**Test 1: -**

Hardware Sample – B413, High

Test setup – No code change

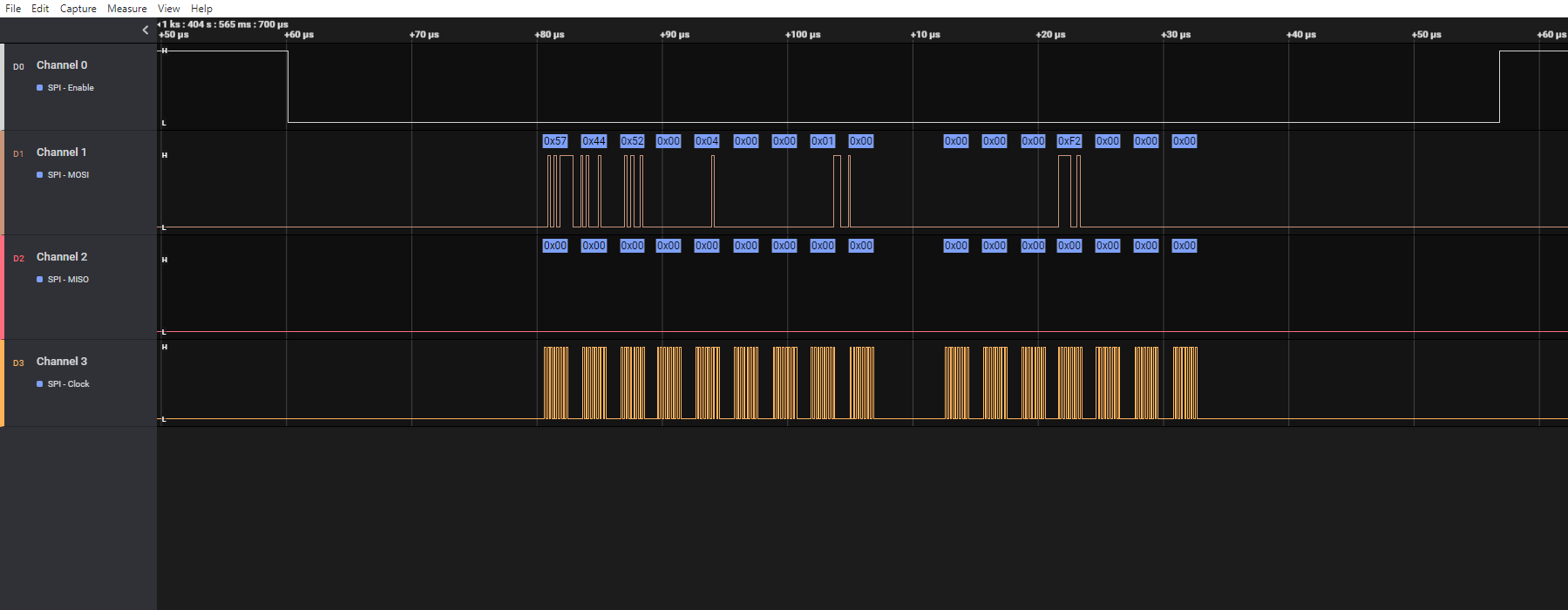


Figure 1– SPI Get version command or MOSI

A picture containing graphical user interface

Description automatically generated

Figure 2 – Tuner Reply or MISO

Status – Data Read OK

**Test 2: -**

Hardware Sample – B413, High

Test setup - FORCE bit disabled using “dynamic\_spi\_node” driver. In PNP driver, in function dab\_read(), following code was added to toggle GPIO.

dab\_read()

{

while(1){

dab\_cs\_assert(dabspi->cs\_gpio);

udelay(100);

