# MDB User Guide

## (CRAY T90™ Series)

**HDM-009-0**

Cray Research Proprietary

**Cray Research, Inc.**

# Record of Revision

| REVISION | DESCRIPTION |
|---|---|
| | August 1995. Original printing. |

Any shipment to a country outside of the United States requires a letter of assurance from Cray Research, Inc.

Requests for copies of Cray Research, Inc. publications should be directed to:

CRAY RESEARCH, INC.
Logistics
6251 South Prairie View Road
Chippewa Falls, WI 54729

Comments about this publication should be directed to:

CRAY RESEARCH, INC.
Hardware Publications and Training
890 Industrial Blvd.
Chippewa Falls, WI  54729

# MDB USER GUIDE

## Figures

*This page intentionally left blank.*

## MSIM Simulator

The mainframe instruction simulator (MSIM) enables you to run diagnostic tests and applications without using mainframe hardware. MSIM runs much like the actual hardware. MSIM is designed to interact with maintenance tools in order to create a realistic maintenance environment. You do not control MSIM directly but instead use MME and the MDB bugger/debugger.

**NOTE:** The MSIM instruction simulator is currently programmed to simulate CRAY T94 and CRAY T916 systems. (1 to 16 CPUs).

There are three main reasons why the CRAY T90 series MSIM simulator was developed:

- to provide Hardware Publications and Training (HPT) instructors with a tool that allows them to conduct their classes without the availability of the real hardware. More specifically, instructors are able to demonstrate the maintenance tools, teach troubleshooting exercises, and reinforce hardware theory using hardware training simulators. Instructors can produce only a limited number of failures on the real hardware.

- to provide field service engineers with a tool that they can use to practice troubleshooting techniques and to learn diagnostics. They also use simulators as refresher training tools. The MSIM and MDB programs are included in MT-T*x.x* maintenance releases.

- to give all students in an HPT classroom the ability to work on the same bug at the same time. Without the MSIM, students have to split up and work in different lab shifts because only a few students can work on a bug at the same time.

## MDB Bugger/Debugger

The MSIM bugger/debugger (MDB) program enables you to duplicate hardware errors or load simulated hardware errors (bugs) to practice hardware troubleshooting. The primary uses of the MDB program are to:

- Modify the configuration of the MSIM.
- Insert hardware faults (load bugs) into MSIM and use MME to troubleshoot the simulated hardware faults, and create and load your own bugs.
- View and change active registers and channel data.

- Set breakpoints to step through diagnostic tests. (This feature was very useful for diagnostic developers who did not have early access to CRAY T90 series hardware.)
- Run one instruction at a time (single-step) of a diagnostic test in a CPU.

## Starting the Mainframe Simulator with the Bugger/Debugger

You can start the MSIM instruction simulator and MDB from the OpenWindows Workspace menu or from a UNIX command prompt.  You can start MME with MSIM or with MSIM and MDB from the MWS workspace menu, as shown in Figure 1:

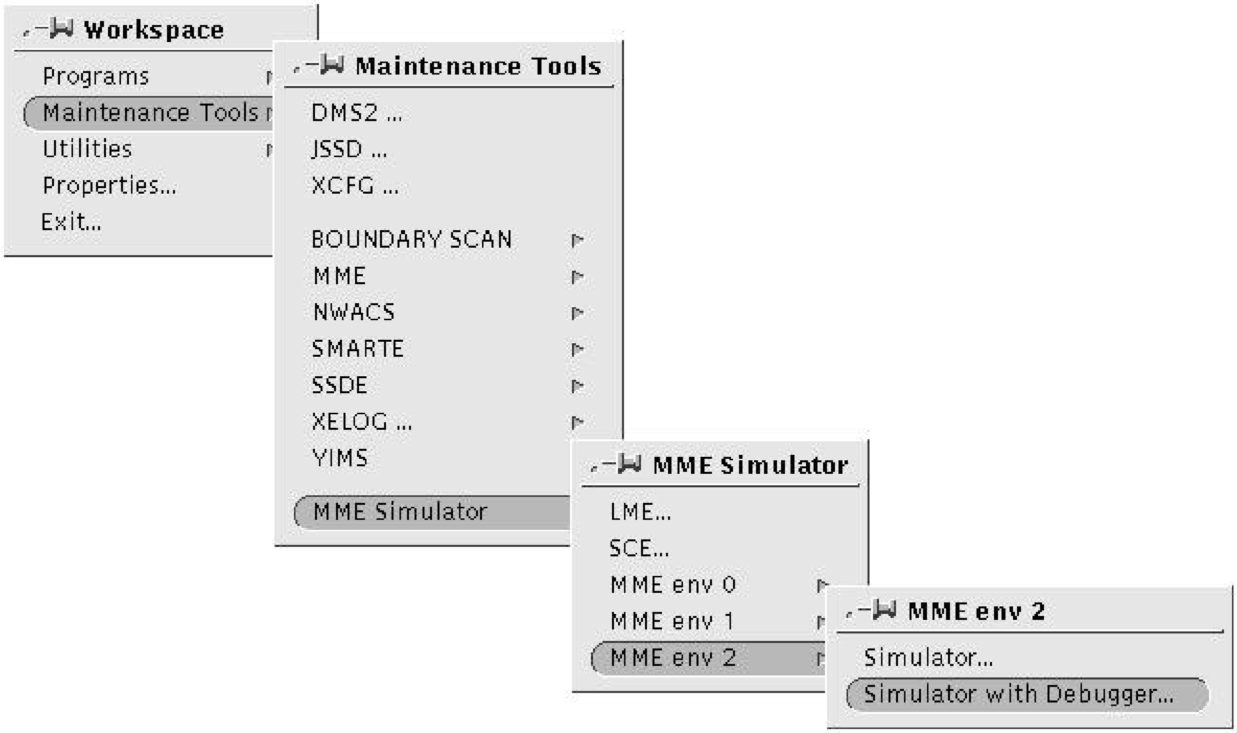
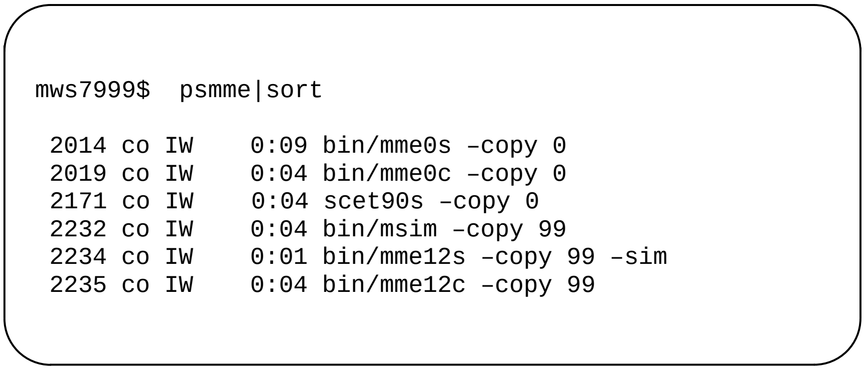


Figure 1.  Workspace Menu Options to Start Environment 1 with the Simulator

The Simulator and Simulator with Debugger commands are programmed in the menus to use the MME `–copy 99` command line option.  For example, choosing the menu command shown in Figure 1 executes the following command:

```
/cri/cme/t32/bin/mme –copy 99 –2 –debug
```

Copy numbers have been implemented with the CRAY T90 series MSIM instruction simulator to enable MSIM (simulated MME) and MDB to run on the MWS at the same time that the real MME is running. (Copy numbers also enable multiple copies of MME to run on the MWS in order to run diagnostics on different partitions of the mainframe.) You can check to see if multiple copies of MME are running by entering the `psmme` (process status mme) command as shown in Figure 2.



