**Observations of Linux SPI Device driver in TI SOC: -**

1. **If do not define “reg” address property in dts then what will happen?**

Ans – “reg” property in dts for addressing SPI slave is mandatory. If we do not give this property in child node of dts, then probe() function will fail.

So bare minimum dts for spi slave device for probe to succeed is as below.

spi\_flash@0 {

compatible = "mir,mirflash";

spi-max-frequency = <1000000>;

reg = <0>;

};

1. **Validate usage of “cs-gpios” dts property for SPI controller.**

Ex –

&spi1 {

pinctrl-names = "default";

pinctrl-0 = <&spi1\_pins>;

status = "okay";

cs-gpios = <0>,<&gpio1 17 0>;

spi\_flash@1 {

compatible = "mir,mirflash";

spi-max-frequency = <1000000>;

reg = <1>;

};

};

In above dts I have used “cs-gpios” property. So, reg = <1> is mapped to GPIO1\_17 and reg =<0> if used will map to native CS of controller i.e. SPI\_CS0.

To verify reg=<1> is mapped to GPIO1\_17, I probed GPIO1\_17 pin and one more important point is that if GPIO is used as CS, then it will be toggled by SPI core by its own and there is no need of explicitly toggle the GPIO CS by client driver.

Other example:-

I further modified dts as below to convert other Gpio0\_7 as chip select. And probed it, this new chip select is also toggling by its own.

&spi1 {

pinctrl-names = "default";

pinctrl-0 = <&spi1\_pins>;

status = "okay";

cs-gpios = <0>,<&gpio1 17 0>,<&gpio0 7 0>;

spi\_flash@2 {

compatible = "mir,mirflash";

spi-max-frequency = <1000000>;

reg = <2>;

};

};

CRO waveform in this test is as below.

![A screenshot of a computer

Description automatically generated with medium confidence]()

Fugure – 1

Above thing is happening due to spi\_set\_cs() function in 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);

}

}

[TODO – verify that is it that both native cs and gpio taken as chip select should not toggle.]

3.**Verify that the member “chip\_select” of struct spi\_board\_info is equivalent to dts “reg” property.**

4. **omap2\_set\_cs() is commented in function omap2\_mcspi\_transfer\_one() , even then it is called from which function?**

To find this stack back trace was taken by calling dump\_stack() in function omap2\_set\_cs(). The stack trace is as below.

101.099940] [<c010d034>] (show\_stack) from [<c0916a2c>] (dump\_stack+0xe0/0x114)

[ 101.107301] [<c0916a2c>] (dump\_stack) from [<c06c954c>] (omap2\_mcspi\_set\_cs+0x20/0xd0)

[ 101.115283] [<c06c954c>] (omap2\_mcspi\_set\_cs) from [<c06c5110>] (spi\_transfer\_one\_message+0x594/0x5f8)

[ 101.124645] [<c06c5110>] (spi\_transfer\_one\_message) from [<c06c5574>] (\_\_spi\_pump\_messages+0x400/0x864)

[ 101.134094] [<c06c5574>] (\_\_spi\_pump\_messages) from [<c06c5c24>] (\_\_spi\_sync+0x240/0x2f0)

[ 101.142320] [<c06c5c24>] (\_\_spi\_sync) from [<c06c5cfc>] (spi\_sync+0x28/0x40)

[ 101.149421] [<c06c5cfc>] (spi\_sync) from [<c0771fb4>] (send\_write\_data+0x80/0xc4)

[ 101.156953] [<c0771fb4>] (send\_write\_data) from [<c07720dc>] (mir\_write+0xe4/0x128)

[ 101.164666] [<c07720dc>] (mir\_write) from [<c02ef210>] (vfs\_write+0xa0/0x17c)

[ 101.171848] [<c02ef210>] (vfs\_write) from [<c02ef478>] (ksys\_write+0x5c/0xe8)

[ 101.179028] [<c02ef478>] (ksys\_write) from [<c0101000>] (ret\_fast\_syscall+0x0/0x28)

[ 101.186727] Exception stack(0xdd53bfa8 to 0xdd53bff0)

$ ./scripts/faddr2line vmlinux spi\_transfer\_one\_message+0x2c

O/p - spi\_transfer\_one\_message+0x2c/0x5cc:

spi\_transfer\_one\_message at drivers/spi/spi.c:1017

Conclusion – omap2\_set\_cs function is also called from spi core i.e. drivers/spi/spi.c from function spi\_transfer\_one\_message(struct spi\_controller \*ctlr, struct spi\_message \*msg){

.

