YAMON™ User's Manual

Document Number: MD00008
Revision 02.14
November 12, 2007

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

Template: nB1.03, Built with tags: 2B

Table of Contents

List of Tables

Chapter 1

Introduction

YAMON ("**Y**et **A**nother **MON**itor") is the ROM monitor used to control and monitor program execution on the evaluation and reference boards from MIPS Technologies Inc. YAMON includes boot code as well as traditional monitor functionality used for loading, executing, and debugging applications. YAMON source code is highly portable to other MIPS-based platforms. The latest version of YAMON can be downloaded from http://www.mips.com.

1.1 Features

YAMON's features include:

- Same binary image is used for all boards and CPUs supported by YAMON. YAMON detects the specific board/CPU at run-time.
- Binary image contains both little- and big-endian code. YAMON detects the endianness at run-time and executes the appropriate code.
- 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, IDE, and serial port support (Ethernet and IDE not available on all boards).
- Configuration of CPU for CPUs supporting this.
- FPU emulation.

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 MIPS Technologies.
- Vector table-based call interface for use by applications; PMON compatible.
- Ethernet for TFTP-load, save, and "ping" support (Ethernet not available on all boards).
- IDE for reading and writing sectors on harddisk / compact flash (IDE not available on all boards).

YAMON actively supports the following boards:

- Malta™

In addition, YAMON supports the following boards, but note that support for these is not guaranteed in future releases:

- SEAD™
- SEAD-2™
- Atlas™

SEAD and SEAD-2 boards can be equipped with CPU cards with MIPS32® 4K® or MIPS64® 5K® class of CPUs. SEAD and SEAD-2 boards are shipped with a system controller ("Basic RTL") included for the on-board FPGA. YAMON supports "Basic RTL" revisions up to and including 01.03 and "MIPS SOC-it® 101" system controller revision 1.1.

Malta and Atlas boards may be equipped with various "Core cards". A core card includes a CPU, a system controller (aka Northbridge), and SDRAM module. Table 1.1 shows the core cards and CPUs supported by YAMON for the Malta and Atlas boards.

Table 1.1 Supported Core Boards and CPUs

	Board Types		
	CoreLV™	**CoreFPGA™**	**QED5261 Board™**
Available CPUs	MIPS32® 4K® class CPUs MIPS64® 5K® class CPUs	MIPS32® 4K® class CPUs MIPS64® 5K® class CPUs MIPS32 M4K™ (in FPGA)	QED RM5261®
System Controller	GT64120®		
Supported by Malta	Yes		
Supported by Atlas			
	Core20K™	**CoreBonito64™**	**CoreFPGA-2™ CoreFPGA-3™**
Available CPUs	MIPS64® 20Kc™ MIPS64® 25Kf™	QED RM5261® QED RM7061A®	MIPS32® 4K® class CPUs MIPS64® 5K® class CPUs MIPS32® 24K® MIPS32® M4K™ MIPS32® 34K™ MIPS32® 74K™ (in FPGA)
System Controller	Bonito64		MIPS SOC-it® 101
Supported by Malta	Yes		Yes
Supported by Atlas	No		Yes
	Core24K®		
Available CPUs	MIPS32® 24K®		
System Controller	MIPS SOC-it® 101		
Supported by Malts	Yes		
Supported by Atlas	Yes		

1.2 YAMON™ Distribution

The board shipped from MIPS Technologies contains 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 (for example, 02.14).

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

yamon-bin-<rev>.zip contains the binary distribution (see Table 1.2).

yamon-*sampleappl*-src-<rev>.tar.gz contains source code for a sample “Hello world” application that uses the YAMON application interface (see Chapter 5, “Application Interface” on page 43). 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. This file is also described in Section 5.6 “Sample Application”.

Table 1.2 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.
go.bat	DOS batch file used for copying yamon-<rev>.fl file to parallel port (lpt1).
go.pl	Perl script used for copying yamon-<rev>.fl file 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 code located at the reset exception vector.
yamon-<rev>_el.map	Linker generated map file for little endian code.
yamon-<rev>_eb.map	Linker generated map file for big endian code.
yamon_api.h	Header file defining the application interface to YAMON (see Chapter 5, “Application Interface” on page 43).

See Chapter 4, “Board Specifics” on page 39 for board-specific instructions on how to upgrade YAMON.

Chapter 2

Getting Started

This chapter describes how to connect your terminal to YAMON, how to use the shell coomand-line interface, and how to load your board with a new revision of YAMON.

2.1 Connecting to YAMON

YAMON supports the two serial ports tty0 and tty1. YAMON uses tty0 as the console port. Chapter 4, “Board Specifics” on page 39 specifies the port mapping for particular boards. By default, the serial port settings are:

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

Some boards (e.g., Malta) allow this default port setting 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 2.3, "Environment Variables"and Chapter 4, “Board Specifics” on page 39.

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

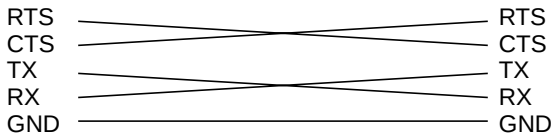

Figure 1 NULL-Modem Cable

By default, hardware flow control is enabled (RTS/CTS). This means that an empty console port connector will cause YAMON to hang in an attempt to write its welcome message. Sometimes, however, YAMON is needed only to boot an application (copy from flash and start) without being connected to a terminal. In that case, while still connected to YAMON, issue the command "`stty -p none`". This will override the default console port setting after reset/power-up, so that hardware flow control will be disabled.

