SiW Touch Driver for Automotive & Large

v A1.01

2020.08.10

Silicon Works Co., Ltd.

History

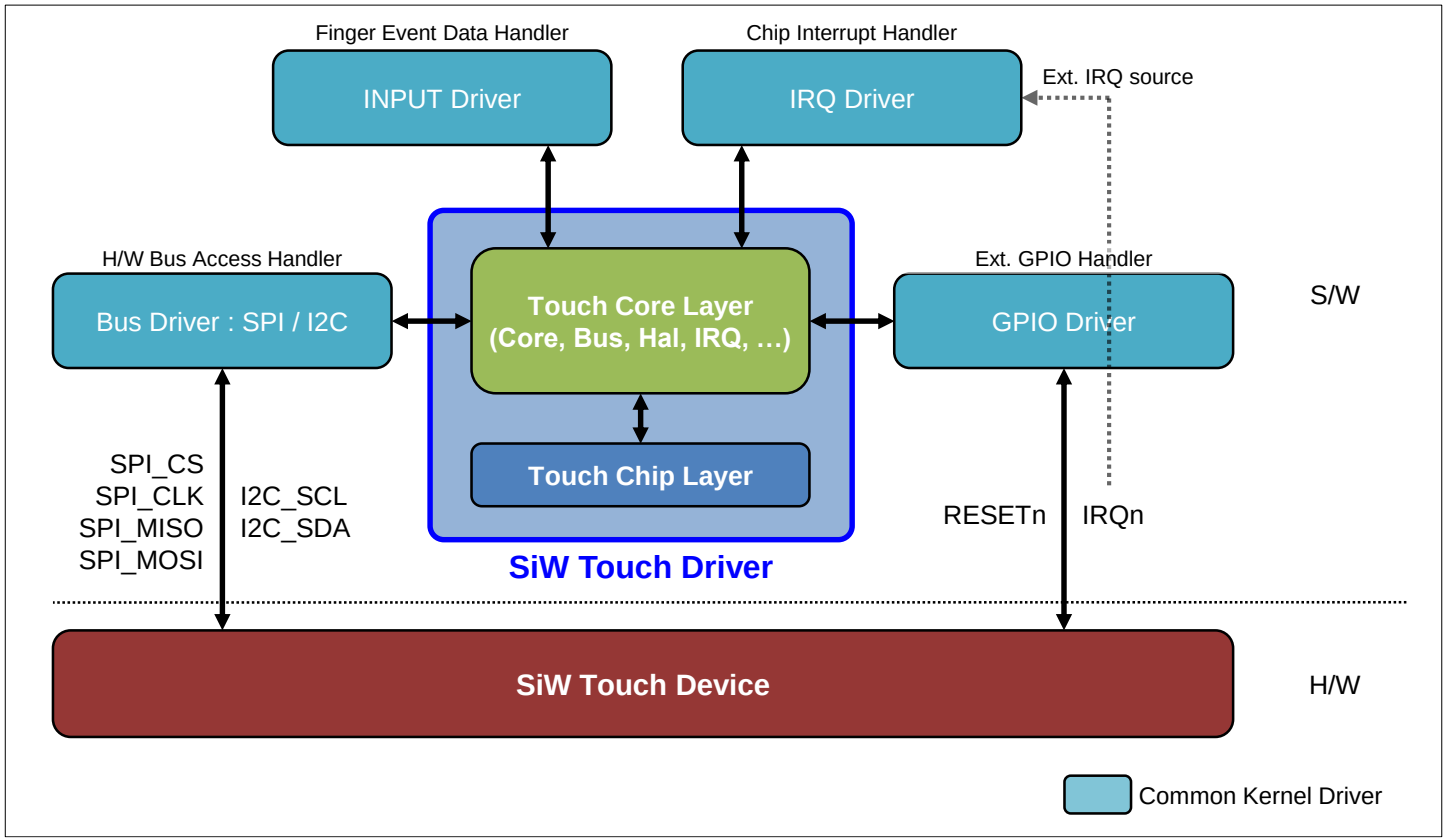
Version	Date	Description
A1.00	2020.06.30	1st release
A1.01	2020.08.10	Add : Appendix-1. Reset Control

Table Of Contents

1. Driver Operation

1.1 Driver Architecture

(1) Overview



[Fig. 1-1] Driver Relationship

1. Driver Operation

1.1 Driver Architecture

(2) SiW Touch Driver Files

Layer	Name	Description
Touch Core Layer	siw_touch.c	Touch Core
	siw_touch_bus.c	Touch Bus I/F main
	siw_touch_bus_i2c.c	Touch Bus I/F - I2C type
	siw_touch_bus_spi.c	Touch Bus I/F - SPI type
	siw_touch_event.c	Touch Input & Event control
	siw_touch_gpio.c	Touch GPIO control
	siw_touch_irq.c	Touch Interrupt control
	siw_touch_of.c	Touch Device Tree analysis
	siw_touch_sysfs.c	Touch Sysfs control
	siw_touch_sys.c	Helper for Touch & System Inter-connection
	siw_touch_misc.c	Device node(/dev/{misc name}) for direct bus access
	siw_touch_hal.c	Touch HAL
	siw_touch_hal_sysfs.c	Touch HAL for Sysfs
Touch Chip Layer	touch_xxxxx.c	Entry configuration for the chipset xxxxx
Build Files	Kconfig / Makefile	

