145101
Level of isolation	Detect Only
Run time	N/A[1]
Off-line system	CMSE - 180
On-line system	DVS - NOS/VE
Test control resides in:	CPU
Source language	META
Source code maintenance	SCU

1. DBUG runs until halted by the operator with a target of 5 passes per minute.

Operational Procedure

Restrictions and User Cautions

1. All instructions must be independently operational for DBUG to test instruction streaming. The following tests must run: FCT1, FCT2, FCT3, FCT5, RCT1, RCT2, and RFST.

2. The virtual addressing mechanism must be operational. DBUG cannot run in real memory address (RMA) mode.

3. DBUG does not test any A170 instructions.

Some instructions, and therefore some hardware, are not tested by DBUG. The RETURN instruction is only used in conjunction with a CALL instruction. The EXCHANGE instruction is not tested but is used to go from job to monitor when required. The SYNC instruction is used at the end of the instruction stream for scoping purposes. INTERRUPT, CPYXS, EXECUTE, and HALT are not tested.

See Instruction List Display for on-line/off-line restrictions.

Loading Procedure

DBUG is assembled using the C180 assembler. The binary output is linked to the binary of virtual level executive (VEXC) using the C180 LINKER/LOADER to form the complete executable module. It can then be written to disk using tape-to-disk utility (TDX). To run off-line, the executable binary is loaded and executed using the CMSE command buffer capability. Consult the MSL15X reference manual for a description of the command buffer structure. To run on-line, consult the Diagnostic Virtual System (DVS On-Line) Usage manual for loading procedures.

Running Procedure

During normal execution, the test runs with default environment set-up values. If bit 36 of PW0 (parameter word 0) is clear, the operator can select values. In this case, VEXC displays the first messages and deals with test environment selection (for example, RUN WITH PAGE FAULTS?, RUN WITH LIMITED WORKING SET SIZE?, etc). Operator input is in the form of (Y)es/(N)o answers. If bit 36 of PWO is set, VEXC uses the information in PW12 to set up the test environment.

The first display from DBUG is a parameter stop display. See Test Commands for a list of available commands. Press the space bar to start random instruction testing.

Parameters and Control Words

Parameters

Tag	RMA Byte Address (Hex)	Bits	Description	Default Setting
	CPU0/CPU1			
PW0	A6100/A7100	0-63		00000000 08014010_{16}
		0-35	Not used	0
		36	Monitor-default run (CMSE EC command)	1
	(1)	37	Monitor bypass	0
	(1)	38	Test mode one	0
	(1)	39	Test mode two	0
		40-44	Not used	
	(2)	45	Batch mode	0
	(2)	46	On-line control	0
	(2)	47	Off-line control	1
		48	Operator stop	0
		49	Initial stop for parameters	1
		50-53	Not used	0
		54	Repeat present instruction/operand/debug code (SRC/CRC)	0
		55-56	Not used	
		57	Repeat present instruction/operand set with all debug codes (SRT/CRT)	0
		58	Log errors (SLE/CLE)	0
		59	Stop on error (SSE/CLE)	1
		60	Stop end of condition	0
		61-62	Not used	0
		63	Stop at end of test (SST/CST)	0
			NOTE (1) Used in test development only. (2) Used by DVS – not for off-line use.	
PW1	A6108/A7108	0-63	Repeat counter	1000_{10}
PW2-PW5			Not used	
PW6	A6130/A7130	0-63	Display select control (set by DS,x command)	0
PW7	A6138/A7138	0-63	Skip pass count (set by P,x command)	1
PW8	A6140/A7140	0-63	PIT value (set by PV,X command)	250_{10}

Tag	RMA Byte Address (Hex)	Bits	Description	Default Setting
PW9	A6148/A7148	0-63	SIT value (set by SV,X command)	200_{10}
PW10-PW11			Not used	
PW12	A6160/A7160	0-63	Monitor control bits. See 3.3.2 for bit definitions	$F00000040_{16}$
PW13	A6168/A7168	0-63	Scope loop control (set by LOOP1/LOOP2 commands)	0
PW14-PW15			Not used	
PW17	A6188/A7188	0-63	Selected debug code (set by DCODE,xx command)	0
PW18	A6190/A7190	0-63	Instr selection control (set by NORM/OMIT/SEL - default = NORM)	0
PW19	A6198/A7198	0-6	PIT control (set by PITON/PITOF command)	1
PW20	A61A0/A71A0	0-63	SIT control (set by SITON/SITOF command)	1
PW21	A61A8/A71A8	0-63	User mask control	1
PW22	A61B0/A71B0	0-63	Instr list length	500_{10}
PW23	A61B8/A71B8	0-63	Optional instruction list (16 words)	0
PW39	A6238/A7238	0-63	Debug list (64 words - 32 2-word entries)	(see DBUG Help Display)

Other Locations Of Interest

Obtain additonal information by displaying CM contents using the CMSE AB and AH commands. Error displays indicate addresses for instruction lists and input/output memory buffers.

Installation Parameters

The Parameter and Control word segment is a segment shared by DBUG (job) and VEXC (monitor). VEXC uses PW12 to set up the running environment.

PW12 is set to $F00000023_{16}$ during compile of DBUG. PW0, bit 36 interacts with PW12. The default is PW0, bit 36 set. When set, VEXC looks at PW12 for environment control bits. These can be changed in the command buffer using the CMSE EC or EB command.

Bit definitions for PW12 are as follows:

Bit	Meaning	Default
28	Run with repetitive page faults	1
29	Run with cache purges	1
30	Run with map purges	1
31	Run with page faults	1
32-63	Number of pages in working set	23_{16}

Control Words

The starting RMA for the parameter word block is given in the initial parameter stop display and in the running message. Control words reside above parameter words.

Tag	RMA Byte Address (Hex)	Description	Default Setting
	CPU0/CPU1		
CW0	A6000/A7000	Program name	ASCII 'RFST'
CW1	A6008/A7008	System type (Not used)	0
CW2	A6010/A7010	Error code	0
CW3	A6018/A7018	Error count	0
CW4	A6020/A7020	Pass count	0
CW5-7		Not used	
CW8	A6040/A7040	Pad count	0
CW9-31		Not used	

Section Index

DBUG is not organized into sections or subsections but rather on a test pass structure. Each test pass consists of the following:

1. Create random instruction stream up to 500 instructions long and create simulated debug buffer based on debug code and debug mask.
2. Create random A/X register values.
3. Create random input/output memory buffers.
4. Disable debug.
5. Execute instructions and save all results (regs/mem/ucr).
6. Enable debug.
7. Restore input values to registers and memory.
8. Execute instructions with debug enabled and save results.
9. Compare results from step 8 with results from step 5.
10. Do steps 6 through 9 using each debug code (READ, WRITE, INSTRUCTION FETCH, BRANCH, CALL, COMBINATION OF ALL CODES).

Each debug code is considered a condition so you may stop the test at the end of each debug code pass (stop end of condition) if you wish.

Operator Communication

Displays

Initial Display

```
DEBUG TEST                  CPU X xxxx PAGES x.x  mm/dd/yy
DBUG   CC   PC=xxxxxxxx   EC=xxxx   TE=xxxx   DC=xx   PA=xxxxx
(COMMENT FIELD)                     PV=xxxxxxxx, SV=xxxxxxxx

   PW0  - XXXX XXXX XXXX XXXX  CONTROL BITS
   PW1  - XXXX XXXX XXXX XXXX  REPEAT COUNTER
   PW6  - XXXX XXXX XXXX XXXX  DISPLAY SELECT
   PW7  - XXXX XXXX XXXX XXXX  STARTING PASS COUNT
   PW8  - XXXX XXXX XXXX XXXX  PIT VALUE
   PW9  - XXXX XXXX XXXX XXXX  SIT VALUE
   PW12 - XXXX XXXX XXXX XXXX  MONITOR CONTROL
   PW13 - XXXX XXXX XXXX XXXX  SCOPE LOOP CONTROL
   PW17 - XXXX XXXX XXXX XXXX  SELECTED DEBUG CODE
   PW18 - XXXX XXXX XXXX XXXX  INSTR SELECT CONTROL
   PW19 - XXXX XXXX XXXX XXXX  PIT CONTROL
   PW20 - XXXX XXXX XXXX XXXX  SIT CONTROL
   PW21 - XXXX XXXX XXXX XXXX  USER MASK CONTROL
   PW22 - XXXX XXXX XXXX XXXX  INSTR LIST LENGTH
```

where:

CPU X	CPU identification (0 or 1)
xxxx PAGES	Number of pages in use if option selected
x.x	Current version of VEXC monitor
CC	One of the following: RU Running SP Stop for parameters SE Stop on error ST Stop at end of test pass RT Repeat test pass RC Repeat inst/operands/pad count
PC	Pass count (DECIMAL)
EC	Error code (HEX) 0001 X reg error 0002 A reg error 0004 Memory error 0008 UCR error 0010 Debug error
TE	Total number of errors (DECIMAL)
DC	Debug code (HEX)
PA	Address of parameter words (HEX-BYTE ADDRESS)
PV	PIT value running with
SV	SIT value running with

Comment Field = Keyboard input error message (see Instruction List Display). If running with small PIT/SIT value, line 3 will display the following information:

```
(Comment Field)  SL=xxxx  PL=xxxx  GZ=xxxx  PV=xxxxxxxx,  SV=xxxxxxxx
```

where

SL Numer of SIT losses
PL Number of PIT losses
GZ Number of PIT interrupts with PIT value greater than zero

Now the operator can enter CSME commands or DBUG test commands. See DBUG Help Display for DBUG commands.

Running Display

```
DEBUG TEST                CPU X xxxx PAGES x.x  mm/dd/yy
DBUG  RU  PC=xxxxxxxx  EC=xxxx   TE= xxxx   DC=xx    PA=xxxxx
(   Comment Line           )    PV= xxxxxxxx    SV=xxxxxxxx
```

DBUG Help Display

```
DEBUG LIST LOCATED AT RMA XXXXXX (PW39)
   LIST ENTRY FORMAT = CCXXXSSSLLLLLLLL
                       XXXXXXXXHHHHHHHH

      WHERE  C = DEBUG CODE
             S = SEGMENT
             L = LOW BYTE ADDR            DEBUG CODE FORMAT
             H = HIGH BYTE ADDR              80 = READ
             X = IGNORED                     40 = WRITE
                                             20 = FETCH
                                             10 = BRANCH
                                             08 = CALL
                                             04 = END OF LIST

   IF CHANGING DEBUG LIST ENTRIES, BE SURE TO SET END OF LIST BIT IN
   LAST ENTRY DESIRED.
```

PW39 is the start of the debug list. There are 64 words in the list (32 two-word entries). The list is preset by the test, but the operator can change the entries by using the CMSE EB or EC commands. Do not use the actual test set-up code segment for debug entries because no simulated debug buffer will be created for that segment. This display is only available off-line.

Initial A/X Register Display

```
            A0-AF        INITIAL A AND X REGISTERS        X0-XF

Reg-0   XXXXXXXXXXXX                                 XXXXXXXXXXXXXXXX
    1   XXXXXXXXXXXX                                 XXXXXXXXXXXXXXXX
    .       .                                            .
    .       .                                            .
    .       .                                            .
Reg-F   XXXXXXXXXXXX                                 XXXXXXXXXXXXXXXX
```

Instruction List Display

```
DEBUG TEST                CPU X xxxx PAGES x.x  mm/dd/yy
DBUG SE  PC=xxxxxxxx  EC=xxxx  TE=xxxx    DC=xx     PA=xxxxx
                                        PV=xxxxxxxx   SV=xxxxxxxx

6B7A2    8BDC  35EF  ADDXQ       XC,XD,35EF
6B7A6    3D8C        ENTP        XC,8
6B7A8    71CC        SUBN,AC,X0  AC,X1  061F0614  0C040633
6B7B2    1002        INCX        X2,0
6B7B4    D5A9  00C8  LBYTS,6     X9,AA,X0,0C8
6B7B8    4AAB  A997  ANDV        AB,XA,AA,997
6B7BC    A9F2  D6F7  SHFX        X2,XF,XD,6F7
6B7C0    DDAE  0153  SBYTS,6     XE,AA,X0,153
6B7C4    D6AD  0734  LBYTS,7     XD,AA,X0,734
6B7C8    9544  0003  BRXNE       X4,X4,0003
6B7CC    26DA        MULX        XA,XD
6B7CE    3AED        CNIF        XD,XE
6B7D0    9322  0007  BRRGE       X2,X2,0007
6B7D4    71CC        SUBN,AC,X0  AC,X1  0C110637  03120648
6B7DE    70CC        ADDN,AC,X0  AC,X1  0609065A  0D130663
6B7E8    2F40        BRDIR       A4,X0
ENTER (F)IRST, (L)AST, (N)EXT, (P)REVIOUS, /RMA, OR (E)XIT
```

The above display converts the instruction list to a format equivalent to the assembly listing output. In this example, the RMA 6B7A2 is used for descriptive purposes. The RMA of each instruction is displayed in column one. The next two columns display the machine code for each instruction. Instructions are one or two parcels long. The mnemonic is displayed next, and the address fields are calculated in the last column. Each screen can display 16 instructions.

The operator may enter:

F – for the first 16 instructions in the list
L – for the last 16 instructions in the list
N – to page forward
P – to page backward

Any RMA between the first address displayed on the first screen of instructions and the last RMA displayed on the last screen of instructions.

If this instruction list started at 6B000 and ended at 6B7E8, entering /6B598 would display 16 instructions starting at RMA 6B598. RMAs entered should be 0 mod 2 (parcel aligned).

E – to exit the display

Error Displays

1. When an error is detected and stop-on-error is set, DBUG displays the following. Addresses are RMA byte addresses. See Running Display also.

```
DEBUG TEST                CPU X xxxx PAGES x.x  mm/dd/yy
DBUG SE  PC=xxxxxxxx  EC=xxxx  TE=xxxx  DC=xx     PA=xxxxx
                                        PV=xxxxxxxx   SV=xxxxxxxx

     ADDRESSES OF BUFFERS -

        INSTRUCTION STREAM  =  XXXXXX
        X REGISTER RESULTS  =  XXXXXX
        A REGISTER RESULTS  =  XXXXXX
        MEMORY RESULTS (1)  =  XXXXXX
        MEMORY RESULTS (2)  =  XXXXXX
        INITIAL X REGISTERS =  XXXXXX
        INITIAL A REGISTERS =  XXXXXX
        INITIAL MEMORY      =  XXXXXX
        INITIAL BDP MEMORY  =  XXXXXX
        UCR RESULTS         =  XXXXXX
        ACT DEBUG BUFFER    =  XXXXXX
        SIM DEBUG BUFFER    =  XXXXXX
```

2. Register error

```
                    NO DEBUG          DEBUG          DIFFERENCE

TYPE 0              X------X          X------X       X------X
TYPE 1              X------X          X------X       X------X
     .                 .                .              .
     .                 .                .              .
     .                 .                .              .
TYPE F              X------X          X------X       X------X
```

Where

TYPE = REG-A for A register error (display 2)

REG-X for X register error (display 3)

3. Memory error

```
STARTING BUFFER ADRS = NO DEBUG=XXXXX  DEBUG=XXXXX

    ADDRS          NO DEBUG          DEBUG          DIFFERENCE

    XXXXX          X------X          X------X       X------X
    XXXXX          X------X          X------X       X------X
      .                .                 .              .
      .                .                 .              .
      .                .                 .              .
    XXXXX          X------X          X------X       X------X
```

This display will print the first 15 memory errors detected at the address displayed. The address displayed for each error is the byte offset relative to the starting buffer addresses. The NO DEBUG and DEBUG starting buffer addresses are RMA byte addresses.

4. DEBUG error results

```
EXPECTED DEBUG TRAP ON AAAAAA              EXP BUFFER AT XXXXXXXXXXXX
DEBUG MASK = XX                         ACT BUFFER AT XXXXXXXXXXXX
DEBUG LIST ENTRY = XXXXXXXXXXXXXXXX
                   XXXXXXXXXXXXXXXX
INSTRUCTION AT P = IIII IIII IIII IIII

                 EXP                 ACT
   YYYYYY  X--------------X                YYYYYY  X--------------X
           X--------------X                        X--------------X
           X--------------X                        X--------------X

      MASK = 7F00FFFFFFFFFFFF
```

If an error is detected in the debug buffers, this display gives necessary information.

- AAAAAA indicates type of debug trap expected (READ, WRITE, FETCH, CALL or BRANCH).
- The addresses of the buffer are the starting address of each buffer (RMA byte if off-line, PVA if on-line).
- The debug list entry that applies to the expected debug trap is displayed.
- II--II displays 8 bytes of instructions starting at the P of the expected debug trap.
- The EXP and ACT displays are the entry preceding the miscompare, the entry that miscompares, and the next entry in the EXPECTED and ACTUAL debug buffers. The mask shows which bits are used in the compare.

5. When a DBUG command is entered incorrectly, DBUG displays the following messages in the comment field in Line 2. A valid command clears the error message.

Message	Explanation
INVALID ENTRY	DBUG does not recognize the command entered, or the command parameters are not acceptable. See DBUG Help Display for acceptable ranges.
INST LIST FULL	The instructions added using the ADD command exceed the instruction buffer.
CANNOT TEST XX	DBUG cannot execute the instruction opcode XX.

Opcodes not tested are:

OF - CPYXS
00 - HALT
03 - INTERRUPT
C0-C7- EXECUTE ALGORITHM
02 - EXCHANGE (Used but not tested)
04 - RETURN- Return is tested in conjunction with a CALL instruction.
01 - SYNC- Sync op is used at the end of the instruction list for scoping purposes.
06 - POP- not tested during on-line execution only.

When running on-line, DBUG runs as an unprivileged job. Therefore, opcode 17 (LPAGE) is not tested, and for opcode 05 (PURGE), only K values of 3 through 7 are used. When running off-line, DBUG tests opcode 17 and all K values for opcode 05. For both off and on-line testing, opcode 9F (BRCR) is only tested with K value of B.

Operator Entries

Test Commands

See Parameter Words for bit descriptions and default values of associated parameter words.

Command	Response
S	Stop test execution.
R	Restart DBUG from initial VEXC entry.
(Space bar)	Continue test execution.
C	Continue test execution (same as space bar).
P,xxxxxxxx	Skips forward to pass count xxxxxxxx prior to test execution. Default equals 1. Input in decimal.
DCODE,xx	Sets debug code to specified value. Input in hex. Test then runs only with specified debug code. 80 = Read 20 = Fetch 08 = Call 40 = Write 10 = Branch 04 = End of list Or any combination of the above.
L,xxx	Set instruction list to length of X instructions. Values 1-500_{10} are acceptable. Default = 500_{10}.
DS,x	Select display X. Display selections are 0 through 10. Default = 0. 0 = Running message 1 = Buffer addresses display (are displayed if error occurs) 2 = A register result display 3 = X register result display 4 = Memory error display 5 = Initial A/X registers 6 = Generalized memory display (on-Line use only) 7 = Initial test display (parameter words) 8 = Debug help message 9 = Debug error results 10 = Instruction list display
PITON	Enables running with small value in PIT (Process Interval Timer). Default equals ON.

Command	Response
PITOF	Disables running with small value in PIT (Process Interval Timer). PIT value will be set to $FFFFFFFF_{16}$ for off-line, $7FFFFFFF_{16}$ for on-line and PW8 ignored.
PV,X	Enter selected value into PIT. Default small PIT value equals 200 microseconds (decimal). Input in decimal.
SITON	Enables running with small value in SIT (System Interval Timer). Default equals on. Not valid on-line.
SITOF	Disables running with small value in SIT. SIT value will be set to $FFFFFFFF_{16}$ for off-line and PW9 ignored. Not valid on-line.
SV,X	Enter selected value into SIT. Default small SIT value equals 250 microseconds (decimal). Input in decimal.
LOOP1	Sets scope mode and continuously executes present instruction/operand set without debug.
LOOP2	Sets scope mode and continuously executes present instruction/operand set with debug enabled.
NORM	Test all instructions. Default equals true.
OMIT	Omit instruction listed in selected list from test list. Default equals false. PW23 is selected list.
SEL	Test only instructions listed in selected list. Default equals false.
ADD,x,y,...	Add opcodes x,y,... to selected list. Opcodes 00-FF are accepted.
DEL,x,y,...	Delete opcodes x,y,... from selected list. Opcodes 00-FF are accepted. An asterisk (*) deletes all opcodes.
ADDB	Add all BDP opcodes to selected list.
ADDV	Add all vector opcodes to selected list.
ADDS	Add all system instruction opcodes to selected list.
ADDF	Add all single precision floating point opcodes to the selected list.
ADDD	Add all double precision floating point opcodes to the selected list.
ADDL	Adds all load and store opcodes to the selected list.
ADDG	Adds all general instructions opcodes to the selected list.
ADDJ	Adds all branch instruction opcodes to the selected list.
DELV	Deletes all vector opcodes from the selected list.
DELB	Deletes all BDP opcodes from the selected list.

Command	Response
DELS	Deletes all system instruction opcodes from the selected list.
DELD	Deletes all double precision floating point opcodes from the selected list.
DELF	Deletes all single precision floating point opcodes from the selected list.
DELL	Deletes all load and store opcodes from selected list.
DELG	Deletes all general instruction opcodes from selected list.
DELJ	Deletes all branch instruction opcodes from the selected list.
MSK	Run with user mask set (FF7F). Default equals set.
NMSK	Run with user mask clear (FF00).
SSE/CSE	Set/Clear stop on error parameter bit.
SRT/CRT	Set/Clear repeat test pass parameter bit.
SST/CST	Set/Clear stop at end of test pass parameter bit.
SLE/CLE	Set/Clear log errors parameter bit.
SRC/CRC	Set/Clear repeat condition parameter bit.
SSC/CSC	Set/Clear stop at end of conditions.

Using Test Commands

Change Test Parameters

Test parameters can be changed before starting DBUG, while it runs, or after it stops. Use DBUG commands or CMSE EC and/or EB commands for off-line. Refer to DVS on-line usage manual for on-line parameter changes.

NOTE

User mask will only be changed by using MSK and NMSK commands at the beginning of a test, end of a test pass, or test restart. This avoids miscompare errors due to trap/no trap differences. When running with user mask clear, BDP invalid data results are undefined and, on some machines, not repeatable. Also, floating point branches cannot be calculated and must be disabled.

On these machines, when NMSK is selected, all BDP opcodes should be disabled. On all machines, single precision floating point opcodes must be disabled. The input sequence is:

ADDB	Add BDP opcodes to selected list
ADDF	Adds single precision floating point opcodes to selected list
OMIT	Omit opcodes in selected list from test list

Bit 49 of PWO is set by default which allows initial parameter stop. If cleared by CMSE EC or EB command, DBUG does not stop for initial parameter changes; the test uses default values for all available parameter settings.

The parameter settings at the time DBUG is initially started are saved and reinstated whenever DBUG is restarted by typing an R command.

Stop on Error

Typing SSE sets the stop-on-error parameter bit and causes DBUG to stop whenever an error is detected. Typing CSE clears the bit and DBUG does not stop on error, but increments the error count if errors are detected. Space bar resumes DBUG execution whenever it is stopped.

Repeat Failing Set of Instruction/Operands

When DBUG stops on error, entry of SRC command sets the repeat-condition parameter bit. Entry of a space causes DBUG to execute the same set of operands and instructions without and with debug enabled. The debug code used is the same code as the failing set used.

Type in CRC to clear repeat-condition bit.

To return to failing situation from test initialization, enter failing pass count with P,xxx command, enter failing debug code with DC,xx command, and enter SRC. The test generates random numbers to selected pass count, begin execution with selected debug code, and repeat execution until CRC is entered.

Note that the error count is incremented during repeat condition execution.

Scope Loop a Failing Set of Instructions/Operands

If a scope loop is desired after DBUG stops on error, type LOOP1 or LOOP2 and a space bar. LOOP1 executes without debug. LOOP2 executes with debug.

DBUG does not communicate with the operator during scope mode. For Scope triggering, a sync (0100) instruction is executed prior to the execution of the instruction stream.

To stop scope looping, clear PW13 with the CMSE instruction EC. The address of this word is displayed in the scope loop message.

Stop After Executing One Set of Random Instructions/Operands/Debug Code

SSC sets stop-at-end-of-condition parameter bit. DBUG stops after execution of the instruction set and the current debug code. Space bar executes next debug code (same instructions) and stop. CSC clears bit and DBUG executes successive debug codes continuously.

Stop After Executing One Set of Random Instructions/Operands

SST sets stop-at-end-of-test parameter bit. DBUG stops after one test pass (execution of the instruction stream with each debug code and then a combination of all codes). Space bar executes next pass (with new random instructions and operands) and stop. CST clears bit and DBUG executes continuously.

Repeat Test Pass

SRT sets repeat-test parameter bit. When this bit is set, DBUG repeats the test pass without updating instruction or operands. All debug codes are repeated. CRT clears bit and DBUG continues to next pass. Note that the error count is incremented during repeat test pass execution.

Stop Test While Executing

An S command stops DBUG execution. Space bar resumes execution.

Display Error Information

Type in DS,x to direct DBUG to print selected display. Refer to Test Commands for description of the x parameter.

Normal Messages

See Running Display and A/X Register Display.

Error Messages

See Error Displays.

Recovery

An unidentified interrupt causes DBUG to exchange to monitor (VEXC) and display a message which describes the error. You can either scope loop this interrupt via a loop offered at the time, or re-start the test by declining the scope loop message. DBUG can also be restarted by rerunning the command buffer.

Applications

Scope Mode

Scope mode instructions are in Using Test Commands.

Tracking Exchanges and Traps

VEXC monitor keeps a number of information buffers that are useful in tracking exchanges and traps. When not running in default mode (bit 36 of PWO clear), VEXC displays the addresses and format of these buffers. By clearing this bit with a CMSE EC or EB command before the test begins, or by clearing it and restarting the test with an R command, this display can be viewed. These tables begin at byte address 25000 hexadecimal for CPU0 and 26000 for CPU1.

Keypoint Test - KYPT 10

Introduction 10-1

Requirements 10-1
- Hardware 10-1
- Software 10-1
- Accessories 10-1
- Characteristics 10-2

Operational Procedure 10-3
- Restrictions and User Cautions 10-3
- Loading Procedure 10-3
  - Running Procedure 10-3
  - Aborts and Recovery 10-4
    - Off-Line 10-4
    - On-Line 10-4
  - Logical Data Structures 10-4
- Parameter and Control Words 10-5
  - Parameters 10-5
  - Control Words 10-6
- Section Index 10-6

Operator Communication 10-7
- Displays 10-7
  - Parameter Display 10-7
  - Running Display 10-8
  - Help Display 10-9
  - Explain Display 10-10
- Operator Entries 10-11
  - Use of KBPON, KBPOF Commands 10-12
- Normal Messages 10-12
- Error Messages 10-13
  - Error Messages for Sections 0 through 7 10-13
  - Error Messages for Sections 8 through 11 10-16
  - PIT/SIT Errors 10-17
- Applications 10-17

Keypoint Test - KYPT 10

Introduction

The CPU Keypoint Test (KYPT) verifies the correct operation of the keypoint instruction for the CYBER 180 systems. It is a model independent test designed to run on all 180 mainframes and to detect errors at the virtual instruction level. The test runs off-line under the Common Maintenance Software Executive (CMSE) and on-line under the Diagnostic Virtual System (DVS).

KYPT interfaces to CMSE via the Virtual Level Executive (VEXC) to request that CMSE display messages, log errors, and input operator request.

Requirements

Hardware

The test is intended for CDC CYBER 180 Models 810, 815, 825, 830, 835, 840, 845, 850, 855, 860, and 990.

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

- 1 Central processor
- 1 Megabyte of central memory
- 1 Magnetic tape or mass storage device
- 1 Display/keyboard device

Software

This test runs off-line under the control of CMSE and on-line under DVS. Off-line interfaces are handled by CMSE and VEXC, and by DVS for the on-line case.

Accessories

None

Characteristics

Description	Characteristics
Program name	KYPT
Test/Diag/Utility	Test
Size of source lines including comments	5250
Size of memory required for execution	1 MB
Size of total assembled binary in bytes	171044
Run time as measured on S3	35 Secs
Level of isolation	Detect Only
Off-line system	CMSE-180
On-line system	NOS/VE-DVS
Test control resides in	CPU
Source language	META
Source code maintenance	SCU

Operational Procedure

Restrictions and User Cautions

All instructions other than the keypoint instruction must be operational in order for KYPT to run and the virtual addressing mechanism must work. KYPT does not test any 170 mode instructions. The keypoint instruction is not tested in monitor mode. See Use of KBPON, KBPOF Commands for details on special cases of test execution.

Loading Procedure

KYPT is assembled using the virtual machine assembler. The binary output is linked to the assembled binary of virtual level executive interface (VEXC) to form the complete executable module. It can then be written to disk using the tape-to-disk utility (TDX) or the CYBER initialization package (CIP) instal- lation utility.

Load the microcode before executing the test.

Command buffer KYPTx (where x is 1 for model 810/815/825/830, 2 for 835, 3 for 840/845/850/855/860 and 5 for 990) exists on the MSL15X 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 chapter of the MSL15X 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,KYPTx 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 chapter of the MSL15X reference manual for procedures.

Consult the MSL15X reference manual for more detail on the command buffer structure. To run on-line, consult the DVS On-Line Usage manual for loading procedures. Only sections 0 through 7 execute on-line.

Running Procedure

During normal execution the test runs with a default environment determined by the values of bit 36 in parameter word 0 (PW00) and the contents of PW12. Default settings are bit 36 = 1 and PW12 = F00000005. See Parameters for the parameter word bit definitions. In this case, the first message is a KYPT parameter display.

If bit 36 of PW00 is clear, then you can determine what options to run with and select values for these options. In this case, the first messages displayed are done by VEXC and deal with environmental selection (RUN WITH PAGE FAULTS?, RUN WITH MAP PURGES?, RUN WITH CACHE PURGES?, RUN WITH REPETITIVE PAGE FAULTS?, RUN WITH LIMITED WORKING SET SIZE?, ENTER NUMBER OF PAGES). Answer the questions with a Y(yes) or N(no), or enter numerical values followed by a carriage return where appropriate.

To select options, clear bit 36 of PW00 with the CMSE EB command in the command buffer after the test loads, or clear the bit and restart the test with an R command.

Aborts and Recovery

Off-Line

An unidentified interrupt causes KYPT to exchange to the VEXC monitor and display a message which describes the error. A message is displayed giving the following values:

1. Monitor condition register (MCR)
2. User condition register (UCR)
3. Untranslatable pointer (UTP)
4. P Register (process virtual address) and (real memory address)

Scope loop this interrupt via a loop offered at the time, or restart the test by declining the scope loop. KYPT can also be restarted by rerunning the command buffer. The system does not abort KYPT.

On-Line

An unidentified interrupt causes KYPT to exchange to the DVS monitor. The error is logged in the job dayfile and the test is aborted. See the DVS On-Line Usage manual for more information.

