



Total Performance. Total Control.™

Netronome Flow Engine

Series i8000/i4000

Boot Monitor and Diagnostics Reference

Product code
060-00001-006

Netronome Flow Engine: Boot Monitor and Diagnostics Reference

Copyright © 2006-2008 Netronome Systems, Inc.

COPYRIGHT

No part of this publication or documentation accompanying this Product may be reproduced in any form or by any means or used to make any derivative work by any means including but not limited to by translation, transformation or adaptation without permission from Netronome Systems, Inc., as stipulated by the United States Copyright Act of 1976. Contents are subject to change without prior notice.

WARRANTY

Netronome warrants that any media on which this documentation is provided will be free from defects in materials and workmanship under normal use for a period of ninety (90) days from the date of shipment. If a defect in any such media should occur during this 90-day period, the media may be returned to Netronome for a replacement.

NETRONOME DOES NOT WARRANT THAT THE DOCUMENTATION SHALL BE ERROR-FREE. THIS LIMITED WARRANTY SHALL NOT APPLY IF THE DOCUMENTATION OR MEDIA HAS BEEN (I) ALTERED OR MODIFIED; (II) SUBJECTED TO NEGLIGENCE, COMPUTER OR ELECTRICAL MALFUNCTION; OR (III) USED, ADJUSTED, OR INSTALLED OTHER THAN IN ACCORDANCE WITH INSTRUCTIONS FURNISHED BY NETRONOME OR IN AN ENVIRONMENT OTHER THAN THAT INTENDED OR RECOMMENDED BY NETRONOME.

EXCEPT FOR WARRANTIES SPECIFICALLY STATED IN THIS SECTION, NETRONOME HEREBY DISCLAIMS ALL EXPRESS OR IMPLIED WARRANTIES OF ANY KIND, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE.

Some jurisdictions do not allow the exclusion of implied warranties, so the above exclusion may not apply to some users of this documentation. This limited warranty gives users of this documentation specific legal rights, and users of this documentation may also have other rights which vary from jurisdiction to jurisdiction.

LIABILITY

Regardless of the form of any claim or action, Netronome's total liability to any user of this documentation for all occurrences combined, for claims, costs, damages or liability based on any cause whatsoever and arising from or in connection with this documentation shall not exceed the purchase price (without interest) paid by such user.

IN NO EVENT SHALL NETRONOME OR ANYONE ELSE WHO HAS BEEN INVOLVED IN THE CREATION, PRODUCTION, OR DELIVERY OF THE DOCUMENTATION BE LIABLE FOR ANY LOSS OF DATA, LOSS OF PROFITS OR LOSS OF USE OF THE DOCUMENTATION OR FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, EXEMPLARY, PUNITIVE, MULTIPLE OR OTHER DAMAGES, ARISING FROM OR IN CONNECTION WITH THE DOCUMENTATION EVEN IF NETRONOME HAS BEEN MADE AWARE OF THE POSSIBILITY OF SUCH DAMAGES. IN NO EVENT SHALL NETRONOME OR ANYONE ELSE WHO HAS BEEN INVOLVED IN THE CREATION, PRODUCTION, OR DELIVERY OF THE DOCUMENTATION BE LIABLE TO ANYONE FOR ANY CLAIMS, COSTS, DAMAGES OR LIABILITIES CAUSED BY IMPROPER USE OF THE DOCUMENTATION OR USE WHERE ANY PARTY HAS SUBSTITUTED PROCEDURES NOT SPECIFIED BY NETRONOME.

Revision History

Date	Revision	Description
2 April 2008	006	Updated support for NFD version 1.3.
1 February 2008	005	Updated support for NFD version 1.2.x.
1 July 2007	004	Support for NFD version 1.2.
1 March 2007	003	New document layout.
15 September 2006	002	New features.
1 June 2006	001	Initial release.

Table of Contents

1. Introduction	6
1.1. Scope	6
1.2. Related Documents	6
1.3. Terminology	6
2. Boot Monitor	7
2.1. Introduction	7
2.2. Boot Monitor console	7
2.3. Recovery Loader console	7
2.4. Flash Memory	8
2.4.1. Store Boot Monitor image	9
2.4.2. Store Recovery Loader image	9
2.4.3. Store Linux Kernel image	10
2.4.4. Store Linux Ramdisk image	12
2.4.5. Store Diagnostics image	13
2.5. Configuration	14
2.6. Boot Monitor Command Reference	16
2.6.1. alias	17
2.6.2. baudrate	18
2.6.3. cache	18
2.6.4. cksum	19
2.6.5. dump	19
2.6.6. exec	20
2.6.7. fconfig	20
2.6.8. fis create	21
2.6.9. fis delete	22
2.6.10. fis erase	22
2.6.11. fis free	23
2.6.12. fis init	23
2.6.13. fis list	23
2.6.14. fis load	24
2.6.15. fis write	24
2.6.16. go	25
2.6.17. help	25
2.6.18. history	25
2.6.19. ip_address	26
2.6.20. load	26
2.6.21. mcmp	27
2.6.22. mcopy	27
2.6.23. mfill	28
2.6.24. ping	28
2.6.25. reset	29
2.6.26. version	29
2.6.27. x	29
3. Diagnostics	31
3.1. Activating the Diagnostic Application	31
3.2. Diagnostic Command Reference	31

3.2.1. ?, h, help	33
3.2.2. banner	33
3.2.3. blockcopy	34
3.2.4. c	34
3.2.5. doff	35
3.2.6. don	35
3.2.7. exit	35
3.2.8. fill	36
3.2.9. h	36
3.2.10. help	37
3.2.11. i2cread	37
3.2.12. i2cwrite	37
3.2.13. ioff	38
3.2.14. ion	38
3.2.15. log	39
3.2.16. p	39
3.2.17. pcicfg	40
3.2.18. pciio	40
3.2.19. pcimem	41
3.2.20. reset	42
3.2.21. r	42
3.2.22. sethoe	42
3.2.23. setverbose	43
3.2.24. sramdeskew	43
3.2.25. test	44
3.2.26. tf	45
3.2.27. t all	46
3.2.28. t ee	46
3.2.29. t ethpci	46
3.2.30. t media	47
3.2.31. t generate	48
3.2.32. t gpio	50
3.2.33. t ixcp	50
3.2.34. t led	51
3.2.35. t mem	51
3.2.36. t msf	52
3.2.37. t pci	53
3.2.38. t pcie	53
3.2.39. t slowport	54
3.2.40. t temp	55
3.2.41. t trng	56
3.2.42. t uart	56
3.2.43. t ueng	56
3.2.44. t xscale	57
3.2.45. w	58
3.2.46. wrv	58

4. Technical Support 59

1. Introduction

1.1 Scope

This manual serves as a reference for the Boot Monitor and diagnostic application of the Netronome Flow Engine (NFE) series i8000/i4000. The Boot Monitor and diagnostic application commands are described in detail.

1.2 Related Documents

Descriptive Name	Filename
Netronome Flow Engine Reference Guide	NFE_RG.pdf
IXDP28x1 ATCA Development Platform System User's Manual	IXDP28X1_SystemUserManual_v5.pdf (Supplied with the Intel® IXA SDK 4.2)

1.3 Terminology

The following terms and abbreviations are used in this document:

Acronym	Description
DLL	Delay Locked Loop
FIS	Flash Image System
GPIO	General Purpose Input/Output
MMU	Memory Management Unit
NFE	Netronome Flow Engine
PCIe	PCI Express
QDR	Quad Data Rate
RAM	Random Access Memory
SDK	Software Development Kit
SRAM	Static RAM
TFTP	Trivial File Transfer Protocol
TRNG	True Random Number Generator
XS	XScale processor

2. Boot Monitor

2.1 Introduction

The Boot Monitor program loads and executes when power is supplied to the NFE hardware platform or when the platform is reset. This program supplies an environment where program images can be loaded from flash memory, over a Ethernet network connection or a RS232 serial cable. These images may be stored to flash memory using the Boot Monitor. A Boot script can be configured to automatically load and execute programs after the Boot Monitor starts.

