

# Bare Metal on the BeagleBone

by Brian Fraser

Last update: November 26, 2014

## This document guides the user through:

1. Installing tools to support a bare metal application.
2. Writing, compiling, and downloading a sample bare metal application.

## Table of Contents

1. TI StarterWare Install and Build.....	2
2. Loading Bare Metal App via U-Boot.....	5
2.1 Booting to Bare Metal and Linux; Set default.....	9
3. Custom Bare Metal Application.....	10
4. Eclipse and Bare Metal Projects.....	11
5.Recovering from Corrupted uEnv.txt.....	12

**Note:** This guide has not yet been tested in the SFU Surrey Linux Lab (SUR4080). Some changes may be needed. Remember that you cannot execute any commands as root (using sudo) in the host OS in the Linux lab!

## Formatting:

1. Host (desktop) commands starting with \$ are Linux console commands:  
`$ echo "Hello world"`
2. Target (board) commands start with #:  
`# echo "On embedded board"`
3. Almost all commands are case sensitive in Linux and U-Boot.

## Revision History:

- Nov 21: Initial version published
- Nov 26: Added info about solving silently failing StarterWare install.

# 1. TI StarterWare Install and Build

1. On the host PC, download StarterWare for AM335x (version **v2.00.01.01**, ~35MB) from:  
<http://www.ti.com/tool/starterware-sitara>
  - Click the “Get Software” button, then select **both**:
    - Linux Installer, and
    - Beaglebone black patch (at bottom).
  - You will need to create a new account with TI. I suggest you enter the “Company” as the university's full name. You will be asked to agree that you are not violating US export restrictions. When asked for the type of business you may want to enter Education.
  - Note that this software is now out of support, meaning it is not being updated. This is not a problem for this course.
2. On the host, execute the `AM335X_StarterWare_02_00_01_01_Setup.bin` interactive command-line installer via the terminal.
  - You may need to change the installer to be executable before running it:  

```
$ chmod +x AM335X_StarterWare_02_00_01_01_Setup.bin
```
  - Install to `~/cmpt433/AM335X_StarterWare_02_00_01_01/`  
(You will likely have to type out the full path without using `~`).
3. Patch the StarterWare install to work with the BeagleBone Black:
  - Copy the downloaded `StarterWare_BBB_support.tar.gz` to the host's StarterWare folder:  
`~/cmpt433/AM335X_StarterWare_02_00_01_01/`
  - In the terminal, change to StarterWare directory and extract it (which is how it installs):  

```
$ cd ~/cmpt433/AM335X_StarterWare_02_00_01_01/  
$ tar xvfz StarterWare_BBB_support.tar.gz
```
4. Install Linaro GCC compiler for bare-metal application development:
  - Create folder for Linaro GCC:  

```
$ mkdir ~/cmpt433/linaro-gcc
```
  - Download the Linux installation tarball (~55MB) of version 4.7-2012q4 from:  
<https://launchpad.net/gcc-arm-embedded/4.7/4.7-2012-q4-major>  
Save file into the `~/cmpt433/linaro-gcc` folder. Expected file name is:  
`gcc-arm-none-eabi-4_7-2012q4-20121208-linux.tar.bz2`
  - Extract the compiler:  

```
$ cd ~/cmpt433/linaro-gcc  
$ tar xvfj gcc-arm-none-eabi-4_7-2012q4-20121208-linux.tar.bz2
```

5. Edit the StarterWare base Makefile to locate the compiler:

- Edit the base Makefile:

```
$ gedit ~/cmpt433/AM335X_StarterWare_02_00_01_01/build/armv7a/gcc/makedefs
```

- Change

```
ifndef PREFIX
PREFIX=arm-none-eabi-
endif
```

to

```
ifndef PREFIX
PREFIX=${LIB_PATH}/bin/arm-none-eabi-
endif
```

6. Setup path to tool-chain:

```
$ export LIB_PATH=${HOME}/cmpt433/linaro-gcc/gcc-arm-none-eabi-4_7-2012q4
```

- Note that this must be done each time you go to build the example files. (The samples distributed for this class set the LIB\_PATH in the makefiles).