A detected uncorrectable error in either KYPT or DVS causes the system to abort KYPT or DVS respectively.

Logical Data Structures

The VEXC monitor keeps information tables that are useful in tracking exchanges and trap interrupts. The addresses and formats of these tables are displayed by VEXC when not running in monitor default mode (bit 36 of PW00 clear). This display can be viewed by clearing this bit with a CMSE EB command in the command buffer after the test loads, or by clearing bit 36 and restarting the test with an R command. These tables begin at byte address 23000(16).

Parameter and Control Words

Parameters

The following table gives a description of the contents of each parameter word. It also provides the exact address (byte/word) and default value, in hex, for easier reference.

Name	Byte/Word Addr	Bits	Description	Default
PW00	66100/CC20		Test control bits	8000010
		0-35	Not used	
		36	Monitor default	1
		37	Monitor bypass	0
		38-45	Not used	0
		46	On-line control	0
		47-48	Not used	
		49	Stop for parameters	0
		50-54	Not used	
		55	Repeat subsection	0
		56	Repeat section	0
		57	Repeat test	0
		58	Log errors	0
		59	Stop on errors	1
		60	Not used	
		61	Stop on subsection	0
		62	Stop on section	0
		63	Stop on test	0
PW01	66108/CC21		Repeat counter Test executes number of passes contained in this counter	0
PW02	66110/CC22		Section select (63-00)	FFF
		52-63	Selects Sections 0-11	
		0-51	Not used	
PW08	66140/CC28		Small PIT value	64
PW09	66148/CC29		Small SIT value	96
PW12	66160/CC2C		Monitor control bits	F00000008
		0-27	Not used	
		28	Run with repetitive page faults	1
		29	Run with cache purges	1
		30	Run with map purges	1
		31	Run with page faults	1
		32-63	Number of pages in working set	8
PW16	66180/CC30		Init KBP value (word addr) (Sections 0-7 only)	E200
PW17	66188/CC31		Keypoint mask reg value	FFFF
PW18	66190/CC32		Enable user supplied KBP (nonzero = ON)	0
PW19	66198/CC33		PIT control (nonzero = ON) Enables running with a small value in the PIT register	1
PW20	661A0/CC34		SIT control (nonzero = ON) Enables running with a small value in the SIT register	1

Control Words

These words are status indicators only and should not be altered by the operator.

Name	Byte/Word Addr	Bits	Description	Default
CW0	66000/CC00		Program name	ASCII KYPT
CW1	66008/CC01		System type	0
CW3	66018/CC03		Error count	0
CW4	66020/CC04		Pass count	0
CW5	66028/CC05		Current section	0
CW6	66030/CC06		Current subsection	0

Section Index

Section	Brief Description
0	Tests the entry recording mechanism with the keypoint enable flag = 0
1	Tests the entry recording mechanism with the keypoint mask register = 0000
2	Tests the correct updating of the keypoint buffer pointer
3	Tests the entry recording mechanism with the keypoint mask register nonzero
4	Verifies that the keypoint free running counter value is unique
5	Verifies that the keypoint class and code fields are correct
6	Verifies the correct interpretation of register Xk in the formation of the keypoint code
7	Verifies that no arithmetic overflow is detected in the formation of the keypoint code
8	Verifies invalid segment interrupts
9	Verifies access violation interrupts
10	Verifies address specification interrupts
11	Verifies page fault interrupts

Operator Communication

Displays

Parameter Display

The parameter display gives the current parameter values and a brief description of all valid parameter words. It also specifies the parameter word address (PA) of parameter word 0 (PW00).

To see a parameter display at any time during test execution enter the command PARAM. The test stops execution and issues the parameter display. Make parameter changes only during initialization or after a test restart.

The parameter display has the following format:

```
                                                                    Line
KYPT  SET  PARAMS  PA=XXXXXX  VRSN = MN/DY/YR                         1
(Comment Field)                                                      2
                                                                     3
PW00  =  XXXX XXXX XXXX XXXX    TEST/DIAG CONTROL BITS               4
PW01  =  XXXX XXXX XXXX XXXX    REPEAT COUNTER                       5
PW02  =  XXXX XXXX XXXX XXXX    SECTION SELECT BITS (63-00)          6
PW08  =  XXXX XXXX XXXX XXXX    SMALL PIT VALUE (DFLT=100 DEC)       7
PW09  =  XXXX XXXX XXXX XXXX    SMALL SIT VALUE (DFLT=150 DEC)       8
PW12  =  XXXX XXXX XXXX XXXX    MONITOR CONTROL BITS                 9
PW16  =  XXXX XXXX XXXX XXXX    INIT KBP VALUE (WORD RMA)           10
PW17  =  XXXX XXXX XXXX XXXX    KEYPOINT MASK REG VALUE             11
PW18  =  XXXX XXXX XXXX XXXX    ENABLE USER SUPPLIED KBP (1=ON)     12
PW19  =  XXXX XXXX XXXX XXXX    PIT CONTROL (1=ON)                  13
PW20  =  XXXX XXXX XXXX XXXX    SIT CONTROL (1=ON)                  14
                                                                     .
                                                                     .
SPACE BAR TO CONTINUE                                               18
```

NOTE

The message INVALID ENTRY appears in the comment field when an invalid command is entered by the operator.

Running Display

The running display has the following format:

```
                                                                    Line

      KYPT  TS  PCXXXX  SXXXX  SBXXXX  TEXXXX                          2
      (Comment Field)  SLXXXX PLXXXX GZXXXX  SV=XXXXXXXX PV=XXXXXXXX   3
```

where:

TS	=	HT	Stopped by operator
		RU	Running
		RS	Repeating section
		RB	Repeating subsection
		SS	Stopped at end of section
		SB	Stopped at end of subsection
		ST	Stopped at end of test
		SE	Stopped on error
		PC	Current pass count (decimal)
		S	Current section (decimal)
		SB	Current subsection (decimal)
		TE	Total number of errors (decimal)
		SL	Number of SIT losses (decimal)
		PL	Number of PIT losses (decimal)
		GZ	Number of PIT interrupts with PIT value greater than zero (decimal)
		PV	Current PIT value (decimal)
		SV	Current SIT value (decimal)

NOTE

The message INVALID ENTRY appears in the comment field when the command entered by the operator is invalid. The message PV=ALL ONES appears if the test is running with a large process interval timer (PIT) value (PIT=FFFFFFFF). The message SV=ALL ONES appears if the test is running with a large system interval timer (SIT) value (SIT=FFFFFFFF).

Help Display

The HELP command stops test execution and produces the following display:

```
                                                                    Line

        LIST OF VALID TEST COMMANDS                                  1
(SPACE),C    BEGIN OR CONTINUE TEST EXECUTION                        2
                                                                     3
R            RESTART TEST FROM BEGINNING                             4
                                                                     5
S            STOP TEST EXECUTION                                     6
                                                                     7
EXPL/HELP    PRESENT EXPLAIN/HELP DISPLAYS                           8
                                                                     9
PARAM        PRESENT PARAMETER DISPLAY                              10
                                                                    11
KBPON/KBPOF  ENABLE/DISABLE USER TO SUPPLY KBP VALUE                12
                                                                    13
PITON/SITON  ENABLE  RUN WITH SMALL  PIT/SIT  VALUE                 14
                                                                    15
PITOF/SITOF  DISABLE RUN WITH SMALL  PIT/SIT  VALUE                 16
                                                                    17
PV(SV),XXXX  SET  PIT/SIT TO XXXX (8 DIGIT DECIMAL VALUE)           18
(Comment Field)                                                     19
```

NOTE

The message INVALID ENTRY appears in the comment field when the command entered by the operator is invalid.

Explain Display

The command EXPL stops test execution and produces the following display:

```
                                                                    Line

KEYPOINT ENTRIES ARE RECORDED IN CENTRAL MEMORY  VIA  PVA             1
SPECIFIED IN KEYPOINT BUFFER POINTER (KBP).  ENTRIES  ARE           2
ENABLED WHEN LOGICAL PRODUCT OF KEYPOINT ENABLE FLAG (KEF)          3
AND JTH BIT IN THE KEYPOINT MASK REGISTER (KMR) IS NONZERO.         4
                                                                    5
EXECUTION OF KEYPOINT  J,XK,Q  INSTRUCTION CAUSES ONE               6
WORD OF KEYPOINT DATA TO BE RECORDED AS FOLLOWS                     7
                                                                    8
  BITS  0-27  RIGHTMOST 28 BITS OF FRC                              9
        28-31  KEYPOINT CLASS NUMBER (=J)                          10
        32-63  KEYPOINT CODE (=XKR+Q)                              11
                                                                   12
AFTER WRITING DATA INTO CM, THE KBP IS INCREMENTED BY 8.           13
USER CAN SELECT STARTING RECORDING WORD ADDR DURING                14
INITIALIZATION OR RESTART BY ENTERING THE KBPON.                   15
COMMAND (OFF-LINE ONLY).                                           16
                                                                   17
(Comment Field)                                                    18
```

NOTE

The message INVALID ENTRY appears in the comment field when the command entered by the operator is invalid.

Operator Entries

The following commands can be used at any time during test execution. KYPT only processes the commands at the end of a section or subsection. If an invalid command is used at a stopped condition, then a diagnostic message is issued and the test waits for a valid command entry to continue. However, if an invalid command is used while the test is running, the test issues the diagnostic message but does not stop execution.

Command	Action Taken
S	Stops test execution.
R	Restarts test from initial monitor entry. Parameter words are not reinitialized.
(Space bar), C	Begins or continues test execution.
PARAM	Displays parameter settings.
HELP	Displays list of valid test commands.
EXPL	Displays keypoint instruction specification.
KBPON	Enables user to supply beginning word address for keypoint data recording by setting PW18. The operator is then prompted for input. (Off-line case only – see Use of KBPON, KBPOF Commands.) Default = off.
KBPOF	Negates KBPON command. Clears PW18.
PITON	Enables running with small value in PIT. Default = on.
PITOF	Disables running with small value in PIT (value set to FFFFFFFF(16)). Enter 8 digit decimal value into PIT.
PV,XXXXXXXX	Enter 8 digit decimal value into PIT. Default small PIT value = 100 microseconds (decimal).
SITON	Enables running with small value in SIT. Default = on.
SITOF	Disables running with small value in SIT (value set to FFFFFFFF(16) for off-line or 7FFFFFFF(16) for on-line).
SV,XXXXXXXX	Enter 8 digit decimal value into SIT. Default small PIT value = 150 microseconds (decimal).
SSE/CSE	Set/clear stop on error parameter bit.
SLE/CLE1	Set/clear log error parameter bit.
SST/CST	Set/clear stop on end of test parameter bit.
SRS/CRS	Set/clear repeat section parameter bit.
SRB/CRB	Set/clear repeat subsection parameter bit.
SRT/CRT	Set/clear repeat test pass parameter bit.

Use of KBPON, KBPOF Commands

The KBPON and KBPOF commands enable and disable respectively user entry of central memory addresses where keypoint data recording and testing is to begin (off-line case only). When a user is running on-line or when the text is executing simultaneously on both CPUs in a dual-CPU system, the user cannot supply a testing address.

NOTE

The KBPON command is effective only if used during initialization or restart. Section 11 is disabled and is not executed when this command is entered.

After entering KBPON, press the space bar to produce the following message:

```
KEYPOINT DATA RECORDING WILL BEGIN AT WORD ADDR=00E200
ANOTHER STARTING ADDRESS DESIRED (Y/N)
```

Press any character other than Y to begin test execution. The running message is displayed. The Y response produces the next message starting on line 18:

```
ENTER NEW ADDRESS (HEX WORD) - MUST BE GREATER THAN 00E400
```

Type in the desired address. If an address less than E400 is entered or a page cannot be allocated for the address you entered, the following message appears on line 19:

```
PAGE UNAVAILABLE - PLEASE RE-ENTER
```

The test gives you three chances to specify a new address. If a page cannot be assigned after three attempts, the test restarts automatically.

Normal Messages

No other messages are displayed except those described previously.

Error Messages

When KYPT encounters an error, it displays an error message with the following format:

```
                                                                    Line

KYPT  SE  PCXXXX  SXXXX  SBXXXX  TEXXXX                              1
(Comment Field)  SLXXXX PLXXXX GZXXXX  SV=XXXXXXXX PV=XXXXXXXX      2

                                                                     3
TEST DESCRIPTION -                                                   4
(Test Description goes here)                                         .
                                                                     .
ERROR MESSAGE                                                        .
```

The first two lines of the display contain the running header message. The test description is a brief message to the operator. This message outlines the failing section or subsection functions to help with error interpretation.

NOTE

The message INVALID ENTRY appears in the comment field when the command entered by the operator is invalid.

Error Messages for Sections 0 through 7

SECT0, SUBSECTs 0 through 15

```
VERIFY RECORDING MECHANISM WITH KEYPOINT ENABLE FLAG = 0
EXECUTES 1 KEYPOINT INSTRUCTION OF CLASS=SUBSECT AND VERIFIES
NO ENTRY IS MADE TO CENTRAL MEMORY.

A KEYPOINT ENTRY WAS MADE WHEN KEYPOINT WRITES TO CENTRAL
MEMORY WERE DISABLED (KEF=0).
```

SECT0, SUBSECT 16

```
VERIFY RECORDING MECHANISM WITH KEYPOINT ENABLE FLAG = 0
EXECUTES ALL KEYPOINT INSTRUCTION CLASSES IN A ROW AND VERIFIES
NO ENTRIES ARE MADE TO CENTRAL MEMORY.

A KEYPOINT ENTRY WAS MADE WHEN KEYPOINT WRITES TO CENTRAL
MEMORY WERE DISABLED (KEF=0).
```

SECT1, SUBSECTs 0 through 15

```
VERIFY RECORDING MECHANISM WITH KEYPOINT MASK REGISTER = 0000
EXECUTES 1 KEYPOINT INSTRUCTION OF CLASS=SUBSECT AND
VERIFIES NO ENTRY IS MADE TO CENTRAL MEMORY.

A KEYPOINT ENTRY WAS MADE WHEN KEYPOINT WRITES TO CENTRAL
MEMORY WERE DISABLED (KMR=0000).
```

SECT1, SUBSECT 16

```
EXECUTES ALL KEYPOINT INSTRUCTION CLASSES IN A ROW AND
VERIFIES NO ENTRIES ARE MADE TO CENTRAL MEMORY.

A KEYPOINT ENTRY WAS MADE WHEN KEYPOINT WRITES TO CENTRAL
MEMORY WERE DISABLED (KMR=0000).
```

SECT2

```
VERIFY KEYPOINT BUFFER POINTER UPDATING CORRECT
EXECUTES RANDOM NUMBER OF KEYPOINT INSTRUCTIONS (LESS THAN
NUMBER OF WORDS ON A PAGE). CALCULATES BYTE NUMBER
DIFFERENCE BETWEEN FINAL AND INITIAL KBP VALUE.  CONVERTS TO
WORD AND VERIFIES IT IS EQUAL TO RANDOM INSTRUCTION COUNT.

UPDATING OF KEYPOINT BUFFER POINTER INCORRECT
                  PVA       RMA (WORD)
START          XXXXXXXXXXXX    XXXXXXXX
END   (EXP)    XXXXXXXXXXXX    XXXXXXXX
END   (RCV)    XXXXXXXXXXXX    XXXXXXXX
```

NOTE

When executing on-line, the RMA information is blank.

SECT3, SUBSECTs 0 through 15

```
VERIFY RECORDING MECHANISM WITH KMR REGISTER NONZERO WITH
ONLY ONE BIT SET IN KMR, EXECUTES ALL KEYPOINT INSTRUCTION
CLASSES IN A ROW AND VERIFIES ONLY ONE ENTRY WAS MADE TO
CENTRAL MEMORY. (SUBSECT X HAS BIT X OF KMR SET, X=0-15.)

A KEYPOINT ENTRY OF CLASS X WAS NOT MADE WHEN CORRESPONDING
BIT TO ENABLE RECORDING SET  (KMR=XXXX).
```

SECT4

— Test Description

```
VERIFY KEYPOINT TIMER VALUE UNIQUE EXECUTES AN FRC READ, 1
KEYPOINT INSTRUCTON, AND A FINAL FRC READ. VERIFIES THAT
KEYPOINT FRC VALUE IN BETWEEN PRE AND POST READ VALUES.
```

— Preread Error Message

```
KEYPOINT FRC VALUE NOT GREATER THAN PREREAD VALUE
PREREAD  VALUE=XXXXXXXX
KEYPOINT VALUE=XXXXXXXX
```

— Post Read Error Message

```
KEYPOINT FRC VALUE NOT LESS THAN POSTREAD VALUE
KEYPOINT VALUE=XXXXXXXX
POSTREAD VALUE=XXXXXXXX
```

SECT5, SUBSECT 0 through 1

```
VERIFY CLASS AND CODE FIELDS CORRECT
EXECUTES KEYPOINT INSTRUCTION OF RANDOM CLASS AND CODE. FROM
ENTRY IN CENTRAL MEMORY, VERIFIES CLASS=J AND CODE=XKR+Q.
```

— Subsect 0 Error Message

```
CLASS NUMBER FIELD OF KEYPOINT ENTRY NOT EQUAL TO J FIELD.
EXPECTED=X
RECEIVED=X
```

— Subsect 1 Error Message

```
CODE FIELD OF KEYPOINT ENTRY NOT EQUAL TO XKR+Q.
EXPECTED=XXXXXXXX
RECEIVED=XXXXXXXX
```

SECT6, SUBSECT 0

```
VERIFIES CORRECT XK INTERPRETATION
ENTERS 1 INTO X0. EXECUTES KEYPOINT INSTRUCTION WITH K=0,Q=0.
FROM ENTRY IN CENTRAL MEMORY VERIFIES CODE=0.

X0 REG INTERPRETED AS EQUAL TO 1, NOT AS EQUAL TO 0.
```

SECT6, SUBSECT 1

```
VERIFIES CORRECT XK INTERPRETATION
SETS BIT 31 IN X1. EXECUTES KEYPOINT WITH Q=1,J=2. FROM ENTRY IN
CENTRAL MEMORY VERIFIES THAT CONCATENATED CLASS AND CODE
FIELD=200000001 AND NOT=300000001.

XK AND NOT XKR WAS USED FOR FORMATION OF KEYPOINT CODE
CONCATENATED CLASS AND CODE FIELD IS AS FOLLOWS
EXPECTED=XXXXXXXXX
RECEIVED=XXXXXXXXX
```

SECT7

```
VERIFY NO ARITHMETIC OVERFLOW DETECTED WHEN FORM CODE
SETS X1R=80000000.  EXECUTES KEYPOINT WITH Q=8000.  FROM ENTRY IN
CENTRAL MEMORY VERIFY THAT CODE=7FFF8000.

ARITHMETIC OVERFLOW WAS DETECTED DURING ADDITION OF XKR AND
Q TO FORM CODE
EXPECTED=XXXXXXXX
RECEIVED=XXXXXXXX
```

Error Messages for Sections 8 through 11

Sections 8 through 11 share a common error message. This message provides expected versus actual values of the following:

- Process virtual address (PVA) and real memory address (RMA) of the P register where the interrupt should have occurred
- The corresponding untranslatable pointer (UTP)

The common error message has the format:

```
          EXP                 ACT

P (PVA)   XXXXXXXXXXXX        XXXXXXXXXXXX
  (RMA)   XXXXXXX             XXXXXXX
UTP       XXXXXXXXXXXX        XXXXXXXXXXXX
```

The following are test descriptions for sections 8 through 11:

SECT8, SUBSECTs 0 and 1

— Subsect 0 Test Description

```
VERIFY INVALID SEGMENT INTERRUPT
INITIALIZES THE KBP TO POINT TO AN INVALID SEGMENT (VL BITS OF
SDE=00).
```

— Subsect 1 Test Description

```
VERIFY INVALID SEGMENT INTERRUPT
THE VALUE OF THE SEGMENT NUMBER IN THE KB IS GREATER THAN
THE SEGMENT TABLE LENGTH.
```

SECT9, SUBSECTs 0, 1 and 2

— Subsect 0 Test Description

```
VERIFY ACCESS VIOLATION INTERRUPT
INITIALIZES THE KBP TO POINT TO A NONWRITABLE SEGMENT.
```

— Subsect 1 Test Description

```
VERIFY ACCESS VIOLATION INTERRUPT
INITIALIZES THE KBP TO POINT TO A SEGMENT WITH R1=A (R1A).
EXECUTES KEYPOINT INSTRUCTION FROM CODE SEGMENT WHICH IS NOT
IN WRITE RING BRACKET OF R1A.
```

— Subsect 2 Test Description

```
VERIFY ACCESS VIOLATION INTERRUPT
INITIALIZES THE KBP TO POINT TO LOCK SEGMENT (LOCK=6).  EXECUTES
KEYPOINT INSTRUCTION FROM CODE SEGMENT (KEY=5)
```

SECT10

```
VERIFY ADDRESS SPEC ERROR INTERRUPT
INITIALIZES KBP PVA WITH BIT 32 SET
```

SECT11, SUBSECTs 0 and 1

— Subsect 0 Test Description

```
VERIFY PAGE FAULT INTERRUPT
INITIALIZES THE KBP TO HAVE A BYTE NUMBER=PAGE SIZE+8
```

— Subsect 1 Test Description

```
VERIFY PAGE FAULT INTERRUPT
EXECUTES MORE KEYPOINT INSTRUCTIONS THAN NUMBER OF WORDS ON A PAGE
```

PIT/SIT Errors

When PIT or SIT errors are encountered, the pertinent information is displayed on line 2 of the running display. Refer to Running Display for details. Note that the total error count is not increased.

Applications

Refer to Operator Entries and Use of KBPON, KBPOF Commands for test applications.

Cache Test - CACH 11

Introduction 11-1

Requirements 11-1
- Hardware 11-1
- Software 11-2
- Accessories 11-2
- Characteristics 11-2

Operational Procedures 11-3
- Restrictions and User Cautions 11-3
- Loading Procedure 11-3
  - Running Procedures 11-3
- Parameters and Control Words 11-4
  - Parameters 11-4
  - Installation Parameters 11-5
  - Control Words 11-6
- Section Index 11-6

Operator Communication 11-7
- Displays 11-7
  - Initial Display 11-7
  - Running Display 11-8
  - Error Displays 11-8
- Operator Entries 11-9
  - Test Commands 11-9
  - Using Test Commands 11-10
    - Change Test Parameters 11-10
    - Use of Condition Commands 11-10
    - Use of SETS Command 11-10
    - Stop on Error 11-10
    - Repeat Failing Condition 11-11
    - Scope Loop a Failing Condition 11-11
    - Stop After Executing One Condition 11-11
    - Repeat Test 11-11
    - Stop Test While Executing 11-11
    - Display Error Information 11-11
- Normal Messages 11-12
- Error Messages 11-12
  - Recovery 11-12
- Applications 11-12
  - Scope Mode 11-12
  - Tracking Exchanges and Traps 11-12

Cache Test - CACH 11

Introduction

CACH is a model independent cache test for CYBER 180 computer systems which have a cache memory. If run on a system with no cache, the test results are unpredictable. CACH is designed to DETECT errors at the virtual instruction level and runs off-line under Common Maintenance System Executive (CMSE).

CACH is not designed to run on-line under the Diagnostic Virtual System (DVS) because some system instructions (for example PURGE) are used which are not allowed on-line, and time-sharing with other jobs destroys the test environment.

CACH tests the CYBER 180 cache buffer by initializing cache sequentially and accessing it randomly. CYBER 180 load/store instructions are used in the testing sequences.

Requirements

Hardware

This test is intended for models 835, 840, 850, 855, 860, and 990 computer systems.

In addition to CMSE requirements, CACH requires the following:

Target Configuration:

- 1 CY180 CPU with 8 MB of memory
- 1 Disk unit supported by CMSE for test residence
- 1 System console
- Cache/page maps/segment maps
- 20 PPs

Minimum Configuration:

- 1 CY180 CPU with 1 MB of memory
- 1 Disk unit supported by CMSE for test residence
- 1 System console
- Cache memory
- 5 PPs

Software

CACH runs under CMSE.

Accessories

None required.

Characteristics

Description	Characteristics
Program name	CACH
Test/Diag/Utility	Test
Size (source lines. Includes comments)	5813
Size (memory required for execution)	1 MB
Size (total assembled binary)	163704
Level of Isolation	Detect only
Run time	2 min.
Off-line system	CMSE-180
On-line system	N/A
Test control resides in:	CPU
Source language	META
Source code maintenance	SCU

Operational Procedures

Restrictions and User Cautions

1. All instructions must be independently operational before running CACH.
2. The virtual addressing mechanism must be operational.
3. CACH cannot run in RMA mode.
4. CACH does not test A170 instructions.
5. Machines running CACH must have a cache memory.
6. CACH does not run on-line under DVS.

Loading Procedure

CACH is assembled using the CYBER 180 assembler. The binary output is linked to the binary of VEXC using the CYBER 180 LINKER/LOADER to form the complete executable module. The binary output can then be written to disk using TDX. To run off-line, the executable binary is loaded and executed using the CMSE command buffer capability. Consult the MSL15X reference manual for a description of the command buffer structure.

Running Procedures

During default execution, the test bypasses monitor options. Bit 37 of Parameter Word 0 (PW0) is set to enable monitor bypass. In this mode, CACH runs with page faulting and map/cache purging disabled. This is done to increase the speed of the test and to avoid destroying the cache environment while page faults are satisfied. The PIT and SIT timers are disabled for the same reason. If the operator chooses to run with monitor options, bit 37 should be cleared. If bit 36 of PW0 is clear, the operator can select values. In this case, the first messages displayed will be done by VEXC and deals with test environment selection (for example, `RUN WITH PAGE FAULTS?, RUN WITH LIMITED WORKING SET SIZE?`). Operator input is in the form of (Y)es/(N)o answers. If bit 36 of PW0 is set, VEXC uses the information in PW12 to set up the test environment.

The first display from CACH is a parameter stop display. See Test Commands for a list of available commands. A space bar starts test execution.

Parameters and Control Words

Parameters

Tag	RMA Byte Address (Hex)	Bits	Description	Default Setting
PW0	66100	0-63		00000000 04014010_{16}
		0-35	Not used	0
		36	Monitor-default run (CMSE EC command)	0
	(1)	37	Monitor bypass	1
	(1)	38	Test mode one	0
	(1)	39	Test mode two	0
		40-44	Not used	
	(2)	45	Batch mode	0
	(2)	46	On-line control	0
	(2)	47	Off-line control	1
		48	Operator stop	0
		49	Initial stop for parameters	1
		50-53	Not used	0
		54	Repeat condition (SRC/CRC)	0
		55-56	Not used	
		57	Repeat test (SRT/CRT)	0
		58	Log errors (SLE/CLE)	0
		59	Stop on error (SSE/CLE)	1
		60	Stop end of condition	0
		61-62	Not used	0
		63	Stop at end of test (SST/CST)	0
			NOTE (1) Used in test development only. (2) Used by DVS – not for off-line use.	
PW1	66108	0-63	Repeat counter	1000_{10}
PW2	66110	0-63	Section select bits	$1FFF_{16}$
PW3-PW5			Not used	
PW6	66130	0-63	Display select control (set by DS,x command)	0
PW7	66138	0-63	Skip pass count (set by P,x command)	1
PW8	66140	0-63	PIT value (set by PV,X command)	100_{10}
PW9	66148	0-63	SIT value (set by SV,X command)	150_{10}
PW10-PW11			Not used	
PW12	66160	0-63	Monitor control bits. See 7.4 for bit definitions.	$F0000000E_{16}$

Tag	RMA Byte Address (Hex)	Bits	Description	Default Setting
PW13	66168	0-63	Scope loop control (set by LOOP1 command).	0
PW14-PW15			Not used	
PW16	66180	0-63	Starting condition count (set by C,xxx command) Default = 1	1
PW17	66188	0-63	Maximum condition count (set by MXC,xxx) Default = 10000	10000_{10}
PW18	66190	0-6	PIT control (set by PITON/PITOF command)	0
PW19	66198	0-63	SIT control (set by SITON/SITOF command)	0
PW20	661A0	0-63	Cache sets enabled (set by SETS,x) Default = 4	4

Installation Parameters

The parameter and control word segment is shared by CACH (job) and VEXC (monitor). PW12 is used by VEXC to set up the running environment.

PW12 is set to $F0000000E_{16}$ during a compile of CACH. PW0, bit 36 interacts with PW12. The default is PW0, bit 36 clear. When set, VEXC looks at PW12 for environment control bits. These can be changed in the command buffer using the CMSE EC or EB command.

Bit definitions for PW12 are as follows:

Bit	Meaning	Default
28	Run with repetitive page faults	1
29	Run with cache purges	1
30	Run with map purges	1
31	Run with page faults	1
32-63	Number of pages in working set	E_{16}

Control Words

The starting RMA for the parameter word block is given in the initial parameter stop display and in the running message. Control words reside above parameter words. The control words are used by the test and are informative only. Parameter words can be altered by the operator.

Tag	RMA Byte Address (Hex)	Description	Default Setting
CW0	66000	Program name	ASCII 'CACH'
CW1	66008	System type (Not used)	0
CW2	66010	Error code	0
CW3	66018	Error count	0
CW4	66020	Pass count	0
CW5	66028	Section number	0
CW6		Not used	
CW7	66038	Condition count	0
CW8-31		Not used	

Section Index

CACH is organized into sections and conditions within each section. The sections are defined as follows:

1. Randomly accesses cache within addresses available using seven load instructions.
2. Randomly accesses cache within addresses available using six load and one store instruction (LXI,LXI,LXI,SXI,LXI,LXI,LXI).
3. Randomly accesses cache within addresses available using five load and two store instructions (LXI,LXI,LXI,SXI,SXI,LXI,LXI).
4. Repeats section 1 within addresses available times two.
5. Repeats section 2 within addresses available times two.
6. Repeats section 3 within addresses available times two.
7. Repeats section 1 using three different segments (that is, three different ASIDs).
8. Repeats section 2 using three different segments.
9. Repeats section 3 using three different segments.
10. Repeats section 7 using load byte instructions with random byte lengths.
11. Repeats section 8 using load/store byte instructions with random byte lengths.
12. Repeats section 9 using load/store byte instructions with random byte lengths.
13. Randomly accesses cache using all load/store instructions within addresses available and using 3 different segments. (Op codes 82,83,84,85,A0,A1,A2,A3,A4,A5,D8,DF).

Each section loops through the load/store sequences for a default of 10,000 conditions. Each condition uses different random numbers to calculate the random cache addresses read and written.

