TP Driver Porting Guide for Qualcomm

TP Driver Porting Guide for Qualcomm	
Project name	Touch panel
Document ref	[Document ref]
Version	3.4
Release date	2021.12.09
Owner	Driver Team, FocalTech
Classification	
Distribution List	
Approval	

22th Floor,Block B,Building 9,Zone II of Shenzhen Bay Eco-Technology Park,
Yuehai Avenue,Nanshan District,Shenzhen,
Guangdong Province, P.R. China

ZIP: 518052
T +86 755 26588222
F +86 755 26712499
E support@focaltech-electronics.com

www.focaltech-systems.com

Revision History

Date	Version	List of changes	Author	Approved by
2021.12.09	3.4	1. Modify app.bin to xxx.bin while using fts_upgrade_bin node. 2. Add description of FTS_PEN_EN.	luoguojin	
2020.12.29	3.3	1. Add more IC support. 2. Modify description in fts_rw_reg node. 3. Modify resolutions in DTS sample.	luoguojin	
2020.04.18	3.2	1. Add more IC support. 2. Suggest disable FTS_AUTO_LIC_UPGRADE_EN for mass production.	luoguojin	
2019.07.31	3.1	1. More IC Support 2. Add SPI DTS reference code 3. Go through the whole architecture again	luoguojin	
2018.12.27	3.0	1.More IC Support 2.Modify descriptions of upgrade function	luoguojin	
2018.3.21	2.2	1.More IC Support	xiaoligen	
2017.12.26	2.1	1. Add FT8719 Support	xiaoligen	
2017.9.19	2.0	1. More IC Support 2. Modify upgrade & nodes 3. Add pinctrl to dtsi	xiaoligen	
2017.06.30	1.4	1. Remove lcd_cfg.i 2. Remove Force Touch	xiaoligen	
2017.03.06	1.3	1. Add test step of "Factory Test" 2. Modify upgrade configuration and description	xiaoligen	
2016.12.29	1.2	1. More IC Support	xiaoligen	
2016.10.31	1.1	1. More IC support 2. Extern mode, gesture update	xiaoligen	
2016.08.30	1.0	1. Initial draft.	xiaoligen	

Contents

1 Abstract

This guide introduces the structure and functions of "Focaltech TP Driver" based on linux environment and a porting reference for your Qualcomm platform, step by step.

Through this guide, you can get how to porting "Focaltech TP Driver" to Qualcomm platform, containing Kconfig/Makefile/DTS modifying, kernel configuration and compiling.

You can also customize your functions of TP, containing MULTI-TOUCH protocol, GESTURE en/dis, ESDCHECK en/dis, FACTORY TEST in driver en/dis, FIRMWARE UPGRADE and so on.

Note: Examples used in the guide are verified in Qualcomm Dragonboard 410c platform, only a reference, maybe different with your platform; please refer to your platform's document when porting the driver into your platform.

2 Interface Setting Recommendation

2.1 I2C interface

I2C speed recommendation: 400K (For all Focaltech's IC)

2.2 SPI interface

Different ICs have different settings, please refers to the following recommendations:

IC Serials	SPI Mode	SPI Speed	Description
FT8719/FT8615	Mode 1	8M	
FT8006P	Mode 1	6M	
FT7251/FT7252	Mode 0	6M	
FT8756/FT8656 FT8009 FT8006S-AA/Later ICs	Mode 0	Follow datasheet	

3 File Structure

The directory of "Focaltech TP Driver" is named as "focaltech_touch" as default, that means all driver files are in it. The directory structure looks like this:

