Embedded Linux

For

BCT RE1

User Guide

Document Reference: Embedded Linux User Guide

Document Issue: 1.2

Contents

Introduction

The content of this document provides information required to start building embedded Linux operating systems for the BCT RE1 platform. It covers:

- The tools and components required for building an embedded Linux operating system
- How to install the build components
- How to compile the Linux Kernel source code for BCT RE1.
- How to install Buildroot and build the sample configuration
- How to boot embedded Linux on the RE1 platform using YAMON

Embedded Linux Components

The components required for building, embedded Linux for the BCT RE1 platform are a cross compiling tool chain, Linux kernel source code and a Buildroot configuration.

A prebuilt toolchain is provided which is configured to compile little-endian MIPS code compatible with RE1 on an X86 desktop PC running Linux. The toolchain includes GCC V4.3.2, Binutils 2.19.1, and uClibc 0.9.30.1.

Linux kernel 2.6.28.7 was ported to work with the BCT RE1 platform.

For compiling a root file system, buildroot V2009.05 is used. Please see http://buildroot.uclibc.org for information related to Buildroot. A sample buildroot configuration is provided which is setup to use the provided toolchain.

The components above have all been tested to compile using a Debian 5 development machine.

Installation of the Embedded Linux build components

As root or as a user with root privileges create a directory called “embedded” in the root of the file system and enter the directory. Issue the following commands to achieve this:

```
cd /
mkdir embedded
cd embedded
```

Download the latest RE1 Linux components from the Blue Chip Technology website:

wget
http://www.bluechiptechnology.co.uk/~bluedownloads/Single_Board_Computers/RE1/Drivers/re1embeddedlinuxV102.tar.bz2

Extract the tar ball by issuing the command:

tar –xvjf re1embeddedlinuxV102.tar.bz2

Once extracted the build components will be laid out in the following structure on the development machine. The first directory (“embedded”) is the folder created in the root of the file system.

```
/embedded ->
        /projects ->
                /bctre1 ->
                        /linux-2.6.28.7-bctre1
                        /buildroot
        /toolchains ->
                /bctre1
```

Compiling the Linux Kernel

Before compiling the Linux Kernel we must set some environment variables. This is to ensure the kernel builds for the correct architecture and can find the cross compiling tool chain. To make this task simpler a script file is provided to configure the environment for a BCT RE1 build. Issue the following commands to run the script:

```
cd /embedded/projects/bctre1
. ./setenv.sh
```

To compile the kernel we must enter the root of the kernel source tree, make some configuration changes and use make to start the compile. Issue the following commands.

```
cd /embedded/projects/bctre1/linux-2.6.28.7-bctre1
cp ./arch/mips/configs/bctre1-defconfig ./.config
make oldconfig
make
```

The compile process should complete in approximately ten minutes, and leave a Linux kernel (vmlinux) in the root of the kernel source tree.

If changes are required to the kernel configuration the command “make menuconfig” can be used to present a menu based configuration utility for the Linux kernel. If any changes are made using the menuconfig tool, the “make” command must be re-issued.

A script file called “makesrec.sh” is provided in the root of the kernel source tree, which will convert the generated Linux kernel into a Motorola S-Record, compatible with the YAMON bootloader and copy the S-Record to the location “/tftpboot/vmlinux.srec”. The folder “/tftpboot” presumes a TFTP server has been setup to serve files from that location.

Building a root file system

A sample buildroot configuration is provided in the location “/embedded/projects/bctre1/buildroot”. Building the root file system with the default configuration requires the following commands to be issued:

```
cd /embedded/projects/bctre1/buildroot
```

make oldconfig

make

The compile process should take approximately thirty minutes, and generate the root file system in the following formats and locations.

Format	Location	Description
Raw file system	/embedded/projects/bctre1/buildroot/project_build_mipsel/bctre1/root	Root file system as a raw file system. Useful for booting the root file system over NFS
Tar ball	/embedded/projects/bctre1/buildroot/binaries/bctre1/rootfs.mipsel.tar	Tar ball of the root file system. Useful for uncompressing straight to formatted SD card
S-Record	/embedded/projects/bctre1/buildroot/binaries/bctre1/rootfs.mipsel.jffs2.srec	S-Record of root file system. Useful for putting the root file system into the onboard NOR flash of the BCT RE1 using YAMON

The default buildroot configuration produces a basic small footprint file system which can easily fit into the onboard NOR flash of the BCT RE1. If a more featured root file system is required the command “make menuconfig” can be used to present a menu based configuration utility for the buildroot package manager. If any changes are made using the menuconfig tool, the “make” command must be re-issued.

