Hermes™ IoT Development Platform Debian Distribution Build Guide

1 Platform Overview

Hermes is built around a BeagleBone Black (BBB) (www.beagleboard.org) single-board computer controlling a wireless sensor reader board for communicating with sensor tags.

2 Flasher Image

The following Hermes released BBB flasher image: **rfmicron-hermes-bbb-release-image-1.0.img.xz** can be downloaded from the following web page:

https://github.com/RFMicron/Hermes/blob/master/BeagleBone_Black_Images.md.

This image can be used to program the exact Debian distribution that it is shipped with the BBB in the Hermes IoT Development Platform. The main goal of this document is to describe the process that creates this image. Users can follow this process to add further customizations for their own purposes.

3 Flasher Image Build Process

The main steps to build the flasher image are:

1. Download and write the Linux distribution baseline to a micro SD card. This step can be performed in a Windows, OS X or Linux machine. As an option, this baseline can be manually built using a Linux machine.
2. Download the Hermes Debian distribution customization packages file and extract the files using a Linux machine.
3. As an option the following Hermes components can be built manually:
    a. Customized Linux kernel
    b. Qt and Qwt
    c. The auxiliary applications
    d. Can-utils
    e. BBB examples
    f. The Hermes demo application
4. Replace the Hermes components that were manually built.
5. Connect the SD card to the Linux machine.

6. Configure the script **customize_SD_Card.rc** and execute it using root privileges. This script copies all the necessary files to the SD card and performs other configurations.
7. Disconnect the SD card.

4 Linux Distribution Baseline

Hermes uses as a baseline a Debian distribution created by Robert C. Nelson that he customized for the BBB. Robert Nelson is an Application Engineer at Digi-Key Electronics and he is a very well-known contributor to the BBB community. The following links show some of his main contributions:

https://github.com/RobertCNelson

https://eewiki.net/display/linuxonarm/BeagleBone+Black

https://plus.google.com/u/0/106813818225399872098/posts

https://rcn-ee.com/

The specific Debian baseline image that is used is: **bone-debian-7.9-lxde-4gb-armhf-2015-11-03-4gb.img.xz** located at:

https://github.com/RFMicron/Hermes/blob/master/BeagleBone_Black_Images.md

It is possible not to use this image and instead rebuild it from sources. This is a laborious and lengthy process that requires advanced Linux skills. For users interested in rebuilding the baseline from sources, the following page created by Robert Nelson, provides all the necessary information:

https://eewiki.net/display/linuxonarm/BeagleBone+Black

This image contains Debian version **7.9** code named Wheezy. It uses the Linux kernel version **3.8.13-bone79**. The distribution contains LXDE (abbreviation for Lightweight X11 Desktop Environment) which is a free desktop environment with comparatively low resource requirements ideal for embedded platforms.

5 Component Files

The components required for the customization of the flasher image are included in the following file: **rfmicron-hermes-debian-customization-packages-1.0.tar.gz**. This file can be downloaded from the following page:

https://github.com/RFMicron/Hermes/tree/master/DebianCustomization

6 Linux Kernel Customization (Optional)

The Linux kernel requires customization for two reasons:

1. Add support for Hermes LCD screen, buttons and LEDs using a cape configuration file.
2. Upgrade and modify the touchscreen device driver used by the Hermes board.

6.1 Operating System Requirements

The Linux kernel must be configured using a recent 64-bit release of Debian, Fedora or Ubuntu. The use of OS virtualization software is not recommended. Nonetheless, the customization for Hermes was successfully done using Ubuntu 14.04 LTS 64-bit running on a virtual machine using Oracle VM Virtual Box version 5.0.14 on Windows 7 Professional 64-bit.

6.2 Compiler Setup

The compiler used for the customization is the compiler recommended by Robert Nelson. It is the Linaro GCC 5.2-2015.11. The compiler can be downloaded and installed using the following three commands:

```
wget -c https://releases.linaro.org/components/toolchain/binaries/5.2-2015.11/arm-linux-gnueabihf/gcc-linaro-5.2-2015.11-x86_64_arm-linux-gnueabihf.tar.xz

tar xf gcc-linaro-5.2-2015.11-x86_64_arm-linux-gnueabihf.tar.xz

export CC=`pwd`/gcc-linaro-5.2-2015.11-x86_64_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-
```

The installation can be verified with the following command:

```
$(CC}gcc --version
```

The output should include the expected compiler version: Linaro GCC 5.2-2015.11.

