COMPILING YOUR FIRST LKM AND LOADING ON THE EVB

In order to compile Embedded Linux Applications \ Modules and download them to the Evaluation board. You need first to complete this guide, please make sure you have done so before continue

This tutorial is divided into 2 parts:

A – The first part emphases on the needed software for the course and includes the following:

1. Install the Arm Cross Compiler
2. Downloading the BeagleBone Green Kernel source-code
3. Configure the build environment
4. Applying the BeagleBone Green kernel configuration
5. Compiling the Kernel Source Code

B – The second part goes through on how to compile your first Kernel module and download it to the EVB.
Once you have installed all the needed Software on your Linux Machine (or Virtual Machine), we can go ahead and start compiling code.

In order to compile and load your first Linux Kernel Module, we need to get the system ready, so some additional preparations are in order (in this precise order):

1. Get to know the BeagleBone Green
2. Connect the EVB to the Laptop \ P.C.
3. Turn on the EVB and make sure Linux is loading.
4. Compile the "hello-word" module on the Linux Machine
5. Copy the hello.ko (kernel-image-module) to the EVB file-system

6. Load the hello.ko module into the EVB kernel

Part A - Installing Course SW

**** You need to run as root just enter:**
Red-Hat \ Centos: ***su***
Debian \ Ubunto: ***sudo su***

Please complete the following steps:

1) Installing the Arm Cross Compiler:
   a) To be able to compile the C code on x86 machine for Arm architecture,we need to install Linux Arm GCC cross compiler.
   b) *Log in as administrator and type:*
   apt-get install gcc-arm-linux-gnueabi
   inside a terminal,or download the package from the link below:
   https://drive.google.com/open?id=1OAoqzhlKZuT_5j1RFg76ZxlwhrZatv_b

   c) After installing the package please run:
   arm-linux-gnueabi-gcc -v
   to check that the installation completed successfully and you can see the current version of cross compiler.

2) Download th **BBG-kernel source-code (kernel version 4.9)**, please do the following:
   a) Download the BBG-kernel source code using one of following links:
   Google drive:
   https://drive.google.com/open?id=1WpPzOEgWbh8qFO0ftcIrL8wTs8HVG1Lk

Github (choose Download ZIP option):
https://github.com/beagleboard/linux/archive/4.9.82-ti-r102.zip

b) Create a new directory called kernels under /usr/src on you pc (so you"ll get **/usr/src/kernels**) , with the following command:
mkdir /usr/src/kernels

c) Move **linux-4.9.82-ti-r102.zip** to /usr/src/kernels with the command:
mv linux-4.9.82-ti-r102.zip /usr/src/kernels

d) Unzip the **linux-4.9.82-ti-r102.zip** file with the command in a same directory:
unzip linux-4.9.82-ti-r102.zip

e) Make sure the kernel source-code directory is named "linux-4.9.82-ti-r102" is it matches the lessons Makefiles:
KERNELDIR=/usr/src/kernels/linux-4.9.82-ti-r102

3) Configure the kernel build environment :

a) By default, kernel build system uses x86 architecture and native gcc compiler.
However, this is not correct for ARM based systems. So, It is mandatory to specify the ARCH and CROSS_COMPILE in order to compile the kernel for ARM architecture.
Based on these definitions, kernel build system understands which architecture and cross compile to be used.
So export them, ‘make’ command automatically fetches these definitions during compilation.

b) Please use the following commands to export them:
export ARCH=arm
export CROSS_COMPILE=arm-linux-gnueabi-

4) Applying the Beaglebone kernel configuration :

a) Open the terminal in a Kernel folder and lf in as Administrator(sudo su).

b) Kernel configuration for Beaglebone black is bb.org_defconfig
To apply this configuration, run below command.
make bb.org_defconfig

c) A textual program will start don't change anything, just press tab and exit, the configuration will be saved in .config file

5) Compiling the Kernel Source Code
make -j6

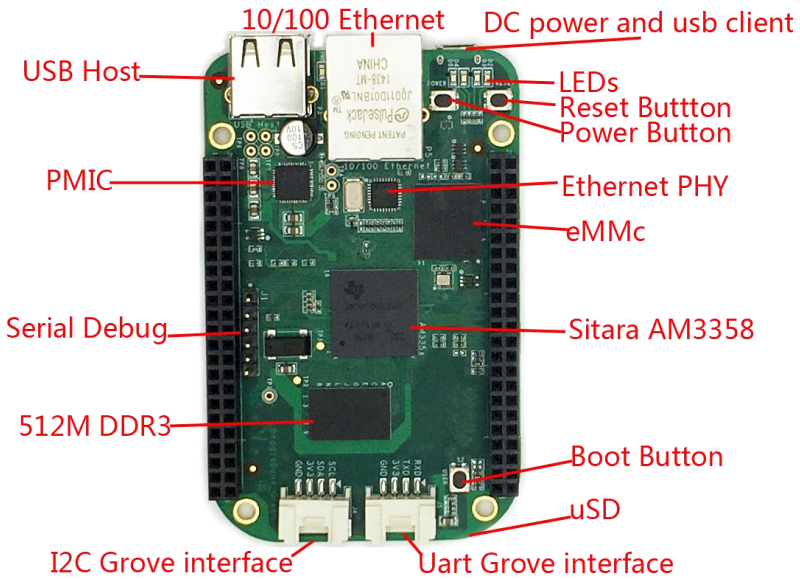
6) After the Kernel finishing the compilation the Kernel headers are ready for use.