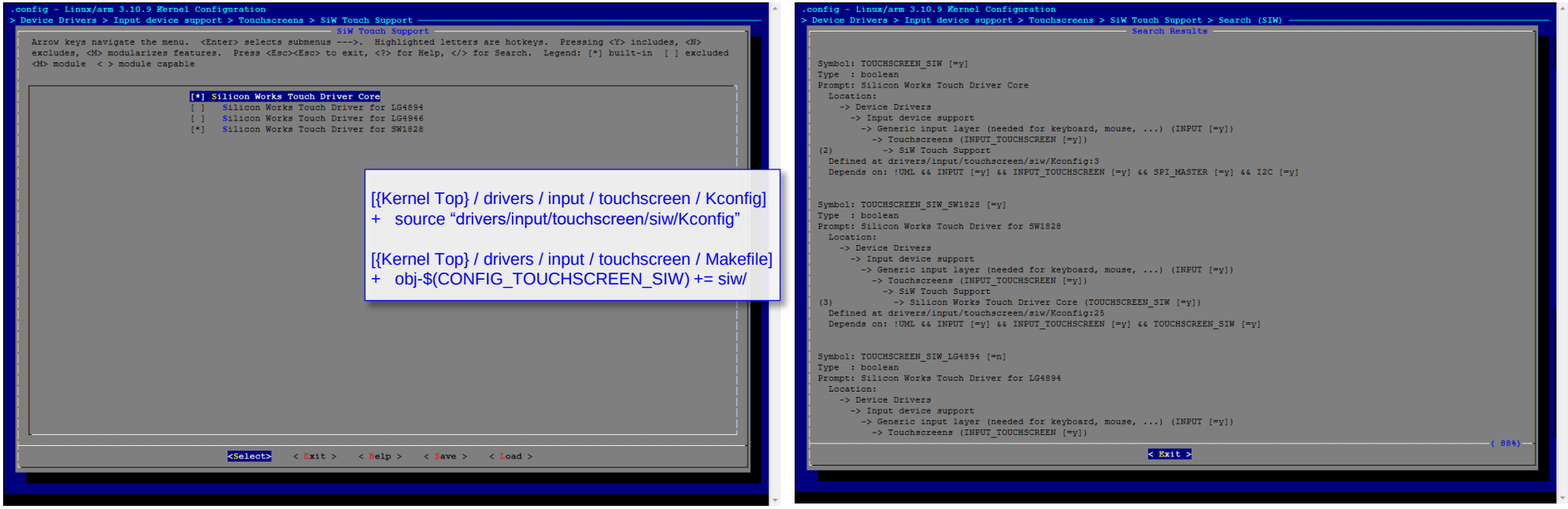
[Table. 1-1] Driver File List

* HAL : Hardware Abstraction Layer

1. Driver Operation

1.1 Driver Architecture

(2) SiW Touch Driver Files - Kconfig



[Fig. 1-2] Kconfig (example)

1. Driver Operation

1.1 Driver Architecture

(2) SiW Touch Driver Files - Test Environment

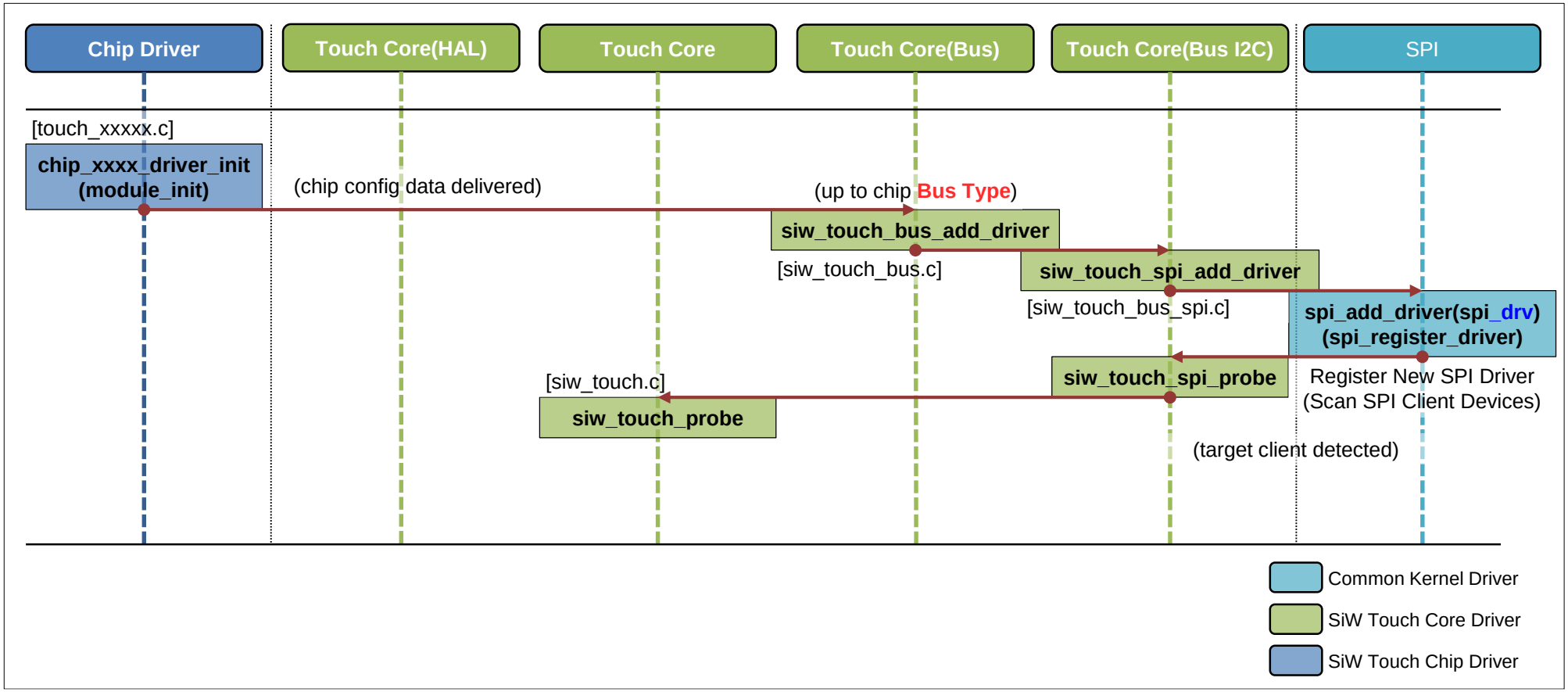
Test Environment		
H/W		Odroid-XU4(Exynos5422)
S/W	Platform Version	Android 4.4.4
		Kernel 3.10.9
	Driver Folder	{Kernel Top} / drivers / input / touchscreen / siw

[Table. 1-2] Test Environment

1. Driver Operation

1.2 Initialization Flow

(1) Probe Sequence - SPI (SW42000A, SW49408, SW49407, LG4946, LG4895)

