SPI Drivers

Serial Peripheral Interface

Proprietary & Confidential. © Aricent 2019

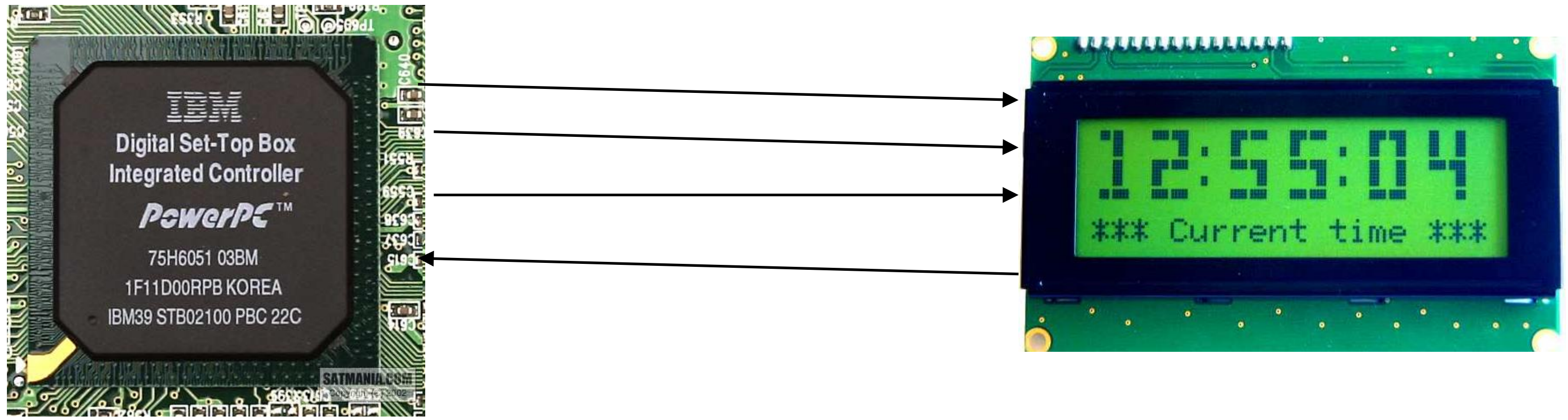
Outline

- What is SPI?
- Basic Serial Peripheral Interface (SPI)
- Capabilities
- Protocol
- SPI Framework
- SPI Framework Components
- SPI Client Driver
- Pro / Cons and Competitor
- Uses
- Conclusion

Proprietary & Confidential. © Aricent 2019

What is SPI?

- Serial Bus protocol
- Fast, Easy to use, Simple
- Everyone supports it



SPI Basics

- A communication protocol using 4 wires
- Also known as a 4 wire bus
- Used to communicate across small distances
- Multiple Slaves, Single Master
- Synchronized

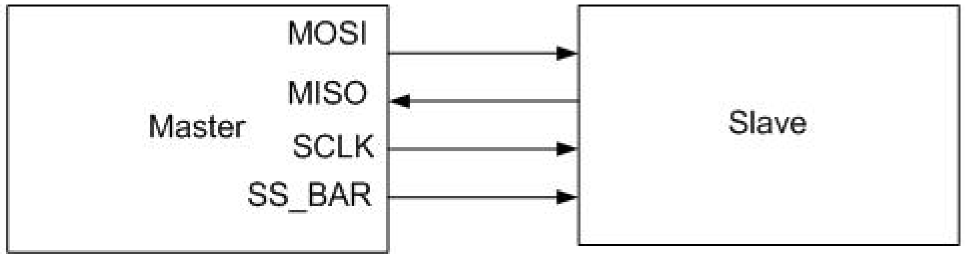
Proprietary & Confidential. © Aricent 2019

Capabilities of SPI

- Always Full Duplex
- Communicating in two directions at the same time
- Transmission need not be meaningful
- Multiple MBPS transmission speed
- Transfers data in 4 to 16 bit characters
- Multiple slaves
- Daisy-chaining possible

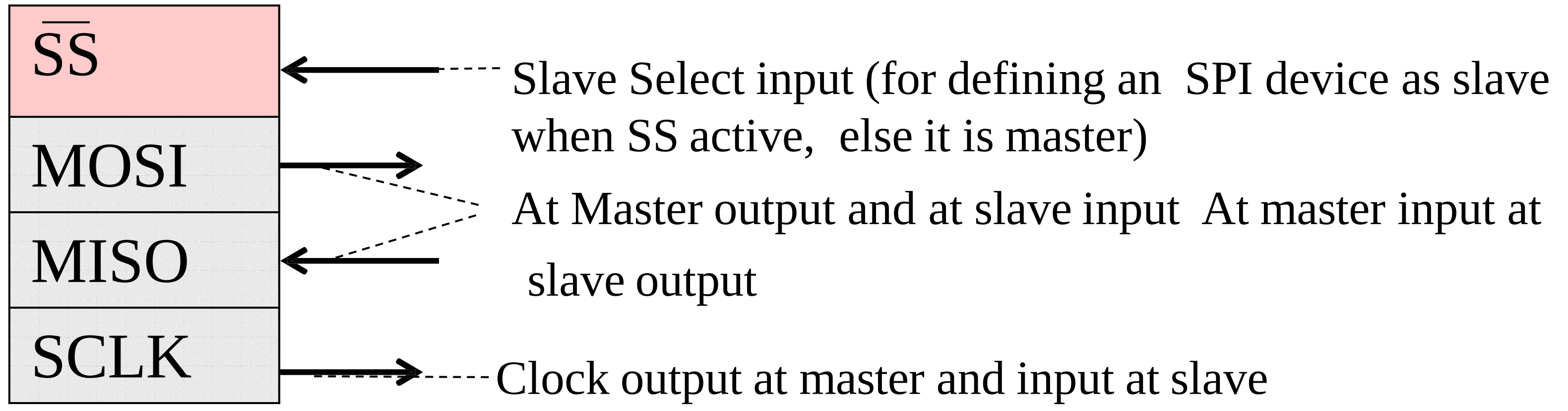
Proprietary & Confidential. © Aricent 2019

Protocol



- Wires:
  - MOSI – Carries data out of Master to Slave
  - MISO – Carries data from Slave to Master
  - Both signals happen for every transmission
  - SCLK – Master produced clock to synchronize data transfer
  - SS_BAR – Unique line to select a slave. Slave Select 1…N
- Master Set Slave Select low
- Master Generates Clock
- Shift registers shift in and out data

