Laird

Smart Technology. Delivered.™

How-to Guide

Integrating Laird Sterling-LWB Wi-Fi and Bluetooth with Freescale i.MX 6 UltraLite Evaluation Kit Under Linux

Version 2.4

REVISION HISTORY

Version	Date	Notes	Approver
2.3	01 Dec 2016	Replacing version 2.2	Andy Dobbing
2.4	04 Apr 2017	Rename Doc	Andy Dobbing

CONTENTS

1 Introduction

This document walks you through the software process of integrating the Laird Sterling-LWB's Wi-Fi and Bluetooth functionality with a Freescale i.MX 6 UltraLite Evaluation Kit running Linux. This walkthrough is intended to be a learning tool for software engineers. The walkthrough results are not sufficient for production release but are sufficient for demonstrating the basic Wi-Fi and Bluetooth functionality of the Sterling-LWB.

We first establish a baseline by building and booting the Freescale-provided Linux distribution without modification. We then modify the device tree and kernel configuration, build the Laird-supported drivers, and finally use the Sterling-LWB module to communicate over an open Wi-Fi network.

Note: Connecting to a secured Wi-Fi network, integrating Bluetooth functionality, and integrating power management are beyond the scope of this document.

2 Required Materials

The following are the required materials for this process:

- Laird Sterling-LWB development Kit with SD card form factor
  - LSR part number 450-0155 (U.FL antenna connector) or 450-0156 (chip antenna)
  - Applicable SD card with any of the following part numbers and revision levels:
    - 940-0137-R2
    - 940-0137-R3
    - 940-0138-R2
    - 940-0138-R3
    - 940-0151-R1
    - 940-0151-R2
    - 940-0152-R1
    - 940-0152-R2

    Note: If your SD card does not have of the above part numbers and revision levels, contact Laird support.

- Freescale i.MX 6 UltraLite Evaluation Kit
  - Part number MCIMX6UL-EVK available from the following site: http://www.nxp.com/products/microcontrollers-and-processors/arm-processors/i.mx-applications-processors/i.mx-6-processors/i.mx6qp/i.mx6ultralite-evaluation-kit:MCIMX6UL-EVK?
  - Power supply for Freescale i.MX 6 UltraLite Evaluation Kit
- 2.4 GHz Wi-Fi router
  - DHCP server enabled
  - SSID **ccopen**
  - No authentication, no encryption
- Micro SD card (4 GB or larger)
- USB-A to Micro-USB cable
- Arduino headers for the i.MX6 UltraLite evaluation kit (example: Adafruit product ID 85)
- Male-female jumper wire
- Soldering tools and consumables (0.1" pitch through-hole)
- Voltmeter
- Computer running Ubuntu 16.04.1 to be used as build host with the following:

  - 40 GB disk space minimum
  - *root* privileges
  - Available USB port
  - Micro SD card slot or equivalent adapter
  - Internet connection

  Available from: http://releases.ubuntu.com/16.04/

- Arbitrary Bluetooth device (computer, smartphone, etc.) configured to be visible to other devices
- ESD mat and wrist strap

The following items are optional:

- Arbitrary Arduino Uno R3-compatible shield (for use as soldering jig)
- Breadboard, header pins, and micro grabber leads for voltmeter
- Header pin leads for voltmeter

3 General Information

We suggest that you budget approximately one business day to execute this procedure, assuming a broadband Internet connection and excluding the time required to collect the required materials.

Adhere to all cautions and warnings included with all materials. The LWB and the i.MX 6 UltraLite evaluation kit can be damaged by electrostatic discharge (ESD); handle accordingly.

We assume the software engineer is:

- Familiar with the **vi** and **nano** text editors
- Capable of soldering a 0.1" pitch through-hole component and shortening the pads of an unpopulated surface mount resistor
- Familiar with the use of a voltmeter

3.1 Walkthrough Terminology

The following terminology is used in this how-to guide:

- LWB – Sterling-LWB development kit with SD card form factor
- Target/board – Freescale i.MX 6 UltraLite evaluation kit
- Pin – Refers to a processor's I/O pad
- Freescale and NXP – Interchangeable terms
- Designation *J123-P4* – Refers to pin 4 of connector 123

For this guide, we used a build host running Ubuntu 16.04.1. While it is possible to use a different version of Ubuntu or an entirely different distribution, doing so may require procedural changes. Build hosts running other than Ubuntu 16.04.1 are beyond the scope of this document.

Note: It is possible to run Ubuntu 16.04.1 without installing it to your hard drive. This is done by installing it to and booting from a USB drive instead. Search the Internet for *persistent live Ubuntu USB* for details.

Note: Laird has acquired LS Research (LSR). We are currently in the process of updating all documentation accordingly.

Links and checksums for referenced documents and downloads are provided at the end of this document. Common problems and their causes are also listed at the end of this document.

Pin assignments vary between different revisions of the LWB SD card. This document only applies to the LWB SD cards listed in the *Required Materials* section of this document. If your SD card is not listed, contact Laird Support.

4 Modifying the Breakout Board

Integrating the Sterling-LWB with the i.MX 6 UltraLite evaluation kit requires that electrical connections are established between the target board and LWB as well as within the target board itself.

Connections between the board and LWB are through the SD card connector and additional discrete wiring. We use the board's Arduino connection points for the discrete connections. The board comes with its Arduino connection points labeled and drilled but not populated with headers. We recommend that you solder Arduino-compatible female headers onto the UltraLite breakout board at J1703, J1704, J1705, and J1706.

It’s helpful to use an arbitrary Arduino Uno R3-compatible shield as a jig for aligning the headers during the soldering process. To do this, follow these steps:

1. Mate all headers with the shield pins prior to soldering.
2. Insert all header pins through the board at once.
3. Perform the necessary soldering.
4. Remove the jig shield when finished.

The necessary connections within the target board are formed by soldering a shunt across the pads of unpopulated resistor R1732 on the UltraLite breakout board (on the top of the board between J1703-P5 and the SODIMM connector hosting the compute module).

5 Building a Baseline System Image

We begin this process with our build host running a fresh install of Ubuntu 16.04.1 with no updates installed.

To create our baseline build, we follow the procedure described in sections 3, 4, and 5 of Freescale's *Yocto Project User's Guide* with some minor deviations.

5.1 Installing the Freescale BSP and its Dependencies

To install the Freescale BSP and its dependencies, follow these steps:

1. Install the software packages that are required by the Freescale BSP.
2. Open a terminal window and use the apt-get command to download and install packages from the Ubuntu distribution's package repository servers.

```
user@buildhost:~$ sudo apt-get update
[...]
Fetched 190 kB in 0s (240 kB/s)
```

```
Reading package lists... Done

user@buildhost:~$ sudo apt-get install gawk wget git-core diffstat unzip
texinfo gcc-multilib build-essential chrpath socat libsdl1.2-dev xterm sed cvs
subversion coreutils texi2html docbook-utils python-pysqlite2 help2man make gcc
g++ desktop-file-utils libgl1-mesa-dev libglu1-mesa-dev
mercurial autoconf automake groff curl lzop asciidoc u-boot-tools
27 upgraded, 182 newly installed, 0 to remove and 164 not upgraded.
Need to get 889 MB of archives.
After this operation, 1,529 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

Messages scroll past as packages download.

3. Install the repo utility which allows you to easily fetch multiple git repositories.

```
user@buildhost:~$ mkdir -p ~/bin
user@buildhost:~$ curl http://commondatastorage.googleapis.com/git-repo-downloads/repo
> ~/bin/repo
   %  Total       %  Received   %   Xferd  Average   Speed  Time    Time    Time   Current
                                           Dload    Upload  Total   Spent   Left   Speed
 100  27759     100  27759      0       0  131k          0  --:--   --:--   --:-   130k
                                                            :--     :--     -:--

user@buildhost:~$ chmod a+x ~/bin/repo
user@buildhost:~$ which repo
/home/user/bin/repo
```

The output of the *which repo* command tells you which program your shell invokes when you run the repo command. We expect it to report the file we just downloaded.

4. Configure your git client.

```
user@buildhost:~$ git config --global user.name "John Doe"
user@buildhost:~$ git config --global user.email "john.doe@localhost"
user@buildhost:~$ git config --list
user.name=John Doe
user.email=john.doe@localhost
```

5. Use the repo command to retrieve Freescale's BSP from Freescale's git repository.
6. Invoke the script to setup our build area.

```
user@buildhost:~$ mkdir -p projects/fsl-release-bsp
user@buildhost:~$ cd projects/fsl-release-bsp
user@buildhost:~/projects/fsl-release-bsp$ repo init -u
git://git.freescale.com/imx/fsl-arm-yocto-bsp.git -b imx-4.1.15-1.0.0_ga
user@buildhost:~/projects/fsl-release-bsp$ repo sync
[...]
Fetching projects: 100% (9/9), done.
user@buildhost:~/projects/fsl-release-bsp$ DISTRO=fsl-imx-fb MACHINE=imx6ulevk
source fsl-setup-release.sh -b build-imx6ul-fb
```

```
[...]
Do you accept the EULA you just read? (y/n)
```

7. Enter **y** if you accept Freescale's license agreement.

   Notice that running the script changed the working directory to your build directory: **~/projects/fsl-release-bsp/build-imx6ul-fb**

   The name of your build directory is determined by the **-b** argument to the **fsl-setup-release.sh** script.

5.2 Building the System Image

To build the system image, use the **bitbake** build system to generate a system image for the board. This command downloads and compiles all of the source code associated with the target. In our experience, this process takes approximately one hour.

```
user@buildhost:~/projects/fsl-release-bsp/build-imx6ul-fb$ bitbake core-image-base
[...]
NOTE: Tasks Summary: Attempted 2268 tasks of which 10 didn't need to be rerun and
all succeeded.
```

When bitbake is complete, an SD card image is produced in the following location:

~/projects/fsl-release-bsp/build-imx6ul-fb/tmp/deploy/images/imx6ulevk/core-image-base-imx6ulevk.sdcard

5.3 Writing the System Image to the SD Card

To write the system image to the SD card, follow these steps:

1. Determine what your build host calls your SD card by doing the following:
   a. Monitor your system's log file (syslog).

   ```
   user@buildhost$ sudo tail -F /var/log/syslog
   ```

   b. Insert the SD card into your build host.
   c. In the syslog's output, find the lines that are similar to the following. Note the value of **mmcblk***.

   ```
   kernel: mmc0: new high speed SDHC card at address 1234
   kernel: mmcblk0: mmc0:1234 SA04G 3.64 GiB
   kernel:  mmcblk0: p1 p2
   ```

   ```
   kernel: EXT4-fs (mmcblk0p2): mounted filesystem with ordered data mode.
   Opts: (null)
   udisksd: Mounted /dev/mmcblk0p2 at /media/user/c576e163-f7cf-4b22-a62d-
   eea2c488ba06 on behalf of uid 1000
   udisksd: Mounted /dev/mmcblk0p1 at /media/user/Boot imx6ul on behalf of
   uid 1000
   ```

In this example, the build host named the SD card device **/dev/mmcblk0**.

d. Press **Ctrl+C** to stop tailing the syslog.

Normally when you write a file to a device such as an SD card, you write the file to a filesystem that exists within a partition on that device. However, this system image binary file defines the entire raw contents of an SD card, including its partition tables and filesystem metadata. You must write this raw image data directly to the SD card device rather than into a filesystem that already exists on the SD card device.

Caution: Performing this write destroys all data currently on the SD card.

When you insert your SD card, Ubuntu mounts its partition(s) and opened one or more file system browser windows.

2. Unmount the SD card from the build host to prevent confusing the build host when the SD card's original filesystems unexpectedly disappear.
3. Use the **dd** command to write the contents of the image directly to the raw SD card device. Notice in the sample below, we are using the name for the SD card that we found in the syslog.
4. Before ejecting the SD card, invoke the **sync** command to force cached writes to be committed to the media.