The Boot Monitor console is accessed via a RS232 serial cable that is attached to the NFE hardware platform. The Recovery Loader is similar to the Boot Monitor and is used to store the Boot Monitor image in flash memory. These two programs are not able to write to their own memory location in flash memory. Only the Recovery Loader is able write a Boot Monitor image into flash memory and only the Boot Monitor is able to write a Recovery Loader image to flash memory. This ensures that the Boot Monitor can be restored if it is accidentally erased from flash memory.

2.2 Boot Monitor console

Follow the steps below to access the Boot Monitor console:

1. Attach a terminal or terminal emulator via a serial cable to the NFE hardware platform's RS-232 serial interface. The factory default settings for this interface are 115200 baud, 8 data bit, No parity and 1 stop bit.
2. Power on the host in which the NFE hardware platform is installed or, if the host has already been powered on, reset the hardware platform using the hardware reset pushbutton, or reset the host platform using its hardware reset button.
3. After the initialization sequence, the following message will appear which indicates that the system is about to execute a preconfigured boot script:

```
== Executing boot script in 5.000 seconds - enter ^C to abort
```

When this message appears, press Ctrl-C to abort the normal boot sequence and enter the boot monitor.

4. After the Boot Monitor image has loaded, a prompt will be displayed:

```
RedBoot>
```

The Boot Monitor is now ready to accept commands.

2.3 Recovery Loader console

Follow the steps below to activate the Recovery Loader and access the Recovery Loader console:

1. Attach a terminal or terminal emulator via a serial cable to the NFE hardware platform's RS-232 serial interface. The factory default settings for this interface are 115200 baud, 8 data bit, No parity and 1 stop bit.
2. Power on the host in which the NFE hardware platform is installed or, if the host has already been powered on, reset the hardware platform using the hardware reset pushbutton, or reset the host platform using its hardware reset button.
3. The following message will appear on the RS-232 Serial console:

```
Recovery loader initializing...
```

As soon as this message appears, type the following three characters:

```
= = 1
```

Type an equal sign, an equal sign again and the digit one. Do not type spaces or press [Enter]. The Recovery Loader will start with the following message:

```
Starting recovery loader.
```

```
Netronome Systems NFE Boot Monitor version 1.31.1-RECOVER
```

4. The Recovery Loader console bootstrap process ends with the display of the prompt:

```
Recover>
```

The Recovery Loader is now ready to accept commands, for example to write a new Boot Monitor image to the flash memory.

2.4 Flash Memory

This section describes how to store various application/data images in the flash memory of the NFE hardware platform. The image to be stored must be copied into memory and must therefore be accessible from a TFTP sever or as an alternative via the terminal emulator that is attached to the RS-232 serial interface using the xmodem or ymodem protocols.

The following table shows a list of the current images that are used and details regarding their memory locations.

Table 2.1. Flash Images and their locations

Name	Flash Address	Memory Address	Length	Entry Point
Recovery Loader	0xC4000000	0xC4000000	0x00100000	0x00000000
Boot Monitor	0xC4100000	0xC4100000	0x00040000	0x00000000
Persistent storage	0xC4200000	0xC4200000	0x00080000	N/A
RAM Disk	0xC4300000	0x2C600000	0x00300000	0x2C600000
Diags	0xC4D00000	0x00280000	0x000A0000	0x00280000
Linux	0xC4DA0000	0x00280000	0x000C0000	0x00280000

The list of program images available on the flash memory can be accessed via the Boot Monitor console. Refer to Section 2.2 for a description on how to access the Boot Monitor console. Refer to Section 2.6.13 for an explanation of the `fis list` command to list the images stored in flash memory.

2.4.1 Store Boot Monitor image

1. Activate the **Recovery Loader** as described in Section 2.3.
2. Ensure the Boot Monitor image is available on a TFTP server. The image is called `redboot_monitor.swp.1.31.1` and is available in the NFE tarball.
3. Load the image into the NFE hardware platform RAM (replacing **xyz** with the IP address of the TFTP server):

```
Recover> load -r -v -m tftp -b 0x04000000 -h xyz redboot_monitor.swp.1.31.1
```

The following message is displayed:

```
Raw file loaded 0x04000000-0x04030f13, assumed entry at 0x04000000
```

To calculate the size of the image, subtract the first value in the message from the second value and add 1:

```
0x04030f13 - 0x04000000 + 1 = 0x30f14
```

4. Alternatively, the Boot Monitor image may be loaded using the RS-232 serial interface. To use this method, enter:

```
Recover> load -r -v -b 0x04000000 -m xmodem
```

then send the file `boot_monitor_v1.1.bin` using the terminal emulator's `xmodem send` command.

5. Next the image will be transferred from RAM to the flash memory. Enter the following two commands:

```
Recover> fis erase -f 0xc4100000 -l 0x30f14
```

```
Recover> fis write -f 0xc4100000 -b 0x10000000 -l 0x30f14
```

and at the prompt to confirm writing, enter: **y**

6. When the flash memory programming has completed, reset the NFE hardware platform using the hardware reset pushbutton, or reset the host platform using its hardware reset button.

2.4.2 Store Recovery Loader image

1. Activate the **Boot Monitor** as described in Section 2.2.
2. Make sure the Recovery Loader image is available on a TFTP server. The image is called `boot_loader_v1.1.bin` and is available in the NFE tarball.
3. Load the image into the NFE hardware platform RAM (replacing **xyz** with the IP address of the TFTP server):

```
RedBoot> load -r -v -b 0x04000000 -h 10.0.0.1 redboot_loader.swp.1.31.1
```

The following message is displayed:

```
Raw file loaded 0x04000000-0x04030f1b, assumed entry at 0x04000000
```

To calculate the size of the image, subtract the first value in the message from the second value and add 1:

$$0x04030f1b - 0x04000000 + 1 = 0x30f1c$$

- Alternatively, the Recovery Loader image may be loaded using the RS-232 serial interface. To use this method, enter:

```
RedBoot> load -r -v -b 0x04000000 -m xmodem
```

then send the file `boot_loader_v1.1.bin` using the terminal emulator's xmodem send command.

- Next the Boot Monitor image will be transferred from RAM to flash. Enter the following two commands:

```
RedBoot> fis erase -f 0xc4000000 -l 0x30f1c
```

```
RedBoot> fis write -f 0xc4000000 -b 0x04000000 -l 0x30f1c
```

and at the prompt to confirm writing, enter: **y**

- When the flash memory programming has completed, reset the NFE hardware platform using the hardware reset pushbutton, or reset the host platform using its hardware reset button.

2.4.3 Store Linux Kernel image

- Activate the **Boot Monitor** as described in Section 2.2.
- Make sure the Linux Kernel image is available on a TFTP server. The image is called `zImage.i8000.r43` (or `zImage.i4000.r43`) and is available in the NFE tarball.
- Load the image into the NFE hardware platform RAM (replacing xyz with the IP address of the TFTP server):

```
RedBoot> load -r -v -m tftp -b 0x00280000 -h xyz zImage.i8000.r43
```

The following message is displayed:

```
Raw file loaded 0x00280000-0x0032fec3, assumed entry at 0x00280000
```

To calculate the size of the image, subtract the first value in the message from the second value and add 1:

$$0x0032fec3 - 0x00280000 + 1 = 0xafec4.$$

- Delete any image named linux from flash. Enter:

```
RedBoot> fis list
```

The output will list images that are stored in flash memory:

Name	FLASH addr	Mem addr	Length	Entry point
... (reserved)	0xc4000000	0xc4000000	0x00100000	0x00000000

RedBoot	0xC4100000	0xC4100000	0x00040000	0x00000000
linuxdata	0xC4200000	0xC4200000	0x00080000	0xFFFFFFFF
ramdisk	0xC4300000	0x2C600000	0x00300000	0x2C600000
diag	0xC4D00000	0x00280000	0x000A0000	0x00280000
linux	0xC4DA0000	0x00280000	0x000C0000	0x00280000
FIS directory	0xC4FE0000	0xC4FE0000	0x0001F000	0x00000000
RedBoot config	0xC4FFF000	0xC4FFF000	0x00001000	0x00000000
...				

