Linux

For

BCT RE2G2

User Guide

Document Reference: BCTRE2G2 Linux User Guide

Document Issue: 1.05

Associated SDK release: 1.04

Author: D Robinson

Contents

Introduction

The content of this document provides information required to start building Linux operating systems for the BCT RE2G2 platform. It covers:

- The tools and components required for building a Linux operating system
- How to install the build components
- How to compile the U-Boot boot loaders
- How to compile the Linux Kernel
- How to boot Linux on the RE2G2 platform using U-boot

Customers already familiar with building Linux for the RE2 platform may still benefit from reading this document as various changes were made to the hardware and software. Please see the, "Porting from RE2 to RE2G2" section for details on differences between the two platforms.

Environment Setup

Required Linux Components

The components required for building Linux for the BCT RE2G2 platform are a cross compiling tool chain, boot loader source code, Linux kernel source code, an Ubuntu root file system, and an X86 Ubuntu 12.04 X64 LTS development machine.

A prebuilt toolchain is provided which is configured to compile ARM V7 code compatible with RE2G2 on an X86 desktop PC running Linux. The toolchain used is from Linaro (2012.01). Alternatively the Ubuntu cross compiler can be used which can be installed by issuing the following commands on the development machine:

```
apt-get install g++-arm-linux-gnueabihf
apt-get install gcc-arm-linux-gnueabihf
```

Linux kernel 3.16.2 was ported to work with the BCT RE2G2 platform.

The root file system tested with BCT RE2G2 is Ubuntu 12.04 LTS (Armhf)

The components above have all been tested to compile using an Ubuntu 12.04 LTS 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
```

Copy the latest RE2G2 Linux components to the "/embedded" directory. Sources can be distributed in different ways, but usually they can be downloaded straight from our website. (http://www.bluechiptechnology.com/). At time of writing re2g2linuxsrcV101.tar.bz2 was the latest release. Download the Linux source code for RE2G2 using the command:

```
wget
http://dl.bluechiptechnology.com/dl/re2g2/software/re2g2linuxsrcv101.tar.bz
2
```

Extract the tar ball by issuing the command:

```
tar xvjf re2g2linuxsrcvxxx.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.

Directory	Directory	**Description**
embedded/projects/bctre2g2	linux-bctre2g2	Kernel source code with configuration for BCT RE2G2
	rootfs	Staging directory used for holding the root file system of the target device during development
	uboot-bctre2g2	Source code for the U-boot boot loader including configuration for BCT RE2G2

Development Machine Setup

Where possible build scripts have been provided for the various components included with the Linux SDK for BCT RE2G2. These scripts presume the following has been setup on the Ubuntu 12.04 LTS development machine.

- A TFTP server serving files from a tftpboot directory in the root of the file system. (/tftpboot)
- An NFS server serving files from an nfs directory in the root of the file system (/nfs). The /nfs directory should be symbolically linked to the root staging directory, "/embedded/projects/bctre2g2/rootfs".

The following links provide information on setting up an NFS and TFTP server.

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

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

Before compiling the various components of Linux we must set some environment variables. This is to ensure the configuration tools build 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 RE2G2 build. Issue the following commands to run the script:

```
cd /embedded/projects/bctre2g2
. ./setenv.sh
```
To use the Linaro toolchain

Or

```
. ./setenvubuntu.sh
```
To use the Ubuntu toolchain

Building Required Components

Creating a root file system

There are many Linux distributions available that are compatible with BCT RE2G2. Ubuntu was chosen as the distribution of choice for this guide, as it has a large pre-compiled package database, and easy to use configuration tools. Ubuntu is not the ideal choice for all Linux projects, but it will allow a basic operating system to be constructed quickly to allow evaluation of the BCT RE2G2.

The process of creating an Ubuntu root file system for BCT RE2G2 consists of the following steps.

- Extract an Ubuntu image into the staging directory
- Add kernel modules and other specific support to the root file system.
- Boot the generated Ubuntu root file system on a BCT RE2G2.
- Configure the Ubuntu root file system using, apt-get, synaptic package manager, or another package manager.

To support the BCT RE2G2 hardware and kernel, various files need to be copied to the root file system. To simplify this process all build components that need to modify the root file system are configured to do so at the staging location, “/embedded/projects/bctre2g2/rootfs”. Therefore we must extract a root file system as a starting point to this location.

For convenience a pre-built root file system can be downloaded from our web site and extracted by issuing the following commands

```
wget
http://dl.bluechiptechnology.com/dl/re2g2/software/re2g2ubuntudemorootfsv101.tar.bz2
```

At this point Linux Kernel modules, and any other specific support must be added to the root file system. The following sections describe this process.

