



# **Intel® Dialogic® System Software for DM3 Architecture Products on Windows**

**Diagnostics Guide**

---

*November 2003*



INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. Intel products are not intended for use in medical, life saving, or life sustaining applications.

Intel may make changes to specifications and product descriptions at any time, without notice.

This Intel® Dialogic® System Software for DM3 Architecture Products on Windows Diagnostics Guide as well as the software described in it is furnished under license and may only be used or copied in accordance with the terms of the license. The information in this manual is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Intel Corporation. Intel Corporation assumes no responsibility or liability for any errors or inaccuracies that may appear in this document or any software that may be provided in association with this document.

Except as permitted by such license, no part of this document may be reproduced, stored in a retrieval system, or transmitted in any form or by any means without express written consent of Intel Corporation.

Copyright © 2002-2003, Intel Corporation

AnyPoint, BoardWatch, BunnyPeople, CablePort, Celeron, Chips, CT Media, Dialogic, DM3, EtherExpress, ETOX, FlashFile, i386, i486, i960, iCOMP, InstantIP, Intel, Intel Centrino, Intel Centrino logo, Intel logo, Intel386, Intel486, Intel740, IntelDX2, IntelDX4, IntelSX2, Intel InBusiness, Intel Inside, Intel Inside logo, Intel NetBurst, Intel NetMerge, Intel NetStructure, Intel SingleDriver, Intel SpeedStep, Intel StrataFlash, Intel TeamStation, Intel Xeon, Intel XScale, IPLink, Itanium, MCS, MMX, MMX logo, Optimizer logo, OverDrive, Paragon, PDCharm, Pentium, Pentium II Xeon, Pentium III Xeon, Performance at Your Command, RemoteExpress, SmartDie, Solutions960, Sound Mark, StorageExpress, The Computer Inside., The Journey Inside, TokenExpress, VoiceBrick, VTune, and Xircom are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

\* Other names and brands may be claimed as the property of others.

Publication Date: November 2003

Document Number: 05-1935-002

Intel Converged Communications, Inc.  
1515 Route 10  
Parsippany, NJ 07054

For **Technical Support**, visit the Intel Telecom Support Resources website at:

<http://developer.intel.com/design/telecom/support>

For **Products and Services Information**, visit the Intel Telecom Products website at:

<http://www.intel.com/design/network/products/telecom>

For **Sales Offices** and other contact information, visit the Where to Buy Intel Telecom Products page at:

<http://www.intel.com/buy/wtb/wtb1028.htm>



# Contents

---

	<b>Revision History</b> .....	4
	<b>About This Publication</b> .....	5
<b>1</b>	<b>Diagnostics Overview</b> .....	9
1.1	Common Diagnostic Tasks .....	9
1.2	Diagnostic Tools .....	9
<b>2</b>	<b>System Requirements</b> .....	11
2.1	Hardware Requirements .....	11
2.2	Software Requirements .....	11
<b>3</b>	<b>CallInfo Reference</b> .....	13
<b>4</b>	<b>CAS Trace Reference</b> .....	15
<b>5</b>	<b>DebugAngel Reference</b> .....	17
<b>6</b>	<b>DebugView Reference</b> .....	19
<b>7</b>	<b>DigitDetector Reference</b> .....	21
<b>8</b>	<b>Dlgsnapshot Reference</b> .....	23
<b>9</b>	<b>DM3Insight Reference</b> .....	25
<b>10</b>	<b>DM3post Reference</b> .....	27
<b>11</b>	<b>Getver Reference</b> .....	29
<b>12</b>	<b>ISDN Trace Reference</b> .....	31
<b>13</b>	<b>KernelVer Reference</b> .....	33
<b>14</b>	<b>MercMon Reference</b> .....	35
<b>15</b>	<b>PDK Trace Reference</b> .....	41
<b>16</b>	<b>Phone Reference</b> .....	43
<b>17</b>	<b>QError Reference</b> .....	45
<b>18</b>	<b>Status Monitor Reference</b> .....	47
<b>19</b>	<b>StrmStat Reference</b> .....	49
<b>20</b>	<b>TSP Config Reference</b> .....	51
<b>21</b>	<b>TSP Monitor Reference</b> .....	53
<b>22</b>	<b>TSP Tracer Reference</b> .....	55
	<b>Index</b> .....	57



## Revision History

---

This revision history summarizes the changes made in each published version of this document.

Document No.	Publication Date	Description of Revisions
05-1935-002	November 2003	Global changes: Removed the chapters about DM3Stderr, DM3Trace, and DM3Kdebug and all references to these tools. <a href="#">Diagnostics Overview</a> : Added references to new diagnostic tools. <a href="#">How to Use This Publication</a> : Updated chapter list to reflect additions and deletions. <a href="#">CAS Trace Reference</a> : New chapter <a href="#">DebugAngel Reference</a> : New chapter <a href="#">Dlgsnapshot Reference</a> : New chapter <a href="#">PDK Trace Reference</a> : New chapter <a href="#">Status Monitor Reference</a> : New chapter <a href="#">TSP Config Reference</a> : New chapter (previously in Administration Guide) <a href="#">TSP Monitor Reference</a> : New chapter (previously in Administration Guide) <a href="#">TSP Tracer Reference</a> : New chapter (previously in Administration Guide)
05-1935-001	November 2002	Initial version of document. Much of the information contained in this document was previously contained in the <i>DM3 Diagnostic Utilities Reference Guide</i> , document number 05-1484-005.



## About This Publication

---

The following topics provide information about this *Intel® Dialogic® System Software for DM3 Architecture Products on Windows Diagnostics Guide*:

- [Purpose](#)
- [Intended Audience](#)
- [How to Use This Publication](#)
- [Related Information](#)

### Purpose

This guide describes the diagnostic tools included with the Intel® Dialogic® System Software release and explains how to use them.

**Note:** All tools described in this publication apply to Intel® NetStructure™ on DM3 architecture boards only. For information about diagnostic tools that can be used with Intel Dialogic Springware boards, refer to the *Dialogic Universal Hardware Diagnostics Guide*.

### Intended Audience

This information is intended for:

- Distributors
- System Integrators
- Toolkit Developers
- Independent Software Vendors (ISVs)
- Value Added Resellers (VARs)
- Original Equipment Manufacturers (OEMs)
- End Users

### How to Use This Publication

Refer to this document after you have installed the hardware and the Intel Dialogic system software that includes the diagnostic tools.

The information in this guide is organized as follows:

- [Chapter 1, “Diagnostics Overview”](#) provides a brief overview of all diagnostic tools.

- [Chapter 2, “System Requirements”](#) lists the hardware and software required to run the diagnostic tools.