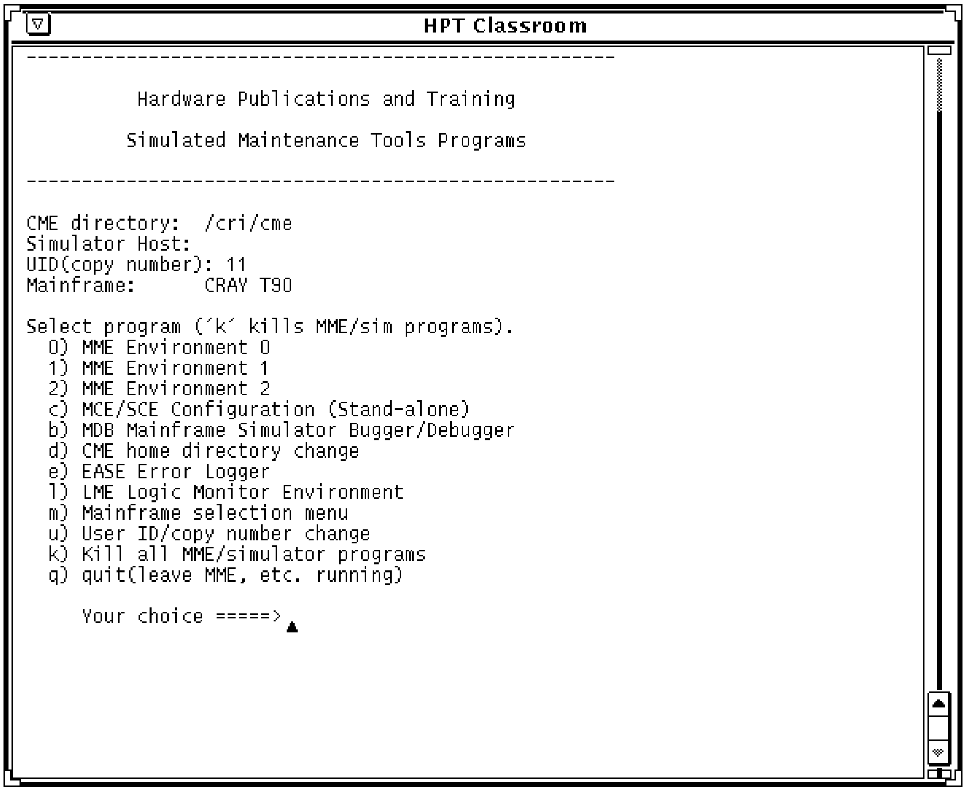
```
mws7999$  psmme|sort

 2014 co IW    0:09 bin/mme0s –copy 0
 2019 co IW    0:04 bin/mme0c –copy 0
 2171 co IW    0:04 scet90s –copy 0
 2232 co IW    0:04 bin/msim –copy 99
 2234 co IW    0:01 bin/mme12s –copy 99 –sim
 2235 co IW    0:04 bin/mme12c –copy 99
```

Figure 2. Displaying MME Processes

## HPT Classroom Environment

You can run MSIM, MDB, and other maintenance tools from the HPT classroom simulator menu. Perform the following procedure to start MME with the simulator and debugger from an HPT classroom workstation.

1. Enter **mme** to display the simulator menu shown in Figure 3.



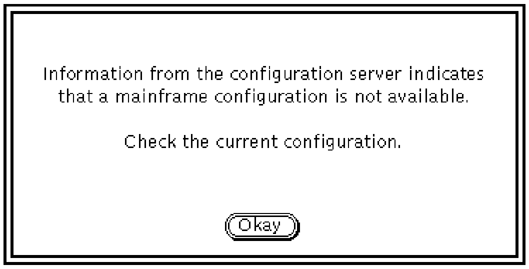
```
----------------------------------------------------

          Hardware Publications and Training

         Simulated Maintenance Tools Programs

----------------------------------------------------

CME directory:  /cri/cme
Simulator Host:
UID(copy number): 11
Mainframe:      CRAY T90

Select program ('k' kills MME/sim programs).
  0) MME Environment 0
  1) MME Environment 1
  2) MME Environment 2
  c) MCE/SCE Configuration (Stand-alone)
  b) MDB Mainframe Simulator Bugger/Debugger
  d) CME home directory change
  e) EASE Error Logger
  l) LME Logic Monitor Environment
  m) Mainframe selection menu
  u) User ID/copy number change
  k) Kill all MME/simulator programs
  q) quit(leave MME, etc. running)

     Your choice =====>
```

Figure 3. HPT Classroom Simulator Environment Menu

2. Start the MME environment you want by entering **0**, **1**, or **2**.

   The first thing the MME server does is attempt to connect to the System Configuration Environment (SCE) server. If the connection is successful, SCE provides MME with the components available for use by the maintenance system. MME automatically configures itself to use these components.

   If a configuration is not available, MME displays the following message:



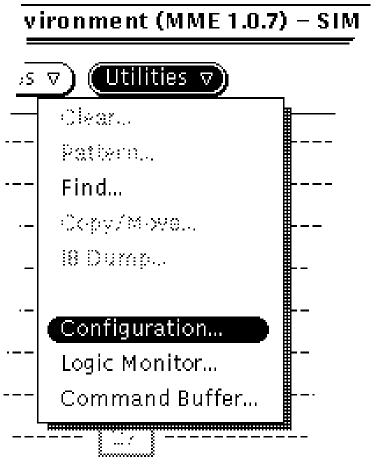
Information from the configuration server indicates that a mainframe configuration is not available.

Check the current configuration.

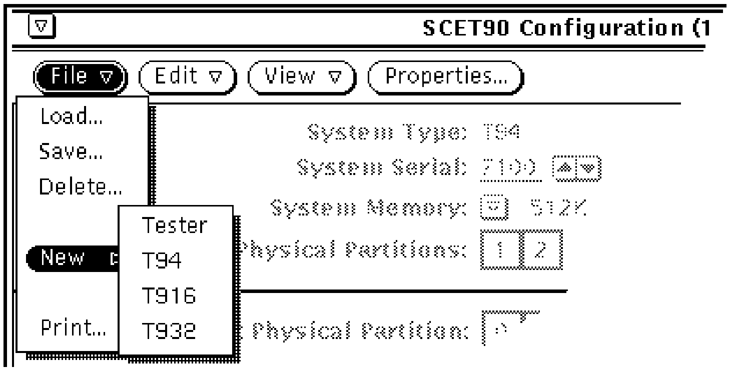
When you see this message, you need to establish a configuration using SCE before you continue using MME, as described in the following steps.

3. Click on (Okay) to close the configuration server message.

4. Select **Utilities –> Configuration** from the MME base window to start the SCE program.



5. Select the **T94** default configuration file. (A configuration file, `SCET90.7001.LAST`, may already exist under the File –> Load menu.)