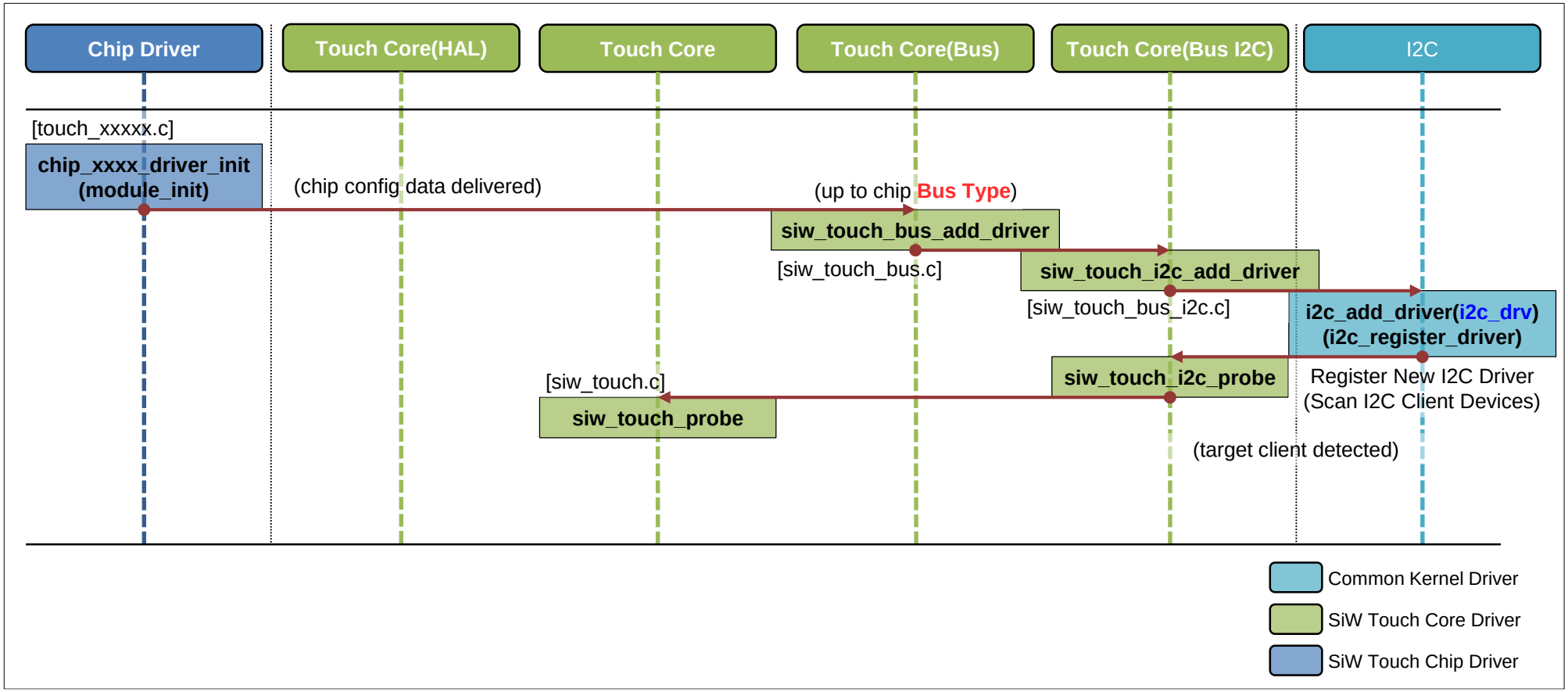


[Fig. 1-3] Initial Probe Sequence (SPI)

1. Driver Operation

1.2 Initialization Flow

(2) Probe Sequence - I2C (SW49501, SW49106, SW46104, LG4951, LG4894, SW42101, SW1828, SW42103, SW17700)

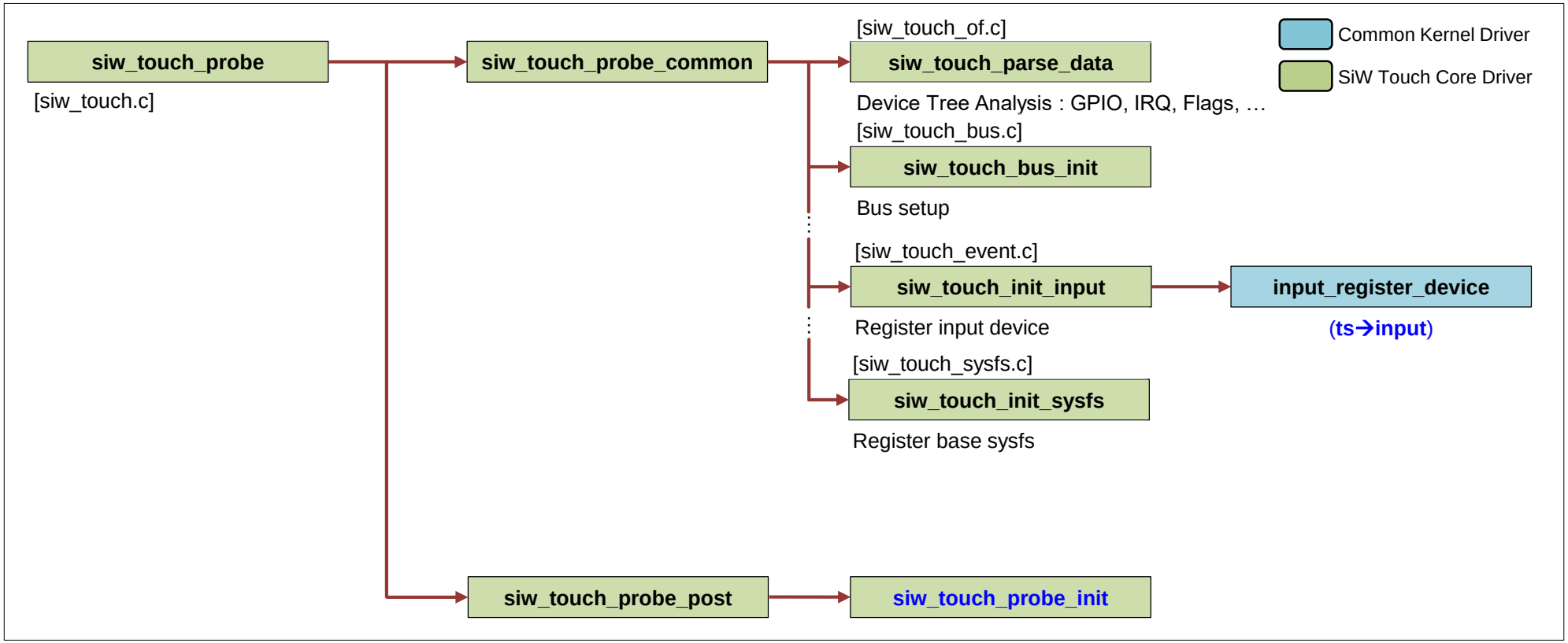


[Fig. 1-4] Initial Probe Sequence (I2C)

1. Driver Operation

1.2 Initialization Flow

(3) siw_touch_probe (1/2)