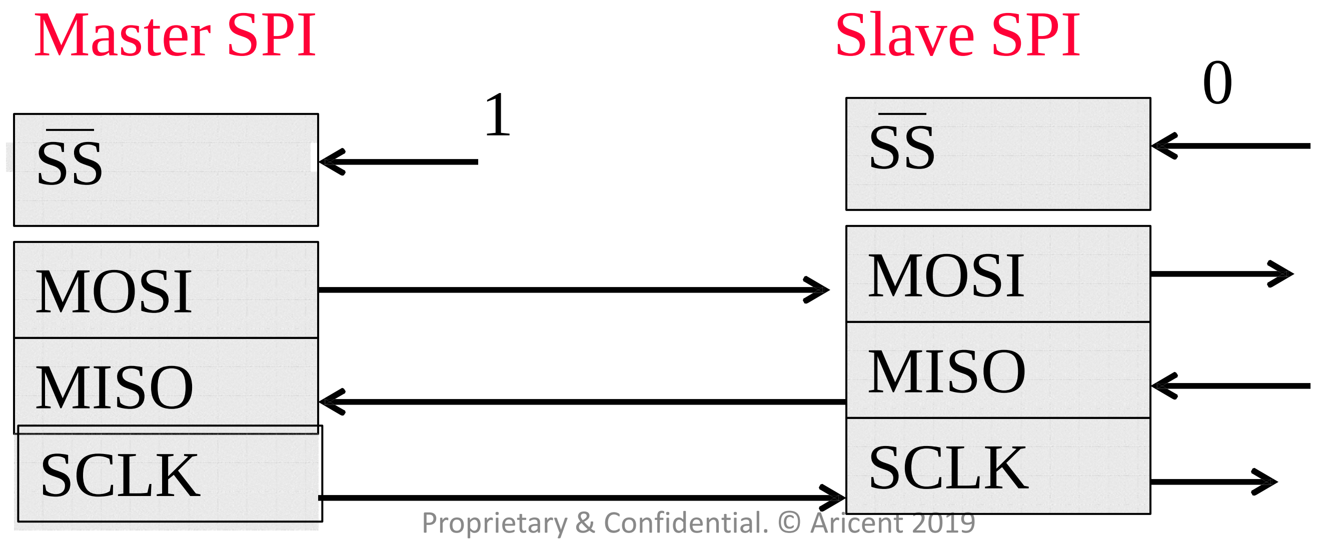
Protocol



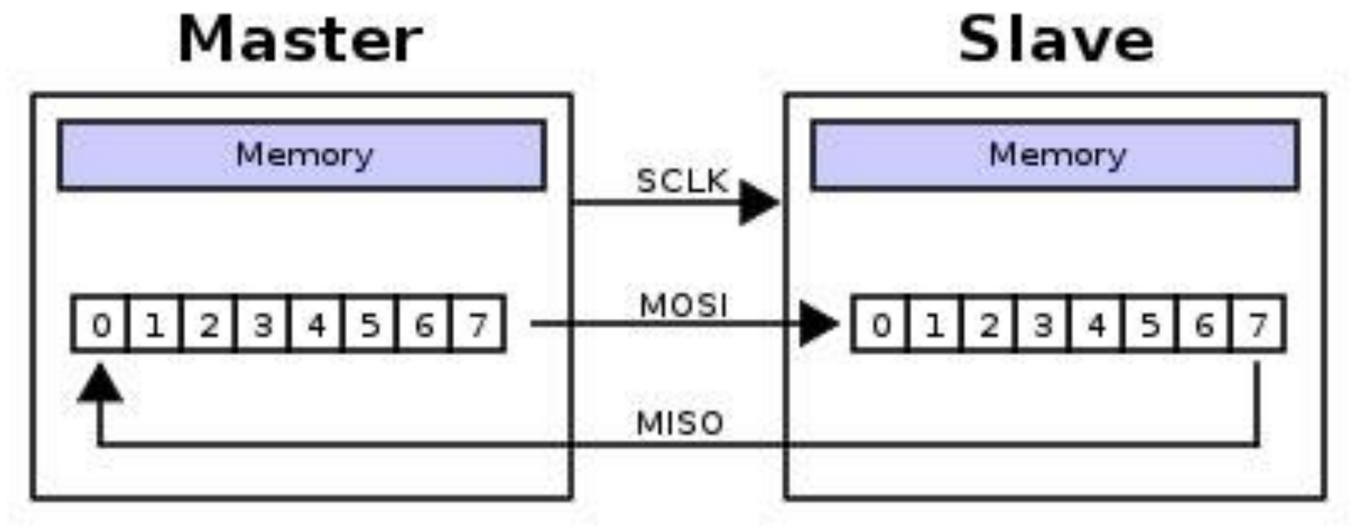
Slave Select input (for defining an SPI device as slave when SS active, else it is master)

At Master output and at slave input At master input at slave output

Clock output at master and input at slave

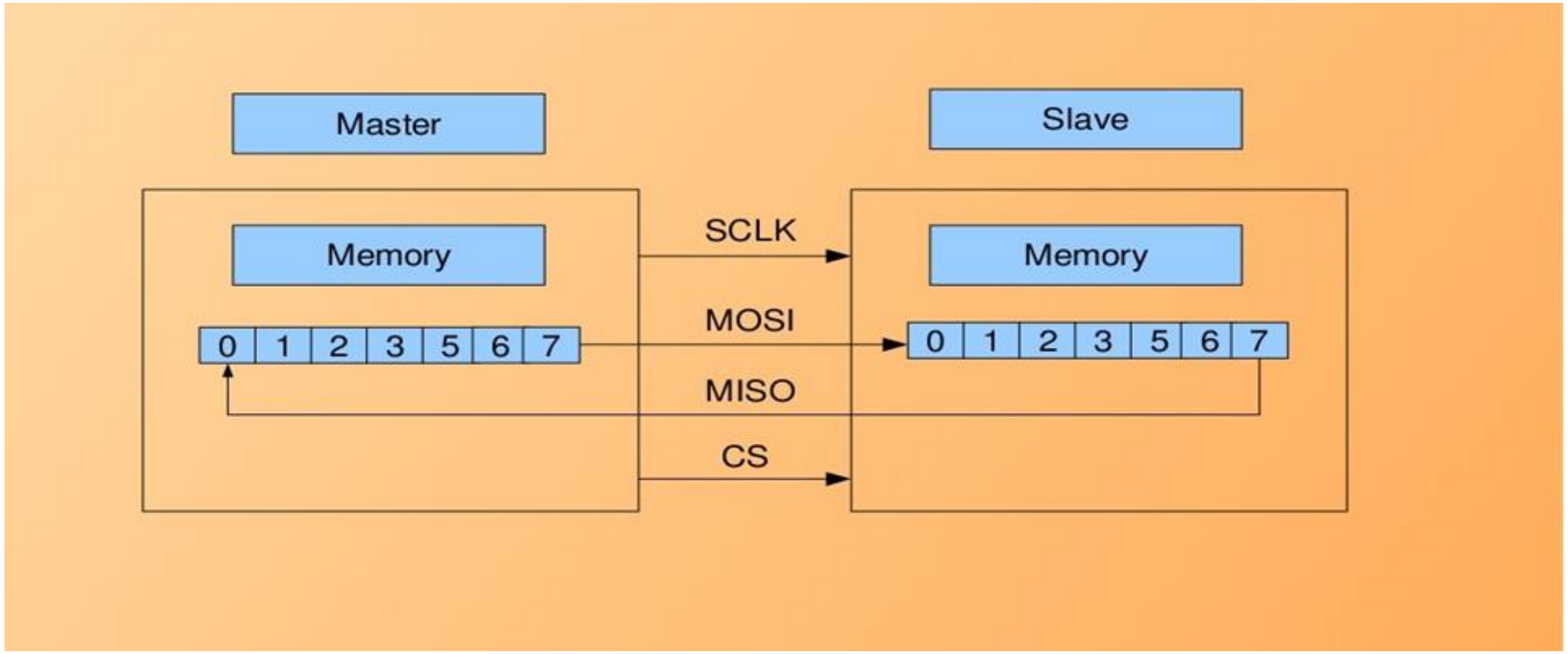


Shifting Protocol



Master shifts out data to Slave, and shift in data from Slave

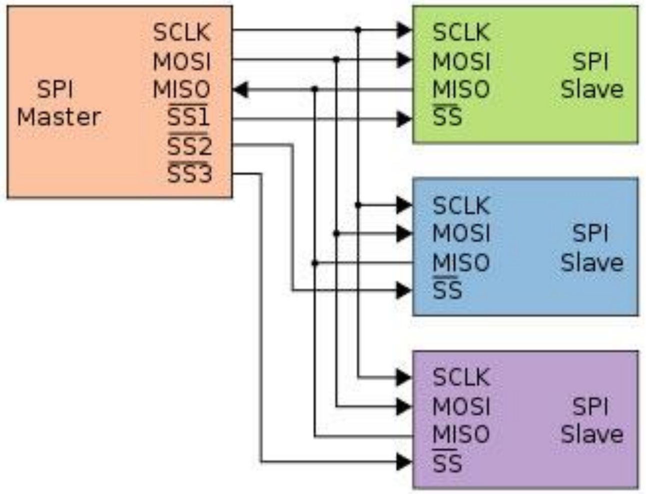
Shifting Protocol



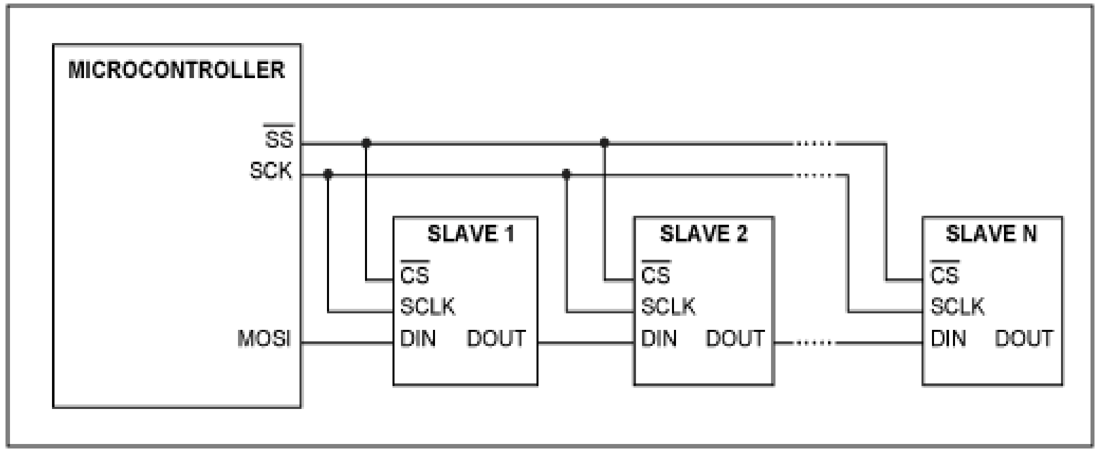
Master shifts out data to Slave, and shift in data from Slave