Operator Communication

Displays

Initial Display

```
C A C H E   T E S T        xxxx PAGES  CACHE-x.x    mm/dd/yy
CACH  CC  PC=xxxxxx S=xxxx COND=xxxxx EC=xxxx TE=xxxx PA=xxxxx
(COMMENT FIELD)                      PV=ALL ONES, SV=ALL ONES

   PW0  - XXXX XXXX XXXX XXXX  CONTROL BITS
   PW1  - XXXX XXXX XXXX XXXX  REPEAT COUNTER
   PW2  - XXXX XXXX XXXX XXXX  SECTION SELECT
   PW6  - XXXX XXXX XXXX XXXX  DISPLAY SELECT
   PW7  - XXXX XXXX XXXX XXXX  STARTING PASS COUNT
   PW8  - XXXX XXXX XXXX XXXX  PIT VALUE
   PW9  - XXXX XXXX XXXX XXXX  SIT VALUE
   PW12 - XXXX XXXX XXXX XXXX  MONITOR CONTROL
   PW13 - XXXX XXXX XXXX XXXX  SCOPE LOOP CONTROL
   PW16 - XXXX XXXX XXXX XXXX  STARTING CONDITION
   PW17 - XXXX XXXX XXXX XXXX  MAX CONDITION COUNT
   PW18 - XXXX XXXX XXXX XXXX  PIT CONTROL
   PW19 - XXXX XXXX XXXX XXXX  SIT CONTROL
   PW20 - XXXX XXXX XXXX XXXX  CACHE SETS ENABLED
```

where:

xxxx PAGES	Number of pages in use if option selected
x.x	Current version of VEXC monitor
CC	One of the following: RU Running SP Stop for parameters SE Stop on error SS Stop end of section ST Stop at end of test RS Repeat section RT Repeat test SC Stop at end of condition RC Repeat condition
PC	Pass count (decimal)
S	Section number running (decimal)
COND	Condition count (decimal)
EC	Error code (not used)
TE	Total number of errors (decimal)
PA	Address of parameter words (hex-byte address)
PV	PIT value running with
SV	SIT value running with

Comment Field = Keyboard input error message. See Error Displays.

If running with small PIT/SIT value, line 3 will display the following information:

```
(Comment Field)  SL=xxxx  PL=xxxx  GZ=xxxx  PV=xxxxxxxx,  SV=xxxxxxxx
```

where:

SL Number of SIT losses
PL Number of PIT losses
GZ Number of PIT interrupts with PIT value greater than zero.

At this time, the operator may enter CSME commands or CACH test commands. See Test Commands for CACH commands.

Running Display

```
C A C H E   T E S T        xxxx PAGES  CACHE-x.x    mm/dd/yy
CACH  RU  PC=xxxxxx S=xxxx COND=xxxxx EC=xxxx TE=xxxx PA=xxxxx
(   Comment Line             )      PV=ALL ONES, SV=ALL ONES
```

Error Displays

1. When an error is detected and stop-on-error is set, CACH displays the following:

```
C A C H E   T E S T        xxxx PAGES  CACHE-x.x    mm/dd/yy
CACH  SE  PC=xxxxxx S=xxxx COND=xxxxx EC=xxxx TE=xxxx PA=xxxxx
                                      PV=ALL ONES, SV=ALL ONES

  CACHE COLUMN = xx       RMA = xxxxxx

  ADDRESS          EXP DATA          ACT DATA
xxxxxxxxxxxx  xxxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
```

   On line 4, the cache column and the Real Memory Address (RMA) of the address in error are displayed. On line 7, the Process Virtual Address (PVA) of the same memory cell that caused the error is displayed along with the expected and actual data.

   See Appendix D for a diagram of the relationship between central memory addresses and cache addresses.

2. When a CACH command is entered incorrectly, CACH displays the following message in the comment field in line 2. A valid command clears the error message.

Message	Explanation
INVALID ENTRY	CACH does not recognize the command entered, or does not accept the command parameters. See Test Commands for acceptable ranges.

Operator Entries

Test Commands

Refer to Parameters for bit descriptions and default values of associated parameter words.

Command	Response
S	Stop test execution.
R	Restart CACH from initial VEXC entry.
(Space bar)	Continue test execution.
P,xxxxxxxx	Skips forward to pass count xxxxxxxx prior to test execution. Default = 1. Input in decimal.
C,xxx	Sets starting condition to xxx. This allows recreation of random numbers associated with any condition. Default = 1. Input in decimal.
MXC,xxx	Set maximum condition count for each section. Default = 10,000 decimal. Input in decimal.
SETS,x	Defines the number of cache sets available. Values 1-4 allowed. Default = 4 sets available. Input in decimal.
DS,x	Select display X. Display selections are 0 through 2. Default = 0. 0 = Running message 1 = Initial test display (Parameter words) 2 = Generalized memory display
PITON	Enables running with small value in Process Interval Timer (PIT). The PIT is a 32 microsecond counter. Default = off.
PITOF	Disables running with small value in PIT. PIT value will be set to -1 and PW8 ignored.
PV,X	Enter selected value into PIT. Default small PIT value = 100 microseconds (decimal). Input in decimal.
SITON	Enables running with small value in System Interval Timer (SIT). The SIT is a 32 microsecond counter. Default = off.
SITOF	Disables running with small value in SIT. SIT value will be set to -1 and PW9 ignored.
SV,X	Enter selected value into SIT. Default small SIT value = 150 microseconds (decimal). Input in decimal.
LOOP	Sets scope mode and continuously executes present condition.

Command	Response
SSE/CSE	Set/Clear stop on error parameter bit.
SRT/CRT	Set/Clear repeat test pass parameter bit.
SST/CST	Set/Clear stop at end of test pass parameter bit.
SLE/CLE	Set/Clear log errors parameter bit.
SRC/CRC	Set/Clear repeat condition parameter bit.
SSC/CSC	Set/Clear stop at end of condition parameter bit.
SSS/CSS	Set/Clear stop at end of section parameter bit.

Using Test Commands

Change Test Parameters

Test parameters can be changed before starting CACH execution, while running, and after stopping use of CACH, CMSE EC, and/or EB commands.

Bit 49 of PW0 is set by default. This allows initial parameter stop. If cleared by CMSE EC or EB commands, CACH does not stop for initial parameter changes; the test uses default values for all available parameter settings. If you restart CACH by typing an R command, the parameters set when CACH is started are saved and reinstated.

Use of Condition Commands

Type C,xxxx to set the starting condition count. This calculates the random numbers used for addressing cache up to xxxx condition. The default starting count is 1.

Enter MXC,xxx to set the maximum number of conditions for each section. The default is $10{,}000_{10}$. When each section has executed the default or selected number of conditions, the next section begins.

Use of SETS Command

Type SETS,x to enter the number of cache sets available. The cache addresses are calculated using input (x * $3FF_{16}$ = addresses available). When disabling cache sets via the Dependent Environment Register (DEC), use this command. After changing the number of sets enabled via the DEC, restart the test. Then enter the SETS,x command.

Stop on Error

Type SSE to set the stop-on-error parameter bit and cause CACH to stop when an error is detected. Type CSE to clear the bit, CACH does not stop on error but increments the error count if errors are detected.

Whenever CACH stops, enter a space to resume execution.

Repeat Failing Condition

When CACH stops on an error, entry of SRC command sets the repeat-condition parameter bit. Entry of a space causes CACH to re-execute the failing set of operands and instructions.

Type in CRC to clear the repeat-condition bit.

To return to a failing situation from test initialization, enter the failing pass count with a P,xxx command, enter failing condition count with C,xxx command, and enter an SRC command. The test generates random numbers to selected pass and condition counts, begins execution, and repeats execution until CRC is entered.

If an error is detected, the error count increments during repeat condition execution.

Scope Loop a Failing Condition

If a scope loop is desired, after CACH has stopped on an error, type LOOP and then a space bar. The test then allows looping series of instructions with the particular random number of the condition that detected the failure.

During scope mode, CACH does not communicate with the operator. For scope triggering, a sync 0100 instruction is executed before execution of the instruction stream.

To stop scope looping, clear PW13 with the CMSE instruction EC. The address of this word is displayed in the scope loop message.

Stop After Executing One Condition

SSC sets the stop-at-end-of-condition parameter bit. CACH will stop after execution of one condition of the instruction set. Space bar executes the next condition, which has the same instructions but with new random addresses, and then stops.

CSC clears the bit and CACH executes successive conditions continuously.

Repeat Test

Set the repeat-test parameter bit with SRT. CACH then repeats the test using the same starting random number seed for each repetition of a pass.

SRT clears the bit and CACH continues to the next pass. Note that, if an error is detected, the error count increments during the repeat test pass execution.

Stop Test While Executing

Use an S command to stop CACH execution. Use a space bar to resume execution.

Display Error Information

Type in DS,x to direct CACH to print a selected display. Refer to Test Commands for a description of the x parameter.

Normal Messages

See Initial Display and Running Display.

Error Messages

See Error Displays.

Recovery

An unidentified interrupt causes CACH to exchange to monitor (VEXC) and display a message that describes the error. The operator can either scope loop this interrupt with a loop offered at the time, or re-start the test by declining the scope loop message. Rerun the command buffer to restart CACH.

The system will not abort CACH.

Applications

Scope Mode

Scope mode instructions are in Using Test Commands.

Tracking Exchanges and Traps

The VEXC monitor keeps a number of informational buffers that are useful in tracking exchanges and traps. When not running in default mode (bit 36 of PW0 clear), VEXC displays the addresses and format of these buffers. The display can be viewed by clearing this bit with a CMSE EC or EB command before the test begins or by clearing it in the command buffer and restarting the test with the command buffer. These buffer tables begin at byte address 23000_{16}.

Model Independent General Design Specification Tests - MIGDS 12

Introduction 12-1
Floating Point Tests 12-1
Integer Tests 12-1
BDP Tests 12-1
System Tests 12-2

Requirements 12-2
Hardware 12-2
Software 12-2
Accessories 12-3
Characteristics 12-3

Operational Procedure 12-4
Restrictions and User Cautions 12-4
Loading Procedure 12-4
Running Procedure 12-4
Parameter and Control Words 12-5
Parameters 12-5
Installation Parameters 12-6
Control Words 12-7
Section Index 12-7

Operator Communication 12-8
Displays 12-8
Initial Display 12-8
Running Display 12-9
Results Displays 12-9
Run All Tests Mode Result Display 12-9
Full Test Mode Result Display 12-11
Single Test Mode Results Display 12-11
Operator Entries 12-12
Test Commands 12-12
Y/N Entries 12-12
R, T, S, C, D Entries 12-12
Test Communication 12-12
Test Control Messages 12-12
Informative Messages 12-16
Operator Error Messages 12-16
Test Error Messages 12-17
Normal Messages 12-17
Error Messages 12-17
Recovery 12-17
Applications 12-18
Scope Mode 12-18
Tracking Exchanges and Traps 12-18
Quick Look (NUMR,EDIT) 12-18

Exceptions - RNGT and CRPT 12-19
RNGT 12-19
CRPT 12-23

Model Independent General Design Specification Tests - MIGDS 12

Introduction

A number of model independent tests are available for testing end case interrupts for the CYBER 180 instruction set. These tests cover BDP instructions, full and half integer instructions, and floating point instructions. They are designed with the same basic test flow and differ only according to the instructions tested. The main purpose of these tests is to fully check out the instruction execution while placing heavy emphasis on end case and stress testing. These tests run off-line on all CYBER 180 systems under CMSE (Common Maintenance System Executive) and on-line under DVS (Diagnostic Virtual System).

The test names and descriptions are as follows:

Floating Point Tests

These tests cover all single and double precision floating point instructions.

- SNGL - tests ADDF, SUBF, MULF, DIVF, CMPF, CNIF, CNFI
- DOBL - tests ADDD, SUBD, MULD, DIVD
- BRCH - tests BRFEQ, BRFNE, BRFGT, BRFGE, BROVR, BRUND, BRINF

Integer Tests

These tests cover all full and half integer instructions (register to register) and all full and half immediate instructions (register to immediate operand).

- FINT - tests ADDX, SUBX, MULX, DIVX, CMPX (FULL INTEGER)
- HINT - tests ADDR, SUBR, MULR, DIVR, CMPR (HALF INTEGER)
- FIMM - tests INCX, DECX, ADDXQ, MULXQ (FULL IMMEDIATE)
- HIMM - tests INCR, DECR, ADDRQ, MULRQ (HALF IMMEDIATE)

BDP Tests

These tests cover BDP numeric and BDP immediate instructions. Additionally, NUMR and EDIT can optionally run in a quick look mode.

- NUMR - tests ADDN, SUBN, MULN, DIVN, CMPN, SCLN, SCLR, MOVN
- BIMM - tests MOVI, CMPI, ADDI
- BYTE - tests CMPB, CMPC, SCNB, TRANB, MOVB
- EDIT - tests the EDIT instruction

System Tests

These tests cover several system instructions and system conditions.

- RNGT – tests ring/key/lock protection mechanisms
- CRPT – test CALL, RETURN, POP

These tests follow a different format from other MIGDS tests. See chapter 5 for a description of CRPT and RNGT.

Requirements

Hardware

These tests are intended for CDC CYBER 170/180 models 810, 815, 825, 830, 835, 840, 845, 850, 855, 860, and 990 computer systems.

Target configuration:

- 1 C180 CPU with 8 MB of memory
- 1 disk unit supported by CMSE for test residence
- 1 system console
- Cache/page maps/segment maps
- 20 PPs

Minimum configuration:

- 1 C180 CPU with 1 MB of memory
- 1 disk unit supported by CMSE for test residence
- 1 system console
- 5 PPs

Software

The MIGDS tests run off-line under CMSE and on-line under DVS.

Accessories

None required.

Characteristics

Description	Characteristics
Program names	SNGL/DOBL/BRCH FINT/HINT/FIMM/HIMM/NUMR BIMM/BYTE/EDIT/CRPT/RNGT
Test/Diag/Utility	Test
Size (memory required for execution)	1 MB
Level of isolation	Detect only
Off-line system	CMSE-180
On-line system	DVS
Test control resides in:	CPU
Source language	META
Source code maintenance	SCU

Operational Procedure

Restrictions and User Cautions

1. The virtual addressing mechanism must be operational.
2. MIGDS tests cannot run in RMA mode.
3. MIGDS tests do not test any A170 instructions.

Loading Procedure

These tests are assembled using the C180 assembler (CPAS180 or ASM180). The binary output is linked to the binary of VEXC and TRPUTL using the C180 LINKER/LOADER to form the complete executable module. It can then be written to disk using TDX. To run off-line, the executable binary is loaded and executed using the CMSE command buffer capability. Consult the MSL15X reference manual for a description of the command buffer structure. To run on-line, consult the DVS On-line Usage manual for loading procedures.

Running Procedure

During default execution, the tests run with default environment set-up values. If bit 36 of Parameter Word 0 (PW0) is clear, the operator can select values. In this case, VEXC displays the first messages and deals with test environment selection. For example, the first messages displayed could be RUN WITH PAGE FAULTS? or RUN WITH LIMITED WORKING SET SIZE? Operator input is in the form of (Y)es/(N)o answers. If bit 36 of PW0 is set, VEXC uses the information in PW12 to set up the test environment.

If bit 49 of PW0 is set, the first display from these tests is a parameter stop display. Refer to Test Commands for a description of this display. The tests then present a series of questions dealing with the test mode of operation and wait for input from the operator. See Test Commands for a list of questions and input required. In the default mode, bit 49 of PW0 is clear, and the tests use the values in the various parameter words for input and do not stop for operator input.

Parameter and Control Words

Parameters

Tag	RMA byte Address (Hex)	Bits	Description	Default Setting
	CPU0/CPU1			
PW0	A6100/A7100	0-63	Control Bits	00000000 08000000
		0-35	Not used	0
		36	Monitor-default run (CMSE EC command)	1
		37	Monitor nypass	0
		38	Test mode one	0
		39	Test mode two	0
		40-44	Not used	0
		45	Batch mode	0
		46	On-line control	0
		47	Off-line control	0
		48	Not used	0
		49	Initial stop for parameters	0
		50-51	Not used	
		52	Quick look mode	0
		53-57	Not used	0
		58	Log errors	0
		59	Stop on every error	0
		60-63	Not used	
PW1	A6108/A7108	0-63	Repeat counter	1
PW2-PW5			Not used	
PW6	A6130/A7130	0-63	Display select control	1
PW7	A6138/A7138	0-63	Skip pass count	1
PW8	A6140/A7140	0-63	Default PIT value	100(10)
PW9	A6148/A7148	0-63	Default SIT value	150(10)
PW10			Not used	
PW11	A6158/A7158	0-63	TRAP capability (0=operative)	0
PW12	A6160/A7160	0-63	Monitor control bits (See Installation Paramters for bit definitions)	F0000000C(16)
PW13	A6168/A7168	0-63	Scope loop control	0
PW14-PW15			Reserved	
PW16	A6180/A7180	0-63	Forced error count	0
PW17	A6188/A7188	0-63	Forced error type	0

NOTE

Bits 32 through 63 of PW12 may vary with each test. The more activity internal to a test, the greater the number of pages necessary to run with. See Installation Parameters.

Installation Parameters

The parameter and control word segment is a segment shared by the test (job) and VEXC (monitor). PW12 is used by VEXC to set up the running environment. PW12 is set to F0000000C(16) during compile of each test. PW0, bit 36 interacts with PW12. The default is PW0, bit 36 set. When set, VEXC looks at PW12 for environment control bits. These can be altered in the command buffer using the CMSE EC or EB commands.

Bit definitions for PW12 are as follows:

Bit	Meaning	Default
27	Run with 32 page table search	0
28	Run with repetitive page faults	1
29	Run with cache purges	1
30	Run with map purges	1
31	Run with page faults	1
32-63	Number of pages in working set	C(16)

The number of pages in the working set will vary with each test. The following table shows the default values for this parameter for each test. If a smaller number is selected, the test runs more slowly, and if the number is too small the test may hang while trying to validate enough pages to continue.

Test Name	Default Value (decimal)
SNGL	2
DOBL	2
BRCH	2
FINT	2
HINT	1
FIMM	2
HIMM	2
NUMR	7
BIMM	6
BYTE	7
EDIT	6
RNGT	0
CRPT	0

Control Words

The starting RMA for the parameter word block is given in the initial parameter stop display and the running message. Control words reside above parameter words. The control words are used by the test and are only informational. Parameter words can be altered by the operator.

Tag	RMA Byte Address (HEX)	Description	Default Setting
	CPU0/CPU1		
CW0	A6000/A7000	PROGRAM NAME	ASCII TEST NAME
CW1	A6008/A7008	SYSTEM TYPE	AA8(16)
CW2	A6010/A7010	ERROR CODE (Not used)	0
CW3	A6018/A7018	ERROR COUNT	0
CW4	A6020/A7020	PASS COUNT	0
CW5-31		Not used	

Section Index

MIGDS's have no selectable test sections.

Operator Communication

Displays

Initial Display

```
HALF WORD INTEGER   PA=XXXXXX  XXXX PAGES  CPU X  X.X  YY/MM/DD
THIS TEST OPERATES IN ONE OF THREE MODES -
    1. ALL PRE-SET OPERANDS THROUGH ALL OP CODES
    2. ALL PRE-SET OPERANDS THROUGH ONE OP CODE
    3. A SINGLE PAIR OF OPERANDS THROUGH ONE OP CODE
    OPERANDS MAY BE CHOSEN FROM ONE OF THREE PLACES -
        1. THE RESULT TABLE - DISPLAYED AFTER EACH FULL TEST
        2. THE OPERAND TABLE WHICH HAS PRE-SET OPERANDS
        3. THEY MAY BE ENTERED FROM THE KEYBOARD
SCOPE LOOPS ARE AVAILABLE FOR ANY CONDITION

     PARTIAL LIST OF PARAMETERS - BYTE ADDRESSED   PARAM 0 BITS
PW0 0A6100    0601 4000  CONTROL BITS              TEST MODE 1-38
PW1 0A6108    0000 0001  REPEAT COUNT              TEST MODE 2-39
PW7 0A6138    0000 0001  STARTING PASS COUNT      STOP PARAMS-49
PW8 0A6140    0000 0064  PIT VALUE                LOG ERRORS -58
PW9 0A6148    0000 0096  SIT VALUE                STOP ON ERR-59

ARE TRAPS INPLEMENTED ON THIS MACHINE  (Y/N)
```

This display gives an overview of the test modes available. The question at the bottom line is the first in a series of questions presented to set up the desired test mode and testing environment. See Test Communication for an explanation of the questions.

Line 1 displays the title of the test and the following information:

PA	=	Parameter block byte address
XXXX PAGES	=	Number of pages in use if option selected
CPU X	=	CPU ident (0 or 1 if dual CPU system)
X.X	=	Current version of VEXC monitor

The partial list of parameters gives the byte address, right most 32 bits, and verbal description of the contents of several frequently used parameter words. The right side of the screen gives the bit number of several frequently used bits in PW0.

Running Display

```
PCXXXXXX TE=X UM=XXXX  OPERN YYY,ZZZ      PV=XXXXXXXX SV=XXXXXXXX
```

The running display is printed on line 7 of the screen:

where:

PC	Pass Count (decimal)
TE	Trap Enable flipflop setting.
	0 traps disabled
	1 traps enabled
UM	User Mask value (Hex)
OPERN	Mnemonic of opcode being run.
YYY	Index to canned operand table for Xj (decimal)
ZZZ	Index to canned operand table for Xk (decimal)
PV/SV	Values being used in the PIT and SIT timers (decimal)

NOTE

For three BDP tests (NUMR/BYTE/EDIT), there will be a third index. This is the index to the canned Ai operand table (scale count index, translate/collate table index, or edit mask index).

Results Displays

Run All Tests Mode Result Display

```
FULL WORD INTEGER   PA = XXXXXX  XXXX PAGES  CPU X  X.X  YY/MM/DD
TOTAL NUMBER OF ERRORS  10051, TOTAL NUMBER OF PASSES     1

                    ADDX  SUBX  MULX  DIVX  CMPX

TRAPS   UM CLR      3233   N/A   N/A   N/A   N/A
DISABLED
        UM SET      3233   N/A   N/A   N/A   N/A

TRAPS   UM CLR      3233   N/A   N/A   N/A   N/A
ENABLED
        UM SET       352   N/A   N/A   N/A   N/A

TOTAL ERRORS       10051   N/A   N/A   N/A   N/A

DISPLAY RESULTS (Y/N)
```

This display is presented at the end of a RUN ALL TESTS mode run. Total number of errors for each operation for each case of TE and UM setting is displayed. The mnemonics for the operations are test dependent. In this case, RUN ALL TESTS mode was selected but only ADDX was enabled. The other operations are flagged as not applicable.

The BDP tests (BYTE/NUMR/BIMM/EDIT) show only the two conditions run: Traps Disabled, UM clear and Traps Enabled, UM set.

If no errors occur, the last line displays 'ANOTHER TEST (Y/N)'.

If displaying errors is desired and a yes is entered to the 'DISPLAY RESULTS' question, the following screen will be presented:

```
 FULL WORD INTEGER   PA=XXXXXX  XXXX PAGES  CPU X  X.X  YY/MM/DD
 TOTAL NUMBER OF ERRORS  10051, TOTAL NUMBER OF PASSES     1
              XK                XJ              X1        UCR P
  1 IN  0000000000000000  0000000000000000                1000
    ACT 0000000000000000  0000000000000000                0000
    EXP 0000000000000000  0000000000000000
24   ADDX  UM=0 TE=0  TRAP EXP=0 ACT=0
  2 IN  0000000000000001  0000000000000000                1000
    ACT 0000000000000001  0000000000000000                0000
    EXP 0000000000000001  0000000000000000
24   ADDX  UM=0 TE=0  TRAP EXP=0 ACT=0
  3 IN  0000000000000002  0000000000000000                1000
    ACT 0000000000000002  0000000000000000                0000
    EXP 0000000000000002  0000000000000000
24   ADDX  UM=0 TE=0  TRAP EXP=0 ACT=0
  4 IN  0000000000000004  0000000000000000                1000
    ACT 0000000000000004  0000000000000000                0000
    EXP 0000000000000004  0000000000000000
24   ADDX  UM=0 TE=0  TRAP EXP=0 ACT=0
 ENTER (N)EXT, (E)ND DISPLAY OR DECIMAL INDEX
```

Line 0 — Test name, etc.

1 — Total number of errors and number of passes run.

2 — Types of registers used in the operation in error (will be test dependent).

3,7,11,15 — Error index into table; input register values for registers used.

4,8,12,16 — Actual register results; actual UCR result; P or P+ type of interrupt (P for P type; for P+ type, a byte offset from P – ex. 02, 04. Blank if no interrupt occurred.)

5,9,13,17 — Expected register values; expected UCR value; expected P or P+ type of interrupt (P for P type and P+ for P+ type. Blank if no interrupt expected.)

6,10,14,18 — opcode and mnemonic of operation in error; setting of UM and TE (zero = clear, 1 = set); whether trap was expected (0 = no, 1 = yes); whether trap occurred (0 = no,1 = yes).

19 — N = Next screen of errors
E = exit error display
Decimal index will display error associated with that index.

In this example, the UCR actual result did not match the UCR expected result.

NOTE

The error display is test dependent and displays the necessary information based on the types of instructions run. For example, DOBL only displays 2 errors per screen due to more information necessary, and HIMM and FIMM display XJ/J and Q fields instead of XJ and X1. BIMM, NUMR, BYTE, and EDIT display BDP instructions and results, with only one error per screen.

Full Test Mode Result Display

```
FULL WORD INTEGER   PA=XXXXXX  XXXX PAGES  CPU X  X.X  YY/MM/DD
NO MIS-MATCHES   ADDF   UM SET   TRAPS ENABLED


ANOTHER TEST (Y/N)
```

In a FULL TEST, one operation is run using a selected setting of TE and UM. All operands are run. In this example, there were no errors. Line 1 indicates that the operation ADDF was run with User Mask set and traps enabled. If there were errors, the test control would go directly into the error display shown on the previous page.

Single Test Mode Results Display

```
FULL WORD INTEGER   PA=XXXXXX  XXXX PAGES  CPU X  X.X  YY/MM/DD
  1 MIS-MATCH    CMPX  UM CLEAR  TRAPS ENABLED
                XK                XJ              X1       UCR P
    IN  0123456789ABCDEF   0FFFFFFFFFFFFFFF
    ACT 0123456789ABCDEF   FFFFFFFFFFFFFFFF   40000000  0000
    EXP 0123456789ABCDEF   0FFFFFFFFFFFFFFF   40000000  0000
 2D  CMPX  UM=0  TE=1  TRAP EXP=0  ACT=0

ENTER (R)EPEAT, (N)EW TEST, OR (T)ERMINATE
```

In a SINGLE TEST, one operation is run using a selected setting of TE and UM. The operator also selects the operands to run either from the results table of a previous run, the preset operand table, or by directly inputting the values to the keyboard.

In this example, the Xj final result was in error. The test can either be restarted ('N' or 'T') or the single test condition can be repeated.

Operator Entries

Test Commands

Y/N Entries

If these tests are run with stop-for-parameter option, the tests present a series of messages/questions which step the operator through the test mode selected (See chapter 24 for a description of the test modes offered and Test Communication for a description of the messages/questions). For ease of use, the tests only require two types of responses. For questions (ex: USER MASK SET Y/N), a 'Y' is interpreted as a YES, and all other replies, including a carriage return, are interpreted as a NO. For messages (ex: ENTER STARTING PASS COUNT), a value is entered in the keyboard. All messages that request values for input also display the base if other than decimal is required.

When entering the operation to be tested, the required input is displayed with parentheses around the shortest form necessary (ex: (A)DDXQ, (M)ULXQ, (I)NCX, (D)ECX). In this case, entering an 'A' would select ADDXQ, as would an 'AD', 'ADD', 'ADDX', or 'ADDXQ'. The parentheses simply indicate the characters necessary to determine the operation selected.

R, T, S, C, D Entries

Once the tests are running, the operator can input the following commands:

Command	Action Taken
R	Restarts the test via a branch back to the beginning of the test code.
T	Terminates the test, which causes an exchange to VEXC, a resetting of monitor's history files, and recreation of the testing environment based on PW12.
S	Stops the test
C	Continues the test after an 'S' has been input
D	Displays the error result table if any errors have occurred.

Test Communication

Test Control Messages

The following is a list of test messages/questions in the order of appearance and comments. These will display if bit 49 of PW0 is set.

Message	Comment
ARE TRAPS IMPLEMENTED ON THIS MACHINE (Y/N)	The Call/Return/Trap hardware will allow trapping to occur. If not working, tests will not expect a trap on normal trap conditions.
IMMEDIATE STOP ON ERROR DISABLED (Y/N)	If NO, test will stop on each error and display error information. If YES, test will complete and accumulate all errors for that final display.

Message	Comment
LOG ERRORS DISABLED (Y/N)	If NO, test will log all errors in the dayfile at the end of a pass. If YES, test will not log errors. See Test Communication.
RUN WITH SMALL PIT (Y/N)	If YES, test will run with a selected value in the Process Interval Timer (PIT) thereby creating additional stress on the processor. If NO response sets PIT to -1 and the next question is bypassed.
ENTER PIT VALUE	Operator can select a PIT setting (decimal). The PIT is a 32 bit microsecond counter – default value is 100 microseconds. A carriage return uses default value.
RUN WITH SMALL SIT (Y/N)	Same as PIT but for the System Interval Timer.
ENTER SIT VALUE	Same as PIT but the default is 150 microseconds.
ENTER STARTING PASS COUNT	Operator can enter a pass count at which to start executing. A carriage return defaults to pass count 1. This is useful if the test has failed at pass XX and is restarted to begin directly at failing pass count XX.
RUN ALL TESTS (Y/N)	If YES, then test will go into the RUN ALL TESTS mode. Otherwise test will skip next four questions.
RUN ALL OPERATIONS (OPCODES) (Y/N)	Operator can run the ALL TESTS mode but can deselect opcodes if answer is NO.
ENTER OPERATIONS YOU DO NOT WANT TO RUN	Operator can deselect operations. Operations are test dependent and are displayed on the screen to indicate necessary input.
RUN SELECTED -J- VALUES (Y/N)	This question only appears in BIMM. Operator can select specific values for the j field of the operations tested.
RUN SELECTED DATA TYPES (Y/N)	This question only appears in BIMM/NUMR/EDIT. Operator can select specific data types to be used by the operations tested.
RUN RANDOM OPERANDS ONLY (Y/N)	Test will use only the randomly generated operands and not the set of canned end case operands.

