Freescale Semiconductor
Application Note

AN2738
Rev. 0, 07/2004

Genesi Pegasos II Firmware

by *Maurie Ommerman*
CPD Applications
Freescale Semiconductor, Inc.
Austin, TX

This application note is the third in a series describing the Genesi Pegasos II system, which contains a PowerPC™ microprocessor, and the various applications of the system.

Contents

1 Introduction

This document describes the firmware used by the Genesi Pegasos II system including some history, some philosophy, and a discussion of many useful commands. The original SmartFirmware users manual was consulted for this document and is available on the MorphOS partition, a printed copy is suppled in this document package. See Section 7, “References,” *SmartFirmware Users Manual.*

2 Terminology

The following terms are used in this document.

Linux OS	Linux Operating system
Debian	One of the flavors of Linux
Yellow Dog	One of the flavors of Linux
Open Firmware	A standard for firmware, IEEE Standard 1275-1994.
SmartFirmware	Specific implementation of Open Firmware used in Genesi Pegasos II.

freescale™ semiconductor

Firmware	The code associated with booting and starting the motherboard

3 What Is Firmware?

Firmware is the first code executed after power on. Dink32, U-boot, Open Firmware, and x86 BIOS are all firmware. Dink32 is proprietary software owned by Freescale, available for free under Freescale license exclusively for Sandpoint.

U-boot is GPL software available free under GPL license for many boards, including ADS, Arcadia, and others.

Open Firmware is implementation proprietary software available for many boards, including ADS, Arcadia, and others.

3.1 What Does Firmware Do?

Firmware is specific to a particular processor family. When the firmware starts, it performs these actions:

- Queries the board
- Determines the processor
- Discovers the memory controller
- Sets the chip set registers to a known state
- Enumerates PCI, serial port, and other board features
- Can boot an operating system automatically

After the firmware boot gains control, the user can do the following:

- Query the board and peripherals
- Set and see non-volatile variables
- Set and see various processor and chipset registers
- Boot into an operating system

3.2 Why Are There So Many Firmwares?

- There are many variants of Windows.
- There are many variants of Linux.
- There are many variants of real time OS.
- There is competition among vendors.

Each has their advantages and uses:

- Dink32 gives Freescale ownership of Sandpoint firmware.
- U-boot is fairly ubiquitous and free within GPL rules and everyone can add and debug it.
- Open Firmware is maintained by a commercial company, CodeGen, and supported by Genesi. The standard is open, the implementation is proprietary.

4 Open Firmware Help

The firmware has several help screens, which are cryptic, but are a good tool for remembering commands and how to use them. The Open Firmware prompt is `ok`. The user may type in any Open Firmware command after the `ok` prompt.

4.1 General Information

This version of Open Firmware is an implementation of the OpenFirmware IEEE Standard 1275-1994 plus errata changes. It is officially called SmartFirmware. Much of the information in this document is from 7, “References,” *SMUserManual.pdf*. Information on how to obtain this pdf is given in the application note *Genesi Pegasos II Debian Linux* (AN2639). The firmware is a Forth engine, essentially a stack-based language, much like Hewlett-Packard calculators, that is, reverse polish. For instance to add two numbers type, “3 4 + .”, which puts 3 then 4 on the stack, adds them, and then pops them off the stack, that is, displays them.

These usability commands are available:

1. Some of the help screens will overflow the terminal window. To allow all command responses to scroll, so they display a few lines and wait for a keypress to continue, use this command to specify the number of lines per page of display.
   - `18 to lines/page` will display 18 lines and wait for a key press.
   - `-1 to lines/page` will display the maximum lines -1 and wait for a key press.
2. Command line history is invoked with the up and down arrows on the keyboard
3. Tab-completion will complete any commands after a minimum of unique characters are typed followed by the Tab key.
4. Setting the number of rows and columns for the screen can be done, however, these commands seem to have no affect.
   - `setenv screen-#columns n`, where *n* = 0 indicates maximum size
   - `setenv screen-#rows n`, where *n* = 0 indicates maximum size
5. To stop any firmware command from continuing type in Ctrl+c.
6. To change the size of the screen display, use the F6 through F9 key.
7. If the firmware locks up and no responses are possible, push the reset button.
8. `help` gives general help, specifically indicating what help submenus are available. help command, gives more specific help for that particular command. Example: help booting displays all the booting parameters, help boot, displays the boot command, although it is very cryptic.
   - `help`
   - `help booting`
   - `help boot`
9. To display any forth variable use the ‘.’ dot command
   ok lines/page. <cr>

Here is an example of the use of the help command:

```
ok: Help

Help category/Forth-word

        where category is one of:
```

```
booting devices nvram vn-vars testing debugging bplan
```

5 Open Firmware Commands

5.1 help bplan

Displays the copyright date.

5.2 help nvram

Displays the commands available for manipulating the non-volatile variables, that is, persistent variables. Persistent variables determine the actions at boot up. They all have a default value that can be reset at any time, or they can be changed to any value, such as a text string value pair.

1. `printenv [var]` is the most useful, it will print all the nvram variables or just a specified one.
   - `printenv` prints all the variables, be sure to set "-1 to lines/page" first.
   - `printenv boot-device` prints just the boot-device string.
   - `printenv boot-command` prints the boot-command string.
   - There are many non-volatile variables and most are self-evident. The important ones are the following:
     - client-ip and server-ip when booting from ethernet
     - diag-switch, diag-file, and diag-device when running diagnostics on the firmware
     - boot-device, boot-file, boot-command, auto-boot-timeout, and auto-boot are variables which collectively specify what the firmware will boot into at start up. These are the non-volatile variables for this case:
       - boot-device is set to ide:0, the first partition on the first IDE channel.
       - boot-file is set to menu, a forth script that displays the menu.
       - boot-command is set to boot, specifying to boot into the boot-file on the boot-device.
       - auto-boot-timeout is set to 500, wait 25 seconds
       - auto-boot is set to true, boot into the menu at start up time.
2. `setenv` will set an nvram variable to a value.
   `setenv var stringvalue`
   - `setenv boot menu` There is no '=' in this command; separators are blanks.
   - `setenv auto-boot? true` True/false variables include the '?' in their name.
3. `set-defaults` will quickly set all variables back to their defaults.
4. `set-default [var]` Resets just the variable specified.
   - `set-default boot-file` Will reset boot-file to linux.eth.
5. `nvedit` and `nvstore` modify the nvramrc script, described in Section 5.7, "help booting" step 3.

5.3 help nv-vars

This command will list all the non-volatile variables and their defaults, which are set by the `set-defaults` command.

5.4 help testing

This is a facility to test out the devices on the Genesi Pegasos II system. It generates more extensive diagnostics when the variable “diag-switch?” is true. This facility is rarely used. Below are the test commands:

- `test [device]`

This command runs a test on a specific device. A command example is `test ide`

- `test-all [device]`

This command runs a test on this device and all it’s subordinate devices. A command example is `test-all ide`

5.5 help debugging

These commands can print out a lot of internal information. Most of these commands are useful for debugging the hardware. Some of these commands follow:

`cpustat` shows the important CPU registers.

`dump-env` shows the open firmware environment.

`dump-all` shows the open firmware state dump, that is, what every variable and device status or value or location.

`dump address length` displays hex and ascii value of all memory from address for length. example: `dump 100000 100`, dump memory starting at 0x100000 for 0x100 bytes.

`mem-stats` displays the memory statistics.

`dump-shipset` should display the important chipset registers. Don’t try it, it will hang.

5.6 help devices

A device path is like a Unix (Linux) file path: /device@address:options/dev2/... The address and options may be omitted. The familiar Unix/Linux commands are available for navigating the device tree.