7. Change the initialization assembly code to disable some hardware (MMU, instruction/data caching...) which is enabled by UBoot and conflicts with our use of interrupts.

- Edit AM335X\_StarterWare\_02\_00\_01\_01/system\_config/armv7a/gcc/init.S

```
$ cd ~/cmpt433/AM335X_StarterWare_02_00_01_01/
```

```
$ gedit system_config/armv7a/gcc/init.S
```

- Find the start of the Entry: process (~line 88) and add the following highlighted lines:

```
@*****
@
@*****
@
@ The reset handler in StarterWare is named as 'Entry'.
@ The reset handler sets up the stack pointers for all the modes. The FIQ and
@ IRQ shall be disabled during this. Then clear the BSS sections and finally
@ switch to the function calling the main() function.
@
Entry:
@
@ 2016: Brian Fraser's Fix for UBoot Messing with Interrupts
@ Disable the MMU, instruction and data caches.
@ Without this, the ISRs seem not to work with the latest UBoot code (2016)
@
        SUB    r0, r0, r0
        MCR    p15, 0, r0, c1, c0, 0
@
@ Set up the Stack for Undefined mode
@
        LDR    r0, =_stack                @ Read the stack address
        MSR    cpsr_c, #MODE_UND|I_F_BIT @ switch to undef mode
        MOV    sp,r0                      @ write the stack pointer
        SUB    r0, r0, #UND_STACK_SIZE   @ give stack space
```

8. Build the StarterWare example programs and utilities

```
$ cd ~/cmpt433/AM335X_StarterWare_02_00_01_01/  
$ cd build/armv7a/gcc/am335x/beaglebone  
$ make
```

9. Verify the build (path is long and takes up two lines):

```
$ ls -lA ~/cmpt433/AM335X_StarterWare_02_00_01_01/binary/armv7a/gcc/am335x/  
beaglebone/uart/Release  
total 100  
-rwxrwxr-x 1 brian brian 8588 Nov 5 21:17 uartEcho.bin  
-rwxrwxr-x 1 brian brian 104475 Nov 5 21:17 uartEcho.out  
-rw-rw-r-- 1 brian brian 8596 Nov 5 21:17 uartEcho_ti.bin
```

10. Troubleshooting:

- If the build fails, ensure you have the `LIB_PATH` environment variable set:

```
$ printenv LIB_PATH  
/home/brian/cmpt433/linaro-gcc/gcc-arm-none-eabi-4_7-2012q4
```

- If the build fails, double check you made the correct modification to the makefile and `init.S`; double check the syntax.
- If the build seems to complete but you are unable to find the `.bin` files (only `.out` files), you may be in the wrong director. Double check that the directory name is correct.
- If install of StarterWare fails silently, run `strace` on it:  
\$ `strace ./AM335X_StarterWare_02_00_01_01_Setup.bin`

If missing `/lib/ld-linux.so.2`, then:

```
$ sudo apt-get install lib32ncurses5
```

## 2. Loading Bare Metal App via U-Boot

This section guides you to configuring U-Boot for loading either your Linux kernel with its root-file-system, or loading a bare metal application via TFTP.

1. Host PC must be configured with a TFTP server. See the driver creation guide for steps on setting one up.
2. On host, make a symbolic link to file to download:

- Create the TFTP folder:  

```
$ mkdir ~/cmpt433/public/baremetal
```
- Copy the UART example (compiled in previous section) to the TFTP public folder:  

```
$ cd ~/cmpt433/AM335X_StarterWare_02_00_01_01/  
$ cd binary/armv7a/gcc/am335x/beaglebone/uart/Release/  
$ cp uartEcho.bin ~/cmpt433/public/baremetal/.
```
- Create the symbolic link:  

```
$ cd ~/cmpt433/public/baremetal  
$ ln -s uartEcho.bin download.bin
```
- General command to use for your future .bin files:  

```
$ ln -s daFileYouWantToDownload.bin download.bin
```
- **Explanation**

U-Boot's `uEnv.txt` file specifies how the board will boot and, in the case of a bare metal application, specifies what image to load from the host. However, we don't want to have to boot to Linux on the target each time we want to change the file to load.

