2nd Generation Touch Sensor Family Integration Guide

Linux Device Driver

Fingerprint Area Sensor System in Mobile Devices

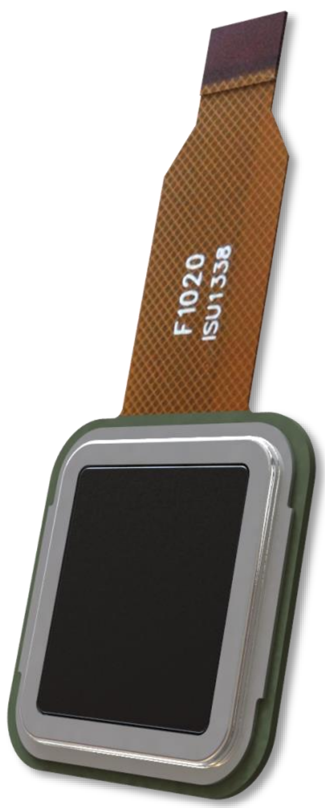

Table of Contents

1. Document Control

1.1. Revision History

Rev	Date	Changes	Author	Approval
A	2014-01-19	New document / DRY + PJ Edits	MM	
B	2014-03-20	Minor update	ZW	
C	2014-05-20	Updated for Touch Sensor Family	MM	
D	2014-06-13	Included driver diagnostics and bring-up tools. Added troubleshooting section. Reviewed, modified and clarified for fpc1150.	PJE/TES	
E	2014-06-30	Minor updated in naming (2nd generation of Touch Sensors)	TES	

1.2. References

The following publications contain material relevant to the FPC1020 integration efforts. These publications are available from Fingerprint Cards AB upon request.

Ref #	Title	Revision
1	*3/750-FPC77XX-AN TSF Integration Guide Android system*	Rev D
2	*330-FPC77XXAN_SAO_Android*	Rev A

1.3. Attributions

Term	Attribution
Android	Google, Inc.
Nexus (phone)	Google, Inc.
Dragonboard APQ8074	Qualcomm, Inc.
Dragonboard APQ8074	Intrinsyc

2. Overview

This document explains the details of the FPC 2nd generation Touch Sensor Family (FPC1020 family) device driver(s) integration into the Linux kernel of an Android platform. As of the time of this document writing, the TSF family of fingerprint devices includes: FPC1020, FPC1021 and FPC1150. These sensors are functionally compatible, with differences in pixel count and layout, for image capture operations.

Note that the sensor is frequently referred to as “fpc1020” in documentation and code. Unless otherwise stated “fpc1020” should be read as “fpc1020 family”

This document is a part of the document suite for the FPC TSF Integration Guide for Android also see [1].The information contained here is intended for a software engineer with experience in Linux kernel device driver modifications and building, access to the kernel sources and make, and the platform Android.

Fingerprint Cards Linux device driver reference design is provided in the context of the Qualcomm Dragonboard APQ8074 development system for mobile platforms. This development system is produced by Intrinsyc www.intrinsyc.com. It is assumed that many mobile devices (smartphones, tablets) are based on this or similar platforms supplied by Qualcomm (see Prerequisites). Other Android/Linux platforms would be similar, though possibly not identical.

There are at least two options of building a Linux kernel driver: as an installable kernel object, or built into the kernel. Each option has its advantages:

- Built into the kernel – provides better software integrity for the end product.
- Installable kernel object – allows iterations and modifications.

3. Prerequisites

3.1. Dragonboard APQ8074 Requirements

The Dragonboard (or the target platform) is operational and is flashed with the appropriate Linux and Android software. These software components are not provided by FPC and they are fixed by the reference board supplier. Intrinsyc APQ8074 BSP1.1 is used.