Aricent
ALTRAN GROUP

Master – Salve Configurations



Master and multiple independent slaves



Some wires have been renamed

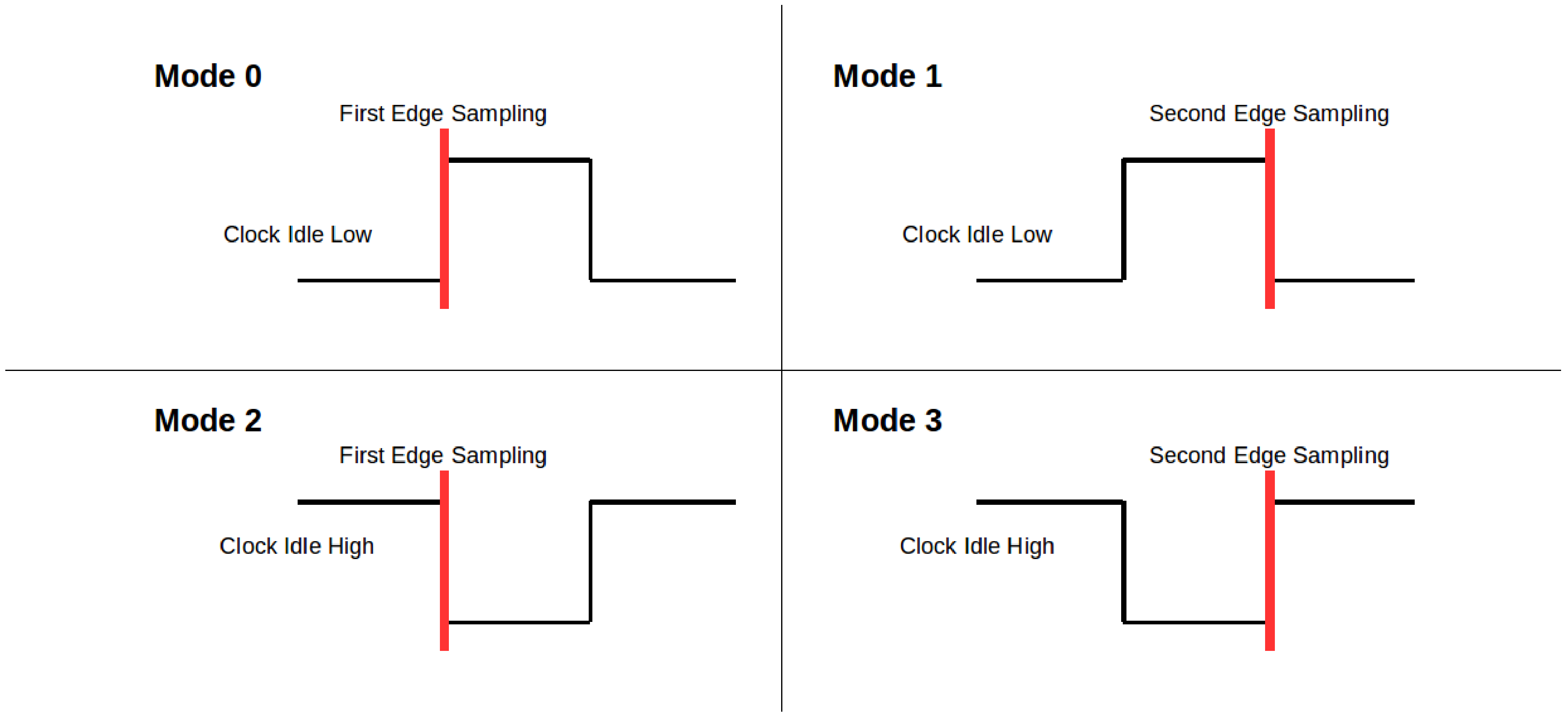
Master and multiple daisy-chained slaves

Clock Phase (Advanced)

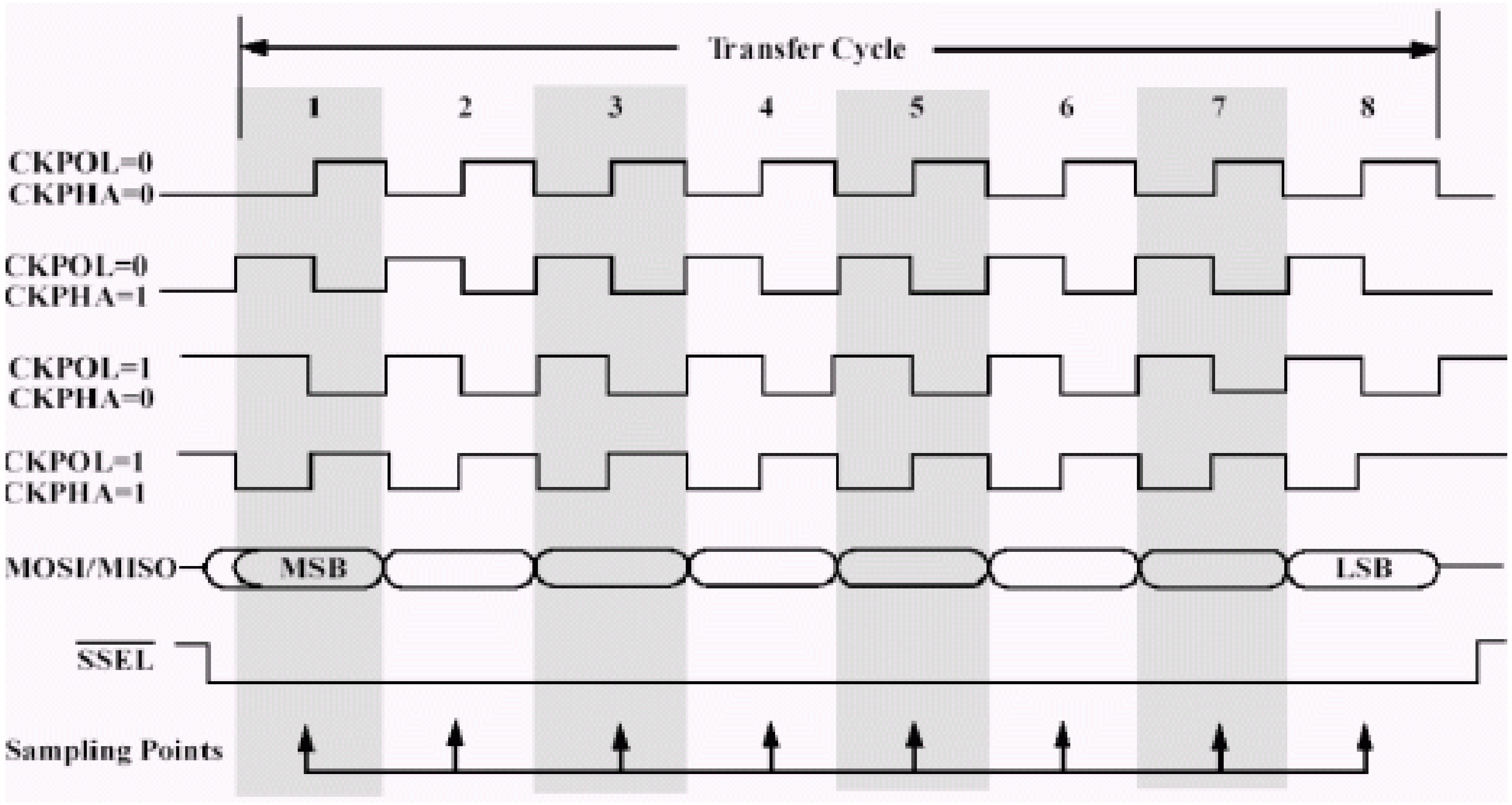
- Two phases and two polarities of clock
- Four modes
- Master and selected slave must be in same mode
- Master must change polarity and phase to communicate with slaves of different numbers

Proprietary & Confidential. © Aricent 2019

4 Modes


Mode 0
First Edge Sampling
Clock Idle Low
Mode 1
Second Edge Sampling
Clock Idle Low
Mode 2
First Edge Sampling
Clock Idle High
Mode 3
Second Edge Sampling
Clock Idle High

