YAMON™ Porting Requirements Specification

Document Number: MD00108
Revision 02.04
Nov 21, 2002

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

Template: S1.08, Build with Conditional Tags: 2B

Table of Contents

YAMON Porting Requirements

MIPS Technologies, Inc.distributes the source code of the YAMON monitor software thus enabling companies to port YAMON to their platforms. The modifications/additions performed by such companies may even be merged into future MIPS releases of YAMON enabling YAMON to support additional platforms or adding new YAMON features.

In order to ensure to avoid confusion concerning "what is YAMON ?", we define requirements for porting YAMON. In case these requirements are not adhered to, the resulting product shall not be called "YAMON".

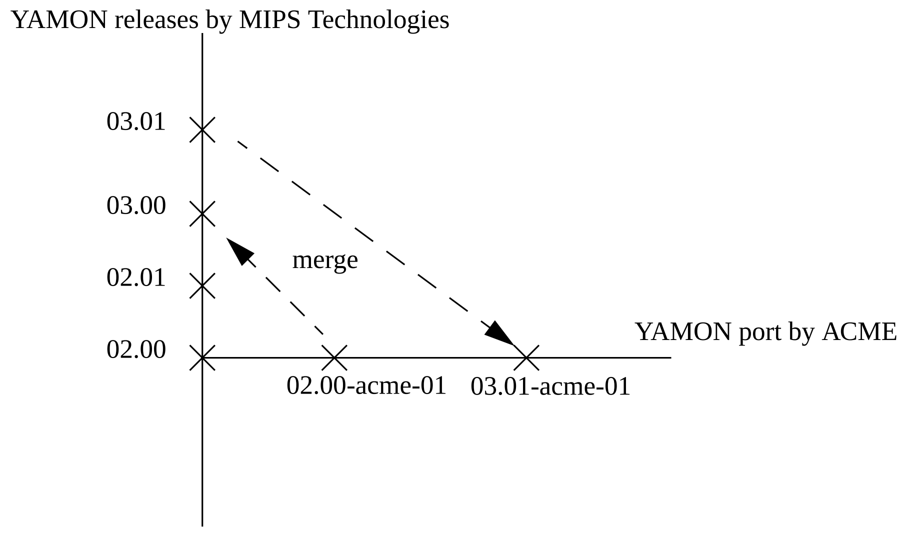
Also, we define recommended porting procedures, which will make it easier for the porting company to merge its ported YAMON with new YAMON releases by MIPS Technologies.

In the case where the modifications/additions made by the porting company are to be included in future YAMON releases by MIPS Technologies, the recommended rules become required rules.

Porting and merging is illustrated in figure 1. The figure illustrates the following steps :

- A (fictional) company "ACME" receives YAMON 02.00 source code.
- ACME ports YAMON to it’s platform and names the product YAMON 02.00-acme-01.
- MIPS Technologies releases YAMON 02.01.
- Some or all modifications/additions by ACME are included in the following release by MIPS Technologies (03.00).
- MIPS Technologies releases YAMON 03.01.
- ACME receives YAMON 03.01 and includes the modifications/additions in it’s next release (03.01-acme-01).

Figure 1 YAMON porting and merging.



This document describes the requirements/recommendations for porting YAMON to new platforms.

Rules are categorized as one of the following :

- Required. These rules must be followed in case the port is to be called "YAMON".
- Recommended. These rules need not be followed, but doing so will ease merging with new YAMON revisions. In case software source code is to be merged into future YAMON releases by MIPS Technologies, the recommended rules become required rules.

1 Revision handling

Required : The following convention must be used for naming YAMON :

```
<REVMAJ>.<REVMIN>-<COMPANY>-<REVISION>
```

REVMAJ and REVMIN are the major and minor revision numbers of the MIPS release of YAMON source used as base for the port.

COMPANY is the company name.

REVISION is the revision number of the ported YAMON version. This number is managed by the company performing the port.

