Mellanox Software Tools (MST) User's Manual

Rev 4.20

Mellanox Software Tools (MST) User's Manual

Document Number: 2125SM

Mellanox Technologies, Inc.
2900 Stender Way
Santa Clara, CA 95054
U.S.A.
www.Mellanox.com

Tel: (408) 970-3400
Fax: (408) 970-3403

Mellanox Technologies Ltd
PO Box 586 Hermon Building
Yokneam 20692
Israel

Tel: +972-4-909-7200
Fax: +972-4-959-3245

Table of Contents

List of Figures

List of Tables

About this Manual

This manual describes the tools and utilities included in the Mellanox Software Tools (MST) package for the architecture platforms x86, IA-64, Intel EM64T, and PPC with Linux as the running operating system.

This MST package includes a script for the installation of any one, two, or all three of the following tools:

- The InfiniHost Mellanox Software Tools (hMST)
- The Infiniscale III Mellanox Software Tools (is3MST)
- The InfiniScale Mellanox Software Tools (isMST)

This manual is organized in the following manner:

- Two introductory chapters, with Chapter 1 describing how to install the tool sets above, and Chapter 2 describing the available operations by the unified MST Driver (for all three tool sets).
- Part I follows, comprised of chapters 2-6, where the hMST tools are described.
- Part II follows, comprised of chapters 7-11, where the is3MST tools are described.
- Part III follows, comprised of chapters 12-15, where the isMST tools are described.
- Part IV follows, containing the User's Manual for MTUSB-1, the USB to I^2C Adapter.
- An Appendix where a special initialization file used by one of the is3MST tools is described.

Intended Audience

This manual is intended for software developers running the Mellanox host and/or switch devices, and writing drivers and applications running above it.

The manual assumes familiarity with the InfiniBand™ architecture specification.

Related Documentation

Please refer to the following documentation as reference to Mellanox InfiniBand Host Channel Adapter and Switch devices.

- InfiniHost MT23108 Programmer's Reference Manual, *Doc. # 2111PM*
- InfiniHost MT23108 Hardware Reference Manual, *Doc. # 2112HM*
- InfiniScale MT43132 Register Specification, *Doc. # 2053RS*
- InfiniScale MT43132 Data Sheet (Hardware Reference Manual), *Doc. # 2052HM*
- InfiniScale III MT47396 Programmer's Reference Manual, *Doc. # 2235PM*
- InfiniScale III MT47396 Hardware Reference Manual, *Doc. # 2231HM*

1 Mellanox Software Tools (MST) Installation

This chapter presents the platforms and operating systems supported by this MST package. It then describes a unified script called *mst_install* which may be used to install one, two, or all three of the tool sets: hMST, is3MST, and isMST.

1.1 Supported Platforms and Operating Systems

The tools in this MST package have been qualified on the architecture platforms and operating systems listed in the following table:

Table 1 - Supported Platforms and Operating Systems

Platform	Operating System	Kernel	GCC
X86	Red Hat Linux Advanced Server 2.1	2.4.9-e.24 (UP; SMP and enterprise)	2.96-108.1
	Red Hat Linux 8.0	2.4.18-14 (UP; SMP and bigmem)	3.2-7
	Red Hat Linux 8.0	2.4.22 (UP; SMP) with gcc 3.2-7	
	Red Hat Linux 8.0	2.4.23 (UP; SMP) with gcc 3.2-7	
	Red Hat Linux 9.0	2.4.20-8 (UP; SMP; bigmem)	3.2.2 20030222 (Red Hat Linux 3.2.2-5)
	Red Hat Enterprise Linux AS 3.0	2.4.21-4.EL (UP; SMP)	3.2.3 20030502 (Red Hat Linux 3.2.3-20)
	Red Hat Enterprise Linux AS 3.0	2.4.21-4.EL big pages & kernel patch	3.2.3 20030502
	SuSe SLES 8.0	2.4.19-64G-SMP	3.2
	SuSe 9.0	2.4.21-99-smp4G	3.2.3
	SuSe 9.0	2.6.3 (UP; SMP; bigmem)	3.3.1
IA-64	Red Hat Linux Advanced Server release 2.1AS (Derry)	2.4.18-e.25smp	2.96-112.7.2
	SuSE SLES-8 (ia64)	2.4.19-SMP	3.2
	Red Hat Enterprise Linux AS 3.0	2.4.21-4.EL	3.2.3 20030502 (Red Hat Linux 3.2.3-20)
Intel EM64T	SuSE 9.0	2.4.21-156-smp	3.3.1
X86_64 (AMD64)	SuSE SLES-8 SP3 (AMD64)	2.4.21-143-smp	3.2.2
	Red Hat Enterprise Linux AS 3.0	2.4.21-4.EL	3.2.3 20030502 (Red Hat Linux 3.2.3-20)
PPC (7455, HDPU Compute Blade) [1]	Yellow Dog Linux release 3.0 (Sirius)	2.4.23-pre5	3.3

1. Supported by hMST only.

1.2 Supported I^2C Adapters

The tools in this MST package can access Mellanox devices attached to the following I^2C Adapters:

- MTUSB USB-to-I2C Adapter (Initial revision of USB-I2C adapter from Mellanox).
- MTUSB-1 USB-to-I2C Adapter (First revision of USB-I2C adapter from Mellanox).
- CALIBRE (ISA to I2C adapter card from Mellanox; requires a PC with an ISA slot).

1.3 Tools Installation

The entire MST package is provided in a single tar file which may be extracted using the following command:

tar -zxvf MST-<platform>-<version>.tgz

Once you untar, you will notice the existence of the following subdirectories:

hmst-<platform> - Installation kit for InfiniHost MT23108 and InfiniHost III-Ex MT25208 tools

ismst-<platform> - Installation kit for InfiniBridge MT21108 and InfiniScale MT43132 tools

is3mst-<platform> - Installation kit for InfiniScale III MT47396 tools

This package provides an installation script named *mst_install*. It enables the installation of one, two, or all three tools sets: hMST, isMST, and is3MST.

When you run *mst_install*, it prompts for the tool set(s) you wish to install. More than one tool set may be specified in the response line, separated by spaces. If none is specified, all three will be installed. The following is an example of an *mst_install* run:

```
# ./mst_install
`/tmp/scripts/mvision' -> `/usr/mst/bin/mvision'
create symbolic link `/usr/bin/mvision' to `/usr/mst/bin/mvision'
`/tmp/scripts/mtrace' -> `/usr/mst/bin/mtrace'
create symbolic link `/usr/bin/mtrace' to `/usr/mst/bin/mtrace'
     1  hmst-i686-3.2.3-release.tgz
     2  ismst-i686-3.2.3-release.tgz
     3  is3mst-i686-3.2.3-release.tgz
Enter the numbers of the desired packages to install (Press Enter for all)>
```

Additionally, three scripts are provided which serve to unify the initiation of applications from the various tools sets. See Table 2, “Unified Initiation Scripts” for description.

Table 2 - Unified Initiation Scripts

Script	Operation
mvision	Prompts for chip type (MTxxx), then runs *infinivision*, *infinivisionEx*, *is3infinivision*, *adevmon* or *gdevmon* as applicable.
mtrace	Prompts for chip type, then runs *itrace* or *is3trace* as applicable.

1.4 I^2C Cable Connection During Device Power-On

During power-on of a device such as MTEK43132 or MTS2400, the device uses the I^2C bus to transfer its run-time program to its internal RAM. Since external signals on the I^2C bus may disturb this process, it is required to disconnect the I2C cable from the device until the power-on process is complete.

1.5 MST Driver Control Operations

1.5.1 Overview

The MST driver provides basic access to Mellanox hardware for use by Mellanox and OEM tools. It is composed of several kernel modules, a linkable access library (libmtcr.a), access utilities (mread, mwrite), and utilities for remote access. A script called *mst* is provided to the user to operate the MST driver functions.

Specifically, the *mst* script enables the user to execute the following operations:

1.5.2 Starting the MST Driver

To start the MST driver, execute the following command:

```
mst start
```

This command:

- Creates special files representing Mellanox devices in the directory /dev/mst.
- Loads appropriate kernel modules.
- Saves PCI configuration headers in the directory /usr/mst/etc/pci.

After successful completion of this command, the set of MST drivers is ready to work.

1.5.3 Stopping the MST Driver

To stop the MST driver, execute the following command:

```
mst stop
```

This command:

- Stops the Mellanox service
- Removes all special files/directories
- Unloads kernel modules

1.5.4 Restarting the MST Driver

To restart the MST drivers, execute the following command:

```
mst restart
```

This command effectively functions as does executing `mst stop` followed by `mst start`.

1.5.5 Displaying the MST Driver Status

To display the status of the MST driver, execute the following command:

```
mst status
```

This command displays a list of the loaded MST kernel modules and the local and remote devices recognized by the host. For example:

```
MST modules:
------------
    MST PCI module loaded
    MST PCI configuration module loaded
    MST Calibre (I2C) module is not loaded
MST devices:
------------
/dev/mst/mtusb                - MTUSB USB to I2C adapter as I2C master
/dev/mst/mt23108_pciconf0     - PCI configuration cycles access.
                                bus:dev.fn=01:01.0 addr.reg=88 data.reg=92
                                Chip revision is: 00
/dev/mst/mt23108_pci_cr0      - PCI direct access.
                                bus:dev.fn=02:00.0 bar=0xf7e00000size=0x100000
                                Chip revision is: 00
/dev/mst/mt23108_pci_ddr0     - PCI direct access.
                                bus:dev.fn=02:00.0 bar=0xe8000000 size=0x8000000
/dev/mst/mt23108_pci_uar0     - PCI direct access.
                                bus:dev.fn=02:00.0 bar=0xf7000000 size=0x800000
/dev/mst/vtop                 - Virtual to physical addresses tranlation driver
```

1.5.6 Starting an MST Server

To start an MST server, execute the following command:

```
mst server start [port]
```

where:

port – IP port where the MST server listens. The default is 23108.

This command starts an MST server to enable an incoming connection.:

1.5.7 Stopping an MST Server

To stop an MST server, execute the following command:

```
mst server stop
```

where:

This command stops an MST server.

1.5.8 Connecting to a Remote Server

To connect to a remote server, execute the following command:

```
mst remote add hostname[:port]
```

where:

hostname – An IP address or a host name, optionally followed by port number. (The default port is 23108.)

This command establishes a connection with the specified host on the specified port. This adds to the local devices list the devices on the remote peer.

1.5.9 Disconnecting from Remote Servers

To remove all remote devices on a remote host, execute the following command:

```
mst remote del hostname[:port]
```

where:

hostname – An IP address or a host name, optionally followed by port number. This must be specified exactly as it is in the `mst remote add` command described above.

This command removes from the local devices list all the devices on the specified host.

1.5.10 Saving PCI Configuration Headers

To save PCI configuration headers, execute the following command:

```
mst save
```

This command saves PCI configuration headers in the directory: /usr/mst/etc/pci. This command is invoked automatically when the “mst run” command is executed.

1.5.11 Loading PCI Configuration Headers

To load PCI configuration headers, execute the following command:

```
mst load
```

This command loads PCI configuration headers from the directory: /usr/mst/etc/pci.

1.5.12 Resetting a Mellanox PCI Device

To reset a Mellanox device attached to the PCI bus, execute the following command:

```
mst reset <device_name>
```

This command resets the device <device_name> and then carries out the command “mst load”. To get a list of currently available Mellanox devices, type “ls /dev/mst” or “mst status”. Devices connected to the PCI bus will have the string “pci” as a part of their name.

Part I: InfiniHost Software Tools and Utilities

The InfiniHost MT23108 software tool set consists of the following:

2 infinivision Tool

2.1 Supported Platforms

See "Supported Platforms and Operating Systems' (page 13).

2.2 Overview

Mellanox supplies the infinivision tool with the InfiniHost MT23108 device. infinivision provides compact and convenient viewing of the modules, registers and sub-fields accessible from the CR bus. infinivision can be used to directly edit CR values described in the *InfiniHost Programmer's Reference Manual (PRM*), such as host command interface registers.

2.3 Installation and Setup

The InifiniVision tool is part of the InfiniHost Tool Kit package, whose requirements and installation procedure are described in "Mellanox Software Tools (MST) Installation' (page 13).

2.4 Operation

The operations of the infinivision tool are described in the following sections:

- Start the infinivision tool (page 21).
- Select the device interface for accessing the registers of the InfiniHost device (page 22).
- Specify the notation (decimal or hex) infinivision displays (page 23).
- Select a register (page 23).
- Modify register values (page 23).
- Save CR space and other values to an external file (page 24.)
- Monitor fields in the Watch window (page 24)

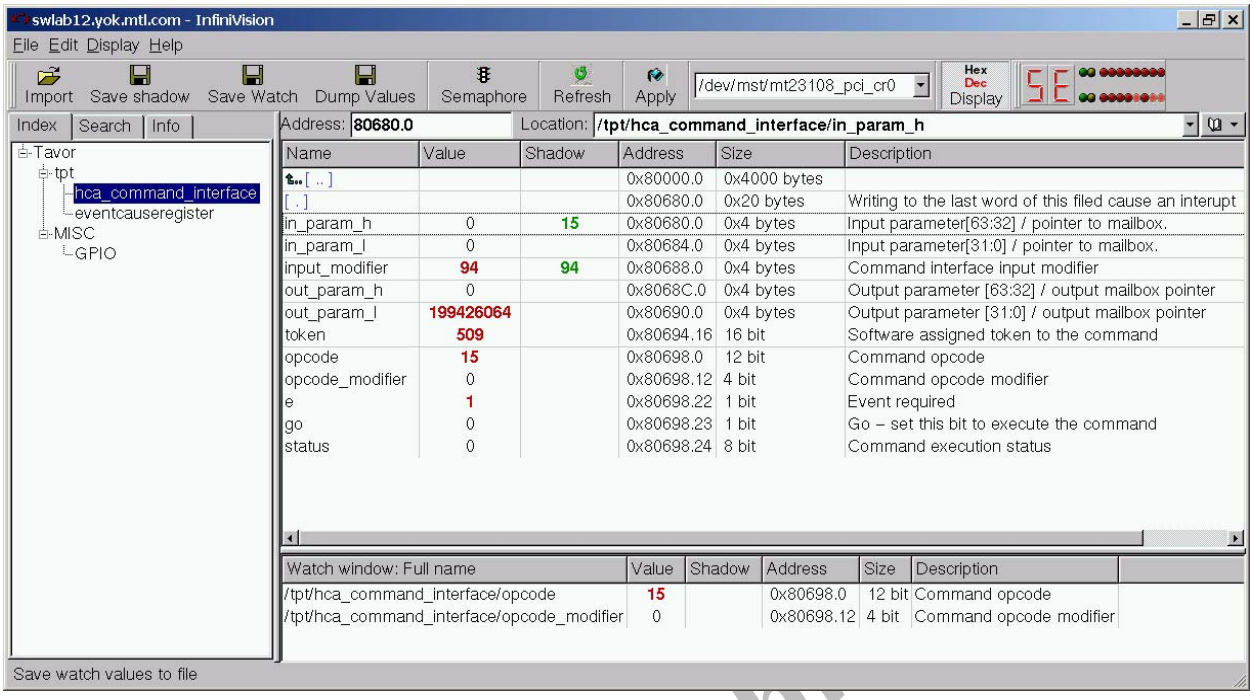
2.4.1 Start the infinivision Tool

The MST driver must be started prior to running infinivision tool. **To start infinivision:**

- Start the MST driver (mst start or mst restart).
- Enter `infinivision` at the command line.