Each of the remaining chapters provides information about an individual diagnostic tool. The chapters are organized in alphabetical order:

- [Chapter 3, “CallInfo Reference”](#) - The CallInfo tool detects call information using the DM3 board’s Telephony Service Provider (TSP) resource.
- [Chapter 4, “CAS Trace Reference”](#) - The CAS Trace tool enables you to track the bit level transitions on a robbed bit or CAS line.
- [Chapter 5, “DebugAngel Reference”](#) - The DebugAngel tool provides low level firmware tracing to aid in low level debugging.
- [Chapter 6, “DebugView Reference”](#) - DebugView is an application that lets you monitor debug output on your local system, or any computer on the network that you can reach via TCP/IP.
- [Chapter 7, “DigitDetector Reference”](#) - The DigitDetector tool provides the ability to detect digits at the local end of a channel connection.
- [Chapter 8, “Dlgsnapshot Reference”](#) - The dlgsnapshot tool uses Intel® Dialogic® system software fault monitoring components to generate a core dump file when a Control Processor (CP), Signal Processor (SP), or Shared RAM (SRAM) fault is detected on a DM3 board.
- [Chapter 9, “DM3Insight Reference”](#) - DM3Insight is a tool used to capture message and stream traffic from the DM3 board device driver.
- [Chapter 10, “DM3post Reference”](#) - The Dm3post tool can perform diagnostics on a stopped board at any time to detect and isolate possible hardware faults.
- [Chapter 11, “Getver Reference”](#) - The Getver tool displays the version of any DM3 board binary file.
- [Chapter 12, “ISDN Trace Reference”](#) - The ISDNtrace tool provides the ability to track Layer 3 (Q.931) messages on the ISDN D-channel. ISDNtrace prints messages on the screen in real time.
- [Chapter 13, “KernelVer Reference”](#) - TheKernelVer tool can be used to verify whether or not a processor has crashed.
- [Chapter 14, “MercMon Reference”](#) - MercMon provides counter information about DM3 board device drivers (Class Driver and Protocol Drivers).
- [Chapter 15, “PDK Trace Reference”](#) - The PDK Trace tool allows those who use a DM3 PDK protocol to log specific information related to the operation of the protocol.
- [Chapter 16, “Phone Reference”](#) - The Phone tool can control a single DM3 resource channel (make calls, wait for calls etc.), monitor channel and call states, and send call control operations to a DM3 Global Call resource.
- [Chapter 17, “QError Reference”](#) - The QError tool displays the string associated with a particular error code returned in a QResultError message from the board.
- [Chapter 18, “Status Monitor Reference”](#) - The Status Monitor tool enables you to track the state of the TSC as well as the state of the bits on a robbed bit or CAS line.
- [Chapter 19, “StrmStat Reference”](#) - The StrmStat tool displays the current state of a set of specified streams.

- [Chapter 20, “TSP Config Reference”](#) - The TSP Config tool sets and retrieves DM3 protocol variant parameters.
- [Chapter 21, “TSP Monitor Reference”](#) - TSP Monitor monitors one or two DM3 Global Call resource channels.
- [Chapter 22, “TSP Tracer Reference”](#) - TSP Tracer traces CAS protocols with timing information and saves trace information to a log file.

## Related Information

Refer to the following documents and websites for more information:

- The appropriate Administration Guide:
  - *Intel Dialogic System Software for PCI Products on Windows Administration Guide*
  - *Intel Dialogic System Release CompactPCI for Windows Administration Guide*
- *SNMP Agent Software for Windows Operating Systems Administration Guide*
- The appropriate Configuration Guide(s):
  - *Intel DM3 Architecture PCI Products on Windows Configuration Guide*
  - *Intel NetStructure Products on DM3 Architecture for CompactPCI on Windows Configuration Guide*
  - *Intel NetStructure IPT Series on Windows Configuration Guide*
- The Release Guide for your System Release
- The online System Release Update
- <http://developer.intel.com/design/telecom/support/> (for technical support)
- <http://www.intel.com/design/network/products/telecom/index.htm> (for product information)





This chapter presents an overview of the diagnostic tools and the debugging/troubleshooting tasks that can be performed with them. The following sections are included:

- [Common Diagnostic Tasks](#) ..... 9
- [Diagnostic Tools](#) ..... 9

## 1.1 Common Diagnostic Tasks

This section provides a list of diagnostic tasks that can be performed by using the various diagnostics tools.

The following tasks can be accomplished using the diagnostic tools:

Checking the hardware

You can use the Dm3post, Getver, KernelVer, and Dlgsnapshot tools to troubleshoot the physical boards in your system.

Checking the network connections

You can use the Phone, Digit Detector, CallInfo and DebugView\* tools to check your board's network connections.

Checking the firmware

You can use the DebugAngel, PDK Trace, and MercMon tools to diagnose and trace problems with a board's firmware.

## 1.2 Diagnostic Tools

This section provides a brief description of the categories of diagnostic tools included with the system software. Topics include:

- [QScript Tools](#)
- [Other Tools](#)

### 1.2.1 QScript Tools

QScript is an object-oriented scripting tool developed for the Intel® NetStructure™ on DM3™ architecture products. The QScript tools are a subset of the diagnostic tools that talk to the various components of a DM3 board (player, recorder etc.). QScript is intended for use while developing demonstration or test programs and is implemented using the Tcl/Tk generic scripting language.

The default location of the QScript tools is *C:\program files\dialogic\qscript* while the Windows batch file used to invoke the QScript tools is *C:\program files\dialogic\bin*.

**Note:** Do not directly run any *<toolname>.qs* files located in *C:\program files\dialogic\qscript*. Script files have been created which call the QScript interpreter to run the *<toolname>.qs* file. To use a QScript tool, specify the tool name, along with any options, at the command line as shown in the various diagnostic tool description sections in this document.

The following diagnostic tools use QScript:

- CallInfo
- CAS Trace
- DigitDetector
- PDK Trace
- Phone
- TSP Config
- TSP Monitor
- TSP Tracer

## 1.2.2 Other Tools

The following diagnostic tools are included with the system release software but do not use QScript:

- DebugAngel
- DebugView\*
- Dlgsnapshot
- Dm3post
- Dm3Insight
- GetVer
- ISDNTrace
- KernelVer
- MercMon
- PDK Trace
- QError
- Status Monitor
- StrmStat

The default location for these files is *C:\program files\dialogic\bin*.

This chapter provides information about the hardware/software requirements needed to run the diagnostic tools. Topics include:

- [Hardware Requirements . . . . . 11](#)
- [Software Requirements . . . . . 11](#)

## 2.1 Hardware Requirements

At least one Intel® NetStructure™ on DM3™ architecture board is required to run the diagnostic tools.

## 2.2 Software Requirements

The Intel® Dialogic® system software is required to use all diagnostic tools.