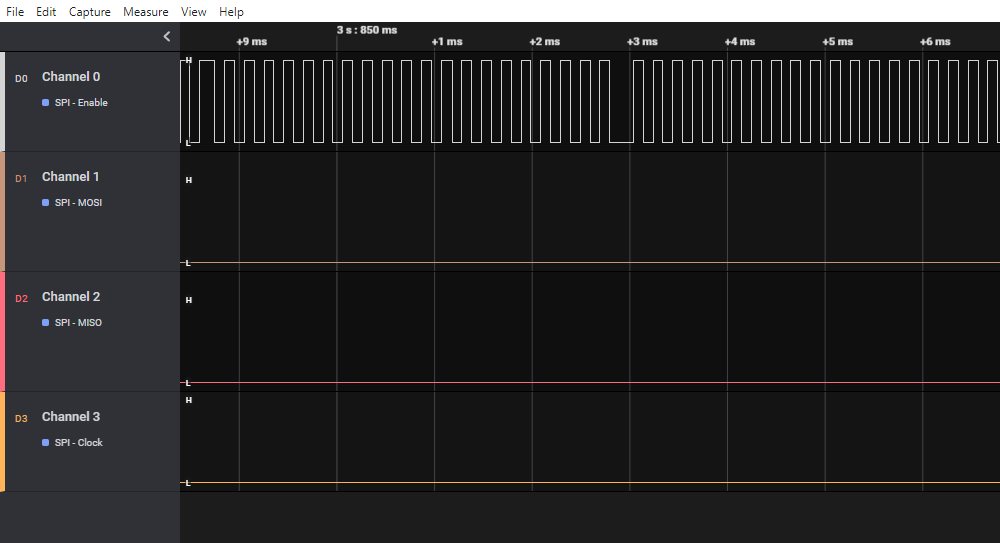
dab\_cs\_deassert(dabspi->cs\_gpio);

udelay(100);

}

}

Result: -



Conclusion 1: - This confirms that SPI\_CS is toggling due to explicit GPIO toggling in PnP driver.

**Test 3: -**

Hardware Sample – B413, High

Test setup – FORCE bit masked using “dynamic\_spi\_node” driver

Test method: -

After loading the PnP driver, when test app was ran then below waveform came and test app was blocked at first read () call in main () function.

Then SPI\_CS became permanently high.

Diagram

Description automatically generated with low confidence

Graphical user interface

Description automatically generated with low confidence

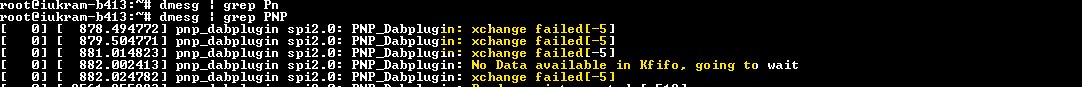


Figure – Expanded view

The question is when FORCE bit is masked, who togged CS as above.

At this stage, number of interrupt counts are only 4, as shown in figure below.

By taking kernel log, it is clear that it is failing at function spi\_xchange() of PNP driver in interrupt routine in response to interrupt pulse coming from tuner.



**Source code in PNP driver –**

static irqreturn\_t send\_spi\_data(int irq, void \*data)

{

dab\_cs\_assert(dabspi->cs\_gpio);

dabspi->txlen = 12;

res = spi\_xchange(dabspi->spi, NULL, dabspi->rx\_buf, dabspi->txlen);

if (res) {

dev\_err(&dabspi->spi->dev, "PNP\_Dabplugin: xchange failed[%d]\n", res);

goto ret;

}

res = dab\_fill\_kfifo(dabspi);

if (res)

dev\_err(&dabspi->spi->dev, "PNP\_Dabplugin: Error %s\n", \_\_func\_\_);

ret:

dab\_cs\_deassert(dabspi->cs\_gpio);

return IRQ\_HANDLED;

}

**Code flow: -**

spi\_xchange() - - > spi\_sync () - - > \_\_spi\_pump\_messages( ) - - > spi\_transfer\_one\_message - - >

omap2\_mcspi\_transfer\_one() - - > omap2\_mcspi\_work\_one() - - > omap2\_mcspi\_txrx\_pio()

**Kernel Log: -**

*[ 0] [ 164.014929] [<80011ac1>] (show\_stack) from [<801be8df>] (dump\_stack+0x87/0xac)*

*[ 0] [ 164.014946] [<801be8df>] (dump\_stack) from [<802a02ed>] (\_\_spi\_pump\_messages+0x2f9/0x54c)*

*[ 0] [ 164.014960] [<802a02ed>] (\_\_spi\_pump\_messages) from [<802a0721>] (\_\_spi\_sync+0x1d1/0x1d8)*

