

# Dansk Data Elektronik A/S

## SMES Diagnostic Programs

User's Manual  
Version 1.0

©Copyright 1995  
Dansk Data Elektronik A/S

**©1995 Dansk Data Elektronik A/S, Denmark**  
**All Rights Reserved**  
**Printed in Denmark**

Version 1.0. Revised and published August 1995

**Stock no.:**

SMES Diagnostic Programs, User's manual: 94400401

SMES Diagnostic Programs, Software: 40030401

**NOTICE**

The information in this document is subject to change without notice. Dansk Data Elektronik A/S, Denmark assumes no responsibility for any errors that may appear in this document.

Supermax is a registered trademark of Dansk Data Elektronik A/S, Denmark.

# SMES Diagnostic Programs

---

<b>Introduction</b>	1
General Description	1-1

---

<b>User interface</b>	2
Description of Special Keys	2-1
Description of Command Interpreter	2-2
Display Modes	2-3
Test and Error Log	2-4

---

<b>baio30x Module</b>	11
Hardware Description	11-1
LEDs Description	11-2
baio30x Commands	11-3

---

<b>cpu30x Module</b>	12
Hardware Description	12-1
LEDs Description	12-2
cpu30x Commands	12-3

**Table of Contents**

*This page is intentionally left blank*

# 1 INTRODUCTION

---

## General Description

Console port	1-1-1
Starting-up	1-1-3
How to Start a Complete Test on All Modules	1-1-4
How to Exit the Diagnostic Programs	1-1-5

**Table of Contents**



# Figures and Tables

---

<b>Figure 1-1-1</b>	Console terminal connected to an SMES system	1-1-2
<b>Figure 1-1-2</b>	Transcript of booting SMES with the Diagnostic Programs	1-1-4

**Table of Contents**

*This page is intentionally left blank*



## General Description

The Diagnostic Programs run separately in each intelligent unit. Consequently it is possible to start programs on all units controlled by *console terminal*.

The main purpose of the Diagnostic Program is to locate a failing unit. If any errors are found the failing unit should be removed from the system and be repaired. It is not the purpose of the Diagnostic Program to specifically point out where on the unit the error was found. Under certain circumstances though, it is possible to point out the failing hardware block by specifying options to a command to state how the hardware should be tested.

All intelligent units in a Supermax Enterprise Server (SMES) are connected to the *global bus*.

### Console port

The *console terminal* must be connected to the *RS232 port 0* on the *baio30x* module with the highest slot (position) number, this unit is called *master unit*. A modem can also be connected to the *console port* instead of the terminal.

It is possible to select which emulation the terminal must use to communicate with the *console port*. You can select two different emulation types, a *VT100* family or a dumb terminal emulation. The *VT100* is default after start-up. Please refer to the *set* command in the "Chapter 11-3 Baio30x Command" to see the detailed description of setting the terminal emulation type. The *console port* is initialized to following:

- 9600 baud.
- 8 data bit.
- 1 stop bit.
- No parity.

The initializing can not be changed.

The *master unit* takes care of all communication between the *console port* and all intelligent units. From the *console port* it is possible to connect to each intelligent unit, by connecting (select/deselect) a unit. Only one unit can be connected at time.

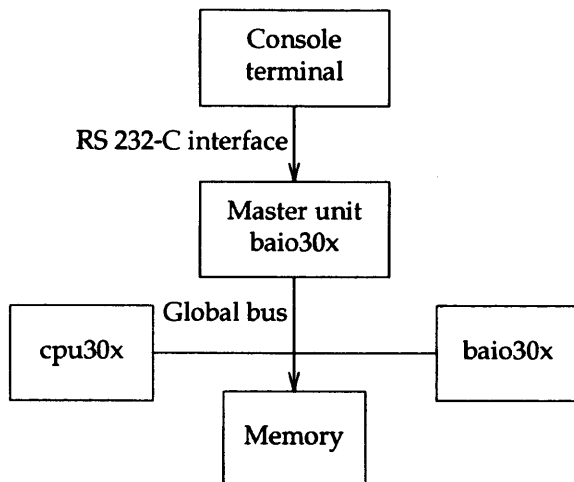


Figure 1-1-1: Console terminal connected to an SMES system

When the *master unit* is connected to another unit, the *master unit* will communicate with the other unit, through the *debug register* on the *control space* located on the *global bus*. There will only be traffic on the *global bus* while the *master unit* is sending or receiving characters to or from the terminal.

While the *master unit* is connected to another unit, the *master unit* will continue testing or waiting for more commands.

## Starting-up

The Diagnostic Programs are available on a 3½" 1.44 MB diskette, which contains Diagnostic Programs for all intelligent units in the Supermax Enterprise Server.

Before booting the Diagnostic Programs, insert the floppy disk in the floptical drive set the **key switch** on the front panel of the Supermax Enterprise Server to **system** and press the **reset button**. If the *SVR4 operating system* is running you must first shutdown the system, before booting the Diagnostic Programs. Please refer to the *Supermax Enterprise Server ABC* manual, to see the detailed description of how to shutdown and reboot the *SVR4 operating system*.

The *prom monitor* on the *baio30x* module will only boot the *master unit*, the *baio30x* diagnostic program will take over and will boot all other intelligent units in the Supermax Enterprise Server. The *master unit* is always selected after the system is booted up.

The following shows an example of booting the Diagnostic Programs:

```

Supermax Boot Monitor - Thu Mar 23 08:35:40 1995 halfway/half
Configuration: C-M---B----- Service computer present.
Slot no = 7. Total memory 16 MB
GetSystemId succeeded. Installation number=0
Masterslot 7. Consoleslot 7. 1 BAIO(s) found
System is booting - type ^c to abort
System is auto starting
16 MB must be initialized
16 MB initialized
Disk TimeOut
Bootdisk 7/1/7/0 - no such device.
Booting from bootdisk 7/1/2/0

Finding configuration on Global bus
Resetting all modules on the Global bus
Starting unit 30 <mem301 module>
Initializing 16 MB Global memory
Loading cpu301 firmware from Boot File System
Starting unit 00 <cpu301 module>
Backplane: BPL304-1 Global bus: 08 slots 33.33 MHz.
Serial no: 1 FCN no.: 0 Local bus: 08 slots 33.33 MHz
Unit Type Module name Serial no FCN no. Module configuration
00 CPU CPU301-1 31 0 MIPS4400/150MHz 1 MB cache
30 MEM MEM301-1 15 147 BaseLo add 0000 MB 016 MB memory
70 IOC BAIO301-1 14 146 MIPS3052/33MHz 008 MB memory
SUBM1 SCU302-1 9 0 128 KB memory
baio 70 ->

```

Figure 1-1-2: Transcript of booting SMES with the Diagnostic Programs

## How to Start a Complete Test on All Modules

When the Supermax Enterprise Server is booted with the diagnostic programs, the units are ready for commands from the command interpreter.

If you want to run a simple test on all modules you can select the units one by one by typing in the following commands:

```
test ; repeat
```

**Return**

To check that the test of the units are complete select all units and see that the pass status is at least one and that no errors have occurred. Then the test is complete.

**NOTE**

The *peripheral* units on the SCSI interface will not be tested in the example.

Please refer to command chapters to see the detailed description of how to run tests on a unit.

## How To Exit the Diagnostic Programs

When you have finished the Diagnostic Programs, the Supermax Enterprise Server can either be powered-down or reset with boot of the *SVR4 operating system*. The reset can be done by pressing the **reset button** on the front panel on the computer or by executing the *reset* command on the *baio30x* module. Before doing this, remove the floppy disk in the floptical drive. Please refer to *Supermax Enterprise Server ABC* manual, to see the detailed description of how to boot the *SVR4 operating system*.

**NOTE**

All test of the *SCSI devices* must be stopped before resetting or powering-down the Supermax Enterprise Server.

*This page is intentionally left blank*

## 2 USER INTERFACE

---

<b>Description of Special Keys</b>	2-1
Select/Deselect Control Keys	2-1-1
▪ Responds From the Modules	2-1-1
Program Control Keys	2-1-2
Command Editor Control Keys	2-1-3

---

<b>Description of Command Interpreter</b>	2-2
Prompt Display	2-2-1
Command Interpreter Control Keys	2-2-2
Command Execution	2-2-2
Command Interpreter Errors	2-2-2
Command History	2-2-3
Command Help Information	2-2-4
Command Options	2-2-5
▪ Digit Value Option	2-2-6
▪ Hex Value Option	2-2-6
▪ Decimal value Option	2-2-7
▪ Double Digit Value Option	2-2-7
▪ Value Range Option	2-2-7
▪ Bit Range Option	2-2-8
▪ String Option	2-2-9
▪ Option Without Parameters	2-2-9
▪ Word Symbol Option	2-2-10
▪ Double Word Symbols Option	2-2-10
▪ Multi Word Symbols Option	2-2-10
▪ Word Symbol or Digit Value Option	2-2-11
▪ Word Symbol or Hex Value Option	2-2-11
▪ Date Option	2-2-12
▪ Time Option	2-2-12
▪ Time Zone Option	2-2-13
▪ Command Name Option	2-2-13

## Table of Contents

Confirm Message	2-2-13
Script Command	2-2-14
Set Parameters	2-2-15
Remote Command	2-2-16

---

<b>Display Modes</b>	2-3
Unit Status Display Mode	2-3-1
SCSI Command Display Mode	2-3-2
Test Display Mode	2-3-2
Short Test Display Mode	2-3-3
Error Display Mode	2-3-4
▪ Error Display Mode with Wait for a Continue Command	2-3-4

---

<b>Test and Error Log</b>	2-4
Test Log	2-4-1
Error Log	2-4-2
▪ Fatal Error Mode	2-4-3



# Figures and Tables

---

<b>Figure 2-2-1</b>	Example of a normal prompt	2-2-1
<b>Figure 2-2-2</b>	Example of a fatal error prompt	2-2-1
<b>Figure 2-2-3</b>	Example of commands to be executed	2-2-2
<b>Figure 2-2-4</b>	Example of a command interpreter error	2-2-3
<b>Figure 2-2-5</b>	Example of a command history	2-2-4
<b>Figure 2-2-6</b>	Example of command help information	2-2-5
<b>Figure 2-2-7</b>	Example of digit value option to a command	2-2-6
<b>Figure 2-2-8</b>	Example of hex value option to a command	2-2-6
<b>Figure 2-2-9</b>	Example of decimal value option to a command	2-2-7
<b>Figure 2-2-10</b>	Example of double digit value option to a command	2-2-7
<b>Figure 2-2-11</b>	Example of value range option to a command	2-2-8
<b>Figure 2-2-12</b>	Example of bit range option to a command	2-2-9
<b>Figure 2-2-13</b>	Example of string option to a command	2-2-9
<b>Figure 2-2-14</b>	Example of option without parameters to a command	2-2-9
<b>Figure 2-2-15</b>	Example of word symbol option to a command	2-2-10
<b>Figure 2-2-16</b>	Example of double word symbols option to a command	2-2-10
<b>Figure 2-2-17</b>	Example of multi word symbols option to a command	2-2-11
<b>Figure 2-2-18</b>	Example of word symbol or digit value option to a command	2-2-11
<b>Figure 2-2-19</b>	Example of word symbol or hex value option to a command	2-2-12
<b>Figure 2-2-20</b>	Example of date option to a command	2-2-12
<b>Figure 2-2-21</b>	Example of time option to a command	2-2-12
<b>Figure 2-2-22</b>	Example of time zone option to a command	2-2-13
<b>Figure 2-2-23</b>	Example of command name option to a command	2-2-13
<b>Figure 2-2-24</b>	Example of confirm message	2-2-14
<b>Figure 2-2-25</b>	Example on options to a typical script command	2-2-14

## Table of Contents

<b>Figure 2-2-26</b>	Example on how to display commands from a script command	2-2-15
<b>Figure 2-2-27</b>	Example of how to see the parameter settings	2-2-15
<b>Figure 2-3-1</b>	Example of a command status display	2-3-1
<b>Figure 2-3-2</b>	Example of a SCSI command display	2-3-2
<b>Figure 2-3-3</b>	Example of a test display	2-3-3
<b>Figure 2-3-4</b>	Example of a short test display	2-3-3
<b>Figure 2-3-5</b>	Example of an error display	2-3-4
<b>Figure 2-3-6</b>	Example of an error display with wait for a continue command	2-3-5

## Description of Special Keys

This chapter describes all special keys used to select/deselect a unit, program control and command editor control keys.

### Select/Deselect Control Keys

The following shows how to select or deselect a unit.

To select the *master unit* type the following key sequence:

**ESC** **#** **Return**

To select a specific unit, type one of the two sequences shown below:

**ESC** **#** **0-F** **Return**

or

**ESC** **#** **0-F** **0-3** **Return**

The keys **0-F** indicate the slot (position) number on the unit and the keys **0-3** indicate the subposition. If the subposition is not selected it is set to 0.

In order to deselect all units in a system type the following key sequence:

**ESC** **#** **Space**

**NOTE**

The **ESC** key can be replaced with **Ctrl-A** in the select/deselect sequence

### Responses From the Modules

The selected unit will respond with a *prompt* or a *unit status*, dependent on whether the unit is inactive or currently running a test. If a unit does not respond to the selected sequence, the cause of the trouble might be one of the following:

- The unit is not installed in the system.

- The diagnostic program is not loaded into the unit.
- The *control space* located on the *global bus* failed in a way that makes it unable to communicate with the unit from the *master unit*.
- The *global bus* or *global memory* failed in a way that makes it unable to load the unit.
- The unit has failed in a way that makes it unable to run the diagnostic program.
- The selected unit is not an intelligent unit - i.e. a memory unit, only *CPU* or *IOC* units are intelligent. Please refer to the *config* command in the the command chapters to see the configuration of the Supermax Enterprise Server.
- Maybe the selected key sequence was not correct.

If the unit does not respond, the previously selected unit will now have been deselected.

## Program Control Keys

It is possible to control program execution by using some special control characters as mentioned below.

**Ctrl-C** This is used to cancel a running test. Program execution stops and the prompt will be displayed on the terminal. It only affects the unit currently selected.

**Ctrl-E** This is used to soft cancel a running test. A soft cancel allows the command currently being executed to complete before the test is stopped and then return to the *prompt*. When running a *peripheral* unit test, it is recommended to use **Ctrl-E** to cancel a running test. **Ctrl-E** only affects the unit currently selected.

**Ctrl-S** This is used to send an *XOFF* character to the *master unit*, which will stop printing on the terminal. If a unit is selected, the program will stop while trying to print on the terminal and an *XON* character must be received before it can continue.

- Ctrl-Q** This is used to send an *XON* character to the program, which will start printing again.
- Ctrl-P** This is used to start printing *SCSI commands* on the terminal, if the unit is running *peripheral* tests. The printing will be turned off automatically, when the unit is deselected. It is necessary to press **Ctrl-P** to start printing again. It only affects the unit currently selected.
- Ctrl-N** This is used to stop *SCSI commands* printing on the terminal. It only affects the unit currently selected.

## Command Editor Control Keys

It is possible to edit a command line by using some special characters as mentioned below.

- Move cursor 1 character to the right.
- ← Move cursor 1 character to the left.
- ↑ Insert character at the cursor position.
- ↓ Delete character at the cursor position.
- Tab** Move cursor 8 characters to the right.
- Del** Delete all characters from the cursor position to the end of the line.
- Return** End editing.

**User Interface**

**Description of Special Keys**

**User Interface**

*This page is intentionally left blank*


## Description of Command Interpreter

The *command interpreter* is used to execute commands on a unit. When the unit displays the *prompt* it is running in *command editor* and is ready for command inputs. The *command interpreter* can execute more than one command from the *command line* buffer. The buffer can hold up to 140 characters.

### Prompt Display

When the prompt is displayed the unit is ready to execute commands from the *command interpreter*. The prompt shows which unit number you are running on.

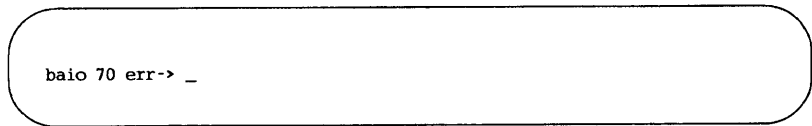
Normally the prompt will look as below:



```
baio 70 -> _
```

Figure 2-2-1: Example of a normal prompt

If the unit has failed with *fatal error mode* the prompt will look as shown below:



```
baio 70 err-> _
```

Figure 2-2-2: Example of a fatal error prompt

**NOTE**

If the prompt is a *fatal error prompt* not all commands can be executed. To cancel this mode you must strike **Ctrl-C**

## Command Interpreter Control Keys

The *command interpreter* has some control keys. They are listed below.

;  
Separates commands.

**Return** Completes editing *command line* buffer. The *command interpreter* then begins executing the commands from the buffer.

## Command Execution

In the following example the *date* command will be executed first and after that the *memory*, *scsi*, and *ethstest* commands will be executed and repeated. The tests will stop when 1000 seconds have passed. The *date* command will only be executed once, because this command cannot be repeated.

```
baio 70 -> date; memory ; scsi; ethstest;repeat -mtime 1000
```

Figure 2-2-3: Example of commands to be executed

A command can be either a *system command* or a *test command*. A *system command* is a command which doesn't test the hardware and a *test command* is a command which tests the hardware.

**NOTE**

Idle LED on the modules, if any, will be turned off while a *test command* is running.

## Command Interpreter Error

After editing the *command line* buffer, the *command interpreter* will analyze the buffer for syntax and command errors. If there is an error the *command interpreter* will print it out and return to the *command editor*.

The example shows the *ethstest* command with an illegal option.



```
baio 70 -> date; memory ; scsi; ethtest -xx;repeat -mtime 1000
Shell: ethtest -xx <-- Illegal option!
```

Figure 2-2-4: Example of a command interpreter error

## Command History

The *command interpreter* has a built-in history facility. The interpreter keeps the last 22 *command line* buffers issued by the operator, and the program allows the operator to re-issue these command lines, possibly with some modifications.

On top of the commands the *command interpreter* has the following editing commands (which all must start in the first character position of the *command line* buffer).

- ?? Gives a numbered list of the last 22 *command line* buffers.
- ? This command makes the interpreter display the most recent *command line* buffer. The operator may then edit the buffer and issue it by pressing the **Return** key.
- ! This command makes the interpreter re-issue the most recent *command line* buffer. The buffer will be displayed and then executed.
- !nn where *nn* is a number (positive, zero, or negative). This command makes the interpreter re-issue *command line* buffer number *nn*. The buffer will be displayed and then executed. The number *nn* may be either of the two numbers displayed for each command with the ?? command.
- ?nn where *nn* is a number (positive, zero, or negative). This command makes the interpreter display *command line* buffer number *nn*. The operator may then edit the buffer and issue it by pressing the **Return** key. The number *nn* may be either of the two numbers displayed for each command with the ?? command.

User Interface	Description of Command Interpreter	User Interface
----------------	------------------------------------	----------------

<i>!</i> string	This command makes the interpreter re-issue the most recent <i>command line</i> buffer whose first characters were <i>string</i> (leading spaces must be included). The buffer will be displayed and then executed.	
-----------------	---	--

<i>?</i> string	This command makes the interpreter display the most recent <i>command line</i> buffer whose first characters were <i>string</i> (leading spaces must be included). The operator may edit the buffer and issue it by pressing the <b>Return</b> key.	
-----------------	---	--

```
baio 70 -> ??
-3 1 date
-2 2 set dispmode=all
-1 3 memory;repeat
-0 4 test;repeat
baio 70 -> ?-1
baio 70 -> memory;repeat
baio 70 -> ?3
baio 70 -> memory;repeat
baio 70 -> ?se
baio 70 -> set dispmode=all
```

Figure 2-2-5: Example of a command history

## Command Help Information

For every command it is possible to display the options available for the command. It can be done in two different ways as shown in the example on the next page.

```

baio 70 -> help memfast
memfast  Memory high speed test
        [-dDEVICE]      [-d{pmem|dmem|nvm|global}]
        [-rADDR_RANGE] [-r{START|#SIZE|START-END|START#SIZE}]
        [-aACCESS]     [-a[word|dword|short|byte][,*]]
        [-cCACHE]      [-c{on|off}]
        [-pPATTERN]    [-p[random|value|setdata|count|cntdown]]
        [-vPAT_VALUE]  [-vLSB_DIGIT[,MSB_DIGIT]]

baio 70 -> memfast -?
memfast  Memory high speed test
        [-dDEVICE]      [-d{pmem|dmem|nvm|global}]
        [-rADDR_RANGE] [-r{START|#SIZE|START-END|START#SIZE}]
        [-aACCESS]     [-a[word|dword|short|byte][,*]]
        [-cCACHE]      [-c{on|off}]
        [-pPATTERN]    [-p[random|value|setdata|count|cntdown]]
        [-vPAT_VALUE]  [-vLSB_DIGIT[,MSB_DIGIT]]

baio 70 ->
    
```

Figure 2-2-6: Example of command help information

**NOTE**

To get *command help information* with the `-?` option it must be the first option to the command.

The *command help information* displays a command text, and after that two columns will be displayed if the command has options. The first column describes which options are available in a short form and the second column describes how the options shall be preset on the command line.

The characters `[` and `]` show that the *command option* is optional. If one or more *command options* are not optional, they must be specified in the order listed in the *command help information*.

## Command Options

An option is used to change a command function. If the command is a test, the options to the command is used to change the test from default. All types of options to a command are described below.

## Digit Value Option

The *digit value option* is used to select a value. It can be specified as decimal or hex digits. To specify that it is a hex value, you must put "0x" in front of the digits. The command interpreter will interpret the digits as hex, if it has the characters "0x" in front of the digits.

```
baio 70 -> repeat -?  
cmd Command example  
    [-nNUMBER] [-n[DIGIT]]  
baio 70 -> cmd -n508  
baio 70 -> cmd -n0x1fc
```

Figure 2-2-7: Example of digit value option to a command

## Hex value option

The *hex value option* is used to select a value on hex digit form without "0x" character before the digits.

```
baio 70 -> cmd -?  
cmd Command example  
    [-nNUMBER] [-n[HEX]]  
baio 70 -> cmd -n1fc
```

Figure 2-2-8: Example of hex value option to a command

## Decimal Value option

The *decimal value option* is used to select a value on decimal digit form.

```
baio 70 -> cmd -?  
cmd Command example  
[-nNUMBER] [-n[DECIMAL]]  
baio 70 -> cmd -n508
```

Figure 2-2-9: Example of decimal value option to a command

## Double Digit Value Option

The *double digit value option* is used to select two values (LSB and MSB value). They can be specified as decimal or hex digits. If the MSB value is not specified it will be set to zero.

```
baio 70 -> cmd -?  
cmd Command example  
[-nNUMBERS] [-nLSB_DIGIT[,MSB_DIGIT]]  
baio 70 -> cmd -n508  
baio 70 -> cmd -n0x1fc  
baio 70 -> cmd -n508,0x14  
baio 70 -> cmd -n0x1fc,20
```

Figure 2-2-10: Example of double digit value option to a command

## Value Range Option

The *value range option* is used to select a start and end value range. The start or the end value can either be on decimal or hex digit format. The "-" character indicates a start and an end value. The "#" character indicates the next digit is size. The "\*" character indicates that a minimum or maximum value shall be used.

The minimum value is used when you want to specify the first memory address for a memory test and you don't know where the first address in the memory starts. The maximum value is used when you want to specify the last memory address for a memory test and you don't know how much memory there is.

If the option is not selected, the command will use default values dependent on which command you are running. Please refer to the specific command in the command chapters to see the detailed description of all parameters for each type of module.

```
baio 70 -> cmd -?
cmd Command example
  [-rNUMBER_RANGE] [-r[START | #SIZE | START-END | START#SIZE]]
baio 70 -> cmd -r512 "Specifies a start value of 512"
baio 70 -> cmd -r#0x14 "Specifies a size of 20"
baio 70 -> cmd -r0x200-532 "Value range from 512 to 532"
baio 70 -> cmd -r512#20 "Value range from 512 to 532"
baio 70 -> cmd -r*-* "Value range from minimum to maximum"
baio 70 -> cmd -r*-532 "Value range from minimum to 532"
baio 70 -> cmd -r* "Specifies a start value of minimum"
baio 70 -> cmd -r##* "Specifies the largest possible size"
baio 70 -> cmd -r*#20 "Value range from minimum to minimum + 20"
```

Figure 2-2-11: Example of value range option to a command

### Bit Range Option

The *bit range option* is used to select bit numbers. The bit numbers can either be on decimal or hex digit format. The ";" character indicates single bit. The "-" character indicates bit range.

```
baio 70 -> cmd -?  
cmd Command example  
    [-bBIT_RANGE] [-b[BIT RANGE FROM 0 TO 64]]  
baio 70 -> cmd -r3-24  
baio 70 -> cmd -r3,5,7,9,0x10  
baio 70 -> cmd -r3-7,9,10-20,24
```

Figure 2-2-12: Example of bit range option to a command

### String Option

The *string option* is used to select text. If the text includes spaces it must be enclosed in quotation marks " ".

```
baio 70 -> cmd -?  
cmd Command example  
    [-tSTRING] [-t[STRING]]  
baio 70 -> cmd -tsingleword  
baio 70 -> cmd -t"More words"
```

Figure 2-2-13: Example of string option to a command

### Option Without Parameters

The *option without parameters* is used to select an option alone.

```
baio 70 -> cmd -?  
cmd Command example  
    [-pOPTION] [-p]  
baio 70 -> cmd -p
```

Figure 2-2-14: Example of option without parameters to a command

## Word Symbol Option

The *word symbol option* is used to select a word symbol name. If the option is not selected the command will use the first word symbol in the *command help information*.

```
baio 70 -> cmd -?  
cmd Command example  
  [-sSYMBOL]  [-s[namex|namey|namew]]  
baio 70 -> cmd -snamex  
baio 70 -> cmd -snamey  
baio 70 -> cmd -snamez
```

Figure 2-2-15: Example of word symbol option to a command

## Double Word Symbols Option

The *double word symbols option* is used to select two different or identical word symbol names. If the second word symbol is not selected, the second word will be the same as the first word in the *command help information*. If the option is not selected, the command will use the first word symbol.

```
baio 70 -> cmd -?  
cmd Command example  
  [-sSYMBOLS] [-s[namex|namey|namew][, *]]  
baio 70 -> cmd -snamex  
baio 70 -> cmd -snamey  
baio 70 -> cmd -snamez, namey  
baio 70 -> cmd -snamey, namex
```

Figure 2-2-16: Example of double word symbols option to a command

## Multi Word Symbols Option

The *multi word symbols option* is used to select multiple word symbol names. If the option is not selected the command will use all word symbol names.



```

baio 70 -> cmd -?
cmd Command example
    [-sSYMBOLS] [-s[namex|namey|namew][,*,*,)]
baio 70 -> cmd -snamex
baio 70 -> cmd -snamey
baio 70 -> cmd -snamez,namey
baio 70 -> cmd -snamey,namex
baio 70 -> cmd -snamey,namex,namew
    
```

Figure 2-2-17: Example of multi word symbols option to a command

### Word Symbol or Digit Value Option

The *word symbol or digit value option* is used to select a word symbol name or a value. The value can be specified as decimal or hex digit. If the option is not selected the command will use the first word symbol in the *command help information*.

```

baio 70 -> cmd -?
cmd Command example
    [-sSYMBOL] [-s[namex|namey|namew][DIGIT]]
baio 70 -> cmd -snamex
baio 70 -> cmd -snamey
baio 70 -> cmd -snamew
baio 70 -> cmd -s255
baio 70 -> cmd -s0xff
    
```

Figure 2-2-18: Example of word symbol or digit value option to a command

### Word Symbol or Hex Value Option

The *word symbol or hex value option* is used to select a word symbol name or a hex value. The value must be specified as hex digits without putting the characters "0x" in front of the digits. If the option is not selected the command will use the first word symbol in the *command help information*.

```
baio 70 -> cmd -?  
cmd Command example  
    [-sSYMBOL] [-s[namex|namey|namew][HEX]]  
baio 70 -> cmd -snamex  
baio 70 -> cmd -snamey  
baio 70 -> cmd -snamew  
baio 70 -> cmd -sff
```

Figure 2-2-19: Example of word symbol or hex value option to a command

### Date Option

The *date option* is used to select a date. A "-" character must be used to separate year, month and date.

```
baio 70 -> cmd -?  
cmd Command example  
    [-dDATE] [-dYYYY-MM-DD]  
baio 70 -> cmd -d1995-04-24
```

Figure 2-2-20: Example of date option to a command

### Time Option

The *time option* is used to select a time. A ":" character must be used to separate hours, minutes and seconds.

```
baio 70 -> cmd -?  
cmd Command example  
    [-tTIME] [-tHH:MM:SS]  
baio 70 -> cmd -t15:13:45
```

Figure 2-2-21: Example of time option to a command

### Time Zone Option

The *time zone option* is used to select a time zone. The time zone is the zone difference from UTC (GMT). The **HH** are the hours and the **MM** are the minutes.

```
baio 70 -> cmd -?
cmd Command example
  [-TIME_ZONE] [-[+|-][HHMM]]
baio 70 -> cmd -0100
baio 70 -> cmd +0200
baio 70 -> cmd +0000
```

Figure 2-2-22: Example of time zone option to a command

### Command Name Option

The *command name option* is used to select a command with or without options.

```
baio 70 -> time -?
time Time on a command
  [command] [COMMAND [OPTIONS]]
baio 70 -> time memfast -dglobal -r#0x1000000
```

Figure 2-2-23: Example of command name option to a command

### Confirm Message

When running a command that has a possibility of destroying data from the *SVR4 operating system* (e.g data on a disk), the command will always display a *confirm message* on the *console terminal* and wait for a confirmation from the user, before starting any commands in the *command line buffer*.

In the example below, the *diskwr* command will only test the disk on SCSI 0 id 1.

```
baio 70 -> diskwr -d00,01
Last warning! The disk (hp3323) on scsi 0 id 0
                Data will now be destroyed (y-OK n-CANCEL)? n
Last warning! The disk (hp3323) on scsi 0 id 1
                Data will now be destroyed (y-OK n-CANCEL)? y
```

Figure 2-2-24: Example of confirm message

If the user has not confirmed the message within 45 seconds, the program will automatically cancel the command and go to the next command in the *command line* buffer.

## Script Command

A *script command* is a command which executes other commands on the unit.

```
baio 70 -> ethernet -?
ethernet Test script for ethernet
          [-dETHERNET_DEVICES] [-d[main][,*,*,]]
          [-tTEST_FUNCTIONS]   [-t[fast|function][,*,*,]]
          [-pDISP_CMDS]        [-p]
```

Figure 2-2-25: Example on options to a typical script command

On all script commands it is possible to see which commands the *script command* will execute. This is always done the following way:

```

baio 70 -> ethernet -p
Test script for ethernet
Device Function Command script
main fast ethctest -dmain
main function ethdiag -dmain -r#0x1000; ethctest -dmain -r*-*; ethctest
-dmain -f1024
baio 70 ->
  
```

Figure 2-2-26: Example on how to display commands from a script command

## Set Parameters

It is possible to set parameters for each unit. These parameters can be changed by the *set* command. Please refer to the *set* command in the command chapters to see the detailed description of all parameters for each type of module.

The different module types have different parameters. To see the parameters of a unit and their actual values do the following:

```

baio 70 -> set
fault-on
cache-on
dispmode=error
repmode=off
repmo=0
term=vt100
scsimode=disc, sync
diskdev=00
tapedev=15
bfsdev=12
baio 70 ->
  
```

Figure 2-2-27: Example of how to see the parameter settings

## Remote Command

When a unit makes another unit execute a command it is a *remote command*. A unit can request a command to more than one unit at a time. This is for example used to test *cache coherence* on the *global bus* between different modules.

A *master remote unit* is the unit which requests a command to another unit. A *slave remote unit* is the unit which acknowledges a command from another unit.

The *slave remote unit* will answer the *master remote unit* on the following conditions:

- When a unit starts running a test and the *remote parameter* is set to **on**.
- When a unit runs a *remote command* on the unit.
- When a unit starts running a test which uses a *remote command*, after the *master remote unit* has started the command.

If the *master remote unit* get no response from all *slave remote units* after 5 minutes the unit will fail with time-out.



If you cancel a running command on a *master remote unit* with **Ctrl-C**, the unit will be hung up until all *slave remote unit* commands are complete or cancelled.

If a test fails when executing a *remote command*, and the unit waits to display the error message, the test must be cancelled before all other *master/slave remote units* can continue.

## Display Modes

There are different types of display modes when a unit is testing. The type of display is dependent on how the *dispmode* parameter is set. Please refer to "Chapter 2-2 Description of Command Interpreter", to see the detailed description of how to set the parameters.

When a unit is not selected it discards all printing until the unit is selected again.

**NOTE**

If the unit is in *error display mode* it will wait for the unit to be selected before it starts printing on the terminal.

## Unit Status Display Mode

The *unit status display mode* is used to see the status of a unit on short form. The displayed status information is unit no, how many tests, errors, passes, and how long time the unit has been testing. Every time the unit status is changed the information will be displayed at the same position on the terminal.

```

baio 70 -> memfast -r** -dglobal
Unit  Running  Tests    Errors    Passes    Time
01   Testing   1         0         0         00:01:06
    
```

Figure 2-3-1: Example of a command status display

**NOTE**

The pass status tells how many times the *command line* buffer has been executed.

The status information will be displayed on the following conditions:

- When a unit is selected and is running tests.
- When the *dispmode parameter* is set to **error** or **status** and a test is started or completed or when an error has occurred.
- When the *dispmode parameter* is set to **test** and the running tests fail.

## SCSI Command Display Mode

The *SCSI command display mode* is used to see the command activities on an *SCSI* interface when running a *peripheral* test on the unit. To enable *SCSI command display mode* type **Ctrl-P** and to disable it type **Ctrl-N**. Every time a new *SCSI* command is started, the information will be displayed at the same position on the terminal.

```
baio 70 -> diskrd -d00 -b0x11200 -s512
Device Command   Blk-address   Blk-length
00      read1      0x00011200   0x00000200
```

Figure 2-3-2: Example of a SCSI command display

## Test Display Mode

The *test display mode* is used to display a test with all command option values on the *test command*. The displayed information is test start time, test passes, test code text, and all information about option values from the command. The information will be displayed when a new test is started.



```

baio 70 -> scsittest -smglobal
Test started at Friday 1995-03-17 - 14:51:20 Pass number: 1
SCSI data transfer test on scsi0
Src-addr.  Des-addr.  Length  Blk-size  Direction  Pattern Cmp Value
0x00760000 0x00290000 0x0100000 0x000100 global->pmem random on 0x2f69f648
  
```

Figure 2-3-3: Example of a test display

The test information will be displayed on the following conditions:

- When the *dispmode* parameter is set to **test** or **errstop** or **all** and a test is started.
- When running the *testlog* command and the error log contains some loggings.

## Short Test Display Mode

The *short test display mode* is used to display a test on a short form. The displayed information is start time, test passes, and test code text. The test information will be displayed when the test is started.

```

baio 70 -> scsittest -smglobal
Test started at Friday 1995-03-17 - 14:51:20 Pass number: 1
SCSI data transfer test on scsi0
  
```

Figure 2-3-4: Example of a short test display

The *short test display mode* will be displayed when the *dispmode* parameter is set to **testshort**.

## Error Display Mode

The *error display mode* is used to see an error from a test, if it has failed. The displayed information is error time, test passes, error code text, and all available information about the error. The information will be displayed when the error has occurred. If the unit is not selected it will wait until it is selected, before it displays the error on the terminal.

```
baio 70 -> scsitest -smglobal
Error occurred at Friday 1995-03-17 - 14:51:29
Data error during SCSI DMA transfer on scsi0 Pass number: 1
Source address:      0x00760a4c <global> Write data: 0x179097c4
Destination address: 0x00290a4c <pmem>   Read data:  0x75097804
```

Figure 2-3-5: Example of an error display

The error information will be displayed on the following conditions:

- When the *dispmode parameter* is set to **testshort** or **errstop** or **all** and an error has occurred.
- When running the *errlog* command, and the error log contains some loggings.

### Error Display Mode with Wait for a Continue Command

If the *dispmode parameter* is set to **errstop** and if an error has occurred the error information is displayed. After that the program will wait for a continue command from the terminal. The following continue commands are available:

- c** This will continue the test.
- Return** This will continue the test.
- b** This will break the current test, and continue with a new test.

```
baio 70 -> scsitest -smglobal
Error occurred at Friday 1995-03-17 - 14:51:29
Data error during SCSI DMA transfer on scsi0 Pass number: 1
Source address:      0x00760a4c <global>  Write data: 0x179097c4
Destination address: 0x00290a4c <pmem>    Read data: 0x75097804

Press command (c=Continue b=Break): _
```

Figure 2-3-6: Example of an error display with wait for a continue command

**NOTE**

If a *fatal error* has occurred it is not possible to continue the test. Instead of waiting for a continue command, the program goes directly to the *fatal error mode*.

*This page is intentionally left blank*

## Test and Error Log

On each unit there are two log buffers used to store information about a running test. These log buffers are a *test log* to store information about a test and an *error log* used to store information about errors from a test.

Every time the *command line* buffer contains a *test command* the log buffers are cleared when the *command interpreter* starts executing the first command from the *command line* buffer.

### Test Log

When a test is started the test information can either be saved and displayed or only be saved in the *test log* buffer. This is dependent on the setting of the *dispmode parameter*.

The *testlog* command is used to display test information from *test log* buffer. Please refer to the command chapters to see the detailed description of the *testlog* command for each type of modules.

The maximum number of tests in the *test log* buffer is 64. If this number is exceeded the oldest log will be deleted and the new ones will be added to the buffer.

When you start a test on a unit the program will do following:

- If the module has an *idle LED* it will be turned off. And it will first be turned on again when the test is complete or cancelled.
- The test information will be saved in the *test log* buffer.
- The test information will be displayed if the *dispmode parameter* is set to do so and the unit is selected.
- The program will continue testing.

**NOTE**

Some *test commands* are split up to more than one test series, for every test series there will be saved test information in the *test log*.

## Error Log

If an error has occurred the error information can either be saved and displayed or only be saved in the *error log* buffer. This is dependent on how the *dispmode* parameter is set.

The *errlog* command is used to display error information from *error log* buffer. Please refer to the command chapters to see the detailed description of *errlog* command for each type of modules.

The maximum number of errors in the *error log* buffer, which has not been displayed yet, is 64. If this number is exceeded an overrun is detected, and the first 64 errors are saved, and the last arrived is discarded.

**NOTE**

Command interpreter errors will not be saved in the *error log* buffer.

If an error occurs on a unit, the program will do following:

- If the module has an *error LED*, it will be turned on. And it will first be turned off when all errors in the *error log* has been displayed on the *console terminal*.
- The error information will be saved in the *error log* buffer. If the *error log* buffer is full the error will be discarded and an overruns count will be counted up.
- Dependent on how the *dispmode* parameter is set the program will wait for the unit to be selected, and then the error information will be displayed.
- The program will continue testing.

### Fatal Error Mode

A *fatal error mode* is if a unit fails with one of following error types:

- An unexpected exception has occurred from the *CPU*.
- A wrong or illegal interrupts is sent to the *CPU*.
- An *error interrupt* sent to the *CPU* when the unit has the *fault parameter* set to **on** (Please refer to the *set* command in the command chapters to see the detailed description).

After the program has saved the error, the program will go straight to the *fatal error mode*.

**NOTE**

In the *fatal error mode*, the program will run without using the interrupt from the *CPU*. Therefore not all commands can be executed. To cancel this mode you must strike **Ctrl-C**

*This page is intentionally left blank*



# 11 BAIO30x MODULE

---

<b>Hardware Description</b>	11-1
-----------------------------	------

---

<b>LEDs Description</b>	11-2
-------------------------	------

---

<b>baio30x Commands</b>	11-3
bfscat command	11-3
bfsload command	11-3
bfsls command	11-3
boot command	11-3
config command	11-3
date command	11-3
debug command	11-3
disk command	11-3
diskexer command	11-3
diskformat command	11-3
diskrblk command	11-3
diskrd command	11-3
diskwr command	11-3
errlog command	11-3
ethdiag command	11-3
ethernet command	11-3
ethmon command	11-3
ethstat command	11-3
ethstest command	11-3
hdlc command	11-3
hdlcdiag command	11-3
hdlcmon command	11-3

## Table of Contents

hdlctest command	11-3
help command	11-3
memaddr command	11-3
memcmp command	11-3
memdata command	11-3
memdisp command	11-3
memexer command	11-3
memfast command	11-3
memgallop command	11-3
memory command	11-3
menu command	11-3
repeat command	11-3
reset command	11-3
scsi command	11-3
scsidiag command	11-3
scsitest command	11-3
scu command	11-3
scucntl command	11-3
sculeds command	11-3
scumon command	11-3
scutest command	11-3
sdconf command	11-3
sddisp command	11-3
sdloadf command	11-3
sdreset command	11-3
set command	11-3
status command	11-3
sysbus command	11-3
tape command	11-3
tapeclr command	11-3
taperd command	11-3
taperet command	11-3
taperew command	11-3
tapewr command	11-3

**Table of Contents**

test command	11-3
testlog command	11-3
time command	11-3
version command	11-3

**Table of Contents**

# Figures and Tables

---

<b>Figure 11-2-1</b>	LEDs status when running selftest or PROM monitor	11-2-1
<b>Figure 11-2-2</b>	LEDs status when running SMES Diagnostic Programs	11-2-1
<b>Figure 11-bfscat-1</b>	Example of the bfscat command	11-3
<b>Figure 11-bfsload-1</b>	Example of the bfsis and bfsload command	11-3
<b>Figure 11-bfsls-1</b>	Example of the bfsls command	11-3
<b>Figure 11-boot-1</b>	Example of using boot command to boot all modules automatic	11-3
<b>Figure 11-boot-2</b>	Example of using boot command to boot all modules manual	11-3
<b>Figure 11-config-1</b>	Example on a configuration of an SMES system	11-3
<b>Figure 11-date-1</b>	Example of using the date command	11-3
<b>Figure 11-disk-1</b>	Example of the disk command displaying all commands from the script	11-3
<b>Figure 11-disk-2</b>	Example of the disk command selecting a few commands from the script	11-3
<b>Figure 11-diskexer-1</b>	Example of using the diskexer	11-3
<b>Figure 11-diskformat-1</b>	Example of using diskformat to format disks	11-3
<b>Figure 11-diskrblk-1</b>	Example of using the diskrblk to reassign a block on a disk	11-3
<b>Figure 11-diskrblk-2</b>	Example of using the diskrblk to display defect list on a disk	11-3
<b>Figure 11-diskrd-1</b>	Example of using the diskrd command to run a seek test	11-3
<b>Figure 11-diskwr-1</b>	Example of the diskwr command	11-3
<b>Figure 11-errlog-1</b>	Example of using the errlog command	11-3
<b>Figure 11-errlog-2</b>	Example of using the errlog command to	11-3

## Table of Contents

	display the latest error	
<b>Figure 11-ethdiag-1</b>	Example of the ethdiag command	11-3
<b>Figure 11-ethernet-1</b>	Example of ethernet command displaying all commands from the script	11-3
<b>Figure 11-ethernet-2</b>	Example of ethernet command selecting a few commands from the script	11-3
<b>Figure 11-ethmon-1</b>	Example of the ethmon command	11-3
<b>Figure 11-ethstat-1</b>	Example of the ethstat command	11-3
<b>Figure 11-ethctest-1</b>	Example of the ethctest command	11-3
<b>Figure 11-hdlc-1</b>	Example of the hdlc command displaying all commands from the script	11-3
<b>Figure 11-hdlc-2</b>	Example of the hdlc command selecting a few commands from the script	11-3
<b>Figure 11-hdlcdiag-1</b>	Example of the hdlcdiag command	11-3
<b>Figure 11-hdlcmon-1</b>	Example of the hdlcmon command	11-3
<b>Figure 11-hdlctest-1</b>	HDLC test plugs used for test of V28 interface	11-3
<b>Figure 11-hdlctest-2</b>	HDLC test plugs used for test of V11 interface	11-3
<b>Figure 11-hdlctest-3</b>	Example of the hdlctest command	11-3
<b>Figure 11-help-1</b>	Example of help command which displays the special keys	11-3
<b>Figure 11-help-2</b>	Example of help command which displays the command help information	11-3
<b>Figure 11-memaddr-1</b>	Example of the memaddr command	11-3
<b>Figure 11-memcmp-1</b>	Example of the memcmp command	11-3
<b>Figure 11-memdata-1</b>	Example of the memdata command	11-3
<b>Figure 11-memdisp-1</b>	Example of the memdisp command	11-3
<b>Figure 11-memexer-1</b>	Example of the memexer command	11-3
<b>Figure 11-memfast-1</b>	Example of the memfast command	11-3
<b>Figure 11-memgallop-1</b>	Example of the memgallop command	11-3
<b>Figure 11-memory-1</b>	Example of memory command displaying all commands from the script	11-3
<b>Figure 11-memory-2</b>	Example of memory command selecting a	11-3

	few commands from the script	
<b>Figure 11-menu-1</b>	Commands on the baio30x module in short format	11-3
<b>Figure 11-menu-2</b>	Commands on the baio30x module in long format	11-3
<b>Figure 11-repeat-1</b>	Example of using the repeat command	11-3
<b>Figure 11-scsi-1</b>	Example of the scsi command displaying all commands from the script	11-3
<b>Figure 11-scsi-2</b>	Example of the scsi command selecting a few commands from the script	11-3
<b>Figure 11-scsidiag-1</b>	Example of the scsidiag command	11-3
<b>Figure 11-scsitest-1</b>	Example of the scsitest command	11-3
<b>Figure 11-scu-1</b>	Example of the scu command displaying all commands from the script	11-3
<b>Figure 11-scucntl-1</b>	Example of the scucntl command	11-3
<b>Figure 11-sculeds-1</b>	Number of LEDs for each type	11-3
<b>Figure 11-sculeds-2</b>	Example of the sculeds command	11-3
<b>Figure 11-scumon-1</b>	Example of the scumon command	11-3
<b>Figure 11-scutest-1</b>	Normal voltage operating range	11-3
<b>Figure 11-scutest-2</b>	Low voltage operating range	11-3
<b>Figure 11-scutest-3</b>	High voltage operating range	11-3
<b>Figure 11-scutest-4</b>	Temperature operating range	11-3
<b>Figure 11-scutest-5</b>	Fan groups	11-3
<b>Figure 11-scutest-6</b>	Example of the scutest command	11-3
<b>Figure 11-sdconf-1</b>	Example of using sdconf to read configuration on all SCSI devices	11-3
<b>Figure 11-sdconf-2</b>	Example of using the sdconf to see which devices that are selected	11-3
<b>Figure 11-sddisp-1</b>	Example of using the sddisp to write a 32 bit value	11-3
<b>Figure 11-sddisp-2</b>	Example of using the sddisp to display a specific device address	11-3
<b>Figure 11-sddisp-3</b>	Example of using the sddisp to display the	11-3

## Table of Contents

	next device address	
<b>Figure 11-sdloadf-1</b>	Example of loading firmware to an SCSI device	11-3
<b>Figure 11-sdreset-1</b>	Example of resetting both SCSI interfaces	11-3
<b>Figure 11-set-1</b>	Example of using set command to display current parameter values	11-3
<b>Figure 11-set-2</b>	Example of using set command to change a parameter	11-3
<b>Figure 11-status-1</b>	Example of the status command	11-3
<b>Figure 11-sysbus-1</b>	Example of using sysbus to read global control space id register	11-3
<b>Figure 11-tape-1</b>	Example of the tape command displaying all commands from the script	11-3
<b>Figure 11-tape-2</b>	Example of the tape command selecting a few commands from the script	11-3
<b>Figure 11-tapeclr-1</b>	Example of the tapeclr command	11-3
<b>Figure 11-taperd-1</b>	Example of the taperd command	11-3
<b>Figure 11-taperet-1</b>	Example of the taperet command	11-3
<b>Figure 11-taperew-1</b>	Example of the taperew command	11-3
<b>Figure 11-tapewr-1</b>	Example of the tapewr command	11-3
<b>Figure 11-test-1</b>	Example of the test command displaying all commands from the script	11-3
<b>Figure 11-test-2</b>	Example of the test command selecting a few commands from the script	11-3
<b>Figure 11-testlog-1</b>	Example of using the testlog command	11-3
<b>Figure 11-testlog-2</b>	Example of using the testlog command to display the latest test	11-3
<b>Figure 11-time-1</b>	Example of the time command	11-3
<b>Figure 11-version-1</b>	Example of the version command	11-3



## Hardware Description

NOT IMPLEMENTED YET.