To delete the Linux image, enter:

```
RedBoot> fis delete linux
```

and at the prompt to confirm writing, enter: **y**

5. Store the new image to flash memory:

```
RedBoot> fis create -b 0x280000 -l 0xafec3 -r 0x280000 -e 0x280000 -f 0xC4DA0000 linux
```

6. Make sure an empty image in flash memory named linuxdata is present:

```
RedBoot> fis list
```

The output will list images that are stored in flash:

Name	FLASH addr	Mem addr	Length	Entry point
...				
(reserved)	0xC4000000	0xC4000000	0x00100000	0x00000000
RedBoot	0xC4100000	0xC4100000	0x00040000	0x00000000
linuxdata	0xC4200000	0xC4200000	0x00080000	0xFFFFFFFF
ramdisk	0xC4300000	0x2C600000	0x00300000	0x2C600000
diag	0xC4D00000	0x00280000	0x000A0000	0x00280000
linux	0xC4DA0000	0x00280000	0x000C0000	0x00280000
FIS directory	0xC4FE0000	0xC4FE0000	0x0001F000	0x00000000
RedBoot config	0xC4FFF000	0xC4FFF000	0x00001000	0x00000000
...				

If the image, linuxdata, is not present, it must be created. Enter:

```
RedBoot> fis create -l 0x80000 -f 0xc4200000 -b 0xc4200000 linuxdata
```

7. Next, erase the contents of this area:

```
RedBoot> fis erase -l 0x80000 -f 0xc4200000
```

and at the prompt to confirm writing, enter: **y**

8. Load the Linux image without a Ramdisk, enter:

```
RedBoot> fis load linux
```

or load the Linux image with a ramdisk, enter:

```
RedBoot> fis load ramdisk
```

```
RedBoot> fis load linux
```

9. To execute the Linux Kernel image, enter one of the commands below. The first is to mount an NFS share as the root filesystem and the second is to use the Ramdisk image that has been loaded into memory as the root filesystem:

```
RedBoot> exec -c "console=ttyS0,115200 ip=bootp root=nfs"
```

or

```
RedBoot> exec -c "console=ttyS0,115200 ip=off root=/dev/ram initrd=0x2c600000"
```

2.4.4 Store Linux Ramdisk image

1. Activate the **Boot Monitor** as described in Section 2.2.
2. Make sure the Ramdisk image is available on a TFTP server. The image is named `ramdisk` and is available in the NFE tarball.
3. Load the image into the NFE hardware platform RAM (replacing `xyz` with the IP address of the TFTP server and `0x2c600000` with `0x0c600000` for an NFE-i4000 series card):

```
RedBoot> load -r -v -b 0x2c600000 -h xyz ramdisk.1.4.1.gz
```

The following message is displayed:

```
Raw file loaded 0x2c600000-0x2c7d7c92, assumed entry at 0x2c600000
```

To calculate the size of the image, subtract the first value in the message from the second value and add 1:

$$0x2c7d7c92 - 0x2c600000 + 1 = 0x1d7c93.$$

4. Remove any image named `ramdisk` from flash memory the same way as described in the previous section. To remove the Ramdisk image, enter:

```
RedBoot> fis delete ramdisk
```

and at the prompt to confirm writing, enter: **y**

5. Store the new image to flash memory:

```
RedBoot> fis create -b 0x2c600000 -l 0x1d7c93 -e 0x2c600000 -r 0x2c600000 -f 0xc4300000  
ramdisk
```

6. To load the ramdisk image and boot the Linux image that uses it, enter:

```
RedBoot> fis load ramdisk
```

```
RedBoot> fis load linux
```

```
RedBoot> exec -c "console=ttyS0,115200 ip=off root=/dev/ram initrd=0x2c600000"
```

2.4.5 Store Diagnostics image

1. Activate the **Boot Monitor** as described in Section 2.2.
2. Make sure the diagnostics image is available on a TFTP server. The image is named `diag_NFE.bin.1.31.0` and is available in the NFE tarball.
3. Load the image into the NFE hardware platform RAM (replacing **xyz** with the IP address of the TFTP server):

```
RedBoot> load -r -v -m tftp -b 0x280000 -h xyz diag_NFE.bin.1.31.0
```

The following message is displayed:

```
Raw file loaded 0x00280000-0x003056d3, assumed entry at 0x00280000
```

To calculate the size of the image, subtract the first value in the message from the second value and add 1:

$$0x003056d3 - 0x00280000 + 1 = 0x856d4$$

4. Alternatively, the Boot Monitor image may be loaded using the RS-232 serial interface. To use this method, enter:

```
RedBoot> load -r -v -b 0x280000 -m xmodem
```

then send the file `diag_NFE.bin.1.31.0` using the terminal emulator's xmodem send command.

5. Next, find the diagnostics image that is listed in the flash:

```
RedBoot> fis list
```

The output will list images that are stored in flash:

Name	FLASH addr	Mem addr	Length	Entry point
...				
RedBoot	0xC4100000	0xC4100000	0x00040000	0x00000000
linuxdata	0xC4200000	0xC4200000	0x00080000	0xFFFFFFFF
ramdisk	0xC4300000	0x2C600000	0x00300000	0x2C600000
diag	0xC4D00000	0x00280000	0x000A0000	0x00280000
...				

If no `diag` entry is listed, go to step 7.

6. Next the image will be transferred from RAM to flash memory. Enter the following two commands:

```
RedBoot> fis erase -f 0xc4D00000 -l 0x856d4
```

```
RedBoot> fis write -f 0xc4D00000 -b 0x280000 -l 0x856d4
```

and at the prompt to confirm writing, enter: **y**

7. If no `diag` entry was found at step 5, transfer the image from ram to flash memory by creating a new `diag` entry in flash memory. Enter the following command: `RedBoot> fis create -b 0x280000 -l 0x856d4 -e 0x280000 -r 0x280000 -f 0xc4d00000 diag`

2.5 Configuration

The `fconfig` command is used to configure the Boot Monitor. A part of this configuration is the “Boot script” which is basically a set of commands, programmed by the user, and executed by the Boot Monitor after a successful initialization. The purpose mainly being to load another operating system such as Linux.

In the Boot Monitor console, execute `fconfig -l` to view the current configuration. If the current settings need to be modified, execute `fconfig` without any parameters to enter an interactive configuration session.

The first set of three questions configures the Boot script and its operation:

1. If the Boot script must be executed, enter **true** at the `Run script at boot:` prompt
2. Enter sequential commands, each one on it's own line, to be executed by the Boot script. To end the script, enter an empty line. That is, at the “>>”-prompt simply press enter without entering any characters on the line.
3. The boot script timeout can be set to the number of seconds the Boot Monitor must wait after successful initialization before starting the Boot script. During this wait, the user may press Ctrl-C to cancel the execution of the Boot script commands and enter the Boot Monitor console.

The ranges and default values of the remaining Boot Monitor configuration questions are discussed in Table 2.2. Refer to Section 2.6.7 for a description of the `fconfig` command.

Table 2.2. Boot Monitor Configuration Parameters

Full name	Description
Run script at boot	<p>When this option is set to true, a boot script is executed when the Boot Monitor starts. If this option is false, the Boot Monitor will start at the Boot Monitor command line prompt.</p> <p>Default: false</p> <p>Range: True/false</p>
Boot script	<p>The boot script to be executed when the Boot Monitor has loaded, must be entered one line at a time. Leave a blank line to indicate the end of the boot script.</p> <p>Default: blank</p> <p>Range: Any text string</p>
Boot script timeout	<p>The time the Boot Monitor waits before the boot script is executed. During this time the user may cancel the execution of the boot script and return to the Boot Monitor prompt.</p> <p>Default: 0</p> <p>Range: 0 - 100000</p>
Use BOOTP for network configuration	<p>If true, the Boot Monitor will use bootp to obtain an IP address from a DHCP server. Note that this option is unavailable for an NFE-i4000 series card.</p> <p>Default: true</p> <p>Range: True/false</p>
Default server IP address	<p>Specify the address of the TFTP server where program images may be loaded from. Note that this option is unavailable for an NFE-i4000 series card.</p> <p>Default: blank</p> <p>Range: 0.0.0.0 – 255.255.255.255</p>
Console baud rate	<p>Set the baud rate of the RS-232 serial interface.</p> <p>Default: 115200</p> <p>Range: 300 - 115200</p>
GDB connection port	<p>Set the TCP port to send GNU Debugger messages over the ethernet port.</p> <p>Default: 9000</p> <p>Range: 1 - 100000</p>
Force console for special debug messages	<p>Enable console for special debug messages. This is not used.</p> <p>Default: false</p> <p>Range: True/false</p>