WARNING: the exported CC environment variable remains valid only for the duration of the terminal session.

6.3 Building the Kernel

Use the following four commands to download, configure and build the kernel, its modules and device trees:

```
git clone https://github.com/RobertCNelson/bb-kernel

cd bb-kernel/
```

```
git checkout origin/am33x-rt-v3.8 -b tmp

./build_kernel.sh
```

This process can last from a fraction of an hour to several hours depending on the computer capabilities. The final products will be located in the **deploy** subdirectory.

6.4 Hermes Cape

Copy the **RFMicron_Hermes_Cape.patch** file, located in the **kernel** subdirectory of the customization packages, to the same directory where the **bb-kernel** directory is located. Apply the patch using the following command:

```
patch –p1 < RFMicron_Hermes_Cape.patch
```

This patch will add the **RFMICRON-HERMES-00A0.dts** cape file in the **bb-kernel/KERNEL/firmware/capes** directory and it will update the **makefile** file in order to include the new cape in the build process.

6.5 Touchscreen Device Driver

For this step use the files **edt_ft5x06_3.19.c** and **RFMicron_Hermes_Touchscreen.patch** located in the **kernel** subdirectory of the customization packages. The original location of the device driver file is at:

https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/tree/drivers/input/touchscreen/edt-ft5x06.c?h=v3.19

Place the files in the same directory where the **bb-kernel** directory is located. Update the file to version 3.19 and apply the Hermes patch using the following commands:

```
rm bb-kernel/KERNEL/drivers/input/touchscreen/edt_ft5x06.c

mv edt_ft5x06_3.19.c bb-kernel/KERNEL/drivers/input/touchscreen/edt_ft5x06.c

patch –p1 < RFMicron_Hermes_Touchscreen.patch
```

6.6 Rebuild the Customized Kernel

Rebuild the kernel in order to include the Hermes cape and touchscreen driver updates. Use the following commands:

```
cd bb-kernel/

./tools/rebuild.sh
```

This process should be fairly fast. Depending on the computer capabilities it could range from seconds to a couple minutes.

6.7 Customized Kernel Files

There are two files of interest that are located at **bb-kernel/deploy**. The files are:

- `3.8.13-bone79.zImage`
- `3.8.13-bone79-modules.tar.gz`

Copy these files to the **kernel** subdirectory of the customization packages**.**

7 Qt 5.5.1 Build Process (Optional)

Unfortunately there is no known way to cross-compile Qt 5.5.1 for the BBB so it can be used both locally and from a cross-compile environment. Therefore this process has to be performed in a BBB. This process requires so much memory and disk space that it is not possible to be performed using only the BBB's onboard eMMC. It has to be done using a micro SD card of at least 8 GB. In order to speed up the process it is highly recommended to use the eMMC for a swap file and for the Qt sources.

7.1 SD Card Preparation

Use an SD card of at least 8 GB. Program the SD card using the baseline image: **bone-debian-7.9-lxde-4gb-armhf-2015-11-03-4gb.img.xz** located at:

https://github.com/RFMicron/Hermes/tree/master/DebianCustomization

The SD card will be programed with two partitions. The first one is about 100 MB in size and its type is FAT. The second partition is about 3.5 GB in size and its type is ext4. The rest of the SD card is unused. The next step is to enlarge the second partition so it can use the rest of the SD card. One way of doing it is through the following steps:

1. Connect the SD card to a Linux computer. Don't boot from the SD card.
2. Install the **GParted** application. This can be done in most distributions using the command: **sudo apt-get install gparted**
3. **GParted** must be executed using root privileges. This a graphical application that will show the partitions.
4. Select the second partition of the SD card and extend it so it covers the rest of the space in the SD card.
5. Then select the **Apply All Operations** from the **Edit** menu to commit the operation. **WARNING**: **GParted must be used with extreme care. A simple undesired operation can render the whole Linux installation useless as well as completely erase all data.**
6. Close the **GParted** application and remove the SD card.