So, we tell U-Boot to load “`download.bin`”, and then on the host we create a symbolic link to whichever file we actually want the target to load. This allows us to easily switch the file being downloaded by making a change on just the host.

3. Figure out (and test!) the command to load your bare metal application onto the BeagleBone for your setup.

- On your host, create a text file somewhere which can hold the UBoot command. It's easier to edit this way in a text editor than inside the UBoot prompt.
- **If using DHCP, you'll use the command (all on one line!):**

```
setenv autoload no;dhcp;setenv loadaddr 0x80000000; setenv serverip
192.168.0.102;setenv tftpboot
/home/user_name/cmpt433/public/baremetal;setenv bootfile
${tftpboot}/download.bin;tftp ${loadaddr} ${bootfile};echo *** Booting
to BareMetal ***;go ${loadaddr};
```

In this command, customize:

- 192.168.0.102: Change to the IP address of your TFTP server.
- user\_name to be the user name you use on your host PC. This must be a full path to the file, not relative to the shared directory.

**If using static IP addresses, you'll use the command (all on one line!):**

```
setenv ipaddr 192.168.2.2;setenv loadaddr 0x80000000; setenv serverip
192.168.2.1;setenv tftpboot
/home/user_name/cmpt433/public/baremetal;setenv bootfile
${tftpboot}/download.bin;tftp ${loadaddr} ${bootfile};echo *** Booting
to BareMetal ***;go ${loadaddr};
```

- Change IP addresses (ipaddr and serverip) accordingly.
- Change user\_name to fit your network and server configuration.

4. Run your bare metal application directly from UBoot:

- Reboot the target and enter UBoot by pressing any key at startup.
- Copy and paste your command from the previous step into UBoot, then execute it to boot:  
=> *Your copy-and-past command goes here*  
=> boot

5. Edit the U-Boot bare-metal script file on the target.

- Boot the target into Linux (without pressing any key at UBoot).
- Copy the current boot file to a backup, and to one for booting Linux:  
# cd /boot  
# cp uEnv.txt uEnv.bak.baremetal  
# cp uEnv.txt uEnvLinux.txt

**This is very important! Without these files you may not be able to boot to Linux!**

- Create a file for loading the bare metal application:  
# nano /boot/uEnvBareMetal.txt

Make the contents of the file be the following (copying your now debugged UBoot command from above for launching your application). (i.e., prepend `uenvcmd=` to your command and make that the contents of the file):

`uenvcmd=YourUbootCommandGoesHere`

## 6. Setup UBoot to UBoot automatically boot to your mare-metal app<sup>1</sup>:

```
$ cp /boot/uEnvBareMetal.txt /uEnv.txt
```

- **To boot back into Linux, you'll need to follow the directions in section 2.1.**
- A note on `uEnv.txt` locations:
  - `/boot/uEnv.txt`: Used for booting a Linux kernel.
  - `/uEnv.txt`: Used for booting a bare metal application.

## 7. Reboot the target to load and run the sample bare metal application.

- If running the `uartEcho.bin`, whatever you type on the screen will be echo'd back to you.
- Note: Board may reboot within 45 seconds of launching bare metal application. This is due to the watchdog timer, and is expected. We'll cover in class how to disable this.