Message	Comment
ENTER NUMBER OF PASSES TO RUN	Enter in decimal. Default is one pass. If multiple passes selected and an error occurs, test will stop at end of pass in error.
ENTER OPERATION TO BE TESTED	When running in FULL TEST mode or SINGLE TEST mode, operator selects operation to run. Operations are test dependent and displayed on the screen to indicate necessary input.
USER MASK SET (Y/N)	User mask set (Y) = FF7F clear (N) = FE00
TRAPS ENABLED (Y/N)	Test will execute with traps enabled or disabled depending upon reply.
RUN TEST WITH ALL OPERANDS-FULL TEST (Y/N)	If YES, a FULL TEST will run one operation using all operands. If NO is entered, a SINGLE TEST will run one operation with one pair of operands and request the operands with the following messages.
INDEXING RESULT TABLE (Y/N)	This will be displayed if there are entries in the Results Table. If YES, operator can rerun an operation from the results table.
INDEXING OPERAND TABLE (Y/N)	Operator can select operands from the preset operand table.
DISPLAY TABLE (Y/N)	If YES, then test will display either the preset operand table or the results table, depending upon which was selected.
ENTER (N)EXT, (E)ND DISPLAY OR DECIMAL INDEX	This is displayed with the operand table. 'N' prints next screen of operands, 'E' exits display of table, and decimal index will display starting from the operand associated with the decimal number entered. Operands are numbered 1-100(10).
ENTER INDICES (XK,XJ)	Operator enters indices from operand table. Test will use operands associated with indices entered. (Number of operands is test dependent)
ENTER INDEX	Test is requesting an index for the results table. Test will then use all information associated with selected index, and will skip the next question.

Message	Comment
ENTER OPERANDS (XK,XJ(HEX))	Test is requesting input for the operation fields if not selecting from the operand or results table. The fields will be test dependent. (16 digits per operand allowed Test will left justify floating point numbers and right justify integers.
SCOPE LOOP (Y/N)	In SINGLE TEST mode, operator can enter scope loop of opcode/operand selected. The address to clear to exit scope mode will be displayed upon entering the loop.
DISPLAY RESULTS (Y/N)	Operator can display error results at the end of the RUN ALL TESTS mode.
ENTER (N)EXT, (E)ND DISPLAY OR DECIMAL INDEX	This line is displayed with the results table. 'N' will display the next screen of errors, 'E' will exit the display, and decimal index will display the error associated with the index selected.
ENTER (R)EPEAT, (N)EW TEST OR (T)ERMINATE	This line is displayed after a SINGLE TEST mode run. 'R' will repeat the condition, 'N' will return to the beginning of the main test code, and 'T' will exchange to monitor for complete reinitialization.
ANOTHER TEST (Y/N)	This line is displayed at the end of the RUN ALL TESTS mode and the FULL TEST mode. If YES, the test returns to the beginning. If NO, the test exchanges to monitor for complete reinitialization.
TEST STOPPED, ENTER C TO CONTINUE	If test has been stopped with an 'S' command, this message will be displayed.
ALL ERRORS NOW BEING DISPLAYED, ENTER ANY KEY TO CONTINUE MESSAGE	If errors have been displayed during execution, and total errors are less than one screenful, this message will be displayed.
CONTINUE TO NEXT ERROR (Y/N)	If stop-on-error is enabled and an error occurs this message will appear. A YES will continue to the next error, and a NO will allow the options of repeating the error, starting a new test or terminating the test.
ERROR DID NOT REPEAT	If stop-on-error is enabled and an error condition has been repeated but the error did not recurr, this message will appear.
ERROR RECURRED	If stop-on-error is enabled and an error condition has been repeated and the error did recurr, this message will appear.

Informative Messages

The following is a list of informative messages which appear on the screen during execution.

Message	Comment
RUNNING ALL OPERATIONS	Will be displayed on line 2 if running ALL TESTS mode.
RUNNING XXXX,XXXX,XXXX,XXXX	Will be displayed on line 2 if only some of the operations are selected. The selected operations will be displayed in mnemonic form.
RUNNING ALL DATA TYPE	Will be displayed on line 4 if running all data types. (For BIMM, NUMR, AND EDIT tests only)
RUNNING DATA TYPES XX,XX,XX	Will be displayed on line 4 if only some of the data types are selected. The types will be displayed. (For BIMM, NUMR, and EDIT tests only)
WRITING ERRORS TO DAYFILE	Will be displayed on line 19 when errors have occurred and Log Errors is enabled. If a large number of errors have occurred, this task may take several minutes to complete and the test will appear to be hung.

Operator Error Messages

The following is a list of error messages associated with operator input. They will appear on line 19 only if the tests are in stop-for-parameter mode and an operator error occurs.

Message	Comment
CANNOT DISABLE ALL OPERATIONS	Operator cannot deselect all operations. At least one operation must be left selected.
UNRECOGNIZABLE OPERATION – RE-ENTER	Test does not recognize input for selecting or deselecting operations. Input must be in the form displayed on the screen.
INDEX ZERO, RE-ENTER	All indexes to the tables start at one.
INDEX TOO LARGE, RE-ENTER	This message will occur if operator selects an Operand Table index or a Results Table index greater than 100. Although only the first 100 errors are saved, the total number of errors is display on the screen.
NO INPUT RECEIVED – RE-ENTER	Test is expecting some form of input and received a carriage return or a space bar.

Message	Comment
ILLEGAL DATA TYPES - RE-ENTER DATA	Test does not recognize input for data types. This message only occurs in BIMM/NUMR/EDIT.
INVALID -J- TYPE - RE-ENTER	Test does not recognize input for j field. This message only occurs in BIMM.
NO ERRORS RECORDED	This message will be displayed if operator types in a 'D' command during test execution, and there are no errors to display.

Test Error Messages

The following is a list of error messages not covered under normal tests error messages:

Message	Comment
PIT GT 0 XXXX OCCURRENCES	This message indicates that the PIT has fired before it decremented to zero. The number of times this occurs is displayed.
NUMBER OF PIT LOSSES = XXXX	This message indicates that the PIT has decremented to zero and not initiated a PIT interrupt.
NUMBER OF SIT LOSSES = XXXX	This message indicates that the SIT has decremented to zero and not initiated a SIT interrupt.

Normal Messages

See Displays and Test Communications.

Error Messages

See Results Displays.

Recovery

An unidentified interrupt will cause these tests to exchange to monitor (VEXC) and display a message which describes the error. The operator can either scope loop this interrupt via a loop offered at the time, or the test can be restarted by declining the scope loop message. The tests can also be restarted by rerunning the command buffer.

The system will not abort the tests.

Applications

Scope Mode

Scope mode instructions are in Test Communication.

Tracking Exchanges and Traps

VEXC monitor keeps a number of informational buffers that are useful in tracking exchanges and traps. The addresses and format of these buffers are displayed by VEXC when bit 36 of PW0 is clear. By clearing this bit with a CMSE EC or EB command before the test is restarted by issuing a (T)ERMINATE command, or by clearing it in the command buffer and rerunning the command buffer, this display can be viewed. These tables begin at byte address 25000(16) for CPU 0 and 26000(16) for CPU 1. Also see VEXC/TRPUTL ERS (ARH7239) for the description and formats of these buffer.

Quick Look (NUMR,EDIT)

If quick look is selected by setting bit 52 of parameter word 0, a subset of opcodes and type codes are run for each test;

Test	Opcodes	Type Codes
NUMR	ADDN,MULN,DIVN MOVN,SCLR	1,5,6,A
EDIT	EDIT	1,5,6,A

Exceptions - RNGT and CRPT

RNGT and CRPT conform to this document in accordance with Requirments and Operational Procedure. See also Test Error Messages, Error Messages, and Tracking Exchanges and Traps. The following information describes how these tests differ in test structure and display formats.

RNGT

This test is designed to verify access control and security by exercising ring bracket protection mechanisms and key/lock protection mechanisms in conjunction with segment access attributes. The test is divided into conditions as described in the initial display (displayed if bit 49 of PW0 is set):

```
RING/KEY/LOCK TEST                                    CPU X

THIS TEST IS DESIGNED TO VERIFY ACCESS CONTROL AND SECURITY
BY EXERCISING RING BRACKET PROTECTION MECHANISMS AND KEY/LOCK
PROTECTION MECHANISMS IN CONJUNCTION WITH SEGMENT ACCESS
ATTRIBUTES.

CONDITIONS 1-15  TEST RING BRACKET PROTECTION MECHANISMS
CONDITIONS 16-22 TEST LOCAL KEY/LOCK ACCESS CONTROL WITHIN
                 ONE RING (15)
CONDITION  23    TEST LOCAL KEY/LOCK ACCESS CONTROL ACROSS
                 RING BOUNDARIES
CONDITIONS 24-25 TEST GLOBAL KEY/LOCK ACCESS CONTROL WITHIN
                 ONE RING (14) (IF IMPLEMENTED)
CONDITIONS 26-27 TEST GLOBAL KEY/LOCK ACCESS CONTROL ACROSS
                 RING BOUNDARIES (IF IMPLEMENTED)
IF AN ERROR OCCURS, ALL NECESSARY INFORMATION WILL BE
DISPLAYED.  TO ENTER SCOPE LOOP, TYPE (Y).  ANY OTHER ENTRY
WILL CONTINUE TEST EXECUTION.
```

The running display is as follows:

```
RNGT  CC  PC XXXXX  CD XX  EC XXXX  PA XXXXXX     MM/DD/YY
                                    PIT=XXXXXXXX, SIT=XXXXXXXX
```

where:

C	RU for running SE for stopped on error
PC	Pass count (decimal)
CD	Condition number (decimal)
EC	Error count (decimal)
PA	Parameter byte address (Hex)
PIT/SIT	Values being used in the PIT/SIT timers (decimal)

Below is a short description of each condition and expected error condition (Expected MCR = 0200 for all error conditions):

Condition	Description	Expected Error
1	RN.15 call RN.6,6,15	
	Call RN.6,8,9–write RN.7,7	None
	Call RN.6,8,9–write RN.8,8	None
	Call RN.6,8,9 on behalf of RN.15–write RN.6,8	Out of Call bracket
2	RN.15 call RN.7,7,15	
	Call RN.6,8,9–write RN.6,6	Out of Write bracket
	Call RN.6,8,9–write RN.8,8	None
	Call RN.6,8,9 on behalf of RN.15–write RN.6,8	Out of Call bracket
3	RN.15 call RN.8,8,15	
	Call RN.6,8,9–write RN.6,6	Out of Write bracket
	Call RN.6,8,9–write RN.7,7	Out of Write bracket
	Call RN.6,8,9 on behalf of RN.15–write RN.6,8	Out of Call bracket
4	RN.15 call RN.6,6,15–write RN.6,8	None
5	RN.15 call RN.7,7,15–write RN.6,8	Out of Write bracket
6	RN.15 call RN.8,8,15–read RN.6,8	None
7	RN.15 call RN.6,8,9	Out of Call bracket
8	RN.15 call RN.6,7,15	
	Read RN.6,6 on behalf of RN.15	Out of Read bracket
9	RN.15 call RN.6,7,15	
	Read RN.15,15 on behalf of RN.15	None
10	RN.15 call RN.6,7,15	
	Read RN.5,5 on behalf of RN.15	Out of read bracket
11	RN.15 call RN.6,7,15	
	Read write-only segment	No read privilege
12	RN.15 write read-only segment	No write privilege
13	RN.15 write write-only segment	None
14	RN.15 read write-only segment	No read privilege
15	RN.15 call read-only segment	No execute privilege
— Local key/lock verification with one ring (15) —		
— Main segment has master (0) local key; executes in RN.15		
16	Write segment with K/L = 9 on write	None
17	Read segment with K/L = 9 on write	None
	Change main segment local K/L = 9	
18	Write segment with K/L = 9 on write	None
19	Read segment with K/L = 9 on write	None
20	Write segment with K/L = 2 on read	None
21	Read sgement with K/L = 2 on read	Local Key .NE. Lock
22	RN.15 call RN.15 with Local K/L = 2	
	Check proper key transformation	None
	Write seg with K/L = 2 on read	None
	Read seg with K/L = 2 on read	None
	Write seg with K/L = 9 on write	Local Key .NE. Lock
	Read seg with K/L = 9 on write	None
— Local key/lock verification across ring boundary —		
23	RN.15 call RN.14 with local K/L = 5	
	Check proper key transformation	None
	Write RN.15 with K/L = 5 on R/W	None
	Read RN.15 with K/L = 5 on R/W	None

Condition	Description	Expected Error
	Write RN.15 with K/L = 2 on read	None
	Read RN.15 with K/L = 2 on read	Local Key .NE. Lock
	Write RN.15 with K/L = 9 on write	Local Key .NE. Lock
	Read RN.15 with K/L = 9 on write	None
	(return to RN.15-check key transformation)	None
— Global key/lock verification within one ring (14) (Tested if implemented)		
— Main segment has master (0) global key; executes in RN.15		
24	RN.15 call RN.14 with global K/L = 12	
	Check proper key transformation	None
	Write seg with K/L = 12 on write	None
	Read seg with K/L = 12 on write	None
	Write seg with K/L = 8 on read	None
	Read seg with K/L = 8 on read	Global Key .NE. Lock
	Call RN.14 with global K/L = 8	Global Key .NE. Lock
	BRDIR to RN.14 with global K/L = 8	Global Key .NE.
	(Return to RN.15-check key transformation)	None
25	RN.15 call RN.14 with global K/L = 8	
	Check proper key transformation	None
	Write seg with K/L = 8 on read	None
	Read seg with K/L = 8 on read	None
	Write seg with K/L = 12 on write	Global Key .NE. Lock
	Read seg with K/L = 12 on write	None
	(Return to RN.15-check key transformation)	None
— Global key/lock verification across ring boundaries (Tested if implemented)		
26	RN.15 call RN.14 with global K/L = 12	
	Check proper key transformation	None
	Write RN.14 with K/L = 12 on R/W	None
	Read RN.14 with K/L = 12 on R/W	None
	Write RN.14 with K/L = 8 on R/W	Global Key .NE. Lock
	Read RN.14 with K/L = 8 on R/W	Global Key .NE. Lock
	RN.14 call RN.13 with master global K/L (0)	
	Carry global key = 12 on call	
	Check proper key transformation	None
	Write RN.14 with K/L = 12 on R/W	None
	Read RN.14 with K/L = 12 on R/W	None
	Write RN.14 with K/L = 8 on R/W	Global Key .NE. Lock
	Read RN.14 with K/L = 8 on R/W	Global Key .NE. Lock
	(Return to RN.14-check key transformation)	None
	(Return to RN.15-check key transformation)	None
27	RN.15 call RN.13 with master (0) global K/L	
	Carry master global K/L on call	
	Write RN.14 with K/L = 12 on R/W	None
	Read RN.14 with K/L = 12 on R/W	None
	Write RN.14 with K/L = 8 on R/W	None
	Read RN.14 with K/L = 8 on R/W	None
	(Return to RN.15-check key transformation)	None

If an error occurs, the following will be displayed:

```
RING/KEY/LOCK TEST                                        CPU X
RNGT  SE   PC    1   CD    18   EC   1   PA 066100    MM/DD/YY
                                        PIT=00000100, SIT=00000150

ERROR - EXPECTED ILLEGAL READ
            EXPECTED                ACTUAL
P       0005E01E00000198      0005E01E00000198
UTP         E01D00000000          101E00000000
CBP     0000000000000000      0000000000000000
MCR                 0200                  0200
DATA    4444444444444444      4444444444444444
AJ          E01D00000000
SDE     86FF010042000000


NUMBER OF PIT LOSSES = 00000
NUMBER OF SIT LOSSED = 00000

SCOPE LOOP  Y/N
```

where:

CPU X	CPU identifier (0 or 1)
P	P register value when interrupt occurred
UTP	value in Untranslatable Pointer Register
CBP	Code Base Pointer on call (when applicable)
MCR	Value in Monitor Condition Register
DATA	Data read or written during error condition (when applicable)
AJ	Value in the AJ register
SDE	Segment Descriptor value for AJ

NUMBER OF PIT/SIT LOSSES will be a count of the number of times the PIT or SIT timers decremented to zero without initiating an interrupt.

A Yes response will put the test into a scope loop on the error conditon. A message will be printed giving the address of the word to clear to exit the scope loop. If a No response is entered, the test will continue execution.

CRPT

This test is designed to verify all end cases involved in CALL/RETURN/POP instructions. The test is divided into sections and conditions as described in the initial display (displayed if bit 49 of PW0 is set):

```
CALL/RETURN/POP TEST                              CPU X      MM/DD/YY

THIS TEST IS DESIGNED TO VERIFY ALL END CASES INVOLVED IN
CALL/RETURN/POP INSTRUCTIONS.

SECTIONS     1-5   TEST END CASES FOR CALLREL
SECTIONS     6-11  TEST END CASES FOR CALLSEG
SECTIONS    12-18  TEST END CASES FOR RETURN
SECTIONS    19-25  TEST END CASES FOR POP

IF AN ERROR OCCURS, ALL NECESSARY INFORMATION WILL BE
DISPLAYED.  TO ENTER SCOPE LOOP, TYPE (Y).  ANY OTHER ENTRY
WILL CONTINUE TEST EXECUTION.
```

The running display is as follows:

```
CRPT  CC  PC XXXXX  SEC XXX  CD XXX   EC XXXX   PA XXXXXX
                                    PIT=XXXXXXXX, SIT=XXXXXXXX
```

where:

CC	=	RU for running SE for stopped on error
PC	=	Pass count (decimal)
SEC	=	Section number (decimal)
CD	=	Condition number (decimal)
EC	=	Error count (decimal)
PA	=	Parameter byte address (Hex)
PIT/SIT	=	Values being used in the PIT/SIT timers (decimal)

Below is a short description of each condition and expected MCR/UCR interrupt settings:

Section	Cond.	Description	Expected MCR/UCR
		— CALLREL INSTRUCTIONS —	
1	1	Bits 56-59 of X0 < 2	1000/0000
2	1	Critical Frame Flag set	None
		— CALLREL INSTRUCTIONS —	
3	1	Current Stack Frame not in writable segment	0200/0000
4	1	SFSA invalid segment	0008/0000
	2	A0 invalid segment (> STL)	0008/0000
5	1	A0 invalid PVA (bit 32=1)	0800/0000
	2	A0 + SFSA invalid PVA (bit 32=1)	0800/0000
		— CALLSEG INSTRUCTIONS —	
6	1	Bits 56-59 of X0 < 2	1000/0000
7	1	Initial P.RN < SDE.R1	0004/0000
8	1	CBP VMID mismatch VMCL	0100/0000
9	1	Call with CBP not in Binding Section	0200/0000
	2	Call w/Binding Section not 'ring-readable'	0200/0000
	3	SFSA not in 'writable' segment	0200/0000
10	1	(P) final not reference a valid segment	0008/0000
	2	Current stack frame invalid segment	0008/0000
	3	Aj not valid segment	0008/0000
	4	Binding Section not valid segment	0008/0000
	5	A0 invalid segment (> STL)	0008/0000
11	1	(P) final not 0 mod 8	0800/0000
	2	(P) final has bit 32 set	0800/0000
	3	(Aj) not 0 mod 8	0800/0000
	4	(Aj) not valid PVA (bit 32=1)	0800/0000
	5	A0 invalid PVA (bit 32=1)	0800/0000
	6	A0 + SFSA invalid PVA (bit 32=1)	0800/0000
		— RETURN INSTRUCTION —	
12	1	(A2) addresses 'non-readable' segment	0200/0000
13	1	Final P.RN < initial A2.RN	0004/0000
14	1	Critical Frame Flag set	0000/0400
	2	Critical Frame Flag set (purge SFSA from Cache)	0000/0400
	3	Critical Frame Flag set (ensure SFSA in Cache)	0000/0400
15	1	(A2) addresses invalid segment	0008/0000
	2	((A)) addresses invalid segment	0008/0000
	3	Return with RN=0 in PSFSA	0008/0000
16	1	'AT' field in PSF descriptor < 2	0100/0000
	2	Final (A0) not equal initial (A2)	0100/0000
	3	VMID mismatch with VMCL	0100/0000
	4	Return to non-zero VMID w/out global priv.	0100/0000

Section	Cond.	Description	Expected MCR/UCR
17	1	(P) final invalid PVA (bit 32=1)	0800/0000
	2	(P) final not 0 mod 2	0800/0000
	3	Initial (A2) invalid PVA	0800/0000
	4	Initial (A2) not 0 mod 8	0800/0000
18	1	Check RIPPLE	None
	2	Check RIPPLE	None
	3	Check RIPPLE	None
		— POP INSTRUCTION —	
19	1	Initial (A2) SDE invalid	0008/0000
	2	POP with RN=0 in PSFSA	0008/0000
20	1	Initial (A2) addresses 'non-readable' segment	0200/0000
21	1	Initial (A2) .ne. word 1 of PSFSA	0100/0000
22	1	Initial (A2) invalid PVA (bit 32=1)	0800/0000
	2	Initial (A2) not 0 mod 8	0800/0000
23	1	Inter-ring POP	0000/0800
24	1	Critical Frame Flag set	0000/0400
25	1	Check POP stack frame	None

If an error occurs, the following will be displayed:

```
CALL/RETURN/POP TEST                          CPU X       MM/DD/YY
CRPT  SE   PC  1   SEC  19   CD   1   EC   1    PA  066100
                                        PIT=00000100, SIT=00000150

ERROR - EXPECTED ILLEGAL POP
              EXPECTED                ACTUAL
P        0000F00A00001DFA        0000F00A00001DFA
UTP           F03F00000000            F03F00000000
CBP      0000000000000000        0000000000000000
MCR                  0008                    0008
UCR                  0000                    0000
AJ           000000000000
SDE      0AFF0B0000000000
INSTRUCTION AT P   06008271

NUMBER OF PIT LOSSES = 00000
NUMBER OF SIT LOSSED = 00000

SCOPE LOOP  Y/N
```

where:

CPU X	CPU identifier (0 or 1)
P	P register value when interrupt occurred
UTP	value in Untranslatable Pointer Register
CBP	Code Base Pointer on call (when applicable)
MCR	Value in Monitor Condition Register
UCR	Value in User Condition Register
AJ	Value in the AJ register
SDE	Segment Descriptor value for AJ

INSTRUCTION AT P is the failing instruction where the interrupt occurred.

NUMBER OF PIT/SIT LOSSES will be a count of the number of times the PIT or SIT timers decremented to zero without initiating an interrupt.

A YES response will put the test into a scope loop on the error conditon. A message will be printed giving the address of the word to clear to exit the scope loop. If a NO response is entered, the test will continue execution.

Central Memory Test - CMEM 13

Program Description 13-1
General 13-1
Section Descriptions 13-1
Section 1 (SECT1) 13-1
Section 2 (SECT2) 13-1
Section 3 (SECT3) 13-1
Section 4 (SECT4) 13-2
Section 5 (SECT5) 13-2
Section 6 (SECT 6) 13-2
Section 7 (SECT7) 13-2
Section 8 (SECT8) 13-3
Section 9 (SECT9) 13-3
Section 10 (SECTA) 13-3
Section 11 (SECTB) 13-3

Central Memory Test - CMEM 13

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:  01010101-010101012
          555555555555555516
```

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:  01010101-010101012
          AAAAAAAAAAAAAAAA16
```

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:  11011111-110111112
          DFDFDFDFDFDFDFDF16
```

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-FF_{16}). The memory test area may be restricted by using the ADDR, p1, p2 command. p1 is FWA and p2 is LWA. 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 3FDB16 equals 00003FDB-00003FDB16
```

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. 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 3FDB16 equals FFFFC024-FFFFC02416
```

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. 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. 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.

Scope loop does not apply to this section.

Random Command Test - RCT1 14

Program Description 14-1
General 14-1
Basic Test Flow 14-1

Random Command Test - RCT1 14

Program Description

General

RCT1 contains test passes and conditions. A test pass is the generation of a random set of instructions that are simulated and executed. The simulated results are compared with the actual results. Conditions within each pass are used to insert different types of purge instructions into the first two parcels of the execution list and to insert PAD instructions (1AFF-ANDX) between each simulated instruction. Purging and padding introduce different timings to the instructions being executed. Options are available to eliminate both purging and padding. Test passes and conditions are repeatable. Conditions can be brought into a scope loop by typing LOOP or SSM. Random instructions are not changed by the repetition of a test pass.

Basic Test Flow

The RCT1 test flow is as follows:

1. Displays operator and parameter messages and processes any keyboard commands received from the operator.

2. Generates a random seed. For each random seed, updates the PASS counter.

3. Generates random instructions, A and X register values, and data in memory based on the random seed generated in step 2.

4. Simulates the random set of instructions using the A and X register values and memory buffer data. Saves A and X register values after simulation and the final memory buffer contents.

5. Packs the random set of instructions into an execution buffer. Inserts the current purge instruction into the first two parcels of the execution buffer.

6. Executes the random set of instructions using the actual machine hardware. Machine execution uses the same A and X register values and random data in the memory buffer. After machine execution, saves the A registers, X registers, and final memory buffer contents.

7. If scope looping is selected by LOOP/SSM commands, the test goes to step 6. If LOOP/SSM is not selected, the test compares the simulated A register, X register, and memory buffer with the final results from the execution of step 6. The test displays miscompares. If the test encountered errors, and the TRIM (TRM) feature is enabled, the test automatically reduces the failing set of instructions to the minimum number needed to cause the failure (usually one). The reduction technique is to replace each instruction with a PAD (1AFF) instruction, one at a time, and still have the error.

8. If stop-on-condition is set, the test stops and indicates end-of-condition stop (SC). If repeat condition is set, the test repeats step 6 and indicates that the test is repeating a condition (RC).

9. Repeats step 6 until all purge combinations have been used (0-5). There are six purge combinations:

   0 – Purge cache, segment maps and page maps this condition
   1 – Don't purge this condition
   2 – Purge cache only this condition
   3 – Don't purge this condition
   4 – Purge segment maps and page maps this condition
   5 – Don't purge this condition

   Parameters are available to disable the insertion of purge instruction combinations (NPURG).

10. Resets the purge condition counter to zero (0). Unpacks the execution buffer into the simulation buffer and inserts one pad instruction between each simulated instruction. Repeats steps 4 through 10 until a PAD count equal to parameter word 17 (default is 10) is reached.

    Parameters are available to disable the insertion of pad instructions (NPAD).

11. If stop-at-end-of-PASS/TEST is selected, the test stops and displays the proper message (ST). If repeat-PASS/TEST is selected, the test resets the pad and purge counters to 0, repeats steps 3 through 11 and displays the proper message (RT). The same random instructions are used during repeat-PASS/TEST.

12. Goes to step 2 to generate a new random number and to begin a new test pass.

Random Command Test - RCT2 15

Program Description 15-1
General 15-1
Section Descriptions 15-1
Sections 0 through 6 - BDP Numeric Instructions 15-1
Section 0 15-2
Section 1 15-2
Section 2 15-2
Section 3 15-3
Section 4 15-3
Section 5 15-3
Section 6 15-4
Sections 7 through 12, Bdp Byte Instructions 15-5
Section 7 15-5
Section 8 15-5
Section 9 15-6
Section 10 15-7
Section 11 15-7
Section 12 15-7
Sections 13 through 15, BDP Immediate Data Instructions 15-8
Section 13 15-8
Section 14 15-9
Section 15 15-9

Random Command Test - RCT2 15

Program Description

General

RCT2 is divided into sections with each section 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 (for example, 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

Sections 0 through 6 - BDP Numeric Instructions

Sections 0 through 6 each test one of several types of BDP numeric instructions. Code common to these seven sections is used to generate the random operands and descriptors. The operands are arranged into sets; one set is associated with each of the 10 instructions that are executed in each section. The common code performs the following functions:

- 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	Set Number
V_OPB	V_OPA	V_MDF0	0
V_OPC	V_OPA	V_MDF1	1
V_OPC	V_OPB	V_MDF2	2
V_OPD	V_OPB	V_MDF3	3
V_OPD	V_OPC	V_MDF4	4
V_OPE	V_OPC	V_MDF5	5
V_OPA	V_OPD	V_MDF6	6
V_OPE	V_OPD	V_MDF7	7
V_OPA	V_OPE	V_MDF8	8
V_OPB	V_OPE	V_MDF9	9

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

Section 0

SC00 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; otherwise, halts and displays:

```
SECT 0 - BDP NUMBERIC SUM - OP 70
```

Section 1

SC01 tests the BDP numeric difference (71) instruction. The section is similar to SC00 except that BDP numeric differences are simulated and executed. The following error message is displayed on error:

```
SECT 2 - BDP NUMERIC DIFFERENCE - OP 71
```

Section 2

SC02 tests the BDP numeric product (72) instruction. The section is similar to SC00 except that BDP numeric products are simulated and executed. The following error message is displayed on error:

```
SECT 2 - BDP NUMERIC DIFFERENCE - OP 72
```

Section 3

SC03 is designed to test the BDP numeric quotient (73) instruction. The section is similar to SC00 except that BDP numeric quotients are simulated and executed. The following error message is displayed on error:

```
SECT 3 - BDP NUMERIC QUOTIENT - OP 73
```

Section 4

SC04 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)	SET 0
Scale operand C (V_OPC) to type A (V_MDF1)	SET 1
Scale operand C (V_OPC) to type B (V_MDF2)	SET 2
Scale operand D (V_OPD) to type B (V_MDF3)	SET 3
Scale operand D (V_OPD) to type C (V_MDF4)	SET 4
Scale round operand E (V_OPE) to type C (V_MDF5)	SET 5
Scale round operand A (V_OPA) to type D (V_MDF6)	SET 6
Scale round operand E (V_OPE) to type D (V_MDF7)	SET 7
Scale round operand A (V_OPA) to type E (V_MDF8)	SET 8
Scale round operand B (V_OPB) to type E (V_MDF9)	SET 9

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

```
SECT 4 - BDP NUMERIC SCALE - OP E4/E5
```

Section 5

SC05 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	SET 0
C (V_OPC) to A (V_MDF1), result to V_MCDR1	SET 1
C (V_OPC) to B (V_MDF2), result to V_MCDR2	SET 2
D (V_OPD) to B (V_MDF3), result to V_MCDR3	SET 3
D (V_OPD) to C (V_MDF4), result to V_MCDR4	SET 4
E (V_OPE) to C (V_MDF5), result to V_MCDR5	SET 5
A (V_OPA) to D (V_MDF6), result to V_MCDR6	SET 6
E (V_OPE) to D (V_MDF7), result to V_MCDR7	SET 7
A (V_OPA) to E (V_MDF8), result to V_MCDR8	SET 8
B (V_OPB) to E (V_MDF9), result to V_MCDR9	SET 9

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

```
SECT 5 - BDP NUMERIC COMPARE - OP 74
```

Section 6

SC06 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 SET	0
operand B (V_OPB) to V_MDF1 SET	1
operand C (V_OPC) to V_MDF2 SET	2
operand D (V_OPD) to V_MDF3 SET	3
operand E (V_OPE) to V_MDF4 SET	4
operand F (V_OPF) to V_MDF5 SET	5
operand G (V_OPG) to V_MDF6 SET	6
operand H (V_OPH) to V_MDF7 SET	7
operand I (V_OPI) to V_MDF8 SET	8
operand J (V_OPJ) to V_MDF9 SET	9

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

