Integrating the Laird Sterling-LWB with NXP i.MX6 UltraLite on Linux

21 June 2018
330-0201-R2.6

Revision History

Version	Date	Notes	Approver
2.3	01 Dec 2016	N/A	(see Laird version control system)
2.4	04 Apr 2017	N/A	(see Laird version control system)
2.5	11 May 2017	N/A	(see Laird version control system)
2.6	16 July 2018	add Japan regulatory info	(Eric Bentley)

Contents

1 Introduction

This document provides a walkthrough for engineers to integrate the Laird Sterling-LWB's Wi-Fi and Bluetooth functionality with a Freescale i.MX6 UltraLite Evaluation Kit running Linux. This walk through is intended to be a learning tool. The results of this walkthrough are not sufficient for production release, but are sufficient for demonstrating the basic Wi-Fi and Bluetooth functionality of the Sterling-LWB.

First, we 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 via an open Wi-Fi network and Bluetooth.

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

2 Required Materials

The following are required for this integration:

- 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 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 is not listed, please contact Laird's customer support team.
- Freescale i.MX6 UltraLite evaluation kit
  - Freescale part number MCIMX6UL-EVK
  - 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.MX6 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 i.MX6 UltraLite evaluation kit (example: Adafruit product ID 85)
- Male-female jumper wire (example: Adafruit product ID 826 / Digi-Key part number 1528-1162-ND)
- Soldering tools and consumables (0.1” pitch through-hole)
- Voltmeter
- Computer running Ubuntu 16.04.1 to be used as build host which includes:
  - 40 GB free disk space minimum
  - *root* privileges
  - Available USB port
  - Micro SD card slot or equivalent adapter
  - Internet connection
  - http://releases.ubuntu.com/16.04/
- Arbitrary other Bluetooth device (computer, smartphone, etc.) configured to be visible to other devices
- ESD mat
- ESD wrist strap

The following 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
- Rectangular connector housing, 2x8 positions, 0.1” pitch (example: Digi-Key part number 952-2037-ND)

3 General Guidance

3.1 Estimated Time to Complete

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

3.2 Cautions and Warnings

Adhere to all cautions and warnings included with all materials. The LWB and the i.MX6 UltraLite Evaluation Kit can be damaged by electrostatic discharge (ESD); handle accordingly.

3.3 Regional Radio Frequency Regulatory Compliance

Radio frequency regulations vary throughout the world. During the integration process, you must select the set of regulations with which the integration is to comply.

For integrations operated within the United States or Canada, select the FCC-compliant options.

For integrations operated within Japan, select the Giteki compliant options.

For integrations operated within ETSI or RCM (AU), select the ETSI-compliant options.

3.4 Prerequisite Skills

We assume that you are:

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

3.5 Terminology

The following terminology is applicable to this document:

- **LWB** - Sterling-LWB Development Kit with SD card form factor
- **Target** or **board** - Freescale i.MX6 UltraLite Evaluation Kit
- **Pin** - May be used to refer to a processor's I/O *pad*
- **Freescale** and **NXP** are interchangeable
- **J123-P4** refers to pin 4 of connector 123

Note: Laird acquired LS Research (LSR). We are in the process of updating documentation to reflect this acquisition.

3.6 Build Host Linux Distribution and Version

This guide was produced using 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 a different procedure. 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. You can do this by installing it to and booting from a USB drive instead. Search the internet for *persistent live Ubuntu USB* for details.

3.7 LWB SD Card Part Numbers and Revision Levels

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

3.8 About This Document

Links for referenced documents and downloads are provided at the end of this document.

Common problems and their causes are listed at the end of this document.

4 Modifying the Breakout Board

To integrate the Sterling-LWB with the i.MX6 UltraLite evaluation kit, you must make connections between the target board and the LWB. You must also make additional modifications within the target board. Connections between the target board and the LWB are carried through both the SD card connector and discrete wiring. Discrete wires interface with the target board through the target board's Arduino connection points.

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 may be helpful to use an arbitrary Arduino Uno R3 compatible shield as a jig for aligning the headers during the soldering process. To do so, mate all headers with the pins of the shield prior to soldering, insert all headers' pins through the board at once, solder, and remove the jig shield when complete.

The necessary modification within the target board is made by soldering a short 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

Begin this process with your 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 document with minor deviations.

5.1 Installing the Freescale BSP and Its Dependencies

To install the software packages that are required by the Freescale BSP, follow these steps:

1. 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.

2. Install the *repo* utility, which enables 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
[...]
user@buildhost:~$ chmod a+x ~/bin/repo
user@buildhost:~$ which repo
/home/user/bin/repo
```

   The output of the *which repo* command tells you specifically which program your shell invokes when you run the *repo* command. It should report the file you just downloaded.

3. 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
```

4. Use the *repo* command to retrieve Freescale's BSP from Freescale's git repository.
5. Invoke the script to set up your 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
[...]
Enable color display in this user account (y/N)? y

repo has been initialized in /home/user/projects/fsl-release-bsp
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)
```

6. Enter *y* to accept Freescale's license agreement.

   Note: Running the script changed the working directory to the build directory, **/projects/fsl-release-bsp/build-imx6ul-fb**. The name of the 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, follow these steps:

1. 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. Doing this takes some time, approximately one hour in our experience.

```
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.
```

2. Notice the system image file that was produced.

```
user@buildhost$ cd ~/projects/fsl-release-bsp/build-imx6ul-fb/tmp/deploy/images/imx6ulevk
user@buildhost:~/projects/fsl-release-bsp/build-imx6ul-fb/tmp/deploy/images/imx6ulevk$
  ls -l core-image-base-imx6ulevk.sdcard

lrwxrwxrwx [...] core-image-base-imx6ulevk.sdcard ->
  core-image-base-imx6ulevk-20170303202501.rootfs.sdcard

user@buildhost:~/projects/fsl-release-bsp/build-imx6ul-fb/tmp/deploy/images/imx6ulevk$
  ls -lh core-image-base-imx6ulevk-20170303202501.rootfs.sdcard

-rw-r--r-- [...] 84M [...] core-image-base-imx6ulevk-20170303202501.rootfs.sdcard
```

5.3 Writing the System Image to the SD Card

The next step is to write the image out to the SD card. To do this, follow these steps:

1. Determine what your build host calls your SD card. Begin monitoring your system's log file, the *syslog*.

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

2. Insert the SD card into your build host.
3. In the syslog's output, find the lines similar to the following and note the value of *mmcblk** (we've deleted timestamps for clarity):

```
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*.