*[ 0] [ 164.014980] [<802a0721>] (\_\_spi\_sync) from [<7f81c13b>] (spi\_xchange+0x52/0x6c [pnp\_dabplugin])*

*[ 0] [ 164.014999] [<7f81c13b>] (spi\_xchange [pnp\_dabplugin]) from [<7f81c371>] (send\_spi\_data+0x34/0x80 [pnp\_dabplugin])*

*[ 0] [ 164.015017] [<7f81c371>] (send\_spi\_data [pnp\_dabplugin]) from [<80066485>] (irq\_thread\_fn+0x15/0x38)*

*[ 0] [ 164.015031] [<80066485>] (irq\_thread\_fn) from [<800666d3>] (irq\_thread+0xfb/0x18c)*

*[ 0] [ 164.015044] [<800666d3>] (irq\_thread) from [<8003e297>] (kthread+0xaf/0xc4)*

*[ 0] [ 164.015058] [<8003e297>] (kthread) from [<8000eb8d>] (ret\_from\_fork+0x11/0x24)*

*[ 0] [ 164.015067] MIR: In function - \_\_spi\_pump\_messages, line no - 1164*

*[ 0] [ 164.015074] HARMAN: Force bit is masked*

*[ 0] [ 165.024700] pnp\_dabplugin spi2.0: RXS timed out*

*[ 0] [ 165.024721] pnp\_dabplugin spi2.0: SPI transfer failed: -5*

*[ 0] [ 165.024733] HARMAN: Force bit is masked*

*[ 0] [ 165.024750] spi\_master spi2: failed to transfer one message from queue*

*[ 0] [ 165.024763] pnp\_dabplugin spi2.0: PNP\_Dabplugin: xchange failed[-5]*

*[ 0] [ 165.733989] CPU: 0 PID: 1686 Comm: irq/243-pnp\_dab Not tainted 4.4.14 #Rel\_Elina\_J6\_IUKRAM\_21465B-21-gf1f4b3213295*

*[ 0] [ 165.734007] Hardware name: Generic DRA72X (Flattened Device Tree)*

*[ 0] [ 165.734038] [<80014501>] (unwind\_backtrace) from [<80011ac1>] (show\_stack+0x11/0x14)*

*[ 0] [ 165.734058] [<80011ac1>] (show\_stack) from [<801be8df>] (dump\_stack+0x87/0xac)*

*[ 0] [ 165.734078] [<801be8df>] (dump\_stack) from [<802a02ed>] (\_\_spi\_pump\_messages+0x2f9/0x54c)*

*[ 0] [ 165.734093] [<802a02ed>] (\_\_spi\_pump\_messages) from [<802a0721>] (\_\_spi\_sync+0x1d1/0x1d8)*

*[ 0] [ 165.734116] [<802a0721>] (\_\_spi\_sync) from [<7f81c13b>] (spi\_xchange+0x52/0x6c [pnp\_dabplugin])*

*[ 0] [ 165.734135] [<7f81c13b>] (spi\_xchange [pnp\_dabplugin]) from [<7f81c371>] (send\_spi\_data+0x34/0x80 [pnp\_dabplugin])*

*[ 0] [ 165.734156] [<7f81c371>] (send\_spi\_data [pnp\_dabplugin]) from [<80066485>] (irq\_thread\_fn+0x15/0x38)*

*[ 0] [ 165.734171] [<80066485>] (irq\_thread\_fn) from [<800666d3>] (irq\_thread+0xfb/0x18c)*

*[ 0] [ 165.734184] [<800666d3>] (irq\_thread) from [<8003e297>] (kthread+0xaf/0xc4)*

*[ 0] [ 165.734200] [<8003e297>] (kthread) from [<8000eb8d>] (ret\_from\_fork+0x11/0x24)*

*[ 0] [ 165.734209] MIR: In function - \_\_spi\_pump\_messages, line no - 1164*

*[ 0] [ 165.734218] HARMAN: Force bit is masked*

*[ 0] [ 166.734688] pnp\_dabplugin spi2.0: RXS timed out*

*[ 0] [ 166.734702] pnp\_dabplugin spi2.0: SPI transfer failed: -5*

*[ 0] [ 166.734709] HARMAN: Force bit is masked*

*[ 0] [ 166.734738] spi\_master spi2: failed to transfer one message from queue*

*[ 0] [ 166.734748] pnp\_dabplugin spi2.0: PNP\_Dabplugin: xchange failed[-5]*

*[ 0] [ 166.734769] CPU: 0 PID: 1686 Comm: irq/243-pnp\_dab Not tainted 4.4.14 #Rel\_Elina\_J6\_IUKRAM\_21465B-21-gf1f4b3213295*