*This page is intentionally left blank*

## LEDs Description

This chapter describes the *LEDs* status on the front of the *baio30x* module.

IDLE	KERNEL	DRIVER	ERROR	Description
off	off	off	off	Slave PROM monitor
off	off	on	off	Selftest running
off	off	on	on	Selftest failed
off	on	on	off	PROM monitor
on	off	on	off	Running in PROM menu or debugger
on	off	on	on	Selftest failed with exception error

Figure 11-2-1: LEDs status when running selftest or PROM monitor

IDLE	KERNEL	DRIVER	ERROR	Description
off	on	off	off	Idle state
off	on	off	on	Idle state and the <i>error log</i> buffer has an error that has not been displayed
on	on	off	off	A <i>test command</i> is running
on	on	off	on	A <i>test command</i> is running and the <i>error log</i> buffer has an error that has not been displayed
on	on	on	on	Reset state during boot

Figure 11-2-2: LEDs status when running SMES Diagnostic Programs

**baio30x module**

**LEDs Description**

**baio30x module**

*This page is intentionally left blank*

**NAME**

*bfscat* – Display file contents from Boot File System

**SYNOPSIS**

*bfsls* FILE\_NAME [-pPAGESTOP]

FILE\_NAME        STRING

[-pPAGESTOP]    [-p]

**DESCRIPTION**

The *bfscat* command is used to display the contents of a file on the BFS (Boot File System). The *SCSI device* which is used for the BFS is specified in the *bfsdev* parameter. Please refer to the *set* command to see the detailed description of the *bfsdev* parameter.

The command has the following options:

**FILE\_NAME**

This option is used to specify a file name which should be displayed. The option must be selected as the first option to the command, and it must be specified as a *string option*.

**-pPAGESTOP**

If this option is selected, the command will write one page with a page number on the *console terminal*, and then it will wait for a **Return** before displaying the next page.

## EXAMPLES

```
baio 70 -> bfscat version -p
Test started at Friday 1995-05-26 - 11:09:50
Display file contents from Boot File System on SCSI device: 12
File name: version

SMES Diagnostic Programs version 1.03 950821

Page 1 -- Press RETURN to continue
baio 70 ->
```

Figure 11-bfscat-1: Example of the bfscat command

## NOTICES

Same as the *bfsls* command.

No character conversion takes place. Therefore, this command should only be used to display ASCII files.

## BUGS

Same as the *bfsls* command.

## COMMAND ERRORS

Same as the *bfsls* command.

The command can fail with the error *File not found on the boot disk*. This is because the file is not on *BFS*.

## REFERENCES

See also commands:  
*bfsload*, *bfsls*, *sdconf*, *set*.

**NAME**

*bfsload* – Load data from Boot File System

**SYNOPSIS**

*bfsload* FILE\_NAME FILE\_LENGTH [-oFILE\_OFFSET] [-mdMEM\_DEVICE] [-mrMEM\_RANGE]

FILE\_NAME                   STRING

FILE\_LENGTH                 DIGIT

[-oFILE\_OFFSET]           [-oDIGIT]

[-mdMEM\_DEVICE]         [-md[pmem | dmem | nvm | global |  
subm1 | subm2 | subm3]]

[-mrMEM\_RANGE]         [-mr[START | #SIZE | START-END |  
START#SIZE]]

**DEFAULTS**

*bfsload* FILE\_NAME FILE\_LENGTH -o0 -mdpmem

**DESCRIPTION**

The *bfsload* command is used to load a file from the BFS (Boot File System) to memory. The SCSI device which is used for the BFS is specified in the *bfsdev* parameter. Please refer to the *set* command to see the detailed description of the *bfsdev* parameter.

The command has the following options:

**FILE\_NAME**

This option is used to specify a file name to be loaded into the memory. The option must be selected as the first option to the command, and it must be specified as a *string option*.

**FILE\_LENGTH**

This option is used to specify the number of bytes to be loaded from the file. The option must be selected as the second option to the command, and it must be specified as a *digit value option*. The number of bytes will always be aligned to 4. If the number of bytes added with the value from the FILE\_OFFSET option is bigger than the size of the file, only the data up to the end of the file will be loaded.

**-oFILE\_OFFSET**

The option is used to specify an offset address in the file from where the data should be loaded. The option must be specified as a *digit value option*. The offset address will always be aligned to 4 by means of cutting. If the offset address is bigger than the size of the file, nothing is loaded from the file.

**-mdMEM\_DEVICE**

This option is used to select the memory where the command has to store the loaded data. The option must be specified as a *word symbol option*. The following memories can be specified:

- pmem** Program memory.
- dmem** Data memory.
- nvm** NVM RAM.
- global** Global memory.
- subm1** Submodule 1. Only available if the submodule is installed and has a local memory mounted.
- subm2** Submodule 2. Only available if the submodule is installed and has a local memory mounted.
- subm3** Submodule 3. Only available if the submodule is installed and has a local memory mounted.

**-mrMEM\_RANGE**

This is used to specify the memory address range where the command has to write the loaded data. The option must be specified as a *value range option*. Default, the command will reserve the next free memory array with the necessary number of bytes, calculated as the value from the **FILE\_LENGTH** option aligned to 0x100.



## EXAMPLES

```

baio 70 -> bfsls
Test started at Friday 1995-05-26 - 11:08:38
List contents of the Boot File System on SCSI device: 12
 2 Block 0x00014-0x00014 Size 512 1995-04-12 - 11:05:49 .
 2 Block 0x00014-0x00014 Size 512 1995-04-12 - 11:05:49 ..
 3 Block 0x00015-0x00015 Size 46 1995-04-12 - 11:09:42 version
baio 70 -> bfsload version 46 -mr0x100000
Test started at Friday 1995-05-26 - 11:10:27
Load data from Boot File System on SCSI device: 12
File name      Length  Offset  Memory range
version        46     0      0x00100000-0x00100100 <pmem>
baio 70 ->

```

Figure 11-bfsload-1: Example of the *bfsls* and *bfsload* command

## NOTICES

Same as the *bfsls* command.

## BUGS

Same as the *bfsls* command.

## COMMAND ERRORS

Same as the *bfsls* command.

The command can fail with the error *File not found on the boot disk*. This is because the file is not on the *BFS*.

If the command returns the error code *Options error*, it is because the memory range is not aligned on 4 bytes addresses or because the specified memory range is too small.

## REFERENCES

See also commands:  
*bfsct*, *bfsls*, *boot*, *sdconf*, *set*.

**bfsload (boot)**

**baio30x command**

**bfsload (boot)**

*This page is intentionally left blank*

## NAME

*bfsls* – List contents of the Boot File System

## SYNOPSIS

*bfsls*

## DESCRIPTION

The *bfsls* command is used to list the contents of the *BFS* (Boot File System) on an *SCSI device*. The *SCSI device* which is used for the *BFS* is specified in the *bfsdev* parameter. Please refer to the *set* command to see the detailed description of the *bfsdev* parameter.

The command always lists the contents, no matter how the *dispmode* parameter is set. The following contents are displayed from each file on the *BFS*:

- *BFS* I-node no.
- *BFS* block addresses.
- Number of bytes.
- Creation time.
- File name.

## EXAMPLES

```
baio 70 -> bfsls
Test started at Friday 1995-05-26 - 11:08:38
List contents of the Boot File System on SCSI device: 12
 2 Block 0x00014-0x00014 Size 512 1995-08-21 - 10:38:01 .
 2 Block 0x00014-0x00014 Size 512 1995-08-21 - 10:38:01 ..
 3 Block 0x00015-0x0022c Size 274336 1995-08-21 - 10:39:31 bioc
 4 Block 0x0022d-0x0033d Size 139696 1995-08-21 - 10:42:20 cpu301dp
baio 70 ->
```

Figure 11-bfsls-1: Example of the *bfsls* command

## NOTICES

The command cannot be executed while running in *fatal error mode*.

If the program has not read the configuration on the *SCSI device* before (this is normally done with the *sdconf* command), the command will automatically read the configuration before reading the *BFS* on the *SCSI device*. This also happens if you change the media on the *SCSI device*.

## BUGS

The Diagnostic Programs cannot handle a *VTOC* (Volume Table Of Contents) on an *SCSI device*, e.g. the *BFS* must be on the first address block on the *SCSI device*.

## COMMAND ERRORS

The command can fail with the error *No boot disk available*. This is because no *SCSI device* is defined in the *bfsdev* parameter. This only happens if the unit is not a *master unit*.

The command can fail with the error *Illegal boot disk format*. This is because there is no *BFS* on the specified *SCSI device*.

## REFERENCES

See also commands:  
*bfsat*, *bfsload*, *boot*, *sdconf*, *set*.

**NAME**

*boot* - Boot all modules on the Global bus from boot disk

**SYNOPSIS**

*boot* [-cCOMMAND] [-manMAN\_STARTUP] [-tMODULE\_TYPE] [-uBOOT\_UNIT] [-fFILE\_NAME] [-oFILE\_OFFSET] [-maMEMORY\_ADD] [-eSTART\_ENTRY]

[-cCOMMAND]                   [-c[boot | conf | reset | meminit | load  
| start | disp]]

[-manMAN\_STARTUP]           [-man]

[-tMODULE\_TYPE]           [-t[mem301 | baio301 | cpu301 |  
[DIGIT]]]

[-uBOOT\_UNIT]               [-uHEX]

[-fFILE\_NAME]               [-fSTRING]

[-oFILE\_OFFSET]           [-oDIGIT]

[-maMEMORY\_ADD]           [-maDIGIT]

[-eSTART\_ENTRY]           [-eDIGIT]

**DEFAULTS**

*boot* -cboot

**DESCRIPTION**

The *boot* command is used to boot the Diagnostic Programs on other modules on the *global bus*. The command will automatically be executed on the *master unit* after it has been booted. I.e it is only necessary to run the command manually if the boot has failed.

Default, the command automatically boots all the modules connected to the *global bus* as described in the following:

- Find configuration on the *global bus*.
- Reset all modules on the *global bus*.
- Start all *mem30x* modules.
- Initialize the *global memory*.
- Load the *baio30x* firmware from the *BFS*.
- Start all *baio30x* modules.
- Load the *cpu30x* firmware from the *BFS*.
- Start all *cpu30x* modules.
- Display the Supermax Enterprise Server configuration.

It is possible to boot the modules step by step manually. This is shown in the second example.

The command will use the *bfsls* and *bfsload* commands to load the diagnostic firmware from the *BFS* to the *global memory*.

The command has the following options:

#### **-cCOMMAND**

This option is used to specify which command the *boot* command shall perform. The option must be specified as a *word symbol option*, and the following types of commands can be specified:

- |              |   |
|--------------|---|
| <b>boot</b>  | This command will boot all the other modules on the <i>global bus</i> automatically.  |
| <b>conf</b>  | This command will find the configuration on the <i>global bus</i> . This is done by accessing all slots (positions) on the <i>global control space</i> one by one to see which modules are connected to the <i>global bus</i> . |
| <b>reset</b> | This command will set all modules connected to the <i>global bus</i> to selective reset state. This is done by writing a selective reset value to the <i>control space control register</i> on all modules on the <i>global</i> |

boot (boot)

baio30x command

boot (boot)

- bus*. This does not include the unit that executes this command.
- meminit** This command will initialize (clear) data in all memories that are connected to the *global bus*. The command will perform a selective reset on all *CPU* and *IOC* units before initializing the memory.
- load** This command will load firmware from the *BFS* to the *global memory*. The firmware to be loaded is specified by the **MODULE\_TYPE** and **FILE\_NAME** options.
- start** This command will start/restart the modules on the *global bus*. The modules to be started are specified by the **MODULE\_TYPE** and **BOOT\_UNIT** options.
- disp** This command will display the configuration of the Supermax Enterprise Server just as the *config* command.

**-manMAN\_STARTUP**

If this option is selected the command will start a unit without enabling the cache and the fault interrupt. This means that the default *cache parameter* on the started unit will be set to **off** instead of **on**, and the *fault parameter* will be set to **poll** instead of **on**. The option only has effect, if the **COMMAND** option is set to **boot** or **start**.

**-tMODULE\_TYPE**

This option is used to specify which type of module the command shall use when loading firmware into the *global memory* or starting the modules. The option must be specified as a *word symbol or digit value option*, and it only has effect, if the **COMMAND** option is set to **load** or **start**. Default, the command will select all types of modules. The following predefined types can be specified:

boot (boot)

baio30x command

boot (boot)

- mem301** This will select all MEMs on the *mem30x* modules.
- baio301** This will select all IOCs on the *baio30x* modules.
- cpu301** This will select all CPUs on the *cpu30x* modules.

**-uBOOT\_UNIT**

This option is used to specify which unit that shall be loaded or started. The option must be specified as a *hex value option*. The first hex digit is the slot (position) number, and the second digit is the subposition number. Default, the command will start all units. The option only has effect, if the **COMMAND** option is set to **load** or **start**.

**-fFILE\_NAME**

This option is used to specify an alternative file name when loading firmware from the *BFS*. It is not necessary to use this feature under normal circumstances, because the program knows the file name. The option must be specified as a *digit value option*, and it only has effect, if the **COMMAND** option is set to **load**.

**-oFILE\_OFFSET**

This option is used to specify an alternative file offset address when loading firmware from the *BFS*. It is not necessary to use this feature under normal circumstances, because the program knows the offset. The option must be specified as a *digit value option*, and it only has effect, if the **COMMAND** option is set to **load**.

**-maMEMORY\_ADD**

This option is used to specify an alternative load address in the global memory when loading the firmware from the *BFS*. It is not necessary to use this feature under normal circumstances, because the program knows the address. The option must be specified as a *digit value option*, and it only has effect, if the **COMMAND** option is set to **load**.



**-eSTART\_ENTRY**

This option is used to specify an alternative firmware start address in the global memory when booting a module. It is not necessary to use this feature under normal circumstances, because the program knows the address. The option must be specified as a *digit value option*, and it only has effect, if the **COMMAND** option is set to **start**.

If no options are given, the *boot* command will boot all modules automatically.

**EXAMPLES**

```

baio 40 -> boot
Finding configuration on Global bus
Resetting all modules on the Global bus
Starting unit 10 <mem301 module>
Initializing 256 MB Global memory
Loading baio301 firmware from the Boot File System
Starting unit 20 <baio301 module>
Starting unit 30 <baio301 module>
Loading cpu301 firmware from the Boot File System
Starting unit 00 <cpu301 module>
Starting unit 01 <cpu301 module>
Starting unit 02 <cpu301 module>
Starting unit 03 <cpu301 module>
Backplane: BPL305-1                Global bus: 05 slots 33.33 MHz.
Serial no: 1          FCN no.: 0    Local bus: 05 slots 33.33 MHz
Unit Type  Module name      Serial no FCN no.  Module configuration
00 CPU     CPU300-1      31       0       MIPS4400/150MHz    01 MB cache
01 CPU     CPU300-1      31       0       MIPS4400/150MHz    01 MB cache
02 CPU     CPU300-1      31       0       MIPS4400/150MHz    01 MB cache
03 CPU     CPU300-1      31       0       MIPS4400/150MHz    01 MB cache
10 MEM     MEM301-1      12       147     BaseLo add 0000 MB 256 MB memory
20 IOC     BAI0301-1    13       146     MIPS3052/33MHz     008 MB memory
30 IOC     BAI0301-1    14       146     MIPS3052/33MHz     008 MB memory
40 IOC     BAI0301-1    8        146     MIPS3052/33MHz     008 MB memory
          SUBM1 SCU302-1    9        0       128 KB memory
baio 40 ->

```

Figure 11-boot-1: Example of using the boot command to boot all modules automatically

```
baio 40 -> boot -econfg
Finding configuration on Global bus
baio 40 -> boot -creset
Resetting all modules on the Global bus
baio 40 -> boot -cstart -tmem301
Starting unit 10 <mem301 module>
baio 40 -> boot -cmeminit
Resetting all CPU and IOC modules on the Global bus
Initializing 256 MB Global memory
baio 40 -> boot -cload -tbaio301
Loading baio301 firmware from the Boot File System
baio 40 -> boot -cstart -tbaio301 -u20
Starting unit 20 <baio301 module>
baio 40 -> boot -cstart -tbaio301 -u30
Starting unit 30 <baio301 module>
baio 40 -> boot -cload -tcpu301
Loading cpu301 firmware from the Boot File System
baio 40 -> boot -cstart -tcpu301 -u00
Starting unit 00 <cpu301 module>
baio 40 -> boot -cstart -tcpu301 -u01
Starting unit 01 <cpu301 module>
baio 40 -> boot -cstart -tcpu301 -u02
Starting unit 02 <cpu301 module>
baio 40 -> boot -cstart -tcpu301 -u03
Starting unit 03 <cpu301 module>
baio 40 ->
```

Figure 11-boot-2: Example of using the boot command to boot all modules manually

## NOTICES

The command cannot be executed while running in *fatal error mode*.

The command does not clear *test log* and *error log*.

When loading firmware from the *BFS* on the *SCSI device*, the *Diagnostic Programs media (floptical)* must be present in the *SCSI drive* before running the command.

If the command only performs a selective reset (**COMMAND** set to **reset**) on all modules on the *global bus*, the *global memory* cannot be accessed (the unit will fail with *missing target acknowledge*) until the *mem30x* has been started again.

## DIAGNOSTICS

If the *boot* command fails, then try to diagnose the modules by means of the *sysbus* command, or perform a memory data bit test in the *global memory*.

## BUGS

The command cannot save the boot information in the *test log* buffer. The information will only be displayed on the *console terminal*, no matter how the *dispmode* parameter is set.

## WARNINGS

Don't run the *boot* command on other units than the *master unit*.

No units on the *global bus* may perform tests on the *global bus* while running the *boot* command.

In certain cases the hardware (*global bus*) can break down (hang-up), if you are running the *boot* command after the Supermax Enterprise Server has been booted with the Diagnostic Programs.

## REFERENCES

See also commands:

*bfsload*, *bfsls*, *config*, *reset*, *set*, *sdconf*, *sysbus*.

**boot (boot)**

**baio30x command**

**boot (boot)**

*This page is intentionally left blank*

config (boot)

baio30x command

config (boot)

## NAME

*config* – System and module configuration

## SYNOPSIS

*config*

## DESCRIPTION

The *config* command is used to display the configuration of the Supermax Enterprise Server system. The configuration information is taken from a table which is made by the *master unit* using the *boot* command.

## EXAMPLES

```

baio 70 -> config
Backplane: BPL304-1                Global bus: 08 slots 33.33 MHz.
Serial no: 1      FCN no.: 0        Local bus: 08 slots 33.33 MHz
Unit Type  Module name  Serial no FCN no.  Module configuration
00 CPU     CPU301-1        30        0         MIPS4400/150MHz    01 MB cache
30 MEM     MEM301-1          15        147       BaseLo add 0000 MB 016 MB memory
70 IOC     BAI0301-1         14        146       MIPS3052/33MHz     008 MB memory
          SUBM1 SCU302-1  2         0         128 KB memory
baio 70 ->
    
```

Figure 11-config-1: Example on a configuration of an SMES system

## NOTICES

The command does not clear *test log* and *error log*.

The command cannot be repeated. It will only run once.

## REFERENCES

See also commands:

*boot*

**config (boot)**

**baio30x command**

**config (boot)**

*This page is intentionally left blank*

**NAME**

*date* – Read or set date

**SYNOPSIS**

*date* [-dDATE] [-tTIME] [TIME\_ZONE]

[-dDATE]            [-dYYYY-MM-DD]

[-tTIME]            [-tHH:MM:SS]

[TIME\_ZONE]        [[+|-][HHMM]]

**DESCRIPTION**

The *date* command is used to display or set the current date and time on the unit.

The startup time is initialized from the *master unit* when it is booting. The *master unit* reads the current date and time from the Service computer submodule and distributes a copy to all other units in the Supermax Enterprise Server.

If no Service computer submodule is installed in the Supermax Enterprise Server, the date and time will be set to the date of the *master unit* version.

The *date* command has the following options:

**-dDATE**

This option is used to change the current date on the unit and it must be specified as a *date option* on the form "YYYY-MM-DD".

**-tTIME**

This option is used to change the current time on the unit and it must be specified as a *time option* on the form "HH:MM:SS".

**TIMEZONE**

This option is used to change the current time zone on the unit and it must be specified as a *time zone option* on the form "+HHMM" or "-HHMM". The time zone must be specified as the difference between the actual time zone and UTC (GMT) in hours and minutes.

date (system)

baio30x command

date (system)

If no options are given, the *date* command displays the current date, time, and time zone of the the unit.

## EXAMPLES

```
baio 70 -> date -d1995-04-06 -t13:40:10 +0200
Thursday 1995-04-06 - 13:40:10 GMT +0200
baio 70 -> date
Thursday 1995-04-06 - 13:40:11 GMT +0200
```

Figure 11-date-1: Example of using the date command

## NOTICES

The command does not clear *test log* and *error log*.

The command cannot be repeated. It will only run once.

When setting date, time or time zone it does not affect other units.

During the "Daylight Saving Time" period the time zone must be changed.

## BUGS

The internal clock on the unit does not run while the unit is running in the *fatal error mode*.

## REFERENCES

See also commands:

*boot*, *scucntl*



debug (system)

baio30x command

debug (system)

**NAME***debug* – Call debugger**SYNOPSIS***debug***DESCRIPTION**

The *debug* command is a diagnostic feature which normally isn't used by the operator. The command is used to debug the Diagnostic Programs on the unit. If *debug* is accidentally entered, use **C**

**Return** to exit.

**NOTICES**

The command doesn't clear *test log* and *error log*.

The command cannot be repeated. It will only run once.

If the **debug button** on the *scu30x* (Service Computer) submodule is activated, all the units in the Supermax Enterprise Server will enter the debugging mode.

**debug (system)**

**baio30x command**

**debug (system)**

*This page is intentionally left blank*

**NAME**

*disk* - Test script for SCSI disk drives

**SYNOPSIS**

```
disk [-mTEST_MODES] [-tTEST_FUNCTIONS] [-pDISP_CMDS]
      [-m[read | write] [*,*,]]
      [-t[fast | function] [*,*,]]
      [-pDISP_CMDS]      [-p]
```

**DEFAULTS**

*disk* -mread,write -tfast,function

**DESCRIPTION**

This command is a *script command*, and it is used to run a complete test of disk drive(s). The command executes other commands which are all tests of disk drives, either read or write tests. It is possible to specify which commands of the *disk script* to be executed. This is done by means of the **TEST\_MODES** and **TEST\_FUNCTIONS** options.

The disk drives used in the tests will be all the disk drives from the *diskdev parameter*. Please refer to the *set* command to see the detailed description of the *diskdev parameter*.

The *disk* command has the following options:

**-mTEST\_MODES**

This option is used to specify which test modes should be selected from the *disk script*. The option must be specified as a *multi word symbol option*, and the following test modes can be specified:

- |              |  |
|--------------|--|
| <b>read</b>  | This test mode will select commands from the script which only read from the disk drives. Data on the disk will not be destroyed.          |
| <b>write</b> | This test mode will select commands from the script which both write and read to/from the disk drives. Data on the disk will be destroyed. |

**-tTEST\_FUNCTIONS**

This option is used to specify which test functions should be selected from the *disk script*. The option must be specified as a *multi word symbol option* and the following test functions can be specified:

- fast** If this test function is selected, the disk drives will only be tested fast and not thoroughly. The selected commands will only use the *program memory* when running the tests.
- function** If this test function is selected, the disk drives will be tested thoroughly. The selected commands will use the *global memory* when running the tests.

**-pDISP\_CMDS**

If this option is specified, the selected commands from the script will only be displayed, not executed.

Without option the command executes all the commands from the scripts.

**EXAMPLES**

```

baio 70 -> disk -p
Test script for SCSI disk drives
Device Function Command script
read fast diskrd -b#0x800; diskrd -b#0x800 -aseek
write fast diskwr -b#0x800 -twrcmp; diskwr -b#0x800 -tcmp -arandom
read function diskrd -aseqin -mdglobal; diskrd -aseek -s1-256 -mdglobal
write function diskwr -aseqin -s1-256 -tcmp -mdglobal; diskwr -arandom
-adfirst -s1-256 -twrcmp -mdglobal

baio 70 ->

```

Figure 11-disk-1: Example of the disk command displaying all commands from the script

disk (disk)

baio30x command

disk (disk)

```
baio 70 -> disk -tfast -mread -p
Device Function Command script
read fast diskrd -b#0x800; diskrd -b#0x800 -aseek
baio 70 ->
```

Figure 11-disk-2: Example of the disk command selecting a few commands from the script

## NOTICES

The command cannot be used while running in *fatal error mode*.

## WARNINGS

If you are using the command to perform write test on a disk drive, a *confirm message* is displayed before destroying data on the disk.

## REFERENCES

See also commands:

*set, sdconf, diskrd, diskwr.*

**disk (disk)**

**baio30x command**

**disk (disk)**

*This page is intentionally left blank*

**NAME**

*diskexer* – Disk exerciser test

**SYNOPSIS**

*diskwr* [-dDISK\_DEVICES] [-bbBLK\_RANGE2] [-bBLK\_RANGE] [-ssBLK\_PER\_CMD2] [-sBLK\_PER\_CMD] [-aACC\_MODES] [-cDISK\_CMDS] [-nNO\_OF\_BLKs] [-mdMEM\_DEVICE] [-mrMEM\_RANGE]

[-dDISK\_DEVICES] [-d[00 | 01 | 02 | 03 | 04 | 05 | 06 | 10 | 11 | 12 | 13 | 14 | 15 | 17][\*,\*,,]]

[-bbBLK\_RANGE2] [-b[START | #SIZE | START-END | START#SIZE]]

[-bBLK\_RANGE] [-b[START | #SIZE | START-END | START#SIZE]]

[-ssBLK\_PER\_CMD2] [-s[START | #SIZE | START-END | START#SIZE]]

[-sBLK\_PER\_CMD] [-s[START | #SIZE | START-END | START#SIZE]]

[-aACC\_MODES] [-a[seqin | seqout | seek | random][\*,\*]]

[-cDISK\_CMDS] [-c[read | write | seek][\*,\*]]

[-nNO\_OF\_BLKs] [-nDIGIT]

[-mdMEM\_DEVICE] [-md[pmem | global]]

[-mrMEM\_RANGE] [-mr[START | #SIZE | START-END | START#SIZE]]

**DEFAULTS**

*diskexer* -dDISKDEV\_PARAMETER -b\*-\* -s32 -aseqin -cread -mdpmem

**DESCRIPTION**

This command can be used to exercise disk drive(s) or it can be used to measure the performance of a disk drive. This is done by performing *SCSI write, read, or seek commands* on the disk drive. If using the command to measure the performance of a disk drive then use the *time* command in connection with the *diskexer* command to

display the time elapsed during the *diskexer* command.

It is possible to specify two sets of parameters (first and second set of parameters) for the disk drive exercise. The following parameters can be set for each set:

- Block address range.
- Blocks per command.
- *SCSI command* (read, write or seek command).
- Access mode in the block address range.

The second set of parameters is only used, if the **BLK\_RANGE2** or **BLK\_PER\_CMD2** options is selected. If the second set of parameters is used during the test, the command executes an *SCSI command* from the first and the second set of parameters, alternately.

The *diskexer* command has the following options:

**-dDISK\_DEVICES**

Same description as the *diskwr* command.

**-bbBLK\_RANGE2**

Same as **BLK\_RANGE** option, but this is used for the second set of parameters.

**-bBLK\_RANGE**

This option is used for the first set of parameters to specify the block address range for the access to the disk. The option must be specified as a *value range option*.

**-ssBLK\_PER\_CMD2**

Same as **BLK\_PER\_CMD** option, but this is used for the second set of parameters.

**-sBLK\_PER\_CMD**

This option is used for the first set of parameters to specify the number of blocks per *SCSI write/read* command during the test. It is also possible to specify a block transfer range. If this is done, the number of blocks per access is calculated on a pseudo random basis for every access to the *SCSI device*. The option must be specified as a *value range option*.



**-aACC\_MODES**

Same description as the *diskwr* command, but the **ACC\_MODES** option must be specified as a *double word symbol option*. The access mode can be different for the two sets of parameters. The first word symbol will be used for the first set of parameters and second for the second set of parameters.

**-cDISK\_CMDS**

This option is used to specify which type of *SCSI commands* the *diskexer* command shall perform during access to a disk drive. The option must be specified as a *double word symbol option*. The *SCSI command* can be different for the two sets of parameters. The first word symbol will be used for the first set of parameters and the second for the second set of parameters. The following *SCSI commands* can be specified:

- |              |  |
|--------------|--|
| <b>read</b>  | This will perform an <i>SCSI read</i> command on the disk drives.  |
| <b>write</b> | This will perform an <i>SCSI write</i> command on the disk drives. |
| <b>seek</b>  | This will perform an <i>SCSI seek</i> command on the disk drives.  |

**-nNO\_OF\_BLKs**

This option is used to specify how many blocks to be accessed on a disk drive. It must be specified as a *digit value option*. Default number of blocks is dependent on the size of block ranges of the two sets of parameters.

**-mdMEM\_DEVICE**

Same description as the *diskwr* command.

**-mrMEM\_RANGE**

Same description as the *diskwr* command, but the calculation of the necessary number of bytes is as follows:  
(disk\_numbers \* block\_per\_cmd \* block\_size) aligned to 0x10000.

## EXAMPLES

```
baio 70 -> time diskexer -d12 -aseqin,seqout -b0#0x100 -bb0xa40#256 -s4 -ss4
Test started at Wednesday 1995-07-12 - 10:46:57
Disk exerciser test on SCSI device: 12
Block address range  Blocks  Access Cmd  Memory range  Device  Blocks
0x00000000-0x00000100  4      seqin read  0x00090000-   pmem   0x00000200
0x00000a40-0x00000b40  4      seqout read  0x000a0000
Command time: 27 second. 20 ms.
baio 70 ->
```

Figure 11-diskexer-1: Example of using the diskexer command

## NOTICES

The command cannot be executed while running in *fatal error mode*.

The maximum number of blocks per *SCSI command* is the size of available memory or up to 16 Mbyte.

It is not possible to write to blocks located before the 4th block on a disk (this does not apply to removable disks). This is because the *disk configuration table* on the disk must not be destroyed.

## COMMAND ERRORS

If the command returns the error code *Options error*, it is because the memory range is not aligned on 4 bytes addresses or because the specified memory range is too small.

## WARNINGS

If you are using the command to write exercise test on the disk drive, a confirm message is displayed before writing on the disk. If this is confirmed with a **(y)**, the data on the disk will be destroyed.

## BUGS

If using the command to measure a write performance of a disk drive, the elapsed time during the *diskexer* command is not correct, because elapsed time during the command also include the confirm message on the *console terminal*. To work around this just repeat the commands and use the second measured time during the *diskexer* command.

## REFERENCES

See also commands:

*set, sdconf, diskrd, diskwr, time*

**diskexer (disk)****baio30x command****diskexer (disk)**

*This page is intentionally left blank*

**NAME**

*diskformat* – Format disk

**SYNOPSIS**

*diskformat* [-dDISK\_DEVICES] [-cCOMMAND] [-gCLR\_GLIST] [-mDISK\_MODEL] [-snSERIALNO] [-sDISK\_SIZE\_MB]

[-dDISK\_DEVICES] [-d[00 | 01 | 02 | 03 | 04 | 05 | 06 | 10 | 11 | 12 | 13 | 14 | 15 | 17][,\*,\*]]

[-cCOMMAND] [-c[format | reformat | diskinit]]

[-gCLR\_GLIST] [-g]

[-mDISK\_MODEL] [-m[micr1528 | micr1588 | micr1674 | micr1684 | micr1924 | micr2105 | micr2112 | seag12550 | seag15150 | seag32550 | hp2244 | hp2247 | hp3323 | hp3724 | ibms1f | ibms2f | ibms4f | insite325]]

[-snSERIALNO] [-snSTRING]

[-sDISK\_SIZE\_MB] [-sDIGIT]

**DEFAULTS**

*diskformat* -dDISKDEV\_PARAMETER -cformat

**DESCRIPTION**

The *diskformat* command is used to format disks or to initialize a *disk configuration table* on a disk. The command is normally used by the production department to format disk drives. It is not usually necessary to do that in the field.

If formatting a disk, the *diskformat* command will first execute an *SCSI mode select* command to setup the disk drive to DDE standard. After that an *SCSI format command* will be performed to format the disk.

If reformatting a disk, only the *SCSI format command* will be performed. After the format command is complete the *disk configuration table* will be initialized on the disk. All data on the disk will be destroyed when formatting a disk.

Normally, the disk is formatted using *primary defect* and *grown defect*

*lists*, but it is possible to clear the grown defect list during the formatting procedure.

The *disk configuration table* on a disk is used to indicate that the disk drive is supported and formatted by DDE. This feature is used by the *SVR4 operating system*.

The *diskformat* command has the following options:

**-dDISK\_DEVICES**

Same description as the *diskwr* command.

**-cCOMMAND**

This option is used to specify what the *diskformat* command should do. The option must be specified as a *word symbol option*. The following command types can be specified:

**format** This type will format the disk to the DDE standard, and after that the *disk configuration table* will be initialized on the disk.

**reformat** This type will reformat the disk, and after that the *disk configuration table* will be initialized on the disk.

**diskinit** This type will only initialize the *disk configuration table* on the disk.

**-gCLR\_GLIST**

This option is used to specify that the *grown defect list* should be cleared before formatting the disk. This option has no effect, if the **COMMAND** option is set to **diskinit**.

**-mDISK\_MODEL**

This option is used to specify the disk drive model. If the program recognizes the disk drive model, it is not necessary to select this option, otherwise the option must be selected and specified as a *word symbol option*.

**-snSERIALNO**

This option is used to specify the serial number on the disk drive. If the program has read the serial number it is not necessary to select this option, otherwise the option must be selected and specified as a *string option* of up to 16 characters. The serial number is used when initializing the *disk configuration table* on the disk.

**-sDISK\_SIZE\_MB**

This option is used to specify the physical disk size for the *SVR4 operating system*. If the program recognizes the disk drive model, it is not necessary to select this option. Otherwise the option must be selected and specified as a *digit value option*. The physical disk size is used when initializing the *disk configuration table* on the disk.

If no options are given, the command will format and initialize all disk drives selected in the *diskdev parameter*. Please refer to the *set command* to see the detailed description of the *diskdev parameter*.

## EXAMPLES

```
baio 70 -> diskformat -d00,01
Last warning! The disk (seag32550) on scsi 0 id 0
      Disk will now be formatted (y=OK n=CANCEL)? y
Last warning! The disk (hp2247) on scsi 0 id 1
      Disk will now be formatted (y=OK n=CANCEL)? y
Test started at Monday 1995-05-15 - 13:20:46
Format disk on SCSI device: 00
Disk model: seag32550      Command: Format disk without clearing grown list
Serial no.: 00135795      System size: 2047 MB
Test started at Monday 1995-05-15 - 13:51:03
Format disk on SCSI device: 01
Disk model: hp2247        Command: Format disk without clearing grown list
Serial no.: 3404A80096    System size: 1000 MB
baio 70 ->
```

Figure 11-diskformat-1: Example of using diskformat to format disks

## NOTICES

The command cannot be executed while running in *fatal error mode*.

The *sdconf* command can be used to see if the disk model or the serial number are recognized by the program.

If selecting the **DISK\_MODEL**, **SERIALNO** or **DISK\_SIZE\_MB** option, it is only possible to format one disk, otherwise the command will return the error code *Options error*.

The *disk configuration table* will not be created on a removable disk.

## WARNINGS

Before the command executes a format/reformat command or initializes a disk, it displays a *confirm message*. If this is confirmed with a  and the *diskformat* command executes a format/reformat command, the data on the disk will be destroyed.



## BUGS

The *diskformat* command will only be able to format one disk at a time, because the command is not able to use the *SCSI disconnect/reconnect feature*.

It is recommended to read (*sdconf* command) the configuration from an *SCSI device* again, if you change the removable medium in the disk drive. If this is not done, the next command to the *SCSI device* will fail with *Unit attention sense key error*.

## REFERENCES

See also commands:

*set, sdconf, diskrd, diskwr, diskexer, diskrbk*

**diskformat (disk)**

**baio30x command**

**diskformat (disk)**

*This page is intentionally left blank*

**NAME**

*diskrblk* – Reassign block or display defect list on disk

**SYNOPSIS**

*diskrblk* [-dDISK\_DEVICE] [-rREASSIGN\_BLK] [-gDISP\_GLIST] [-pDISP\_PLIST] [-bDISP\_BLKFORMAT]

[-dDISK\_DEVICE]                   [-d[00 | 01 | 02 | 03 | 04 | 05 | 06 | 10  
| 11 | 12 | 13 | 14 | 15 | 17]]

[-rREASSIGN\_BLK]               [-rDIGIT]

[-gDISP\_GLIST]               [-g]

[-pDISP\_PLIST]               [-p]

[-bDISP\_BLKFORMAT]       [-b]

**DESCRIPTION**

This command is used to display the primary or grown defect list from a disk drive or it can be used to reassign a block on the disk drive.

The disk drive is able to handle media defects, and it has two different sets of defect information:

- *Primary defect list.* This list is supplied by the manufacturer and is resident on the drive.
- *Grown defect list.* This list contains defects which have been identified to the drive using the reassign block command.

These defect lists can be displayed by means of the **DISP\_GLIST** or **DISP\_PLIST** options.

If a hard error occurs during a read or write operation, you must use the *diskrblk* command to reassign the defective block address.

The *diskrblk* command requests the disk drive to reassign a defective block to a spare block reserved for this purpose. If a spare block is available on the track, where the defective block is located, this block will be used, otherwise a spare block on another track will be used. The address of the defective block is added to the *grown defect list*. It is **NOT** necessary to format the drive after using the reassign block command.

The *diskrblk* command has the following options:

**-dDISK\_DEVICE**

Same description as the *diskwr* command. But only one disk drive can be specified, because the option is a *word symbol option*. If the option is not selected, the first disk drive in the *diskdev* parameter will be selected. Please refer to the *set* command to see the detailed description of the *diskdev* parameter.

**-rREASSIGN\_BLK**

When this option is selected the command reassigns the specified block address on the *SCSI device*. This option overrules all other options.

**-gDISP\_GLIST**

If this option is selected the *grown defect list* is displayed.

**-pDISP\_PLIST**

If this option is selected the *primary defect list* is displayed.

**-bDISP\_BLKFORMAT**

If this option is selected, and the **DISP\_GLIST** or **DISP\_PLIST** option is selected too, the defect list is displayed in block format, otherwise it is displayed in sector, head and cylinder format.

If one of the **REASSIGN\_BLK**, **DISP\_GLIST** or **DISP\_PLIST** options is not selected, the command has no effect.

## EXAMPLES

```
baio 70 -> diskrblk -d10 -r10000
Last warning! The disk (hp2247) on scsi 1 id 0
          Data block 0x2710 will now be reassigned (y=OK n=CANCEL)? y
Test started at Monday 1995-05-15 - 15:46:24
Reassign block on disk on SCSI device: 10
Block address: 0x00002710
baio 70 ->
```

Figure 11-diskrblk-1: Example of using the diskrblk to reassign a block on a disk

```
baio 70 -> diskrblk -d10 -g
Test started at Monday 1995-05-15 - 15:46:38
Display defect on a disk on SCSI device: 10
Display grown defect list on sector format
Defect list no: 1   Cylinder: 17   Head: 4
Number of defects: 1
baio 70 ->
```

Figure 11-diskrblk-2: Example of using the diskrblk to display defect list on a disk

## NOTICES

The command cannot be executed while running in *fatal error mode*.

The program can fail with *Recovered error sense key error*, while displaying the *primary or grown defect list* on block format from a disk drive. The reason is that the disk drive does not support block format. Try to display on sector format instead by leaving out the `DISP_BLKFORMAT` option.

It is not possible to reassign a block on removeable disk.

diskrblk (disk)

baio30x command

diskrblk (disk)

If you use this command to display the defect list, it overrules the setting of the *dispmode* parameter.

**WARNINGS**

If you are using the command to reassign a block on a disk, a *confirm* message is displayed before destroying data in the block.

**REFERENCES**

See also commands:

*set, sdconf, diskrd, diskwr, diskexer, diskformat*

## NAME

*diskrd* – Disk read (CRC) test

## SYNOPSIS

*diskrd* [-dDISK\_DEVICES] [-bBLK\_PER\_CMD] [-sBLK\_TRANSFER] [-aACC\_MODE] [-mdMEM\_DEVICE] [-mrMEM\_RANGE]

[-dDISK\_DEVICES] [-d[00 | 01 | 02 | 03 | 04 | 05 | 06 | 10 | 11 | 12 | 13 | 14 | 15 | 17][,\*,\*]]

[-bBLK\_RANGE] [-b[START | #SIZE | START-END | START#SIZE]]

[-sBLK\_PER\_CMD] [-s[START | #SIZE | START-END | START#SIZE]]

[-aACC\_MODE] [-a[seqin | seqout | seek | random]]

[-mdMEM\_DEVICE] [-md[pmem | global]]

[-mrMEM\_RANGE] [-mr[START | #SIZE | START-END | START#SIZE]]

## DEFAULTS

*diskrd* -dDISKDEV\_PARAMETER -b\*-\* -s32 -aseqin -mdpmem

## DESCRIPTION

This command tests disk drive(s). This is done by reading data from the disk drive to the memory. The test does not check the data, but the disk drive performs a CRC (Cyclic Redundancy Check) check while the reading the data.

The *diskrd* command has the following options:

### -dDISK\_DEVICES

Same description as the *diskwr* command.

### -bBLK\_RANGE

Same description as the *diskwr* command.

### -sBLK\_PER\_CMD

Same description as the *diskwr* command.

diskrd (disk)

baio30x command

diskrd (disk)

**-aACC\_MODE**

Same description as the *diskwr* command.

**-mdMEM\_DEVICE**

Same description as the *diskwr* command.

**-mrMEM\_RANGE**

Same description as the *diskwr* command, but the calculation of the necessary number of bytes is as follows:

(disk\_numbers \* block\_per\_cmd \* block\_size) aligned to 0x10000.

If no options are given, the *diskrd* command will run a CRC test from the first to the last block address on all disk drives selected in the *diskdev* parameter. Please refer to the *set* command to see the detailed description of the *diskdev* parameter.

**EXAMPLES**

The example below shows how the *diskrd* command is used to run a seek read test from block address 0x186a0 to 0x286a0 on the disk drives mounted on SCSI interface 0 id's 0 and 1. The number of blocks per SCSI command will be random between 4 and 32. The data from the disk drives are stored in *global memory* from address 0x2a0000 to 0x2c0000.

```

baio 70 -> diskrd -d00,01 -aseek -s4-32 -b100000#0x10000 -mdglobal
Test started at Friday 1995.05.12 - 11:45:14
Disk read (CRC) test on SCSI device: 00 01
Block address range  Blocks  Access Memory range      Device
0x000186a0-0x000286a0 4 -32  seek  0x002a0000-0x002c0000 global
baio 70 ->

```

Figure 11-diskrd-1: Example of using the *diskrd* command to run a seek test



## NOTICES

The command cannot be executed while running in *fatal error mode*.

The maximum number of blocks per *SCSI command* is up to 16 Mbyte or the size of available memory.

If the number of blocks to be tested is not divisible by the blocks per *SCSI read command*, the number of blocks per *SCSI command* will be reduced for the last access to disk drive.

If more than one disk is tested at the same time, the start block address must be specified as a smaller number than the number of blocks of the smallest disk. If the end block address is specified as a larger number than the number of blocks of the smallest disk, the test will use the specified number of block addresses on the larger disks and, when testing the smallest disk, it will only use the existing number of block addresses on this disk.

## COMMAND ERRORS

If the command returns the *Options error* error code, it is because the memory range is not aligned on 4 bytes addresses or because the specified memory range is too small.

## REFERENCES

See also commands:

*set, sdconf, sddisp, diskwr, diskexer, diskrbk, diskformat*

**diskrd (disk)**

**baio30x command**

**diskrd (disk)**

*This page is intentionally left blank*

**NAME**

*diskwr* – Disk write/read test

**SYNOPSIS**

*diskwr* [-dDISK\_DEVICES] [-bBLK\_PER\_CMD] [-sBLK\_TRANSFER] [-adDUMMY\_ACC] [-aACC\_MODE] [-tTYPE] [-pPATTERN] [-vPAT\_VALUE] [-mdMEM\_DEVICE] [-mrMEM\_RANGE]

[-dDISK\_DEVICES] [-d[00 | 01 | 02 | 03 | 04 | 05 | 06 | 10 | 11 | 12 | 13 | 14 | 15 | 17][,\*,,]]

[-bBLK\_RANGE] [-b[START | #SIZE | START-END | START#SIZE]]

[-sBLK\_PER\_CMD] [-s[START | #SIZE | START-END | START#SIZE]]

[-adDUMMY\_ACC] [-ad[off | first | last | seqin | seqout | seek | random]]

[-aACC\_MODE] [-a[seqin | seqout | seek | random]]

[-tTYPE] [-t[wrcmp | cmp | write | read | check]]

[-pPATTERN] [-p[random | value | count | cntdown | current]]

[-vPAT\_VALUE] [-vDIGIT]

[-mdMEM\_DEVICE] [-md[pmem | global]]

[-mrMEM\_RANGE] [-mr[START | #SIZE | START-END | START#SIZE]]

**DEFAULTS**

*diskwr* -dDISKDEV\_PARAMETER -b\*-\* -s32 -adoff -aseqin -twrcmp -prandom -vRANDOM -mdpmem

**DESCRIPTION**

This command tests disk drive(s). This is done by writing and reading data from a disk drive to the memory while checking the data.

