CRAY T90 SERIES MME DIAGNOSTIC TESTS AND UTILITIES

Tables

Description of this Document

This document provides quick-reference information for all diagnostic tests and utilities you can use with the Mainframe Maintenance Environment (MME) application.

This document is one component of the MME documentation set, which includes the following documents:

CRAY T90 Series MME User Guide, publication number HDM-xxx-0.

This document describes using the MME environments for troubleshooting. It includes information about the internal functionality of MME.

CRAY T90 Series MME Interface Reference, publication number HDM-xxx-0.

This document describes the interfaces used with MME environments 0, 1, and 2. It also describes all available menu button commands.

CRAY T90 Series MME Diagnostic Tests and Utilities, publication number HDM-xxx-0.

This document provides quick-reference information for all diagnostic tests and utilities you can use with MME.

Notational Conventions

This document uses the following notational conventions:

- All memory addresses and section numbers are octal values.
- The –> symbol indicates holding the MENU mouse button down and moving the mouse pointer to the next menu item.
- *Italic* type indicates a variable.

ENVIRONMENT 0 TESTS

This section describes the environment 0 tests included in the offline diagnostic release. Table 1 lists the environment 0 tests and the modules that you can test with them. Run the tests in the order listed, beginning at the top of Table 1.

Table 1. Modules Tested by Environment 0 Tests

Test	Modules
BS Communication	BS
Configuration (Basic)	All
Memory	CPU
I/O Error Correction	CPU
Logic Monitor	All
Exchange	CPU
Instruction Buffer	CPU
Configuration (Advanced)	All
Miscellaneous	*Not implemented yet*

For more information about running the environment 0 tests, refer to the *CRAY T90 Series MME User Guide* document, publication number HDM-xxx-0.

BS Communication Test

The boundary scan (BS) communication test checks the communication capabilities of the boundary scan module. This test loops back module data, loops back port data, echoes the module function word, echoes the channel function word, echoes the port function word, loads the port register and reads the value, and checks for front-end interface (FEI) errors (for example, parity errors on the channel).

Configuration (Basic) Test

The configuration (basic) test checks the CPU-to-memory configuration. For each CPU, this test checks all memory group, section, subsection, and bank configuration settings for a given chassis type. This test also checks CPU memory addressing (reads and writes) when a CPU is configured in upper 256-Kword addressing mode.

The sequence for testing memory configuration settings is as follows:

1. The test configures memory with the maximum memory configuration settings for the chassis type.
2. The test patterns memory with an address pattern and uses direct memory access (DMA) transfers through the maintenance channel to verify that the pattern can be read back correctly.
3. The test asserts loop controller functions through the maintenance channel to set up the configuration being tested.
4. The test uses maintenance channel DMA, with the CPU under test providing the memory access path, to read the memory data. The test ensures the memory data is stripped according to the memory configuration settings.
5. The test repeats Steps 1 through 4 for all applicable memory group, section, subsection, and bank configuration settings.

The sequence for testing upper 256-Kword addressing mode is as follows:

1. The test configures memory with the maximum memory configuration settings for the chassis type.
2. The test patterns the upper 256 Kwords of memory with an address pattern; these memory writes start at the absolute maximum memory value minus 256 Kwords. The test uses DMA transfers through the maintenance channel to verify that the pattern can be read back correctly.
3. The test asserts loop controller functions through the maintenance channel to set up 256-Kword addressing mode.
4. The test uses maintenance channel DMA, with the 256-Kword mode CPU providing the memory access path, to verify the upper 256 Kwords of memory data.

5. The test patterns the upper 256 Kwords of memory with a compliment addressing pattern.

6. The test asserts loop controller functions through the maintenance channel to clear 256 Kword addressing mode.

7. The test uses maintenance channel DMA to read the upper 256 Kwords of memory and verifies the pattern.

Memory Test

The memory test checks mainframe memory. This test:

- Writes the MME buffer with a pattern, writes the MME buffer data to mainframe memory with error correction enabled, reads the mainframe memory data back into the MME buffer, and compares the data written to mainframe memory with the data read from mainframe memory
- Tests the block length of direct memory access (DMA) write and read function words
- Writes the MME buffer with a pattern, writes the MME buffer data to mainframe memory with error correction disabled, reads the mainframe memory data back into the MME buffer, and compares the data written to mainframe memory with the data read from mainframe memory
- Tests the address bits in a DMA function word

I/O Error-correction Test

The input/output (I/O) error-correction test checks the single-error correction/double-error detection (SECDED) and single-byte correction/double-byte detection (SBCDBD) mechanisms of the mainframe. This test:

- Writes data that forces a correctable SECDED error and verifies that the data error is corrected
- Writes data that forces an uncorrectable SECDED error and verifies that the data error is detected
- Writes various data patterns and reads and verifies the check bits

- Writes various data patterns and reads and verifies the checkbytes
- Writes a series of checkbytes, reads the checkbytes back, and verifies the checkbyte storage mechanism
- Writes data that forces a correctable SBCDBD error and verifies that the data error is corrected
- Writes data that forces an uncorrectable SBCDBD error and verifies that the data error is detected

Logic Monitor Test

The logic monitor test checks the ability of a logic monitor to record preselected values, to trigger on preselected values, and to perform breakpoint functions for preselected values. The results from these test functions are compared with expected values to ensure the logic monitor is functioning properly.

Exchange Test

The exchange test checks the exchange mechanism of the mainframe. This test sends 40_8 words of a data pattern to the MME buffer, performs a DMA transfer of the MME buffer data to mainframe memory, and exchanges the data in mainframe memory into the CPU. Then this test exchanges the data in the CPU back out to mainframe memory and performs a DMA transfer from mainframe memory to the MME buffer. Finally, this test compares the original data transferred from the MME buffer with the data received from mainframe memory to ensure the exchange mechanism is functioning properly.

NOTE: This test does not issue any instructions in the CPU(s). This test uses maintenance functions to perform the exchanges.

Instruction Buffers Test

The instruction buffers test checks the instruction buffers. This test writes the MME buffer with a data pattern, writes the MME buffer data to mainframe memory, and loads the instruction buffers from mainframe memory. Then, this test writes the contents of selected instruction buffers into mainframe memory and reads the data from mainframe memory into

the MME buffer. Finally, this section compares the original data transferred from the MME buffer with the data received from mainframe memory.

NOTE: This test does not issue any instructions in the CPU(s). This test uses maintenance functions to manipulate the instruction buffers.

Configuration (Advanced) Test

The configuration (advanced) test checks I/O group and shared group configuration settings and interprocessor interrupts within shared groups for each CPU. This test runs a Cray Assembly Language (CAL) program, called cfa.t, in the mainframe. This test sets parameters for the cfa.t program, deadstarts CPUs, and analyzes the results, which are stored in mainframe memory.

The cfa.t program has parameters for I/O group testing, shared group testing, and interprocessor testing. The configuration (advanced) test sequence modifies these parameters to test the modules you select.

The result of the I/O group testing is a table of LOSP and VHISP channels that responded to the channel status reads. The result of the shared group testing is a table of SB00 data from register read attempts of all selected clusters. The result of the interprocessor testing is a table of CPU numbers that indicates which CPUs responded to the interprocessor interrupts.

Miscellaneous Test

This test is not implemented yet.