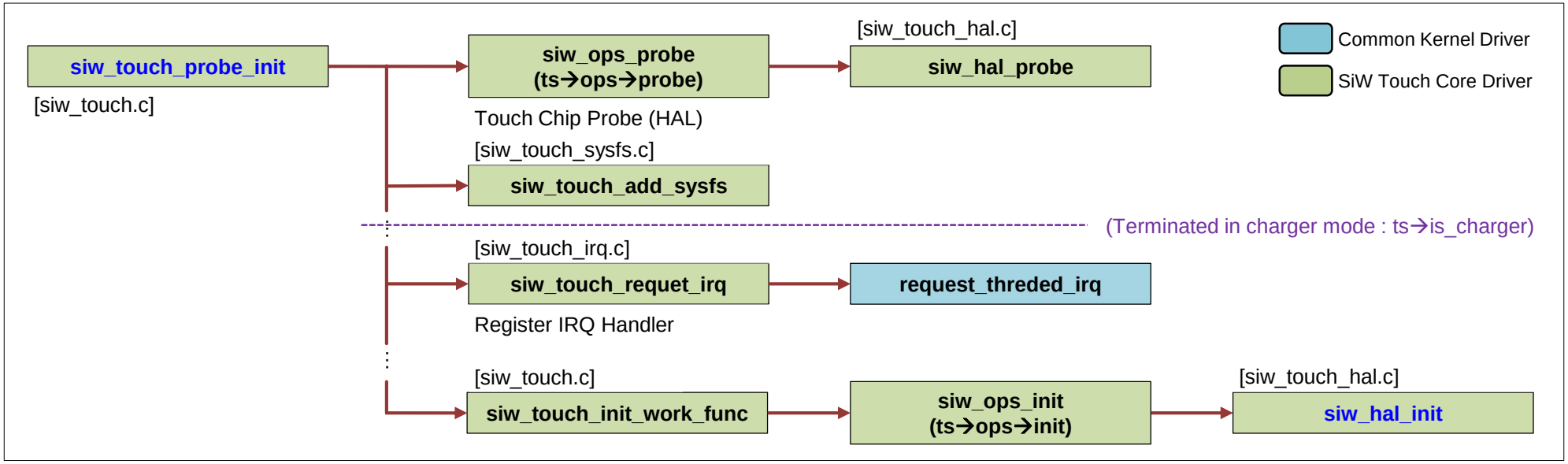


[Fig. 1-5-1] Inside operation of siw_touch_probe (1/2)

1. Driver Operation

1.2 Initialization Flow

(3) siw_touch_probe (2/2) - Actual HW access for touch device



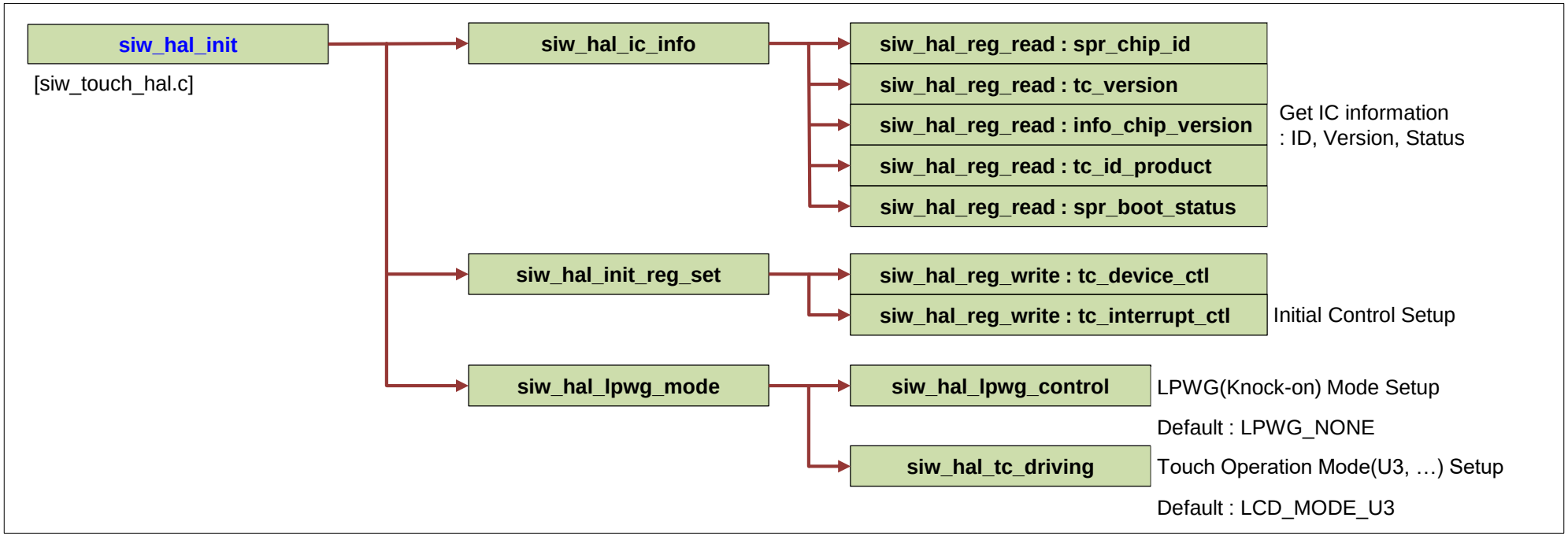
[Fig. 1-5-2] Inside operation of siw_touch_probe (2/2)

- The **siw_touch_probe_init** can be postponed by TOUCH_USE_PROBE_INIT_LATE option and the post processing(ts->init_late) can be triggered by either of the following
  (1) via sysfs - echo 0x55AA > /sys/devices/virtual/input/siw_touch_input/init_late

