



YAMON™ User's Manual

Document Number: MD00008

Revision 2.01

January 31, 2001

**MIPS Technologies, Inc.
1225 Charleston Road
Mountain View, CA 94043-1353**

Copyright © 1999-2000 MIPS Technologies, Inc. All rights reserved.

Unpublished rights reserved under the Copyright Laws of the United States of America.

This document contains information that is proprietary to MIPS Technologies, Inc. (“MIPS Technologies”). Any copying, modifying or use of this information (in whole or in part) which is not expressly permitted in writing by MIPS Technologies or a contractually-authorized third party is strictly prohibited. At a minimum, this information is protected under unfair competition laws and the expression of the information contained herein is protected under federal copyright laws. Violations thereof may result in criminal penalties and fines.

MIPS Technologies or any contractually-authorized third party reserves the right to change the information contained in this document to improve function, design or otherwise. MIPS Technologies does not assume any liability arising out of the application or use of this information. Any license under patent rights or any other intellectual property rights owned by MIPS Technologies or third parties shall be conveyed by MIPS Technologies or any contractually-authorized third party in a separate license agreement between the parties.

The information contained in this document constitutes one or more of the following: commercial computer software, commercial computer software documentation or other commercial items. If the user of this information, or any related documentation of any kind, including related technical data or manuals, is an agency, department, or other entity of the United States government (“Government”), the use, duplication, reproduction, release, modification, disclosure, or transfer of this information, or any related documentation of any kind, is restricted in accordance with Federal Acquisition Regulation 12.212 for civilian agencies and Defense Federal Acquisition Regulation Supplement 227.7202 for military agencies. The use of this information by the Government is further restricted in accordance with the terms of the license agreement(s) and/or applicable contract terms and conditions covering this information from MIPS Technologies or any contractually-authorized third party.

MIPS, R3000, R4000, R5000, R8000 and R10000 are among the registered trademarks of MIPS Technologies, Inc., and R4300, R20K, MIPS16, MIPS32, MIPS64, MIPS-3D, MIPS I, MIPS II, MIPS III, MIPS IV, MIPS V, MDMX, SmartMIPS, 4K, 4Kc, 4Km, 4Kp, 5K, 5Kc, 20K, 20Kc, EC, MGB, SOC-it, SEAD, YAMON, ATLAS, JALGO, CoreLV and MIPS-based are among the trademarks of MIPS Technologies, Inc.

All other trademarks referred to herein are the property of their respective owners.

Table of Contents

1	Introduction	4
2	General Issues	5
2.1	Connecting to YAMON	5
2.2	Shell	5
2.2.1	Number Formats	7
2.2.2	YAMON Distribution	8
3	Shell Commands	10
4	Environment Variables	19
4.1	Ethernet Support	21
5	Special Issues	22
5.1	Address Validation	22
5.2	Exception Handling	22
5.3	Cache Issues	22
6	Memory Layout	24
7	Board Specifics	25
7.1	Atlas/Malta	25
7.1.1	Upgrading YAMON	25
7.2	SEAD	26
7.2.1	Upgrading YAMON	26
7.3	SEAD-2	26
7.3.1	Upgrading YAMON	26
8	Application Interface	27
8.1	Entry	27
8.2	Exit	28
8.3	Functions	28
8.4	Environment Variables	29
8.5	Sample Application	29
9	GDB Interface	31
9.1	Connecting to GDB	31
9.2	GDB Remote Protocol Specification	31
9.3	GDB Remote Protocol Requests	32
10	Motorola S-records	38
11	Flash Support	40
12	Diagnostics and Error Messages	42
Appendices		45
A	References	45
B	Revision History	45

1 Introduction

This document constitutes the user manual for the YAMON™ ROM monitor.

YAMON (“Yet Another MONitor”) is the ROM monitor used on MIPS Technologies evaluation and reference boards.

The target audience for this document is users of those boards. This would typically be engineers developing hardware or software including compilers, RTOS and other tools.

Currently, the following boards/CPUs are supported by YAMON :

- Atlas™ with MIPS32 4K™ or MIPS64 5K™ class of CPUs.
- Atlas with QED RM5261@.
- Malta™ with MIPS32 4K or MIPS64 5K class of CPUs.
- Malta with QED RM5261@.
- SEAD™ with MIPS32 4K or MIPS64 5K class of CPUs.
- SEAD-2™ with MIPS32 4K or MIPS64 5K class of CPUs.

SEAD boards are supported for “Basic RTL” revisions up to and including 01.03.

This document describes the interfaces to YAMON. This includes a command line interface, a debugger interface and an application interface.

The same binary image is used for all boards and CPUs supported by YAMON. YAMON detects the specific board/CPU at run-time.

The binary image contains both little- and big-endian code. YAMON detects the endianness at run-time and executes the appropriate code.

Some of the main YAMON features are:

- RAM size/type detection and auto configuration.
- Shell with command line history and editing.
- Traditional shell commands (load, go, dump etc.).
- PCI auto detection and auto configuration (PCI not available on all boards).
- Ethernet and serial port support (Ethernet not available on all boards).
- Configuration of CPU for CPUs supporting this.

YAMON supports the following interfaces :

- Command line interface through serial port.
- Debug interface through dedicated debug serial port. Interface conforms to GNU-GDB “Standard Remote Protocol” with extensions for SDE-GDB from Algorithmics.
- Vector table based call interface for use by applications.
- Ethernet for TFTP-load and “ping” support (Ethernet not available on all boards).

2 General Issues

This section describes how to connect your terminal to YAMON, how to use the shell and how to load your board with a new revision of YAMON.

2.1 Connecting to YAMON

YAMON supports 2 serial ports named tty0 and tty1. YAMON uses tty0 as the console port. [Section 7, "Board Specifics"](#) specifies the port mapping for specific boards.

By default, the serial port settings are :

38400 baud, 8 databits, no parity, 1 stopbit, hardware flowcontrol (RTS/CTS).

Some boards (e.g. Atlas) allow this default to be changed and stored in environment variables, so that they survive a reset. If you are not sure what the current serial port settings are, you may restore the above factory default settings by the mechanisms described in [Section 4, "Environment Variables"](#) and [Section 7, "Board Specifics"](#).

A standard "NULL-modem" cable with the wiring shown below should be used to connect the board to your terminal or PC.

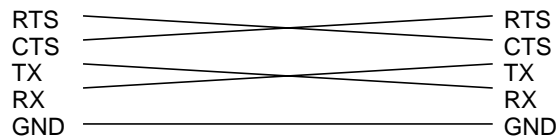


Figure 1 NULL-Modem Cable

2.2 Shell

After a reset, the YAMON shell starts by displaying the following sign-on message and then starts prompting the user for commands. The example shows YAMON running on an Atlas board with a MIPS 4Kc processor. The sign-on message slightly differs for other boards/CPUs. Also, the software compilation date/time may differ.

YAMON ROM Monitor, Revision 02.00.
Copyright (c) 1999-2000 MIPS Technologies, Inc. - All Rights Reserved.

For a list of available commands, type 'help'.

```

Compilation time =          Sep 11 2000  13:06:28
Board type/revision =       0x00 (Atlas) / 0x00
Core board type/revision =  0x01 (CoreLV) / 0x01
FPGA revision =            0x0007
MAC address =              00.d0.a0.00.00.32
Board S/N =                0000000050
PCI bus frequency =        33.33 MHz
Processor Company ID =     0x01 (MIPS Technologies, Inc.)
Processor ID/revision =    0x80 (MIPS 4Kc) / 0x01
Endianness =              Little
CPU/Bus frequency =       80 MHz / 40 MHz
Flash memory size =       36 MByte
SDRAM size =              32 MByte
First free SDRAM address = 0x80094150

```

YAMON>

The shell implements command line history and editing. Previous commands may be recalled by typing Ctrl-p or arrow-up.

The shell accepts the control codes shown in [Table 1](#).

VT-100 control sequences are used for the arrows ("ESC[A", "ESC[B", "ESC[C", "ESC[D").

Table 1 Command Line Recall/editing Commands

Name	Description
Ctrl-p / arrow-up	Recall previous command in command stack (do not perform it).
Ctrl-n / arrow-down	Recall next command in command stack (do not perform it).
Ctrl-a	Move to first character.
Ctrl-e	Move to last character.
Ctrl-b / arrow-left	Move one character left.
Ctrl-f / arrow-right	Move one character right.
Ctrl-d	Delete character at cursor position.
Ctrl-h / DEL	Delete character to the left of cursor position.
Ctrl-k	Delete characters from cursor position to end of line.
Ctrl-u	Delete line.
Ctrl-c	Cancel current line.
TAB	Command line completion.

Commands may be auto completed by pressing TAB. Also, the shell attempts to auto complete commands when parsing them. However, minimum 2 characters must be typed before auto completing. For example, if the user enters "he", the command "help" will be performed.

The special characters shown in [Table 2](#) are recognized by the shell parser.

Table 2 Shell Parser Special Characters

Character	Meaning
\$ (dollar)	The following is an environment variable, which will be expanded by the parser.
“ (double quote)	The following is a string, which should not be parsed as individual tokens (must be terminated by another double quote). Environment variables within the quotes are expanded.
‘ (single quote)	The following is a string, which should not be parsed as individual tokens (must be terminated by another single quote). Environment variables within the quotes are not expanded.
;	Used to separate commands. Example : YAMON> echo hello; echo world hello world YAMON>
+	A line beginning with : +<count>; will be repeated ‘count’ times or until a command fails. Example : YAMON> +3; echo hello hello hello hello YAMON>
\ (backslash)	The following character should not be considered a special character. The sequence \\ will generate a \ character. If the backslash is the last character on a line, the command will continue on the next line. Example : YAMON> echo “hello \ ? world” hello world YAMON>

Most commands may be stopped by typing Ctrl-C. An RS232 break on tty0 is equivalent to either Ctrl-C or reset depending on the board (see [Section 7, "Board Specifics"](#)).