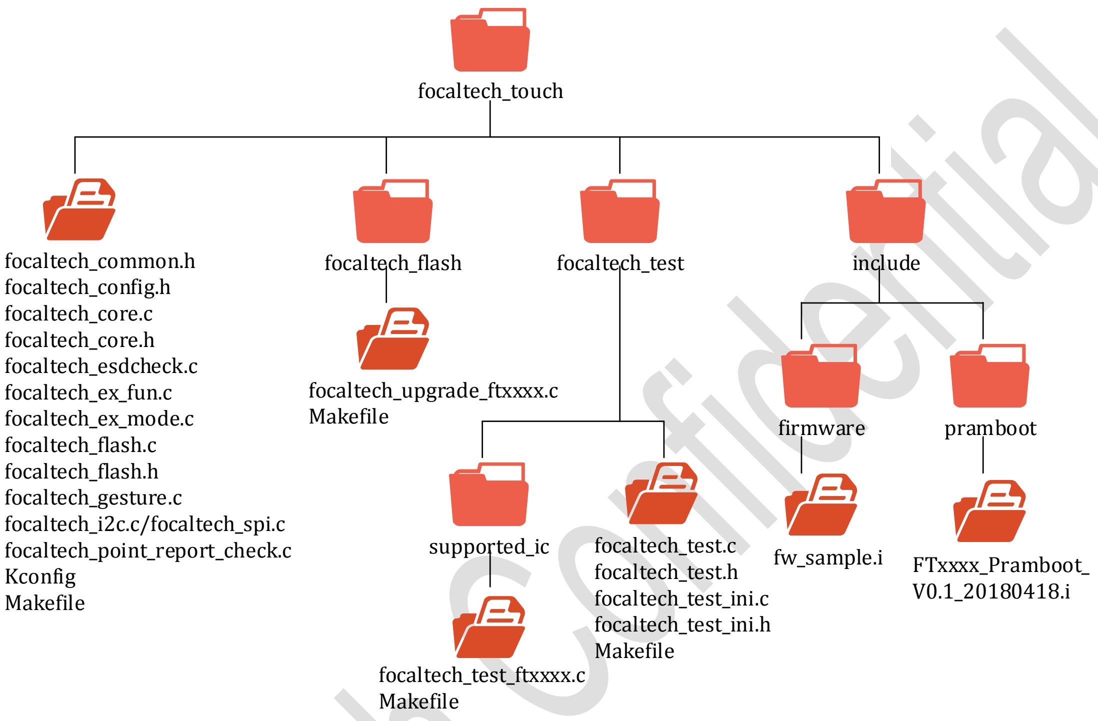


Figure1: Structure of "Focaltech TP Driver"

Table 1 shows the detail description of driver files:

Component	Files	Attribute	Description
Complier	Makefile Kconfig	Required	Use for kernel compiling and configuration.
Main	focaltech_common.h focaltech_config.h focaltech_core.c focaltech_core.h	Required	The main function of TP driver, that containing driver registration, bus interface (SPI/I2C) initialization, suspend/resume, multi-touch protocol support and so on. You can customize functions of TP driver by modifying focaltech_config.h.
Interface	focaltech_i2c.c focaltech_spi.c	Required	Bus communication with I2C/SPI; **Warning**: either I2C or SPI can be used at the same time.
Upgrade	focaltech_flash.c focaltech_flash.h focaltech_flash/	Optional	Code related to firmware upgrade; **Warning**: a. There isn't focaltech_flash/ directory when use SPI interface;

	include/firmware/ include/pramboot/		b. There isn't include/pramboot/ directory when don't support pramboot.
esdcheck	focaltech_esdcheck.c	Optional	Use to process ESD check function.
gesture	focaltech_gesture.c	Optional	Use to process gesture function.
sysfs/proc	focaltech_ex_fun.c	Optional	Create sysfs/proc node, use to communicate with APK or ADB.
factory test	focaltech_test/	Optional	Use for factory test; recommend to use APK for factory test.
Others	focaltech_ex_mode.c	Optional	Code to process cover/glove/charger functions.
	focaltech_point_report_check.c		Use to auto report all points' up events if no-touch timeout; Warning: although it's ok, but not recommend to use it, especially in beginning of a project.

Table 1: Driver Files description

Warning: If you want to remove the optional files, you should customize your sourcecode for compiling pass.

4 Porting TP Driver to Qualcomm Platform

4.1 Copy driver files into kernel

Copy “focaltech_touch” directory into kernel directory(kernel/drivers/input/touchscreen);

Then modify Kconfig/Makefile:

a. Add the line below to kernel/drivers/input/touchscreen/Kconfig:

```
source "drivers/input/touchscreen/focaltech_touch/Kconfig"
```

b. Add the line below to kernel/drivers/input/touchscreen/Makefile:

```
obj-$(CONFIG_TOUCHSCREEN_FTS)         +=  focaltech_touch/
```

4.2 Check and enable “Foaltech Touchscreen” driver

You can achieve it via two methods:

Modify menuconfig

Modify default kernel config file

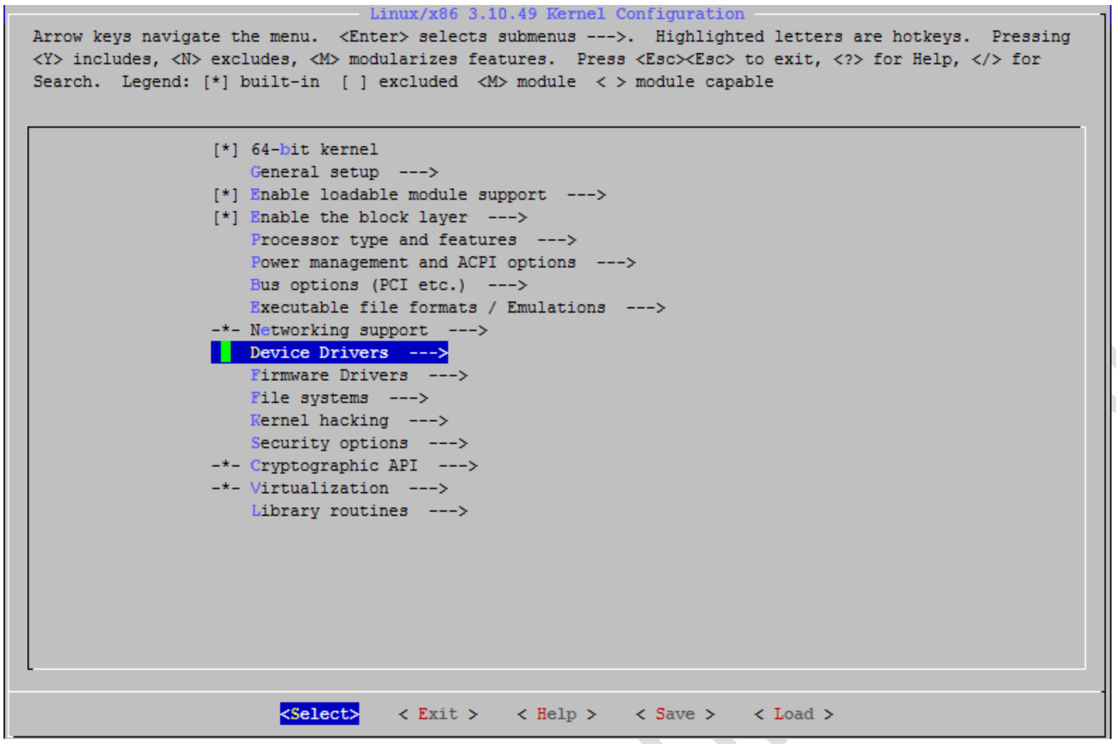
Modify menuconfig

Use “make menuconfig” command to call menuconfig, the following example for reference:

```
$ source build/envsetup.sh
$ lunch msm8916_64-userdebug
$ cd kernel
$ make menuconfig
```

After you execute “make menuconfig” command, then kernel configuration menu will be shown in Figure 2, now you should check “Foaltech Touchscreen” driver in following path: “**Device Drivers -> Input Device Support -> Touchscreens -> Focaltech Touchscreen”**

As mentioned above, the default directory name of driver is “focaltech_touch”, so if you want to modify the driver directory name, you can modify it in following path: “**Device Drivers -> Input Device Support -> Touchscreens -> “Focaltech Touchscreen” -> “Focaltech ts directory name”**



```
                  Linux/x86 3.10.49 Kernel Configuration
  Arrow keys navigate the menu.  <Enter> selects 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  [ ] excluded  <M> module  < > module capable

                  [*] 64-bit kernel
                      General setup  --->
                  [*] Enable loadable module support  --->
                  [*] Enable the block layer  --->
                      Processor type and features  --->
                      Power management and ACPI options  --->
                      Bus options (PCI etc.)  --->
                      Executable file formats / Emulations  --->
                  -*- Networking support  --->
                      Device Drivers  --->
                      Firmware Drivers  --->
                      File systems  --->
                      Kernel hacking  --->
                      Security options  --->
                  -*- Cryptographic API  --->
                  -*- Virtualization  --->
                      Library routines  --->

                <Select>    < Exit >    < Help >    < Save >    < Load >
```

Figure 2: Kernel configuration menu

Modify default kernel config file

Generally the default kernel configuration file (similar to defconfig) exists in directory: kernel/arch/arm64/config/, you should add the following code into the kernel configuration file:

```
CONFIG_TOUCHSCREEN_FTS=y
CONFIG_TOUCHSCREEN_FTS_DIRECTORY="focaltech_touch"
```

How to confirm that your modification has come into effect?

You can search .config file, if you can find "CONFIG_TOUCHSCREEN_FTS=y", that means your modification has come into effect, otherwise not.

.config path for reference: out/target/product/msm8916_64/obj/KERNEL_OBJ/.config

4.3 Configure DTS

Qualcomm dts file path: kernel/arch/arm64/boot/dts/qcom/apq8016-sbc.dtsi

I2C Interface Example:

```
/*
 * KEY_BACK: 158
 * KEY_MENU: 139
 * KEY_HOMEPAGE: 172
 * KEY_SEARCH: 217
```

```
    */
    focaltech@38{
            compatible = "focaltech,fts";                    /* do not modify */
            reg = <0x38>;                                    /* do not modify */
            interrupt-parent = <&msm_gpio>;                  /* INT pin */
            interrupts = <13 0x2>;
            //vdd-supply = <&pm8916_l15>;
            //vcc_i2c-supply = <&pm8916_l16>;
            focaltech,reset-gpio = <&msm_gpio 12 0x01>;          /* RST pin */
            focaltech,irq-gpio = <&msm_gpio 13 0x02>;            /* INT pin */
            focaltech,max-touch-number = <5>;
            focaltech,display-coords =  <0 0 1079 1919>;         /* resolution */

            /* pinctrl config */
            pinctrl-names = "pmx_ts_active","pmx_ts_suspend","pmx_ts_release";
            pinctrl-0 = <&ts_int_active>;
            pinctrl-1 = <&ts_int_suspend>;
            pinctrl-2 = <&ts_release>;

            /* key settings */
           /* focaltech,have-key;
           focaltech,key-number = <3>;
           focaltech,keys = <139 172 158>;                     /* key codes*/
           focaltech,key-x-coords = <200 600 800>;           /* keys x coords */
           focaltech,key-y-coords = <2000 2000 2000>;  /* keys y coords */
           */
    };
```