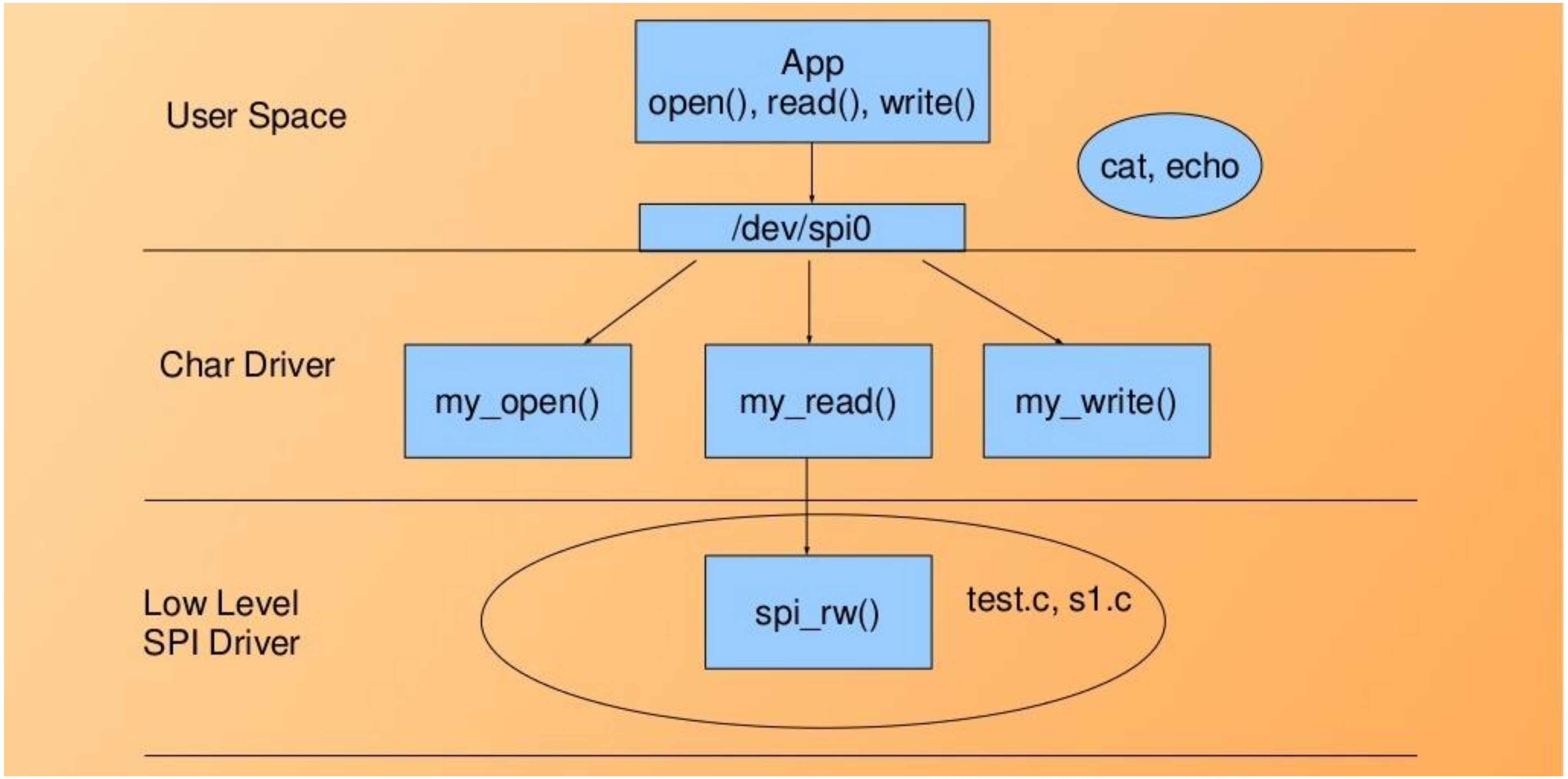
Timing Diagram



Proprietary & Confidential. © Aricent 2019

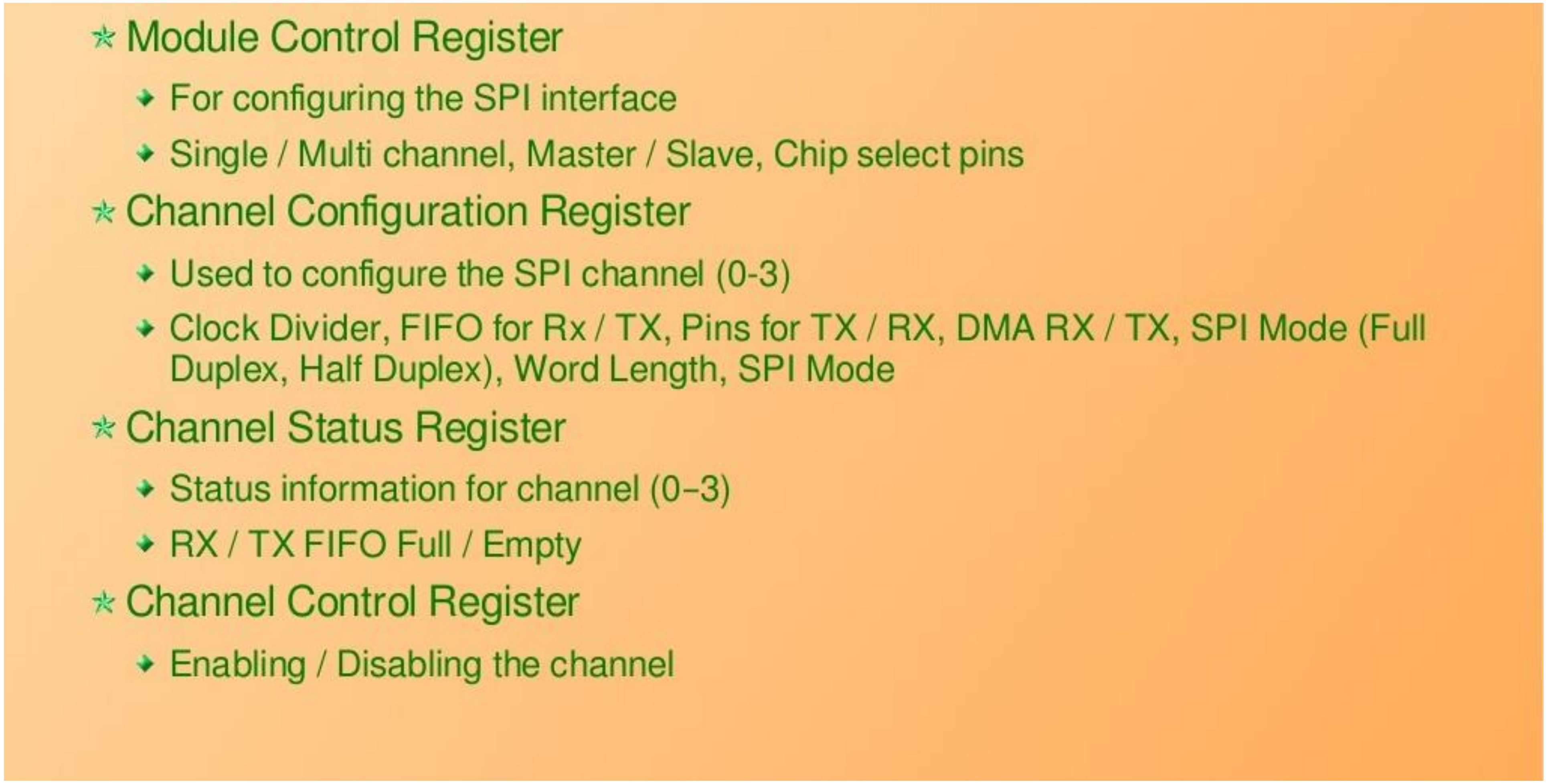
Aricent
.ALTRAN GROUP

SPI Character Driver Framework



Proprietary & Confidential. © Aricent 2019

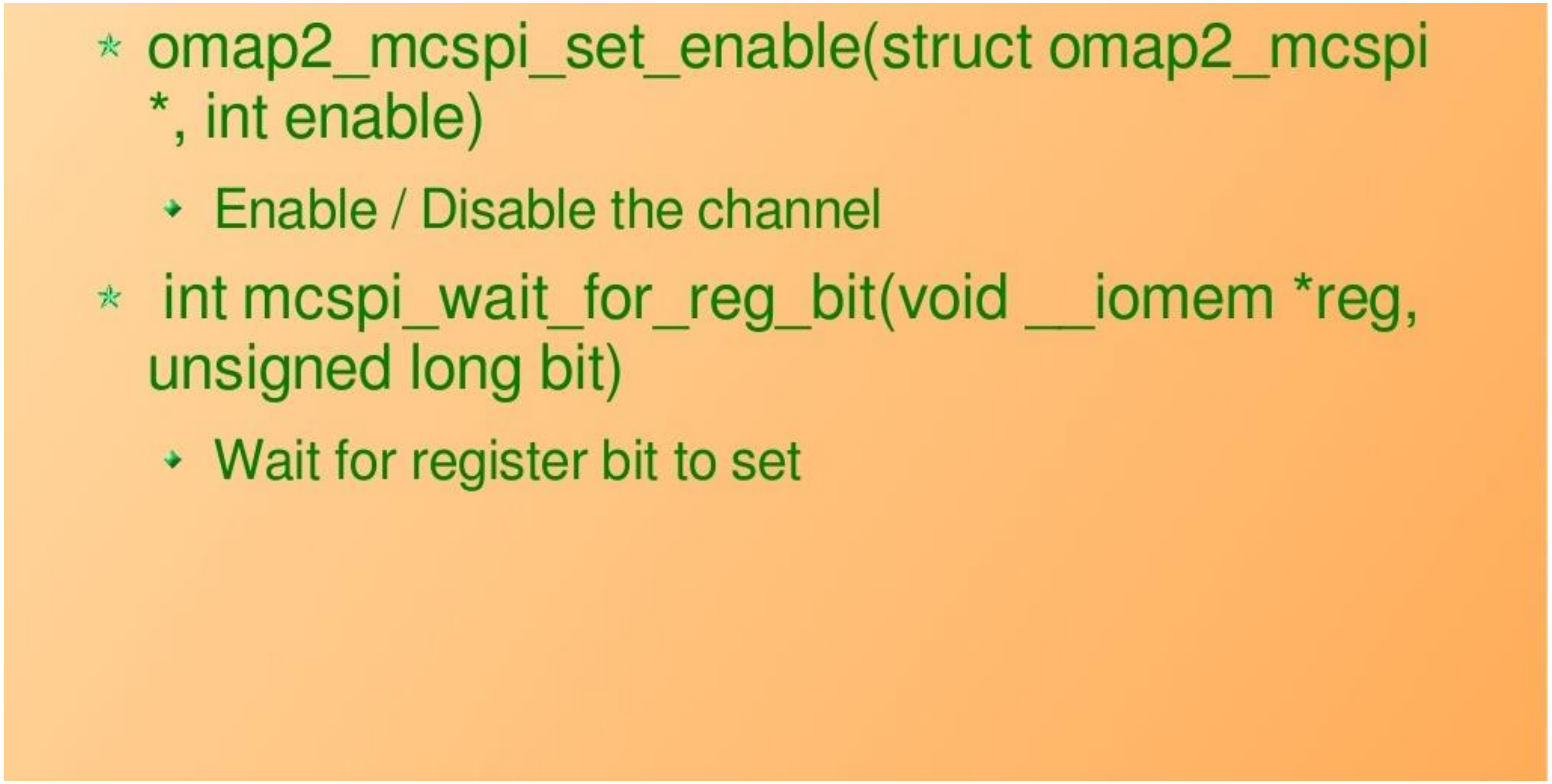
AM335X Registers



- Module Control Register
  - For configuring the SPI interface
  - Single / Multi channel, Master / Slave, Chip select pins
- Channel Configuration Register
  - Used to configure the SPI channel (0-3)