Example (assuming YAMON 02.01 is ported by a company called "ACME") :

```
02.01-ACME-01
```

Required : The name must be used for at least the following purposes :

- Sign-on information from YAMON. This is controlled through the symbols REVMAJ and REVMIN defined in bin/makefile.
- Environment variable "yamonrev". This is controlled through the symbols REVMAJ and REVMIN defined in bin/makefile.
- Names of files used for distributing YAMON (binary files and source files).
  Example :
  ```
  yamon-bin-02.00-acme-01.zip
  yamon-src-02.00-acme-01.tar.gz
  ```

2 Look and feel

Required : The "look and feel" of YAMON must not be changed. By "look and feel" we mean the interfaces to YAMON. The 3 interfaces to YAMON are the shell (user interface), the GDB stub (debugger interface) and the application interface (vector table). Requirements for each of these interfaces are described in the following subsections.

2.1 Shell

Required : Shell commands may only be removed in case they are not supported due to hardware restrictions.

Required : Shell commands may be modified as long as changes are kept backward compatible. So, new options may be added, but the meaning of existing options must not be altered.

New shell commands may be added.

Recommended : Shell commands are removed/added through :

```
arch/shell/platform/shell_platform.c
```

The table below lists the commands included in YAMON 02.01 from MIPS Technologies. The second column indicates whether the command is required or may be removed when porting YAMON (in case platform does not support command).

Command	Required (y/n)
. (continuation command)	y
cache	y
cksum	y
compare	y
copy	y
cp0	y
date	n
dis	y
dump	y
echo	y
edit	y
eeprom	n
erase	n
fill	y
flush	y
fread	y
fwrite	y
gdb	y
go	y
help	y
info	y
load	y

Command	Required (y/n)
off	n
pcicfg	n
ping	n
port	y
reset	n
scpu	n
search	y
setenv	n
sleep	y
stty	y
test	y
tlb	n
unsetenv	n

2.2 GDB-stub

The GDB-stub is implemented in :

```
shell/gdb.c
```

Required : The GDB remote protocol consists of a set of commands. The set of commands supported by the “gdb” command must not be altered. In case GDB commands are modified/added/deleted in order to support a variant of the GDB Remote Protocol, this must be done in one of the following ways :

- By adding command line options to the “gdb” command.
- By adding a new command (e.g. “gdb<variant>”). The original "gdb" command must be retained.

2.3 Application interface

The application interface is described in [1,2]. As a historical note it may be observed, that the application interface for the first couple of vectors is compatible with that of pmon. As a technical detail it may be noted, that all the application interface vectors branch directly to an interface function in the file shell/appl_if.S, with the function code set up in register t0 in the branch delay slot. The interface function stores all registers, sets up a minimal YAMON context and calls directly into the appl_shell_func() routine in file shell/go.c.

Required : Existing functions must not be modified or removed.

Required : New functions must be added at the reserved vector locations. Vector locations reserved for this use are in the range from 0xbfc00600 to 0xbfc006fc.

Recommended : Adding a new function may be done following the steps :

- Allocate a function code in include/shell_api.h (Example : SHELL_FUNC_PRINT_CODE).
- Update the definition of SHELL_FUNC_COUNT in include/shell_api.h to span the new function code.
- Add an entry in vector table defined in arch/reset/bootvector/reset.S at the base (0xbfc00600) for vectors defined by companies porting YAMON
- Add support for the function in file shell/go.c

3 Conditional compilation

Recommended : Modifications to YAMON should be done using conditional compilation. We recommend using the following naming convention for the symbol used for conditional compilation :

```
_ARCH_<COMPANY>_<PLATFORM>
```

<COMPANY> is replaced with the name of the company.

<PLATFORM> is replaced with the name of the platform.

Example (company = “ACME”, platform = “TRAP”) :

```
#ifdef _ARCH_ACME_TRAP
... code
#endif /* #ifdef _ARCH_ACME_TRAP */
```

4 Directory structure.

Recommended : Platform specific code should be kept in the arch directory.

The arch directory includes subdirectories for platform specific features. Each of these directories include one or more of the following subdirectories :

- platform
- platform/core
- cpu

The "platform" directories hold board specifics (see chapter 5).