```
user@buildhost$ sudo umount /dev/mmcblk0*

user@buildhost$ sudo dd if=~/projects/fsl-release-bsp/build-imx6ul-
fb/tmp/deploy/images/imx6ulevk/core-image-base-imx6ulevk.sdcard of=/dev/mmcblk0
bs=1M && sync

84+0 records in
84+0 records out
88080384 bytes (88 MB, 84 MiB) copied, 15.2377 s, 5.8 MB/s

user@buildhost$
```

5. When the image is finished writing and syncing, physically eject the SD card from your build host.
6. Install the SD card into the J301 hinged micro-SD card cage on top of the board’s compute module; do not insert your SD card into your board’s breakout board’s full-size SD card slot.

5.4 Connecting a Serial Terminal to the Board

When you boot your board, you'll want to interact with it through a monitor and keyboard. Since the board lacks a monitor and keyboard, cable it to your build host and interact with it through a serial terminal program such as minicom. To complete this process, follow these steps:

1. Install minicom.

```
user@buildhost$ sudo apt-get install minicom
```

Your build host's kernel treats the serial connection as a device file named **/dev/ttyUSB*** (***** represents some integer).

Normal users cannot access these device files by default so you must add your user to a privileged group.

2. Add your user to the build host's **dialout** group located in the **/etc/group** file.

```
user@buildhost$ sudo nano /etc/group
[...]
dialout:x:20:user
[...]
```

3. Press **Ctrl+o** to save, then **Ctrl+x** to exit the nano editor.

   The change to the /etc/group file is not effective immediately. To force the change to take effect, follow these steps:

   a. Close your Terminal window.
   b. Log out of your desktop session.
   c. Log back into your desktop session.
   d. Open a new Terminal window.

4. Verify that the *dialout* group is included in your group list.

```
user@buildhost$ id
uid=1000(user) gid=1000(user)
groups=1000(user),4(adm),20(dialout),24(cdrom),27(sudo),30(dip),
46(plugdev),113(lpadmin),128(sambashare)
```

5. Before you can establish a serial terminal, you must determine which specific device file represents your serial connection to your board. Monitor the syslog again.

```
user@buildhost$ sudo tail -F /var/log/syslog
```

6. Connect a USB cable between your build host and the board's J1901 micro-USB jack.

7. When messages appear in the syslog, disconnect the USB cable.

8. Inspect the log output for a message similar to the following.

```
Oct  7 16:44:40 buildhost kernel: [114634.216893] cp210x ttyUSB4:
cp210x converter now disconnected from ttyUSB4
```

9. Make note of the value of ttyUSB*.

10. Press **Ctrl+c** to stop tailing the syslog.

11. Reconnect the USB cable between the build host and the board's J1901 jack.

12. Open a serial terminal session with the board by starting the minicom program and supplying the value of ttyUSB* observed in the log.

```
user@buildhost$ minicom -D /dev/ttyUSB4
```

13. Press **Ctrl+a** followed by **z** to open minicom's menu.

14. Type **o** to configure Minicom.

```
+-------------------------------------------------------------------+
|                      Minicom Command Summary                      |
|                                                                   |
|              Commands can be called by CTRL-A <key>               |
|                                                                   |
|               Main Functions                  Other Functions    |
|                                                                   |
| Dialing directory..D  run script (Go)....G | Clear Screen.......C |
| Send files.........S  Receive files......R | cOnfigure Minicom..O |
| comm Parameters....P  Add linefeed.......A | Suspend minicom....J |
| Capture on/off.....L  Hangup.............H | eXit and reset.....X |
| send break.........F  initialize Modem...M | Quit with no reset.Q |
| Terminal settings..T  run Kermit.........K | Cursor key mode....I |
| lineWrap on/off....W  local Echo on/off..E | Help screen........Z |
| Paste file.........Y  Timestamp toggle...N | scroll Back........B |
| Add Carriage Ret...U                                              |
|                                                                   |
|             Select function or press Enter for none.              |
+-------------------------------------------------------------------+
```

15. Select **Serial port setup** from the menu.

```
+-----[configuration]------+
| Filenames and paths      |
| File transfer protocols  |
| Serial port setup        |
| Modem and dialing        |
| Screen and keyboard      |
| Save setup as dfl        |
| Save setup as..          |
| Exit                     |
+--------------------------+
```

16. Ensure that your settings appear as follows with your specific value set for Serial Device:

```
+-----------------------------------------------------------------------+
| A -    Serial Device      : /dev/ttyUSB*                              |
| B - Lockfile Location     : /var/lock                                 |
| C -   Callin Program      :                                           |
| D -  Callout Program      :                                           |
| E -    Bps/Par/Bits       : 115200 8N1                                |
| F - Hardware Flow Control : No                                        |
| G - Software Flow Control : No                                        |
|                                                                       |
|    Change which setting?                                              |
+-----------------------------------------------------------------------+
```

17. Exit the Serial port setup menu to return to the [configuration] menu.

18. Select **Save setup as dfl** to save your changes to minicom's default settings.

```
+------[configuration]-------+
| Filenames and paths        |
| File transfer protocols    |
| Serial port setup          |
| Modem and dialing          |
| Screen and keyboard        |
| Save setup as dfl          |
| Save setup as..            |
| Exit                       |
+----------------------------+
```

19. Exit the menus.

5.5 Selecting the Boot Device

To select the boot device, follow these steps:

1. Configure the board to boot from the micro SD card.
2. On the compute module, set the switches as follows:

SW601	D1 off, D2 off, D3 on, D4 off
SW602	D1 on, D2 off

5.6 Booting the System Image

To boot the system image, follow these steps:

1. With your serial terminal established, power up your board.

   You should see something similar to the following:

```
U-Boot 2015.04imx_v2015.04_4.1.15_1.2.0_ga+gede7538 (Oct 07 2016 - 14:51:35)

CPU:   Freescale i.MX6UL rev1.0 at 396 MHz
CPU:   Temperature 30 C
Reset cause: POR
Board: MX6UL 14x14 EVK
I2C:   ready
DRAM:  512 MiB
MMC:   FSL_SDHC: 0, FSL_SDHC: 1
*** Warning - bad CRC, using default environment
```

   You should then see scrolling messages, followed by the login prompt:

```
Freescale i.MX Release Distro 4.1.15-1.2.0 imx6ulevk /dev/ttymxc0

imx6ulevk login:
```

2. Enter the user name **root**. The board's shell prompt appears:

```
root@imx6ulevk:~#
```

Congratulations! You've built and booted a system image.

Capture an observation that you will reference in the next section:

```
root@imx6ulevk:~# cat /sys/kernel/debug/pinctrl/pinctrl-maps | grep -C 5 UART5

device 21a4000.i2c
state default
type CONFIGS_PIN (3)
controlling device 20e0000.iomuxc
pin MX6UL_PAD_UART5_TX_DATA
config 0001b8b0

device 21a4000.i2c
state default
type CONFIGS_PIN (3)
controlling device 20e000.iomuxc
pin MX6UL_PAD_UART5_RX_DATA
config 0001b8b0
device 2080000.pwm
state default
type MUX_GROUP (2)
```

6 Modifying the System Image

To support the LWB, you must create a new system image for the board by doing the following:

- Allocating a pin on the i.MX 6 UltraLite processor via the device tree for:
  - Toggling the LWB's Wi-Fi subsystem between its *enabled* and *reset* states.
  - Toggling the LWB's Bluetooth subsystem between its *enabled* and *reset* states.
- Removing potentially conflicting drivers from the kernel.
- Adding drivers and firmware.

6.1 Reestablishing the Build Environment

The build environment variables were cleared when you previously closed the Terminal window earlier. You must re-source the build environment. To do this, follow these steps:

1. Exit from the Minicom serial terminal.
2. Press **Ctrl+a** followed by **x**.
3. When minicom asks you if you want to leave, select **Yes**.
4. Change to the BSP directory and source your build environment, as shown by the commands below.

```
user@buildhost$ cd ~/projects/fsl-release-bsp/
user@buildhost:~/projects/fsl-release-bsp$ source setup-environment build-
imx6ul-fb/

Welcome to Freescale Community BSP

The Yocto Project has extensive documentation about OE including a
reference manual which can be found at:
    http://yoctoproject.org/documentation

For more information about OpenEmbedded see their website:
    http://www.openembedded.org/

You can now run 'bitbake <target>'

Common targets are:
    core-image-minimal
    meta-toolchain
    meta-toolchain-sdk
    adt-installer
    meta-ide-support

Your configuration files at build-imx6ul-fb/ have not been touched.
user@buildhost:~/projects/fsl-release-bsp/build-imx6ul-fb$
```

Notice that the script has changed your working directory to the build's directory.

6.2 Configuring the Device Tree

You must configure a pin on the iMX 6 UltraLite processor to drive the WL_REG_ON and BT_REG_ON lines that control the LWB's internal power regulators. These regulators provide the ability to reduce the LWB's power consumption and to reinitialize the LWB device.

Note: These regulators only control power distribution *within* the LWB. The target board has separate regulators that control whether power is delivered *to* the LWB.

WL_REG_ON controls power to the LWB's Wi-Fi subsystem, while BT_REG_ON controls power to the LWB's Bluetooth subsystem. When one is driven high, the corresponding subsystem is enabled; when one is driven low, the corresponding subsystem is held in reset. The Broadcom 4343W data sheet discusses WL_REG_ON and BT_REG_ON in detail.

The binding of pins to various functions is configured through the device tree. The kernel consumes a binary form of the device tree. This binary form is referred to as any of the following: Device Tree Binary, Device Tree Blob, Flat Device Tree. The Device Tree Compile (DTC) is used to translate a human-readable device tree source file into binary form.

6.2.1 Choosing Pins

You must first decide which processor pins (pads) to use for WL_REG_ON and BT_REG_ON. When deciding, consider the following for each signal:

- The pin must be capable of producing digital output.
- It must be controllable by software.
- The line must be driven exclusively by the processor.
- Driving the pin must not interfere with the board's ability to perform other necessary functions.
- There must be a reasonable means of connecting the signal to the LWB.
- The pin should be selected to minimize rework of the board.

- The pin should be selected to minimize software configuration effort.

For this instruction guide, we selected UART5_TX_DATA for the WL_REG_ON signal and processor pin UART5_RX_DATA for the BT_REG_ON signal after reviewing the following Freescale documents:

- Compute module schematic (*SPF-28617_C1.pdf*)
- Breakout board schematic (*SPF-28616_C.pdf*)
- i.MX6 UltraLite Reference Manual (*IMX6ULRM.pdf*)

Each signal has different names within different contexts:

LWB	WL_REG_ON	BT_REG_ON
LWB Header	J3-P11	J3-P13
Arduino Pinout	I2C_SCL (D19)	I2C_SDA (D18)
i.MX6UL Breakout Board Header	J1704-P10	J1704-P9
i.MX6UL Breakout Board Node	ISC2_SCL	I2C2_SDA
Breakout Board/ Compute Module Interconnect Pin	79	68
i.MX6UL Compute Module Node	UART5_TXD	UART5_RXD
i.MX6UL Processor Ball	F17	G13
i.MX6UL Processor Pad	UART5_TX_DATA	UART5_RX_DATA
i.MX6UL GPIO Bank / Pin	Bank 1 Pin 30	Bank 1 Pin 31

The following are the reasons we selected the UART5_TX_DATA and UART5_RX_DATA:

- The pins can be configured for general purpose input/output (GPIO), which can be a software controlled digital output.
- Using these pins does not interfere with the board's normal operation because:
  - They are assigned by default to communicate over an I2C bus
  - We do not need to access any of the I2C devices that are attached to the I2C bus
  - The I2C slave devices on the bus do not respond to the DC signals we drive
- No other devices drive these lines because:
  - There are no other I2C master devices on the bus
  - The I2C slave devices on the bus do not respond to the DC signals we drive
- The signals are accessible through the board's Arduino headers which are reasonably accessible for prototyping work.
- Aside from populating the Arduino headers and soldering a shunt across the pads of an unpopulated resistor, no further board rework is required; particularly, no surface mount devices need to be populated, no devices need to be removed, no traces need to be cut, and no jumper wires need to be soldered onto devices.
- The pins are allocated by default to a bus with unutilized devices, so software reconfiguration is minimal.

6.2.2 Removing Pins' Default Bindings

To remove the pin's default binding, follow these steps:

1. Open your device tree source file in a text editor.