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

   Typically, 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 itself, not 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 inserted your SD card, Ubuntu mounted the card's partition(s) and opened one or more file system browser windows. You must unmount the SD card from the build host to avoid confusing the build host when the card's filesystems are overwritten.

   Use the *dd* command to write the contents of the image directly to the raw SD card device. Notice that this is the SD card name found in the syslog.

5. 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$
```

6. Once the image is finished writing and syncing, physically eject the SD card from your build host.
7. Install the SD card into the J301 hinged micro-SD card cage atop your board's compute module. Do not insert the SD card into the 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, you can cable it to your build host and interact with it through a serial terminal program such as *minicom*.

To connect a serial terminal to the board, 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***, where ***** represents some integer.

   Because normal users cannot access these device files by default, you must add your user to a privileged group.

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

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

3. Press **Ctrl+o** to save it.
4. Press **Ctrl+x** to exit the *nano* editor.

   Your change to the **/etc/group** file does not take effect immediately. To force the change to take effect, do the following:

   (a) Close your Terminal window.
   (b) Log out of your desktop session.
   (c) Log back in to your desktop session.
   (d) Open a new Terminal window.

   You can list the groups your user is associated with to verify that the change took effect.

5. 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)
```

6. 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
```

7. Connect a USB cable between your build host and the board's J1901 micro-USB jack.
8. Wait for messages to appear in the syslog, then disconnect the USB cable.
9. 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
```

10. Make note of the valueof **ttyUSB***.
11. Press **Ctrl+c** to stop tailing thesyslog.
12. Reconnect the USB cable between the build host and and the board's J1901 jack.
13. 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
```

14. Press **Ctrl+a**, then **z** to open minicom'smenu.

```
+-----------------------------------------------------------------+
|                      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. Type **o** to configureminicom.

```
+-----[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. Select **Serial port setup** from the menu.
17. 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?                                               |
+-----------------------------------------------------------------------+
```

18. Exit the Serial Port Setup menu and return to the **[configuration]** menu.
19. 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                     |
+--------------------------+
```

20. Exit the menu.

5.5 Selecting the Boot Device

Configure the board to boot from the micro SD card. 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

With your serial terminal established, power on 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
```

...followed by scrolling messages and finally the login prompt:

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

imx6ulevk login:
[...]
root@imx6ulevk:~#
```

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

Let's 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 20e0000.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 your board by doing the following:

- Allocating pins on the i.MX6 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 your kernel.
- Adding the drivers and firmware.

6.1 Reestablishing the Build Environment

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

1. Exit from the minicom serial terminal.
   (a) Press **Ctrl+a**.
   (b) Press **x**.
   (c) Select **Yes** when minicom asks if you want to leave.
2. 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$
```

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

6.2 Configuring the Device Tree

You must configure a pin on the i.MX6 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 datasheet 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; the binary form is called many things, including the Device Tree Binary, Device Tree Blob, and Flat Device Tree. The Device Tree Compiler (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 pins (pads) on the processor to use for WL_REG_ON and BT_REG_ON. For each signal, consider the following:

- The pin must be capable of producing digital output.
- The pin 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.

After reviewing the following Freescale documents, we selected processor pin **UART5_TX_DATA** for our WL_REG_ON signal and processor pin **UART5_RX_DATA** for our BT_REG_ON signal.

- 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, as presented in the following table.

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
SODIMM Connector 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

We selected the UART5_TX_DATA and UART5_RX_DATA pins for the following reasons:

- 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 for the following reasons:
  - 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 that 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 that 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 must be populated, no devices must be removed, no traces must be cut, and no jumper wires must 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 pins' default bindings, 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 marked in the following listing to prevent your 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 /* delete */
                                MX6UL_PAD_UART5_RX_DATA__I2C2_SDA 0x4001b8b0 /* delete */
                        >;
                };
[...]
};
[...]
```

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 your pins for GPIO, you must configure up to three memory-mapped registers per pin, including the following:

- Mux control register
- Pad control register (pad configuration register)
- DAISY register (input register), if applicable

Each register instance is discussed in detail in its own section of the *i.MX6 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 five numeric fields define the address offsets of the corresponding (1) mux control register, (2) pad configuration register, (3) DAISY (input) register, and the values to be loaded into the (4) mux control register and (5) DAISY register, respectively. These address offsets and register values are defined in the *i.MX6 UltraLite Reference Manual*.

Incorporate the following marked lines into your device tree file to configure your 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 /* add */
                                MX6UL_PAD_UART5_RX_DATA__GPIO1_IO31     0x3031 /* add */
                        >;
                };
[...]
```

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 registers. The values were 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.MX6 UltraLite Reference Manual* to configure the following properties for our pins:

- 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)

Save your changes (Ctrl+o) and exit (Ctrl+x).

6.2.4 Testing GPIO Pins Under Manual Control

At this point, the substantial changes you made warrant testing. To do 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$ cd ~/projects/fsl-release-bsp/build-imx6ul-fb
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 earlier.
3. Open a serial terminal to the board, insert the SD card, boot the board, and log in as you did previously.

4. Use the debug filesystem to verify that that your device tree changes had the desired effect. 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_UART5_RX_DATA
config 00003031

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

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

6. Use the **/sys/class/gpio** facilities to verify that you can manually toggle the pins. Change to the **/sys/class/gpio** directory.

   Recall that your pins are at GPIO bank 1 pin 30 and GPIO bank 1 pin 31; this implies their GPIO pin numbers are 30 and 31 overall.