This command displays the following window.

Figure 1: HCA Command Interface



2.4.2 Select the Device Interface to Access the InfiniHost Device

After infinivision has been started, you can select the interface used to access the registers of the InfiniHost device.

To select the device interface to access the device registers:

- From the select box (at the top and center of the window), select the device interface used to access the device registers.

The interfaces available in the select list are determined dynamically by the MST driver. The following interfaces may be available:

- **mtusb –** enables access via the I2C interface.
- **mtusb-1** - an improved version of mtusb.
- **mt23108_pci_cr0** – enables access via the PCI interface
- **mt23108_pciconf0** – enables access via the PCI interface in the case where the on-board EEPROM has intentionally been disabled (EEPROM_disable jumper is set). This is typically done if the firmware image has been serverly corrupted and normal boot is not possible. Before selecting this interface, you need to do the following:

  1. Shut down and disconnect the system.
  2. Remove the Mellanox board.
  3. Connect the GP5 jumper.
  4. Reconnect the board and reboot.

2.4.3 Specify the Notation Displayed

You can set the infinivision graphical interface to display values (in the "Value" and "Shadow" columns) in either decimal or hexadecimal notation, as follows:

To select the notation displayed.

- Click the **Hex Dec Display** button at the top of the window to toggle between decimal and hexadecimal notation for the values listed in the Value and Show columns.

2.4.4 Select a Register

The infinivision tool automatically reads the registers and the fields of the InfiniHost device through the device interface selected. (The procedure to select the device interface is described above, in "Select the Device Interface to Access the InfiniHost Device" on page 22. These registers and their associated fields are displayed in the infinivision navigation pane (on the left side of the screen).

To choose the register and fields you want to read or modify, expand the tree of the navigation pane and select the specific register to display.

To facilitate navigation among the different registers and fields, the infinivision utility includes a bookmark mechanism. Bookmark a particular register and return to the display for this bookmarked register at a later time, as follows:

To create and use bookmarks.

1. While the register to be bookmarked is selected, click the bookmark (book) icon located to the right of the Location list box.
2. Return to a bookmarked register by selecting a bookmark from the drop-down list opened by the arrow next to the bookmark icon.

2.4.5 Modify Register Values

The procedures to set and modify register values involve the "Shadow" mechanism – a buffer storing specified values until they are applied via the Apply button. The main screen of the infinivision interface contains a column labeled "Shadow," where the specified values are displayed in green before they are applied (written) to the register, after which they are displayed in black.

The infinivision utility enables you to view and modify the values assigned to registers and subfields of the InfiniHost device. You can:

- Load the default value of a particular register field to the Shadow column.
- Load the default values for all fields of a register to the Shadow column.
- Load the values from a previously saved file to the Shadow column.
- Modify register values, using the values in the Shadow column.
- Dump all CR space registers to a file.
- Save all Shadow values to a file.

To load to the Shadow column the default value of a particular register field:

1. Place the cursor in the field (the row) whose default value is to be loaded.
2. Right-click and select **Load Default value to Shadow**.
   This writes the value to the Shadow column, where it is displayed in green.

To load to the Shadow column the default values of all the fields of a register:

1. Place the cursor in a field of the register whose default values are to be loaded.
2. Right-click and select **Load ALL default values to Shadow**.
   This writes the values to the Shadow column, where they are is displayed in green.

To load values from a previously saved file to the Shadow column:

1. Place the cursor in a field of the register whose default values are to be loaded.
2. Select File | Import Shadow from the menu bar. This displays the Import Shadow from select box, enabling you to browse to the file (*.temt) you want to import.
3. This writes the values to the Shadow column, where they are is displayed in green.

To modify the values contained in a register.

- Click the Apply button located at the top of the screen.
  This applies the value displayed in the Shadow column to the Value column and writes it to the hardware register. This changes the Shadow column entry to black.

To dump the entire register space to a file.

1. Select the interface for accessing the device registers (as explained above, in “To select the device interface to access the device registers:” on page 22.
2. Select **File | Dump All Values to File** ... This displays the **Select target file for CR** space dialog box.
3. Browse to or type in a file name.
4. Click the Save button.

To save all Shadow values to a file.

1. Select the interface for accessing the device registers (as explained above, in “To select the device interface to access the device registers:” on page 22.
2. Select **File | Save Shadow**... This displays the **Export Shadow by Address** dialog box.
3. Browse to or type in a file name.
4. Click the Save button.

2.4.6 Monitor Fields in the Watch Window

The Watch window of at the bottom of the infinivision screen enables you to monitor fields of different nodes simultaneously. infinivision additionally enables you to set the Watch window automatically to update its display five times a second, and to save the values of the watch window to file.

To display the Watch window:

- Select **Display | Watch** from the menu bar.

To set the Watch window to update its display automatically:

- Select **Display | Watch timer updates** from the menu bar.

To save the watch window values to file:

- Select **File | Save Watch Values to File...** from the menu bar. This displays a dialog box where you specify/browse to the file where the watch values are saved.

To save the watch window Field names to file:

Select **File | Save Watch Configuration to File...** from the menu bar. This displays a dialog box where you specify the name of watch configuration file.

To restore the watch window Fields from file:

Select **File | Restore Watch Configuration from File...** from the menu bar. This displays a dialog box where you specify the name of watch configuration file whose fields you want to monitor.

3 itrace Utility

3.1 Supported Platforms

See "Supported Platforms and Operating Systems' (page 13).

3.2 Overview

The iTrace utility prints trace messages generated by iRISC processors of the InfiniHost device. These trace messages inform developers of software drivers about internal status, events, critical errors, etc., for each iRISC. Trace messages generated by iRISCs are stored in packed format in the trace buffer (usually allocated in DDR).

The iTrace utility both extracts and displays these messages.

iTrace is a command line application, controlled by command line parameters. It prints trace messages in text format to the console.

3.3 Installation and Setup

The iTrace utility is part of the MST toolkit, whose requirements and installation procedure are described in "Mellanox Software Tools (MST) Installation' (page 13).

3.4 Operation

You can run the iTrace utility only after the debug firmware had been loaded and the SYS_ENABLE command had been executed.

> Note: You can execute the "System Enable" command manually, as you may want to do if you want to run iTrace without the driver.
>
> 1. Write 0x00800001 to CR-Space register 0x80698
> This can be done with mwrite, as in the following example:
> **/usr/mst/bin/mwrite /dev/mst/mt23108_pci_cr0 0x80698 0x00800001**
>
> 2. Read the value from same address (0x80698). If you get 1, the SYS.ENA is OK
> This may be done by mread, for example:
> **/usr/mst/bin/mread /dev/mst/mt23108_pci_cr0 0x80698**

The MST driver must be started prior to running iTrace tool. **To start iTrace:**

- Start the MST driver (mst start or mst restart).
- Enter the following command:

```
iTrace [options...] IRISC_NAME
```

where:

IRISC_NAME – is the iRISC whose traces are to be printed. This can be specified once anywhere in the command line as a special option without the leading hyphen. For example:

```
itrace ntu -w
itrace -w ntu
```

iTrace recognizes the following iRISC names:

- exus
- exur
- tcu
- tpt
- qpc
- ntu

[options] can specify any of the following:

-h, --help – Displays help about iTrace usage.

-m, --mask=*TRACE_MASK* – Sets the Trace Mask.

To enable generating trace messages for iRISC, the trace_mask register must be set according to the specifications in the “Driver Debug Hooks” chapter in the *InfiniHost Programmer’s Reference Manual (PRM)*. Setting / clearing bits of the trace_mask register enables/disables generating specific types of trace messages. The `TRACE_MASK` parameter must be a hexadecimal or decimal number and its value will be written into the trace_mask register. Changing trace_mask will not change or remove messages previously stored in the trace buffer, so disabled types of messages still can be displayed by iTrace if they were previously generated.

Example: `itrace exur –m 0xFF0`

```
itrace -m=0xffffffff tcu
```

This generates output regarding the Sequence numbers, timestamps, and records of operations, such as the following:

```
IRISC Trace Viewer (Mellanox MT23108/InfiniHost, V3.10, Jan 16 2003 15:53:03)
FW Version: 1.0000.18 15/04/2003 10:20:59

(00000001 25b9a565) INFO: receive_ipc from irisc 4 opcode=0x01 #Data-dwords=7
(00000002 25b9a7df) INFO: IPCdata[00]=0x00000000
(00000003 25b9a859) INFO: IPCdata[01]=0x00000000
(00000004 25b9a8d2) INFO: IPCdata[02]=0x00000010
(00000005 25b9a94b) INFO: IPCdata[03]=0x00000000
(00000006 25b9aa0c) INFO: IPCdata[04]=0x00000000
(00000007 25b9aad7) INFO: IPCdata[05]=0x00200f90
(00000008 25b9aba2) INFO: IPCdata[06]=0x00800023
(00000009 25b9b47b) CMD_IF: cmdif_initiator: got command OpCode=0x23 Event? 0. will use
ptr 0x0x11e374 for HCR:
(0000000a 25b9b66a) DEBUG: ...[000]:00000000
(0000000b 25b9b6ec) DEBUG: ...[004]:00000000
(0000000c 25b9b766) DEBUG: ...[008]:00000010
(0000000d 25b9b7df) DEBUG: ...[00c]:00000000
(0000000e 25b9b890) DEBUG: ...[010]:00000000
(0000000f 25b9b95a) DEBUG: ...[014]:00200f90
(00000010 25b9ba25) DEBUG: ...[018]:00800023
(00000011 25b9cc53) CMD_IF: cmdif_committer: CMDIF_DO_CMDPROLOG
(00000012 25b9cedc) CMD_IF: CMD_IF: conf_special_qp:                conf_qp_num=0x10,
```

```
special_qp_num=0x000000
(00000013 25b9dee8) MAD: init_traps for port 1
(00000014 25b9e342) MAD: init_traps for port 2
.
.
.
```

-w, --wait – Runs iTrace in wait mode. iTrace will exit only if you press <Ctrl-C>. This is not the default behavior of iTrace. Without the –w option, iTrace will exit if there have been no new traces in the last 0.5 seconds.

-d, --d=*DEVICE* – Specifies the name of the MST device driver for accessing the cr-space. The default value is: /dev/mst/mt23108_pci_cr0. This option is useful for accessing trace buffers through I2C devices.

To run iTrace via the I2C interface, use this option to specify the following:

--d=*device*, where the *device* is an I2C device (such as InfiniBridge, or Caliber).

--**noddr** – Sets iTrace not to directly access DDR for reading the trace buffer but to use the NSI Gateway instead. By default, iTrace access DDR directly. If cr-space device specified in the –d parameter is one of the I2C devices, –noddr is switched on.

--**no-propel** – Sets iTrace not to animate the propeller in wait mode (-w option). By default, animation is enabled.

-v, --**version** – Prints version and exits

-c, --**color** – Enables color in trace output.

-D, --**dump** – Dumps the trace buffer and exits. This option is useful for debugging iTrace – it dumps the contents of the trace buffer in row format.

Note: Typing `--help` at the command line displays manual pages describing the syntax of the iTrace utility.

4 FLINT Utility

4.1 Overview

The FLINT (Flash interface) utility enables you to burn from command line the Flash memory from a binary image. The FLINT utility also enables you to execute various operations on Flash memory from the command line.

Note:

The FLINT utility does not keep the existing GUID, which you need to specify. See below, in "Burning the Entire Flash from a Raw Binary Image" on page 33, for the syntax for specifying the GUID.

Burning with the FLINT utility is not Failsafe.

The FLINT utility is supplied with its source code, which can serve as a reference for customer modifications.

4.2 Installation and Setup

The FLINT utility is part of the MST toolkit, whose requirements and installation procedure are described in Mellanox Software Tools (MST) Installation (page 13). The MST installation copies the FLINT tool to /usr/mellanox/bin.

4.3 Usage

```
flint [switches...] <command> [parameters...]
```

4.3.1 Switch Description

-bsn <BSN> - Mellanox Board Serial Number (BSN). Valid BSN format is:
MTxxxxx[-]R[xx]ddmmyy-nnn[-cc]
Commands affected: burn

-d[evice] <device> - Device flash is connected to.
Commands affected: all

-guid <GUID> - Base value for up to 4 GUIDs which are automatically assigned the following values:
guid -> node GUID
guid+1 -> port1
guid+2 -> port2
guid+3 -> system image GUID.
Commands affected: burn

-guids <GUIDs...> - 4 GUIDs must be specified here. These GUIDs will be assigned to:
node, port1, port2 and system image GUID respectively.

Commands affected: burn

-h[elp] - Prints this message and exits

-i[mage] <image> - Binary image file.
Commands affected: burn, verify

-nofs - Burn image not in failsafe manner.

-s[ilent] - Do not print burn progress flyer.
Commands affected: burn

-y[es] - Non-interactive mode. Assume the answer "yes" to all questions.
Commands affected: all

-vsd <vendor-specific-data> -A VSD string, composed of up to 208 characters, will be written to the VSD section in the flash. If not specified, the current VSD will be preserved.

-psid <Parameter-Set-ID> - The PSID string, composed of up to 16 characters, indicates the FW parameter which was used for the image generation. If not specified, current PSID will be preserved.

4.4 Operation

The FLINT utility enables you to do the following operations:

- Burn entire flash from raw binary image (page 33).
- Read one dword from flash (page 33).
- Write one dword to flash (page 34).
- Write one dword to flash without sector erase (page 34).
- Erase a sector (page 34).
- Verify entire flash (page 35).
- Query miscellaneous FW and flash parameters (page 35).

The MST driver must be started prior to running flint tool. **To start flint:**

- Start the MST driver (mst start or mst restart).
- Enter the following command:

```
flint -d <device> <command> [parameters...]
```

The following sections provide the command line syntax for each of the FLINT utility operations, together with an example of its use.

4.5 Burning the Entire Flash from a Raw Binary Image

The FLINT utility enables you to burn the Flash from a binary image.

To burn the entire Flash from a raw binary image, use the following command line:

```
flint -d <device> b file [GUID...]
```

where:

device – Device on which the flash is burned.

file – Name of the image file.

GUID (optional) – One or four GUIDs.