SPI Interface Example:

```
    spi@78b9000 {
            focaltech@0 {
                    compatible = "focaltech,fts";
                    reg = <0x0>;
                    spi-max-frequency = <6000000>;
                    interrupt-parent = <&msm_gpio>;
                    interrupts = <13 0x2>;
                    focaltech,reset-gpio = <&msm_gpio 12 0x01>;
                    focaltech,irq-gpio = <&msm_gpio 13 0x02>;
                    focaltech,max-touch-number = <5>;
```

```
                focaltech,display-coords =  <0 0 1079 1919>;
                pinctrl-names = "pmx_ts_active","pmx_ts_suspend","pmx_ts_release";
                pinctrl-0 = <&ts_int_active>;
                pinctrl-1 = <&ts_int_suspend >;
                pinctrl-2 = <&ts_release>;
        };
};
```

You can also refer to docs/focaltech-ts.txt for detail description

4.4 Compile kernel and generate boot.img

```
$ make bootimage –j4
```

5 Driver Customization

You can modify the file of focaltech_config.h to customize your driver components.

FTS_CHIP_TYPE

Set the focaltech chip type which current driver support. The value of this macro must be consistent with the IC type that you use in your project.

FTS_DEBUG_EN

Enable/Disable debug log.

FTS_MT_PROTOCOL_B_EN

Set the linux multi-touch protocol, Enable: Protocol B, Disable: Protocol A.

FTS_REPORT_PRESSURE_EN

Whether register and report pressure (ABS_MT_PRESSURE) or not? Enable: Register and report pressure, Disable: Not.

Strongly recommend setting the value to enable as default, and report actual and variant pressure to applications.

Warning: Don't report constant pressure to applications; If pressure from firmware is constant, you should set it to disable.

FTS_PEN_EN

Enable/Disable stylus pen support.

FTS_GESTURE_EN

Enable/Disable gesture function.

FTS_ESDCHECK_EN

Enable/Disable ESD check function. It can reset IC to normal state when esd damage is checked.

FTS_TEST_EN

Enable/Disable factory test function.

Strongly recommend using APK for factory test.

FTS_POWER_SOURCE_CUST_EN

Enable/Disable power for outcell IC when need. Default value for outcell IC is enable, so if you don't need to set the power of IC, please set it to disable manually.

FTS_PINCTRL_EN

Enable/Disable pinctrl function for INT/RST gpio pin

FTS_AUTO_UPGRADE_EN/ FTS_AUTO_LIC_UPGRADE_EN/
FTS_GET_MODULE_NUM/
FTS_MODULE_ID/ FTS_MODULE2_ID/ FTS_MODULE3_ID/
FTS_MODULE_NAME/ FTS_MODULE2_NAME/ FTS_MODULE3_NAME/
FTS_UPGRADE_FW_FILE/ FTS_UPGRADE_FW2_FILE/ FTS_UPGRADE_FW3_FILE

These macros are related to upgrade function, please refer to upgrade function for detail description.

5.1 Upgrade function

This chapter describes the whole functions of upgrade. You can get how to enable upgrade function when power on, how to set upgrade firmware, how to distinguish different modules etc.

TP driver will initialize the upgrade function when booting kernel. So we can use upgrade function after TP driver initialization completion, for example, use ADB tool to operate sysfs node, or use APK to operate proc node.

Focaltech provides three methods to upgrade firmware: upgrade during kernel booting, upgrade using ADB (sysfs), upgrade using APK (proc).

5.1.1 Enable upgrade function during kernel booting

FTS_AUTO_UPGRADE_EN

By default, upgrade from power on is disable, which avoids some unfamiliar person to use it to break the default valid firmware in IC. After you are familiar with upgrade function, you must enable upgrade function during kernel booting in your project, which can upgrade firmware of IC automatically when you put a new firmware instead of old firmware.

How to enable upgrade function during kernel booting, set FTS_AUTO_UPGRADE_EN to 1.

FTS_AUTO_LIC_UPGRADE_EN

If the IC you use in your project supports LCD initial code upgrade, you should set FTS_AUTO_LIC_UPGRADE_EN to 1, then TP driver will confirm to upgrade LCD initial code or not automatically.