Commands are case-sensitive.

2.2.1 Number Formats

The following number formats are supported (case insensitive) :

- <number> : Hex
- 16/<number> : Hex
- 0x <number> : Hex
- 0X <number> : Hex

- 10/<number> : Decimal
- 8/<number> : Octal

By default, all numbers except numbers used for IP-addresses and IP-subnet masks are in hexadecimal format.

2.2.2 YAMON Distribution

The board shipped from MIPS Technologies will contain YAMON in non-volatile memory. Depending on the specific board, this may be flash memory or (E)PROM.

YAMON is released as three files :

- `yamon-src-<rev>.tar.gz`
- `yamon-bin-<rev>.zip`
- `yamon-sampleappl-src-<rev>.tar.gz`

<rev> is replaced with the specific revision number (e.g. 02.00).

`yamon-src-<rev>.tar.gz` contains the YAMON source file tree.

`yamon-bin-<rev>.zip` contains the binary distribution (see [Table 3](#)).

`yamon-sampleappl-src-<rev>.tar.gz` contains source code for a sample “Hello world” application that uses the YAMON application interface (see [Section 8, "Application Interface"](#)). The revision number <rev> for the sample application is not correlated to the revision number of YAMON itself. The sample application includes a README file describing the application. It is also described in [Section 8.5, "Sample Application"](#).

Table 3 YAMON Distribution Zip File Contents

Name	Description
RELEASE	ASCII text file containing release notes.
yamon-<rev>.fl	YAMON image in format required for parallel download.
deleteall.fl	File used for clearing the required sections of Monitor flash.
lock.fl	File used for locking the required sections of Monitor flash.
go.bat	DOS batch file used for copying deleteall.fl, yamon-<rev>.fl and lock.fl files to parallel port (lpt1).
go.pl	Perl script used for copying deleteall.fl, yamon-<rev>.fl and lock.fl files to parallel port on computers running Linux or Solaris.
yamon-<rev>.bin	YAMON image in binary format. Used for programming an (E)PROM device.
reset-<rev>.dis	Disassembly of the code located at the reset exception vector (0xbfc00000).
yamon-<rev>_el.dis	Disassembly of the little endian code.
yamon-<rev>_eb.dis	Disassembly of the big endian code.
reset-<rev>.map	Linker generated map file for the code located at the reset exception vector.
yamon-<rev>_el.map	Linker generated map file for the little endian code.
yamon-<rev>_eb.map	Linker generated map file for the big endian code.
yamon_api.h	Header file defining the application interface to YAMON (see Section 8, "Application Interface").

See [Section 7, "Board Specifics"](#) for board specific instructions on how to upgrade YAMON.

3 Shell Commands

The following shell commands are available (sorted in alphabetical order).

Not all commands are available for all boards/CPUs due to hardware dependencies.

The command “help” will display a list of the commands. “help <command>” will display help on a specific command.

NAME	.
SYNOPSIS	. (continuation command)
DESCRIPTION	This command is a continuation command for “dump”, so that more data can be dumped without the user having to repeat the previous “dump” with a new address.
OPTIONS	None

NAME	cache
SYNOPSIS	cache [on off]
DESCRIPTION	Enable/disable caches by setting K0 field of CPU CONFIG register. The command will flush D-cache before the K0 field is modified. If no parameter is supplied, the current cache setting is displayed.
OPTIONS	None

NAME	cksum
SYNOPSIS	cksum <address> <size>
DESCRIPTION	Computes 32-bit CRC for the specified memory area. The CRC algorithm and polynomial is the same which is being used by the Unix "cksum" command.
OPTIONS	None

NAME	compare
SYNOPSIS	compare <address1> <address2> <size>
DESCRIPTION	Compares the two specified memory areas. If a difference is encountered during the compare, the address of the first mis-match will be reported.
OPTIONS	None

NAME	copy
SYNOPSIS	copy [-f] <src> <dst> <size>
DESCRIPTION	The number of bytes specified by <size> are copied from <src> to <dst>. Both the source and destination can be located anywhere in the address space. The copy command knows the address areas for the flash memories in the system and is able to program them. If the destination is flash, the destination area must be cleared using the “erase” command prior to the copy operation. Note that the copy command prevents the user from overwriting the environment flash area. Unless the -f option is applied, caches are flushed before and after the copy operations (D-cache writeback and invalidate, I-cache invalidate).
OPTIONS	-f Do not flush caches

NAME	cp0
SYNOPSIS	cp0 [<name> [<value>]]
DESCRIPTION	<p>Read/write CP0 register(s).</p> <p>If no arguments are applied, all CP0 registers are shown. If the name of a CP0 register is applied, this register is written or read depending on whether a value is applied or not. Writing a CP0 register takes effect immediately and should be done with care since it may crash YAMON. Some of the CP0 registers are optional and not available for all CPUs.</p> <p>See Section 8.3, "Functions" for description of context shift handling.</p>
OPTIONS	None

NAME	date (Atlas/Malta only)
SYNOPSIS	date [[[cc]yy]mmddHHMM[.ss]] (HH in 24 hour format)
DESCRIPTION	<p>Set or read date and time.</p> <p>If no argument is given, the present date and time is output. If an argument is specified, it is used to set the date and time on the real time clock.</p> <p>The argument must have the format [[cc]yy]mmddHHMM[.ss] where the hours HH are represented in 24h format (00-23).</p> <p>The output is shown as Day Mon dd HH:MM:SS ccyy.</p>
OPTIONS	None

NAME	dis
SYNOPSIS	dis [-m] <address> [<count>]
DESCRIPTION	<p>Disassemble code starting at <address> (only MIPS32 instructions). <count> (default 16) specifies the number of instructions to disassemble.</p>
OPTIONS	-m Prompt user for keypress after each screen of data

NAME	dump
SYNOPSIS	dump [-m][-8 -16 -32] <address> [<size>]
DESCRIPTION	<p>Dumps data from address range starting at <address>.</p> <p>The default data display width is 8 bits. The <size> parameter specifies the number of bytes to dump (default is 256).</p> <p>The continuation command “.” works together with “dump”.</p>
OPTIONS	-m Prompt user for keypress after each screen of data -8 Dump data in units of bytes -16 Dump data in units of halfwords -32 Dump data in units of words

NAME	echo
SYNOPSIS	echo <data>
DESCRIPTION	<p>The echo command echoes all of its arguments to the console.</p> <p>This offers a convenient way to examine the contents of specific environment variables (e.g. “echo \$ipaddr”).</p>
OPTIONS	None

NAME	edit
SYNOPSIS	edit [-8 -16 -32] <address>
DESCRIPTION	<p>Edit memory contents starting at <address>. The default data width is 8 bits. Edit mode is exited by typing '?' or Ctrl-C.</p> <p>During the edit, a data element can be left untouched by just pushing Enter. The edit will then continue with the next data element.</p> <p>Typing '-' will cause the address to be decremented.</p> <p>Data is entered using hexadecimal number format (with no leading "0x").</p>
OPTIONS	<p>-8 Edit data in units of bytes</p> <p>-16 Edit data in units of halfwords</p> <p>-32 Edit data in units of words</p>

NAME	erase
SYNOPSIS	erase [-s -e]<address> <size>]
DESCRIPTION	<p>Erase flash memory.</p> <p>An option may be applied specifying which flash region to erase.</p> <p>If no such option is applied, the address range to be erased is specified by the <address> and <size> parameters.</p> <p>If no such range is specified either, the range corresponding to the default option is assumed (if there is a default option, this is platform specific).</p> <p>If (and only if) the -e option (erase environment flash) is applied, the system environment variables are reinitialized to factory default values.</p> <p>If a range is specified, all flash blocks touched by the range are cleared. The block size depends on the flash memory type used by the board. The blocks to be cleared are displayed, and the user is asked for confirmation before the operation is performed.</p> <p>Erasing a large flash area takes time. It can easily take several minutes to erase a 32 MByte area.</p> <p>Any set flash sector lock bits will be cleared before the sector is erased. If they cannot be cleared (e.g. due to hardware protection of the lock bits), the command will fail.</p>
OPTIONS	<p>-e Erase and reinitialize entire environment area.</p> <p>-s Erase entire system flash (default option). Only available on some platforms.</p>

NAME	fill
SYNOPSIS	fill [-8 -16 -32] <address> <size> <data>
DESCRIPTION	<p>Fills the specified memory area starting at <address>. Default data width is 8 bits. <size> specifies the size of the area to fill (expressed in bytes).</p>
OPTIONS	<p>-8 Fill data in units of bytes</p> <p>-16 Fill data in units of halfwords</p> <p>-32 Fill data in units of words</p>

NAME	flush
SYNOPSIS	flush [-i -d]
DESCRIPTION	<p>Flush cache(s). By default, the D-cache is flushed first, followed by an I-cache invalidate. This behavior can be changed by the "-i" and "-d" options.</p> <p>The D-cache flush operation is in reality a write-back of dirty lines (write-back caches only) followed by an invalidate operation.</p>
OPTIONS	<p>-i Invalidate I-cache</p> <p>-d Flush D-cache</p>