7. Echo these overall pin numbers into the *export* file and observe that new directories *gpio30* and *gpio31* now exist to represent our 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
```

8. Change into 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/gpio30# ls -1
active_low
device
directio
n edge
power
subsyste
m uevent
value
root@imx6ulevk:/sys/class/gpio/gpio30# echo out > direction
```

9. 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.

10. 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
```

11. Observe that the voltage is now 3.3 V. Change your pin's value back to a logical 0.

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

12. Observe that the voltage is again 0.0 V.

13. Repeat this exercise for your other pin.

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

14. Using a voltmeter, measure the voltage of J1704-P9 with respect to ground. Ground can be found at J1705-P7. Observe that the voltage is 0.0 V.
15. Change your 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
```

16. Observe that the voltage is now 3.3 V. Change your 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 your electrical path is intact and that you have correctly modified your device tree.

17. Capture one final observation that you will reference 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  2625mV  3400mV
    20ca000.usbphy                                              3200mV  3200mV
    20c9000.usbphy                                              3200mV  3200mV
 regulator-dummy                  0    9      0    0mV     0mA     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
```

18. Exit the serial terminal as you did before.

6.2.5 Binding GPIO Pins as Regulators

Toggling the pin's value manually is useful for testing in a development environment, but we would like a production system to control their values autonomously. You will use your pair of GPIO pins to control a pair of power regulators within the LWB. You can implement such control by employing Linux's *regulator* facility.

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 you'll configure your LWB's regulators to constantly stay on.

Again, be aware that the regulators dealt with in this discussion only manage the distribution of power within the LWB; separate regulators belonging to the target board control whether power is delivered to the LWB.

Edit the device tree file.

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

Define a pair of regulators and associate them with your GPIO pins by adding the following 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 {                              /* add */
                        compatible = "regulator-fixed";             /* add */
                        regulator-name = "wl";                      /* add */
                        gpio = <&gpio1 30 GPIO_ACTIVE_HIGH>;        /* add */
                        startup-delay-us = <100>;                   /* add */
                        enable-active-high;                         /* add */
                        regulator-min-microvolt = <3300000>;        /* add */
                        regulator-max-microvolt = <3300000>;        /* add */
                        regulator-always-on;                        /* add */
                };                                                  /* add */

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

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 update your system image and re-test.

```
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.
2. Open a serial terminal to the board, insert the SD card, boot the board, and log in as you did previously.
3. Using the voltmeter, observe that the system has automatically driven your pins to 3.3 V during startup. Also notice that your LWB's regulators now appear 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  3300mV  3300mV
 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  2625mV  3400mV
    20ca000.usbphy                                               3200mV  3200mV
    20c9000.usbphy                                               3200mV  3200mV
 regulator-dummy                  0    9      0     0mV     0mA     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 your pins, you can no longer manipulate them 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

Before you build your own drivers for use with the LWB, you must determine whether any conflicting drivers are built into 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 will cause a conflict.

```
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 must rebuild the kernel and exclude it in the process.

2. Exit your serial terminal so that you can return to running commands on the build host. Press **Ctrl+a** then **x**.
3. Select **Yes** when minicom asks if you want to leave.
4. 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
```

5. Exclude the cfg80211 module from the kernel by navigating the menu to **Networking Support > Wireless**, highlighting the *cfg80211 - wireless configuration API* item, and pressing **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 >       │
 └─────────────────────────────────────────────────────────────────────┘
```

6. Escape back up to the main menu.
7. Exclude wireless LAN drivers from the kernel: navigate the menu to **Device Drivers > Network device support**, highlight **Wireless LAN**, 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 >       │
 └───────────────────────────────────────────────────────────────────────┘
```

8. Escape back up to the main menu, then escape from the configuration menu.
9. When prompted, choose to save the new configuration.
10. Copy your new configuration file to where 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 detected that your kernel configuration is tainted, it rebuilds 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.
[...]
```

6.4 Writing the System Image to the SD Card

Write the 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$
```

6.5 Mounting the SD Card’s Root File System

You must copy drivers and firmware onto the target’s root file system, which is the second partition on the SD card. With the SD card in the build host, mount its second partition.

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

6.6 Building and Installing the Wi-Fi Drivers

6.6.1 Downloading and Extracting the Driver Backport Package

1. Download the driver backport source code package. Using your web browser, visit https://www.lairdtech.com/products/sterling-lwb
2. Select the Kits & Software tab.
3. Download *Sterling-LWB Software Version Backports (930-0075)* to **/home/user/930-0075.zip**.
4. Extract the driver backport source code.

```
user@buildhost$ cd ~
user@buildhost:~$ ls 930-0075.zip
930-0075.zip
user@buildhost:~$ unzip 930-0075.zip
Archive:  930-0075.zip
  inflating: backports-laird-3.5.4.2.tar.bz2
user@buildhost:~$ tar xf backports-laird-3.5.4.2.tar.bz2 -C projects/
user@buildhost:~$ cd ~/projects/laird-backport-3.5.4.2/
user@buildhost:~$ ls -l
[...]
drwxrwxr-x [...] backport-include
drwxrwxr-x [...] compat
-rw-rw-r-- [...] COPYING
drwxrwxr-x [...]
defconfigs drwxrwxr-x
[...] drivers drwxrwxr-x
[...] include drwxrwxr-x
[...] kconf
-rw-rw-r-- [...] Kconfig
-rw-rw-r-- [...] Kconfig.package.hacks
-rw-rw-r-- [...] Kconfig.sources
-rw-rw-r-- [...] MAINTAINERS
-rw-rw-r-- [...] Makefile
-rw-rw-r-- [...] Makefile.build
-rw-rw-r-- [...] Makefile.kernel
-rw-rw-r-- [...] Makefile.real
drwxrwxr-x [...] net
drwxrwxr-x [...] scripts
```