1. Driver Operation

1.2 Initialization Flow

(4) siw_hal_init

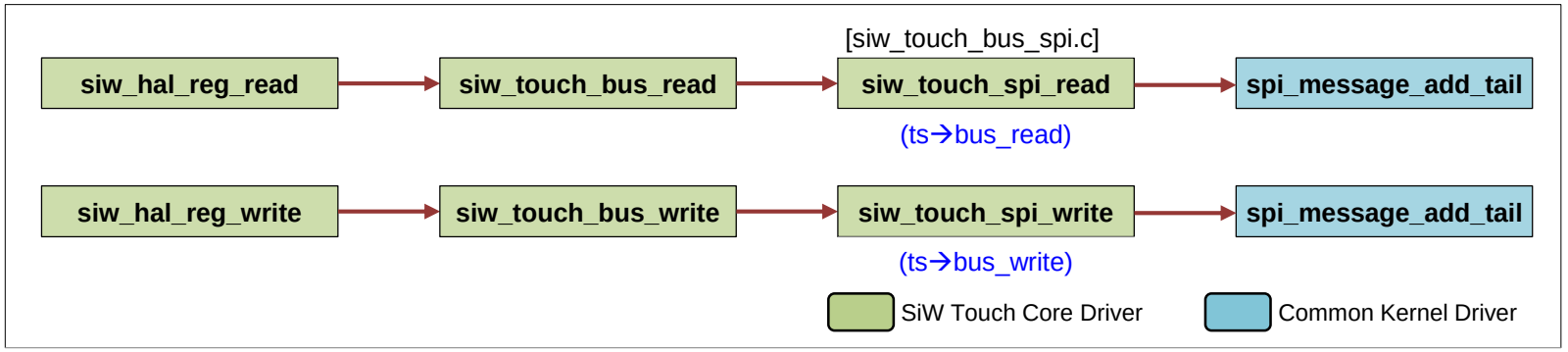


[Fig. 1-6] Inside operation of siw_hal_init

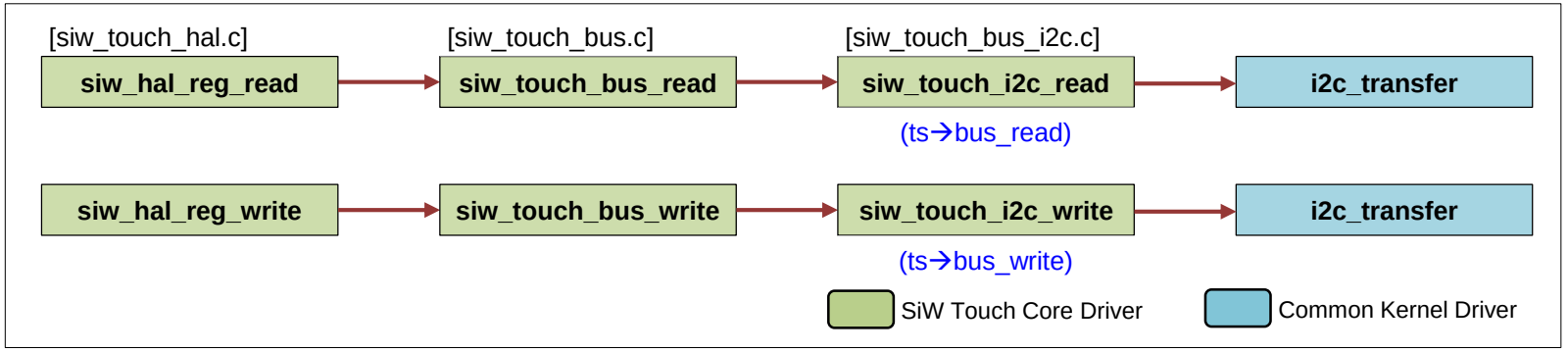
1. Driver Operation

1.3 Operation

(1) Bus Access



[Fig. 1-7] Bus Access Flow for SPI type

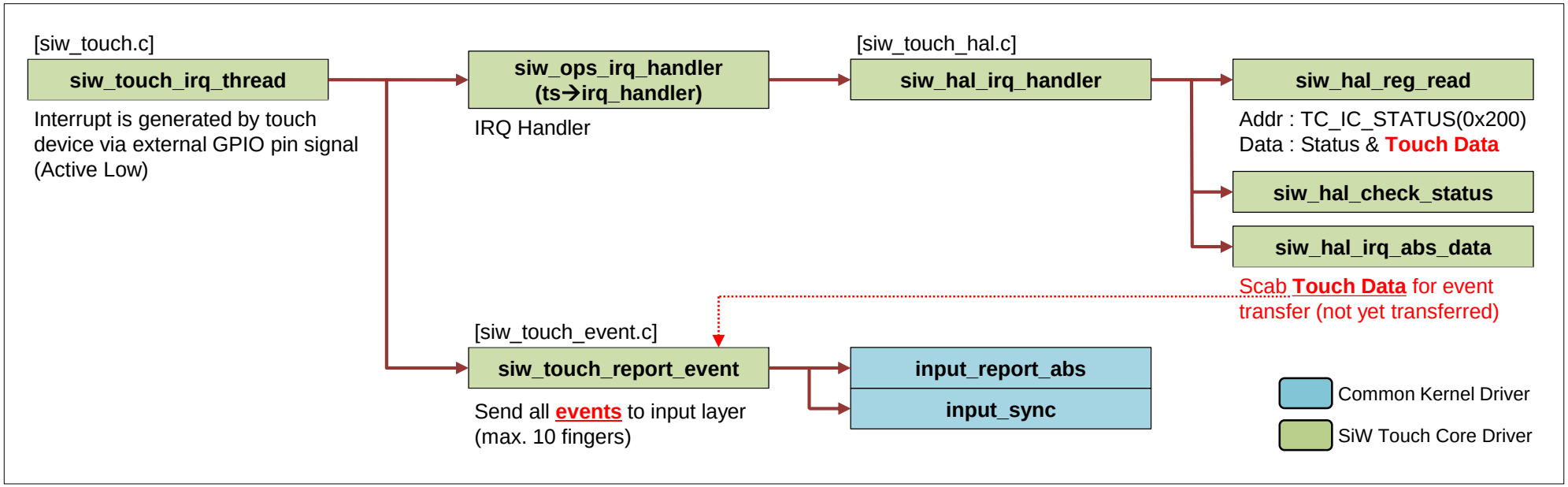


[Fig. 1-8] Bus Access Flow for I2C type

1. Driver Operation

1.3 Operation

(2) IRQ Handler (when touch event detected)



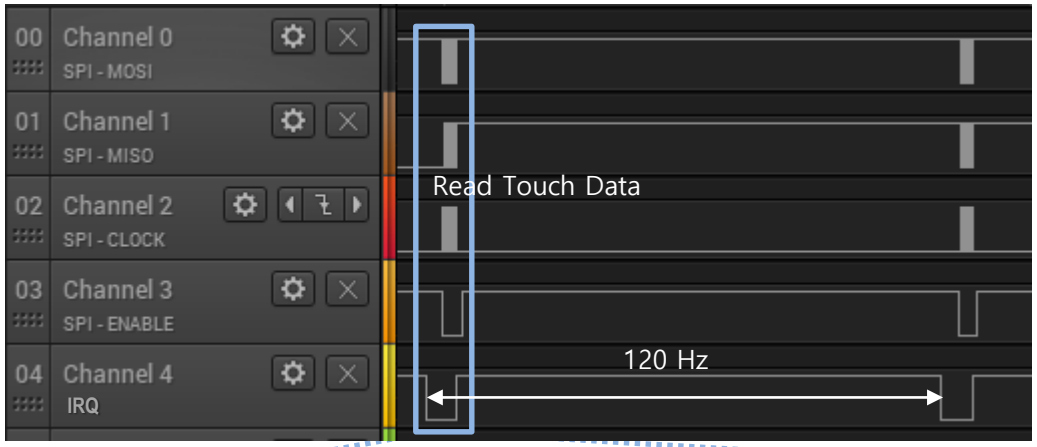
[Fig. 1-9] Interrupt Handling

- An appropriate external interrupt connection shall be guaranteed for the accuracy of this IRQ operation
- IRQ Flags
  Recommended flag setup value is **0x2002**((**IRQF_TRIGGER_FALLING**(0x02) | **IRQF_ONESHOT**(0x2000)), however, some problematic chipset may call handler routine twice at both edge, falling and rising.
  In this case, use **0x2008**((**IRQF_TRIGGER_LOW**(0x08) | **IRQF_ONESHOT**(0x2000)) instead of **0x2002**