Full name	Description
Network debug at boot time	Turn network debugging at boot time on or off. Default: false Range: True/false
Default network device	Specify the default network device. Note that this option is unavailable for an NFE-i4000 series card. Default: i82559_eth0 Range: i82559_eth0

The example below shows a complete Boot Monitor configuration with a Boot script that loads a Linux Kernel and its accompanying Ramdisk image from flash memory into memory and executes it. Please note the empty line following the Boot script “exec”-command.

```
RedBoot> fconfig
Run script at boot: true
Boot script:
Enter script, terminate with empty line
>> fis load ramdisk
>> fis load linux
>> exec -c "console=ttyS0,115200 ip=off root=/dev/ram initrd=0x2c600000"
>>
Boot script timeout (1000ms resolution): 10
Use BOOTP for network configuration: true
Default server IP address: 10.0.0.1
Console baud rate: 115200
GDB connection port: 9000
Force console for special debug messages: false
Network debug at boot time: false
Default network device: i82559_eth0
Update RedBoot non-volatile configuration - continue (y/n)? y
RedBoot> reset
```

To load the kernel from a TFTP server (xyz) and remote mount the file system with NFS, the Boot script must be configured as follows:

```
:
>> load -r -v -m tftp -b 0x00280000 -h xyz kernel_filename
>> exec -c "console=ttyS0,115200 ip=bootp root=nfs"
>>
:
```

2.6 Boot Monitor Command Reference

This section describes the Boot Monitor commands in detail. The commands are summarized in the following table.

Table 2.3. Boot Monitor Commands

Command	Description
alias	Manage aliases kept in Flash memory.
baudrate	Set or query the console's baud rate.
cache	Manage machine caches (data and instruction).
cksum	Show POSIX checksum for a specified range of memory.
dump	Display memory contents.
exec	Execute an image with the MMU off.
fconfig	Manage configuration kept in Flash memory.
fis create	Create an image in the Flash Image System (FIS) directory.
fis delete	Delete an image in the FIS directory.
fis erase	Erase a portion of flash memory.
fis free	Display areas of flash memory that are currently not in use.
fis init	Initialize the FIS.
fis list	List images currently available in the FIS.
fis load	Transfer an image from flash to RAM.
fis write	Write data from memory to flash.
go	Execute code at a specified location.
help	Display help information.
history	Display command history.
ip_address	Set or change the IP address
load	Download programs or data.
ncmp	Compare two blocks of memory.
ncpy	Copy memory from one address to another.
mfill	Fill a block of memory with a pattern.
ping	Verify network connectivity.
reset	Reboot the platform.
version	Display Boot Monitor version.
x	Display a range of memory (hex-dump).

2.6.1 alias

Manage aliases kept in flash memory.

Syntax:

```
alias name [value]
```

Parameters:

name	Name for the alias.
value	Value for the alias (if this parameter is not given, the current value of the alias will be displayed). An alias may represent multiple commands by separating them with a semi-colon(;). Use an escape character(\) to add a quotation mark("").

Example:

```
alias bootnfs "fis load linux; exec -c \"console=ttyS0,115200 ip=bootp root=nfs\""
```

Usage:

```
%{bootnfs}
```

2.6.2 baudrate

Set or query the system console baudrate.

Syntax:

```
baudrate [-b <rate>]
```

Parameters:

b	Set the given console baud rate. The following values are valid: 600, 1200, 2400, 4800, 9600, 14400, 19200 , 38400, 57600, 115200
---	---



Note

If the `-b` parameter is not specified, the command will display the current value.

Example:

```
baudrate -b 115200
```

2.6.3 cache

Manage machine caches (data and instruction).

Syntax:

```
cache [ON | OFF]
```

Parameters:

ON	Command enables the data and instruction caches.
OFF	Command disables the data and instruction caches.

Returns:

0 OK
<0 or >0 Unspecified error.

Example:

cache ON

2.6.4 cksum

Count and show a 32-bit POSIX checksum for a specified range of memory. This command is used to verify and compare large numbers of memory locations.

Syntax:

cksum -b <location> -l <length>

Parameters:

b <location>	Start of the memory range. It must be a four byte aligned address.
l <length>	Length of the memory range. It must be at least four bytes and a number that is divisible by four.

Example:

cksum -b 0xa000000 -l <length>

2.6.5 dump

Display the content of a memory range.

Syntax:

dump -b <location> [-l <length>] [-s] [-1|-2|-4]

Parameters:

b <location>	Start of memory range.
l <length>	Length of memory range (default is 32).
s	Format data using Motorola™ S-records.

1	Access one byte at a time. Only the least 8 significant bits of the pattern will be used. This is the default.
2	Access two bytes at a time. Only the least 16 significant bits of the pattern will be used.
4	Access four bytes at a time. Only the least 32 significant bits of the pattern will be used.

Example:

```
dump -b 0x100000 -2
```

2.6.6 exec

Execute an image with the Memory Management Unit (MMU) disabled.

Syntax:

```
exec [-w <timeout>] [-b <load addr> [-l <length>]] [-r <ramdisk addr> [-s <ramdisk length>]] [-c <"kernel command line">] [-t <target>] [<entry_point>]
```

Parameters:

w <timeout>	Time (in seconds) to wait before execution (default value is 0).
b <load_addr>	Memory address of the image.
l <length>	Length of the image.
r <ramdisk addr>	Memory address of "initrd" style RAMdisk passed to Linux Kernel image.
s <ramdisk length>	Length of RAMdisk image passed to Linux Kernel image.
c <kernel command line>	Command line parameters to pass to image.
t <target>	Final physical address to copy image to.
<entry_point>	Execution starting address for image.

Example:

```
exec -c "console=ttyS0,115200 root=/dev/partition_in_redboot noinitrd init=/bin/boot" -w 1 0x0d008000
```

2.6.7 fconfig

Manage configuration kept in flash memory. When the command is invoked, a prompt appears for each value. The value must be typed in full. Pressing Enter will keep the current value.

Syntax:

```
fconfig [-i] [-l] [-n] [-f] [-d] nickname [value]
```

Parameters:

i	Reset configuration database to default state.
l	List current configuration without changing the values.
n	Display parameter “nicknames” rather than the full names.
f	Display full parameter names.
d	Edit when backspace cannot be used.
nickname <value>	Change or display only this parameter.



Note

When executing `fconfig` without the `-d` option, backspace over the displayed value and enter a new value.

Example:

```
fconfig -l
```

2.6.8 fis create

Create an image in the FIS directory. The image must be loaded into RAM before it is copied to flash memory. Use the `load` command to load the file into RAM.

Syntax:

```
fis create -b <data_address> -l <length> [-f <flash_address>] [-e <entry>] [-r  
<relocation_address>] [-s <data_length>] [-n] name
```

Parameters:

b <data_address>	Memory address of image to be written to flash.
l <length>	Length of flash area to be occupied (this value may be calculated based on the load command).
f <flash_address>	Flash memory address of area to be occupied (default value is the first available block that is large enough).
e <entry>	Entry address for executable image.
r <relocation_address>	Memory address where image should be relocated to by the load command.
s <data_length>	Actual length of image.
n	No image data written to flash, only the FIS directory is updated.

name	Name of flash image.
------	----------------------

Example:

```
fis create -b 0x280000 -l 0x1e0000 -f 0x90020000 monitor
```

2.6.9 fis delete

Remove an image from the FIS directory. This image will be erased and the name will be removed from the FIS directory.

Syntax:

```
fis delete name
```

Parameters:

name	Name of flash image to be deleted.
------	------------------------------------

Example:

```
fis delete diag
```

2.6.10 fis erase

Erase a portion of the flash memory.

Syntax:

```
fis erase -f <flash_address> -l <length>
```

Parameters:

f <flash_address>	Starting flash memory address.
l <length>	Length of flash area to be erased.

Example:

```
fis erase -f 0x90500000 -l 0x100000
```

2.6.11 fis free

Display areas of flash memory that are currently not in use. A block containing non-erased content is considered in use.

Syntax:

```
fis free
```

2.6.12 fis init

Initialize the FIS.



Warning

Execution of this command will cause the loss of information stored in the Flash Image System (FIS).