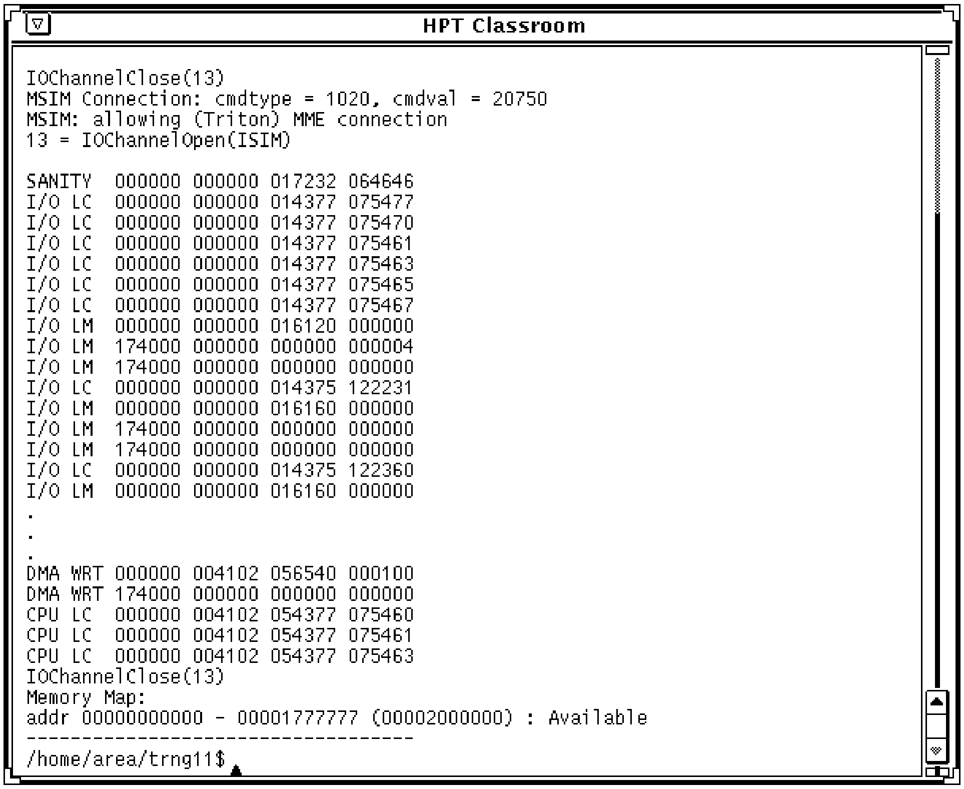


System Type: T94
System Serial: 7100
System Memory: 512K
Physical Partitions: 1 2
Physical Partition:

**NOTE:** The MSIM instruction simulator does not support CRAY T932 systems.

6. Click on Apply and then quit the SCET90 window.

   SCE configuration data may be echoed to a console or shell tool window as shown in the following window.



```
IOChannelClose(13)
MSIM Connection: cmdtype = 1020, cmdval = 20750
MSIM: allowing (Triton) MME connection
13 = IOChannelOpen(ISIM)

SANITY  000000 000000 017232 064646
I/O LC  000000 000000 014377 075477
I/O LC  000000 000000 014377 075470
I/O LC  000000 000000 014377 075461
I/O LC  000000 000000 014377 075463
I/O LC  000000 000000 014377 075465
I/O LC  000000 000000 014377 075467
I/O LM  000000 000000 016120 000000
I/O LM  174000 000000 000000 000004
I/O LM  174000 000000 000000 000000
I/O LC  000000 000000 014375 122231
I/O LM  000000 000000 016160 000000
I/O LM  174000 000000 000000 000000
I/O LM  174000 000000 000000 000000
I/O LC  000000 000000 014375 122360
I/O LM  000000 000000 016160 000000
.
.
.
DMA WRT 000000 004102 056540 000100
DMA WRT 174000 000000 000000 000000
CPU LC  000000 004102 054377 075460
CPU LC  000000 004102 054377 075461
CPU LC  000000 004102 054377 075463
IOChannelClose(13)
Memory Map:
addr 00000000000 - 00001777777 (00002000000) : Available
----------------------------------------
/home/area/trng11$
```

7. Enter **b** from the classroom simulator menu to start the MSIM bugger/debugger (Figure 4, page 9).

## Using MDB to Configure the Mainframe Simulator

When MDB is started, it attempts to connect with a simulator version of MME and copy the configuration data that MSIM is using (this configuration data is defined by using SCE). When you bring up MDB, it reads and loads whatever configuration MSIM was set to.

After you have started the mainframe simulator, you can configure the mainframe simulator or use the default configuration. To configure the mainframe simulator, perform the following procedure:

1. Start MDB with the simulator and debugger to display the windows shown in Figure 4.

**NOTE:** The MSIM Configuration window opens automatically when you start MDB. You can either use the default configuration, or you can configure MSIM yourself.

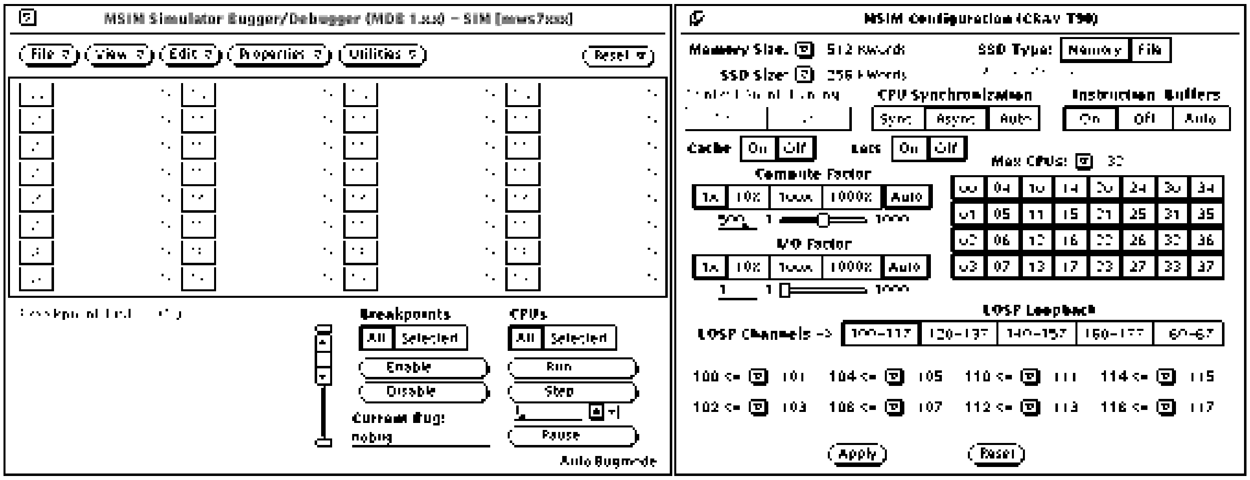


Figure 4. MSIM Simulator Bugger/Debugger and Configuration Windows

2. Select the appropriate memory size from the Memory Size ▽ menu.

   **NOTE:** The memory size of MSIM on HPT classroom workstations is set at 512 Kwords.

3. Select the type of simulated SSD you want from the SSD Type: selection area (for running SSD diagnostic programs).

   Click on Memory to use the MWS-E memory. Click on File to use a file on the hard disk. Click in the Dir: field and type the name of the directory you want to use.

4. Select the appropriate SSD size from the SSD Size ▽ menu.

**NOTE:** The Control Point Tuning options have not been implemented.

5. Select one of the following CPU Synchronization types:

   Sync causes CPUs to take turns executing so that they are always running the same instructions at the same time (default).

   Async does not synchronize instruction execution.

   Auto *not currently implemented.*

6. Click on the appropriate setting from the Instruction Buffers area:

   On enables instruction buffers in the simulator.

   Off disables instruction buffers in the simulator.

   Auto *not currently implemented.*

7. Click on the appropriate setting from the Cache area:

   On enables simulation of cache hardware.

   Off disables cache hardware simulation.

8. Click on the appropriate setting from the Lats area:

   On enables LATs in the simulator for LAT testing.

   Off disables LATs in the simulator for LAT testing.