If 4 GUIDS are present here, they will be assigned as node, port1, port2 and system image GUIDs, respectively.
If only one GUID is present here, it will be assigned as node GUID. Its values +1, +2 and +3 will be assigned as port1, port2 and system image GUID, respectively.
If nothing is present here - no GUIDs will be changed.

Note: The GUID may be defined as 16-digit hexadecimal numbers with any delimiter between them. If less than 16 digits are provided, leading zeros will be inserted.

Examples:

```
flint -d /dev/mst/mt23108_pci_cr0 b image1.bin 12345678deadbeef
```

This burns flash from the "image1.bin" file, and assigns Node UID to 12345678deadbeef, port1 GUID to 12345678deadbef0, port2 GUID to 12345678deadbef1 and system image GUID to 12345678deadbef12

```
flint -d /dev/mst/mt23108_pci_cr0 b image1.bin 12:34:56:78:de:ad:be:ef
12,34,56,78,de,ad,be,e0 34.56.78.de.ad.be.e2 '56 78 de ad be e5'
```

This burns flash from the "image1.bin" file and assigns Node GUID to 12345678deadbeef, port1 GUID to 12345678deadbee0, port2 GUID to 00345678deadbee2 and system image GUID to 00005678deadbe5

```
flint -d /dev/mst/mt23108_pci_cr0 b image1.bin
```

This burns flash from the "image1.bin" file and does not change GUIDs – i.e., assigns them as they are presented in the image.

4.6 Reading a Word from Flash

To read one dword from Flash memory, use the following command line:

```
flint -d <device> rw addr
```

where:

device – the device the dword is read from.

addr – the address of the word to read.

Example:

```
flint -d /dev/mst/mt23108_pci_cr0 rw 0x20
```

4.7 Writing a dword to Flash

To write one dword to Flash memory, use the following command line:

```
flint -d <device> ww addr data
```

where:

device – the device the dword is written to.

addr – the address of the word to write.

data – the value of the word.

Example:

```
flint -d /dev/mst/mt23108_pci_conf01 ww 0x10008 0x5a445a44
```

4.8 Writing a dword to Flash Without Sector Erase

To write one dword to Flash memory without sector erase , use the following command line:

```
flint -d <device> wwne addr data
```

where:

device – the device the dword is written to.

addr – the address of the word to write.

data – the value of the word.

Example:

```
flint -d /dev/mst/mt23108_pci_cr0 wwne 0x10008 0x5a445a44
```

Note that a result may be dependent on the Flash type. Usually, bitwise and between the specified word and the previous Flash contents will be written to the specified address.

4.9 Erasing a Sector

To erase a sector that contains a specified address, use the following command line:

```
flint -d <device> e addr
```

where:

device – the device the sector is erased from.

addr – the address of a word in the sector that you want to erase.

Example:

```
flint -d /dev/mst/mtusb e 0x1000
```

4.10 Verifying the Entire Flash

To verify the entire Flash, use the following command line:

```
flint -d <device> v
```

where:

device – the device the sector is erased from.

Example:

```
flint -d /dev/mst/mtusb v
```

4.11 Query FW and flash parameters

To query miscellaneous FW and flash parameters, use the following command line:

```
flint -d <device> q
```

where:

device – Device on which the query is run.

Example:

```
flint -d /dev/mst/mt23108_pci_cr0 query
```

5 mstdump Utility

5.1 Overview

The mstdump utility dumps device internal configuration data. The data can be used by mellanox engineers for hardware troubleshooting.

5.2 Installation and Setup

mstdump is part of the MST toolkit, whose requirements and installation procedure are described in Tools Installation (page 14). The installation copies the mstdump utility to /usr/bin.

5.3 Operation

The MST driver must be started prior to running mstdump tool. **To start mstdump:**

- Start the MST driver (mst start or mst restart).
- Enter am mstdump command that complies with the following command syntax:

  mstdump <device file> > <dump file>

Example:

```
[root@swlab20 root]#   mstdump /dev/mst/mt23108_pci_cr0 > mt23108.dmp
```

This dumps the internal configuration data of the device mt23108_pci_cr0 into the file mt23108.dmp.

Part II: InfiniScale III Software Tools and Utilities

The InfiniScale III MT47396 software tool set consists of the following:

6 is3burn EEPROM Management Tool

6.1 Overview

The is3burn tool enables you either to burn a firmware image to the EEPROM residing on your InfiniScale III board or to save an EEPROM image to file for future use.

Prior to running this tool, the MST Driver must be started (mst start or mst restart).

6.2 Syntax

```
is3burn {OPTIONS}
```

The following *OPTIONS* may be specified in an arbitrary order. All options may be used in abbreviated form, but the abbreviation must be unique.

-silent – Print errors only.

-help – Print help message and exit.

-nowarn – Suppress some warnings.

-simulate – Simulate a burn process, but do not actually burn, even when the -burn option is specified.

-noverify – Do not verify EEPROM after burning.

-nofs – Do not create a FailSafe image. If this option is not specified, a FailSafe image is created by default.

-version – Print is3burn version number and exit.

-noread – Do not read image from board in order to obtain GUIDs. By default, the image is read.

-fw <*FW image filename*> – Specify the Mellanox-supplied FW image file.

-conf <*input config file filename*> – Specify user-configureable (.INI) file (See Note 2 below). See Appendix on page 89 for description of initialization file.

-wrconf <*output config file filename*> – Write user-configureable (.INI) file with updated values.

-burn <*device*> – Burn to specified device. The device list can be obtained by running the `MST status` command.

-image <*input EEPROM image filename*> – Specify an input EEPROM image.

-wrimage <*output EEPROM image filename*> – Write an EEPROM image to the specified file.

-reindeer – The target board is MTS2400 (Reindeer).

-format <output EEPROM image format> – Output image format. May be BINARY, DWORDS, or IMAGE. The default is IMAGE.

-*GROUP.PARAM* <*Value*> – Set the value of the specified parameter. Available parameters are determined by the firmware and can be seen in the configuration file generated by the -wrconf command. (See notes below).

Notes:

1. Some parameters of the *-GROUP.PARAM* option will not take effect unless the -noread option is specified. These parameters are *Special.sysimage_GUID*, *Special.node_descr,* and *Special.Node_GUID*.
2. Command line has higher priority than the initialization file supplied in -conf option. Values given to parameters in command line will override those defined in initialization file.

One of the input files, <*FW image*> or <*input EEPROM image*>, must be specified (but not both of them). According to the type of input file specified, the tool works in one of two modes:

1. <*FW image*> file is specified.
   The is3burn tool reads and parses this image. There are definitions of firmware parameters as well as firmware code itself inside the firmware image. When the firmware image is read and parsed successfully, is3burn may read miscellaneous firmware parameters from the configuration file or command line (miscellaneous options in the form *-GROUP.PARAM* <*Value*>). The tool modifies firmware accordingly and may burn the resulting firmware image to device (-burn option) or simply dump it to file (-wrimage option). The configuration file (-wrconf option) may also be written.

   Examples:

   ```
   is3burn -fw is3fw.afw -burn /dev/mst/mtusb-1
   ```

   Burns firmware from the is3fw.afw file to a target InfiniScale III device attached to the /dev/mst/mtusb-1 device. Does not modify parameters.

   ```
   is3burn -fw is3fw.afw -wrimage is3fw.img
   ```

   Writes EEPROMs image to file is3fw.img. This file may be used later for burning.

   ```
   is3burn -fw is3fw.afw -wrconf my_conf.ini
   ```

   Reads firmware from the is3fw.afw file and writes parameter initialization information to the my_conf.ini file. The file may be edited by user and used later in is3burn. The initialization file format is described in “.INI File Format” on page 89 in the Appendix.

   ```
   is3burn -fw is3fw.afw -conf my_conf.ini -burn /dev/mst/mtusb-1
   ```

   Reads firmware from the is3fw.afw file, reads user-specific parameter initialization from the file my_conf.ini, and burns the resulting image to a target InfiniScale III device attached to the /dev/mst/mtusb-1 device

   ```
   is3burn -fw is3fw.afw -PLL.PLLStabilizationTime 150 -wrimage is3fw.img
   ```

   Reads firmware from the is3fw.afw file, changes the PLLStabilizationTime value to 150, and writes the EEPROM image to the file is3fw.img.

   ```
   is3burn -fw is3fw.afw -conf my_conf.ini
   -PLL.PLLStabilizationTime 150 -burn /dev/mst/mtusb-1
   ```

   Reads firmware from the is3fw.afw file, reads user-specific parameter initialization from the my_conf.ini file, changes the PLLStabilizationTime value to 150, and burns the resulting image to a target InfiniScale III device attached to the /dev/mst/mtusb-1 device.

2. <input EEPROM image filename> is specified.

   The is3burn tool knows nothing about the details of the firmware image. The only action it takes is to burn the image to the device. The only options that can be used with the -image option are: -silent, -burn, -simulate, -verify.

   Example:

```
is3burn -image is3fw.img -burn /dev/mst/mtusb-1
```

Reads an already compiled image from is3fw.img and burns it to a target InfiniScale III device attached to the /dev/mst/mtusb-1 device.

6.3 The Initialization File

The is3burn tool uses an initialization file. This file can be written automatically (see the -wrconf option), which writes the default parameter values to file, and it can also be modified and read by the is3burn tool (see the -conf option).

It is possible to read in multiple initialization files; thus you can maintain different logical parameter sets.

Note: Each parameter value can be modified also in the -GROUP.PARAMETER command line option.

The structure of the initialization file is described in the Appendix of this document.

7 is3infinivision

7.1 Overview

Mellanox supplies the is3infinivision tool with the InfiniScale III device. is3infinivision provides compact and convenient viewing of the modules, registers and sub-fields accessible from the CR bus. is3infinivision can be used to directly edit CR values described in the *InfiniScale III MT47396 Programmer's Reference Manual (PRM*), such as host command interface registers.

7.2 Operations

The operations of the is3infinivision tool are described in the following sections:

- Start the is3infinivision tool (page 45).
- Select the device interface for accessing the registers of the InfiniScale III device (page 46).
- Specify the notation (decimal or hex) is3infinivision displays (page 46).
- Select a register (page 47).
- Modify register values (page 47).
- Save CR space and other values to an external file (page 48.)
- Monitor fields in the Watch window (page 48)

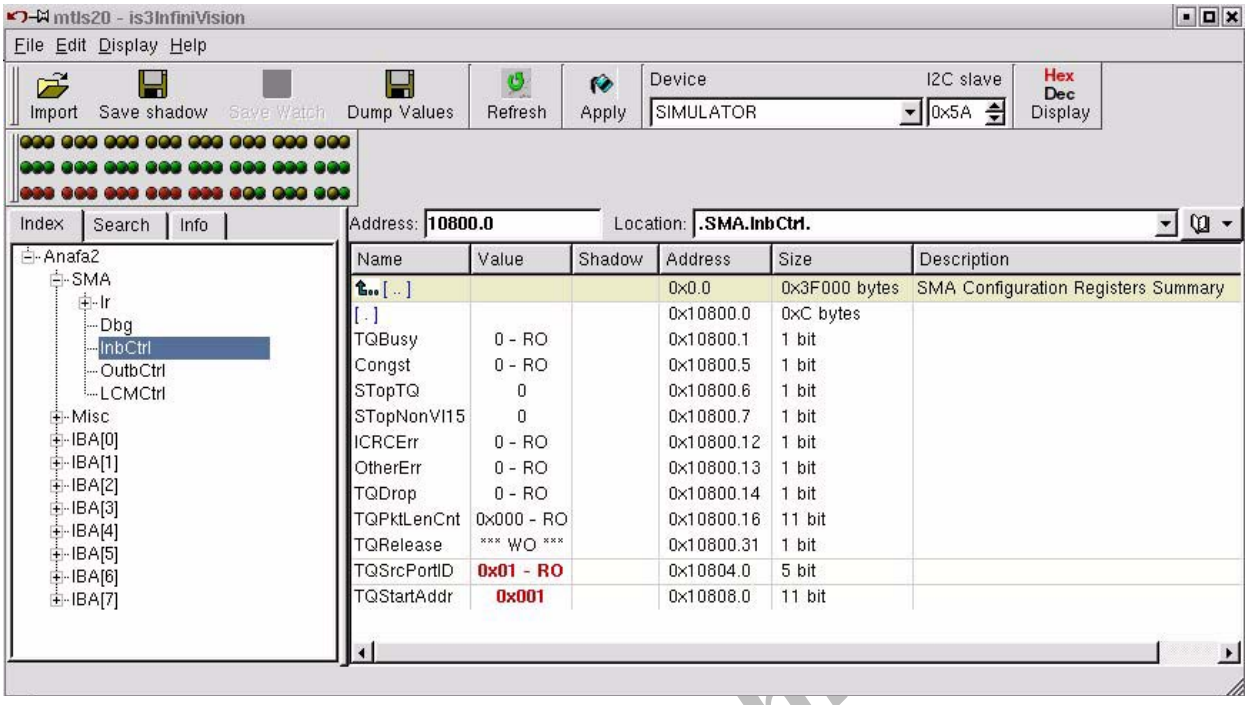
7.2.1 Start the is3infinivision Tool

The MST driver must be started prior to running is3infinivision tool. **To start is3infinivision:**

- Start the MST driver (mst start or mst restart).
- Enter `is3infinivision` at the command line.

This command displays the following window.

Figure 2: HCA Command Interface



7.2.2 Select the Device Interface to Access the InfiniScale III Device

After is3infinivision has been started, you can select the interface used to access the registers of the InfiniScale III device.

To select the device interface to access the device registers:

- From the Device select box (at the top and center of the window), select the device interface used to access the device registers.

The interfaces available in the select list are determined dynamically by the MST driver. The following interfaces may be available:

- **mtusb –** enables access via the I2C interface.
- **mtusb-1** - an improved version of mtusb
- **mt47396_ppc0** – enables access via the PPC interface

7.2.3 Specify the Data Notation Displayed

You can set the is3infinivision graphical interface to display values (in the “Value” and “Shadow” columns) in either decimal or hexadecimal notation, as follows:

To select the data notation displayed.

- Click the **Hex Dec Display** button at the top right of the window to toggle between decimal and hexadecimal notation for the values listed in the Value and Shadow columns.

7.2.4 Select a Register

The is3infinivision tool automatically reads the registers of the InfiniScale III device through the device interface selected. (The procedure to select the device interface is described above, in "Select the Device Interface to Access the InfiniScale III Device" on page 46. These registers are displayed in the is3infinivision navigation pane on the left side of the screen).

To choose the register or register fields you want to read or modify, expand the tree of the navigation pane and select the specific registers to display.

To facilitate navigation among the different registers and fields, the is3infinivision utility includes a bookmark mechanism. Bookmark a particular register and return to the display for this bookmarked register at a later time, as follows:

To create and use bookmarks.