Compiling the Linux Kernel

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/bctre2g2/linux-bctre2g2
make bctre2g2_defconfig
./build.sh
```

The compile process should complete in approximately six minutes, and leave a Linux kernel (uImage) at, "./arch/arm/boot/uImage", and, "/tftpboot/uImage.bin"

build.sh is an example of a script file that simplifies the process of building BCT RE2G2 Linux components. Please study these files for an understanding of their purpose.

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 "./build.sh" command must be re-issued.

Once the kernel has been compiled, the kernel modules must be copied to the root file system. Issuing the following command performs this task.

./installmodulesrootfs.sh

NOTE: Part of the kernel compilation process requires the uboot mkimage tool be present on the development PC. In Ubuntu this package can be installed by issuing the following command.

"**apt-get install uboot-mkimage**"

U-Boot Bootloader – Ported (http://www.denx.de/wiki/U-Boot)

U-Boot 2014.07 has been ported to work with RE2G2. It includes both a primary and secondary bootloader, which have the purpose of initialising the hardware, and booting a Linux operating system.

To build U-Boot for BCT RE2G2 issue the following commands.

```
cd /embedded/projects/bctre2g2/uboot-bctre2g2
./buildre2g2.sh
```

The compiled boot loaders are located at,

"/embedded/projects/bctre2g2/uboot-bctre2g2/u-boot.img"

And

"/embedded/projects/bctre2g2/uboot-bctre2g2/MLO"

Booting BCT RE2G2

So far this document has described how to set up a build environment and how to build the various components of a Linux operating system for BCT RE2G2. The components we have built are as follows.

Component	Location
Ubuntu Root File System	/embedded/projects/bctre2g2/rootfs
Linux Kernel	/embedded/projects/bctre2g2/linux-bctre2g2/arch/arm/boot/uImage
U-Boot	/embedded/projects/bctre2g2/uboot-bctre2g2/u-boot.img /embedded/projects/bctre2g2/uboot-bctre2g2/MLO

This section of the document describes how to use these components to boot and configure Ubuntu Linux on a BCT RE2G2.

BCT RE2 Hardware Setup

Linux and U-boot for BCT RE2G2 heavily relies on access to a serial console. By default U-boot and Linux are configured to use the RS232 port available on P11 of the RE2. By default the board is set to communicate at 115200, 8, n, 1. Before turning on the BCT-RE2 for the first time it is recommended that this port be connected to a PC with terminal emulator software running. E.g. HyperTerminal.

Creating a bootable SD-Card

To boot from an SD-Card the BCT RE2G2 board requires the first partition to be formatted as FAT and be active. A Linux root file system is not compatible with FAT partitions, and requires an EXT2 or EXT3 partition on the SD-Card. To satisfy both of these requirements an SD card must be dual partitioned to have both a fat, and EXT partition. The following steps detail how to create a bootable SD Card compatible with BCT RE2G2. The size of SD Card required depends on the size of the final root file system, although 8GB or greater is recommended.

1. Plug an SD card into the development machine via a USB adapter
2. Issue command, "**`dmesg | tail`**", to determine the root device name of the SD Card. E.g. /dev/sdf
3. Issue the following commands followed by return.
    a. **`fdisk /dev/sdf`** - replace /dev/sdf with device name of SD Card determined in step 2.
    b. **`p`** – print the current partition table
    c. **`d`** – Delete command
    d. **`1`** – Delete existing partition 1