9. Specify the Compute Factor by either entering the value you want into the Compute Factor field or by dragging the slide bar to the value you want. Click on the 1X, 10X, 100X, 1000X, or Auto compute factor by which you want to multiply the value in the Compute Factor field. (If you have 50 typed into the field and 100X set, the Compute Factor is 5,000.)

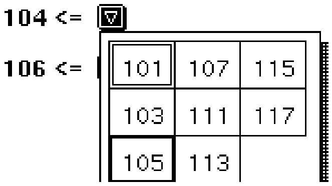
   The Compute Factor specifies the number of instructions the simulator performs before MME and MDB are updated. The higher the compute factor, the higher the pass counts. The lower the Compute Factor, the more often MME and MDB are updated with data. The higher the Compute Factor, the faster the simulator should run because it does not have to stop to update the display as often.

10. Specify the I/O Factor either by entering the value you want into the I/O Factor field or by dragging the slide bar to the value you want. Click on the 1X, 10X, 100X, 1000X, or Auto I/O Factor by which you want to multiply the value in the I/O Factor field.

    The I/O Factor determines the speed at which the simulator updates channel activity. The higher the specified number, the faster channel activity is updated.

11. Select the low-speed channel number you want to configure from the LOSP Loopback area.

12. Specify the channel connections by choosing the connector number from the ▽ next to each channel indicator:



13. From the Max CPUs: ▽ menu, choose the maximum number of CPUs.

    If you click on an individual CPU under Max CPUs:, you configure the CPU. If you deselect a CPU under Max CPUs:, you remove the CPU from the configuration.

14. Perform one of the following steps:

    - Click on Apply to change the values in MSIM to the values that you have just selected.

    - Click on Reset to cancel your changes and restore the previous values.

# Using Bugs

With MDB, you can load system-supplied bugs into the MSIM instruction simulator and run the MME control points to identify simulated hardware faults.

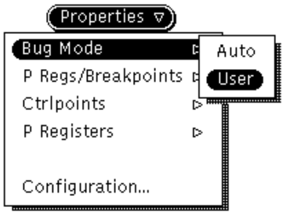
The following system-supplied user bug files are available:

- `bugcache1`
- `bugcache2`
- `bugcache3`
- `buglat1`
- `bugmem1`
- `bugmem2`
- `bugmem3`
- `bugreg1`
- `bugreg1`
- `bugshr1`
- `bugshr2`
- `bugshr3`
- `bugshr4`
- `clear_mem_cache_bug`
- `clear_reg_shr_bug`

Refer to the files in `/cri/cme/t32/rel/msim/bugdoc` for detailed information about the bugs and for troubleshooting hints.

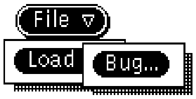
## Loading and Applying a Bug into MSIM

To load and apply a bug into the simulator, perform the following procedure:

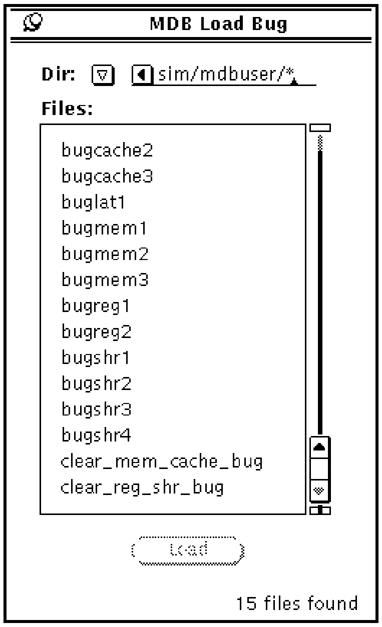


1. Choose **Properties –> Bug Mode –> User**, as shown at the left.

   **NOTE:** There are no auto-mode bug files.



2. Choose **File –> Load –> Bug**, as shown at the left.
   The MDB Load Bug window appears:



Dir: ▽ sim/mdbuser/*

Files:

bugcache2
bugcache3
buglat1
bugmem1
bugmem2
bugmem3
bugreg1
bugreg2
bugshr1
bugshr2
bugshr3
bugshr4
clear_mem_cache_bug
clear_reg_shr_bug

15 files found

Figure 5. MDB Load Bug Window

3. Choose the Dir: ▽ to specify the directory you want to use or triple click on the Dir field, type the directory name, and press the Return key.

   **NOTE:** Bug files are located in the `/cri/cme/t32/usr/msim/mdbuser` directory.

4. In the Files scroll box, click on the bug you want to load.

5. Click on Load; MDB loads the bug into the simulator. The currently loaded bug is indicated in the Current Bug field.

   **NOTE:** You can also double-click on the bug to load it.

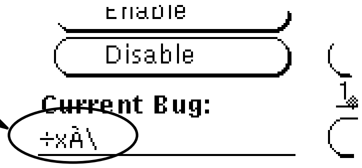
6. Choose **Utilities –> Bugmaker –> Shared** to apply shared register or register bugs or choose **Utilities –> Bugmaker –> Memory** to apply memory, cache, or LAT bugs.

7. From the MSIM Bugmaker window, click on the Apply button to apply the bug that is currently loaded.

## Removing a Bug

To remove a bug from the CRAY T90 series MSIM instruction simulator, you must load and apply one of the two clear bugs, as shown in the bottom of the MDB Load Bug window (Figure 5). Using the Edit –> Remove Bug command will not completely remove the bug. If you use the Edit –> Remove Bug command, you may receive garbled text in the Current Bug: field.

Garbled text indicates that the bug is not completely removed.



Current Bug:
÷xÂ\

- Load the clear_mem_cache_bug to remove memory, cache, and LAT bugs.

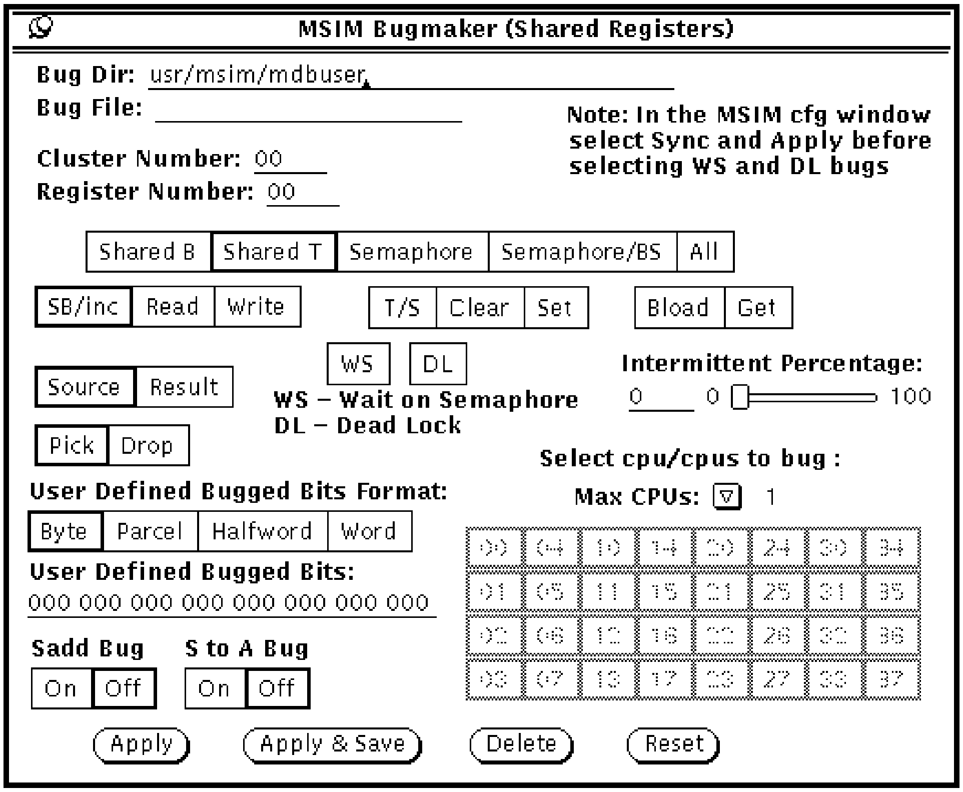
- Load the clear_reg_shr_bug to remove register and shared register bugs.