It is possible to select different types of tests using the **TYPE** option. The following tests can be performed:

- Read data from a disk without checking data.
- Read data from a disk while checking data.
- Write data to a disk.
- Write and read data to a disk while checking data.
- Write data to a disk, and read and check the written data afterwards.

It is possible to specify different data patterns to be written on the disks. This is done by means of the **PATTERN** and **PAT\_VALUE** options.

It is also possible to specify that a dummy read access shall be performed before the real write or read access is performed. The dummy read access will read one block of data without checking the data.

It is not possible to read or write from blocks located before the 4th block on a disk (this does not apply to removable disks). This is because the *disk configuration table* must not be destroyed on the harddisk.

The *diskwr* command has the following options:

#### **-dDISK\_DEVICES**

This option is used to select which disk drive(s) should be used. The option must be specified as a *multi word symbol option*. The *sdconf* command must read the configuration from a disk drive, before the disk drive is available in the word symbol list. If the command is executed without this option, it selects the disk drive(s) specified in the *diskdev parameter*. Please refer to the *set* command to see the detailed description of the *diskdev parameter*.

**-bBLK\_RANGE**

This option is used to specify a block address range where the write or read access shall be done on the disk. The option must be specified as a *value range option* and the number blocks tested on a disk drive is the size of the range.

**-sBLK\_PER\_CMD**

This option is used to specify the number of blocks per *SCSI write/read command* during the test. It is also possible to specify a block transfer range. If this is done the number of blocks per access is calculated on a pseudo random basis for every access to the *SCSI device*. The option must be specified as a *value range option*.

**-adDUMMY\_ACC**

This option is used to specify how and where a dummy read access shall be performed in the block address range on the disk drive. If the option is not selected no dummy access will be performed. The option must be specified as a *word symbol option*, and the following dummy read access modes can be specified:

- off** No dummy access will be performed.
- first** A dummy access will be performed on the first block address on the disk.
- last** A dummy access will be performed on the last block address on the disk.
- seqin** A dummy access will be performed in the block address range from the outer to the inner track on the disk drive.
- seqout** A dummy access will be performed in the block address range from the inner to the outer track on the disk drive.
- seek** A dummy access will be performed in the block address range from the outer to the inner track and from the inner to the outer track (and so on) on the disk drive.

diskwr (disk)

baio30x command

diskwr (disk)

**random** A dummy access will be performed random in the block address range. For every dummy access the block address is calculated on a pseudo random basis.

**-aACC\_MODE**

This option is used to specify how and where to perform the write or read access in the block address range on the disk drive. The option must be specified as a *word symbol option*, and the following access modes can be specified:

**seqin** The write/read access will be performed in the block address range from the outer to the inner track on the disk drive.

**seqout** The write/read access will be performed in the block address range from the inner to the outer track on the disk drive.

**seek** The write/read access will be performed in the block address range from the outer to the inner track and from the inner to the outer track (and so on) on the disk drive.

**random** The write/read access will be performed random in the block address range. For every write/read access, the block address is calculated on a pseudo random basis.

**-tTYPE**

This option is used to specify which test the command shall perform. The option must be specified as a *word symbol option* and the following types of test can be specified:

**wrcmp** The test will perform a write access and a read access to the same disk blocks and the data will be checked. This is done for all blocks in the specified block address range on the disk drives. If a dummy read access is selected, it will be performed between the write and the read access.

- cmp** The test will first perform write accesses to the specified block address range on the disk drives. After that it will perform read accesses and check the data.
- write** The test will perform write accesses to the specified block address range on the disk drives.
- read** The test will perform read accesses to the specified block address range on the disk drives without checking the data.
- check** The test will perform read accesses to the specified block address range on the disk drives and check the data.

**-pPATTERN**

This option is used to specify a data pattern type together with the **PAT\_VALUE** option. The data pattern is used to generate data to the disk or to compare data from the disk. If the **TYPE** option is set to **read**, this option has no effect. The **PATTERN** option must be specified as a *word symbol option*, and the following pattern types can be specified:

- random** This pattern type will generate a 32 bit random data pattern from the **PAT\_VALUE**. If the same **PAT\_VALUE** is specified, the same random pattern is generated.
- value** This pattern type will generate a 32 bit data value specified with the **PAT\_VALUE** option.
- count** This pattern type will generate a 32 bit count data pattern calculated as follows:  
 $\text{PAT\_VALUE} + (\text{block address} * \text{block size}) + (\text{offset} / 4)$ .
- cntdown** This pattern type will generate a 32 bit count-down data pattern calculated as follows:  
 $\text{PAT\_VALUE} + (\text{block address} * \text{block size}) - (\text{offset} / 4)$ .

**current** This pattern type will use the current values in the memory address range.

#### **-vPAT\_VALUE**

This option is used to specify a 32 bit data value for the pattern type. The option must be specified as a *digit value option*. If not specified, the default value is a value from the internal system clock. If the **PATTERN** option is set to **current** or the **TYPE** option is set to **read**, this option has no effect.

#### **-mdMEM\_DEVICE**

This option is used to select which memory the test has to use. The option must be specified as a *word symbol option*. The following memories can be specified:

**pmem** Program memory.

**global** Global memory.

#### **-mrMEM\_RANGE**

This option is used to specify the memory address range where the test has to write/read data to/from the disk drives. The option must be specified as a *value range option*. Default, the command will reserve the next free memory array with the necessary number of bytes, calculated as follows:

(*disk\_numbers \* block\_per\_cmd \* block\_size \* 2*) aligned to 0x10000.



## EXAMPLES

```

baio 70 -> diskwr -tcmp -adrandom -mdglobal -pcount -v0
Last warning! The disk (seag32550) on scsi 0 id 0
          Data will now be destroyed (y=OK n=CANCEL)? y
Test started at Wednesday 1995.05.24 - 12:25:20
Disk write/read test on SCSI device: 00
Block address range  Blocks  Access Dummy  Memory range  Device Pattern
0x00000004-0x00216481 32      seqin  random  0x02500000-  global count
                               cmp          0x02510000      0x00000000
baio 70 ->

```

Figure 11-diskwr-1: Example of the diskwr command

## NOTICES

The command cannot be executed while running in *fatal error mode*.

The maximum number of blocks per *SCSI command* is the size of available memory or up to 16 Mbyte.

If the number of blocks to be tested is not divisible by the blocks per *SCSI read command*, the number of blocks per *SCSI command* will be reduced on the last access to disk drive.

If more than one disk is tested at the same time, the start block address must be specified as a smaller number than the number of blocks of the smallest disk. If the end block address is specified as a larger number than the number of blocks of the smallest disk, the test will use the specified number of block addresses on the larger disks and, when testing the smallest disk, it will only use the existing number of blocks on this disk. It is not possible to read or write the first 1024 bytes on a disk.

diskwr (disk)

baio30x command

diskwr (disk)

## WARNINGS

If setting the **TYPE** option to **check**, you must be sure that the compared data has been written on the disk, else the program will find incorrect data errors on the disks.

If you are using the command to perform write test on the disk drive, a confirm message is displayed before writing on the disk. If this is confirmed with a , the data on the disk will be destroyed.

## COMMAND ERRORS

If the command returns the error code *Options error*, it is because the memory range is not aligned on 4 bytes addresses or because the specified memory range is too small.

## REFERENCES

See also commands:

*set, sdconf, sddisp, diskrd, diskrblk, diskformat*

## NAME

*errlog* – Display errors from the error log

## SYNOPSIS

*errlog* [-nNUMBER] [-old]

[-nNUMBER] [-nDIGIT]

[-old] [-old]

## DESCRIPTION

The *errlog* command is used to display error information which is stored in the *error log* buffer. The *error log* buffer is cleared every time the *command line* buffer contains a *test command* and the *command interpreter* starts executing the first command from the *command line* buffer.

The maximum number of errors in the *error log* buffer, which has not been displayed yet, is 64. If this number is exceeded an overrun is detected, and the first 64 errors are saved, and the last arrived is discarded.

The *errlog* command has the following options:

### -nNUMBER

This option is used to display a number of the latest errors which are stored in the *error log* buffer. The option must be specified as a *digit value option*.

### -old

This option is used to display error information which is stored in the *error log* buffer, whether it has been displayed before or not.

If no options are given, the command displays all error information from the *error log* which has not been displayed before.

## EXAMPLES

In the example on the next page, the *errlog* command has been executed. It displays all error information from the *error log* buffer which has not been displayed before. If you execute the command again, it will not display any error information.

errlog (system)

baio30x command

errlog (system)

```
baio 70 -> errlog
Error occurred at Friday 1995-03-17 - 14:51:29
Data error during SCSI DMA transfer on scsi0 Pass number: 1
Source address: 0x00760a4c <global> Write data: 0x179097c4
Destination address: 0x00290a4c <pmem> Read data: 0x75097804

Error occurred at Friday 1995-03-17 - 14:51:30
Data error during SCSI DMA transfer on scsi0 Pass number: 1
Source address: 0x00760a50 <global> Write data: 0x17909750
Destination address: 0x00290a50 <pmem> Read data: 0x75097800
Number of errors: 2
Number of overruns: 0
baio 70 -> errlog
Number of errors: 2
Number of overruns: 0
baio 70 ->
```

Figure 11-errlog-1: Example of using the errlog command

The example below shows how to display only the latest error information from the *error log* buffer.

```
baio 70 -> errlog -old -n1
Error occurred at Friday 1995-03-17 - 14:51:30
Data error during SCSI DMA transfer on scsi0 Pass number: 1
Source address: 0x00760a50 <global> Write data: 0x17909750
Destination address: 0x00290a50 <pmem> Read data: 0x75097800
Number of errors: 2
Number of overruns: 0
baio 70 ->
```

Figure 11-errlog-2: Example of using the errlog command to display the latest error

## NOTICES

The command does not clear *test log* and *error log*.

The command cannot be repeated. It will only run once.

The error information can also be displayed when the error has occurred. This depends on how the *dispmode* parameter is set.

## REFERENCES

See also commands:

*set, status, testlog*

**errlog (system)**

**baio30x command**

**errlog (system)**

*This page is intentionally left blank*

**NAME**

*ethdiag* – Ethernet diagnose test

**SYNOPSIS**

*ethdiag* [-dDEVICE] [-rADDR\_RANGE] [-nNO\_OF\_TEST]

[-dDEVICE]                    [-d[main]]

[-rADDR\_RANGE]            [-r[START | #SIZE | START-END |  
START#SIZE]]

[-nNO\_OF\_TEST]            [-nDIGIT]

**DEFAULTS**

*ethdiag* -dmain -r#0x100000

**DESCRIPTION**

The *ethdiag* command performs a simple diagnostics test on an *ethernet controller*. The test will be performed as follows:

- The command initializes 16 bytes in the *data memory*.
- The command performs a diagnose test on the *ethernet controller*. The test writes in a certain register in the controller. The controller writes the results in the *data memory* where the memory has been initialized.
- The command checks the result in the *data memory*.
- The command repeats this sequence for every 16 bytes in the selected memory array.

The *ethdiag* command has the following options:

**-dDEVICE**

Same description as the *ethctest* command.

**-rADDR\_RANGE**

Same description as the *ethctest* command.

### **-nNO\_OF\_TEST**

This option is used to specify how many times to repeat a diagnose test. It must be specified as a *digit value option*. Default number of tests depends on how many bytes are reserved for the test, calculated as follows: Size of the memory array / 16.

If no options are given, the *ethdiag* command performs 65536 diagnostics tests on the **main ethernet**.

### **EXAMPLES**

```
baio 70 -> ethdiag -r#256 -n0x1000
Test started at Friday 1995-05-26 - 14:24:21
Ethernet diagnose test on Main ethernet
Memory address range  No of test
0x00000000-0x00000100  4096
baio 70 ->
```

Figure 11-ethdiag-1: Example of the ethdiag command

### **NOTICES**

The command cannot be executed while running in *fatal error mode*.

If the specified length of the memory address range is smaller than 16 bytes aligned to a 16 bytes address, no test will be performed.

### **COMMAND ERRORS**

If the command returns the error code *Options error*, then it is because the memory range is not aligned to a 4 bytes address.

### **REFERENCES**

See also commands:  
*ethernet, ethmon, ethtest*



## NAME

*ethernet* – Test script for ethernet

## SYNOPSIS

*ethernet* [-dETHERNET\_DEVICES] [-tTEST\_FUNCTIONS] [-pDISP\_CMDS]

[-dETHERNET\_DEVICES] [-d[main] [,\*,\*]]

[-tTEST\_FUNCTIONS] [-t[fast | function] [,\*,\*]]

[-pDISP\_CMDS] [-p]

## DEFAULTS

*ethernet* -dmain -tfast,function

## DESCRIPTION

This command is a *script command*, and it is used to run a complete test of the ethernet circuit that is located on the *baio30x* module. The command executes other commands which all are tests of the *ethernet controller* and the internal network interface. It is possible to specify which ethernet to be tested and which commands of the *ethernet script* to be executed. This is done by means of the **ETHERNET\_DEVICES** and **TEST\_FUNCTIONS** options.

The *ethernet* command has the following options:

### -dETHERNET\_DEVICES

This option is used to select which ethernet circuit to be tested. The option must be specified as a *multi word symbol option*, and only one *ethernet controller* can be specified at the moment:

**main** The *main ethernet* that is located on the *baio30x* module.

**-tTEST\_FUNCTIONS**

This option is used to specify which test functions should be selected from the *ethernet script*. The option must be specified as a *multi word symbol option* and the following test functions can be specified:

- fast**            If this test function is selected, the *ethernet controller* will only be tested fast and not thoroughly.
- function**       If this test function is selected, the *ethernet controller* and internal network interface will be tested thoroughly.

**-pDISP\_CMDS**

If this option is specified, the selected commands from the script will only be displayed, not executed.

Without option the command executes all the commands from the scripts.

**EXAMPLES**

```

baio 70 -> ethernet -p
Test script for ethernet
Device Function Command script
main fast ethtest -dmain
main function ethdiag -dmain -r#0x1000; ethtest -dmain -r*-*; ethtest
-dmain -f1024
baio 70 ->
    
```

Figure 11-ethernet-1: Example of ethernet command displaying all commands from the script

ethernet (ethernet)

baio30x command

ethernet (ethernet)

```
baio 70 -> ethernet -tfast -p
Test script for ethernet
Device  Function  Command script
main    fast    ethtest -omain
baio 70 ->
```

Figure 11-ethernet-2: Example of ethernet command selecting a few commands from the scrip

## NOTICES

The command cannot be used while running in *fatal error mode*.

## REFERENCES

See also commands:

*ethdiag*, *ethtest*, *ethstat*.

**ethernet (ethernet)**

**baio30x command**

**ethernet (ethernet)**

*This page is intentionally left blank*

## NAME

*ethmon* – Ethernet network monitor

## SYNOPSIS

```
ethmon [-dDEVICE] [-rADDR_RANGE] [-nNO_OF_FRM] [-fFRAME_SIZE] [-iINTERFACE] [-tTYPE] [-pPATTERN] [-vPAT_VALUE]
```

[-dDEVICE]	[-d[main]]
[-rADDR_RANGE]	[-r[START   #SIZE   START-END   START#SIZE]]
[-nNO_OF_FRM]	[-nDIGIT]
[-fMAX_FRAME_SIZE]	[-fDIGIT]
[-iINTERFACE]	[-i[module   intern   aui   thin   tpe]]
[-tTYPE]	[-t[non   all   error   netadd   srcadd   destadd   erradd]]
[-aNETWORK_ADDR]	[-vDIGIT]

## DEFAULTS

```
ethmon -dmain -r#0x100000 -n0x1000 -f1518 -iaui -mnon -a0
```

## DESCRIPTION

The *ethmon* command is used to monitor an ethernet network. This is done by configuring the *ethernet controller* to receive all frames on the network.

The CPU and the *ethernet controller* communicate by using the *data memory* to exchange commands, status information, and descriptors in order to enable the controller to receive and transmit frames on a network. The number of command and receive descriptor buffers in the memory is depending on the size of the memory array and the maximum number of data bytes per frame, specified with the *MAX\_FRAME\_SIZE* option.

It is possible to select a different type of frame monitoring. This can be used to specify a specific frame from the network and save the frame information in the *error log* buffer. This is done by means of the **TYPE** and **NETWORK\_ADD** options. The following frame information will be saved:

- Frame source and destination network address.
- Frame length.
- Monitor frame number.
- Ethernet *RFD* (Receive frame descriptor) memory address and *RFD* status.

While the command is running the receive statistics and ethernet status will be updated every time a frame is received. This information can be displayed by means of the *ethstat* command.

The *ethmon* command has the following options:

**-dDEVICE**

Same description as the *ethtest* command.

**-rADDR\_RANGE**

Same description as the *ethtest* command.

**-nNO\_OF\_FRM**

This option is used to specify how many ethernet frames to be received. It must be specified as a *digit value option*. Default, the command will receive 65536 frames.

**-fMAX\_FRAME\_SIZE**

This option is used to specify the maximum number of data bytes per frame the command shall reserve for data in the receive buffer. It must be specified as a *digit value option*, and the number of bytes must be in the range of 8 to 1518.

**-iINTERFACE**

Same description as the *ethtest* command. The **module** and **intern** network interface cannot be used in this test.

**-tTYPE**

This option is used to specify which type of monitoring the command shall perform. I.e. which of the received frames the command shall store in the *error log* buffer. The option must be specified as a *word symbol option*, and the following types can be specified:

- non**           The command will not store any frames in the *error log* buffer.
- all**            The command will store all frames in the *error log* buffer.
- error**          The command will store all frames with error in the *error log* buffer.
- netadd**        The command will store a frame in the *error log* buffer, if the frame has same source or destination network address as on the specified by the **NETWORK\_ADDR** option.
- srcadd**        The command will store a frame in the *error log* buffer, if the frame has same source network address as on the specified by the **NETWORK\_ADDR** option.
- destadd**       The command will store a frame in the *error log* buffer, if the frame has same destination network address as on the specified by the **NETWORK\_ADDR** option.
- erradd**        The command will store a frame in the *error log* buffer, if the frame failed and has same source or destination network address as on the specified by the **NETWORK\_ADDR** option.

**-aNETWORK\_ADDR**

This option is used to specify the lower 24 bits of the source or destination network address. The option must be specified as a *digit value option*. If the **TYPE** option is set to **non**, **all** or **error**, this option has no effect. Default the network address is zero.

**EXAMPLES**

```

baio 70 -> ethmon
Test started at Monday 1995-06-19 - 11:56:38
Ethernet monitor test on Main ethernet
Memory address range No of frame Frm size Network Type Network_add
0x00100000-0x00200000 65536 1518 aui non
baio 70 ->

```

Figure 11-ethmon-1: Example of the ethmon command

**NOTICES**

The command cannot be executed while running in *fatal error mode*.

If the specified length of the memory address range is smaller than 8192 bytes aligned to a 16 bytes address, no monitor test will be performed.

**COMMAND ERRORS**

If the command returns the error code *Options error*, it is because the memory range is not aligned on a 16 bytes address.

The program can fail with *Time-out during wait for receiving a frame error*, while the command is running. The reason is either that the selected network interface is not connected to the network or that the **INTERFACE** option is set to **module** or **intern**.



ethmon (ethernet)

baio30x command

ethmon (ethernet)

**BUGS**

If there are heavy load conditions on the network, and the ethernet frame information from the *errlog* is displayed on the terminal while the command is running, the program will not be able to maintain monitoring of the network. Instead, the program will terminate with the *Ethernet receiver unit failed* error.

**REFERENCES**

See also commands:  
*ethdiag, ethstat, ethtest*

**ethmon (ethernet)**

**baio30x command**

**ethmon (ethernet)**

*This page is intentionally left blank*

## NAME

*ethstat* – Display ethernet network statistics

## SYNOPSIS

*ethstat* [-dDEVICE] [-clrCLR\_STAT] [-tSTAT\_TYPE]

[-dDEVICE]            [-d[main]]

[-clrCLR\_STAT]       [-clr]

[-tSTAT\_TYPE]       [-t[rec | transmit | stat][\*,\*]]

## DEFAULTS

*ethstat* -dmain -trec,transmit,stat

## DESCRIPTION

The *ethstat* command is used to display various information about ethernet network statistics and status. This information is updated every time the *ethstest* (ethernet transmission test) or *ethmon* command (Ethernet network monitor) is executed. It is possible to clear the statistics with the *CLR\_STAT* option. The following information can be displayed:

- Network number on the ethernet.
- Receive statistics.
- Transmit statistics.
- Ethernet status.

The following receive statistics are available:

- Number of received OK frames.
- Number of received bytes.
- Number of received bad frames.
- Number of alignment errors.
- Number of CRC errors.
- Number of overrun errors.
- Number of collisions detected while receiving frames.

The following transmit statistics are available:

- Number of transmitted OK frames.
- Number of transmitted bytes.
- Number of transmitted bad frames.
- Number of heartbeats missing.
- Number of defers to traffic.
- Number of collisions.
- Distribution of number of collisions.

The following ethernet status are available:

- Current number of buffers/bytes on free command descriptor buffers.
- Current number of buffers/bytes on used command descriptor buffers.
- Current number of buffers/bytes on free receive descriptor buffers.
- Current number of buffers/bytes on used receive descriptor buffers.

The *ethstat* command has the following options:

**-dDEVICE**

Same description as the *ethtest* command.

**-clrCLR\_STAT**

If this option is selected, all the ethernet statistics will be cleared. If you want to display the statistics before the statistics is cleared, the **STAT\_TYPE** option must be selected too.

**-tSTAT\_TYPE**

This option is used to specify which ethernet information should be displayed. The option must be specified as a *multi word symbol option*. Default, the command displays all ethernet information from the ethernet, unless the **CLR\_STAT** option is selected. The following ethernet information can be specified:

- rec**            The receive statistics will be displayed.
- transmit**     The transmit statistics will be displayed.
- stat**           The ethernet status will be displayed.

The network number on the ethernet will only be displayed, if the statistics or status are displayed.

If no options are given, the *ethstat* command will display all statistics and status information from the *main ethernet*.

**EXAMPLES**

```

baio 70 -> ethstat
Network number on Main ethernet is 080075a20008
Receive   Receive   Receive   Alignment  CRC           Overrun   Collisions
OK frames bytes    bad frames errors  errors      errors    detected
0x00001000 0x00100000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
Transmit  Transmit  Transmit  Heartbeat  Defer to    Number of
OK frames bytes    bad frames missing traffic      Collisions
0x00001000 0x00100000 0x00000000 0x00000000 0x000000ff 0x00000000
Distribution of number of collisions:
 1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
00003 00001 00000 00001 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
Command free      Command used      Receive free      Receive used
Buffer Bytes     Buffer Bytes      Buffer Bytes      Buffer Bytes
0x0006e9 0x0006e900 0x000000 0x00000000 0x0006eb 0x0006eb00 0x000000 0x00000000
baio 70 ->
    
```

Figure 11-ethstat-1: Example of the ethstat command

**ethstat (ethernet)****baio30x command****ethstat (ethernet)****NOTICES**

The command does not clear *test log* and *error log*.

If the command is used to display ethernet information, the information will always be displayed, no matter how the *dispmode parameter* is set.

**REFERENCES**

See also commands:

*ethernet, ethtest, ethmon*

**NAME**

*ethtest* – Ethernet transmission test

**SYNOPSIS**

*ethtest* [-dDEVICE] [-rADDR\_RANGE] [-nNO\_OF\_FRM] [-fFRAME\_SIZE] [-iINTERFACE] [-tTYPE] [-pPATTERN] [-vPAT\_VALUE]

[-dDEVICE]                    [-d[main]]

[-rADDR\_RANGE]            [-r[START | #SIZE | START-END | START#SIZE]]

[-nNO\_OF\_FRM]            [-nDIGIT]

[-fFRAME\_SIZE]           [-fDIGIT]

[-iINTERFACE]           [-i[module | intern | aui | thin | tpe]]

[-tTYPE]                   [-t[compare | loop | send | tdrCmd]]

[-pPATTERN]              [-p[random | value | count | cntdown | current]]

[-vPAT\_VALUE]            [-vDIGIT]

**DEFAULTS**

*ethtest* -dmain -r#0x100000 -n0x1000 -f256 -imodule -tcompare -prandom -vX

**DESCRIPTION**

The *ethtest* command is used to perform an ethernet transmission from an *ethernet controller* to a network interface. This is done by transmitting an ethernet frame to the network interface and then making it loop back to the *ethernet controller*.

The *CPU* and the *ethernet controller* communicate via the *data memory* to exchange commands, status information, and descriptors in order to enable the controller to receive and transmit frames on a network. The number of command and transmit/receive descriptor buffers in the memory is depending on the size of the memory array and the number of data bytes per frame, specified by the **FRAME\_SIZE** option.

The network interface, which is a LAN (Local Area Network) interface, is connected to the *ethernet controller* through a *manchester encoder/decoder*. The network interface can be treated as either an internal or external network. This is determined by means of the **INTERFACE** option.

It is possible to select different types of tests by means of the **TYPE** option. The following tests can be performed:

- Transmit a frame to the network and receive the frame back again to the *ethernet controller*. After each transmitted frame, the received data will be compared to the transmitted data.
- Transmit a frame to the network and receive the frame back again to the *ethernet controller*. The received data will not be checked.
- Transmit frames to the network. The transmitted frames will be checked for correct status.
- Perform a *TDR* (Time Domain Reflectometry) command on the network.

Before transmitting frames or performing a *TDR* command on the network, the command will check the communication between the CPU and the *ethernet controller* by executing the following *ethernet controller* commands: **NOP**, **DIAGNOSE**, **IA-SETUP**, **MC-SETUP** and **CONFIG-SETUP**. If an error occurs, no further testing will be done by the *ethtest* command.

It is possible to set the number of frames to be transmitted and the number of data bytes to be transmitted per frame. This is done by means of the **NO\_OF\_FRM** and **FRAME\_SIZE** options. The data patterns in a transmitted frame can be specified with the **PATTERN** and **PAT\_VALUE** options.

While the command is running the transmit/receive statistics and ethernet status will be updated every time a frame is received. This information can be displayed by the *ethstat* command.



The *ethtest* command has the following options:

#### **-dDEVICE**

This option is used to select which *ethernet controller* the test has to use. The option must be specified as a *word symbol option*. Only one *ethernet controller* can be specified at the moment:

**main**     The *main ethernet* that is located on the *baio30x* module.

#### **-rADDR\_RANGE**

This option is used to specify the memory address range in the *data memory* which the *ethernet controller* has to use. The option must be specified as a *value range option*. Default, the command will reserve the next free memory array with the size of 0x100000 bytes.

#### **-nNO\_OF\_FRM**

This option is used to specify how many times to transmit an ethernet frame. It must be specified as a *digit value option*. Default number of tests is depending on how many bytes are reserved for the test calculated as follows: Size of memory array / frame size.

#### **-fFRAME\_SIZE**

This option is used to specify the number of data bytes in an ethernet frame. It must be specified as a *digit value option*, and number of bytes must be in the range from 8 to 1518.

#### **-iINTERFACE**

This option is used to specify which network interface the *ethernet controller* has to use. The option must be specified as a *word symbol option*, and the following interface types can be specified:

**module**     Internal network, loop-back in the *manchester encoder/decoder*.

**intern**     Internal network, loop-back in the *ethernet controller*.

**au**             External network, based on *10Base5 AUI* (Ethernet) interface. The ethernet network cable must be connected to the *au* interface on the ethernet, before running this test.

ethctest (ethernet)

baio30x command

ethctest (ethernet)

- thin** External network, based on *10Base2 AUI* (Cheaper-net) interface. The cheaper-net network cable must be connected to the *thin interface* on the ethernet, before running this test.
- tpe** External network, based on *10Base-T* (Twisted Pair Ethernet) interface. The twisted pair network cable must be connected to the *tpe interface* on the ethernet, before running this test.

**-tTYPE**

This option is used to specify which test the command shall perform. The option must be specified as a *word symbol option*, and the following types of test can be specified:

- compare** This test will make the *ethernet controller* transmit the data to the network and make the data loop back to the receiver in the controller. The transmitted and received frame will be checked for correct status, and the received data will be compared with the transmitted data.
- loop** This test will make the *ethernet controller* transmit the data to the network and make the data loop back to the receiver in the controller. The transmitted and received frame will be checked for correct status, but the received data will not be checked.
- send** This test will make the *ethernet controller* only transmit data to the network. The receiver in the controller will be disabled in this test.
- tdrcmd** This test will make the *ethernet controller* run a *TDR* (Time Domain Reflectometry) command on the network. This is a mechanism to detect open or short circuits on the network and to measure their distance from the diagnosing station. When this test mode is used, the options **PATTERN**, **PAT\_VALUE**, and **FRAME\_SIZE** options have no effect.

**-pPATTERN**

This option is used to specify a data pattern type together with the **PAT\_VALUE** option. The **PATTERN** option must be specified as a *word symbol option*, and the following pattern types can be specified:

- random** This pattern type will generate a 32 bit random data pattern from the **PAT\_VALUE**. If the same **PAT\_VALUE** is specified, the same random pattern is generated.
- value** This pattern type will generate a 32 bit data value specified with the **PAT\_VALUE** option.
- count** This pattern type will generate a 32 bit count data pattern calculated as follows:  $\text{PAT\_VALUE} + (\text{offset} / 4)$ .
- cntdown** This pattern type will generate a 32 bit count-down data pattern calculated as follows:  $\text{PAT\_VALUE} - (\text{offset} / 4)$ .
- current** This pattern type will use the current values in the memory address range.

**-vPAT\_VALUE**

This option is used to specify a data value for the pattern type. The option must be specified as a *digit value option*. Default, the value is a value from the internal system clock. The option only has effect, if the **PATTERN** option is not set to **current**.

## EXAMPLES

```
baio 70 -> ethctest -iaui -f1024 -n1024
Test started at Friday 1995-05-26 - 14:24:40
Ethernet transmission test on Main ethernet
Memory address range No of frame Frm-size Network Type Pattern Value
0x00000000-0x00100000 1024 1024 aui compare random 0x2fc5e4a8
baio 70 ->
```

Figure 11-ethctest-1: Example of the ethctest command

## NOTICES

The command cannot be executed while running in *fatal error mode*.

If the **INTERFACE** option is set to **au**i, **thin** or **tpe** (external loop-back), the selected network interface on the ethernet must be connected with a network cable to the network, before running this test.

If the specified length of the memory address range is smaller than 8192 bytes aligned to a 16 bytes address, no test will be performed.

## WARNINGS

If the **TYPE** option is set to **send** and external loop-back is selected, the network can be loaded up to 100%, while transmitting frames to the network.

## COMMAND ERRORS

If the command returns the error code *Options error*, then it is because the memory range is not aligned on a 4 bytes address.

During heavy load conditions on the network a *Transmitted ethernet frame failed* error or *Missing an ethernet frame* error may occur.

ethtest (ethernet)

baio30x command

ethtest (ethernet)

## REFERENCES

See also commands:

*ethdiag, ethernet, ethmon, ethstat*

**ethtest (ethernet)**

**baio30x command**

**ethtest (ethernet)**

*This page is intentionally left blank*

**NAME**

*hdlc* – Test script for HDLC

**SYNOPSIS**

*hdlc* [-dHDLC\_DEVICES] [-tTEST\_FUNCTIONS] [-pDISP\_CMDS]

[-dHDLC\_DEVICES]        [-d[subm1 | subm2 | subm3] [,\*,,]]

[-tTEST\_FUNCTIONS]     [-t[fast | function] [,\*,,]]

[-pDISP\_CMDS]         [-p]

**DEFAULTS**

*hdlc* -dALL\_HDLC\_SUBMODULS -tfast,function

**DESCRIPTION**

This command is a *script command*, and it is used to run a complete test of the *hdlc30x* submodules that are installed on the *baio30x* module. The command executes other commands which are all tests of the *hdlc30x* submodules. It is possible to specify which submodules and which commands of the *hdlc script* to be executed. This is done by means of the HDLC\_DEVICES and TEST\_FUNCTIONS options.

The *hdlc* command has the following options:

**-dHDLC\_DEVICES**

This option is used to select which *hdlc30x* submodule to be tested. The option must be specified as a *multi word symbol option*, and the following HDLC submodules can be specified:

**subm1**     Submodule 1. Only available if the *hdlc30x* submodule is installed.

**subm2**     Submodule 2. Only available if the *hdlc30x* submodule is installed.

**subm3**     Submodule 3. Only available if the *hdlc30x* submodule is installed.

**-tTEST\_FUNCTIONS**

This option is used to specify which test functions should be selected from the *hdlc script*. The option must be specified as a *multi word symbol option*, and the following test functions can be specified:

- fast** If this test function is selected, the *HDLC* submodule will only be tested fast and not thoroughly. The selected commands will only run internal loop-back tests on the submodule.
- function** If this test function is selected, the *HDLC* submodule will be tested thoroughly. The selected commands will run external loop-back tests on the submodule, if the special test plugs are mounted on the *HDLC* port interface.

**-pDISP\_CMDS**

If this option is specified, the selected commands from the script will only be displayed, not executed.

Without options the command executes all the commands from the scripts.



## EXAMPLES

```

baio 70 -> hdlc -p
Test script for HDLC
Device Function Command script
subm1 fast hdlcdiag -dsubm1; hdlctest -dsubm1 -iintern
subm2 fast hdlcdiag -dsubm2; hdlctest -dsubm2 -iintern
subm3 fast hdlcdiag -dsubm3; hdlctest -dsubm3 -iintern
subm1 function hdlcdiag -dsubm1 -n256; hdlctest -dsubm1 -b2400; hdlctest
-dsubm1 -b9600; hdlctest -dsubm1 -b38400 -n256
subm2 function hdlcdiag -dsubm2 -n256; hdlctest -dsubm2 -b2400; hdlctest
-dsubm2 -b9600; hdlctest -dsubm2 -b38400 -n256
subm3 function hdlcdiag -dsubm3 -n256; hdlctest -dsubm3 -b2400; hdlctest
-dsubm3 -b9600; hdlctest -dsubm3 -b38400 -n256
baio 70 ->

```

Figure 11-hdlc-1: Example of the hdlc command displaying all commands from the script

```

baio 70 -> hdlc -dsubm2 -tfast -p
Test script for HDLC
Device Function Command script
subm2 fast hdlcdiag -dsubm2; hdlctest -dsubm2 -iintern
baio 70 ->

```

Figure 11-hdlc-2: Example of the hdlc command selecting a few commands from the script

## NOTICES

The command cannot be used while running in *fatal error mode*.

## REFERENCES

See also commands:

*hdlcdiag*, *hdlctest*.

hdlc (hdlc)

baio30x command

hdlc (hdlc)

*This page is intentionally left blank*

**NAME**

*hdlcdiag* – HDLC controller diagnose test

**SYNOPSIS**

*hdlcdiag* [-dDEVICE] [-pPORT] [-fFUNCTION] [-nNO\_OF\_TEST]

[-dDEVICE]           [-d[subm1 | subm2 | subm3]]

[-pPORT]           [-p[BIT RANGE FROM 0 TO 3]]

[-fFUNCTION]       [-f[reg | intr][,\*,\*]]

[-nNO\_OF\_TEST]   [-nDIGIT]

**DEFAULTS**

*hdlcdiag* -dFIRST\_HDLC\_SUBMODULE p0-3 -freg,intr -n16

**DESCRIPTION**

The *hdlcdiag* command performs a simple diagnostics test on the *HDLC controllers* on the *hdlc30x* submodule. This is done by running a register test and an interrupt test. This is very useful to diagnose an error in the *HDLC controller* when the controller communicates with the *CPU*.

It is possible to select which type of test the command shall perform and which *HDLC* port it shall use on the *hdlc30x* submodule. This is done by means of the **FUNCTION** and **PORT** options.

The register test will test the communication between the *CPU* and the *HDLC controller*. This is done by writing an 8 bit counter in the timer register of the HDLC controller and checking data after each write. Between the write and read access, a zero value will be written to another register of the HDLC controller. The test runs 256 times and the ports are tested one by one.

The interrupt test is used to test, that an interrupt from an *HDLC controller* can be received by the *CPU*. The program will first test that there are no interrupts from the *HDLC controllers*. After that the program will initiate an interrupt from all *HDLC controllers* and test that the interrupts are made. Finally, the program will initiate an interrupt from the *HDLC ports* one by one. All interrupts from the *HDLC controllers* are made by using the internal timer function on the controller.

The *hdlcdiag* command has the following options:

**-dDEVICE**

Same description as the *hdlctest* command.

**-pPORT**

Same description as the *hdlctest* command.

**-fFUNCTION**

This option is used to specify which test functions the command shall perform. The option must be specified as a *multi word symbol option* and the following tests can be specified:

**reg** If this test function is selected, the command will perform register tests in the *HDLC controller*.

**intr** If this test function is selected, the command will perform interrupt tests on the *HDLC submodule*.

**-nNO\_OF\_TEST**

This option is used to specify how many times the register and interrupt tests shall be performed. It must be specified as a *digit value option*.

## EXAMPLES

```

baio 70 -> hdlcdiag -p0-1
Test started at Monday 1995-07-03 - 09:36:56
HDLC controller diagnose test on submodule 2
Function  HDLC port      Number of tests
reg       0-1             16

Test started at Monday 1995-07-03 - 09:36:57
HDLC controller diagnose test on submodule 2
Function  HDLC port      Number of tests
intr     0-1             16

baio 70 ->

```

Figure 11-hdlcdiag-1: Example of the hdlcdiag command

## NOTICES

The command cannot be executed while running in *fatal error mode*.

The command stores test information in the *test log* buffer for each type of test function.

## REFERENCES

See also commands:  
*hdlc*, *hdlctest*, *hdlcmon*

**hdlcdiag (hdlc)**

**baio30x command**

**hdlcdiag (hdlc)**

*This page is intentionally left blank*

**NAME**

*hdlcmon* – HDLC monitor line status

**SYNOPSIS**

*hdlcmon* [-dDEVICE] [-pPORT]

[-dDEVICE]      [-d[subm1 | subm2 | subm3]]

[-pPORT]        [-p[BIT RANGE FROM 0 TO 3]]

**DEFAULTS**

*hdlcmon* -dFIRST\_HDLC\_SUBMODULE -p0-3

**DESCRIPTION**

The *hdlcmon* command is used to monitor all input signal lines from the external *HDLC* interface on an *hdlc30x* submodule. The command will display various information about the *HDLC* interface. The following information will be displayed for each *HDLC* port:

- *HDLC* interface type (*V28* or *V11*).
- Current status on *CTS* (Clear To Send).
- Current status on *DSR* (Data Set Ready).
- Current status on *CD* (Carrier Detect).
- Current status on *RI* (Ring Indicator).
- Current status on *DATA* (Receive Data).
- Current status on *RCLK* (Input Receive Clock).
- Current status on *TCLK* (Input Transmit Clock).

Before monitoring the command will set the *RTS* output signal to active and the *DTR* to passive. The output clock signals *RCLK* and *TCLK* are not generated.

The *hdlcmon* command has the following options:

**-dDEVICE**

Same description as the *hdlctest* command.

**-pPORT**

Same description as the *hdlctest* command.

**EXAMPLES**

```
baio 70 -> hdlcmon
Test started at Monday 1995-07-03 - 09:37:25
HDLC monitor line status on submodule 2

Port  Type  CTS   DSR   CD    RI    DATA  RCLK  TCLK
0     V28   passive passive passive passive  no    no
1     V28   passive passive passive passive  no    no
2     V11   passive passive passive         no    no
3     V11   passive passive passive         no    no
baio 70 ->
```

Figure 11-hdlcmon-1: Example of the hdlcmon command

**NOTICES**

The command cannot be executed while running in *fatal error mode*.

The command always displays the line status from the *HDLC* interface, no matter how the *dispmode parameter* is set.

**REFERENCES**

See also commands:

*hdlc*, *hdlcdiag*, *hdlctest*, *set*



## NAME

*hdlctest* – HDLC transmission test

## SYNOPSIS

*hdlctest* [-dDEVICE] [-pPORT] [-fFUNCTION] [-iINTERFACE] [-bBAUDRATE] [-nNO\_OF\_TEST]

[-dDEVICE] [-d[subm1 | subm2 | subm3]]

[-pPORT] [-p[BIT RANGE FROM 0 TO 3]]

[-fFUNCTION] [-f[modem | data][\*,\*,]]

[-iINTERFACE] [-i[default | intern | extern]]

[-bBAUDRATE] [-b[9600 | 1200 | 2400 | 4800 | 19200 | 38400 | 76800 | 153600]]

[-nNO\_OF\_TEST] [-nDIGIT]

## DEFAULTS

*hdlctest* -dFIRST\_HDLC\_SUBMODULE -p0-3 -fmodem,data  
-idefault -b9600 -n16

## DESCRIPTION

The *hdlctest* command is used to run a modem signal test and a data transmission test on the *hdlc30x* submodule. This can either be done via internal loop-back in the *HDLC controller* or external loop-back in the *HDLC interface*. The command is very useful to diagnose an error on the *HDLC interface*.

It is possible to select which type of test and which *HDLC* port the command shall perform on the submodule. This is done by means of the **FUNCTION** and **PORT** options.

The modem signal test will test all modem and clock signals between the *HDLC controller* and the interface. Modem output signals from the controller are *RTS*, *DTR*, *TxCLK*, and *TxRCLK* and the modem input signals to the controller are *CTS*, *DSR*, *CD*, *RI*, *RxCLK*, and *RxRCLK*. The test will activate and deactivate output signals and check the input signals for correct level. The modem signal test will run on all selected *HDLC* ports, before signals are changed. No test will be performed, if the **INTERFACE** option is set

to intern (internal loop-back).

The data transmission test will test the *HDLC* interface as follows:

- The *HDLC controller* is initialized to HDLC mode and then all ports are tested by transmitting characters within the range from 0x00 to 0xff, from a port to the same port and checking that a received character is the same as the one transmitted.
- The program will transmit 31 characters plus one control byte on the first port and then transmit 31 characters, different from the first 32, plus one control byte on the next port and so on. After all the ports have transmitted the characters, the program will check the characters on the first port and then check the characters on the next port and so on. Before checking received data, the program will check the transmit and receive status for errors.

The *hdlctest* command has the following options:

#### **-dDEVICE**

This option is used to select which *hdlc30x* submodule the test has to use. The option must be specified as a *word symbol option*, and the following submodules can be specified:

- subm1** Submodule 1. Only available if the *hdlc30x* submodule is installed.
- subm2** Submodule 2. Only available if the *hdlc30x* submodule is installed.
- subm3** Submodule 3. Only available if the *hdlc30x* submodule is installed.

#### **-pPORT**

This option is used to select the *HDLC* ports you want to test. The option must be specified as a *bit range option*. Default, the program will use all four ports on the *hdlc30x* submodule.

**-fFUNCTION**

This option is used to specify which test functions the command shall perform. The option must be specified as a *multi word symbol option*, and the following tests can be specified:

- modem** If this test function is selected, the command will perform a modem signal test on the *HDLC* interface.
- data** If this test function is selected, the command will perform a data transmission test on the *HDLC* interface.

**-iINTERFACE**

This option is used to specify which *HDLC* interface the test has to use. The option must be specified as a *word symbol option*, and the following interface types can be specified:

- default** If this is selected, the program will select either internal or external loop-back depending on whether a special *HDLC test plug* is mounted on an *HDLC* port. If an *HDLC test plug* is mounted on an *HDLC* port, the program will select external loop-back. Otherwise internal loop-back will be selected.
- intern** Internal loop-back in the *HDLC controller*. The transmitted data will be looped-back to the receiver by the controller itself.
- extern** External loop-back on the *HDLC* interface. The output modem signals on *HDLC* interface will be looped-back to the input modem signals, and the transmitted data will be looped-back to the receiver on the *HDLC* interface. All this is done by mounting a special *HDLC test plug* in each selected *HDLC* port on the interface.

**-bBAUDRATE**

This option is used to specify which baudrate the *HDLC controller* has to use. The option must be specified as a *word symbol option*, and the following baudrates can be specified:

hdlctest (hdlc)

baio30x command

hdlctest (hdlc)

- 9600 Data transmission with 9600 baud.
- 1200 Data transmission with 1200 baud.
- 2400 Data transmission with 2400 baud.
- 4800 Data transmission with 4800 baud.
- 19200 Data transmission with 19200 baud.
- 38400 Data transmission with 38400 baud.
- 76800 Data transmission with 76800 baud.
- 153600 Data transmission with 153600 baud (only internal loop-back).

**-nNO\_OF\_TEST**

This option is used to specify how many times the modem and data transmission tests shall be performed. It must be specified as a *digit value option*.

In order to test external loop-back on the *HDLC* interface, you must use two set of *HDLC test plugs*, strapped as shown below.

V28 interface, port 0 - 1			
pin 2	↔	pin 3	
pin 4	↔	pin 5	
pin 4	↔	pin 22	
pin 6	↔	pin 8	
pin 6	↔	pin 20	
port 0, pin 14	↔	pin 15, port 1	
port 0, pin 15	↔	pin 14, port 1	
pin 16	↔	pin 17	

Figure 11-hdlctest-1: HDLC test plugs used for test of V28 interface

V11 interface, port 2 - 3			
pin 2	↔	pin 4	
port 2, pin 3	↔	pin 9, port 3	
pin 5	↔	pin 8	
pin 6	↔	pin 13	
pin 7	↔	pin 19	
port 2, pin 9	↔	pin 3, port 3	
pin 10	↔	pin 11	
pin 10	↔	pin 12	
pin 14	↔	pin 16	
port 2, pin 15	↔	pin 21, port 3	
pin 17	↔	pin 20	
pin 18	↔	pin 25	
port 2, pin 21	↔	pin 15, port 3	
pin 22	↔	pin 23	
pin 22	↔	pin 24	

Figure 11-hdlctest-2: HDLC test plugs used for test of V11 interface

**EXAMPLES**

```

baio 70 -> hdlctest -b38400 -n256 -idefault
Test started at Monday 1995-07-03 - 09:37:08
HDLC transmission test on submodule 2
Function  HDLC ports      Interface  Baudrate  Number of tests
modem    0-3                    extern    38400     256

Test started at Monday 1995-07-03 - 09:37:08
HDLC transmission test on submodule 2
Function  HDLC ports      Interface  Baudrate  Number of tests
data     0-3                    extern    38400     256

baio 70 ->
    
```

Figure 11-hdlctest-3: Example of the hdlctest command

## NOTICES

The command cannot be executed while running in *fatal error mode*.

If the **INTERFACE** option is set to **extern** (external loop-back), the selected *HDLC* ports must be mounted with a special *HDLC test plug* on the *HDLC* interface, before running this test.

## DIAGNOSTICS

If the test finds errors, then try to diagnose them by means of the *hdlcdiag* command.