Syntax:

```
fis init [-f]
```

Parameters:

f	All blocks of flash memory (except for the boot blocks) will be erased as part of the initialization procedure.
---	---

Example:

```
fis init
```

2.6.13 fis list

List images currently available in the Flash Image System (FIS).

Syntax:

```
fis list [-c] [-d]
```

Parameters:

c	Show image checksum instead of memory address.
d	Show image data length instead of amount of flash occupied by image.

Example:

```
fis list -d
```

2.6.14 fis load

Transfer an image from Flash Image System (FIS) to RAM. After the image has been loaded, it can be executed using the go command.

Syntax:

```
fis load [-b <load_address>] [-c] name
```

Parameters:

b <load_address>	Memory address where the image should be loaded (default value is the address associated with the image).
c	Compute and print the checksum of the image after it has been loaded.
name	The name of the file as show in the FIS directory.

Example:

```
fis load -b 0x280000 diag
```

2.6.15 fis write

Write data from memory to Flash Image System (FIS).

Syntax:

```
fis write -b <data_address> -l <length> -f <flash_address>
```

Parameters:

b <data_address>	Memory address of data to be written to flash.
l <length>	Length of flash area to be occupied (this value may be calculated based on the load command).
f <flash_address>	Address of flash area to be occupied (default - first available block that is large enough to store the image).

Example:

```
fis write -b 0x280000 -l 0x100000 -f 0x90200000
```

2.6.16 go

Execute code at a specified RAM location.

Syntax:

```
go [-w <timeout>] [-c] [-n] [entry]
```

Parameters:

w <timeout>	Executes code after a given period of time in seconds.
c	Executes code with cache enabled.
n	Executes code with network driver stopped.
entry	Memory address of code to execute.

Example:

```
go 0x400000
```

2.6.17 help

Display help on available commands.

Syntax:

```
help <topic>
```

Parameters:

<topic>	Command to display help for.
---------	------------------------------

Example:

```
help go
```

2.6.18 history

Display command history.

Syntax:

```
history
```

Parameters:

None.

Example:

history

2.6.19 ip_address

Set or change the IP address.

Syntax:

ip_address [-b] [-l <local_ip_address>[/<mask_len>]] [-h <server_address>]

Parameters:

b	Display BOOTP information.
h <server_address>	Set default server IP address.
l <local_ip_address>[/<mask_len>]]	Set the local IP address.

Example:

ip_address -l 10.1.1.231/8

2.6.20 load

Download applications or data to the NFE.

Syntax:

load [-v] [-r] [-m <protocol>] [-h <server_IP_address>] [-p <TCP Port>] [-b <location>]
[file_name]

Parameters:

v	Display a small “spinner” while load is in progress.
r	Raw (or binary) data.
m	Protocol used for transfer: <ul style="list-style-type: none"> • tftp • xmodem

	• ymodem
h <server_IP_address>	IP address of host with TFTP service as well as requested data.
p <TCP_port>	Alternate TCP port to use.
b <location>	Memory address to load data into.
<file_name>	Name of the file to be transferred.

Example:

```
load -r -m xmodem -b 0x400000
```

2.6.21 mcmp

Compare two blocks of memory.

Syntax:

```
mcmp -s <location> -d <location> -l <length> [-1 | -2 | -4]
```

Parameters:

s <location>	Start memory address of first block.
d <location>	Start memory address of second block.
l <length>	Number of bytes in block to compare.
[1 2 4]	Compare Bytes, Words or Double words (default).

Returns:

Nothing if the two blocks match.

Example:

```
mcmp -s 0xa0000000 -d 0xa0000040 -l 64
```

2.6.22 mcopy

Copy memory from one address to another.

Syntax:

```
mcopy -s <addr> -d <addr> -l <length> [-1 | -2 | -4]
```

Parameters:

s <location>	Start memory address of source block.
d <location>	Start memory address of destination block.
l <length>	Number of bytes in bock to copy.
[1 2 4]	Copy Bytes, Words or Double words (default).

Example:

```
mcopy -s 0xa0000000 -d 0xa0000040 -l 64
```

2.6.23 mfill

Fill a block of memory with a pattern.

Syntax:

```
mfill -b <addr> -l <length> [-p <pattern>] [-1 | -2 | -4]
```

Parameters:

b <address>	Start memory address.
l <length>	Number of bytes to fill (in decimal).
p <pattern>	Fill pattern.
[1 2 4]	Fill format Bytes, Words or Double words (default).

Example:

```
mfill -b 0xa0000000 -l 64 -p 0xabcddefab
```

2.6.24 ping

Verify network connectivity.

Syntax:

```
ping [-v] [-i <local_IP_address>] [-l <length>] [-n <count>] [-t <timeout>] [-r <rate>]
-h <server_IP_address>
```

Parameters:

v	Verbose mode, display information on every packet.
i <local_IP_address>	IP address the Boot Monitor should use. The default value is set by the fconfig command or obtained using BOOTP protocol.
l <length>	Length of the ICMP data payload.

n <count>	Number of packets to be sent.
t <timeout>	How long to wait for the round-trip to complete (in milliseconds).
r <rate>	Time between successive sends (in milliseconds).
h <server_IP_address>	IP address of destination host.

Example:

```
ping -h 10.1.0.100 -n 100
```

2.6.25 reset

Reboot the NFE platform.

Syntax:

```
reset
```

Parameters:

None.

2.6.26 version

Display Boot Monitor version information.

Syntax:

```
version
```

Parameters:

None.

2.6.27 x

Display a range of memory (hex-dump).

Syntax:

```
x -b <location> [-l <length>] [-s] [-1 | -2 | -4]
```

Parameters:

b <location>	Memory address to display.
l <length>	Number of bytes to display in decimal. Default 32 bytes.
s	Format data using Motorola™ S-records.
[1 2 4]	Group output as Bytes e.g FF, Words e.g 12FF or Doublewords e.g 1234ABCD.

Example:

```
x -b 0xa0000000 -l 64
```

3. Diagnostics

3.1 Activating the Diagnostic Application

The NFE hardware platform's diagnostic application is factory programmed into the hardware platform's flash memory together with the Boot Monitor and Recovery Loader. A list of all the applications programmed into the flash memory can be viewed by typing the following command at the Boot Monitor prompt (“RedBoot>”):

```
RedBoot> fis list
```

The diagnostic application is typically saved using the name `diag`. It provides a suite of utilities to test all the hardware components of the NFE. To execute the diagnostic application (or any other application on the flash memory) always follow these steps:

1. Load the application from flash memory into RAM using the following Boot Monitor command:

```
RedBoot> fis load <application_name>
```

Normally one would type:

```
RedBoot> fis load diag
```

2. Execute the application with the following Boot Monitor command:

```
RedBoot> go
```

Diagnostic commands are entered at the diagnostic application's prompt (“DIAG>”). The command `exit` causes the diagnostic application to exit and return to the Boot Monitor environment.

3.2 Diagnostic Command Reference

This section describes the diagnostic commands in detail. The commands are summarized in the following table.

Table 3.1. Diagnostic Commands

Command	Description
? or h or help	Display diagnostic commands help information.
banner	Print hardware and software information.
blockcopy	Copy memory block.
c	Change the contents at a memory address.
doff	Disable data cache.
don	Create page table, enable data cache and enable writeback.
exit	Exit the diagnostic application.
fill	Fill memory with pattern.
i2cread	Read from an I ² C device and display results.
i2cwrite	Write to an I ² C device.
ioff	Disable instruction cache.
ion	Enable instruction cache.
l[og]	Log management.
p	Display contents of memory.
pcicfg	Read/write data from/to a PCI device's configuration space.
pciio	Read/write data from/to PCI I/O space.
pcimem	Read/write from/to PCI memory space.
reset	Reset NFE hardware platform.
r and w	General read/write memory access.
sethoe	Set the halt on error value.
setverbose	Set the verbose output level for all diagnostic commands.
sramdeskew	Find optimal SRAM channel deskew and DLL settings.
t or test	Display a list of the available test commands within the diagnostic image.
tf	Enable or disable test flags.
t[est] all	Test the system with tests that are enabled by test flags.
t[est] ee	Access EEPROM data.
t[est] ethpci	Test 82559ER management Ethernet.
t[est] media	Configure test media for multiple ports.
t[est] generate	Tests frame transmit and receive with media usage.
t[est] gpio	Perform GPIO tests.
t[est] ixcp	Tests IXCP210 asymmetric cryptographic processor.
t[est] led	Perform LED tests.
t[est] mem	Perform memory tests.
t[est] msf	Perform MSF tests.