NAME	<code>gdb</code>
SYNOPSIS	<code>gdb [-v][-c] [. <args>]</code>
DESCRIPTION	<p>Setup connection to GDB debugger on port <code>tty1</code>. The Standard GDB remote protocol is used.</p> <p>If the user application is not currently running, and <code>Ctrl-C</code> is typed at the console port, YAMON will leave GDB mode and return to the command prompt.</p> <p><code><args></code> is broken up in substrings and passed to the application. The list of arguments to be passed must begin with a <code>'</code>. The <code>'</code> is not passed as an argument. The first argument (<code>argv[0]</code>) will be the string <code>'gdb'</code>.</p> <p>Section 8.1, "Entry" describes how arguments are passed to the application and how the application context is initially setup.</p> <p>The application may return to YAMON by jumping to the address specified in <code>ra</code> or by calling the <code>exit(rc)</code> function supplied by YAMON.</p> <p>The verbose (<code>-v</code>) option will cause the commands from the GDB host and the responses from YAMON to be displayed on the console port.</p> <p>The checksum off (<code>-c</code>) option will disable validation of the checksum used in GDB commands. This is useful in case the user wishes to enter commands manually. Two checksum characters should still be used in the commands, but the values are don't care.</p>
OPTIONS	<p><code>-v</code> Display messages from and to GDB host</p> <p><code>-c</code> Disable check of GDB checksum</p>

NAME	<code>go</code>
SYNOPSIS	<code>go [. <address> [<args>]]</code>
DESCRIPTION	<p>Execute application code. If a target address is not specified, the address obtained from the last successful <code>'load'</code> command (if any) is used as the target address. If arguments for the user program need to be specified, the default execution address can be referenced by a <code>'</code>.</p> <p>Section 8.1, "Entry" describes how arguments are passed to the application and how the application context is initially setup.</p> <p>The application may return to YAMON by jumping to the address specified in <code>ra</code> or by calling the <code>exit(rc)</code> function supplied by YAMON.</p>
OPTIONS	None

NAME	<code>help</code>
SYNOPSIS	<code>help [<command>]</code>
DESCRIPTION	<p><code>"help"</code> with no parameter shows a list of all the available commands.</p> <p>To get more detailed help on a particular command, specify the command name as an argument to <code>"help"</code>.</p> <p>When specifying a command as an argument to <code>help</code>, command completion will be performed if at least two characters of the command name have been specified.</p>
OPTIONS	None

NAME	info
SYNOPSIS	info [board cpu memory uart boot all pci isa lan]
DESCRIPTION	<p>Display information on the requested item (default boot).</p> <p>The following information displays can be requested:</p> <pre>board : Board properties cpu : CPU properties memory : Memory properties uart : Serial ports statistics boot : Info displayed after reset all : All info pci (Atlas/Malta only) : PCI auto discovery/auto configuration isa (Malta only) : ISA bus configuration lan (Atlas/Malta only) : Ethernet statistics</pre>
OPTIONS	None

NAME	load
SYNOPSIS	<pre>load [-r] ([tftp:][//<ipaddr>][/<filename>]) ([asc:] [/(tty0 tty1)])</pre>
DESCRIPTION	<p>Load image from serial port or Ethernet to RAM or flash (depending on address). The only image type currently supported is SREC.</p> <p>On platforms supporting both Ethernet and serial port, the default protocol is taken from the environment variable 'bootprot'. On platforms without Ethernet, the only (and default) protocol is 'asc'.</p> <p>If loading from serial port, the default port is taken from the environment variable 'bootserport'.</p> <p>If loading from Ethernet, the IP address of the TFTP server and the filename may be specified. If an IP address is not specified, it is taken from the environment variable 'bootserver'. If a filename is not specified, it is taken from the environment variable 'bootfile'. Note that the TFTP protocol limits the maximum file size (SREC file size, not actual binary data size) to 32 MBytes (even 16 MBytes on some TFTP servers). If this limit is exceeded, the load will fail.</p> <p>For the currently supported formats, the execution entrypoint of the image is embedded in the image. This address is saved such that the 'go' command can use it as the default entrypoint.</p> <p>During the load operation, the current load address will be shown on the 8-position hex display (if present).</p> <p>Note that the load command prevents the user from overwriting the environment flash area.</p>
OPTIONS	-r Retry on ARP timeout (until load succeeds or Ctrl-c is typed)

NAME	<pre>off (Atlas/SEAD)</pre> <p>Note : There is no "off" command on Malta in version 02.00 of YAMON even though the hardware supports it.</p>
SYNOPSIS	off
DESCRIPTION	Turn off board.
OPTIONS	None

NAME	pcicfg (Atlas/Malta only)
SYNOPSIS	pcicfg [-8 -16 -32] ([-r] <bus> <dev> <func> <addr> [<range>]) (-w <bus> <dev> <func> <addr> <val>)
DESCRIPTION	Read a value/range or write a value to PCI configuration space. All arguments are hexadecimal. Range parameter indicates the number of bytes to read. Default action is read. Default width is 32 bit.
OPTIONS	-r Read PCI configuration space -w Write to PCI configuration space -8 See data in units of bytes -16 See data in units of halfwords -32 See data in units of words

NAME	ping
SYNOPSIS	ping ipaddr [<datagramsize>]
DESCRIPTION	ping - send ICMP ECHO_REQUEST packets to network host. This command is typically used to verify end-to-end network functionality & connectivity in a debug or bring-up situation. An ICMP ECHO_REQUEST packet must be replied to with an ICMP ECHO_REPLY packet from the remote host ('ipaddr'). The ICMP ECHO packet will contain data with the specified size. The default datagramsize is 64 bytes, minimum is 0 bytes and maximum is 1472 bytes. The maximum size is constrained by the Ethernet upper frame size limit (IP segmentation is not supported). If the optional datagramsize parameter is not within the valid range, the default size of 64 bytes will be used. The ping command will stop when the first reply is received from the remote host. If no replies are received, depending on whether the MAC-address of the destination path has been resolved and kept in a cache, ARP or ICMP_ECHO REQUEST packets are retransmitted up to 3 times before an appropriate error message is finally returned. The user may stop the ping command at any time using Ctrl-C.
OPTIONS	None

NAME	port
SYNOPSIS	port [-a] [-8 -16 -32] <address> [<value>]
DESCRIPTION	Perform a read or write operation to the specified <address> with the specified data width (default 32 bits). If <value> is specified, this value is written, otherwise a read operation is performed and the result is displayed. The command checks the validity of the specified address (see Section 5.1, "Address Validation"). This check can be turned off using the '-a' option. The port command will result in exactly one read or write operation with the specified data width. This makes it useful for accessing registers in peripheral devices.
OPTIONS	-8 Access data byte -16 Access data halfword -32 Access data word -a Allow invalid address

NAME	reset (Atlas/Malta/SEAD-2)
SYNOPSIS	reset
DESCRIPTION	Performs a hardware reset of the board.
OPTIONS	None

NAME	scpu (only available if CPU is configurable).
SYNOPSIS	Depends on CPU. Type 'help scpu' at prompt to view synopsis.
DESCRIPTION	<p>Configure or view current cpu configuration.</p> <p>scpu does not by default modify the semi-permanent scpu setting recorded in the environment variable 'cpuconfig'.</p> <p>By default, cpuconfig is an empty string, implying processor specific hardware reset configuration.</p> <p>Use the '-p' option if you want to set the environment variable.</p> <p>Use 'unsetenv cpuconfig' if you wish to reset cpuconfig to an empty string.</p> <p>The following operations are available :</p> <p>Display available settings. Display current configuration. Edit configuration. Setup configuration based on environment variable. Reset configuration to hardware default. Store current configuration in environment variable.</p> <p>'scpu' without options or parameters displays the current configuration.</p>
OPTIONS	<p>-a Display available configurations. -u Configure based on environment variable. -r Reset configuration to hardware reset value. -p Commit configuration to environment variable.</p> <p>Other options are CPU specific. Type 'help scpu' at prompt to view options.</p>

NAME	search
SYNOPSIS	search [-asc -hex] <address> <size> <string>
DESCRIPTION	<p>Search for string in the memory area specified by <address> and <size>.</p> <p>Default string type is ASCII. If the search string contains spaces, remember to use quotes around the string.</p> <p>If searching for a hex string, the search pattern must be entered as a number of two-digit hexcodes without spaces inbetween.</p>
OPTIONS	<p>-asc Search for ASCII string -hex Search for hex string</p>

NAME	setenv
SYNOPSIS	setenv [<variable> [<value>]]
DESCRIPTION	<p>Set the specified environment variable.</p> <p>If no variable is specified, all environment variables are displayed.</p> <p>If no value is specified, the variable is set to the NULL value.</p> <p>When setting a R/W system variable, the value is first validated.</p>
OPTIONS	None

NAME	sleep
SYNOPSIS	sleep <ms>
DESCRIPTION	<p>Halt shell for the specified number of milliseconds.</p> <p>Note that the default number format is hexadecimal.</p>
OPTIONS	None

NAME	stty
SYNOPSIS	stty [-tty0 -tty1] [-b -u [-p][<baudrate>][n o e][7 8][1 2]]
DESCRIPTION	<p>Setup or view serial port setup. Default port is tty0. -b,-u,-p apply to the default port if no port is specified.</p> <p>The possible baudrates are generally 75-460800, but not all baudrates are supported by all platforms. Use 'stty -tyx -b' to get a list of the supported baudrates for a specific port.</p> <p>Available parity settings are n (none), o (odd), e (even). Available databits are 7 and 8. Available stopbits are 1 and 2.</p> <p>When changing the parameters for a tty which is being used (e.g. the console), some strange characters may appear as a result.</p> <p>Also note that stty does not by default modify the semi-permanent tty setting recorded in the environment variables. Use the '-p' option if you want to set the environment variable for the specific tty as well.</p>
OPTIONS	<p>-u Force environment settings for port to take effect -p Transfer current settings for port to environment -b List supported baud rates for the specified port -tty0 Setup port 0 - default -tty1 Setup port 1</p>

