Loading Images with RedBoot

RedBoot is the bootloader currently used on EMAC's Cirrus EP93xx-based products. These include the SoM-9307, PPC-E7, and iPac-9302. This page explains the process of using RedBoot for programming the root flash on EMAC boards.

Setup

In order to access RedBoot, the board will need to be connected through the standard serial connection described in the EMAC OE Getting Started Document. The local network will also need to have a TFTP server available. This is easily installed on most Linux distributions through the package manager. Free TFTP servers exist for Windows as well. The board will need to have an Ethernet connection to the local network. The images that are to be programmed onto the board should be placed in the TFTP server root directory so that they can be loaded from RedBoot.

Accessing RedBoot

After all connections have been made, start the serial terminal application and apply power to the board. Press Ctrl+C several times on the serial terminal until you see the `RedBoot>` prompt. If the kernel begins to boot, reset the board and try again (the standard timeout before booting the kernel is one second). On most versions of RedBoot, you will see either a '+' or a message from RedBoot before it boots the kernel, but some configurations have this disabled.

Once you are at the RedBoot prompt, a list of available commands can be found by typing `help` as shown below:

```
RedBoot> help
Manage aliases kept in FLASH memory
   alias name [value]
Manage machine caches
   cache [ON | OFF]
boot a WIndows CE 5.0 image
   ce [-v <validate only>] [-b <image location>]
Display/switch console channel
   channel [-1|<channel number>]
Compute a 32bit checksum [POSIX algorithm] for a range of memory
   cksum -b <location> -l <length>
Display disks/partitions.
   disks
Display (hex dump) a range of memory
   dump -b <location> [-l <length>] [-s] [-1|2|4]
Execute an image - with MMU off
   exec [-w timeout] [-b <load addr> [-l <length>]]
        [-r <ramdisk addr> [-s <ramdisk length>]]
        [-c "kernel command line"] [<entry_point>]
Manage FLASH images
   fis {cmds}
Manage configuration kept in FLASH memory
   fconfig [-i] [-l] [-n] [-f] [-d] | [-d] nickname [value]
Execute code at a location
   go [-w <timeout>] [entry]
Help about help?
   help [<topic>]
Read CE INI file into global variables
   iniparse -b <mem_base>
Set/change IP addresses
   ip_address [-l <local_ip_address>] [-h <server_address>]
Load a file
```

```
   load [-r] [-v] [-h <host>] [-m <varies>] [-c <channel_number>]
        [-b <base_address>] <file_name>
Compare two blocks of memory
   mcmp -s <location> -d <location> -l <length> [-1|-2|-4]
Fill a block of memory with a pattern
   mfill -b <location> -l <length> -p <pattern> [-1|-2|-4]
test a section of memory by writing a changing pattern and verifying
   mtest -b <start> -e <end> -d <display increment>
Network connectivity test
   ping [-v] [-n <count>] [-l <length>] [-t <timeout>] [-r <rate>]
        [-i <IP_addr>] -h <IP_addr>
read a raw file to a disk partition, ignoring any filesystem
   rawread -b <mem_base> -l <image_length> <file name>
write a raw file to a disk partition, ignoring any filesystem
   rawwrite -b <mem_base> -l <image_length> <file name>
Reset the system
   reset
run a script of redboot commands
   script -b <mem_base>
display system parameters
   system
configure tftp client
   tftpconfig [-t <timeout>]  [-r <retries>]
Display RedBoot version information
   version
Display (hex dump) a range of memory
   x -b <location> [-l <length>] [-s] [-1|2|4]
```

When entering data into RedBoot, the command line interface does not support moving the cursor (i.e. using the cursor/arrow keys). The backspace key must be used to edit any information that has been entered.