5. Configure the environment variables that enable the driver build process to use 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. Perform a sanity check of these environment variables.

```
user@buildhost$ "${CROSS_COMPILE}gcc" --version
arm-poky-linux-gnueabi-gcc (GCC) 5.2.0
Copyright (C) 2015 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
user@buildhost$ echo "${ARCH}"
arm
user@buildhost$ ls "${KLIB_BUILD}/.config"
/home/user/projects/fsl-release-bsp/build-imx6ul-fb/tmp/work/
  imx6ulevk-poky-linux-gnueabi/linux-imx/4.1.15-r0/build/.config
```

6.6.2 Configuring for FCC Compliance

Generate the configuration that governs the driver build process.

```
user@buildhost:~/projects/laird-backport-3.5.4.2$ $ make defconfig-lwb-fcc
make[2]: 'conf' is up to date.
#
# configuration written to .config
#

user@buildhost:~/projects/laird-backport-3.5.4.2$
```

6.6.3 Configuring for ETSI Compliance

Generate the configuration that governs the driver build process.

```
user@buildhost:~/projects/laird-backport-3.5.4.2$ $ make defconfig-lwb-etsi
make[2]: 'conf' is up to date.
#
# configuration written to .config
#

user@buildhost:~/projects/laird-backport-3.5.4.2$
```

6.6.4 Configuring for Giteki Compliance

Generate the configuration that governs the driver build process.

```
user@buildhost:~/projects/laird-backport-3.5.4.2$ $ make defconfig-lwb-jp
Generating local configuration database from kernel ... done.
make[2]: 'conf' is up to date.
#
# configuration written to .config
#

user@buildhost:~/projects/laird-backport-3.5.4.2$
```

6.6.5 Building and Installing

To build and install the Wi-Fi drivers, follow these steps:

1. Invoke *make* to build the drivers.

```
user@buildhost:~/projects/laird-backport-3.5.4.2$ make
make[5]: 'conf' is up to date.
[...]
  CC      /home/user/projects/laird-backport-3.5.4.2/compat/compat.mod.o
  LD [M]  /home/user/projects/laird-backport-3.5.4.2/compat/compat.ko
  CC      /home/user/projects/laird-backport-3.5.4.2/drivers/net/wireless/
             brcm80211/brcmfmac/brcmfmac.mod.o
  LD [M]  /home/user/projects/laird-backport-3.5.4.2/drivers/net/wireless/
             brcm80211/brcmfmac/brcmfmac.ko
  CC      /home/user/projects/laird-backport-3.5.4.2/drivers/net/wireless/
             brcm80211/brcmutil/brcmutil.mod.o
  LD [M]  /home/user/projects/laird-backport-3.5.4.2/drivers/net/wireless/
             brcm80211/brcmutil/brcmutil.ko
  CC      /home/user/projects/laird-backport-3.5.4.2/net/wireless/cfg80211.mod.o
  LD [M]  /home/user/projects/laird-backport-3.5.4.2/net/wireless/cfg80211.ko
user@buildhost:~/projects/laird-backport-3.5.4.2$
```

2. Observe that four kernel modules (**.ko* files) are built.
3. Copy these kernel modules to the target's root file system.
4. Verify the results.

```
user@buildhost:~/projects/laird-backport-3.5.4.2$ find . -name '*.ko'
./drivers/net/wireless/brcm80211/brcmfmac/brcmfmac.ko
./drivers/net/wireless/brcm80211/brcmutil/brcmutil.ko
./net/wireless/cfg80211.ko
./compat/compat.ko
user@buildhost:~/projects/laird-backport-3.5.4.2$ find . -name '*.ko' | \
  sudo xargs cp --parents -t /mnt/lib/modules/4.1.15-1.2.0+g77f6154/kernel/
user@buildhost:~/projects/laird-backport-3.5.4.2$ find \
  /mnt/lib/modules/4.1.15-1.2.0+g77f6154/ -name 'compat.ko' -o \
  -name 'brcmfmac.ko' -o -name 'brcmutil.ko' -o -name 'cfg80211.ko'
/mnt/lib/modules/4.1.15-1.2.0+g77f6154/kernel/compat/compat.ko
/mnt/lib/modules/4.1.15-1.2.0+g77f6154/kernel/drivers/net/wireless/brcm80211/brcmutil/
  brcmutil.ko
/mnt/lib/modules/4.1.15-1.2.0+g77f6154/kernel/drivers/net/wireless/brcm80211/brcmfmac/
  brcmfmac.ko
/mnt/lib/modules/4.1.15-1.2.0+g77f6154/kernel/net/wireless/cfg80211.ko
```

6.7 Building and Installing the Bluetooth Drivers

To build and install the Bluetooth drivers, follow these steps:

1. You need to execute the *brcm_patchram_plus* utility on our target board at runtime to load firmware into the volatile memory of the LWB device's Bluetooth subsystem.
2. Download the *brcm_patchram_plus* package. Using your web browser, visit https://github.com/LairdCP/brcm_patchram/releases
3. Download *brcm_patchram_plus_1.1* in tar.gz format.
4. Save it as **/home/user/brcm_patchram-brcm_patchram_plus_1.1.tar.gz**.
5. Extract the tarball.

```
user@buildhost$ cd ~
user@buildhost:~$ ls brcm_patchram-brcm_patchram_plus_1.1.tar.gz
brcm_patchram-brcm_patchram_plus_1.1.tar.gz
user@buildhost:~$ tar xf brcm_patchram-brcm_patchram_plus_1.1.tar.gz -C projects/
user@buildhost:~$ cd ~/projects/brcm_patchram-brcm_patchram_plus_1.1/
user@buildhost:~/projects/brcm_patchram-brcm_patchram_plus_1.1$ ls -l
[...]
-rw-rw-r-- [...] brcm_patchram_plus.1
-rw-rw-r-- [...] brcm_patchram_plus.c
-rw-rw-r-- [...] brcm_patchram_plus_h5.c
-rw-rw-r-- [...] brcm_patchram_plus_usb.c
-rw-rw-r-- [...] LICENSE
-rw-rw-r-- [...] Makefile
-rw-rw-r-- [...] README.md
```

6. Open a shell with an environment configured for building conventional Makefile projects with the cross-development toolchain.

```
user@buildhost$ cd ~/projects/fsl-release-bsp/build-imx6ul-fb/
user@buildhost:~/projects/fsl-release-bsp/build-imx6ul-fb$ bitbake \
  -c devshell core-image-base
```

   A new terminal window appears.