1. While the register/field to be bookmarked is selected, click the bookmark (book) icon located to the right of the Location list box.
2. Return to a bookmarked register by selecting a bookmark from the drop-down list opened by the arrow next to the bookmark icon.

7.2.5 Modify Register Values

The procedures to set and modify register values involve the "Shadow" mechanism – a buffer storing specified values that can be applied via the Apply button. The main screen of the is3infinivision tool contains a column labeled "Shadow," where the specified values are displayed in green before they are applied (written) to the register, after which they are displayed in black.

The is3infinivision tool enables you to view and modify the values assigned to registers and subfields of the InfiniScale III device. You can:

- Load the default value of a particular register field to the Shadow column.
- Load the default values for all fields of a register to the Shadow column.
- Load the values from a previously saved file to the Shadow column.
- Modify register values, using the values in the Shadow column.
- Dump all CR space registers to a file.
- Save all Shadow values to a file.

To load to the Shadow column the default value of a particular register field: 1.

1. Place the cursor in the field (the row) whose default value is to be loaded.
2. Right-click and select **Load Default value to Shadow**.
   This writes the value to the Shadow column, where it is displayed in green.

To load to the Shadow column the default values of all the fields of a register:

1. Place the cursor in a field of the register whose default values are to be loaded.
2. Right-click and select **Load ALL default values to Shadow**.
   This writes the values to the Shadow column, where they are is displayed in green.

To load values from a previously saved file to the Shadow column:

1. Place the cursor in a field of the register whose default values are to be loaded.

2. Select **File** | **Import Shadow** from the menu bar. This displays the Import Shadow from select box, enabling you to browse to the file (*.temt) you want to import.
3. This writes the values to the Shadow column, where they are is displayed in green.

To modify the values contained in a register:

- Click the Apply button located at the top of the screen.
  This applies the value displayed in the Shadow column to the Value column and writes it to the hardware register. This changes the Shadow column entry to black.

To dump the entire register space to a file:

1. Select the interface for accessing the device registers (as explained above, in “To select the device interface to access the device registers:” on page 46.)
2. Select **File | Dump All Values to File** ... This displays the **Select target file for CR space** dialog box.
3. Browse to or type in a file name.
4. Click the Save button.

To save all Shadow values to a file.

1. Select the interface for accessing the device registers (as explained above, in “To select the device interface to access the device registers:” on page 46.
2. Select **File | Save Shadow**... This displays the **Export Shadow by Address** dialog box.
3. Browse to or type in a file name.
4. Click the Save button.

7.2.6 Monitor Fields in the Watch Window

The Watch window at the bottom of the is3infinivision screen enables you to monitor fields of different nodes simultaneously. is3infinivision additionally enables you to set the Watch window automatically to update its display five times a second, and to save the values of the watch window to file.

To display the Watch window:

- Select **Display | Watch** from the menu bar.

To set the Watch window to update its display automatically:

- Select **Display | Watch timer updates** from the menu bar.

To save the watch window values to file:

- Select **File | Save Watch Values to File...** from the menu bar. This displays a dialog box where you specify/browse to the file where the watch values are saved.

To save the watch window Field names to file:

Select **File | Save Watch Configuration to File...** from the menu bar. This displays a dialog box where you specify the name of watch configuration file.

To restore the watch window Fields from file:

Select **File | Restore Watch Configuration from File...** from the menu bar. This displays a dialog box where you specify the name of watch configuration file whose fields you want to monitor.

8 is3itrace Utility

8.1 Overview

The is3itrace utility prints trace messages generated by iRISC processor of the InfiniScale III device. These trace messages inform developers of software drivers about internal status, events, critical errors, etc. Trace messages generated by the iRISC are stored in packed format in the firmware trace buffer.

The is3itrace utility both extracts and displays these messages.

is3itrace is a command line application, controlled by command line parameters. It prints trace messages in text format to the console.

8.2 Operation

The MST driver must be started prior to running is3itrace tool. **To start is3itrace:**

- Start the MST driver (mst start or mst restart).
- Run is3itrace with the following command line syntax:

  `is3itrace [`*`options`*`...]`

where [*options*] can specify any of the following:

-b, **--behavior**=*BEHAVIOR* – Sets the behavior of is3itrace when the trace buffer is full (0=none, 1=discard new, 2=wait).

-d, --d=*DEVICE* – Specifies the name of the MST device driver for accessing the cr-space. Currently, there is no default device.

This option is useful for accessing trace buffers through I2C devices.

To run is3itrace via the I2C interface, use this option to specify the following:

--d=*device*, where the *device* is an I2C device (such as InfiniBridge, or Caliber). (You can additionally use the --i2c option, described below).

-D, **--dump** – Dumps the trace buffer and exits. This option is useful for debugging is3itrace – it dumps the contents of the trace buffer in row format.

-e, --error=*ERRORL* – Sets the maximum level of the error messages generated by the firmware.

-f, **--hashfile**=***HASHFILE*** – itrace format strings file (default: ir-objs/hashfile)

-h, --help – Displays help about is3itrace usage.

--i2c**=*I2C_SLAVE*** – Sets i2c_slave address. InfiniScale III default: 0x51

-m, --mask=*TRACE_MASK* – Sets the Trace Mask.

Setting /clearing bits of the trace_mask register enables/disables generating specific types of trace messages. The `TRACE_MASK` parameter must be a hexadecimal or decimal number and its value will be written into the

trace_mask register. Changing trace_mask will not change or remove messages previously stored in the trace buffer, so disabled types of messages still can be displayed by is3itrace if they were previously generated.

Example: `is3itrace –m 0xFF0`

`is3itrace -m=0xfffffffff`

This generates output regarding the Sequence numbers, timestamps, and records of operations, such as the following:

```
IRISC Trace Viewer (Mellanox MT47396/InfiniScale III, V3.0, Oct 13 2003 12:29:22)
FW Version: 0.0000.0 00/00/00 0:0:0

1  1454a050:  uitrace.c:86      :Initial itrace message 1 2 3
2  1454a0e4:  boot2.c:546     :Welcome to boot2
3  145da209:  load_sections.c:409     :Starting load section code
4  145e1b20:  load_sections.c:531     :Special Structures section
5  145e1c39:  load_sections.c:271     :Struct ID 1 - System Image GUID
6  145e1e17:  load_sections.c:280     :Struct ID 2 - Node Description
7  145e26d5:  load_sections.c:289     :Struct ID 3 - Board ID
8  145e27dc:  load_sections.c:271     :Struct ID 1 - System Image GUID
9  145e29c9:  load_sections.c:389     :Unknown special structure type 0
10  145e3ee6:  load_sections.c:449     :Address-Data pairs section
11  145e66d4:  load_sections.c:465     :Block Initialization section.
                                        Addr = 0x3f804, Size = 32 bytes  , 8 dwords
12  145f45cc:  load_sections.c:497     :Read-Modify-Write section
13  14678e26:  load_sections.c:465     :Block Initialization section. Addr = 0x20000,
                                        Size = 73444 bytes  , 18361 dwords
14  157e75ed:  load_sections.c:525     :Jump-Address section.
                                        Jump Address = 0x20000020
15  157e857a:  load_sections.c:538     :Section Loader reached Last Data Record
.
.
```

-n, **--start**=START_NO or **--start**=now. Sets first message number to display.

--**no-propel** – Eliminates the activity display (propeller) in wait mode (-w option). By default, animation is enabled.

-t, --trace=*TRACEL* – Sets the maximum level of the trace messages to be generated by the firmware.

-v, **--version** – Prints version and exits.

-w, --wait – Runs is3itrace in wait mode. is3itrace will exit only if you press <Ctrl-C>. This is not the default behavior of is3itrace. Without the –w option, is3itrace will exit if there have been no new traces in the last 0.5 seconds.

9 i2c Utility

9.1 Overview

The i2c utility provides low level access to the I2C bus on MTEK43132-M16-5 (Gnu) and MTEK43132-M96-2P (Gazelle) switch platforms, enabling you to read or write data.

9.2 Installation and Setup

The i2c utility is part of the MST toolkit, whose requirements and installation procedure are described in Tools Installation (page 14).

9.3 Operation

The MST driver must be started prior to running i2c tool. **To start i2c:**

- Start the MST driver (mst start or mst restart).
- Run i2c with the following command line syntax:

  `i2c [OPTIONS] <device> <cmd> <i2c_addr> <addr> [<data>]`

where `[OPTIONS]` can be the following:

`-h` – Prints this message.

`-a` <***addr_width***> – Sets address width (in bytes) to the specified value. May be 0, 1, 2 or 4. Default is 1.

`-d` <***data_width***> – Sets data width (in bytes) to the specified value. May be 1, 2 or 4. Default is 1.

`-x <data_len>` – Presents each byte of data as two hexadecimal digits (such as 013C20343B). Note that this option is mutually exclusive with the "-d" option.

The remaining parameters are:

<***device***> – Valid MST device.

<*cmd*> – Command. May be "r[ead]" or "w[rite]".

<*i2c_addr*> – I2C slave address.

<***addr***> – Address (of length *addr_width*) inside I2C target device to read/write operation.

<***data***> – Data (bytes of length *data_width*) to write to target device.

Note that <*addr*> value is ignored if <*addr_witdh*> == 0

All parameters are interpreted as hexadecimal values. The application return code is zero only when read/write was successfully completed.

Examples:

1. Read two bytes from address 0 of target I2C device at address 0x50:

```
> i2c -a 1 -d 2 /dev/mst/calibre r 0x50 0x00
0000
```

2. Write two bytes to the address above then read them:

```
> i2c -a 1 -d 2 /dev/mst/calibre w 0x50 0x00 0x1234
> i2c -a 1 -d 2 /dev/mst/calibre r 0x50 0x00
3412
```

3. Read (as separate) 16 bytes in hexadecimal format starting from address 0 of the target device above:

```
> i2c -a 1 -x 16 /dev/mst/calibre r 0x50 0x00
12340000000000000000000000000000
```

10 isw Utility

10.1 Overview

The isw utility provides access, via the I2C MST device, to one of the boards/chips on the MTEK43132-M96-2P/ MTEK43132-M16-S (Gazelle/Gnu) switch platforms. You use this utility to connect to a device on a Leaf or Spine board of the switch platform, after which you can run the other tools affecting devices, such as:

- adevmon: Device Monitoring Utility (page 75)
- EMT Tool (page 59)
- eburn Utility (page 77)

10.2 Installation and Setup

The isw utility is part of the MST toolkit, whose requirements and installation procedure are described in Tools Installation (page 14).

10.3 Operation

The MST driver must be started prior to running isw tool. **To start isw:**

- Start the MST driver (mst start or mst restart).
- Run isw with the following command line syntax:

  `isw` [*OPTIONS*] <*DEVICE*> <*TARGET*>

where [OPTIONS] are:

`-gazelle` – Board type is MTEK43132-M96-2P (Gazelle). This is the default.

`-ch` – Uses chassis access I2C bridges (MTEK43132-M96-2P; Gazelle only).

-`gnu` – Board type is MTEK43132-M16-S (Gnu).

`-leaf` – Board type is standalone MTEK43132-MSX12-4x (LEAF).

`-spine` – Board type is standalone MTEK43132-SP8 (SPINE).

`-h` – Prints this message.

`DEVICE` – A valid MST device.

- ***TARGET*** – For the MTEK43132-M96-2P (Gazelle) board, this should be in the format <*BRD*><*CHIP*>|`CH`, where:

  <*BRD*> is a board ID, such as L1-L12 or S1-S6 or PS (Power supply)
  <*CHIP*> is a chip ID, such as A1-A7 (InfiniScale 1-7)
  CH represents Chassis).
  Note that if BRD is PS, no CHP is required.

For an MTEK43132-M16-S (GNU) board, TARGET is simply the CHIP definition (A1-A7).

Examples:

```
isw /dev/mst/calibre          L1A3
isw /dev/mst/calibre          L2A7
isw /dev/mst/calibre          S3A2
isw /dev/mst/mtusb            L4CH
isw /dev/mst/mtusb            mtusbPS
isw -gnu /dev/mst/mtusb       A4
```

Part III: InfiniScale Mellanox Software Tools (isMST) and Utilities

The InfiniScale MT43132 software tool set (isMST) contains the following:

- "EMT Tool' (page 59)
- "adevmon: Device Monitoring Utility' (page 75)
- "eburn Utility' (page 77)
- "gz_burn Utility' (page 79)

In addition to the tools listed above, isMST also includes the following two tools which have been presented in Part II of this book as part of the is3MST tool set:

- "i2c Utility' (page 53)
- "isw Utility' (page 55)

11 EMT Tool

11.1 Overview

This chapter describes the Mellanox EEPROM Management Tool (EMT). EMT enables you to burn Firmware and its default boot-time configuration to the EEPROMs residing on boards hosting Mellanox InfiniScale and InfiniBridge devices. EMT also enables you to prepare an image that you can save and later burn to the EEPROM via the command line.

You can use EMT for burning Firmware via the following interfaces:

- **PCI**
- **I^2C** – CALIBRE ISA I^2C Card
- **USB I^2C**

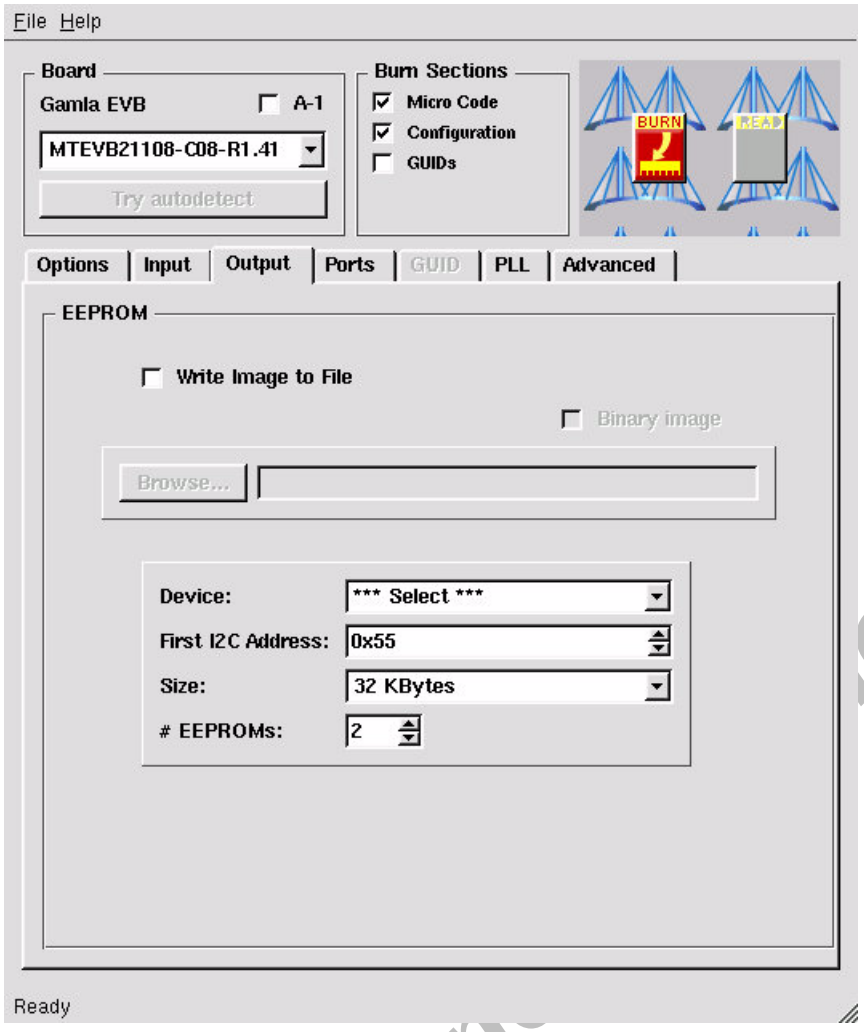
The support for these interfaces on the supported operating systems is described in Section 11.3, "Selecting the Type of Board," on page 60.