## COMMAND ERRORS

If the command returns the error code *Value in range to high*, it is because the baudrate is set to 153600 and the interface is set to external loop-back.

## REFERENCES

See also commands:

*hdlc*, *hdlcdiag*, *hdlcmon*

**NAME**

*help* – Display command information and options

**SYNOPSIS**

*help* [-allDISPLAY\_COMMANDS] [command]

[-allDISPLAY\_COMMANDS]

[-all]

[command] [COMMAND [OPTIONS]]

**DESCRIPTION**

The *help* command is used to display *command help information*. The command can display special keys or *command help information* for one or for all command(s).

The *help* command has the following options:

**-all**

If this option is selected, the *help* command displays *command help information* for all commands on the *baio30x* module.

**command**

If a command name is specified, the *help* command displays *command help information* only for the selected command.

If no options are given, the *help* command displays all special keys.

## EXAMPLES

```
baio 70 -> help
SPECIAL KEYS:
ESC # SP   Deselect a unit. ESC <--> CTRL-A
ESC # CR   Select the master unit. ESC <--> CTRL-A
ESC # U CR Select a unit where U is slot/subpos. ESC <--> CTRL-A
CTRL-C     Cancel a running test
CTRL-E     Soft cancel a running test
CTRL-Q     Start printing on the terminal (XON)
CTRL-S     Stop printing on the terminal (XOFF)
CTRL-P     Start printing SCSI command on the terminal
CTRL-N     Stop printing SCSI command on the terminal
CTRL-L     Move curser one position to the right
CTRL-H     Move curser one position to the left
CTRL-K     Insert single character
CTRL-J     Delete single character
DEL        Delete to end of line
TAB        Move curser 8 positions to the right
??         Display last commands
?xx        Find and edit previous command line
!xx        Find and run previous command line
;          Separate commands
help -all  Display help information for all commands
help CMD   Display help information for the command
menu      Display commands
```

Figure 11-help-1: Example of help command which displays the special keys



```
baio 70 -> help help
help      Display command information and options
          [-allDISPLAY_COMMANDS] [-all]
          [command]                [COMMAND [OPTIONS]]
baio 70 -> help -?
help      Display command information and options
          [-allDISPLAY_COMMANDS] [-all]
          [command]                [COMMAND [OPTIONS]]
baio 70 ->
```

Figure 11-help-2: Example of help command which displays the command help information

## NOTICES

The command does not clear *test log* and *error log*.

The command cannot be repeated. It will only run once.

## REFERENCES

See also commands:

*menu*

**help (system)**

**baio30x command**

**help (system)**

*This page is intentionally left blank*

memaddr (memory)

baio30x command

memaddr (memory)

**NAME***memaddr* – Memory address bit test**SYNOPSIS***memaddr* [-dDEVICE] [-rADDR\_RANGE] [-aACCESS] [-cCACHE] [-bADDR\_BITS] [-vPAT\_VALUE]

[-dDEVICE] [-d[pmem | dmem | nvm | global | subm1 | subm2 | subm3]]

[-rADDR\_RANGE] [-r[START | #SIZE | START-END | START#SIZE]]

[-aACCESS] [-a[word | dword | short | byte] [,\*]]

[-cCACHE] [-c[on | off]]

[-bADDR\_BITS] [-b[BIT RANGE FROM 0 TO 63]]

[-vPAT\_VALUE] [-v[zero | ones | [DIGIT]]]

**DEFAULTS***memaddr* -dpmem -r#0x100000 -aword -con -b0-63 -vzero**DESCRIPTION**

The *memaddr* command is used to test a single address bit in a memory. This is very useful to diagnose fault on an address bit in the memory.

The test will write a value in a memory. The value is dependent on which address bit is tested. If the current address bit is one, the value from the *PAT\_VALUE* option is used. And if the address bit is zero, the inverted value is written. After writing all data, the memory is checked for correct data contents. All this is repeated for all address bits which are selected with the *ADDR\_BITS* option in the same memory array.

The *memaddr* command has the following options:

**-dDEVICE**

Same description as the *memfast* command.

**-rADDR\_RANGE**

Same description as the *memfast* command.

memaddr (memory)

baio30x command

memaddr (memory)

**-aACCESS**

Same description as the *memfast* command.

**-cCACHE**

Same description as the *memfast* command.

**-bADDR\_BITS**

This option is used to select which address bits you want to test. The option must be specified as a *bit range option*. Default, the bit range will be set to all address bits in the selected memory.

**-vPAT\_VALUE**

This option is used to specify a 32 bit data value which is written when the address bit is one. If the address bit is zero the inverted value is written. The option must be specified as a *word symbol or digit value option*, and the following predefined values can be specified:

**zero**    The same value as 0x00000000.

**ones**    The same value as 0xffffffff.

**EXAMPLES**

```

baio 70 -> memaddr -r*-* -dcmem -b4,6
Test started at Tuesday 1995-06-06 - 10:48:43
Memory address bit test in Data memory
Start address End address Cache Write Read Address bit Pattern value
0x00000000 0x00400000 on word word 4 0x00000000

Test started at Tuesday 1995-06-06 - 10:48:44
Memory address bit test in Data memory
Start address End address Cache Write Read Address bit Pattern value
0x00000000 0x00400000 on word word 6 0x00000000
baio 70 ->
    
```

Figure 11-memaddr-1: Example of the memaddr command

memaddr (memory)

baio30x command

memaddr (memory)

## NOTICES

The command cannot be used while running in *fatal error mode*.

The command will store test information in the *test log* buffer for each tested address bit.

It is only possible to test address bits up to the number of address bits in the selected memory. If you select address bits over that number, the command will ignore these bits.

## DIAGNOSTICS

If the test finds errors, then try to diagnose them with the *memfast* or the *memdata* commands.

## COMMAND ERRORS

If the command returns the error code *Options error*, it is because the memory range is not aligned on a 4 bytes address or an 8 bytes address if using *dword access*.

## REFERENCES

See also commands:

*set, memory, memfast, memdata, memgallop, memcmp, memdisp*

**memaddr (memory)**

**baio30x command**

**memaddr (memory)**

*This page is intentionally left blank*

**NAME**

*memcmp* – Memory move and compare test

**SYNOPSIS**

*memcmp* [-dDEVICE] [-rADDR\_RANGE] [-mdMOVE\_DEV] [-maMOVE\_ADDR] [-aACCESS] [-cCACHE] [-scSRC\_CHECK] [-pPATTERN] [-vPAT\_VALUE]

[-dDEVICE]	[-d[pmem   dmem   nvm   global   subm1   subm2   subm3]]
[-rADDR_RANGE]	[-r[START   #SIZE   START-END   START#SIZE]]
[-mdMOVE_DEV]	[-d[pmem   dmem   nvm   global   subm1   subm2   subm3]]
[-maMOVE_ADDR]	[-ma[DIGIT]]
[-aACCESS]	[-a[word   dword   short   byte   word4   word8   word16   word32] [,*]]
[-cCACHE]	[-c[on   off] [,*]]
[-scSRC_CHECK]	[-sc[on   off]]
[-pPATTERN]	[-p[random   value   count   cntdown   current]]
[-vPAT_VALUE]	[-v[DIGIT]]

**DEFAULTS**

*memcmp* -dpmem -r#0x100000 -mdSAME\_AS\_DEVICE -aword -con -scon -prandom -vX

**DESCRIPTION**

The *memcmp* command is used to test a memory by moving and comparing data from one memory array to another. The memory arrays can be in different memory devices.

The test will be performed as follows:

- The source memory array is initialized with a data pattern type, specified with the **PATTERN** and **PAT\_VALUE** options.
- The data will be copied from the source memory array to the destination memory array.
- The destination memory is read and compared with the source memory.
- The source memory is read and it is checked that correct data was used for initializing. This is only done if the **SRC\_CHECK** option is set to **on** and the **PATTERN** option is not set to **current**.

The *memcmp* command has the following options:

**-dDEVICE**

Same description as the *memfast* command.

**-rADDR\_RANGE**

Same description as the *memfast* command.

**-mdMOVE\_DEV**

Same description as the **DEVICE** option, only this is the destination memory device. Default, the command will set the option to the same memory as the source memory device.

**-maMOVE\_ADDR**

This option is used to specify a start memory address for the destination memory array. The size of the destination memory array will always be the same as the source memory array. The option must be specified as a *digit value option*. Default, the command will reserve the next free destination memory array.

**-aACCESS**

This option is used to specify which type of access the command shall perform when accessing the source and destination memory. The option must be specified as a *double word symbol option*. The access type can be different for the source and destination memory. The first word symbol will be used for the source memory and the second for the destination memory.



The following access types can be specified:

- word**      *Word accesses* will be performed to the memory.
- dword**     *Double word accesses* will be performed to the memory (only applicable when using the *global memory*).
- short**     *Short accesses* will be performed to the memory.
- byte**      *Byte accesses* will be performed to the memory.
- word4**     *4 Word accesses* will be performed to the memory (only applicable when using the *global memory*).
- word8**     *8 Word accesses* will be performed to the memory (only applicable when using the *global memory*).
- word16**    *16 Word accesses* will be performed to the memory (only applicable when using the *global memory*).
- word32**    *32 Word accesses* will be performed to the memory (only applicable when using the *global memory*).

#### **-cCACHE**

This option is used to select whether the *CPU cache* shall be used or not, while reading in the source and destination memory. The option must be specified as a *double word symbol option*. The cache mode can be different for source and destination memory. The first word symbol will be used for the source memory and the second for the destination memory. The following cache modes can be specified:

- on**        The *CPU cache* is used when executing read access to the memory.
- off**       The *CPU cache* is not used when executing read access to the memory.

#### **-scSRC\_CHECK**

This option is used to specify the data check mode for the source memory. The option must be specified as a *word symbol option*, and the following two check modes can be specified:

**memcmp (memory)**

**balc30x command**

**memcmp (memory)**

- on** Data check will be performed on the source memory contents. This will only be done if the **PATTERN** option is not set to **current**.
- off** No check will be performed on the source memory contents.

**-pPATTERN**

This option is used to specify a data pattern type together with the **PAT\_VALUE** option. The data pattern is used to initialize the source memory array. The **PATTERN** option must be specified as a *word symbol option*, and the following pattern types can be specified:

- random** This pattern type will generate a 32 bit random data pattern from the **PAT\_VALUE**. If the same **PAT\_VALUE** is specified, the same random pattern is generated.
- value** This pattern type will generate a 32 bit data value specified with the **PAT\_VALUE** option.
- count** This pattern type will generate a 32 bit count data pattern calculated as follows: **PAT\_VALUE + (offset / 4)**.
- cntdown** This pattern type will generate a 32 bit count-down data pattern calculated as follows: **PAT\_VALUE - (offset / 4)**.
- current** This pattern type will use the current values in the source memory address range.

**-vPAT\_VALUE**

This option is used to specify a data value for the pattern type. The option must be specified as a *digit value option*. Default, the value is a value from the internal system clock. The option only has effect if the **PATTERN** option is not set to **current**.

## EXAMPLES

In the example below the *memcmp* command will perform the test as follows:

- The command writes a 32 bit counter in the *global memory* from address 0x400000 to 0x700000.
- Afterwards the data will be copied from the *global memory* to the *data memory* address range from 0x100000 to 0x400000.
- Then the *data memory* contents will be compared with the *global memory* by reading the data from both memories.
- Finally, when the comparing is complete, the command will check the source memory contents for the above described 32 bit counter.

The access type will always be *byte accesses* in the *global memory* and *word accesses* in the *data memory*.

```

baio 70 -> memcmp -dglobal -mddmem -ma0x100000 -r0x400000#0x300000
-abyte,word -pcount -v0
Test started at Friday 1995-06-16 - 14:49:50
Memory move and compare test in Global memory
Dir  Address      Memory  Access  Cache  Compare  Length      Pattern
src  0x00400000  global  byte    on     on       0x00300000  count
dest 0x00100000  dmem    word    on     on
baio 70 ->
    
```

Figure 11-memcmp-1: Example of the memcmp command

memcmp (memory)

baio30x command

memcmp (memory)

## NOTICES

The command cannot be used while running in *fatal error mode*.

It is recommended to set the **ACCESS** option to **word32**, if you want to use the *global memory* as source memory and the **PATTERN** option is set to **current**. This is due to the *cache coherence* on the *global bus* (the source data can be in a *CPU cache* instead of in the *global memory*).

## DIAGNOSTICS

If the test finds errors, then try to diagnose them by means of the *memfast* command.

## COMMAND ERRORS

If the command returns the error code *Options error*, it is because the memory range is not aligned on a 4 bytes address or because the size of the access type is wrong. (For example, when selecting a 32 *Word access* the address must be aligned on an 128 bytes address, a *Double word access* must be aligned on an 8 bytes address etc.)

## BUGS

The **ADDR\_RANGE** option cannot be set to **\*-\***, **\*\*#** or **#\***, if the source and destination memory is the same memory device or the size of the source memory is bigger than the size of destination memory.

## REFERENCES

See also commands:

*set, memory, memfast, memdata, memaddr, memgallop, memdisp*

**NAME**

*memdata* – Memory data bit test

**SYNOPSIS**

*memdata* [-dDEVICE] [-rADDR\_RANGE] [-aACCESS] [-cCACHE] [-bDATA\_BITS] [-vPAT\_VALUE]

[-dDEVICE] [-d[pmem | dmem | nvm | global | subm1 | subm2 | subm3]]

[-rADDR\_RANGE] [-r[START | #SIZE | START-END | START#SIZE]]

[-aACCESS] [-a[word | dword | short | byte] [\*]]

[-cCACHE] [-c[on | off]]

[-bDATA\_BITS] [-b[BIT RANGE FROM 0 TO 63]]

[-vPAT\_VALUE] [-v[zero | ones | [DIGIT]]]

**DEFAULTS**

*memdata* -dpmem -r#0x100000 -aword -con -b0-63 -vzero

**DESCRIPTION**

The *memdata* command is used to test a single data bit in a memory. This is very useful to diagnose fault on a data bit in the memory.

The test will write a value in a memory. The value depends on which data bit is tested. The value is calculated as the value from the **PAT\_VALUE** option, inverted with the current data bit. After writing all data, the memory is checked for correct data contents. All this is repeated for all data bits which are selected by the **DATA\_BITS** option in the same memory array.

The *memdata* command has the following options:

**-dDEVICE**

Same description as the *memfast* command.

**-rADDR\_RANGE**

Same description as the *memfast* command.

**-aACCESS**

Same description as the *memfast* command.

**-cCACHE**

Same description as the *memfast* command.

**-bDATA\_BITS**

This option is used to select the data bits you want to test. The option must be specified as a *bit range option*. Default, the bit range will be set to all data bits in the selected memory.

**-vPAT\_VALUE**

This option is used to specify a data value for the test. The number of data bits used in the value depends on how many data bits the selected memory has, but if the number of data bits is 64, the 32 bit value is copied to the data bits 32 to 63. The option must be specified as a *word symbol or digit value option*, and the following predefined values can be specified:

**zero**    The same value as 0x00000000.

**ones**    The same value as 0xffffffff.

## EXAMPLES

```

baio 70 -> memdata -dnvm -v0xffffffff
Test started at Tuesday 1995-06-06 - 10:46:34
Memory data bit test in NVM memory
Start address End address Cache Write Read Data bit Pattern value
0x00004000 0x00008000 off byte byte 0 0xffffffff

Test started at Tuesday 1995-06-06 - 10:46:35
Memory data bit test in NVM memory
Start address End address Cache Write Read Data bit Pattern value
0x00004000 0x00008000 off byte byte 1 0xffffffff

Test started at Tuesday 1995-06-06 - 10:46:35
Memory data bit test in NVM memory
Start address End address Cache Write Read Data bit Pattern value
0x00004000 0x00008000 off byte byte 7 0xffffffff
baio 70 ->

```

Figure 11-memdata-1: Example of the memdata command

## NOTICES

The command cannot be used while running in *fatal error mode*.

The command stores test information in the *test log* buffer for each tested data bit.

It is only possible to test data bit up to the number of data bits in the selected memory. If you select data bits over that number, the command will ignore these bits.

## DIAGNOSTICS

If the test finds errors, then try to diagnose them by means of the *memfast* or the *memaddr* commands.

## COMMAND ERRORS

If the command returns the error code *Options error*, it is because the memory range is not aligned on a 4 bytes address or an 8 bytes address if using *dword access*.

**REFERENCES**

See also commands:

*set, memory, memfast, memaddr, memgallop, memcmp, memdisp*



**NAME**

*memdisp* – Display memory contents

**SYNOPSIS**

*memdisp* [-dDEVICE] [-rADDR\_RANGE] [-aACCESS] [-cCACHE]

[-dDEVICE] [-d[pmem | dmem | nvm | global | subm1  
| subm2 | subm3]]

[-rADDR\_RANGE] [-r[START | #SIZE | START-END |  
START#SIZE]]

[-aACCESS] [-a[word | dword | short | byte | word4 |  
word8 | word16 | word32]]

[-cCACHE] [-c[on | off]]

**DEFAULTS**

*memdisp* -dpmem -r0#256 -aword -con

**DESCRIPTION**

The *memdisp* command is used to display data contents from a memory on the *console terminal*. The command is very useful to diagnose memory faults which occur while running a memory test.

The memory to be displayed is specified in the **DEVICE** option. The memory address and number of bytes which is displayed is depended on the start and size value of the **ADDR\_RANGE** option. The memory range will not be reserved while reading from the memory. This means that the memory array can be used from another module while reading the data.

The command has the following options:

**-dDEVICE**

Same description as the *memfast* command.

**-rADDR\_RANGE**

This option is used to specify the memory address range where the command has to read the data to be displayed. The option must be specified as a *value range option*.

**-aACCESS**

This option is used to specify which type of access the command shall perform during the read access to a memory. The option must be specified as a *word symbol option*. The following access types can be specified:

- word**      *Word accesses* will be performed to the memory.
- dword**     *Double word accesses* will be performed to the memory (only applicable when using the *global memory*).
- short**      *Short accesses* will be performed to the memory.
- byte**        *Byte accesses* will be performed to the memory.
- word4**      4 *Word accesses* will be performed to the memory (only applicable when using the *global memory*).
- word8**      8 *Word accesses* will be performed to the memory (only applicable when using the *global memory*).
- word16**     16 *Word accesses* will be performed to the memory (only applicable when using the *global memory*).
- word32**     32 *Word accesses* will be performed to the memory (only applicable when using the *global memory*).

**-cCACHE**

Same description as the *memfast* command.

## EXAMPLES

```

baio 70 -> memdisp -dglobal -aword32 -r0x800000#128
Reading in Global memory from address 0x00800000 to 0x00800080
Address      0          4          8         12
0x00800000  647a5634  411a1305  1db9cfd6  fa598ca7  dzV4A.....Y..
0x00800010  d6f94978  b3990649  9038c31a  6cd87feb  ..Ix...I.8..l...
0x00800020  49783cbc  2617f98d  02b7b65e  df57732f  Ix<. &.....^..Ws/
0x00800030  bbf73000  9896ecd1  7536a9a2  51d66673  ..0.....u6..Q.fs
0x00800040  2e762344  0b15e015  e7b59ce6  c45559b7  .v#D.....UY.
0x00800050  a0f51688  7d94d359  5a34902a  36d44cfb  ...e..YZ4.*6.L.
0x00800060  137409cc  f013c69d  ccb3836e  a953403f  .t.....n.S@?
0x00800070  85f2fd10  6292b9e1  3f3276b2  1bd23383  ....b...?2v...3.
baio 70 ->

```

Figure 11-memdisp-1: Example of the memdisp command

## NOTICES

The command does not clear *test log* and *error log*.

The command always displays the data contents, no matter how the *dispmode parameter* is set.

It is recommended to set the **ACCESS** option to **word32**, if you want to display the real data (the data is not necessary in the *global memory*). This is due to the *cache coherence* on the *global bus*.

## COMMAND ERRORS

If the command returns the error code *Options error*, it is because the memory range is not aligned on a 4 bytes address or because of the size of the access type. (For example, when selecting a 32 *Word access* the address must be aligned on an 128 bytes address, a *Double word access* must be aligned on an 8 bytes address etc.)

## REFERENCES

See also commands:

*memfast, memdata, memaddr, memgallop, memexer, memcmp.*

**memdisp (memory)**

**baio30x command**

**memdisp (memory)**

*This page is intentionally left blank*

**NAME**

*memexer* – Memory exerciser

**SYNOPSIS**

*memexer* [-dDEVICE] [-rADDR\_RANGE] [-aACCESS] [-cCACHE] [-tTYPE] [-mMODE] [-pPATTERN] [-vPAT\_VALUE]

[-dDEVICE]	[-d[pmem   dmem   nvm   global   subm1   subm2   subm3]]
[-rADDR_RANGE]	[-r[START   #SIZE   START-END   START#SIZE]]
[-aACCESS]	[-a[word   dword   short   byte   word4   word8   word16   word32]]
[-cCACHE]	[-c[on   off]]
[-tTYPE]	[-t[read   check   write   cmp]]
[-mMODE]	[-m[reserv   raw]]
[-pPATTERN]	[-p[random   value   setdata   count   cntdown]]
[-vPAT_VALUE]	[-vLSB_DIGIT[,MSB_DIGIT]]

**DEFAULTS**

*memexer* -dpmem -r#0x100000 -aword -con -tread -mreserv -prandom -vX

**DESCRIPTION**

The *memexer* command is used to exercise write or read accesses in a memory. This is very useful to diagnose a memory fault in the memory. It is possible to select different types of tests by means of the **TYPE** option. The following tests can be performed:

- The test performs only read accesses with or without checking the read data.
- The test performs only write accesses.
- The test performs write accesses and afterwards it reads the memory contents and checks it.

The *memexer* command has the following options:

**-dDEVICE**

Same description as the *memfast* command.

**-rADDR\_RANGE**

Same description as the *memfast* command. If the **MODE** option is set to **raw**, the command will not reserve a free memory array. Default, the command will use the memory range from 0 up to 0x10000.

**-aACCESS**

Same description as the *memdisp* command.

**-cCACHE**

Same description as the *memfast* command.

**-tTYPE**

This option is used to specify which test the command shall perform. The option must be specified as a *word symbol option*, and the following types of test can be specified:

- read**      This test will perform read accesses to the specified memory address range. No check of the data will be done.
- check**     This test will perform read accesses to the specified memory address range. The read data will be checked with the specified pattern.
- write**     This test will perform write accesses to the specified memory address range.
- cmp**        This test will perform write accesses to the specified memory address range and afterwards read the memory contents and compare it to the written value.

**-mMODE**

This option is used to specify which mode the command has to use. The option must be specified as a *word symbol option*, and the following modes can be specified:

memexer (memory)

baio30x command

memexer (memory)

- reserv** In this mode, the command will reserve the memory array as the normal procedure.
- raw** In this mode, the command will not reserve the memory array as the normal procedure. This means that it is possible to access a memory array that is reserved for another purpose (e.g code and stack segments used by the program itself or by another unit).

**-pPATTERN**Same description as the *memfast* command.**-vPAT\_VALUE**Same description as the *memfast* command.**EXAMPLES**

```

baio 70 -> memexer -twrite -pcount -v0 -r0x100000-0x200000
Test started at Monday 1995-03-20 - 11:07:29
Memory exerciser in Program memory
Memory address range  Cache Access Mode Type Pattern Value
0x00100000-0x00200000 on word reserv write count 0x00000000
baio 70 -> memexer -tcheck -pcount -v0 -r0x100000-0x200000
Test started at Monday 1995-03-20 - 11:07:39
Memory exerciser in Program memory
Memory address range  Cache Access Mode Type Pattern Value
0x00100000-0x00200000 on word reserv check count 0x00000000
baio 70 ->

```

Figure 11-memexer-1: Example of the memexer command

**NOTICES**

The command cannot be used while running in *fatal error mode*.

It is recommended to set the **ACCESS** option to **word32**, if you want to use the *global memory* and the **TYPE** option is set to **check**. This is due to the *cache coherence* on the *global bus* (the data can be in a *CPU cache* instead of the *global memory*).

memexer (memory)

balo30x command

memexer (memory)

## COMMAND ERRORS

If the command returns the error code *Options error*, it is because the memory range is not aligned on a 4 bytes address, or because the size of the access type is false. (For example, when selecting a 32 *Word access* the address must be aligned on a 128 bytes address, a *Double word access* must be aligned on an 8 bytes address etc.).

## WARNINGS

If setting the **TYPE** option to **check**, you must be sure that the data for the comparison has been written in the memory, otherwise the program will find incorrect data in the memory.

If the **MODE** option is set to **raw** and the **TYPE** option is set to **write** or **cmp**, the command will not reserve the memory array. I.e. that it is possible to perform write access in a memory array that is reserved for another purpose (e.g code and stack segments used by the program itself or by another unit). This means that the Diagnostic Programs can break down.

## REFERENCES

See also commands:

*memaddr, memcmp, memdata, memfast, memgallop, set*



**NAME**

*memfast* – Memory high speed test

**SYNOPSIS**

*memfast* [-dDEVICE] [-rADDR\_RANGE] [-aACCESS] [-cCACHE] [-pPATTERN] [-vPAT\_VALUE]

[-dDEVICE] [-d[pmem | dmem | nvm | global | subm1 | subm2 | subm3]]

[-rADDR\_RANGE] [-r[START | #SIZE | START-END | START#SIZE]]

[-aACCESS] [-a[word | dword | short | byte] [,\*]]

[-cCACHE] [-c[on | off]]

[-pPATTERN] [-p[random | value | setdata | count | cntdown]]

[-vPAT\_VALUE] [-vLSB\_DIGIT[,MSB\_DIGIT]]

**DEFAULTS**

*memfast* -dpmem -r#0x100000 -aword -con -prandom -vX

**DESCRIPTION**

The *memfast* command is used to perform a fast test in a memory. This is done by writing a pattern in the memory and afterwards reading the memory contents and comparing it to the written value.

The *memfast* command has the following options:

**-dDEVICE**

This option is used to select which memory to be tested. The option must be specified as a *word symbol option*, and the following memories can be specified:

**pmem** Program memory.

**dmem** Data memory.

**nvm** NVM RAM.

memfast (memory)

baio30x command

memfast (memory)

- global** Global memory.
- subm1** Submodule 1. Only available if the submodule is installed and has a local memory mounted.
- subm2** Submodule 2. Only available if the submodule is installed and has a local memory mounted.
- subm3** Submodule 3. Only available if the submodule is installed and has a local memory mounted.

**-rADDR\_RANGE**

This option is used to specify the memory address range where the test has to write and read data in the memory. The option must be specified as a *value range option*. Default, the command will reserve the next free memory array with the necessary number of bytes up to 1 Mb.

**-aACCESS**

This option is used to specify which type of access the command shall perform during the write and read access to a memory. The option must be specified as a *double word symbol option*. The access type can be different for the write and the read access. The first word symbol will be used for write access and the second for read access. The following access types can be specified:

- word** *Word accesses* will be performed to the memory.
- dword** *Double word accesses* will be performed to the memory (only applicable when using the *global memory*).
- short** *Short accesses* will be performed to the memory.
- byte** *Byte accesses* will be performed to the memory.

**-cCACHE**

This option is used to select whether the *CPU cache* shall be used, while writing and reading in the memory. The option must be specified as a *word symbol option*, and the following cache modes can be specified:

- on**     The *CPU cache* is used when executing write/read access to the memory.
- off**    The *CPU cache* is not used when executing write/read access to the memory.

**-pPATTERN**

This option is used to specify a data pattern type together with the **PAT\_VALUE** option. The **PATTERN** option must be specified as a *word symbol option*, and the following pattern types can be specified:

- random**   This pattern type will generate a 32 bit random data pattern from the **PAT\_VALUE**. If the same **PAT\_VALUE** is specified, the same random pattern is generated.
- value**    This pattern type will generate a 32 bit data value specified with the **PAT\_VALUE** option.
- setdata**   This is the same as **value** pattern, but the number of data bits in the value depends on how many data bits the selected memory has.
- count**    This pattern type will generate a 32 bit count data pattern calculated as follows: **PAT\_VALUE** + (offset / 4).
- cntdown**   This pattern type will generate a 32 bit count-down data pattern calculated as follows: **PAT\_VALUE** - (offset / 4).

**-vPAT\_VALUE**

This option is used to specify a data value for the pattern type. The option must be specified as a *double digit value option*. Default, the **LSB\_DIGIT** value is a value from the internal

system clock and the *MSB\_DIGIT* value is set to zero. The *MSB\_DIGIT* value only has effect if the *DEVICE* option is set to *global* and the *PATTERN* option is set to *setdata*.

## EXAMPLES

In the example below the *memfast* command tests the *global memory* from address *0x400000* to *0x400080* by using *byte access* while writing and using *word access* while reading. The pattern which is written is a *64 bit value*.

After the test is complete a *memdisp* command is executed to display the memory contents.

```

baio 70 -> memfast -dglobal -r0x400000#0x80 -abyte,word -psetdata
-v0x89abcdef,0x01234567
Test started at Monday 1995-03-20 - 11:48:35
Memory high speed test in Global memory
Start address End address Cache Write Read Type Pattern value
0x00400000 0x00400080 on byte word setdata 0x0123456789abcdef

baio 70 -> memdisp -dglobal -r0x400000#0x20
Reading in Global memory from address 0x00400000 to 0x00400020
Address 0 4 8 12
0x00400000 01234567 89abcdef 01234567 89abcdef .#Eg....#Eg....
0x00400010 01234567 89abcdef 01234567 89abcdef .#Eg....#Eg....
baio 70 ->

```

Figure 11-memfast-1: Example of the *memfast* command

## NOTICES

The command cannot be used while running in *fatal error mode*.

## DIAGNOSTICS

If the test finds errors, then try to diagnose them by means of the *memdata* or the *memaddr* command.

## COMMAND ERRORS

If the command returns the error code *Options error*, it is because the memory range is not aligned on a 4 bytes address or an 8 bytes address if using *dword access*.

## REFERENCES

See also commands:

*set, memory, memaddr, memdata, memgallop, memcmp, memdisp.*

*This page is intentionally left blank*

memgallop (memory)

baio30x command

memgallop (memory)

**NAME***memgallop* – Memory galloping test**SYNOPSIS***memgallop* [-dDEVICE] [-rADDR\_RANGE] [-aACCESS] [-cCACHE] [-vPAT\_VALUE]

[-dDEVICE] [-d[pmem | dmem | nvm | global | subm1 | subm2 | subm3]]

[-rADDR\_RANGE] [-r[START | #SIZE | START-END | START#SIZE]]

[-aACCESS] [-a[word | dword | short | byte] [,\*]]

[-cCACHE] [-c[on | off]]

[-vPAT\_VALUE] [-v[zero | ones | [DIGIT]]]

**DEFAULTS***memgallop* -dpmem -r#0x100000 -aword -con -vzero**DESCRIPTION**

The *memgallop* command is used to test a memory with a galloping value. This is very useful to diagnose an error in the memory where a *DRAM chip* fails. The command will perform the test as described below:

- The command will initialize a memory space with the inverted value from the `PAT_VALUE` option, also called the passive value.
- Then an active value (the value from the `PAT_VALUE` option) is written to the address been tested.
- Afterwards, all the neighbouring addresses of the first address are tested. A neighbouring address is an address which only differs in one bit from the original address. It must still contain the passive value.

- When all neighbouring addresses are tested, the passive value is written back in the original address.
- This is repeated for all addresses in the memory space.

The *memgallop* command has the following options:

**-dDEVICE**

Same description as the *memfast* command.

**-rADDR\_RANGE**

Same description as the *memfast* command.

**-aACCESS**

Same description as the *memfast* command.

**-cCACHE**

Same description as the *memfast* command.

**-vPAT\_VALUE**

This option is used to specify a galloping value for the test. The number of data bits used in the value depends on how many data bits the selected memory has. For a 64 bit memory the test only uses a 32 bit galloping value. The option must be specified as a *word symbol or digit value option*, and the following predefined values can be specified:

**zero** Same as value of 0x00000000.

**ones** Same as value of 0xffffffff.



## EXAMPLES

```

baio 70 -> memgallop -dglobal -vones -abyte,word
Test started at Tuesday 1995-06-06 - 13:04:25
Memory galloping test in Global memory
Start address End address Cache Write Read Active value Passive value
0x00200000 0x00300000 on byte word 0xffffffff 0x00000000
baio 70 ->

```

Figure 11.3-1: Example of the memgallop command

## NOTICES

The command cannot be used while running in *fatal error mode*.

## DIAGNOSTICS

If the test finds errors, then try to diagnose them by means of the *memfast*, *memaddr*, or the *memdata* command.

## BUGS

It is not possible to set the **ACCESS** option to **dword**, because the *double word access* is not implemented in this test.

## COMMAND ERRORS

If the command returns the error code *Options error*, it is because the memory range is not aligned on a 4 bytes address or because the **ACCESS** option is set to **dword**.

## REFERENCES

See also commands:

*set*, *memory*, *memfast*, *memaddr*, *memdata*, *memcmp*, *memdisp*

**memgallop (memory)**

**baio30x command**

**memgallop (memory)**

*This page is intentionally left blank*

**NAME**

*memory* – Test script for memory

**SYNOPSIS**

*memory* [-dMEMORY\_DEVICES] [-tTEST\_FUNCTIONS] [-pDISP\_CMDS]

[-dMEMORY_DEVICES]	[-d[pmem   dmem   nvm   global   subm1   subm2   subm3] [*,*,]]
[-tTEST_FUNCTIONS]	[-t[fast   function] [*,*,]]
[-pDISP_CMDS]	[-p]

**DEFAULTS**

*memory* -dpmem, dmem, nvm, global, subm1, subm2, subm3  
-tfast, function

**DESCRIPTION**

This command is a *script command*, and it is used to run a complete test of the memories connected to the *baio30x* module. The command executes other commands which are all memory tests. It is possible to specify which memories to be tested and which commands of the *memory script* to be executed. This is done by means of the **MEMORY\_DEVICES** and **TEST\_FUNCTIONS** options.

The *memory* command has the following options:

**-dMEMORY\_DEVICES**

This option is used to select the memories to be tested. The option must be specified as a *multi word symbol option*, and the following memories can be specified:

<b>pmem</b>	Program memory.
<b>dmem</b>	Data memory.
<b>nvm</b>	NVM RAM.
<b>global</b>	Global memory.

No memory test is defined for the **subm1**, **subm2** and **subm3** memories.

**-tTEST\_FUNCTIONS**

This option is used to specify which test functions should be selected from the *memory script*. The option must be specified as a *multi word symbol option*, and the following test functions can be specified:

**fast** If this test function is selected, the memories will only be tested fast and not thoroughly.

**function** If this test function is selected, the memories will be tested thoroughly.

**-pDISP\_CMDS**

If this option is specified, the selected commands from the script will only be displayed, not executed.

Without option the command executes all the commands from the scripts.

EXAMPLES

```

baio 70 -> memory -p
Test script for memory
Device Function Command script
pmem fast memfast -dpmem; memcmp -dpmem; memgallop -dpmem -v0;
memgallop -dpmem -vones
dmem fast memfast -ddmem; memcmp -ddmem; memgallop -ddmem -v0;
memgallop -ddmem -vones
nvm fast memfast -dnvm; memcmp -dnvm; memgallop -dnvm -v0; memgallop
-dnvm -vones
global fast memfast -dglobal; memcmp -dglobal -aword32; memgallop
-dglobal -v0; memgallop -dglobal -vones
pmem function memdata -dpmem -v0; memdata -dpmem -vones; memaddr -dpmem
-v0; memaddr -dpmem -vones
dmem function memdata -ddmem -v0; memdata -ddmem -vones; memaddr -ddmem
-v0; memaddr -ddmem -vones
nvm function memdata -dnvm -v0; memdata -dnvm -vones; memaddr -dnvm -v0;
memaddr -dnvm -vones
global function memdata -dglobal -v0; memdata -dglobal -vones; memaddr
-dglobal -v0; memaddr -dglobal -vones; memcmp -dglobal
-abyte -r#0x10000; memcmp -dglobal -ashort -r#0x10000;
memcmp -dglobal -aword -r#0x10000; memcmp -dglobal -aword4
-r#0x10000; memcmp -dglobal -aword8 -r#0x10000; memcmp
-dglobal -aword16 -r#0x10000; memfast -dglobal -r#*

baio 70 ->

```

Figure 11-memory-1: Example of memory command displaying all commands from the script

```

baio 70 -> memory -dglobal -tfast -p
Test script for memory
Device Function Command script
global fast memfast -dglobal; memcmp -dglobal -aword32; memgallop
-dglobal -v0; memgallop -dglobal -vones

baio 70 ->

```

Figure 11-memory-2: Example of memory command selecting a few commands from the script

memory (memory)

baio30x command

memory (memory)

## NOTICES

The command cannot be used while running in *fatal error mode*.

## REFERENCES

See also commands:

*memfast, memaddr, memdata, memgallop, memcmp.*

**NAME**

*menu* - Display all commands

**SYNOPSIS**

*menu* [-ILONG\_FORMAT]  
[-ILONG\_FORMAT] [-l]

**DEFAULTS**

menu

**DESCRIPTION**

The *menu* command is used to display all commands on the *baio30x* module. The command only has one option as shown below:

- l If this option is specified, all commands will be displayed in long format, i.e. the display command names followed by a command text.

If no options are given, the command displays all commands on the unit in short format, i.e. only the command names.

**EXAMPLES**

```
baio 70 -> menu
date      debug      errlog      help        menu        repeat
set       status      testlog     time        version     bfscat
bfsload   bfsis         boot        config      reset       sysbus
test      memory      memfast     memcmp      memaddr     memdata
memgallop memexer      memdisp     ethernet    ethdiag     ethtest
ethmon    ethstat      scsi        scsidiag    scsitest    hdlc
hdlcdiag  hdlctest    hdlcomon   scu         sculeds     scutest
scumon    scuctl       sdreset     sdconf      sddisp      sdloadf
disk      diskrd       diskwr      diskexer    diskformat  diskrbk
tape      taperd       tapewr      taperew     taperet     tapeclr
baio 70 ->
```

Figure 11-menu-1: Commands on the baio30x module in short format

```
baio 50 -> menu -l
date      Read or set date
debug     Call debugger
errlog    Display errors from the error log
help      Display command information and options
menu      Display all commands
repeat    Repeat commands
set       Set or display parameters
status    Display test status
testlog   Display test information from the test log
time      Time a command
version   Display program version
bfscat    Display file contents from Boot File System
bfsload   Load data from Boot File System
bfsls     List contents of the Boot File System
boot      Boot all modules on the Global bus from boot disk
config    System and module configuration
reset     Reset system
sysbus    System Bus exerciser
test      Test script for all tests
memory    Test script for memory
memfast   Memory high speed test
memcmp    Memory move and compare test
memaddr   Memory address bit test
memdata   Memory data bit test
memgallop Memory galloping test
memexer   Memory exerciser
memdisp   Display memory contents
ethernet  Test script for ethernet
ethdiag   Ethernet diagnose test
ethctest  Ethernet transmission test
ethmon    Ethernet network monitor
ethstat   Display ethernet network statistics
scsi      Test script for scsi controller
scsidiag  SCSI diagnose test
scsitest  SCSI data transfer test
hdlc      Test script for HDLC
hdlcdiag  HDLC controller diagnose test
hdlctest  HDLC transmission test
hdlcmon   HDLC monitor line status
scu       Test script for Service Computer Unit
sculeds   Service Computer LEDs' test
scutest   Service Computer test
scumon    Service Computer monitoring
scucntl   Service Computer control
```

(continued on next page)



```

sdreset      Reset SCSI interface
sdconf       Read SCSI device configuration
sddisp       Display SCSI device data
sdloadf      Download firmware to SCSI device
disk         Test script for SCSI disk drives
diskrd       Disk read (CRC) test
diskwr       Disk write/read test
diskexer     Disk exerciser test
diskformat   Format disk
diskrblk     Reassign block or display disk defect on disk
tape         Test script for SCSI tape drives
taperd       Tape read test
tapewr       Tape write/read test
taperew      Rewind tape
taperet      Retain tape
tapeclr      Erase tape
baio 50 ->
    
```

Figure 11-menu-2: Commands on the baio30x module in long format

## NOTICES

The command does not clear *test log* and *error log*.

The command cannot be repeated. It will only run once.

## REFERENCES

See also commands:

*help*

*This page is intentionally left blank*

**NAME**

*repeat* – Repeat commands

**SYNOPSIS**

*repeat* [-mMODE] [NUMBER]

[-mMODE] [-m[on | pass | time | cmd | test | error | off]]

[NUMBER] [DIGIT]

**DEFAULTS**

*repeat* -mon 1

**DESCRIPTION**

The *repeat* command is used to select how many times *test commands* should be executed from the *command line* buffer.

The *repeat* command has the following options:

**-mMODE**

This option is used to specify a repeat mode. It must be specified as a *word symbol option*. One of the following modes can be specified:

- |             |  |
|-------------|--|
| <b>off</b>  | This mode does not repeat commands.  |
| <b>on</b>   | This mode repeats commands forever.  |
| <b>pass</b> | This mode repeats commands until the number of <i>command line</i> buffer passes has reached the value of the <b>NUMBER</b> option.  |
| <b>time</b> | This mode repeats commands until the running time measured in seconds has passed the value of the <b>NUMBER</b> option.              |
| <b>cmd</b>  | This mode repeats commands until the number of commands has passed the value of the <b>NUMBER</b> option.                            |
| <b>test</b> | This mode repeats commands until the number of tests in the <i>test log</i> buffer has passed the value of the <b>NUMBER</b> option. |

repeat (system)

baio30x command

repeat (system)

**error** This mode repeats commands until the number of errors in the *error log* buffer has passed the value of the **NUMBER** option.

### NUMBER

This option is used to specify how many times to repeat the commands. It must be specified as a *digit value option*. If the **MODE** option is set to **off** or **on**, this option has no effect.

If no options are given, the *repeat* command repeats all the *test commands* from the *command line* buffer forever.

### EXAMPLES

```
baio 70 -> memfast; ethtest; repeat
baio 70 -> memfast; ethtest; repeat -mtime 60
```

Figure 11-repeat-1: Example of using the repeat command

In the first example all the *test commands* will be repeated forever, and in the second example all the *test commands* will be repeated for 60 seconds.

### NOTICES

The command does not clear *test log* and *error log*.

The command cannot be executed while running in *fatal error mode*.

If the *repeat* command is not selected in the *command line* buffer, the *repmode parameter* and the *repmo parameter* from the *set* command are used instead.

repeat (system)

baio30x command

repeat (system)

**BUGS**

If the **MODE** option is set to **time**, the commands will maybe run longer time than specified with the **NUMBER** option. This is because the program does not check the time when performing a test.

**REFERENCES**

See also commands:

*set, status*

**repeat (system)**

**baio30x command**

**repeat (system)**

*This page is intentionally left blank*

**reset (boot)**

**baio30x command**

**reset (boot)**

**NAME**

*reset* – Reset system

**SYNOPSIS**

*reset*

**DESCRIPTION**

The *reset* command is used to reset the entire Supermax Enterprise Server. This has the same effect as pressing the **reset button** on the front panel of the Supermax Enterprise Server.

**WARNINGS**

The Supermax Enterprise Server will reset when running the command.

**REFERENCES**

See also commands:

*boot*

**reset (boot)**

**baio30x command**

**reset (boot)**

*This page is intentionally left blank*



**NAME**

*scsi* - Test script for scsi controller

**SYNOPSIS**

*scsi* [-dSCSI\_DEVICES] [-tTEST\_FUNCTIONS] [-pDISP\_CMDS]

[-dSCSI\_DEVICES]           [-d[scsi0 | scsi1] [\*,\*,]]

[-tTEST\_FUNCTIONS]       [-t[fast | function] [\*,\*,]]

[-pDISP\_CMDS]           [-p]

**DEFAULTS**

*scsi* -dscsi0,scsi1 -tfast,function

**DESCRIPTION**

This command is a *script command*, and it is used to run a complete test of the *SCSI* circuits that are mounted on the *baio30x* module. The command executes other commands which are all tests of the *SCSI controller*. It is possible to specify which *SCSI controller* and which commands of the *scsi script* to be executed. This is done by means of the **SCSI\_DEVICES** and **TEST\_FUNCTIONS** options.

The *scsi* command has the following options:

**-dSCSI\_DEVICES**

This option is used to select which *SCSI controller* to be tested. The option must be specified as a *multi word symbol option*, and the following *SCSI controllers* can be specified:

**scsi0**     *SCSI controller 0.*

**scsi1**     *SCSI controller 1.*

**-tTEST\_FUNCTIONS**

This option is used to specify which test functions should be selected from the *scsi script*. The option must be specified as a *multi word symbol option*, and the following test functions can be specified:

- fast** If this test function is selected, the *SCSI controller* will only be tested fast and not thoroughly. The selected commands will only use the *program memory* when running the tests.
- function** If this test function is selected, the *SCSI controller* will be tested thoroughly. The selected commands will use the *global memory* when running the tests.

**-pDISP\_CMDS**

If this option is specified, the selected commands from the script will only be displayed, not executed.

Without option the command executes all the commands from the scripts.

**EXAMPLES**

```

baio 70 -> scsi -p
Test script for scsi controller
Device Function Command script
scsi0 fast scsidiag -dscsi0; scsitest -dscsi0
scsi1 fast scsidiag -dscsi1; scsitest -dscsi1
scsi0 function scsidiag -dscsi0 -r*-*; scsitest -dscsi0; scsitest -dscsi0
-smglobal -b0x11c; scsitest -dscsi0 -dmglobal -b0x11c
scsi1 function scsidiag -dscsi1 -r*-*; scsitest -dscsi1; scsitest -dscsi1
-smglobal -b0x11c; scsitest -dscsi1 -dmglobal -b0x11c
baio 70 ->

```

Figure 11-scsi-1: Example of the scsi command displaying all commands from the script

scsi (scsi)

baio30x command

scsi (scsi)

```
baio 70 -> scsi -tfast -p
Test script for scsi controller
Device  Function  Command script
scsi0   fast    scsidiag -dscsi0; scsitest -dscsi0
scsi1   fast    scsidiag -dscsi1; scsitest -dscsi1
baio 70 ->
```

Figure 11-scsi-2: Example of the scsi command selecting a few commands from the script

## NOTICES

The command cannot be used while running in *fatal error mode*.

The command will not test the *SCSI* interface.

## REFERENCES

See also commands:

*scsidiag*, *scsitest*.