Parameter	Rating
Android O/S Version	Jelly Bean 4.2.2
Linux Version	3.4.0

3.2. Hardware Integration Completed

It is assumed that the FPC1020 fingerprint sensor has been attached to the hardware platform

3.3. Linux Kernel Sources

Obtain Linux kernel sources for the platform. Make sure that you can build kernel for the platform.
Note: Intrinsyc provides a script, "download-and-build", that results in the download of all required sources for the Dragonboard. The kernel source is automatically downloaded as part of the Intrinsyc BSP source.

4. Device Driver Sources and Target

4.1. Download Sources

Obtain the source files for the device driver from FPC in the supplied package.
Note: the exact names of the source files are subject to modification.

4.2. General Board/Kernel Preparations

The kernel must be prepared to accept the fpc1020 device drive. The 8074 Dragonboard kernel 3.4.0 uses Devicetree/Open firmware for driver configuration. Several modifications to files in `./kernel/arch/arm/boot` and `./kernel/arch/arm/mach-msm` are required. The details are not described in this document.

Modifications required include:

- Device tree definitions for the SPI controller and fpc1020 driver.
- Modifications to the "board" file.
- Clock configuration for the SPI driver
- I/O pin setup
- Power supply configuration

Target the driver to the platform HW choices:

- SPI port
- Interrupt GPIO
- Reset GPIO
- Power supply considerations

4.3. FPC TSF Driver Built Into the Kernel

In order to make a TSF device driver part of the kernel it must be built from within the kernel source tree. This section lists the required steps for the TSF linux driver integration into the kernel.

- Copy source files into the platform Linux source tree, e.g. `linux/drivers/input/misc`
- Add `CONFIG_INPUT_FPC1020` to the kernel menu configuration.
- Add `FPC1020` to the `Makefile`.
- Configure the kernel to include `fpc1020`.
- Make the new kernel and/or Android image

4.4. Driver as a Kernel Module

In order to build a TSF device driver as a loadable module it may either be built from within the kernel source tree or externally by referring to the kernel source. Note that the kernel sources are required even though the kernel itself is not rebuilt.

In order to build from within the source tree, follow the steps for built-in driver as per above but select "module" rather than "yes" for the `fpc1020` driver option.

Optionally the driver module may be built outside the kernel tree. In that case the source tree must still be referred to from the `Makefile`.

In both cases a module, `fpc1020.ko`, should be created. The module is unique for a specified platform and kernel. It is not possible to load a module built for another kernel.

4.5. Driver compatibility

As stated above, the touch sensor family includes several different devices sharing common functionality. These sensors share a common driver. Hence, the "fpc1020" driver supports the sensors included within the TSF range. Some users may find it confusing to refer to "1020" even if the actual sensor is another device.

5. Linux Device Driver Installation

If the TSF driver (fpc1020) was properly built, then the device driver can be installed by:

```
insmod fpc1020.ko
```

Check that the driver is loaded by:

```
lsmod
```

If the driver was built into the kernel it will load automatically when the target board is rebooted. In either case there will be diagnostic output in `/proc/kmesg`.

Once loaded, and successfully initialized, the driver will create a character device:

```
/dev/fpc1020
```

and add sysFS entries in

```
/sys/bus/spi/devices/[spi x.y]/
```

6. Driver post-configuration

6.1. Filesystem preparations

It is essential to set the file mode (permissions) for the device and settings such that the device and settings can be read and written e.g.:

```
chmod 777 /dev/fpc1020
chmod 777 /sys/bus/spi/devices/spi2.2/setup
chmod 777 /sys/bus/spi/devices/spi2.2/diag
```

Add the device "/dev/fpc1020" to udevd (e.g. in ueventd.qcom.rc for Dragonboard) to make file mode settings persistent.

6.2. Driver configuration

The TSF driver itself requires no configuration. It automatically configures sensor parameters based on recommended settings for the device detected. The driver will not load if it cannot match a detected sensor to its list of known configurations.

6.3. Image capture parameters

Some parameters, e.g. ADC settings, must be configured individually for each product. These settings must be tuned to match physical characteristics of the sensor packaging, e.g. coating thickness or mechanical constraints.