Warning: You must use all.bin/all.i for LCD initial code upgrade. And we strongly recommend set it to 0(disabled) for mass production, maybe you can enable it for debug usage.

5.1.2 Distinguish different modules

Generally, there are more than one module in one project. These modules may use different glasses, may be packaged from different module manufacturer which make these modules having different configurations; Different modules means different firmware. So when you upgrade firmware to IC, you must confirm the firmware used by host upgrade matches your module. TP driver provides the following mechanism for it:

FTS_GET_MODULE_NUM

Set number of the modules supported by your project. You need keep it to 0 when your project only has one module, no need to set it to 1; but when your project has more than one module, for example, 2 or 3 or ..., you must set this macro. When its value is greater than or equal to 2, these macros are used to distinguish different modules, such as: FTS_MODULE_ID, FTS_MODULE_NAME, and so on.

FTS_MODULE_ID/ FTS_MODULE2_ID/ FTS_MODULE3_ID

Module's ID to distinguish different modules, generally means module's vendor id(GLASS_ID << 8 + VENDOR_ID), also maybe mean GPIOs or LCM ID. Meaning of these macros are different followed different project.

Macro FTS_MODULE_ID and FTS_UPGRADE_FW_FILE are combined to use. The detail usage is the following:

When TP driver check module id matching with FTS_MODULE_ID, then TP driver will use firmware of FTS_UPGRADE_FW_FILE to upgrade;

Similarly, FTS_MODULE2_ID is related to FTS_UPGRADE_FW2_FILE;

FTS_MODULE3_ID is related to FTS_UPGRADE_FW3_FILE.

FTS_UPGRADE_FW_FILE/ FTS_UPGRADE_FW2_FILE/ FTS_UPGRADE_FW3_FILE

The firmware's name, usually is similar to "include/firmware/xxx.i" (xxx means your actual firmware's name). The firmware will be included into sourcecode, which forms firmware array buffer.

If you want to use new firmware to replace old firmware, you should modify these macros firstly, then compile kernel again.

FTS_MODULE_NAME/ FTS_MODULE2_NAME/ FTS_MODULE3_NAME

Module's vendor name, should be similar to "tianma", "boe" or others.

If you use request_firmware() function to get firmware, you need set these macros. These macros are used to form the firmware's name, which are used to be the 2nd parameter of request_firmware() function, format is the following:

"focaltech_ts_fw_" + FTS_MODULE_NAME

For example:

Set FTS_MODULE_NAME to "boe", then firmware's name should be "focaltech_ts_fw_boe.bin" in /system/etc/firmware/ or else (by customer's system).

How to configure your upgrade setting

a. If you want upgrade firmware during kernel booting, please set FTS_AUTO_UPGRADE_EN to 1.

b. If you want to upgrade LCD initial code, please set FTS_AUTO_LIC_UPGRADE_EN to 1.
Please set it to 0 for mass production.

c. How to configure upgrade firmware, there are several situations you may encounter:

One module supported.

Set FTS_GET_MODULE_NUM to 0

Set FTS_UPGRADE_FW_FILE to correct firmware name

Set FTS_MODULE_NAME to correct value if you want to get firmware using request_firmware() function

Two modules supported.

Set FTS_GET_MODULE_NUM to 2

Set FTS_MODULE_ID & FTS_MODULE2_ID to correct value

Set FTS_UPGRADE_FW_FILE & FTS_UPGRADE_FW2_FILE to correct firmware

name
Set FTS_MODULE_NAME & FTS_MODULE2_NAME to correct value if you want to get firmware using request_firmware() function

Three modules supported.

Set FTS_GET_MODULE_NUM to 3
Set FTS_MODULE_ID & FTS_MODULE2_ID & FTS_MODULE3_ID to correct value
Set FTS_UPGRADE_FW_FILE & FTS_UPGRADE_FW2_FILE & FTS_UPGRADE_FW3_FILE to correct firmware name
Set FTS_MODULE_NAME & FTS_MODULE2_NAME & FTS_MODULE3_NAME to correct value if you want to get firmware using request_firmware() function

Four or more modules supported.