**NOTE:** After loading one of the clear bugs, you should click on the (Apply) button in the appropriate MSIM Bugmaker window to ensure that the clear bug is applied.

## Creating Shared Register Bugs

Choose **Utilities –> Bugmaker** to begin creating shared register bugs. The MSIM Bugmaker (Shared Registers) window appears, as shown in Figure 6.

**NOTE:** To access the bugmaker utility, MDB must be in user bug mode.



MSIM Bugmaker (Shared Registers)

Bug Dir: usr/msim/mdbuser

Bug File:

Cluster Number: 00

Register Number: 00

Note: In the MSIM cfg window select Sync and Apply before selecting WS and DL bugs

WS – Wait on Semaphore
DL – Dead Lock

Intermittent Percentage: 0

Select cpu/cpus to bug :

Max CPUs: 1

User Defined Bugged Bits Format:

User Defined Bugged Bits: 000 000 000 000 000 000 000 000

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 00 | 04 | 10 | 14 | 20 | 24 | 30 | 34 |
| 01 | 05 | 11 | 15 | 21 | 25 | 31 | 35 |
| 02 | 06 | 12 | 16 | 22 | 26 | 32 | 36 |
| 03 | 07 | 13 | 17 | 23 | 27 | 33 | 37 |

**NOTES:** The Source and Result options do not work.

Intermittent Percentage indicates the percent of time the bug is not active. (0% means the bug is always active.)

**KEY**

All = All Shared Registers
Apply = Loads the bug into the simulator
Apply & Save = Loads the bug and saves it as a file
Bload = Broadside Load
BS = Broadside
DL = Deadlock
Get = Get Function
SB/inc. = Shared B Incremental
T/S = Test and Set Instruction
W/S = Wait on Semaphore

Figure 6. MSIM Bugmaker (Shared Registers) Window

### Procedure for Creating Register Bugs

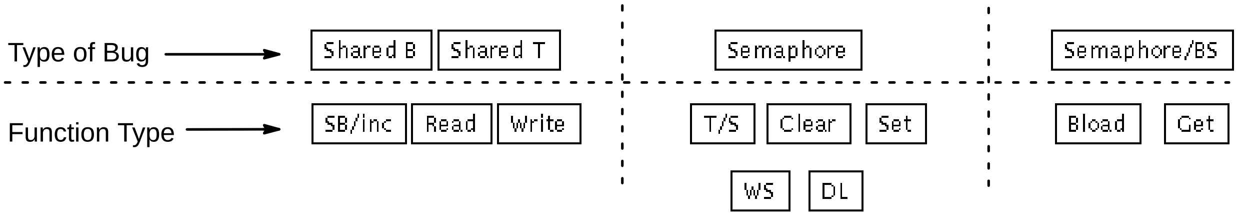
You should be aware of the following points before you create a shared register bug:

- MDB allows you to create bugs (shared register and memory) that do not match CRAY T90 series hardware. You need to define bugs that correspond to the hardware.
- If you have a predefined shared register bug loaded, that bug may appear in the Shared Registers window. You will not be able to modify, rename, and save the bug. The Current Bug field in MDB should read nobug before you create your own bugs.
- If you have a predefined bug loaded and you click on the Delete button, you will delete the bug from your `msim/mdbuser/` directory.

Perform the following procedure to create a shared B register or shared T register bug:

1. In the Bug File field, enter a name for the bug you are creating.
2. Enter the Cluster Number and Register Number.

   (Use an octal number 0 through 21 for the cluster number.)
3. Select the type of shared register bug you want to create along with one of the function types for that type of bug.



**NOTE:** Currently, the Source and Result options do not work.

4. Click on Pick or Drop to pick or drop a bit.
5. Click on Byte, Parcel, Halfword, or Word to specify the format for the bit mask of bits you want to bug.

6. In the User Defined Bugged Bits field, enter a bit mask that indicates the bit(s) you want to bug.

   **NOTE:** Enter shared B register bugged bits in the lower half of the bit mask and semaphore/BS bugged bits in the upper half.

7. Define a value in the Intermittent Percentage field.

   Use the slide bar or type in a value to set the percentage of time you want the bug to be inactive. This value specifies how often the bug will not be active. (0% means the bug will always be active.)

8. Click on the CPUs you want to bug. (The Max CPUS: ▽ menu does not work.)

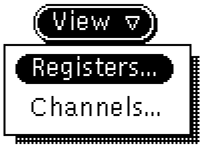
9. To apply the bug without saving it, click on (Apply). To apply and save the bug, click on (Apply & Save).

   If a bug with the same name already exists, a popup message appears that asks if you want to overwrite the existing bug.

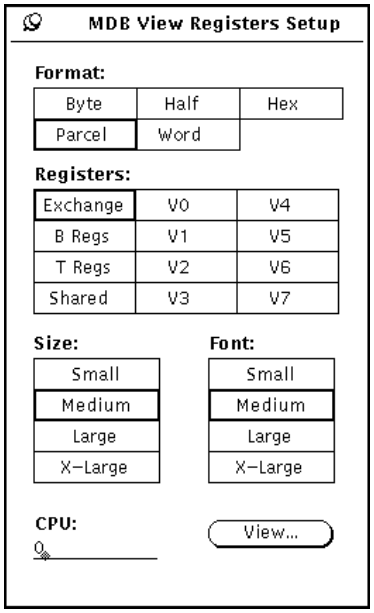
   **NOTE:** To delete the bug selected in the bug file, click on (Delete). To reset the window to the last loaded bug, click on (Reset).

## Viewing the Contents of Registers

Perform the following procedure to view the contents of registers.



1. Choose **View –> Registers**, as shown at the left. MDB displays the MDB View Registers Setup window:



2. Click on a Format [Byte, Half the (halfword), Hex (hexadecimal), Parcel, or Word] to indicate in which format the register contents should be displayed.