ENVIRONMENTS 1 AND 2 TESTS AND UTILITIES

This section describes the environment 1 and environment 2 diagnostic tests and utilities included in the offline diagnostic release.

Diagnostic Tests

Table 2 lists the diagnostics you can use to test the various functional units and registers in a CRAY T90 series mainframe.

Table 2. Diagnostics Used to Test Registers and Functional Units

Register or Functional Unit	Diagnostic Test(s)
Address registers	asb.t asf.t
Scalar registers	asb.t asf.t svb.t svf.t
Address and scalar add functional units	asb.t asf.t
B and T registers	btv.t
Address and scalar logical functional units	asb.t asf.t
Address and scalar mask functional units	asb.t asf.t
Address and scalar population, parity, and leading zero functional units	asb.t asf.t
A register shift functional unit	asb.t asf.t
S register shift functional unit	asb.t asf.t
Address multiply functional unit	asb.t asf.t
Integer multiply functional unit	fpb.t (section 6)
Vector registers	btv.t svb.t svf.t

Table 2. Diagnostics Used to Test Registers and Functional Units (continued)

Register or Functional Unit	Diagnostic Test(s)
Vector logical functional unit	svb.t svf.t
Iota and vector mask functional units	fpb.t svb.t
Vector add functional unit	svb.t svf.t
Vector shift functional unit	svb.t svf.t
Gather and scatter functional units	btv.t (sections 21, 22, 23)
Floating add functional unit	fpb.t fpt.t
Reciprocal approximation functional unit	fpb.t fpt.t
Floating multiply functional unit	fpb.t fpt.t
Bit matrix multiply (BMM) functional unit	fpb.t svf.t
Exchange	eft.t ept.t
Fetch	eft.t fbt.t
Issue	ctt.t rict.t rit.t
Memory control	cmt.t mpct.t smt.t
Logical address translation (LAT)	lat.t
Cache	cach.t
Performance monitor	pmt.t
Shared registers	mcst.t srt.t srct.t
LOSP channel	etem.t lsp.t
VHISP channel	vhs.t
CRAY C90 mode	cmp.t

Table 3 lists all available diagnostic tests you can load as control points in environments 1 and 2. This table provides a quick-reference description of each test section. Refer to the actual listings for detailed information about the tests.

For more information about viewing the listings, refer to "View –> Listing –> Current" and "View –> Listing –> Other" in the "Environments 1 and 2" section of the *CRAY T90 Series MME Interface Reference* document, publication number HDM-xxx-0.

NOTE: Any instruction preceded by EIS in Table 3 is part of the extended instruction set. For more information about any of the instructions listed in Table 3, refer to the *CRAY T90 Series Instruction Descriptions* document, publication number HTM-xxx-0.

Table 3. Environments 1 and 2 Diagnostic Tests

Test†	Description	Section Information			
		Number	Filename	Env‡	Function
asb.t	Basic A and S register chip and path test	0	asb00.t	1 2	Tests the basic error check and operand read routines
		1	asb01.t	1 2	Tests the following instructions: 100*i*00*nm* A*i* *nm*,0 110*i*00*nm* *nm*,0 A*i* 120*i*00*nm* S*i* *nm*,0 130*i*00*nm* *nm*,0 S*i*
		2	asb02.t	1 2	Tests the following instruction: 022*ijk* A*i* exp
		3	asb03.t	1 2	Tests the following instructions: 046*ijk* S*i* S*j*\S*k* EIS 046*ijk* A*i* A*j*\A*k*
		4	asb04.t	1 2	Tests the following instruction: 015000*nm* JSN exp
		5	asb05.t	1 2	Tests the following instructions: 030*ijk* A*i* A*j*+A*k* 031*ijk* A*i* A*j*–A*k*

† A `.t` extension indicates the test program was assembled in Triton mode.

‡ The Env column indicates which environment a section runs in: 1 indicates the section runs in environment 1 only; 1 2 indicates the section runs in environment 1 or 2.

Table 3. Environments 1 and 2 Diagnostic Tests (continued)

Test †	Description	Section Information			
		Number	Filename	Env ‡	Function
asb.t (cont.)	Basic A and S register chip and path test	6	asb06.t	[1][2]	Tests the following instructions: 060*ijk* Si Sj+Sk 061*ijk* Si Sj–Sk
		7	asb07.t	[1][2]	Tests the following instructions: 020*i*00*nm* Ai exp 020*i*20*nm* Ai :exp 020*i*40*nm* :exp Ai 021*i*00*nm* Ai exp
		10	asb10.t	[1][2]	Tests the following instructions: 040*i*00*nm* Si exp 040*i*20*nm* Si exp 040*i*40*nm* Si exp 041*i*00*nm* Si exp
		11	asb11.t	[1][2]	Tests the following instructions: 010000*nm* JAZ exp 011000*nm* JAN exp 012000*nm* JAP exp 013000*nm* JAM exp
		12	asb12.t	[1][2]	Tests the following instructions: 014000*nm* JSZ exp 016000*nm* JSP exp 017000*nm* JSM exp
		13	asb13.t	[1][2]	Tests the following instruction: 023*ij*0 Ai Sj
		14	asb14.t	[1][2]	Tests the following instructions: 071*i*0*k* Si Ak 071*i*1*k* Si +Ak
		15	asb15.t	[1][2]	Tests the following instructions: 024*ijk* Ai Bjk 025*ijk* Bjk Ai

† A `.t` extension indicates the test program was assembled in Triton mode.

‡ The Env column indicates which environment a section runs in: [1] indicates the section runs in environment 1 only; [1][2] indicates the section runs in environment 1 or 2.

Table 3. Environments 1 and 2 Diagnostic Tests (continued)

Test †	Description	Section Information			
		Number	Filename	Env ‡	Function
asb.t (cont.)	Basic A and S register chip and path test	16	asb16.t	1 2	Tests the following instructions: 074*ijk* S*i* T*jk* 075*ijk* T*jk* S*i*
		17	asb17.t	1 2	Tests the following instruction: 032*ijk* A*i* A*j**Ak
		20	asb20.t	1 2	Tests the following instructions: EIS 044*ijk* A*i* A*j*&A*k* EIS 045*ijk* A*i* #A*k*&A*j* EIS 047*ijk* A*i* #A*j*\A*k* EIS 050*ijk* A*i* A*j*!A*i*&A*k* EIS 051*ijk* A*i* A*j*!A*k*
		21	asb21.t	1 2	Tests the following instructions: 044*ijk* S*i* S*j*&S*k* 045*ijk* S*i* #S*k*&S*j* 047*ijk* S*i* #S*j*\S*k* 050*ijk* S*i* S*j*!S*i*&S*k* 051*ijk* S*i* S*j*!S*k*
		22	asb22.t	1 2	Tests the following instructions; EIS 052*ijk* A0 A*i*<*jk* EIS 053*ijk* A0 A*i*>*jk* EIS 054*ijk* A*i* A*i*<*jk* EIS 055*ijk* A*i* A*i*>*jk*
		23	asb23.t	1 2	Tests the following instructions: EIS 056*ijk* A*i* A*i*,A*j*<A*k* EIS 057*ijk* A*i* A*i*,A*j*>A*k*
		24	asb24.t	1 2	Tests the following instructions: 052*ijk* S0 S*i*<*jk* 053*ijk* S0 S*i*>*jk* 054*ijk* S*i* S*i*<*jk* 055*ijk* S*i* S*i*>*jk*
		25	asb25.t	1 2	Tests the following instructions: 056*ijk* S*i* S*i*,S*j*<S*k* 057*ijk* S*i* S*i*,S*j*>S*k*

† A `.t` extension indicates the test program was assembled in Triton mode.