Configuring RedBoot

Before the new images can be loaded, RedBoot must be configured to match the local network settings. Although it is possible to use a local DHCP/BOOTP server to lease an IP address to the board through RedBoot, EMAC recommends using a static IP address. If DHCP is enabled in RedBoot and the board cannot access the server or obtain a lease, it will take approximately 30 seconds before timing out and proceeding with the boot process. Contact your IT department for a valid static IP address to be used. You will also need the subnetwork mask, default gateway IP address, and TFTP server IP address. The examples below assume that the network settings are as follows:

IP address	10.0.2.41
Default Gateway	10.0.2.1
Subnet Mask	255.255.255.0
TFTP Server IP	10.0.2.60

The `fconfig` command is used to set configuration data in RedBoot. When run with no options, `fconfig` will start an interactive prompt for setting all configuration variables. This is seldom necessary as generally most of the options are left as default. To see the current configuration, run `fconfig -l` or `fconfig -l -n` to trigger using variable names rather than descriptions. The `-n` flag is handy because the variable names are needed when setting only one variable.

The following is an example of the default settings (note that some of these settings are board specific):

```
RedBoot> fconfig -l
Run script at boot: true
Boot script:
```

```
.. fis unlock -f 0x60000000 -l 0x1fdffff
.. fis load zImage
.. exec -c "root=/dev/mtdblock2 rootfstype=jffs2 console=ttyAM"

Boot script timeout (1000ms resolution): 1
Use BOOTP for network configuration: false
Gateway IP address: 10.0.2.1
Local IP address: 10.0.2.14
Local IP address mask: 255.255.255.0
Default server IP address: 10.0.2.60
DNS server IP address: 10.0.2.1
Set eth0 network hardware address [MAC]: false
GDB connection port: 9000
Force console for special debug messages: false
Network debug at boot time: false
RedBoot> fconfig -l -n
boot_script: true
boot_script_data:
.. fis unlock -f 0x60000000 -l 0x1fdffff
.. fis load zImage
.. exec -c "root=/dev/mtdblock2 rootfstype=jffs2 console=ttyAM"

boot_script_timeout: 1
bootp: false
bootp_my_gateway_ip: 10.0.2.1
bootp_my_ip: 10.0.2.14
bootp_my_ip_mask: 255.255.255.0
bootp_server_ip: 10.0.2.60
dns_ip: 10.0.2.1
ep93xx_esa: false
gdb_port: 9000
info_console_force: false
net_debug: false
```

The values in the second listing show the variable names rather than descriptions.

Run the following commands to configure RedBoot using the network settings listed above. Replace the values with the correct settings for your local network. Note that if you change a variable from its current value, RedBoot will prompt you to commit the value to non-volatile storage. Type 'y' and press Enter when this occurs so that the new setting will be stored in flash.

```
fconfig bootp false
fconfig bootp_my_gateway_ip 10.0.2.1
fconfig bootp_my_ip 10.0.2.41
fconfig bootp_my_ip_mask 255.255.255.0
fconfig bootp_server_ip 10.0.2.60
```

Note that if DHCP/BOOTP is desired, set the `bootp` variable to `true` rather than `false` and leave the other configuration options blank except for `bootp_server_ip`. See above for a discussion on DHCP versus static configuration.

RedBoot uses a bootscript that is run automatically on boot to load and execute the Linux kernel. The values specified here are somewhat hardware-specific. In most cases, the default EMAC values should be used. In general, the commands will perform the following steps:

1. Unlock the NOR flash
2. Load the kernel image from flash into RAM
3. Execute the kernel, passing it a set of boot arguments

To change the boot script, run the following commands:

```
fconfig boot_script true
fconfig boot_script_timeout 1
fconfig boot_script_data
```

After the last command above has been entered, a prompt will appear to `Enter script, terminate with empty line.` The current value of the `boot_script_data` variable will be printed as well. Enter the desired boot script line-by-line at the prompts. After the last line of the boot script, leave the next line empty and press enter to signal that all data has been entered. An example is shown below:

```
RedBoot> fconfig boot_script_data
boot_script_data:
.. fis unlock -f 0x60000000 -l 0x1fdffff
.. fis load zImage
.. exec -c "root=/dev/mtdblock2 rootfstype=jffs2 console=ttyAM"
Enter script, terminate with empty line
>> fis unlock -f 0x60000000 -l 0x1fdffff
>> fis load zImage
>> exec -c "root=/dev/mtdblock2 rootfstype=jffs2 console=ttyAM"
>>
Update RedBoot non-volatile configuration - continue (y/n)? y
... Unlock from 0x61fc0000-0x61fc1000: .
... Erase from 0x61fc0000-0x61fc1000: .
... Program from 0x03fde000-0x03fdf000 at 0x61fc0000: .
... Lock from 0x61fc0000-0x61fc1000: .
```