7. Modify its prompt to explicitly mark its terminal as the devshell.

```
root@buildhost:~/projects/fsl-release-bsp/build-imx6ul-fb/tmp/work/
imx6ulevk-poky-linux-gnueabi/core-image-base/1.0-r0/core-image-base-1.0# export \
  PS1="[devshell] $PS1"
[devshell] root@buildhost:~/projects/fsl-release-bsp/build-imx6ul-fb/tmp/work/
imx6ulevk-poky-linux-gnueabi/core-image-base/1.0-r0/core-image-base-1.0#
```

8. Within the devshell, change to the *brcm_patchram_plus* directory.

```
[devshell] root@buildhost:~/projects/fsl-release-bsp/build-imx6ul-fb/tmp/work/
imx6ulevk-poky-linux-gnueabi/core-image-base/1.0-r0/core-image-base-1.0# cd \
  ~/projects/brcm_patchram-brcm_patchram_plus_1.1/
[devshell] root@buildhost:~/projects/brcm_patchram-brcm_patchram_plus_1.1# ls -l
[...]
-rw-rw-r-- [...] LICENSE
-rw-rw-r-- [...] Makefile
-rw-rw-r-- [...] README.md
-rw-rw-r-- [...] brcm_patchram_plus.1
-rw-rw-r-- [...] brcm_patchram_plus.c
-rw-rw-r-- [...] brcm_patchram_plus_h5.c
-rw-rw-r-- [...] brcm_patchram_plus_usb.c
```

9. Invoke *make* to perform the build.
10. Once the executable *brcm_patchram_plus* is produced, exit the devshell.

```
[devshell] root@buildhost:~/projects/brcm_patchram-brcm_patchram_plus_1.1# make
[...]
[devshell] root@buildhost:~/projects/brcm_patchram-brcm_patchram_plus_1.1# ls -l
-rw-rw-r-- [...] LICENSE
-rw-rw-r-- [...] Makefile
-rw-rw-r-- [...] README.md
-rwxr-xr-x [...] brcm_patchram_plus
-rw-rw-r-- [...] brcm_patchram_plus.1
-rw-rw-r-- [...] brcm_patchram_plus.c
-rw-rw-r-- [...] brcm_patchram_plus.o
-rwxr-xr-x [...] brcm_patchram_plus_h5
-rw-rw-r-- [...] brcm_patchram_plus_h5.c
-rw-r--r-- [...] brcm_patchram_plus_h5.o
-rwxr-xr-x [...] brcm_patchram_plus_usb
-rw-rw-r-- [...] brcm_patchram_plus_usb.c
-rw-r--r-- [...] brcm_patchram_plus_usb.o
[devshell] root@buildhost:~/projects/brcm_patchram-brcm_patchram_plus_1.1# exit
```

11. Within the original terminal window, copy the *brcm_patchram_plus* executable to the target's root file system and verify the result.

```
user@buildhost$ 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
user@buildhost:~/projects/brcm_patchram-brcm_patchram_plus_1.1$ ls -l \
  /mnt/usr/bin/brcm_patchram_plus
-rwxr-xr-x [...] /mnt/usr/bin/brcm_patchram_plus
```

6.8 Installing the Firmware

6.8.1 FCC Compliance

For FCC compliance, follow these steps:

1. Download the firmware package. Using your web browser, visit https://www.lairdtech.com/products/sterling-lwb
2. Select the Kits & Software tab.
3. Download *Sterling-LWB Firmware Package FCC (USA/Canada) (480-0079)* to **/home/user/480-0079.zip**.
4. Extract the firmware.

```
user@buildhost$ cd ~
user@buildhost:~$ ls 480-0079.zip
480-0079.zip
user@buildhost:~$ unzip 480-0079*.zip
Archive:  480-0079.zip
  inflating: 480-0079.tar.bz2
user@buildhost:~$ tar xjf 480-0079.tar.bz2
user@buildhost:~$ cd lib/firmware/brcm/
user@buildhost:~/lib/firmware/brcm$ ls -l
total 4
lrwxrwxrwx [...] 4343w.hcd -> ./bcm4343w/4343w.hcd
drwxr-xr-x [...] bcm4343w
lrwxrwxrwx [...] brcmfmac43430-sdio.bin -> ./bcm4343w/brcmfmac43430-sdio.bin
lrwxrwxrwx [...] brcmfmac43430-sdio.clm_blob -> ./bcm4343w/brcmfmac43430-sdio.clm_blob
lrwxrwxrwx [...] brcmfmac43430-sdio.txt -> ./bcm4343w/brcmfmac43430-sdio.txt
```

5. Create the destination directory on the target's root file system.
6. Copy the firmware files into that directory.
7. Verify the result.

```
user@buildhost:~/lib/firmware/brcm$ sudo mkdir /lib/firmware/brcm
user@buildhost:~/lib/firmware/brcm$ sudo cp -ra * /lib/firmware/brcm/
user@buildhost:~/lib/firmware/brcm$ ls -l /lib/firmware/brcm/total
total 4
lrwxrwxrwx [...] 4343w.hcd -> ./bcm4343w/4343w.hcd
drwxr-xr-x [...] bcm4343w
lrwxrwxrwx [...] brcmfmac43430-sdio.bin -> ./bcm4343w/brcmfmac43430-sdio.bin
lrwxrwxrwx [...] brcmfmac43430-sdio.clm_blob -> ./bcm4343w/brcmfmac43430-sdio.clm_blob
lrwxrwxrwx [...] brcmfmac43430-sdio.txt -> ./bcm4343w/brcmfmac43430-sdio.txt
```

6.8.2 ETSI Compliance

For ETSI compliance, follow these steps:

1. Download the firmware package. Using your web browser, visit https://www.lairdtech.com/products/sterling-lwb
2. Select the Kits & Software tab.
3. Download *Sterling-LWB Firmware Package ETSI / RCM (480-0080)* to **/home/user/480-0080.zip**.