2.2 Using the Shell Command Line Interface

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 a Malta board with a MIPS 34Kf processor. The sign-on message differs slightly for other boards/CPUs. Also, the software compilation date/time may differ.

```
YAMON ROM Monitor, Revision 02.14.
Copyright (c) 1999-2007 MIPS Technologies, Inc. - All Rights Reserved.

For a list of available commands, type 'help'.
Compilation time =              Nov 9 2007 17:35:01
Board type/revision =           0x02 (Malta) / 0x00
```

```
Core board type/revision =      0x09 (CoreFPGA-3) / 0x00
System controller/revision =    MIPS SOC-it 101 OCP / 1.3   SDR-FW-1:1
FPGA revision =                 0x0001
MAC address =                   00.d0.a0.00.03.2e
Board S/N =                     0000000566
PCI bus frequency =             25 MHz
Processor Company ID/options =  0x01 (MIPS Technologies, Inc.) / 0x00
Processor ID/revision =         0x95 (MIPS 34Kf) / 0x21
Endianness =                    Little
CPU/Bus frequency =             32 MHz / 32 MHz
Flash memory size =             4 MByte
SDRAM size =                    64 MByte
First free SDRAM address =      0x800b6750
```

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 2.1.

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

Table 2.1 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, a 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 2.2 are recognized by the shell parser.

Table 2.2 Shell Parser Special Characters

Character	Meaning
$ (dollar)	The following is an environment variable, which will be expanded by the parser. The environment variable may be enclosed in braces {} to distinguish it from surrounding text. Example: YAMON> set var " in " YAMON> echo embedded${var}string embedded in string
" (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 or ctrl-c is detected. If 'count' is zero, line will be repeated indefinitely (or until command fails or ctrl-c is detected). 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 can be stopped by typing Ctrl-C. An RS-232 break on tty0 is equivalent to either Ctrl-C or reset, depending on the board (see Chapter 4, "Board Specifics" on page 39).

Commands are case-sensitive.

2.2.1 Number Formats

The following number formats (case-insensitive) are supported:

- <*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 Shell Commands

This section describes each shell command. The commands are listed in alphabetical order.

The command "`help`" will display a list of the commands. The command "`help` <*command*>" will display help on a specific command.

Because of hardware dependencies, not all commands are available for all boards/CPUs.

NAME	.
SYNOPSIS	. (continuation/repeat command)
DESCRIPTION	This command is a continuation command for "dis", “dump” and "eeprom" commands, so that more data can be dumped without the user having to repeat the previous command with a new address. If the previous command was anything else than "dis", "dump" or "eeprom", this command acts like a simple repeat command.
OPTIONS	None

NAME	cache (synopsis and description are shown for a 5Kc)
SYNOPSIS	cache [<value> | off | on | wb | wt | wtall]
DESCRIPTION	Configure KSEG0 cache algorithm by setting K0 field of CPU CONFIG register. Second level cache may be enabled/disabled as well if CPU supports this. If no parameter is supplied, the current cache setting is displayed. The command will cause caches to be flushed. K0 settings are CPU specific. The mappings are shown below (a specific value may be given instead of these symbols). off: 2 (Uncached) on: 3 (Write-back, write allocate) wb: 3 (Write-back, write allocate) wt: 0 (Write-through, no write allocate) wtall: 1 (Write-through, write allocate) For CPU’s that support an L2 cache, this command will also allow you to disable or enable the L2 cache. l2on: L2 cache enable l2off: L2 cache disable
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> | ([-t n] [-<0..7>] [-32|-64] <regnum>)) [<value>]]
DESCRIPTION	Read/write CP0 register(s). If no arguments are applied, all CP0 registers are read. A register may be selected by name (case insensitive) or number (and possibly select value using -0,-1,-2,...,-7 option). On MT capable processors, TC specific registers can be selected by specifying the -t option. YAMON binds TC0 to VPE0 and TC1 to VPE1, so VPE specific registers may also be specified using the -t option. Select 0 is default. If a value is given, this value is written to the register, otherwise the register is read. 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. Settings of CP0 registers are also applied to user applications (started with ’`go`’ or ’`gdb`’) except for *STATUS*, for which the *IE* field is cleared. TLB related registers as well as *COUNT* and *COMPARE* are undefined when user application is first started. See Section 5.4, "Functions" for description of context-shift handling.
OPTIONS	-t n Select TC n (MT capable processors) -0 Select = 0 (default option) -1 Select = 1 -2 Select = 2 -3 Select = 3 -4 Select = 4 -5 Select = 5 -6 Select = 6 -7 Select = 7 -32 Access 32-bit of register (default option) -64 Access 64-bit of register (64 bit CPUs only)

NAME	cp1 (if CP1 is available)
SYNOPSIS	cp1 [(<name> | <regnum>) [<value>]]
DESCRIPTION	Read/write CP1 control register(s). If no arguments are applied, all CP1 control registers are read. A register may be selected by name or number. If a value is given, this value is written to the register, otherwise the register is read. Writing a CP1 control register takes effect immediately. Settings of CP1 control registers are also applied to user applications (started with 'go' or 'gdb'). See Section 5.4, "Functions" for description of context shift handling.
OPTIONS	None

NAME	date (Atlas/Malta only)
SYNOPSIS	date [[[cc]yy]mmddHHMM[.ss]] (HH in 24 hour format)
DESCRIPTION	Set or read date and time. 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. The argument must have the format [[cc]yy]mmddHHMM[.ss] where the hours HH are represented in 24h format (00-23). The output is shown as Day Mon dd HH:MM:SS ccyy.
OPTIONS	None

NAME	dis
SYNOPSIS	dis [-m] [-16] <address> [<count>]
DESCRIPTION	Disassemble code starting at <address>. Disassembles MIPS64/MIPS32 instructions. Includes MIPS-3D™ and MIPS16e™ ASEs. Includes MIPS64/MIPS32 Release 2 instructions. <count> (default 16) specifies the number of instructions to disassemble. The continuation command "." works together with "dis".
OPTIONS	-m Prompt user for keypress after each screen of data -16 Disassemble MIPS16e code