*[ 0] [ 166.734777] Hardware name: Generic DRA72X (Flattened Device Tree)*

*[ 0] [ 166.734792] [<80014501>] (unwind\_backtrace) from [<80011ac1>] (show\_stack+0x11/0x14)*

*[ 0] [ 166.734807] [<80011ac1>] (show\_stack) from [<801be8df>] (dump\_stack+0x87/0xac)*

*[ 0] [ 166.734820] [<801be8df>] (dump\_stack) from [<802a02ed>] (\_\_spi\_pump\_messages+0x2f9/0x54c)*

*[ 0] [ 166.734832] [<802a02ed>] (\_\_spi\_pump\_messages) from [<802a0721>] (\_\_spi\_sync+0x1d1/0x1d8)*

*[ 0] [ 166.734846] [<802a0721>] (\_\_spi\_sync) from [<7f81c13b>] (spi\_xchange+0x52/0x6c [pnp\_dabplugin])*

*[ 0] [ 166.734863] [<7f81c13b>] (spi\_xchange [pnp\_dabplugin]) from [<7f81c371>] (send\_spi\_data+0x34/0x80 [pnp\_dabplugin])*

*[ 0] [ 166.734880] [<7f81c371>] (send\_spi\_data [pnp\_dabplugin]) from [<80066485>] (irq\_thread\_fn+0x15/0x38)*

*[ 0] [ 166.734893] [<80066485>] (irq\_thread\_fn) from [<800666d3>] (irq\_thread+0xfb/0x18c)*

*[ 0] [ 166.734904] [<800666d3>] (irq\_thread) from [<8003e297>] (kthread+0xaf/0xc4)*

*[ 0] [ 166.734916] [<8003e297>] (kthread) from [<8000eb8d>] (ret\_from\_fork+0x11/0x24)*

*[ 0] [ 166.734925] MIR: In function - \_\_spi\_pump\_messages, line no - 1164*

*[ 0] [ 166.734932] HARMAN: Force bit is masked*

*[ 0] [ 167.744690] pnp\_dabplugin spi2.0: RXS timed out*

*[ 0] [ 167.744710] pnp\_dabplugin spi2.0: SPI transfer failed: -5*

*[ 0] [ 167.744721] HARMAN: Force bit is masked*

*[ 0] [ 167.744736] spi\_master spi2: failed to transfer one message from queue*

*[ 0] [ 167.744748] pnp\_dabplugin spi2.0: PNP\_Dabplugin: xchange failed[-5]*

*[ 0] [ 167.836026] pnp\_dabplugin spi2.0: PNP\_Dabplugin: No Data available in Kfifo, going to wait*

*[ 0] [ 183.047580] pnp\_dabplugin spi2.0: PNP\_Dabplugin: Read was interrupted [-512]*

From above log, it is found that the below function is failing on masking FORCE bit.

ret = master->transfer\_one\_message(master, master->cur\_msg);

static int spi\_master\_initialize\_queue(struct spi\_master \*master)

{

…..

if (!master->transfer\_one\_message)

master->transfer\_one\_message = spi\_transfer\_one\_message;

………

}

static int spi\_transfer\_one\_message(struct spi\_master \*master, struct spi\_message \*msg)

{

spi\_set\_cs(msg->spi, true);

ret = master->transfer\_one(master, msg->spi, xfer);

if (ret < 0) {

dev\_err(&msg->spi->dev, "SPI transfer failed: %d\n", ret);

goto out;

}

}

Error - SPI transfer failed due to master->transfer\_one( )

**File – drivers/spi/spi-omap2-mcspi.c**

static int omap2\_mcspi\_probe(struct platform\_device \*pdev)

{

……….

master->transfer\_one = omap2\_mcspi\_transfer\_one;

……….