NAME	test
SYNOPSIS	test [-l] [-m] [<module> [<module arguments>]]
DESCRIPTION	<p>The test command can perform a number of self-tests on different modules. If no module is supplied, all available modules are tested and a final pass/fail status is indicated. If a module is specified, only this module is tested.</p> <p>If the option "-m" is applied and no module is specified, a list of the available modules is displayed.</p> <p>If the option "-m" is applied and a module is specified, additional information about the module test and the optional arguments is displayed.</p> <p>If the option "-l" is applied, all available modules are tested repetitively, until Ctrl-C is pressed or a test fails. The "-l" option cannot be specified together with other options or arguments.</p>
OPTIONS	<p>-m List available test modules -l Loop default tests until Ctrl-C is pressed</p>

NAME	tlb (only available if CPU has TLB).									
SYNOPSIS	tlb -i [<index> <pagesize> <va> <g> <asid> <pa0> <c0> <d0> <v0> <pa1> <c1> <d1> <v1>]									
DESCRIPTION	<p>Display or edit TLB.</p> <p>In case there are no parameters, the contents of the TLB is displayed. In case (all) parameters are available, the TLB entry at the requested index is written.</p> <p>The number of TLB entries is CPU specific.</p> <p>Available settings of 'pagesize' are :</p> <pre>0x1000 4kB 0x4000 16kB 0x10000 64kB 0x40000 256kB 0x100000 1MB 0x400000 4MB 0x1000000 16MB</pre> <p>Available settings of c0/c1 (cache algorithm for even/odd page) are processor specific. However, the values 2 and 3 are typically reserved for Uncached (2) and Cacheable (3) modes. Values 0..7 are available.</p> <p>Other parameters are :</p> <p>va : Virtual base address of even/odd pair of pages. g : GLOBAL setting ('n' -> ASID is used, 'y' -> Ignore ASID). asid : ASID setting (only relevant if g = 'n'). pa0 : Physical base address of even page. d0 : DIRTY setting of even page ('y' -> write enabled, 'n' -> write protected). v0 : VALID setting of even page ('y' -> valid, 'n' -> not valid). pa1 : Physical base address of odd page. d1 : DIRTY setting of odd page ('y' -> write enabled, 'n' -> write protected). v1 : VALID setting of odd page ('y' -> valid, 'n' -> not valid).</p> <p>Example :</p> <pre>TLB index = 2 Pagesize = 4kB Global mapping (i.e. ASID ignored) ASID = 0xff (but ignored) Cache algorithm = 3 (Cacheable) Both pages valid</pre> <table border="1"> <thead> <tr> <th>Virtual address</th> <th>Physical address</th> <th>Dirty (i.e. write enabled)</th> </tr> </thead> <tbody> <tr> <td>0x00000000</td> <td>0x00200000</td> <td>Yes</td> </tr> <tr> <td>0x00001000</td> <td>0x00300000</td> <td>No</td> </tr> </tbody> </table> <p>tlb 2 4kB 0 y ff 200000 3 y 300000 3 n y</p>	Virtual address	Physical address	Dirty (i.e. write enabled)	0x00000000	0x00200000	Yes	0x00001000	0x00300000	No
Virtual address	Physical address	Dirty (i.e. write enabled)								
0x00000000	0x00200000	Yes								
0x00001000	0x00300000	No								
OPTIONS	-i Initialize TLB									

NAME	unsetenv
SYNOPSIS	unsetenv <variable> (-u -s)+
DESCRIPTION	<p>Unset specified environment variable. If a user created variable is specified, it will be removed from the environment. A system variable will not be removed, but will instead be set to the default value.</p> <p>By using the options below, all user and/or system variables can be unset using a single command.</p>
OPTIONS	-u Delete all user variables. -s Reset all read/write (R/W) system variables to default values.

4 Environment Variables

YAMON has support for environment variables stored in non volatile memory (flash). YAMON creates a number of environment variables, and the user may create his own using the “setenv” command.

The shell parser expands environment variables. For example, the command “load \$file” will expand to “load /test.rec” in case the user has created an environment variable “file” using the command “setenv file /test.rec”.

Environment variables also provide a means to pass data to applications. The “go” command will set CPU register a2 to the address of an array of structures holding the environment variables. This is described in [Section 8, "Application Interface"](#).

YAMON creates a number of environment variables. These are either Read Only (RO) or Read/Write (R/W). RO variables may not be modified nor deleted. R/W variables may be modified, but the new values are first validated by YAMON. R/W variables may also be reset to factory default settings using the “unsetenv” command.

[Table 4](#) holds YAMON created environment variables that are common to all boards.

[Table 5](#) holds YAMON created environment variables common to boards supporting Ethernet.

[Table 6](#) holds YAMON created environment variable “baseboardserial” common to boards supporting a serial number.

[Table 7](#) holds YAMON created environment variable “cpuconfig” available in case CPU is configurable. Some implementations of MIPS 4K, 5K class of processors allow configuration of cache and/or MMU type. The format of “cpuconfig” depends on the configurable parameters of the specific CPU.

The “Takes effect” column indicates when changes take effect (only applicable for R/W variables).

Table 4 Common Environment Variables

Name	Description	Format	Default	Access	Takes effect
start	Start command that will be executed after a reset. YAMON will give the user the option to press Ctrl-C within 2 seconds to cancel execution of the start command.	string	Empty string	R/W	After reset unless cancelled.
memsize	Size of RAM in bytes	0XXXXXXXX	RAM size detected	RO	N/A
modetty0 modetty1	Serial port settings.	<baudrate>, <parity>, <databits>, <stopbits>, <flowcontrol>	38400,n,8,1,hw	R/W	See below
bootserport	This is the default serial port used by the “load” command.	tty0 tty1	tty0	R/W	immediately
prompt	String used for shell prompt. Also displayed in LED display.	string	YAMON	R/W	immediately

Table 5 Environment Variables for Platforms Supporting Ethernet

Name	Description	Format	Default	Access	Takes effect
ethaddr	MAC address	XX.XX.XX.XX.XX.XX	Factory hardwired	RO	N/A
ipaddr	IP address	ddd.ddd.ddd.ddd	0.0.0.0	R/W	Immediately
subnetmask	Subnetmask	ddd.ddd.ddd.ddd	0.0.0.0	R/W	Immediately
gateway	Default gateway	ddd.ddd.ddd.ddd	0.0.0.0	R/W	Immediately
bootprot	Default boot protocol used by “load” command.	tftp asc	tftp	R/W	Immediately
bootserver	Default TFTP fileservier. This is the ip-address used for the TFTP fileservier if user does not specify otherwise in the “load” command.	ddd.ddd.ddd.ddd	0.0.0.0	R/W	Immediately
bootfile	Default TFTP file. This is the name of the file loaded if user does not specify otherwise in the “load” command.	String	Empty string	R/W	Immediately

Table 6 Board Serial Number

Name	Description	Format	Default	Access	Takes effect
baseboardserial	Serial number of board.	DDDDDDDDDD	Factory hardwired	RO	N/A

Table 7 CPU Configuration

Name	Description	Format	Default	Access	Takes effect
cpuconfig	CPU configuration in case I-cache, D-cache, MMU are configurable.	[<icache bytes per way>, <icache associativity>, <dcache bytes per way>, <dcache associativity>, (tlb fixed)]	Empty string => Hardware default settings	R/W	See below
cpuconfig	CPU configuration in case I-cache and D-cache are configurable.	[<icache bytes per way>, <icache associativity>, <dcache bytes per way>, <dcache associativity>]	Empty string => Hardware default settings	R/W	See below
cpuconfig	CPU configuration in case MMU is configurable.	[tlb fixed]	Empty string => Hardware default settings	R/W	See below

Modifying the environment variables “modetty0” or “modetty1” will not modify the serial port settings until an “stty -u” command is issued. On the other hand, the “stty” command will immediately take effect, but in order to store the new settings in the corresponding environment variable, an “stty -p” command must be issued.

On some boards, the tty settings specified by the environment variables will be applied after a reset. On other boards, fixed settings will be used after a reset (see [Section 7, "Board Specifics"](#)).

Modifying the environment variable “cpuconfig” will not modify the CPU configuration until an “scpu -u” command is issued. On the other hand, the “scpu” command will immediately take effect, but in order to store the new settings in the environment variable, an “scpu -p” command must be issued.

On some boards, the CPU configuration specified by the environment variable will be applied after a reset. On other boards, hardware default settings will be used after a reset (see [Section 7, "Board Specifics"](#)).

The user may create his own environment variables using the setenv command. These variables may be both modified and deleted.

On some boards, factory default settings may be restored by a board specific mechanism (see [Section 7, "Board Specifics"](#)). Note however that this will reset all YAMON created environment variables (serial port settings, cpu configuration and LAN related variables like IP address, default TFTP bootserver etc.).

On all boards, factory default settings may also be restored by using the “unsetenv -s” command. User created environment variables may be deleted by using the “unsetenv -u” command.

Should the environment flash area for some reason become corrupted, the entire area may be erased using the “erase -e” command. User created environment variables will be lost and environment variables created by YAMON will be reset to factory default settings.

4.1 Ethernet Support

Ethernet is supported on Atlas and Malta boards.

The IP-address for the board must be assigned by the local network administrator only. Using an unapproved IP-address may lead to duplicated IP-address network error events, which can disrupt network operation.

