**SPI Driver Basics: -**

1. **SPI Modes: -**

CPOL = 0 - > Idle state of SPI\_CLK is low

CPHA = 0 - > Data latched or shifted out on falling edge

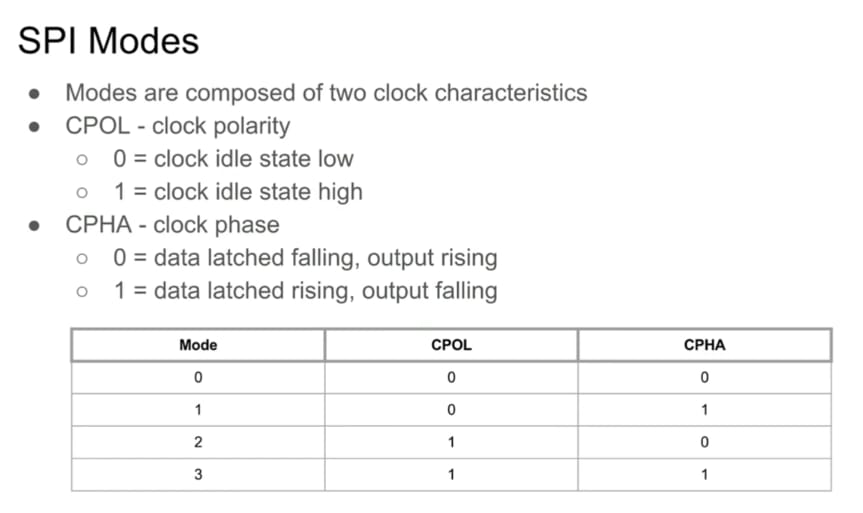


Figure – 1

**SPI Mode = 0: -**

In SPI Mode = 0, CPOL = 0 and CPHA = 0. So, idle state of clock is zero and since CPHA = 0, so data is latched or shifted out on falling edge and data is sampled or captured on rising edge of clock. It is shown in figure – 2 below.



Figure – 2: SPI\_MODE = 0

From figure – 2, it is clear that in SPI Mode = 0, data are latched or shifted out on falling edge and data are sampled or captured on clock’s rising edge.

**SPI Mode = 1: -**

In SPI Mode = 1, CPOL = 0 and CPHA = 1. So, idle state of clock is zero and since CPHA = 1, so data is latched or shifted out on rising edge and data is sampled or captured on falling edge of clock. It is shown in figure – 3 below.



Figure – 3: SPI\_MODE = 1

Diagram

Description automatically generated

If CPHA = 1, then data changes or latches at rising edge for both MOSI and MISO signal.

1. **Types of SPI: -**

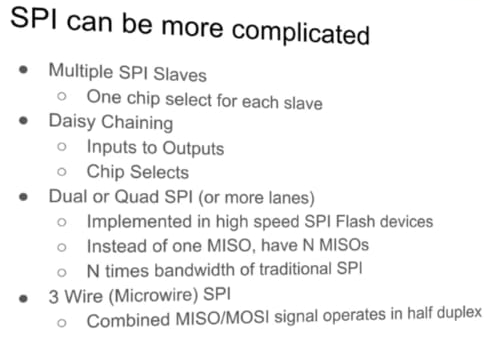


Figure – 2

1. **Linux SPI Driver: -**

There are two types of SPI driver –

1. SPI controller driver
2. SPI protocol driver

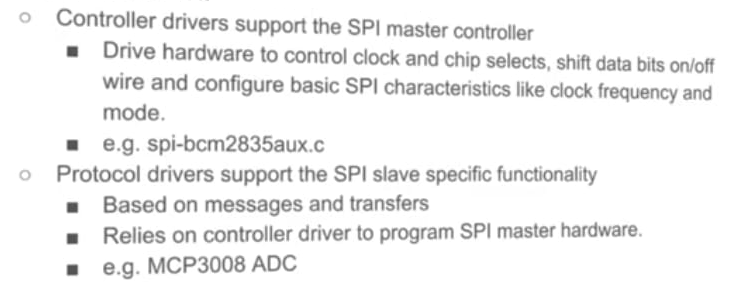


Figure – 3

1. **Linux SPI communication: -**

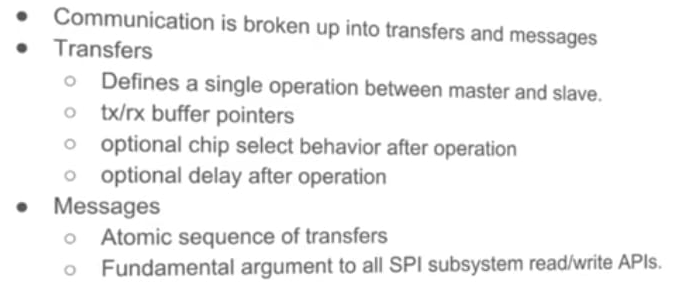


Figure – 4

struct spi\_message is basically argument to all SPI subsystem read/write APIs.