This chapter provides reference information about the CallInfo tool.

The CallInfo tool detects call information using the DM3 board's Telephony Service Provider (TSP) resource. The CallInfo tool confirms that the TSP component is working correctly by monitoring select messages on a per call basis.

To use CallInfo, specify the call-related messages you want to monitor as follows:

1. Launch the CallInfo tool from the command line.
2. Click the **Action** menu on the CallInfo display. Highlight **Select Ids** and select the group of call-related events you want to trace. The following menu selections are available:
  - CallInfoSet: TSP-related call messages
  - IE Set: Information elements of ISDN-related messages (focuses on small pieces of information)
  - ISDNMsgSet: ISDN-related messages
3. A window of events specific to the event group that you selected will appear. Select the messages you want to trace by clicking on them.
4. After you select messages, click the **Action** menu and choose **DetectEvt**. The tool will start monitoring the messages you selected. As a new call comes in, it will write over the old call information in the CallInfo display. A separate console window will open that tracks messages coming in and shows the message sequence.
5. If you want to stop tracing events you selected and select other events to trace, click the **Action** menu and select **CancelEvt**. Then follow steps 2 through 4 again.
6. To exit the CallInfo tool completely, select **Action > Exit** or close the window.

## Options

The CallInfo tool uses the following command line options:

-board <n>

Logical ID of board (required). Use the PBL utility (Linux) or the DCM configuration utility (Windows) to obtain the board's logical ID.

-line <n>

Line number that the tool will monitor (optional). The default value is 1.

-chan<n>

Channel number that the tool will monitor (optional). The default value is 1.

The following example runs the CallInfo tool on board 0, line 1, channel 1:

```
callinfo -board 0 -line 1 -chan 1
```



This chapter provides reference information about the CAS Trace tool.

The CAS Trace tool enables you to track the bit level transitions on a robbed bit or CAS line. CAS Trace prints messages on the screen in real time. The CAS Trace tool can be configured to output the messages into a circular log file.

The CAS Trace tool is used for a class of problems similar to those you would troubleshoot using the TSP Monitor tool<sup>1</sup>, but on a larger scale: an entire span rather than just one channel. CAS Trace is used for long-term debugging and logging for problems with timing, hung channels, and improper bit states.

The CAS Trace tool consumes a portion of the CPU for each trunk that is logged and may generate an exception if CTRL-C is used to exit the tool.

It may take several seconds to start the monitoring on all the trunks

The CAS instance is not valid until the line is in service, so you will have to run an application or the Lineadmin and Phone tools to set the channel in-service before using this application. For information about the Lineadmin tool, refer to the Administration Guide for the system release. For information about the Phone tool, refer to [Chapter 16, “Phone Reference”](#).

## Options

The CAS Trace tool uses the following command line options:

-board <board list>

Logical ID of board (or boards) to trace (optional). The default is the lowest ID present in the system. Use the PBL utility (Linux) or the DCM configuration utility (Windows) to obtain the board's logical ID.

-line <list of lines>

Line number that the tool will monitor (optional). The default value is 1.

-k <file size>

Size of each file to create in kilobytes (optional). If this option is not present the file size will be infinite.

-# <number of files to create>

The number of files to trace to in a circular fashion. Each file will be the size specified with -k (optional). The default is a single file.

-f <filename>

This will be the base filename. The board, line, and file numbers will be appended to the end of this name (optional). The default is *CasTrace.log*.

---

1. For more information about TSP Monitor, refer to [Chapter 21, “TSP Monitor Reference”](#).

**-nodisplay**

This flag will turn off the screen output to help reduce CPU overhead (optional). The default is to have the screen display on. The display can also be toggled on and off by pressing **d** during runtime.

**-t1 / -e1**

This flag will tell the system if you are using a T1 protocol or and E1 protocol (optional). The default is T1. If you specify this incorrectly, you may not be able to initialize and monitor the upper channels.

The following will run the CAS Trace tool on boards 0 and 1 for all four spans. Each line will trace to two 1 MB files:

```
CAStrace -board 0 1 -line 1 2 3 4 -# 2 -k 1000
```

The output will look like the example below:

```
[      timestamp      ] B## L## T## Rx=ABCD Tx=ABCD +delta (mS)
-----
[10/29 23:33:33.218] B0 L1 T1 Rx=1100 Tx=0000
[10/29 23:33:33.828] B0 L1 T2 Rx=0000 Tx=0000
[10/29 23:33:33.828] B0 L1 T1 Rx=1100 Tx=1100 +610
[10/29 23:33:34.531] B0 L1 T2 Rx=1100 Tx=0000 +703
[10/29 23:33:34.593] B0 L1 T2 Rx=1100 Tx=1100 +62
[10/29 23:33:34.609] B0 L1 T2 Rx=1100 Tx=0000 +16
[10/29 23:33:34.718] B0 L1 T3 Rx=1100 Tx=0000
[10/29 23:33:35.468] B0 L1 T1 Rx=0000 Tx=1100 +1640
[10/29 23:33:35.500] B0 L1 T3 Rx=1100 Tx=1100 +782
[10/29 23:33:35.562] B0 L1 T1 Rx=0000 Tx=0000 +94
[10/29 23:33:35.656] B0 L1 T4 Rx=1100 Tx=0000
[10/29 23:33:35.656] B0 L1 T1 Rx=0000 Tx=0000 +94
[10/29 23:33:36.625] B0 L1 T5 Rx=1100 Tx=0000
[10/29 23:33:36.843] B0 L1 T1 Rx=1100 Tx=0000 +1187
[10/29 23:33:36.906] B0 L1 T1 Rx=1100 Tx=1100 +63
[10/29 23:33:36.921] B0 L1 T1 Rx=1100 Tx=0000 +15
[10/29 23:33:37.203] B0 L1 T3 Rx=0000 Tx=1100 +1703
[10/29 23:33:37.328] B0 L1 T6 Rx=1100 Tx=0000
[10/29 23:33:37.328] B0 L1 T3 Rx=0000 Tx=0000 +125
[10/29 23:33:37.421] B0 L1 T3 Rx=0000 Tx=0000 +93
[10/29 23:33:37.843] B0 L1 T4 Rx=1100 Tx=1100 +2187
[10/29 23:33:37.875] B0 L1 T2 Rx=1100 Tx=1100 +3266
[10/29 23:33:37.875] B0 L1 T5 Rx=1100 Tx=1100 +1250
```

The output is separated into the following columns:

- Timestamp - the time at which the event is received down to msec
- B## - the board's logical ID
- L## - the line number
- T## - the timeslot
- Rx=ABCD - the state of the Receive bits
- Tx=ABCD - the state of the Transmit bits
- Delta - the time between the last transition in msec (if this is blank, it is the initial bit state)



This chapter provides reference information about the DebugAngel tool.

