SmartFirmware User Manual

This document describes the SmartFirmware User Interface, and how to use it to manipulate device trees, view and set parameter values, and to control boot-up procedures.

Basics

SmartFirmware is an implementation of the OpenFirmware IEEE standard 1275-1994 plus errata changes.

The basis of SmartFirmware is a Forth engine. Forth is essentially a stack-based language, much like the well-known Hewlett-Packard calculators. Forth "words" are names that are executed when they are typed in at the "ok" prompt.

For instance, to add two numbers, type "3 4 +. cr" at the prompt. This sequence first pushes the number '3' then the number '4' onto the stack. The '+' then pops and adds the top two stack elements and pushes the result on the stack. The "." pops and displays the sum on the top of the stack, namely the value '7'. The "cr" simply displays a carriage-return and doesn't affect the stack.

This document expands upon the built-in "help" Forth word.

Devices

A device is a node in a tree, much as a file in a Unix file-path. A device name is usually of the form "/name@addr1,addr2:options/dev2/...". All the portions of a device path are machine-dependent. An example of a path is "/duart@C0800A00/uart@1:19200,9600,8,1,N". The "@addr" and ":options" portions of the name may be left out, in which case the first matching name is used ("/duart/uart" for the previous example).

Devices may also have aliases for convenience. For instance, the alias "tty" may refer to "/duart@C0800A00/uart@0", and may then be referred to with "tty:19200,9600,8,1,N" with options if desired.

Displaying

Devices can be listed using the "ls" command from the SmartFirmware User Interface. The "dev" or "cd" commands (which are synonyms) moves around the device tree. The "pwd" command shows the current device being accessed.

The command "show-devs" lists all devices in the device tree. The command ".properties" can be used to display all property lists at any node in the device tree, after using "dev" to first open that device.

pwd

Print current (working) device path.

ls

Show all devices at the current point in the device tree.

dev <device path>

cd <device path>

Change the currently open device to the one specified on the command-line. The path may be relative to the current device. The path may also be the string ".." to open the parent of the current device. The device will also parse any optional arguments to configure the device, if any are provided.

Example: cd /duart@C0800A00:9600

show-devs

Show all the devices in the device tree.

.properties

Show all the properties of the current device.

words

Show all the methods of the current device, or all the Forth words if there is no current device.

devalias [name string]

Show all the device aliases currently defined if nothing is on the command line. Otherwise, create a device alias called "name" with the value of "string". This is handy to avoid typing in long path names again and again. Useful in "nvramrc" scripts (below).

Example: devalias tty /duart@C0800A00/uart@0

Parameter settings

Parameter settings are stored in non-volatile memory (NVRAM). They affect a variety of SmartFirmware activities from bootup to running custom scripts to setting the size of the display window.

Parameters are simply string value pairs that usually have a default value (stored in ROM) and an active value (stored in NVRAM). They may be displayed, set, and restored using the following commands.

The supported parameter variables are described in the next section.

The following commands are used to manipulate NVRAM parameter settings:

printenv [var]

Display all parameter variables, their current values, and their default values if no "var" is specified on the command-line. Otherwise just display the values for that variable.

Example: printenv diag-switch?

setenv var string value

Set the value of "var" to be "string value". Everything up to the end of the line, including spaces, will be stored in NVRAM.

Example: setenv diag-switch? true

set-default var

Set the value of "var" to be its default value. This is handy to restore some variable whose value may have gotten scrambled for some reason.

set-defaults

Set all parameter values back to their default values. This is handy if NVRAM has become corrupted.

nvedit

Edit the script stored in parameter "nvramrc". This opens up a very simple Emacs-style text editor with the contents of the "nvramrc" variable.

Type ^C (Control-C) to exit. Most of the usual Emacs commands may be used to move the cursor around, including ^F (forward-char), ^B (backward-char), ^N (next-line), and ^P (previous-line).

This command does not alter the contents of "nvramrc" but merely edits a copy in memory. See "nvstore", "nvquit", "nvrecover", and "nvrun" (below).

nvstore

Stores the contents of the current "nvedit" buffer into the parameter variable "nvramrc". See "nvedit" (above).

nvquit

Erase the contents of the nvedit buffer without storing it. Prompts the user to confirm before erasing the buffer. Does not alter the contents of "nvramrc". See "nvedit" (above).

nvrecover

Attempts to recover the contents of nvedit after it has been erased. Normally "nvstore" is supposed to erase the contents of the edit buffer. SmartFirmware does not do this, so nvrecover is effectively non-functional. See "nvedit" and "nvstore" (above).

nvrun

Execute the contents of the nvedit buffer as Forth code. Handy way to make sure that the "nvramrc" script works before committing it to NVRAM. See "nvedit" (above).

nvalias alias device

Add a "devalias" to the "nvramrc" script to create this alias at bootup. If there is already a "devalias" command there, edit it to point to this new device. Finally, run the new "devalias" command so the alias is available for immediate use. This is a short-cut operation for most typical uses of "nvramrc".