```
user@buildhost:~/projects/fsl-release-bsp$ nano ~/projects/fsl-release-
bsp/build-imx6ul-fb/tmp/work-shared/imx6ulevk/kernel-
source/arch/arm/boot/dts/imx6ul-14x14-evk.dts
```

2. Delete the lines that have been struck out in the following list to prevent the pins from being allocated to the second I2C controller.

[...]

&iomuxc {
[...]
    imx6ul-evk {
[...]
        pinctrl_i2c2: i2c2grp {
            fsl,pins = <
                ~~MX6UL_PAD_UART5_TX_DATA__I2C2_SCL 0x4001b8b0~~
                ~~MX6UL_PAD_UART5_RX_DATA__I2C2_SDA 0x4001b8b0~~
            >;
        };
[...]
};
[...]

3. Save your changes to the file by pressing **Ctrl+o**.

6.2.3 Binding Pins as GPIO

The processor's I/O Multiplexer Controller (iomuxc) hardware governs the peripheral functions and signal routings associated with the processor's pins. It is managed by the kernel's Pin Control (pinctrl) subsystem, which consumes the binary device tree as input. To configure the pin for GPIO, you must configure up to three memory-mapped registers per pin, including the following:

- MUX control register
- Pad control register (also known as pad configuration register)
- DAISY Register (also known as input register), if applicable

Each of these register instances is discussed in detail in its own section of the *i.MX 6 UltraLite Reference Manual*.

The device tree file includes a processor-specific device tree include file, *imx6ul.dtsi*, which in turn includes the file *imx6ul-pinfunc.h*. The *imx6ul-pinfunc.h* file defines macros that, if incorporated into a device tree, configure a pin's MUX control register, pad control register, and DAISY (input) control register for a specific role.

```
[...]
/*
 * The pin function ID is a tuple of
 * <mux_reg conf_reg input_reg mux_mode input_val>
 */
[...]
#define MX6UL_PAD_UART5_TX_DATA__GPIO1_IO30              0x00BC 0x0348 0x0000 0x5 0x0
#define MX6UL_PAD_UART5_TX_DATA__ECSPI2_MOSI             0x00BC 0x0348 0x054C 0x8 0x0
#define MX6UL_PAD_UART5_TX_DATA__UART5_DCE_TX            0x00BC 0x0348 0x0000 0x0 0x0
#define MX6UL_PAD_UART5_TX_DATA__UART5_DTE_RX            0x00BC 0x0348 0x0644 0x0 0x4
#define MX6UL_PAD_UART5_TX_DATA__ENET2_CRS               0x00BC 0x0348 0x0000 0x1 0x0
#define MX6UL_PAD_UART5_TX_DATA__I2C2_SCL                0x00BC 0x0348 0x05AC 0x2 0x2
#define MX6UL_PAD_UART5_TX_DATA__CSI_DATA14              0x00BC 0x0348 0x04FC 0x3 0x0
#define MX6UL_PAD_UART5_TX_DATA__CSU_CSU_ALARM_AUT00     0x00BC 0x0348 0x0000 0x4 0x0
#define MX6UL_PAD_UART5_RX_DATA__UART5_DCE_RX            0x00C0 0x034C 0x0644 0x0 0x5
#define MX6UL_PAD_UART5_RX_DATA__UART5_DTE_TX            0x00C0 0x034C 0x0000 0x0 0x0
#define MX6UL_PAD_UART5_RX_DATA__ENET2_COL               0x00C0 0x034C 0x0000 0x1 0x0
#define MX6UL_PAD_UART5_RX_DATA__I2C2_SDA                0x00C0 0x034C 0x05B0 0x2 0x2
#define MX6UL_PAD_UART5_RX_DATA__CSI_DATA15              0x00C0 0x034C 0x0500 0x3 0x0
#define MX6UL_PAD_UART5_RX_DATA__CSU_CSU_INT_DEB         0x00C0 0x034C 0x0000 0x4 0x0
#define MX6UL_PAD_UART5_RX_DATA__GPIO1_IO31              0x00C0 0x034C 0x0000 0x5 0x0
#define MX6UL_PAD_UART5_RX_DATA__ECSPI2_MISO             0x00C0 0x034C 0x0548 0x8 0x1

[...]
```

A macro is provided for every valid (pin, function) pair. The first part of the macro name identifies a pin and the second part selects the function. The numeric fields define the address offsets of the corresponding mux control register, pad configuration register, and DAISY (input) register, followed by the values to be loaded into the mux control register DAISY registers, respectively. These address offsets and register values are defined in the i.MX 6 UltraLite Reference Manual.

Incorporate the following bold lines into our device tree file to configure our pins for GPIO:

```
[...]
#include <dt-bindings/input/input.h>
#include "imx6ul.dtsi"
[...]
&iomuxc {
	pinctrl-names = "default";
	pinctrl-0 = <&pinctrl_hog_1>;
	imx6ul-evk {
		pinctrl_hog_1: hoggrp-1 {
			fsl,pins = <
				MX6UL_PAD_UART1_RTS_B__GPIO1_IO19	0x17059
				MX6UL_PAD_GPIO1_IO05__USDHC1_VSELECT	0x17059
				MX6UL_PAD_GPIO1_IO09__GPIO1_IO09	0x17059
```

```
                        MX6UL_PAD_SNVS_TAMPER0__GPIO5_IO00      0x80000000
                        MX6UL_PAD_UART5_TX_DATA__GPIO1_IO30     0x3031
                        MX6UL_PAD_UART5_RX_DATA__GPIO1_IO31
```

Our macros configure the following:

- Pin UART5_TX_DATA for GPIO and associate it with GPIO bank 1 pin 30
- Pin UART5_RX_DATA for GPIO and associate it with GPIO bank 1 pin 31

The numeric arguments to the macros specify the values of the pad control. The values are derived using the **SW_PAD_CTL_PAD_UART5_TX_DATA SW PAD** and **SW_PAD_CTL_PAD_UART5_RX_DATA SW PAD** bit definitions from the *i.MX 6 UltraLite Reference Manual* to configure the following properties for your pin:

- Hysteresis disabled (don't care)
- 100K ohm internal pull-down resistor
- Open drain disabled
- Slow speed (don't care)
- Fast slew rate (don't care)

6.2.4 Testing GPIO Pins Under Manual Control

Making these substantial changes warrants some testing. To perform this testing, follow these steps:

1. Create an updated system image. Behind the scenes, the first bitbake recipe below invokes **DTC** to compile your modified device tree source into a binary.

```
user@buildhost:~/projects/fsl-release-bsp/build-imx6ul-fb$ bitbake linux-imx -f -c
deploy

[...]

NOTE: Tainting hash to force rebuild of task /home/user/projects/fsl-release-
bsp/sources/meta-fsl-bsp-release/imx/meta-bsp/recipes-kernel/linux/linux-imx_4.1.15.bb,
do_deploy

[...]

NOTE: Tasks Summary: Attempted 268 tasks of which 253 didn't need to be rerun and all
succeeded.

[...]

user@buildhost:~/projects/fsl-release-bsp/build-imx6ul-fb$ bitbake core-image-base

[...]

WARNING: /home/user/projects/fsl-release-bsp/sources/meta-fsl-bsp-release/imx/meta-
bsp/recipes-kernel/linux/linux-imx_4.1.15.bb.do_deploy is tainted from a forced run

[...]

NOTE: Tasks Summary: Attempted 2268 tasks of which 2253 didn't need to be rerun and all
succeeded.

[...]
```

2. Write the image to the SD card as you did previously.

3. Open a serial terminal to the board, insert the SD card, boot the board, and log in as you did previously.

   You can use the debug filesystem to verify that the device tree changes had the desired effect.

4. Re-cat the pinctrl-maps file.

```
root@imx6ulevk:~# cat /sys/kernel/debug/pinctrl/pinctrl-maps | grep -C 5 UART5_

device 20e0000.iomuxc
state default
type CONFIGS_PIN (3)
controlling device 20e0000.iomuxc
pin MX6UL_PAD_UART5_TX_DATA
config 00003031

device 20e0000.iomuxc
state default
type CONFIGS_PIN (3)
controlling device 20e0000.iomuxc
pin MX6UL_PAD_UART3UART5_RX_DATA
config 00003031

device regulators:regulator-gpio
state default
type MUX_GROUP (2)
```

   Notice that the pins were previously associated with an **i2c** device but are now associated with the **iomuxc** device. Also notice that the pins' config values are now 0x3031, the values that were specified along with the macros in the device tree file.

   You can use the **/sys/class/gpio** facilities to verify that you can manually toggle the pin.

5. Change to the **/sys/class/gpio** directory.
   Recall that your pins are on GPIO bank 1 as pin 30 and GPIO bank 1 pin 31. This implies that their GPIO pin numbers are 30 and 31 overall.
6. Echo these overall pin numbers into the export file. A new directory (gpio25) now exists to represent your pins.

```
root@imx6ulevk:~# cd /sys/class/gpio/
root@imx6ulevk:/sys/class/gpio# ls -1
export
gpio132
gpiochip0
gpiochip128
gpiochip32
gpiochip504
gpiochip64
gpiochip96
unexport
root@imx6ulevk:/sys/class/gpio# echo 30 > export
root@imx6ulevk:/sys/class/gpio# echo 31 > export
root@imx6ulevk:/sys/class/gpio# ls -1
```

```
export
gpio30
gpio31
gpiochip0
gpiochip128
gpiochip32
gpiochip504
gpiochip64
gpiochip96
unexport
```

7. Switch to the new **gpio30** directory and configure your pin as an output pin by echoing **out** into its *direction* file.

```
root@imx6ulevk:/sys/class/gpio# cd gpio30
root@imx6ulevk:/sys/class/gpio/gpio25# ls -1
active_low
device
direction
edge
power
subsystem
uevent
value
root@imx6ulevk:/sys/class/gpio/gpio30# echo out > direction
```

8. Using a voltmeter, measure the voltage of J1704-P10 with respect to ground. Ground can be found at J1705-P7. Observe that the voltage is ~0.0 V.

9. Change your pin's value to a logical **1** by echoing a **1** into the pin's *value* file.

```
root@imx6ulevk:/sys/class/gpio/gpio30# echo 1 > value
```

   Observe that the voltmeter now reads ~3.3 V.

10. Change your pin's value back to a logical **0**.

```
root@imx6ulevk:/sys/class/gpio/gpio30# echo 0 > value
```

   The voltage is again ~0.0 v.

Now repeat the process for the other pin.

```
root@imx6ulevk:/sys/class/gpio/gpio30# cd ../gpio31/
root@imx6ulevk:/sys/class/gpio/gpio31# echo out > direction
```

1. Using a voltmeter, measure the voltage of J1704-P9 with respect to ground. Ground can be found at J1705-P7. Observe that voltage is ~0.0 V.
2. Change the pin's value to a logical **1** by echoing a **1** into the pin's *value* file.

```
root@imx6ulevk:/sys/class/gpio/gpio31# echo 1 > value
```

Observe that the voltmeter now reads ~3.3 V.

3. Change the pin's value back to a logical **0**.