NAME	disk
SYNOPSIS	disk [-f] (id [hda|hdb|hdc|hdd]) | (read hda|hdb|hdc|hdd <sector> <count> <addr>) | (write hda|hdb|hdc|hdd <sector> <count> <addr>) (Malta only)
DESCRIPTION	Command for copying data to/from IDE harddisk or compact flash module. The disks are named the following way : Primary master: hda Primary slave: hdb Secondary master: hdc Secondary slave: hdd When a single device is attached to an interface, it is recommended to set it as master. Otherwise, the device may not be detected immediately following a reset. For example, a "disk id" command may not detect the device if executed directly following a reset, but will detect the device after a few seconds. Depending on the configuration, a command executed directly following a reset may take up to 30 seconds to complete (not including the time required for reading/writing data). Only LBA addressing is supported. Description: "disk id" Lists disk parameters (ID, size) for all disks available or the particular one (hda/hdb/hdc/hdd) requested. "disk read" Reads <count> sectors starting at <sector>. Data is written to <addr>. "disk write" Writes <count> sectors starting at <sector>. Data is read from <addr>. If a read operation is performed, and <addr> is flash, the destination area must be cleared using the ’erase’ command prior to the disk operation. Unless the -f option is applied, caches are flushed before and after a read operation (D-cache writeback and invalidate, I-cache invalidate).
OPTIONS	-f Do not flush caches

NAME	dump
SYNOPSIS	dump [-m][-8|-16|-32] <address> [<size>]
DESCRIPTION	Dumps data from address range starting at <address>. The default data display width is 8 bits. The <size> parameter specifies the number of bytes to dump (default is 256). The continuation command “.” works together with “dump”.
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	The echo command echoes all of its arguments to the console. This offers a convenient way to examine the contents of specific environment variables (e.g., “echo $ipaddr”).
OPTIONS	None

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

NAME	eeprom
SYNOPSIS	eeprom [-m] <dev> <address> [<size>]
DESCRIPTION	Dumps <size> bytes (default 256) from eeprom device <dev> starting at <address>. The continuation command "." works together with "eeprom". Available settings of 'dev' parameter are: sys: Device for storing system data. spd0: Device used for SDRAM parameters (SPD).
OPTIONS	-m Prompt user for keypress after each screen of data

NAME	erase
SYNOPSIS	erase [-s|-e|<address> <size>]
DESCRIPTION	Erase flash memory. An option may be applied specifying which flash region to erase. If no such option is applied, the address range to be erased is specified by the <address> and <size> parameters. 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). If (and only if) the -e option (erase environment flash) is applied, the system environment variables are reinitialized to factory default values. 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. Erasing a large flash area takes time. It can easily take several minutes to erase a 32 MByte area. 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.
OPTIONS	-e Erase and reinitialize entire environment area. -s Erase entire system flash (default option). Only available on some platforms.

NAME	fill
SYNOPSIS	fill [-8|-16|-32] <address> <size> <data>
DESCRIPTION	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).
OPTIONS	-8 Fill data in units of bytes -16 Fill data in units of halfwords -32 Fill data in units of words

NAME	flush
SYNOPSIS	flush [-i|-d]
DESCRIPTION	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. The D-cache flush operation is in reality a write-back of dirty lines (write-back caches only) followed by an invalidate operation.
OPTIONS	-i Invalidate I-cache -d Flush D-cache

NAME	fpu
SYNOPSIS	fpu [on|off] [emul [on|off|stat|clear]] [fs [on|off]] [fr [on|off]]
DESCRIPTION	Controls the floating point unit and FPU emulator. The above specified parameters can be set in the environment variable “fpu” to control settings at startup. The FPU emulator supports all FP operations specified in MIPS32/MIPS64 - except 64 bit addressing, Paired Single and extended (ASE) FP operations.
OPTIONS	<blank>Show current status. Applicable when FPU is present onEnable the hardware FPU. offDisable the hardware FPU Applicable when FPU emulator is compiled in emulShow current status of emulator. emul onEnable the emulator. If the system has a hardware FPU which is enabled, the emulator will handle denormalized and out of range numbers. If the system lacks an FPU or the FPU is disabled, the emulator will handle *all* FP operations. emul offDisable the emulator. If the system has a hardware FPU which is enabled, FP operations on denormalized and out of range numbers will cause an exception. If the system lacks an FPU or the FPU is disabled, *all* FP operations will cause an exception. emul statShow emulator statistics. emul clearClear emulator statistics. Always applicable fsShow current status of FS. fs onEnable the flush to zero bit. When the FS bit is set, denormalized numbers will be replaced by zero (This is not IEEE compliant). fs offDisable the flush to zero bit. Denormalized numbers must be handled by the emulator. frShow current status of FR. fr onEnable the FR bit. All 32 double-sized FP registers are exposed. fr offDisable the FR bit. Only 16 double-sized FP registers (even numbers) are exposed.

NAME	fread
SYNOPSIS	fread tftp://<ipaddr>/<filename> <address>
DESCRIPTION	Load binary image to RAM or flash (depending on address) from TFTP server. Note that the exact limitation on the filesize in the TFTP protocol is 33553919 bytes (appr. 32 Mbytes). Any file larger than this size cannot be transferred.
OPTIONS	None

NAME	fwrite
SYNOPSIS	fwrite tftp://<ipaddr>/<filename> <address> <size>
DESCRIPTION	Save binary image from RAM or flash (depending on address) to TFTP server. Note that the exact limitation on the filesize in the TFTP protocol is 0x1fffdff bytes (appr. 32 Mbytes). Any file larger than this size cannot be transferred.
OPTIONS	None