4. Extract the firmware.

```
user@buildhost$ cd ~
user@buildhost:~$ ls 480-0080.zip
480-0080.zip
user@buildhost:~$ unzip 480-0080*.zip
Archive:  480-0080.zip
  inflating: 480-0080.tar.bz2
user@buildhost:~$ tar xjf 480-0080.tar.bz2
user@buildhost:~$ cd lib/firmware/brcm/
user@buildhost:~/lib/firmware/brcm$ ls -l
total 4
lrwxrwxrwx [...] 4343w.hcd -> ./bcm4343w/4343w.hcd
drwxr-xr-x [...] bcm4343w
lrwxrwxrwx [...] brcmfmac43430-sdio.bin -> ./bcm4343w/brcmfmac43430-sdio.bin
lrwxrwxrwx [...] brcmfmac43430-sdio.clm_blob -> ./bcm4343w/brcmfmac43430-sdio.clm_blob
lrwxrwxrwx [...] brcmfmac43430-sdio.txt -> ./bcm4343w/brcmfmac43430-sdio.txt
```

5. Create the destination directory on the target’s root file system.
6. Copy the firmware files into that directory.
7. Verify the result.

```
user@buildhost:~/lib/firmware/brcm$ sudo mkdir /lib/firmware/brcm
user@buildhost:~/lib/firmware/brcm$ sudo cp -ra * /lib/firmware/brcm/
user@buildhost:~/lib/firmware/brcm$ ls -l /lib/firmware/brcm/total
total 4
lrwxrwxrwx [...] 4343w.hcd -> ./bcm4343w/4343w.hcd
drwxr-xr-x [...] bcm4343w
lrwxrwxrwx [...] brcmfmac43430-sdio.bin -> ./bcm4343w/brcmfmac43430-sdio.bin
lrwxrwxrwx [...] brcmfmac43430-sdio.clm_blob -> ./bcm4343w/brcmfmac43430-sdio.clm_blob
lrwxrwxrwx [...] brcmfmac43430-sdio.txt -> ./bcm4343w/brcmfmac43430-sdio.txt
```

6.8.3 Giteki Compliance

For Giteki compliance, follow these steps:

1. Download the firmware package. Using your web browser, visit https://www.lairdtech.com/products/sterling-lwb
2. Select the Kits & Software tab.
3. Download *Sterling-LWB Firmware Package Giteki (Japan) (480-0116)* to **/home/user/480-0116.zip**.
4. Extract the firmware.

```
user@buildhost$ cd ~
user@buildhost:~$ ls 480-0116*.zip
480-0116-5.0.0.28.zip
user@buildhost:~$ unzip 480-0116*.zip
Archive:  480-0116-5.0.0.28.zip
  inflating: 480-0116.tar.bz2
user@buildhost:~$ tar xjf 480-0116.tar.bz2
user@buildhost:~$ cd lib/firmware/brcm/
user@buildhost:~/lib/firmware/brcm$ ls -l
total 4
lrwxrwxrwx [...] 4343w.hcd -> ./bcm4343w/4343w.hcd
drwxr-xr-x [...] bcm4343w
lrwxrwxrwx [...] brcmfmac43430-sdio.bin -> ./bcm4343w/brcmfmac43430-sdio.bin
lrwxrwxrwx [...] brcmfmac43430-sdio.clm_blob -> ./bcm4343w/brcmfmac43430-sdio.clm_blob
lrwxrwxrwx [...] brcmfmac43430-sdio.txt -> ./bcm4343w/brcmfmac43430-sdio.txt
```

5. Create the destination directory on the target's root file system.
6. Copy the firmware files into that directory.
7. Verify the result.

```
user@buildhost:~/lib/firmware/brcm$ sudo mkdir /lib/firmware/brcm
user@buildhost:~/lib/firmware/brcm$ sudo cp -ra * /lib/firmware/brcm/
user@buildhost:~/lib/firmware/brcm$ ls -l /lib/firmware/brcm/total
total 4
lrwxrwxrwx [...] 4343w.hcd -> ./bcm4343w/4343w.hcd
drwxr-xr-x [...] bcm4343w
lrwxrwxrwx [...] brcmfmac43430-sdio.bin -> ./bcm4343w/brcmfmac43430-sdio.bin
lrwxrwxrwx [...] brcmfmac43430-sdio.clm_blob -> ./bcm4343w/brcmfmac43430-sdio.clm_blob
lrwxrwxrwx [...] brcmfmac43430-sdio.txt -> ./bcm4343w/brcmfmac43430-sdio.txt
```

6.9 Unmounting the SD Card's Root File System

Now that you are done copying files to the target's root file system, unmount the SD card.

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

Physically eject it from the build host.

7 Installing the LWB Hardware

To install the LWB hardware, follow these steps:

1. Power down the board.
2. Ensure that the LWB's jumpers are configured as follows:
   - **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 to LWB	J1704-P10	J3-P11
BT_REG_ON	Host to LWB	J1704-P9	J3-P13
Bluetooth UART:	-	-	-
Tx	Host to LWB	J1703-P2	J3-P10
Rx	LWB to Host	J1703-P1	J3-P8
RTS	Host to LWB	J1703-P4	J3-P6
CTS	LWB to Host	J1703-P3	J3-P14

   The UART signal names in the table above are assigned per RS-232 conventions, treating the target board as our Data Terminal Equipment (DTE) and the LWB as our Data Circuit-Terminating Equipment (DCE).

   Be aware that documentation, schematics, code, and configuration files for the target board and the LWB do not consistently adhere to this convention.