}

Hence, finally omap2\_mcspi\_transfer\_one() is failing due to spi\_sync() call from IRQ routine of Pnp driver. The function omap2\_mcspi\_transfer\_one() in turn calls the function omap2\_mcspi\_work\_one().

Then it calls omap2\_mcspi\_txrx\_pio().

**Actual failure: -**

spi\_xchange() - - > spi\_sync () - - > \_\_spi\_pump\_messages( ) - - > spi\_transfer\_one\_message - - >

omap2\_mcspi\_transfer\_one() - - > omap2\_mcspi\_work\_one() - - > omap2\_mcspi\_txrx\_pio()

*omap2\_mcspi\_txrx\_pio() {*

*if (mcspi\_wait\_for\_reg\_bit(chstat\_reg, OMAP2\_MCSPI\_CHSTAT\_RXS) < 0) {*

*dev\_err(&spi->dev, "RXS timed out\n");*

*goto out;*

*}*

*}*

static int mcspi\_wait\_for\_reg\_bit(void \_\_iomem \*reg, unsigned long bit)

{

unsigned long timeout;

timeout = jiffies + msecs\_to\_jiffies(1000);

while (!(readl\_relaxed(reg) & bit)) {

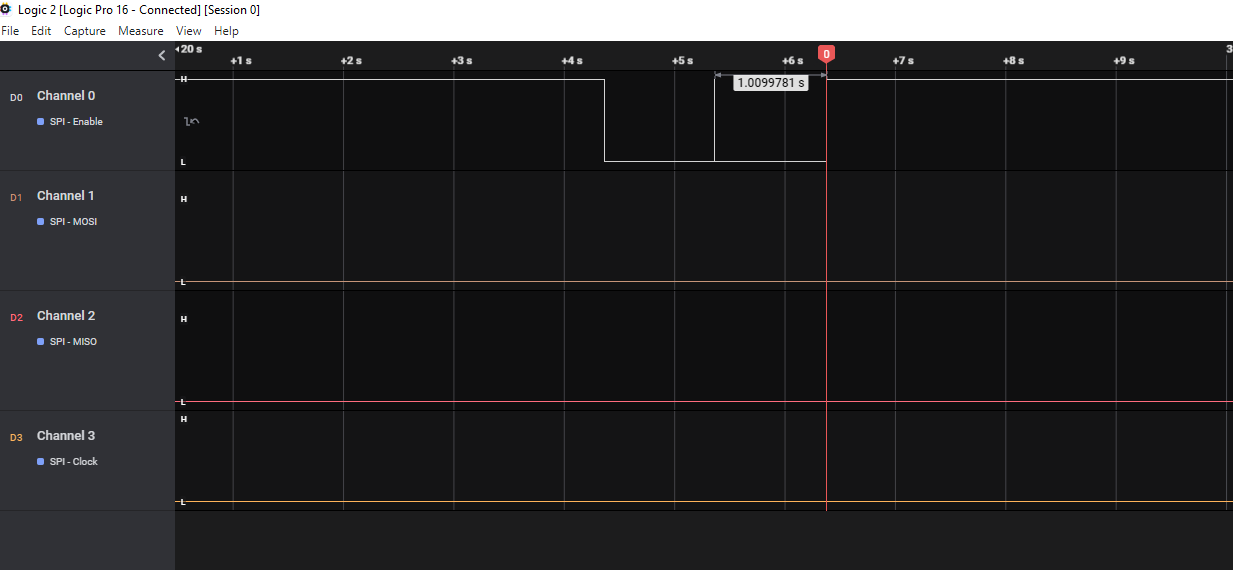
……

}

This error comes when data is not received with 1 second time out.

Hardware register - MCSPI\_CH0STAT, AM335X TRM Pg – 4942.

Observation: - Immediately after Tuner reset, it gives two pulses. But we can see from below figure is that SPI\_CLK does not generate. It is the reason it is failing due to “RXS” timeout of 1000ms.