NAME	gdb
SYNOPSIS	gdb [-v][-c] [. <args>]
DESCRIPTION	Setup connection to GDB debugger on port tty1. The Standard GDB remote protocol is used. If the user application is not currently running, and Ctrl-C is typed at the console port, YAMON will leave GDB mode and return to the command prompt. <args> is broken up in substrings and passed to the application. The list of arguments to be passed must begin with a ".". The "." is not passed as an argument. The first argument (argv[0]) will be the string "gdb". Section 5.1 "Entry" describes how arguments are passed to the application and how the application context is initially setup. The application may return to YAMON by jumping to the address specified in ra or by calling the exit(rc) function supplied by YAMON. The verbose (-v) option will cause the commands from the GDB host and the responses from YAMON to be displayed on the console port. The checksum off (-c) 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.
OPTIONS	-v Display messages from and to GDB host -c Disable check of GDB checksum

NAME	go
SYNOPSIS	go [?|.|<address> [<args>]]
DESCRIPTION	Execute application code. If a target address is not specified, the address obtained from the last successful "load" command (if any) is used as the target address. This address may be determined by issuing a "go ?" command. The application will not be executed in this case. If arguments for the user program need to be specified, the default execution address can be referenced by a ".". Section 5.1 "Entry" describes how arguments are passed to the application and how the application context is initially setup. The application may return to YAMON by jumping to the address specified in ra or by calling the exit(rc) function supplied by YAMON.
OPTIONS	None

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

NAME	info
SYNOPSIS	info [boot | board | cpu | sysctrl | memory | uart | all | pci | ide | isa | lan]
DESCRIPTION	Display information on the requested item (default boot). The following information displays can be requested: boot: Info displayed after reset board: Board properties cpu: CPU properties sysctrl: System Controller/Basic RTL properties memory: Memory properties uart: Serial ports statistics all: All info pci (Atlas/Malta only): PCI autodiscovery/autoconfiguration ide (Malta only): IDE configuration isa (Malta only): ISA bus configuration lan (Atlas/Malta only): Ethernet statistics
OPTIONS	None

NAME	load
SYNOPSIS	load [-r] ([tftp:][//<ipaddr>][/<filename>]) | ([asc:] [//(tty0|tty1)])
DESCRIPTION	Load image from serial port or Ethernet to RAM or flash (depending on address). The only image type currently supported is SREC. 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". If loading from serial port, the default port is taken from the environment variable "bootserport". 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 exact limitation on the filesize in the TFTP protocol is 33553919 bytes (appr. 32 Mbytes). Any file larger than this size cannot be transferred. 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. During the load operation, the current load address will be shown on the 8-position hex display (if present). Note that the load command prevents the user from overwriting the environment flash area.
OPTIONS	-r Retry on ARP timeout (until load succeeds or Ctrl-c is typed)

NAME	off (Atlas/SEAD only)
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 (Atlas/Malta only)
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 resoluted 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 2.5.1 “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 only)
SYNOPSIS	reset
DESCRIPTION	Performs a hardware reset of the board.
OPTIONS	None

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

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

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

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

NAME	stty
SYNOPSIS	stty [-tty0|-tty1] [-b|-u| [-p][<baudrate>][n|o|e][7|8][1|2][hw|none]]
DESCRIPTION	Setup or view serial port setup. Default port is tty0. -b,-u,-p apply to the default port if no port is specified. The possible baudrates are generally 75-460800, but not all baudrates are supported by all platforms. Use "stty -ttyx -b" to get a list of the supported baudrates for a specific port. Available parity settings are n (none), o (odd), e (even). Available databits are 7 and 8. Available stopbits are 1 and 2. Available flowctrl settings are hw and none When changing the parameters for a tty which is being used (e.g., the console), some strange characters may appear as a result. 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.
OPTIONS	-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

NAME	test
SYNOPSIS	test [-l] | [-m] [<module> [<module arguments>]]
DESCRIPTION	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. If the option “-m” is applied and no module is specified, a list of the available modules is displayed. If the option “-m” is applied and a module is specified, additional information about the module test and the optional arguments is displayed. 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.
OPTIONS	-m List available test modules -l Loop default tests until Ctrl-C is pressed

NAME	tlb (if CPU has TLB).
SYNOPSIS	tlb (-i [-s]) | (<index> <pagesize> <va> <g> <asid> <pa0> <c0> <d0> <v0> <pa1> <c1> <d1> <v1>)
DESCRIPTION	Display or edit TLB. Some CPUs (MIPS32/MIPS64 Release 2 or later only) may support small pages (1kB). The '-s' option is only available when running on such a CPU. '-s' is used to toggle between enabling and disabling small pages. It may only be used in conjunction with '-i' since toggling this state requires the TLB to be reinitialised. In case there are no parameters, the contents of the TLB is displayed. If small pages are available, the state of this feature (enabled/disabled) is also displayed. In case (all) parameters are available, the TLB entry at the requested index is written. The number of TLB entries is CPU specific. Available settings of <pagesize> are : 0x400 | 1kB (if small pages are enabled) 0x1000 | 4kB 0x4000 | 16kB 0x10000 | 64kB 0x40000 | 256kB 0x100000 | 1MB 0x400000 | 4MB 0x1000000 | 16MB 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. Other parameters are : va : Virtual base address of even/odd pair of pages. The va specified is the one used for the even page, so it must be aligned to pagesize * 2. 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. Must be aligned to pagesize. 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. Must be aligned to pagesize. d1 : DIRTY setting of odd page ('y' -> write enabled, 'n' -> write protected). v1 : VALID setting of odd page ('y' -> valid, 'n' -> not valid). Example : TLB index = 2 Pagesize = 4kB Global mapping (i.e. ASID ignored) ASID = 0xff (but ignored) Cache algorithm = 3 (Cacheable) Both pages valid Virtual address Physical address Dirty (i.e. write enabled) --- 0x00000000 0x00200000 Yes 0x00001000 0x00300000 No tlb 2 4kB 0 y ff 200000 3 y y 300000 3 n y
OPTIONS	-i Initialize TLB -s Toggle small (1kB) page support (on CPUs supporting this)

NAME	unsetenv
SYNOPSIS	unsetenv <variable> | (-u|-s)+
DESCRIPTION	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. By using the options below, all user and/or system variables can be unset using a single command.
OPTIONS	-u Delete all user variables. -s Reset all read/write (R/W) system variables to default values.