=Part B – Running your first LKM

a) **Plug in BeagleBone via USB**

b) Use the provided USB cable to plug BeagleBone into your computer. This will both power the board and provide a development interface. BeagleBone will boot Linux from 4GB eMMC and operates as a flash drive. This provides you with a local copy of the documentation and drivers.

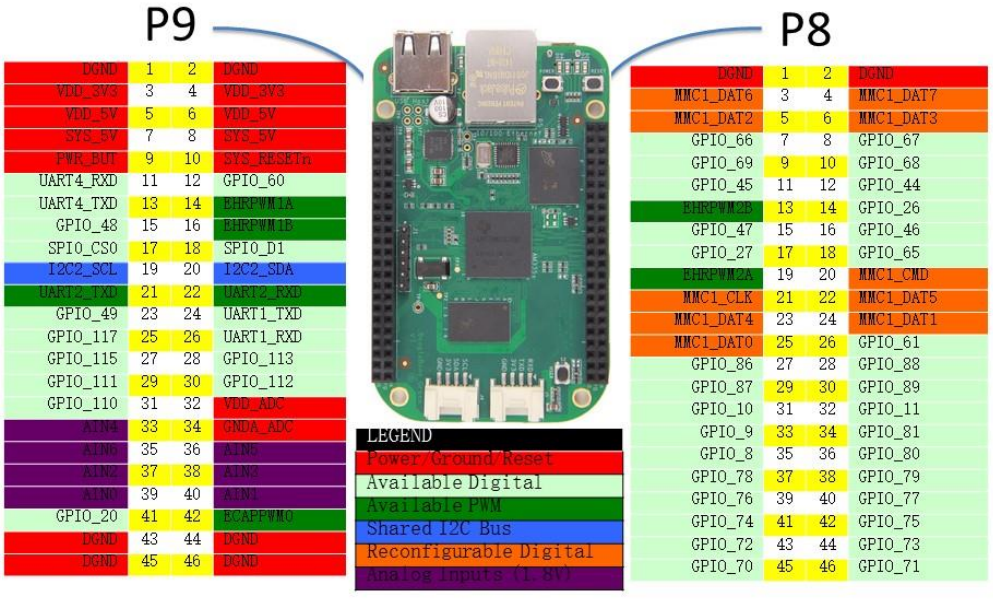
b) BeagleBone Green hardware details



c) Beaglebone capes I/O available configurations.

Cape Expansion Headers

-



a) **Connecting to the EVB SSH server:**
   a) After the Linux complete the booting process on EVB, he setup the SSH server and open virtual port on a host PC. We can use this virtual port to connect to the server and access the BBG terminal.
   b) To connect to the SSH server,open the Terminal and type the following command:
      ssh debian@192.168.7.2
   c) When asked for password type: **temppwd.**
   d) You now logged in a EVB terminal and can use all bash commands to control the board.

4) Compiling the "hello-word" module on the Linux Machine

a. Download the lesson from the site: Lesson-4: Linux Kernel Modules
b. Unzip the lesson, we'll be focusing on lesson-4.1, go to that directory
c. Open the Makefile with an editor (gedit for example) Make sure that the following are correct:
   i. The " build=INTEL" is remarked so the kernel will use the else <section> and build the module for ARM architecture.

   ii. The Cross-Compiler: *arm-linux-gnueabi-* is working properly and is in configured within the PATH environment variable.

iii. The KERNELDIR variable is mapped to the correct kernel source code location.
KERNELDIR=/usr/src/kernels/linux-4.9.82-ti-r102

d. Go ahead and compile the lesson-4.1 (hello world) kernel module using the make command:
> *make*

e. If there were no errors, you should get a ***hello.ko*** file

5) Copying the hello.ko (kernel-image-module) to the EVB file-system can be done in following way:

a. Open the terminal window from the compiled module dir.

b. Type the following command:
scp hello.ko debian@192.168.7.2:/home/debian

c. When asked for password enter: **temppwd**. The file will be upload to the selected path using SSH server.

d. Run the ls command from home directory to check the the file uploaded successfully.

6) Loading the hello.ko module into the EVB kernel

a. Before loading the hello.ko module' clean the kernel message buffer:
dmesg -c

b. load the hello.ko module:
insmod hello.ko

from the module directory.

c. make sure the module has loaded:
lsmod (look for the module name)

d. have a look at the modules messages:
dmesg -c

7) Unloading the hello.ko module from the EVB's kernel

a. Use the following command:
rmmod hello.ko

b. have a look at the modules messages:
dmesg -c

c. *Enjoy.*