```
SECT 6 - BDP NUMERIC MOVE OP 75
```

Sections 7 through 12, Bdp Byte Instructions

Sections 7 through 12 each test one of several types of BDP byte instructions. The sections use either one, three, or six sets of instructions and operands. The code common to these sections fills a 256-byte field V_BUFA with random data from V_RNB.

Section 7

SC07 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	SET 0
V_BUFC, L2 and V_BUFA, L1 to V_MDF0+1	SET 1
V_BUFA, L1 and V_BUFC, L2 to V_MDF0+2	SET 2
V_BUFA, L2 and V_BUFB, L2 to V_MDF0+3	SET 3
V_BUFA, L2 and V_BUFB, L1 to V_MDF0+4	SET 4
V_BUFC, L1 and V_BUFA, L2 to V_MDF0+5	SET 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; otherwise, halts and displays:

```
SECT 7 - BDP BYTE COMPARE - OP 77
```

Section 8

SC08 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	SET 0
V_BUFA, L2 and V_BUFB, L1 to V_SDF0+1	SET 1
V_BUFB, L1 and V_BUFA, L2 to V_SDF0+2	SET 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	SET 0
V_BUFA, L2 and V_BUFB, L1 to V_MDF0+1	SET 1
V_BUFB, L1 and V_BUFA, L2 to V_MDF0+2	SET 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, otherwise, halts and displays:

  ```
  SECT 8 - BDP BYTE COMPARE COLLATED - OP E9
  ```

Section 9

SC09 tests the BDP byte scan (F3) 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)	SET 0
V_BUFE (fixed 32 bytes, ones)	SET 1
V_BUFB (random bytes)	SET 2

- 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	SET 0
V_BUFE result to V_MDF0+1	SET 1
V_BUFB result to V_MDF0+2	SET 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, otherwise, halts and displays:

  ```
  SECT 9 - BDP BYTE SCAN - OP F3
  ```

Section 10

SC10 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, otherwise, halts and displays:

```
SECT 10 - BDP BYTE TRANSLATE - OP EB
```

Section 11

SC11 tests the BDP byte move (76) 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, otherwise, halts and displays:

```
SECT 11 - BDP BYTE MOVE - OP 76
```

Section 12

SC12 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; otherwise, halts and displays:

```
SECT 12 - BDP BYTE EDIT - OP ED
```

Sections 13 through 15, BDP Immediate Data Instructions

Sections 13 through 15 each test one of the BDP immediate data instructions. In each section, sets of ten instructions and operands are employed. The code common to these sections performs the following operations:

- 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

Section 13

SC13 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, otherwise, halts and displays:

```
SECT 13 - BDP IMMEDIATE DATA MOVE - OP F9
```

Section 14

SC14 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 instruc- tions 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; otherwise, halts and displays:

```
SECT 14 - BDP IMMEDIATE DATA COMPARE - OP FA
```

Section 15

SC15 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, otherwise, halts and displays:

```
SECT 15 - BDP IMMEDIATE DATA ADD - OP FB
```

Fixed Operand Command Tests - FCT1,2,3,5 16

FCT1 Test 16-1

FCT2 Test 16-1

FCT3 Test 16-1

FCT5 Test 16-2

Fixed Operand Command Tests - FCT1,2,3,5 16

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 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, chapter 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.

Virtual Mode Instruction Level Test – FCT9 17

Program Description 17-1
General 17-1
Controller Operation 17-1
CPU Operation 17-1
Environment Interface (EI) 17-1
Testing Without EI 17-2
Testing With EI 17-2
Section Descriptions 17-3
Section 0 17-3
Sections 1 and 2 17-4
Sections 3 and 4 17-6
Section 5 17-8
Section 6 17-11
Section 7 17-15
Section 8 17-16
Section 9 17-17
Section 10 17-19
Section 11 17-21
Section 12 17-21

Virtual Mode Instruction Level Test - FCT9 17

Program Description

General

This chapter describes the operation of the test and the testing done within each section. This chapter 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 processor's do not finish by the end of this period, a processor's 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 EIs 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 deadstart, 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.

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. 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

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

Subsection	Description

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.

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.

4 Subsection 4 (section 1 only) checks that a 60-bit negative value read into XJ by the CR (660) command is properly extended. A value of minus one (1's complement) is written to CM by the control PP. The CP reads this into X0 with the CR (660) and then executes a 52 command (SA7 X0+K) to store a pattern in memory. If the sign is not extended, an AOR error should occur since the effective memory address will be very large instead of being K-1.

The following comparisons are made:

1. Checks that the CP halts at the correct address.
2. Checks for the page fault bit in the MCR.
3. Checks that C170 register A7 points to K-1.
4. Checks that a few words below K-1 have not been overwritten.
5. Checks for the correct pattern at K-1.
6. Checks that the -1 at location K, which is read by the CR command, has not been overwritten.

Sections 3 and 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. Ten conditions are executed with a 21-bit FLE (15 MSB's true) and then the same conditions are repeated with a 24-bit FLE (18 MSB's true).

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)= 373,777
9	(Xk)=7,777,677

Subsection	Description
	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. 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 undefined fields immediately to the left of RAE and FLE in the 180 exchange package (C180 bits 20-31) are set to nonzero values. The same also applies to the C170 bits 30-59 in register XK which contains the UEM address. In all three cases these fields should be ignored by the hardware. The patterns used are:

Condition	
0	data = octal 2525252525252525
1	data = octal 5252525252525252
2	data = octal 7070707070707070
3	data = octal 0707070707070707

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.

4 Subsection 4 (section 3 only) checks that a 60-bit negative value read into XJ by the RX (014) command is properly extended. For further information refer to section 1, subsection 4 which is identical except for the command tested.

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.

The first 50 conditions are executed with a 21 bit FLE (15 MSB's true). The remaining conditions use a 24-bit FLE (18 MSB's true).

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.

Subsection	Description

3 Tests the transmission of data patterns by the REC instruction. The patterns used are:

Condition	
0	data = octal 25252525252525252525
1	data = octal 52525252525252525252
2	data = octal 70707070707070707070
3	data = octal 07070707070707070707

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[1]
1	0	1	1
2	–1	2	1

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

1. This condition tested in previous subsection.

Subsection	Description

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.

Subsection	Description
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 (that is, 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.

Subsection	Description

The first 50 conditions are executed with a 21-bit FLE (15 MSB's true). The remaining conditions use a 24-bit FLE (18 MSB's true).

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:

Condition	
0	data = octal 25252525252525252525
1	data = octal 52525252525252525252
2	data = octal 70707070707070707070
3	data = octal 07070707070707070707

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	1^2
1	0	1	1
2	-1	2	1

2. This condition tested in previous subsection.

Subsection	Description

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 (that is, 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.

Subsection	Description

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, X0 upper instead of A0 is used for the block transfer from 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 7

This section tests the error exits of the CRXj and CWXj instructions. Subsection 0 sets up processor registers and common data.

Subsection **Description**

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 indicates 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 indicates 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.
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.

Subsection 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 indicates 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.

Subsection 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 indicates 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

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.

Subsection	Description
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 indicates 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	USM 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

Subsection	Description
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.
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.

Subsection	Description
1	Tests for a Privileged Instruction Fault interrupt when a RT instruction is encountered.

Condition	Operation Tested	Traps	Expected 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

Subsection	Description

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.

Exchange Test - EXCH 18

Program Description 18-1
General 18-1
Structure 18-1
CP Executable Code 18-1
Exchange Packages 18-1
Section and Subsection Address Tables 18-2
Control Command Tables 18-2
Execution Sequence 18-2
Setting Up The Test Condition 18-2
Exchange Sequence 18-3
Data Checking 18-3
Repeat Condition and Scope Loops 18-3
Slave PP Control 18-4
Slave Triggering Via PP Channels 18-4
Communication Via Central Memory 18-4
Synchronized Versus Free-Running Control 18-5

Section Descriptions 18-6
CYBER 170 Program Stop Sequence 18-6
CYBER 170 X Register Shift Sequence 18-7
Section 0 - Test Initialization (SECT 0) 18-7
Section 1 - CEJ/MEJ Commands (SECT 1) 18-8
Section 2 - Exchange Addressing (SECT 2) 18-10
Section 3 - 26XX Execution In CYBER 170 Job And Monitor Mode (SECT 3) . 18-11
Section 4 - 26XX Sets MCR 5 In Executive State Job Mode (SECT 4) 18-13
Section 5 - 26XX Execution In Executive State Job Mode (SECT 5) 18-13
Section 6 - 26XX Execution In Executive State Monitor Mode (SECT 6) 18-15
Section 7 - Multiple 2600 Exchanges and B0 Checking (SECT 7) 18-17
Section 8 - 26XX Exchanges and Block Writes In CYBER 170 Mode (SECT 8) 18-19
Subsection 0 18-19
Subsection 1 18-20
Subsection 2 18-20
Subsection 3 18-20
Subsection 4 18-20
Section 9 - 2600 Exchanges and CP Stop/Start In CYBER 170 Mode (SECT 9) 18-21
Section 10 - 2600 Exchanges and External Interrupts (SECT 10) 18-24
Section 11 - 2600 Exchanges and SIT/PIT Interrupts (SECT 11) 18-26
Section 12 - Error Exit, Illegal Instructions (SECT 12) 18-29
Section 13 - Error Exit, Address Out of Range (SECT 13) 18-32
Section 14 - Error Exit, Floating-Point Infinite (SECT 14) 18-35
Subsections 1 to 4 - Infinite error, CYBER 170 job mode, exit selected 18-35
Subsection 2 Conditions 18-35
Subsection 3 Conditions 18-36
Subsection 4 Conditions 18-36
Subsection 5 18-36
Subsection 6 18-36
Subsection 7 18-36
Section 15 - Error Exit, Floating-Point Indefinite (SECT 15) 18-38
Subsections 1 to 5 - Indefinite Error, CYBER 170 Job Mode Exit Selected .. 18-38
Subsection 1 Conditions 18-38
Subsection 2 Conditions 18-39
Subsection 3 Conditions 18-39
Subsection 4 Conditions 18-40

Subsection 5 18-40
Subsection 7 18-40
Subsection 8 18-40
Section 16 – Exchange Addressing, Selectable Addresses (SECT 16) 18-42
Section 17 – OS Bounds Register Test (SECT 17) 18-43
Subsection 1 – Comparison Checks, Even Numbered Conditions 18-43
Subsection 1 – Comparison Checks, Odd Numbered Conditions 18-44
Subsection 2 18-44
Subsection 3 18-44
Section 18 – State Switch With X0 Sign Bit (SECT 18) 18-44
Section 19 – A170 UEM Read/write and Block Copy Commands in ESM Mode . 18-45
Subsection 0 18-45
Subsection 1 18-45
Subsection 2 18-45
Subsection 3 18-46
Subsection 4 18-46
Subsection 5 18-46
Subsection 6 18-46
Subsection 7 18-46
Subsection 8 18-46
Subsection 9 18-46

Exchange Test - EXCH 18

Program Description

General

This chapter 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 elements described in the following paragraphs.

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 various memory locations may be performed. Frequently, the exchange count read back to CM 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 CM 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 CM. 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 CM. 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 CM 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 CM. 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.

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

CMP.02 Checks that the PTL register was loaded properly

CMP.03 Checks that the PTA register was loaded properly

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:

Condition 0 - Execute Program Stop in CYBER 170 monitor mode
Condition 1 - Execute Program Stop in CYBER 170 job mode
Condition 2 - Execute CEJ command in CYBER 170 monitor mode
Condition 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.

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.

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 command 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.

Subsection 2 provides initialization for subsections 3 and 4.

Subsection 3 checks the C170 registers in the C180 job package after the execution of a C170 program stop command in C170 monitor mode. A CYBER 180 exchange command (op code 02) starts execution at the stop command which causes an immediate exchange interrupt back to the C180 monitor. Except for the exit mode halt flag, all C170 registers and flags should contain their initial values (except the C170 B0 register).

Fourteen conditions are executed which are all identical except for the initial pattern of bits which are set up in the 12-bit field containing the C170 exit mode and flag bits.

Subsection 4 is similar to subsection 3 except that fields in words 5 through 24 of the C180 job package which must be ignored (during load) by the C170 process are set to nonzero values. These same fields are undefined during store and are therefore ignored by the test during comparison checks.

CMP.01 Checks the value of the P register in the C170 exchange to MA package.

CMP.02 Checks the P register value in RA.

CMP.03 Checks the P register in the C180 job exchange package.

CMP.04 Checks the P register in the register file dump.

The following comparisons all perform checks on the C180 job exchange package:

CMP.05 Exit mode, flags, and RAC in word 4.

CMP.06 MF, FLC, EMHF, and MA in words 5 and 6.

CMP.07 RAE and FLE in words 7 and 8.

CMP.08 C170 A0 through A7 in words 9 through 16.

CMP.09 C170 B1 through B7 in words 18 through 24.

CMP.10 C170 X0 through X7 in words 25 through 32.

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 2^3 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 third 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 an 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:

Condition 0 - 2600, CYBER 170 Job mode, Swap Per A, MF No Change
Condition 1 - 2610, CYBER 170 Job mode, Swap Per A, MF Is Set
Condition 2 - 2620, CYBER 170 Job mode, Swap Per MA, MF Is Set
Condition 3 - 2600, CYBER 170 Mon mode, Swap Per A, MF No Change
Condition 4 - 2610, CYBER 170 Mon mode, No Swap, MF No Change
Condition 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 countdown 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:

Condition 0 - 2600, Response in CYBER 170 Job, Swap Per A, MF No Change
Condition 1 - 2610, Response in CYBER 170 Job, Swap Per A, MF Is Set
Condition 2 - 2620, Response in CYBER 170 Job, Swap Per MA, MF Is Set
Condition 3 - 2600, Response in CYBER 170 Mon, Swap Per A, MF No Change
Condition 4 - 2610, Response in CYBER 170 Mon, No Swap, MF No Change
Condition 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:

1. After the initial deadstart, the executive state monitor process performs some initialization and then switches to the executive state job mode via an EXCHANGE.

2. 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.

3. 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.

4. 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).

5. 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.

6. 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.

7. For conditions 4 and 5 the program stop referred to in step 5 above causes an immediate executive state exchange to executive state monitor mode.

8. For conditions 0 through 3, either CYBER 170 exchange to A, or exchange to MA package is selected and determines which of two short CYBER 170 programs are executed. In both cases, a unique flag is set 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.

9. For conditions 1 through 3 the PS command will now cause a return to executive state monitor mode.

10. 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. In this case, both the exchange to A and exchange to MA flags will be set up.

11. 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 1 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 4 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 2 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 4 and 5 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 4 and 5 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 10 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:

Condition 0 - 2600, Response in CYBER 170 Job, Swap Per A, MF No Change
Condition 1 - 2610, Response in CYBER 170 Job, Swap Per A, MF Is Set
Condition 2 - 2620, Response in CYBER 170 Job, Swap Per MA, MF Is Set
Condition 3 - 2600, Response in CYBER 170 Mon, Swap Per A, MF No Change
Condition 4 - 2610, Response in CYBER 170 Mon, No Swap, MF No Change
Condition 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:

1. 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.

2. 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.

3. 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.

4. 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).

5. 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.

6. 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.

7. 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.

8. 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.

9. For conditions 1 through 3 the PS command will now cause a return to executive state monitor mode.

10. 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. In this case, both the exchange to A and exchange to MA flags will be set up.

11. 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 1 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 3 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 4 and 5 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 4 and 5 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 10 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. 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. The contents of A0 were saved in X5 after the trap (step 2 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 specifies 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. 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 CM 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 occurrence 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. Both sets of 49 conditions are controlled by 2 tables, each occupying 49 words (one for each condition) of CM. 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:

1. 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.

2. 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.

3. After the EXCHANGE to job, EXCHITC will trigger the slave PP or PPs to commence execution of their 2600 commands. The processor does not exit the executive state job process to perform the swap in response to a PP exchange request.

4. Each slave commanded to execute 2600 commands will perform the following sequence:

   a. Executes the delay defined for itself in the control table entry for the current condition.

   b. Loads into its A register the address of its specific CYBER 170 exchange package.

   c. Performs a CRDL command to read the first word of the exchange package which contains P, A0, and B0.

   d. 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.

   e. Executes a keypoint command and then immediately afterward the 2600 command.

   f. 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.

   g. The B0 portion of the word is examined. Only if the value is zero is the incrementing of a "B0 miss count" skipped.

   h. Each active slave PP returns to substep a and repeats the sequence until the required number of 2600 commands have been executed.

5. 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 CM. 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 CM 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 CM. 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 CM 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:

1. 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.

2. 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.

3. 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.

4. 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. If the scope mode is selected, EXCHITC will deadstart the CP and loop back to step 1 above; otherwise, comparisons will commence as defined below.

CMP.01 Upon completion of each condition, each slave writes into CM 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 CM 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 operation.

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. 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. The 14 conditions are controlled by 2 tables, each occupying 14 words (one for each condition) of CM. 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:

1. After the initial deadstart, the executive state monitor process sets up a CYBER 170 exchange package and an executive state job exchange package using two copy sequences. A sequence complete flag in CM is cleared at this time. At the end of the condition sequence, this flag is 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.

2. 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).

3. 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.

4. EXCHANGE OUT – While the X register shift sequence and PP exchange activity are occurring, the EXCHITC PP will commence an exchange out sequence which is described in the following steps:

   a. A stop function is sent to the CP on the MAC channel.

   b. EXCHITC checks the halt bit in the CP status summary register; if it is set, it continues to substep c. 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.

c. 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.

d. EXCHITC initiates a deadstart to cause a half exchange out which writes the process registers to a executive state exchange package in CM. 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.

e. 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.

5. RELOAD - EXCHITC now commences a reload operation which consists of the following operations:

   a. A deadstart is initiated which causes microcode initialization including setting default values into various registers including some or all of those just saved in PP memory.

   b. A clear errors function is issued to the CP.

   c. A reload sequence is initiated in which the register contents saved in PP memory in step 4e above are now written back into the same registers.

6. EXCHANGE IN - The EXCHITC PP completes the stop/start sequence by executing the exchange in sequence which is as follows:

   a. A master clear function and then a clear errors function are sent to the CP.

   b. 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.

   c. The preserve exchange bit in the DEC register is cleared.

   d. The B0 word in the CYBER 170 exchange package in CM is set to a nonzero value.

   e. 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 4a above, and it should continue execution.

   f. 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 7 below.

   g. 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 6e above).

If this is the case, the sequence returns to step 4 above and the commencement of another stop/start operation. If B0 does not clear, step 6f above and this step are repeated several times before EXCHITC exits to step 7. A failure of the process to restart properly will be detected by errors during comparison checks.

7. 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 1 above; otherwise, comparisons will commence as defined below.

CMP.01 Upon completion of each condition, each slave writes into CM 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 7 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 CM contain all ones.

Section 10 - 2600 Exchanges and External Interrupts (SECT 10)