2.3 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 containing the environment variables. This is described in Chapter 5, "Application Interface" on page 43.

YAMON creates a number of environment variables. These are either Read Only (RO) or Read/Write (R/W). RO variables may not be modified or 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 2.3 describes environment variables created by YAMON that are common to all boards.

Table 2.4 describes environment variables created by YAMON that are common to all boards supporting Ethernet.

Table 2.5 describes the environment variable `baseboardserial`, created by YAMON and common to boards supporting a serial number.

Table 2.6 describes the environment variable `cpuconfig`, created by YAMON and common to boards with a configurable CPU. Some implementations of MIPS 4K and 5K class of processors allow configuration of the 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 2.3 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.
`startdelay`	Number of seconds YAMON will wait for the user to press Ctrl-C to cancel execution of the start command. If set to 0, the start command will not be executed.	string	Empty string	R/W	After reset
`memsize`	Size of RAM in bytes	0xXXXXXXXX	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
`fpu`	Control FPU and/or FPU emulator	[on|off] [fs [on|off]] [fr [on|off]] [emul [on|off|stat|clear]]	Not set	R/W	After reset
`linewidth`	Sets the terminal width	number	Not set (80)	R/W	After reset
`linemax`	Sets the number of lines output before output is suspended. Set to 0 to disable paging	number	Not set (10)	R/W	After reset
`softendian`	Select the desired endianness of the board in software instead of via the hardware switch (if the board supports it)	number or string (0 or Little, 1 or Big, 2 or Hardware)	Hardware	R/W	After reset
`yamonrev`	YAMON revision	<major>.<minor>	02.14	RO	N/A

Table 2.4 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 fileserver. This is the ip-address used for the TFTP fileserver 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 2.5 Board Serial Number

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

Table 2.6 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 Chapter 4, “Board Specifics” on page 39).

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 Chapter 4, “Board Specifics” on page 39).

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 Chapter 4, “Board Specifics” on page 39). 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 using the “unsetenv -s” command. User-created environment variables may be deleted by using the “unsetenv -u” command.

Should the environment flash area 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.

2.4 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., they 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.

2.5 Special Issues

2.5.1 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 256 MB RAM, and a 64 MB RAM module is used, accesses above 64 MB and below 256 MB range will cause an error message.

Some commands perform validation of flash ranges (see Chapter 8, “Flash Support” on page 59).

2.5.2 Exception Handling

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

If an “unexpected” exception occurs in the YAMON context, the YAMON context is displayed (CPU registers 0..31 and selected CP0 registers as well as FPU registers if applicable), YAMON waits for one second, and then attempts to restart at the prompt.

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

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 5.4, "Functions").

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

```
EJTAG : 0x80000a00
NMI   : 0x80000a80
```

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.

Note that prior to YAMON 02.07, YAMON used the following addresses for EJTAG/NMI exception handling:

```
OLD_EJTAG : 0x80000300
OLD_NMI   : 0x80000380
```

These addresses clash with the vectored interrupt table when an external interrupt controller (EIC) is used. For compatibility, YAMON installs a jump from `EJTAG`/`NMI` to `OLD_EJTAG`/`OLD_NMI` to allow older applications to work when an EIC is not being.

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 exception. NMI exceptions are very useful for debugging “run-away” applications.

When an MT-capable processor is in use, YAMON does not take any special action to prior to jumping at the NMI vector. This allows an application to handle NMI’s independently on each VPE. The default YAMON NMI handler disables all thread contexts and VPEs except VPE0/TC0, which is used to run the NMI handler.

2.5.3 Cache Issues

IA potential source of errors is the access of two memory locations contained within the same cache line, both cached and uncached. If, for example, an address is written uncached, it may be overwritten later by hardware (in case of writeback caches) if the corresponding cache line was valid at the time and later evicted.

Another potential source of errors is the loading of an application to memory. Since this is done by writing the instructions to memory using `store word` instructions (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` and `gdb` commands.

I-cache invalidation is followed by a call to `sys_flush_pipeline`(), which performs an `eret` in order to flush the CPU pipeline. This is necessary because instructions in the cache line(s) being invalidated may already be in the CPU pipeline.

The `copy` and `disk` commands also flush the caches before and after copying data. This is done because these commands are expected to be frequently used for moving applications between, for example, Flash and RAM, and this requires flushing the D-cache and invalidating the I-cache, as previously described. If the user does not want the command to flush the caches, he may use the `-f` option.

Otherwise, 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 which take care of flushing the caches:

- `load`
- `gdb`
- `copy`
- `disk`
- `scpu`
- `cache`

If YAMON has been configured to run uncached (using the `cache` command), cache flushing will be disabled, so TLB mapped, cached applications should not be loaded.

Chapter 3

Memory Layout

This chapter describes YAMON's memory map. The memory map is shown in the following tables, and refer also to [1], [2], and [5].

The addresses of functions that YAMON provides to applications for printing data to the terminal, for cache flush, exiting the application, etc., are shown in Table 5.2. The actual functions are located at the base address named "YAMON functions" in the tables below.