At boot, YAMON issues a warning message if either the “ipaddr” or “subnetmask” have not been set (i.e. are set to the factory default setting “0.0.0.0”).

When downloading S-records, it is checked, that the “ipaddr” and “subnetmask” are set and further it is checked that the board and the TFTP-server are on the same subnet. If they are not, “gateway” must be set to the default gateway.

A ping-server is active (responds to ICMP-ECHO-requests) whenever “ipaddr” has been set.

On the Atlas board, the Ethernet driver is polled rather than interrupt driven. This means that the ping-server will not respond while commands are being executed. For example, potentially time-consuming commands like “cksum” and “erase” may cause YAMON not to respond for a long period.

5 Special Issues

5.1 Address Validation

YAMON commands that require an address (e.g. port, edit, dump etc.) will validate the address. This includes the following :

- Alignment.
- In case of TLB-mapped address space, the TLB setup is verified. This includes test for TLB miss as well as write protection of page.
- In case of RAM address space, it is verified that RAM is available for the access. For example, if the platform supports 128 MB RAM, and a 64 MB RAM module is used, only the first 64 MB are available.

Some commands perform validation of flash ranges (see [Section 11, "Flash Support"](#)).

5.2 Exception Handling

If an “unexpected” exception occurs, while executing a user application, the application is terminated, the user context (CPU registers 0..31 and selected CP0 registers) is displayed and YAMON returns to the prompt.

If an “unexpected” exception occurs in YAMON context, the YAMON context (CPU registers 0..31 and selected CP0 registers) is displayed and YAMON enters an infinite loop.

When an “unexpected” exception occurs, the CP0 EPC or ERROREPC (depending on exception type) is displayed in the ASCII LED display.

By “unexpected” is meant an exception (including interrupts) for which no handler has been registered. Interrupt handlers are installed for the interrupts used by device drivers. Also, a user application may register handlers for specific exceptions (see [Section 8.3, "Functions"](#)).

If an NMI or EJTAG exception is detected, YAMON will jump to the following addresses :

EJTAG : 0x80000300
NMI : 0x80000380

YAMON has installed code at these addresses (just like it has done at for example the general exception vector address 0x80000180) that will jump to the exception handling functions of YAMON, but applications may install their own code at the above vector addresses.

Unless applications have taken over EJTAG exceptions, they will be handled as any other exception.

Unless an application has taken over NMI exceptions, they will be handled as any other unexpected exceptions. NMI exceptions are very useful for debugging “run-away” applications.

5.3 Cache Issues

It is a potential source of errors to access two memory locations contained within the same cache line both cached and uncached. If, for example, an address is written uncached, it may later be overwritten if the corresponding cache line is evicted.

Another potential source of errors is when an application is loaded to memory. Since this is done by writing the instructions to memory (D-cache domain), it is important that the I-cache is invalidated, so that it will be refilled before

executing new instructions. To make sure the I-cache refill is performed on the correct data, the D-cache must be flushed to physical memory after loading the application. Also, if the application executes uncached, it is important to flush the D-cache before starting to load the application. All of this is handled by the “load” command.

YAMON does not usually flush the caches “behind the back” of the user, so if the user issues an “edit” command to uncached memory, and a memory location within the same cache line has previously been accessed cached, it is the responsibility of the user to make sure the D-cache has been flushed by issuing the “flush -d” command.

The following commands are the only commands where YAMON takes care of flushing the caches :

- load
- gdb
- copy
- scpu
- cache

The copy command flushes the caches before and after copying data. This is done since the copy command is expected to be frequently used for moving applications between e.g. flash and RAM. Doing this requires the I-cache to be invalidated as described above. If the user does not want the copy command to flush caches, he may apply the “-f” option.

6 Memory Layout

The memory map for YAMON is shown in the following table (see also [1], [2] and [3]).

“Vector table” is the table described in [Section 8.3, "Functions"](#). It is a table with addresses of functions that YAMON provide to applications for printing data to the terminal, cache flush, exiting application etc. The actual functions are located at the base address named “YAMON functions” in the table below.

Table 8 Memory Map (Physical Addresses)

	Atlas / Malta	SEAD / SEAD-2
SDRAM base	0x00000000 (Max 128MB)	0x00000000 (32 MB)
YAMON functions	0x00001000	0x00001000
YAMON image	0x00005000	0x00005000
PCI Memory space	0x08000000 (Max 256 MB)	N/A
PCI I/O space	0x18000000 (Max 62 MB)	N/A
System controller	0x1be00000 (2 MB)	N/A
SDRAM controller	N/A	0x1b000000 (1 MB)
GPIO module	N/A	0x1b100000 (1 MB)
Performance module	N/A	0x1b200000 (1 MB)
System flash	0x1c000000 (32 MB)	0x1c000000 (31.75 MB)
Monitor flash	0x1e000000 (4 MB)	N/A
Environment flash (environment variables)	0x1e3e0000 (128 kB)	0x1dfc0000 (256 kB)
FPGA (interrupt controller, timer, LED display etc.)	0x1f000000 (12 MB)	N/A
SRAM	N/A	0x1e000000 (4 MB)
Peripheral controller registers	N/A	0x1e800000 (4 MB)
Peripheral bus devices	N/A	0x1f000000 (12 MB)
Bootcode base	0x1fc00000 (4 MB)	0x1fc00000 (1 MB)
Vector table	0x1fc00500	0x1fc00500
Little endian code	0x1fc10000	0x1fc10000
Big endian code	0x1fc58000	0x1fc58000

On the Malta board, the hardware will redirect accesses to the “Bootcode” range (based at 0x1fc00000) to “Monitor flash” (0x1e000000).

On the Atlas board, the hardware will redirect accesses to the “Bootcode” range (based at 0x1fc00000) to either “Monitor flash” (0x1e000000) or the upper 4MB of “System flash” based on jumper settings (see [1]).

7 Board Specifics

This section (including the table below) describes the issues specific to the boards supported by YAMON.

“Break effect on tty0” describes the effect of an RS232 break on port tty0. On the Atlas and Malta boards, the reset triggered by “Break on tty0” may be disabled as described in [1] and [2]. This involves writing 0 to the BRKRES register (address 0xbf000508)

Table 9 Board Specific Properties

	Atlas / Malta	SEAD / SEAD-2
tty0 position.	Top (Atlas) / Left (Malta)	Top
tty1 position.	Bottom (Atlas) / Right (Malta)	Bottom
Serial port settings following a reset.	Taken from environment variables “modetty0” and “modetty1”.	Always 38400, n, 8, 1, hw
CPU configuration following a reset.	Taken from environment variable ‘cpuconfig’ if the CPU is configurable.	CPU hardware default value.
Break effect on tty0	Triggers hardware reset.	Same as Ctrl-C
Ethernet support.	Atlas : 10 Mbps, half duplex. Malta : 10/100 Mbps, half/full duplex.	N/A
Switch used to reset environment variables to factory default settings.	S5-4 followed by reset.	N/A
SDRAM modules supported (PC100).	16MB..256MB	32MB..256MB
Max SDRAM utilized.	128MB	32MB
Required SDRAM parameters.	Must support Burst length = 8 and CAS latency = 2	Mixed-mode module configurations not allowed. (SDRAM banks, DIMM banks) = (2, 2) or (4, any). (rows, columns) = (11, min 9) or (12, min 8). Burst length will be set to 1. CAS latency will be set to 2 if SDRAM supports this, otherwise 3.

7.1 Atlas/Malta

This section describes issues specific to the Atlas and Malta boards.

7.1.1 Upgrading YAMON

On the Atlas and Malta board, the following sequence will program YAMON in flash memory :

- Copy the zip file to your work directory.
- Unzip the file.
- Connect parallel download cable from parallel port on your computer to the Atlas or Malta board using the procedure described in [1] and [2].
- Set switch S5-1 (“PROG”).

- Copy deleteall.fl, yamon-<rev>.fl, lock.fl to the parallel port.
This can also be done by executing go.bat (DOS) or the Perl script go.pl (Linux/Solaris).
- When the parallel download has completed, the display will show the text “FINISHED”.
- Unset switch S5-1.
- Press the reset button.

7.2 SEAD

This section describes issues specific to the SEAD board.

7.2.1 Upgrading YAMON

YAMON is contained in an (E)PROM. Update YAMON by replacing the (E)PROM device (U23). The file required for programming the (E)PROM device is yamon-<rev>.bin.

7.3 SEAD-2

This section describes issues specific to the SEAD-2 board.

7.3.1 Upgrading YAMON

YAMON is contained in flash. YAMON is updated using the USB port as described in [4]. The file required for programming the (E)PROM device is yamon-<rev>.bin.

8 Application Interface

YAMON may be used to load and execute user applications. The details of this are described in this section.

8.1 Entry

Applications may be loaded by YAMON using the “load” command and executed using the “go” command (or under GDB control using the “gdb” command).

The application calling convention uses the standard “argc, argv” approach.

The go command will setup the registers as described in [Table 10](#) before jumping to the requested address (obtained from the previous load or as a parameter to “go”).

Table 10 Initial Application Context

Field	Value
reg4 (a0)	Set to the argument count.
reg5 (a1)	Pointer to array of strings holding the arguments. argv[0] == “go” in case application is started by “go” command. argv[0] == “gdb” in case application is started by “gdb” command.
reg6 (a2)	Pointer to table holding environment variables.
reg7 (a3)	Size of memory (in bytes).
reg29 (sp)	4 words below top of memory range reserved for user stack (size defined by symbol SYS_APPL_STACK_SIZE defined in include/sys_api.h).
reg31 (ra)	Return address. Application may jump to this address in order to exit and return to YAMON.
Other CPU registers	0
cp0_status	Same as YAMON context except that IE bit is cleared thus disabling interrupts.
epc	Entry point of application obtained from “load” command or as a parameter to “go”.
Other CP0 registers	Identical to YAMON context.