1. Driver Operation

1.3 Operation

(2) IRQ Handler - SPI protocol example



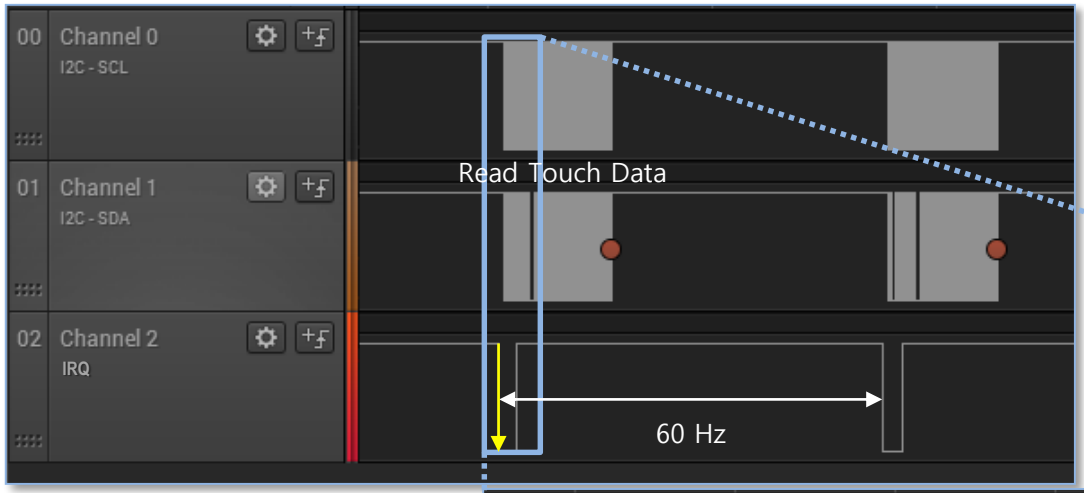
- The regular period of touch IRQ is 60Hz or 120Hz.
- Reading time for 132 bytes data shall be terminated in a given period or the IRQ sync distortion will happen.
- Reading data twice in single IRQ section is not permitted because 'invalid IRQ state' may be detected in 2nd reading

[Fig. 1-10] IRQ & SPI protocol

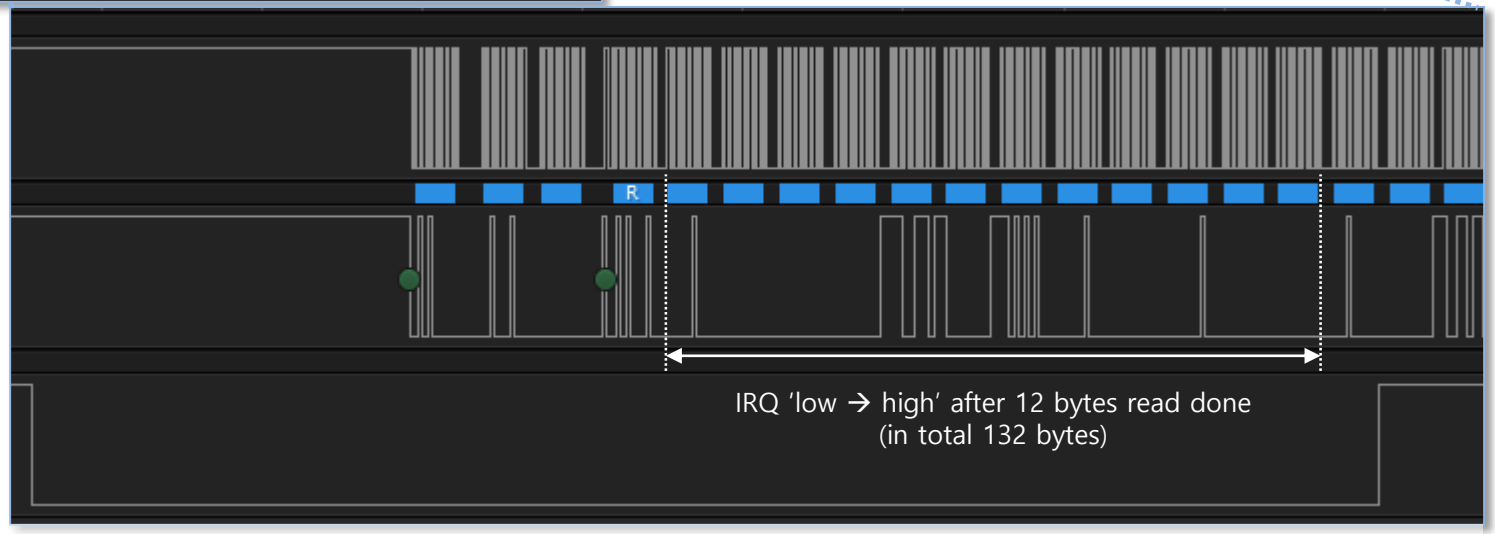
1. Driver Operation

1.3 Operation

(2) IRQ Handler - I2C protocol example



- The regular period of touch IRQ is 60Hz or 120Hz.
- Reading time for 132 bytes data shall be terminated in a given period or the IRQ sync distortion will happen.
- Reading data twice in single IRQ section is not permitted because 'invalid IRQ state' may be detected in 2nd reading