Table 3.1 Memory Map (Physical Addresses) for Atlas™ and Malta™

Name	Atlas/Malta with Galileo System Controller	Malta with Bonito64 System Controller	Malta with MIPS SOC-it 101
SDRAM base	0x00000000 (Max 256MB)		
YAMON functions	0x00001000		
YAMON code	0x00005000		
PCI Memory space (as seen from CPU)	0x10000000 (128 MB) plus 0x18200000 (60 MB)	0x10000000 (192 MB)	0x10000000 (128 MB - v1.0) or 0x10000000 (176 MB - v1.1)
PCI Memory space (as seen on PCI)			
PCI I/O space (as seen from CPU)	0x18000000 (2 MB)	0x1fd00000 (1 MB)	0x1b000000 (8MB)
PCI I/O space (as seen from PCI)	Atlas: 0x18000000 Malta: 0x00000000	0x00000000	0x00000000
System controller	0x1be00000 (2 MB)	0x1fe00000 (256 kB)	0x1bc00000 (4 MB)
System flash	Atlas: 0x1c000000 (32 MB) Malta: N/A .		
Monitor flash	0x1e000000 (4 MB including 128 kB for Environment flash)		
Environment flash (environment variables)	0x1e3e0000 (128 kB)		
FPGA (interrupt controller, timer, LED display etc.)	0x1f000000 (12 MB)		
Bootcode base	0x1fc00000 (4 MB)	0x1fc00000 (1 MB)	0x1fc00000 (4 MB)
Vector table	0x1fc00500		
Little endian YAMON image	0x1fc10000		
Big endian YAMON image	0x1fc78000		

Table 3.2 Memory Map (Physical Addresses) for SEAD™

Name	SEAD / SEAD-2 with Basic RTL	SEAD-2 with MIPS SOC-it 101
SDRAM base	0x00000000 (Max 32 MB)	0x00000000 (256 MB)
YAMON functions	0x00001000	
YAMON code	0x00005000	
SDRAM controller	0x1b000000 (1 MB)	N/A
GPIO module	0x1b100000 (1 MB)	N/A
Performance module	0x1b200000 (1 MB)	N/A
System controller	N/A	0x1bc00000 (4 MB)
System flash	0x1c000000 (32 MB including 256 kB for Environment flash)	
Monitor flash	N/A	
Environment flash (environment variables)	0x1dfc0000 (256 kB)	
SRAM	0x1e000000 (4 MB)	
Peripheral controller registers	0x1e800000 (4 MB)	N/A
Peripheral bus devices	0x1f000000 (12 MB)	
Bootcode base	0x1fc00000 (1 MB)	
Vector table	0x1fc00500	
Little endian code	0x1fc10000	
Big endian code	0x1fc78000	

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” (0x1dc00000) based on jumper settings (see [1]).

Chapter 4

Board Specifics

This chapter describes the issues that are specific to the boards supported by YAMON.

4.1 Overview

“Break effect on tty0” describes the effect of an RS-232 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 be BRKRES register (address 0xbf000508).

Table 4.1 YAMON™ 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..512MB	32MB..512MB
Max SDRAM utilized.	256 MB	Depends on System Controller: Basic RTL: 32 MB MIPS SOC-it 101: 256 MB

Table 4.1 YAMON™ Board-specific Properties

	Atlas / Malta	SEAD / SEAD-2
Required SDRAM parameters.	Gallileo/Bonito64 system controller: No mixed-mode module configuration. Must support Burst length = 8 and CAS latency = 2	Basic RTL: 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.
	MIPS SOC-it 101 system controller: Mixed-mode module size supported, DDR supported, Parity supported. Requirement: SDRAM banks = 4.	MIPS SOC-it 101 system controller: Mixed-mode module size supported, DDR not connected, Parity not connected. Requirement: SDRAM banks = 4.

4.2 Upgrading YAMON on Atlas™/Malta™ Boards

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 `yamon-<rev>.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.

4.3 Upgrading YAMON on SEAD™ Board

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`

4.4 Upgrading YAMON on SEAD-2™ Board

YAMON is contained in flash. YAMON is updated using the USB port as described in [4]. The file required for programming the flash is `yamon-<rev>.fl`, contained in the zip file.

Chapter 5

Application Interface

This chapter describes how YAMON is used to load and execute user applications.

5.1 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 5.1 before jumping to the requested address (obtained from the previous `load` or as a parameter to `go`).

Table 5.1 Initial Application Context

Field	Value
`$4 (a0)`	Set to the argument count.
`$5 (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.
`$6 (a2)`	Pointer to table holding environment variables.
`$7 (a3)`	Size of memory (in bytes).
`$29 (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`).
`$31 (ra)`	Return address. Application may jump to this address in order to exit and return to YAMON.
Other CPU registers	0
FPU registers (if available)	0
CP0 *STATUS*	Same as YAMON context except that IE bit is cleared thus disabling interrupts.
CP0 *EPC*	Entry point of application obtained from `load` command or as a parameter to `go`.
Other CP0 registers	Identical to YAMON context.
FPU control registers (if available)	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 2.2.2, "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 2.3, "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 5.4, "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 initialise gp if using gp-relative addressing
- Interrupts are disabled (CP0 *STATUS* register *IE* bit is cleared).
- Caches are written back to memory if required.

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

5.2 Shadow Sets

For MIPS32/MIPS64 Release 2 CPUs supporting register shadow sets, shadow set 0 is used by YAMON. When the application is started, the "Current shadowset" will be 0. The application is allowed to change shadow sets, and YAMON will change the current set back to 0 when the application exits.

If a user application causes an exception, YAMON will dump the contents of the shadow set that was in use when the exception was taken.

5.3 Exit

Because YAMON resides in RAM, it is possible for an application to corrupt YAMON. When this happens, the following requirements must be met in order to return to YAMON:

- YAMON must not be overwritten. 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/CP1 control registers and the register banks (registers $0..$31 and the FPU general purpose registers) are stored/restored by the context shift mechanism used by YAMON. YAMON also manages shadow register sets (see Section 5.2, "Shadow Sets").

5.4 Functions

YAMON provides a set of functions (refer to Table 5.2) callable by applications. Pointers to these functions are 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`. There are no requirements for stack space, because the YAMON stack is used during the function call.