7.2 eMMC Configuration

Connect the SD card to a BBB and boot from the SD card. The onboard eMMC has two partitions that will be mounted at **/media/BEAGLEBONE_** and **/media/rootfs**. Erase the contents of the second partition using the following command:

```
sudo rm –r /media/rootfs
```

WARNING: **All contents on the second partition of the BBB's eMMC will be erased.**

7.3 Swap File Setup

Use the following commands to create a swap file in the BBB's eMMC:

```
sudo touch /media/rootfs/swap.img

sudo chmod 600 /media/rootfs/swap.img

sudo dd if=/dev/zero of=/media/rootfs/swap.img bs=1024k count=1200

sudo mkswap /media/rootfs/swap.img

sudo swapon /media/rootfs/swap.img
```

Verify that the new swap file is working using the command:

```
swapon –s
```

WARNING: **If the BBB is restarted then the command `sudo swapon /media/rootfs/swap.img` must be executed again. To automatically activate the swap file after system reboot add the line: `/media/rootfs/swap.img none swap sw 0 0` to the file `/etc/fstab`.**

7.4 Build Process

Create a directory for the Qt sources using the following commands:

```
sudo mkdir /media/rootfs/q

sudo chown debian:debian /media/rootfs/q
```

Download the Qt 5.5.1 source code for Linux file: **qt-everywhere-opensource-src-5.5.1.tar.xz** located at:

https://github.com/RFMicron/Hermes/tree/master/DebianCustomization

Extract the contents to the newly created folder using the command:

```
tar xf qt-everywhere-opensource-src-5.5.1.tar.xz –C /media/rootfs/q
```

Open the configuration file using the following command:

```
nano /media/rootfs/q/qt-everywhere-opensource-src-
.5.5.1/qtbase/mkspecs/devices/linux-beagleboard-g++/qmake.conf
```

Change the **QT_QPA_DEFAULT_PLATFORM** variable value from **eglfs** to **xcb** and the value of the **COMPILER_FLAG –mfloat-abi** from **softfp** to **hard**.

Next create a directory for the Qt target location using the following commands:

```
sudo mkdir /usr/local/qt5

sudo chmod 777 /usr/local/qt5
```

Then create a folder for the Qt build process:

```
cd ..

mkdir qt-build
```

Next configure the build process with the following command:

```
cd qt-build

/media/rootfs/q/qt-everywhere-opensource-src-5.5.1/configure –prefix
/usr/local/qt5 –release –opensource –confirm-license –no-largefile –qt-xcb –
no-compile-examples –v –sysroot /
```

Finally, build and install Qt with the following commands:

```
make

make install
```

The build process will take about 18 hours. This time can be much longer in BBBs with slower eMMCs. At the end, the tar file can be created with the following command:

```
tar –czf qt5.tar.gz /usr/local/qt5
```

8 Qwt 6.1.2 Build Process (Optional)

Qwt is an extension to the libraries of the Qt Project. The Qwt library contains widgets and components which are primarily useful for technical and scientific purposes.

The build process must be performed in a BBB that has Qt 5.5.1 installed. The following steps describe the build process:

1. Download the source code from: https://sourceforge.net/projects/qwt/files/qwt/

2. Extract the source code with the following command: **tar xvf qwt-6.1.2.tar.bz2**
3. Then execute the following commands:
    a. `cd qwt-6.1.2`
    b. `qmake qwt.pro`
    c. `make`
    d. `sudo make install`
4. The binaries will be installed at: **/usr/local/qwt-6.1.2**. Create the tar file with the following command:
    a. `tar –czvf qwt-6.1.2.tar.gz /usr/local/qwt-6.1.2`

9 Package Auxiliary Applications (Optional)

The Hermes distribution includes three applications:

1. Florence. This is a virtual keyboard.
2. jEdit. This is a text editor specialized for programming tasks.
3. Minicom. This is a command line utility useful for UART testing.

The packages required to install the three applications and their dependencies are located in the **apps** subdirectory. Users can manually download the packages and build the tar files using the script **downloadPackages.rc** located in the **apps** subdirectory. Follow the instructions described at the top of the script to properly configure the package sources. The script must be executed using a BBB with Debian 7.9.