```
root@imx6ulevk:/sys/class/gpio/gpio31# echo 0 > value
```

You have successfully toggled your pins' outputs which confirms that the electrical path is intact and that you have correctly modified your device tree.

The following screen capture is referenced in the next section.

```
root@imx6ulevk# cat /sys/kernel/debug/regulator/regulator_summary
 regulator                use    open   bypass  voltage  current      min      max
-------------------------------------------------------------------------------------
 gpio_dvfs                  0       1        0   1300mV      0mA   1300mV   1400mV
   cpu0                                                            1300mV   1300mV
 VSD_3V3                    0       1        0   3300mV      0mA   3300mV   3300mV
   2190000.usdhc                                                   3300mV   3400mV
 can-3V3                    0       2        0   3300mV      0mA   3300mV   3300mV
   2094000.can                                                        0mV      0mV
   2090000.can                                                        0mV      0mV
 vddsoc                     0       1        0   1175mV      0mA    725mV   1450mV
   cpu0                                                            1175mV   1175mV
 cpu                        0       1        0    950mV      0mA    725mV   1450mV
   cpu0                                                             950mV    950mV
 vdd3p0                     2       2        0   3200mV      0mA   2625mW   3400mV
   20ca000.usbphy                                                  3200mV   3200mV
   20c9000.usbphy                                                  3200mV   3200mV
 regulator-dummy            0       9        0      0mV      0mV      0mV      0mV
   0-000e                                                             0mV      0mV
   0-000e                                                             0mV      0mV
 2184200.usb                                                          0mV      0mV
 2184000.usb                                                          0mV      0mV
 2184800.usbmisc                                                      0mV      0mV
 2188000.ethernet                                                     0mV      0mV
 20b4000.ethernet                                                     0mV      0mV
 21c8000.lcdif                                                        0mV      0mV
 backlight                                                            0mV      0mV
```

4. Exit the serial terminal as before.

6.2.5 Binding GPIO Pins as Regulators

Toggling your pins' values manually is useful for testing in a development environment, but having a production system controlling their values autonomously is preferable. We use the pair of GPIO pins to control a pair of power regulators within the LWB. Conveniently, Linux includes a facility specifically for managing such regulators.

A regulator can be enabled or disabled according to system power states and device drivers can manipulate regulators to disable power to idle hardware. The details of power management are beyond the scope of this document so we'll configure our LWB's regulators to constantly stay on.

Again, note that the regulators we deal with in this document only manage the distribution of power within the LWB; separate regulators belonging to the target board control whether power is delivered to the LWB.

```
user@buildhost$ nano ~/projects/fsl-release-bsp/build-imx6ul-fb/tmp/work-
shared/imx6ulevk/kernel-source/arch/arm/boot/dts/imx6ul-14x14-evk.dts
```

1. Define a pair of regulators and associate them with your GPIO pins by adding the following bold lines to your device tree:

```
[...]
/ {
        regulators {
        [...]
                reg_gpio_dvfs: regulator-gpio {
                        compatible = "regulator-gpio";
                        pinctrl-names = "default";
                        pinctrl-0 = <&pinctrl_dvfs>;
                        regulator-min-microvolt = <1300000>;
                        regulator-max-microvolt = <1400000>;
                        regulator-name = "gpio_dvfs";
                        regulator-type = "voltage";
                        gpios = <&gpio5 3 GPIO_ACTIVE_HIGH>;
                        states = <1300000 0x1 1400000 0x0>;
                };

                reg_wl: regulator-wl {
                        compatible = "regulator-fixed";
                        regulator-name = "wl";
                        gpio = <&gpio1 30 GPIO_ACTIVE_HIGH>;
                        startup-delay-us = <100>;
                        enable-active-high;
                        regulator-min-microvolt = <3300000>;
                        regulator-max-microvolt = <3300000>;
                        regulator-always-on;

                };

                reg_bt: regulator-bt {
                        compatible = "regulator-fixed";
                        regulator-name = "bt";
                        gpio = <&gpio1 31 GPIO_ACTIVE_HIGH>;
                        startup-delay-us = <100>;
                        enable-active-high;
                        regulator-min-microvolt = <3300000>;
                        regulator-max-microvolt = <3300000>;
                        regulator-always-on;
                };
        };
[...]
```

2. Save your changes to the file (Ctrl+o) and exit (Ctrl+q).

6.2.6 Testing GPIO Pins Under Regulator Control

This is a good point to stop and test again, so update your system image:

```
user@buildhost:~/projects/fsl-release-bsp/build-imx6ul-fb$ bitbake linux-imx -f -c deploy

[...]
NOTE: Tasks Summary: Attempted 268 tasks of which 257 didn't need to be rerun and all
succeeded.
[...]
user@buildhost:~/projects/fsl-release-bsp/build-imx6ul-fb$ bitbake core-image-base
[...]
NOTE: Tasks Summary: Attempted 2268 tasks of which 2253 didn't need to be rerun and all
succeeded.
[...]
```

1. Write the image to the SD card: Open a serial terminal to the board, insert the SD card, boot the board, and log in.
2. Using the voltmeter, observe that the system has automatically driven your pins to ~3.3 V during startup. Also notice that the LWB's regulators now appears in the system's regulator summary.

```
root@imx6ulevk# cat /sys/kernel/debug/regulator/regulator_summary
 regulator             use   open  bypass  voltage  current      min      max
-------------------------------------------------------------------------------
 gpio_dvfs               0      1       0  1300mV       0mA  1300mV   1400mV
   cpu0                                                      1300mV   1300mV
 bt                      0      0       0  3300mV       0mA    3300     3300
                                                                 mV       mV
 Wl                      0      0       0  3300mV       0mA  3300mV   3300mV
 VSD_3V3                 0      1       0  3300mV       0mA  3300mV   3300mV
   2190000.usdhc                                             3300mV   3400mV
 can-3V3                 0      2       0  3300mV       0mA  3300mV   3300mV
   2094000.can                                                  0mV      0mV
   2090000.can                                                  0mV      0mV
 vddsoc                  0      1       0  1175mV       0mA   725mV   1450mV
   cpu0                                                      1175mV   1175mV
 cpu                     0      1       0   950mV       0mA   725mV   1450mV
   cpu0                                                       950mV    950mV
 vdd3p0                  2      2       0  3200mV       0mA  2625mW   3400mV
   20ca000.usbphy                                            3200mV   3200mV
   20c9000.usbphy                                            3200mV   3200mV
 regulator-dummy         0      9       0     0mV       0mV     0mV      0mV
   0-000e                                                       0mV      0mV
   0-000e                                                       0mV      0mV
 2184200.usb                                                    0mV      0mV
 2184000.usb                                                    0mV      0mV
 2184800.usbmisc                                                0mV      0mV
```

```
2188000.ethernet                                              0mV      0mV
20b4000.ethernet                                              0mV      0mV
21c8000.lcdif                                                 0mV      0mV
backlight                                                     0mV      0mV
```

Now that you've given the regulator subsystem control of the pins, we can no longer manipulate it manually through /sys/class/gpio. Note that the pinctrl-maps still lists them as allocated to the **iomuxc** device.

```
root@imx6ulevk:~# cat /sys/kernel/debug/pinctrl/pinctrl-maps | grep -C 5 UART5_
device 20e0000.iomuxc
state default
type CONFIGS_PIN (3)
controlling device 20e0000.iomuxc
pin MX6UL_PAD_UART5_TX_DATA
config 00003031

device 20e0000.iomuxc
state default
type CONFIGS_PIN (3)
controlling device 20e0000.iomuxc
pin MX6UL_PAD_UART5_RX_DATA
config 00003031

device regulators:regulator-gpio
state default
type MUX_GROUP (2)
```

6.3 Configuring the Kernel

This section shows you how to build our own drivers for use with the LWB. Before you build your own driver, you must determine whether any conflicting drivers are built in to your kernel. To do this, follow these steps:

1. Within a serial terminal to your board, list the modules built into the kernel and search for any that would cause a conflict.
2. On the build host, open a minicom serial terminal to the board following the same procedure as before. Remember to verify the settings in minicom's configuration menu.
3. Run the following commands on the board to determine whether any relevant modules are built into our kernel.

```
root@imx6ulevk:~# cat /lib/modules/$(uname -r)/modules.builtin | grep
'compat.ko\|cfg80211.ko\|brcmutil.ko\|brcmfmac.ko'

kernel/net/wireless/cfg80211.ko
root@imx6ulevk:~#
```

   The cfg80211 module is built into the kernel, so you'll need to rebuild the kernel and exclude it in the process.

4. Exit the serial terminal to return to running commands on the build host. Press **Ctrl+a** > **x**. When minicom asks if you want to leave, select **Yes**.

5. Start the kernel's configuration menu utility. It appears in its own window but takes some time to appear the first time you use it.

```
user@buildhost:~/projects/fsl-release-bsp/build-imx6ul-fb$ bitbake -c menuconfig linux-
imx
```

6. Exclude the cfg80211 module from the kernel by selecting **Network Support** > **Wireless** from the menu. Then highlight the *cfg80211 – wireless configuration API* item and press **n**.

```
.config - Linux/arm 4.1.15 Kernel Configuration
 → Networking support → Wireless ───────────────────────────────────────────────────
────────────────────────────────
   ┌─────────────────────────────────────────────────────────── Wireless ─────────────
──────────────────────────────────────────┐
   │  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty │
   │  submenus ----).  Highlighted letters are hotkeys.  Pressing <Y>        │
   │  includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to │
   │  exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ]         │
   │ ┌──────────────────────────────────────────────────────────────────────────────
────────────────────────────────────────────┐  │
   │ │     --- Wireless                                                     │  │
   │ │     < >   cfg80211 - wireless configuration API                      │  │
   │ │           *** CFG80211 needs to be enabled for MAC80211 ***          │  │
   │ │                                                                      │  │
   │ │                                                                      │  │
   │ │                                                                      │  │
   │ │                                                                      │  │
   │ │                                                                      │  │
   │ │                                                                      │  │
   │ │                                                                      │  │
   │ └──────────────────────────────────────────────────────────────────────────────
────────────────────────────────────────────┘  │
   ├────────────────────────────────────────────────────────────────────────────────
──────────────────────────────────────────────────┤
   │       <Select>    < Exit >    < Help >    < Save >    < Load >           │
   └────────────────────────────────────────────────────────────────────────────────
──────────────────────────────────────────────────┘
```

7. **Escape** back to the main menu.

8. Exclude wireless LAN drivers from the kernel by navigating to **Device Drivers** > **Network device support** from the menu. Then highlight the wireless LAN item and press **n**.

```
.config - Linux/arm 4.1.15 Kernel Configuration
 → Device Drivers → Network device support ─────────────────────────────────────────
────────────────────────
   ┌────────────────────────────────────────────── Network device support ───────────
─────────────────────────────────┐
   │  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty │
   │  submenus ----).  Highlighted letters are hotkeys.  Pressing <Y>        │
   │  includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to │
```

```
  │  exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ]       │
  │ ┌──────↑(-)────────────────────────────────────────────────────────────
──────────────────────────────────────┐  │
  │ │     [*]   Ethernet driver support  --->                              │  │
  │ │     -*-   PHY Device support and infrastructure  --->                │  │
  │ │     < >   Micrel KS8995MA 5-ports 10/100 managed Ethernet switch     │  │
  │ │     < >   PPP (point-to-point protocol) support                      │  │
  │ │     < >   SLIP (serial line) support                                 │  │
  │ │     <*>   USB Network Adapters  --->                                 │  │
  │ │     [ ]   Wireless LAN  ----                                         │  │
  │ │           *** Enable WiMAX (Networking options) to see the WiMAX driv│  │
  │ │     [ ]   Wan interfaces support  ----                               │  │
  │ │     [ ]   ISDN support  ----                                         │  │
  │ └──────────────────────────────────────────────────────────────────────────
───────────────────────────────────────┘  │
  ├────────────────────────────────────────────────────────────────────────
────────────────────────────────────────────┤
  │       <Select>    < Exit >    < Help >    < Save >    < Load >        │
  └────────────────────────────────────────────────────────────────────────
────────────────────────────────────────────┘
```

9. Escape back up to the main menu and then to the configuration menu. When prompted, choose to save the new configuration.

10. Copy the new configuration file to the location that the build system will look for it:

```
user@buildhost:~/projects/fsl-release-bsp/build-imx6ul-fb$ cp -f
./tmp/work/imx6ulevk-poky-linux-gnueabi/linux-imx/4.1.15-r0/build/.config
./tmp/work-shared/imx6ulevk/kernel-source/arch/arm/configs/imx_v7_defconfig
```

11. Make the build system ingest its new configuration file and flag the build targets that depend upon it as dirty (tainted).