    e. Delete all partitions 2 – 4 as necessary
    f. **`n`** – Create new partition
    g. **`p`** – Primary partition
    h. **`1`** – Partition 1
    i. Return for default start cylinder
    j. **`+50M`** - Create a 50MB partition
    k. **`t`** – Set partition type
    l. **`1`** - Set partition type of partition 1
    m. **`c`** – Set partition type to fat
    n. **`n`** – Create new partition
    o. **`p`** – Primary partition
    p. **`2`** – Partition 2
    q. Return for default start cylinder
    r. Return for default end cylinder
    s. **`t`** – Set partition type
    t. **`2`** - Set partition type of partition 2
    u. **`83`** – Set partition type to Linux
    v. **`a`** – Set partition active
    w. **`1`** - Set partition 1 active
    x. **`w`** – Write partitions tables to SD-Card
4. Issue command, "**`sync`**", to flush the SD-Card
5. Unplug and reconnect the SD-Card
6. Issue command, "**`dmesg | tail`**", to determine the SD Card partitions. E.g. /dev/sdf1, /dev/sdf2
7. Issue command, "**`mkfs.msdos /dev/sdf1`**", to format the first SD-Card partition as FAT.
8. Issue command, "**`mkfs.ext3 /dev/sdf2`**", to format the second SD-Card partition as ETX3.

Copying Linux files to SD-Card

After creating a bootable SD-Card, the Linux file system and modules can be copied to the media. U-boot, and the Linux kernel should be copied to FAT partition on the SD-Card, and the root file system should be copied to the EXT3 partition. Issue the following commands to mount the individual partitions, and copy the required files.

`mkdir /media` – create a directory for mounting media devices

`mkdir /media/sdcard` – create a directory for mounting SD-Cards.

`mount /dev/sdf1 /media/sdcard` – mount the first partition of SD-Card to /media/sdcard

`cp /embedded/projects/bctre2g2/uboot-bctre2g2/MLO /media/sdcard/` - copy X-Loader to SD-Card. **MLO must be the first file copied**

`cp /embedded/projects/bctre2g2/uboot-bctre2g2/u-boot.img /media/sdcard/` - copy u-boot to /media/sdcard

`cp /embedded/projects/bctre2g2/linux-bctre2g2/arch/arm/boot/uImage /media/sdcard/uImage.bin` – copy linux kernel to /media/sdcard as uImage.bin

`umount /media/sdcard` - un-mount SD-Card boot partition

`mount /dev/sdf2 /media/sdcard` – mount the second partition of SD-Card to /media/sdcard

`cp -rp /embedded/projects/bctre2g2/rootfs/* /media/sdcard/` - copy entire root file system including sub directories to /media/sdcard

`umount /media/sdcard` - un-mount SD-Card root file system partition

Remove SD-Card from system

Booting Ubuntu Linux on BCT RE2G2 from SD-Card

Setup the BCT RE2G2 hardware with Ethernet, DVI monitor, and RS232 (p11) connected to a PC terminal. Insert the SD-Card, and power on the BCT RE2G2. Booting the entire Linux system from SD-Card is the default configuration of U-boot, so the system should boot into Ubuntu without requiring special configuration.

Note: If there is firmware in the on-board NAND flash the RE2G2 will not automatically boot from SD-Card, but from the on-board flash. This default behaviour can be changed by asserting the engineering pin while powering on the BCT RE2G2. See the BCT RE2G2 user guide for details.

BCT RE2G2 Serial Ports

The UARTs in the RE2G2 are mapped as follows:

RE2G2 Header	Linux Device Name
P11(RS232 RX + RS232 TX)	/dev/ttyO2
P11(CRX1N CRX1P CTX1N CTX1P)	/dev/ttyO1
P10	/dev/eser0

/dev/ttyO1 is an RS485 / RS422 compatible port which has a transmit enable signal. This signal is controlled using GPIO 137. From the Linux console this signal can be manipulated using the commands:

```
echo 1 >> /sys/class/gpio/gpio137/value
echo 0 >> /sys/class/gpio/gpio137/value
```

Configuring Ubuntu for BCT RE2G2

U-Boot Environment Variables

U-boot parameters can be modified to change the configuration of the Linux kernel. Parameters can be changed by halting the boot process shortly after power on. When the message, “Hit any key to stop autoboot: 1” is displayed, press a key to halt auto boot.

U-boot parameters can be changed with the command setenv in the following format.

```
setenv <parameter> <setting>
```

When the required settings have been changed they can be saved by issuing the command, “**saveenv**”.

The following table lists the BCT RE2G2 specific U-Boot commands. For comprehensive information on u-boot please refer to the following link.

http://www.denx.de/wiki/view/DULG/Manual

Parameter	Description
defaultdisplay	dvi = using DVI / HDMI display lcd = using BCT RE2 compatible LCD
dvimode	Sets the display resolution and colour depth. The example below sets the display to a resolution of 800x480 with 32bits per pixel. **Must be set regardless of whether the display is DVI or LCD** 800x480MR-32@60
bctlcd	Selects the LCD to use when defaultdisplay is set to lcd. Supported settings are: urt8089 – 640x480 urt8253 – 320x240 urt8044 – 320x240 urt8065 – 320x240 urt8173 – 800x480 lcd71800480 – (800x480) BCT 7" display with PM9

Limiting CPU frequency

For customers who wish to clock back their 1GHz RE2G2 to similar performance to the original 600MHz/720MHz RE2, this can be achieved by using the “maxcpu” variable.

maxcpu=600 (Limit the RE2 to 600Mhz)
maxcpu=800 (Limit the RE2 to 800Mhz)
maxcpu=1000 (Limit the RE2 to 1000Mhz)

The easiest way to pass the parameter to the kernel is to modify the u-boot mmcargs variable.

1. Break into u-boot by pressing any key into the serial console when notified shortly after power on.
2. Type, “edit mmcargs”
3. Append the variable to the existing string
4. Press enter
5. Type, “save” followed by enter to save the new mmcargs variable.
6. Reboot the RE2G2

Booting from NAND Flash

Although the Linux images provided by Blue Chip Technology for BCT RE2G2, are designed to run from an SD Card, it is possible to boot the entire system out of NAND flash. This section with detail how to copy the XLDR and UBOOT boot loaders into NAND flash. The instructions presume that a bootable SD Card has been prepared for the RE2, that includes the files MLO, and u-boot.img in the FAT formatted partition 1.

The NAND flash on RE2 is partitioned under Linux as follows:

Start Address - End Address
0x000000000000-0x000000080000 : "MLO"
0x000000080000-0x000000180000 : "U-Boot"
0x000000180000-0x0000001a0000 : "U-Boot Env"
0x0000001a0000-0x0000005a0000 : "Kernel"
0x0000005a0000-0x000020000000 : "File system"

1. Powe on the RE2 with the bootable SD card inserted, and monitor the debug serial port using a terminal emulator.
2. When u-boot messages are display on terminal. “Hit any key” to halt the system boot.
3. At the u-boot console type the following commands
    a. “`mmc rescan`”. To re-detect the SD Card media
    b. “`fatload mmc 0:1 80200000 MLO`”. To load MLO from the SD Card into memory
    c. “`nandecc hw`”. To set hardware ECC correction
    d. “`nand erase 0 80000`”. To erase the region of NAND containing MLO.
    e. “`nand write 80200000 0 80000`”. To write MLO to NAND.
    f. “`fatload mmc 0:1 80200000 u-boot.img`”. To load u-boot.img from the SD Card into memory
    g. “`nandecc hw`”. To set software ECC correction
    h. “`nand erase 80000 100000`”. To erase the region of NAND containing u-boot.
    i. “`nand write 80200000 80000 100000`”. To write u-boot to NAND.
4. Remove the SD Card and power cycle the RE2G2.
5. MLO and UBOOT will now boot from the RE2G2 NAND flash.

For further reading please visit, http://elinux.org/BeagleBoardNAND

Known Issues

1. The PowerVR graphics accelerator is currently not supported. This is due to an incompatibility between the kernel supported by the RE2G2, and the Ubuntu 12.04 library formats.
2. The DSP video accelerator is currently not supported. This is due to an incompatibility between the kernel supported by the RE2G2, and the Ubuntu 12.04 library formats.

Porting from RE2 to RE2G2

Customers migrating from RE2 to RE2G2 should be aware of some hardware and software differences between the two platforms. Most changes are self documented in the u-boot and Linux kernel source code. The files to reference are:

a. **uboot-bctre2g2/board/bct/bctre2g2/bctre2g2.c**
b. **uboot-bctre2g2/board/bct/bctre2g2/bctre2g2.h**
c. **uboot-bctre2g2/include/configs/bctre2g2.h**
d. **linux-bctre2g2/arch/arm/mach-omap2/board-bctre2g2.c**

The table below highlights the main differences between the two platforms.

Feature	**RE2**	**RE2G2**
ARM ABI	SoftFP	HardFP
Ubuntu	10.04 LTS / 12.04 LTS	12.04 LTS
X-loader (MLO)	X-loader	X-loader is superseded by u-boot spl. U-boot now builds the MLO file.
U-boot	V2011.06	V2014.07
Kernel	V2.6.38.8	V3.16.2
GPIO control	Control using GPIO application	GPIO's are directly exported by the kernel and can be controlled using SYSFS
P6 GPIO mappings (1 - 12)	OMAP GPIOs 164, 163, 186, 156, 28, 29, 26, 27, 25, 161, 42, 41	OMAP GPIOs 164, 163, 186, 156, 28, 29, 26, 27, 25, 71, 42, 41
RS485 transmit enable	OMAP GPIO 158	OMAP GPIO 137
WiFi Module	Wi2Wi W2CBW003	Murata LBEE5ZSTNC
Touch screen	4 wire resistive	4 wire resistive or projected capacitive.

GPIO and SYSFS

The recommended way to access the GPIO is using the SYSFS interface. This can be done using the command line (or scripts), or can be done from inside an application.

The Linux GPIO documentation can be found here:
https://www.kernel.org/doc/Documentation/gpio/sysfs.txt

This following page also has some useful examples:
http://falsinsoft.blogspot.co.uk/2012/11/access-gpio-from-linux-user-space.html

By default, the GPIOs on RE2G2 are already exported by the kernel, so you don't need to do that again. For example for GPIO1, the SYSFS name would be gpio164 and the following command would read the pin value

```
cat /sys/class/gpio/gpio164/value
```