BCT RE1 NOR flash configuration

The BCT RE1 features 32MB’s of on board NOR flash which is partitioned into five partitions as follows.

Partition Number	Linux Partition Name	Physical Start Address	Physical End Address	Partition Size	Description
0	mtdblock0	0xBE000000	0xBF800000	24MB’s	Root File System
1	mtdblock1	0xBF800000	0xBFC00000	4MB’s	Linux Kernel
2	mtdblock2	0xBFC00000	0xBFD00000	1MB’s	YAMON boot loader
3	mtdblock3	0xBFD00000	0xBFFC0000	2.75MB’s	Spare
4	mtdblock4	0XBFFC0000	0x20000000	256KB’s	YAMON boot loader Variables

There is a 2.75MB hole in the flash configuration due to a combination of where the YAMON boot loader can reside and the fact that the Linux kernel is too big to fit in this space.

YAMON Bootloader

The first software to execute after the BCT RE1 is powered on is the YAMON boot loader. YAMON performs the following functions:

- Initialises the hardware to a known state
- Allows the contents of onboard flash to be updated over Ethernet
- Allows command line parameters to be passed to the Linux kernel

- Allows the device to automatically boot Linux at start-up.

This section will demonstrate how to configure YAMON to boot Embedded Linux in different configurations. For more detailed information on YAMON please see the, “YAMON reference manual”.

Booting to the YAMON prompt

Configuration of YAMON is performed using a command driven interface though a serial terminal emulator. Connect a NULL modem cable between COM2 of the RE1 and a serial port on a desktop PC. Open a terminal emulator on the desktop PC (E.g. Minicom for Linux, or HyperTerminal for Windows) and open the serial port with settings, 115200 baud, 8 data bits, 1 stop bit, no parity, and no flow control.

After powering on the RE1 the following should be seen on the terminal screen:

```
YAMON ROM Monitor, Revision 02.19DB1100.
Copyright (c) 1999-2000 MIPS Technologies, Inc. - All Rights Reserved.

For a list of available commands, type 'help'.

Compilation time =          Aug 20 2008  10:59:51
MAC address =              00.c0.46.00.00.6d
Processor Company ID =      0x03
Processor ID/revision =     0x02 / 0x04
Endianness =               Little
CPU =                    336 MHz
Flash memory size =         32 MByte
SDRAM size =               64 MByte
First free SDRAM address =   0x8008c604

YAMON>
```

Configuring YAMON

To configure YAMON for downloading S-Record files over Ethernet some environment variables must be configured. Below are some sample commands which configure YAMON with the IP address 10.0.0.251, subnet mask 255.255.255.0, and target TFTP server address of 10.0.0.37. These specific IP settings should be altered to values compatible with the target network.

```
set bootfile vmlinux.srec
set bootprot tftp
set bootserver 10.0.0.37
set ipaddr 10.0.0.251
set subnetmask 255.255.255.0
```

The following environment variables are optional but will save on typing when booting Linux. Each variable is a Linux command line parameter for booting the root file system from a different medium. The 10.0.0.37 address in the “nfs” environment variable is a sample IP address of a Network File System server, and should be changed to a value compatible with the target network.

```
set nfs 'ip=dhcp root=/dev/nfs rw nfsroot=10.0.0.37:/nfs/root'
set sdcard 'root=/dev/mmcblk0p1 rootwait rootfstype=ext2 rw'
set nor 'root=/dev/mtdblock0 rootfstype=jffs2 rw'
```

Booting Linux

This section describes how to download and boot a Linux kernel using YAMON and presumes the following setup.