‡ The Env column indicates which environment a section runs in: 1 indicates the section runs in environment 1 only; 1 2 indicates the section runs in environment 1 or 2.

Table 3. Environments 1 and 2 Diagnostic Tests (continued)

Test †	Description	Section Information			
		Number	Filename	Env‡	Function
asb.t (cont.)	Basic A and S register chip and path test	26	asb26.t	1 2	Tests the following instructions: EIS 042*ijk* A*i* <exp EIS 043*ijk* A*i* >exp
		27	asb27.t	1 2	Tests the following instructions: 042*ijk* S*i* <exp 043*ijk* S*i* >exp
		30	asb30.t	1 2	Tests the following instructions: 026*ij*2 A*i* PA*j* 026*ij*3 A*i* QA*j* 027*ij*1 A*i* ZA*j* 026*ij*0 A*i* PS*j* 026*ij*1 A*i* QS*j* 027*ij*0 A*i* ZS*j*
asf.t	Comprehensive A and S register functional unit test	0	asf00.t	1 2	Tests the logical functional unit and the A and S register paths to it
		1	asf01.t	1 2	Tests the add functional unit and the A and S register paths to it
		2	asf02.t	1 2	Tests the shift functional unit and the A and S register paths to it
		3	asf03.t	1 2	Tests the population, parity, and leading zero count functional units and the A and S register paths to them
		4	asf04.t	1	Tests the address multiply functional unit and the A register paths to it; tests address multiply interrupts
		5	asf05.t	1 2	Tests a randomly chosen functional unit (logical, add, shift, population, parity, leading zero, or address multiply functional unit)
btv.t	B, T, and V register integrity and port test	0	btv00.t	1 2	Tests the A to B register-to-register paths and B register chip integrity
		1	btv01.t	1 2	Tests the S to T register-to-register paths and T register chip integrity
		2	btv02.t	1 2	Tests the S to V register-to-register paths and V register chip integrity

† A `.t` extension indicates the test program was assembled in Triton mode.

‡ The Env column indicates which environment a section runs in: 1 indicates the section runs in environment 1 only; 1 2 indicates the section runs in environment 1 or 2.

Table 3. Environments 1 and 2 Diagnostic Tests (continued)

Test †	Description	Section Information			
		Number	Filename	Env ‡	Function
btv.t (cont.)	B, T, and V register integrity and port test	3	btv03.t	1 2	Tests a random selection of A, B, S, T, or V register paths and chip integrity
		4	btv04.t	1 2	Tests port A addressing, data, and control using B register block transfers
		5	btv05.t	1 2	Tests port C addressing, data, and control using B register block transfers; tests B register array addressing, data, and control with B write references
		6	btv06.t	1 2	Tests B register array parity generation, data and parity storage, and parity checking
		7	btv07.t	1 2	Tests port A addressing, data, and control using T register block transfers
		10	btv10.t	1 2	Tests port C addressing, data, and control using T register block transfers; tests T register array addressing, data, and control with T write references
		11	btv11.t	1 2	Tests T register array parity generation, data and parity storage, and parity checking
		12	btv12.t	1 2	Tests ports A, B, and C paths with concurrent port activity
		13	btv13.t	1 2	Tests ports A and A' addressing, data, and control using V register block reads; tests V register addressing, data, and control with V read references
		14	btv14.t	1 2	Tests ports B and B' addressing, data, and control using V register block reads; tests V register addressing, data, and control with V read references
		15	btv15.t	1 2	Tests ports C and C' addressing, data, and control using V register block writes; tests V register addressing, data, and control with V read references
		16	btv16.t	1 2	Tests V reference chaining control and concurrent V array reads and writes
		17	btv17.t	1 2	Tests ports A, B, and C paths and control with concurrent port activity

† A `.t` extension indicates the test program was assembled in Triton mode.

‡ The Env column indicates which environment a section runs in: 1 indicates the section runs in environment 1 only; 1 2 indicates the section runs in environment 1 or 2.

Table 3. Environments 1 and 2 Diagnostic Tests (continued)

Test†	Description	Section Information			
		Number	Filename	Env‡	Function
btv.t (cont.)	B, T, and V register integrity and port test	20	btv20.t	1 2	Tests V register array parity generation, data and parity storage, and parity checking
		21	btv21.t	1 2	Tests ports A and A' gather V*k* address and V*j* data paths, stack, and control
		22	btv22.t	1 2	Tests ports B and B' gather V*k* address and V*j* data paths, stack, and control
		23	btv23.t	1 2	Tests ports C and C' scatter V*k* address and V*j* data paths, stack, and control
		24	btv24.t	1 2	Tests ports A, B, and C paths and control with concurrent port activity (concurrent gather and scatter reference test)
		25	btv25.t	1 2	Tests ports A and A' and B and B' double gather (A0, A*k*, and V*k* address paths; V*k* stack; V*i* data paths; and control)
		26	btv26.t	1 2	Tests breakpoint base and limit registers, breakpoint and logical address compare, and breakpoint interrupt
		27	btv27.t	1 2	Tests randomly selected port A, B, or C paths and control with concurrent port activity
cach.t	Cache test	0	cach00.t	1 2	Tests ability to enable and disable the scalar cache by using the exchange package global cache enable mode, by using the 002501 ESC and 002601 DSC instructions, and by using the exchange package logical address table (LAT) cache enable modes
		1	cach01.t	1 2	Verifies that scalar cache is invalidated by an exchange and by the 002601 DSC instruction
		2	cach02.t	1 2	Tests the cache using combinations of scalar load and store functions

† `A.t` extension indicates the test program was assembled in Triton mode.

‡ The Env column indicates which environment a section runs in: 1 indicates the section runs in environment 1 only; 1 2 indicates the section runs in environment 1 or 2.

Table 3. Environments 1 and 2 Diagnostic Tests (continued)

Test †	Description	Section Information			
		Number	Filename	Env‡	Function
cach.t (cont.)	Cache test	3	cach03.t	[1][2]	Tests the short and long scalar load (bypasses data cache) with the following instructions: 10*hi*20*nm* A*i* exp,A*h*,BC 10*hi*60*pnm* A*i* exp,A*h*,BC 12*hi*20*nm* S*i* exp,A*h*,BC 12*hi*60*pnm* S*i* exp,A*h*,BC
		4	cach04.t	[1][2]	Tests cache interaction with block (B, T, and V registers) load and store functions
		5	cach05.t	[1][2]	Tests the page register line valids
		6	cach06.t	[1]	Tests page register address compare
		7	cach07.t	[1][2]	Tests basic functionality of cache performance monitors (which count cache hits and misses)
		10	cach10.t	[1][2]	Tests data integrity of the cache random access memory (RAM) array
		11	cach11.t	[1]	Tests the destination code in the SR4 status register for error-correcting code (ECC) errors logged for cache reads (runs in maintenance mode)
		12	cach12.t	[1]	Tests ability to enable and disable cache parity maintenance mode; verifies that parity is forced and parity errors are reported correctly from each cache port (runs in maintenance mode)
cct.t	CPU clock test	0	cct00.t	[1]	Tests the real-time clock (RTC) write and read paths, the RTC +1 adder, and the RTC monitor mode write protection
		1	cct01.t	[1]	Tests the load, countdown, and reload functions of the programmable clock and tests the programmable clock interrupts and related instructions
cmp.t	C90 mode compatibility test	0	cmp00.t	[1][2]	Tests the C90 mode instructions and verifies compatibility with the CRAY C90 instruction set

† `A.t` extension indicates the test program was assembled in Triton mode.