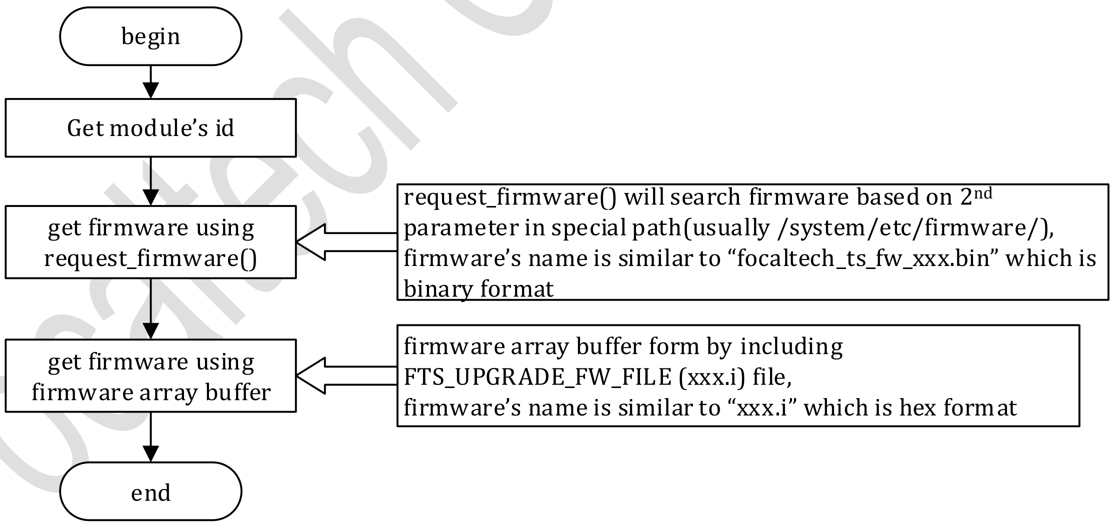
You must FTS_MODULE4_ID/ FTS_UPGRADE_FW4_FILE/ FTS_MODULE4_NAME or more. And use rules above to set corresponding macros.

d. After step a~c configure completely, you should push xxx.i into /include/firmware/ directory; or push focaltech_ts_fw_xxx.bin into /system/etc/firmware/ directory.

5.1.3 Flowchart of getting firmware

Firstly, TP driver will get firmware via request_firmware() function that is the standard library function of linux kernel.

Secondly, TP driver will get firmware directly using firmware array buffer formed by including FTS_UPGRADE_FW_FILE (xxx.i) file when request_firmware() fail to get firmware.


begin
Get module's id
get firmware using request_firmware()
request_firmware() will search firmware based on 2nd parameter in special path(usually /system/etc/firmware/), firmware's name is similar to "focaltech_ts_fw_xxx.bin" which is binary format
get firmware using firmware array buffer
firmware array buffer form by including FTS_UPGRADE_FW_FILE (xxx.i) file, firmware's name is similar to "xxx.i" which is hex format
end

NOTE:

1. If your project supports more than three modules, you should define the following macros by yourself to distinguish more modules: FTS_MODULE4_ID/ FTS_UPGRADE_FW4_FILE/ FTS_MODULE4_NAME and so on.

2. If you want to get firmware using customization method, you should ignore all macros

above, and use your method to get firmware before firmware upgrade. Of course, you should modify the source code by yourself.

3. Firmware has two format: xxx.bin and xxx.i, which are different so that you can't mix to use, that means you can't rename xxx.bin to xxx.i, or xxx.i to xxx.bin directly, you should choose different format in different situation.

4. IC types(FT8607 FT8006M FT8201 FT7250 FT8006U FT8006S FT8739 FT8006P FT8613S FT8756 FT8009 FT8302 FT7251 FT7252and all new IDC) use all.i fw to upgrade, others use app.i.

5.2 Factory test function

5.2.1 Enable factory test

Please set FTS_TEST_EN to 1 to enable factory test in driver, enable FTS_TEST_EN in focaltech_config.h:

```
#define FTS_TEST_EN                    1
```

5.2.2 How to do factory test

After enable factory test in driver, TP driver will generate a sysfs node called "fts_test". We should use this sysfs node to run factory test and get the test results. The detail steps are the following:

a. Push test configuration file(xxx.ini) into /sdcard/

```
> adb root
> adb remount
> adb push xxx.ini /sdcard/
```

b. Use adb command to run factory test

```
> adb shell
# cd /sys/bus/i2c/devices/*-0038 ("*" stands for the i2c bus no. for TP)
# echo xxx.ini > fts_test
```

c. Get test results

Wait test finished, then test results will be generated under /sdcard/ directory. There are two result files: testdata.csv & testresult.txt, which you can confirm factory test is pass or not from. Also you can check kernel log to check test result.

Use pull command to pull out the test result files:

```
> adb pull /sdcard/testdata.csv d:\
> adb pull /sdcard/testresult.txt d:\
```