Command	Description
t[est] pci	Perform PCI tests.
t[est] pcie	Perform PCI Express functionality test.
t[est] slowport	Perform slowport tests.
t[est] temp	Perform temperature sensor test.
t[est] trng	Test the true random number generator.
t[est] uart	Perform UART tests.
t[est] ueng	Perform a microengine test.
t[est] xscale	Perform Intel XScale® scratchpad and core component tests.
wrv	Write-Read-Verify memory.

3.2.1 ?, h, help

Print the help screen for a given diagnostic command.

Syntax:

```
? [command] h [command] help [command]
```

Parameters:

command	Specific command to display help screen for.
---------	--



Note

The command name parameter is optional – if omitted, help is displayed for all diagnostic commands.

Example:

```
? fill
```

3.2.2 banner

Print a banner containing basic hardware and software information.

Syntax:

```
banner
```

3.2.3 blockcopy

Copy a memory block.

Syntax:

```
blockcopy <low> <high> <dest>
```

Parameters:

low	Start memory address of block to copy.
high	End memory address of block to copy.
dest	Destination memory address of the block.

Example:

```
blockcopy A0000000 A000003f A0000050
```

3.2.4 c

Interactively change the contents of a specified memory address. This command allows the value of a 32-bit(d), 16-bit(w) or 8-bit(b) memory address to be changed. That is, once the command and the memory address has been entered, the command will prompt for the new value. To exit without entering a value (current value unchanged), press the <ESC> key.

The following options are available in the address editing interface:

- ?: Display this help text
- Space: Re-read memory location
- Enter: Write value or move to next location
- ESC: Leave change mode
- Backspace: Delete last digit or re-enter last entered char
- Control-U: Delete line (clear value)
- + or -: Move to prev / next location
- ' or ": Enable/disable char entry mode
- ~: Enable/disable bit toggling mode
- \$ or x: Select hex or dec mode
- When in bit toggling mode: Enter bit number (2 decimal digits) to toggle
- When in decimal mode: Press 0-9 to enter digits
- When in hex mode: Press 0-9 or A-F to enter digits



Note

All memory addresses are entered in hexadecimal format without the *0x* prefix.

Syntax:

```
c<bwd> <addr> c <addr> cb <addr> cw <addr> cd <addr>
```

Parameters:

addr	Memory address to change.
------	---------------------------

Example:

```
c 1000000
```

3.2.5 doff

Disable data cache.

Syntax:

```
doff
```

Parameters:

None.

3.2.6 don

Create page table, enable data cache and enable writeback.

Syntax:

```
don
```

Parameters:

None.

3.2.7 exit

Exit the diagnostic application.

Syntax:

```
exit
```

Parameters:

None.

3.2.8 fill

Fill a range of memory with a user specified quadword.

Syntax:

```
fill <low> <high> <val>
```

Parameters:

low	Start address of memory range to fill.
high	End address of memory range to fill.
val	Quad word value used to fill each memory location of the memory range.

Example:

```
fill A0000000 A0000020 BEEFF00D
```

3.2.9 h

Print the help screen for a given diagnostic command. Same as help.

Syntax:

```
h [command]
```

Parameters:

command	Specific command to display help.
---------	-----------------------------------

Example:

```
h fill
```

3.2.10 help

Print the help screen for a given diagnostic command.

Syntax:

```
help [command]
```

Parameters:

command	Specific command to display help.
---------	-----------------------------------

Example:

```
help fill
```

3.2.11 i2cread

Read from an I²C device and display the results.



Note

All memory addresses are entered in hexadecimal format without the *0x* prefix.

Syntax:

```
i2cread <slave_addr> <offset_addr> <count>
```

Parameters:

slave_addr	Address of the I ² C device on the bus.
offset_addr	Memory address inside the I ² C device.
count	Number of bytes read (decimal).

Example:

```
i2cread 1 0 10
```

3.2.12 i2cwrite

Write string data into the memory of an I²C device.



Note

All memory addresses are entered in hexadecimal format without the *0x* prefix.

Syntax:

```
i2cwrite <slave_addr> <offset_addr> <data>
```

Parameters:

slave_addr	Address of the I ² C device on the bus.
offset_addr	Memory address inside the I ² C device.
data	Hexadecimal string to write to I ² C device memory.

Example:

```
i2write 4 30 123456789abcdef123456
```

3.2.13 ioff

Disable instruction cache.

Syntax:

```
ioff
```

Parameters:

None.

3.2.14 ion

Enable instruction cache.

Syntax:

```
ion
```

Parameters:

None.

3.2.15 log

Log management.

Syntax:

```
log <command> [eventID]
```

Parameters:

command	String to specify one of the following: <ul style="list-style-type: none"> • c clear event log • d dump event log • dp dump not passed events • df dump failed events • da dump aborted events • ds dump skipped events • di dump invalid events • h this help • f fill event log eventID = RREELLPP • r read selected event • rn read next event
eventID	Selected event ID.

Example:

```
log r 0
```

3.2.16 p

Display the contents of memory that starts at address `addr` and is `size` bytes long.



Note

All memory addresses are entered in hexadecimal format without the `0x` prefix.

Syntax:

```
p<bwd> <addr> [size] p <addr> [size] pb <addr> [size] pw <addr> [size] pd <addr> [size]
```

Parameters:

addr	Starting address of a memory block in hexadecimal.
size	Number of bytes to display in hexadecimal.

Example:

```
p 800000 80
```

3.2.17 pcicfg

Read (r) or write (w) a byte (b), word (w) or dword (d) from/to a PCI device.



Note

All memory addresses are entered in hexadecimal format without the *0x* prefix.

Syntax:

```
pcicfg[rw][bwd] pcicfgrb <bus> <slot> <addr> pcicfgwb <bus> <slot> <addr> <data> pcicfgrw  
<bus> <slot> <addr> pcicfgww <bus> <slot> <addr> <data> pcicfgrd <bus> <slot> <addr>  
pcicfgwd <bus> <slot> <addr> <data>
```

Parameters:

bus	PCI bus number of the device.
slot	PCI slot number of the device.
addr	Memory address that will be written to or read from.
data	Data to write.

Example:

```
pcicfgrb 1 0 0
```

3.2.18 pciio

Read (r) or write (w) a byte (b), word (w) or dword (d) from/to a PCI IO.



Note

All memory addresses are entered in hexadecimal format without the *0x* prefix.

Syntax:

```
pciio[rw][bwd] pciiorb <addr> <rept> <silent> pciiowb <addr> <data> <rept> pciiorw <addr>
<rept> <silent> pciioww <addr> <data> <rept> pciiorw <addr> <rept> <silent> pciiowd <addr>
<data> <rept>
```

Parameters:

addr	Memory address that will be written to or read from.
rept	Repeat this on a loop. Can be set as follows: <ul style="list-style-type: none"> • 0 perform a read or write one time only. • 1 perform a read or write on a loop until a key is pressed.
data	Data to write.
silent	Do not trace any information on the console: <ul style="list-style-type: none"> • 0 display information on the console. • 1 run in silent mode.

Example:

```
pciiorb 1 0 0
```

3.2.19 pcimem

Read (r) or write (w) a byte (b), word (w) or dword (d) from/to memory.

Syntax:

```
pcimem[rw][bwd] pcimemrb <addr> <rept> <silent> pcimemwb <addr> <data> <rept> <silent>
pcimemrw <addr> <rept> <silent> pcimemww <addr> <data> <rept> <silent> pcimemrd <addr>
<rept> <silent> pcimemwd <addr> <data> <rept> <silent>
```

Parameters:

addr	Memory address that will be written to/read from.
rept	Repeat this on a loop. Can be set as follows: <ul style="list-style-type: none"> • 0 perform a read or write one time only. • 1 perform a read or write on a loop until a key is pressed.
data	Data to write to the memory address.
silent	Do not trace any information on the console: <ul style="list-style-type: none"> • 0 display information on the console. • 1 run in silent mode.

Example:

3.2.20 reset

Reset NFE hardware platform.

Syntax:

```
reset
```

Parameters:

None.

3.2.21 r

General read of a byte (b), word (w) or dword (d).