10 Can-utils Build Process (Optional)

The Hermes distribution includes Can-utils. These utilities are very useful for testing and troubleshooting the CAN BUS interface. Follow the next steps to build and package the utilities:

```
1. git clone https://github.com/linux-can/can-utils.git
2. cd can-utils
3. ./autogen.sh
4. ./configure
5. make
6. sudo make install
7. cd ..
8. tar –czvf can-utils.tar.gz can-utils
```

The previous steps must be executed using a BBB with Debian 7.9.

11 BeagleBone Black Examples (Optional)

The Hermes distribution includes the examples from the following book: **Exploring BeagleBone: Tools and Techniques for Building with Embedded Linux**. The author of the book is **Derek Molloy** and the ISBN is **1118935128**. The book and these examples are very useful in understanding the BBB capabilities. Users can directly download these examples using the following command:

```
git clone https://github.com/derekmolloy/exploringBB.git
```

12 Hermes Demo Application Build Process (Optional)

The Hermes application can be built using the following commands:

```
1. cd hermes
2. qmake hermes.pro
3. make
```

13 Installation Scripts

There are two installation scripts. The first one, **hermesSetScript.rc**, copies all the required files to the SD card. This script must be executed using root privileges. The second one, **hermesRunOnceScript.rc**, is copied by the first script to the BBB's debian account home folder (`/home/debian`) where it is executed the first time the BBB boots. This script removes itself at the end of its execution. The second script installs the auxiliary applications and performs other configurations.

13.1 hermesSetScript.rc

Before executing this script it must be configured to target the right path where the SD card is mounted. This can be done by setting the **ROOTFS** variable at the top of the script. This script performs the following tasks:

1. Replaces the Linux kernel and its modules.
2. Adds the Hermes cape configuration files. It also copies these files to the **/home/debian/hermesDeviceTrees** folder so users can easily review them.
3. Copies the file **99-gpio.rules**. This file enables the use of GPIO without having to use root privileges.
4. Installs Qt and Qml.
5. Adds commands to initialize the CAN BUS interface.
6. Adds the Hermes source code and compiled application.
7. It adds the file **libfm.conf**. The purpose of this file is to enable single-click selection and execution. This is very useful for the touchscreen interface.
8. It copies the Florence, jEdit and Minicom installation packages.

9. It copies the Can-utils and the BBB examples.
10. It copies the file **pointercal.xinput**. The purpose of this file is to disable the screen calibration which is not needed for a capacitive touchscreen.
11. It add the **hermesRunOnceScript.rc**.
12. It turns the SD card into a flasher card.

13.2 hermesRunOnceScript.rc

This script is automatically executed only one time after the first boot of the operating system. It performs the following tasks:

1. Enables the use of Qwt.
2. Enables the use of the single-click feature.
3. Creates the Hermes demo application icon in the desktop.
4. Installs the Florence, jEdit and Minicom applications and removes all the installation packages at the end.
5. It removes itself.

14 Notices

RFMicron, Inc., ("RFMicron") conditionally delivers this document to a single, authorized customer ("Customer"). Neither receipt nor possession hereof confers or transfers any rights in, or grants any license to, the subject matter of any drawings, designs, or technical information contained herein, nor any right to reproduce or disclose any part of the contents hereof, without the prior written consent of RFMicron.

RFMicron reserves the right to make changes, at any time and without notice, to information published in this document, including, without limitation, specifications and product descriptions. This document supersedes and replaces all information delivered prior to the publication hereof. RFMicron makes no representation or warranty, and assumes no liability, with respect to accuracy or use of such information.

Customer is solely responsible for the design and operation of its applications and products using RFMicron products. It is the Customer's sole responsibility to determine whether the RFMicron product is suitable and fit for Customer's applications and products planned, as well as for the planned application and use of Customer's end user(s). RFMicron accepts no liability whatsoever for any assistance provided to Customer at Customer's request with respect to Customer's applications or product designs. Customer is advised to provide appropriate design and operating safeguards to minimize the risks associated with its applications and products.

RFMicron makes no representation or warranty, and assumes no liability, with respect to infringement of patents and/or the rights of third parties, which may result from any

assistance provided by RFMicron or from the use of RFMicron products in Customer's applications or products.

RFMicron represents and Customer acknowledges that this product is neither designed nor intended for use in life support appliances, devices, or systems where malfunction can reasonably be expected to result in personal injury.

This product is covered by U.S. patents 7586385 and 8081043; other patents pending. Chameleon® and Magnus® are trademarks of RFMicron.