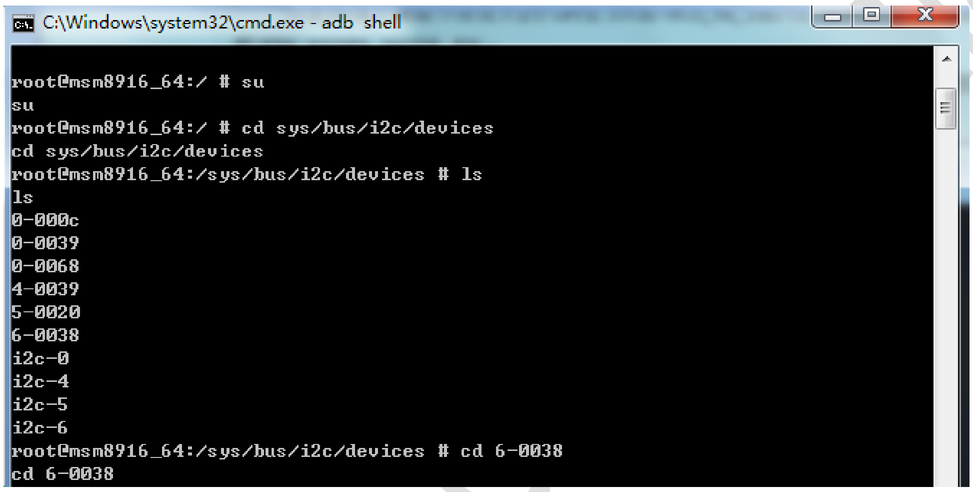
6 Sysfs Nodes

6.1 Where are these sysfs nodes

a. I2C interface

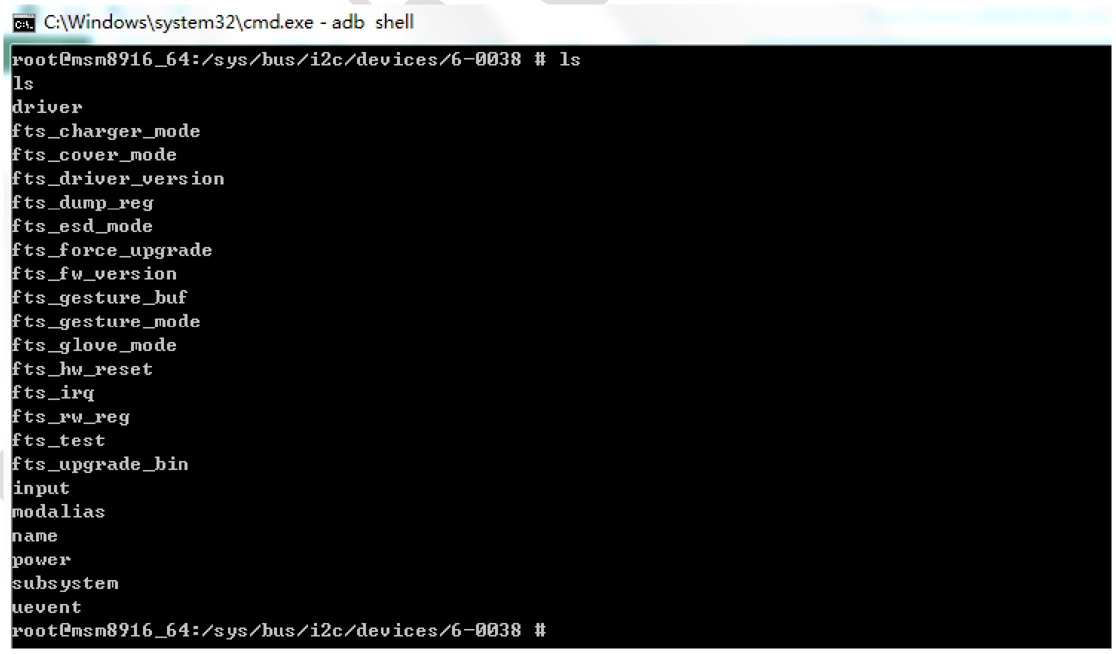
When interface is I2C, then sysfs nodes will be generated under /sys/bus/i2c/devices/*-0038/ directory, "*" stands for I2C bus number depending on your project design.

For example:



C:\Windows\system32\cmd.exe - adb shell

```
root@msm8916_64:/ # su
su
root@msm8916_64:/ # cd sys/bus/i2c/devices
cd sys/bus/i2c/devices
root@msm8916_64:/sys/bus/i2c/devices # ls
ls
0-000c
0-0039
0-0068
4-0039
5-0020
6-0038
i2c-0
i2c-4
i2c-5
i2c-6
root@msm8916_64:/sys/bus/i2c/devices # cd 6-0038
cd 6-0038
```



C:\Windows\system32\cmd.exe - adb shell

```
root@msm8916_64:/sys/bus/i2c/devices/6-0038 # ls
ls
driver
fts_charger_mode
fts_cover_mode
fts_driver_version
fts_dump_reg
fts_esd_mode
fts_force_upgrade
fts_fw_version
fts_gesture_buf
fts_gesture_mode
fts_glove_mode
fts_hw_reset
fts_irq
fts_rw_reg
fts_test
fts_upgrade_bin
input
modalias
name
power
subsystem
uevent
root@msm8916_64:/sys/bus/i2c/devices/6-0038 #
```

b. SPI interface

Similarly, when interface is SPI, then sysfs nodes will be generated under /sys/bus/spi/devices/spi*/ directory, "*" stands for SPI bus number depending on your project design.