```
user@buildhost:~/projects/fsl-release-bsp/build-imx6ul-fb$ bitbake -f -c
copy_defconfig linux-imx
[...]
NOTE: Tasks Summary: Attempted 3 tasks of which 2 didn't need to be rerun and
all succeeded.
[...]
```

12. Update the system image. Because the build system has detected that your kernel configuration is tainted, it rebuild your kernel in the process.

```
user@buildhost:~/projects/fsl-release-bsp/build-imx6ul-fb$ bitbake core-image-
base
[...]
WARNING: /home/user/projects/fsl-release-bsp/sources/meta-fsl-bsp-
release/imx/meta-bsp/recipes-kernel/linux/linux-imx_4.1.15.bb.do_copy_defconfig
is tainted from a forced run
[...]
NOTE: Tasks Summary: Attempted 2268 tasks of which 2238 didn't need to be rerun
and all succeeded.
```

```
[...]
```

13. Rewrite the image to the SD card.

```
user@buildhost$ sudo umount /dev/mmcblk0*
user@buildhost$ sudo dd if=~/projects/fsl-release-bsp/build-imx6ul-
fb/tmp/deploy/images/imx6ulevk/core-image-base-imx6ulevk.sdcard of=/dev/mmcblk0
bs=1M && sync

84+0 records in
84+0 records out
88080384 bytes (88 MB, 84 MiB) copied, 15.2377 s, 5.8 MB/s
```

6.4 Writing the System Image to the SD Card

To write the system image to the SD card, follow these steps:

1. Write your updated system image to the SD card.

```
user@buildhost$ sudo umount /dev/mmcblk0*
user@buildhost$ sudo dd if=~/projects/fsl-release-bsp/build-imx6ul-
fb/tmp/deploy/images/imx6ulevk/core-image-base-imx6ulevk.sdcard of=/dev/mmcblk0
bs=1M && sync

84+0 records in
84+0 records out
88080384 bytes (88 MB, 84 MiB) copied, 15.2377 s, 5.8 MB/s

user@buildhost$
```

Do not eject the SD card because you'll be making some additional modifications to it.

6.5 Building the Drivers

To build the driver modules, follow these steps:

1. Download the LWB driver source code and firmware.
2. Using your web browser, visit http://www.lairdtech.com/product-categories/embedded-wireless/wi-fi-and-wi-fi-bt-modules
1. Select the Sterling-LWB product, go to the Kits & Software tab, and download the Linux backports package *Sterling-LWB Software Version Backports (930-0075)* to **/home/user/930-0075.zip**.

   The binary firmware packages are available for two regulatory regions.
2. Select the firmware package which matches the region in which you intend to operate your end device.
   - For the USA or Canada, use the *Sterling-LWB Firmware Package (480-0079)* to **/home/user/480-0079.zip**.
   - For the ETSI / Rest of the World use the *Sterling-LWB Firmware Package (480-0080)* to **/home/user/480-0080.zip.**

Table 1: Regional firmware packages

Regulatory Domain	Firmware Package Name	md5 sum
USA or Canada	480-0079.zip	53bcd37030ce4cde6f4bea83385dd425
ETSI / Rest of the World	480-0080.zip	aa833f0bfa7933103b917bbfec3d2560

3. Compare the md5 checksums of your files to those in Table 1 to verify you're using the same versio.
3. Extract the LWB driver source code and firmware.

```
user@buildhost$ cd ~
user@buildhost:~$ ls 930-0075.zip
930-0075.zip
user@buildhost:~$ md5sum 930-0075.zip
85db1b2f5c296e80e7ad783a33ec0c10  930-0075.zip
user@buildhost:~$ unzip 930-0075.zip
Archive:  930-0075.zip
  inflating: backports-laird-3.5.2.19.tar.bz2
user@buildhost:~$ tar xf backports-laird-3.5.2.19.tar.bz2 -C projects/
user@buildhost:~$ ls 480-0079.zip
480-0079.zip
user@buildhost:~$ md5sum 480-0079.zip
53bcd37030ce4cde6f4bea83385dd425  480-0079.zip
user@buildhost:~$ unzip 480-0079.zip
Archive:  480-0079.zip
   creating: 480-0079/
 extracting: 480-0079/lwb-bcm4343w-20160725.tar.bz2
user@buildhost:~$ tar xf 480-0079/lwb-bcm4343w-20160725.tar.bz2 -C
projects/
user@buildhost:~$ ls -1 projects/lwb-bcm4343w-20160725/
BCM4343A1_001.002.009.0038.0000_Generic_UART_37_4MHz_wlbga_ref_OTP.hcd
bcmdhd_4343w-04_15_2016.cal
fw_bcmdhd_4343w-04_15_2016.bin
fw_bcmdhd_mfgtest_4343w-04_15_2016.bin
```

Your *projects* directory should appear as follows (irrelevant columns deleted).

```
user@buildhost:~$ cd projects/
user@buildhost:~/projects$ ls -l
total 16
drwxrwxr-x [...] fsl-release-bsp
drwxrwxr-x [...] laird-backport-3.5.2.19
drwxrwxr-x [...] lwb-bcm4343w-20160725
```

4. Generate the configuration that guides the build process.

```
user@buildhost:~/projects/laird-backport-3.5.2.19$ defconfig-lwb-fcc
Generating local configuration database from kernel ... done.
cc -Wall -Wmissing-prototypes -Wstrict-prototypes -O2 -fomit-frame-pointer   -c
-o conf.o conf.c
cc -Wall -Wmissing-prototypes -Wstrict-prototypes -O2 -fomit-frame-pointer   -c
-o zconf.tab.o zconf.tab.c
In file included from zconf.tab.c:2503:0:
menu.c: In function 'get_symbol_str':
menu.c:561:18: warning: 'jump' may be used uninitialized in this function [-
Wmaybe-uninitialized]
     jump->offset = r->len - 1;
                  ^
menu.c:515:19: note: 'jump' was declared here
  struct jump_key *jump;
                   ^
cc   conf.o zconf.tab.o   -o conf
boolean symbol HWMON tested for 'm'? test forced to 'n'
boolean symbol HWMON tested for 'm'? test forced to 'n'
#
# configuration written to .config
#
```

For devices operating in the Rest of the World, the configuration is built with the following command:

```
make defconfig-lwb-esti
```

5. Set the environment variables that tell the driver build about your cross-development environment.

```
user@buildhost:$ export CROSS_COMPILE="${HOME}/projects/fsl-release-bsp/build-
imx6ul-fb/tmp/sysroots/x86_64-linux/usr/bin/arm-poky-linux-gnueabi/arm-poky-
linux-gnueabi-"
user@buildhost$ export ARCH="arm"
user@buildhost$ export KLIB_BUILD="${HOME}/projects/fsl-release-bsp/build-
imx6ul-fb/tmp/work/imx6ulevk-poky-linux-gnueabi/linux-imx/4.1.15-r0/build"
```

6. Invoke *make* to build the drivers.

```
user@buildhost:~/projects/laird-backport-3.5.2.19$ make
Generating local configuration database from kernel ... done.
[...]
  Building modules, stage 2.
  MODPOST 4 modules
  CC      /home/user/projects/laird-backport-3.5.2.19/compat/compat.mod.o
  LD [M]  /home/user/projects/laird-backport-3.5.2.19/compat/compat.ko
  CC      /home/user/projects/laird-backport-
3.5.2.19/drivers/net/wireless/brcm80211/brcmfmac/brcmfmac.mod.o
  LD [M]  /home/user/projects/laird-backport-
3.5.2.19/drivers/net/wireless/brcm80211/brcmfmac/brcmfmac.ko
  CC      /home/user/projects/laird-backport-
3.5.2.19/drivers/net/wireless/brcm80211/brcmutil/brcmutil.mod.o
  LD [M]  /home/user/projects/laird-backport-
3.5.2.19/drivers/net/wireless/brcm80211/brcmutil/brcmutil.ko
  CC      /home/user/projects/laird-backport-
3.5.2.19/net/wireless/cfg80211.mod.o
  LD [M]  /home/user/projects/laird-backport-3.5.2.19/net/wireless/cfg80211.ko
```

Four kernel modules were built. You'll copy these onto the board's Root File System in a moment.

```
user@buildhost:~/projects/laird-backport-3.5.2.19$ find . -name '*.ko'
./drivers/net/wireless/brcm80211/brcmfmac/brcmfmac.ko
./drivers/net/wireless/brcm80211/brcmutil/brcmutil.ko
./net/wireless/cfg80211.ko
./compat/compat.ko
```

7. Close the current shell.

```
user@buildhost:~/projects/laird-backport-3.5.2.19$ exit
```

8. Open a new Terminal and re-source the build environment.
9. Create an installer for a software development kit (SDK) for our target board. This takes a while.

```
user@buildhost:~$ cd projects/fsl-release-bsp/
user@buildhost:~/projects/fsl-release-bsp$ source setup-environment build-
imx6ul-fb/

Welcome to Freescale Community BSP

The Yocto Project has extensive documentation about OE including a
reference manual which can be found at:
    http://yoctoproject.org/documentation

For more information about OpenEmbedded see their website:
    http://www.openembedded.org/

You can now run 'bitbake <target>'

Common targets are:
    core-image-minimal
    meta-toolchain
    meta-toolchain-sdk
    adt-installer
    meta-ide-support

Your configuration files at build-imx6ul-fb/ have not been touched.
user@buildhost:~/projects/fsl-release-bsp/build-imx6ul-fb$ bitbake core-image-
base -c populate_sdk
[...]
NOTE: Tasks Summary: Attempted 2459 tasks of which 1711 didn't need to be rerun
and all succeeded.
[...]
```

10. Install the SDK by running the SDK installer.

```
user@buildhost:~/projects/fsl-release-bsp/build-imx6ul-fb$
./tmp/deploy/sdk/fsl-imx-fb-glibc-x86_64-core-image-base-cortexa7hf-vfp-neon-
toolchain-4.1.15-1.2.0.sh

Freescale i.MX Release Distro SDK installer version 4.1.15-1.2.0
================================================================
Enter target directory for SDK (default: /opt/fsl-imx-fb/4.1.15-1.2.0):
~/projects/fsl-imx-fb/4.1.15-1.2.0
You are about to install the SDK to "/home/user/projects/fsl-imx-fb/4.1.15-
1.2.0". Proceed[Y/n]? y
Extracting SDK.................................done
Setting it up...done
SDK has been successfully set up and is ready to be used.
Each time you wish to use the SDK in a new shell session, you need to source
the environment setup script e.g.
 $ . /home/user/projects/fsl-imx-fb/4.1.15-1.2.0/environment-setup-cortexa7hf-
vfp-neon-poky-linux-gnueabi
```

11. Source the SDK's environment variable.