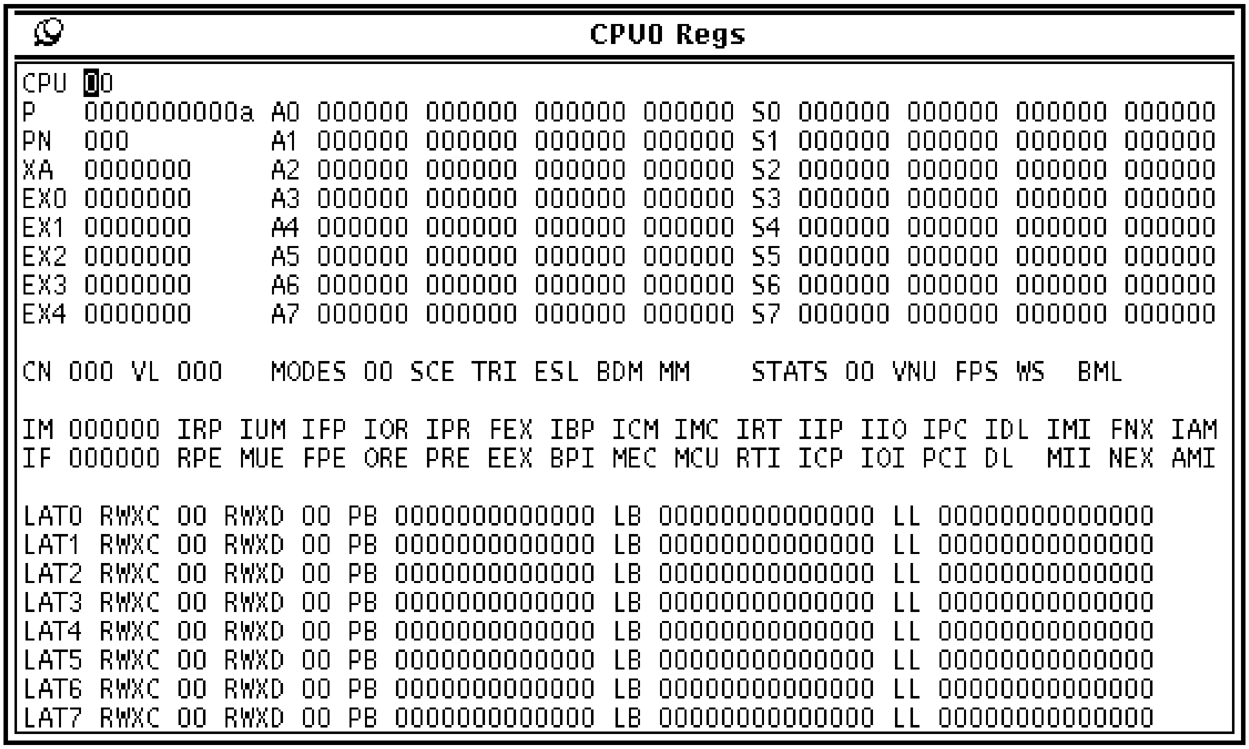
3. Specify which Registers you want to view by clicking on one of the following selections: Exchange, B Regs, T Regs, Shared, V0, V1, V2, V3, V4, V5, V6, or V7.

4. Click on a Size [Small, Medium, Large, or X-Large (extra large)] to specify the size of the window that will be displayed.

5. Click on a Font [Small, Medium, Large, or X-Large (extra large)] to specify the size of the font that will be displayed.

6. Double click on the CPU field. Type the number of the CPU that you want to use.

7. Click on View... . MDB displays the specified register. For example, if you select CPU 0 exchange registers, the following window appears:



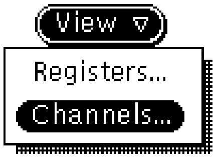
CPU0 Regs

```
CPU 00
P   0000000000a A0 000000 000000 000000 000000 S0 000000 000000 000000 000000
PN  000         A1 000000 000000 000000 000000 S1 000000 000000 000000 000000
XA  0000000     A2 000000 000000 000000 000000 S2 000000 000000 000000 000000
EX0 0000000     A3 000000 000000 000000 000000 S3 000000 000000 000000 000000
EX1 0000000     A4 000000 000000 000000 000000 S4 000000 000000 000000 000000
EX2 0000000     A5 000000 000000 000000 000000 S5 000000 000000 000000 000000
EX3 0000000     A6 000000 000000 000000 000000 S6 000000 000000 000000 000000
EX4 0000000     A7 000000 000000 000000 000000 S7 000000 000000 000000 000000

CN 000 VL 000   MODES 00 SCE TRI ESL BDM MM    STATS 00 VNU FPS WS  BML

IM 000000 IRP IUM IFP IOR IPR FEX IBP ICM IMC IRT IIP IIO IPC IDL IMI FNX IAM
IF 000000 RPE MUE FPE ORE PRE EEX BPI MEC MCU RTI ICP IOI PCI DL  MII NEX AMI

LAT0 RWXC 00 RWXD 00 PB 0000000000000 LB 00000000000000 LL 00000000000000
LAT1 RWXC 00 RWXD 00 PB 0000000000000 LB 00000000000000 LL 00000000000000
LAT2 RWXC 00 RWXD 00 PB 0000000000000 LB 00000000000000 LL 00000000000000
LAT3 RWXC 00 RWXD 00 PB 0000000000000 LB 00000000000000 LL 00000000000000
LAT4 RWXC 00 RWXD 00 PB 0000000000000 LB 00000000000000 LL 00000000000000
LAT5 RWXC 00 RWXD 00 PB 0000000000000 LB 00000000000000 LL 00000000000000
LAT6 RWXC 00 RWXD 00 PB 0000000000000 LB 00000000000000 LL 00000000000000
LAT7 RWXC 00 RWXD 00 PB 0000000000000 LB 00000000000000 LL 00000000000000
```

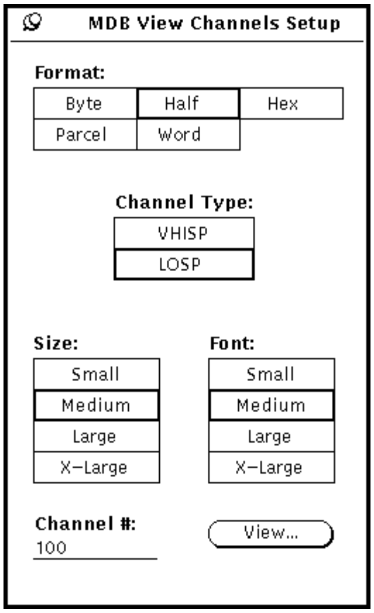
The registers window displays simulated values located in the registers of the MSIM program.

## Viewing Channel Data

To view channel data, perform the following procedure:



1. Choose **View –> Channels**, as shown at the left. MDB displays the MDB View Channels Setup window:



MDB View Channels Setup

Format:
Byte | Half | Hex
Parcel | Word

Channel Type:
VHISP
LOSP

Size:
Small
Medium
Large
X–Large

Font:
Small
Medium
Large
X–Large

Channel #: 100

View...

2. Click on a Format [Byte, Half (halfword), Hex (hexadecimal), Parcel, or Word] to indicate in which format the register contents should be displayed.

3. Click on a Channel [VHISP (very high-speed channel) or LOSP (low-speed channel)] to specify the type of channel data you want to view.

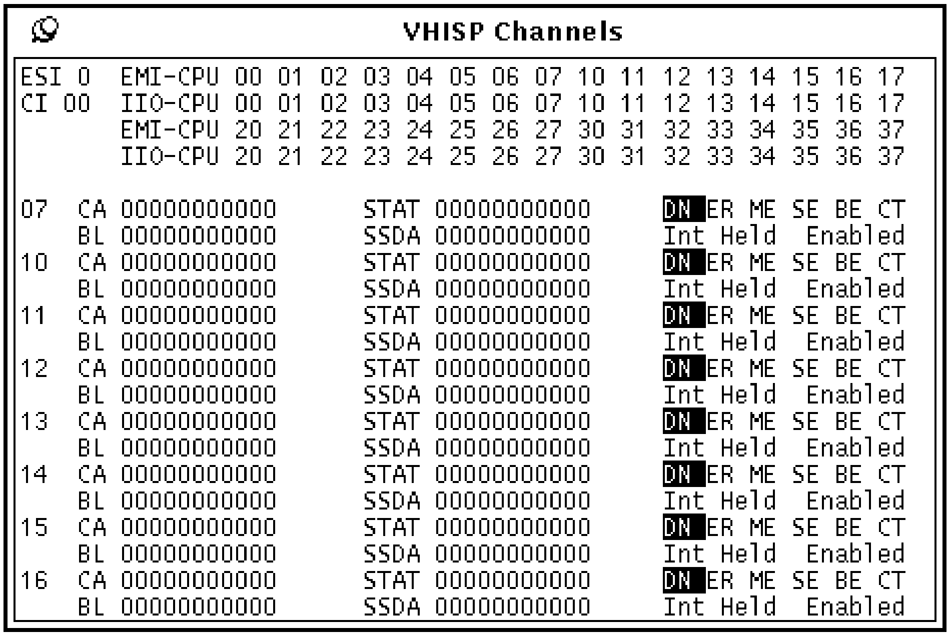
4. Click on a Size [Small, Medium, Large, or X-Large (extra large)] to specify the size of the window that will be displayed.