  - Clock Divider, FIFO for Rx / TX, Pins for TX / RX, DMA RX / TX, SPI Mode (Full Duplex, Half Duplex), Word Length, SPI Mode
- Channel Status Register
  - Status information for channel (0–3)
  - RX / TX FIFO Full / Empty
- Channel Control Register
  - Enabling / Disabling the channel

Aricent
ALTRAN GROUP

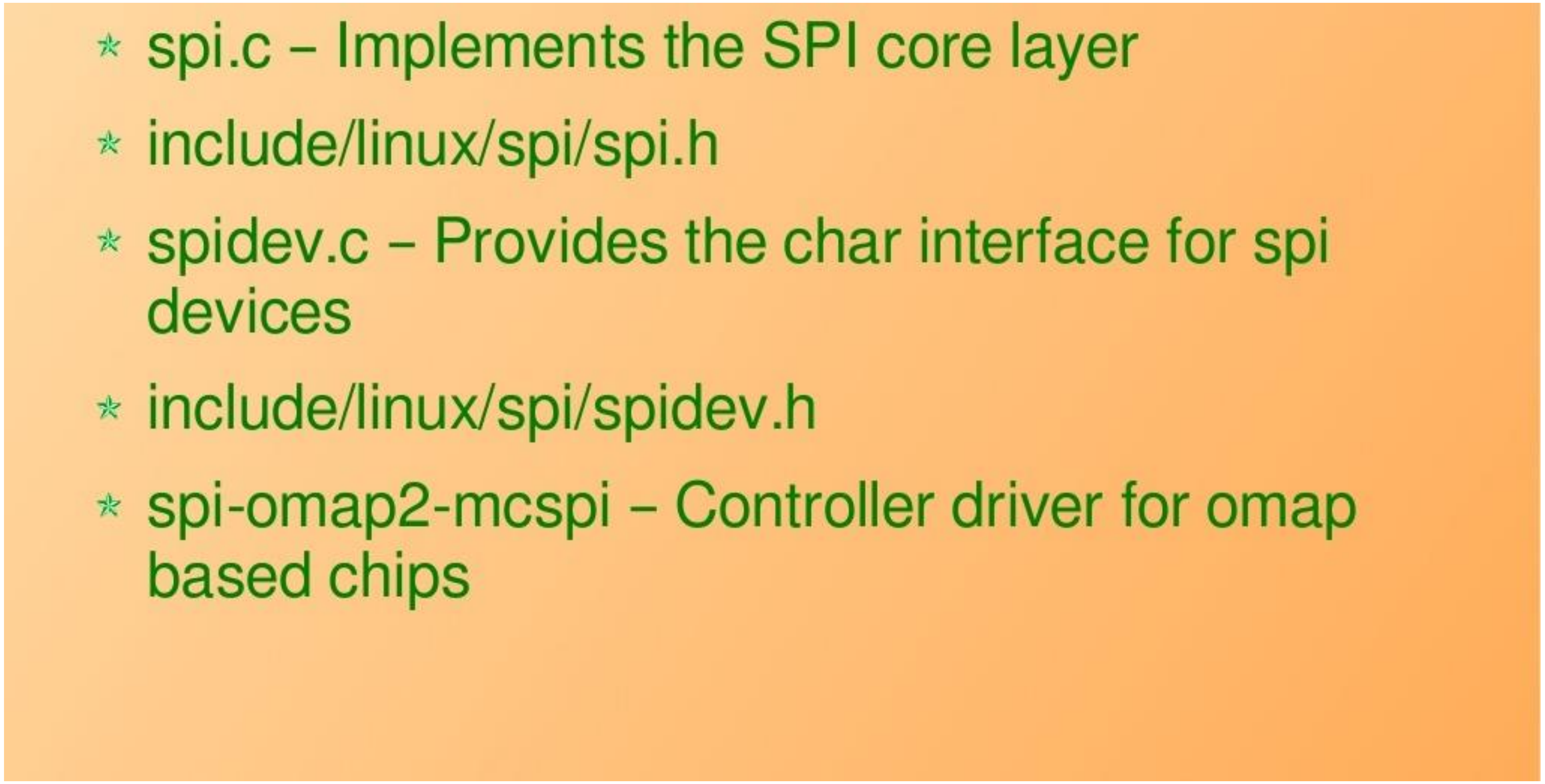
AM335X SPI APIs



- omap2_mcspi_set_enable(struct omap2_mcspi *, int enable)
  - Enable / Disable the channel
- int mcspi_wait_for_reg_bit(void __iomem *reg, unsigned long bit)
  - Wait for register bit to set