```
user@buildhost:~/projects/fsl-release-bsp/build-imx6ul-fb$ source
/home/user/projects/fsl-imx-fb/4.1.15-1.2.0/environment-setup-cortexa7hf-vfp-
neon-poky-linux-gnueabi
```

You must execute the *brcm_patchram_plus* utility on your target board at runtime to load the firmware into the LWB's Bluetooth subsystem.

12. Download the *brcm_patchram_plus* package.
13. Compare its md5sum against the one shown below.

```
user@buildhost$ cd ~
user@buildhost:~$ wget
https://github.com/LairdCP/brcm_patchram/archive/brcm_patchram_plus_1.1.tar.gz
user@buildhost:~$ md5sum brcm_patchram_plus_1.1.tar.gz
3c03e03ce4ce11ea131702779906c6b3  brcm_patchram_plus_1.1.tar.gz
user@buildhost:~$ tar xf brcm_patchram_plus_1.1.tar.gz -C projects/
user@buildhost:~$ ls -1 projects/
brcm_patchram-brcm_patchram_plus_1.1
```

```
fsl-imx-fb
fsl-release-bsp
laird-backport-3.5.2.19
lwb-bcm4343w-20160725
user@buildhost:~$ cd projects/brcm_patchram-brcm_patchram_plus_1.1/
user@buildhost:~/projects/brcm_patchram-brcm_patchram_plus_1.1$ ls -1
brcm_patchram_plus.1
brcm_patchram_plus.c
brcm_patchram_plus_h5.c
brcm_patchram_plus_usb.c
LICENSE
Makefile
README.md
user@buildhost:~/projects/brcm_patchram-brcm_patchram_plus_1.1$ make
[warnings scroll past]
arm-poky-linux-gnueabi-gcc  -march=armv7-a -mfloat-abi=hard -mfpu=neon -
mtune=cortex-a7 --sysroot=/home/user/projects/fsl-imx-fb/4.1.15-
1.2.0/sysroots/cortexa7hf-vfp-neon-poky-linux-gnueabi -Wl,-O1 -Wl,--hash-
style=gnu -Wl,--as-needed  brcm_patchram_plus_usb.o  -lbluetooth -o
brcm_patchram_plus_usb
user@buildhost:~/projects/brcm_patchram-brcm_patchram_plus_1.1$ ls -1
brcm_patchram_plus
brcm_patchram_plus.1
brcm_patchram_plus.c
brcm_patchram_plus_h5
brcm_patchram_plus_h5.c
brcm_patchram_plus_h5.o
brcm_patchram_plus.o
brcm_patchram_plus_usb
brcm_patchram_plus_usb.c
brcm_patchram_plus_usb.o
LICENSE
Makefile
README.md
```

14. If it matches, extract the tarball, change to its source code directory, and invoke *make* to build the package. *Make* cross-compiles *brcm_patchram_plus* according to the environment variables that are configured by the SDK's *environment-setup-[…]* script.
15. Observe that running *make* produced the executable *brcm_patchram_plus*.

6.6 Installing the Firmware and Driver Modules to the System Image

To install the firmware and driver modules to the system image, follow these steps:

1. Mount the board's Root File System (the SD card's second partition). Your SD card should still be in your build host.

```
user@buildhost$ sudo mount /dev/mmcblk0p2 /mnt
```

2. Copy the LWB's firmware files onto the board's Root File System. To do this, you must first create the directory where they belong.

```
user@buildhost$ cd /mnt/lib/firmware
user@buildhost:/mnt/lib/firmware$ sudo mkdir brcm
```

3. Copy the files into the directory and inspect the result.

```
user@buildhost$ cd ~/projects/lwb-bcm4343w-20160725/
user@buildhost:~/projects/lwb-bcm4343w-20160725$ sudo cp bcmdhd_4343w-
04_15_2016.cal brcmfmac43430-sdio.txt fw_bcmdhd_4343w-04_15_2016.bin
brcmfmac43430-sdio.bin
BCM4343A1_001.002.009.0038.0000_Generic_UART_37_4MHz_wlbga_ref_OTP.hcd
/mnt/lib/firmware/brcm/
```

Your directory contents should match the following (irrelevant columns deleted).

```
user@buildhost:/mnt/lib/firmware/brcm$ ls -l
total 396
-rwxr-xr-x
BCM4343A1_001.002.009.0038.0000_Generic_UART_37_4MHz_wlbga_ref_OTP.hcd
-rwxr-xr-x bcmdhd_4343w-04_15_2016.cal
lrwxrwxrwx brcmfmac43430-sdio.bin -> fw_bcmdhd_4343w-04_15_2016.bin
lrwxrwxrwx brcmfmac43430-sdio.txt -> bcmdhd_4343w-04_15_2016.cal
-rwxr-xr-x fw_bcmdhd_4343w-04_15_2016.bin
```

4. Copy the drivers you built along with their directory structure onto the board's Root File System.

```
user@buildhost:/mnt/lib/firmware/brcm$ cd ~/projects/laird-backport-3.5.2.19
user@buildhost:~/projects/laird-backport-3.5.2.19$ find . -name '*.ko' | sudo
xargs cp --parents -t /mnt/lib/modules/4.1.15-1.2.0+g77f6154/kernel/
```

```
user@buildhost:~/projects/laird-backport-3.5.2.19$ cd ~/projects/brcm_patchram-
brcm_patchram_plus_1.1/
user@buildhost:~/projects/brcm_patchram-brcm_patchram_plus_1.1$ sudo cp
brcm_patchram_plus /mnt/usr/bin
```

5. Review the result.

```
user@buildhost:~/projects/laird-backport-3.5.2.19$ cd /mnt/
user@buildhost:/mnt$ find . -name 'compat.ko' -o -name 'brcmfmac.ko' -o -name
'brcmutil.ko' -o -name 'cfg80211.ko' -o -name
'BCM4343A1_001.002.009.0038.0000_Generic_UART_37_4MHz_wlbga_ref_OTP.hcd' -o -
name 'brcm_patchram_plus'
./usr/bin/brcm_patchram_plus
find: `./lost+found': Permission denied
./lib/modules/4.1.15-1.2.0+g77f6154/kernel/net/wireless/cfg80211.ko
./lib/modules/4.1.15-
1.2.0+g77f6154/kernel/drivers/net/wireless/brcm80211/brcmutil/brcmutil.ko
./lib/modules/4.1.15-
1.2.0+g77f6154/kernel/drivers/net/wireless/brcm80211/brcmfmac/brcmfmac.ko
./lib/modules/4.1.15-1.2.0+g77f6154/kernel/compat/compat.ko
```

```
./lib/firmware/brcm/BCM4343A1_001.002.009.0038.0000_Generic_UART_37_4MHz_wlbga
_ref_OTP.hcd
```

6. You are now done configuring the SD card so you can unmount it.

```
user@buildhost:/mnt$ cd ~
user@buildhost:~$ sudo umount /dev/mmcblk0*
umount: /dev/mmcblk0: not mounted
umount: /dev/mmcblk0p1: not mounted
```

7. Physically eject the SD card from the build host.

7 Connecting the LWB to the Board

To connect the LWB module to the board, follow these steps:

1. Power the board off.
2. Ensure that the LWB's jumpers are configured as:
   - J4: *VCCI/O* and *3.3V* pins shorted together
   - J5: Disconnected
   - J6: Both pins shorted together
   - J7: *VBATT* and *SDIO* pins shorted together
3. Insert the LWB into the board's SD card slot.
4. Use jumper wires to make the connections listed in the table below.

Note: These LWB pin numbers are only valid for the specific LWB part numbers listed in the *Required Materials* section of this document.

Signal	Signal Direction	Target Board	LWB
WL_REG_ON	Host > LWB	J1704-P10	J3-P11
BT_REG_ON	Host > LWB	J1704-P9	J3-P13
Bluetooth UART:			
Tx	Host > LWB	J1703-P2	J3-P10
Rx	LWB > Host	J1703-P1	J3-P8
RTS	Host > LWB	J1703-P4	J3-P6
CTS	LWB > Host	J1703-P3	J3-P14

We've assigned the UART signal names in the table above per RS-232 conventions, given we treat the target board as our *Data Terminal Equipment (DTE)* and the *LWB as our Data Circuit-Terminating Equipment (DCE)*.

Note: Documentation, schematics, code, and configuration files for the target board and the LWB do not consistently adhere to the convention defined here.

5. On the build host, open a minicom serial terminal to the board following the same procedure as you did previously.

8 USING WI-FI

8.1 Manually Loading Driver Modules

To manually load driver modules, follow these steps:

1. Insert the SD card.
2. Power up the board.
3. Log in as *root*.
   Notice that the kernel lists no wireless network interfaces.

```
root@imx6ulevk:~# ip link show wlan0
ip: can't find device 'wlan0'
```

   Observe that the wireless drivers are not built into the kernel, as indicated by this empty result.

```
root@imx6ulevk:~# cat /lib/modules/$(uname -r)/modules.builtin | grep -e compat
-e cfg80211 -e brcmutil -e brcmfmac
root@imx6ulevk:~#
```

   Also observe that the wireless driver modules are not loaded into the kernel, as indicated by this empty result.

```
root@imx6ulevk:~# lsmod | grep -e compat -e cfg80211 -e brcmutil -e brcmfmac
root@imx6ulevk:~#
```

4. Manually load your driver modules, one by one, and watch their output messages.

```
root@imx6ulevk:~# insmod /lib/modules/4.1.15-
1.2.0+g77f6154/kernel/compat/compat.ko
Loading modules backported from Linux version v4.1.13-85-gf76a7a8
Backport generated by backports.git 8011bc7

root@imx6ulevk:~# insmod /lib/modules/4.1.15-
1.2.0+g77f6154/kernel/net/wireless/cfg80211.ko
cfg80211: Calling CRDA to update world regulatory domain
cfg80211: World regulatory domain updated:
cfg80211:  DFS Master region: unset
cfg80211:   (start_freq - end_freq @ bandwidth), (max_antenna_gain, max_eirp),
(dfs_cac_time)
cfg80211:   (2402000 KHz - 2472000 KHz @ 40000 KHz), (N/A, 300 mBm), (N/A)
cfg80211:   (2457000 KHz - 2482000 KHz @ 40000 KHz), (N/A, 300 mBm), (N/A)
cfg80211:   (2474000 KHz - 2494000 KHz @ 20000 KHz), (N/A, 300 mBm), (N/A)
cfg80211:   (5170000 KHz - 5250000 KHz @ 40000 KHz), (N/A, 300 mBm), (N/A)
cfg80211:   (5735000 KHz - 5835000 KHz @ 40000 KHz), (N/A, 300 mBm), (N/A)

root@imx6ulevk:~# insmod /lib/modules/4.1.15-
1.2.0+g77f6154/kernel/drivers/net/wireless/brcm80211/brcmutil/brcmutil.ko

root@imx6ulevk:~# insmod /lib/modules/4.1.15-
1.2.0+g77f6154/kernel/drivers/net/wireless/brcm80211/brcmfmac/brcmfmac.ko
brcmfmac: brcmf_sdio_drivestrengthinit: No SDIO Drive strength init done for
chip 43430 rev 1 pmurev 24
usbcore: registered new interface driver brcmfmac
brcmfmac: brcmf_c_preinit_dcmds: Firmware version = wl0: Dec 29 2015 15:56:15
version 7.45.41.24 (r608913) FWID 01-4e412465
brcmfmac: brcmf_cfg80211_reg_notifier: not a ISO3166 code
```

The driver modules are now listed as loaded into the kernel.

```
root@imx6ulevk:~# lsmod | grep -e compat -e cfg80211 -e brcmutil -e brcmfmac
brcmfmac              187936  0
brcmutil                8292  1 brcmfmac
cfg80211              233253  1 brcmfmac
compat                  1744  2 cfg80211,brcmfmac
```

The kernel now enumerates a wireless network interface named *wlan0*.