.

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

.

.

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

.

.

}

If we see SPI controller drivers probe function i.e. omap2\_mcspi\_probe() then we find that there is assignment of or registration of low level functions to spi core.

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

{

.

.

master->set\_cs = omap2\_mcspi\_set\_cs;

.

.

}

==========🡺

The function omap2\_mcspi\_transfer\_one() is the function which has code to clear SINGLE bit of MODUL\_CTRL register to enable Chip Select Per Byte.

[ 62.290420] [<c091680c>] (dump\_stack) from [<c06c8828>] (omap2\_mcspi\_transfer\_one+0x50/0x1150)

[ 62.299087] [<c06c8828>] (omap2\_mcspi\_transfer\_one) from [<c06c4c6c>] (spi\_transfer\_one\_message+0xf0/0x5f8)

[ 62.308885] [<c06c4c6c>] (spi\_transfer\_one\_message) from [<c06c5574>] (\_\_spi\_pump\_messages+0x400/0x864)

[ 62.318334] [<c06c5574>] (\_\_spi\_pump\_messages) from [<c06c5c24>] (\_\_spi\_sync+0x240/0x2f0)

[ 62.326561] [<c06c5c24>] (\_\_spi\_sync) from [<c06c5cfc>] (spi\_sync+0x28/0x40)

[ 62.333655] [<c06c5cfc>] (spi\_sync) from [<c0771d94>] (send\_write\_data+0x80/0xc4)

[ 62.341186] [<c0771d94>] (send\_write\_data) from [<c0771ebc>] (mir\_write+0xe4/0x128)

[ 62.348891] [<c0771ebc>] (mir\_write) from [<c02ef210>] (vfs\_write+0xa0/0x17c)

[ 62.356072] [<c02ef210>] (vfs\_write) from [<c02ef478>] (ksys\_write+0x5c/0xe8)

[ 62.363252] [<c02ef478>] (ksys\_write) from [<c0101000>] (ret\_fast\_syscall+0x0/0x28)

[ 62.370950] Exception stack(0xdac69fa8 to 0xdac69ff0)

[ 62.376035] 9fa0: 00000074 00000000 00000003 00021048 00000010 00000000

[ 62.384261] 9fc0: 00000074 00000000 00000000 00000004 00000000 00000000 b6f04000 becc9c7c

[ 62.392484] 9fe0: 00000004 becc9c48 b6e6f5b3 b6df8746

[ 62.397684] MIR: Value of OMAP2\_MCSPI\_CHCONF - 200123fc

[ 62.402948] MIR: Clearing SINGLE bit for CS\_Per\_Word

**5. omap2\_mcspi\_probe() function call back trace:-**

[ 1.424675] [<c0916dcc>] (dump\_stack) from [<c06c99e8>] (omap2\_mcspi\_probe+0x38/0x560)

[ 1.432651] [<c06c99e8>] (omap2\_mcspi\_probe) from [<c06190cc>] (platform\_drv\_probe+0x48/0x98)

[ 1.441230] [<c06190cc>] (platform\_drv\_probe) from [<c0616f90>] (really\_probe+0x21c/0x2c0)

[ 1.449545] [<c0616f90>] (really\_probe) from [<c0617194>] (driver\_probe\_device+0x5c/0x160)

[ 1.457858] [<c0617194>] (driver\_probe\_device) from [<c061738c>] (\_\_driver\_attach+0xf4/0xf8)

[ 1.466346] [<c061738c>] (\_\_driver\_attach) from [<c06151a4>] (bus\_for\_each\_dev+0x74/0xc0)

[ 1.474572] [<c06151a4>] (bus\_for\_each\_dev) from [<c0616358>] (bus\_add\_driver+0x170/0x204)

[ 1.482886] [<c0616358>] (bus\_add\_driver) from [<c0618130>] (driver\_register+0x74/0x108)

[ 1.491030] [<c0618130>] (driver\_register) from [<c0103040>] (do\_one\_initcall+0x84/0x324)

[ 1.499263] [<c0103040>] (do\_one\_initcall) from [<c0d01254>] (kernel\_init\_freeable+0x3d8/0x4c4)

[ 1.508016] [<c0d01254>] (kernel\_init\_freeable) from [<c092b6f8>] (kernel\_init+0x8/0x110)

[ 1.516241] [<c092b6f8>] (kernel\_init) from [<c01010b4>] (ret\_from\_fork+0x14/0x20)