‡ The Env column indicates which environment a section runs in: [1] indicates the section runs in environment 1 only; [1][2] indicates the section runs in environment 1 or 2.

Table 3. Environments 1 and 2 Diagnostic Tests (continued)

Test †	Description	Section Information			
		Number	Filename	Env ‡	Function
cmt.t	Comprehensive memory test	0	cmt00.t	[1]	Tests CPU port memory maintenance modes [substitute V*k* to checkbytes (CBs), disable single-byte correction (SBC), substitute CBs to V*i*, disable error log, and input port resume (runs in maintenance mode)]
		1	cmt01.t	[1]	Tests correctable and uncorrectable memory interrupts and interrupt modes (runs in maintenance mode)
		2	cmt02.t	[1]	Tests memory error-reporting stacks in each memory section; tests the memory error status path from each section to the SR4, SR5, and SR6 status registers; and tests the path from the status registers to each S register (runs in maintenance mode)
		3	cmt03.t	[1]	Tests the check-bit read and write paths to and from memory; tests the ports C and C' write check-bit generation and sections 0 through 7 check-bit generation (runs in maintenance mode)
		4	cmt04.t	[1]	Tests error detection and correction in all memory sections (runs in maintenance mode)
		5	cmt05.t	[1]	Tests data, addressing, and control paths and stacks on the memory and network modules (runs in maintenance mode)
		6	cmt06.t	[1]	Tests check-bit memory and network paths and stacks; tests check-bit memory storage (runs in maintenance mode)
		7	cmt07.t	[1]	Tests memory storage (runs in maintenance mode)
ctt.t	CPU timing test	0	ctt00.t	[1][2]	Verifies correct operation of a CPU through simulated timings of CPU instruction issue

† A `.t` extension indicates the test program was assembled in Triton mode.

‡ The Env column indicates which environment a section runs in: [1] indicates the section runs in environment 1 only; [1][2] indicates the section runs in environment 1 or 2.

Table 3. Environments 1 and 2 Diagnostic Tests (continued)

Test †	Description	Section Information			
		Number	Filename	Env ‡	Function
ctt.t (cont.)	CPU timing test	1	ctt01.t	[1]	Tests memory reference timing [verifies proper timing for memory, memory ports, and port arbitration (runs in maintenance mode)]
		2	ctt02.t	[1]	Tests exchange, jump, and fetch timing (runs in maintenance mode)
		3	ctt03.t	[1]	Tests interrupt timing (runs in maintenance mode)
		4	ctt04.t	[1]	Tests shared register instruction timing
eft.t	Exchange fetch test	0	eft00.t	[1]	Tests an exchange and fetch with conflicts (runs in maintenance mode)
		1	eft01.t	[1]	Tests instruction buffer parity and data integrity (runs in maintenance mode)
		2	eft02.t	[1]	Tests wait exchange conditions (runs in maintenance mode)
		3	eft03.t	[1]	Verifies that exchange data is valid before use (runs in maintenance mode)
		4	eft04.t	[1][2]	Tests an exchange on multiple-parcel instructions (runs in maintenance mode)
		5	eft05.t	[1]	Tests an exchange with concurrent interrupts (runs in maintenance mode)
		6	eft06.t	[1][2]	Tests the vector not used (VNU) bit (runs in maintenance mode)
ept.t	Exchange parameters test	0	ept00.t	[1]	Checks the exit address (XA) and exchange address (EA) registers read and write instruction functionality; verifies the EA and XA registers data integrity; verifies normal and exit instructions; and checks the EA and XA address paths to and from memory
		1	ept01.t	[1]	Tests the exchange paths to and from memory
		2	ept02.t	[1]	Tests instructions that modify exchange parameters

† A `.t` extension indicates the test program was assembled in Triton mode.

‡ The Env column indicates which environment a section runs in: [1] indicates the section runs in environment 1 only; [1][2] indicates the section runs in environment 1 or 2.

Table 3. Environments 1 and 2 Diagnostic Tests (continued)

Test †	Description	Section Information			
		Number	Filename	Env‡	Function
ept.t (cont.)	Exchange parameters test	3	ept03.t	[1]	Tests instruction decode of monitor instruction interrupt and nonmonitor instruction interrupt opcodes; verifies that interrupt on monitor instruction (IMI) is disabled when monitor mode (MM) is set
		4	ept04.t	[1]	Tests a short series of exchange and random exchange parameter instruction sequences
etem.t	End-to-end mainframe LOSP test	0	etem.t	[1][2]	Produces random activity on LOSP channel pairs connected to an IOS-E or CRAY T3D system (rotates under the run system)
fbt.t	Fetch and branch test	0	fbt00.t	[1]	Tests instruction buffers 0 through 7 basic addressing and control
		1	fbt01.t	[1]	Tests instruction stack addressing and control
		2	fbt02.t	[1]	Tests instruction buffer in-stack and out-of-stack control
		3	fbt03.t	[1]	Tests prefetch control
		4	fbt04.t	[1]	Tests multiple-parcel instructions that are split across several instruction buffers
		5	fbt05.t	[1]	Tests the ability of the instruction stack chips to hold random data
		6	fbt06.t	[1]	Tests in-stack and out-of-stack jumps
		7	fbt07.t	[1]	Tests branch conditional control
		10	fbt10.t	[1][2]	Tests the P register and the fetch address path to memory
		11	fbt11.t	[1][2]	Tests the P register +1 and prefetch +1 adder paths and functionality
		12	fbt12.t	[1][2]	Tests the jump expression address path to the P register
		13	fbt13.t	[1][2]	Tests B register chip addressing and data integrity
		14	fbt14.t	[1][2]	Tests B register-to-P register path

† A `.t` extension indicates the test program was assembled in Triton mode.

‡ The Env column indicates which environment a section runs in: [1] indicates the section runs in environment 1 only; [1][2] indicates the section runs in environment 1 or 2.