## 8. Troubleshooting:

- If unable to load your application via UBoot, it may display a useful error message before rebooting. In which case, you can either capture the output to a file for analysis, or try to power-down the target (pull its USB power) fast enough when the message appears.
- If you are unable to download the file `download.bin`, ensure that you have correctly created the file link. Do a directory listing on the `baremetal/` folder and see what `download.bin` links to (points to) and ensure that target folder exists correctly.

```
$ ls -la ~/cmpt433/public/baremetal
```

```
..
-rwxrwxr-x 1 brian brian 8256 Nov 17 23:33 bm_uart.bin
lrwxrwxrwx 1 brian brian 11 Nov 17 23:33 download.bin -> bm_uart.bin
-rwxrwxr-x 1 brian brian 8488 Nov 17 23:23 uartEcho.bin
```

- Ensure you edited the contents of `/uEnv.txt` correctly: must have IP configuration correct and user name in path correctly. Try the command out via the UBoot prompt first.
- Ensure your bare metal script is in `/uEnv.txt` on the target. If you place it in the `/boot/` folder you may prevent Linux from booting, and cause it to fail to load your bare metal application.
- You may find that rebooting the target and retrying to load the bare metal application via TFTP again may work if it initially failed.

1 In UBoot versions which have a boot macro which uses the `bootenv` variable correctly, one may instead boot into UBoot (pressing any key at startup), and then setting the `bootenv` environment variable to select the boot script.

```
=> setenv bootenv uEnvBareMetal.txt
```

```
=> boot
```

## 9. Sample capture of full boot process, booting into `bm_uart.bin`; its output in bold.

```
U-Boot 2016.03-00001-g148e520 (Jun 06 2016 - 11:27:44 -0500), Build: jenkins-
github_Bootloader-Builder-395

        Watchdog enabled
I2C:    ready
DRAM:   512 MiB
Reset Source: Global external warm reset has occurred.
Reset Source: Global warm SW reset has occurred.
Reset Source: Power-on reset has occurred.
MMC:    OMAP SD/MMC: 0, OMAP SD/MMC: 1
Using default environment

Net:    <ethaddr> not set. Validating first E-fuse MAC
cpsw, usb_ether
Press SPACE to abort autoboot in 2 seconds
Card did not respond to voltage select!
gpio: pin 56 (gpio 56) value is 0
gpio: pin 55 (gpio 55) value is 0
gpio: pin 54 (gpio 54) value is 0
gpio: pin 53 (gpio 53) value is 1
Card did not respond to voltage select!
Card did not respond to voltage select!
switch to partitions #0, OK
mmc1(part 0) is current device
Scanning mmc 1:1...
gpio: pin 56 (gpio 56) value is 0
gpio: pin 55 (gpio 55) value is 0
gpio: pin 54 (gpio 54) value is 0
gpio: pin 53 (gpio 53) value is 1
switch to partitions #0, OK
mmc1(part 0) is current device
gpio: pin 54 (gpio 54) value is 1
Checking for: /uEnv.txt ...
264 bytes read in 30 ms (7.8 KiB/s)
gpio: pin 55 (gpio 55) value is 1
Loaded environment from uEnv.txt
Importing environment from mmc ...
Checking if uenvcmd is set ...
gpio: pin 56 (gpio 56) value is 1
Running uenvcmd ...
link up on port 0, speed 100, full duplex
BOOTP broadcast 1
DHCP client bound to address 192.168.0.113 (236 ms)
link up on port 0, speed 100, full duplex
Using cpsw device
TFTP from server 192.168.0.102; our IP address is 192.168.0.113
Filename '/home/brian/cmpt433/public/baremetal/download.bin'.
Load address: 0x80000000
Loading: #
        1.3 MiB/s
done
Bytes transferred = 8256 (2040 hex)
*** Booting to BareMetal ***
## Starting application at 0x80000000 ...
Demo bare metal UART application
Note: This may reset in about 45seconds... why?
Count = 10000000
Count = 20000000
Count = 30000000
Count = 40000000
Count = 50000000
```



## 2.1 Booting to Bare Metal and Linux; Set default

1. In UBoot, you can change the boot configuration<sup>2</sup>:

- **Boot to Linux by wiping contents of /uEnv.txt**

```
=> ext4write mmc 1:1 0x82000000 /uEnv.txt 0  
=> boot
```