11.2 Starting the EMT Tool

The MST driver must be started prior to running EMT tool. **To start EMT:**

- Start the MST driver (mst start or mst restart).
- Enter `EMT` at the command line.

Upon starting, the EMT utility displays the following initial **EMT – EEPROM Manager Tool** window:

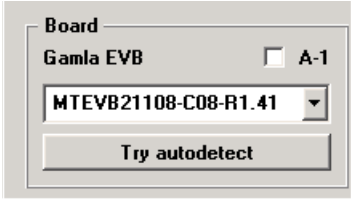


11.3 Selecting the Type of Board

The EMT utility enables you to select the board to whose EEPROMs you want to burn the Firmware.

The upper left corner of the main window of EMT contains the following Board frame, enabling you to choose (from a list of supported boards) the board where you want to burn the Firmware.

Note: JNI's MT21108 (GAMLA) board is also supported. (For information about the default configuration of this board, see "JNI MT21108 Board: Default Configuration" on page 73.).



To select the board to which EMT burns the firmware:

1. From the select box, choose the board.
   – ***or*** –
   Click **Try autodetect** to automatically detect and select the board where the firmware will be burned.
2. From the Board frame (shown above) located at the top of the main window, choose the board where you want to burn the firmware.

The A-1 check box indicates which version of the InfiniBridge (Gamla) MT21108 device is included with this board. If the box is checked, the board is using the most current version of the InfiniBridge device; if the box is not checked, the board includes a previous version of the device.

11.4 Selecting Parts of the EEPROM to Burn

EMT enables you to burn only selected parts of an EEPROM while leaving other parameters unchanged. You can select the parts you want to change in the Burn Sections frame of the EMT interface, located to the right of the Board frame.

To select the parts of the EEPROM to change:

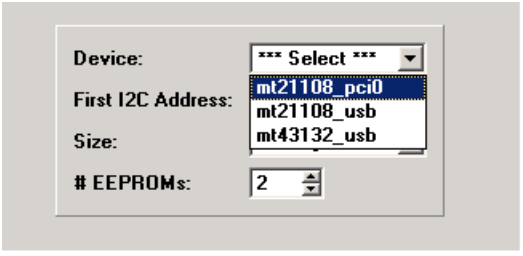
- In the following Burn Sections frame, select the sections of the EEPROM you want to change.

Each of the three options has its own configuration tab that is enabled only when its corresponding option is selected.

- Selecting **Micro Code** enables the Input tab, where you specify the Ucode and configuration files that EMT burns to the EEPROMs or saves to file. This tab and its operations are described in Section 11.6, "Selecting the Firmware that Is Burned to EEPROMs," on page 63
- Selecting **Configuration** enables the Options tab, where you set the operational mode of the device to which you are burning the firmware. This tab and its operations are described in Section 11.7, "Selecting the Operational Mode of the Device," on page 65.
- Selecting **GUID** enables the GUID tab, where you configure/change the board's GUID. This tab and its options are described in Section 11.8, "Configuring the GUID of the Board," on page 69.

11.5 Selecting the Interface for Burning the EEPROM

In the following section of the Options tab (of the main EMT window), you can select the interface EMT uses to burn the EEPROM:

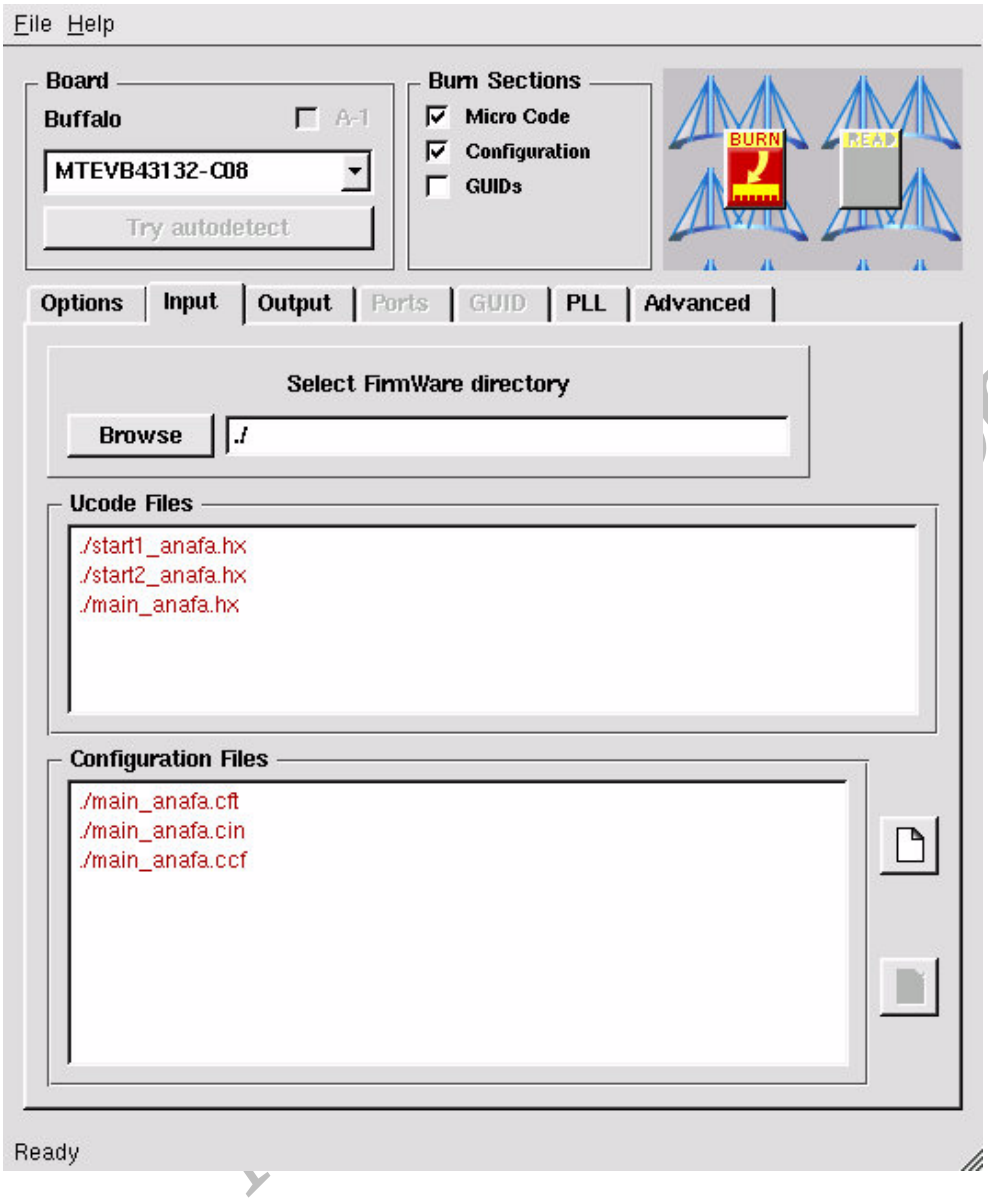


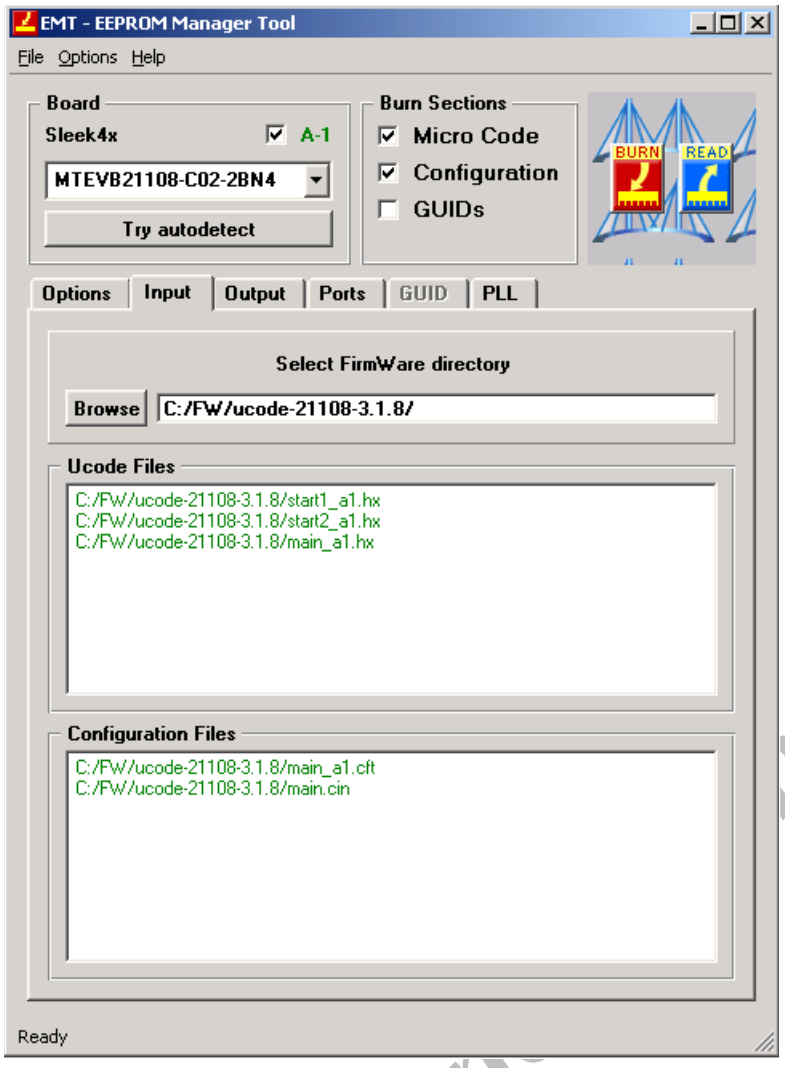
These boxes and fields enable you to specify the following:

- **Device** – Displays a list of interfaces through which EMT can carry out the burning process.
- **First I^2C Address** – The address of the EEPROM on the I2C bus.
- **Size** – The size of the EEPROM on the board.
- # EEPROMS – The number of EEPROMs to which EMT burns the firmware.

11.6 Selecting the Firmware that Is Burned to EEPROMs

The following Input tab defines the Ucode and Configuration files constituting the Firmware that EMT burns to the EEPROMs or saves to file:





These files are chosen automatically by EMT once the board type is selected in the Board frame.

> Note: If the files are displayed in green, EMT has found them in their default location. If the files are displayed in red, EMT has not located them. All the files must reside in the same directory and you must browse (via the Browse button) to the files' location.

11.6.1 Saving the Firmware to File

After you have specified the Firmware that you want to burn to the EEPROM, EMT enables you to save the Firmware to an image on a file that you can burn to a target EEPROM later, via the command interface. Burning the saved Firmware image via the command interface is described in "Running the EMT Command Line" on page 73.

To save the firmware to file:

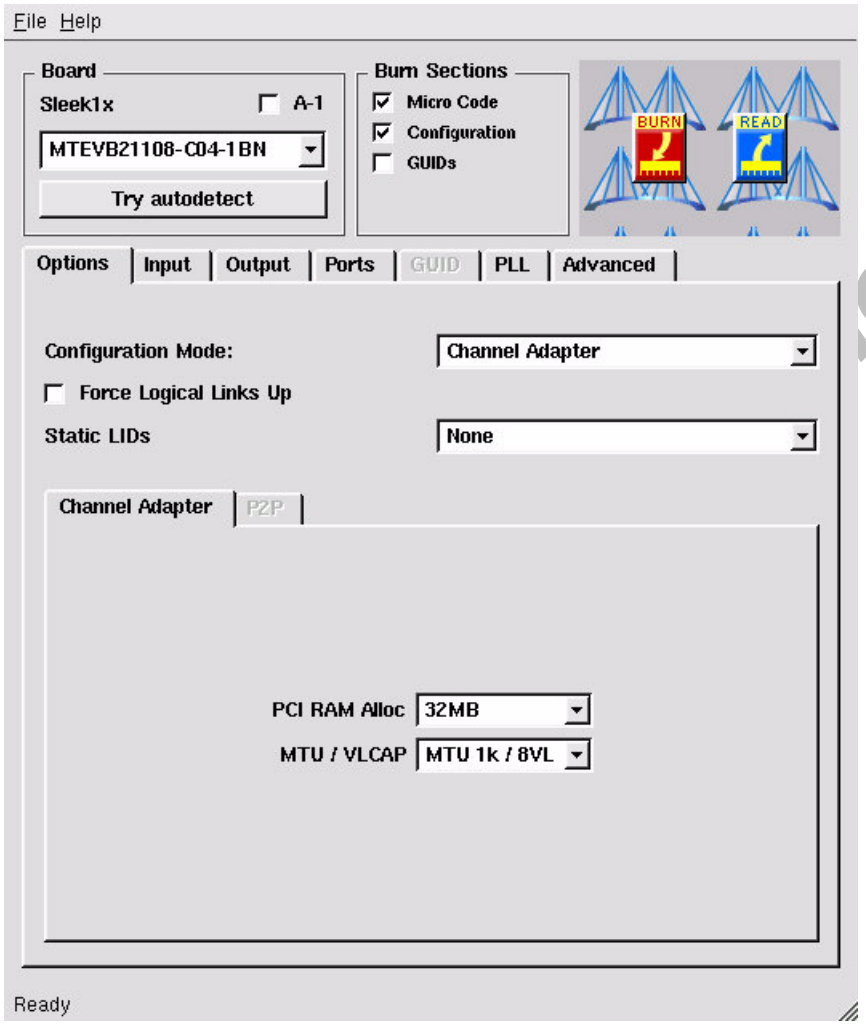
1. Select the Write Image to File check box, located near the top of the Output tab of the main EMT window.
2. Click the Browse button and specify the name and location where this file will reside.