**scsi (scsi)**

**baio30x command**

**scsi (scsi)**

*This page is intentionally left blank*

## NAME

*scsidiag* – SCSI diagnose test

## SYNOPSIS

```
scsidiag [-dDEVICE] [-fFUNCTION] [-regREGISTERS] [-rADDR_RANGE] [-aACCESS] [-nNO_OF_TEST] [-pPATTERN] [-vPAT_VALUE]
```

[-dDEVICE]            [-d[scsi0 | scsi1]]

[-fFUNCTION]        [-f[reg | script][\*,\*,]]

[-regREGISTERS]    [-reg[BIT RANGE FROM 0 TO 31]]

[-rADDR\_RANGE]    [-r[START | #SIZE | START-END | START#SIZE]]

[-aACCESS]        [-a[word | dword | short | byte][\*,\*]]

[-nNO\_OF\_TEST]    [-nDIGIT]

[-pPATTERN]       [-p[random | value | count | cntdown | current]]

[-vPAT\_VALUE]     [-vDIGIT]

## DEFAULTS

```
scsidiag -dscsi0 -freg,script -reg0-7 -r#0x100000  
-aword -n4096 -prandom -vX
```

## DESCRIPTION

The *scsidiag* command performs a simple diagnostics test on an *SCSI controller*. This is done by running a register test and an *SCSI script* test executed by the controller in the *program memory*. This is very useful to diagnose an error in the *SCSI controller* and in the communication between the controller and the *CPU* and the *program memory*. It is possible to select which type of test the command shall perform on the *SCSI controller*. This is done by means of the **FUNCTION** option.

The *SCSI controller* register test will run a write/read test on certain registers in the controller. It is possible to specify up to 8 different registers in the controller. This is done by means of the **REGISTERS** option, which can have the values 0 - 7.

First the program will write 32 bit data value in all registers, and then it will read and check each register for the correct value. This will be repeated until the number of tests has passed the value of the `NO_OF_TEST` option. The written data pattern is specified with the `PATTERN` and `PAT_VALUE` options.

The *SCSI controller script* test will be performed as follows:

- The memory array is initialized with a 32 bit value that consists of an illegal *SCSI script* instruction added with the memory address.
- The program will initialize a 256 byte *SCSI script* in the memory range, that consists of different types of *SCSI controller* instructions.
- Then the program will start the *SCSI controller* to execute the script. After the controller has executed the script, the program will check the status and the script results in the controller.
- The script initializing, executing, and status checking will be repeated until the number of tests have passed the value of the `NO_OF_TEST` option.

The *scsidiag* command has the following options:

**-dDEVICE**

Same description as the *scsitest* command.

**-fFUNCTION**

This option is used to specify which test functions the command shall perform. The option must be specified as a *multi word symbol option* and the following tests can be specified:

**reg** If this test function is selected, the command will perform register tests in the *SCSI controller*.

**script** If this test function is selected, the command will initialize the *SCSI controller* to run *SCSI script* tests in the *data memory*.

**-regREGISTERS**

This option is used to specify which *SCSI controller* registers shall be used during the register test. The option must be specified as a *bit range option*. Default, all 8 registers in the *SCSI controller* are selected. If the **FUNCTION** option is not set to **reg**, this option has no effect.

**-rADDR\_RANGE**

This option is used to specify a memory address range in the *program memory* where the *SCSI controller* has to execute the *SCSI scripts*. The option must be specified as a *value range option*. This is only used, when the **FUNCTION** option is set to **script**. Default, the command will reserve the next free memory array with the size of 0x10000 bytes.

**-aACCESS**

This option is used to specify which type of access the command shall perform during write and read access to an *SCSI controller* register. The option must be specified as a *double word symbol option*. The access type can be different for the write and the read access. The first word symbol will be used for write access and the second for read access. The following access types can be specified:

- word**      *Word accesses* will be performed to the register.
- dword**     *Word accesses* will be performed to the register.
- short**     *Short accesses* will be performed to the register.
- byte**      *Byte accesses* will be performed to the register.

If the **FUNCTION** option is not set to **reg**, this option has no effect.

**-nNO\_OF\_TEST**

This option is used to specify how many times the register test and the *SCSI script* test shall be performed. It must be specified as a *digit value option*. Default number of test for the register tests is 4096. For the *SCSI script* test it depends on how many bytes are reserved for the *SCSI script* test, calculated as follows:  
Size of memory / 256.

**-pPATTERN**

Same description as the *scsitest* command. If the **FUNCTION** option is not set to **reg**, this option has no effect.

**-vPAT\_VALUE**

Same description as the *scsitest* command. If the **FUNCTION** option is not set to **reg**, this option has no effect.

**EXAMPLES**

```

baio 70 -> scsidiag -r**
Test started at Tuesday 1995-06-27 - 14:04:44
SCSI diagnose test on scsi0
Address/register range  Function  No of test  Access type  Pattern value
0-7                    reg      4096       word  word  random 0x2ff01e0b

Test started at Tuesday 1995-06-27 - 14:04:44
SCSI diagnose test on scsi0
Address/register range  Function  No of test  Access type  Pattern value
0x00090000-0x00400000  script   14080
baio 70 ->
    
```

Figure 11-scsidiag-1: Example of the scsidiag command

**NOTICES**

The command cannot be executed while running in *fatal error mode*.

When running the *SCSI script* test, and the memory range size is smaller than 256 bytes no *SCSI script* test will be performed.

The command stores test information in the *test log* buffer for each type of test function.



## COMMAND ERRORS

If the command returns the error code *Options error*, it is because the memory range is not aligned on a 4 bytes address.

## REFERENCES

See also commands:

*scsi*, *scsitest*

*This page is intentionally left blank*

**NAME**

*scsitest* – SCSI data transfer test

**SYNOPSIS**

*scsitest* [-saSRC\_ADD] [-smSRC\_MEM] [-daDESC\_ADD] [-dmDESC\_MEM] [-dDEVICE] [-sMEM\_SIZE] [-bBLOCK\_SIZE] [-cCOMPARE] [-pPATTERN] [-vPAT\_VALUE]

[-saSRC_ADD]	[-saDIGIT]
[-smSRC_MEM]	[-sm[pmem   global]]
[-daDESC_ADD]	[-daDIGIT]
[-dmDESC_MEM]	[-dm[pmem   global]]
[-dDEVICE]	[-d[scsi0   scsi1]]
[-sMEM_SIZE]	[-sDIGIT]
[-bBLOCK_SIZE]	[-bDIGIT]
[-cCOMPARE]	[-c[on   off]]
[-pPATTERN]	[-p[random   value   count   cntdown   current]]
[-vPAT_VALUE]	[-vDIGIT]

**DEFAULTS**

*scsitest* -dscsi0 -saX -smpmem -daX -dmpmem -dscsi0  
-s0x100000 -b0x100 -con -prandom -vX

**DESCRIPTION**

The *scsitest* command is used to perform an *SCSI controller* data transfer test. This is done by initializing the *SCSI controller* to execute an *SCSI script* in the *program memory*. The *SCSI script* will copy the data from one memory array to another. This is very useful to diagnose faults on *SCSI controller* circuits. The command will not check the *SCSI interface* that is connected to the *SCSI controller*.

The test will be performed as follows:

- The source memory array is initialized with a data pattern type, specified with the **PATTERN** and **PAT\_VALUE** options.
- Now the *SCSI controller* will be initialized to execute the *SCSI script* that copies a block of data from the source to the destination memory. The number of bytes in a block (block transfer size) is specified with the **BLOCK\_SIZE** option.
- The copied data in the destination memory is read and compared with the source memory contents. This is only done, if the **COMPARE** option is set to **on**.
- The data copying and comparison will be repeated until all data have been copied.

The *scsitest* command has the following options:

**-saSRC\_ADD**

This option is used to specify a start memory address for the source memory array. The option must be specified as a *digit value option*. Default, the command will reserve the next free source memory array. The size that will be reserved in the source memory depends on the value of the **MEM\_SIZE** option.

**-smSRC\_MEM**

This option is used to select which memory to be used as the source memory. The option must be specified as a *word symbol option*, and the following memories can be specified:

**pmem**    *Program memory.*

**global**    *Global memory.*

**-daDESC\_ADD**

Same description as the **SRC\_ADD** option, only this is the destination start memory address.

**-dmDESC\_MEM**

Same description as the **SRC\_MEM** option, only this is the destination memory device.

**-dDEVICE**

This option is used to select which *SCSI controller* the test has to use. The option must be specified as a *word symbol option*, and the following *SCSI controllers* can be specified:

**scsi0**     *SCSI controller 0.*

**scsi1**     *SCSI controller 1.*

**-sMEM\_SIZE**

This option is used to specify the number of bytes to be reserved in the source and destination memory. It must be specified as a *digit value option*. Default, the command will reserve up to 1 Mb in the source and destination memory.

**-bBLOCK\_SIZE**

This option is used to specify the block transfer size. I.e. the number of bytes that are copied every time the *SCSI controller* executes the *SCSI script*. It must be specified as a *digit value option*. Default, the command will copy 256 bytes per block transfer.

**-cCOMPARE**

This option is used to specify the compare mode for the source and destination memory array. The option must be specified as a *word symbol option*, and the following two compare modes can be specified:

**on**        Data comparison will be performed.

**off**        No data comparison will be performed.

**-pPATTERN**

This option is used to specify a data pattern type together with the **PAT\_VALUE** option. The data pattern is used to initialize the source memory array. The **PATTERN** option must be specified as a *word symbol option*, and the following pattern types can be specified:

- random** This pattern type will generate a 32 bit random data pattern from the **PAT\_VALUE**. If the same **PAT\_VALUE** is specified, the same random pattern is generated.
- value** This pattern type will generate a 32 bit data value specified with the **PAT\_VALUE** option.
- count** This pattern type will generate a 32 bit count data pattern calculated as follows: **PAT\_VALUE** + (offset / 4).
- cntdown** This pattern type will generate a 32 bit count-down data pattern calculated as follows: **PAT\_VALUE** - (offset / 4).
- current** This pattern type will use the current values in the source memory address range.

**-vPAT\_VALUE**

This option is used to specify a data value for the pattern type. The option must be specified as a *digit value option*. Default, the value is a value from the internal system clock. The option only has effect if the **PATTERN** option is not set to **current**.

## EXAMPLES

```

baio 70 -> scsitest -smglobal -pcount -v0 -b0x10000 -dscsil
Test started at Monday 1995-06-26 - 14:00:25
SCSI data transfer test on scsil
Src-addr.  Des-addr.  Length  Blk-size  Direction  Cmp Pattern Value
0x00200000 0x00090000 0x0100000 0x0100000 global->pmem on count 0x00000000
baio 70 ->

```

Figure 11-scsitest-1: Example of the scsitest command

## NOTICES

The command cannot be executed while running in *fatal error mode*.

The source and destination memory cannot both be set to *global memory*.

If the number of bytes to be copied is not divisible by the block transfer size, the block transfer size will be reduced for the last copying to the destination memory.

If the **COMPARE** option is set to **on** and block transfer size (**BLOCK\_SIZE** option) is not set to a 4 bytes aligned value, the comparison will not be performed on some of the data in the beginning or in the end of the transferred blocks. This is because the program can only compare 32 bit data, if the source and destination address are aligned on a 4 bytes address. If the block transfer size is smaller than 4, the comparison will not be performed.

## DIAGNOSTICS

If the test finds errors, then try to diagnose them by means of the *scsiddiag* command.

## COMMAND ERRORS

If the command returns the error code *Options error*, it is because the **MEM\_SIZE**, **SRC\_ADD** or **DESC\_ADD** options is not aligned on a 4 bytes address, or because the options **SRC\_MEM** and **DESC\_MEM** are both set to *global memory*.

## REFERENCES

See also commands:

*scsi*, *scsidiag*, *sdconf*



**NAME**

*scu* – Test script for Service Computer Unit

**SYNOPSIS**

```
scu [-dSCU_DEVICES] [-tTEST_FUNCTIONS] [-pDISP_CMDS]
[-dSCU_DEVICES]      [-d[subm1 | subm2 | subm3] [*,*,]]
[-tTEST_FUNCTIONS]   [-t[function | leds] [*,*,]]
[-pDISP_CMDS]       [-p]
```

**DEFAULTS**

```
scu -dALL_SCU_SUBMODULE -tfunction,leds
```

**DESCRIPTION**

This command is a *script command*, and it is used to run a complete test of the *scu30x* (Service Computer Unit) submodules that are installed on the *baio30x* module. This includes checking of all possible external input/output signals that are connected to the *scu* submodule. The command executes other commands which are all tests of the *scu30x* submodules. It is possible to specify which submodules and which commands from the *scu script* to be executed. This is done by means of the **SCU\_DEVICES** and **TEST\_FUNCTIONS** options.

The *scu* command has the following options:

**-dSCU\_DEVICES**

This option is used to select which *scu30x* submodule to be tested. The option must be specified as a *multi word symbol option*, and the following *scu30x* submodules can be specified:

- subm1**     Submodule 1. Only available if the *scu30x* submodule is installed.
- subm2**     Submodule 2. Only available if the *scu30x* submodule is installed.
- subm3**     Submodule 3. Only available if the *scu30x* submodule is installed.

### -tTEST\_FUNCTIONS

This option is used to specify which test functions should be selected from the *scu script*. The option must be specified as a *multi word symbol option*, and the following test function can be specified:

- function**      This test function will test the *scu30x* submodule thoroughly including some external input signals that are monitored by the submodule.
- leds**            This test function will turn on and off all *LEDs* that are connected externally to the *scu30x* submodule.

### -pDISP\_CMDS

If this option is specified, the selected commands from the script will only be displayed, not executed.

Without options the command executes all the commands from the scripts.

### EXAMPLES

```

baio 40 -> scu -p
Test script for Service Computer Unit
Device Function Command script
subm1 function memfast -dsubm1; memcmp -dsubm1; mengallop -dsubm1; scutest
-dsubm1;
subm2 function memfast -dsubm2; memcmp -dsubm2; mengallop -dsubm2; scutest
-dsubm2;
subm3 function memfast -dsubm3; memcmp -dsubm3; mengallop -dsubm3; scutest
-dsubm3;
subm1 leds sculeds -dsubm1 -ton; sculeds -dsubm1 -toff -w20;
subm2 leds sculeds -dsubm2 -ton; sculeds -dsubm2 -toff -w20;
subm3 leds sculeds -dsubm3 -ton; sculeds -dsubm3 -toff -w20;
    
```

Figure 11-hdlc-1: Example of the scu command displaying all commands from the script

### NOTICES

The command cannot be used while running in *fatal error mode*.

The Service Computer must be connected to the external input/output signals in the Supermax Enterprise Server, before running this test.

#### REFERENCES

See also commands:

*memcmp, memfast, memgallop, scutest.*

**scu (scu)**

**baio30x command**

**scu (scu)**

*This page is intentionally left blank*

**NAME**

*scucntl* – Service Computer control

**SYNOPSIS**

*scucntl* [-dDEVICE] [-cCOMMAND] [-vVALUE]

[-dDEVICE]            [-d[subm1 | subm2 | subm3]]

[-cCOMMAND]        [-c[lcdoff | lcdon | lcdcont | getrtc | setrtc |  
pownorm | powlow | powhigh | powoff | set-  
fan0 | setfan1]]

[-vVALUE]           [-vDIGIT]

**DEFAULTS**

*scucntl* -dFIRST\_SCU\_SUBMODULE -v0

**DESCRIPTION**

The *scucntl* command is used to execute a control command on the *scu30x* submodule (Service Computer Unit). The command that should be executed is specified by the **COMMAND** options. The following control commands can be executed:

- Read or set the *RTC* (Real Time Clock).
- *LCD* display controls.
- Power supply controls.
- Fan level.

After the Diagnostic Programs have been started, the following will be done on the Service Computer Unit:

- Current date and time is set on the unit to the *RTC*.
- Turn off the *LCD* display.
- Set the *LCD* display contrast to 0.
- Set the output voltage on the +5V and +3.3V supply is set to its nominal value.
- Set the fan level for group 0 and 1 to maximum level.

The *scucntl* command has the following options:

**-dDEVICE**

Same description as the *scutest* command.

**-cCOMMAND**

This option is used to specify what the *scucntl* command should do. The option must be specified as a *word symbol option*. The following command types can be specified:

- |                |  |
|----------------|--|
| <b>lcdoff</b>  | This command will turn off the <i>LCD</i> display.   |
| <b>lcdon</b>   | This command will clear and turn on the <i>LCD</i> display.  |
| <b>lcdcont</b> | This command will set the <i>LCD</i> display contrast together with the <b>VALUE</b> option. The display contrast can be set to a value between 0 and 255.             |
| <b>getrtc</b>  | This command will set current date and time on the unit to the <i>RTC</i> on the <i>scu30x</i> submodule.  |
| <b>setrtc</b>  | This command will set the <i>RTC</i> on the <i>scu30x</i> submodule to the current date and time on the unit.  |
| <b>pownorm</b> | This command will set the output voltage on the +5V and +3.3V supply to its nominal value (5.00 and 3.30 Volt).  |
| <b>powlow</b>  | This command will set the output voltage on the +5V and +3.3V supply to its low value (4.75 and 3.14 Volt).  |
| <b>powhigh</b> | This command will set the output voltage on the +5V and +3.3V supply to its high value (5.25 and 3.46 Volt).   |
| <b>powoff</b>  | This command will turn off the power to the Supermax Enterprise Server. This will only work if the <b>key switch</b> on the front panel of the computer is set to run. |

scuctl (scu)

baio30x command

scuctl (scu)

**setfan0** This command will set the fan level for group 0 together with the **VALUE** option. The fan level can be set to a value between 1 and 32. After 30 seconds a watchdog time-out will set the fan to maximum level.

**setfan1** Same as the **setfan0**, but for fan group 1.

#### **-vVALUE**

This option is used to specify a value to be used by the command when the **COMMAND** option is set to **lcdcont**, **setfan0** or **setfan1**. The value must be specified as a *digit value option*.

Without the **COMMAND** option the command turns off the **LCD** display.

#### **EXAMPLES**

```
baio 70 -> scuctl -cpowlow
The power supply output voltage will be set to low value
baio 70 ->
```

Figure 11-scuctl-1: Example of the scuctl command

#### **WARNINGS**

If the command is used to turn off the power on the power supply, the operation is suicidal.

If the command is used to set the fan levels, it is recommended not to repeat the command, because the program does not measure the temperature in the Supermax Enterprise Server. The result of this can be that the temperature in the computer will raise, and the modules may be damaged!

**scucntl (scu)****baio30x command****scucntl (scu)****BUGS**

The command cannot save the *SCU* control information in the *test log* buffer. The information will only be displayed on the *console terminal*.

**REFERENCES**

See also commands:

*date, scu, sculeds, scumon, scutest*



**NAME**

*sculeds* – Service Computer LEDs test

**SYNOPSIS**

```
sculeds [-dDEVICE] [-fFUNCTION] [-tLED_TEST] [-wWAIT_MS] [-rROW_POS] [-cCOLUMN_POS] [-vLED_VALUE]
[-dDEVICE]           [-d[subm1 | subm2 | subm3]]
[-fFUNCTION]         [-f[activity | key | disk | lcd][L*,*,]]
[-tLED_TEST]         [-t[step | off | on | manual]]
[-wWAIT_MS]          [-wDIGIT]
[-rROW_POS]           [-rDIGIT]
[-cCOLUMN_POS]       [-cDIGIT]
[-vLED_VALUE]        [-vDIGIT]
```

**DEFAULTS**

```
sculeds -dFIRST_SCU_SUBMODULE -factivity,key,disk,lcd
-tstep -w0 -rALL -cALL -v0xffffffff
```

**DESCRIPTION**

The *sculeds* command is used to exercise the *LEDs* that are connected externally to the *scu30x* submodule (Service Computer Unit). This is done by turning the *LEDs* off and on. Different types of *LED* tests can be selected by means of the *LED\_TEST* option.

The following 4 types of *LEDs* can be exercised on the *scu30x* submodule:

- *Activity LEDs* mounted on the front of the Supermax Enterprise Server panel.
- *Button LED* mounted on the front of the Supermax Enterprise Server panel.
- *Disk LEDs* mounted on the front of the disk(s).
- *LCD display* mounted on the front of the Supermax Enterprise Server panel.

It is possible to exercise a specific *LED*. This is done by means of the

ROW\_POS, COLUMN\_POS and LED\_VALUE options. The following table is a description of the maximum number of LEDs for each type:

Type	Numbers	Rows	Columns	LED value
Activity	16 LEDs	1	1	16 bits
Key	Up to 4 LEDs	1	1	4 bits
Disk	Up to 512 LEDs	8	16	4 bits
LCD	1024 entries	8	128	1 bit

Figure 11-sculeds-1: Number of LEDs for each type

The *sculeds* command has the following options:

**-dDEVICE**

Same description as the *scutest* command.

**-fFUNCTION**

This option is used to specify which type of LEDs the command has to use. The option must be specified as a *multi word symbol option* and the following LEDs can be specified:

- activity** If this function is selected, the command will use the *Activity LEDs*.
- key** If this function is selected, the command will use the *Button LED*.
- disk** If this function is selected, the command will use the *Disk LEDs*.
- lcd** If this function is selected, the command will use the *LCD display*.

**-tLED\_TEST**

This option is used to specify which LED tests the command shall perform. The option must be specified as a *word symbol option*, and the following tests can be specified:

- step** This will first turn all *LEDs* off and then it will turn the *LEDs* on and off one by one.
- off** This will turn the *LEDs* off.
- on** This will turn the *LEDs* on.
- manual** This will turn the *LEDs* off or on, depending on the value of the *LED\_VALUE* option.

**-wWAIT\_MS**

This option is used to specify a delay between each access to the *LEDs*. This is very useful, if you want to see a single change of the *LEDs*, while running the *LED* step test. It must be specified as a *digit value option* and the value is given in milliseconds.

**-rROW\_POS**

This option is used to specify a specific *LED* row address. It must be specified as a *digit value option* and a value up to 7 can be specified. Default, the command will use all *LEDs* row addresses.

**-cCOLUMN\_POS**

This option is used to specify a specific *LED* column address. It must be specified as a *digit value option* and a value up to 127 can be specified. Default, the command will use all *LED* column addresses.

**-vLED\_VALUE**

This option is used to specify an *LED* bit manually. A zero on a bit indicates that the *LED* will be turned off and a one that it will be turned on. The option must be specified as a *digit value option*. If the *LED\_TEST* option is not set to **manual**, this option has no effect.

## EXAMPLES

```
baio 70 -> sculeds -w50
Test started at Monday 1995-07-03 - 09:40:22
Service Computer LEDs test on submodule 1
Function Command Row Column Delay ms. Leds value
activity step all all 50

Test started at Monday 1995-07-03 - 09:40:23
Service Computer LEDs test on submodule 1
Function Command Row Column Delay ms. Leds value
key step all all 50

Test started at Monday 1995-07-03 - 09:40:23
Service Computer LEDs test on submodule 1
Function Command Row Column Delay ms. Leds value
disk step all all 50

Test started at Monday 1995-07-03 - 09:40:49
Service Computer LEDs test on submodule 1
Function Command Row Column Delay ms. Leds value
lcd step all all 50
baio 70 ->
```

Figure 11-sculeds-2: Example of the sculeds command

## NOTICES

The command cannot be executed while running in *fatal error mode*.

The Service Computer must be connected to the external *LEDs* in the Supermax Enterprise Server, before running this test.

To check whether the *LEDs* can be turned off and on correctly, it is recommended to check this by visually inspection.

## REFERENCES

See also commands:

*scucntl*, *scutest*, *scumon*

**NAME**

*scumon* – Service Computer monitoring

**SYNOPSIS**

*scumon* [-dDEVICE] [-fFUNCTION]

[-dDEVICE] [-d[subm1 | subm2 | subm3]]

[-fFUNCTION] [-f[rtc | power | volt | temp | fan | input |  
lcd][,\*,\*]]

**DEFAULTS**

*scumon* -dsubm1 -frtc, power, volt, temp, fan, input, lcd

**DESCRIPTION**

The *scumon* command is used to monitor a *scu30x* submodule (Service Computer Unit) including all external input signals that are connected to the submodule. The command displays all monitoring information. The command also checks the monitoring information for correct values. The results is only displayed, not saved in the *error log* buffer. The following monitor information can be displayed:

- Date and time from the *RTC* (Real Time Clock).
- Status from the power supplies.
- Measurements of the output voltages of the power supplies.
- Measurements of the temperature in the Supermax Enterprise Server cabinet.
- Status and levels from the fans.
- Status from the general input signals.
- Status from the *LCD* display.

The *scumon* command has the following options:

**-dDEVICE**

Same description as the *scutest* command.

**-fFUNCTION**

This option is used to specify which monitor functions the command shall display. The option must be specified as a *multi word symbol option*, and the following functions can be specified:

- rtc** Same description as the *scutest* command, but the status information is only displayed.
- power** Same description as the *scutest* command, but the status information is only displayed.
- volt** Same description as the *scutest* command, but the status information is only displayed.
- temp** Same description as the *scutest* command, but the status information is only displayed.
- fan** Same description as the *scutest* command, but the status information is only displayed.
- input** If this function is selected, the command will display the following general input signals to the *scu30x* sub-module:
- *Reset button.*
  - *Key switch (System).*
  - *Shutdown button.*
  - *Modem input.*
  - *UPS input.*
  - *Alarm input.*
- lcd** If this function is selected, the command will display the *LCD* status and the current *LCD* display contrast value.

### EXAMPLES

```

baio 70 -> scumon
Test started at Monday 1995-07-03 - 09:39:06
Service Computer monitoring on submodule 1

Real Time Clock (RTC) status.
Date from RTC:    Monday 1995-07-03 - 09:39:07 GMT -0100  Clock ok
Power supply status.
Power supply no:  0    1    2    3    4    5    6    7
Installed:        yes  yes  no   no   no   no   no   no
Temperature status:ok  ok
Power status:     ok   ok
Voltages measurements.
A/D channel no:  0    1    2    3    4    5    6    7
Expected voltage: 3.30 5.00 12.00      2.10 2.10
Measured voltage: 3.41 5.05 12.12      2.10 2.08
Voltages check:  ok  ok  ok           ok  ok
Temperature measurements.
A/D channel no:  0    1    2    3    4    5    6    7
Temperature group: in0  out0 out0  in1  out1 out1  out1  out1
Measured temp:   24.3 25.7 25.3 23.9 27.6 26.2
Temperature check:ok  ok  ok  ok  ok  ok
Fan control.
Fan group no:    0      1
Fan level:       100.0% 100.0%
Fan status:      ok    ok
Input signal status.
Input signal:    Reset  System  Shutdown  Modem  UPS  Alarm
Signal status:  off   on    off      off   unmount unmount
LCD status.
LCD display:    off    Contrast: 0
baio 70 ->

```

Figure 11-scumon-1: Example of the scumon command

## NOTICES

The command cannot be executed while running in *fatal error mode*.

The command always displays the Service Computer monitoring information, no matter how the *dispmode parameter* is set.

When displaying the monitor information about the fan levels, it is not necessarily the real fan levels (because of a watchdog time-out). It is only information about what level the *scucntl* command has set them to.

## DIAGNOSTICS

Execute the *sculeds* and *scutest* commands to run a complete test of the *scu30x* submodule.

## REFERENCES

See also commands:

*scucntr*, *sculeds*, *scutest*, *set*



**NAME**

*scutest* – Service Computer test

**SYNOPSIS**

*scutest* [-dDEVICE] [-fFUNCTION] [-tTYPE]

[-dDEVICE] [-d[subm1 | subm2 | subm3]]

[-fFUNCTION] [-f[nvm | rtc | power | volt | temp | fan][,\*,\*,]]

[-tTYPE] [-t[extern | intern]]

**DEFAULTS**

*scutest* -fnvm,rtc,power,volt,temp,fan -textern  
-dFIRST\_SCU\_SUBMODULE

**DESCRIPTION**

The *scutest* command is used to run a complete test of the *scu30x* submodule (Service Computer Unit) which also includes checking of some external input signals connected to the submodule (e.g power supply status, voltage and temperature checking, and fan tacho status). It is possible to test the submodule without checking the external input signals. This is done by means of the **TYPE** option.

It is also possible to select which type of test function the command shall perform on the submodule. This is done by means of the **FUNCTION** option.

The *scutest* command has the following options:

**-dDEVICE**

This option is used to select which *scu30x* submodule the test has to use. The option must be specified as a *word symbol option*, and the following submodules can be specified:

- subm1** Submodule 1. Only available if the *scu30x* submodule is installed.
- subm2** Submodule 2. Only available if the *scu30x* submodule is installed.
- subm3** Submodule 3. Only available if the *scu30x* submodule is installed.

**-fFUNCTION**

This option is used to specify which test functions the command shall perform. The option must be specified as a *multi word symbol option*, and the following tests can be specified:

- nvm** If this function is selected, the command will perform a memory high speed test in the *NVM* with random data pattern.
- rtc** If this function is selected, the command will check whether the *RTC* is running and the date, time, and time-zone are valid.
- power** If this function is selected, the command will check the power and temperature status from the power supplies in the Supermax Enterprise Server.
- volt** If this function is selected, the command will measure the output voltages of the power supplies and check whether the values are in the range of the predefined minimum and maximum values. All voltage measurements are average values of 8 measurements.

The following tables describe the compare values for the different voltage channels:

Ch no.	Voltage	Minimum	Maximum
0	3.30V	2.97V	3.63V
1	5.00V	4.75V	5.25V
2	12.00V	11.30V	12.70V
4	2.10V	2.01V	2.19V

Figure 11-scutest-1: Normal voltage operating range

Ch no.	Voltage	Minimum	Maximum
0	3.14V	2.81V	3.47V
1	4.75V	4.50V	5.00V
2	12.00V	11.30V	12.70V
4	2.10V	2.01V	2.19V

Figure 11-scutest-2: Low voltage operating range

Ch no.	Voltage	Minimum	Maximum
0	3.46V	3.13V	3.79V
1	5.25V	5.00V	5.50V
2	12.00V	11.30V	12.70V
4	2.10V	2.01V	2.19V

Figure 11-scutest-3: High voltage operating range

**temp** If this function is selected, the command will measure the temperature in the Supermax Enterprise Server cabinet and check whether the values are in the range of the predefined minimum and maximum values. All temperature measurements are average values of 8 measurements. The following table describes the compare values for the different channels:

Grp	Ch	Type	Minimum	Maximum
0	0	Input	10.0°C	38.0°C
	1	Output	10.0°C	45.0°C
	2	Output	10.0°C	45.0°C
1	3	Input	10.0°C	38.0°C
	4	Output	10.0°C	58.0°C
	5	Output	10.0°C	58.0°C
	6	Not used		
	7			

Figure 11-scutest-4: Temperature operating range

**fan** If this function is selected, the command will check the tachometer status from the fans that are mounted in the Supermax Enterprise Server cabinet. The following table describes the location for the different fan groups:

Group	Unit	Fan location
0	0	Front devices
	1	Not used
1	0	Hardware modules
	1	Not used

Figure 11-scutest-5: Fan groups

**-tTYPE**

This option is used to specify which test type the command has to use. The option must be specified as a *word symbol option*, and the following two test types can be specified:

**extern** This will test the *scu30x* submodule as described above. This means that the submodule and some external input signals that are connected to the submodule will be tested.

**intern** Only internal test will be performed on the *scu30x* submodule. This means that no check will be performed on the following external input signals:

- Power supply status.
- Power supply output voltages.
- Temperature sensors.
- Fan tacho status.

## EXAMPLES

```
baio 70 -> scutest
Test started at Monday 1995-07-03 - 09:38:49
Service Computer test on submodule 1
Function: nvm           Test type: extern

Test started at Monday 1995-07-03 - 09:38:49
Service Computer test on submodule 1
Function: rtc          Test type: extern

Test started at Monday 1995-07-03 - 09:38:49
Service Computer test on submodule 1
Function: power        Test type: extern

Test started at Monday 1995-07-03 - 09:38:49
Service Computer test on submodule 1
Function: volt         Test type: extern

Test started at Monday 1995-07-03 - 09:38:50
Service Computer test on submodule 1
Function: temp         Test type: extern

Test started at Monday 1995-07-03 - 09:38:50
Service Computer test on submodule 1
Function: fan          Test type: extern

baio 70 ->
```

Figure 11-scutest-6: Example of the scutest command

## NOTICES

The command cannot be executed while running in *fatal error mode*.

The command is not able to check the external *LEDs* and the *LCD* display. In order to do that execute the *sculeds* command.

If the **TYPE** option is set to **extern**, the Service Computer must be connected to the external input/output signals in the Supermax Enterprise Server, before running this test.

## DIAGNOSTICS

If the test finds errors, then try to diagnose them by means of the *scumon* command.

If the test finds errors in the *NVM RAM*, then try to diagnose them by means of the memory group commands (eg. *memfast*).

## REFERENCES

See also commands:

*memfast, scu, scucntr, sculeds, scumon*

**scutest (scu)**

**baio30x command**

**scutest (scu)**

*This page is intentionally left blank*



sdconf (SCSI device)

baio30x command

sdconf (SCSI device)

**NAME***sdconf* – Read SCSI device configuration**SYNOPSIS***sdconf* [-dSCSI\_DEVICES]

[-dSCSI\_DEVICES] [-d[00 | 01 | 02 | 03 | 04 | 05 | 06 | 10 | 11  
| 12 | 13 | 14 | 15 | 17 | scsi0 | scsi1][,\*,\*]]

**DESCRIPTION**

The *sdconf* command is used to read and display configuration on an *SCSI device*. The command can either read the configuration from a selected *SCSI device* or it can show configuration of the *SCSI devices* which are already available for other commands.

The command always displays the configuration on the *SCSI devices*, no matter how the *dispmode parameter* is set.

When reading the configuration on an *SCSI device* the following is performed:

- Check that it is present.
- Initialize synchronous and wide data transfer, depending on how the *scsimode parameter* is set and what the device can do.
- Read *SCSI device type*.
- Power-up the device, if it is a disk device type.
- Wait for the device to be ready for access.
- Read manufacturer's name, serial number and the firmware revision number.
- Read the block size and number of blocks on the device.

When the command has successfully connected to the *SCSI device*, the *SCSI channel/id number* is available for other commands using the *SCSI device*.

The program also inserts the *SCSI channel/id number* in the *diskdev parameter* or *tapedev parameter* dependent on the device type. (Removable disks can not be inserted in the parameter). If no device is accessible, the device number will be deleted from the parameters.

**NOTE**

The *scsimode*, *diskdev* and *tapedev* parameters can be changed by the *set* command.

When displaying the configuration the following information is displayed:

- SCSI channel and ID number.
- SCSI *device* type (disk, tape. etc).
- Manufacturer's name, serial number, and the device revision number. If the program recognizes the device, a short manufacturer name is displayed too.
- Block size and number of blocks on the device.
- The *SCSI device* interface level and information telling whether the device can perform synchronous and wide data transfer.

The *sdconf* command only has one option as shown below:

**-dSCSI\_DEVICES**

This option is used to select from which *SCSI device(s)* the configuration should be read. The option must be specified as a *multi word symbol option*.

Without option the command only displays the *SCSI devices* whose configuration has already been read by the *sdconf* command.

EXAMPLES

```

baio 70 -> sdconf -dscsi0,scsil
Test started at Monday 1995-05-08 - 10:08:54
Read SCSI device configuration on SCSI device: scsi0 scsil

CH ID Type      Model                      Rev.  Blocks  Number of blocks
0 0 disk        SEAGATE ST32550N          0013  1024    0x00216481 (2137MB)
      seag32550 Sn: 00135795          ANSI=2 <synch>
0 1 No device
0 2 No device
0 3 No device
0 4 No device
0 5 No device
0 6 No device
1 0 No device
1 1 No device
1 2 disk        INSITE I325VM             0387  512     0x00000b40 (1440KB)
      insite325 Sn: unknown           ANSI=1
1 3 No device
1 4 No device
1 5 No device
1 7 tape        TANDBERG TDC 3800        -04:  512     0x00106800 (525MB)
      tb3800 Sn: unknown             ANSI=0
baio 70 ->

```

Figure 11-sdconf-1: Example of using sdconf to read configuration on all SCSI devices

sdconf (SCSI device)

baio30x command

sdconf (SCSI device)

```

baio 70 -> sdconf
CH ID Type      Model                      Rev.  Blocks  Number of blocks
0  0  disk      SEAGATE ST32550N          0013  1024    0x00216481 (2137MB)
      seag32550 Sn: 00135795      ANSI=2 <synch>
1  2  disk      INSITE I325VM            0387  512     0x00000b40 (1440KB)
      insite325 Sn: unknown      ANSI=1
1  7  tape      TANDBERG TDC 3800        -04:  512     0x00106800 (525MB)
      tb3800   Sn: unknown      ANSI=0
baio 70 ->

```

Figure 11-sdconf-2: Example of using the sdconf to see which devices that are selected

## NOTICES

The command cannot be executed while running in *fatal error mode*.

The command does not clear *test log* and *error log*.

The *baio30x* module reserves SCSI channel 0 id 7 and SCSI channel 1 id 6 for its own SCSI controller's id.

## BUGS

It is recommended to read the configuration again from an SCSI device, if you changed the removable media from the SCSI device. If this is not done, the next command to the SCSI device will fail with *Unit attention sense key error*.

## REFERENCES

See also commands:

*set, sdreset, bfsls, diskrd, taperd*

**NAME**

*sddisp* – Display SCSI device data

**SYNOPSIS**

*sddisp* [-dSCSI\_DEVICE] [-bBLK\_ADD] [-rOFFSET\_RANGE] [-nNEXT\_ADD] [-pPREV\_ADD] [-wWRITE] [-vVALUE]

[-dSCSI\_DEVICE]            [-d[00 | 01 | 02 | 03 | 04 | 05 | 06 | 10 | 11 | 12 | 13 | 14 | 15 | 17 | scsi0 | scsi1]]

[-bBLK\_ADD]                [-bDIGIT]

[-oOFFSET\_RANGE]         [-o[START | #SIZE | START-END | START#SIZE]]

[-nNEXT\_ADD]              [-n]

[-pPREV\_ADD]              [-p]

[-wWRITE]                  [-w]

[-vVALUE]                  [-vDIGIT]

**DEFAULTS**

*sddisp* -d00 -b0 -o0-256

**DESCRIPTION**

The *sddisp* command is used to display data contents from an *SCSI device* on the *console terminal*. It can also be used to write one 32 bit value on an *SCSI device*.

The address which is used for reading or writing data on the *SCSI device*, will be calculated as follows: The block address value from the **BLK\_ADD** option is multiplied with the device block size, and the result is added to the start offset address from the **OFFSET\_RANGE** option. The number of bytes which are displayed depends on the size of the value from the **OFFSET\_RANGE** option.

The program only reads one block at a time, and if the *SCSI device* is a tape drive, the program will use rewind and space *SCSI commands* to find the position on the tape.

The *sddisp* command has the following options:

**-dSCSI\_DEVICE**

This option is used to select which *SCSI device* to read or write on. The option must be specified as a *word symbol option*.

**-bBLK\_ADD**

This option is used to specify a block address on the *SCSI device*. It must be specified as a *digit value option*. If the *NEXT\_ADD* or *PREV\_ADD* option is selected, this option has no effect.

**-oOFFSET\_RANGE**

This option is used to specify an address offset range on the *SCSI device*. It must be specified as a *value range option*. If the *NEXT\_ADD* or *PREV\_ADD* option is selected, this option has no effect.

**-nNEXT\_ADD**

If this option is selected the next 256 byte data will be displayed from the end position of the latest *sddisp* command. If the *WRITE* option is selected, this option has no effect.

**-pPREV\_ADD**

If this option is selected the previous 256 byte data will be displayed to the start position of the latest *sddisp* command. If the *WRITE* option is selected, this option has no effect.

**-wWRITE**

If this option is selected the command will write one 32 bit value on the *SCSI device* address. The value to be written must be specified by means of the *VALUE* option.

**-vVALUE**

This option is used to specify a 32 bit value to be written on the *SCSI device*. The value must be specified as a *digit value option*. The option is only used if the **WRITE** option is selected.

Without options the command displays the first 256 bytes data on the *SCSI device* channel 0 id 0.

**EXAMPLES**

```

baio 70 -> sddisp -b24 -o0x180#16 -d12 -w -v0x12345678
Last warning! The disk (insite325) on scsi 1 id 2
          Data block 0x18 will now be changed (y=OK n=CANCEL)? y
Test started at Tuesday 1995-05-09 - 14:17:22
Display SCSI device data on SCSI device: 12
Writing data 0x12345678 on block 0x00000018 offset 0x0180
baio 70 ->
    
```

Figure 11-sddisp-1: Example of using the sddisp to write a 32 bit value

```

baio 70 -> sddisp -d12 -b0 -o1024#16
Test started at Tuesday 1995-05-09 - 14:14:09
Display SCSI device data on SCSI device: 12
Reading block 0x00000002 offset 0x00000000 - 0x00000010
Address      0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15
0x00000400  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
baio 70 ->
    
```

Figure 11-sddisp-2: Example of using the sddisp to display a specific device address

```

baio 70 -> sddisp -n -d12
Test started at Tuesday 1995-05-09 - 14:15:00
Display SCSI device data on SCSI device: 12
Reading block 0x00000002 offset 0x00000010 - 0x00000110
Address      0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
0x00000410  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0x00000420  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0x00000430  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0x00000440  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0x00000450  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0x00000460  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0x00000470  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0x00000480  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0x00000490  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0x000004a0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0x000004b0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0x000004c0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0x000004d0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0x000004e0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0x000004f0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0x00000500  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
baio 70 ->

```

Figure 11-sddisp-3: Example of using the sddisp to display the next device address

**NOTICES**

The command cannot be executed while running in *fatal error mode*.

Before running the command the *SCSI device* in question must be selected by the *sdconf* command.

It is only possible to write a 32 bit value on an *SCSI device*, if the *SCSI device* is a disk drive.

If selecting the *scsi0* word symbol in the *SCSI\_DEVICE* option, the *SCSI channel/id* will be replaced by *00* and *scsi1* will be replaced by *10*.

The command always displays the data contents, no matter how the *dispmode parameter* is set.



## WARNINGS

If you are using the command to write a 32 bit value on the *SCSI device*, a *confirm message* is displayed before writing on the *SCSI device*.

## COMMAND ERRORS

If the command returns the error code *No device available*, it is because the *sdconf* command has not read the configuration on the selected *SCSI device*.

## BUGS

If the *SCSI device* is a tape drive, it is not possible to read data after a *file mark* on the tape.

## REFERENCES

See also commands:  
*set, sdconf*

**sddisp (SCSI device)**

**baio30x command**

**sddisp (SCSI device)**

*This page is intentionally left blank*

**NAME**

*sdloadf* – Download firmware to SCSI device

**SYNOPSIS**

*sdloadf* -mrMEM\_RANGE [-mdMEM\_DEVICE] [-dSCSI\_DEVICES]

-mrMEM\_RANGE            -mr[START | #SIZE | START-END | START#SIZE]

[-mdMEM\_DEVICE]        [-md[pmem | global]]

[-dSCSI\_DEVICE]        [-d[00 | 01 | 02 | 03 | 04 | 05 | 06 | 10 | 11 | 12 | 13 | 14 | 15 | 17 | scsi0 | scsi1]]

**DEFAULTS**

*sdloadf* -d00 -mdpmem

**DESCRIPTION**

The *sdloadf* command is used to down-load SCSI firmware from the memory to the SCSI device.

Before running the command the SCSI device in question must be selected by the *sdconf* command, and the firmware must be loaded into program or global memory.

After the *sdloadf* command has been executed, it is necessary to restart the selected SCSI device by means of the *sdconf* command.

The command has the following options:

**-mrMEM\_RANGE**

The option is used to specify the memory address range where the *sdloadf* command has to read the firmware. The start and the end address must be specified precisely to the address range where the firmware is placed in the memory. The option must be selected as the first option to the command, and it must be specified as a *value range option*.

**-mdMEM\_DEVICE**

This option is used to select from which memory the *sdloadf* has to read the firmware. The option must be specified as a *word symbol option*. The following memories can be specified:

sdloadf (SCSI device)

baio30x command

sdloadf (SCSI device)

**pmem** Program memory.

**global** Global memory.

### -dSCSI\_DEVICE

This option is used to select which *SCSI device* the firmware must be down-loaded to. The option must be specified as a *word symbol option*.

The **MEM\_RANGE** option must always be specified to the command.

### EXAMPLES

The example on the next page, shows how to down-load firmware to an *SCSI device*. First the firmware must be loaded into the *global memory* from the boot device. This is done by means of the *bfsload* command. It is then possible to down-load the firmware from the *global memory* to an *SCSI device* using the *sdloadf* command.

```

baio 70 -> bfsload c2lp0013.lod 261632 -mr0x200000#261632 -mdglobal
Test started at Monday 1995-03-20 - 11:07:06
Load data from Boot File System on SCSI device: 12
File name      Length  Offset  Memory range
c2lp0013.lod   261632  0       0x00200000-0x0023fe00 <global>

baio 70 -> sdloadf -d00 -mr0x200000#261632 -mdglobal
Last warning! The disk (seag32550) on scsi 0 id 0
              Data will now be down-loaded (y=OK n=CANCEL)? y
Test started at Monday 1995-03-20 - 11:08:03
Download firmware to SCSI device on SCSI device: 00
Transfer length: 261632 Memory range: 0x00200000-0x0023fe00 <global>
baio 70 -> sdconf -d00
CH ID Type      Model                Rev.  Blocks  Number of blocks
0 0 disk        SEAGATE ST32550N     0013  1024    0x00216481 (2137MB)
          seag32550 Sn: 00135795          ANSI=2 <synch>
baio 70 ->

```

Figure 11-sdloadf-1: Example of loading firmware to an SCSI device

### NOTICES

The command cannot be executed while running in *fatal error mode*.

If the SCSI firmware is loaded to the *global memory*, do not use the *global memory* for test from other units.

If selecting the `scsi0` word symbol in the `SCSI_DEVICE` option, the SCSI channel/id will be replaced by `00` and `scsi1` will be replaced by `10`.

### WARNINGS

Before down-loading the firmware to the *SCSI device*, the program displays a *confirm message*.

### COMMAND ERRORS

If the command returns the error code *Options error*, it is because the memory range is not aligned on a 4 bytes address.

### REFERENCES

See also commands:  
*set, sdconf, bfsls, bfsload*

**sdloadf (SCSI device)**

**baio30x command**

**sdloadf (SCSI device)**

*This page is intentionally left blank*

**NAME**

*sdreset* – Reset SCSI interface

**SYNOPSIS**

*sdreset* [-dSCSI\_DEVICES]

[-dSCSI\_DEVICES] [-d[00 | 01 | 02 | 03 | 04 | 05 | 06 | 10 | 11  
| 12 | 13 | 14 | 15 | 17 | scsi0 | scsi1][,\*,,]]

**DESCRIPTION**

The *sdreset* command performs a reset on the selected *SCSI* interface. It only has one option as shown below:

**-dSCSI\_DEVICES**

This option is used to specify which *SCSI* interface should be reset. It must be specified as a *multi word symbol option*.

Without the option the command has no effect.

**EXAMPLES**

```
baio 70 -> sdreset -dscsi0,scsi1
Test started at Monday 1995-03-20 - 12:04:09
Reset SCSI interface on SCSI device: scsi0 scsi1
baio 70 ->
```

Figure 11-sdreset-1: Example of resetting both SCSI interfaces

**NOTICES**

The command cannot be executed while running in *fatal error mode*.

If a specific *SCSI device* is selected, the *SCSI* interface on that *SCSI device* is reset.