After the configuration above has been entered, reset the board using the reset button or `reset` command and press Ctrl+C to interrupt the boot process and return to the RedBoot> prompt.

Preparing the Flash

If you are starting with a flash that has not been pre-programmed or want to completely erase the existing image, the flash will need to be formatted. To do this, run the `fis init -f` command. Note that this does not erase the RedBoot image or configuration partition. This command will take several minutes to run. This command is not necessary if the goal is simply to overwrite an existing filesystem or kernel image without changing the existing partition values.

Loading Images

Images are loaded and programmed to the board using a two-step process. First, the image is loaded to RAM through the network using the TFTP protocol. Once the image has been loaded into RAM, the image may be programmed to the flash using the `fis` command.

Loading the Kernel

The kernel is stored in a small binary partition of the flash separate from the root filesystem. The example below assumes that the kernel image is stored on the TFTP server in a file named `zImage-2.6.25`. The following steps are required to program the new kernel image:

1. Unlock the flash (necessary only if the image will be programmed to the flash):

```
fis unlock -f 0x60000000 -l 0x1fdffff
```

2. Load the kernel image via TFTP:

```
load -r -v -b 0x80000 zImage-2.6.25
```

3. At this point the image may be booted without programming to flash. This feature is very helpful for development and testing. Also note that changing the `boot_script_data` variable so that `fis load zImage` is replaced with the TFTP load command above will cause the board to load the kernel from TFTP at

each boot. To manually execute the kernel from RAM, run the exec command from the boot script, i.e.

```
exec -c "root=/dev/mtdblock2 rootfstype=jffs2 console=ttyAM"
```

4. After the image has been loaded into RAM, it may be programmed to the flash:

```
fis create -b 0x80000 -l 0x200000 zImage
```

Loading the Filesystem

The default filesystem for EMAC boards using RedBoot is a JFFS2 image stored on the NOR flash. The steps below describe the process of programming a new image via TFTP assuming the image is stored on the TFTP server with the name `emac-image.jffs2`.

1. Unlock the flash:

```
fis unlock -f 0x60000000 -l 0x1fdffff
```

2. Load the image to RAM (this will take several minutes):

```
load -r -v -b 0x300000 emac-image.jffs2
```

3. Program the image to flash (do not proceed with this step unless the image has successfully been loaded into RAM):

```
fis create -b 0x300000 -l 0x1c00000 jffs2
```

Note that the length of the image programmed above was 28 MB (0x1c00000). This is the default size for boards with 32 MB and 64 MB flash chips. For other boards, the image length will need to be adjusted accordingly.

Scripting

An additional feature of RedBoot is the ability to execute scripts of commands. This is handy for repetitive tasks and programming several boards with the same image. A RedBoot script is simply a text file containing the commands that should be run listed line-by-line in the correct order. This file can then be loaded via TFTP and executed. For example, to script the programming process detailed above, a script file would be created and stored on the TFTP server with the following contents:

```
fis unlock -f 0x60000000 -l 0x1fdffff
load -r -v -b 0x80000 zImage-2.6.25
fis create -b 0x80000 -l 0x200000 zImage
fis unlock -f 0x60000000 -l 0x1fdffff
load -r -v -b 0x300000 emac-image.jffs2
fis create -b 0x300000 -l 0x1c00000 jffs2
```

Assuming that the script file was saved with the name `flash-script`, it could be loaded and executed from RedBoot through the following commands:

```
load -r -v -b 0x50000 flash-script
script -b 0x50000
```