In particular, note :

- a0 is set to the number of white space separated tokens on the command line after expanding environment variables. The first token is “go” or “gdb”. The ‘address’ or ‘.’ tokens are not counted nor included in the argv array (see “go” and “gdb” command syntax in [Section 3, "Shell Commands"](#)).
- a1 is set to the address of an array of pointers to strings containing the tokens.
- a2 points to an array of structures holding the environment variables (see [Section 8.4, "Environment Variables"](#)).
- a3 is set to the SDRAM memory size (number of bytes).
- ra holds the return address for returning to YAMON. An application can return to YAMON by jumping to this address. Alternatively, the application may call the exit function supplied by YAMON (see [Section 8.3, "Functions"](#)).
- sp (stack pointer) is set to point 4 words from the top of an area reserved for user stack (stack size can be found by issuing the “info memory” command). Available user stack can be assumed to be at least 0x1000 bytes.
- gp (global pointer) is not setup, so applications should setup gp if using gp-relative addressing
- Interrupts are disabled (CP0 STATUS register IE bit is cleared).

YAMON will disable hardware devices performing DMA before starting an application.

When the user application calls a YAMON function (see [Section 8.3, "Functions"](#)), the YAMON setup of CP0 registers will be restored. The user setup will be restored before returning to the application. This also includes the HI and LO registers.

8.2 Exit

Since YAMON resides in RAM, it is possible for an application to corrupt YAMON. The following requirements must be met in order to return to YAMON :

- YAMON must not be corrupted. YAMON resides in RAM starting at address 0. The first free RAM address is displayed by the sign-on message (also available by issuing command “info”).
- Hardware configuration must not be altered. State of hardware devices is not stored/restored by YAMON.

However, the state of CP0 registers and the register bank (registers \$0..\$31) are stored/restored by the context shift mechanism used by YAMON.

8.3 Functions

YAMON provides a set of functions callable by applications. Pointers to these function are located in a table located at physical address 0x1fc00500. Functions must be called via KSEG0 addresses.

Functions are called using the standard MIPS calling convention, where the first arguments are passed in registers a0..a3.

YAMON restores its private context (including CP0 registers and registers \$0..\$31) before performing the functions. There are no requirements for stack space since the above context shift also switches to the YAMON stack during the function call.

The function table has the following layout:

Table 11 YAMON Function Vector Table (Base Address 0x1fc00500)

Offset	Function
0x0	Reserved
0x4	print_count
0x8	Reserved
0xc	Reserved
0x10	Reserved
0x14	Reserved
0x18	Reserved
0x1c	Reserved
0x20	exit
0x24	Reserved
0x28	Reserved
0x2c	flush_cache

Table 11 YAMON Function Vector Table (Base Address 0x1fc00500) (Continued)

Offset	Function
0x30	Reserved
0x34	print
0x38	register_cpu_isr
0x3c	deregister_cpu_isr
0x40	register_ic_isr
0x44	deregister_ic_isr
0x48	register_esr
0x4c	deregister_esr
0x50	getchar
0x54	syscon_read

Descriptions of the functions and macros for access to the functions are available in the header file “yamon_api.h” contained in the binary distribution (as well as the include directory of YAMON in the source distribution).

An application may return to YAMON by either jumping to the address stored in register ra (KSEG0) when application was invoked, or by calling exit. In the first case, the contents of register v0 will be displayed by the shell. In the second case, the parameter (rc) passed to exit (in register a0) will be displayed by the shell.

8.4 Environment Variables

As mentioned above, register a2 points to an array of structures holding the environment variables.

The structures have the following layout :

```
typedef struct
{
    char *name;
    char *val;
}
t_yamon_env_var;
```

name points to a zero-terminated array of characters containing the name of the environment variable.

val points to a zero-terminated array of characters containing the value of the environment variable.

The last structure in the array has both name and val pointers with NULL values.

Since the pointers point to the actual environment variables stored in RAM and not a copy of these, the application is responsible for not corrupting these, at least if it intends to return to YAMON.

8.5 Sample Application

A sample “Hello world” application is available for demonstrating the application interface. The application prints “Hello world” on the terminal by looking up the address of the YAMON “print” function and calling it.

The source files, makefile and linker script are tar'ed and gzipped in the following file:

```
yamon-sampleappl-src-<rev>.tar.gz
```

<rev> is the revision number of the application. The revision number is not correlated to the revision number of YAMON.

The application has been build and tested in the following environment :

- Host running Sun Sparc Solaris 2.6.
- GNU Make version 3.77.
- GNU compiler tools (gcc, ld, objcopy, objdump) :
Cygnus GNUPro Embedded ToolSuite with MIPS support (e.g. gcc-2.9) or
Algorithmics SDE-MIPS 3.1.
- GNU gunzip tool.

The makefile contains the following expressions, which define the names of the compiler tools.

```
TOOLCHAIN = cygnus

ifeq ($(TOOLCHAIN),cygnus)
CC          = mips-elf-gcc
LD          = mips-elf-ld
OBJCOPY     = mips-elf-objcopy
OBJDUMP     = mips-elf-objdump
endif

ifeq ($(TOOLCHAIN),sde)
CC          = sde-gcc
LD          = sde-ld
OBJCOPY     = sde-objcopy
OBJDUMP     = sde-objdump
endif
```

Please replace this according to your setup.

By typing 'make', the following files are generated :

- appl.rec
- appl.elf
- appl.map

appl.rec is the s-record file to be used for loading the application (using the "load" command).

appl.elf holds the image in ELF format.

appl.map is a linker generated map file for the application.

9 GDB Interface

YAMON may function as a “GDB-STUB” supporting a subset of the “GDB Remote Protocol” interface as defined by GDB version 4.18. YAMON supports the 64 bit extensions defined by Algorithmics for the SDE-GDB debugger.

GDB may be downloaded from the “Free Software Foundation” web site at <http://www.fsf.org>.

GDB must be configured using the `--target=mips` option.

9.1 Connecting to GDB

YAMON enters “GDB-STUB” mode after issuing the “gdb” command. This means that YAMON starts polling for requests from GDB on port tty1. When YAMON is waiting for GDB requests, GDB-STUB mode may be abandoned by typing Ctrl-C on port tty0.

Initial application parameters may be applied as described in [Section 3, "Shell Commands"](#). The initial application context is the same as described for the “go” command (see [Table 10](#)).

Applications may register Exception Service Routines (ESRs) as described in [Section 8.3, "Functions"](#). However, the YAMON gdb stub must “own” the BREAK exception.

Setting breakpoints in exception handling code is not supported.

GDB assumes by default 9600 baud, so use the “`stty -tty1 9600`” command to configure tty1 accordingly.

On the host side, you should issue the following GDB commands :

```
set endian little (if YAMON is running little-endian).
target remote /dev/ttya (or whatever port is being used on the host).
```

9.2 GDB Remote Protocol Specification

The GDB remote protocol encodes requests from GDB to target in the following format :

```
$<command>[<data>]#<CSUM1><CSUM2>
```

<command> and <data> must be ASCII characters and cannot include characters '\$' or '#'.

CSUM1 and CSUM2 are ASCII hex representation of an 8-bit checksum of <command> and <data> fields. The most significant nibble is sent first. Checksum is calculated as the modulo 255 sum of the characters representing <command> and <data> (i.e. not including '\$' and '#').

No additional characters are allowed within a request (including space, TAB, CR/LF). However, all characters are ignored between requests until a '\$' is received.

The command field is case-sensitive (for example, 'g' and 'G' are two different commands).

The data and CSUM fields are case-insensitive.

YAMON immediately responds with either a "+" or a "-" character. "+" in case command was received successfully and "-" if checksum characters were missing or the checksum indicated an error.

When YAMON is done performing the requested command, YAMON will transmit the following data to the GDB host (CSUM in lower case) :

```
$<data>#<CSUM1><CSUM2>
```

<data> depends on the specific command. No additional characters are transmitted (including CR/LF).

Two error messages are used :

```
E01 : Illegal format.
E02 : Illegal address.
```

E01 is used for commands with illegal formats, while E02 is used if GDB requests access to an illegal address.

Addresses / register values are represented in hexadecimal format with the most significant byte first (i.e. big endian). Values may consist of 1 to 8 hex digits in case of 32 bit commands and 1 to 16 hex digits in case of 64 bit commands. No leading format identifiers are allowed (i.e. no "0x").

Example :

GDB may request to read 4 bytes starting from location 0xa0200000 by issuing the following command :

```
$ma0200000,4#80
```

YAMON will reply with the following sequence (assuming data read = 0x01, 0x02, 0x03, 0x04)

```
+$01020304#8a
```

9.3 GDB Remote Protocol Requests

The following requests are supported :

Format does not include \$, #, CSUM fields and the replies assume that no errors occurred. Error messages are described in the previous section.