It is not necessary to execute the *sdreset* command after the *SCSI* interface has failed during execution of an *SCSI command*, since the interface is automatically reset before the next *SCSI command* is executed.

**sdreset (SCSI device)**

**baio30x command**

**sdreset (SCSI device)**

## **REFERENCES**

See also commands:

*sdconf*



**NAME**

*set* - Set or display parameters

**SYNOPSIS**

*set* [PARAMETER]

<b>[PARAMETER]</b>	[PARAMETER[=VALUE]]
<b>fault</b>	[off   poll   on]
<b>cache</b>	[off   code   data   code/data   test   code/test   data/test   on]
<b>dispmode</b>	[error   status   test   errstop   testshort   all   off]
<b>repmode</b>	[on   pass   time   cmd   test   error   off]
<b>repno</b>	[DIGIT]
<b>term</b>	[non   dumterm   vt100]
<b>scsimode</b>	[non   disc   sync   wide][,*,*]
<b>diskdev</b>	[00   01   02   03   04   05   06   10   11   12   13   14   15   17][,*,*]
<b>tapedev</b>	[00   01   02   03   04   05   06   10   11   12   13   14   15   17][,*,*]
<b>bfsdev</b>	[00   01   02   03   04   05   06   10   11   12   13   14   15   17   scsi0   scsi1]

**DESCRIPTION**

The *set* command is used to show or change certain parameters which are used to set the units in different modes or to set default values for certain commands. It is possible to assign different values to the different units. But only one parameter can be changed at a time. If no options are given, the command displays the current values for all parameters.

The following is a description of all parameters:

### fault

The *fault parameter* is used to disable/enable error checking from the *error status* and on the *global memory* modules.

*Error status* checking includes errors from time-out (bus error exception) and parity circuitry on the *baio30x* board, and errors from the *CPU* and *SCSI Agent* (*bad data identifier, missing target acknowledge and parity errors*). These errors can be checked by reading a hardware error status register. If an error occurs an interrupt to the *CPU* can be generated.

*Global memory* modules checking includes errors from parity and ECC (Error Check Code) circuitry on the memory modules. These errors can only be checked by reading a hardware status register on the memory modules.

The *fault parameter* must be specified as a *word symbol option* and the following fault modes are shown below:

- off** In this mode no fault check will be done on *status interrupt* and *global memory* modules.
- poll** In this mode fault will be checked on *status interrupt* by reading the register (no interrupt will be generated to the *CPU*) and fault will also be checked on *global memory* modules.
- on** In this mode fault will be checked on *status interrupt* by generating an interrupt to the *CPU*, and fault will also be checked on *global memory* modules.

**NOTE**

If an error occurs from the *error interrupt*, the program goes straight to the *fatal error mode*. An exception error from the *CPU* or wrong or illegal interrupts to the *CPU* also invokes the *fatal error mode*.

### cache

The *cache parameter* is used to disable or enable the *CPU cache* in different cache modes specified as a *word symbol option*. The cache modes are shown below:

- |                  |   |
|------------------|---|
| <b>off</b>       | In this mode the <i>CPU cache</i> is not used.  |
| <b>code</b>      | In this mode the <i>CPU cache</i> is only used when it executes an instruction fetch.   |
| <b>data</b>      | In this mode the <i>CPU cache</i> is only used when it executes a write/read access used by the Diagnostic program (e.g stack).                 |
| <b>code/data</b> | In this mode the <i>CPU cache</i> is used when it executes an instruction fetch or executes a write/read access used by the Diagnostic program. |
| <b>test</b>      | In this mode the <i>CPU cache</i> is only used when it executes a write/read access to a memory from a test.                                    |
| <b>code/test</b> | In this mode the <i>CPU cache</i> is used when it executes an instruction fetch or a write/read access to a memory from a test.                 |
| <b>data/test</b> | In this mode the <i>CPU cache</i> is used when it executes a write/read access.   |
| <b>on</b>        | In this mode the <i>CPU cache</i> is used both for instruction fetch and write/read access.   |

The *CPU cache* mode which controls the write/read access to a memory from a test can be overruled by certain commands.

### dispmode

The *dispmode parameter* is used to set the type of information to display when the unit is testing. Please refer to "Chapter 2-3 Display modes" to see the detailed description of the different display modes.

The following display modes can be set, and the mode must be specified as a *word symbol option*:

- error**            When a test starts the *unit status display mode* is displayed. If an error occurs the *error display mode* will be displayed instead.
- status**            Only the *unit status display mode* is displayed in this mode.
- test**              When a test starts the *test display mode* is displayed. If an error occurs the *unit status display mode* will be displayed instead.
- errstop**            When a test starts the *test display mode* is displayed. If an error occurs the *error display mode* will be displayed and afterwards the program will wait for a continue command from the terminal.
- testshort**        When a test starts the *short test display mode* is displayed. If an error occurs the *error display mode* will be displayed.
- all**                When a test starts the *test display mode* is displayed. If an error occurs the *error display mode* will be displayed.
- off**                In this mode nothing will be displayed when running tests.

#### repmode

The *repmode parameter* is used to set which repeat mode to use when not using the *repeat* command. The following repeat modes can be set, and they must be specified as a *word symbol option*:

- off**                This mode does not repeat commands.
- on**                 This mode repeats commands forever.
- pass**              This mode repeats commands until the number of *command line* buffer passes has reached the value of the *repmo parameter*.

- time** This mode repeats commands until the running time measured in seconds has passed the value of the *repro* parameter.
- cmd** This mode repeats commands until the number of commands has passed the value of the *repro* parameter.
- test** This mode repeats commands until the number of tests in the *test log* buffer has passed the value of the *repro* parameter.
- error** This mode repeats commands until the number of errors in the *error log* buffer has passed the value of the *repro* parameter.

The *repro* parameter can be overruled by the *repeat* command.

#### **repro**

The *repro* parameter is used to specify how many times to repeat the commands. It must be specified as a *digit value option*. If the *repro* parameter option is set to **off** or **on**, the *repro* parameter has no effect. The parameter can be overruled by the *repeat* command.

#### **term**

The *term* parameter is used to set the type of communication between the *master unit* and the *console terminal*. The following terminal types can be set, and they must be specified as a *word symbol option*:

- non** This type is only used when the unit is not a *master unit* and it cannot be changed.
- vt100** This type can be used when the terminal emulating is set up to *VT102*, *VT220* or *VT320* family, either 7 or 8 bit are supported.
- dumterm** This type can only be used when the terminal is a non-*vt100* type. When this parameter is used, the terminal output will be slower. It is usually used when the terminal is an older type.

## NOTE

The *term* parameter can only be set on the *master* unit.

**scsimode**

The *scsimode* parameter is used to disable/enable different modes on the *SCSI* interfaces. Every time the *scsimode* parameter is changed, all *SCSI* devices will be initialized with the new change, when they execute the next *SCSI* command to the *SCSI* device. The following modes can be set, and they must be specified as a *multi word* symbol option:

- non** This mode will disable all *SCSI* modes.
- disc** This mode enables the *disconnect/reconnect* option for all *SCSI* devices. It is a feature in the *SCSI* interface which enables the target to free the *SCSI* bus during execution of a command.

## NOTE

This version of the program is not able to handle pending commands to each device on the *SCSI* bus.

- sync** This mode enables *synchronous data transfer* for all *SCSI* devices which implement this feature.
- wide** This mode enables *16 bit wide data transfer* for all *SCSI* devices which implement this feature.

### diskdev

The *diskdev* parameter is used set default SCSI channel/id numbers on disk drives which are used in disk group commands. It must be specified as a *multi word symbol option*.

**NOTE**

The *sdconf* command must have read the configuration from a disk drive, before the disk drive is available in the *word symbol* list. It cannot be added to the list during set.

### tapedev

The *tapedev* parameter is used set default SCSI channel/id numbers on tape drives which are used in tape group commands. It must be specified as a *multi word symbol option*.

**NOTE**

The *sdconf* command must have read the configuration from a tape drive, before the tape drive is available in the *word symbol* list. It cannot be added to the list during set.

### bfsdev

The *bfsdev* parameter is used to set the SCSI channel/id number on a SCSI device that content a *boot file system*.

The parameter will always be initialized to the SCSI device where the Diagnostic programs has been booted from, but if the unit is not a *master unit*, the parameter is initialized to no SCSI device.

**NOTE**

If selecting the *scsi0* word symbol, the SCSI device will be replaced by 00 and *scsi1* will be replaced by 10.

## EXAMPLES

```
baio 70 -> set
fault-on
cache-on
dispmode=error
repmode=off
repmo=0
term=vt100
scsimode=disc, sync
diskdev=00
tapedev=07
bfsdev=12
baio 70 ->
```

Figure 11-set-1: Example of using set command to display current parameter values

```
baio 70 -> set dispmode=all
baio 70 ->
```

Figure 11-set-2: Example of using set command to change a parameter

## NOTICES

The command does not clear *test log* and *error log*.

The command cannot be repeated. It will only run once.

## REFERENCES

See also commands:

*repeat*, *memfast*, *bfsls*, *sdconf*, *diskrd*, *taperd*



**NAME**

*status* – Display test status

**SYNOPSIS**

*status*

**DESCRIPTION**

The *status* command is used to display the status after executing one or more *test commands*.

The *status* command shows the following status information:

- When the test is started and for how long time it has been executed.
- Number of commands.
- Number of tests.
- Number of errors.
- Number of passes.

The status information is cleared every time the *command line* buffer contains a *test command* and the *command interpreter* starts executing the first command from the *command line* buffer. When the commands are complete or cancelled, the status information will be saved.

## EXAMPLES

```
baio 70 -> memfast -ddmem -r*-* ; ethtest
baio 70 -> status
Status for test started at..: Monday 1995-03-20 - 18:42:31
Executed test time.....: 00:00:04
Number of commands.....: 2
Number of tests.....: 2
Number of errors.....: 0
Number of passes.....: 1
baio 70 ->
```

Figure 11-status-1: Example of the status command

## NOTICES

The command does not clear *test log* and *error log*.

The command cannot be repeated. It will only run once.

## REFERENCES

See also commands:

*testlog*, *errlog*

**NAME**

*sysbus* – System bus exerciser

**SYNOPSIS**

*sysbus* [-caCA\_CMD] [-csCS\_CMD] [-pPOSITION] [-sSUBPOS] [-oOFFSET\_ADD] [-vVALUE]

[-caCA\_CMD] [-ca[stat | cntl | debug | cntwr | cntrd | write | read | reset]]

[-csCS\_CMD] [-cs[cntl | stat | statclr | id | fcn | debug | intr | buserr | berrclr | cpuconf | mem-lowbase | memupbase | memsize | mem-synb | count | write | read]]

[-pPOSITION] [-pDIGIT]

[-sSUBPOS] [-sDIGIT]

[-oOFFSET\_ADD] [-oDIGIT]

[-vVALUE] [-vDIGIT]

**DEFAULTS**

*sysbus* -castat -pOWN -sOWN -o0 -v0

**DESCRIPTION**

The *sysbus* command is used to exercise the *CPU agent* or *control space* on the *global bus*. This is very useful to diagnose fault on a module which fails with errors on the *global bus*, when executing the *boot* command.

The command performs a write or read access to a register on the *global control space* when the *CS\_CMD* option is specified, otherwise the *CPU agent* on the module is accessed.

If the *global control space* is accessed, it is possible to select which module on the *global bus* you want to select. This is specified by means of the *POSITION* and *SUBPOS* options.

The **CA\_CMD** or **CS\_CMD** option specifies the register to be accessed. If the register is a read register, the register information will be displayed on the *console terminal*. If it is a write register the value to be written must be specified by the **VALUE** option.

The command will always display information about its currently selected options.

The *sysbus* command has the following options:

**-caCA\_CMD**

This option is used to select which register on the *CPU agent* to be displayed or changed. The option will be overruled by the **CS\_CMD** option. The option must be specified as a *word symbol option*, and the following registers can be specified:

- |              |  |
|--------------|--|
| <b>stat</b>  | Read the status from the <i>CPU agent status register</i> .  |
| <b>cntl</b>  | Write a value to the <i>CPU agent control register</i> .   |
| <b>debug</b> | Read the value from the <i>CPU agent debug register</i> .  |
| <b>cntwr</b> | Write a value to the <i>CPU agent count register</i> .   |
| <b>cntrd</b> | Read the value from the <i>CPU agent count register</i> .  |
| <b>write</b> | Write a 32 bit value to a <i>CPU agent register</i> . The <b>OFFSET_ADD</b> option specifies the offset address for the register.  |
| <b>read</b>  | Read a 32 bit value from a <i>CPU agent register</i> . The <b>OFFSET_ADD</b> option specifies the offset address for the register. |
| <b>reset</b> | Reset the <i>CPU agent</i> and the <i>SCSI agent</i> .   |

**-csCS\_CMD**

This option is used to select which register on the *global control space* to be displayed or changed. The option must be specified as a *word symbol option*, and the following registers can be specified:

- cntl** Write a value to a *control space control register*.
- stat** Read the status from a *control space status register*.
- statclr** Clear information on a *control space status register*.
- id** Read the values from a *control space module id*.
- fcn** Read the values from a *control space module fcn*.
- debug** Write a value to a *control space debug register*.
- intr** Write an interrupt to a *control space interrupt register*.
- buserr** Read the status from a *control space bus error register*.
- berrclr** Clear information on a *control space bus error register*.
- cpuconf** Read the configuration from a *control space processor module register*.
- memlowbase** Write start address (Mbytes) for the low memory partition to a *control space lower base address register*.
- memupbase** Write start address (Mbytes) for the high memory partition to a *control space lower base address register*.
- memsize** Read the memory size from a *control space size register*.
- memsynb** Read the ECC status from a *control space syndrome bits register*.

sysbus (system)

baio30x command

sysbus (system)

count	Read the value from a <i>control space 1 MHz count register</i> .
write	Write a 32 bit value to a <i>control space register</i> . The <b>OFFSET_ADD</b> option specifies the offset address to the register.
read	Read a 32 bit value from a <i>control space register</i> . The <b>OFFSET_ADD</b> option specifies the offset address to the register.

**-pPOSITION**

This option is used to specify the module position when accessing the *control space* on the *global bus*. The option only has effect if the **CS\_CMD** option is specified. The **POSITION** option must be specified as a *digit value option*.

**-sSUBPOS**

This option is used to specify the module subposition when accessing the *control space* on the *global bus*. The option only has effect if the **CS\_CMD** option is specified. The **SUBPOS** option must be specified as a *digit value option*.

**-oOFFSET\_ADD**

This option is used to specify an offset address to a register on the *CPU agent* or the *global control space*. This is necessary if you want to read from or write to a register which is not defined by the **CA\_CMD** or **CS\_CMD** option. The option must be specified as a *digit value option*, and the offset address will always be aligned to 4 bytes by cutting the value. The option only has effect if the **CA\_CMD** or **CS\_CMD** option is set to **write** or **read**.

**-vVALUE**

This option is used to specify a 32 bit value to be written in the selected register. The value must be specified as a *digit value option*, and it is only used if the selected register is a write register.

Without options the command displays the *CPU agent status register*.

## EXAMPLES

```
baio 70 -> sysbus -csid -p0 -s0
Read from id register on global Control Space position 0 subpos 0
Module name: CPU300-1.....
Module type: 0x00000003
Serial no: 0x0000001f
baio 70 ->
```

Figure 11-sysbus-1: Example of using sysbus to read global control space id register

## NOTICES

The command does not clear *test log* and *error log*.

The command always displays the bus information, no matter how the *dispmode parameter* is set.

The command always checks the *Error status* except for the parity circuitry on the *baio30x* board, no matter how the *fault parameter* is set.

## WARNINGS

If using the command to write to a register on the *CPU agent* or *global control space*, you must be sure that the correct value is written. Otherwise, the Diagnostic Programs can indicate incorrect errors on a unit or even break down.

## COMMAND ERRORS

The command will fail with *missing target acknowledge* error, if trying to access a module on a position which is not installed in the Supermax Enterprise Server.

## BUGS

The command will not save the exerciser information in the *test log* buffer. The information will only be displayed on the *console terminal*, no matter how the *dispmode parameter* is set.

**sysbus (system)**

**baio30x command**

**sysbus (system)**

## REFERENCES

See also commands:

*boot, config, reset.*



**NAME**

*tape* – Test script for SCSI tape drives

**SYNOPSIS**

```
tape [-mTEST_MODES] [-tTEST_FUNCTIONS] [-pDISP_CMDS]
      [-m[read | write] [*,*,]]
      [-t[fast | function] [*,*,]]
      [-pDISP_CMDS]      [-p]
```

**DEFAULTS**

*tape* -mread,write -tfast,function

**DESCRIPTION**

This command is a *script command*, and it used to run a complete test of tape drive(s). The command executes other commands which are all tests of tape drives, either read or write tests. It is possible to specify which commands of the *tape script* to be executed. This is done by means of the **TEST\_MODES** and **TEST\_FUNCTIONS** options.

The tape drives used in the tests will be all the tape drives from the *tapedev parameter*. Please refer to the *set* command to see the detailed description of the *tapedev parameter*.

The *tape* command has the following options:

**-mTEST\_MODES**

This option is used to specify which test modes to be selected from the *tape script*. The option must be specified as a *multi word symbol option*, and the following test modes can be specified:

- |              |  |
|--------------|--|
| <b>read</b>  | This test mode will select commands from the script which only read from the tape drives. Data on the tape will not be destroyed.          |
| <b>write</b> | This test mode will select commands from the script which both write and read to/from the tape drives. Data on the tape will be destroyed. |

### -tTEST\_FUNCTIONS

This option is used to specify which test functions to be selected from the *tape script*. The option must be specified as a *multi word symbol option*, and the following test functions can be specified:

**fast** If this test function is selected, the tape drives will only be tested fast and not thoroughly. The selected commands will only use the *program memory* when running the tests.

**function** If this test function is selected, the tape drives will be tested thoroughly, the selected commands will use the *global memory* when running the tests.

### -pDISP\_CMDS

If this option is selected, the commands which are selected from the script will only be displayed, not executed.

Without option the command executes all the commands from the scripts.

## EXAMPLES

```
baio 70 -> tape -p
Test script for SCSI tape drives
Device  Function  Command script
read   fast      taperd -s64
write  fast      tapewr -s8-128 -b#0x10000 -tcmp
read   function  taperd -s64 -mdglobal
write  function  tapewr -s64 -tcmp -mdglobal
baio 70 ->
```

Figure 11-tape-1: Example of the tape command displaying all commands from the script

```
baio 70 -> tape -tfunction -p
Device Function Command script
read function taperd -s64 -mdglobal
write function tapewr -s64 -tcmp -mdglobal
baio 70 ->
```

Figure 11-tape-2: Example of the tape command selecting a few commands from the script

## NOTICES

The command cannot be used while running in *fatal error mode*.

## WARNINGS

If you are using the command to perform write test on a tape drive, a *confirm message* is displayed before destroyed data on the tape.

## REFERENCES

See also commands:  
*set, sdconf, taperd, tapewr.*

*This page is intentionally left blank*

## NAME

*tapeclr* – Erase tape

## SYNOPSIS

*tapeclr* [-dTape\_DEVICES]

[-dTape\_DEVICES] [-d[00 | 01 | 02 | 03 | 04 | 05 | 06 | 10 | 11  
| 12 | 13 | 14 | 15 | 17][,\*,\*]]

## DEFAULTS

*tapeclr* -dTapeDEV\_PARAMETER

## DESCRIPTION

This command causes the tape drive(s) to erase the entire tape. After the erasing the tape will be rewound to its beginning. The *tapeclr* command only has one option as shown below:

### -dTape\_DEVICES

Same description as the *tapewr* command.

If no options are given, the command will erase the entire tape on all tape drives selected in the *tapedev parameter*. Please refer to the *set* command to see the detailed description of the *tapedev parameter*.

## EXAMPLES

```
baio 70 -> tapeclr
Last warning! The tape (tb3800) on scsi 1 id 7
      Tape will now erased (y=OK n=CANCEL)? y
Test started at Friday 1995-05-12 - 11:56:04
Erase tape on SCSI device: 17
baio 70 ->
```

Figure 11-tapeclr-1: Example of the *tapeclr* command

## NOTICES

The command cannot be executed while running in *fatal error mode*.

**tapeclr (tape)****baio30x command****tapeclr (tape)****WARNINGS**

Before erasing a tape, the program displays a *confirm message* and if this is confirmed with a  **y** , the data on the tape will be destroyed.

**REFERENCES**

See also commands:

*set, sdconf, sddisp, taperd, tapewr, taperew, taperet*

**NAME**

*taperd* – Tape read test

**SYNOPSIS**

*taperd* [-dTAPE\_DEVICES][-bBLK\_RANGE] [-sBLK\_PER\_CMD] [-mdMEM\_DEVICE] [-mrMEM\_RANGE]

[-dTAPE\_DEVICES] [-d[00 | 01 | 02 | 03 | 04 | 05 | 06 | 10 | 11 | 12 | 13 | 14 | 15 | 17][,\*,\*]]

[-bBLK\_RANGE] [-b[START | #SIZE | START-END | START#SIZE]]

[-sBLK\_PER\_CMD] [-s[START | #SIZE | START-END | START#SIZE]]

[-mdMEM\_DEVICE] [-md[pmem | global]]

[-mrMEM\_RANGE] [-mr[START | #SIZE | START-END | START#SIZE]]

**DEFAULTS**

*taperd* -dTapeDev\_PARAMETER -b\*-\* -s32 -mddmem

**DESCRIPTION**

This command tests tape drive(s). This is done by reading data from the tape drive to the memory. The test does not check the data on the tape.

The command will first rewind the tape and then read data from the tape. If the **BLK\_RANGE** option is specified to a non-zero start block address, the program will perform an *SCSI space command* to find the specified position on the tape, before the command will start reading the data. The *taperd* command has the following options:

**-dTape\_DEVICES**

Same description as the *tapewr* command.

**-bBLK\_RANGE**

Same description as the *tapewr* command.

**-sBLK\_PER\_CMD**

Same description as the *tapewr* command.

**-mdMEM\_DEVICE**

Same description as the *tapewr* command.

**-mrMEM\_RANGE**

Same description as the *tapewr* command, but the calculation of the necessary number of bytes is as follows:

(*tape\_numbers* \* *block\_per\_cmd* \* *block\_size*) aligned to 0x10000.

If no options are given, the *taperd* command will run a read test on all tape drives selected in the *tapedev* parameter from the beginning of the tape to a *file mark* or until there are no more data on the tape. Please refer to the *set* command to see the detailed description of the *tapedev* parameter.

**EXAMPLES**

```

baio 70 -> taperd -b100-200
Test started at Thursday 1995-06-01 - 13:42:35
Tape read test on SCSI device: 17
Block address range  Blocks  Memory range      Device
0x00000064-0x000000c8 32    0x000d0000-0x000e0000 pmem
baio 70 ->

```

Figure 11-taperd-1: Example of the taperd command

**NOTICES**

The command cannot be executed while running in *fatal error mode*.

The maximum number of blocks per *SCSI command* is up to 16 Mbyte or the size of available memory.

If the number of blocks to be tested is not divisible by the blocks per *SCSI read command*, the number of blocks per *SCSI command* will be reduced for the last access to tape drive.



**taperd (tape)****baio30x command****taperd (tape)****BUGS**

It is not possible to read data after a *file mark* on the tape.

**COMMAND ERRORS**

If the command returns the error code *Options error*, it is because the memory range is not aligned on 4 bytes addresses or because the specified memory range is too small.

**REFERENCES**

See also commands:

*set, sdconf, sddisp, tapewr, taperew, taperet, tapeclr*

*This page is intentionally left blank*

**NAME**

*taperet* – Retain tape

**SYNOPSIS**

*taperet* [-dTAPPE\_DEVICES]

[-dTAPPE\_DEVICES] [-d[00 | 01 | 02 | 03 | 04 | 05 | 06 | 10 | 11  
| 12 | 13 | 14 | 15 | 17][,\*,,]]

**DEFAULTS**

*taperet* -dTAPPEDEV\_PARAMETER

**DESCRIPTION**

This command causes the tape drive(s) to retain the tape. This is done by rewinding the tape to the beginning, winding it to the end position and then rewinding it to the beginning again. The *taperet* command only has one option as shown below:

**-dTAPPE\_DEVICES**

Same description as the *tapewr* command.

If no options are given, the command will retain the tape on all tape drives selected in the *tapedev* parameter. Please refer to the *set* command to see the detailed description of the *tapedev* parameter.

**EXAMPLES**

```
baio 70 -> taperet
Test started at Friday 1995-05-12 - 11:58:15
Retain tape on SCSI device: 17
baio 70 ->
```

Figure 11-taperet-1: Example of the *taperet* command

**NOTICES**

The command cannot be executed while running in *fatal error mode*.

**REFERENCES**

See also commands:

*set, sdconf, sddisp, taperd, tapewr, taperew, tapeclr*

**NAME**

*taperew* – Rewind tape

**SYNOPSIS**

*taperew* [-dTAPE\_DEVICES]

[-dTAPE\_DEVICES] [-d[00 | 01 | 02 | 03 | 04 | 05 | 06 | 10 | 11  
| 12 | 13 | 14 | 15 | 17][,\*,\*]]

**DEFAULTS**

*taperew* -dTAPEDEV\_PARAMETER

**DESCRIPTION**

This command causes the tape drive(s) to rewind the tape to the beginning. The *taperew* command only has one option as shown below:

**-dTAPE\_DEVICES**

Same description as the *tapewr* command.

If no options are given, the command will rewind the tape to the beginning on all tape drives selected in the *tapedev parameter*. Please refer to the *set* command to see the detailed description of the *tapedev parameter*.

**EXAMPLES**

```
baio 70 -> taperew -d17
Test started at Friday 1995-05-12 - 11:55:10
Rewind tape on SCSI device: 17
baio 70 ->
```

Figure 11-taperew-1: Example of the *taperew* command

**NOTICES**

The command cannot be executed while running in *fatal error mode*.

taperew (tape)

baio30x command

taperew (tape)

## REFERENCES

See also commands:

*set, sdconf, sddisp, taperd, tapewr, taperet, tapeclr*

**NAME**

*tapewr* – Tape write/read test

**SYNOPSIS**

*tapewr* [-dTAPE\_DEVICES][-bBLK\_RANGE] [-sBLK\_PER\_CMD] [-wDELAY\_MS] [-tTYPE] [-pPATTERN] [-vPAT\_VALUE] [-mdMEM\_DEVICE] [-mrMEM\_RANGE]

<b>[-dTAPE_DEVICES]</b>	[-d[00   01   02   03   04   05   06   10   11   12   13   14   15   17][*,*,]]
<b>[-bBLK_RANGE]</b>	[-b[START   #SIZE   START-END   START#SIZE]]
<b>[-sBLK_PER_CMD]</b>	[-s[START   #SIZE   START-END   START#SIZE]]
<b>[-wDELAY_MS]</b>	[-wDIGIT]
<b>[-tTYPE]</b>	[-t[cmp   write   read   check]]
<b>[-pPATTERN]</b>	[-p[random   value   count   cntdown   current]]
<b>[-vPAT_VALUE]</b>	[-vDIGIT]
<b>[-mdMEM_DEVICE]</b>	[-md[pmem   global]]
<b>[-mrMEM_RANGE]</b>	[-mr[START   #SIZE   START-END   START#SIZE]]

**DEFAULTS**

*tapewr* -dTAPPEDEV\_PARAMETER -b\*-\* -s32 -w0 -tcmp -prandom -vX -mddmem

**DESCRIPTION**

This command tests tape drive(s). This is done by writing and reading data from a tape drive to the memory while checking the data.

It is possible to select different types of tests using the **TYPE** option. The following tests can be performed:

- Read data from a tape without checking data.
- Read data from a tape while checking data.
- Write data to a tape.
- Write data to a tape, and read and check the written data afterwards.

It is possible to specify different data patterns to be written on the tapes. This is done by means of the **PATTERN** and **PAT\_VALUE** options.

It is also possible to specify a delay between two *SCSI commands* to a tape drive which is used for testing *non-streaming mode* on the tape drives.

The *tapewr* command has the following options:

#### **-dTAPE\_DEVICES**

This option is used to select which tape drive(s) should be tested. The option must be specified as a *multi word symbol option*. The *sdconf* command must have read the configuration from a tape drive, before the tape drive is available in the word symbol list. If the command is executed without this option, it selects the tape drive(s) specified in the *tapedev parameter*. Please refer to the *set* command to see the detailed description of the parameter.

#### **-bBLK\_RANGE**

This option is used to specify a block address range for the write or read access on the tape. The option must be specified as a *value range option*. The size of the range defines how many blocks to test on a tape drive. If no end block address is specified, the command will continue writing data to the tape drive until the end of the tape or the command will continue reading data from the tape drive until a *file mark* is read or until there are no more data on the tape.



**-sBLK\_PER\_CMD**

This option is used to specify the number of blocks per *SCSI write/read command* when performing the test. It is also possible to specify a block transfer range. If this is done the number of blocks per access is calculated on a pseudo random basis for each access to the *SCSI device*. The option must be specified as a *value range option*.

**-wDELAY\_MS**

This option is used to specify a delay between two *SCSI commands* to a tape drive. It must be specified as a *digit value option*. The value is given in milliseconds.

**-tTYPE**

This option is used to specify which test the command shall perform. The option must be specified as a *word symbol option*, and the following types of test can be specified:

- cmp**           The test will first perform write accesses to the specified block address range on the disk drives. Afterwards it will perform read accesses and finally check the data.
- write**         The test will perform write accesses to the specified block address range on the tape drives.
- read**          The test will perform read accesses to the specified block address range on the tape drives without checking the data.
- check**         The test will perform read accesses to the specified block address range on the tape drives and check the data.

**-pPATTERN**

This option is used to specify a data pattern type together with the **PAT\_VALUE** option. The data pattern is used to generate data to the tape or to compare data from the tape. If the **TYPE** option is set to **read**, this option has no effect.

The **PATTERN** option must be specified as a *word symbol option*, and the following pattern types can be specified:

- random** This pattern type will generate a 32 bit random data pattern from the **PAT\_VALUE**. If the same **PAT\_VALUE** is specified, the same random pattern is generated.
- value** This pattern type will generate a 32 bit data value specified with the **PAT\_VALUE** option.
- count** This pattern type will generate a 32 bit count data pattern calculated as follows:  
 $\text{PAT\_VALUE} + (\text{block address} * \text{block size}) + (\text{offset} / 4)$ .
- cntdown** This pattern type will generate a 32 bit count-down data pattern calculated as follows:  
 $\text{PAT\_VALUE} + (\text{block address} * \text{block size}) - (\text{offset} / 4)$ .
- current** This pattern type will use the current values in the memory address range.

#### **-vPAT\_VALUE**

This option is used to specify a 32 bit data value for the pattern type. The option must be specified as a *digit value option*. If not specified, the default value is a value from the internal system clock. If the **PATTERN** option is set to **current** or the **TYPE** option is set to **read**, this option has no effect.

#### **-mdMEM\_DEVICE**

This option is used to select which memory the test has to use. The option must be specified as a *word symbol option*. The following memories can be specified:

- pmem** Program memory.
- global** Global memory.

**-mrMEM\_RANGE**

This option is used to specify the memory address range where the test has to read data from the tape drives. The option must be specified as a *value range option*. Default, the command will reserve the next free memory array with the necessary number of bytes, calculated as follows:

(tape\_numbers \* block\_per\_cmd \* block\_size \* 2) aligned to 0x10000.

If no options are given, the *tapewr* command will run a write/read test on all tape drives selected in the *tapedev* parameter from the beginning to the end of the tape. Please refer to the *set* command to see the detailed description of the *tapedev* parameter.

**EXAMPLES**

```

baio 70 -> tapewr -b#0x1000 -pcount -v0 -mdglobal mr0x1000000
Last warning! The tape (hp1533) on scsi 1 id 7
      Data will now be destroyed (y=OK n=CANCEL)? y
Test started at Friday 1995-06-02 - 11:29:51
Tape write/read test on SCSI device: 17
Block address range  Blocks  Delay-ms Type  Memory range Device Pattern
0x00000000-0x00001000  32      0      cmp  0x01000000-  global count
                                0x01010000  0x00000000

baio 70 -> addisp -b0xffff -o#32 -d17
Test started at Friday 1995-06-02 - 11:30:50
Display SCSI device data on SCSI device: 17
Reading block 0x00000fff offset 0x00000000 - 0x00000020
Address      0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15
0x003fff00  00 0f ff 00 00 0f ff 01 00 0f ff 02 00 0f ff 03 .....
0x003ffc10  00 0f ff 04 00 0f ff 05 00 0f ff 06 00 0f ff 07 .....
baio 70 ->

```

Figure 11-tapewr-1: Example of the tapewr command

## NOTICES

The command cannot be executed while running in *fatal error mode*.

The maximum number of blocks per *SCSI command* is the size of available memory or up to 16 Mbyte.

If the number of blocks to be tested is not divisible by the blocks per *SCSI write/read command*, the number of blocks per *SCSI command* will be reduced for the last access to the tape drive.

If the **TYPE** option is specified to **write** or **cmp**, the start block address in the option **BLK\_RANGE** must be set to zero.

## WARNINGS

If setting the **TYPE** option to **check**, you must be sure that the compared data have been written on the tape, else the program will find incorrect data on the tapes.

If you are using the command to perform write test on the tape drive, a confirm message is displayed before writing on the tape. If this is confirmed with a **(Y)**, the data on the disk will be destroyed.

## BUGS

It is not possible to read data after a *file mark* on the tape.

## COMMAND ERRORS

If the command returns the error code *Options error*, it is because the memory range is not aligned on 4 bytes addresses or because the specified memory range is too small.

## REFERENCES

See also commands:

*set, sdconf, sddisp, taperd, taperew, taperet, tapeclr*

## NAME

*test* – Test script for all tests

## SYNOPSIS

*test* [-tTEST\_FUNCTIONS] [-pDISP\_CMDS]

[-tTEST\_FUNCTIONS] [-t[memory | scsi | ethernet | hdlc | scu] [\*,\*]]

[-pDISP\_CMDS] [-p]

## DEFAULTS

*test* -tmemory,scsi,ethernet,hdlc,scu

## DESCRIPTION

This command is a *script command*, and it is used to run a complete test of the *baio30x* module including the *global memory* and the sub-modules which are connected to the module. The command executes other *script commands* which are all tests of the memories, ethernet, and scsi controls, hdlc and service computer submodules. It is possible to specify which test groups of the *test script* to be executed. This is done by means of the **TEST\_FUNCTIONS** option.

The *test* command has the following options:

### -tTEST\_FUNCTIONS

This option is used to specify which test groups should be selected from the *test script*. The option must be specified as a *multi word symbol option*, and the following test groups can be specified:

- |                 |  |
|-----------------|--|
| <b>memory</b>   | If this is selected, the <i>memory</i> command will be executed.   |
| <b>scsi</b>     | If this is selected, the <i>scsi</i> command will be executed.     |
| <b>ethernet</b> | If this is selected, the <i>ethernet</i> command will be executed. |

test(system)

baio30x command

test(system)

- hdlc            If this is selected, the *hdlc* command will be executed.
- scu            If this is selected, the *scu* command will be executed.

**-pDISP\_CMDS**

If this option is specified, the selected commands from the script will only be displayed, not executed.

Without option the command executes all the commands from the scripts.

**EXAMPLES**

```

baio 70 -> test -p
Test script for all tests
Function Command script
memory memory
scsi scsi
ethernet ethernet
hdlc hdlc
scu scu
baio 70 ->
  
```

Figure 11-test-1: Example of the test command displaying all commands from the script

```

baio 70 -> test -p -memory
Test script for all tests
Function Command script
memory memory
baio 70 ->
  
```

Figure 11-test-2: Example of the test command selecting a few commands from the script

## NOTICES

The command cannot be used while running in *fatal error mode*.

The *SCSI devices* which are connected to the module will not be tested.

## REFERENCES

See also commands:

*ethernet, hdlc, memory, scsi, scu.*

test (system)

baio30x command

test (system)

*This page is intentionally left blank*



## NAME

*testlog* – Display test information from the test log

## SYNOPSIS

*testlog* [-nNUMBER] [-old]

[-nNUMBER] [-nDIGIT]

[-old] [-old]

## DESCRIPTION

The *testlog* command is used to display test information which is stored in the *test log* buffer. The *test log* buffer is cleared every time the *command line* buffer contains a *test command* and the *command interpreter* starts executing the first command from the *command line* buffer.

The maximum number of tests in the *test log* buffer is 64. If this number is exceeded the oldest log will be deleted and the new ones will be added to the buffer.

The *testlog* command has the following options:

### -nNUMBER

This option is used to display test information about a number of the latest tests which is stored in the *test log* buffer. The option must be specified as *digit value option*.

### -old

This option is used to display test information which is stored in the *test log* buffer, whether it has been displayed before or not.

If no options are given the command, it displays all test information from the *test log* buffer which has not been displayed before.

## EXAMPLES

In the example on the next page, the *testlog* command has been executed. It displays all test information from the *test log* buffer which has not been displayed before. If you execute the command again, it will not display any test information.

```

baio 70 -> testlog
Test started at Thursday 1995-05-04 - 12:54:34
SCSI diagnose test on scsi0
Address/register range  Function  No of test  Access type  Pattern value
0-7                    reg      2048      word word   random 0x2fa95b2a

Test started at Thursday 1995-05-04 - 12:54:34
SCSI diagnose test on scsi0
Address/register range  Function  No of test  Access type  Pattern value
0x00100000-0x00200000  script   4096

Number of tests: 2
baio 70 -> testlog
Number of tests: 2
baio 70 ->
    
```

Figure 11-testlog-1: Example of using the testlog command

The example below shows how to display only the latest test information from the *test log* buffer.

```

baio 70 -> testlog -nl -old
Test started at Thursday 1995-05-04 - 12:54:34
SCSI diagnose test on scsi0
Address/register range  Function  No of test  Access type  Pattern value
0x00100000-0x00200000  script   4096

Number of tests: 2
baio 70 ->
    
```

Figure 11-testlog-2: Example of using the testlog command to display the latest test

**NOTICES**

- The command does not clear *test log* and *error log*.
- The command cannot be repeated. It will only run once.
- The test information can also be displayed when a test is started. This depends on how the *dispmode parameter* is set.

**testlog (system)**

**baio30x command**

**testlog (system)**

## REFERENCES

See also commands:

*set, status, errlog*

**testlog (system)**

**balo30x command**

**testlog (system)**

*This page is intentionally left blank*

time (test)

baio30x command

time (test)

**NAME**

*time* – Time a command

**SYNOPSIS**

*time* command

**command**      **COMMAND [OPTIONS]**

**DESCRIPTION**

The *time* command is used to measure the time a command needs to complete execution.

The *time* command will execute a selected command that is specified as a *command name option*. When the selected command is complete, the *time* command will display the time elapsed during execution of the selected command. The elapsed time will be shown in ms and in seconds.

The options to the *time* command, must always be specified.

**EXAMPLES**

```
baio 70 -> time memfast -dpmem -r#0x100000
Command time: 0 second. 190 ms.
baio 70 ->
```

---

Figure 11-time-1: Example of the time command

**NOTICES**

The command cannot be executed while running in *fatal error mode*.

**REFERENCES**

See also commands:  
*status*.

time (test)

baio30x command

time (test)

*This page is intentionally left blank*

**NAME**

*version* – Display program version

**SYNOPSIS**

*version*

**DESCRIPTION**

The *version* command displays the version of the boot prom and the current version of the Diagnostic Program which has been loaded to the unit.

**NOTICES**

The command does not clear *test log* and *error log*.

The command cannot be repeated. It will only run once.

**EXAMPLES**

```
baio 70 -> version
Debugger PROM:      Version 1.00      Date 1995-03-20 - 10:00:19
Diagnostic program: Version 1.03      Date 1995-08-21 - 10:23:14
baio 70 ->
```

Figure 11-version-1: Example of the version command

**version (system)**

**baio30x command**

**version (system)**

*This page is intentionally left blank*



# 12 CPU30x MODULE

---

<b>Hardware Description</b>	12-1
-----------------------------	------

---

<b>LEDs Description</b>	12-2
-------------------------	------

---

<b>cpu30x Commands</b>	12-3
cache command	12-3
cachediag command	12-3
cachestat command	12-3
cachetest command	12-3
config command	12-3
date command	12-3
debug command	12-3
errlog command	12-3
help command	12-3
memaddr command	12-3
memcmp command	12-3
memdata command	12-3
memdisp command	12-3
memexer command	12-3
memfast command	12-3
memgallop command	12-3
memory command	12-3
memtas command	12-3
menu command	12-3
remote command	12-3
repeat command	12-3
set command	12-3

## Table of Contents

status command	12-3
sysbus command	12-3
test command	12-3
testlog command	12-3
time command	12-3
version command	12-3

# Figures and Tables

---

<b>Figure 12-2-1</b>	LEDs status when running selftest	12-2-1
<b>Figure 12-2-2</b>	LEDs status when running SMES Diagnostic Programs	12-2-1
<b>Figure 12-cache-1</b>	Example of the cache command displaying all commands from the script	12-3
<b>Figure 12-cache-2</b>	Example of cache command selecting a few commands from the script	12-3
<b>Figure 12-cachediag-1</b>	Example of the cachediag command	12-3
<b>Figure 12-cachestat-1</b>	Example of cachestat command displaying secondary cache status	12-3
<b>Figure 12-cachestat-2</b>	Example of cachestat command displaying primary data cache status	12-3
<b>Figure 12-cachetest-1</b>	Example of the cachetest command	12-3
<b>Figure 12-config-1</b>	Example on a configuration of an SMES system	12-3
<b>Figure 12-date-1</b>	Example of using the date command	12-3
<b>Figure 12-errlog-1</b>	Example of using the errlog command	12-3
<b>Figure 12-errlog-2</b>	Example of using the errlog command to display the latest error	12-3
<b>Figure 12-help-1</b>	Example of help command which displays the special keys	12-3
<b>Figure 12-help-2</b>	Example of help command which displays the command help information	12-3
<b>Figure 12-memaddr-1</b>	Example of the memaddr command	12-3
<b>Figure 12-memcmp-1</b>	Example of the memcmp command	12-3
<b>Figure 12-memdata-1</b>	Example of the memdata command	12-3
<b>Figure 12-memdisp-1</b>	Example of the memdisp command	12-3

## Table of Contents

<b>Figure 12-memexer-1</b>	Example of the memexer command	12-3
<b>Figure 12-memexer-2</b>	Example of the memexer command executes as a remote command	12-3
<b>Figure 12-memfast-1</b>	Example of the memfast command	12-3
<b>Figure 12-memgallop-1</b>	Example of the memgallop command	12-3
<b>Figure 12-memory-1</b>	Example of memory command displaying all commands from the script	12-3
<b>Figure 12-memory-2</b>	Example of memory command selecting a few commands from the script	12-3
<b>Figure 12-memtas-1</b>	Example of the memtas command	12-3
<b>Figure 12-menu-1</b>	Commands on the cpu30x module in short format	12-3
<b>Figure 12-menu-2</b>	Commands on the cpu30x module in long format	12-3
<b>Figure 12-remote-1</b>	Example of using the remote command	12-3
<b>Figure 12-repeat-1</b>	Example of using the repeat command	12-3
<b>Figure 12-set-1</b>	Example of using set command to display current parameter values	12-3
<b>Figure 12-set-2</b>	Example of using set command to change a parameter	12-3
<b>Figure 12-status-1</b>	Example of the status command	12-3
<b>Figure 12-sysbus-1</b>	Example of using sysbus to read global control space id register	12-3
<b>Figure 12-test-1</b>	Example of the test command displaying all commands from the script	12-3
<b>Figure 12-testlog-1</b>	Example of using the testlog command	12-3
<b>Figure 12-testlog-2</b>	Example of using the testlog command to display the latest test	12-3
<b>Figure 12-time-1</b>	Example of the time command	12-3
<b>Figure 12-version-1</b>	Example of the version command	12-3

## Hardware Description

NOT IMPLEMENTED YET.

*This page is intentionally left blank*

## LEDs Description

This chapter describes the *LEDs* status on the front of the *cpu30x* module.

The *Ax LED* and *Bx LED* are only mounted on the *cpu30x* module, if the module has *CPU agents* mounted on the *local bus*.

ERRORx	BUSYx	Ax	Bx	Description
off	off	off	off	Reset state
off	off	off	on	Selftest running
off	on	off	on	Running in PROM menu or debugger
on	off	off	on	Selftest failed
on	on	off	on	Selftest failed with exception error

Figure 12-2-1: LEDs status when running selftest

ERRORx	BUSYx	Ax	Bx	Description
off	off	off	off	Reset state during boot
off	off	on	off	Idle state
off	on	on	off	A <i>test command</i> is running
on	off	on	off	Idle state and the <i>error log</i> buffer has an error that has not been displayed
on	on	on	off	A <i>test command</i> is running and the <i>error log</i> buffer has an error that has not been displayed

Figure 12-2-2: LEDs status when running SMES Diagnostic Programs

*This page is intentionally left blank*



**NAME**

*cache* - Test script for cache

**SYNOPSIS**

*cache* [-dMEMORY\_DEVICES] [-tTEST\_FUNCTIONS] [-pDISP\_CMDS]

[-dMEMORY\_DEVICES]      [-d[pmem | dmem | nvm | global |  
subm1 | subm2 | subm3] [\*,\*,]]

[-tTEST\_FUNCTIONS]      [-t[fast | coherent] [\*,\*,]]

[-pDISP\_CMDS]            [-p]

**DEFAULTS**

*cache* -dglobal -tfast,coherent

**DESCRIPTION**

This command is a *script command*, and it is used to run a complete test of the CPU cache (*primary and secondary cache*) located on the *pmd30x* submodule and the *cache snooper* located on the *cpu30x* module. The command executes other commands which are all cache tests. It is possible to specify which memory to be used while testing the cache and which commands of the *cache script* to be executed. This is done by means of the **MEMORY\_DEVICES** and **TEST\_FUNCTIONS** options.

The *cache* command has the following options:

**-dMEMORY\_DEVICES**

This option is used to select the memories to be used when testing the cache. The option must be specified as a *multi word symbol option*, and the following memories can be specified:

**global**      *Global memory.*

**local**        Only available if the module has a *local bus* mounted and a memory module is installed on the *local bus*.

**-tTEST\_FUNCTIONS**

This option is used to specify which test functions should be selected from the *cache script*. The option must be specified as a *multi word symbol option*, and the following test functions can be

specified:

**fast** If this test function is selected, the *primary and secondary cache* will be tested fast and not thoroughly.

**coherent** If this test function is selected, the *primary and secondary cache* and the *cache snooper* will be tested thoroughly. This is done by running commands that test the *cache coherence* between other CPU submodules on the *cpu30x* module.

**-pDISP\_CMDS**

If this option is specified, the selected commands from the script will only be displayed, not executed.

Without option the command executes all the commands from the scripts.