- **Boot to bare metal by copying /boot/uEnvBareMetal.txt to /uEnv**

```
=> ext4load mmc 1:1 0x82000000 /boot/uEnvBareMetal.txt  
=> ext4write mmc 1:1 0x82000000 /uEnv.txt ${filesize}  
=> boot
```

- Since this actually overwrites the /uEnv.txt file, the change will stay in effect until you overwrite the file.
- Note that to boot Linux, the boot file is /boot/uEnv.txt, but for booting bare metal it is /uEnv.txt because the UBoot loading scripts treat those two files differently.

2. You can set the default boot option if you are booted into Linux by changing the uEnv.txt file on the target's MMC:

- Setup to boot to **Linux**:

```
# rm /uEnv.txt
```

Setup to boot to **bare metal**:

```
# cp /boot/uEnvBareMetal.txt /uEnv.txt
```

- Once changed, when the board reboots it will execute the desired boot option.

3. In U-Boot, you can list the files in the /boot/uboot/ folder:

```
=> ls mmc 1:1
```

- This can be useful because you need to know the name of the uEnv.txt file if you wish to select one to boot (because you can't use Linux to "ls" the folder if you can't boot Linux).

```
=> ls mmc 1:1 /boot  
<DIR>      4096 .  
<DIR>      4096 ..  
<DIR>      4096 dtbs  
<DIR>      4096 uboot  
          1336 uEnv.opt1  
           492 SOC.sh  
          1401 uEnv.bak.uboot  
3300682 System.map-4.4.12-ti-r31  
147437 config-4.4.12-ti-r31  
4817115 initrd.img-4.4.12-ti-r31  
          1367 uEnv.bak.audio  
          1400 uEnv.txt  
7777640 vmlinuz-4.4.12-ti-r31  
          1400 uEnv.bak.baremetal  
          1400 uEnvLinux.txt  
           264 uEnvBareMetal.txt
```

2 On systems which support the bootenv in UBoot correctly, you can use:

to boot to **Linux**

```
=> setenv bootenv uEnvLinux.txt; boot
```

to boot to **bare metal**

```
=> setenv bootenv uEnvBareMetal.txt; boot
```

### 3. Custom Bare Metal Application

The following files are required to build your own custom bare metal application:

- **foo.c:**  
Your application code which is expected to run bare metal.
- **load\_script.lds:**  
Load script which controls how the linker builds the final executable (.bin).
- **makefile:**  
builds and links the application. Uses the StarterWare base Makefile to set many of the configuration options.

To create your own application:

1. Copy the example code for the `bm_uart` project on the course website to a new folder. Likely in your `~/cmpt433/work` folder.
2. Rename the copied `bm_uart.c` file to match your application's purpose.
3. Edit the `makefile`:
  - Change `APPNAME` to be your C file's name.
  - Add any extra libraries that are required to run.
4. Run `make` to cross-compile the code for the `arm-none-eabi` environment and copy the final binary file (.bin) to the TFTP public folder for download.
5. If using the above suggested way of having U-Boot load the file `download.bin` as a symbolic link to the desired file, you'll have to update the link:

```
$ cd ~/cmpt433/public/baremetal/  
$ rm download.bin  
$ ln -s foo.bin download.bin
```