- Linux development PC configured with IP address 10.0.0.37
- Linux development PC configured with a TFTP server, serving from directory "/tftpboot"
- Built Linux kernel, converted to S-Record format and copied to "/tftpboot" directory. See the "Compiling the Linux Kernel" section for details on achieving this.
- YAMON configured with environment variables as configured in the section, "Configuring YAMON"

From the YAMON command prompt type the command, "load". This will cause YAMON to contact the TFTP server at address 10.0.0.37, and download the vmlinux.srec file into memory. The following output should be seen at the YAMON console.

```
YAMON> load
About to load tftp://10.0.0.37/vmlinux.srec
Press Ctrl-C to break
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
Start = 0x801043c0, range = (0x80100000,0x804882b3), format = SREC
YAMON>
```

The above output means that YAMON copied the file vmlinux.srec into memory starting at address 0x801043c0 and ending at address 0x804882b3. The kernel entry point for this particular kernel is at address 0x801043c0. NOTE: These addresses will be different on a kernel by kernel basis.

Now the Linux kernel is in memory it is possible to execute the kernel. Issue the command, "go 0x801043c0". This will cause YAMON to start executing code at address 0x801043c0 which is where our kernel entry point is located. The following output should be seen at the console.

```
 YAMON> go 0x801043c0
Linux version 2.6.28.7 (root@drobinson-linux) (gcc version 4.3.2 (crosstool-NG-1.4.1) ) #105 Tue Aug 11 14:59:55 BST 2009
CPU revision is: 02030204 (Au1100)
Blue Chip Technology RE1
(PRID 02030204) @ 336 MHz
BCLK switching enabled!
Determined physical RAM map:
 memory: 04000000 @ 00000000 (usable)
Zone PFN ranges:
  Normal   0x00000000 -> 0x00004000
Movable zone start PFN for each node
early_node_map[1] active PFN ranges
    0: 0x00000000 -> 0x00004000
Built 1 zonelists in Zone order, mobility grouping on.  Total pages: 16256
Kernel command line:  console=ttyS0,115200
Primary instruction cache 16kB, VIPT, 4-way, linesize 32 bytes.
Primary data cache 16kB, 4-way, VIPT, no aliases, linesize 32 bytes
PID hash table entries: 256 (order: 8, 1024 bytes)
CPU frequency 336.00 MHz
Console: colour dummy device 80x25
Dentry cache hash table entries: 8192 (order: 3, 32768 bytes)
Inode-cache hash table entries: 4096 (order: 2, 16384 bytes)
Memory: 60108k/65536k available (2920k kernel code, 5352k reserved, 541k data, 156k init, 0k highmem)
Calibrating delay loop... 334.84 BogoMIPS (lpj=669696)
Mount-cache hash table entries: 512
net_namespace: 288 bytes
NET: Registered protocol family 16
<5>SCSI subsystem initialized
usbcore: registered new interface driver usbfs
usbcore: registered new interface driver hub
usbcore: registered new device driver usb
NET: Registered protocol family 2
IP route cache hash table entries: 1024 (order: 0, 4096 bytes)
TCP established hash table entries: 2048 (order: 2, 16384 bytes)
TCP bind hash table entries: 2048 (order: 1, 8192 bytes)
TCP: Hash tables configured (established 2048 bind 2048)
TCP reno registered
NET: Registered protocol family 1
JFFS2 version 2.2. © 2001-2006 Red Hat, Inc.
msgmni has been set to 117
io scheduler noop registered
io scheduler anticipatory registered
io scheduler deadline registered
io scheduler cfq registered (default)
au1100fb: LCD controller driver for AU1100 processors
au1100fb: Panel=CRT_640x480_16 Mode=default
Console: switching to colour frame buffer device 80x30
ucb1x00-ts: registered
Serial: 8250/16550 driver4 ports, IRQ sharing disabled
serial8250.9: ttyS0 at MMIO 0x11100000 (irq = 8) is a 16550A
console [ttyS0] enabled
serial8250.9: ttyS1 at MMIO 0x11200000 (irq = 9) is a 16550A
serial8250.9: ttyS2 at MMIO 0x11400000 (irq = 11) is a 16550A
loop: module loaded
au1000_eth version 1.6 Pete Popov <ppopov@embeddedalley.com>
eth0: Au1xx0 Ethernet found at 0x10500000, irq 36
au1000_eth_mii: probed
eth0: attached PHY driver [Generic PHY] (mii_bus:phy_addr=0:00, irq=-1)
Driver 'sd' needs updating - please use bus_type methods
BCTRE1 Flash: probing 32-bit flash bus
BCTRE1 Flash: Found 2 x16 devices at 0x0 in 32-bit bank
NOR chip too large to fit in mapping. Attempting to cope...
 Amd/Fujitsu Extended Query Table at 0x0040
```

```
BCTRE1 Flash: CFI does not contain boot bank location. Assuming top.
number of CFI chips: 1
cfi_cmdset_0002: Disabling erase-suspend-program due to code brokenness.
Reducing visibility of 65536KiB chip to 32768KiB
Creating 5 MTD partitions on "BCTRE1 Flash":
0x00000000-0x01800000 : "User FS"
0x01800000-0x01c00000 : "raw kernel"
0x01c00000-0x01d00000 : "YAMON"
0x01d00000-0x01fc0000 : "Spare"
0x01fc0000-0x02000000 : "YAMON env"
ohci_hcd: USB 1.1 'Open' Host Controller (OHCI) Driver
au1xxx-ohci au1xxx-ohci.0: Au1xxx OHCI
au1xxx-ohci au1xxx-ohci.0: new USB bus registered, assigned bus number 1
au1xxx-ohci au1xxx-ohci.0: irq 34, io mem 0x10100000
usb usb1: configuration #1 chosen from 1 choice
hub 1-0:1.0: USB hub found
hub 1-0:1.0: 2 ports detected
Initializing USB Mass Storage driver...
usbcore: registered new interface driver usb-storage
USB Mass Storage support registered.
mice: PS/2 mouse device common for all mice
au1xmmc: au1xmmc_init
Registered led device: mmc0
au1xxx-mmc: MMC Controller 0 set up at B0600000 (mode=pio)
usbcore: registered new interface driver usbhid
usbhid: v2.6:USB HID core driver
Advanced Linux Sound Architecture Driver Version 1.0.18rc3.
ALSA AC97: Driver Initialized
ALSA device list:
  #0: AMD Au1000--AC97 ALSA Driver
TCP cubic registered
NET: Registered protocol family 15
RPC: Registered udp transport module.
RPC: Registered tcp transport module.
Root-NFS: No NFS server available, giving up.
VFS: Unable to mount root fs via NFS, trying floppy.
VFS: Cannot open root device "<NULL>" or unknown-block(2,0)
Please append a correct "root=" boot option; here are the available partitions:
1f00         24576 mtdblock0 (driver?)
1f01          4096 mtdblock1 (driver?)
1f02          1024 mtdblock2 (driver?)
1f03          2816 mtdblock3 (driver?)
1f04           256 mtdblock4 (driver?)
Kernel panic - not syncing: VFS: Unable to mount root fs on unknown-block(2,0)
```