The DebugAngel tool provides low level firmware tracing to aid in low level debugging. Running as a Windows service, it polls the DM3 boards in the system and posts **qPrintf()** statements from the resources and DM3 kernel to a log file.

Once the service is running, no further action is needed. Any changes to your system (for example, new boards) are automatically detected. Information is logged to the file *DebugAngel.log* in the *%INTEL\_DIALOGIC\_DIR%\log* directory.

## Command Line Options

To install the service and start it running in automatic mode, enter the command:

```
DebugAngel -install
```

The DebugAngel tool uses the following options when it is invoked from the command line.

**Note:** Command line options are mutually exclusive.

- instonly  
Installs the service without starting it.
- start  
Starts the service.
- stop  
Stops the service.
- manual  
Changes the service startup mode to manual.
- auto  
Changes the service startup mode to automatic (default).
- remove  
Removes the service (and stops it if it was started).
- status  
Shows the service status.

## Additional Configuration Options

Additional configuration options for DebugAngel are available under the Windows registry key *HKEY\_LOCAL\_MACHINE\SOFTWARE\Dialogic\DebugAngel*.

**Warning:** Incorrect manipulation of the Windows registry can render your system unusable, requiring that you reinstall Windows. Only a system administrator qualified to modify the registry should change the DebugAngel configuration.

The configuration options in the registry include:

DebugLevel -> 0

1 writes the log information to DebugView.

MaxFileSize -> 0

0 = no max. When the max size is reached, the file is truncated to 0.

LogFile -> default file name

AutoRename -> 1

1 (default) avoids erasing the file when the computer is restarted; the file is renamed (name).bak. Other options are: 0 = don't back up 2 = rename file to (name).(Day/Time\_String)  
3 = uses multiple files.

MaxFiles -> 1

Used by AutoRename 3 (default is 1 file).

This chapter describes the DebugView\* diagnostic tool.

DebugView is an application that lets you monitor debug output on your local system, or any computer on the network that you can reach via TCP/IP. It can display both kernel-mode and Win32 debug output generated by standard debug print APIs, so you don't need a debugger to catch the debug output your applications or device drivers generate, and you don't need to modify your applications or drivers to use non-Windows debug functions in order to view its debug output.

To download this freeware and get full instructions for using it, go to <http://www.sysinternals.com/ntw2k/freeware/debugview.shtml>.



This chapter provides reference information about the DigitDetector tool.

The DigitDetector tool demonstrates the use of a DM3 board's Tone Generator and Signal Detector components. It provides the ability to detect digits at the local end of a channel connection.

To use the DigitDetector tool, you need a physical connection to the board. For example, two network interface trunks can be looped or you might use a connection between end users. You can then use the Phone tool at one end of the channel to generate dialed digits that can be detected by the Digit Detector tool.

## Options

The DigitDetector tool uses the following command line options:

-board <n>

Logical ID of board (required). Use the PBL utility (Linux) or the DCM configuration utility (Windows) to obtain the board's logical ID.

-line <n>

Line number that the tool will monitor for digits (optional). The default value is 1.

-chan<n>

Channel number that the tool will monitor for digits (optional). The default value is 1.

The following example runs the DigitDetector tool on board 0, line 1, channel 1:

```
digitdetector -board 0 -line 1 -chan 1
```



This chapter provides reference information about the dlgsnapshot tool.

The dlgsnapshot tool uses Intel® Dialogic® system software fault monitoring components to generate a core dump file when a Control Processor (CP), Signal Processor (SP), or Shared RAM (SRAM) fault is detected on a DM3 board. When a fault is detected, a core dump file is created in the *Program Files\Dialogic\log* directory and the board that the fault occurred on is taken out of service (stopped) without causing any other interruptions to the system. To reactivate the out-of-service board, you must restart it.

When a core dump file is generated, you can send a .zip file containing the contents of the *Program Files\Dialogic\log* directory along with the Windows event viewer system log file (\*.evt) to Intel Dialogic support services for debugging purposes.

Each core dump file is named according to the type of fault detected and the date/time the fault occurred. The naming convention of the core dump files is

faulttype MM\_DD\_YY HH\_NNxx.bin

where:

- *faulttype* denotes the type of fault (CPDump, SPDump, or SRAM)
- *MM* is a two-digit number indicating the month
- *DD* is a two-digit number indicating the day
- *YY* is a two-digit number indicating the year
- *HH* is a two-digit number indicating the hour
- *NN* is a two-digit number indicating the minute
- *xx* indicates whether the fault occurred in the AM or PM

**Note:** If multiple DSPs on the same board generate a fault, the core dump file will only contain information about the DSP that generated the initial fault.

## Options

Besides running automatically when a fault is detected, the dlgsnapshot tool can also be run on demand from the command line. Running dlgsnapshot from the command line may also stop the board, depending on the options selected.

If you try to run the dlgsnapshot tool on a board version that doesn't support it, you will get a message that dlgsnapshot is not supported and the command failed.

The dlgsnapshot tool uses the following command line options:

**-f**<number 1 to 6>

This option, which is **mandatory**, indicates which print buffer will be dumped. Values are as follows:

- 1 – SRAM
- 2 – DRIVER\_BOARD\_STATE\_DUMP
- 3 – DRIVER\_BOARD\_COUNTER\_DUMP
- 4 – DOWNLOAD\_OFFDIAG
- 5 – PROCESSOR\_DUMP\_ONLY (this value requires that you use the **-r** option)
- 6 – DUMP\_ALL (this value requires that you use the **-r** option)

**-a**<AUID>

Dumps the print buffer of the processor specified with the **-r** option on the board with the given Addressable Unit IDentifier (AUID).

**-r**<processor number>

The **-r** option is mandatory with **-f** option 5 or 6.

**-l**<logical ID>

Dumps the print buffer of the processor specified with the **-r** option on the board with the given logical ID.

**-p**<physical slot number>

Dumps the print buffer of the processor specified with the **-r** option on the board with the given physical bus number.

**-b**<PCI bus number> **-s**<PCI slot number>

Dumps the print buffer of the processor with the **-r** option on the board with the given PCI bus number and given PCI slot number.

**-h**

Displays the help screen.

**-v**

Displays the version number of dlgsnapshot.

The following shows the usage of the command and its options:

```
dlgsnapshot [-a -r -f] [-l -r -f] [-p -r -f] [-b -s -r -f] [-h] [-v]
```



This chapter provides information about the DM3Insight tool.