This section is similar to section 8 except 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 CM. 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:

1. 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.

2. 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.

3. 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. The processor does not exit the executive state job process to perform the swap in response to a PP exchange request.

4. 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.

5. This step only occurs if a trap interrupt occurs as a result of an external interrupt arriving during step 4 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 4 above, another interrupt request may be issued before a RETURN is made to the point of interruption in step 4 above.

6. 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 1 above; otherwise, comparisons begin as defined below.

CMP.01 Upon completion of each condition, each slave writes into CM 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 CM 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 CM. 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 specifies 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 execution 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:

1. 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 CM for the processor. The CP masks the SIT and PIT interrupt counts and saves them in CM. 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.

2. 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.

3. 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.

4. JOB TRAP – If or when a job trap interrupt occurs, a job trap routine performs the following operations:

   a. The MCR bits from the job trap stack frame are checked for a PIT interrupt.

   b. The job process PIT interrupt count in CM is decremented by one.

   c. 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 4e.

   d. If the count was nonzero a CPYXS command sets the PIT to the test interval to generate a subsequent job process PIT interrupt.

   e. A RETURN command switches the processor back to the CYBER 170 X register shift sequence.

5. EXCHANGE INTERRUPT – When an exchange interrupt occurs, the processor returns to the executive state monitor mode where the following steps occur:

   a. The MCR bits in the executive state job exchange package are checked for a SIT interrupt.

   b. The SIT interrupt count in CM is decremented by one.

   c. The SIT bit in the MCR of the job package is cleared.

   d. 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 5f.

   e. A CPYXS command sets the SIT to the test interval to cause a subsequent SIT interrupt.

   f. The processor performs an EXCHANGE to return to the job mode and the CYBER 170 shift sequence.

6. 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 5 above. A monitor process trap routine would perform the following operations:

   a. The MCR bits from the monitor trap stack frame are checked. If all bits are false skip to step 6e.

   b. If the SIT bit is set, the SIT interrupt count in CM is decremented by one.

   c. 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 6f.

   d. The SIT is set up with the SIT interval value to cause a subsequent SIT interrupt. Skip to step 6f.

   e. 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.

   f. The processor executes a RETURN command to switch back to the monitor process described in step 5 above.

7. 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. If the scope mode is selected, EXCHITC will deadstart the CP and loop back to step 1 above; otherwise, comparisons will commence as defined below.

CMP.01 Upon completion of each condition, each slave writes into CM 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 CM; it should have been decremented to zero.

CMP.04 This compare checks the job process PIT interrupt count in CM; 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 CM 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

Cond. No.	Command	Violation
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 CM 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 CM. Only that portion of CM 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 (2^{48}) 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 (2^{48}) 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 (2^{48}) 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 2^{49}) 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 (2^{49}) 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.

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 referred 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.

Subsection 0 is used for initialization only. Subsections 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

Command	Xj +,-	Xk to	Xi	Cond. No.
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

Command	Xj +,-	Xk to	Xi	Cond. No.
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

Command	Xj *,/	Xk to	Xi	Cond. No.
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

Command	Xj *,/	Xk to	Xi	Cond. No.
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

Command	Xj *	Xk to	Xi	Cond. No.
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

Command	Xj /	Xk to	Xi	Cond. No.
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 P to zero after which this package is swapped (P in this package will 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 (2^{50}) 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 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 2^0 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 subsequent 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. 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 segments.

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 PPs). 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 similar 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 countdown 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 similar to the previous except that a 2600 (EXN) command is employed; the hardware should react in the same manner. In order to speed execution, all even numbered conditions are skipped (that is, 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 condition 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 19 - A170 UEM Read/write and Block Copy Commands in ESM Mode

This section tests the single word CM/UEM read and write commands RX (014) and WX (015) and the CM/UEM block copy commands REC (011) and WEC (012). Tests are performed in the ESM mode only. The tests are executed for models 810, 830, 840, 845, 850, 855 and 860 only. All subsections of section 19 are automatically skipped for other models.

The EXCH page table resides at byte address 20,000 (16) and has been modified to provide for one to one mapping of PVAs to RMAs up to 128 megabytes. The PP controller used with test EXCH obtains the CM size from CMSE and ensures that testing is within the appropriate limits. The test assumes that CMSE is resident in CM and therefore never accesses the last megabyte.

Subsection 0

This subsection performs the common initialization required by all subsections of section 19.

Subsection 1

Subsection 1 tests the addressing of central memory and unified extended memory (ESM mode) using the REC instruction. The extended block copy flag is clear and the CM and UEM addresses are contained in A0 and X0 respectivley. The test checks that the block of received data in CM begins and ends at the correct addresses. The reference addresses RAC and RAE are both equal to zero.

Subsection 2

Subsection 2 tests the addressing of central memory and unified extended memory (ESM mode) by the REC instruction. This section is similar to the previous subsection except that the block copy flag is set and the CM and UEM addresses are both contained in X0.

Subsection 3

Subsection 3 tests the addressing of central memory and unified extended memory (ESM mode) by the REC instruction. The block copy flag is clear and the CM and UEM addresses are contained in the 170 registers A0 and X0. An extra loop is added so that for each UEM address tested, the REC command repeats through a total of ten conditions and thus checks various addresses in CM. The length of the transfer is 16 words.

Subsection 4

Subsection 4 tests the addressing of central memeory and unified extended memory (ESM mode) using the WEC instruction. The extended block copy flag is clear and the CM and UEM addresses are contained in A0 and X0 respectively. A block of 32 words is written from CM to UEM and checked. The reference addresses RAC and RAE are both equal to zero.

Subsection 5

Subsection 5 tests the addressing of central memory and unified extended memory (ESM mode) by the WEC instruction. This section is similar to the previous subsection except that the block copy flag is set and the CM and UEM addresses are both contained in X0.

Subsection 6

Subsection 6 tests the addressing of central memory and unified extended memory (ESM mode) by the WEC instruction. The block copy flag is clear and the CM and UEM addresses are contained in the 170 registers A0 and X0. An extra loop is added so that for each UEM address tested the WEC command is repeated via a total of ten conditions to check a variety of addresses in CM. The length of the transfer is 16 words.

Subsection 7

Subsection 7 tests the addressing of central memory and unified extended memory (ESM mode) using the RX instruction. Ten conditions using different data patterns are executed for each different test address. RAE for UEM access is zero. RAC used for CM references is set 4200 (8) word = 4400 (16) byte.

Subsection 8

Subsection 8 tests the addressing of central memory and unified extended memory (ESM mode) using the WX instruction. The subsection is similar to section 7 except that the source and destination are reversed.

Subsection 9

Subsection 9 performs an Address Out of Range check for the address in UEM specified by RAE+FLE. Four conditions are employed for the REC and WEC commands (two with and two without the block copy mode). Two additional conditions check the RX and WX commands. These six conditions first execute with the AOR exit bit set and are repeated with the exit bit cleared.

Trap Test - TRAP 19

Program Description 19-1
General 19-1
Structure 19-1
Central Processor Executable Code 19-1
Exchange Packages 19-1
Section and Subsection Address Tables 19-2
Control Command Tables 19-2
Execution Sequence 19-2

Section Descriptions 19-4
Section Initialization 19-4
SECT 00 - MCR Bit Tests 19-5
SECT 01 - UCR Bit Tests 19-7
MCR/UCR Function Tests 19-9
MCR Function Test Comparisons - Sections 2 to 17 19-11
UCR Function Test Comparisons - Sections 18 to 33 19-11
SECT 02 - MCR 00 - Detected Uncorrectable Error 19-12
SUB02_1 19-12
SUB02_2 19-12
SECT 03 - MCR 01 - Not Used 19-12
SECT 04 - MCR 02 - Short Warning 19-13
SUB04_1 19-13
SUB04_2 19-13
SECT 05 - MCR 03 - Instruction Specification Error 19-14
SUB05_1 19-14
SUB05_2 19-14
SUB05_3 19-14
SUB05_4 19-14
SUB05_5 19-14
SECT 06 - MCR 04 - Address Specification Error 19-14
SUB06_1 19-14
SUB06_2 19-14
SUB06_3 19-15
SUB06_4 19-15
SECT 07 - MCR 05 - Exchange Request 19-15
SECT 08 - MCR 06 - Access Violation 19-15
SUB08_1 19-15
SUB08_2 19-15
SUB08_3 19-15
SUB08_4 19-15
SUB08_5 19-15
SUB08_6 19-16
SECT 09 - MCR 07 - Environment Specification Error 19-16
SUB09_1 19-16
SUB09_2 19-16
SUB09_3 19-16
SUB09_4 19-16
SECT 10 - MCR 08 - External Interrupt 19-16
SUB10_1 19-16
SECT 11 - MCR 09 - Page Table Search 19-17
SUB11_1 19-17
SUB11_2 19-17
SUB11_3 19-17

SUB11_4 19-17
SUB11_5 19-17
SECT 12 - MCR 10 - System Call 19-17
SECT 13 - MCR 11 - System Interval Timer 19-18
SUB13_1 19-18
SECT 14 - MCR 12 - Invalid Segment/Ring=0 19-18
SUB14_1 19-18
SUB14_2 19-18
SUB14_3 19-18
SUB14_4 19-18
SUB14_5 19-18
SECT 15 - MCR 13 - Outward Call/Inward Return 19-19
SUB15_1 19-19
SUB15_2 19-19
SECT 16 - MCR 14 - Soft Error Log 19-19
SUB16_1 19-19
SUB16_2 19-19
SECT 17 - MCR 15 - Trap Exception 19-20
SUB17_1 19-20
SECT 18 - UCR 00 - Priviledged Instruction Fault 19-20
SUB18_1 19-20
SUB18_2 19-20
SUB18_3 19-20
SECT 19 - UCR 01 - Unimplemented Instruction 19-21
SUB19_1 19-21
SUB19_2 19-21
SECT 20 - UCR 02 - Free Flag 19-21
SECT 21 - UCR 03 - Process Interval Timer 19-21
SUB21_1 19-21
SECT 22 - UCR 04 - Inter-Ring POP 19-21
SUB22_1 19-21
SECT 23 - UCR 05 - Critical Frame Flag 19-22
SUB23_1 19-22
SUB23_2 19-22
SECT 24 - UCR 06 - Keypoint 19-22
SUB24_1 19-22
SECT 25 - UCR 07 - Divide Fault 19-22
SUB25_1 19-22
SUB25_2 19-22
SUB25_3 19-23
SECT 26 - UCR 08 - Debug 19-23
SUB26_1 19-23
SECT 27 - UCR 09 - Arithmetic Overflow 19-23
SUB27_1 19-23
SUB27_2 19-24
SECT 28 - UCR 10 - Exponent Overflow 19-24
SUB28_1 19-24
SECT 29 - UCR 11 - Exponent Underflow 19-24
SUB29_1 19-24
SECT 30 - UCR 12 - FP Loss of Significance 19-24
SUB30_1 19-24
SECT 31 - UCR 13 - FP Indefinite 19-25
SUB31_1 19-25
SECT 32 - UCR 14 - Arithmetic Loss of Significance 19-25
SUB32_1 19-25
SUB32_2 19-25

SECT 33 - UCR 15 - Invalid BDP Data 19-25
SUB33_1 19-25
SECT 34 - Multiple Interrupt/Priority Checks 19-26
SUB34_1 19-26
SUB34_2 19-26
SUB34_3 19-27
SECT 35 - MCR Bit Left in Memory 19-27
SUB35_1 19-27
SUB35_2 19-28
SUB35_3 19-28
SUB35_4 19-29
SECT 36 - Reserved 19-29
SECT 37 - Model 810/815/825/830 Processor Model Dependent Tests 19-29
SUB37_1 19-30
SUB37_2 19-30
SECT 38 - Model 835 Processor Model Dependent Tests 19-30
SUB38_1 19-30
SECT 39 19-31
SECT 40 - Model 990 Processor Model Dependent Test 19-31
SECT 41 - Keypoint Function Test 19-31

Trap Test - TRAP 19

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 elements described in the following paragraphs.

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. (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. 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.

Subsection	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	Enabled	All but 1 set	Exchange	Halt or Stack
00_0	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.

Subsection	Mode	Traps	Mask Bits	MCR 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.

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.

Subsection	Mode	Traps	Mask Bits	MCR 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).

All conditions which execute in CYBER 170 mode will be skipped automatically if the processor being tested does not have 170 capability.

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).

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.

1. In all cases the untranslatable pointer (UTP) is checked to ensure it is set to correct value.

2. A check is performed to ensure the test sequence has been executed correctly. This usually consists of checking value of P in:

   a. the register file dump,

   b. the job or monitor exchange pack or

   c. 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.

3. The register file dump or Job exchange package is checked to ensure that MCR or UCR bits have been set correctly (whichever applies).

4. 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 – Sections 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 deadstart 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). 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.

SUB06_4

This subsection uses a KEYPOINT (op code B1(16)) command to generate an address specification error. The Bn field of the keypoint buffer pointer referenced by the command will have bit 32 equal to 1.

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.

SUB08_5

A write access violation is generated in this subsection by use of a KEYPOINT (op code = B1(16)) command. The keypoint buffer pointer references a segment whose segment descriptor WP field defines the segment as nonwritable.

SUB08_6

A write access violation is generated in this subsection by use of a KEYPOINT (op code = B1(16)) command. The keypoint buffer pointer utilizes a ring number (X'F') greater than ring R1 of the segment descriptor for the segment in which the keypoint buffer resides.

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 segment/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 search 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.

SUB11_5

This subsection uses a KEYPOINT command to generate the interrupt. The page descriptors for this section for page 2 are set up in the same manner as described for subsection 11_2.

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.

SUB14_5

This subsection generates an invalid segment interrupt by employing a KEYPOINT (op code = B1(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 KEYPOINT command.

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 segment/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 search 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.

SUB11_5

This subsection uses a KEYPOINT command to generate the interrupt. The page descriptors for this section for page 2 are set up in the same manner as described for subsection 11_2.

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.

SUB14_5

This subsection generates an invalid segment interrupt by employing a KEYPOINT (op code = B1(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 KEYPOINT command.

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. Specifically, 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 a DIVX (op code 27(16)) command with a divisor of zero is employed to generate a divide fault and also a trap exception condition. The test runs with traps enabled in job and monitor modes. When the divide fault 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.

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 privileged segment.

SUB18_3

This section generates a privileged 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 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 (2^{62}). 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 conditions 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 simultaneously 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 conditions 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. Similarly 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. 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 – Model 990 Processor Model Dependent Test

This section is similar to that for model 835 described in section 38. The principal difference is the data entered in the PTM which causes the uncorrected error.

SECT 41 – Keypoint Function Test

This section provides a check of the timer field in the keypoint buffer entry and the addressing and incrementing of the keypoint buffer pointer. These tests are intended to supplement those provided in FCT1, 2, 3, and 5 which check other functions of the keypoint command and the keypoint buffer pointer (KBP).

Subsection 1 executes two conditions. In the first of these two conditions the keypoint command is executed 64 times causing the automatic incrementing of the KBP. The KC field of each buffer entry should increment from 0 to 63 and the timer field in each case should be greater than the previous. These fields are checked after each execution of the keypoint command and will cause immediate execution of a HALT command if they are not correct. A normal HALT command is executed after 64 keypoint executions.

The second condition is similar but does 65 keypoints, the last of which falls beyond a page boundary. The processor should halt on the keypoint command with a page fault (MCR bit 9 (57)).

Both conditions operate in the monitor mode with traps disabled and monitor and user mask register bits clear.

Random Fast/Slow Test - RFST 20

Program Description 20-1
General 20-1
Basic Test Flow 20-1

Random Fast/Slow Test - RFST 20

Program Description

General

RFST generates a random set of instructions and operands with each pass. This random set of instructions is executed and the results saved. Then the instruction stream is executed again, and results saved (0 pads). These results are then compared and any difference is flagged as an error. In the next step, one pad instructon is inserted between each instruction, and the set of instructions with pads is executed, and the results are saved. These results are then compared with the original nonpadded results and any difference is flagged as an error. The pad count is incremented by one and the set of instructions with pads is executed again with results saved. Results are compared with the original nonpadded results. This sequence is repeated with pad count incremented to 16 decimal. This constitutes a test pass. Each succeeding pass generates a new set of random instructions and operands.

NOTE

Running with the first pad count equal to 0 is essentially repeating the fast loop. This verifies that no errors occur when re-executing the instruction stream.

Basic Test Flow

The basic steps of each pass are:

1. Display running message.
2. Process any keyboard input.
3. Generate random instructions, A/X register values, and random data memory buffers.
4. Execute the set of instructions using the A/X registers and memory buffers. Save A/X register values and memory data.
5. Insert O pad instructions between each instruction in the instruction set being used. Reinitialize A/X/memory with initial random values. Execute padded instruction set and save A/X/memory values.
6. Compare first execution results with padded execution results. Any differences are considered errors and will be reported.
7. If a miscompare occurs, and TRIM mode is selected, then RFST reduces the failing instruction set to the minimum needed to cause failure. This is done by removing each instruction, one at a time, and re- executing to check if the error still occurs. A default mode (TRIM) will re-execute the test one time after removal of each instruction. Use of the TRIMI command allows for re-execution of 256 times to check for intermittent errors. TRIMR is identical to TRIM except that the instructions are replaced with $1ADD_{16}$ and $1ACC_{16}$ instructions, so that the failing instruction(s) do not change position (address). TRIMB executes TRIM as far as possible while still retaining the error and then executes TRIMR to further isolate the failing instruction(s). TRIMI executes with either or both trim modes.

8. Increase the pad count by one and repeat padded execution and compare steps 5 and 6. Do this for pad counts up to 16.