## EXAMPLES

```

cpu 00 -> cache -p
Test script for cache
Device Function Command script
global fast cachediag -ftag -mlimit; cachediag -fdata -mlimit
local fast cachediag -ftag -mlimit
global coherent cachetest -dglobal -n32 -r#0x80000 -frdexc; cachetest
-dglobal -n32 -r#0x80000 -frdsh; cachetest -dglobal -n32
-r#0x80000 -frwexc; cachetest -dglobal -n32 -r#0x80000
-frwsh; memtas -dglobal -cshared -n32 -aoff -r#0x180000;
memtas -dglobal -n16 -cshared
local coherent cachetest -dlocal -n32 -r#0x80000 -frdexc; cachetest
-dlocal -n32 -r#0x80000 -frdsh; cachetest -dlocal -n32
-r#0x80000 -frwexc; cachetest -dlocal -n32 -r#0x80000
-frwsh; memtas -dlocal -cshared -n32 -aoff -r#0x180000;
memtas -dlocal -n16 -cshared

cpu 00 ->

```

Figure 12-cache-1: Example of the cache command displaying all commands from the script

```

cpu 00 -> cache -p -tcoherent
Test script for cache
Device Function Command script
global coherent cachetest -dglobal -n32 -r#0x80000 -frdexc; cachetest
-dglobal -n32 -r#0x80000 -frdsh; cachetest -dglobal -n32
-r#0x80000 -frwexc; cachetest -dglobal -n32 -r#0x80000
-frwsh; memtas -dglobal -cshared -n32 -aoff -r#0x180000;
memtas -dglobal -n16 -cshared

cpu 00 ->

```

Figure 12-cache-2: Example of cache command selecting a few commands from the script

## NOTICES

The command cannot be used while running in *fatal error mode*.

## BUGS

If you are running the command and the *remote parameter* is set to on (Please refer to the *set* command) please note the following:

- When the command runs the *memtas* or *cachetest* command, this command will perform a *remote command* on the other CPU units. These units (*slave remote units*) must respond to the *remote command*. This is only possible, if the units execute commands from the *command line* buffer. If the *master remote unit* has not got a response from all *slave remote units* after 5 minutes the unit will fail with time-out.
- If you cancel a running command on a *master remote unit* with **Ctrl-C**, the unit will be hung up until all *slave remote unit* commands are complete or cancelled.
- If a test fails when executing a *remote command*, and the unit waits to display the error message, the test must be cancelled before all other *master/slave remote units* can continue.

## REFERENCES

See also commands:

*cachediag*, *cachestat*, *cachetest*, *memtas*, *set*.

**NAME**

*cachediag* – Cache diagnose test in tag/data field

**SYNOPSIS**

*cachediag* [-dDEVICE] [-rADDR\_RANGE] [-fFUNCTION] [-mMODE] [-pPATTERN] [-vPAT\_VALUE]

[-dDEVICE]            [-d[scache | icache | dcache]]

[-rADDR\_RANGE]      [-r[START | #SIZE | START-END | START#SIZE]]

[-fFUNCTION]        [-f[tag | data]]

[-mMODE]            [-m[limit | all]]

[-pPATTERN]        [-p[random | value | count | cntdown]]

[-vPAT\_VALUE]      [-vDIGIT]

**DEFAULTS**

*cachediag* -dscache -r#0x100000 -ftag,data -mlimit  
-prandom -vX

**DESCRIPTION**

The *cachediag* command is used to test of the *CPU cache* (*primary and secondary cache* memory) located on the *pm30x* (*CPU*) submodule. This is done by testing the tag and data fields in the cache memory. This is very useful to diagnose an error in the *primary and secondary cache* memory.

It is possible to select which type of *CPU cache* and which type of test the command shall use. This is done by means of the **DEVICE** and **FUNCTION** options.

The cache tag field test will test the tag fields in the cache memory. This is done by writing a data pattern in the cache tag fields. After that the tag fields will be checked. All this will be done by means of a special *CPU cache* instruction.

The cache data test will test the data fields in the cache memory. This is done by writing a data pattern in the cache data fields. After that the data fields will be checked.

The *CPU cache exception* and *CPU cache* will be disabled, before running the tests. When the command is completed or cancelled, the command will clear the cache, before it sets the cache to the current state of the *cache* and *cache\_ecc* parameter.

The *cachediag* command has the following options:

#### **-dDEVICE**

This option is used to select which cache the command has to use. The option must be specified as a *word symbol option*, and the following caches can be specified:

- scache**     *Secondary cache with a size from 1 to 4 Mbyte.*
- icache**     *Primary instruction cache with a size from 8 to 32 Kbyte.*
- dcache**     *Primary data cache with a size from 8 to 32 Kbyte.*

The cache size depends on which type of *CPU* submodule is installed on the *cpu30x* module.

#### **-rADDR\_RANGE**

This option is used to specify the memory address range in the *global memory* which the command has to use. The option must be specified as a *value range option*. Default, the command will reserve the next free memory array which can hold the cache.

#### **-fFUNCTION**

This option is used to specify which test functions the command shall perform. The option must be specified as a *multi word symbol option*, and the following tests can be specified:

- tag**     If this test function is selected, the command will perform a test of the tag field in the cache.
- data**     If this test function is selected, the command will perform a test of the data in the cache.

**-mMODE**

This option is used to specify which mode the command has to use while running the test. The option only has effect, if the command runs the cache tag field test. The option must be specified as a *word symbol option*, and the following modes can be specified:

**limit** In this mode, the state bits in the cache tag field will not be tested. This is recommended if running the test in the Supermax Enterprise Server with more than one CPU because of inconsistency of the *cache coherence* between the CPU modules.

**all** In this mode, all bits in tag fields will be tested.

**-pPATTERN**

Same description as the *cachetest* command.

**-vPAT\_VALUE**

Same description as the *cachetest* command.

**EXAMPLES**

```

cpu 00 -> cachediag
Test started at Monday 1995-03-20 - 12:40:21
Cache diagnose test in tag/data field in Global memory
Cache  Cache address range  Function  Mode  Pattern  value
scache 0x00900000-0x00a00000 tag      limit  random  0x2f6d77b5

Test started at Monday 1995-03-20 - 12:40:24
Cache diagnose test in tag/data field in Global memory
Cache  Cache address range  Function  Mode  Pattern  value
scache 0x00900000-0x00a00000 data      limit  random  0x2f6d77b5
cpu 00 ->
    
```

Figure 12-cachediag-1: Example of the cachediag command

cachediag (cache)

cpu30x command

cachediag (cache)

## NOTICES

The command cannot be used while running in *fatal error mode*.

If the **ADDR\_RANGE** option is set to **\*-\***, **\*\*#** or **#\***, the command will reserve the next free memory array with the necessary size.

if the **DEVICE** option is set to **icache** and the **FUNCTION** is set to **data**, no test will be performed.

## COMMAND ERRORS

If the command returns the error code *Options error*, it is because the memory range is not aligned on a 4 bytes address or because of the size of memory address range is greater than the size of the cache.

## REFERENCES

See also commands:  
*cachestat, cachetest, set*



## NAME

*cachestat* – Display cache tag status

## SYNOPSIS

*cachestat* [-dDEVICE] [-rCACHE\_OFFSET]

[-dDEVICE] [-d[socache | icache | dcache]]

[-rCACHE\_OFFSET] [-r[START | #SIZE | START-END | START#SIZE]]

## DEFAULTS

*cachestat* -dsocket -r\*-\*

## DESCRIPTION

The *cachestat* command is used to display various status information about the *primary and secondary cache* on the unit. This information is read from the tag field in the cache. It is very useful to see the cache status of the unit, if a *CPU cache exception* occurs or a coherence test has failed.

The following cache information will be displayed for each cache line:

- Offset address in the cache.
- Tag field value.
- Data ECC status on the cache line.
- Memory address.
- Cache state (Invalid, clean/dirty exclusive, shared or dirty shared).
- Tag field parity/ECC status.

The *cachestat* command has the following options:

### -dDEVICE

Same description as the *cachediag* command.

### -rCACHE\_OFFSET

This option is used to specify the offset range where the command has to read the tag status in the cache. The option must be specified as a *value range option*. The offset range can also be

a memory address range. In this case the command will only use the offset address to the cache. Default, the command will display all the tag fields in the cache.

### EXAMPLES

```

cpu 00 -> cachestat -r#256
Cache tag status from Secondary cache
Offset  Tag value  ECC  Memory add  Cache state  Status
0x000000 0x00000000 0x21 0x00040000  Dirty exclusive
0x000080 0x00000000 0x26 0x000400080  Dirty exclusive
cpu 00 ->
    
```

Figure 12-cachestat-1: Example of cachestat command displaying secondary cache status

```

cpu 00 -> cachestat -r#32 -dicache
Cache tag status from Primary data cache
Offset  Tag value  ECC  Memory add  Cache state  Status
0x000000 0x00002881 0x4f 0x000028000  Clean exclusive
0x000010 0x00002881 0xff 0x000028010  Clean exclusive
cpu 00 ->
    
```

Figure 12-cachestat-2: Example of cachestat command displaying primary data cache status

### NOTICES

The command does not clear *test log* and *error log*.

The command always displays the cache status information, no matter how the *dispmode parameter* is set.

### REFERENCES

See also commands:

*cachediag, cachetest, memtas, set*

**NAME**

*cachetest* – Cache coherence test

**SYNOPSIS**

*cachetest* [-dDEVICE] [-rADDR\_RANGE] [-uUNIT\_NO] [-fFUNCTION] [-nNO\_OF\_TEST] [-pPATTERN] [-vPAT\_VALUE]

[-dDEVICE]            [-d[global | local]]

[-rADDR\_RANGE]      [-r[START | SIZE | START-END | START#SIZE]]

[-uUNIT\_NO]         [-u[all | own | HEX]]

[-fFUNCTION]        [-f{rdexc | rdsh | rwexc | rwsh}[,\*]]

[-nNO\_OF\_TEST]     [-nDIGIT]

[-pPATTERN]        [-p[random | value | count | cntdown]]

[-vPAT\_VALUE]      [-vDIGIT]

**DEFAULTS**

*cachetest* -dglobal -r#0x100000 -uall -fALL -n4 -prandom -vX

**DESCRIPTION**

The *cachetest* command is used to test the *cache coherence* between the CPU units in a Supermax Enterprise Server. The command requests other units to execute a *remote command* that will perform a memory read test in a memory, at same the time as the *cachetest* command itself performs the same test. This is very useful to diagnose an error on the *cache snooper* located on the *cpu30x* module.

The test will be performed as follows:

- The memory array is initialized with a data pattern type, specified with the **PATTERN** and **PAT\_VALUE** options. The CPU will perform the access with the *coherent exclusive* cache algorithm.
- Then the command will request another unit(possibly more units) to perform a *remote command*. The *slave remote units* can be specified with the **UNIT\_NO** option.

- When all *slave remote units* have made a response to the *master remote unit*, the units begin testing. This is done by performing a number of read or read/write *word accesses* to the same address. All the units will check the read data for correct contents. All this is repeated for all addresses in the memory space. It is possible to specify the access types and the number of times to repeat the access. This is done by means of the **FUNCTION** and **NO\_OF\_TEST** options.

When a *slave remote unit* performs the test, it will execute the *memexer* command. This means that the *slave remote units* will always log the test in its own *test log* buffer. If a unit finds errors during the test, it will also log the error in its own *error log* buffer.

The *cachetest* command has the following options:

#### **-dDEVICE**

This option is used to select which memory the test has to use. The option must be specified as a *word symbol option*, and the following memories can be specified:

**global**     *Global memory.*

**local**       *Local memory.* This is only available if the module has a *local bus* mounted, and a memory module is installed on the *local bus*.

#### **-rADDR\_RANGE**

This option is used to specify the memory address range where the test has to write and read data in the memory. The option must be specified as a *value range option*. Default, the command will reserve the next free memory array with the necessary number of bytes up to 1 Mb.

#### **-uUNIT\_NO**

This option is used to specify the **slave remote unit** that also shall perform the *cache coherence* test. The option must be specified as a *word symbol or hex value option*. If specified as a hex value, the first hex digit is the slot (position) number and the second digit is the subposition number of the *slave remote unit*, otherwise the following predefined units can be specified:

- own** No *remote command* will be requested to another unit. (Only this unit will perform the test). This is default, if the *remote parameter* is set to **off**.
- all** All other *CPU* units will be requested to execute a *remote command* that perform the test. This is default, if the *remote parameter* is set to **on**.

### **-fFUNCTION**

This option is used to specify which test function the command shall perform during access to a memory. The option must be specified as a *double word symbol option*. The test function can be different for the *master and slave remote unit*. The first word symbol will be used for *master remote unit* and second for the *slave remote units*. The following test functions can be specified:

- rdexc** This will perform a read access with the *coherent exclusive* cache algorithm.
- rdsh** This will perform a read access with the *coherent exclusive on write* cache algorithm.
- rwexc** This will perform a read and write access with the *coherent exclusive* cache algorithm.
- rwsh** This will perform a read and write access with the *coherent exclusive on write* cache algorithm.

### **-nNO\_OF\_TEST**

This option is used to specify how many times a unit shall access the same address. It must be specified as a *digit value option*.

### **-pPATTERN**

This option is used to specify a data pattern type together with the **PAT\_VALUE** option. The **PATTERN** option must be specified as a *word symbol option*, and the following pattern types can be specified:

cachetest (cache)

cpu30x command

cachetest (cache)

<b>random</b>	This pattern type will generate a 32 bit random data pattern from the <b>PAT_VALUE</b> . If the same <b>PAT_VALUE</b> is specified, the same random pattern is generated.
<b>value</b>	This pattern type will generate a 32 bit data value specified with the <b>PAT_VALUE</b> option.
<b>count</b>	This pattern type will generate a 32 bit count data pattern calculated as follows: <b>PAT_VALUE</b> + (offset / 4).
<b>cntdown</b>	This pattern type will generate a 32 bit count-down data pattern calculated as follows: <b>PAT_VALUE</b> - (offset / 4).

**-vPAT\_VALUE**

This option is used to specify a data value for the pattern type. The option must be specified as a *digit value option*. Default, the value is a value from the internal system clock.

## EXAMPLES

```

cpu 00 -> cachetest -u01
Test started at Monday 1995-03-20 - 12:41:20
Cache coherence test in Global memory
Memory range      Unit  Functions      Numbers  Pattern value
0x00a00000-0x00b00000 all  rdexc - rdexc  4        random  0x2f6d77f0

Test started at Monday 1995-03-20 - 12:41:31
Cache coherence test in Global memory
Memory range      Unit  Functions      Numbers  Pattern value
0x00a00000-0x00b00000 all  rdsh  - rdsh    4        random  0x2f6d77f0

Test started at Monday 1995-03-20 - 12:41:39
Cache coherence test in Global memory
Memory range      Unit  Functions      Numbers  Pattern value
0x00a00000-0x00b00000 all  rwexc - rwexc  4        random  0x2f6d77f0

Test started at Monday 1995-03-20 - 12:41:51
Cache coherence test in Global memory
Memory range      Unit  Functions      Numbers  Pattern value
0x00a00000-0x00b00000 all  rwsh  - rwsh    4        random  0x2f6d77f0
cpu 00 ->

```

Figure 12-cachetest-1: Example of the cachetest command

## NOTICES

The command cannot be used while running in *fatal error mode*.

## COMMAND ERRORS

If the command returns the error code *Options error*, it is because the memory range is not aligned on a 4 bytes address or because you have selected a *remote unit (CPU)* that is not installed.

## BUGS

When running the command and performing the test on other *CPU* units (this is done default if the *remote parameter* is to *on*) please note the following:

- The command will perform a *remote command* to the other *CPU* units. These units (*slave remote units*) must make a response to the *remote command*. This is only possible, if the units execute commands from the *command line* buffer. If the *master remote unit* has not got a response from all *slave remote units* after 5 minutes the unit will fail with time-out.
- If you cancel a running command on a *master remote unit* with **Ctrl-C** , the unit will be hung up until all *slave remote unit* commands are complete or cancelled.
- If a test fails when executing a *remote command*, and the unit waits to display the error message, the test must be cancelled before all other *master/slave remote units* can continue.

## REFERENCES

See also commands:

*cachediag, cachestat, memexer, memtas, set*



**NAME**

*config* – System and module configuration

**SYNOPSIS**

*config*

**DESCRIPTION**

The *config* command is used to display the configuration of the Supermax Enterprise Server system. The configuration information is taken from a table which is made by the *master unit* using the *boot* command.

**EXAMPLES**

```

cpu 00 -> config
Backplane: BPL304-1                Global bus: 08 slots 33.33 MHz.
Serial no: 1          FCN no.: 0    Local bus: 08 slots 33.33 MHz
Unit Type  Module name  Serial no FCN no.  Module configuration
00 CPU     CPU301-1      30        0          MIPS4400/150MHz    01 MB cache
30 MEM     MEM301-1       15        147        BaseLo add 0000 MB 016 MB memory
70 IOC     BAI0301-1        14        146        MIPS3052/33MHz    008 MB memory
          SUBM1 SCU302-1    2         0          128 KB memory
cpu 00 ->
    
```

Figure 12-config-1: Example of a configuration of an SMES system

**NOTICES**

The command does not clear *test log* and *error log*.

The command cannot be repeated. It will only run once.

**config (boot)**

**cpu30x command**

**config (boot)**

*This page is intentionally left blank*

**NAME**

*date* – Read or set date

**SYNOPSIS**

*date* [-dDATE] [-tTIME] [TIME\_ZONE]

[-dDATE]            [-dYYYY-MM.DD]

[-tTIME]            [-tHH:MM:SS]

[TIME\_ZONE]        [[+|-][HHMM]]

**DESCRIPTION**

The *date* command is used to display or set the current date and time on the unit.

The startup time is initialized from the *master unit* when it is booting. The *master unit* reads the current date and time from the Service computer submodule and distributes a copy to all other units in the Supermax Enterprise Server.

If no Service computer submodule is installed in the Supermax Enterprise Server, the date and time will be set to the date of the *master unit* version.

The *date* command has the following options:

**-dDATE**

This option is used to change the current date on the unit, and it must be specified as a *date option* on the form "YYYY-MM.DD".

**-tTIME**

This option is used to change the current time on the unit, and it must be specified as a *time option* on the form "HH:MM:SS".

**TIMEZONE**

This option is used to change the current time zone on the unit, and it must be specified as a *time zone option* on the form "+HHMM" or "-HHMM". The time zone must be specified as the difference between the actual time zone and UTC (GMT) in hours and minutes.

If no options are given, the *date* command displays the current date, time, and time zone of the unit.

date (system)

cpu30x command

date (system)

## EXAMPLES

```
cpu 00 -> date -d1995-04-06 -t13:40:10 +0200
Thursday 1995-04-06 - 13:40:10 GMT +0200
cpu 00 -> date
Thursday 1995-04-06 - 13:40:11 GMT +0200
```

Figure 12-date-1: Example of using the date command

## NOTICES

The command does not clear *test log* and *error log*.

The command cannot be repeated. It will only run once.

When setting date, time or time zone it does not affect other units.

During the "Daylight Saving Time" period the time zone must be changed.

## BUGS

The internal clock on the unit does not run while the unit is running in the *fatal error mode*.

## REFERENCES

See also commands:

debug (system)

cpu30x command

debug (system)

**NAME***debug* – Call debugger**SYNOPSIS***debug***DESCRIPTION**

The *debug* command is a diagnostic feature which normally isn't used by the operator. The command is used to debug the Diagnostic Programs on the unit. If *debug* is accidentally entered, use **C**

**Return** to exit.

**NOTICES**

The command doesn't clear *test log* and *error log*.

The command cannot be repeated. It will only run once.

If the **debug button** on the *scu30x* (Service Computer) submodule is activated, all the units in Supermax Enterprise Server will enter the debugging mode.

**debug (system)**

**cpu30x command**

**debug (system)**

*This page is intentionally left blank*

errlog (system)

cpu30x command

errlog (system)

**NAME**

*errlog* - Display errors from the error log

**SYNOPSIS**

*errlog* [-nNUMBER] [-old]

[-nNUMBER] [-nDIGIT]

[-old] [-old]

**DESCRIPTION**

The *errlog* command is used to display error information which is stored in the *error log* buffer. The *error log* buffer is cleared every time the *command line* buffer contains a *test command* and the *command interpreter* starts executing the first command from the *command line* buffer.

The maximum number of errors in the *error log* buffer, which has not been displayed yet, is 64. If this number is exceeded an overrun is detected, and the first 64 errors are saved, and the last arrived is discarded.

The *errlog* command has the following options:

**-nNUMBER**

This option is used to display a number of the latest errors which are stored in the *error log* buffer. The option must be specified as a *digit value option*.

**-old**

This option is used to display error information which is stored in the *error log* buffer, whether it has been displayed before or not.

If no options are given, the command displays all error information from the *error log* which has not been displayed before.

**EXAMPLES**

In the example on the next page, the *errlog* command has been executed. It displays all error information from the *error log* buffer which has not been displayed before. If you execute the command again, it will not display any error information.

errlog (system)

cpu30x command

errlog (system)

```
cpu 00 -> errlog
Error occurred at Tuesday 1995-07-11 - 10:26:14
Memory read fault in Global memory
Memory address: 0x00200000   Write data: 0x2a275082 <word access>
                             Read data: 0x00000000 <word access>

Error occurred at Tuesday 1995-07-11 - 10:26:14
Memory read fault in Global memory
Memory address: 0x00200004   Write data: 0x2a275083 <word access>
                             Read data: 0x00000000 <word access>

Number of errors: 2
Number of overruns: 0
cpu 00 -> errlog
Number of errors: 2
Number of overruns: 0
cpu 00 ->
```

Figure 12-errlog-1: Example of using the errlog command

The example below shows how to display only the latest error information from the *error log* buffer.

```
cpu 00 -> errlog -old -n1
Error occurred at Tuesday 1995-07-11 - 10:26:14
Memory read fault in Global memory
Memory address: 0x00200004   Write data: 0x2a275083 <word access>
                             Read data: 0x00000000 <word access>

Number of errors: 2
Number of overruns: 0
cpu 00 ->
```

Figure 12-errlog-2: Example of using the errlog command to display the latest error

## NOTICES

The command does not clear *test log* and *error log*.

The command cannot be repeated. It will only run once.



**errlog (system)****cpu30x command****errlog (system)**

The error information can also be displayed when the error has occurred. This depends on how the *dispmode parameter* is set.

**REFERENCES**

See also commands:

*set, status, testlog*

**errlog (system)**

**cpu30x command**

**errlog (system)**

*This page is intentionally left blank*

**NAME**

*help* – Display command information and options

**SYNOPSIS**

*help* [-all]DISPLAY\_COMMANDS] [command]

[-all]DISPLAY\_COMMANDS]  
[-all]

[command] [COMMAND [OPTIONS]]

**DESCRIPTION**

The *help* command is used to display *command help information*. The command can display special keys or *command help information* for one or for all command(s).

The *help* command has the following options:

**-all**

If this option is selected, the *help* command displays *command help information* for all commands on the *cpu30x* module.

**command**

If a command name is specified, the *help* command displays *command help information* only for the selected command.

If no options are given, the *help* command displays all special keys.

## EXAMPLES

```
cpu 00 -> help
SPECIAL KEYS:
ESC # SP  Deselect a unit. ESC <--> CTRL-A
ESC # CR  Select the master unit. ESC <--> CTRL-A
ESC # U CR Select a unit where U is slot/subpos. ESC <--> CTRL-A
CTRL-C    Cancel a running test
CTRL-E    Soft cancel a running test
CTRL-Q    Start printing on the terminal (XON)
CTRL-S    Stop printing on the terminal (XOFF)
CTRL-P    Start printing SCSI command on the terminal
CTRL-N    Stop printing SCSI command on the terminal
CTRL-L    Move cursor one position to the right
CTRL-H    Move cursor one position to the left
CTRL-K    Insert single character
CTRL-J    Delete single character
DEL       Delete to end of line
TAB       Move cursor 8 positions to the right
??        Display last commands
?xx       Find and edit previous command line
!xx       Find and run previous command line
;         Separate commands
help -all Display help information for all commands
help CMD  Display help information for the command
menu      Display commands
```

---

Figure 12-help-1: Example of help command which displays the special keys

help(system)

cpu30x command

help(system)

```
cpu 00 -> help help
help      Display command information and options
          [-allDISPLAY_COMMANDS] [-all]
          [command]                [COMMAND [OPTIONS]]

cpu 00 -> help -?
help      Display command information and options
          [-allDISPLAY_COMMANDS] [-all]
          [command]                [COMMAND [OPTIONS]]

cpu 00 ->
```

---

Figure 12-help-2: Example of help command which displays the command help information

## NOTICES

The command does not clear *test log* and *error log*.

The command cannot be repeated. It will only run once.

## REFERENCES

See also commands:

*menu*

**help (system)**

**cpu30x command**

**help (system)**

*This page is intentionally left blank*

**NAME**

*memaddr* – Memory address bit test

**SYNOPSIS**

*memaddr* [-dDEVICE] [-rADDR\_RANGE] [-aACCESS] [-cCACHE] [-bADDR\_BITS] [-vPAT\_VALUE]

[-dDEVICE] [-d[global | local]]

[-rADDR\_RANGE] [-r[START | #SIZE | START-END | START#SIZE]]

[-aACCESS] [-a[word | dword | short | byte] [,\*]]

[-cCACHE] [-c[on | off | shared | update]]

[-bADDR\_BITS] [-b[BIT RANGE FROM 0 TO 63]]

[-vPAT\_VALUE] [-v[zero | ones | [DIGIT]]]

**DEFAULTS**

*memaddr* -dglobal -r#0x100000 -aword -con -b0-63 -vzero

**DESCRIPTION**

The *memaddr* command is used to test a single address bit in a memory. This is very useful to diagnose fault on an address bit in the memory.

The test will write a value in a memory. The value is dependent on which address bit is tested. If the current address bit is one, the value from the *PAT\_VALUE* option is used. And if the address bit is zero, the inverted value is written. After writing all data, the memory is checked for correct data contents. All this is repeated for all address bits which are selected with the *ADDR\_BITS* option in the same memory array.

The *memaddr* command has the following options:

**-dDEVICE**

Same description as the *memfast* command.

**-rADDR\_RANGE**

Same description as the *memfast* command.

memaddr (memory)

cpu30x command

memaddr (memory)

**-aACCESS**

Same description as the *memfast* command.

**-cCACHE**

Same description as the *memfast* command.

**-bADDR\_BITS**

This option is used to select which address bits you want to test. The option must be specified as a *bit range option*. Default, the bit range will be set to all address bits in the selected memory.

**-vPAT\_VALUE**

This option is used to specify a 32 bit data value which is written when the address bit is one. If the address bit is zero the inverted value is written. The option must be specified as a *word symbol or digit value option*, and the following predefined values can be specified:

**zero**     The same value as 0x00000000.

**ones**     The same value as 0xffffffff.

**EXAMPLES**

```

cpu 00 -> memaddr -r*#0x200000 -b4,6
Test started at Tuesday 1995-06-06 - 10:48:43
Memory address bit test in Global memory
Start address End address Cache Write Read Address bit Pattern value
0x00200000 0x00400000 on word word 4 0x00000000

Test started at Tuesday 1995-06-06 - 10:48:44
Memory address bit test in Global memory
Start address End address Cache Write Read Address bit Pattern value
0x00200000 0x00400000 on word word 6 0x00000000
cpu 00 ->
    
```

Figure 12-memaddr-1: Example of the memaddr command



memaddr (memory)

cpu30x command

memaddr (memory)

## NOTICES

The command cannot be used while running in *fatal error mode*.

The command will store test information in the *test log* buffer for each tested address bit.

It is only possible to test address bits up to the number of address bits in the selected memory. If you select address bits over that number, the command will ignore these bits.

## DIAGNOSTICS

If the test finds errors, then try to diagnose them with the *memfast* or the *memdata* commands.

## COMMAND ERRORS

If the command returns the error code *Options error*, it is because the memory range is not aligned on a 4 bytes address or an 8 bytes address if using *dword access*.

## REFERENCES

See also commands:

*set, memory, memfast, memdata, memgallop, memcmp, memdisp*

**memaddr (memory)**

**cpu30x command**

**memaddr (memory)**

*This page is intentionally left blank*

**NAME**

*memcmp* – Memory move and compare test

**SYNOPSIS**

*memcmp* [-dDEVICE] [-rADDR\_RANGE] [-mdMOVE\_DEV] [-maMOVE\_ADDR] [-aACCESS] [-cCACHE] [-scSRC\_CHECK] [-pPATTERN] [-vPAT\_VALUE]

[-dDEVICE]                [-d[global | local]]

[-rADDR\_RANGE]        [-r[START | #SIZE | START-END | START#SIZE]]

[-mdMOVE\_DEV]        [-d[global | local]]

[-maMOVE\_ADDR]       [-ma[DIGIT]]

[-aACCESS]            [-a[word | dword | short | byte] [,\*]]

[-cCACHE]            [-c[on | off | shared | update] [,\*]]

[-scSRC\_CHECK]       [-sc[on | off]]

[-pPATTERN]           [-p[random | value | count | cntdown | current]]

[-vPAT\_VALUE]        [-v[DIGIT]]

**DEFAULTS**

*memcmp* -dglobal -r#0x100000 -mdSAME\_AS\_DEVICE -aword  
-con -scon -prandom -vX

**DESCRIPTION**

The *memcmp* command is used to test a memory by moving and comparing data from one memory array to another. The memory arrays can be in different memory devices. The test will be performed as follows:

- The source memory array is initialized with a data pattern type, specified with the **PATTERN** and **PAT\_VALUE** options.
- The data will be copied from the source memory array to the destination memory array.

memcmp (memory)

cpu30x command

memcmp (memory)

- The destination memory is read and compared with the source memory.
- The source memory is read and it is checked that correct data was used for initializing. This is only done if the **SRC\_CHECK** option is set to **on** and the **PATTERN** option is not set to **current**.

The *memcmp* command has the following options:

**-dDEVICE**

Same description as the *memfast* command.

**-rADDR\_RANGE**

Same description as the *memfast* command.

**-mdMOVE\_DEV**

Same description as the **DEVICE** option, only this is the destination memory device. Default, the command will set the option to the same memory as the source memory device.

**-maMOVE\_ADDR**

This option is used to specify a start memory address for the destination memory array. The size of the destination memory array will always be the same as the source memory array. The option must be specified as a *digit value option*. Default, the command will reserve the next free destination memory array.

**-aACCESS**

Same description as the *memfast* command, but the access type can be different for the source and destination memory. The first word symbol will be used for the source memory and the second for the destination memory.

**-cCACHE**

Same description as the *memfast* command, but the cache algorithm can be different for source and destination memory. The first word symbol will be used for the source memory and the second for the destination memory.

**-scSRC\_CHECK**

This option is used to specify the data check mode for the source memory. The option must be specified as a *word symbol option*, and the following two check modes can be specified:

- on** Data check will be performed on the source memory contents. This will only be done if the **PATTERN** option is not set to **current**.
- off** No check will be performed on the source memory contents.

**-pPATTERN**

This option is used to specify a data pattern type together with the **PAT\_VALUE** option. The data pattern is used to initialize the source memory array. The **PATTERN** option must be specified as a *word symbol option*, and the following pattern types can be specified:

- random** This pattern type will generate a 32 bit random data pattern from the **PAT\_VALUE**. If the same **PAT\_VALUE** is specified, the same random pattern is generated.
- value** This pattern type will generate a 32 bit data value specified with the **PAT\_VALUE** option.
- count** This pattern type will generate a 32 bit count data pattern calculated as follows: **PAT\_VALUE** + (offset / 4).
- cntdown** This pattern type will generate a 32 bit count-down data pattern calculated as follows: **PAT\_VALUE** - (offset / 4).
- current** This pattern type will use the current values in the source memory address range.

**-vPAT\_VALUE**

This option is used to specify a data value for the pattern type. The option must be specified as a *digit value option*. Default, the value is a value from the internal system clock. The option only

memcmp (memory)

cpu30x command

memcmp (memory)

has effect if the PATTERN option is not set to current.

## EXAMPLES

In the example below the *memcmp* command will perform the test as follows:

- The command writes a 32 bit counter in the *global memory* from address 0x400000 to 0x700000.
- Afterwards the data will be copied to address 0x700000 to 0xa00000.
- Then the source memory contents will be compared with the destination memory by reading the data from both address ranges.
- Finally, when the comparing is complete, the command will check the source memory contents for the above described 32 bit counter.

The access type will always be *byte accesses* in the source memory and *word accesses* in the destination memory.

```

cpu 00 -> memcmp -dglobal -r0x400000#0x300000 -abyte,word -pcount -v0
Test started at Friday 1995-06-16 - 14:49:50
Memory move and compare test in Global memory
Dir  Address      Memory  Access  Cache  Compare  Length      Pattern
src  0x00400000    global  byte   on     on       0x00300000  count
dest 0x00700000    global  word   on     on              0x00000000
cpu 00 ->
    
```

Figure 12-memcmp-1: Example of the memcmp command

## NOTICES

Same description as the *memfast* command.

## DIAGNOSTICS

If the test finds errors, then try to diagnose them by means of the *memfastc* command.

memcmp (memory)

cpu30x command

memcmp (memory)

## COMMAND ERRORS

If the command returns the error code *Options error*, it is because the memory range is not aligned on a 4 bytes address or an 8 bytes address if using *dword access*.

## BUGS

The `ADDR_RANGE` option cannot be set to `*-*`, `*#*` or `#*`, if the source and destination memory is the same memory device or the size of the source memory is bigger than the size of the destination memory.

## REFERENCES

See also commands:

*set, memory, memfast, memdata, memaddr, memgallop, memdisp*

**memcmp (memory)**

**cpu30x command**

**memcmp (memory)**

*This page is intentionally left blank*



## NAME

*memdata* – Memory data bit test

## SYNOPSIS

*memdata* [-dDEVICE] [-rADDR\_RANGE] [-aACCESS] [-cCACHE] [-bDATA\_BITS] [-vPAT\_VALUE]

[-dDEVICE]                   [-d[global | local]]

[-rADDR\_RANGE]           [-r[START | #SIZE | START-END | START#SIZE]]

[-aACCESS]               [-a[word | dword | short | byte] [,\*]]

[-cCACHE]               [-c[on | off | shared | update]]

[-bDATA\_BITS]           [-b[BIT RANGE FROM 0 TO 63]]

[-vPAT\_VALUE]           [-v[zero | ones | [DIGIT]]]

## DEFAULTS

*memdata* -dglobal -r#0x100000 -aword -con -b0-63 -vzero

## DESCRIPTION

The *memdata* command is used to test a single data bit in a memory. This is very useful to diagnose fault on a data bit in the memory.

The test will write a value in a memory. The value depends on which data bit is tested. The value is calculated as the value from the *PAT\_VALUE* option, inverted with the current data bit. After writing all data, the memory is checked for correct data contents. All this is repeated for all data bits which are selected by the *DATA\_BITS* option in the same memory array.

The *memdata* command has the following options:

**-dDEVICE**

Same description as the *memfast* command.

**-rADDR\_RANGE**

Same description as the *memfast* command.

**-aACCESS**

Same description as the *memfast* command.

**-cCACHE**

Same description as the *memfast* command.

**-bDATA\_BITS**

This option is used to select the data bits you want to test. The option must be specified as a *bit range option*. Default, the bit range will be set to all data bits in the selected memory.

**-vPAT\_VALUE**

This option is used to specify a 32 bit data value for the test. The 32 bit value is copied to the data bits 32 to 63. The option must be specified as a *word symbol or digit value option*, and the following predefined values can be specified:

**zero**     The same value as 0x00000000.

**ones**     The same value as 0xffffffff.

## EXAMPLES

```

cpu 00 -> memdata -v0xffffffff -abyte
Test started at Tuesday 1995-06-06 - 10:46:34
Memory data bit test in Global memory
Start address End address Cache Write Read Data bit Pattern value
0x00200000 0x00300000 on byte byte 0 0xffffffff

Test started at Tuesday 1995-06-06 - 10:46:35
Memory data bit test in Global memory
Start address End address Cache Write Read Data bit Pattern value
0x00200000 0x00300000 on byte byte 1 0xffffffff

Test started at Tuesday 1995-06-06 - 10:46:35
Memory data bit test in Global memory
Start address End address Cache Write Read Data bit Pattern value
0x00200000 0x00300000 on byte byte 63 0xffffffff
cpu 00 ->

```

Figure 12-memdata-1: Example of the memdata command

## NOTICES

The command cannot be used while running in *fatal error mode*.

The command stores test information in the *test log* buffer for each tested data bit.

## DIAGNOSTICS

If the test finds errors, then try to diagnose them by means of the *memfast* or the *memaddr* commands.

## COMMAND ERRORS

If the command returns the error code *Options error*, it is because the memory range is not aligned on a 4 bytes address or an 8 bytes address if using *dword access*.

## REFERENCES

See also commands:

*set, memory, memfast, memaddr, memgallop, memcmp, memdisp*

**memdata (memory)**

**cpu30x command**

**memdata (memory)**

*This page is intentionally left blank*

**NAME**

*memdisp* – Display memory contents

**SYNOPSIS**

*memdisp* [-dDEVICE] [-rADDR\_RANGE] [-aACCESS] [-cCACHE]

[-dDEVICE]            [-d[global | local]]

[-rADDR\_RANGE]      [-r[START | #SIZE | START-END |  
START#SIZE]]

[-aACCESS]           [-a[word | dword | short | byte]]

[-cCACHE]            [-c[on | off | shared | update]]

**DEFAULTS**

*memdisp* -dglobal -r0#256 -aword -con

**DESCRIPTION**

The *memdisp* command is used to display data contents from a memory on the *console terminal*. The command is very useful to diagnose memory faults which occur while running a memory test.

The memory to be displayed is specified in the **DEVICE** option. The memory address and the number of bytes which is displayed is depended on the start and size value of the **ADDR\_RANGE** option. The memory range will not be reserved while reading from the memory. This means that the memory array can be used from another module while reading the data.

The command has the following options:

**-dDEVICE**

Same description as the *memfast* command.

**-rADDR\_RANGE**

This option is used to specify the memory address range where the command has to read the data to be displayed. The option must be specified as a *value range option*.

**-aACCESS**

This option is used to specify which type of access the command shall perform during the read access to a memory. The option must be specified as a *word symbol option*. The following

access types can be specified:

- word**      *Word accesses* will be performed to the memory.
- dword**    *Double word accesses* will be performed to the memory.
- short**     *Short accesses* will be performed to the memory.
- byte**      *Byte accesses* will be performed to the memory.

**-cCACHE**

Same description as the *memfast* command.

**EXAMPLES**

```

cpu 00 -> memdisp -dglobal -r0x800000#128
Reading in Global memory from address 0x00800000 to 0x00800080
Address 0 4 8 12
0x00800000 647a5634 411a1305 1db9cfd6 fa598ca7 dzV4A.....Y..
0x00800010 d6f94978 b3990649 9038c31a 6cd87feb ..Ix...I.8..l...
0x00800020 49783cbc 2617f98d 02b7b65e df57732f Ix<.s.....^..ws/
0x00800030 bbf73000 9896ecd1 7536a9a2 51d66673 ..0.....u6..Q.fs
0x00800040 2e762344 0b15e015 e7b59ce6 c45559b7 .v#D.....UY.
0x00800050 a0f51688 7d94d359 5a34902a 36d44cfb .....e..YZ4.*6.L.
0x00800060 137409cc f013c69d ccb3836e a953403f .t.....n.S@?
0x00800070 85f2fd10 6292b9e1 3f3276b2 1bd23383 ....b...?2v...3.
cpu 00 ->
    
```

Figure 12-memdisp-1: Example of the memdisp command

**NOTICES**

The command does not clear *test log* and *error log*.

The command always display the data contents, no matter how the *dispmode* parameter is set.

**COMMAND ERRORS**

If the command returns the error code *Options error*, it is because the memory range is not aligned on a 4 bytes address or an 8 bytes address if using *dword access*.

memdisp (memory)

cpu30x command

memdisp (memory)

## REFERENCES

See also commands:

*memfast, memdata, memaddr, memgallop, memexer, memcmp.*

**memdisp (memory)**

**cpu30x command**

**memdisp (memory)**

*This page is intentionally left blank*



**NAME**

*memexer* – Memory exerciser

**SYNOPSIS**

*memexer* [-dDEVICE] [-rADDR\_RANGE] [-aACCESS] [-cCACHE] [-tTYPE] [-mMODE] [-pPATTERN] [-vPAT\_VALUE]

[-dDEVICE] [-d[global | local]]

[-rADDR\_RANGE] [-r[START | #SIZE | START-END | START#SIZE]]

[-aACCESS] [-a[word | dword | short | byte]]

[-cCACHE] [-c[on | off | shared | update]]

[-tTYPE] [-t[read | check | write | cmp | tas]]

[-mMODE] [-m[reserv | raw]]

[-pPATTERN] [-p[random | value | setdata | count | cntdown]]

[-vPAT\_VALUE] [-vLSB\_DIGIT[,MSB\_DIGIT]]

**DEFAULTS**

*memexer* -dglobal -r#0x100000 -aword -con -tread  
-mreserv -prandom -vX

**DESCRIPTION**

The *memexer* command is used to exercise write or read accesses in a memory. This is very useful to diagnose a memory fault in the memory.

It is possible to select different types of tests by means of the **TYPE** option. The following tests can be performed:

- The test performs only read accesses with or without checking the read data.
- The test performs only write accesses.
- The test performs write accesses and afterwards it reads the memory contents and checks it.

The *memexer* command has the following options:

**-dDEVICE**

Same description as the *memfast* command.

**-rADDR\_RANGE**

Same description as the *memfast* command. If the **MODE** option is set to **raw**, the command will not reserve a free memory array. Default, the command will use the memory range from 0 up to 0x10000.

**-aACCESS**

Same description as the *memdisp* command.

**-cCACHE**

Same description as the *memfast* command.

**-tTYPE**

This option is used to specify which test the command shall perform. The option must be specified as a *word symbol option*, and the following types of test can be specified:

- |              |  |
|--------------|--|
| <b>read</b>  | This test will perform read accesses to the specified memory address range. No check of the data will be done.   |
| <b>check</b> | This test will perform read accesses to the specified memory address range. The read data will be checked with the specified pattern.                    |
| <b>write</b> | This test will perform write accesses to the specified memory address range.   |
| <b>cmp</b>   | This test will perform write accesses to the specified memory address range and afterwards read the memory contents and compare it to the written value. |
| <b>tas</b>   | <b>Do not use this test, it is not implemented yet.</b>  |

**-mMODE**

This option is used to specify which mode the command has to use. The option must be specified as a *word symbol option*, and the following modes can be specified:

memexer (memory)

cpu30x command

memexer (memory)

**reserv** In this mode, the command will reserve the memory array as the normal procedure.

**raw** In this mode, the command will not reserve the memory array as the normal procedure. This means that it is possible to access a memory array that is reserved for another purpose (e.g code and stack segments used by the program itself or by another unit).

**-pPATTERN**

Same description as the *memfast* command.

**-vPAT\_VALUE**

Same description as the *memfast* command.

**EXAMPLES**

```

cpu 00 -> memexer -twrite -pcount -v0 -r0x200000-0x300000
Test started at Monday 1995-03-20 - 11:07:29
Memory exerciser in Global memory
Memory address range  Cache Access Mode Type Pattern Value
0x00200000-0x00300000 on word reserv write count 0x00000000
cpu 00 -> memexer -tcheck -pcount -v0 -r0x200000-0x300000
Test started at Monday 1995-03-20 - 11:07:39
Memory exerciser in Global memory
Memory address range  Cache Access Mode Type Pattern Value
0x00200000-0x00300000 on word reserv check count 0x00000000
cpu 00 ->
    
```

Figure 12-memexer-1: Example of the memexer command

memexer (memory)

cpu30x command

memexer (memory)

```

cpu 00 ->
Test started at Monday 1995-03-20 - 12:41:20
Remote memory exerciser in Global memory from unit 01
Memory address range  Cache  Access  Mode  Type  Pattern Value
0x00a00000-0x00b00000  on      word   raw   check  random  0x2fd77f0
cpu 00 ->

```

Figure 12-memexer-2: Example of the memexer command executed as a remote command

## NOTICES

The command cannot be used while running in *fatal error mode*.

When another unit is running the *memtas* or *cachetest* command and it requests the unit to perform a *remote command*, this command will be executed as the *remote command*. Please refer to the *remote command* for a detailed description.

## COMMAND ERRORS

If the command returns the error code *Options error*, it is because the memory range is not aligned on a 4 bytes address or an 8 bytes address if using *dword access*.

## WARNINGS

If setting the **TYPE** option to **check**, you must be sure that the data for the comparison has been written in the memory, otherwise the program will find incorrect data in the memory.

If the **MODE** option is set to **raw** and the **TYPE** option is set to **write** or **cmp**, the command will not reserve the memory array. I.e. that it is possible to perform write access in a memory array that is reserved for another purpose (e.g code and stack segments used by the program itself or by another unit). This means that the Diagnostic Programs can break down.

## REFERENCES

See also commands:

*memaddr*, *memcmp*, *memdata*, *memfast*, *memgallop*, *remote*, *set*

**NAME**

*memfast* – Memory high speed test

**SYNOPSIS**

*memfast* [-dDEVICE] [-rADDR\_RANGE] [-aACCESS] [-cCACHE] [-pPATTERN] [-vPAT\_VALUE]

[-dDEVICE]            [-d[global | local]]

[-rADDR\_RANGE]      [-r[START | #SIZE | START-END | START#SIZE]]

[-aACCESS]           [-a[word | dword | short | byte] [,\*]]

[-cCACHE]            [-c[on | off | shared | update]]

[-pPATTERN]          [-p[random | value | setdata | count | cntdown]]

[-vPAT\_VALUE]        [-vLSB\_DIGIT[,MSB\_DIGIT]]

**DEFAULTS**

*memfast* -dglobal -r#0x100000 -aword -con -prandom -vX

**DESCRIPTION**

The *memfast* command is used to perform a fast test in a memory. This is done by writing a pattern in the memory and afterwards reading the memory contents and comparing it to the written value.

The *memfast* command has the following options:

**-dDEVICE**

This option is used to select which memory to be tested. The option must be specified as a *word symbol option*, and the following memories can be specified:

**global**      *Global memory*

**local**        *Local memory.* This is only available, if the module has a *local bus* mounted and a memory module is installed on the *local bus*.

**-rADDR\_RANGE**

This option is used to specify the memory address range where the test has to write and read data in the memory. The option must be specified as a *value range option*. Default, the command will reserve the next free memory array with the necessary number of bytes up to 1 Mb.

**-aACCESS**

This option is used to specify which type of access the command shall perform during the write and read access to a memory. The option must be specified as a *double word symbol option*. The access type can be different for the write and the read access. The first word symbol will be used for write access and the second for read access. The following access types can be specified:

- word**      *Word accesses* will be performed to the memory.
- dword**     *Double word accesses* will be performed to the memory.
- short**     *Short accesses* will be performed to the memory.
- byte**      *Byte accesses* will be performed to the memory.