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

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'
```

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

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

5. Observe that the wireless driver modules have not been loaded into the kernel, as indicated by the following empty result:

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

6. 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 v4.3-15586-g7eee5e4
Backport generated by backports.git f2cb36e

root@imx6ulevk:~# insmod /lib/modules/4.1.15-1.2.0+g77f6154/kernel/net/wireless/cfg80211.ko
cfg80211: World regulatory domain updated:
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, 3000 mBm), (N/A)
cfg80211:   (5170000 KHz - 5250000 KHz @ 80000 KHz, 160000 KHz AUTO), (N/A, 2300 mBm), (N/A)
cfg80211:   (5250000 KHz - 5330000 KHz @ 80000 KHz, 160000 KHz AUTO), (N/A, 2300 mBm), (0 s)
cfg80211:   (5490000 KHz - 5730000 KHz @ 160000 KHz), (N/A, 2300 mBm), (0 s)
cfg80211:   (5735000 KHz - 5835000 KHz @ 80000 KHz), (N/A, 3000 mBm), (N/A)
cfg80211:   (57240000 KHz - 63720000 KHz @ 2160000 KHz), (N/A, 4000 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
brcmfmac: brcmf_c_preinit_dcmds: Firmware version = wl0: Nov 14 2016 00:32:49
  version 7.45.41.32 (r662793) FWID 01-8b473c3f
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

Your new interface is not yet 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 act of connecting to networks and its settings are contained in the /etc/wpa_supplicant.conf file. Edit this file.

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

2. Press **i** to switch vi from command mode to edit mode.
3. 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
}
```

Note: The network.ssid property specifies your network's name (its SSID) to be *ccopen*.

4. Press **Escape** to exit edit mode and to re-enter command mode.
5. Type *:wq* to write your changes to disk, then quit.
6. Verify that your wireless router is on and ready. Its SSID should be set to *ccopen* and its DHCP server should be enabled.
7. Start wpa_supplicant. Temporarily run it in the foreground so that you can easily see 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
IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready
wlan0: Trying to associate with SSID 'ccopen'
IPv6: ADDRCONF(NETDEV_CHANGE): wlan0: link becomes ready
wlan0: Associated with 64:a0:e7:dd:2e:ad
wlan0: CTRL-EVENT-CONNECTED - Connection to 64:a0:e7:dd:2e:ad completed [id=0 id_str=]
```

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

8. 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 is no longer reported as *Not connected*. It now shows that wlan0 is connected to the *ccopen* network.

```
root@imx6ulevk:~# iw wlan0 link
Connected to 64:a0:e7:dd:2e:ad (on wlan0)
        SSID: ccopen
        freq: 2462
        RX: 218 bytes (2 packets)
        TX: 642 bytes (7 packets)
        signal: -38 dBm
        tx bitrate: 24.0 MBit/s

        bss flags:      short-preamble short-slot-time
        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,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:25:ca:07:0c:d5 brd ff:ff:ff:ff:ff:ff
    inet6 2001:1890:1230:f42e:225:caff:fe07:cd5/64 scope global dynamic
       valid_lft 2591992sec preferred_lft 604792sec
    inet6 fe80::225:caff:fe07:cd5/64 scope link
       valid_lft forever preferred_lft forever
root@imx6ulevk:~# ip route show
root@imx6ulevk:~#
```

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 10.1.44.173...
Lease of 10.1.44.173 obtained, lease time 28800
/etc/udhcpc.d/50default: Adding DNS 10.1.44.35
/etc/udhcpc.d/50default: Adding DNS 10.1.44.32
/etc/udhcpc.d/50default: Adding DNS 8.8.8.8
```

Your device now has an IP address (in our example, it's 10.1.44.173).

```
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:07:0c:d5 brd ff:ff:ff:ff:ff:ff
    inet 10.1.44.173/24 brd 10.1.44.255 scope global wlan0
       valid_lft forever preferred_lft forever
    inet6 2001:1890:1230:f42e:225:caff:fe07:cd5/64 scope global dynamic
       valid_lft 2591946sec preferred_lft 604746sec
```

You now have a default route through your router (in our example, the address is 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
```

2. Pass some IP traffic over your Wi-Fi link by pinging your 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=355.885 ms
64 bytes from 10.1.44.1: seq=1 ttl=255 time=111.497 ms
64 bytes from 10.1.44.1: seq=2 ttl=255 time=157.660 ms

--- 10.1.44.1 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 111.497/208.347/355.885 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

Although you manually loaded your driver modules with *insmod*, they do not automatically reload after a reboot. To have your modules loaded during system initialization, tell the module dependency database to search for and add all new modules.

```
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.

```
root@imx6ulevk:~# reboot

[...]

imx6ulevk login: root

root@imx6ulevk:~# lsmod | grep -e compat -e cfg80211 -e brcmutil -e brcmfmac
brcmfmac              184694  0
cfg80211              217202  1 brcmfmac
compat                  4526  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:07:0c:d5 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. This requires that the WL_REG_ON signal be asserted during boot.

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 0 bytes and received 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
```

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/4343w-07_25_2016.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
```

Note: 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 has been transmitting and receiving bytes. Also notice that the 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
```

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 nearby visible Bluetooth devices. Ensure that you have another Bluetooth device nearby, powered on, and configured to be visible to other devices, then 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 your 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

Congratulations on successfully integrating your LWB. How you proceed from this point depends upon your specific application.

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
- Refining power management

Potential next steps with Bluetooth:

- Bringing up your interface automatically 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 (USA/Canada) (FCC)
480-0079.zip
https://www.lairdtech.com/products/sterling-lwb
Select the Kits & Software tab.

Laird Sterling-LWB Firmware (ETSI / RCM) (ETSI)
480-0080.zip

https://www.lairdtech.com/products/sterling-lwb
Select the Kits & Software tab.

Laird Sterling-LWB Firmware (Japan) (Giteki)
480-0116.zip
https://www.lairdtech.com/products/sterling-lwb
Select the Kits & Software tab.

Laird Sterling-LWB Driver Backports
930-0075.zip
https://www.lairdtech.com/products/sterling-lwb
Select the Kits & Software tab.