It is normal for the kernel to panic as we did not specify where the root file system should be obtained from.

To specify the location of the root file system, some command arguments must be passed to the Linux kernel. This is achieved by issuing the YAMON go command in the format:

go <address> . <arguments>

The following table shows some examples of passing Linux command line arguments using YAMON.

YAMON command	Description
go 0x801043c0 . $nfs	Boots Linux and mounts the root file system using an NFS server. This environment variable presumes that an NFS server has been setup on the desktop machine (10.0.0.37) and the root file system files are in the location, “/nfs/root”
go 0x801043c0 . $nor	Boots Linux and mounts the root file system in the on-board NOR flash of the RE1 device. This environment

	presumes that the root file system is present in partition 0 of the RE1 flash. See the following section for details.
go 0x801043c0 . $sdcard	Boots Linux and mounts the root file system in an ext2 formatted SD card.

Booting the root file system from NOR flash

Before Linux can mount the root file system as a JFFS2 file system, the file system created in section "Building a root file system" must be copied into NOR flash using YAMON.

On the development machine copy the rootfs.mipsel.jffs2.srec file to the TFTP server directory.

cp /embedded/projects/bctre1/buildroot/binaries/bctre1/rootfs.mipsel.jffs2.srec /tftpboot

At the YAMON prompt type, "fill 81000000 1800000 ff". This will cause YAMON to fill 24MB's of flash with 0xFF.

At the YAMON prompt type, "load /rootfs.mipsel.jffs2.srec". This will cause YAMON to download the rootfs.mipsel.jffs2.srec file from the TFTP server into memory.