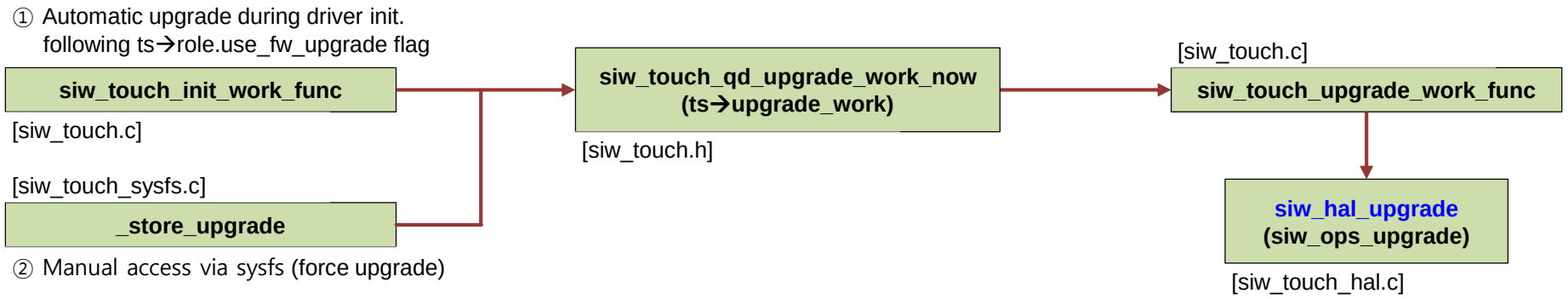


[Fig. 1-11] IRQ & I2C protocol

1. Driver Operation

1.3 Operation

(3) FW Upgrade (core layer)



1. If TOUCH_USE_FW_BINARY flag used
1-1 Default upgrade (through version comparison)
do upgarde using **binary header** link
1-2 echo {bin} > fw_upgrade
do force-upgrade using **binary header** link (same as 1-1)
1-3 echo /.../fw_img > fw_upgrade
do force-upgrade using **request_firmware** (relative path)
1-4 echo {root}/.../fw_img > fw_upgrade
do force-upgrade using **normal file open** control (absolute path)

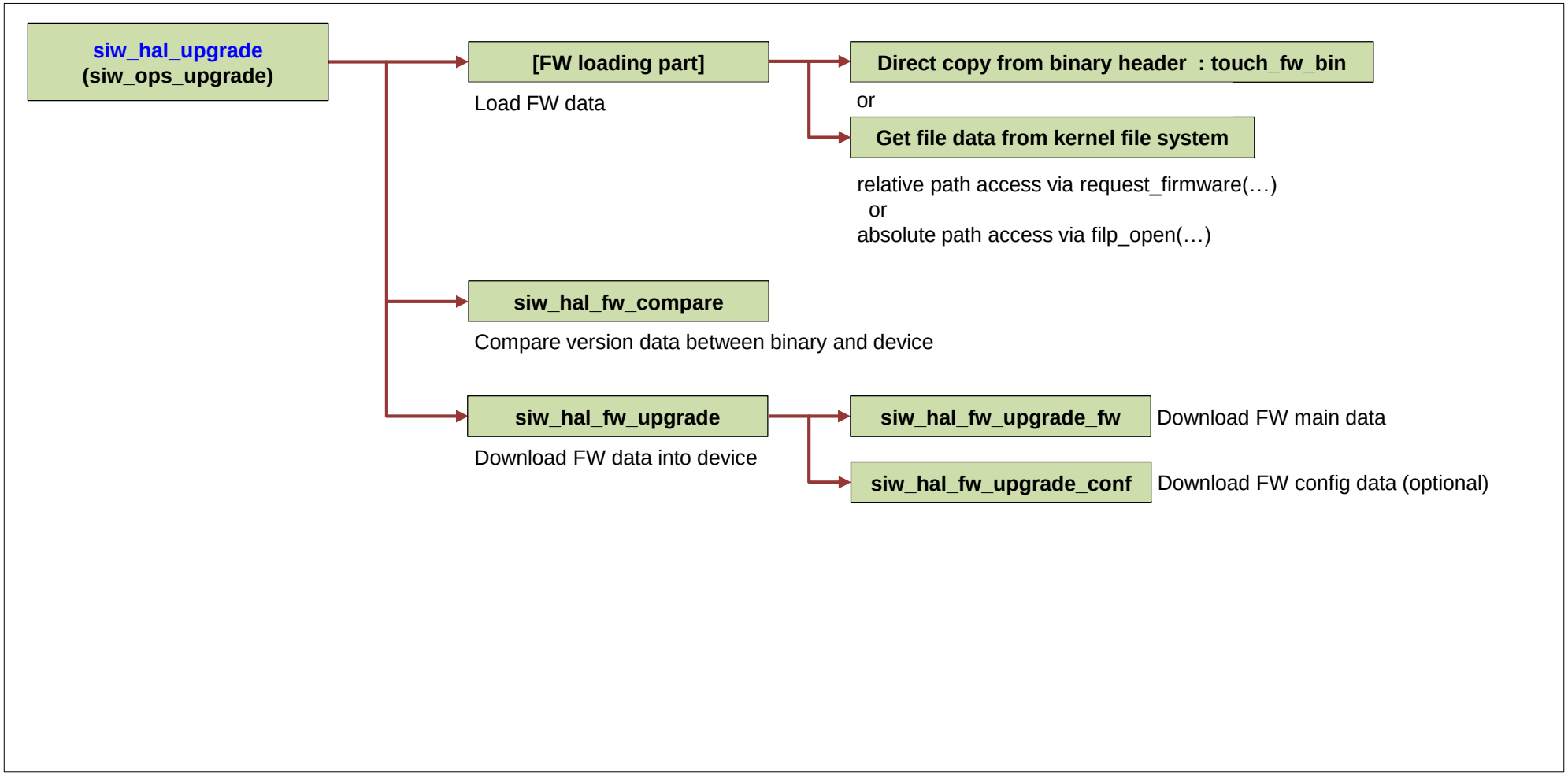
2. Else
2-1 Default upgrade (through version comparison)
do upgarde using request_firmware (relative path)
2-2 echo /.../fw_img > fw_upgrade
do force-upgrade using **request_firmware** (relative path)
2-3 echo {root}/.../fw_img > fw_upgrade
do force-upgrade using **normal file open** control (absolute path)

[Fig. 1-12-1] FW Upgrade (1/2)

1. Driver Operation

1.3 Operation

(3) FW Upgrade (hal layer)



[Fig. 1-12-2] FW upgrade (2/2)