Table 3. Environments 1 and 2 Diagnostic Tests (continued)

Test†	Description	Section Information			
		Number	Filename	Env‡	Function
fbt.t (cont.)	Fetch and branch test	15	fbt15.t	1 2	Tests P register-to-B00 path on return jumps
		16	fbt16.t	1 2	Tests the indirect jump reference address path to memory
		17	fbt17.t	1 2	Tests the indirect jump address path to the P register
		20	fbt20.t	1 2	Tests P register-to-B00 path on indirect return jump
		21	fbt21.t	1 2	Verifies that B00 remains unchanged after executing all jumps that are not return jumps
		22	fbt22.t	1 2	Tests the instruction buffer address registers and the coincidence logic
		23	fbt23.t	1	Verifies that exchanges, the 005100 J B*jk* instruction, and indirect jumps invalidate the instruction buffer address registers (runs in maintenance mode)
		24	fbt24.t	1 2	Tests random in-stack and out-of-stack jump sequences (stationary buffers)
		25	fbt25.t	1 2	Tests random in-stack and out-of-stack jump sequences (dynamic buffers)
fpb.t	Basic floating-point and bit matrix multiply chip and path test	0	fpb00.t	1	Tests the following instructions: 071*i*40 S*i* 0.4 071*i*30 S*i* 0.6 071*i*50 S*i* 1.0 071*i*60 S*i* 2.0 071*i*70 S*i* 4.0
		1	fpb01.t	1	Tests the following instruction: 071*i*2*k* S*i* +FA*k*
		2	fpb02.t	1	Tests the following instructions: 062*ijk* S*i* S*j*+FS*k* 063*ijk* S*i* S*j*–FS*k*

† A `.t` extension indicates the test program was assembled in Triton mode.

‡ The Env column indicates which environment a section runs in: 1 indicates the section runs in environment 1 only; 1 2 indicates the section runs in environment 1 or 2.

Table 3. Environments 1 and 2 Diagnostic Tests (continued)

Test †	Description	Section Information			
		Number	Filename	Env‡	Function
fpb.t (cont.)	Basic floating-point and bit matrix multiply chip and path test	3	fpb03.t	[1]	Tests the following instructions: 170*ijk* V*i* S*j*+FV*k* 172*ijk* V*i* S*j*–FV*k* 171*ijk* V*i* V*j*+FV*k* 173*ijk* V*i* V*j*–FV*k*
		4	fpb04.t	[1]	Tests the following instructions: 064*ijk* S*i* S*j**FS*k* 065*ijk* S*i* S*j**HS*k* 066*ijk* S*i* S*j**RS*k* 067*ijk* S*i* S*j**IS*k*
		5	fpb05.t	[1]	Tests the following instruction sequence: Express jump 062*ijk* S0 S*j*+FS*k* 017*ijkm* JSM exp 064*ijk* S0 S*j**FS*k* 017*ijkm* JSM exp **NOTE:** This section tests the capability to perform these instructions in sequence. It does not test the express jump timing.
		6	fpb06.t	[1]	Tests the following instruction: 166*ijk* V*i* S*j**V*k*
		7	fpb07.t	[1]	Tests the following instructions: 160*ijk* V*i* S*j**FV*k* 162*ijk* V*i* S*j**HV*k* 164*ijk* V*i* S*j**RV*k*
		10	fpb10.t	[1]	Tests the following instructions: 161*ijk* V*i* V*j**FV*k* 163*ijk* V*i* V*j**HV*k* 165*ijk* V*i* V*j**RV*k* 167*ijk* V*i* V*j**IV*k*

† A `.t` extension indicates the test program was assembled in Triton mode.

‡ The Env column indicates which environment a section runs in: [1] indicates the section runs in environment 1 only; [1][2] indicates the section runs in environment 1 or 2.

Table 3. Environments 1 and 2 Diagnostic Tests (continued)

Test †	Description	Section Information			
		Number	Filename	Env‡	Function
fpb.t (cont.)	Basic floating-point and bit matrix multiply chip and path test	11	fpb11.t	[1]	Tests the following instructions: 174*ij*1 V*i* PV*j* 174*ij*2 V*i* QV*j* 174*ij*3 V*i* ZV*j*
		12	fpb12.t	[1]	Tests the following instructions: 070*ij*0 S*i* /HS*j* 174*ij*0 V*i* /HV*j*
		13	fpb13.t	[1]	Tests the following instruction: 070*ij*1 V*i* CI,S*j*&VM
fpt.t	Floating-point test	0	fpt00.t	[1][2]	Tests floating-point addition (canned and simulated answers)
		1	fpt01.t	[1][2]	Tests floating-point multiplication (canned and simulated answers)
		2	fpt02.t	[1][2]	Tests floating-point reciprocal functionality (canned and simulated answers)
		3	fpt03.t	[1][2]	Tests floating-point addition (comprehensive: actual results compared to simulated results)
		4	fpt04.t	[1][2]	Tests floating-point integer multiplication (comprehensive: actual results compared to simulated results)
		5	fpt05.t	[1][2]	Tests floating-point multiplication (comprehensive: actual results compared to simulated results)
		6	fpt06.t	[1][2]	Tests floating-point reciprocal (comprehensive: actual results compared to simulated results)
		7	fpt07.t	[1][2]	Tests chained functional units (simulated answers)
		10	fpt10.t	[1][2]	Tests floating-point range errors
		11	fpt11.t	[1][2]	Performs vector population, parity, and leading zero tests (comprehensive: actual results compared to simulated results)

† A `.t` extension indicates the test program was assembled in Triton mode.

‡ The Env column indicates which environment a section runs in: [1] indicates the section runs in environment 1 only; [1][2] indicates the section runs in environment 1 or 2.

Table 3. Environments 1 and 2 Diagnostic Tests (continued)

Test †	Description	Section Information			
		Number	Filename	Env‡	Function
fpt.t (cont.)	Floating-point test	12	fpt12.t	1 2	Performs comprehensive IOTA test
		13	fpt13.t	1 2	Tests floating-point addition and multiplication express jumps (checks express jump sign bits with timing test)
		14	fpt14.t	1	Tests floating-point interrupt enable and disable instructions, interrupt mode bits, and interrupt status bit
		15	fpt15.t	1 2	Tests floating-point addition, multiplication, reciprocal, IOTA, vector population, vector parity, and vector leading zero functional units (random selection)
lat.t	Logical address translation (LAT) test	0	lat00.t	1 2	Verifies that each exchange package LAT entry can be enabled for execute, read, and write modes; verifies that the LAT dirty bits are set properly
		1	lat01.t	1 2	Verifies that each exchange package LAT entry can be disabled for execute, read, and write modes
		2	lat02.t	1	Tests each bit in the logical base of the execute LATs for coincidence
		3	lat03.t	1	Tests each bit in the logical base of the execute LATs for no-coincidence
		4	lat04.t	1	Tests each bit in the logical base of the read and write LATs for coincidence (each port is tested separately)
		5	lat05.t	1	Tests each bit in the logical base of the read and write LATs for no-coincidence (each port is tested separately)
		6	lat06.t	1	Tests each bit in the logical limit of the execute LATs for coincidence
		7	lat07.t	1	Tests each bit in the logical limit of the execute LATs for no-coincidence
		10	lat10.t	1	Tests each bit in the logical limit of the read and write LATs for coincidence (each port is tested separately)

† A `.t` extension indicates the test program was assembled in Triton mode.

‡ The Env column indicates which environment a section runs in: 1 indicates the section runs in environment 1 only; 1 2 indicates the section runs in environment 1 or 2.