9. A scope loop is offered for either the set of instructions without the pads, or the test padded set.

Debug Test - DBUG 21

Program Description 21-1
General 21-1
Basic Test Flow 21-1

Debug Test - DBUG 21

Program Description

General

DBUG generates a random set of instruction and operands each pass. A simulated debug buffer is created based on the debug code which saves the debug index and P-counter for each instruction which causes a debug trap. The random instruction stream is executed with debug disabled, and all results are saved. Debug is then enabled and the random instructions are executed again. The results are saved and compared to the run with debug disabled. The actual debug buffer (created by the trap handler) is compared to the simulated buffer to ensure that debug occurred correctly. Each individual debug code is used with a combination of all codes (READ, WRITE, FETCH, BRANCH, CALL). After executing with each code and then the combination, one test pass is complete. Each succeeding pass generates a new set of random instructions and operands.

Basic Test Flow

The basic steps of each pass are:

1. Display running message.
2. Process any keyboard input.
3. Generate random instructions, A/X register values, and random data memory buffers. Generate simulated debug buffer.
4. Execute the set of instructions using the A/X registers and memory buffers with debug disabled. Save A/X register values and memory data.
5. Pick up debug code and enable debug. Reinitialize A/X/memory with initial random values. Execute instruction stream with debug enabled. Save A/X/memory values.
6. Compare first execution results with debug execution results. Any differences are considered an error and are reported.
7. Use the next debug code, repeat debug enabled execution, and compare results (steps 5 and 6). Repeat for all debug codes.
8. A scope loop is offered for either the execution of instructions without debug, or execution with debug enabled.

Keypoint Test - KYPT 22

Program Description 22-1
General 22-1
Detailed 22-1

Keypoint Test - KYPT 22

Program Description

General

KYPT tests the normal and abnormal operation of the system instruction keypoint. In the normal phase, sections 0 through 7, no interrupts associated with the instruction are expected to occur during its execution. In the abnormal phase, sections 8 through 11, a condition is established such that the execution of the instruction causes an exchange interrupt to monitor. The abnormal phase tests the exception conditions which are associated with the instruction.

Detailed

Each error message has a description of each section and/or subsection. See chapter 10.

Cache Test - CACH 23

Program Description 23-1
General 23-1
Basic Test Flow 23-1

Cache Test - CACH 23

Program Description

General

CACH tests the addressing and control sequences within the cache memory. Although CACH does not directly test the data written to the cache, data patterns are used to realize the primary objective. To check the access and control sequences with the cache, the test loads it sequentially with an address pattern of 0-FFF_{16} (default size of cache). The testing sequences first read from cache randomly within addresses 0-FFF_{16}, and then check that the data is correct (data = address read). The next test reads from cache randomly within addresses 0-$1FFF_{16}$. This causes cache to reload because the addresses used are twice the size of the cache. The data is then checked again (data = address read). Next, the test randomly accesses cache within addresses 0-FFF_{16}, but uses three different segment numbers, that is, three different ASIDs. This causes cache to reload because of tag miscompares. Cache is based on an SVA (ASID plus byte number), and that is the tag that determines a cache hit or miss.

Basic Test Flow

Each section does the following:

1. Presets cache to an initialization pattern and purges cache.
2. Writes address pattern to memory (cache bypass segment).
3. Loads address pattern to cache by LXI instructions from a non-cache bypass segment.
4. Loads registers with random addresses.
5. Executes a series of load, or load and store, instructions.
6. Checks that data read equals the random addresses used on the loads.
7. Reports any errors in the data read.
8. Does steps 4 through 7 for a number of iterations in parameter word 17 (default = 10,000).

The data loaded into cache is the address of each cache word. For sections that use three different segment numbers, the left half of the data word holds the segment number. This forms the address when reporting an error. Sections that use only one address register do not have the segment number in the cache data.

Model Independent General Design Specification Tests - MIGDS 24

Program Description 24-1
General 24-1

Model Independent General Design Specification Tests - MIGDS 24

Program Description

General

These tests are organized into a test pass structure rather that into sections or subsections. They run in 4 modes. The default mode (RUN ALL TESTS) runs all operations (instructions) using all operands. The basic steps are as follows:

1. Run all operations using all operands with Traps disabled and User Mask clear.
2. Run all operations using all operands with Traps disabled and User Mask set.
3. Run all operations using all operands with Traps enabled and User Mask clear.
4. Run all operations using all operands with Traps enabled and User Mask set.

The other three modes can only be entered if the stop-for-parameter bit (bit 49 of PW0) is set. In these modes the test questions/messages step the operator through the different test modes.

The second mode (RUN ALL TESTS - not default run) is similar to the default mode except that the operator can deselect operations, or select only random operands.

The third mode (FULL TEST) runs one selected operation through all operands but with a selected setting in the TE (Trap Enable Flipflop) and UM (User Mask).

The fourth mode (SINGLE TEST) runs one selected operation through one selected set of operands. The state of TE and UM is selected, and in this mode a scope loop is offered.

NOTE

Due to the length of the BDP tests (NUMR/BIMM/EDIT/BYTE), only steps 1 and 4 of the basic steps described above are available – traps disabled-user mask clear and traps enabled-user mask set.

Glossary A

Glossary A

A

A register
Address register

ADU
Assembly/disassembly unit

ASCII
American standard code for information interchange

ASID
Active segment identifier

B

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

C

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

D

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

C

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

E

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

G

GE
Greater than or equal

GK
Global key

GL
Global lock

GT
Greater than

I

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

J

JEP
Job mode exchange package

JPS
Job process state pointer

K

KC
Keypoint code

KCN
Keypoint class number

KEF
Keypoint enable flag

KM
Keypoint mask

L

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

M

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

N

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

O

OCF
On-condition flag

OI
Options installed

ON
Occurrence number

OP
Operation code

OS
Operating system

P

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

Q

QLT1
Quick look test

R

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, availability, 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

S

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

T

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

U

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

V

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

W

WAR
Word assembly register

WCB
Write Check Bits

WDR
Word disassembly register

WP
Write access control (segment descriptor field)

X

XP

Execute access control (segment descriptor field)

Z

ZF

Zero field toggle (BDP edit instruction)

RCT1 and RCT2 Test/Usage Sets B

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	
30	099	Add floating point		1
31	100	Subtract floating point		1

Op Code	Ref. No.	Short Name	Usage	Test
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

Op Code	Ref. No.	Short Name	Usage	Test
9A	111	Branch >, floating point		1
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		

Exchange Sequence Diagrams C

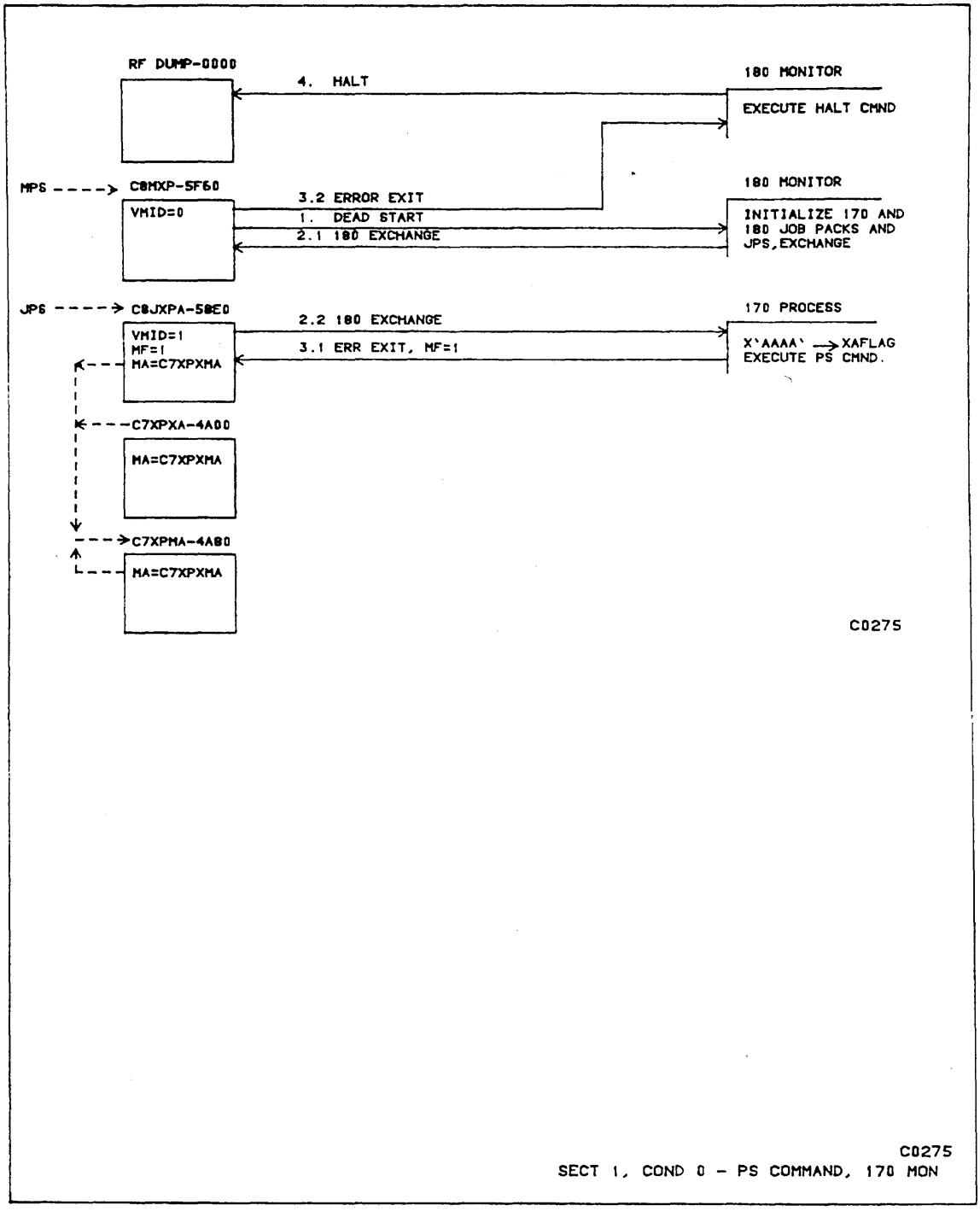
Exchange Sequence Diagrams C

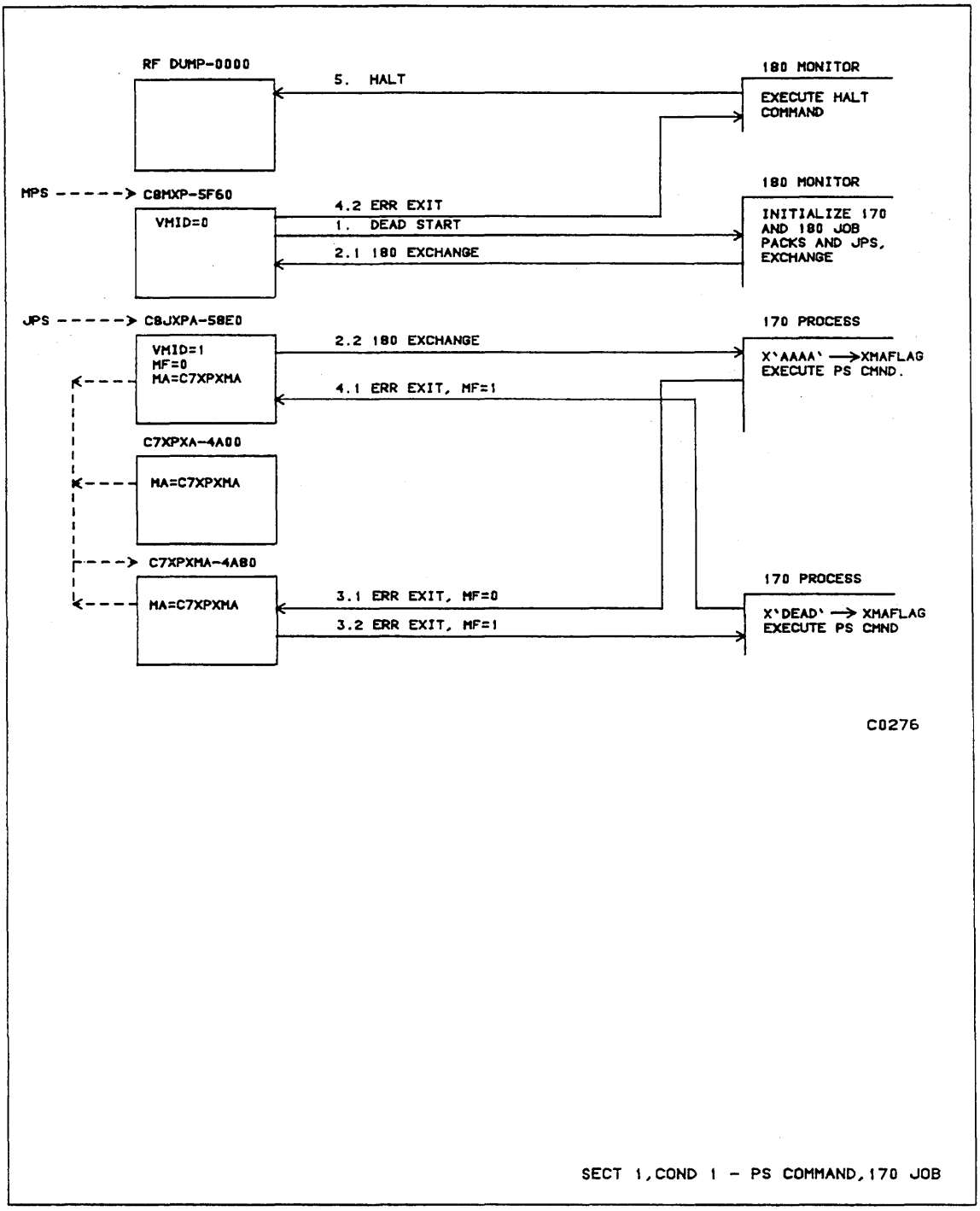
The following pages provide diagrams illustrating the sequence of exchanges in the various sections of the test. No diagram is provided for section 4. Some diagrams serve more than one section.

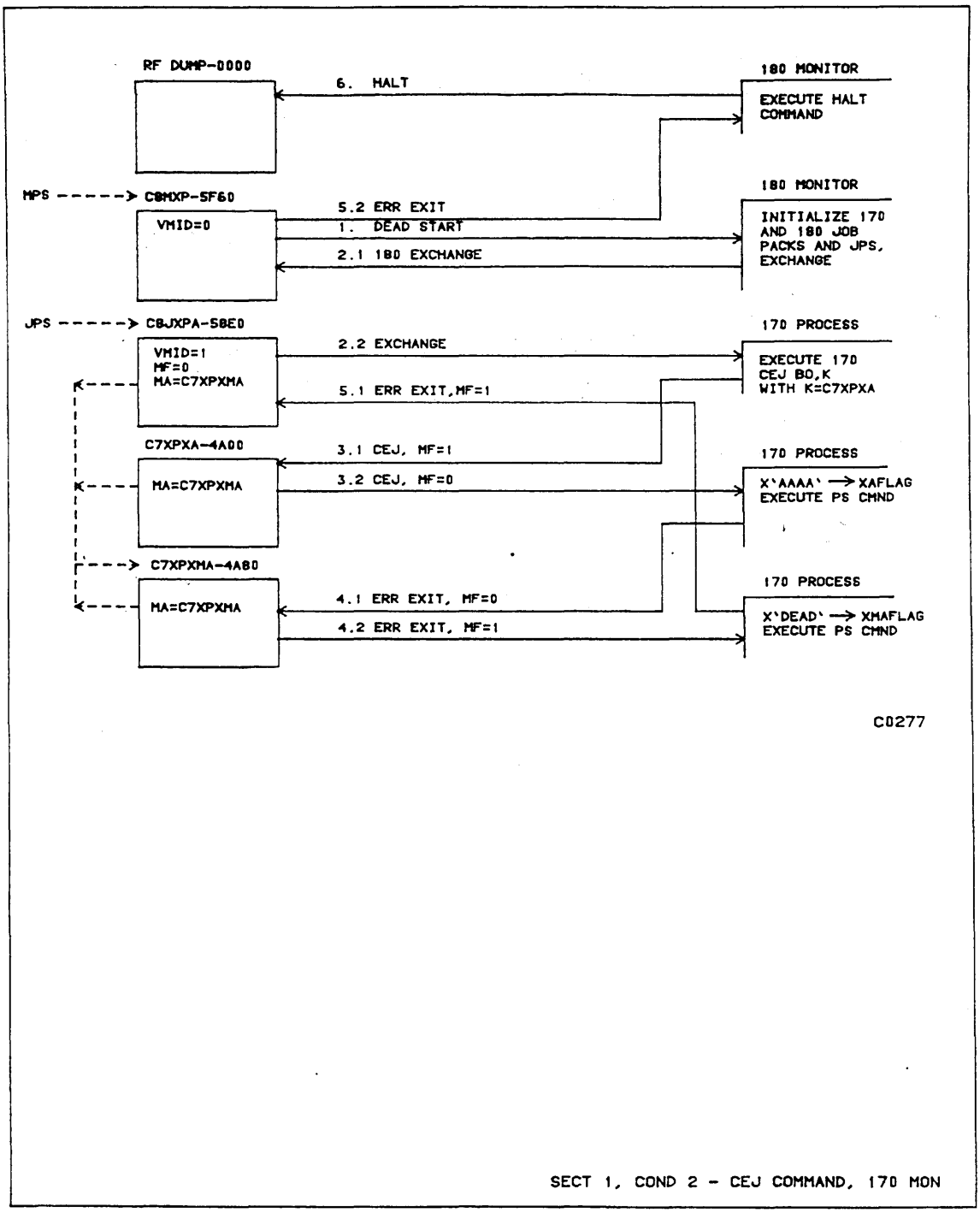
In each diagram, the exchange packages are on the left. The blocks on the right briefly describe the function of an associated process. The connecting lines indicate the movement of a process' registers from an exchange package in memory into the central processor's registers or vice versa. Each line bears a number showing the sequence step and note indicating the event which triggers the action. Dashed lines indicate a pointing action.

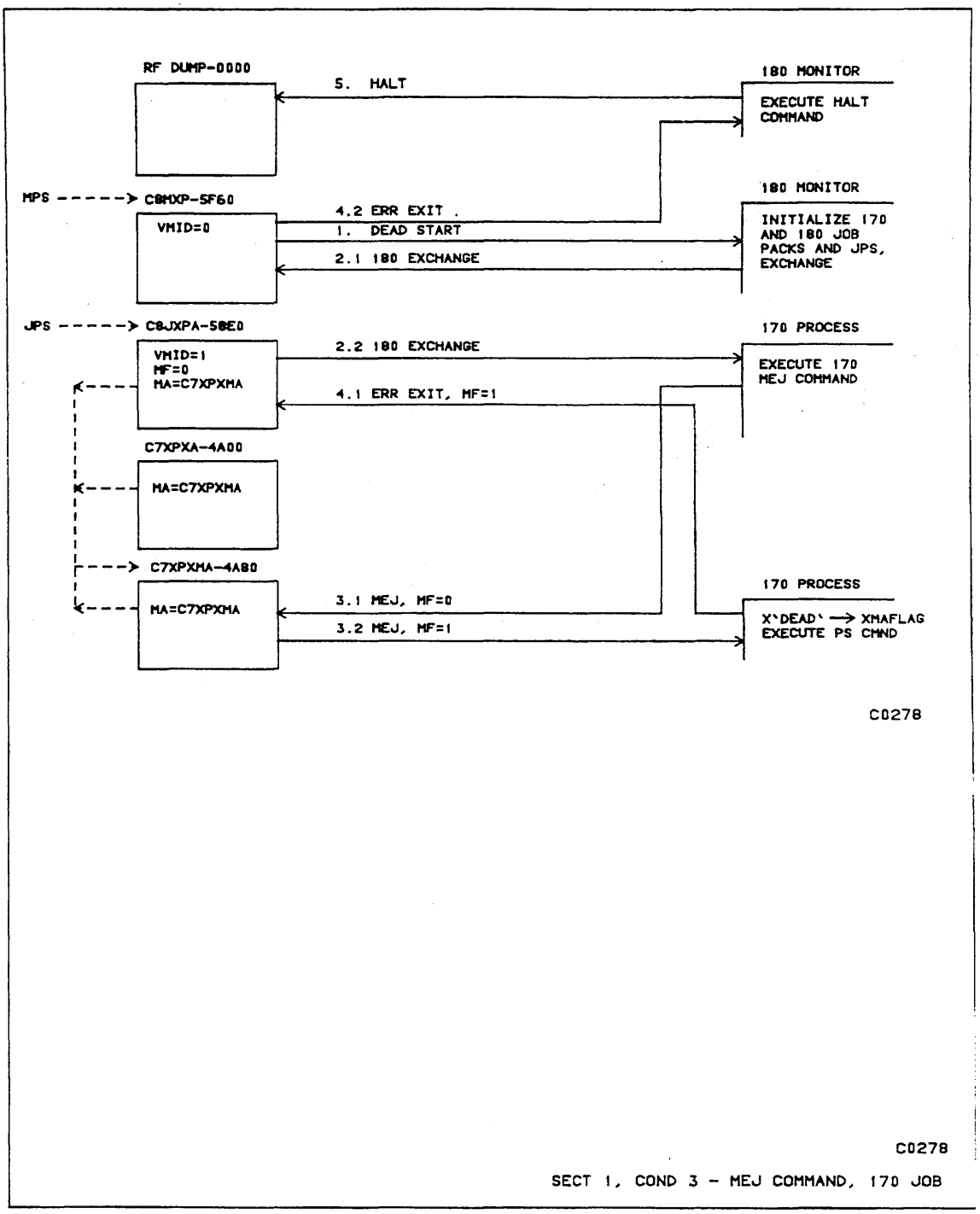
Each exchange package shows the assembly listing name of the package and its location (hex byte address) in central memory. A package name (label) which begins with C8 refers to a CYBER 180 package; C7 refers to a CYBER 170 package.

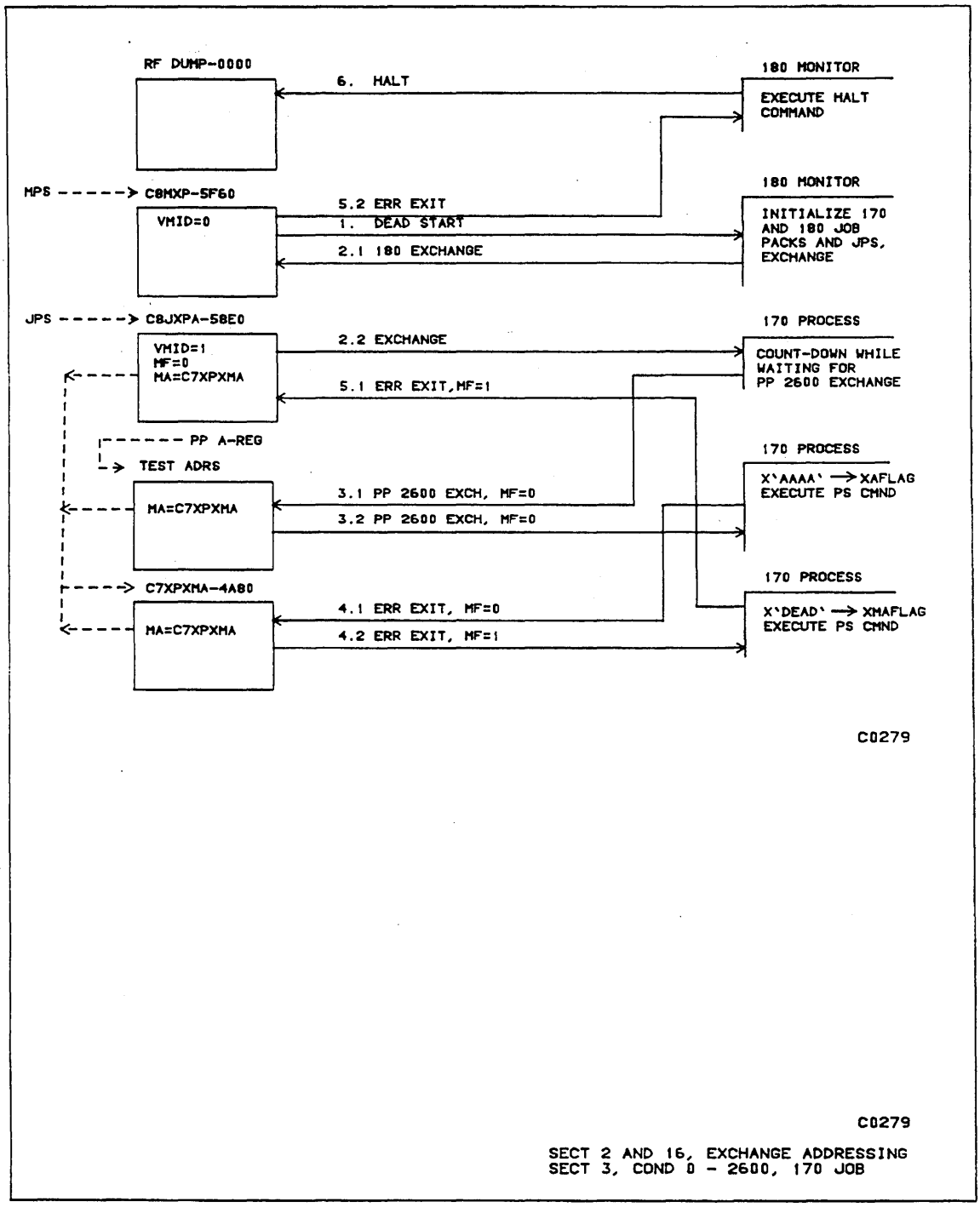
Some sequence steps or a series of steps may be repeated many times during the execution of a given condition. The diagrams only provide an overall view; reference should be made to the descriptions with each test section in part II-6 of this manual for additional details.

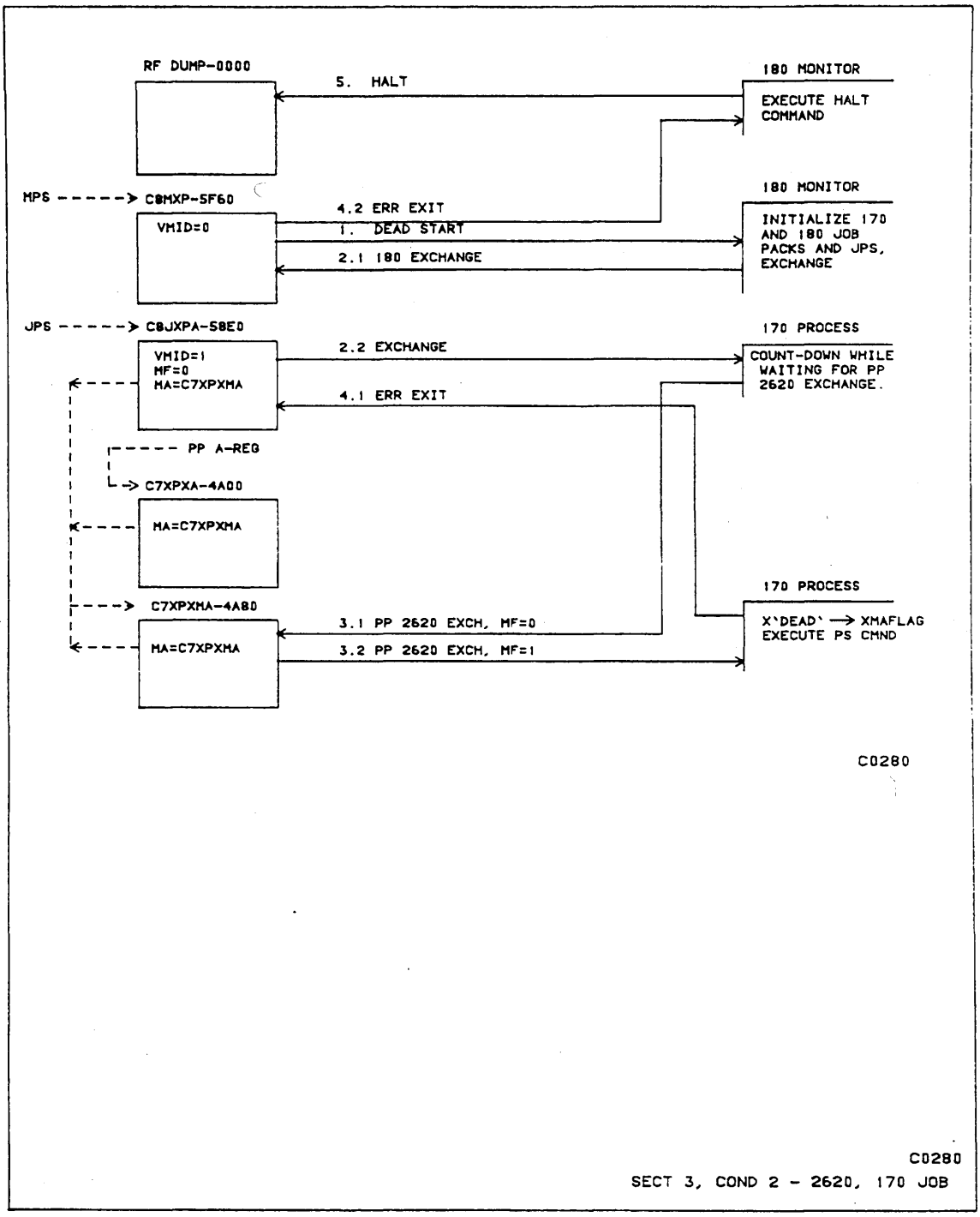

RF DUMP-0000
4. HALT
180 MONITOR
EXECUTE HALT CMND
MPS
C8MXP-5F60
VMID=0
3.2 ERROR EXIT
1. DEAD START
2.1 180 EXCHANGE
180 MONITOR
INITIALIZE 170 AND
180 JOB PACKS AND
JPS,EXCHANGE
JPS
C8JXPA-58E0
VMID=1
MF=1
MA=C7XPXMA
2.2 180 EXCHANGE
3.1 ERR EXIT, MF=1
170 PROCESS
X'AAAA' → XAFLAG
EXECUTE PS CMND.
C7XPXA-4A00
MA=C7XPXMA
C7XPMA-4A80
MA=C7XPXMA
C0275
C0275
SECT 1, COND 0 - PS COMMAND, 170 MON

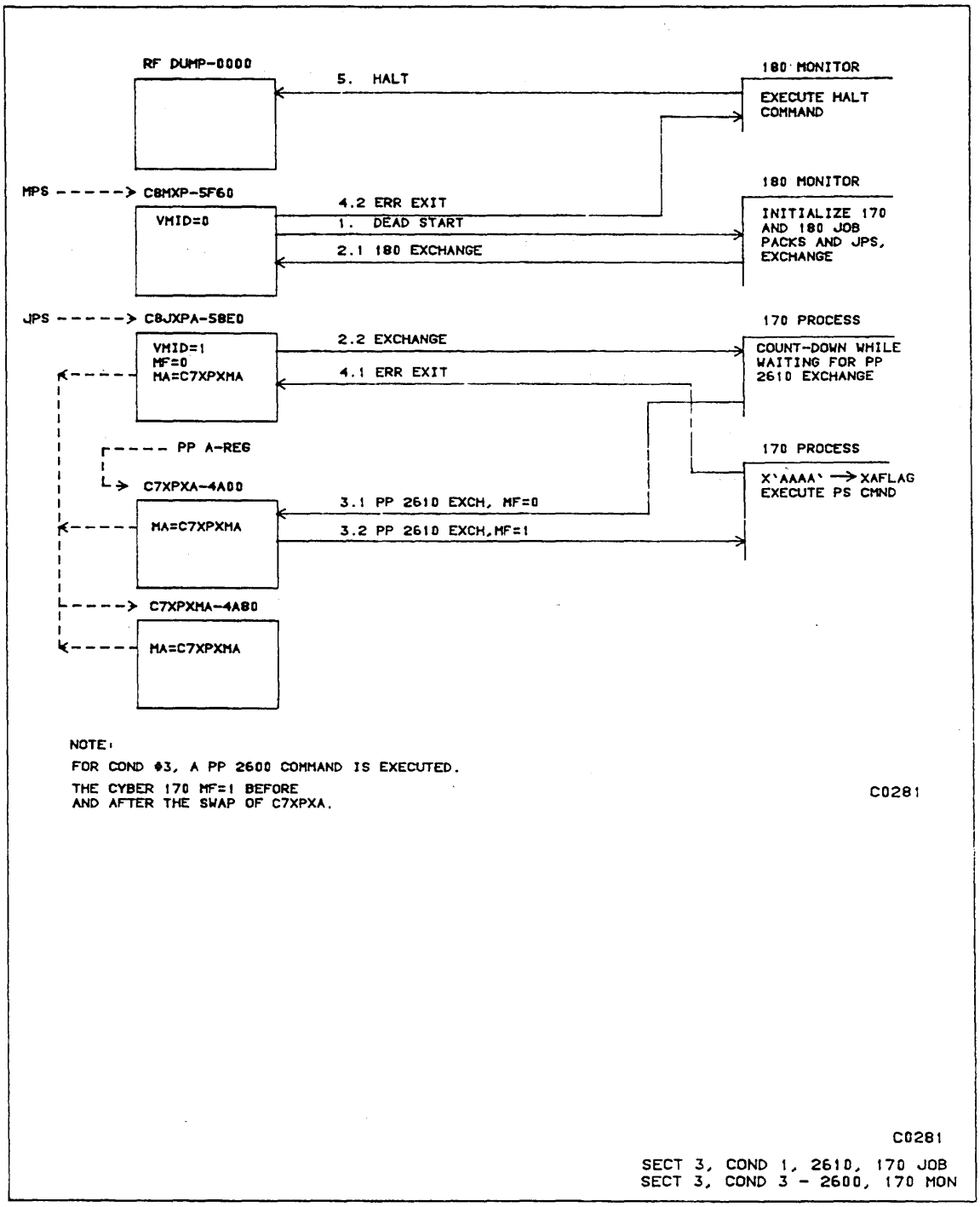

RF DUMP-0000
5. HALT
180 MONITOR
EXECUTE HALT COMMAND
MPS
C8MXP-5F60
VMID=0
4.2 ERR EXIT
1. DEAD START
2.1 180 EXCHANGE
180 MONITOR
INITIALIZE 170 AND 180 JOB PACKS AND JPS, EXCHANGE
JPS
C8JXPA-58E0
VMID=1
MF=0
MA=C7XPXMA
2.2 180 EXCHANGE
4.1 ERR EXIT, MF=1
170 PROCESS
X'AAAA' →XMAFLAG
EXECUTE PS CMND.
C7XPXA-4A00
MA=C7XPXMA
C7XPXMA-4A80
MA=C7XPXMA
3.1 ERR EXIT, MF=0
3.2 ERR EXIT, MF=1
170 PROCESS
X'DEAD' → XMAFLAG
EXECUTE PS CMND
C0276
SECT 1,COND 1 - PS COMMAND,170 JOB

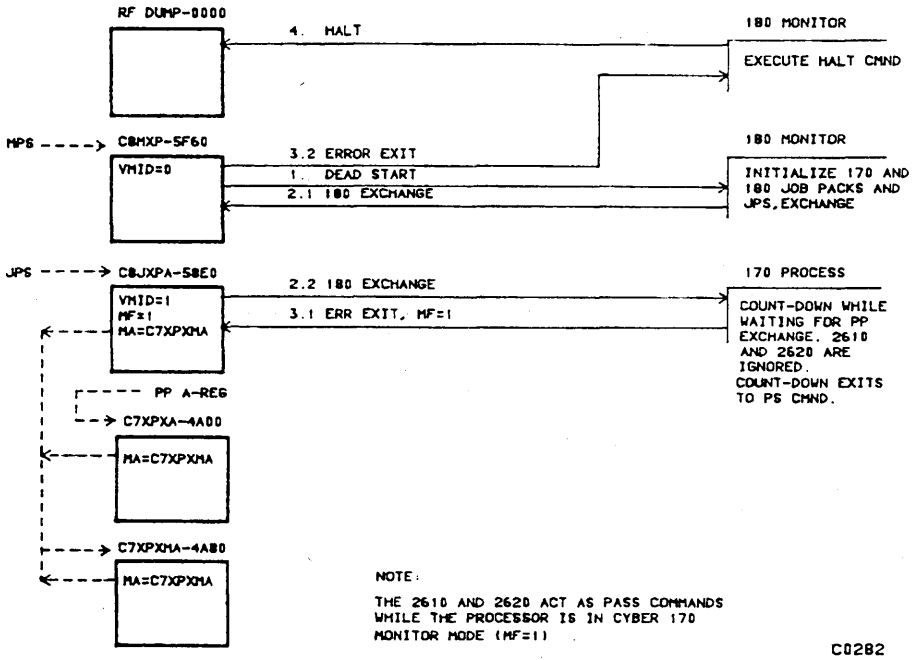

RF DUMP-0000
6. HALT
180 MONITOR
EXECUTE HALT COMMAND
MPS
C8MXP-5F60
VMID=0
5.2 ERR EXIT
1. DEAD START
2.1 180 EXCHANGE
180 MONITOR
INITIALIZE 170 AND 180 JOB PACKS AND JPS, EXCHANGE
JPS
C8JXPA-58E0
VMID=1
MF=0
MA=C7XPXMA
2.2 EXCHANGE
5.1 ERR EXIT,MF=1
170 PROCESS
EXECUTE 170 CEJ BO,K WITH K=C7XPXA
C7XPXA-4A00
MA=C7XPXMA
3.1 CEJ, MF=1
3.2 CEJ, MF=0
170 PROCESS
X'AAAA' → XAFLAG
EXECUTE PS CMND
C7XPXMA-4A80
MA=C7XPXMA
4.1 ERR EXIT, MF=0
4.2 ERR EXIT, MF=1
170 PROCESS
X'DEAD' → XMAFLAG
EXECUTE PS CMND
C0277
SECT 1, COND 2 - CEJ COMMAND, 170 MON


RF DUMP-0000
5. HALT
180 MONITOR
EXECUTE HALT COMMAND
MPS
C8MXP-5F60
VMID=0
4.2 ERR EXIT .
1. DEAD START
2.1 180 EXCHANGE
180 MONITOR
INITIALIZE 170 AND 180 JOB PACKS AND JPS, EXCHANGE
JPS
C8JXPA-58E0
VMID=1
MF=0
MA=C7XPXMA
2.2 180 EXCHANGE
4.1 ERR EXIT, MF=1
170 PROCESS
EXECUTE 170 MEJ COMMAND
C7XPXA-4A00
MA=C7XPXMA
C7XPXMA-4A80
MA=C7XPXMA
3.1 MEJ, MF=0
3.2 MEJ, MF=1
170 PROCESS
X'DEAD' --> XMAFLAG
EXECUTE PS CMND
C0278
C0278
SECT 1, COND 3 - MEJ COMMAND, 170 JOB


RF DUMP-0000
6. HALT
180 MONITOR
EXECUTE HALT COMMAND
MPS
C8MXP-5F60
VMID=0
5.2 ERR EXIT
1. DEAD START
2.1 180 EXCHANGE
180 MONITOR
INITIALIZE 170 AND 180 JOB PACKS AND JPS, EXCHANGE
JPS
C8JXPA-58E0
VMID=1
MF=0
MA=C7XPXMA
2.2 EXCHANGE
5.1 ERR EXIT,MF=1
170 PROCESS
COUNT-DOWN WHILE WAITING FOR PP 2600 EXCHANGE
PP A-REG
TEST ADRS
MA=C7XPXMA
3.1 PP 2600 EXCH, MF=0
3.2 PP 2600 EXCH, MF=0
170 PROCESS
X'AAAA' → XAFLAG
EXECUTE PS CMND
C7XPXMA-4A80
MA=C7XPXMA
4.1 ERR EXIT, MF=0
4.2 ERR EXIT, MF=1
170 PROCESS
X'DEAD' → XMAFLAG
EXECUTE PS CMND
C0279
C0279
SECT 2 AND 16, EXCHANGE ADDRESSING
SECT 3, COND 0 - 2600, 170 JOB

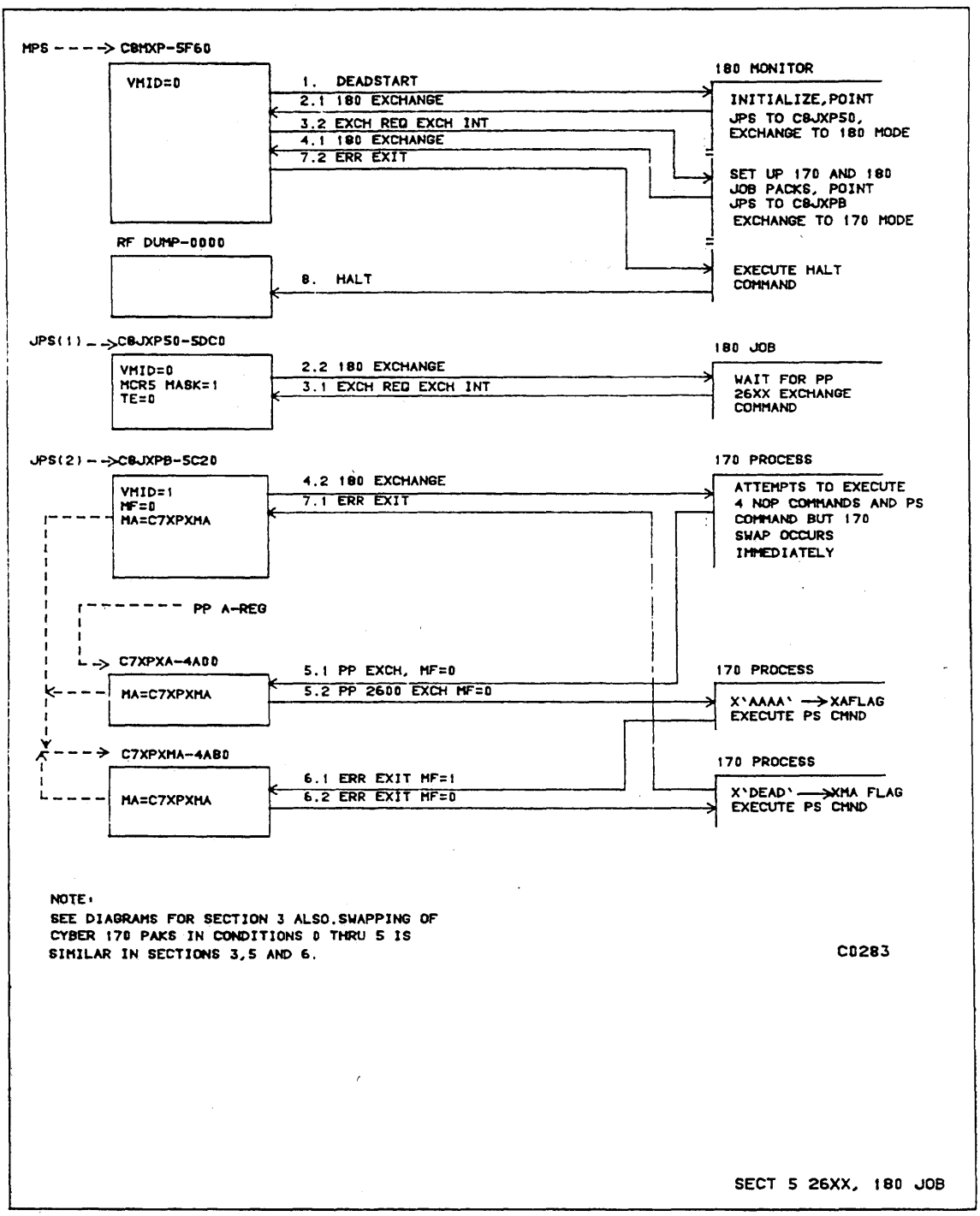

RF DUMP-0000
5. HALT
180 MONITOR
EXECUTE HALT COMMAND
MPS
C0MXP-5F60
VMID=0
4.2 ERR EXIT
1. DEAD START
2.1 180 EXCHANGE
180 MONITOR
INITIALIZE 170 AND 180 JOB PACKS AND JPS, EXCHANGE
JPS
C0JXPA-58E0
VMID=1
MF=0
MA=C7XPXMA
2.2 EXCHANGE
4.1 ERR EXIT
170 PROCESS
COUNT-DOWN WHILE WAITING FOR PP 2620 EXCHANGE.
PP A-REG
C7XPXA-4A00
MA=C7XPXMA
C7XPXMA-4A80
MA=C7XPXMA
3.1 PP 2620 EXCH, MF=0
3.2 PP 2620 EXCH, MF=1
170 PROCESS
X'DEAD' → XMAFLAG
EXECUTE PS CMND
C0280
C0280
SECT 3, COND 2 - 2620, 170 JOB

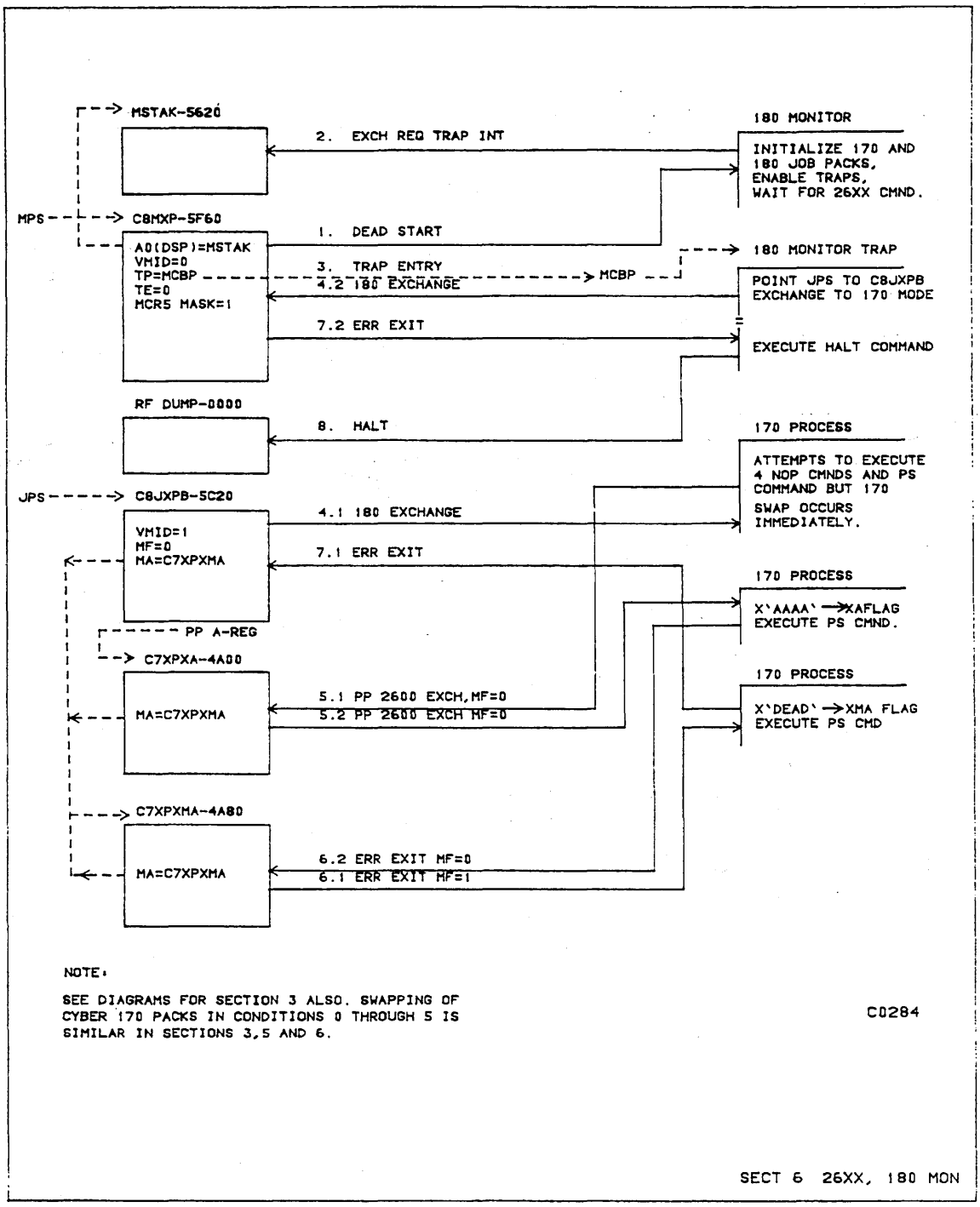

RF DUMP-0000
5. HALT
180 MONITOR
EXECUTE HALT COMMAND
MPS
C8MXP-5F60
VMID=0
4.2 ERR EXIT
1. DEAD START
2.1 180 EXCHANGE
180 MONITOR
INITIALIZE 170 AND 180 JOB PACKS AND JPS, EXCHANGE
JPS
C8JXPA-5BE0
VMID=1
MF=0
MA=C7XPXMA
2.2 EXCHANGE
4.1 ERR EXIT
170 PROCESS
COUNT-DOWN WHILE WAITING FOR PP 2610 EXCHANGE
PP A-REG
C7XPXA-4A00
MA=C7XPXMA
3.1 PP 2610 EXCH, MF=0
3.2 PP 2610 EXCH,MF=1
170 PROCESS
X'AAAA' → XAFLAG
EXECUTE PS CMND
C7XPXMA-4A80
MA=C7XPXMA
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

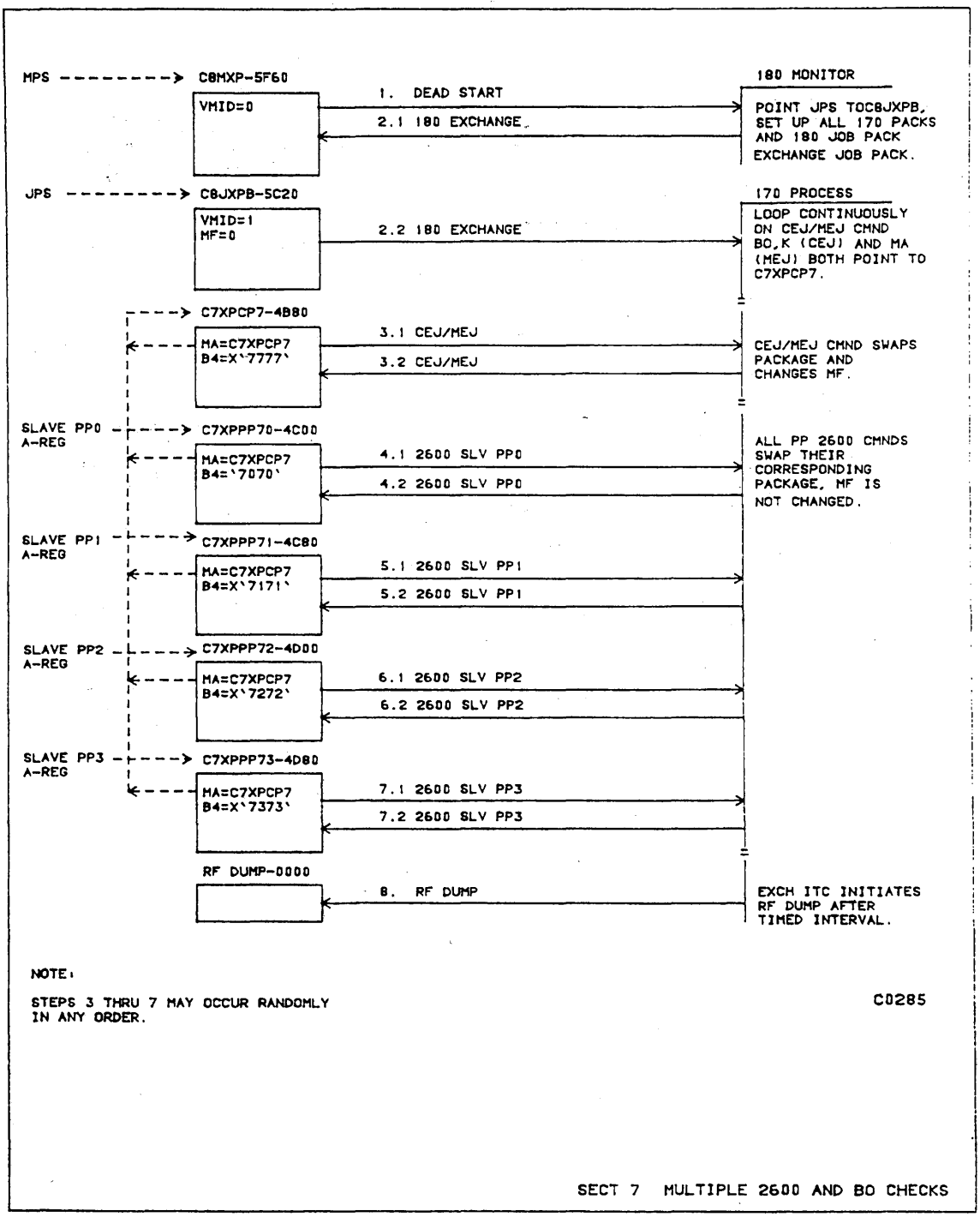


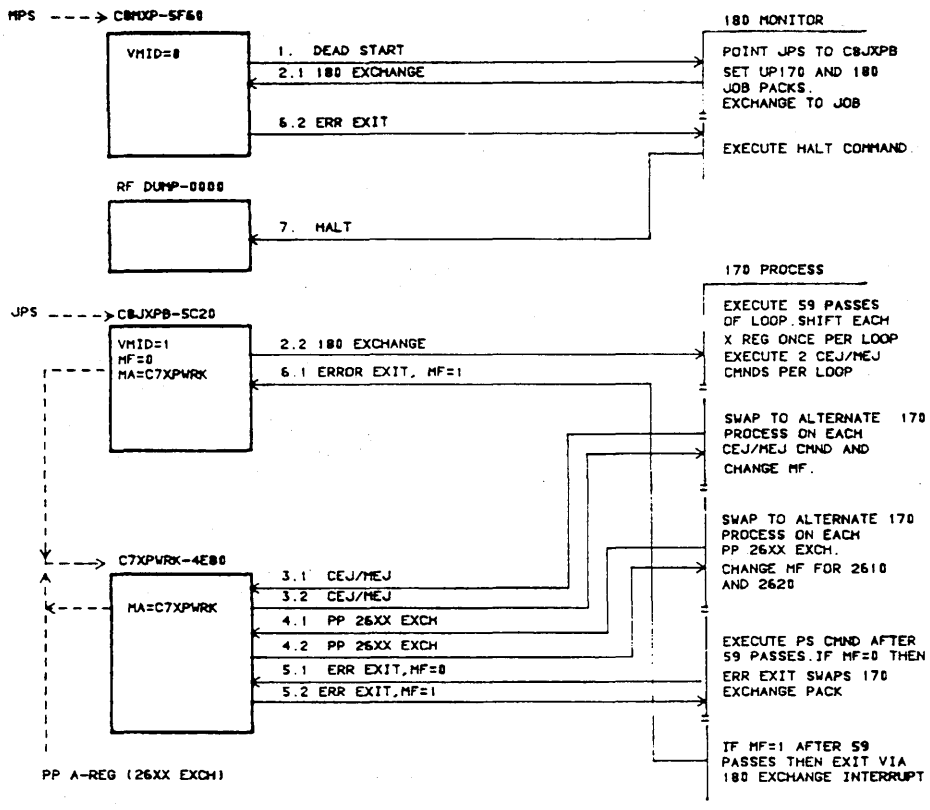
C0282

SECT 3, COND 4 - 2610, 170 MON
SECT 3, COND 5 - 2620, 170 MON

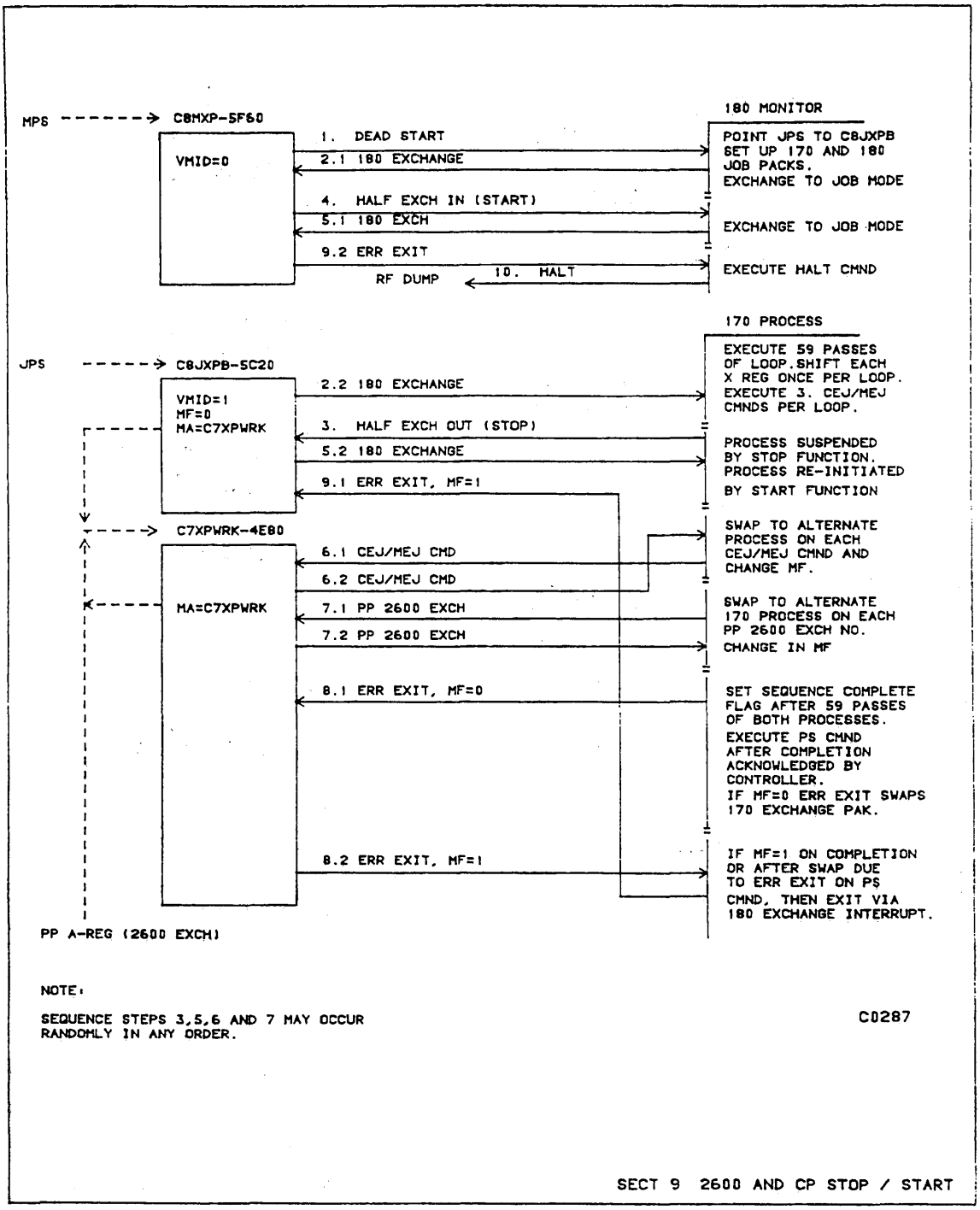

MPS ----> C8MXP-5F60
VMID=0
1. DEADSTART
2.1 180 EXCHANGE
3.2 EXCH REQ EXCH INT
4.1 180 EXCHANGE
7.2 ERR EXIT
180 MONITOR
INITIALIZE,POINT JPS TO C8JXP50, EXCHANGE TO 180 MODE
SET UP 170 AND 180 JOB PACKS, POINT JPS TO C8JXPB EXCHANGE TO 170 MODE
RF DUMP-0000
8. HALT
EXECUTE HALT COMMAND
JPS(1) ---> C8JXP50-5DC0
VMID=0
MCR5 MASK=1
TE=0
2.2 180 EXCHANGE
3.1 EXCH REQ EXCH INT
180 JOB
WAIT FOR PP 26XX EXCHANGE COMMAND
JPS(2) ---> C8JXPB-5C20
VMID=1
MF=0
MA=C7XPXMA
4.2 180 EXCHANGE
7.1 ERR EXIT
170 PROCESS
ATTEMPTS TO EXECUTE 4 NOP COMMANDS AND PS COMMAND BUT 170 SWAP OCCURS IMMEDIATELY
PP A-REG
C7XPXA-4A00
MA=C7XPXMA
5.1 PP EXCH, MF=0
5.2 PP 2600 EXCH MF=0
170 PROCESS
X'AAAA' ---> XAFLAG
EXECUTE PS CMND
C7XPXMA-4AB0
MA=C7XPXMA
6.1 ERR EXIT MF=1
6.2 ERR EXIT MF=0
170 PROCESS
X'DEAD' ---> XMA FLAG
EXECUTE PS CMND
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

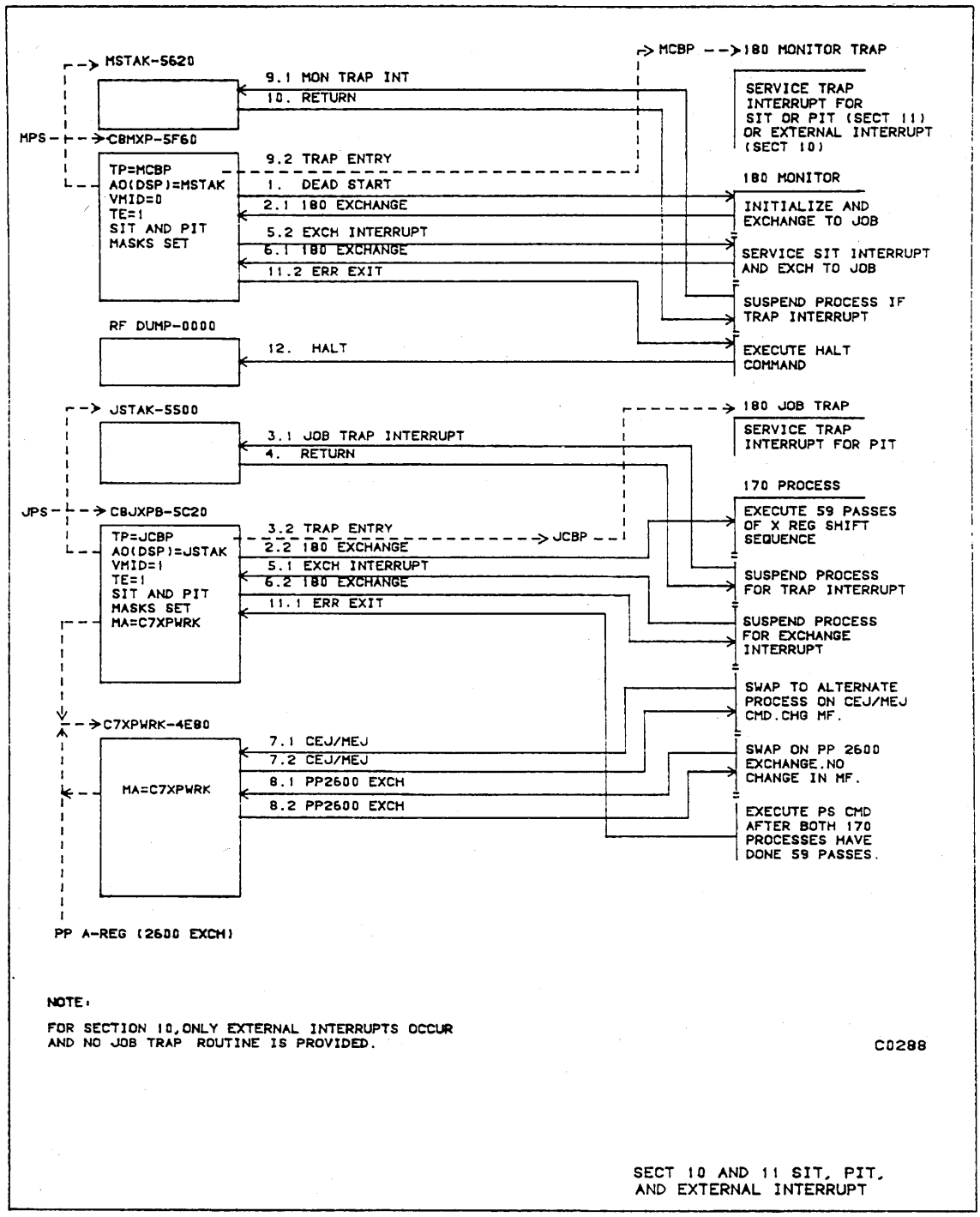

MSTAK-5620
180 MONITOR
2. EXCH REQ TRAP INT
INITIALIZE 170 AND 180 JOB PACKS, ENABLE TRAPS, WAIT FOR 26XX CMND.
MPS
C8MXP-5F60
1. DEAD START
A0(DSP)=MSTAK
VMID=0
TP=MCBP
TE=0
MCR5 MASK=1
3. TRAP ENTRY
180 MONITOR TRAP
MCBP
4.2 180 EXCHANGE
POINT JPS TO C8JXPB
EXCHANGE TO 170 MODE
7.2 ERR EXIT
EXECUTE HALT COMMAND
RF DUMP-0000
8. HALT
170 PROCESS
ATTEMPTS TO EXECUTE 4 NOP CMNDS AND PS COMMAND BUT 170 SWAP OCCURS IMMEDIATELY.
JPS
C8JXPB-5C20
4.1 180 EXCHANGE
VMID=1
MF=0
MA=C7XPXMA
7.1 ERR EXIT
170 PROCESS
X'AAAA' →XAFLAG
EXECUTE PS CMND.
PP A-REG
C7XPXA-4A00
170 PROCESS
5.1 PP 2600 EXCH,MF=0
MA=C7XPXMA
5.2 PP 2600 EXCH MF=0
X'DEAD' →XMA FLAG
EXECUTE PS CMD
C7XPXMA-4A80
6.2 ERR EXIT MF=0
MA=C7XPXMA
6.1 ERR EXIT MF=1
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

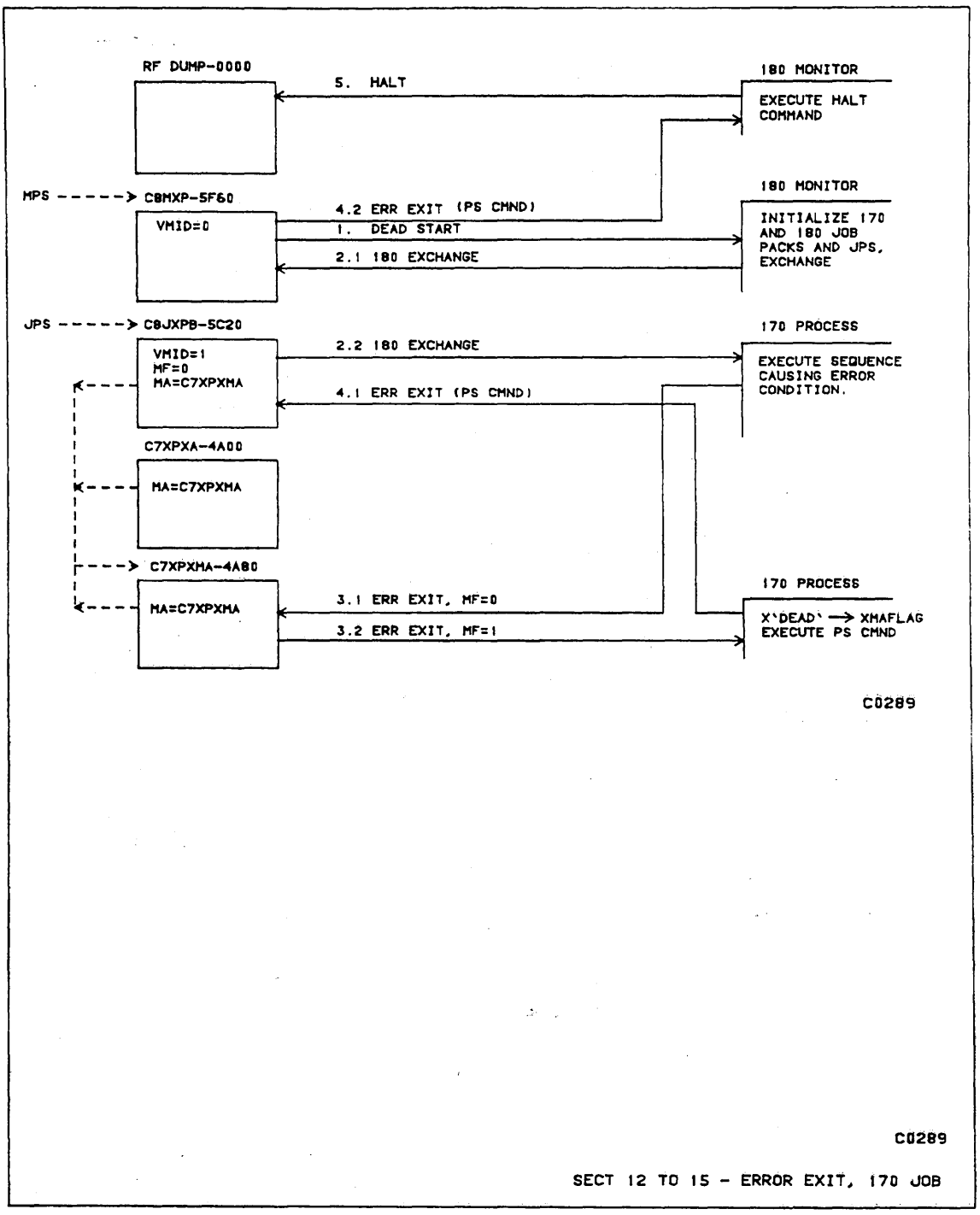

MPS
C8MXP-5F60
VMID=0
1. DEAD START
2.1 180 EXCHANGE
180 MONITOR
POINT JPS TOC8JXPB, SET UP ALL 170 PACKS AND 180 JOB PACK EXCHANGE JOB PACK.
JPS
C8JXPB-5C20
VMID=1
MF=0
2.2 180 EXCHANGE
170 PROCESS
LOOP CONTINUOUSLY ON CEJ/MEJ CMND BO,K (CEJ) AND MA (MEJ) BOTH POINT TO C7XPCP7.
C7XPCP7-4B80
MA=C7XPCP7
B4=X'7777'
3.1 CEJ/MEJ
3.2 CEJ/MEJ
CEJ/MEJ CMND SWAPS PACKAGE AND CHANGES MF.
SLAVE PP0 A-REG
C7XPPP70-4C00
MA=C7XPCP7
B4='7070'
4.1 2600 SLV PP0
4.2 2600 SLV PP0
ALL PP 2600 CMNDS SWAP THEIR CORRESPONDING PACKAGE, MF IS NOT CHANGED.
SLAVE PP1 A-REG
C7XPPP71-4C80
MA=C7XPCP7
B4=X'7171'
5.1 2600 SLV PP1
5.2 2600 SLV PP1
SLAVE PP2 A-REG
C7XPPP72-4D00
MA=C7XPCP7
B4=X'7272'
6.1 2600 SLV PP2
6.2 2600 SLV PP2
SLAVE PP3 A-REG
C7XPPP73-4D80
MA=C7XPCP7
B4=X'7373'
7.1 2600 SLV PP3
7.2 2600 SLV PP3
RF DUMP-0000
8. RF DUMP
EXCH ITC INITIATES RF DUMP AFTER TIMED INTERVAL.
NOTE:
STEPS 3 THRU 7 MAY OCCUR RANDOMLY IN ANY ORDER.
C0285
SECT 7 MULTIPLE 2600 AND BO CHECKS

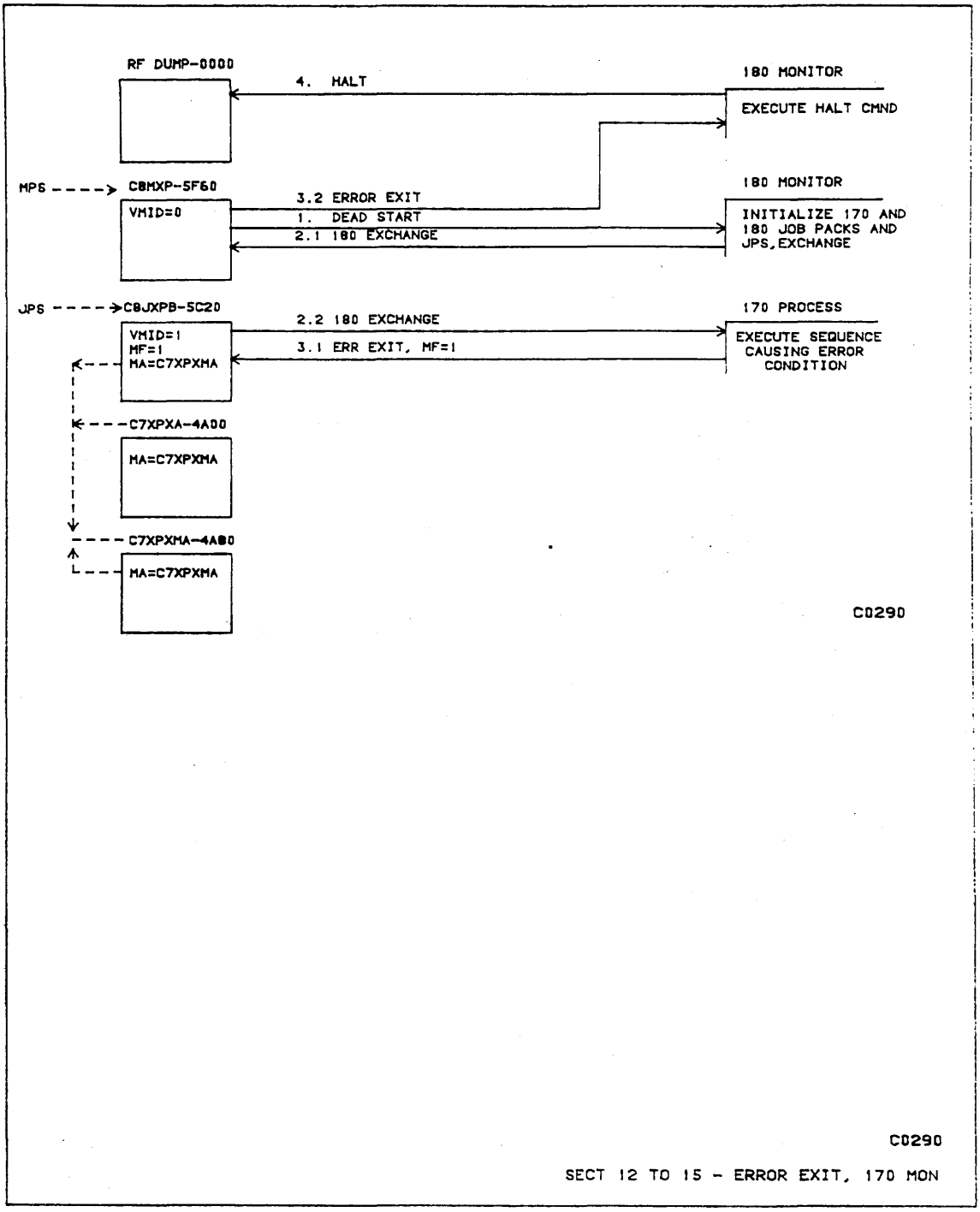


SECT 8 26XX BLOCK READ AND WRITE


MPS
C8MXP-5F60
VMID=0
1. DEAD START
2.1 180 EXCHANGE
4. HALF EXCH IN (START)
5.1 180 EXCH
9.2 ERR EXIT
RF DUMP
10. HALT
180 MONITOR
POINT JPS TO C8JXPB
SET UP 170 AND 180
JOB PACKS.
EXCHANGE TO JOB MODE
EXCHANGE TO JOB MODE
EXECUTE HALT CMND
170 PROCESS
EXECUTE 59 PASSES
OF LOOP.SHIFT EACH
X REG ONCE PER LOOP.
EXECUTE 3. CEJ/MEJ
CMNDS PER LOOP.
JPS
C8JXPB-5C20
VMID=1
MF=0
MA=C7XPWRK
2.2 180 EXCHANGE
3. HALF EXCH OUT (STOP)
5.2 180 EXCHANGE
9.1 ERR EXIT, MF=1
PROCESS SUSPENDED
BY STOP FUNCTION.
PROCESS RE-INITIATED
BY START FUNCTION
C7XPWRK-4E80
MA=C7XPWRK
6.1 CEJ/MEJ CMD
6.2 CEJ/MEJ CMD
7.1 PP 2600 EXCH
7.2 PP 2600 EXCH
SWAP TO ALTERNATE
PROCESS ON EACH
CEJ/MEJ CMND AND
CHANGE MF.
SWAP TO ALTERNATE
170 PROCESS ON EACH
PP 2600 EXCH NO.
CHANGE IN MF
8.1 ERR EXIT, MF=0
SET SEQUENCE COMPLETE
FLAG AFTER 59 PASSES
OF BOTH PROCESSES.
EXECUTE PS CMND
AFTER COMPLETION
ACKNOWLEDGED BY
CONTROLLER.
IF MF=0 ERR EXIT SWAPS
170 EXCHANGE PAK.
8.2 ERR EXIT, MF=1
IF MF=1 ON COMPLETION
OR AFTER SWAP DUE
TO ERR EXIT ON PS
CMND, THEN EXIT VIA
180 EXCHANGE INTERRUPT.
PP A-REG (2600 EXCH)
NOTE:
SEQUENCE STEPS 3,5,6 AND 7 MAY OCCUR
RANDOMLY IN ANY ORDER.
C0287
SECT 9 2600 AND CP STOP / START


MCBP
180 MONITOR TRAP
MSTAK-5620
9.1 MON TRAP INT
10. RETURN
SERVICE TRAP INTERRUPT FOR SIT OR PIT (SECT 11) OR EXTERNAL INTERRUPT (SECT 10)
MPS
C8MXP-5F60
9.2 TRAP ENTRY
180 MONITOR
TP=MCBP
AO(DSP)=MSTAK
VMID=0
TE=1
SIT AND PIT MASKS SET
1. DEAD START
2.1 180 EXCHANGE
INITIALIZE AND EXCHANGE TO JOB
5.2 EXCH INTERRUPT
6.1 180 EXCHANGE
SERVICE SIT INTERRUPT AND EXCH TO JOB
11.2 ERR EXIT
SUSPEND PROCESS IF TRAP INTERRUPT
RF DUMP-0000
12. HALT
EXECUTE HALT COMMAND
JSTAK-5500
180 JOB TRAP
3.1 JOB TRAP INTERRUPT
4. RETURN
SERVICE TRAP INTERRUPT FOR PIT
170 PROCESS
JPS
C8JXPB-5C20
3.2 TRAP ENTRY
JCBP
EXECUTE 59 PASSES OF X REG SHIFT SEQUENCE
TP=JCBP
AO(DSP)=JSTAK
VMID=1
TE=1
SIT AND PIT MASKS SET
MA=C7XPWRK
2.2 180 EXCHANGE
5.1 EXCH INTERRUPT
6.2 180 EXCHANGE
11.1 ERR EXIT
SUSPEND PROCESS FOR TRAP INTERRUPT
SUSPEND PROCESS FOR EXCHANGE INTERRUPT
SWAP TO ALTERNATE PROCESS ON CEJ/MEJ CMD.CHG MF.
C7XPWRK-4E80
7.1 CEJ/MEJ
7.2 CEJ/MEJ
8.1 PP2600 EXCH
8.2 PP2600 EXCH
MA=C7XPWRK
SWAP ON PP 2600 EXCHANGE.NO CHANGE IN MF.
EXECUTE PS CMD AFTER BOTH 170 PROCESSES HAVE DONE 59 PASSES.
PP A-REG (2600 EXCH)
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


RF DUMP-0000
5. HALT
180 MONITOR
EXECUTE HALT COMMAND
MPS
C8MXP-5F60
VMID=0
4.2 ERR EXIT (PS CMND)
1. DEAD START
2.1 180 EXCHANGE
180 MONITOR
INITIALIZE 170 AND 180 JOB PACKS AND JPS, EXCHANGE
JPS
C8JXPB-5C20
VMID=1
MF=0
MA=C7XPXMA
2.2 180 EXCHANGE
4.1 ERR EXIT (PS CMND)
170 PROCESS
EXECUTE SEQUENCE CAUSING ERROR CONDITION.
C7XPXA-4A00
MA=C7XPXMA
C7XPXMA-4A80
MA=C7XPXMA
3.1 ERR EXIT, MF=0
3.2 ERR EXIT, MF=1
170 PROCESS
X'DEAD' → XMAFLAG
EXECUTE PS CMND
C0289
C0289
SECT 12 TO 15 - ERROR EXIT, 170 JOB


RF DUMP-0000
4. HALT
180 MONITOR
EXECUTE HALT CMND
MPS
C8MXP-5F60
VMID=0
3.2 ERROR EXIT
1. DEAD START
2.1 180 EXCHANGE
180 MONITOR
INITIALIZE 170 AND
180 JOB PACKS AND
JPS, EXCHANGE
JPS
C8JXPB-5C20
VMID=1
MF=1
MA=C7XPXMA
2.2 180 EXCHANGE
3.1 ERR EXIT, MF=1
170 PROCESS
EXECUTE SEQUENCE
CAUSING ERROR
CONDITION
C7XPXA-4A00
MA=C7XPXMA
C7XPXMA-4A80
MA=C7XPXMA
C0290
C0290
SECT 12 TO 15 - ERROR EXIT, 170 MON

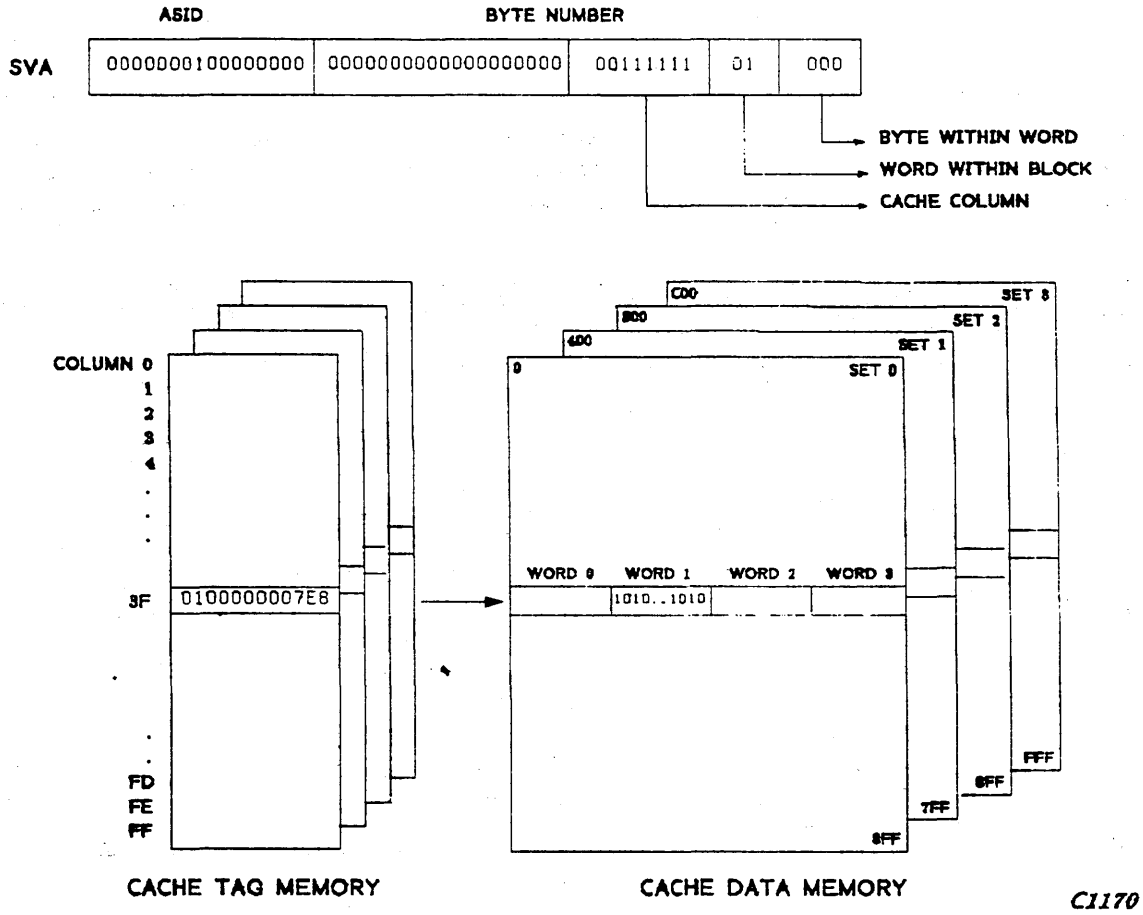
Cache Addressing and Layout D

Cache Addressing and Layout D

Given a PVA of B00F000007E8, the segment number is used to index into the Segment Table. From there, the active segment identifier (ASID) is found and concatenated with the byte portion of the PVA to form the SVA.

PVA = B00F000007E8 DATA AT B00F000007E8 = 1010101010101010

SVA = 0100000007E8



For each SVA, bits 51-58 are the hash into the cache tag memory--this is the column address. Bits 59-60 determine the word within the cache block associated with that column. Bits 61-63 indicate the byte within the word. On a write, FIF0 determines which cache set is to be loaded. On a read, a cache hit is determined by matching the SVA with the cache tag entry. There are 4,096 64-bit word locations in cache data memory. Therefore, word addresses 0-FFF fill cache completely.

Comments (continued from other side)

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

FOLD

FOLD

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

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

POSTAGE WILL BE PAID BY ADDRESSEE

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

We would like your comments on this manual to help us improve it. Please take a few minutes to fill out this form.

Who are you?

- ☐ Manager
- ☐ Systems analyst or programmer
- ☐ Applications programmer
- ☐ Operator
- ☐ Other _____

How do you use this manual?

- ☐ As an overview
- ☐ To learn the product or system
- ☐ For comprehensive reference
- ☐ For quick look-up
- ☐ Other _____

What programming languages do you use? _____

_____

How do you like this manual? Answer the questions that apply.

Yes	Somewhat	No	
☐	☐	☐	Does it tell you what you need to know about the topic?
☐	☐	☐	Is the technical information accurate?
☐	☐	☐	Is it easy to understand?
☐	☐	☐	Is the order of topics logical?
☐	☐	☐	Can you easily find what you want?
☐	☐	☐	Are there enough examples?
☐	☐	☐	Are the examples helpful? (☐ Too simple? ☐ Too complex?)
☐	☐	☐	Do the illustrations help you?
☐	☐	☐	Is the manual easy to read (print size, page layout, and so on)?
☐	☐	☐	Do you use this manual frequently?

Comments? If applicable, note page and paragraph. Use other side if needed.

Check here if you want a reply: ☐

Name _____ Company _____

Address _____ Date _____

_____ Phone _____

Please send program listing and output if applicable to your comment.

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

LITHO IN U.S.A

CONTROL DATA