Syntax:

```
r[bwd] <addr> [repeatcount] [verbose] rb <addr> [repeatcount] [verbose] rw <addr>
[repeatcount] [verbose] rd <addr> [repeatcount] [verbose]
```

Parameters:

addr	Memory address of a word to read in hexadecimal.
repeatcount	Number of times (in decimal) the word should be read.
verbose mode	Can be set to any of the following: <ul style="list-style-type: none"> • 0 verbose • 1 silent

Example:

```
rw 80000000 1 0
```

3.2.22 sethoe

Set the halt on error value.

Syntax:

```
sethoe <val>
```

Parameters:

val	Halt on error value.
-----	----------------------

Example:

```
sethoe 3
```

3.2.23 setverbose

Set the verbose output level for all diagnostic commands.

Syntax:

```
setverbose <level>
```

Parameters:

level	Verbose level. Can be set to any of the following: <ul style="list-style-type: none">• 0 disable all trace messages.• 1 enable ERROR trace messages.• 2 enable WARNING trace messages.• 3 enable INFO trace messages.• 4 enable EVENT and PROGRESS trace messages.• 5 enable DEBUG_0 trace messages.
-------	---

Example:

```
setverbose 3
```

3.2.24 sramdeskew

Find optimal DLL and deskew values.

Syntax:

Setting DLL/deskew:

```
sramdeskew s <channel> <DLL> <deskew>
```

where the parameters DLL and deskew are decimal values.

Testing DLL/deskew:

Example:

```
test
```

3.2.26 tf

Enable test flags. Causes given device to be included or excluded from tests in the `test all` command.

Syntax:

```
tf <test_type_ID> <status>
```

Parameters:

test_type_id	<p>Test identifier. Can be any of the following:</p> <ul style="list-style-type: none"> • 1 LED • 2 GPIO • 3 MSF • 4 SLOWPORT • 5 UENG (microengine) • 6 XSCALE • 7 UART • 8 MEDIA • 9 MEM • A ETHPCI • B PCI • C EE • D TRNG
status	<p>Test flag. Can be any of the following:</p> <ul style="list-style-type: none"> • 0 do not perform test. • 1 perform test.

Example:

```
tf 4 1
```

3.2.27 t all

Test the system with tests that are enabled by test flags.

Syntax:

```
t all
```

3.2.28 t ee

I2C EEPROM test.

Syntax:

```
t ee <option> t ee r <offset> [count]
```

Parameters:

option	Indicates the test to perform. Can be set as follows: <ul style="list-style-type: none"> • v EEPROM version information. • d EEPROM database display. • r read and display contents of EEPROM. • c EEPROM checksum validation.
offset	Offset address from start of EEPROM in hexadecimal.
count	The number of bytes in hexadecimal to display.

Example:

```
t ee r 0 7f
```

3.2.29 t ethpci

Test the management/debug Ethernet interface.

Syntax:

```
t ethpci <option>
```

Parameters:

option	<p>Indicates the test to perform. Can be set as follows:</p> <ul style="list-style-type: none"> • h this message. • p presence test. • r read contents of EEPROM. • t backup EEPROM, test write/read of entire EEPROM, then restore EEPROM. • w read in formatted data from stdin, then write to EEPROM • e erase EEPROM (all words to 0xFFFF).
--------	---

Example:

```
t ethpci r
```

3.2.30 t media

Perform an on-board IXF1110 Media Access Controller(MAC) and on-board PHY interface test. The test performs IXF1110 and PHY initialization and starts receive and transmit microBlocks (uBlocks). The uBlocks send and receive frames at wire speed. The uBlocks send frames with its checksum, making it possible to receive frames on any interface or by another board running the same test. The system loopback can be activated so that frames received on any interface can be put back on the front of the backplane. Fiber and copper system loopback are both available.

Syntax:

```
t media <mode> <test> [ports] [speed]
```

Parameters:

mode	<p>Test mode. Can be set as follows:</p> <ul style="list-style-type: none"> • b Base board mode. • d Disable tests and reset media configuration for all modes • c Configuration trace mode (port config and link states). • h Help. • hm Help (mode only). • ht Help (test only). • hp Help (ports only). • hs Help (speed and type only).
test	<p>Test type to perform. Can be set as follows:</p> <ul style="list-style-type: none"> • r All MAC and PHY registers test. • rm All MAC registers test.

	<ul style="list-style-type: none"> • rmd Default MAC registers test. • rmu Update MAC registers test. • rp All PHY registers test. • rpd Default PHY registers test. • rpu Update PHY registers test. • c Basic MAC counter show (selected counters only). • cf Full MAC counter show. • ct Transmit MAC counter show. • cr Receive MAC counter show. • cs Special MAC counter show.
ports	<p>Comma separated list of ports to be tested. Can be set as follows:</p> <ul style="list-style-type: none"> • BASE<#port>, e.g BASE1 • BASE All base ports.
speed	<p>Port speed. Can be set as follows:</p> <ul style="list-style-type: none"> • 0 Auto negotiation of speed • 100 100 Mhz • 1000 1 GHz.

Example:

```
t media b r BASE1,BASE2 1000
```

3.2.31 t generate

Start the traffic generator based on configuration set by the t media command. The test starts receive and transmit microblocks. The microblocks send and receive frames with wire speed. The microblocks send frames with checksum that makes possible to receive frame on any interface or even by another board running the same test.

Syntax:

```
t generate <mode> <test> [sizeMin] [sizeMax] [pattern] [burst] [frame]
```

Parameters:

mode	<p>Test mode. Can be set as follows:</p> <ul style="list-style-type: none"> • e Enable frame test (training disabled) • et Enable frame test (training enabled) • c Enable frame test (CSIX frame on any HW type)
------	--

	<ul style="list-style-type: none"> • ct Enable frame test (CSIX frame on any HW type) • d Disable frame test • l Enable loopback test (training disabled) • lt Enable loopback test (training enabled) • p Enable CSIX performance test (training disabled) • pt Enable CSIX performance test (training enabled) • s Show frame test • h Help • hm Help (mode only) • ht Help (test only) • hp Help (frame params only)
test	<p>Test type to perform. Can be set as follows:</p> <ul style="list-style-type: none"> • s Selected frame size test, default pattern • i Incremental frame size test, default pattern • d Decremental frame size test, default pattern • r Random frame size test, default pattern • *s * frame size test, selected pattern (default) • *i * frame size test, incremental byte pattern • *ib * frame size test, incremental byte pattern • *iw * frame size test, incremental word pattern • *id * frame size test, incremental dword pattern • *d * frame size test, decremental byte pattern • *db * frame size test, decremental byte pattern • *dw * frame size test, decremental word pattern • *dd * frame size test, decremental dword pattern • *r Selected frame size test, default pattern • c ME Counter show (selected counters only) • cf Full ME counter show • ct Transmit ME counter show • cr Receive ME counter show • co Other ME counter show • cn Clear ME counter
sizeMin/mapDev0	<ul style="list-style-type: none"> • Minimal frame size value

	<ul style="list-style-type: none"> • Port mapping on Device 0.
sizeMax/mapDev1	<ul style="list-style-type: none"> • Maximal frame size value • Port mapping on Device 0.
pattern/mapDev2	<ul style="list-style-type: none"> • Pattern frame fill value • Port mapping on Device 0.
burst	32 bits of transmit burst counter.
frame	32 bits of transmit frame counter.

Example:

```
t generate e ii 40 600 0 1 5c1
```

3.2.32 t gpio

Perform GPIO tests that check presence, registers, edge detection and level detection.

Syntax:

```
t gpio <option> [loop]
```

Parameters:

option	<p>Indicates the test to perform. Can be set as follows:</p> <ul style="list-style-type: none"> • d GPIO detection test • r GPIO register test • e GPIO edge detection test • l GPIO level detection test • a all GPIO tests • h help
loop	Specifies the number of times the test is performed (default is 1).

Example:

```
t gpio l
```

3.2.33 t ixcp

IXCP 210 asymmetric cryptographic processor tests.