The "platform/core" directories hold “Core card" specifics (see chapter 6). A Core card is defined for the Atlas and Malta platforms from MIPS Technologies. A Core card holds the CPU, the North Bridge controller (memory controller and PCI bridge) as well as the SDRAM.

The "cpu" directories hold cpu specifics (see chapter 7).

5 Product ID

The boards supplied by MIPS Technologies use the address 0xbfc00010 for the “REVISION” register used for board identification. The format of the REVISON register is :

Table 1 REVISION register layout

Bits	field name	Function
31:8	reserved	Board - specific extensions.
7:4	PROID	Identifies the basic system (motherboard) type
3:0	PRORV	Identifies the basis system (motherboard) revision

The following table defines the PROID values assigned to the various motherboard types.

Table 2 PROID field assignments

Board	PROID
Atlas	0x0
SEAD	0x1
Malta	0x2
SEAD2	0x3
Third-party product	0xE
reserved	0xF

The value 0xE is allocated for third-party products ("Non MIPS products"). The layout of bits 31:8 are different for "Non MIPS products" as shown below :

Table 3 3rd party REVISION register layout

Bits	Field name	Function	Initial value
31:24	reserved		0x0
23:16	MANPD	Manufacturer’s product ID	-
15:8	MANID	Manufacturer ID code	-
7:4	PROID	Basic system type	0xE
3:0	PRORV	Product revision	-

Throughout YAMON, board specific code is executed based on the global variable :

```
sys_platform
```

sys_platform holds the value of the PROID field. The value 0xE is reserved for "Non MIPS products".

The specific product is identified by the MANID/MANPD fields of the REVISION register (in case of "Non MIPS" products).

MANID field is allocated by MIPS Technologies, while MANPD is allocated by the company.

Recommended : A platform needs not support the REVISION register, but the YAMON port must set the following variables :

- sys_platform (should be set to 0xE).
- sys_manid (allocated by MIPS Technologies).
- sys_manpd (allocated by company owning board).

The above variables are set in file :

```
arch/reset/init.S
```

Note, that sys_manid and sys_manpd were added in YAMON 02.01. When porting earlier YAMON revisions, the variables must be added by the company performing the port. This also applies to the symbol PRODUCT_THIRD_PARTY_ID used in the example below. Symbols identifying the products should be added to the file :

```
arch/include/product.h
```

The following example illustrates a case where support for some platforms is removed, while support for a new platform “TRAP” from the company “ACME” is added :

```
switch( sys_platform )
{
#ifndef _ARCH_ACME_TRAP /* Code not used in ACME port */

  case PRODUCT_ATLASA_ID :
    /* Atlas specific code */
    ...
    break;
  case PRODUCT_MALTA_ID :
    /* Malta specific code */
    ...
    break;
  case PRODUCT_SEAD_ID  :
    /* SEAD-1 specific code */
    ...
    break;
  case PRODUCT_SEAD2_ID :
    /* SEAD-2 specific code */
    ...
    break;

#else

  case PRODUCT_THIRD_PARTY_ID :
    /* Non MIPS product */

    switch( sys_manid ) /* Select manufacturer */
    {
      case MANID_ACME :
        /* ACME product */
        switch( sys_manpd ) /* Select platform */
        {
          case MANPD_TRAP :
            /* ACME Trap specific code */
            ...
            break;
```

```
                default :
                  break;
              }
              break;
            default :
              break;
          }

      #endif /* #ifndef _ARCH_ACME_TRAP */

        default :
          break;
      }
```

A YAMON port may support one or more platforms depending on the symbols defined at compile time.

6 Core card ID

A "Core card" is a hardware module used on the Malta and Atlas platforms from MIPS Technologies. A Core card includes the CPU and the Northbridge (memory controller and PCI bridge) as well as SDRAM. This section only applies when adding support for a new Core card for Malta and Atlas.

Malta and Atlas encode bits 31:8 of the “REVISION” register (see chapter 5) the following way :

Table 4 Atlas/Malta REVISION register layout

Bits	field name	Function
31:16	reserved	reads as 0
15:10	CORID	Identifies the Core card type.
9:8	CORRV	Identifies the Core card revision.