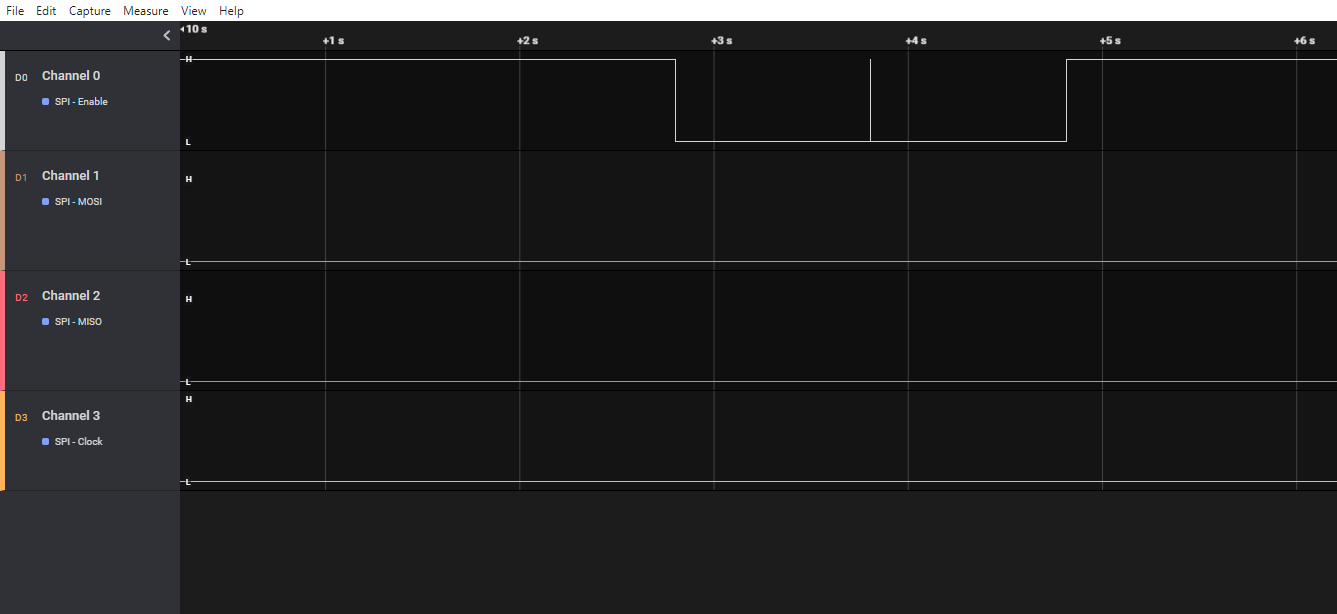


**Test 4: -**

Behavior after Reset.

Test setup: -

FORCE bit masked. Test app programmed to generate Reset.



**Test 5: -**

Behavior just after Power ON Reset.

Test setup: -

FORCE bit unmasked. GPIO toggling in PNP driver is commented.

Test app programmed to generate Reset.



**Test 6: -**

Behavior just after Power ON Reset.

Test setup: -

FORCE bit unmasked. GPIO toggling in PNP driver is enabled back.

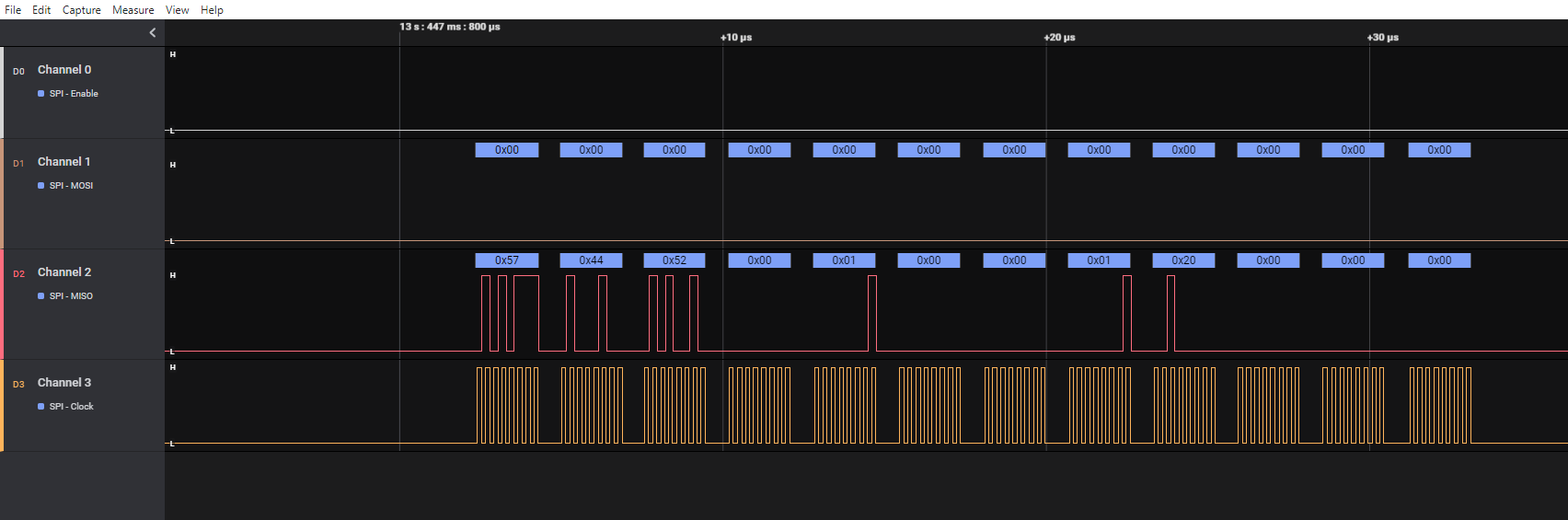
Test app programmed to generate Reset.