Extended operations	
Format	!
Description	Use the extended remote protocol (only relevant for "R" command). Sticky -- only needs to be set once.
Reply	OK

Last signal	
Format	?
Description	Reply the current reason for stopping.
Reply	S05

Write 32 bit registers	
Format	G <data> (data consists of 90*8 hex digits)
Description	<p>Write 32 bit registers. Each register is represented by 4 bytes of register data each encoded as two hex digits. In total, 90 registers are written.</p> <p>The register sequence is the following :</p> <p>\$0 .. \$31</p> <p>CP0 STATUS Special register LO Special register HI CP0 BADVADDR CP0 CAUSE CP0 EPC</p> <p>\$fp0..\$fp31 (ignored by YAMON)</p> <p>CP1 FSR (ignored by YAMON) CP1 FIR (ignored by YAMON) Frame pointer (ignored by YAMON) Dummy register (ignored by YAMON)</p> <p>CP0 INDEX CP0 RANDOM CP0 ENTRYLO0 CP0 ENTRYLO1 CP0 CONTEXT CP0 PAGEMASK CP0 WIRED CP0 register 7 (ignored by YAMON) CP0 BADVADDR (ignored by YAMON) CP0 COUNT CP0 ENTRYHI CP0 COMPARE CP0 STATUS (ignored by YAMON) CP0 CAUSE (ignored by YAMON) CP0 EPC (ignored by YAMON) CP0 PRID</p>
Reply	OK

Write 64 bit registers	
Format	H <data> (data consists of 90*16 hex digits)
Description	<p>Write 64 bit registers. Each register is represented by 8 bytes of register data each encoded as two hex digits. In total, 90 registers are written.</p> <p>The register sequence is the following :</p> <p>\$0 .. \$31</p> <p>CP0 STATUS Special register LO Special register HI CP0 BADVADDR CP0 CAUSE CP0 EPC</p> <p>\$fp0..\$fp31 (ignored by YAMON)</p> <p>CP1 FSR (ignored by YAMON) CP1 FIR (ignored by YAMON) Frame pointer (ignored by YAMON) Dummy register (ignored by YAMON)</p> <p>CP0 INDEX CP0 RANDOM CP0 ENTRYLO0 CP0 ENTRYLO1 CP0 CONTEXT CP0 PAGEMASK CP0 WIRED CP0 register 7 (ignored by YAMON) CP0 BADVADDR (ignored by YAMON) CP0 COUNT CP0 ENTRYHI CP0 COMPARE CP0 STATUS (ignored by YAMON) CP0 CAUSE (ignored by YAMON) CP0 EPC (ignored by YAMON) CP0 PRID</p>
Reply	OK

Set Thread	
Format	Hg Hc
Description	Ignored by YAMON
Reply	OK

Read registers	
Format	g[S]
Description	<p>Read registers.</p> <p>'S' (default 4) is the number of bytes for a register (4 in case of 32 bit registers, 8 in case of 64 bit registers).</p> <p>Each register is represented by 4 or 8 bytes of register data each encoded as two hex digits. In total, 90 registers are read.</p> <p>The register sequence is the following :</p> <p>\$0 .. \$31</p> <p>CP0 STATUS Special register LO Special register HI CP0 BADVADDR CP0 CAUSE CP0 EPC</p> <p>\$fp0..\$fp31 (set to 0)</p> <p>CP1 FSR (set to 0) CP1 FIR (set to 0) Frame pointer (set to 0) Dummy register (set to 0)</p> <p>CP0 INDEX CP0 RANDOM CP0 ENTRYLO0 CP0 ENTRYLO1 CP0 CONTEXT CP0 PAGEMASK CP0 WIRED CP0 register 7 (set to 0) CP0 BADVADDR (again) CP0 COUNT CP0 ENTRYHI CP0 COMPARE CP0 STATUS (again) CP0 CAUSE (again) CP0 EPC (again) CP0 PRID</p>
Reply	<data> 'data' consists of 90*8 (32 bit registers) or 90*16 (64 bit registers) hex digits.

Read single register	
Format	r<NN>[:S]
Description	<p>Read value of register 'NN'</p> <p>'NN' corresponds to the register number as listed above for the 'g' command. As such, NN must be less than 90.</p> <p>Width depends on 'S' (default 4). S is 4 in case of 32 bit register and 8 in case of 64 bit register.</p>
Reply	<data> 'data' consists of 8 or 16 hex digits (depending on 32 or 64 bit registers).

Restart / Write Single Register	
Format	R (if running extended protocol) R<NN>[:S]=XX (if not running extended protocol).
Description	If running extended protocol (set by the '!' command), this command will cause application context to be reinitialised. If not running extended protocol, the 'R' command is identical to the 'P' command (see below).
Reply	OK

Write Single Register	
Format	P<NN>[:S]=XX
Description	Set register 'NN' to the value 'XX'. 'NN' corresponds to the register number as listed above for the 'g' command. As such, NN must be less than 90. Width depends on 'S' (default 4). S is 4 in case of 32 bit register and 8 in case of 64 bit register.
Reply	OK

Read memory	
Format	m <address>,<length>
Description	Read memory bytes. <address> is address, <length> is the byte count. Each byte in the reply is described by two hex digits.
Reply	<data>

Write memory	
Format	M<address>,<length>:<data>
Description	Write memory bytes. <address> is address, <length> is the byte count, <data> are the bytes to be written. Each byte is described by two hex digits.
Reply	OK

Continue	
Format	C <sig>[:<address>]
Description	Continue with signal <sig> (hex signal number). Signal is ignored by YAMON. Run from address given by <address> unless “;<address>” is omitted, in which case we resume at address stored in EPC/ERROREPC. Reply will be sent next time a breakpoint is encountered.
Reply	S05 (indicating TRAP)

Continue	
Format	c [<address>]
Description	Same as 'C' command except for the missing sig value (which is ignored anyway). Run from address given by <address> unless <address> is omitted, in which case we resume at address stored in EPC/ERROREPC. Reply will be sent next time a breakpoint is encountered.
Reply	S05 (indicating TRAP)

Singlestep	
Format	S <sig>[;<address>]
Description	Singlestep with signal <sig> (hex signal signal number). Signal is ignored by YAMON. Same as 'C' except that YAMON will setup breakpoint(s) (using BREAK instruction) causing singlestepping to be performed. Original instructions are restored by YAMON, so that this operation is transparent to GDB.
Reply	S05 (indicating TRAP)

Singlestep	
Format	s <address>
Description	Same as 'S' command except for the missing sig value (which is ignored anyway). Same as 'c' except that YAMON will setup breakpoint(s) (using BREAK instruction) causing singlestepping to be performed. Original instructions are restored by YAMON, so that this operation is transparent to GDB.
Reply	S05 (indicating TRAP)

Detach	
Format	D
Description	Identical to the Kill ("k") command described below.
Reply	OK

Kill	
Format	k
Description	Same as 'D' command. Will cause YAMON to finish "gdb" command and return to the YAMON prompt.
Reply	OK

10 Motorola S-records

The load command supports loading Motorola S-records.

An S record file is an ASCII file consisting of eight different record types (S0/S1/S2/S3/S5//S7/S8/S9). The format of a record is :

```
<type> <length> <address> <data> <checksum>
```

<type> identifies the record type. It consists of the characters “S0”, “S1”, “S2”, “S3”, “S5”, “S7”, “S8” or “S9”.

“S0” records are used for descriptive information identifying the following block of S records. “S0” records are ignored by YAMON.

“S1”, “S2”, “S3” records hold data/instructions to be stored in memory. “S1” are used for 2-byte addresses, “S2” are used for 3-byte addresses, and “S3” are used for 4-byte addresses.

“S5” records contain the number of S1, S2, and S3 records transmitted in a block. “S5” records are ignored by YAMON.

“S7”, “S8”, “S9” records hold the start address of the application. “S9” are used for 2-byte addresses, “S8” are used for 3-byte addresses, and “S7” are used for 4-byte addresses.

<length> is the number of bytes (each byte consisting of two ASCII characters holding the hexadecimal value) included in the <address>, <data> and <checksum> fields.

<address> field holds the address where <data> is to be stored in case of “S1”, “S2” and “S3” records, or the starting address of the code in case of “S7”, “S8” or “S9” records.

<address> is encoded using 2,3 or 4 bytes (i.e. 4, 6 or 8 ASCII characters) depending on the address width (2 in case of “S1” or “S9”, 3 in case of “S2” or “S8”, 4 in case of “S3” or “S7”).

<data> is the actual data to be stored in case of “S1”, “S2” and “S3” records or the information held in an “S0” record.

<csum> contains a one byte (2 ASCII characters) checksum calculated as the one’s complement of the sum of all the bytes from the <length> field through the end of the <data> field.

Between the <csum> field and the record termination character (see next paragraph), space or tab characters are accepted.

Each record can be terminated with either a carriage return or a line feed. Note: by invalid S-record detection, the reported line number of the invalid record is calculated as follows:

- if no line feeds have been detected during load, lines are counted as the number of carriage returns being parsed, starting with line number 1.
- when the first line feed is detected, the line number is set to 1, and the following lines are now counted as the number of line feeds being parsed.

Empty lines with space or tab characters, terminated with carriage return or line feed are accepted.

All valid letter-characters are not case-sensitive and mixed-case records are accepted.

One record of either type “S7”, “S8” or “S9” terminates the file and is mandatory.

Any S-record must not contain more than 200 characters including white space, tab and termination characters.