At the YAMON prompt type, "erase be000000 1800000". This will cause YAMON to erase 24 MB's of flash which corresponds to partition 0 of BCT RE1 NOR flash. See section "BCT RE1 NOR flash configuration" for details.

At the YAMON prompt type, "copy 81000000 be000000 1800000". This will cause YAMON to copy the root file system from memory to flash.

It is now possible to load Linux and boot the root file system from on-board NOR flash. Issue the following commands to achieve this:

load

go 0x801043c0 . $nor

The first boot of Linux with a root file system in NOR flash will longer than normal.

Once the system has booted a login prompt will be shown the LCD display. The default username is "root", with no password.

Booting Linux from NOR flash

The Linux kernel must always boot from memory. Thus far all demonstrations have required the Linux Kernel to be loaded into memory over Ethernet, before jumping to the kernel start address. The principle of booting the Linux kernel from NOR flash is similar to that of booting over Ethernet. Once the image is in NOR flash, the Kernel should be copied from NOR flash to memory, before jumping to the kernel start address.

At the YAMON prompt type, "load". This causes the Linux kernel to be copied into memory over Ethernet .

At the YAMON prompt type, “erase BF800000 400000”. This causes 4MB’s of flash to be erased. This corresponds to partition 1 of BCT RE1 NOR flash. See section “BCT RE1 NOR flash configuration” for details.

At the YAMON prompt type, “copy 80100000 BF800000 400000”. This causes YAMON to copy the Linux kernel from memory to flash.

YAMON has a special environment variable called “start”, which when configured automatically gets executed at power on. It is possible to use this variable to automatically boot Linux at power on without having to issue any YAMON commands.

At the YAMON prompt type, “set start 'copy BF800000 80100000 400000 ; go \

801043c0 . root=/dev/mtdblock0 rootfstype=jffs2'

The above is a concatenation of two commands, each separated by a semi colon. The first command copies the Linux kernel out of flash into its original memory location. The second command jumps to the Linux kernels entry point with a command argument telling the kernel that the JFFS2 root file system in NOR flash should be used.

TFTP server and NFS server setup

The process of setting up a TFTP and NFS server in Linux can vary dependent on the distribution being used. The following links provide guidance for setting up these servers on Debian.

http://www.debianhelp.co.uk/nfs.htm

http://www.debianhelp.co.uk/tftp.htm

Linux Kernel Environment Variables

To configure the Linux kernel at start up it is possible to pass environment variables using YAMON. As previously mentioned the method passing parameters to the Linux kernel using YAMON is to issue the “go” command in the following format:

go <address> . <arguments>

The table below lists some environment variables supported by the Linux kernel for BCT RE1

Linux Parameter	Description
ip=dhcp root=/dev/nfs rw nfsroot=10.0.0.37:/nfs/root	Causes Linux to get an IP address from a DHCP server and load a root file system from an NFS server located at address 10.0.0.37.
root=/dev/mmcblk0p1 rootwait rootfstype=ext2 rw	Causes Linux to load a root file system from partition 1 of an SD card formatted as EXT2
root=/dev/mtdblock0 rootfstype=jffs2 rw	Causes Linux to load a root file system

	from partition 0 of on-board flash.
video=au1100fb:panel:URT_8089	Causes Linux to setup the frame buffer for use with the URT 8089 LCD (640*480)
video=au1100fb:panel:URT_8253	Causes Linux to setup the frame buffer for use with the URT 8253 LCD (480*272)
video=au1100fb:panel:URT_8173	Causes Linux to setup the frame buffer for use with the URT 8173 LCD (800*480)

Linux Demo Image

The demo image included with BCT-RE1 Linux development kits includes the following sample applications:

Matchbox – http://matchbox-project.org

To test this type, “matchbox-session-ts” at the command prompt.

Qtopia – http://qt.nokia.com/

Many sample applications exist for Qtopia. To test them, navigate to “/usr/local/Trolltech/QtEmbedded-4.5.1-mips/demos”, enter the directory of the demo to test, and run the demo with a “ –qws” parameter.

E.g.

1. cd /usr/local/Trolltech/QtEmbedded-4.5.1-mips/demos
2. cd textedit
3. ./textedit -qws