5. Click on a Font [Small, Medium, Large, or X-Large (extra large)] to specify the size of the font that will be displayed.

6. Double click on the Channel # field. Type the number of the channel you want to use.

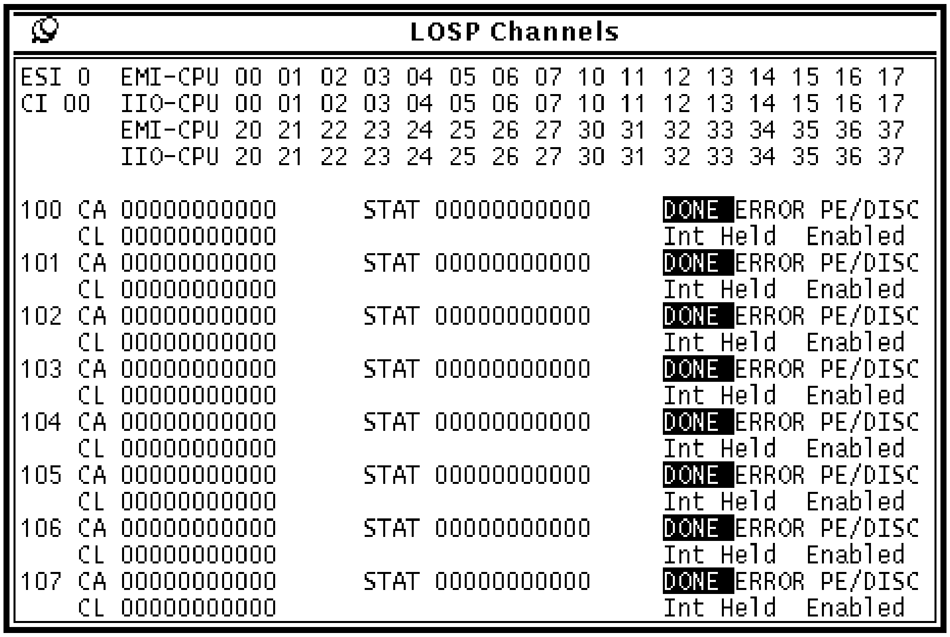
7. Click on View.... The channel data is displayed in a window. Refer to Figure 7 for an example VHISP Channels window or to Figure 8 for an example LOSP Channels window.

**NOTE:** The channel numbers inside the VHISP Channels window should start at number 20. The numbers may start at 07 as shown in Figure 7; if this happens, try resizing the window from the menu located inside the VHISP Channels window. Note that channel tests were not tested and may not function properly.



```
VHISP Channels

ESI 0   EMI-CPU 00 01 02 03 04 05 06 07 10 11 12 13 14 15 16 17
CI 00   IIO-CPU 00 01 02 03 04 05 06 07 10 11 12 13 14 15 16 17
        EMI-CPU 20 21 22 23 24 25 26 27 30 31 32 33 34 35 36 37
        IIO-CPU 20 21 22 23 24 25 26 27 30 31 32 33 34 35 36 37

07   CA 00000000000      STAT 00000000000     DN ER ME SE BE CT
     BL 00000000000      SSDA 00000000000     Int Held  Enabled
10   CA 00000000000      STAT 00000000000     DN ER ME SE BE CT
     BL 00000000000      SSDA 00000000000     Int Held  Enabled
11   CA 00000000000      STAT 00000000000     DN ER ME SE BE CT
     BL 00000000000      SSDA 00000000000     Int Held  Enabled
12   CA 00000000000      STAT 00000000000     DN ER ME SE BE CT
     BL 00000000000      SSDA 00000000000     Int Held  Enabled
13   CA 00000000000      STAT 00000000000     DN ER ME SE BE CT
     BL 00000000000      SSDA 00000000000     Int Held  Enabled
14   CA 00000000000      STAT 00000000000     DN ER ME SE BE CT
     BL 00000000000      SSDA 00000000000     Int Held  Enabled
15   CA 00000000000      STAT 00000000000     DN ER ME SE BE CT
     BL 00000000000      SSDA 00000000000     Int Held  Enabled
16   CA 00000000000      STAT 00000000000     DN ER ME SE BE CT
     BL 00000000000      SSDA 00000000000     Int Held  Enabled
```

Figure 7. VHISP Channel Data Display



```
LOSP Channels

ESI 0   EMI-CPU 00 01 02 03 04 05 06 07 10 11 12 13 14 15 16 17
CI 00   IIO-CPU 00 01 02 03 04 05 06 07 10 11 12 13 14 15 16 17
        EMI-CPU 20 21 22 23 24 25 26 27 30 31 32 33 34 35 36 37
        IIO-CPU 20 21 22 23 24 25 26 27 30 31 32 33 34 35 36 37

100 CA 00000000000      STAT 00000000000     DONE ERROR PE/DISC
    CL 00000000000                           Int Held  Enabled
101 CA 00000000000      STAT 00000000000     DONE ERROR PE/DISC
    CL 00000000000                           Int Held  Enabled
102 CA 00000000000      STAT 00000000000     DONE ERROR PE/DISC
    CL 00000000000                           Int Held  Enabled
103 CA 00000000000      STAT 00000000000     DONE ERROR PE/DISC
    CL 00000000000                           Int Held  Enabled
104 CA 00000000000      STAT 00000000000     DONE ERROR PE/DISC
    CL 00000000000                           Int Held  Enabled
105 CA 00000000000      STAT 00000000000     DONE ERROR PE/DISC
    CL 00000000000                           Int Held  Enabled
106 CA 00000000000      STAT 00000000000     DONE ERROR PE/DISC
    CL 00000000000                           Int Held  Enabled
107 CA 00000000000      STAT 00000000000     DONE ERROR PE/DISC
    CL 00000000000                           Int Held  Enabled
```

Figure 8. LOSP Channel Data Display

## Using Breakpoints

A breakpoint consists of a P register address value and a list of CPUs assigned to that breakpoint. When you use a breakpoint and run control points in MME, the CPU stops issuing control point instructions when the P register value reaches the breakpoint for an assigned CPU. A CPU does not begin issuing instructions again until you press Step or Run.

Using breakpoints, you can perform the following tasks:

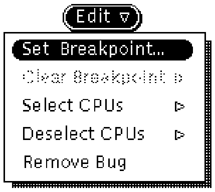
- Set a breakpoint for one or more CPUs.
- Replace an existing breakpoint with a new P register value or assigned CPU(s).
- Enable a breakpoint for each assigned CPU; when you set a breakpoint, the simulator automatically enables the breakpoint.
- Disable a breakpoint for each assigned CPU. This maintains the breakpoint and the assigned CPU relationship, but the breakpoint will not trigger in the assigned CPUs.
- Assign a CPU to one or more breakpoints. Clear or erase a breakpoint.
- Assign more than one CPU to one breakpoint, and all the breakpoints have the same assigned CPUs.
- Assign more than one CPU to more than one breakpoint, and some of the breakpoints have the some of the CPUs assigned, while other breakpoints have other CPUs assigned.
- Select from one to four CPUs for the currently selected breakpoint or for all defined breakpoints (MSIM supports a maximum of four CRAY T90 series CPUs.)