MIPS Technologies allocates IDs for Core cards for Malta and Atlas.

The following variable is used for the Core card ID :

```
sys_corecard
```

The above variable is set in file :

```
arch/reset/init.S
```

Core card IDs are allocated in the file :

```
arch/include/product.h
```

7 Processor ID

Recommended : Processors are identified using the “Company ID” and “Processor ID” fields of the CP0 PRID register. The contents of these two fields are masked and stored in the global variable :

```
sys_processor
```

sys_processor is set in the file :

```
arch/reset/init.S
```

New processors may be defined in the file :

```
arch/include/mips.h
```

Example :

```
#define MIPS_4Kc        ( (C0_PRID_COMP_MIPS <<   \
                            C0_PRID_COMP_SHF) |  \
                           (C0_PRID_PRID_4Kc  << \
                            C0_PRID_PRID_SHF) \
                          )
```

8 Drivers

Drivers not relevant for a particular platform may be removed as described in [2].

New drivers may be added.

Recommended : Each driver is identified by major and minor device numbers as described in [2]. New major numbers may be allocated in file :

```
include/sysdev.h
```

Recommended : Driver major numbers should be referenced using symbols rather than absolute values in order to avoid problems when merging with new YAMON releases.

9 SYSCON objects

Many parts of the code, in particular the shell and the drivers, do not have any information of the specific platform, but acquires platform specific data (for example memory mapping of devices, serial port data etc.) and other centralized system data through a “System Configuration” (SYSCON) module (see [2]).

SYSCON objects are defined by the enumerated type :

```
t_syscon_ids
```

defined in file :

```
arch/include/syscon_api.h
```

Recommended : New objects may be added, but should always be added after the existing objects. Software should reference the objects using symbols rather than absolute values in order to avoid problems when merging with new YAMON releases.

10 Modules

YAMON modules may be removed if not applicable for the particular platform. New modules may be added. Conditional compilation should be used.

This is done as described in [2].

11 Bugfixes

Bugfixes are allowed. Like any other changes, we recommend to use conditional compilation. We encourage the following additional procedures :

- Comment the bugfix using the following style :

```
/* BUGFIX !
 *      Description : <Description of bugfix>
 */
```

- Report the bug to MIPS Technologies (support@mips.com). Please attach the file that was fixed.

12 Removing code

Recommended : Code not relevant for a particular platform may be removed. However, this should be done using conditional compilation.

13 Endianness

YAMON supports run-time detection of endianness. This feature may be removed for platforms not supporting both little- and big-endian or simply in order to save flash/eprom space. Limiting YAMON to one endianness will reduce the image size by appr. 50 %.

The makefile used for building YAMON (bin/makefile) will build YAMON in both big- and little-endian and concatenate the two images. Furthermore, endianness independent reset code is built. This code detects endianness and jumps to the corresponding image. So, if endianness is restricted to only little- or only big-endian, the make files bin/Makefile, fpuemul/Makefile and the reset code (arch/reset/bootvector/reset.S) should be modified.

14 Legal headers and related issues.

14.1 Copyright notice in YAMON sign-on text

This is to be defined.

14.2 Legal header in modified YAMON source files

This is to be defined

14.3 Legal header in new source files

This is to be defined.

References

[1] YAMON(tm) User’s Manual,
MD00008

[2] YAMON(tm) Reference Manual,
MD00009

A Revision History

In the left hand page margins of this document you may find vertical change bars to note the location of significant changes to this document since its last release. Significant changes are defined as those which you should take note of as you use the MIPS IP. Changes to correct grammar, spelling errors or similar may or may not be noted with change bars. Change bars will be removed for changes which are more than one revision old.

Please note: Limitations on the authoring tools make it difficult to place change bars on changes to figures. Change bars on figure titles are used to denote a potential change in the figure itself.

Revision	Date	Description
01.00	December 12, 2000	Initial revision
01.01	November 7, 2001	Updated to latest template
02.03	September 17, 2002	Updated to YAMON v2.03
02.04	November 21, 2002	Updated to YAMON v2.04