Table 3. Environments 1 and 2 Diagnostic Tests (continued)

Test †	Description	Section Information			
		Number	Filename	Env‡	Function
lat.t (cont.)	Logical address translation (LAT) test	11	lat11.t	1	Tests each bit in the logical limit of the read and write LATs for no-coincidence (each port is tested separately)
		12	lat12.t	1	Tests each bit in the physical bias for the execute LATs and tests the bias adder
		13	lat13.t	1	Tests each bit in the physical bias for the read and write LATs (each port is tested separately) and tests the bias adder
		14	lat14.t	1 2	Tests all combinations of overlapping execute LATs to force multiple LAT errors
		15	lat15.t	1 2	Tests all combinations of overlapping read and write LATs to force multiple LAT errors (each port is tested separately)
		16	lat16.t	1 2	Tests the program range error for various combinations of interrupt enables
		17	lat17.t	1 2	Tests the operand range error for various combinations of interrupt enables
		20	lat20.t	1	Verifies that a block transfer instruction works correctly when the memory block spans multiple LATs
		21	lat21.t	1 2	Tests random LATs with random code (tests everything that sections 0 through 20 test)
lsp.t	LOSP channel test	0	lsp00.t	1	Tests basic functionality of LOSP channel control, data paths, and simple addressing
		1	lsp01.t	1	Tests LOSP channel interrupts on a single CPU
		2	lsp02.t	1	Tests the LOSP channel addressing
		3	lsp03.t	1	Tests the LOSP channel limits and large transfer lengths
		4	lsp04.t	1	Tests special-case LOSP channel transfers

† A `.t` extension indicates the test program was assembled in Triton mode.

‡ The Env column indicates which environment a section runs in: 1 indicates the section runs in environment 1 only; 1 2 indicates the section runs in environment 1 or 2.

Table 3. Environments 1 and 2 Diagnostic Tests (continued)

Test †	Description	Section Information			
		Number	Filename	Env‡	Function
lsp.t (cont.)	LOSP channel test	5	lsp05.t	1	Tests the 033 instruction source and destination timers on the shared module(s) (runs in single or multiple CPUs)
		6	lsp06.t	1	Verifies that privileged channel instructions executed in user mode cannot access the channel hardware
		7	lsp07.t	1	Tests SBDDBD error correction and reporting, tests parity associated with the LOSP channels, and tests error reporting to the CPU status registers and the error logger (runs in single or multiple CPUs and runs in maintenance mode)
		10	lsp10.t	1	Tests LOSP channel multiple-CPU interrupts (runs in single or multiple CPUs)
		11	lsp11.t	1	Tests LOSP channel pairs concurrently with background shared interference code in all clusters (runs in single or multiple CPUs)
		12	lsp12.t	1	Tests LOSP channel pairs separately and concurrently with background interference code on all memory ports of all CPUs (runs in single or multiple CPUs)
		13	lsc.t	1 2	Produces random activity on selected LOSP channel pairs (rotates under the run system)
		14	chn.t	1 2	Produces random activity on selected LOSP and VHISP channels with memory port interference (runs in single or multiple CPUs and rotates under the run system)
mcst.t	Mutiple-CPU startup test	0	mcst00.t	1	Verifies the 0014*j*1 SIPI and 001402 CIPI instructions with multiple CPUs (runs in multiple CPUs)
		1	mcst01.t	1	Verifies the real-time clock with multiple CPUs (runs in multiple CPUs)

† A `.t` extension indicates the test program was assembled in Triton mode.

‡ The Env column indicates which environment a section runs in: 1 indicates the section runs in environment 1 only; 1 2 indicates the section runs in environment 1 or 2.

Table 3. Environments 1 and 2 Diagnostic Tests (continued)

Test †	Description	Section Information			
		Number	Filename	Env‡	Function
mcst.t (cont.)	Mutiple-CPU startup test	2	mcst02.t	[1]	Tests network and memory module conflicts with multiple CPUs (runs in multiple CPUs)
		3	mcst03.t	[1]	Tests shared module conflicts with multiple CPUs (runs in multiple CPUs)
		4	mcst04.t	[1]	*Not implemented yet*
		5	mcst05.t	[1]	Tests multiple-CPU error logging hardware (runs in maintenance mode and in multiple CPUs)
		6	mcst06.t	[1]	Verifies that multiple CPUs can perform basic scalar memory references without errors (runs in multiple CPUs)
mpct.t	Memory and port confidence test	0	mpct00.t	[1][2]	Tests memory and network data and address paths (runs in single or multiple CPUs and runs in maintenance mode)
		1	mpct01.t	[1][2]	Tests memory storage (runs in single or multiple CPUs)
		2	mpct02.t	[1][2]	Tests CPU port data and address paths and control with multiple-CPU concurrent port activity [uses ports A, B, and C and runs in multiple CPUs (runs in single or multiple CPUs)]
		3	mpct03.t	[1]	Tests CPU exchange and fetch port data and address path with multiple-CPU concurrent port activity [uses ports D and E (runs in single or multiple CPUs)]
		4	mpct04.t	[1]	Tests random port activity in a multiple CPU environment (runs in single or multiple CPUs)
pmt.t	Performance monitor test	0	pmt00.t	[1]	Verifies that all performance monitor counters can be cleared (runs single or multiple CPUs)
		1	pmt01.t	[1]	Verifies functionality of performance monitor 5, subcounter 5 (runs in maintenance mode)

† `A.t` extension indicates the test program was assembled in Triton mode.

‡ The Env column indicates which environment a section runs in: [1] indicates the section runs in environment 1 only; [1][2] indicates the section runs in environment 1 or 2.

Table 3. Environments 1 and 2 Diagnostic Tests (continued)

Test †	Description	Section Information			
		Number	Filename	Env‡	Function
pmt.t (cont.)	Performance monitor test	2	pmt02.t	1	Verfies that counters can increment in maintenance mode (runs in single or multiple CPUs and runs in maintenance mode)
rict.t	Random instruction conflict test	0	rict00.t	1	Tests randomly selected instructions in a multiple CPU environment (runs in single or multiple CPUs and runs in maintenance mode)
rit.t	Random instruction test	0	rit00.t	1 2	Tests for register and scalar (1-, 2-, and 3-parcel) conflicts using random operands (rotates under the run system)
		1	rit01.t	1 2	Tests for register and scalar/vector instruction (3 parcel) conflicts using random operands (rotates under the run system)
		2	rit02.t	1 2	Performs single-CPU random instruction test (rotates under the run system)
smt.t	Scalar memory test	0	smt00.t	1 2	Performs bank address path test (runs in maintenance mode)
		1	smt01.t	1 2	Performs bank data path test (runs in maintenance mode)
		2	smt02.t	1 2	Performs chip address path test [A*h* to port adder, EXP to port adder, and CPU to memory module addressing (runs in maintenance mode)]
		3	smt03.t	1 2	Tests memory chip data integrity (runs in maintenance mode)
		4	smt04.t	1 2	Tests the scalar memory reference port (port C) adder (runs in maintenance mode)
		5	smt05.t	1 2	Tests the scalar memory reference port (port C) adder using A and S register reads and random operands (runs in maintenance mode)

† A `.t` extension indicates the test program was assembled in Triton mode.

‡ The Env column indicates which environment a section runs in: 1 indicates the section runs in environment 1 only; 1 2 indicates the section runs in environment 1 or 2.