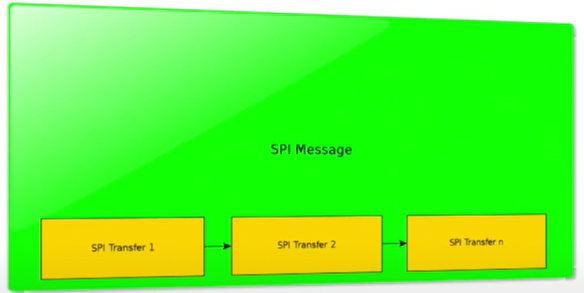


Figure – 5

1. **Adding a SPI device to a system: -**

**A picture containing text

Description automatically generated**

Figure - 6

1. **SPI device DT binding: -**

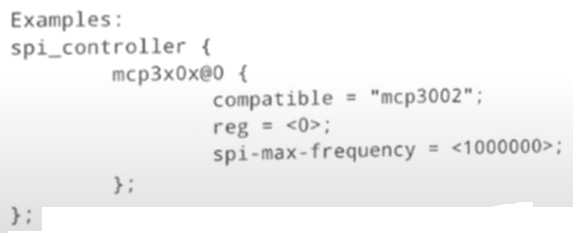
****

Figure – 7

1. **Adding SPI device using board file: -**

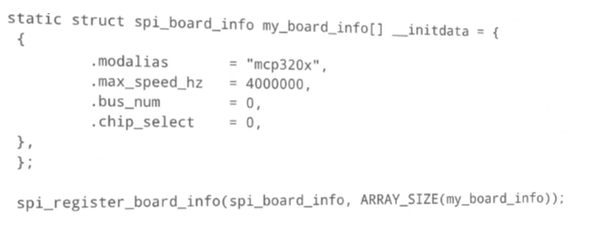
****

Figure - 8

1. **SPI protocol driver: -**

Follows standard driver model.

**Text

Description automatically generated**

Figure – 9

1. **SPI sub system kernel APIs: -**

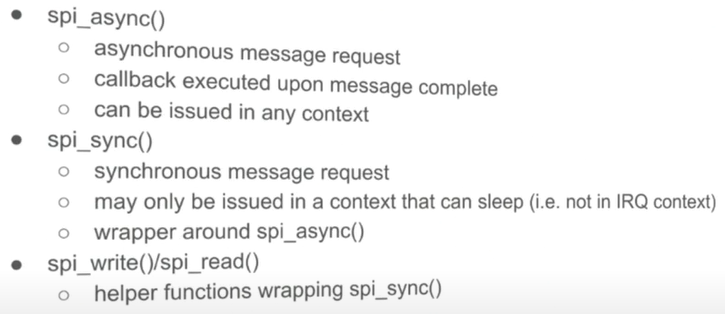
****

Figure - 10

**Graphical user interface, text, application, email

Description automatically generated**

Figure – 11

1. **SPI controller driver: -**

**Graphical user interface, text, application

Description automatically generated**

Figure – 12

transfer\_one model allows us to let the core support GPIO chip select.

1. **User space driver - spidev: -**

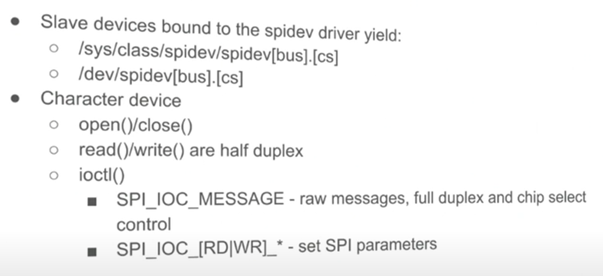
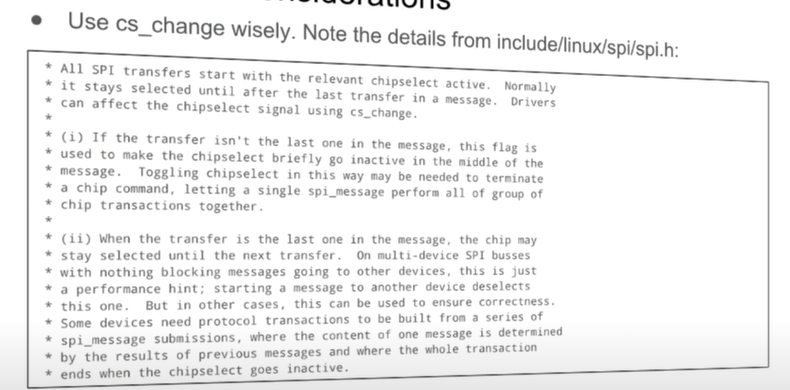


Figure – 13

SPI is inherently a full duplex bus.

1. **“spi\_transfer” cs\_change flag: -**

****

1. **Performance considerations: -**

Graphical user interface, text, application

Description automatically generated

1. **Performance tools: -**

**Graphical user interface, text

Description automatically generated**

1. **SPI core API: -**

Source code – drivers/spi/spi.c

1. spi\_transfer\_one\_message(): -

This is a standard implementation of transfer\_one\_message() for drivers which implement a transfer\_one() operation.

It does following things: -

* It takes struct spi\_message as argument. As we know that spi\_message is having multiple spi\_transfer.

This function iterates through various “struct spi\_transfer” in “struct spi\_message” and then it invokes lower-level function of controller for transfer to take place.

static int spi\_transfer\_one\_message(struct spi\_controller \*ctlr,

struct spi\_message \*msg)

{

list\_for\_each\_entry(xfer, &msg->transfers, transfer\_list) {

…..

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

…..

= = > wait for a timeout for transfer completion

}

* This function also introduces some delay after each transfer, by using following piece of code.

struct spi\_transfer \*xfer;

f (xfer->delay\_usecs) {

u16 us = xfer->delay\_usecs;

if (us <= 10)

udelay(us);

else

usleep\_range(us, us + DIV\_ROUND\_UP(us, 10));

}

* This function is also responsible for setting CS line as per user choice as below.

if (xfer->cs\_change) {

if (list\_is\_last(&xfer->transfer\_list,

&msg->transfers)) {

keep\_cs = true;

} else {

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

udelay(10);

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

}

}

If spi\_transfer - > cs\_change is chosen 1 then CS changes as below: -

If spi\_transfer is not last, then CS line is made high only for 10 µs then it is made low again.

Similarly if spi\_transfer is last in the list, then it will maintain CS line low only, if SPI\_CS is set as active low.