Aricent
ALTRAN GROUP

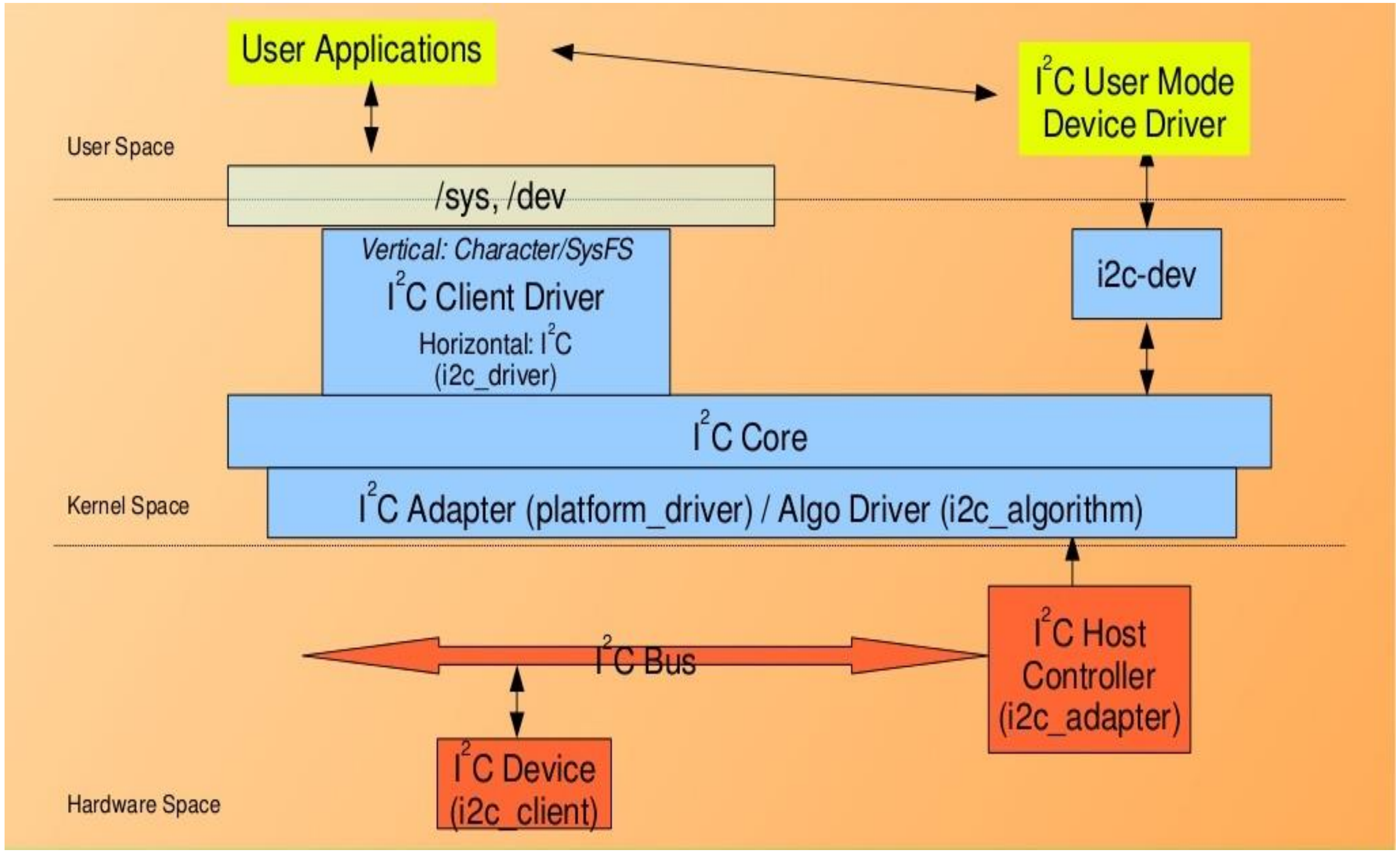
SPI Framework



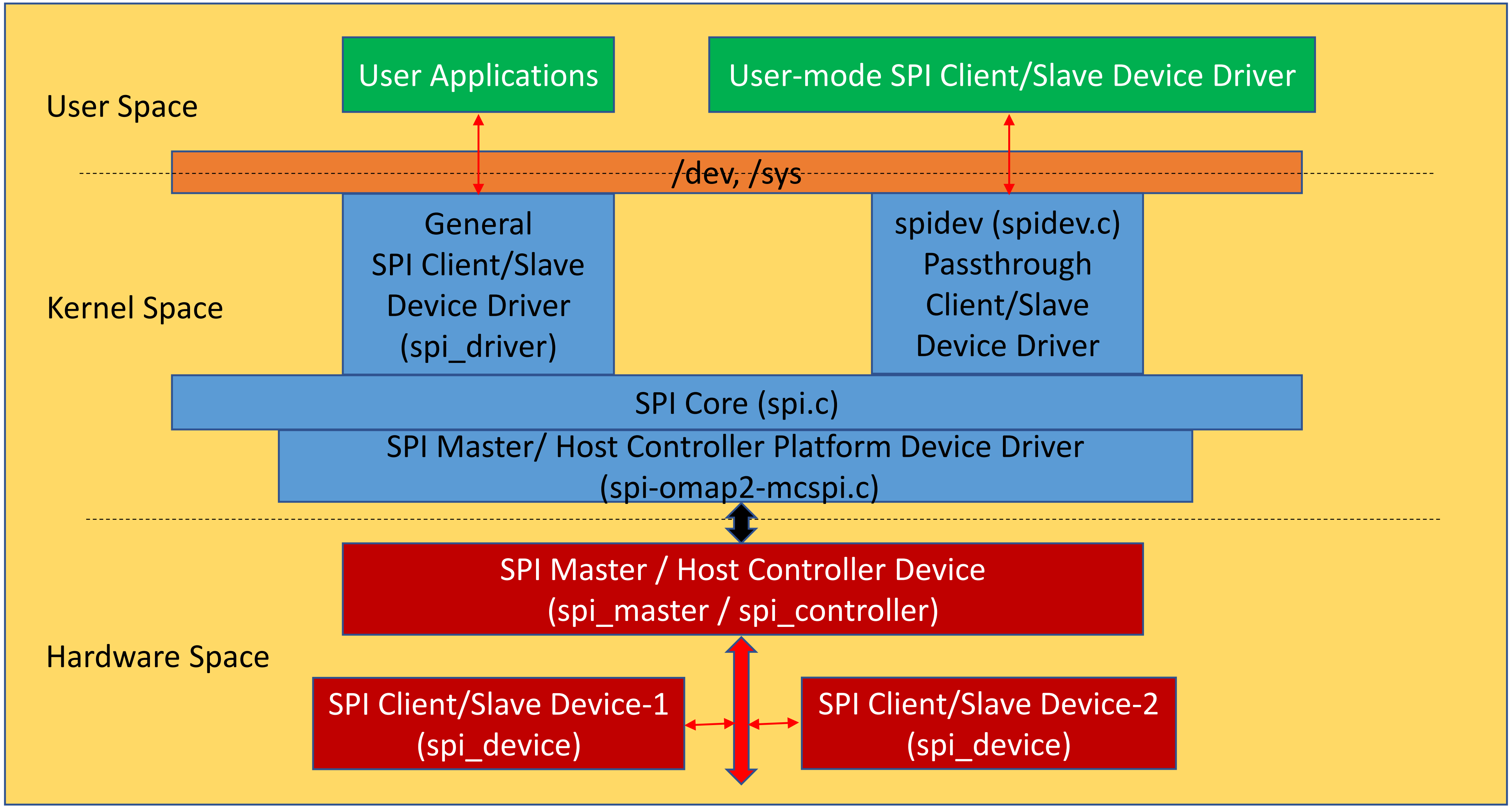
- spi.c – Implements the SPI core layer
- include/linux/spi/spi.h
- spidev.c – Provides the char interface for spi devices
- include/linux/spi/spidev.h
- spi-omap2-mcspi – Controller driver for omap based chips