Table 5.2 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`
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).

The `syscon_read` function is used for access to various YAMON objects. Object IDs are defined in the header file `syscon_api.h` in the YAMON source distribution.

Note that MIPS32/MIPS64 Release 2 CPUs support disabling the CP0 *COUNT* register. If an application disables the counter, it is not allowed to access the following objects through the `syscon_read` function:

```
SYSCON_BOARD_GET_MILLISEC_ID
SYSCON_FILE_BATCH_ACCESS_ID
```

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.

5.5 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;
```

where:

`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.

5.6 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 built 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-2.96) or
  Algorithmics SDE-MIPS 4.0b or
  Algorithmics SDE-MIPS 4.1 or
  MIPS Technologies MIPS SDE Toolkit

- GNU gunzip tool

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

```
TOOLCHAIN = cygnus

ifeq ($(TOOLCHAIN),cygnus)
CC        = mipsisa32-elf-gcc
LD        = mipsisa32-elf-ld
OBJCOPY   = mipsisa32-elf-objcopy
OBJDUMP   = mipsisa32-elf-objdump
endif

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

You must modify the above code according to your setup.

By typing "`make`", the following files are generated:

- `appl.rec`
- `appl.elf`
- `appl.bin`
- `appl.map`
- `appl.dis`

where:

- `appl.rec` is the S-record file to be used for loading the application (using the "load" command).
- `appl.elf` contains the image in ELF format.
- `appl.bin` contains the image in binary format.
- `appl.map` is a linker generated map file for the application.
- `appl.dis` is the disassembled code.

Chapter 6

GDB Interface

This chapter describes YAMON's GDB interface.

6.1 Introduction

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.

6.2 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 2.2.2, "Shell Commands". The initial application context is the same as described for the `go` command (see Table 5.1).

Applications may register Exception Service Routines (ESRs) as described in Section 5.4, "Functions". However, the YAMON gdb stub must "own" the BREAK exception.

Setting breakpoints in exception-handling code is not supported.

GDB assumes 9600 baud by default, so either modify the baud rate used by GDB, or 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).
```

6.3 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 the 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, or 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 different commands). The data and checksum fields are case-insensitive.

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

When YAMON has performed the requested command, it transmits the following data to the GDB host (CSUM1, CSUM2 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 (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 four 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
```

6.4 GDB Remote Protocol Requests

The tables below list the requests supported by YAMON. The Format does not include $, #, and CSUM fields, and the replies assume that no errors occurred. Error messages are described in Chapter 9, "Diagnostics and Error Messages" on page 63.

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	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. The register sequence is the following: $0 .. $31 CP0 STATUS Special register LO Special register HI CP0 BADVADDR CP0 CAUSE CP0 EPC $fp0..$fp31 (ignored by YAMON) CP1 FSR (ignored by YAMON) CP1 FIR (ignored by YAMON) Frame pointer (ignored by YAMON) Dummy register (ignored by YAMON) 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
Reply	OK

Write 64-bit Registers	
Format	H <data> (data consists of 90*16 hex digits)
Description	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. The register sequence is the following: $0 .. $31 CP0 STATUS Special register LO Special register HI CP0 BADVADDR CP0 CAUSE CP0 EPC $fp0..$fp31 (ignored by YAMON) CP1 FSR (ignored by YAMON) CP1 FIR (ignored by YAMON) Frame pointer (ignored by YAMON) Dummy register (ignored by YAMON) 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
Reply	OK

Set Thread	
Format	Hg | Hc
Description	Ignored by YAMON
Reply	OK

Read Registers	
Format	g[S]
Description	Read registers. ‘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). Each register is represented by 4 or 8 bytes of register data each encoded as two hex digits. In total, 90 registers are read. The register sequence is the following: $0 .. $31 CP0 STATUS Special register LO Special register HI CP0 BADVADDR CP0 CAUSE CP0 EPC $fp0..$fp31 (set to 0) CP1 FSR (set to 0) CP1 FIR (set to 0) Frame pointer (set to 0) Dummy register (set to 0) 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
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	Read value of register ‘NN’ ‘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	<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 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 <adddress>
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

Chapter 7

Motorola S-records

This chapter describes the YAMON `load` command's support for 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.

<*checksum*> 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 <*checksum*> 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: In case of invalid S-record detection, the reported line number of the invalid record is calculated as follows:

- If no line feeds have been detected during the 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 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.

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

Table 7.1 summarizes YAMON’s S-record loader support.

Table 7.1 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 displays an example of a valid S-record file:

```
S00E00006170706C2E656C2E726563DD
S31A80100000F0FFBD270800BEAF21F0A0031000C4AF1400C5AF1836
S31A8010001500C6AF00BF013C000820AC00BF023C0808428C000020
S31A8010002A00000400C2AF0400C28F000000003B00432C03006054
S31A8010003F14000000001E0004080000000000BF013C000820AC08
S31A8010005400BF023C0808428C000000000400C2AF0400C28F005C
S31A80100069BF013C100422AC0C0004080000000211000002100A4
S31A8010007E04080000000021E8C0030800BE8F1000BD270800E0CE
S30A801000930300000000CF
S705801000006A
```

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

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

Chapter 8

Flash Support

This chapter describes YAMON's support for flash memory.

The areas of flash memory decoded by YAMON are shown in Table 8.1.