Example: nvalias tty /duart@C0800A00/uart@0

nvunalias alias

Remove the specified "devalias" command for creating this alias at bootup from the "nvramrc" script, if there is one there.

Parameter variables

Parameter variables are manipulated by the commands described in the previous section. Most are used during bootup and are guarded by boolean parameters to turn features on or off. All these variables may not be present on all systems. For instance, if the system does not support booting, none of the boot parameter vars would be of much use.

diag-switch?

Boolean: "true" or "false". Switch on diagnostic-mode if true, which turns on extended tests, more verbose output, and alters the boot commands below.

secondary-diag?

Boolean: "true" or "false". Switch secondary-diagnostics on or off at bootup. This is not a standard OpenFirmware parameter and controls additional comprehensive test routines for SmartFirmware.

auto-boot?

Boolean: "true" or "false". If true, execute the word in "boot-command" after the standard bootup process. Otherwise run the Forth interpreter on the console and display an "ok" prompt.

auto-boot-timeout

Integer: number of milliseconds, usually 500. This is the amount of time to wait (in milliseconds) for a key-press to abort auto-boot. This is not a standard OpenFirmware parameter.

boot-command

String: usually "boot". The command to execute to boot the system if the variable "auto-boot?" is true.

boot-device

String: usually "disk". The device to use to load the boot image when "boot" is executed. The string is usually an alias to the actual device.

boot-file

String: usually blank. The file to load from the "boot-device" when "boot" is executed.

diag-device

diag-file

Like "boot-device" and "boot-file", but are used if "diag-switch" is true when "boot" is executed. Typical values are "net" and "diag" respectively, where "net" is typically an alias to a network device.

use-nvramrc?

Boolean: "true" or "false". If true, execute the contents of "nvramrc" at bootup.

nvramrc

The script to execute at bootup. This script is only executed if "use-nvramrc?" is set to true (below).

The normal bootup sequence is: "probe-all install-console banner". If an "nvramrc" script is provided, it should perform the above three in the same order to make sure that the device tree is probed and ready. If these commands are not executed in the "nvramrc" script, then they will be executed by SmartFirmware once the script finishes running.

input-device

String: usually "keyboard". The device path to use for the console input. The device "keyboard" is usually an alias to the actual input device determined in some machine-dependent fashion.

output-device

String: usually "screen". The device to use for console output. Again, this is usually an alias to the actual device.

screen-#rows

screen-#columns

Integer: usually 0. The number of rows and columns desired for the console output. If zero, the largest allowable number will be used depending upon the font used.

inverse-video

Boolean: "true" or "false". Display text on the console as black-on-white if this parameter is true, else as white-on-black. This is not a standard OpenFirmware parameter.

oem-banner?

Boolean: "true" or "false". If true, display the contents of "oem-banner" when the command "banner" is executed in place of the default banner string.

oem-banner

String to display when the "banner" command is executed, if "oem-banner?" is set to true.

oem-logo?

Boolean: "true" or "false". If true, display the bitmap in "oem-logo" in front of the banner when the command "banner" is executed. Otherwise a default logo (or no logo) will be displayed.

oem-logo

Bitmap: 64x64x1 (512 bytes). The bitmap to display if "oem-logo?" is true. The contents may be machine-dependent.

Booting

The bootup process is as follows, as required by the OpenFirmware standard. The process is largely machine-dependent but the steps are the same for all platforms:

1. Power-on self-test
2. System initialization
3. Evaluate the script "nvramrc" if "use-nvramrc?" is true.
4. If the script was not executed or if there was no console after the script finished executing, then:
    1. Execute "probe-all" (evaluates FCode)
    2. Execute "install-console"
    3. Execute "banner"
5. Secondary diagnostics, if "secondary-diag?" is true, and other system-dependent initialization.
6. Default boot if "auto-boot?" is true and no key is held down.
7. Run the Forth command interpreter (if not booted)

Commands for booting include:

boot [device] [args]

Boot the specified device with arguments, if any. If args is not specified, use the contents of the parameter "boot-file" or "diag-file" depending on the current setting of "diag-switch?". If device is not specified, use "boot-device" or "diag-device" instead. "boot" is the same as a "load" followed by a "go" (below).