Table

Description automatically generated

Timeline

Description automatically generated



Conclusion: -

As SPI\_CS is set using Pin Muxing as GPIO4\_9 using DTS, so CS is toggling because of PNP driver

only. But when we mask FORCE bit then SPI\_CLK is not generated from OMAP SPI Master.

So, for SPI\_CS to work FORCE bit should be enabled, if GPIO is used for toggling the SPI\_CS.

As per kernel version – 4.19 below code was found for spi\_set\_cs().

File – drivers/spi/spi.c

static void spi\_set\_cs(struct spi\_device \*spi, bool enable)

{

if (spi->mode & SPI\_CS\_HIGH)

enable = !enable;

if (gpio\_is\_valid(spi->cs\_gpio)) {

gpio\_set\_value(spi->cs\_gpio, !enable);

/\* Some SPI masters need both GPIO CS & slave\_select \*/

if ((spi->controller->flags & SPI\_MASTER\_GPIO\_SS) &&

spi->controller->set\_cs)

spi->controller->set\_cs(spi, !enable);

} else if (spi->controller->set\_cs) {

spi->controller->set\_cs(spi, !enable);

}

}

File – drivers/spi/spi-omap2-mcspi.c

static int omap2\_mcspi\_work\_one(struct omap2\_mcspi \*mcspi,

struct spi\_device \*spi, struct spi\_transfer \*t)

{

……

if (gpio\_is\_valid(spi->cs\_gpio))

omap2\_mcspi\_set\_cs(spi, spi->mode & SPI\_CS\_HIGH);

}

From above function it is clear that even if we use DTS property “cs-gpio = < &gpio4 9 >, the omap FORCE bit is getting changed accordingly for CS to toggle.

In our case, PNP driver is toggling GPIO but as we are not using DTS property cs-gpio, so SPI core’s function spi\_set\_cs() is choosing omap2\_mcspi\_set\_cs() to toggle FORCE bit.

**Note: -**

GPIO CS pin muxing for SPI for PNP DAB tuner.

For B413 IUK Ram High variant hardware the dts file is “arch/arm/boot/dts/

j6eco\_inkram\_high\_generic. dtsi”. In this “dab-cs-pinmux” is defined as below.

*dabplugin {*

*dab-cs-gpio = <115>;*

*dab-irq-gpio = <230>;*

*dab-cs-pinmux = <0x4A0035C8 0x0006000E>;*

*dab-pm-disable;*

*};*

From page number - 71 of DRA74x data sheet:

Ball No – D11 is “vout1\_clk” by default and it can be multiplexed to behave as “spi3\_cs0”

in Mux-Mode = 8 and it can behave as “gpio4\_9” in Mux-Mode = 14.

From page number – 4429 of DRA74x TRM:

Control module core base address for “CTRL\_CORE\_PAD\_VOUT1\_CLK” be 0x4A00 35C8.

As per “Pad Configuration Register” in page – 4376, the ball D11 is set to behave as “GPIO4\_9”

and receiver is enabled to work in bidirectional mode.