Table 3. Environments 1 and 2 Diagnostic Tests (continued)

Test †	Description	Section Information			
		Number	Filename	Env ‡	Function
srct.t	Shared register comprehensive test	0	srct00.t	1 2	Tests master CPU write SM register data, slave CPU read data, and test and set on four possible states of paired SM registers (runs in single or multiple CPUs)
		1	srct01.t	1 2	Tests arbitration of SB and ST register operations (runs in multiple CPUs)
		2	srct02.t	1 2	Tests arbitration of SM register operations (runs in multiple CPUs)
		3	srct03.t	1 2	Tests cluster attachment and detachment (runs in single or multiple CPUs)
		4	srct04.t	1 2	Tests a random shared instruction sequence (runs in multiple CPUs)
		5	srct05.t	1 2	Tests IP, DL, and PC interrupts while random shared register instruction sequences are running (runs in multiple CPUs)
srt.t	Basic shared register data test	0	srt00.t	1 2	Tests ability of shared B registers, shared T registers, and semaphore registers to hold basic data patterns [0's and 1's (rotates under the run system)]
		1	srt01.t	1	Tests the SR option's RAM array parity checking and parity error reporting (rotates under the run system)
		2	srt02.t	1 2	Tests cluster addressing for shared B registers, shared T registers, and semaphore registers (rotates under the run system)
		3	srt03.t	1 2	Tests shared B register fetch and increment instructions (rotates under the run system)
		4	srt04.t	1 2	Tests semaphore set and clear instructions (rotates under the run system)
		5	srt05.t	1 2	Tests jump test and set instructions (rotates under the run system)

† `A.t` extension indicates the test program was assembled in Triton mode.

‡ The Env column indicates which environment a section runs in: 1 indicates the section runs in environment 1 only; 1 2 indicates the section runs in environment 1 or 2.

Table 3. Environments 1 and 2 Diagnostic Tests (continued)

Test †	Description	Section Information			
		Number	Filename	Env‡	Function
srt.t (cont.)	Basic shared register data test	6	srt06.t	[1][2]	Tests test and set, deadlock, and WS exchange flag (rotates under the run system)
		7	srt07.t	[1][2]	Tests deadlock interrupts with PCI and verifies deadlock interrupts do not occur in monitor mode (rotates under the run system)
		10	srt10.t	[1][2]	Tests random shared register instruction sequences in a single CPU (rotates under the run system)
svb.t	Basic S and V register chip and path test	0	svb00.t	[1]	Tests the following instructions: 00200*k* VL Ak 023*i*01 A*i* VL
		1	svb01.t	[1]	Tests the following instructions: 176*i*00 V*i* ,A0,1 1770*j*0 ,A0,1 V*j*
		2	svb02.t	[1]	Tests the following instructions: 176*i*0*k* V*i* ,A0,A*k* 1770*jk* ,A0,A*k* V*j*
		3	svb03.t	[1]	Tests the following instructions: 073*i*30 A*i* VM1 0030*j*0 VM S*j* 073*i*00 S*i* VM 0030*j*1 VM1 S*j* 073*i*10 S*i* VM1 0030*j*2 VM A*j* 073*i*20 A*i* VM 0030*j*3 VM1 A*j* 073*i*30 A*i* VM1
		4	svb04.t	[1]	Tests the following instructions: 076*ijk* S*i* V*j*,A*k* 077*ijk* V*i*,A*k* S*j*

† A `.t` extension indicates the test program was assembled in Triton mode.

‡ The Env column indicates which environment a section runs in: [1] indicates the section runs in environment 1 only; [1][2] indicates the section runs in environment 1 or 2.

Table 3. Environments 1 and 2 Diagnostic Tests (continued)

Test †	Description	Section Information			
		Number	Filename	Env ‡	Function
svb.t (cont.)	Basic S and V register chip and path test	5	svb05.t	[1]	Tests the following instructions: 154*ijk* V*i* S*j*+V*k* 155*ijk* V*i* V*j*+V*k* 156*ijk* V*i* S*j*–V*k* 157*ijk* V*i* V*j*–V*k*
		6	svb06.t	[1]	Tests the following instructions: 140*ijk* V*i* S*j*&V*k* 141*ijk* V*i* V*j*&V*k* 142*ijk* V*i* S*j*!V*k* 143*ijk* V*i* V*j*!V*k* 144*ijk* V*i* S*j*\V*k* 145*ijk* V*i* V*j*\V*k*
		7	svb07.t	[1]	Tests the following instructions: 146*ijk* V*i* S*j*!V*k*&VM 147*ijk* V*i* V*j*!V*k*&VM
		10	svb10.t	[1]	Tests the following instructions: 150*ijk* V*i* V*j*<A*k* 151*ijk* V*i* V*j*>A*k*
		11	svb11.t	[1]	Tests the following instructions: 152*ijk* V*i* V*j*,V*j*<A*k* 153*ijk* V*i* V*j*,V*j*>A*k*
		12	svb12.t	[1]	Tests the following instructions: 005400 150*ij*0 V*i* V*i*<V0 005400 151*ij*0 V*i* V*i*>V0
		13	svb13.t	[1]	Tests the following instruction: 005400 152*ijk* V*i* V*j*,A*k*
		14	svb14.t	[1]	Tests the following instructions: 1750*j*0 VM V*j*,Z 1750*j*1 VM V*j*,N 1750*j*2 VM V*j*,P 1750*j*3 VM V*j*,M

† A `.t` extension indicates the test program was assembled in Triton mode.

‡ The Env column indicates which environment a section runs in: [1] indicates the section runs in environment 1 only; [1][2] indicates the section runs in environment 1 or 2.

Table 3. Environments 1 and 2 Diagnostic Tests (continued)

Test †	Description	Section Information			
		Number	Filename	Env‡	Function
svb.t (cont.)	Basic S and V register chip and path test	15	svb15.t	[1]	Tests the following instructions: 175*ij*4 V*i*,VM V*j*,Z 175*ij*5 V*i*,VM V*j*,N 175*ij*6 V*i*,VM V*j*,P 175*ij*7 V*i*,VM V*j*,M
		16	svb16.t	[1]	Tests the following instructions: 005400 153*ij*0 V*i* V*j*,CVM 005400 153*ij*1 V*i* V*j*,EVM
		17	svb17.t	[1]	Tests the following instructions: 174*ij*4 BMM LV*j* 174*ij*5 BMM UV*j* 070*ij*6 S*i* S*j**BT 174*ij*6 V*i* V*j**BT
svf.t	Comprehensive S and V functional unit test	0	svf00.t	[1][2]	Tests the enable and disable second vector logical (ESL mode) flag
		1	svf01.t	[1][2]	Tests the vector logical functional unit and the S and V register paths to it
		2	svf02.t	[1][2]	Tests the vector shift functional unit and the S and V register paths to it
		3	svf03.t	[1][2]	Tests the vector add and subtract functional unit and the S and V register paths to it
		4	svf04.t	[1][2]	Tests the vector logical compress index functional unit and the S and V register paths to it
		5	svf05.t	[1][2]	Tests the bit matrix multiply (BMM) functional unit
		6	svf06.t	[1][2]	Tests the bit matrix load (BML) status flag
		7	svf07.t	[1][2]	Tests vector functional units chaining timing
		10	svf10.t	[1][2]	Tests vector functional units chaining data integrity

† `A.t` extension indicates the test program was assembled in Triton mode.

‡ The Env column indicates which environment a section runs in: [1] indicates the section runs in environment 1 only; [1][2] indicates the section runs in environment 1 or 2.