DM3Insight is a tool used to capture message and stream traffic from the DM3 board device driver. DM3Insight can be used for the following:

- Debugging/Understanding applications, driver, kernel by capturing traffic and analyzing the trace output
- Viewing messages that are encountered only between the device driver and the DM3 board
- Eliminating orphan messages and streams
- Calculating the turnaround time for messages
- Viewing messages from the board that have incorrect source/destination address or transaction Id and therefore never reach the application
- Viewing error messages from the board like Std\_MsgError or QMsgUndelivered, which are not handled (or reported) by simple applications

For complete information about configuring and using the DM3Insight tool, refer to the DM3Insight online help file (*DM3Insight.chm*). The default directory for this file is *C:\program files\Dialogic\bin*.



This chapter provides reference information about the Dm3post tool.

The Dm3post tool, sometimes referred to as “POST-on-demand”, can perform diagnostics on a stopped board at any time to detect and isolate possible hardware faults.

The DM3 board must be in a ‘Stopped’ state before the Dm3post tool can be run. Dm3post will reset the specified board, forcing the Control Processor (CP) POST diagnostics to run. Dm3post will then retrieve the POST results from the SRAM and provide a PASS/FAIL indication to the user. The board will remain in a stopped state and the user will be required to restart the board.

**Note:** Signal Processors (SP) are not diagnosed by Dm3post. However, SP health is verified as part of the board’s download process. Therefore sufficient diagnostic coverage for hardware is obtained when a board passes POST and is successfully downloaded.

## Options

The Dm3post tool uses the following command line options:

- s<n>  
slot number (required). Use the DCM configuration utility to obtain the board’s slot number.
- b<n>  
bus number (optional if there is only one bus OR if slot number is unique)
- h  
displays the tool’s help screen
- v  
displays the program version

The following example runs Dm3post tool on a board in slot 17, bus 0:

```
dm3post -s17 -b0
```

You will get the following response:

```
Do you wish to continue (y/n)?
```

If you answer Y, the following will be printed to the screen:

```
dm3post processing...
```

The success/failure message will be printed to the screen when POST is complete.



This chapter provides reference information about the Getver tool.

The Getver tool displays the version of any DM3 board binary file. It scans the binary for the standard DM3 version string and prints it to the screen.

The following example scans for the *qvs\_t1.mlm* file and prints its version string to the screen:

```
getver ../data/qvs_t1.mlm
```



This chapter provides reference information about the ISDNtrace tool.

The ISDNtrace tool provides the ability to track Layer 3 (Q.931) messages on the ISDN D-channel. ISDNtrace prints messages on the screen in real time. This trace information can also be captured into a file.

The ISDNtrace tool consumes a portion of the CPU for each trunk that is logged and may generate an exception if CTRL-C is used to exit the tool.

## Options

The ISDNtrace tool uses the following command line options:

-b<n>

Logical ID of board (required). Use the PBL utility (Linux) or the DCM configuration utility (Windows) to obtain the board's logical ID.

-d<n>

The D-channel number (trunk number) on the specified board (required). The default value is 1.

-f<file>

Output file name (required to save output in a file).

**Note:** A space is used after the -f option but not after -b or -d options.

The following example runs the ISDNtrace tool on board 0, D-channel 1 and prints the output to a *trace.txt* file:

```
isdntrace -b0 -d1 -f trace.txt
```





This chapter provides reference information about the KernelVer tool.

The KernelVer tool queries the board's kernel running on a particular processor for its version number. This tool can be used to verify whether or not a processor has crashed.

## Options

The KernelVer tool uses the following command line options:

-b<n>

Logical ID of board (required). Use the PBL utility (Linux) or the DCM configuration utility (Windows) to obtain the board's logical ID.

--d<level>

Do not modify. Leave this at the default value of 0.

-f<file>

Output file name (required to save output in a file).

-p<n>

Processor number (required). Lowest allowable value is 1.

-l<n>

Number of times the program will retrieve the version (optional). You can use this option to repeatedly ping the board's kernel to generate message traffic.

-h

Displays the help screen.

-v

Displays the version number.

The following example runs the KernelVer tool on board 1, processor 2:

```
kernelver -b1 -p2
```



This chapter provides reference information about the MercMon tool.

MercMon provides counter information about DM3 board device drivers (Class Driver and Protocol Drivers). These drivers maintain various counters that can aid in monitoring system activities and interpreting system behaviors.

The MercMon tool provides counter information about a DM3 board's class driver and protocol driver.

## Class Driver Counters

This section provides details about the class driver counters that are used by the MercMon tool. The class driver counters are as follows:

**Note:** All counters are on a per board basis except for FailMpathFind and FailStrmFind.

### CompleteReads

returns the size (in bytes) of all the read requests completed by the protocol driver (i.e., the actual size in KB of all the data read from the board)

### CompleteSends

returns the size (in bytes) of all the messages sent to or received from the board

### CompleteWrites

returns the size (in bytes) of all the write requests completed by the Protocol Driver (i.e., the actual size in KB of all the data written to the board)

### FailMpathFind

returns the number of times a request to get a message handle (Mpath) failed (global counter)

### FailStrmFind

returns the number of times a request to get a stream handle failed (global counter)

### NumCanTakes

returns the number of "can takes" received for that board

### NumReads

returns the total number of read requests sent down to the protocol driver

### NumSends

returns the total number of messages sent down to the protocol driver

### NumWrites

returns the total number of write requests sent down to the protocol driver

### NumWriteSplit

returns the number of write requests, which were split for that board

### NumOpenedStreams

returns the number of open streams on that board at any given time

NumCloseStreamErr	returns the number of stream close requests failed on that particular board
NumOpenStreamErr	returns the number of stream open requests failed on that particular board
NumStreamClose	returns the number of stream close requests on that particular board
NumStreamOpen	returns the number of stream open requests on that particular board
SizeReads	returns the size (in bytes) of all the messages posted down to the protocol driver
SizeSends	returns the size (in bytes) of all the messages sent down to the protocol driver
SizeWrites	returns the size (in bytes) of all the write requests sent down to the protocol driver
TimeoutReads	returns the number of read requests that timed out
TimeoutSends	returns the number of messages that timed out while waiting to be sent to the board or awaiting a reply from the board
TimeoutWrites	returns the number of write requests that timed out

## Protocol Driver Counters

This section provides details about the protocol driver counters that are used by the MercMon tool. The protocol driver counters are as follows:

MsgsInPerSramSession	returns the number of messages read from the SRAM in one session
MsgsOutPerSramSession	returns the number of messages written to the SRAM in one session
NoInDataDPCisr	returns the number of times when we received an interrupt but there was no data transfer between the SRAM and the HOST
StrmsOutPerSramSession	returns the number of streams written to the SRAM in one session (i.e., number of data blocks written)
StrmsInPerSramSession	returns the number of streams read from the SRAM in one session (i.e., the number of data blocks read)
TotalAsyncMsgQDone	returns the number of requests completed in the AsyncMsgQ