## 4. Eclipse and Bare Metal Projects

1. Open Eclipse and create new C project for existing makefile.
2. Point to the folder of your application containing the makefile.
3. Next you can add StarterWare's .h files for the project's includes. This is optional, but it will allow Eclipse to give better error feedback and coding support as you write your code.
  - Right click your project's name in the Project Explorer.  
Select Properties → C/C++ General → Paths and Symbols
  - “Includes” tab → “GNU C” language, and click the Add... button.
  - Select “File system”... and select the full path for directory to include.
    - Directories to include are (note: don't use the ~; write full path name):  
~/cmpt433/AM335X\_StarterWare\_02\_00\_01\_01/include  
~/cmpt433/AM335X\_StarterWare\_02\_00\_01\_01/include/hw  
~/cmpt433/AM335X\_StarterWare\_02\_00\_01\_01/include/armv7a  
~/cmpt433/AM335X\_StarterWare\_02\_00\_01\_01/include/armv7a/am335x
4. To allow Eclipse to show you the implementation for C functions in the StarterWare library:
  - In Eclipse, create new C project with Makefile for AM335X\_StarterWare\_02\_00\_01\_01
  - Set folder to: ~/cmpt433/AM335X\_StarterWare\_02\_00\_01\_01
  - You don't need to set the compiler prefix, or any other options; we're not building this, we are using it as a reference.
  - In your application's project (such as for bm\_uart):
    - Right-click project → Properties → Project References
    - Check the AM335X\_StarterWare\_02\_00\_01\_01 project.
  - Now, when you control-click a library function call, you'll be able to view the full source.

## 5. Recovering from Corrupted uEnv.txt

If you edit `/boot/uEnv.txt` and it becomes corrupted, or you load a device tree which does not support the on-board eMMC then your board may fail to boot. These steps should help you recover.

1. View your board's boot process using the serial port on the board (via the `screen` program).

- If you see:  
Checking for: `/uEnv.txt` ...  
Checking for: `/boot.scr` ...  
Checking for: `/boot/boot.scr` ...  
Checking for: `/boot/uEnv.txt` ...  
\*\* Invalid partition 2 \*\*  
....

It likely means you have deleted `/boot/uEnv.txt`

2. Reboot your board (may need to use reset button on BeagleBone).

When booting begins, you should see something like:

```
U-Boot SPL 2016.03-00001-g148e520 (Jun 06 2016 - 11:27:44)
Trying to boot from MMC
bad magic
```

```
U-Boot 2016.03-00001-g148e520 (Jun 06 2016 - 11:27:44 -0500), Build:
jenkins-github_Bootloader-Builder-395
```

```
        Watchdog enabled
I2C:    ready
DRAM:   512 MiB
Reset Source: Global external warm reset has occurred.
Reset Source: Power-on reset has occurred.
MMC:    OMAP SD/MMC: 0, OMAP SD/MMC: 1
Using default environment
```

```
Net:    <ethaddr> not set. Validating first E-fuse MAC
cpsw, usb_ether
Press SPACE to abort autoboot in 2 seconds
=>
```

3. Press SPACE as soon as it begins booting to enter the UBoot prompt.

4. Listing files in the `/boot/` folder:

```
=> ext4ls mmc 1:1 /boot
```

5. Copy a file, changing SOURCE and TARGET as needed:

```
=> ext4load mmc 1:1 0x82000000 SOURCE
=> ext4write mmc 1:1 0x82000000 TARGET ${filesize}
```

- For example, to make a backup copy of your current `uEnv.txt` use:  
=> `ext4load mmc 1:1 0x82000000 /boot/uEnv.txt`  
=> `ext4write mmc 1:1 0x82000000 /boot/uEnv.bak.uboot ${filesize}`
- For example, to restore `/boot/uEnv.bak.audio` use:  
=> `ext4load mmc 1:1 0x82000000 /boot/uEnv.bak.audio`  
=> `ext4write mmc 1:1 0x82000000 /boot/uEnv.txt ${filesize}`

- Note the `${filesize}` variable is set when you do an `ext4load` command.
6. Boot the board, which loads `/boot/uEnv.txt`:  
`=> boot`
  7. Display a file:  
`=> ext4load mmc 1:1 0x82000000 /boot/uEnv.txt`  
`=> md 0x82000000 ${filesize}`
  8. Troubleshooting:
    - List the files on the eMMC, view their content, and try and find a `uEnv.txt` file you want to boot!
    - You can possibly boot from a uSD card and use that to access the eMMC. This is beyond the scope of this guide.
    - If all else fails, you can wipe the BeagleBone and return to a clean state using a uSD card.