**6. SPI client driver’s probe function call back trace: -**

[ 1.919367] [<c0916dcc>] (dump\_stack) from [<c0772504>] (mir\_spi\_probe+0x1c/0x2a4)

[ 1.926998] [<c0772504>] (mir\_spi\_probe) from [<c06c2698>] (spi\_drv\_probe+0x7c/0xa0)

[ 1.934796] [<c06c2698>] (spi\_drv\_probe) from [<c0616f90>] (really\_probe+0x21c/0x2c0)

[ 1.942674] [<c0616f90>] (really\_probe) from [<c0617194>] (driver\_probe\_device+0x5c/0x160)

[ 1.950987] [<c0617194>] (driver\_probe\_device) from [<c061738c>] (\_\_driver\_attach+0xf4/0xf8)

[ 1.959475] [<c061738c>] (\_\_driver\_attach) from [<c06151a4>] (bus\_for\_each\_dev+0x74/0xc0)

[ 1.967701] [<c06151a4>] (bus\_for\_each\_dev) from [<c0616358>] (bus\_add\_driver+0x170/0x204)

[ 1.976015] [<c0616358>] (bus\_add\_driver) from [<c0618130>] (driver\_register+0x74/0x108)

[ 1.984158] [<c0618130>] (driver\_register) from [<c0103040>] (do\_one\_initcall+0x84/0x324)

[ 1.992391] [<c0103040>] (do\_one\_initcall) from [<c0d01254>] (kernel\_init\_freeable+0x3d8/0x4c4)

[ 2.001146] [<c0d01254>] (kernel\_init\_freeable) from [<c092b6f8>] (kernel\_init+0x8/0x110)

[ 2.009372] [<c092b6f8>] (kernel\_init) from [<c01010b4>] (ret\_from\_fork+0x14/0x20)

**7. omap2\_mcspi\_setup\_transfer() function call back trace:-**

[ 1.590400] [<c0916d0c>] (dump\_stack) from [<c06c94e4>] (omap2\_mcspi\_setup\_transfer+0x34/0x430)

[ 1.599149] [<c06c94e4>] (omap2\_mcspi\_setup\_transfer) from [<c06c8034>] (omap2\_mcspi\_setup+0x154/0x188)

[ 1.608608] [<c06c8034>] (omap2\_mcspi\_setup) from [<c06c33e4>] (spi\_setup+0xac/0x1d0)

[ 1.616487] [<c06c33e4>] (spi\_setup) from [<c06c3588>] (spi\_add\_device+0x80/0x158)

[ 1.624103] [<c06c3588>] (spi\_add\_device) from [<c06c3ea8>] (spi\_register\_controller+0x50c/0x9e0)

[ 1.633028] [<c06c3ea8>] (spi\_register\_controller) from [<c06c43ac>] (devm\_spi\_register\_controller+0x30/0x70)

[ 1.642998] [<c06c43ac>] (devm\_spi\_register\_controller) from [<c06c9d44>] (omap2\_mcspi\_probe+0x464/0x560)

[ 1.652624] [<c06c9d44>] (omap2\_mcspi\_probe) from [<c06190cc>] (platform\_drv\_probe+0x48/0x98)

[ 1.661200] [<c06190cc>] (platform\_drv\_probe) from [<c0616f90>] (really\_probe+0x21c/0x2c0)

[ 1.669513] [<c0616f90>] (really\_probe) from [<c0617194>] (driver\_probe\_device+0x5c/0x160)

[ 1.677826] [<c0617194>] (driver\_probe\_device) from [<c061738c>] (\_\_driver\_attach+0xf4/0xf8)

[ 1.686313] [<c061738c>] (\_\_driver\_attach) from [<c06151a4>] (bus\_for\_each\_dev+0x74/0xc0)

[ 1.694537] [<c06151a4>] (bus\_for\_each\_dev) from [<c0616358>] (bus\_add\_driver+0x170/0x204)

[ 1.702849] [<c0616358>] (bus\_add\_driver) from [<c0618130>] (driver\_register+0x74/0x108)

[ 1.710990] [<c0618130>] (driver\_register) from [<c0103040>] (do\_one\_initcall+0x84/0x324)

[ 1.719219] [<c0103040>] (do\_one\_initcall) from [<c0d01254>] (kernel\_init\_freeable+0x3d8/0x4c4)

[ 1.727971] [<c0d01254>] (kernel\_init\_freeable) from [<c092b638>] (kernel\_init+0x8/0x110)

[ 1.736195] [<c092b638>] (kernel\_init) from [<c01010b4>] (ret\_from\_fork+0x14/0x20)