Laird Sterling-LWB Data Sheet
330-0190.pdf
https://www.lairdtech.com/products/sterling-lwb
Select the 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.MX6 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=product

NXP i.MX6 UltraLite 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

12.1.1 ”apt”: Resource Temporarily Unavailable

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?
```

Remarks: 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.

12.1.2 ”apt install”: Abort

Problem: An *apt-get install* command aborts with the following error message after user confirms 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.
```

Remarks: Cause uncertain, though the command succeeds when retried.

12.1.3 ”bitbake”: Host Distribution Not Validated

Problem: Receive warning message when running bitbake on 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."
```

Remarks: Also received this warning, but have observed no ill effects.

12.1.4 ”bitbake”: Failed to Fetch

Problem: bitbake reports:

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

Remarks: bitbake is trying to retrieve source code from a repository server. The server is temporarily or permanently unreachable, so 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 prior to exiting:

```
\textbf{Note:} Tasks Summary: Attempted [...] tasks of which [...] didn't need to be rerun
and all succeeded.
```

12.2 Target Board

12.2.1 ”uboot”: Bad CRC

Problem: Target board’s U-Boot bootloader displays 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
```

Remarks: This is the intended behavior. U-Boot is telling us 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.

12.2.2 ”random: nonblocking pool initialized”

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

Remark: This appears to be normal for the board.

12.2.3 ”ip link”: No Wireless Interface Listed

Problem: No wireless network interface listed by ”ip link”

Common cause: Drivers did not load successfully.

12.2.4 ”brcmfmac”: Firmware Load Failed

Problem: While loading brcmfmac.ko driver, receive 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.

12.2.5 ”brcmfmac”: Normal Output Messages Missing

Problem: While loading 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 pmurev 24$
brcmfmac: brcmf_c_preinit_dcmds: Firmware version = wl0: Nov 14 2016 00:32:49$
  version 7.45.41.32 (r662793) FWID 01-8b473c3f$
brcmfmac: brcmf_cfg80211_reg_notifier: not a ISO3166 code
```

Common cause: Wi-Fi hardware module 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

12.2.6 "brcmfmac": "No SDIO Drive strength init"

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

Common cause: This is not a problem and is indicative of normal operation. If you do not see this message, then there is a problem.

12.2.7 "brcm_patchram_plus": Same Data Written Repeatedly

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

Common cause: 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 (FCC)
  - /lib/firmware/brcm/4343w.hcd -> ./bcm4343w/4343w.hcd
  - /lib/firmware/brcm/bcm4343w/4343w.hcd
  - /lib/firmware/brcm/bcm4343w/brcmfmac43430-sdio.bin -> ./brcmfmac43430-sdio-prod.bin
  - /lib/firmware/brcm/bcm4343w/brcmfmac43430-sdio.clm_blob
  - /lib/firmware/brcm/bcm4343w/brcmfmac43430-sdio-fcc.txt
  - /lib/firmware/brcm/bcm4343w/brcmfmac43430-sdio-prod.bin
  - /lib/firmware/brcm/bcm4343w/brcmfmac43430-sdio.txt -> ./brcmfmac43430-sdio-fcc.txt
  - /lib/firmware/brcm/brcmfmac43430-sdio.bin -> ./bcm4343w/brcmfmac43430-sdio.bin
  - /lib/firmware/brcm/brcmfmac43430-sdio.clm_blob -> ./bcm4343w/brcmfmac43430-sdio.clm_blob
  - /lib/firmware/brcm/brcmfmac43430-sdio.txt -> ./bcm4343w/brcmfmac43430-sdio.txt

- Firmware Files and Symlinks (ETSI)
  - /lib/firmware/brcm/4343w.hcd -> ./bcm4343w/4343w.hcd
  - /lib/firmware/brcm/bcm4343w/4343w.hcd
  - /lib/firmware/brcm/bcm4343w/brcmfmac43430-sdio.bin -> ./brcmfmac43430-sdio-prod.bin
  - /lib/firmware/brcm/bcm4343w/brcmfmac43430-sdio.clm_blob
  - /lib/firmware/brcm/bcm4343w/brcmfmac43430-sdio-etsi.txt
  - /lib/firmware/brcm/bcm4343w/brcmfmac43430-sdio-prod.bin
  - /lib/firmware/brcm/bcm4343w/brcmfmac43430-sdio.txt -> ./brcmfmac43430-sdio-etsi.txt
  - /lib/firmware/brcm/brcmfmac43430-sdio.bin -> ./bcm4343w/brcmfmac43430-sdio.bin
  - /lib/firmware/brcm/brcmfmac43430-sdio.clm_blob -> ./bcm4343w/brcmfmac43430-sdio.clm_blob
  - /lib/firmware/brcm/brcmfmac43430-sdio.txt -> ./bcm4343w/brcmfmac43430-sdio.txt
- Firmware Files and Symlinks (Giteki)
  - /lib/firmware/brcm/4343w.hcd -> ./bcm4343w/4343w.hcd
  - /lib/firmware/brcm/bcm4343w/4343w.hcd
  - /lib/firmware/brcm/bcm4343w/brcmfmac43430-sdio.bin -> ./brcmfmac43430-sdio-prod.bin
  - /lib/firmware/brcm/bcm4343w/brcmfmac43430-sdio.clm_blob
  - /lib/firmware/brcm/bcm4343w/brcmfmac43430-sdio-jp.txt
  - /lib/firmware/brcm/bcm4343w/brcmfmac43430-sdio-prod.bin
  - /lib/firmware/brcm/bcm4343w/brcmfmac43430-sdio.txt -> ./brcmfmac43430-sdio-jp.txt
  - /lib/firmware/brcm/brcmfmac43430-sdio.bin -> ./bcm4343w/brcmfmac43430-sdio.bin
  - /lib/firmware/brcm/brcmfmac43430-sdio.clm_blob -> ./bcm4343w/brcmfmac43430-sdio.clm_blob
  - /lib/firmware/brcm/brcmfmac43430-sdio.txt -> ./bcm4343w/brcmfmac43430-sdio.txt
- Bluetooth serial terminal device: /dev/ttymxc1