TotalBadSramAddr	returns the number of times we tried reading a message/data (streams) from the SRAM but the SRAM address was wrong
TotalBadSramAddrAlloc	returns the number of times the system tried to allocate a data block but the address was wrong
TotalBadSramAddrRelease	returns the number of times the system tried to release a data block but the address was wrong
TotalBadSramDataCount	returns the total number of times we got a data block of size greater than the SRAM_DATA_BLOCK_SIZE
TotalBadSramOffset	returns the number of times we tried reading data (streams) from the SRAM but the offset calculation was incorrect
TotalBadSramOffsetAlloc	returns the number of times we allocated data block but the offset was incorrect
TotalBigMessagesRcvd	returns the number of times we read a “BIG” message from the SRAM (i.e., message size greater than 24 bytes)
TotalBigMessagesSent	returns the number of times we wrote a “BIG” message to the SRAM (i.e., message size greater than 24 bytes)
TotalBogusInterrupts	Not Used
TotalDpcOverruns	returns the total number of DPC overruns (i.e., we were not expecting an interrupt but we got one)
TotalDmaInterrupts	returns the total number of interrupts received from the board for performing DMA
TotalFatSramBlocks	returns the number of times the system received a “CHAINED” data block (i.e., the data blocks are linked together)
TotalIrpsCancelled	returns the total number of cancelled requests in any queue
TotalMsgInQ	returns the number of requests in the Message In Queue, which are awaiting a reply from a board
TotalMsgInQDone	returns the number of requests completed from the Message In Queue (i.e., requests completed and sent to the application)
TotalMsgInSram	returns the total number of messages read from the SRAM

- TotalMsgOutQDone**  
returns the number of requests completed from the Message Out Queue (these requests are sent to the board and then moved to the Message In Queue, if expecting a reply, or completed and sent to the application)
- TotalMsgOutSram**  
returns the total number of messages written to the SRAM
- TotalMsgOverruns**  
returns the total number of overruns for the orphan message queue (i.e., we had an orphan message but there was no space in the Orphan Message Queue)
- TotalMsgTimeouts**  
returns the total number of requests timed out while waiting to be written to the SRAM or awaiting a reply from the SRAM (i.e. either in the Message In Q, Message Out Q or the Async Message Q)
- TotalOrphanMsgsv**  
returns the total number of Orphan Messages (i.e., messages which were read from the SRAM but do not have any pending requests from the application)
- TotalOrphanMsgVolume**  
returns the size (in bytes) of the messages present in the Orphan message queue at any given time
- TotalOrphanStrms**  
returns the total number of Orphan Streams in the Orphan Stream Table (i.e., data which was read from the SRAM but did not have any pending Read Requests from the application)
- TotalOrphanStrmMatches**  
returns the total number of streams matched in the Orphan Stream table
- TotalOrphanStrmVolume**  
returns the size (in bytes) of the data present in the Orphan Stream table at any given time
- TotalSramDataFull**  
returns a count of the number of times the HOST tried to write a stream (data) to the SRAM but the SRAM was full
- TotalSramGrantInterrupts**  
returns the number of “HOST SRAM PENDING” interrupts
- TotalSramGrantLost**  
not used
- TotalSramInterrupts**  
returns the total number of “NORMAL” interrupts received from the board (e.g., there is something to read from the SRAM)
- TotalSramMsgFull**  
returns a count of the number of times the HOST tried to write a message to the SRAM but the SRAM was full
- TotalStrmInQ**  
returns the number of read requests pending in the Stream In Queue

**TotalStrmInQDone**

returns the number of read requests completed from the Stream In Queue (i.e., read requests completed and sent to the application)

**TotalStrmInSram**

returns the number of data packets read from the SRAM

**TotalStrmOutQ**

returns the number of writes performed by the user on that particular board. It is also incremented whenever the application sends an End Of Stream command.

**TotalStrmOutQDone**

returns the number of Write requests completed from the Stream Out Queue (i.e., Write requests completed and sent to the application)

**TotalStrmOutSram**

returns the number of data packets written to the SRAM

**TotalStrmOverruns**

returns the number of Overruns for the Orphan Stream Table (e.g., we had an Orphan Stream but we had exceeded the maximum amount of memory allocated for the Orphan Stream Table or we could not allocate memory.)

**TotalStrmTimeouts**

returns the total number of Read or Write Requests timed out (from the Stream In Queue or the Stream Out Queue)

**TotalUnknownInterrupts**

returns the total number of unknown interrupts received from the board

## Options

The MercMon tool uses the following command line options:

**/f<file>**

log file name (default is *mercmon.log*)

**/t<n>**

display timer. The time interval (in milliseconds) between each screen refresh (default is 1000).

**/l<n>**

logging interval. The time interval (in seconds) between each log file update (default is 600).

**/w**

enable/disable logging to a file (default is enabled)

**/?**

displays the help screen

The following example enables logging to the *mercmon.log* file every five minutes and refreshes the screen every two seconds:

```
mercmon /t 2000 /l 300
```





This chapter provides reference information about the PDK Trace tool.

The DM3 PDK Protocol Trace (PDK Trace) tool allows those who use a DM3 PDK protocol to log specific information related to the operation of the protocol. The PDK Trace tool is useful for protocol developers because it can trace the runtime states, input signals, output signals, and decision branches of the SDL protocol. The PDK Trace tool allows you to specify which channels on the board to begin tracing. This can be a single channel on one trunk, a single channel on multiple trunks, a range of channels on one trunk, or a range of channels on multiple trunks.

The PDK Trace tool requires the Global Call Protocols, which can be downloaded from the Intel Telecom Support Resources web site (<http://resource.intel.com/telecom/support/download/index.htm>).

The PDK Trace tool's executable name is *pdktrace.exe* for the Windows operating system and *pdktrace* for the Linux operating system. The PDK Trace tool is a command line application that will create a system process thread to interact with the DM3 system to capture trace data. This data will be streamed to a file on the host system.

Upon completion of tracing, the data will be stored on the host system in a file specified when tracing was started (default: *pdktrace.log*). The file is in an unreadable binary format and will need to be converted to a readable format. For help with this, contact technical support (<http://resource.intel.com/telecom/support/contact.htm>).

## Options

The PDK Trace tool uses the following command line options:

**-b#**

This **required** option specifies the logical board ID of the board to trace.

**Example:** `pdktrace -b0` where 0 is the logical board ID of the destination board.

**-l[#] or -l[#-#]**

Specify which trunk(s) the channels to be traced are located on. The default value is 1 (trunk 1).

**Example 1:** `pdktrace -b0 -l[1] /* Single trunk */`

**Example 2:** `pdktrace -b0 -l[1-4] /* Range of trunks */`

**-c[#] or -c[#-#]**

Specify which channel(s) on the specified trunks to trace. The default value is 1 (channel 1).