To summarize the YAMON S-record loader support, following table is applicable:

Table 12 Motorola S-record Types

Type	Description	Accept	Ignore
S0	informational		x
S1	2 byte address	x	
S2	3 byte address	x	
S3	4 byte address	x	
S5	block count		x
S7	4 byte start address	x	
S8	3 byte start address	x	
S9	2 byte start address	x	

The following paragraph displays an example of a valid S-record file:

```
S00E00006170706C2E656C2E726563DD
S31A80100000F0FFBD270800BEAF21F0A0031000C4AF1400C5AF1836
S31A8010001500C6AF00BF013C000820AC00BF023C0808428C000020
S31A8010002A00000400C2AF0400C28F000000003B00432C03006054
S31A8010003F14000000001E0004080000000000BF013C000820AC08
S31A8010005400BF023C0808428C000000000400C2AF0400C28F005C
S31A80100069BF013C100422AC0C00040800000000211000002100A4
S31A8010007E04080000000021E8C0030800BE8F1000BD270800E0CE
S30A801000930300000000CF
S705801000006A
```

Loading this file will result in the following info from YAMON:

```
Start = 0x80100000, range = (0x80100000,0x80100097) format = SREC
```

11 Flash Support

YAMON decodes and knows about following flash areas:

Table 13 Flash Areas on Supported Boards

	Atlas	Malta	SEAD/SEAD-2
Total (system+monitor+env) [MB]	36	4	32
System [MB]	32	N/A	31.75
System block size [kB]	256	N/A	256
System phys. mem map	0x1C00.0000-0x1DFF.FFFF	N/A	0x1C00.0000-0x1DFB.FFFF
Monitor [MB]	3.872	3.872	N/A
Monitor block size [kB]	128	128	N/A
Monitor, part1, phys. mem map YAMON-boot image (1MB) (lock-bit protected)	0x1E00.0000-0x1E0F.FFFF	0x1E00.0000-0x1E0F.FFFF	N/A
Monitor, part2, phys. mem map user application (2.872 MB)	0x1E10.0000-0x1E3D.FFFF	0x1E10.0000-0x1E3D.FFFF	N/A
Environment [kB]	128	128	256
Environment block size [kB]	128	128	256
Environment phys. mem map	0x1E3E.0000-0x1E3F.FFFF	0x1E3E.0000-0x1E3F.FFFF	0x1DFC.0000-0x1DFF.FFFF

- “System” flash is available on some boards only. It is used for general user applications.
- “Monitor” flash is available on some boards only. 1 MByte is reserved for the YAMON-boot image, and approximately 3 MByte are free for user application.
- “Environment” flash is reserved for saving YAMON environment variables.

Note: The environment flash area is decoded by use of the YAMON flash related commands, but any modification (copy, erase or load) will be rejected by YAMON and an error message will be returned. The physical address space 0x1FC0.0000-0x1FFF.FFFF (MIPS boot area) is not decoded by any of the flash related commands; i.e. no flash programming will occur if data are written to these addresses.

Following YAMON commands operate on flash devices:

- “copy”: When the destination address is decoded as a flash memory space, this command will program the flash with the source contents. Address and size may be on any byte-boundary. The destination area must be erased before copying. Caches are flushed before and after copying.
- “erase”: Flash memory is erased block-wise. This command calculates the closest possible address range of the decoded flash to be erased and prompts the user to confirm. Any lock-bits set in the specified address area will be cleared if possible.
- “load”: Downloading S-records may include flash programming if the S-record is bound to a flash memory area. This area must be erased before downloading. Caches are flushed before and after downloading.
- “setenv” & “unsetenv”: Any environment variable update is saved in the environment flash. Normally, the user should never touch the environment flash area, unless a re-initialization is requested with the “erase -e” command.

- “stty” and “scpu” : These are special cases of the “setenv”-command, as attributes may be written to the environment flash as regular environment variables.

All flash programming commands will check-read and verify all bytes being programmed to detect flash-failure or un-erased flash memory areas.

Note : The Atlas system flash may be write protected with the DIP switch S1-3 and any YAMON flash command attempt to erase or program the system flash, when write-protect is asserted, implies an error report like:

```
Error : Flash is write protected
Hint  : Disable write protection: Switch S1-3
```

Flash-memory addresses may be specified as cached (kseg0) or uncached (kseg1) in any YAMON flash related command.

Flash lock-bits are, except for the “erase” command, not handled by YAMON. However, YAMON detects if any block lock-bits are set and reports an error, if the erase or programming operation fails like:

```
Error : Some sectors are locked
Diag  : Flash status: (bc000000)=00920092
Hint  : Unlock sector(s) before programming
```

Note : To clear any block lock-bit(s) in the Atlas and Malta environment or monitor flash areas, the MFWR jumper (Atlas=JP8, Malta=JP1) must be fitted. Any attempt to clear the lock-bit(s) without having the MFWR jumper fitted, will result in an error report like:

```
Error : Environment FLASH is lock-bit protected
Diag  : Flash status: (be3e0000)=00a200a2
Hint  : Disable 'clear lock-bit' protection: (MFWR-jumper) must be fitted
```

or

```
Error : Some MONITOR FLASH sector(s) locked
Diag  : Flash status: (0xbe000000)=0x00a200a2
```

The monitor flash area on Atlas and Malta is generally made available to the YAMON user, except the lock-bit protected sectors, which keep the YAMON code.

12 Diagnostics and Error Messages

The following diagnostics messages are written to the ASCII LED display during startup (in sequence order).

Not all messages apply to all boards due to hardware dependencies.

You will usually not be able to see most of these messages unless something goes wrong during boot.

Table 14 ASCII LED Display Diagnostic Messages

Message	Description
Power On	Displayed by hardware if no core board is available (Atlas and Malta only).
Flash DL	Displayed by hardware if board has been placed in flash programming mode (Atlas and Malta only).
CPU	CPU specific initialisation.
CACHE	Cache sized being detected.
ICACHE	I-cache being initialized.
DCACHE	D-cache being initialized.
BOARD	Board detection.
SEAD	SEAD specific initialisation.
SEAD-2	SEAD-2 specific initialisation.
ATLAS	Atlas specific initialisation.
MALTA	Malta specific initialisation.
SPD	Access to Serial Presence Detect device.
RAM	RAM test/clear.
RAM_HILO	Perform test of RAM Hi, medium, lo addresses.
RAM_TEST	Perform test of RAM used by YAMON.
CLEAR	Clearing RAM.
COPYTEXT	Copy text segments to RAM.
COPYDATA	Copy data segments to RAM.
STACK	Setting sp and gp.
INFO	Store various system info in RAM.
CINFO	Store various cache info in RAM.
FIRSTC	About to call first C-function.
IO	Initializing YAMON module "IO".
EXCEP	Initializing YAMON exception handler module.
RTC	Initializing YAMON RTC driver.
FREQ	Initializing YAMON module "FREQ".
FREQCPU	Detecting CPU frequency.

Table 14 ASCII LED Display Diagnostic Messages (Continued)

Message	Description
FREQBUS	Detecting bus frequency.
PCI	Performing PCI auto configuration.
GT64120	Galileo GT64120 North Bridge detected.
PCI WAIT	Pause for a while to allow target devices on PCI backplane to boot.
IIC	Initializing YAMON IIC driver.
EEPROM	Initializing YAMON eeprom driver.
FLASH	Initializing YAMON flash driver.
SYSENV	Initializing YAMON module "SYSENV".
ENV	Initializing YAMON module "ENV".
SERIAL	Initializing YAMON Serial drivers.
SAA_LAN	Initializing SAA9730 Lan driver.
AMD_LAN	Initializing AMD Lan driver.
NET	Initializing YAMON module "NET".
CPU_U	Updating CPU configuration based on environment variable 'cpuconfig'. Only done on some boards (see Section 7, "Board Specifics") and only done if CPU is configurable.
INITDONE	Initialisation done.
<prompt>	YAMON shell executing. This message may be selected using the "prompt" environment variable, which by default has the value "YAMON".

The following error messages are written to the ASCII LED display in case of errors. The messages starting with "E:" are used during startup.

Table 15 ASCII LED Display Error Messages

Message	Description
Hex number	Exception occurred. Value represents EPC / ERROREPC
NMI	NMI detected.
E:BOARD	Unknown board.
E:CORE	Unknown core board.
E:CPU	Unknown processor.
E:NO_RAM	RAM "Serial Presence Detect" device failed (probably missing SDRAM module).
E:RAM_CL	SDRAM does not support CAS latency 2.
E:RAM_BL	SDRAM does not support burst length 8.
E:RAM_EC	Unsupported error check for SDRAM module (ECC).
E:RAM_DB	Too many SDRAM device banks (>4).

Table 15 ASCII LED Display Error Messages (Continued)

Message	Description
E:RAM_CF	General SDRAM configuration error.
E:RAM_LO	Lowest RAM address failed.
E:RAM_MI	Middle RAM address failed.
E:RAM_HI	High RAM address failed.
E:RAM_W	RAM test failed during word access.
E:RAM_B	RAM test failed during byte access.
E:NB_CFG	PCI configuration cycle to SAA9730 device failed.
E:NB_DEC	North Bridge setup (decoding) failed.
E:LAN	Lan driver initialisation failed. This message should not be seen, since YAMON will not halt if LAN initialisation fails.
E:P_ALLO	Not enough space for auto configuration
E:P_RANG	Illegal ram range for PCI mapping.
E:P_UNKN	Unknown PCI device on board (should never happen).
E:STRUCT	Structural error in YAMON code detected (should never happen).
E:UNKNWN	Unknown error code (should never happen).

Appendices

A References

- [1] Atlas(tm) User's Manual,
MD00005
- [2] Malta(tm) User's Manual,
MD00048
- [3] SEAD(tm) Basic RTL User's Manual,
MD00017
- [4] SEAD-2(tm) Basic Package Getting Started,
MD00062
- [5] YAMON(tm) Reference Manual,
MD00009

B Revision History

Revision	Date	Description
01.00	99/12/15	Initial revision for YAMON 01.00
01.01	99/02/08	Updated for YAMON 01.01
01.02	00/03/22	Updated copyright notice
02.00	00/09/11	Updated for YAMON 02.00
02.01	01/01/31	Document layout modified