1. `show-devs` —This non-Unix like command displays all the devices available on the system. Devices are packages, CPUs, PCI devices and others, in short every physical and logical device on the system.
   - `show-devs cpus` shows all the CPU info, its type and caches.
   - `show-devs /packages` shows various packages that are available, such as terminal-emulator or deblocker.
   - `show-devs ide` shows the hard drives, which is more useful (see `devalias` below).
2. Navigate to these directories and subdirectories with the `cd` command and display them with the `ls` command.
   - `cd /cpus`
   - `ls`
   - `cd PowerPC,74x7`
   - `ls` —to see the caches
3. The most useful navigation is with the file system on the master IDE drive on IDE channel 0, or any of the other drives on this system. In this case, there is only this one drive.
4. `devalias` indicates what the default aliases are. This can save a lot of typing.

— `devalias` shows us that IDE is the same as /pci@80000000/ide@C,1/disk@0,0
`cd ide`
`ls` shows all the partitions on this device.

5. `.properties` (note the preceding dot) gives all the properties of the current device.
— `cd ide`
`.properties` gives us the name, device_type, position, IDE channel, master/slave, partition, and type, such as ATA.
6. `words` shows all the methods of the current device.
— `cd ide`
`words` gives us open close, read-blocks, etc.
7. Changing to a partition directly is not allowed. Change to the partition table with the following commands:
– `cd ide`
— `ls` displays all the partitions as listed below.

```
RDB partition 0 <FFS>: <boot> (0x444F5301)

RDB partition 1 <SFSS>:<MOS> (0x53465300)

RDB partition 2 <SFS>: <MOS-DATA> (0x53415300)

RDB partition 3 <LNX>: <swap> (0x4C4E5800)

RDB partition 4 <LNX>: <debian> (0x4C4E5800)

RDB partition 5 <LNX>: <YDL> (0x4C4E5800)
```

8. List the files on a partition with the `ls` command by specifying the hard drive and the partition. However, reading or modifying any of these files is not allowed.
ls /pci/ide/disk@a,b:c, where *a* is the IDE channel, *b* is the master (0) or the slave (1), and *c* is the partition. All files and directories will be listed, directories are indicated as such and show the size.
— `ls /pci/ide/disk@0,0:0` will list all the files on the first (zero) partition
— `ls /pci/ide/disk@0,0:1` will list all the files on the second (one) partition.
— `ls /pci/ide/disk@0,0:1 utilities.info` will display the size of the file.
— `ls /pci/ide/disk@0,0:1 MorphOS` will list the files and subdirectories under MorphOS.

Aliases can be used in place of specific disks (see `devalias` above)

— `ls ide:4` is equivalent to `ls /pci/ide/disk@0,0:4` which lists all the files and directories on partition 4.
— `ls ide:4 /boot` lists the files and subdirectories under /boot. An example of the display with this command is shown below:

```
System.map-2.4.22-powerpc

System.map-2.4.25-powerpc

System.map-2.6.4-pegasos

config-2.4.22-powerpc

config-2.4.25-powerpc

config-2.6.4-pegasos
```

```
first.b
patches-2.4.22-powerpc
patches-2.4.25-powerpc
second.b
vmlinuz-2.4.22-powerpc
vmlinuz-2.4.25-powerpc
vmlinuz-2.6.4-pegasos
```

5.7 help booting

Displays the commands available for booting.

Booting performs these operations, according to the manual:

1. Power-on-self-test
2. System initialization
3. Evaluate the script “nvramrc” if “use-nvramrc?” is true
   — In this case, “use-nvramrc” is true, but since no ftp boot, it fails and step 4 begins.
4. If the script was not executed or if there was no console after the script finished executing, then:
   a) Execute `probe-all` (evaluates FCode)
   — This command should only be executed once, subsequent calls will hang the system.
   b) Execute `install-console`
   — This command should only be executed once, subsequent calls will hang the system
   c) Execute `banner`
   d) Secondary diagnostics, if `secondary-diag` is true, and other system-dependent initialization.
   — In this case it is false.
   e) Run boot command, if `auto-boot?` is true and no key is held down.
   — In this case it is true and the boot-command is `boot`, and boot-file is menu. The menu script is run from the /boot partition, which is the 0 partition on the 0 IDE channel.
   — If the user chooses option 5 in the menu, then f) below is run.
   f) Run the Forth command interpreter (if not booted)