Proprietary & Confidential. © Aricent 2019

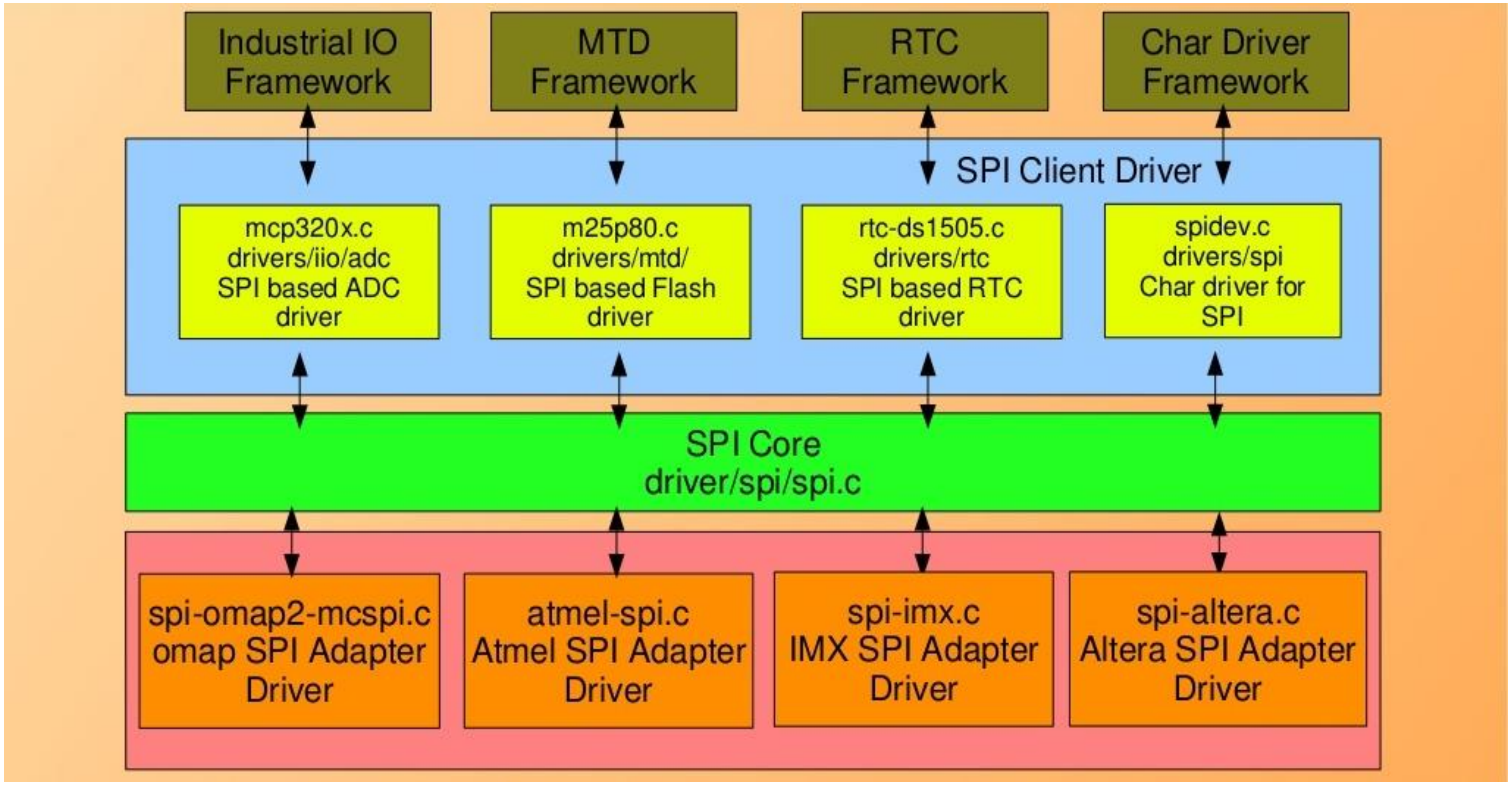
SPI Subsystem


User Applications
I²C User Mode Device Driver
User Space
/sys, /dev
Vertical: Character/SysFS
I²C Client Driver
Horizontal: I²C (i2c_driver)
i2c-dev
I²C Core
Kernel Space
I²C Adapter (platform_driver) / Algo Driver (i2c_algorithm)
I²C Bus
I²C Host Controller (i2c_adapter)
I²C Device (i2c_client)
Hardware Space

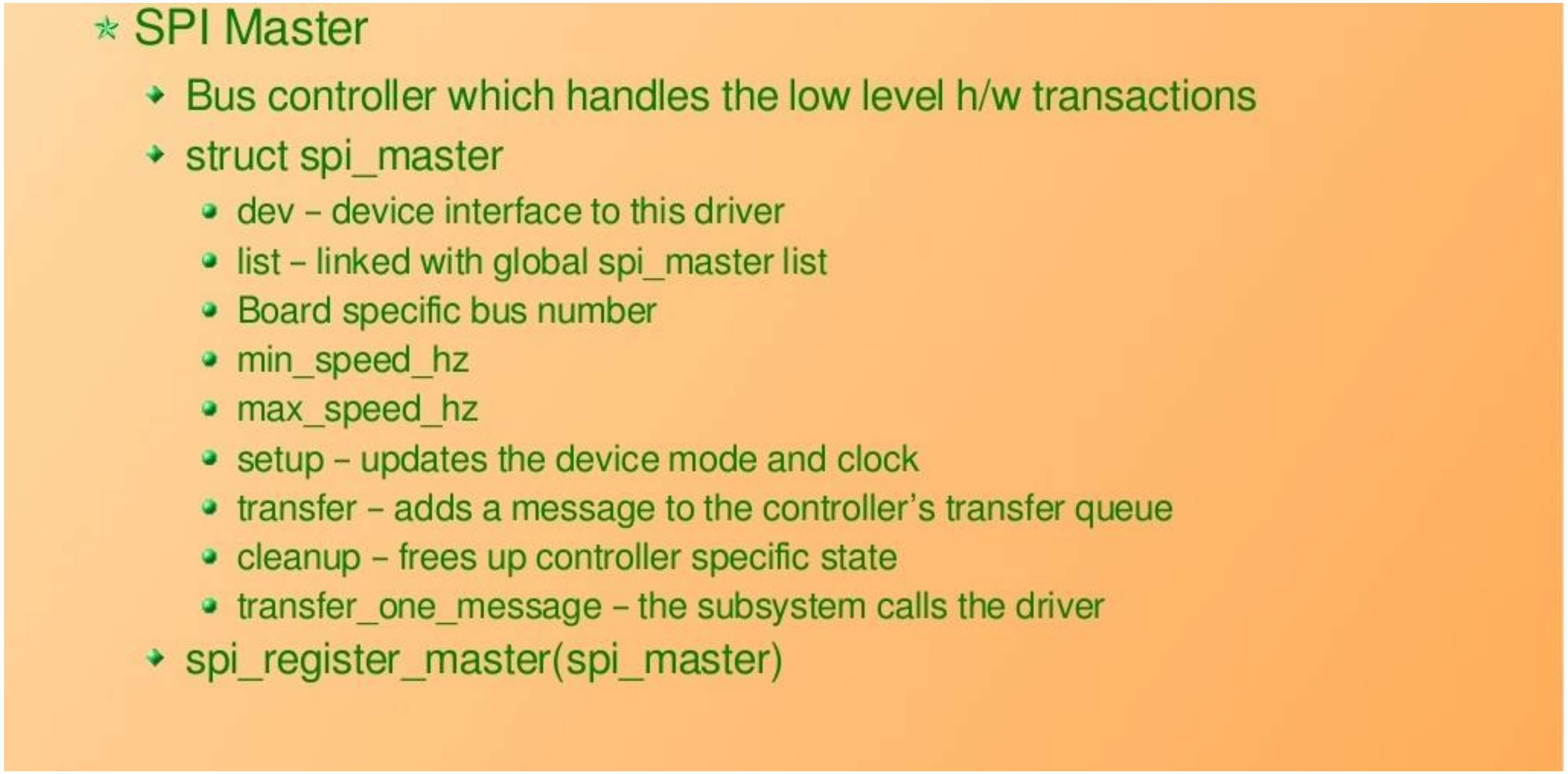
SPI Subsystem


User Space
User Applications
User-mode SPI Client/Slave Device Driver
/dev, /sys
Kernel Space
General SPI Client/Slave Device Driver (spi_driver)
spidev (spidev.c) Passthrough Client/Slave Device Driver
SPI Core (spi.c)
SPI Master/ Host Controller Platform Device Driver (spi-omap2-mcspi.c)
SPI Master / Host Controller Device (spi_master / spi_controller)
Hardware Space
SPI Client/Slave Device-1 (spi_device)
SPI Client/Slave Device-2 (spi_device)

SPI Framework


Industrial IO Framework
MTD Framework
RTC Framework
Char Driver Framework
SPI Client Driver
mcp320x.c drivers/iio/adc SPI based ADC driver
m25p80.c drivers/mtd/ SPI based Flash driver
rtc-ds1505.c drivers/rtc SPI based RTC driver
spidev.c drivers/spi Char driver for SPI
SPI Core driver/spi/spi.c
spi-omap2-mcspi.c omap SPI Adapter Driver
atmel-spi.c Atmel SPI Adapter Driver
spi-imx.c IMX SPI Adapter Driver
spi-altera.c Altera SPI Adapter Driver