```
root@imx6ulevk:~# ip link show wlan0

7: wlan0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop qlen 1000

    link/ether 00:25:ca:07:01:95 brd ff:ff:ff:ff:ff:ff
```

8.2 Connecting to a Wi-Fi Access Point

The new interface is not currently connected to a network.

```
root@imx6ulevk:~# iw wlan0 link
Not connected.
```

To connect your device to a network, follow these steps:

1. Define the settings for connecting to the network.

   The wpa_supplicant program manages the process of connecting to networks. Its settings are contained in the **/etc/wpa_supplicant.conf** file.

2. Edit the **/etc/wpa_supplicant.conf**.

```
root@imx6ulevk:~# vi /etc/wpa_supplicant.conf
```

3. Press **i** to switch vi from command mode to edit mode.
4. Paste the following as the file's contents.

```
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=0
update_config=1

network={
        ssid="ccopen"
        key_mgmt=NONE
}
```

   Notice that the network.ssid property specifies your network's name (SSID) as **ccopen**.

5. Press **ESC** to exit edit mode and to re-enter command mode.
6. Type *:wq* to write your changes to the disk and quit.
7. Verify that your wireless router is on and ready. Ensure that the SSID is set to **ccopen** and its DHCP server is enabled.
8. Start the wpa_supplicant.

Note: In our case, we temporarily ran the supplicant in the foreground to easily display its messages.

```
root@imx6ulevk:~# wpa_supplicant -i wlan0 -c /etc/wpa_supplicant.conf
Successfully initialized wpa_supplicant
rfkill: Cannot open RFKILL control device
brcmfmac: brcmf_add_if: ERROR: netdev:wlan0 already exists
brcmfmac: brcmf_add_if: ignore IF event
wlan0: Trying to associate with SSID 'ccopen'
cfg80211: Calling CRDA for country: US
cfg80211: Regulatory domain changed to country: US
cfg80211:  DFS Master region: FCC
cfg80211:   (start_freq - end_freq @ bandwidth), (max_antenna_gain, max_eirp),
(dfs_cac_time)
cfg80211:   (2402000 KHz - 2472000 KHz @ 40000 KHz), (N/A, 300 mBm), (N/A)
cfg80211:   (5170000 KHz - 5250000 KHz @ 40000 KHz), (N/A, 300 mBm), (N/A)
cfg80211:   (5250000 KHz - 5330000 KHz @ 40000 KHz), (N/A, 300 mBm), (0 s)
cfg80211:   (5490000 KHz - 5600000 KHz @ 40000 KHz), (N/A, 300 mBm), (0 s)
cfg80211:   (5650000 KHz - 5710000 KHz @ 40000 KHz), (N/A, 300 mBm), (0 s)
cfg80211:   (5735000 KHz - 5835000 KHz @ 40000 KHz), (N/A, 300 mBm), (N/A)
cfg80211:   (57240000 KHz - 63720000 KHz @ 2160000 KHz), (N/A, 0 mBm), (N/A)
wlan0: Associated with 54:78:1a:42:b0:d0
wlan0: CTRL-EVENT-CONNECTED - Connection to 54:78:1a:42:b0:d0 completed [id=0
id_str=]
wlan0: CTRL-EVENT-REGDOM-CHANGE init=COUNTRY_IE type=COUNTRY alpha2=US
```

The CTRL-EVENT-CONNECTED message indicates that the wpa_supplicant successfully connected to your network.

9. Press **Ctrl+c** to kill the wpa_supplicant, then restart the wpa_supplicant in the background, out of your way.

```
root@imx6ulevk:~# wpa_supplicant -B -i wlan0 -c /etc/wpa_supplicant.conf
```

The status of wlan0's link no longer displays *Not connected*. Rather, it shows that wlan0 is connected to the ccopen network.

```
root@imx6ulevk:~# iw wlan0 link
Connected to 54:78:1a:42:b0:d0 (on wlan0)
        SSID: ccopen
        freq: 2412
        signal: -42 dBm
        tx bitrate: 65.0 MBit/s

        bss flags:
        dtim period:    1
        beacon int:     102
```

8.3 Configuring Internet Protocol

Although your device is connected at the Wi-Fi level, it’s not yet configured at the Internet Protocol networking level. It has no IP address and no useful IP routes.

```
root@imx6ulevk:~# ip address list wlan0
7: wlan0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop qlen 1000
    link/ether 00:25:ca:07:01:95 brd ff:ff:ff:ff:ff:ff
root@imx6ulevk:~# ip route show
root@imx6ulevk:~#
```

To configure your Internet protocol, follow these steps:

1. Start your DHCP client to request your IP address and routing configuration from the network's DHCP server.

   ```
   root@imx6ulevk:~# udhcpc -i wlan0 -n -q
   udhcpc (v1.23.2) started
   Sending discover...
   Sending select for 10.1.99.43...
   Lease of 10.1.99.43 obtained, lease time 14400
   /etc/udhcpc.d/50default: Adding DNS 10.1.44.32
   ```

   You now have an IP address. In our example below, it’s 10.1.44.142.

   ```
   root@imx6ulevk:~# ip address list wlan0
   7: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
       link/ether 00:25:ca:07:01:95 brd ff:ff:ff:ff:ff:ff
       inet 10.1.44.142/24 brd 10.1.44.255 scope global wlan0
          valid_lft forever preferred_lft forever
       inet6 2001:1890:1230:f42e:225:caff:fe07:195/64 scope global dynamic
          valid_lft 2591954sec preferred_lft 604754sec
   ```

   You also have a default route through our router. In our example below, it has the address 10.1.44.1.

   ```
   root@imx6ulevk:~# ip route show
   default via 10.1.44.1 dev wlan0  metric 10
   10.1.44.0/24 dev wlan0  src 10.1.44.142
   ```

   The following displays our example of IP traffic being passed over the Wi-Fi link by pinging the router three times.

   ```
   root@imx6ulevk:~# ping -c 3 10.1.44.1
   PING 10.1.44.1 (10.1.44.1): 56 data bytes
   64 bytes from 10.1.44.1: seq=0 ttl=255 time=159.746 ms
   64 bytes from 10.1.44.1: seq=1 ttl=255 time=31.492 ms
   64 bytes from 10.1.44.1: seq=2 ttl=255 time=88.698 ms

   --- 10.1.44.1 ping statistics ---
   3 packets transmitted, 3 packets received, 0% packet loss
   round-trip min/avg/max = 31.492/93.312/159.746 ms
   ```

Congratulations! You've brought the LWB up on your board and have successfully passed traffic.

8.4 Automatically Loading Driver Modules at System Startup

Though you manually loaded your driver modules with *insmod*, they are not automatically reloaded after a reboot. If you want your modules to be loaded during system initialization, you can tell the module dependency database to search for and add all new modules using the following:

```
root@imx6ulevk:~# depmod -a
```

You can demonstrate that the drivers load automatically by rebooting, inspecting the loaded kernel modules, and verifying that your wireless network interface is listed (see below).

```
root@imx6ulevk:~# reboot

[...]

imx6ulevk login: root
List loaded kernel modules and search for the drivers we built

root@imx6ulevk:~# lsmod | grep -e compat -e cfg80211 -e brcmutil -e brcmfmac
brcmfmac              187936  0
cfg80211              233253  1 brcmfmac
compat                  1744  2 cfg80211,brcmfmac
brcmutil                8292  1 brcmfmac

root@imx6ulevk:~# ip link show wlan0
7: wlan0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop qlen 1000
    link/ether 00:25:ca:07:01:95 brd ff:ff:ff:ff:ff:ff
```

9 Using Bluetooth

Note: The target board only delivers power to the LWB via the SD card slot when the target board detects the LWB's Wi-Fi module during boot. This requires that the WL_REG_ON signal is asserted.

Observe that no Bluetooth interfaces are enumerated.

```
root@imx6ulevk:~# hciconfig
root@imx6ulevk:~#
```

Also observe that the board's second UART (which connects the host processor to the LWB's Bluetooth subsystem) has so far transmitted zero (0) bytes and received zero (0) bytes.

```
root@imx6ulevk:~# cat /proc/tty/driver/IMX-uart
serinfo:1.0 driver revision:
0: uart:IMX mmio:0x02020000 irq:19 tx:1873 rx:66 RTS|DTR|DSR|CD
1: uart:IMX mmio:0x021E8000 irq:235 tx:0 rx:0 DSR|CD
```

We use the *brcm_patchram_plus* utility to load the LWB Bluetooth subsystem with its firmware image and initialize the host's Bluetooth stack.

```
root@imx6ulevk:~# brcm_patchram_plus -d --patchram
/lib/firmware/brcm/BCM4343A1_001.002.009.0038.0000_Generic_UART_37_4MHz_wlbga_ref_
OTP.hcd --enable_hci --no2bytes --tosleep 1000 /dev/ttymxc1 &
[...lots of hex dump scrolls past...]
received 7
04 0e 04 01 4e fc 00
writing
01 03 0c 00
received 7
04 0e 04 01 03 0c 00
Done setting line discpline
```

Notice that the arguments to *brcm_patchram_plus* reference the **.hcd* firmware file and */dev/ttymxc1*. */dev/ttymxc1* is the device file for the board's UART that we wired to the LWB's Bluetooth subsystem.

Observe that Bluetooth interface *hci0* is now enumerated and that the associated UART is transmitting and receiving bytes. Also notice that our Bluetooth device has MAC address 00:25:CA:07:0C:D6.

```
root@imx6ulevk:~# hciconfig
hci0:   Type: BR/EDR  Bus: UART
        BD Address: 00:25:CA:07:0C:D6  ACL MTU: 1021:8  SCO MTU: 64:1
        DOWN
        RX bytes:675 acl:0 sco:0 events:35 errors:0
        TX bytes:427 acl:0 sco:0 commands:35 errors:0

root@imx6ulevk:~# cat /proc/tty/driver/IMX-uart
serinfo:1.0 driver revision:
0: uart:IMX mmio:0x02020000 irq:19 tx:117501 rx:284 RTS|DTR|DSR|CD
1: uart:IMX mmio:0x021E8000 irq:235 tx:35074 rx:1669 RTS|CTS|DTR|DSR|CD
```

Now bring up the Bluetooth interface.

```
root@imx6ulevk:~# hciconfig hci0 up
root@imx6ulevk:~# hciconfig
hci0:   Type: BR/EDR  Bus: UART
        BD Address: 00:25:CA:07:0C:D6  ACL MTU: 1021:8  SCO MTU: 64:1
        UP RUNNING
        RX bytes:1343 acl:0 sco:0 events:69 errors:0
        TX bytes:850 acl:0 sco:0 commands:69 errors:0
```

Use *hcitool* to scan for visible Bluetooth devices near us. Ensure that you have another Bluetooth device nearby, powered on, and configured to be visible to other devices.

Run the following command.

```
root@imx6ulevk:~# hcitool scan
Scanning ...
        94:65:9C:A2:12:B8       n/a
        04:76:6E:71:86:FF       Alice
        00:17:23:E0:41:E9       Bob
        00:16:A4:07:9F:6E       Charlie
```

Ping one of our remote devices.