The boot command can be used to boot any acceptable executable and supply arguments using this form:

`boot [bootable kernel] [arguments for the kernel]` —where the bootable kernel is an elf image, such as a linux kernel, or an amiga image.

For example: `boot ide:4 boot/vmlinuz-2.6.4-pegasos root=/dev/hda5`

—indicates to boot the linux kernel, vmlinuz-2.6.4-pegasos in the /boot directory, on the fourth partition of the first IDE channel master device (remember IDE is an alias for /pci/ide/disk@0,0), and supply the parameter “root=/dev/hda5” to the kernel, which will tell linux to get its root file system from partition 4 (the fifth partition counting from zero).

Another example from the file shown in Section 6, “Boot-file menu”: `boot ide:0 boot.img`

—indicates to boot the MorphOS kernel boot.img from the root of the first partition on the first IDE channel on the master drive and supply no arguments.

6 Boot-file menu

The boot-file menu is on the /boot, that is, partition 0 of IDE 0 master drive. This menu can only be accessed in Debian Linux because firmware does not have the ability to list or edit these file contents. Using the `mount` command, as in the example below, allows access to the files.

`mount -t affs /dev/hda1 /mnt/temp1` —where `affs` is the amiga fast file system type.

`cat menu`

The file is shown below, the numbers are not part of the forth language and are supplied here to help in the explanation of the file. To change this file, be careful only to change the lines explained here, otherwise it may no longer work.

Line numbers:

- 12: `my-max-boot-num` is the number of items displayed
- 13: `my-boot-default` is the default choice
- 14: `my-boot-delay` is a delay value, I'm not sure how this correlates with the boot-boot-timeout variable.They are used for the wait loop in lines 38-75 and referenced in line 80
- 15-26 is the boot menu that is displayed.
- 27-35 is the command issued for each option chosen, denoted by the case.
  example: 1 is for MorphOS, command is boot ide:0 boot.img, boot the boot.img file on IDE 0, partition 0, which is the MorphOS boot image.
- example: 3 is for debian kernel 2.6, command is boot ide:4, directory /boot, kernel file vmlinuz-2.6.25-powerpc, using the parameters to linux of: root=/dev/hda5 hdc=ide-scsi.
  This will start 2.6 debian linux.
- Changing any of these lines in 20-24, require a corresponding change in lines 30-34. Thus, this menu can be easily changed to specify any number of boot options.
- Without this menu facility, the firmware user would be forced to type in the boot command each time see 5.7, "help booting".
- 38 -75 implements the wait time for a key stroke.

```
1  \ FORTH is identifed by a forth comment at first line
   2  \
   3  \ terminal control stuff
   4  \
   5  : TTY.CSI d# 27 EMIT ASCII [ EMIT ;
   6  : TTY.HOME    TTY.CSI ASCII H EMIT ;
   7  : TTY.CLR_EOS TTY.CSI ASCII J EMIT ;
   8  : TTY.HOME_CLR TTY.HOME TTY.CLR_EOS ;
   9  \
```

```
10  \ boot menu stuff
11  \
12  : my-max-boot-num 5 ;
13  : my-boot-default 3 ;
14  : my-boot-delay d# 300 ; \ unit = 100 ms
15  : my-print-menu ( -- )
16    TTY.HOME_CLR
17    ."  "                                    cr
18    ." Pegasos boot menu"                    cr
19    ."  "                                    cr
20    ." 1: MorphOS"                           cr
21    ." 2: Debian GNU/Linux 2.4 kernel"       cr
22    ." 3: Debian GNU/Linux 2.6 kernel"       cr
23    ." 4: Yellow Dog Linux 2.4 kernel"       cr
24    ." 5: return to OF prompt"               cr
25    ."  "                                    cr
26  ;
27  : my-boot-case ( num -- )
28    ."  " cr
29    case
30      1 of " ide:0 boot.img"      endof
31      2 of " ide:4 boot/vmlinuz-2.4.25-powerpc root=/dev/hda5 hdc=ide-scsi"
endof
32      3 of " ide:4 boot/vmlinuz-2.6.4-pegasos root=/dev/hda5"     endof
33      4 of " ide:0 vmlinuz-2.4.24-pegasos root=/dev/hda6 hdc=ide-scsi
video=radeon:1024x768"      endof
34      5 of " none" endof
35    endcase
36    $boot
37  ;
38  : my-input-num ( wait-period max-boot-num default-num -- boot-num )
39    1 \ loop-inc = 1
40    3 pick 0 do
```

```
41      0d emit
42      ." press 1-"
43      ( wait-period max-boot-num default-num loop-inc )
44      2 pick ascii 0 + emit
45      dup 1 = if
46        ."  within "
47        3 pick i - d# 10 / .d
48        ."  seconds"
49      then
50      ."  (default: "
51      over ascii 0 + emit
52      ." ) :                    "
53      d# 100 ms
54      key? if
55         key
56         ( wait-period max-boot-num default-num loop-inc key )
57         dup 0d = if \ return pressed
58           drop leave
59         then
60
61         ascii 0 -
62         ( wait-period max-boot-num default-num loop-inc num )
63         dup 1 5 pick
64        ( wait-period max-boot-num default-num loop-inc num num 1 max-boot-num )
65         between if
66           rot drop swap leave
67         then
68
69         ( wait-period max-boot-num default-num loop-inc num )
70         2drop 0  \ loop-inc = 0
71      then
72    dup +loop
```

```
    73    drop
    74    ( wait-period max-boot-num boot-num )
    75    nip nip
    76  ;
    77
    78
    79  my-print-menu
80  my-boot-delay my-max-boot-num my-boot-default my-input-num
    81  my-boot-case
```

7 References

The following documents describe the various applications of the Genesi Pegasos II system.

1. Freescale application note AN2666, *Genesi Pegasos II Setup*
2. Freescale application note AN2736, *Genesi Pegasos II Boot Options*
3. Freescale application note AN2739, *Genesi Pegasos II Debian Linux*
4. Freescale application note AN2744, *PMON Module, an Example of Writing Kernel Module Code for Debian 2.6 on Genesi Pegasos II*
5. Freescale application note AN2743, *Software Analysis on Genesi Pegasos II Using PMON and Altivec*
6. *SmartFirmware User Manual -SFUserManual.pdf, SmartFirmware.pdf, and PegasosFirmware.txt-* available on the MorphOS partition and supplied with this application note.

For assistance or answers to any question on the information that is presented in this document, send an e-mail to risc10@freescale.com.

8 Document Revision History

Table 1 provides a revision history for this application note.

Table 1. Document Revision History

Revision Number	Date	Change(s)
0	07/14/04	Initial release.

How to Reach Us:

USA/Europe/Locations Not Listed:
Freescale Literature Distribution
P.O. Box 5405,
Denver, Colorado 80217
1-480-768-2130
(800)-521-6274

Japan:
Freescale Semiconductor Japan Ltd.
SPS, Technical Information Center
3-20-1, Minami-Azabu, Minato-ku
Tokyo 106-8573, Japan
81-3-3440-3569

Asia/Pacific:
Freescale Semiconductor H.K. Ltd.
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T. Hong Kong
852-26668334

Learn More:
For more information about Freescale Semiconductor products, please visit
http://www.freescale.com

AN2738
Rev. 0
07/2004

freescale™
semiconductor