The "Hardware Abstraction Layer" (HAL) can update driver parameters defined by the HAL configuration file (typically /etc/fpc1020.conf). HAL parameters override the default driver settings. HAL operation is not covered in this document.

6.4. Android input device mapping

Some driver functions, e.g. "wake-up" requires additional Android platform integration to work.

The TSF driver issues linux "input" events that are mapped to Android events by means of an input configuration file (fpc1020.idc) and a keyboard layout map (fpc1020.kl). These files must be deployed to the target filesystem. Sample files are distributed with the driver. The Android input system is not covered in this document.

7. Driver diagnostics and trouble-shooting

7.1. Driver loaded OK

The driver exports its functions using the Linux filesystem. Hence, several operations may be performed manually by reading (and writing) the files provided by the driver. Note that these operations do not require Android. Linux terminal access, as super user, is sufficient for the basic sensor test operations. However, no biometrical operations are available in this context. Sensor operations are limited to raw image capture and diagnostics.

These include:
The device file (`/dev/fpc1020`), the configuration directory (`[spi_dir]/setup`), and diagnostics directory (`[spi_dir]/diag`).

In order to assist the initial platform bring-up a simple script based tool is included with the source distribution (./tools/linux). A sample screenshot is shown below.

```
Target
File  Edit  View  Search  Terminal  Help

*******************************
*** Fpc1020 Diagnostic tool ***
*******************************
(l)oad module
(u)nload module

1- Check chip id
2- Dump settings

3- Sensor selftest

4- Capture, wait-and-capture
5- Capture, single image
6- Capture, single image multicapture
7- Capture, checkerboard test

(r)ead sensor register

(q)uit

Enter selection:
```

Further instructions are provided in the README file included with the tool.

Images captured will be stored in the platform directory selected. Multiple captures can be stored. In order to visualize an image it must first be transferred to the host computer and then the raw byte stream must be converted to an image. Each raw image file contains sequential pixel data, one byte per pixel, line by line.

There are several ways to do this: A simple solution is to use GIMP (www.gimp.org) and import the saved image as raw data. The image geometry, as set by the sensor crop settings, must be entered manually (e.g. 192x192 pixels for the un-cropped fpc1020 sensor).

7.2. Troubleshooting, driver not loading.

So, the driver did not load properly? There several reasons why the driver may fail. These include (in order, but not limited to):

- All hardware connections. Check all wires for proper connections, no short circuits etc.
- Check all power supplies. Note that some platforms (e.g. the Dragonboard) do not power certain supplies unless there is a device registered to use it.
- I/O configurations: all GPIOs and the SPI bus signals must be properly configured in the kernel.
- Kernel preparations, i.e. is the SPI bus setup ready to connect to the TSF driver? The SPI bus driver registers expected devices under /sys/bus/spi/devices/…
- TSF driver probe() function called? The driver issues helpful debugging diagnostics to the kernel log. However, probe() is not called unless the kernel can match a driver to an expected SPI device.
- Probe called but fails? Check the log output carefully. For example, probe() may fail if requested memory is not available or if the SPI bus is not working properly.
- FPC Device mismatch? Verify that suggested HW-identification matches a device supported by the driver.

Generally the kernel log output is a good information source for sensor diagnostics.

7.3. Troubleshooting, image capture

The driver did load, but image capture fails? Possible causes include (but not limited to):

- Check file modes (as described above)
- Check user privileges, must match the file modes

Check and adjust the ADC parameters, found in directory (`[spi_dir]/setup`), if:

- Finger detection not working."Finger up" or "finger down" conditions not recognized.
- Images are weak or too light.
- Images are too dark.

8. Contact Information

Fingerprint Cards AB offers world-wide technical support for Android integration efforts.

To obtain information about the local support, please contact Fingerprint Cards AB.

Fingerprint Cards AB (FPC)
P.O. Box 2412
Kungsportsplatsen 2
SE-411 10 Gothenburg
Sweden

Phone: +46 (0)31 60 78 20
Web: www.fingerprints.com/contact/