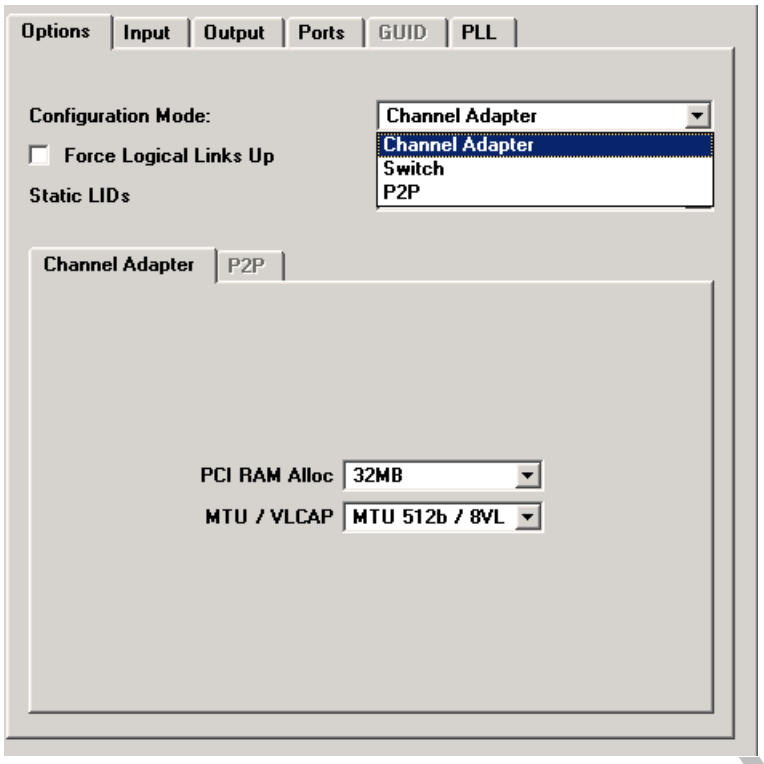
11.7 Selecting the Operational Mode of the Device

EMT enables you to set the operational mode of the device to which you are burning the Firmware. You can configure the device to function as:

- A channel adapter, in which case it functions as an HCA and/or TCA.
- An IB switch.
- A PCI-to-PCI bridge over an IB link.

You can set the operational mode and its associated parameters via the Options tab of the main EMT window. The following tab is enabled when the Configuration check box in the Burns Section frame is selected:



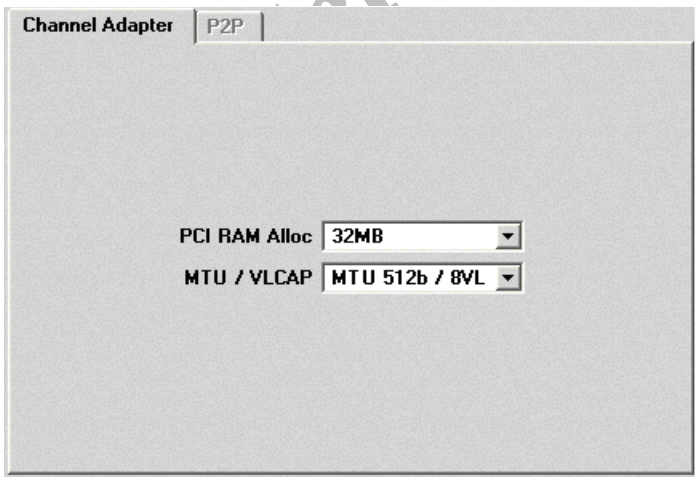


To set the mode of operation of the device you are burning:

- Select the mode of operation from the Configuration Mode select box. The options are:

  Channel Adapter – Sets the device to function as a channel adapter (HCA and/or TCA). The parameters associated with the Channel Adapter operational mode are described below, in "Setting Parameters for Channel Adapter Operations" on page 67.
  Switch – Sets the device to function as an IB switch.
  P2P – Sets the device to function as a PCI-to-PCI bridge over an IB link. The parameters associated with the P2P operational mode are described below, in "Setting Parameters for Channel Adapter Operations" on page 67.

When Channel Adapter is selected in the Configuration Mode select box, the following Channel Adapter tab is enabled:



This tab contains the following fields where you set parameters for the Channel Adapter operational mode:

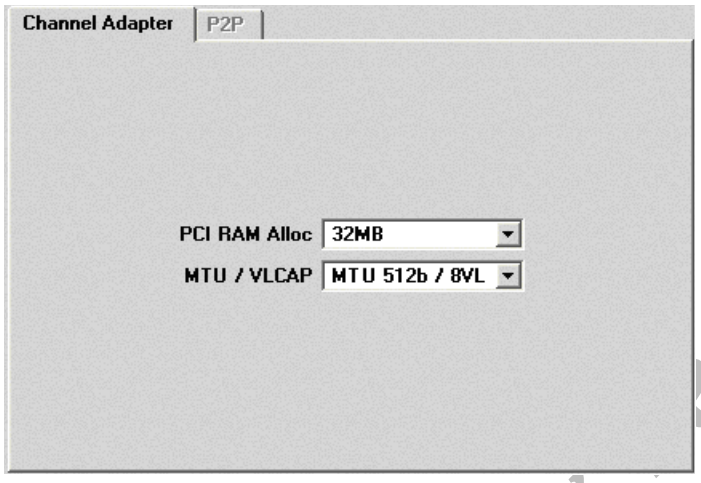
- PCI RAM Alloc – Specifies the amount of RAM the device will request from the PCI controller.
- MTU/VLCAP – Specifies the Maximum Transfer Unit (MTU) and Virtual Lane Capabilities (VLCAP) that are supported by the device.

This Options tab additionally has the following fields setting operational parameters of the device:

- **Force Logical Links Up–** When selected, sets the device to force logical links at boot-time.
- **Static LIDs** – Enables you to select the static LID corresponding to the logical port in use by the board.

11.7.1 Setting Parameters for Channel Adapter Operations

When Channel Adapter is selected in the Configuration Mode select box, the following Channel Adapter tab is enabled:

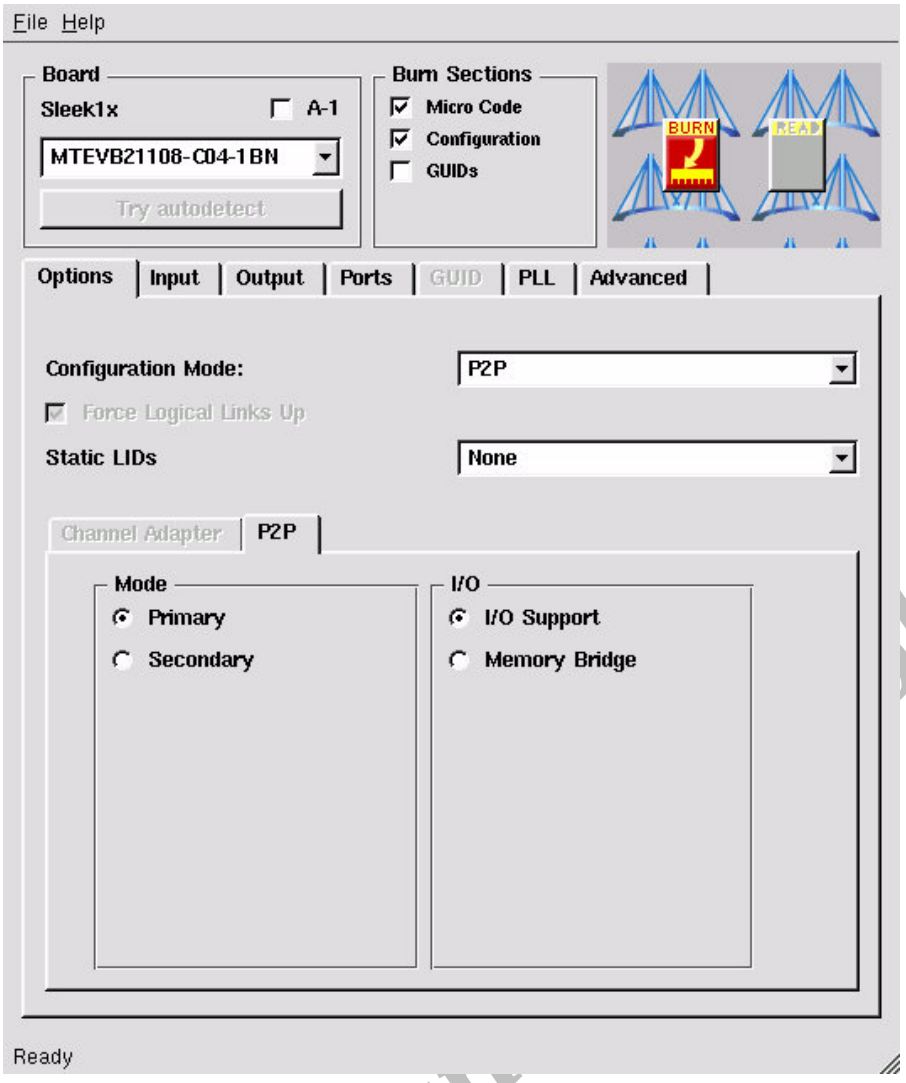


This tab contains the following fields where you set parameters for the Channel Adapter operational mode:

- PCI RAM Alloc – Specifies the amount of RAM the device will request from the PCI controller.
- MTU/VLCAP – Specifies the Maximum Transfer Unit (MTU) and Virtual Lane Capabilities (VLCAP) that are supported by the device.

11.7.2 Setting PCI-to-PCI Operations

If P2P is selected in the Configuration Mode select box, the following P2P tab is enabled:



This tab contains the following fields where you set parameters for the PCI-to-PCI (P2P) operational mode:

Static LIDs – Assigns a static LID (local ID) for both the Primary and Secondary EVB InfiniBridge boards. For both boards you need to select the static LID corresponding to the logical port in use.

The P2P tab includes the Mode and I/O sections, with the following options:

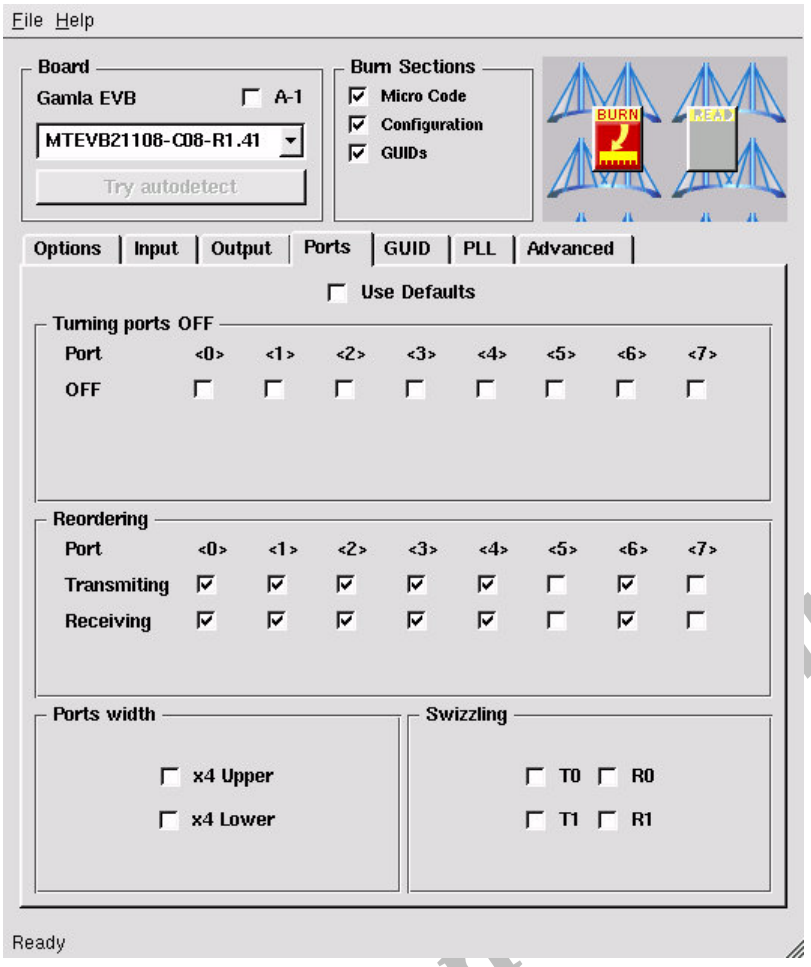
Mode – Specifies whether the selected board functions in Primary or Secondary mode.

I/O – Specifies the types of transactions that the P2P bridge transfers between the end nodes of the P2P bridge.

I/O Support – Sets the P2P bridge to transfer PCI transactions and memory transactions between the end nodes.
Memory Bridge – Sets the P2P bridge to transfer only memory transactions between the end nodes of the bridge.

11.7.3 Controlling the Device's Ports

The following Ports tab enables you to control the ports of the device whose EEPROMs you have selected to burn.



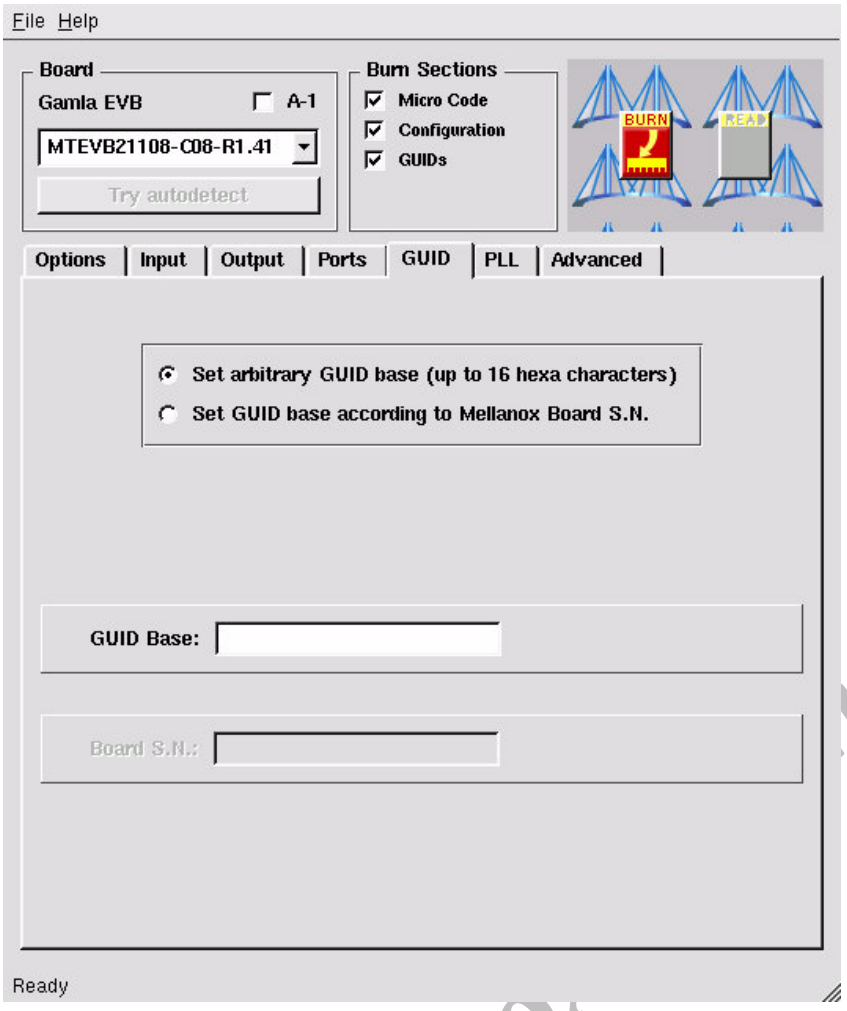
The Ports tab contains the following options:

- **Use defaults** – Sets the device to use the default ports.
- **Turning ports OFF** – Turns off a selected port by stopping the power supply to it.
- **Reordering** – reorders the bits of the port, for both the Transmit and Receive directions. The reordering of bits can be changed from MSB to LSB, and vice versa.
- **Ports width** – Defines whether the upper four ports, lower four ports, or both, work in 4x mode.
  Note: This is relevant for the MT21108 device.
- **Swizzling** – Port BIT reordering for boards with two 4x ports:

  T0 and R0 = Port 0
  T1 and R1 = Port 1

11.8 Configuring the GUID of the Board

If the GUIDs check box is selected in the Burns Sections frame, the GUID tab is enabled. This tab enables you to define the global identifier of the board to which you are burning the Firmware.