**Example 1:** `pdktrace -b0 -l[1] -c[5] /* Single channel */`

**Example 2:** `pdktrace -b0 -l[1-4] -c[1-30] /* Range of channels */`

**-f[filename]**

Specify the name of a file on the host system to write the trace data to. The default is *pdktrace.log*.

-i

This option is **required** only when you **first** use the PDK Trace tool. This option is used to initialize the DM3 Tracer Component in the firmware.

**Note:** This option should only be used the first time the utility is executed after the board is downloaded

-v

Prints the version number of the utility.

-, -h

Prints the help screen (command line options) for the utility.

**Warning:** Due to memory constraints in the embedded DM3 system, the tool will limit the number of channels that can be traced simultaneously to 60. Trying to trace more than 60 channels per board can cause unpredictable results.

## Sample Scenarios

The following are some sample scenarios in which the PDK Trace tool can be used and the command line options used to specify each configuration. These sample scenarios assume that the logical ID of the board being used is 0.

**Scenario 1:** For board 0, trace channel 1 on trunk 1

```
pdktrace -b0
pdktrace -b0 -l [1]
pdktrace -b0 -l [1] -c [1]
```

**Scenario 2:** For board 0, trace channels 10-20 on trunk 1

```
pdktrace -b0 -c [10-20]
pdktrace -b0 -l [1] -c [10-20]
```

**Scenario 3:** For board 0, trace channel 1 on trunks 1-4

```
pdktrace -b0 -l [1-4]
pdktrace -b0 -l [1-4] -c [1]
```

**Scenario 4:** For board 0, trace channels 1-24 on trunks 1-4

```
pdktrace -b0 -l [1-4] -c [1-24]
```

**Note:** If any of the above scenarios were being executed on a board for the first time after it was downloaded, the -i command line option should also be used as follows:

```
pdktrace -b0 -l [1-4] -c [1-24] -i
```

This chapter provides reference information about the Phone tool.

The Phone tool uses the TSC and ToneGen instances and requires a TSC component. The Phone tool can control a single DM3 resource channel (make calls, wait for calls etc.), monitor channel and call states, and send call control operations to a DM3 Global Call resource.

When using Dynamic Routing configurations, the Audio Control tool (described in the Administration Guide for the system release) and the audio buttons in other tools, such as the Phone tool, will not work.

## Options

The Phone tool uses the following command line options:

-b<n>

Logical ID of board (required). Use the PBL utility (Linux) or the Intel® Dialogic® Configuration Manager (Windows) to obtain the board's logical ID.

-l<n>

Line number (optional). The default value is 1.

-chan<n>

Channel number (optional). The default value is 1.

The following example runs the Phone tool on board 1, line 1:

```
phone -b1 -l1
```



This chapter provides reference information about the QError tool.

The QError tool displays the string associated with a particular error code returned in a QResultError message from the board.

**Note:** QError displays error code strings for DM3 kernel errors only.

The error code string is generated with a Perl\* script directly from the header files, so it as accurate as the comments in the header files. The source to the QError code is arranged so that the “get error string” function can be pulled out and used in any application.

## Options

The QError tool uses the following command line options:

- b<n>  
Base of input number (hexadecimal is 16, decimal is 10, default is 16).
- d<level>  
Do not modify. Leave this at the default value of 0.
- h  
Displays the help screen.
- v  
Displays the version number.

The following example runs the QError tool on error code 28008:

```
qerror 28008
```

The output from this example would be as follows:

```
ERROR==> 0x28008 (163848)
Kernel Error:
Cluster does not exist or cannot be found.
```



This chapter provides reference information about the Status Monitor tool.

The Status Monitor tool enables you to track the state of the TSC as well as the state of the bits on a robbed bit or CAS line. You can use Status Monitor for both troubleshooting and administration. For troubleshooting, you can monitor the call state for problems such as hung channels. For administration, you can watch call progress and channel usage.

The Status Monitor tool consumes a portion of the CPU for each trunk that is logged and may generate an exception if CTRL-C is used to exit the tool. It may take a long time to start the monitoring: about 1 minute per board

The CAS instance is not valid until the line is in service, so you will have to run an application or the Lineadmin and Phone tools to set the channel in-service before using the bit monitoring feature. For information about the Lineadmin tool, refer to the Administration Guide for the system release. For information about the Phone tool, refer to [Chapter 16, “Phone Reference”](#).

The Status Monitor tool is best viewed in resolutions greater than 1024 x 768.

## Options

This section describes the one option for the Status Monitor tool and a sample of Status Monitor output.

Status Monitor has the following option:

-board <board list>

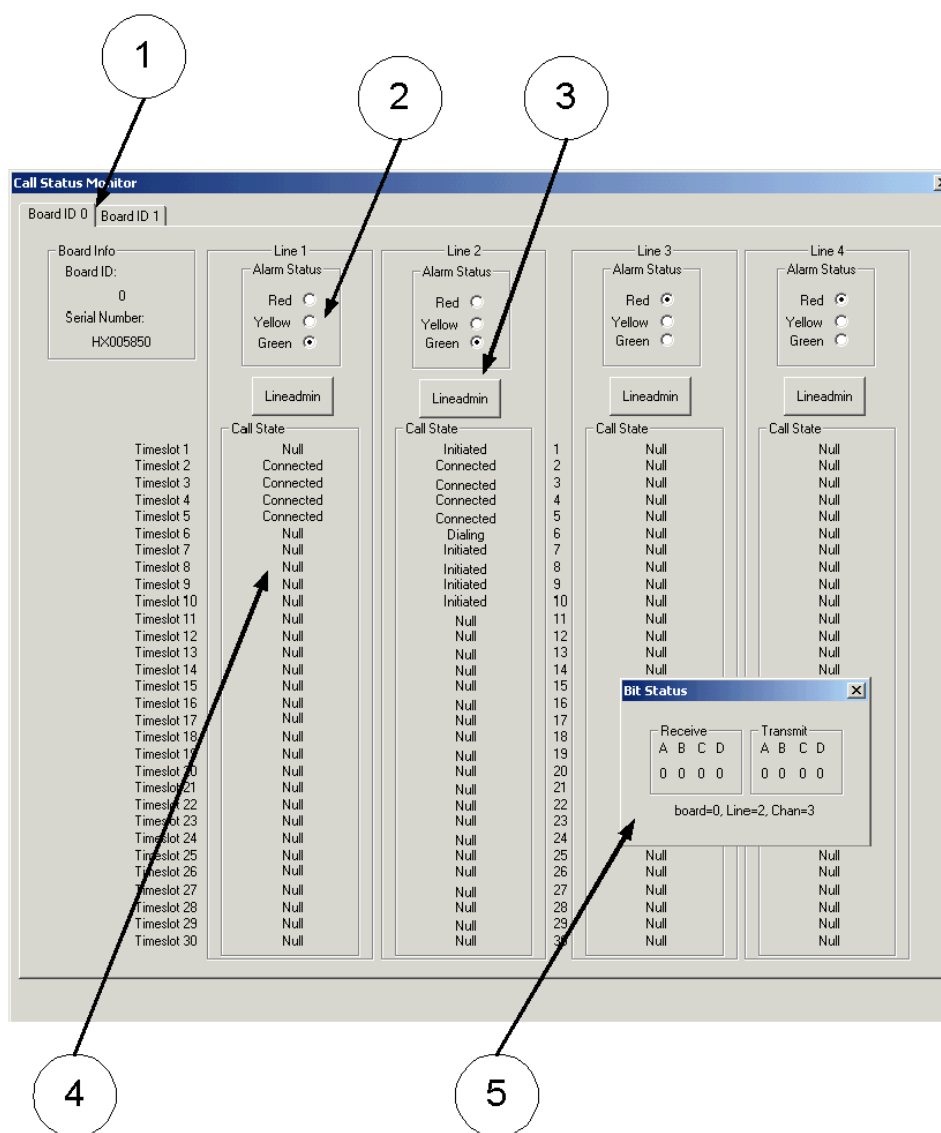
Logical ID of the board(s) to trace (optional). The default is to monitor all DM3 boards in the system. Use the DCM configuration utility to obtain the board's logical ID.