## Setting a Breakpoint

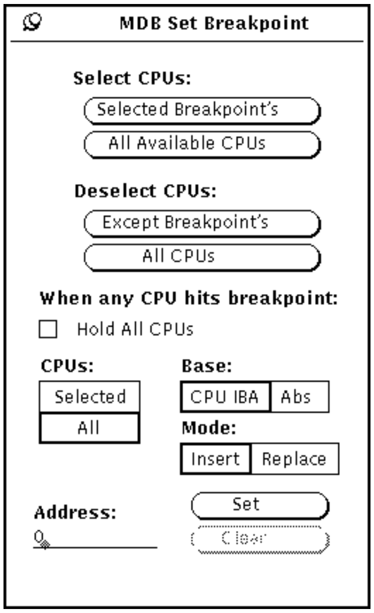
To set a breakpoint, perform the following procedure:

1. Start an MME environment and load a control point. A control point must be loaded before a breakpoint can be set in MDB.

2. Start the MDB bugger/debugger program.



3. Choose **Edit –> Set Breakpoint** to display the MDB Set Breakpoint window:



4. Define which CPUs you want assigned to the breakpoint by performing one of the following actions.

   - Use the default CPUs: All setting.

   - Click on individual CPUs in the MSIM Simulator Bugger/Debugger base window to select (or deselect) CPUs.

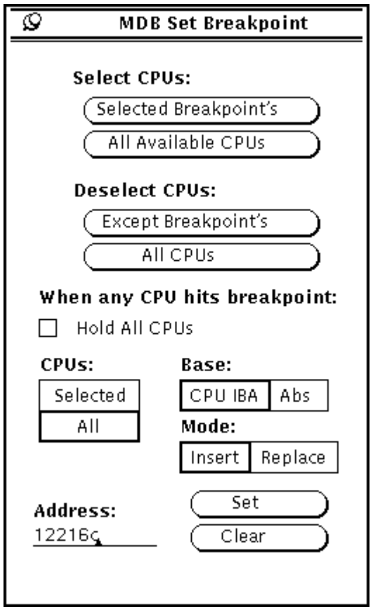
5. Perform one of the following actions:

   - To hold all CPUs when the breakpoint is reached in any CPU, click on the `Hold All CPUs` check box. Go to Step 6.

   - To allow other CPUs to keep running when a CPU reaches a breakpoint, go to Step 6.

6. Change the default Base: or Mode: values if needed.

7. Click on the Address: field, and type the value you want. This example uses 12216c in the Address: field to set the breakpoint at 12126c, as shown:

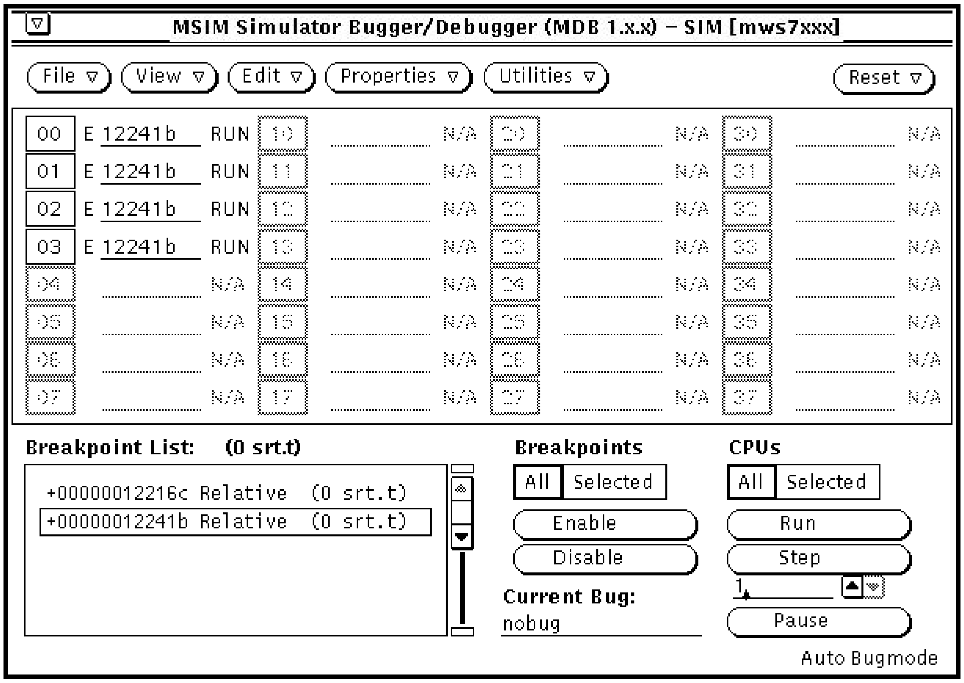


Address:
12216c

8. Click on Set .

9. Repeat Step 4 through Step 8 to set another breakpoint (this example uses an address of 12241b for the second breakpoint). Refer to Figure 9.

**NOTE:** Clicking on individual breakpoints listed in the Breakpoint List: area enables you to quickly see which CPUs are assigned to each breakpoint.



| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | E 12241b | RUN | 10 | | N/A | 20 | | N/A | 30 | | N/A |
| 01 | E 12241b | RUN | 11 | | N/A | 21 | | N/A | 31 | | N/A |
| 02 | E 12241b | RUN | 12 | | N/A | 22 | | N/A | 32 | | N/A |
| 03 | E 12241b | RUN | 13 | | N/A | 23 | | N/A | 33 | | N/A |
| 04 | | N/A | 14 | | N/A | 24 | | N/A | 34 | | N/A |
| 05 | | N/A | 15 | | N/A | 25 | | N/A | 35 | | N/A |
| 06 | | N/A | 16 | | N/A | 26 | | N/A | 36 | | N/A |
| 07 | | N/A | 17 | | N/A | 27 | | N/A | 37 | | N/A |

Breakpoint List: (0 srt.t)

```
+00000012216c Relative  (0 srt.t)
+00000012241b Relative  (0 srt.t)
```

Current Bug: nobug

Figure 9. Breakpoint Set at Locations 12216c and 12241b

10. With a breakpoint selected, click the Go button in the MME window to run and monitor the control point.

    You can select the **View –> Listing –> Current** command from the MME window to view the listing and instruction addresses for the control point. The **View –> Runtime Information –> Current** command displays runtime information for the current control point and provides updated information as the control point runs.

## Running Selected CPUs or All CPUs

Before performing this procedure, ensure that you have selected the appropriate box: All or Selected CPUs.

To run selected CPUs or all CPUs, click on Run. The specified CPU(s) (All or Selected) begin issuing instructions until one of the following things occurs:

- A breakpoint is reached
- You click on Pause to pause the CPU(s)
- You click on Step to step the CPU(s)
- MME master clears the simulated mainframe

## Using Step Mode to Run a CPU or All CPUs

Before starting this procedure, ensure that you have selected one or more CPUs and All or Selected CPUs, as appropriate.

To use step mode to run a selected CPU or all CPUs, perform the following procedure:

1. Set a breakpoint in MDB, as described in Step 1 through Step 8 of the “Setting a Breakpoint” procedure.
2. With a breakpoint selected, click on the Go button in the MME window and wait until the breakpoint address is reached.
3. Click on the ▼ or ▲ button to decrease or increase the number of instructions that will be run each time you click on the Step button.
4. Click on Step. The selected CPU or all CPUs are now in step mode (depending on whether Selected or All is specified for the CPUs selection). The CPU(s) in step mode issues the specified number of instructions (unless a breakpoint is reached) and then pauses.

   Watch the MME window to ensure that P register or CIP values change when you use step mode to run the CPU(s).