Syntax:

t ixcp <option>

Parameters:

<code>option</code>	<p>Indicates the test to perform. Can be set as follows:</p> <ul style="list-style-type: none"> • i ixcp 210 initialization and seed RNG. • r ixcp 210 presence test. • p ixcp 210 CSR Register display. • s ixcp 210 decypher encrypted message.
---------------------	---

Example:

t ixcp s

3.2.34 t led

Test the Light Emitting Diode (LED) indicators and displays mounted on the NFE hardware platform. The installed indicators and displays may vary according to the revision of the NFE hardware platform.

Syntax:

t led

Parameters:

None.

3.2.35 t mem

Perform memory tests. The tests covers all types of memory and various test methods such as walking ones or walking zeros.

Syntax:

t mem <option> <test_option> <start_addr> <size> <data> <loops> <verbose>

Parameters:

<code>option</code>	<p>Indicates the memory type to test. Can be set as follows:</p> <ul style="list-style-type: none"> • s0 SRAM memory test channel 0 only • s2 SRAM memory test channel 2 only
---------------------	---

	<ul style="list-style-type: none"> • s3 SRAM memory test channel 3 only • d DRAM memory test • a all memory types • h help
test_option	<p>Sets the test method to use. Can be set as follows:</p> <ul style="list-style-type: none"> • w1 walking one test • w0 walking zero test • w both walking tests • c5 checker 0x5A5A5A5A test • ca checker 0xA5A5A5A5 test • c both checker tests • u unique test • b address bus test • p pattern test (a value for data is required) • i incremental test • e ECC test (only applicable to DRAM) • a all tests
start_addr	Memory address where the test should begin.
size	Number of bytes to test.
data	Data used for pattern test (mandatory for the pattern (p) test).
loops	Specifies the number of times a test is performed (default value is 1).
verbose	Sets the verbose level for the test. This setting is only valid during the initial execution of a looped command.

Example:

```
t mem s0 w 0 1000
```

3.2.36 t msf

Perform Media Switch Fabric(MSF) tests.

Syntax:

```
t msf <option> [loop]
```

Parameters:

option	Indicates the test to perform. Can be set as follows: <ul style="list-style-type: none"> • d MSF register default test • r MSF register read/write test • a all MSF tests • h help
loop	Number of times to repeat the test, default 1.

Example:

```
t msf r 2
```

3.2.37 t pci

Perform PCI tests. Test if all requested PCI devices are present and PCI interrupts can be generated.

Syntax:

```
t pci <option>
```

Parameters:

option	Indicates the test to perform. Can be set as follows: <ul style="list-style-type: none"> • p PCI presence test • a all PCI tests • h help
--------	--

Example:

```
t pci p
```

3.2.38 t pcie

Test PCI Express functionality.

Syntax:

Variants:

- t pcie tlp-send <addr> <idx0> [idx1] Send the contents of a tx-buffer to a specified PCIe address.
- t pcie tx-load <idx> <val0> [val1] ... Load a tx-buffer with dwords.
- t pcie tx-copy-from-sram <idx> <sram-addr> Copy a 64 byte block from sram to tx-buffer.

- `t pcie tx-load-pattern <idx> <pattern>` Load tx-buffer with a specified pattern.
- `t pcie read-to-sram <reg-addr> <dwords> <sram-addr>` Burst-mode generic read of dwords from CSR to sram.
- `t pcie write-from-sram <reg-addr> <dwords> <sram-addr>` Burst-mode generic write of dwords from sram to CSR register.
- `t pcie load-sram <pattern> <dwords> <sram-addr>` Load sram address with data according to pattern.
- `t pcie enable` Enable PCIe interface.
- `t pcie disable` Disable PCIe interface.
- `t pcie rx-buffer <buf-idx0> [buf-idx1]...` Memory dump of rx-buffers.
- `t pcie tlp-get` Display received tlp.
- `t pcie send <addr> <val0> [val1]...` Send a double word to a PCIe address using XScale.
- `t pcie recv` Read first DW of rx-buffer.
- `t pcie xssend <tlp0> [tlp1]...` Write specified values to XS_SEND CSR.

Parameters:

- `reg-addr` hex CSR address from 0000-5000
- `dwords` number of dwords to write to or read from sram
- `sram-addr` sram memory address
- `buf-idx0`, `buf-idx1` zero-based decimal values or ranges, e.g. 0 5 7-9 30-63
- `idx0`, `idx1`, `idx2`, `idx3` zero-based hex index of buffer
- `idx` zero-based hex index of buffer
- `pattern` code denoting a pattern - one of the following values:
 - 1: C6 93 F0 5A C6 93 F0 5A C6 93 F0 5A ...
 - 2: FF FF FF FF ...
 - 3: 00 00 00 00 ...
 - 4: F0 F0 F0 F0 ...
 - 5: 0F 0F 0F 0F ...
 - 6: 00 01 02 03 04 05 06 07 08 09 0a 0b 0c ...
 - 7: 'X' 'X' 'X' 'X' ...

Example:

```
t pcie tlp-send 2bc00000 0
```

3.2.39 t slowport

Perform slowport tests.

Syntax:

```
t slowport <option> [loop]
```

Parameters:

option	Indicates the test to perform. Can be set as follows: <ul style="list-style-type: none"> • d slowport register default test • r slowport register write-read test • a all slowport tests • h help
loop	Specifies the number of times the test is performed (default is 1).

Example:

```
t slowport d
```

3.2.40 t temp

Perform network processor temperature tests.

Syntax:

```
t temp <option> t temp v <level> t temp t <threshold>
```

Parameters:

option	Indicates the test to perform. Can be set as follows: <ul style="list-style-type: none"> • s status: Displays current min/max/current/threshold temperatures. • v set verbose level • t set temperature threshold level • h help message
level	0 to 4. default 0
threshold	Temperature in degrees C. Default 85.

Example:

```
t temp s t temp t 80
```

3.2.41 t trng

Test the True Random Number Generator (TRNG).

Syntax:

```
t trng
```

Parameters:

None.

Example:

```
t trng
```

3.2.42 t uart

Perform UART tests. User intervention is required to finish the test.

Syntax:

```
t uart <command>
```

Parameters:

command	Indicates a test method to perform. Can be set as follows: <ul style="list-style-type: none">• d dump all UART readable registers• r UART register test• a all UART tests• h help
---------	--

Example:

```
t uart r
```

3.2.43 t ueng

Perform microengine (uEngine) tests.

Syntax:

t ueng <option> [microengine] [loop] or t ueng q <channel> [microengine] [loop]

Parameters:

option	Indicates a test method to perform. Can be set as follows: <ul style="list-style-type: none"> • m MSF register test • r register test • s control store test • t timers and counters test • a all microengine tests • h help
q <channel>	QDR access stress test (valid channel numbers are 0, 2, and 3).
microengine	Specifies the microengine number on which the test(s) should be run. Valid numbers are 0 to 7 or 16 to 23. Specifying 99 will run the test on all microengines.
loop	Specifies how many times the test(s) should be performed (default is 1).

Example:

t ueng c 1

3.2.44 t xscale

Perform XScale scratchpad and core component tests. The tests cover IRQs and internal scratch memory tests.

Syntax:

t xscale <option> [loop]

Parameters:

option	Indicates a test option to perform. Can be set as follows: <ul style="list-style-type: none"> • i XScale interrupt test • r XScale scratchpad ring test • s XScale scratchpad pattern test • a all tests • h help
loop	Specifies how many times the test is performed (default is 1).

Example:

t xscale s 1

3.2.45 w

General write of a byte (b), word (w) or dword (d).

Syntax:

```
w[bwd] <addr> <data> [repeatcount] [verbose] wb <addr> <data> [repeatcount] [verbose] ww
<addr> <data> [repeatcount] [verbose] wd <addr> <data> [repeatcount] [verbose]
```

Parameters:

addr	Memory address of a word to read in hexadecimal.
data	Data to write to memory address in hexadecimal.
repeatcount	Number of times (in decimal) the word should be read.
verbose mode	Can be set to any of the following: <ul style="list-style-type: none"> • 0 verbose • 1 silent

Example:

```
ww 80000000 ffff 1 0
```

3.2.46 wrv

Write-Read-Verify memory.

Syntax:

```
wrv <location> <size>
```

Parameters:

location	Memory address in hex.
size	Number of double words (DW) in decimal.

Example:

```
wrv A0000000 16
```

4. Technical Support

To obtain additional information, or to provide feedback, please email <support@netronome.com> or contact the nearest **Netronome Systems** technical support representative.