```
root@imx6ulevk:~# l2ping -c 3 00:17:23:E0:41:E9
Ping: 00:17:23:E0:41:E9 from 00:25:CA:07:0C:D6 (data size 44) ...
44 bytes from 00:17:23:E0:41:E9 id 0 time 70.79ms
44 bytes from 00:17:23:E0:41:E9 id 1 time 41.90ms
44 bytes from 00:17:23:E0:41:E9 id 2 time 32.08ms
3 sent, 3 received, 0% loss
```

10 What Next?

Now that you've successfully integrated your LWB, how you proceed from here depends on your specific applications.

Potential next steps with Wi-Fi:

- Statically setting your IP address and routes
- Bringing up your interface automatically during boot
- Starting your DHCP client automatically during boot
- Securing your Wi-Fi connection

Potential next steps with Bluetooth:

- Automatically bringing up your interface during boot
- Pairing with another device
- Exercising the functionality associated with one or more Bluetooth profiles

11 References

11.1 Sterling-LWB

Laird Sterling-LWB Firmware

Firmware is located at http://www.lairdtech.com/product-categories/embedded-wireless/wi-fi-and-wi-fi-bt-modules Select **Sterling-LWB > Kits & Software** tab.

For devices operating in the USA or Canada select the FCC firmware 480-0079.zip (md5sum: 53bcd37030ce4cde6f4bea83385dd425)
http://www.lairdtech.com/product-categories/embedded-wireless/wi-fi-and-wi-fi-bt-modules

For devices operating in the Rest of the World select the ETSI firmware 480-0080.zip (md5sum: aa833f0bfa7933103b917bbfec3d2560)

Laird Sterling-LWB Driver Backports

930-0075.zip (md5sum: 85db1b2f5c296e80e7ad783a33ec0c10)
http://www.lairdtech.com/product-categories/embedded-wireless/wi-fi-and-wi-fi-bt-modules

Select **Sterling-LWB > Kits & Software** tab.

Laird Sterling-LWB Data Sheet
330-0190.pdf (md5sum: 6b179957f835bd985a5b596454b6ff3b)
http://www.lairdtech.com/product-categories/embedded-wireless/wi-fi-and-wi-fi-bt-modules

Select **Sterling-LWB > Documentation** tab.

Broadcom BCM4343W Data Sheet
Radio with Bluetooth 4.1, an FM Receiver, and Wireless Charging.pdf
(md5sum: 79d98b25a14f0f79644feceaf304c3aa)
http://www.cypress.com/documentation/datasheets/bcm4343w-single-chip-ieee-80211-bgn-macbasebandradio-bluetooth-41-fm?source=search&keywords=BCM4343WKUBG&cat=technical_documents

11.2 NXP i.MX 6 UltraLite

Freescale Document Archive - L4.1.15_1.2.0_LINUX_DOCS

- Freescale Yocto Project User's Guide
  fsl-yocto-L4.1.15_1.2.0-ga.tar.gz (md5sum: 709b2511919c5df20bd89f3b2a520f29)
  https://www.nxp.com/webapp/Download?colCode=L4.1.15_1.2.0_LINUX_DOCS&location=null&fasp=1&WT_TYPE=Supporting%20Information&WT_VENDOR=FREESCALE&WT_FILE_FORMAT=gz&WT_ASSET=Documentation&fileExt=.gz&Parent_nodeId=1276810298241720831102&Parent_pageType=produc

NXP i.MX 6UltraLite Applications Processor Reference Manual
IMX6ULRM.pdf (md5sum: 02e4cca2f1c356a313ff97e9b6185b9d)
http://www.nxp.com/files/32bit/doc/ref_manual/IMX6ULRM.pdf?fasp=1&WT_TYPE=Reference%20Manuals&WT_VENDOR=FREESCALE&WT_FILE_FORMAT=pdf&WT_ASSET=Documentation&fileExt=.pdf

NXP i.MX UltraLite EVK Design Files

- Computer Module Schematic - SPF-28617_C1.pdf
- Breakout Board Schematic - SPF-28616_C.pdf

MCIMX6UL-EVK_DESIGNFILES.zip (md5sum: c589d6668979b5f5467a5f89d4987f6e)
http://cache.nxp.com/files/32bit/hardware_tools/schematics/MCIMX6UL-EVK_DESIGNFILES.zip?fsrch=1&sr=1&pageNum=1

11.3 Linux and Open Source

Ubuntu 16.04.1
ubuntu-16.04.1-desktop-amd64.iso (md5sum: 17643c29e3c4609818f26becf76d29a3)
http://releases.ubuntu.com/16.04/

brcm_patchram_plus
brcm_patchram_plus_1.1.tar.gz (md5sum: 3c03e03ce4ce11ea131702779906c6b3)
https://github.com/LairdCP/brcm_patchram/archive/brcm_patchram_plus_1.1.tar.gz

repo
repo (md5um: 6ee3b113832e982b9b085ad0863e4351)
http://commondatastorage.googleapis.com/git-repo-downloads/repo

12 FAQ and Common Problems

12.1 Build Host

Problem: An *apt* package manager command such as *apt-get update* or *apt-get install* fails with the following error messages:

```
user@buildhost:~$ sudo apt-get update
[sudo] password for user:
Hit:1 http://us.archive.ubuntu.com/ubuntu xenial InRelease
Get:2 http://us.archive.ubuntu.com/ubuntu xenial-updates InRelease [95.7
kB]
Hit:3 http://us.archive.ubuntu.com/ubuntu xenial-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu xenial-security InRelease
Fetched 95.7 kB in 5s (17.2 kB/s)
E: Could not get lock /var/lib/dpkg/lock - open (11: Resource temporarily
unavailable)
E: Unable to lock the administration directory (/var/lib/dpkg/), is
another process using it?
```

Common cause: An *apt* command is currently running on the system. For example, you may have explicitly invoked an *apt* command in a different terminal or in the background of the current terminal and it is still running or the operating system may be checking for or installing updates without your knowledge.

Problem: An *apt-get install* command aborts with the following error message after the user confirms a desire to download and install packages:

```
After this operation, 862 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Abort.
```

Common cause: Uncertain, although the command succeeds when retried.

Problem: Receive the following warning message when running bitbake on the build host:

```
WARNING: Host distribution "Ubuntu-16.04" has not been validated with
this version of the build system; you may possibly experience unexpected
failures. It is recommended that you use a tested distribution.
```

Answer: We have received the same warning but have observed no ill effects.

Problem: bitbake reports the following:

```
WARNING: Failed to fetch URL http://[...], attempting MIRRORS if
available.
```

Common cause: bitbake is trying to retrieve source code from a repository server. Because the server is temporarily or permanently unreachable, bitbake tries another server that might have the desired content. This is not a problem as long as bitbake successfully retries the content from a mirror server; verify that bitbake displays a message indicating all tasks succeeded just prior to exiting:

```
NOTE: Tasks Summary: Attempted [...] tasks of which [...] didn't need to
be rerun and all succeeded.
```

12.2 Target Board

Problem: Target board's U-Boot bootloader displays the following warning message Warning - bad CRC, using default environment.

```
U-Boot 2015.04imx_v2015.04_4.1.15_1.2.0_ga+gede7538 (Oct 07 2016 -
14:51:35)

CPU:   Freescale i.MX6UL rev1.0 at 396 MHz
CPU:   Temperature 30 C
Reset cause: POR
Board: MX6UL 14x14 EVK
I2C:   ready
DRAM:  512 MiB
MMC:   FSL_SDHC: 0, FSL_SDHC: 1
*** Warning - bad CRC, using default environment
```

Common cause: This is the intended behavior. U-Boot is indicating that it did not find U-Boot environment data saved to the SD card and U-Boot is therefore falling back to its compiled-in defaults. We want U-Boot to use its compiled-in defaults.

Problem: Shortly after booting and/or logging in, target board's console displays the following message: *random: nonblocking pool is initialized*

Common cause: This appears to be normal for the board.

Problem: No wireless network interface listed by *ip link*.

Common cause: The drivers did not load successfully.

Problem: While loading brcmfmac.ko driver, receive the following error message:
brcmfmac_sdio mmc0:0001:1: Direct firmware load for brcm/brcmfmac43430-sdio.bin failed with error -2

Common cause: Symlinks to firmware files are not valid.

Problem: While loading the *brcmfmac.ko* driver, some messages are missing.

Expected:

```
root@imx6ulevk:~# insmod /lib/modules/4.1.15-
1.2.0+g77f6154/kernel/drivers/net/wireless/brcm80211/brcmfmac/brcmfmac.ko
brcmfmac: brcmf_sdio_drivestrengthinit: No SDIO Drive strength init done
for chip 43430 rev 1 p4
usbcore: registered new interface driver brcmfmac
brcmfmac: brcmf_c_preinit_dcmds: Firmware version = wl0: Dec 29 2015
15:56:15 version 7.45.41.24 (r608913) FWID 05
brcmfmac: brcmf_cfg80211_reg_notifier: not a ISO3166 code
```

Actual:

```
root@imx6ulevk:~# insmod /lib/modules/4.1.15-
1.2.0+g77f6154/kernel/drivers/net/wireless/brcm80211/brcmfmac/brcmfmac.ko
usbcore: registered new interface driver brcmfmac
```

Common cause: Wi-Fi hardware module is not detected:

- SD Card not correctly inserted in slot
- SD Card's power configuration jumpers incorrectly configured
- SD Card not receiving power
- Module's "WL_REG_ON" line not driven high

Problem: While loading the *brcmfmac.ko* driver, receive the following message: *No SDIO Drive strength init*.

Common cause: Symlinks to firmware files are not valid.

Problem: The *brcm_patchram_plus* utility slowly reports writing over and over again when debug output is enabled.

Common cause: The board cannot communicate with LWB. This may indicate LWB is not receiving power or that there is a problem with the UART wiring.

---

13 Paths of Interest

13.1 Build Host

- BSP build directory: ~/projects/fsl-release-bsp/build-imx6ul-fb
- SD card image: ~/projects/fsl-release-bsp/build-imx6ul-fb/tmp/deploy/images/imx6ulevk/core-image-base-imx6ulevk.sdcard
- System log: /var/log/syslog
- Target serial terminal device: /dev/ttyUSB[N]
- Kernel configuration from "menuconfig": ~/projects/fsl-release-bsp/build-imx6ul-fb/tmp/work/imx6ulevk-poky-linux-gnueabi/linux-imx/4.1.15-r0/build/.config
- Kernel configuration to build: ~/projects/fsl-release-bsp/build-imx6ul-fb/tmp/work-shared/imx6ulevk/kernel-source/arch/arm/configs/imx_v7_defconfig
- SD card device: /dev/mmcblk[N]
- Device tree source: ~/projects/fsl-release-bsp/build-imx6ul-fb/tmp/work-shared/imx6ulevk/kernel-source/arch/arm/boot/dts/imx6ul-14x14-evk.dts
- Kernel documentation: ~/projects/fsl-release-bsp/build-imx6ul-fb/tmp/work-shared/imx6ulevk/kernel-source/Documentation

13.2 Target Board

- Kernel Modules (in "insmod" order)
  - /lib/modules/4.1.15-1.2.0+g77f6154/kernel/compat/compat.ko
  - /lib/modules/4.1.15-1.2.0+g77f6154/kernel/net/wireless/cfg80211.ko
  - /lib/modules/4.1.15-1.2.0+g77f6154/kernel/drivers/net/wireless/brcm80211/brcmutil/brcmutil.ko
  - /lib/modules/4.1.15-1.2.0+g77f6154/kernel/drivers/net/wireless/brcm80211/brcmfmac/brcmfmac.ko
- Utilities
  - /usr/bin/brcm_patchram_plus
- Firmware Files and Symlinks
  - /lib/firmware/brcm/bcmdhd_4343w-04_15_2016.cal
  - /lib/firmware/brcm/fw_bcmdhd_4343w-04_15_2016.bin
  - /lib/firmware/brcm/brcmfmac43430-sdio.bin -> fw_bcmdhd_4343w-04_15_2016.bin
  - /lib/firmware/brcm/brcmfmac43430-sdio.txt -> bcmdhd_4343w-04_15_2016.cal
  - lib/firmware/brcm/BCM4343A1_001.002.009.0038.0000_Generic_UART_37_4MHz_wlbga_ref_OTP.hcd
- Bluetooth serial terminal device: */dev/ttymxc1*