**-cCACHE**

This option is used to select which type of cache algorithm the CPU shall perform while writing and reading in the memory. The option must be specified as a *word symbol option*, and the following cache algorithms can be specified:

- on**        The CPU cache is used and the CPU will perform the access with the *coherent exclusive on write* cache algorithm.
- off**        The CPU cache is not used when writing and reading in the memory.
- shared**    The CPU cache is used and the CPU will perform the access with the *coherent exclusive* cache algorithm.

**update** The *CPU cache* is used and the *CPU* will perform the access with the *coherent update on write cache* algorithm. This cache mode is not supported on the *cpu30x* module at the moment.

### **-pPATTERN**

This option is used to specify a data pattern type together with the *PAT\_VALUE* option. The *PATTERN* option must be specified as a *word symbol option*, and the following pattern types can be specified:

**random** This pattern type will generate a 32 bit random data pattern from the *PAT\_VALUE*. If the same *PAT\_VALUE* is specified, the same random pattern is generated.

**value** This pattern type will generate a 32 bit data value specified with the *PAT\_VALUE* option.

**setdata** This is the same as *value* pattern, but the number of data bits in the value depends on how many data bits the selected memory has.

**count** This pattern type will generate a 32 bit count data pattern calculated as follows:  $PAT\_VALUE + (offset / 4)$ .

**cntdown** This pattern type will generate a 32 bit count-down data pattern calculated as follows:  $PAT\_VALUE - (offset / 4)$ .

### **-vPAT\_VALUE**

This option is used to specify a data value for the pattern type. The option must be specified as a *double digit value option*. Default, the *LSB\_DIGIT* value is a value from the internal system clock and the *MSB\_DIGIT* value is set to zero. The *MSB\_DIGIT* value only has effect if the *DEVICE* option is set to *global* and the *PATTERN* option is set to *setdata*.

## EXAMPLES

In the example below the *memfast* command tests the *global memory* from address 0x400000 to 0x400080 by using *byte access* while writing and using *word access* while reading. The pattern which is written is a 64 bit value.

After the test is complete a *memdisp* command is executed to display the memory contents.

```

cpu 00 -> memfast -dglobal -r0x400000#0x80 -abyte,word -psetdata
-v0x89abcdef,0x01234567
Test started at Monday 1995-03-20 - 11:48:35
Memory high speed test in Global memory
Start address End address Cache Write Read Type Pattern value
0x00400000 0x00400080 on byte word setdata 0x0123456789abcdef

cpu 00 -> memdisp -dglobal -r0x400000#0x20
Reading in Global memory from address 0x00400000 to 0x00400020
Address 0 4 8 12
0x00400000 01234567 89abcdef 01234567 89abcdef .#Eg.....#Eg....
0x00400010 01234567 89abcdef 01234567 89abcdef .#Eg.....#Eg....
cpu 00 ->

```

Figure 12-memfast-1: Example of the memfast command

## NOTICES

The command cannot be used while running in *fatal error mode*.

## DIAGNOSTICS

If the test finds errors, then try to diagnose them by means of the *memdata* or the *memaddr* command.

## COMMAND ERRORS

If the command returns the error code *Options error*, it is because the memory range is not aligned on a 4 bytes address or an 8 bytes address if using *dword access*.



memfast (memory)

cpu30x command

memfast (memory)

## REFERENCES

See also commands:

*set, memory, memaddr, memdata, memgallop, memcmp, memdisp.*

**memfast (memory)**

**cpu30x command**

**memfast (memory)**

*This page is intentionally left blank*

**NAME**

*memgallop* – Memory galloping test

**SYNOPSIS**

*memgallop* [-dDEVICE] [-rADDR\_RANGE] [-aACCESS] [-cCACHE] [-vPAT\_VALUE]

[-dDEVICE] [-d[global | local]]

[-rADDR\_RANGE] [-r[START | #SIZE | START-END | START#SIZE]]

[-aACCESS] [-a[word | dword | short | byte] [,\*]]

[-cCACHE] [-c[on | off | shared | update]]

[-vPAT\_VALUE] [-v[zero | ones | [DIGIT]]]

**DEFAULTS**

*memgallop* -dglobal -r#0x100000 -aword -con -vzero

**DESCRIPTION**

The *memgallop* command is used to test a memory with a galloping value. This is very useful to diagnose an error in the memory where a *DRAM chip* fails. The command will perform the test as described below:

- The command will initialize a memory space with the inverted value from the `PAT_VALUE` option, also called the passive value.
- Then an active value (the value from the `PAT_VALUE` option) is written to the address been tested.
- Afterwards, all the neighbouring addresses of the first address are tested. A neighbouring address is an address which only differs in one bit from the original address. It must still contain the passive value.
- When all neighbouring addresses are tested, the passive value is written back in the original address.
- This is repeated for all addresses in the memory space.

The *memgallop* command has the following options:

**-dDEVICE**

Same description as the *memfast* command.

**-rADDR\_RANGE**

Same description as the *memfast* command.

**-aACCESS**

Same description as the *memfast* command.

**-cCACHE**

Same description as the *memfast* command.

**-vPAT\_VALUE**

This option is used to specify a 32 bit galloping value for the test. The option must be specified as a *word symbol or digit value option*, and the following predefined values can be specified:

**zero** Same as value of 0x00000000.

**ones** Same as value of 0xffffffff.

## EXAMPLES

```
cpu 00 -> memgallop -dglobal -vones -abyte,word
Test started at Tuesday 1995-06-06 - 13:04:25
Memory galloping test in Global memory
Start address End address Cache Write Read Active value Passive value
0x00200000 0x00300000 on byte word 0xffffffff 0x00000000
cpu 00 ->
```

Figure 12.3-1: Example of the *memgallop* command

## NOTICES

The command cannot be used while running in *fatal error mode*.

## DIAGNOSTICS

If the test finds errors, then try to diagnose them by means of the *memfast*, *memaddr*, or the *memdata* command.

**BUGS**

It is not possible to set the **ACCESS** option to **dword**, because the *double word access* is not implemented in this test.

**COMMAND ERRORS**

If the command returns the error code *Options error*, it is because the memory range is not aligned on a 4 bytes address or an 8 bytes address if using *dword access*.

**REFERENCES**

See also commands:

*set, memory, memfast, memaddr, memdata, memcmp, memdisp*

**memgallop (memory)**

**cpu30x command**

**memgallop (memory)**

*This page is intentionally left blank*

memory (memory)

cpu30x command

memory (memory)

**NAME***memory* – Test script for memory**SYNOPSIS***memory* [-dMEMORY\_DEVICES] [-tTEST\_FUNCTIONS] [-pDISP\_CMDS]

[-dMEMORY\_DEVICES] [-d[global | local] [\*,\*,]]

[-tTEST\_FUNCTIONS] [-t[fast | function] [\*,\*,]]

[-pDISP\_CMDS] [-p]

**DEFAULTS**

memory -dglobal,local -tfast,function

**DESCRIPTION**

This command is a *script command*, and it is used to run a complete test of the memories connected to the *cpu30x* module. The command executes other commands which are all memory tests. It is possible to specify which memories to be tested and which commands of the *memory script* to be executed. This is done by means of the **MEMORY\_DEVICES** and **TEST\_FUNCTIONS** options.

The *memory* command has the following options:

**-dMEMORY\_DEVICES**

This option is used to select the memories to be tested. The option must be specified as a *multi word symbol option*, and the following memories can be specified:

**global**     *Global memory*

**local**       *Local memory.* This is only available, if the module has a *local bus* mounted and a memory module is installed on the *local bus*.

**-tTEST\_FUNCTIONS**

This option is used to specify which test functions should be selected from the *memory script*. The option must be specified as a *multi word symbol option*, and the following test functions can be specified:

memory (memory)

cpu30x command

memory (memory)

- fast** If this test function is selected, the memories will only be tested fast and not thoroughly.
- function** If this test function is selected, the memories will be tested thoroughly.

**-pDISP\_CMDS**

If this option is specified, the selected commands from the script will only be displayed, not executed.

Without options the command executes all commands from the scripts.

**EXAMPLES**

```

cpu 00 -> memory -p
Test script for memory
Device Function Command script
global fast memfast -dglobal -r#0x800000; memcmp -dglobal -r#0x800000;
memgallop -dglobal -r#0x180000 -v0; memgallop -dglobal
-r#0x180000 -vones
local fast memfast -dlocal -r#0x800000; memcmp -dlocal -mdglobal
-r#0x800000; memgallop -dlocal -r#0x180000 -v0; memgallop
-dlocal -r#0x180000 -vones
global function memfast -dglobal -r#0x800000; memdata -dglobal -r#0x200000
-v0; memdata -dglobal -r#0x200000 -vones; memaddr -dglobal
-r#0x200000 -v0; memaddr -dglobal -r#0x200000 -vones;
memcmp -dglobal -r#0x200000; memcmp -dglobal; memgallop
-dglobal -r#0x200000 -v0; memgallop -dglobal -r#0x200000
-vones; memfast -dglobal -r#*
local function memfast -dlocal -r#0x800000; memdata -dlocal -r#0x200000
-v0; memdata -dlocal -r#0x200000 -vones; memaddr -dlocal
-r#0x200000 -v0; memaddr -dlocal -r#0x200000 -vones;
memcmp -dlocal -r#0x200000; memcmp -dlocal -mdglobal;
memgallop -dlocal -r#0x200000 -v0; memgallop -dlocal
-r#0x200000 -vones; memfast -dlocal -r#*

cpu 00 ->

```

Figure 12-memory-1: Example of memory command displaying all commands from the script



memory (memory)

cpu30x command

memory (memory)

```
cpu 00 -> memory -dglobal -tfast -p
Test script for memory
Device Function Command script
global fast memfast -dglobal -r#0x800000; memcmp -dglobal -r#0x800000;
memgallop -dglobal -r#0x180000 -v0; memgallop -dglobal
-r#0x180000 -vones
cpu 00 ->
```

Figure 12-memory-2: Example of memory command selecting a few commands from the script

## NOTICES

The command cannot be used while running in *fatal error mode*.

## REFERENCES

See also commands:

*memfast*, *memaddr*, *memdata*, *memgallop*, *memcmp*.

**memory (memory)**

**cpu30x command**

**memory (memory)**

*This page is intentionally left blank*

**NAME**

*memtas* – Memory Test-And-Set test

**SYNOPSIS**

*memtas* [-dDEVICE] [-rADDR\_RANGE] [-cCACHE] [-uUNIT\_NO] [-aRANDOM\_ACC] [-nNO\_OF\_TAS] [-vVALUE]

[-dDEVICE] [-d[global | local]]

[-rADDR\_RANGE] [-r[START | #SIZE | START-END | START#SIZE]]

[-cCACHE] [-c[on | off | shared | update]]

[-uUNIT\_NO] [-u[all | own | HEX]]

[-aRANDOM\_ACC] [-a[on | off]]

[-nNO\_OF\_TAS] [-nDIGIT]

[-vVALUE] [-vDIGIT]

**DEFAULTS**

*memtas* -dglobal -r#0x100000 -con -uall -aon -n16 -vX

**DESCRIPTION**

The *memtas* command is used to test the *cache coherence* between the CPU units in an Supermax Enterprise Server. The command requests other units to execute a *remote command* that will perform TAS (Test And Set) test in a memory, at the same time as the *memtas* command performs the same test. This is very useful to diagnose an error on the *cache snooper* located on the *cpu30x* module.

The TAS is a feature which makes it possible to lock the memory between CPU units. The TAS access is made by means of a special CPU read and write instruction. The access will be performed as a *word access* with the cache algorithm by means of the CACHE option.

The bits in the *TAS* value have the following meaning:

- Bit 0-19 are used for a data pattern, specified with the **VALUE** option.
- Bit 20-27 are used to save the unit that has locked the memory.
- Bit 28-30 are not used. They are always zero.
- Bit 31 is used as a lock bit. If it is one, a unit has locked the memory, and the *TAS* value is a locked *TAS* value. If it is zero, no unit has locked the memory, and the *TAS* value is an unlocked *TAS* value.

If the size of the address range is bigger than 0x100000 or the **RANDOM\_ACC** option is set to **on**, the size of the *TAS* value will be 64 bits instead of 32 bits. This means that the first 32 bits will be used for the *TAS* access and the second 32 bits will be used to perform a read/write random access or a write/read cache flush access.

The test will be performed as follows:

- The memory array is initialized with a data pattern specified with the **VALUE** option.
- Then the command requests another unit (or more units) to perform a *remote command*. The *slave remote units* can be specified with the **UNIT\_NO** option.
- When all the *slave remote units* have made a response to the *master remote unit*, they begin testing.
- Now the units begin performing a number of *TAS* accesses to the same address using the locked and the unlocked *TAS* value alternately. All this is repeated for all addresses in the memory space.

It is possible to specify that a random read/write access shall be performed between the *TAS* accesses. This is done by means of the **RANDOM\_ACC** option. The read access will be performed on a random address in the second 32 bits of the *TAS* value, before the command performs the *TAS* access on a memory address. For every read access the memory address is calculated on a pseudo random basis. The write access will be performed on the same address as the

read address access after the command has unreserved a *TAS* value.

If the size of the address range is bigger than 0x100000, a write/read cache flush access will be performed automatically. This is done by performing a write and read access in the second 32 bits of the *TAS* value on the *TAS* address plus 0x100000 (*Secondary cache size*). If the calculated address is outside the memory address range, the address is set to the start address plus 4 in the address range. The write access will be performed before the *TAS* access, and the read access will be performed after the command has reserved the *TAS* value.

When a *slave remote unit* performs the test, it will execute the *memexer* command. I.e. a *slave remote unit* will always log the test in its own *test log* buffer. If the unit finds errors during the test, it will also log them in its own *error log* buffer.

The *memtas* command has the following options:

**-dDEVICE**

Same description as the *cachetest* command.

**-rADDR\_RANGE**

Same description as the *cachetest* command.

**-cCACHE**

This option is used to select which type of cache algorithm the *CPU* shall use while writing and reading in the memory. The option must be specified as a *word symbol option*, and the following cache algorithms can be specified:

- on**            The *CPU cache* is used. The *CPU* will perform the access with the *coherent exclusive on write* cache algorithm.
- off**            This cache mode cannot be used.
- shared**        The *CPU cache* is used. The *CPU* will perform the access with the *coherent exclusive* cache algorithm.
- update**        The *CPU cache* is used. The *CPU* will perform the access with the *coherent update on write* cache algorithm. This cache mode is not supported on the *cpu30x* module at the moment.

memtas (memory)

cpu30x command

memtas (memory)

**-uUNIT\_NO**

Same description as the *cachetest* command.

**-aRANDOM\_ACC**

This option is used to specify the random access mode. The option must be specified as a *word symbol option*, and the following two modes can be specified:

**on**     A read/write random access will be performed in the memory address range.

**off**     No random access will be performed.

**-nNO\_OF\_TAS**

This option is used to specify how many times a unit shall perform a *TAS* access on the same address. The number must be specified as a *digit value option*.

**-vVALUE**

This option is used to specify a 20 bit data pattern. The option must be specified as a *digit value option*. Default, the data pattern is a value from the internal system clock.

**EXAMPLES**

```
cpu 00 -> memtas -u01 -cshared
Test started at Monday 1995-03-20 - 12:42:56
Memory Test-And-Set test in Global memory
Start address End address Cache Unit Random Numbers Value
0x00a00000 0x00b00000 shared 01 on 4 0x003d7850
```

Figure 12-memtas-1: Example of the memtas command

**NOTICES**

The command cannot be used while running in *fatal error mode*.

**DIAGNOSTICS**

If the test finds errors, then try to diagnose them by means of the *cachetest* command.

## COMMAND ERRORS

If the command returns the error code *Options error*, it is because the memory range is not aligned on a 4 bytes address or because the selected *remote unit (CPU)* is not installed or because the **CACHE** option is set to **off**.

## BUGS

When the command performs the test on other *CPU* units (this is done by default, if the *remote parameter* is set to **on**) please note the following:

- The command will perform a *remote command* to the other *CPU* units. These units (*slave remote units*) must make a response to the *remote command*. This is only possible, if the units execute commands from the *command line* buffer. If the *master remote unit* has not got response from all *slave remote units* after 5 minutes, the unit will fail with time-out.
- If you cancel a running command on a *master remote unit* with **Ctrl-C**, the unit will be hung up until all *slave remote unit* commands are complete or cancelled.
- If a test fails when executing a *remote command*, and the unit waits to display the error message, the test must be cancelled before all other *master/slave remote units* can continue.

## REFERENCES

See also commands:  
*cachetest*, *memexer*, *set*.

**memtas (memory)****cpu30x command****memtas (memory)**

*This page is intentionally left blank*



**NAME**

*menu* - Display all commands

**SYNOPSIS**

*menu* [-ILONG\_FORMAT]  
 [-ILONG\_FORMAT] [-I]

**DEFAULTS**

*menu*

**DESCRIPTION**

The *menu* command is used to display all commands on the *cpu30x* module. The command only has one option as shown below:

- l If this option is specified, all commands will be displayed in long format, i.e. the display command names followed by a command text.

If no options are given, the command displays all commands on the unit in short format, i.e. only the command names.

**EXAMPLES**

```

cpu 00 -> menu
date      debug      errlog      help        menu        remote
repeat    set          status      testlog     time        version
config    sysbus      test        memory      memfast     mencmp
memaddr   memdata     mengallop  mentas     memexer     mendisp
cache     cachediag   cachetest  cachestat
cpu 00 ->
    
```

Figure 12-menu-1: Commands on the *cpu30x* module in short format

```
cpu 00 -> menu -1
date      Read or set date
debug     Call debugger
errlog    Display errors from the error log
help      Display command information and options
menu      Display all commands
remote    Response on remote command
repeat    Repeat commands
set       Set or display parameters
status    Display test status
testlog   Display test information from the test log
time      Time a command
version   Display program version
config    System and module configuration
sysbus    System Bus exerciser
test      Test script for all tests
memory    Test script for memory
memfast   Memory high speed test
memcmp    Memory move and compare test
memaddr   Memory address bit test
memdata   Memory data bit test
memgallop Memory galloping test
mentas    Memory Test-And-Set test
memexer   Memory exerciser
memdisp   Display memory contents
cache     Test script for cache
cachediag Cache diagnose test in tag/data field
cachetest Cache coherence test
cachestat Display cache tag status
cpu 00 ->
```

Figure 12-menu-2: Commands on the cpu30x module in long format

## NOTICES

The command does not clear *test log* and *error log*.

The command cannot be repeated. It will only run once.

## REFERENCES

See also commands:

*help*

remote (system)

cpu30x command

remote (system)

**NAME**

*remote* – Response on remote command

**SYNOPSIS**

*remote*

**DESCRIPTION**

The *remote* command waits until a *remote command* is requested from a *master remote unit* which is running the *memtas* or the *cachetest* command (*cache coherence test*). Then the remote unit automatically executes the *remote command* which is a *memexer* command.

The *remote* command is very useful when the unit only shall execute *remote commands* and not other *test commands*.

**EXAMPLES**

```
cpu 00 -> remote ; repeat
```

Figure 12-remote-1: Example of the remote command

**NOTICES**

The command cannot be used while running in *fatal error mode*.

If the *remote parameter (set command)* is set to **on**, the response to the first *remote command* will be made by the *command interpreter* instead of the *remote command*.

**REFERENCES**

See also commands:  
*cachetest, memexer, memtas, set*

remote (system)

cpu30x command

remote (system)

*This page is intentionally left blank*

**NAME**

*repeat* – Repeat commands

**SYNOPSIS**

*repeat* [-mMODE] [NUMBER]

[-mMODE] [-m[on | pass | time | cmd | test | error | off]]

[NUMBER] [DIGIT]

**DEFAULTS**

*repeat* -mon 1

**DESCRIPTION**

The *repeat* command is used to select how many times *test commands* should be executed from the *command line* buffer.

The *repeat* command has the following options:

**-mMODE**

This option is used to specify a repeat mode. It must be specified as a *word symbol option*. One of the following modes can be specified:

- |             |  |
|-------------|--|
| <b>off</b>  | This mode does not repeat commands.  |
| <b>on</b>   | This mode repeats commands forever.  |
| <b>pass</b> | This mode repeats commands until the number of <i>command line</i> buffer passes has reached the value of the <b>NUMBER</b> option.  |
| <b>time</b> | This mode repeats commands until the running time measured in seconds has passed the value of the <b>NUMBER</b> option.              |
| <b>cmd</b>  | This mode repeats commands until the number of commands has passed the value of the <b>NUMBER</b> option.                            |
| <b>test</b> | This mode repeats commands until the number of tests in the <i>test log</i> buffer has passed the value of the <b>NUMBER</b> option. |

**repeat (system)****cpu30x command****repeat (system)**

**error** This mode repeats commands until the number of errors in the *error log* buffer has passed the value of the **NUMBER** option.

### NUMBER

This option is used to specify how many times to repeat the commands. It must be specified as a *digit value option*. If the **MODE** option is set to **off** or **on**, this option has no effect.

If no options are given, the *repeat* command repeats all the *test commands* from the *command line* buffer forever.

### EXAMPLES

```
cpu 00 -> memfast; memcmp; repeat
cpu 00 -> memfast; memcmp; repeat -mtime 60
```

Figure 12-repeat-1: Example of using the repeat command

In the first example all the *test commands* will be repeated forever, and in the second example all the *test commands* will be repeated for 60 seconds.

### NOTICES

The command does not clear *test log* and *error log*.

The command cannot be executed while running in *fatal error mode*.

If the *repeat* command is not selected in the *command line* buffer, the *repmode parameter* and the *repro parameter* from the *set* command are used instead.

### BUGS

If the **MODE** option is set to **time**, the commands will maybe run longer time than specified with the **NUMBER** option. This is because the program does not check the time when performing a test.

### REFERENCES

See also commands:

*set, status*

repeat (system)

cpu30x command

repeat (system)

*This page is intentionally left blank*



**NAME**

*set* – Set or display parameters

**SYNOPSIS**

*set* [PARAMETER]

[PARAMETER]	[PARAMETER[=VALUE]]
<b>fault</b>	[off   poll   on]
<b>cache</b>	[off   code   data   code/data   test   code/test   data/test   on]
<b>cache_ecc</b>	[on   off]
<b>remote</b>	[on   off]
<b>dispmode</b>	[error   status   test   errstop   testshort   all   off]
<b>repmode</b>	[on   pass   time   cmd   test   error   off]
<b>repno</b>	[DIGIT]

**DESCRIPTION**

The *set* command is used to show or change certain parameters which are used to set the units in different modes or to set default values for certain commands. It is possible to assign different values to the different units. But only one parameter can be changed at a time. If no options are given, the command displays the current values for all parameters.

The following is a description of all parameters:

**fault**

The *fault parameter* is used to disable/enable error checking from the *error status* and on the *global/local memory* modules.

*Error status* checking includes errors from the *CPU Agent* located on the *global or local bus (bad data identifier, missing target acknowledgement and parity errors)*. These errors can be checked by reading a hardware error status register. If an error occurs an interrupt to the *CPU* can be generated.

*Global/local memory* modules checking includes errors from the parity and ECC (Error Check Code) circuitry on the memory modules. These errors can only be checked by reading a hardware status register on the memory modules.

The *fault* parameter must be specified as a *word symbol option* and the following fault modes are shown below:

- off** In this mode no fault check will be done on *status interrupt* and *global/local memory* modules.
- poll** In this mode fault will be checked on *status interrupt* by reading the register (no interrupt will be generated to the CPU), and fault will also be checked on *global/local memory* modules.
- on** In this mode fault will be checked on *status interrupt* by generating an interrupt to the CPU, and fault will also be checked on *global/local memory* modules.

**NOTE**

If an error occurs from the *error interrupt*, the program goes straight to the *fatal error mode*. An exception error from the CPU or wrong or illegal interrupts to the CPU also invoke the *fatal error mode*.

**cache**

The *cache* parameter is used to disable or enable the CPU cache in different cache modes. It must be specified as a *word symbol option*. The cache modes are shown below:

- off** In this mode the CPU cache is not used.
- code** In this mode the CPU cache is only used when it executes an instruction fetch. The *coherent exclusive on write* cache algorithm is used to do that.
- data** In this mode the CPU cache is only used when it executes a write/read access used by the Diagnostic program (e.g stack). The *coherent exclusive* cache algorithm is used to do that.

set (system)

cpu30x command

set (system)

- code/data** In this mode the *CPU cache* is used when it executes an instruction fetch or executes a write/read access used by the Diagnostic program.
- test** In this mode the *CPU cache* is only used when it executes a write/read access to a memory from a test. The *coherent exclusive* cache algorithm is used to do that.
- code/test** In this mode the *CPU cache* is used when it executes an instruction fetch or write/read access to a memory from a test.
- data/test** In this mode the *CPU cache* is used when it executes a write/read access.
- on** In this mode the *CPU cache* is used both for instruction fetch and write/read access.

The *CPU cache* mode which controls the write/read access to a memory from a test can be overruled by certain commands.

#### cache\_ecc

The *cache\_ecc* parameter is used to disable or enable the *CPU cache exception*. The *CPU cache exception* occurs when an ECC error is detected in the primary or secondary cache or when a parity error is detected on the *CPU* interface to the *CPU agents*. The parameter must be specified as a *word symbol option* and the following two modes can be set:

- on** In this mode the *CPU cache exception* is enabled in the *CPU*.
- off** In this mode the *CPU cache exception* is disabled in the *CPU*.

**NOTE**

If an ECC error is detected in the cache in this mode, the program can indicate various other errors or even break down.

**remote**

The *remote parameter* is used to specify whether the unit shall answer a *remote command* from another unit, before the unit executes a *test command* from the *command interpreter* buffer. The parameter must be specified as a *word symbol option*, and the following two modes can be set:

- on** In this mode, the unit will answer a *remote command* from another unit.
- off** In this mode, the unit will not respond to a *remote command* from another unit. Response to a *remote command* can only be made, if the *remote*, *mementas* or *cachetest* command also performs a *remote command* to another unit.

**dispmode**

The *dispmode parameter* is used to set the type of information to be displayed when the unit is testing. Please refer to "Chapter 2-3 Display Modes" to see the detailed description of the different display modes. The following display modes can be set, and the mode must be specified as a *word symbol option*:

- error** When a test starts the *unit status display mode* is displayed. If an error occurs the *error display mode* will be displayed instead.
- status** Only the *unit status display mode* is displayed in this mode.
- test** When a test starts the *test display mode* is displayed. If an error occurs the *unit status display mode* will be displayed instead.
- errstop** When a test starts the *test display mode* is displayed. If an error occurs the *error display mode* will be displayed, and afterwards the program will wait for a continue command from the terminal.

- testshort** When a test starts the *short test display mode* is displayed. If an error occurs the *error display mode* will be displayed.
- all** When a test starts the *test display mode* is displayed. If an error occurs the *error display mode* will be displayed.
- off** In this mode nothing will be displayed when running tests.

### repmode

The *repmode parameter* is used to set the repeat mode when you are not using the *repeat* command. The following repeat modes can be set, and they must be specified as a *word symbol option*:

- off** This mode does not repeat commands.
- on** This mode repeats commands forever.
- pass** This mode repeats commands until the number of *command line* buffer passes has reached the value of the *repno parameter*.
- time** This mode repeats commands until the running time measured in seconds has passed the value of the *repno parameter*.
- cmd** This mode repeats commands until the number of commands has passed the value of the *repno parameter*.
- test** This mode repeats commands until the number of tests in the *test log* buffer has passed the value of the *repno parameter*.
- error** This mode repeats commands until the number of errors in the *error log* buffer has passed the value of the *repno parameter*.

The *repno parameter* can be overruled by the *repeat* command.

set (system)

cpu30x command

set (system)

## repno

The *repno* parameter is used to specify how many times to repeat the commands. It must be specified as a *digit value option*. If the *repmode* parameter option is set to **off** or **on**, the *repno* parameter has no effect. The parameter can be overruled by the *repeat* command.

## EXAMPLES

```
cpu 00 -> set
fault-on
cache-on
cache_ecc-on
remote-on
dispmode=error
repmode=off
repno=0
cpu 00 ->
```

Figure 12-set-1: Example of using the set command to display current parameter values

```
cpu 00 -> set cache_ecc-on; set cache-on
cpu 00 ->
```

Figure 12-set-2: Example of using the set command to change a parameter

## NOTICES

The command does not clear *test log* and *error log*.

The command cannot be repeated. It will only run once.

## REFERENCES

See also commands:

*cachediag*, *cachetest*, *memfast*, *mementas*, *repeat*

**NAME**

*status* – Display test status

**SYNOPSIS**

*status*

**DESCRIPTION**

The *status* command is used to display the status after executing one or more *test commands*.

The *status* command shows the following status information:

- When the test is started and for how long time it has been executed.
- Number of commands.
- Number of tests.
- Number of errors.
- Number of passes.

The status information is cleared every time the *command line* buffer contains a *test command* and the *command interpreter* starts executing the first command from the *command line* buffer. When the commands are complete or cancelled, the status information will be saved.

## EXAMPLES

```
cpu 00 -> memfast -r0x800000-0x1000000 ; memcmp -r#0x400000
cpu 00 -> status
Status for test started at..: Monday 1995-03-20 - 10:42:31
Executed test time.....: 00:00:07
Number of commands.....: 2
Number of tests.....: 2
Number of errors.....: 0
Number of passes.....: 1
cpu 00 ->
```

Figure 12-status-1: Example of the status command

## NOTICES

The command does not clear *test log* and *error log*.

The command cannot be repeated. It will only run once.

## REFERENCES

See also commands:

*testlog*, *errlog*



## NAME

*sysbus* – System bus exerciser

## SYNOPSIS

*sysbus* [-caCA\_CMD] [-csCS\_CMD] [-lLOCAL\_BUS] [-pPOSITION] [-sSUBPOS] [-oOFFSET\_ADD] [-vVALUE]

[-caCA_CMD]	[-ca[stat   cntl   intrclr   debug   cntwr   cntrd   write   read]]
[-csCS_CMD]	[-cs[cntl   stat   statclr   id   fcn   debug   intr   buserr   berrclr   cpuconf   mem-lowbase   memupbase   memsize   mem-synb   count   write   read]]
[-lLOCAL_BUS]	[-l]
[-pPOSITION]	[-pDIGIT]
[-sSUBPOS]	[-sDIGIT]
[-oOFFSET_ADD]	[-oDIGIT]
[-vVALUE]	[-vDIGIT]

## DEFAULTS

*sysbus* -castat -pOWN -sOWN -o0 -v0

## DESCRIPTION

The *sysbus* command is used to exercise the *CPU agent* or *control space* on the *global/local bus*. This is very useful to diagnose fault on a module which fails with errors on the *global bus*, when executing the *boot* command.

The command performs a write or read access to a register on the *control space* when the *CS\_CMD* option is specified, otherwise the *CPU agent* on the module is accessed.

If the *control space* is accessed, it is possible to specify which module on the *global/local bus* to select. This is specified by means of the *LOCAL\_BUS*, *POSITION* and *SUBPOS* options.

The **CA\_CMD** or **CS\_CMD** option specifies the register to be accessed. If the register is a read register, the register information will be displayed on the *console terminal*. If it is a write register the value to be written must be specified by the **VALUE** option.

The command will always display information about its currently selected options.

The *sysbus* command has the following options:

**-caCA\_CMD**

This option is used to select which register on the *CPU agent* to be displayed or changed. The option will be overruled by the **CS\_CMD** option. The option must be specified as a *word symbol option*, and the following registers can be specified:

- |                |   |
|----------------|---|
| <b>stat</b>    | Read the status from the <i>CPU agent status register</i> .   |
| <b>cntl</b>    | Write a value to the <i>CPU agent control register</i> .  |
| <b>intrclr</b> | Clear <i>CPU</i> interrupts from the <i>control space interrupt register</i> .  |
| <b>debug</b>   | Read the value from the <i>CPU agent debug register</i> .   |
| <b>cntwr</b>   | Write a value to the <i>CPU agent count register</i> .  |
| <b>cntrd</b>   | Read the value from the <i>CPU agent count register</i> .   |
| <b>write</b>   | Write a 32 bit value to a <i>CPU agent</i> register. The <b>OFFSET_ADD</b> option specifies the offset address for the register.  |
| <b>read</b>    | Read a 32 bit value from a <i>CPU agent</i> register. The <b>OFFSET_ADD</b> option specifies the offset address for the register. |

**-csCS\_CMD**

This option is used to select which register on the *control space* to be displayed or changed. The option must be specified as a *word symbol option*, and the following registers can be specified:

- |                   |  |
|-------------------|--|
| <b>cntl</b>       | Write a value to a <i>control space control register</i> .   |
| <b>stat</b>       | Read the status from a <i>control space status register</i> .  |
| <b>statclr</b>    | Clear information on a <i>control space status register</i> .  |
| <b>id</b>         | Read the values from a <i>control space module id</i> .  |
| <b>fcn</b>        | Read the values from a <i>control space module fcn</i> .   |
| <b>debug</b>      | Write a value to a <i>control space debug register</i> .   |
| <b>intr</b>       | Write an interrupt to a <i>control space interrupt register</i> .  |
| <b>buserr</b>     | Read the status from a <i>control space bus error register</i> .   |
| <b>berrclr</b>    | Clear information on a <i>control space bus error register</i> .   |
| <b>cpuconf</b>    | Read the configuration from a <i>control space processor module register</i> .                                     |
| <b>memlowbase</b> | Write start address (Mbytes) for the low memory partition to a <i>control space lower base address register</i> .  |
| <b>memupbase</b>  | Write start address (Mbytes) for the high memory partition to a <i>control space lower base address register</i> . |
| <b>memsize</b>    | Read the memory size from a <i>control space size register</i> .   |
| <b>memsynb</b>    | Read the ECC status from a <i>control space syndrome bits register</i> .   |
| <b>count</b>      | Read the value from a <i>control space 1 MHz count register</i> .  |

sysbus (system)

cpu30x command

sysbus (system)

- write** Write a 32 bit value to a *control space* register. The **OFFSET\_ADD** option specifies the offset address to the register.
- read** Read a 32 bit value from a *control space* register. The **OFFSET\_ADD** option specifies the offset address to the register.

**[-ILOCAL\_BUS]**

If this option is specified, the *local bus* is accessed instead of the *global bus*. This option is only valid, if the *cpu30x* module has *CPU agents* mounted on the *local bus*.

**-pPOSITION**

This option is used to specify the module position when accessing the *control space* on the *global/local bus*. The option has only effect if the **CS\_CMD** option is specified. The **POSITION** option must be specified as a *digit value option*.

**-sSUBPOS**

This option is used to specify the module subposition when accessing the *control space* on the *global/local bus*. The option only has effect if the **CS\_CMD** option is specified. The **SUBPOS** option must be specified as a *digit value option*.

**-oOFFSET\_ADD**

This option is used to specify an offset address to a register on the *CPU agent* or the *control space*. This is necessary if you want to read from or write to a register which is not defined by the **CA\_CMD** or **CS\_CMD** option. The option must be specified as a *digit value option*, and the offset address will always be aligned to 4 bytes by cutting the value. The option only has effect if the **CA\_CMD** or **CS\_CMD** option is set to **write** or **read**.

**-vVALUE**

This option is used to specify a 32 bit value to be written in the selected register. The value must be specified as a *digit value option*, and it is only used if the selected register is a write register.

Without options the command displays the *global CPU agent status register*.

## EXAMPLES

```
cpu 00 -> sysbus -csid -p7 -s0
Read from id register on global Control Space position 0 subpos 0
Module name: BAI0301-1.....
Module type: 0x00000001
Serial no: 0x0000000e
cpu 00 ->
```

Figure 12-sysbus-1: Example of using sysbus to read global control space id register

## NOTICES

The command does not clear *test log* and *error log*.

The command always displays the bus information, no matter how the *dispmode parameter* is set.

The command always checks the *Error status* except for the parity circuitry on the *cpu30x* board, no matter how the *fault parameter* is set.

## WARNINGS

If using the command to write to a register on the *CPU agent* or *control space*, you must be sure that the correct value is written. Otherwise, the Diagnostic Programs can indicate incorrect data on a unit or even break down.

## COMMAND ERRORS

The command will fail with *missing target acknowledge error*, if trying to access a module on a position which is not installed in the Supermax Enterprise Server.

## BUGS

The command will not save the exerciser information in the *test log* buffer. The information will only be displayed on the *console terminal*, no matter how the *dispmode parameter* is set.

**sysbus (system)**

**cpu30x command**

**sysbus (system)**

## **REFERENCES**

See also commands:

*config*.

**NAME**

*test* – Test script for all tests

**SYNOPSIS**

*test* [-tTEST\_FUNCTIONS] [-pDISP\_CMDS]  
 [-tTEST\_FUNCTIONS] [-t[memory | cache] [\*,\*,]]  
 [-pDISP\_CMDS] [-p]

**DEFAULTS**

*test* -tmemory,cache

**DESCRIPTION**

This command is a *script command*, and it is used to run a complete test of the *pmd30x* submodule and the *cpu30x* module including the memories which are connected to the *cpu30x* module. The command executes other *script commands* which are all tests of the memories, cache. It is possible to specify which test groups of the *test script* to be executed. This is done by means of the **TEST\_FUNCTIONS** option.

The *test* command has the following options:

**-tTEST\_FUNCTIONS**

This option is used to specify which test groups should be selected from the *test script*. The option must be specified as a *multi word symbol option*, and the following test groups can be specified:

- memory**      If this is selected, the *memory* commands will be executed.
- cache**        If this is selected, the *cache* commands will be executed.

**-pDISP\_CMDS**

If this option is specified, the selected commands from the script will only be displayed, not executed.

Without option the command executes all commands from the scripts.

## EXAMPLES

```
cpu 00 -> test -p
Test script for all tests
Function  Command script
memory   memory
cache    cache
cpu 00 ->
```

Figure 12-test-1: Example of the test command displaying all commands from the script

## NOTICES

The command cannot be used while running in *fatal error mode*.

## BUGS

When the command runs the *cache script command* with the *remote parameter* set to **on** (Please refer to the *set command*) please note the following:

- When the *test* command runs the *mentas* or *cachetest* command from the *cache script command*, the commands will perform a *remote command* to the other CPU units. These units (*slave remote units*) must make a response to the *remote command*. This is only possible, if the units execute commands from the *command line buffer*. If the *master remote unit* has not got response from all *slave remote units* after 5 minutes, the unit will fail with time-out.
- If you cancel a running command on a *master remote unit* with **Ctrl-C**, the unit will be hung up until all *slave remote unit* commands are complete or cancelled.
- If a test fails when executing a *remote command*, and the unit waits to display the error message, the test must be cancelled before all other *master/slave remote units* can continue.



**test (system)**

**cpu30x command**

**test (system)**

## REFERENCES

See also commands:

*cache, cachetest, memory, memtas, set.*

**test (system)****cpu30x command****test (system)**

*This page is intentionally left blank*

## NAME

*testlog* – Display test information from the test log

## SYNOPSIS

*testlog* [-nNUMBER] [-old]

[-nNUMBER] [-nDIGIT]

[-old] [-old]

## DESCRIPTION

The *testlog* command is used to display test information which is stored in the *test log* buffer. The *test log* buffer is cleared every time the *command line* buffer contains a *test command* and the *command interpreter* starts executing the first command from the *command line* buffer.

The maximum number of tests in the *test log* buffer is 64. If this number is exceeded the oldest log will be deleted and the new ones will be added to the buffer.

The *testlog* command has the following options:

### -nNUMBER

This option is used to display test information about a number of the latest tests which is stored in the *test log* buffer. The option must be specified as *digit value option*.

### -old

This option is used to display test information which is stored in the *test log* buffer, whether it has been displayed before or not.

If no options are given the command, it displays all test information from the *test log* buffer which has not been displayed before.

## EXAMPLES

In the example on the next page, the *testlog* command has been executed. It displays all test information from the *test log* buffer which has not been displayed before. If you execute the command again, it will not display any test information.

testlog (system)

cpu30x command

testlog (system)

```

cpu 00 -> testlog
Test started at Tuesday 1995-07-11 - 09:07:34
Cache diagnose test in tag/data field in Global memory
Cache  Cache address range  Function  Mode  Pattern  value
scache 0x00b00000-0x00c00000 tag      limit  random  0x3002cbf6

Test started at Tuesday 1995-07-11 - 09:07:34
Cache diagnose test in tag/data field in Global memory
Cache  Cache address range  Function  Mode  Pattern  value
scache 0x00b00000-0x00c00000 data     limit  random  0x3002cbf6

Number of tests: 2
cpu 00 -> testlog
Number of tests: 2

```

Figure 12-testlog-1: Example of using the testlog command

The example below shows how to display only the latest test information from the *test log* buffer.

```

cpu 00 -> testlog -old -n1
Test started at Tuesday 1995-07-11 - 09:07:34
Cache diagnose test in tag/data field in Global memory
Cache  Cache address range  Function  Mode  Pattern  value
scache 0x00b00000-0x00c00000 data     limit  random  0x3002cbf6

Number of tests: 2
cpu 00 ->

```

Figure 12-testlog-2: Example of using the testlog command to display the latest test

## NOTICES

The command does not clear *test log* and *error log*.

The command cannot be repeated. It will only run once.

The test information can also be displayed when a test is started. This depends on how the *dispmode parameter* is set.

**REFERENCES**

See also commands:

*set, status, errlog*

*This page is intentionally left blank*

time (test)

cpu30x command

time (test)

**NAME***time* - Time a command**SYNOPSIS***time* command**command**      **COMMAND [OPTIONS]****DESCRIPTION**

The *time* command is used to measure the time a command needs to complete execution.

The *time* command will execute a selected command that is specified as a *command name option*. When the selected command is complete, the *time* command will display the time elapsed during execution of the selected command. The elapsed time will be shown in ms and in seconds.

The options to the *time* command, must always be specified.

**EXAMPLES**

```
cpu 00 -> time memfast -r#0x400000
Command time: 0 second. 340 ms.
cpu 00 ->
```

Figure 12-time-1: Example of the time command

**NOTICES**

The command cannot be executed while running in *fatal error mode*.

**REFERENCES**

See also commands:  
*status*.

**time (test)**

**cpu30x command**

**time (test)**

*This page is intentionally left blank*



**NAME**

*version* – Display program version

**SYNOPSIS**

*version*

**DESCRIPTION**

The *version* command displays the version of the boot prom and the current version of the Diagnostic Programs which has been loaded to the unit.

**NOTICES**

The command does not clear *test log* and *error log*.

The command cannot be repeated. It will only run once.

**EXAMPLES**

```
cpu 00 -> version
Debugger PROM:   Version 1.01   Date 1995-04-12 - 08:55:08
Diagnostic program: Version 1.03   Date 1995-08-21 - 10:16:02
cpu 00 ->
```

Figure 12-version-1: Example of the version command

*This page is intentionally left blank*

**Release Note**  
**Package No. 40030401**  
**Dansk Data Elektronik A/S**

<b>Product name:</b>	SMES Diagnostic Programs
<b>Stock number:</b>	40030401
<b>Version:</b>	1.03
<b>Date:</b>	1995-08-29
<b>Completed by:</b>	OK/BD

Please read this release note when you receive the package.

## Package List

### First Installation:

**Floppy disk:**

40030401 1 3½" 1.44 Mb diskette. SMES Diagnostic Programs 1.03

**Documents:**

1 Release Note (this document)

94400401 1 SMES Diagnostic Programs User's Manual Version 1.0

### Installation Update:

**Floppy disk:**

40030401 1 3½" 1.44 Mb diskette. SMES Diagnostic Programs 1.03

**Documents:**

1 Release note (this document)

94400401 1 SMES Diagnostic Programs User's Manual Version 1.0

## Labeling

### Label on floppy disk:

SMES Diagnostic Programs
Version 1.03 1995-08-21
Stock Number 40030401
© Dansk Data Elektronik A/S 1(1)

## Product Features

The SMES Diagnostic Programs can test the Supermax Enterprise Server hardware modules.

## Requirements

### Disk Space:

None.

## Release Information

### Changes since latest release:

- This is the first release of the SMES Diagnostic Programs.

### Errors corrected since latest release:

- This is the first release of the SMES Diagnostic Programs.

### Recognized errors or inconveniences:

- If the Supermax Enterprise Server has more than 508 MB global memory and a cpu30x module is installed, the memory above 508 MB cannot be tested. This is because the cpu30x Diagnostic Program has a limit of 508 MB. The baio30x Diagnostic Program has a limit of 1792 MB global memory.

#### WORKAROUND:

To test the global memory up to 1792 MB execute the following command sequence on the master baio30x unit:

```
boot -creset; boot -cstart -tmem301; boot -cmeminit
```

Now the baio30x unit is ready to test the global memory (e.g. `memory -dglobal` or `memfast -dglobal -r*-*` commands).

- If you have changed the removable media from an SCSI device and start a new test of the SCSI device, the SCSI command will fail with "Unit attention sense key error".

#### WORKAROUND:

Read the configuration again from the SCSI device, before starting a new test. This is done by running the `"sdconf -dSCSI_DEVICE"` command on the baio30x unit.

- Be very careful when testing the global memory if the memory address range are not aligned to 128 bytes (the cache block size), e.g. `memfast -dglobal -r0x400028` or `memfast -dglobal -r#0x28`. This is because the cache coherence between the cpu30x module and the baio30x modules are not complete. If you did not align the memory address range, it is possible that the program will find incorrect data in the global memory.

- When running the `test`, `cache`, `memtas` or `cachetest` command on a cpu30x unit in a Supermax Enterprise Server with more than one CPU submodule on the cpu30x module, it is necessary to run some test commands on the other CPU units while running the command. This is because the command will request other CPU units to perform a remote command and the remote command can only be responded to when running test commands. If this is not done, the command will fail with "Remote time-out" after no response for 5 minutes.

If you cancel the command on the master remote unit with `CTRL-C`, the unit will be hung-up until all slave remote unit commands are completed or cancelled.

- When the SCSI disconnect/reconnect option is enabled on the baio30x unit, the program is not able to handle pending commands to each device on the SCSI bus (only one SCSI command will be executed on a SCSI bus).

- The internal clock on the unit does not run while the unit is running in the fatal error mode (fatal error is a mode which shows that an exception error from the CPU occurs or wrong, illegal or error interrupts to

**Release Note**  
**Package No. 40030401**  
**Dansk Data Elektronik A/S**

the CPU occurs).

- When running the SMES Diagnostic Programs, the two fan groups will always be set to maximum level.

## **Comments:**

**This Diagnostic Programs contains versions of the programs for:**

- The baio30x module. Version 1.03
- The cpu30x module. Version 1.03

## **First Installation**

Please read the User's Manual prior to product installation.  
Then follow the procedure described under 'General Description'.

## **Installation Update**

Please read the User's Manual prior to product installation.  
Then follow the procedure described under 'General Description'.

## **Installation Guide**

**SMES Diagnostic Programs version 1.03:**

- SMES Diagnostic Programs User's Manual, version 1.0 included.

## **Installation Update Guide**

**SMES Diagnostic Programs version 1.03:**

- SMES Diagnostic Programs User's Manual, version 1.0 included.

*- END OF DOCUMENT -*