2. Device Tree (example for I2C)

- Definition of I2C client device for SW49501 (refer to DTS example files for more information)

```
&i2c_1 {                                  // indicates parent device : I2C_1 adapter block
   sw17700@28 {                           // define new client device(sw17700) and slave addr. is 0x28
      status = "okay";
      compatible = "siw,sw17700";         // compatible name (see touch_xxxxxx.c)
      reg = <0x28>;                       // slave addr. : 0x28
      interrupt-parent = <&gpx1>;         // interrupt source : GPIO group gpx1
      interrupts = <6 0x02>;              // index 6(0~7) in gpx1 external interrupts
      irqflags = <0x2002>;                // IRQF_ONESHOT(0x2000) | IRQF_TRIGGER_FALLING(0x2)
      chip_flags = <0>;
      reset-gpio = <&gpx1 7 GPIO_ACTIVE_LOW>;          // index 7 in gpx1
      irq-gpio = <&gpx1 6 GPIO_ACTIVE_LOW>;            // index 6 in gpx1

      /* Caps */
      max_x = <1920>;
      max_y = <720>;
      max_pressure = <0xff>;
      max_width = <15>;
      max_orientation = <1>;
      max_id = <10>;
      /* role */
      hw_reset_delay = <210>;
      sw_reset_delay = <90>;
      use_lpwg = <0>;
      use_lpwg_test = <0>;
      /* firmware */
      use_firmware = <1>;          // enable firmware control
      use_fw_upgrade = <1>;        // auto-update during driver initialization
      fw_image = "siw/sw17700/LA103WF5_0_01.img";          // in android -> /lib/firmware/siw/..
      //fw_image = “{root}/sdcard/siw/sw17700/LA103WF5_0_01.img”;   // absolute path
      …
   };
};
```

[gpx1 definition in exynos5422 pinctrl device tree]

```
...
   pinctrl@13400000 {
   …
      gpx1: gpx1 {
         …
                  interrupt-controller;
                  interrupt-parent = <&combiner>;
                  #interrupt-cells = <2>;
                  interrupts = <28 0>, <28 1>, <29 0>, <29 1>,
                               <30 0>, <30 1>, <31 0>, <31 1>;
      };
      …
   };
…
```

- This example has been established based on odroidx-xu4(exynos5422) platform
- The detail configuration shall be modified up to main chipset.

2. Device Tree (example for SPI)

- Definition of SPI client device for SW42902 (refer to DTS example files for more information) (1/2)

```
&spi_1 {                                    // indicates parent device : SPI_1 block
   status = “okay”;
   samsung,spi-src-clk = <0>;
   num-cs = <1>;

   sw42902@0 {                              // define new spi device(SW42902)
      status = "okay";
      compatible = "siw,sw42902";           // compatible name (see touch_xxxxxx.c)
      reg = <0>;
      interrupt-parent = <&gpx1>;           // interrupt source : GPIO group gpx1
      interrupts = <6 0x02>;                // index 6(0~7) in gpx1 external interrupts
      irqflags = <0x2002>;                  // IRQF_ONESHOT(0x2000) | IRQF_TRIGGER_FALLING(0x2)
      chip_flags = <0>;
      reset-gpio = <&gpx1 7 GPIO_ACTIVE_LOW>;          // index 7 in gpx1
      irq-gpio = <&gpx1 6 GPIO_ACTIVE_LOW>;            // index 6 in gpx1

      /* Caps */
      max_x = <1440>;
      max_y = <3120>;
      max_pressure = <0xff>;
      max_width = <15>;
      max_orientation = <1>;
      max_id = <10>;
      /* role */
      hw_reset_delay = <100>;
      sw_reset_delay = <90>;
      use_lpwg = <0>;
      use_lpwg_test = <0>;
      /* firmware */
      use_firmware = <1>;          // enable firmware control
      use_fw_upgrade = <1>;        // auto-update during driver initialization
      fw_image = “siw/sw42902/B3W68SIME_0_01.img";          // in android -> /lib/firmware/siw/..
      …
```

- This example has been established based on odroidx-xu4(exynos5422) platform
- The detail configuration shall be modified up to main chipset.

Appendix-1. Reset Control

A1.1 HW Control

* The reset control of touch chipset is highly recommended to use a general GPIO connected to TCH_RSTn.

```
<siw_touch_of.c>
pin_val = siw_touch_of_gpio(dev, np, "reset-gpio", &pin_flags);
pins->reset_pin = pin_val;
```

```
<siw_touch_hal.c>
static void __siw_hal_init_gpio_reset(struct device *dev)
{
    ...
    ret = siw_touch_gpio_init(dev, reset_pin, SIW_HAL_GPIO_RST);    // ➔ gpio_request
    ...
}

static void __siw_hal_set_gpio_reset(struct device *dev, int val)
{
    ...
    siw_touch_gpio_direction_output(dev, reset_pin, !!(val));       // ➔ gpio_direction_output
    ...
}
```

A1.2 SW Reset

But, there are some systems are not designed to use TCH_RSTn as default configuration.
In this case, HW reset works only as Power-On-Reset and there is no way to support HW reset by GPIO.
The second way to do reset operation is SW reset and this requires adding a flag setup : TOUCH_SKIP_RESET_PIN

```
<touch_xxxxxxx.c>
#define CHIP_FLAGS              (0 |            \
                                 TOUCH_SKIP_RESET_PIN |          \
                                 0)
```

This flag enables SW reset instead of HW reset control skipping reset gpio control.

Probe	siw_touch 5-0028: start dts parsing siw_touch 5-0028: reset pin ignored … siw_touch 5-0028: hw_reset_quirk activated
Runtime	siw_touch 5-0028: HW Reset(Async) siw_touch 5-0028: run sw reset (reset gpio deactivated)

[Log Example : When TOUCH_SKIP_RESET_PIN is enabled]

Appendix-1. Reset Control

A1.3 Custom Interface

* Touch driver supports an additional interface for customized touch reset.

```
<siw_touch.h>
struct siw_touch_fquirks {      //function quirks
    ...
    int (*gpio_init_reset)(struct device *dev);           // siw_hal_init_gpio_reset > fquirks→gpio_init_reset
    int (*gpio_free_reset)(struct device *dev);           // siw_hal_free_gpio_reset > fquirks→gpio_free_reset
    int (*gpio_set_reset)(struct device *dev, int val);   // siw_hal_set_gpio_reset  > fquirks→gpio_set_reset
    ...
};
```

```
<touch_xxxxxxx.c>
static int __custom_gpio_init_reset(struct device *dev)
{
    return 0;
}

static int __custom_gpio_free_reset(struct device *dev)
{
    return 0;
}

static int __custom_gpio_set_reset(struct device *dev, int val)
{
    /*
     * If val == GPIO_OUT_ZERO, set TCH_RSTn low
     * If val == GPIO_OUT_ONE, set TCH_RSTn high
     */
    return 0;
}

static const struct siw_touch_pdata chip_pdata = {
    ...
    .fquirks = {
        .gpio_init_reset    = __custom_gpio_init_reset,
        .gpio_free_reset    = __custom_gpio_free_reset,
        .gpio_set_reset     = __custom_gpio_set_reset,
    },
};
```