Table 8.1 Flash Memory Areas on Supported Boards

	Atlas	Malta	SEAD/SEAD-2
Total flash (System+Monitor+Environment) [MB]	36	4	32
System flash [MB]	32	N/A	31.75
System flash block size [kB]	256	N/A	256
System flash physical memory map	0x1C00.0000-0x1DFF.FFFF	N/A	0x1C00.0000-0x1DFB.FFFF
Monitor flash [MB]	3.875	3.875	N/A
Monitor flash block size [kB]	128	128	N/A
Monitor flash - YAMON boot image (1MB) (lock-bit protected) - phys. mem map	0x1E00.0000-0x1E0F.FFFF	0x1E00.0000-0x1E0F.FFFF	N/A
Monitor flash - user application (2.875 MB) - physical memory map	0x1E10.0000-0x1E3D.FFFF	0x1E10.0000-0x1E3D.FFFF	N/A
Environment flash [kB]	128	128	256
Environment flash block size [kB]	128	128	256
Environment flash physical memory 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 YAMON-boot image, and 3.875 MByte are free for user application.
- Environment flash is reserved for saving YAMON environment variables.

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

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.

- `disk`: When reading from IDE to flash, this command will program the flash with the source (harddisk/compact flash) contents. Address may be on any byte-boundary. The destination area must be erased before performing the operation. Caches are flushed before and after the disk operation.

- `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.

- `fread`: When transferring data to flash, this command will program the flash with the data. Address may be on any byte-boundary. The destination area must be erased before performing the operation

- `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 such as:

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

Flash-memory addresses may be specified as KSEG0 or KSEG1 (uncached) 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. For example:

```
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, such as:

```
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 contain the YAMON code.

Chapter 9

Diagnostics and Error Messages

This chapter describes the diagnostics messages written to the ASCII LED display during startup (in the order shown in the table).

Not all messages apply to all boards because of hardware dependencies.

Diagnostic messages are shown in Table 9.1 Most of these messages will not be displayed, unless there is an error during boot.

Table 9.1 ASCII LED Display Diagnostic Messages

Message	Description
Power On	Displayed by hardware if no core board is available or core board is not recognized by YAMON (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.
PRAM_CLR	SDRAM with parity: Clear all parity check bytes.
RAM	About to do basic RAM test.
RAM_HILO	Write test pattern to RAM Hi, middle, Lo addresses.
RAM_TEST	Perform test of RAM used by YAMON.
CLEAR	Clearing YAMON’s bss segment in RAM.
COPYTEXT	Copy YAMON’s text segments to RAM.
COPYDATA	Copy YAMON’s data segments to RAM.

Table 9.1 ASCII LED Display Diagnostic Messages (Continued)

Message	Description
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. HINT: If YAMON boot halts at this point, it could be that the PCI frequency is set too high for the Core board installed.
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.
FREQBUS	Detecting bus frequency.
PCI	Performing PCI auto configuration.
GT64120	Galileo GT64120 North Bridge detected.
BONITO64	Bonito64 North Bridge detected.
SOCit101	MIPS SOC-it 101 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”.
IDE	Initializing YAMON module "IDE".
OPTIMIZE	Optimizing North Bridge settings.
CPU_U	Updating CPU configuration based on environment variable `cpuconfig`. Only done on some boards (see Chapter 4, “Board Specifics” on page 39 and only done if CPU is configurable.
INITDONE	Initialisation done. Interrupts are now enabled.

Table 9.1 ASCII LED Display Diagnostic Messages (Continued)

Message	Description
<prompt>{<*version*>}	YAMON shell executing. This message may be selected using the “prompt” environment variable, which by default has the value “YAMON”. When "prompt" is equal to its default value, the display will also show YAMON’s major/minor revision numbers. Example: YAMON212.

The error messages shown in Table 9.1 are written to the ASCII LED display in case of errors. The messages beginning with “E:” are used during startup.

Table 9.2 ASCII LED Display Error Messages

Message	Description
Hex number	Exception occurred. Value represents EPC / ERROREPC .
NMI	NMI detected.
E:CPU	Unknown processor.
E:NO_RAM	RAM “Serial Presence Detect” device failed (probably missing SDRAM module).
E:RAM_WH	Illegal SDRAM width (must be 64 bit).
E:RAM_MB	Asymmetrical DIMM banks (not supported by Bonito64).
E:RAM_SZ	Illegal RAM size (>256 MB not supported by Bonito64).
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).
E:RAM_CF	General SDRAM configuration error.
E:RAM_HILO	Processor stuck while writing lowest, middle or high RAM address.
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_CW	PCI configuration write cycle failed.
E:NB_CR	PCI configuration read cycle failed.
E:NB_DEC	North Bridge setup (decoding) failed.
E:P_ALLO	Not enough space for PCI auto configuration

Table 9.2 ASCII LED Display Error Messages (Continued)

Message	Description
E:P_RANG	Illegal ram range for PCI mapping.
E:P_CFG	Unspecified error during PCI auto configuration.
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).
WRONGEND	The system failed to set endian desired endianness (should never happen)

Appendix A

References

1. YAMON™ Reference Manual
   MIPS document:MD00009

2. YAMON™ Errata
   MIPS document: MD00032

3. Atlas™ User's Manual
   MIPS document: MD00005

4. Malta™ User's Manual
   MIPS document: MD00048

5. SEAD™ Basic RTL User's Manual
   MIPS document: MD00017

6. SEAD-2™ Basic Package Getting Started
   MIPS document: MD00062

Appendix B

Revision History

Change bars (vertical lines) in the margins of this document indicate significant changes in the document since its last release. Change bars are removed for changes that are more than one revision old.

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	01/01/24	Document layout modified
02.02	01/07/27	Updated for YAMON 02.02
02.03	02/09/17	Updated for YAMON 02.03
02.04	02/11/21	Updated for YAMON 02.04
02.05	03/12/10	Updated for YAMON 02.05
02.06	04/03/24	Updated for YAMON 02.06
02.07	04/10/14	Updated for YAMON 02.07
02.08	05/03/31	Updated for YAMON 02.08
02.09	05/07/01	Updated for YAMON 02.09
02.10	05/10/14	Updated for YAMON 02.10
02.11	06/02/16	Updated for YAMON 02.11
02.12	06/07/04	Updated for YAMON 02.12
02.14	07/11/12	Updated for YAMON 02.14