This tab contains the following fields which let you define the GUID for the board you are burning:

- **Set abitrary GUID base (up to 16 hexadecimal characters)** – Selecting this button enables the GUID Base box, where you can specify a...???
- **Set GUID base according to Mellanox Board S.N.** – Selecting this button enables the Board S.N. box, where you enter the serial number of your Mellanox-supplied board.
-
- **GUID Base** – The 64-bit Global Unique Identifier assigned to the device.

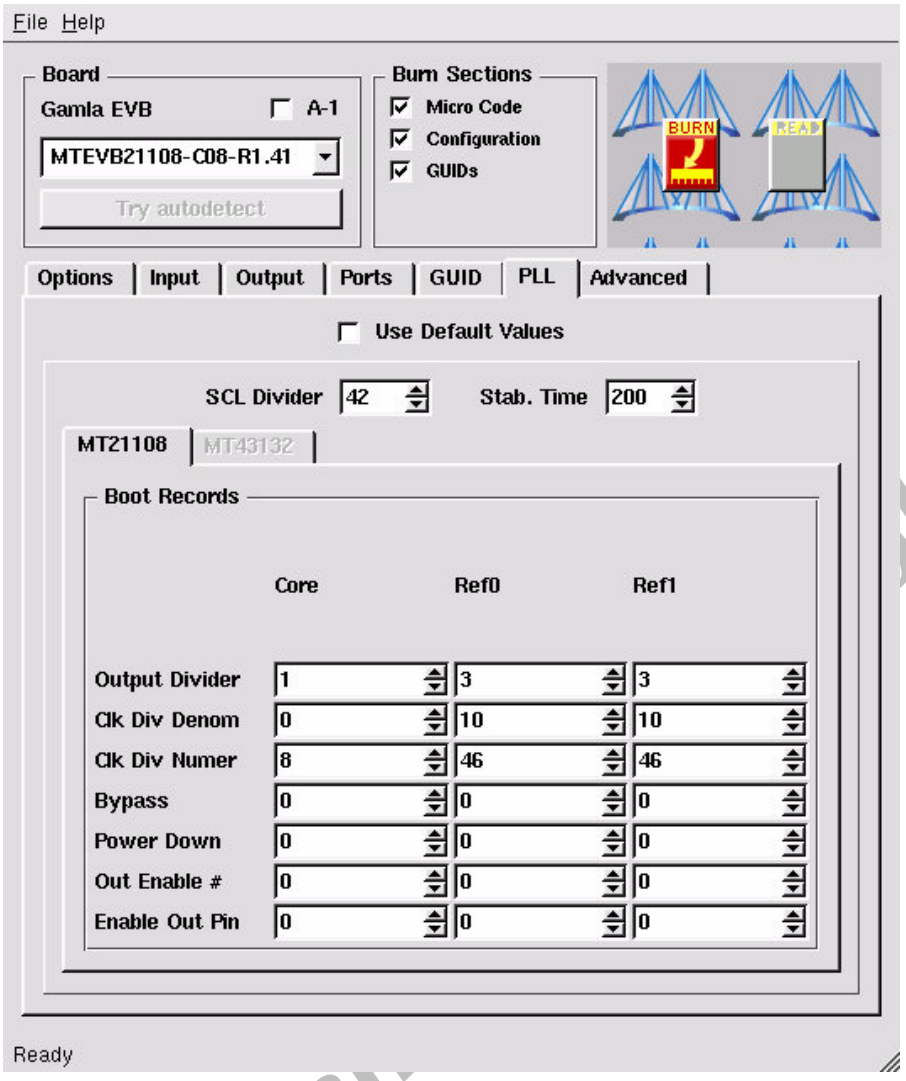
  Note: This tab is not relevant for devices operating in the P2P mode.

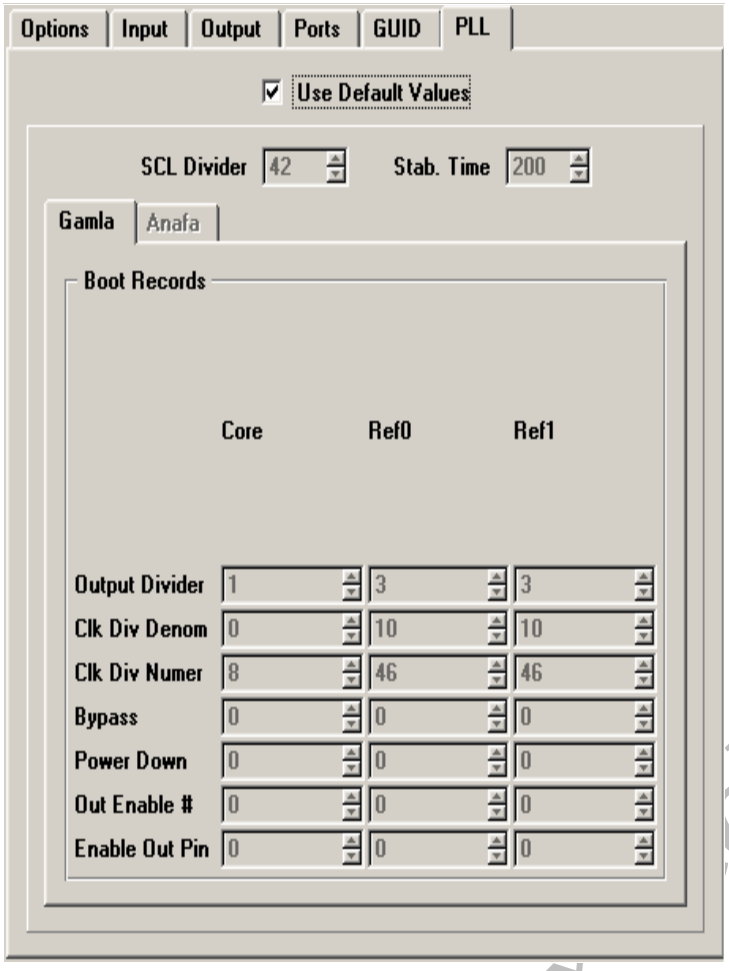
- **Use Board S.N** – Selecting this check box enables the Board S.N. text box, where you enter the serial number of your Mellanox-supplied board.
- **GUID Base** – The 64-bit Global Unique Identifier assigned to the device.

  Note: This tab is not relevant for devices operating in the P2P mode.

11.8.1 Setting the Internal Clocks of the Device

The following PLL tab enables you to change and tune the internal frequencies of the device.





When you have selected a board with the InfiniBridge (MT21108) device, the Gamla tab is enabled.

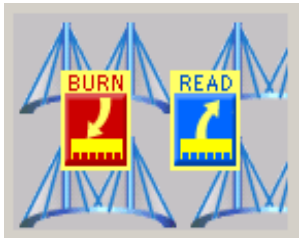
When you have selected a board with the InfiniScale (MT4312) device, the Anafa tab is enabled.

11.9 Burning the Firmware to the EEPROMs

After you have finished all the configuration tasks for the target device and the Firmware to be burned, you can use the EMT interface to burn the Firmware and configuration to the selected board's EEPROMs.

To burn the firmware to the EEPROMs:

- Click the following **BURN** button:



EMT displays a Progress dialog box showing the progress of the burning operation.

The READ button reads the some (not all) of the current configuration from the board.

11.10 Running the EMT Command Line

The EMT command line enables you to burn a previously prepared image. The creation of this image is enabled by the Output tab, and is described in Section 11.6.1, "Saving the Firmware to File," on page 64.

To display EMT command line syntax:

- Start EMT with the "-help" parameter. The following short help message is displayed:

Usage:

/emt [***EEPROM_image_file***] –bm <DEV> [***options***...]**Switches**:

-bsn <BSN> – Force board serial number to BSN.
-bm <DEV> – Burn using dev.
-guid <GUID> – Force GUID base to GUID.
-bs <I2C_slave> – I2C address of the first EEPROM; 0x55 by default.
-es <SIZE> – Size of each EEPROM in bytes; 0x8000 by default.
-ne <N> – Number of EEPROMS; 2 by default.

The parameters in this syntax are as follows:

EEPROM_image_file – Filename of the EEPROM image previously prepared by the EMT utility (see Section 11.6.1, "Saving the Firmware to File," on page 64).

-bsn – Overwrites the Board Serial Number, even if it is different from that in the EEPROM image file.

-guid – Overwrites the Base GUID, even if it is different from that in the EEPROM image file.

-bs – Sets the EEPROEM I^2C slave address.

-es – Sets the size of each EEPROM.

-ne – Sets the number of EEPROMs.

Example

```
emt my_image_file -guid 16H
```

11.11 JNI MT21108 Board: Default Configuration

EMT supports the JNI MT21108 board.

The following describes the default JNI board configuration programmed by EMT:

board name = "JNI-HCA-IBP-1x02"
nick name = "IBP-1x02"
eeprom
I2C slave address = 0x51
EEPROM size = 32KB

Number of EEPROMs = 2
Port width = 0 0 (1x)
Swizling = 0 0 0 0 (no)
Port off = 1 1 1 1 1 0 0 1
TX port reordering = 1 1 1 1 1 1 1 1
RX port reordering = 1 1 1 1 1 0 1 0
Board ID = 8
CA Ucode files = start1_ca.hx, start2_ca.hx, main_ca.hx
CA Ucode data files = main.cft, main_mod.cft, main.cin

12 adevmon: Device Monitoring Utility

12.1 Overview

Like the Infinivision tool, adevmon provides compact and convenient viewing of the modules, registers and sub-fields accessible from the CR bus. You can use the monitoring utilities to directly edit CR values, and you can easily customize the utility for your individual projects through an external configuration file.

For information about this device monitoring utility, see the "infinivision" chapter in Part I of this manual. The screens and operations of infinivision and adevmon are identical, with the following exception:

adevmon is started by running: `% adevmon`

13 eburn Utility

eburn is a command line utility that burns firmware to the EEPROM of an InfiniScale device.

13.1 Installation and Setup

eburn is part of the MST toolkit, whose requirements and installation procedure are described in Tools Installation (page 14). The installation copies the eburn utility to /usr/bin.

13.2 Operation

The MST driver must be started prior to running eburn tool. **To start eburn:**

- Start the MST driver (mst start or mst restart).
- Run eburn with the following command line syntax:

```
eburn <firmware_image_file> [options...]
```

where:

firmware_image_file – the firmware image that is burned to the EEPROM of an InfiniScale device. This image can be prepared via the EMT utility (see "EMT Tool" on page 59).

Note: You should not save the firmware image in binary format.

options can be:

-h – Print this message

-s – Silent mode – do not print progress.

-v – Verify only, do not burn.

-bsn <*BSN*> – Force board serial number to *BSN*.

-bm <*DEV*> – Burn using dev

-guid <*GUID*> – Force GUID base to GUID. May have special value "keep" (see below).

Note that if "-bsn" switch specified, the GUIDs will be calculated and burn according to this BSN by default. If you do not wish to burn GUIDs, specify "-guid keep". If neither "-bsn" nor "-guid" are specified, GUID won't be burn. If both "-bsn" and "-guid" are specified and "-guid" value isn't "keep", both BSN and GUID will be burned according to specified values.

14 gz_burn Utility

14.1 Overview

This release of gz_burn includes the following main features:

- Burns all/selected InfiniScale devices in these switch platforms: MTEK43132-M96-2P (Gazelle) and MTEK 14332-M16-5 (Gnu).
- Burns blade info eeprom.
- Gets the status of FW in all/selected InfiniScale devices.

gz_burn can work after it has been connected to the MTEK43132-M96-2P (Gazelle) network I2C bus or directly to the MTEK43132-MSX12-4x/MTEK4312-SP8 (Leaf/Spine). Selection of the blade and/or InfiniScale device enables working with a subset of blades (L*, S*, L[1-3]) and a subset of InfiniScale (L*A1, S2A[12]). gz_burn can be used for MTEK 43132-M16-5 (Gnu) switch platforms.

14.2 Installation and Setup

The gz_burn utility is part of the MST toolkit, whose requirements and installation procedure are described in Tools Installation (page 14). The MST installation copies the gz_burn utility to /usr/bin.

14.3 Operation

The MST driver must be started prior to running gz_burn tool. **To start gz_burn:**

- Start the MST driver (mst start or mst restart).
- Run gz_burn with the following command line syntax:

  `gz_burn` [*`OPTIONS`*]

where [OPTIONS] can be:

`-hw` – Selects HW device type. Supported devices are: *gazelle_i2c, gnu_i2c*. The default device is *gazelle_i2c*.

`-d|--dev` – Selects MST device to access HW boards. The default mst device is */dev/mst/mtusb*.

`--emt` – Runs the emt utility. The default is "emt".

`-h|--help|--man` – Displays a help message, in the form of an extended manual page.

Bus/Device Selection

-p – Specifies bus selection pattern.

Operations

-b|--burn – Performs burn operation.

-s| --stat – Prints a status report.

--bi – Burns blade info eeprom on chassis bus.

Additional Parameters

-i|--image – Specifies the image to burn. For MTEK43132-M96-2P (Gazelle) this must be a filename. For MTEK43132 -M16-S (Gnu) this must be the name of the directory containing images.

--bsn – Specifies BSN base. By default, bsn is not programmed.

---guid – Specifies GUID base for the blade. By default, GUID is not programmed.

--bv – Specifies blade version information.