The following will run the Status Monitor tool on boards 0 and 1:

```
StatusMon -board 0 1
```

Figure 1 provides an example of output from the Status Monitor tool.

Figure 1. Example of Status Monitor Output



The following numbered list corresponds to the labels in Figure 1.

1. You can toggle between monitored boards by clicking on the tab that represents the board ID.
2. This part of the display shows the current alarm state on the trunk (red, yellow, or green).
3. Use this button to invoke the Lineadmin tool to view and set additional alarms.
4. This part of the display allows you to monitor the call state for each line on the board.
5. Clicking on a line will cause the bit information to be displayed in this window. (All 1's will be displayed for ISDN lines.)



This chapter provides reference information about the StrmStat tool.

The StrmStat tool displays the current state of a set of specified streams. Possible states of a stream as shown by the StrmStat display are as follows:

- closing
- closed
- close failed
- opening
- opened for write
- opened for read
- open failed

## Options

The StrmStat tool uses the following command line options:

- b<n>  
logical ID of the board (required). Use the DCM configuration utility to obtain the board's logical ID.
- f<file>  
log file name (default is *strmstat.log*)
- i<n>  
number of streams on which to report (default is 120)
- s<n>  
first stream to report (default is 1)
- ?  
displays the program help screen (optional)

The following command displays the stream state for the stream IDs 1-16 on board 0:

```
strmstat /b 0 /i 16
```



This chapter provides reference information about the TSP Config tool.

The TSP Config tool sets and retrieves DM3 protocol variant parameters. Using this tool, you can change protocol variant parameters dynamically from one call to the next.

This tool can be used on any DM3 product that has a T1 CAS or T1/E1 ISDN TSP resource. The tool does not work for R2MF protocols or Analog protocols.

## Options

This section describes the options for the TSP Config tool.

TSP Config has the following options:

**-board <n>**

Board number (required). Use the Listboards utility to obtain the board number. For information about the Listboards utility, refer to the Administration Guide for the system release.

**-id <n>**

Protocol variant id (1 - 32), whose parameters you are configuring dynamically. For example, `tspconfig -board 0 -id 2`, will display the parameters of protocol variant id 2.

**Example:** This example runs the TSP Config tool on board 1:

```
tspconfig -board 1
```



This chapter provides reference information about the TSP Monitor tool.

TSP Monitor performs the following:

- Monitors one or two DM3 Global Call resource channels.
- Traces all levels of the protocol including:
  - DM3 Global Call resource call control operations from clients
  - DM3 Global Call resource call state changes
  - DM3 Global Call resource channel state changes
  - CAS signaling bits
- Launches an audio tool for recording/playback purposes on the channel(s). It plays audio data (a file from the host) using the TSP.
- Checks timing via point and click on GUI
- Tunes protocols and identifies configuration problems

## Options

This section describes the options for the TSP Monitor tool.

TSP Monitor has the following options:

-board <*n*>

Board number (required). Use the Listboards utility to obtain the board number. For information about the Listboards utility, refer to the Administration Guide for the system release.

-line <*n*>

Line number (optional, default is 1)

-chan <*n*>

Channel number (optional, default is 1)

**Example:** This example runs the TSP Monitor tool on board 0, line 1, channel 1:

```
tspmon -board 0 -line 1 -chan 1
```



This chapter provides reference information about the TSP Tracer tool.

TSP Tracer traces CAS protocols with timing information and saves trace information to a log file. The TSP Tracer tool does not support ISDN protocol tracing. For ISDN protocol tracing, use the TSP Monitor tool, which supports all protocols.

## Options

This section describes the options for the TSP Tracer tool.

The TSP Tracer tool has the following command line options:

-board *<n>*

Board number (required). Use the Listboards utility to obtain the board number. For information about the Listboards utility, refer to the Administration Guide for the system release.

-line *<n>*

Line number (optional, default is 1)

-chan *<n>*

Channel number (optional, default is 1)

**Example:** This example runs the TSP Tracer utility on board 0, line 1, channel 1:

```
tsptrace -board 0 -line 1 -chan 1
```





## B

binary file 29  
BL 13

## C

CallInfo tool 13  
CAS Trace tool 15  
class driver counters 35  
Control Processor 27

## D

DebugAngel tool 17  
Digit Detector tool 21  
dlgsnapshot tool 23  
DM3 PDK Protocol Trace 41  
DM3Insight.chm 25

## G

Getver tool 29  
Global Call 43

## H

hardware requirements 11

## I

ISDN D-channel 31  
ISDN-related messages 13  
ISDNtrace tool 31

## K

KernelVer tool 33

## L

Logical ID 13

## M

mercmon  
    class driver counters 35  
    protocol driver counters 36

## P

PDK Trace tool 41  
Perl 45  
Phone tool 43  
protocol driver counters 36

## Q

QError 45  
QError tool 45  
QResultError 45

## S

Signal Detector 21  
Signal Processors 27  
software requirements 11  
Status Monitor tool 47  
StrmStat tool 49  
strmstat.log 49

## T

Telephony Service Provider 13  
Tone Generator 21  
TSC 43  
TSP 13  
TSP Config utility 51  
TSP Monitor utility 53  
TSP Trace utility 55

## U

utilities  
    TSP Config 51  
    TSP Monitor 53  
    TSP Trace 55

## V

version string 29