Table 3. Environments 1 and 2 Diagnostic Tests (continued)

Test †	Description	Section Information			
		Number	Filename	Env‡	Function
svf.t (cont.)	Comprehensive S and V functional unit test	11	svf11.t	[1][2]	Checks the ability of the vector logical, add, and shift functional units to clear the vector not used (VNU) flag
		12	svf12.t	[1][2]	Performs randomly selected vector functional unit tests (logical, shift, add/subtract, or compress index)
vhs.t	VHISP channel test	0	vhs00.t	[1]	Tests basic functionality of the solid-state storage device (SSD), VHISP channel control, VHISP data paths, and VHISP simple addressing
		1	vhs01.t	[1]	Test VHISP channel interrupts on a single CPU
		2	vhs02.t	[1]	Tests VHISP channel CPU addressing
		3	vhs03.t	[1]	Tests the SSD/VHISP channel data and addressing
		4	vhs04.t	[1]	Tests the SSD/VHISP channel block length registers
		5	vhs05.t	[1]	Tests the channel abort hardware in the VHISP channels and the SSD
		6	vhs06.t	[1]	Tests single-error correction/ double-error detection (SECDED) and single-byte correction/double-byte detection (SBCDBD) for the VHISP channels (runs in single or multiple CPUs and runs in maintenance mode)
		7	vhs07.t	[1]	Tests VHISP channel multiple-CPU interrupts (runs in single or multiple CPUs)
		10	vhs10.t	[1]	Tests VHISP channels with background shared interference code in all clusters (runs in single or multiple CPUs)
		11	vhs11.t	[1][2]	Tests the VHISP channels separately and concurrently with background code on all memory ports of all CPUs (runs in single or multiple CPUs)

† A `.t` extension indicates the test program was assembled in Triton mode.

‡ The Env column indicates which environment a section runs in: [1] indicates the section runs in environment 1 only; [1][2] indicates the section runs in environment 1 or 2.

Table 3. Environments 1 and 2 Diagnostic Tests (continued)

Test †	Description	Section Information			
		Number	Filename	Env‡	Function
vhs.t (cont.)	VHISP channel test	12	vhc.t	[1][2]	Produces random activity on all selected VHISP channels (rotates under the run system)
		13	chn.t	[1][2]	Produces random activity on selected LOSP and VHISP channels with memory port interference (runs in single or multiple CPUs and rotates under the run system)

† A `.t` extension indicates the test program was assembled in Triton mode.

‡ The Env column indicates which environment a section runs in: [1] indicates the section runs in environment 1 only; [1][2] indicates the section runs in environment 1 or 2.

Diagnostic Utilities

Table 4 lists the diagnostic utilities that you can load as control points in environments 1 and 2. This table provides a quick-reference description for each section of the utilities. Refer to the actual listings for detailed information about the utilities.

For more information about viewing the listings, refer to "View –> Listing –> Current" and "View –> Listing –> Other" in the "Environments 1 and 2" section of the *MME Interface Reference* document, publication number HDM-xxx-0.

Table 4. Environments 1 and 2 Diagnostic Utilities

Utility †	Section Information			
	Number	Filename	Env ‡	Description
clr.t	0	clr.t	[1]	Clears interrupts, registers, memory, and shared resources (runs in multiple CPUs)
	1	clrmem.t	[1]	Clears interrupts and memory with scalar writes (rotates under the run system)
	2	clrreg.t	[1]	Clears interrupts and registers (rotates under the run system and runs in multiple CPUs)

† A `.t` extension indicates the utility was assembled in Triton mode.

‡ The Env column indicates which environment a section runs in: [1] indicates the section runs in environment 1 only; [1][2] indicates the section runs in environment 1 or 2.

Table 4. Environments 1 and 2 Diagnostic Utilities (continued)

Utility †	Section Information			
	Number	Filename	Env ‡	Description
clr.t (cont.)	3	clrshr.t	[1]	Clears interrupts and shared resources
	4	clrssd.t	[1]	Clears all selected very high-speed (VHISP) channels and 1 to 4 selected solid-state storage devices (SSDs) on each channel
	5	clrv.t	[1]	Clears interrupts and memory with vector writes (rotates under the run system)
dc.t	0	dc.t	[1]	Contains the diagnostic controller code (runs in multiple CPUs)
diag.t	0	diag.t	[1][2]	Provides a loop frame template for you to write loops in binary (runs in multiple CPUs and rotates under the run system)
find.t	0	find.t	[1][2]	Searches for the data pattern you enter in absolute memory location 1000 (rotates under the run system) **NOTE:** Loading this utility overwrites mainframe memory to the location labelled IDATA. Use the Utiliies –> Find command to do a complete search of memory.
hdump.t	0	hdump.t	[1][2]	Dumps an exchange package and all CPU registers (runs in multiple CPUs)
loop.t	0	loop.t	[1]	Provides a frame for you to write low level loops in binary
patt.t	0	patt00.t	[1][2]	Patterns memory with an address pattern [scalar or scalar logical address pattern (rotates under the run system)]
	1	patt01.t	[1][2]	Patterns memory with an address pattern [scalar, scalar logical, scalar addition, vector, or vector addition address patterns (rotates under the run system)]
scan.t	0	scan.t	[1][2]	Reads all memory with selected CPUs and logs information (full address, processor number, and data read) in a table starting at memory location ERRTBL for any memory error interrupts (runs in multiple CPUs and rotates under the run system)
ssdsz.t	0	ssdsz.t	[1]	Calculates the memory size of one or more SSDs connected to the mainframe

† A `.t` extension indicates the utility was assembled in Triton mode.

‡ The Env column indicates which environment a section runs in: [1] indicates the section runs in environment 1 only; [1][2] indicates the section runs in environment 1 or 2.

Reader Comment Form

Title: **CRAY T90™ Series**
MME Diagnostic Tests and Utilities
Preliminary Information

Number: **HDM-xxx-0**
December 1994

Your feedback on this publication will help us provide better documentation in the future. Please take a moment to answer the few questions below.

For what purpose did you primarily use this document?

_____Troubleshooting
_____Tutorial or introduction
_____Reference information
_____Classroom use
_____Other - please explain _____

Using a scale from 1 (poor) to 10 (excellent), please rate this document on the following criteria and explain your ratings:

_____Accuracy _____
_____Organization _____
_____Readability _____
_____Physical qualities (binding, printing, page layout) _____
_____Amount of diagrams and photos _____
_____Quality of diagrams and photos _____

Completeness (Check one)

_____Too much information _____

_____Too little information _____

_____Just the right amount of information

Your comments help Hardware Publications and Training improve the quality and usefulness of your publications. Please use the space provided below to share your comments with us. When possible, please give specific page and paragraph references. We will respond to your comments in writing within 48 hours.

NAME _____
JOB TITLE_____
FIRM_____
ADDRESS_____
CITY_____STATE_____ZIP_____
DATE_____

[or attach your business card]

CRAY RESEARCH, INC.

CUT ALONG THIS LINE

Fold

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO 6184 ST. PAUL, MN

POSTAGE WILL BE PAID BY ADDRESSEE

CRAY RESEARCH, INC.

Attn: Hardware Publications and Training
890 Industrial Boulevard
Chippewa Falls, WI 54729

Fold

STAPLE