--leaf|--spine – Forces gz_burn to work over leaf/spine only.

For information about additional options, run `gz_burn --man`.

OTHER_OPTIONS

-v|--ver – Prints version and exits.

-c – Prints configuration report.

-r|retry – Specifies number of retries for burning and other operations.

--no-color – Do not use color in output.

Description

gz_burn allows you to retrieve the status of, or burn specific devices on, the MTEK43132-M96-2P (Gazelle) I2c Management Bus. In general, command line options specified to gz_burn must define desired operation(s), bus(es) on which this operation should be performed and additional parameters for operation.

The following operations can be requested. They are listed in the same order that gz_burn performs them if more then one operation is requested.

`burn` – -b|--burn

`get status -s|--stat` – This is the default operation.

gz_burn can be run on a subset of MTEK43132-M96-2P (Gazelle) I2c buses. Currently supported I2c buses are:

S[1..4]A[1..3] InfiniScale on MTEK43132-SP8 (Spines) (Example S1A2)

S[1..4]CH Chassis Bus on MTEK43132-SP8 (Spines) (Example S2CH)

L[1..8]A[1..7] InfiniScale on MTEK43132-MSX12-4x (leafs) (Example L2A4)

L[1..8]CH Chassis Bus on MTEK43132-MSX12-4x (leafs) (Example L7CH)

To select bus(es) on which you want to perform an operation, you can specify perl/grep like pattern by -p option. By default, the pattern is '*', which means all buses. S* is all MTEK43132-SP8 (spines), L* is all MTEK43132-MSX12-4x (leafs), L*CH is chassis bus for all MTEK43132-MSX12-4x (leafs), e.t.c. Additional information is given in the following EXAMPLES section.

Examples

`gz_burn -b -i=gazelle.eeprom' – Burns all InfiniScale in MTEK43132-MSX12-4x (Gazelle) with an image from the file gazelle.eeprom.

`gz_burn -s' – Gets the full MTEK43132-M96-2P (Gazelle) status.

`gz_burn -b -p *CH -s' – Gets the status of the entire chassis bus.

`gz_burn -b -p L2A* -i=gazelle.eeprom' - Burns InfiniScale devices on MTEK43132-MSX12-4x 2 (Leaf 2).

gz_burn -d /dev/mst/calibre -s – This might print a status report similar to the following:

```
Connected to bus Gazelle I2C Primary bus
Disconnecting HUB Seg3,4 to prevent collisions
Present Boards: L1(v1) L2(v1) L3(v1) L4(v1) L8(v1) S1(v1) S2(v1)
Gazelle I2C Primary bus Status Report
-------------------------------------
L8A1  FW 05.00.0000-071 GUID=0002c901080482c0 BSN=MT31320B2403030131
L8A2  FW 05.00.0000-071 GUID=0002c901080482c8 BSN=MT31320B2403030132
L8A3  FW 05.00.0000-071 GUID=0002c901080482d0 BSN=MT31320B2403030133
L8A4  FW 05.00.0000-071 GUID=0002c901080482d8 BSN=MT31320B2403030134
L8A5  FW 05.00.0000-071 GUID=0002c901080482e0 BSN=MT31320B2403030135
L8A6  FW 05.00.0000-071 GUID=0002c901080482e8 BSN=MT31320B2403030136
L8A7  FW 05.00.0000-071 GUID=0002c901080482f0 BSN=MT31320B2403030137
L8CH  Blade EEP: ver=1 BSN=MT31320B240303013

L4A1  FW 05.00.0000-071 GUID=0002c90108048a30 BSN=MT31320B022403030301
L4A2  FW 05.00.0000-071 GUID=0002c90108048a38 BSN=MT31320B022403030302
L4A3  FW 05.00.0000-071 GUID=0002c90108048a40 BSN=MT31320B022403030303
L4A4  FW 05.00.0000-071 GUID=0002c90108048a48 BSN=MT31320B022403030304
L4A5  FW 05.00.0000-071 GUID=0002c90108048a50 BSN=MT31320B022403030305
L4A6  FW 05.00.0000-071 GUID=0002c90108048a58 BSN=MT31320B022403030306
L4A7  FW 05.00.0000-071 GUID=0002c90108048a60 BSN=MT31320B022403030307
L4CH  Blade EEP: ver=1 BSN=MT31320B02240303030

L3A1  FW 05.00.0000-071 GUID=0002c90108048480 BSN=MT31320B2403030171
L3A2  FW 05.00.0000-071 GUID=0002c90108048488 BSN=MT31320B2403030172
L3A3  FW 05.00.0000-071 GUID=0002c90108048490 BSN=MT31320B2403030173
L3A4  FW 05.00.0000-071 GUID=0002c90108048498 BSN=MT31320B2403030174
L3A5  FW 05.00.0000-071 GUID=0002c901080484a0 BSN=MT31320B2403030175
L3A6  FW 05.00.0000-071 GUID=0002c901080484a8 BSN=MT31320B2403030176
L3A7  FW 05.00.0000-071 GUID=0002c901080484b0 BSN=MT31320B2403030177
L3CH  Blade EEP: ver=1 BSN=MT31320B240303017

L2A1  FW 05.00.0000-071 GUID=0002c90107fdb270 BSN=MT31220B2003030261
L2A2  FW 05.00.0000-071 GUID=0002c90107fdb278 BSN=MT31220B2003030262
L2A3  FW 05.00.0000-071 GUID=0002c90107fdb280 BSN=MT31220B2003030263
L2A4  FW 05.00.0000-071 GUID=0002c90107fdb288 BSN=MT31220B2003030264
L2A5  FW 05.00.0000-071 GUID=0002c90107fdb290 BSN=MT31220B2003030265
L2A6  FW 05.00.0000-071 GUID=0002c90107fdb298 BSN=MT31220B2003030266
L2A7  FW 05.00.0000-071 GUID=0002c90107fdb2a0 BSN=MT31220B2003030267
```

```
L2CH  Blade EEP: ver=1 BSN=MT31220B200303026

L1A1  FW 05.00.0000-071 GUID=0002c90107f88a10 BSN=MT31320B1703030081
L1A2  FW 05.00.0000-071 GUID=0002c90107f88a18 BSN=MT31320B1703030082
L1A3  FW 05.00.0000-071 GUID=0002c90107f88a20 BSN=MT31320B1703030083
L1A4  FW 05.00.0000-071 GUID=0002c90107f88a28 BSN=MT31320B1703030084
L1A5  FW 05.00.0000-071 GUID=0002c90107f88a30 BSN=MT31320B1703030085
L1A6  FW 05.00.0000-071 GUID=0002c90107f88a38 BSN=MT31320B1703030086
L1A7  FW 05.00.0000-071 GUID=0002c90107f88a40 BSN=MT31320B1703030087
L1CH  Blade EEP: ver=1 BSN=MT31320A240203005

S2A1  FW 05.00.0000-071 GUID=0002c90107fdafd0 BSN=MT31220B2003030201
S2A2  FW 05.00.0000-071 GUID=0002c90107fdafd8 BSN=MT31220B2003030202
S2A3  FW 05.00.0000-071 GUID=0002c90107fdafe0 BSN=MT31220B2003030203
S2CH  Blade EEP: ver=1 BSN=MT31220B20030302

S1A1  FW 05.00.0000-071 GUID=0002c90107fdab00 BSN=MT31220B2003030091
S1A2  FW 05.00.0000-071 GUID=0002c90107fdab08 BSN=MT31220B2003030092
S1A3  FW 05.00.0000-071 GUID=0002c90107fdab10 BSN=MT31220B2003030093
S1CH  Blade EEP: ver=1 BSN=MT31220B200303009

PROBLEM SUMMARY ====================================================
ERROR: L1A3: bad BSN, doesn't match L1CH
ERROR: L1A4: bad BSN, doesn't match L1CH
ERROR: L1A5: bad BSN, doesn't match L1CH
ERROR: L1A6: bad BSN, doesn't match L1CH
ERROR: L1A7: bad BSN, doesn't match L1CH
ERROR: L1A1: bad BSN, doesn't match L1CH
ERROR: L1A2: bad BSN, doesn't match L1CH
```

Part IV: USB to I^2C Adapter

This part includes the User's Manual for MTUSB-1, Mellanox's USB to I^2C Adapter.

15 MTUSB-1 USB to I2C Adapter

15.1 Overview

The MTUSB-1 is a USB to I^2C-bus adapter. This chapter provides the user with hardware and software installation instructions. It also describes a functionality test which can be run after adapter HW and SW installation to verify that communication over the USB and I^2C busses has been established correctly.

15.1.1 Scope

It is highly recommended that this chapter is carefully read in its entirety before any attempt to install and/or use the hardware and software contents of this package.

15.1.2 Revision History

Table 3 - MTUSB-1 Revision History

Revision	Change	Date
0.0	Initial version	25-Mar-2004

15.1.3 Package Contents

Please make sure that your package contains the following items and that they are in good condition.

- One page with instructions pointing to this User's Manual
- One MTUSB-1 device
- One USB Cable: USB_A to USB_B 1.8m
- One I2C Cable: 9pin male/male 1.8m
- One Converter Cable: 9pin female to 3pin 0.3m

15.1.4 System Requirements

The MTUSB-1 is a USB device which may be connected to any Personal Computer with a USB Host Adapter (USB Standard 1.1) and having the following specifications:

- Linux operating system (See "Supported Platforms" on page 85. below).
- At least one USB connection port (USB-Standard 1.1).
- MST Tool driver set (supplied by Mellanox).

15.1.5 Supported Platforms

MTUSB-1 supported platforms are the same as those of the MST tools package. See "Supported Platforms and Operating Systems" on page 13.

15.2 Hardware Installation

To install the MTUSB-1 hardware, please follow these steps:

1. Connect one end of the I2C cable to the MTUSB-1 and the other end to the system/board you wish to control via the I2C interface. If the system/board uses a 3-pin connector instead of a 9-pin connector, connect the provided converter cable as an extension to the I2C cable, then connect the converter cable to the system/board.
2. Connect one end of the USB cable to the MTUSB-1 and the other end to the PC.

15.3 Software Installation

The MTUSB-1 device requires that the Mellanox Software Tools (MST) package be installed on the Linux PC to which MTUSB-1 is connected. (See Chapter 1 of *MST User's Manual* for installation instructions.) Once you have MST installed, you may verify that your MTUSB-1 device is detected by MST software.

To detect your MTUSB-1 device, follow these steps:

1. Start the MST drivers by entering:

```
mst start          (or mst restart if mst start was run earlier)
```

2. To obtain the list of MST devices enter:

```
mst status
```

If MTUSB-1 has been correctly installed, "***mst status***" should include the following device in the device list it generates:

/dev/mst/mtusb-1

For further details related to MST operations, please refer to *MST User's Manual*.

15.4 MTUSB-1 Functionality Test

After you have verified that the MTUSB-1 device has been detected by the MST driver, you can run a special test called '*mtusb1_test*' which is normally installed under /usr/mst/bin (or wherever the mst/bin subdirectory resides). This test will check the functionality of the MTUSB-1 device (and its environment) on various levels such as: electrical (no device short circuits), firmware version, communication with the driver on USB bus, communication with target on I^2C bus, and others.

The test performs the following steps:

1. Checks Firmware version of the device for correctness and proper burning. This includes testing the USB connection of driver to MTUSB-1 device for correctness.
2. Requests the user to disconnect from the I^2C target in order to verify that MTUSB-1 is free of internal short circuits.
3. Requests the user to reconnect the target to MTUSB-1 in order to ensure communication over the I^2C bus is correct. The test will drive the I^2C bus signals SDA and SCL in a sequence aimed to verify that no shorts hinder their operation.
4. Finally, it scans for all slave devices connected to the I^2C bus and lists their addresses. This will enable the user to verify that all expected I^2C devices exist and respond (assuming the user has the list of I^2C slave devices and their addresses).

Following is an example of an *mtusb1_test* run:

```
> /usr/mst/bin/mtusb1_test
Check FW Version : PASSED
Please disconnect I2C cable from target device. Press Enter...
SDA SCL Lines Idle status test : PASSED
SDA SCL HI/LOW transition test : PASSED
Please Connect I2C cable to target device. Press Enter...
SDA SCL Lines Target test : PASSED
SDA SCL HI/LOW Target transition test: PASSED
Send I2C STOP to Target : PASSED
I2C Slave devices Scan. Found devices: 0x48 0x51 0x52 0x53 0x54 0x55 x6c 0x77
Test PASSED
```

Appendix A: InfiniScale III MT47396 Firmware Initialization File (.INI)

The Mellanox Technologies firmware burning tool **is3burn** requires a firmware (FW) file to get initialization and configuration information. (Please see is3burn documentation.) This file is either supplied by Mellanox, or by the user. The command line option "-fw" of is3burn uses a *Mellanox-supplied* FW (image) file; the "-conf" option uses a *user-supplied* FW configuration (**.INI**) file. (For information on is3burn and how to call the .INI file, see "is3burn EEPROM Management Tool" on page 41.)

Even if the user does not have a FW configuration file ready, it is possible to have is3burn create one. However, the user still needs some pre-existing FW file for is3burn to use in this process. Once this configuration file is created, its initialization and configuration information can be modified to suit the user's specific system.

To begin with, the .INI file is a text file composed of several initialization and configuration *sections*. The user may choose to include all sections and all attribute settings in the final .INI file, and modify some of the attributes as required. Alternatively, the user may choose to keep only the sections with changes to the existing settings, with only those attributes that are to be modified.

This appendix includes the following sections:

> Note: For the full details of each .INI section, please refer to the *InfiniScale III MT47396 Firmware Release Notes*.

A.1 .INI File Format

The .INI file is actually a concatenation of (a part or all) section specific initialization and configuration settings. Each section in the .INI file starts with its name between square brackets, e.g. [EEPROM], [General], etc. The section name is followed by one or more lines of configuration settings and comments, as in the partial .INI file shown below. Note that comment lines start with a semicolon.

Example:

```
[EEPROM]
; This is a comment line
amount = 0x4
eeprom1_address = 0x56
eeprom1_size = 64

[General]
PortsBufferingMode = SAF
DEVID = 0xb924
VID = 0x2c9
AutoPowerSave = Enable
; End of (partial) .INI file
```

A.2 List of .INI File Sections

The .INI file sections are:

- EEPROM
- PSID
- General
- Special
- ENP0
- MISC
- IB_TO_HW_MAP
- IB_TO_LED_MAP
- PortDisable
- LinkWidthSupp
- PLL
- REV_LANE
- Polarity
- SERDES_Equa_CFG
- SERDES_OutPut_Voltage
- SERDES_Pre_Amp_OutPut
- SERDES_Pre_Emp_Out
- SERDES_Pre_Emp_Pre_Amp
- Credits_Time

Note: For the full details of each .INI section, please refer to the *InfiniScale III MT47396 Firmware Release Notes*.