Example: boot /flash pterm

load [device] [args]

As boot above, except do not actually boot, but do everything else to prepare an image for booting. Run the "go" command to actually boot.

go

Boot the image that was prepared with "load" above.

probe-all

Probes the system for all devices and builds the device tree. Used in the "nvramrc" script. Should only be called once at bootup.

install-console

Selects and installs a console from the device tree, typically a screen and keyboard or a serial-port. Used in the "nvramrc" script. Should only be called once at bootup.

banner

Display the bootup system banner, optionally with a custom color logo. This is usually called in the "nvramrc" script at bootup, but is safe to execute multiple times.

Net-booting

Booting over a network device is probably the most useful feature of SmartFirmware. SmartFirmware supports the standard UDP/IP protocols of BOOTP, DHCP, or RARP for configuring a network device, and TFTP for loading an images over that network device.

The commands ‘‘load’’ and ‘‘boot’’ (described above) when used with a network device accept the following optional device-specific arguments as follows:

`net:[dopt,][prto,]sia,fname,cia,gia,bret,tret [args]`

‘‘net’’ may be a device alias or a device-path to an ethernet device.

‘‘dopt’’ are device-specific options. SmartFirmware’s builtin ethernet drivers support the strings ‘‘promiscuous’’, ‘‘speed=10’’, ‘‘speed=100’’, ‘‘duplex=full’’, and ‘‘duplex=half’’, assuming the device supports the specific capability. The options may be in any order and must be separated by commas. Additional options may also be supported by a specific driver.

‘‘prto’’ is optional and specifies which network boot protocol to be used. SmartFirmware supports the strings ‘‘bootp’’, ‘‘dhcp’’ or ‘‘rarp’’ to force BOOTP, DHCP, or RARP protocols to be used. BOOTP is the default on most platforms (although Sparcs may use RARP).

‘‘sia’’ is the server internet address in dotted octet or hexadecimal format of the server that will be used for TFTP.

‘‘fname’’ is the name of the file to be downloaded over the network using TFTP.

‘‘cia’’ is the internet address that SmartFirmware will use for its local (client) interface address.

‘‘giadr’ is the internet address of the gateway between the local interface and the server if such a gateway is present.

‘‘bret’’ and ‘‘tret’’ are decimal numbers and indicate the number of times that retries will be sent before the BOOTP or TFTP operations fail.

‘‘args’’ are the optional arguments passed to the program when it is executed.

All device arguments are optional, but the commas separating the arguments are required. Any missing fields are filled in by the information returned using BOOTP/DHCP/RARP protocols. Thus the simplest way to boot over a network device is simply ‘‘boot net’’. ‘‘boot net:,file’’ boots a specific file over the net.

Testing

It is possible to exercise some of the test methods from the Forth command prompt. If the parameter variable "diag-switch?" is true, then more verbose and more thorough tests are turned on. It may be useful to run the non-verbose commands at bootup, if desired. The commands are as follows.

test [device]

Test the specified device (or the currently open device if none is specified) by executing its "selftest" method. This may also turn on LEDs or other indicators that testing is in progress.

test-all [device]

Run all "selftest" methods recursively at and below the specified device (or "/" if no device is specified). This runs all tests on the requested node, then on all its children, and their children, and so on.Typically used to run all the tests on all devices from "/".

Changing the console

The console device used may be changed at any time, or selected during bootup. To change it during bootup, the typical method is to either change the NVRAM parameters "input-device" and "output-device" to the desired console device, or add "devalias" commands to "nvramrc" to change the aliases for "screen" and "keyboard" to the desired console.

Otherwise, the console may be changed on-the-fly using the following Forth words. Note that they expect their parameters on the Forth stack and not on the command-line as do most of the commands above.

input (device-str str-len --)

Used by first pushing a string containing the desired device onto the Forth stack, then executing the word "input". The console input will be immediately changed unless an error occurs. Example: " /duart/uart@0" input

output (device-str str-len --)

Changes the current console output device.

io (device-str str-len --)

Changes both the input and the output device at one time. Only useful if the device is capable of both input and output.

Forth variables

Forth variables may be viewed and set using standard Forth words. Some additional variables have been added for SmartFirmware that are not part of the OpenFirmware standard.

To view a Forth variable called, say, "lines/page", use "." to display the value on the top of the stack and "cr" to print a carriage-return:

```
ok lines/page . cr
```

To set a Forth variable, the new value must be first pushed onto the Forth stack before using the "to" word:

```
ok -1 to lines/page
```

lines/page

The current number of lines per page specified for automatic pagination of the output. If the value is zero, no pagination is performed. If the value is negative, it is subtracted from the number of lines in the display. Otherwise the value specifies how many lines are to be displayed before a "More" prompt temporarily stops the output.

scroll-step

The number of lines to scrol l the display by when at the bottom of the screen. This can speed up slower frame buffers by scrolling by 4 lines instead of 1, for instance.