Warning: Sysfs nodes may be different with different driver version.

6.2 Component Nodes

Gesture Component: fts_gesture_mode / fts_gesture_buf which are generated when set FTS_GESTURE_EN to 1.

a. Sysfs node: fts_gesture_mode

Usage:

```
# echo 0 > fts_gesture_mode // Disable Gesture
# echo 1 > fts_gesture_mode // Enable Gesture
# cat fts_gesture_mode        // show current Gesture status
```

b. Sysfs node: fts_gesture_buf

Usage:

```
# cat fts_gesture_buf          // Get gesture buffer information
```

ESD Check component: fts_esd_mode which are generated when set FTS_ESDCHECK_EN to 1.

Sysfs node: fts_esd_mode

Usage:

```
# echo 0 > fts_esd_mode          // disable ESD
# echo 1 > fts_esd_mode          // enable ESD
# cat fts_esd_mode               // show current ESD check status
```

After executing command “echo 0 to fts_esd_mode”, the whole esd check function will be disabled until executing command “echo 1 > fts_esd_mode”.

Glove/Cover/Charger component: fts_glove_mode/ fts_cover_mode/ fts_charger_mode.

a. Sysfs node: fts_glove_mode

Usage:

```
# echo 0 > fts_glove_mode        // Disable glove mode
# echo 1 > fts_glove_mode        // Enable glove mode
# cat fts_glove_mode             // show current glove status
```

b. Sysfs node: fts_cover_mode

Usage:

```
# echo 0 > fts_cover_mode        // Disable cover mode
# echo 1 > fts_cover_mode        // Enable cover mode
# cat fts_cover_mode             // show current cover status
```

c. Sysfs node: fts_charger_mode

Usage:

```
# echo 0 > fts_charger_mode      // Disable charger mode
# echo 1 > fts_charger_mode      // Enable charger mode
```

# cat fts_charger_mode // show current charger status

6.3 Debugging Nodes

Sysfs debugging nodes are used to get TP driver information, and debug TP driver, which makes us debug TP driver easier and more convenient.

How to enable sysfs debugging nodes

a. Sysfs node: fts_driver_info

Usage:

# cat fts_driver_info // show driver info, including version, resolution, INT etc

b. Sysfs node: fts_fw_version

Usage:

# cat fts_fw_version // show firmware version

c. Sysfs node: fts_dump_reg

Usage:

# cat fts_dump_reg // show key register' values

d. Sysfs node: fts_rw_reg which use to read/write register.

Usage:

- Read register (only one byte)

  # echo xx > fts_rw_reg // xx stands for register address, hex format

  # cat fts_rw_reg // get value of register xx

- Write register (only one byte)

  #echo xxaa > fts_rw_reg // xx stands for register address, aa stands for value

- Read plenty of data from register (multiple bytes)

  # echo 1xxzz > fts_rw_reg //1 stands for read operation

  # cat fts_rw_reg // get value from register xx ~ (xx + zz)

- Write plenty of data to register (multiple bytes)

#echo 0xxzzaabbcc... > fts_rw_reg // 0 stands for write operation, aabbcc... stands for data you want to write to address xx

xx stands for register address, aa/bb/cc stands for value, zz stands for length; All data are hex format, and must be two characters.

For example:

Read byte from register 0x00

# echo 00 > fts_rw_reg

# cat fts_rw_reg

Read 10 bytes from register 0xD3

```
        # echo 1D30A > fts_rw_reg
        # cat fts_rw_reg
```

e. Sysfs node: fts_upgrade_bin which use to upgrade firmware manually.

Usage:

```
> adb push xxx.bin /sdcard/
> adb shell
# cd /sys/bus/xxx/devices/yyy    // xxx: i2c/spi, yyy: detail bus number of TP mounting
# echo xxx.bin > fts_upgrade_bin
```

f. Sysfs node: fts_irq

Usage:

```
# echo 0 > fts_irq       // execute disable_irq function
# echo 1 > fts_irq       // execute enable_irq function
# cat fts_irq            // show irq_depth, 0: irq is enabled, others: irq is disabled
```

g. Sysfs node: fts_hw_reset

Usage:

```
# cat fts_hw_reset       // execute TP reset
```

h. Sysfs node: fts_boot_mode, only for SPI protocol

Usage:

```
# echo 0 > fts_boot_mode     // make SPI protocol to firmware mode
# echo 1 > fts_boot_mode     // make SPI protocol to boot mode
# cat fts_boot_mode          // show current SPI protocol mode
```

i. Sysfs node: fts_log_level

Usage:

```
# echo log_level > fts_log_level     // log_level will be 0/1/2...
# cat fts_log_level                  // get current log level
```