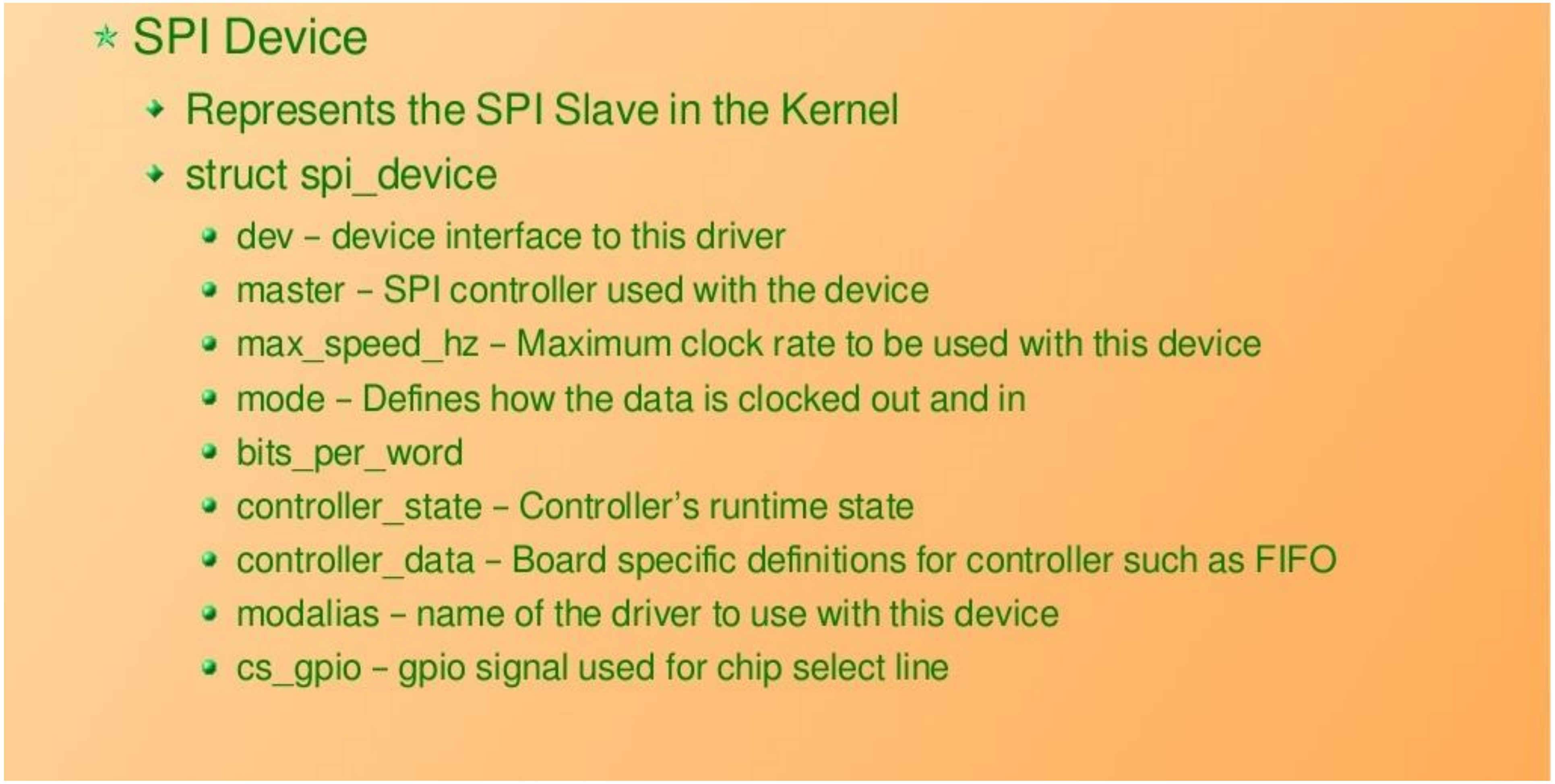
Framework Components



- SPI Master
  - Bus controller which handles the low level h/w transactions
  - struct spi_master
    - dev – device interface to this driver
    - list – linked with global spi_master list
    - Board specific bus number
    - min_speed_hz
    - max_speed_hz
    - setup – updates the device mode and clock
    - transfer – adds a message to the controller's transfer queue
    - cleanup – frees up controller specific state
    - transfer_one_message – the subsystem calls the driver
  - spi_register_master(spi_master)

Proprietary & Confidential. © Aricent 2019

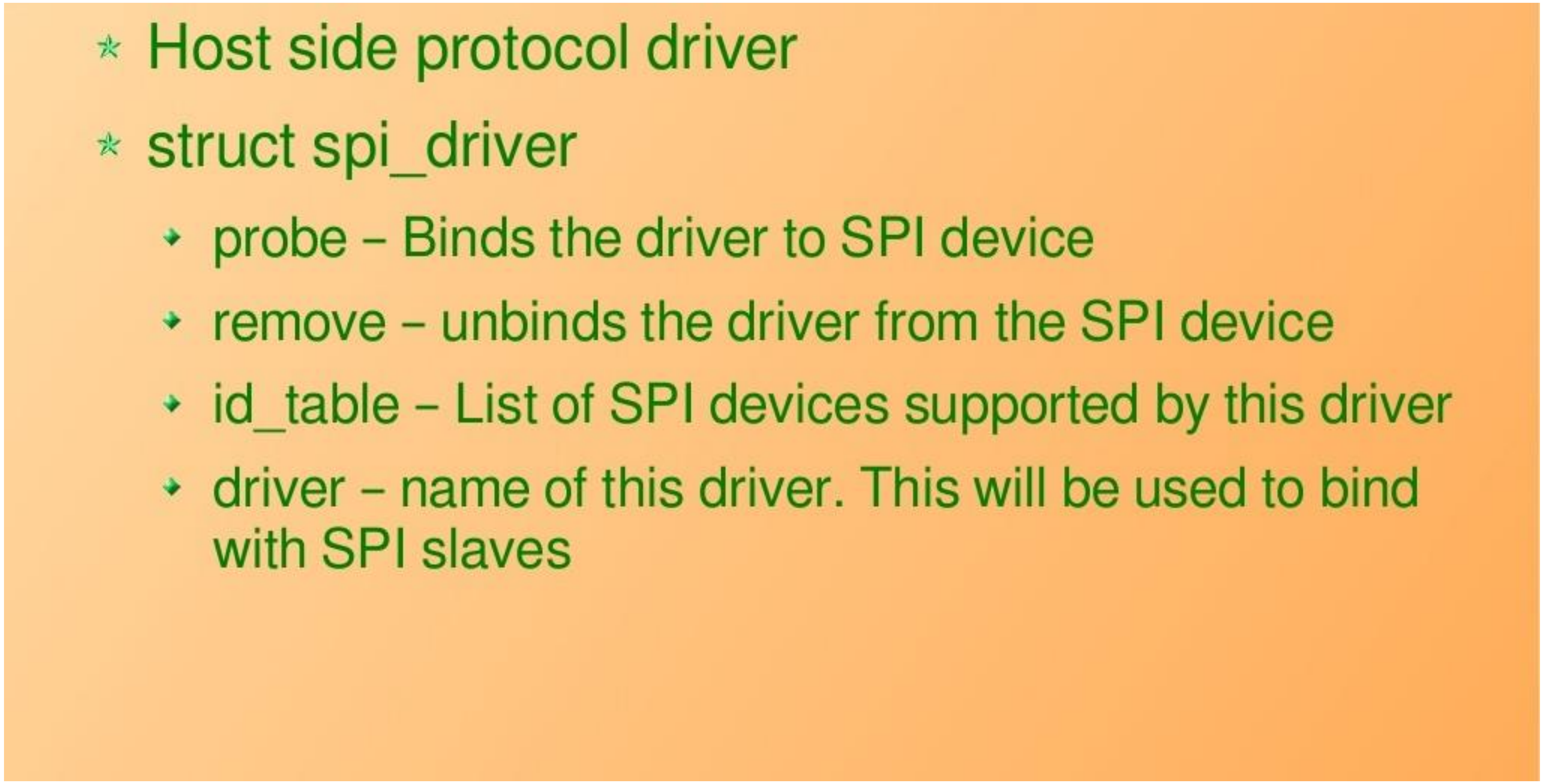
Framework Components….


- SPI Device
  - Represents the SPI Slave in the Kernel
  - struct spi_device
    - dev – device interface to this driver
    - master – SPI controller used with the device
    - max_speed_hz – Maximum clock rate to be used with this device
    - mode – Defines how the data is clocked out and in
    - bits_per_word
    - controller_state – Controller's runtime state
    - controller_data – Board specific definitions for controller such as FIFO
    - modalias – name of the driver to use with this device
    - cs_gpio – gpio signal used for chip select line

Proprietary & Confidential. © Aricent 2019

Aricent
ALTRAN GROUP

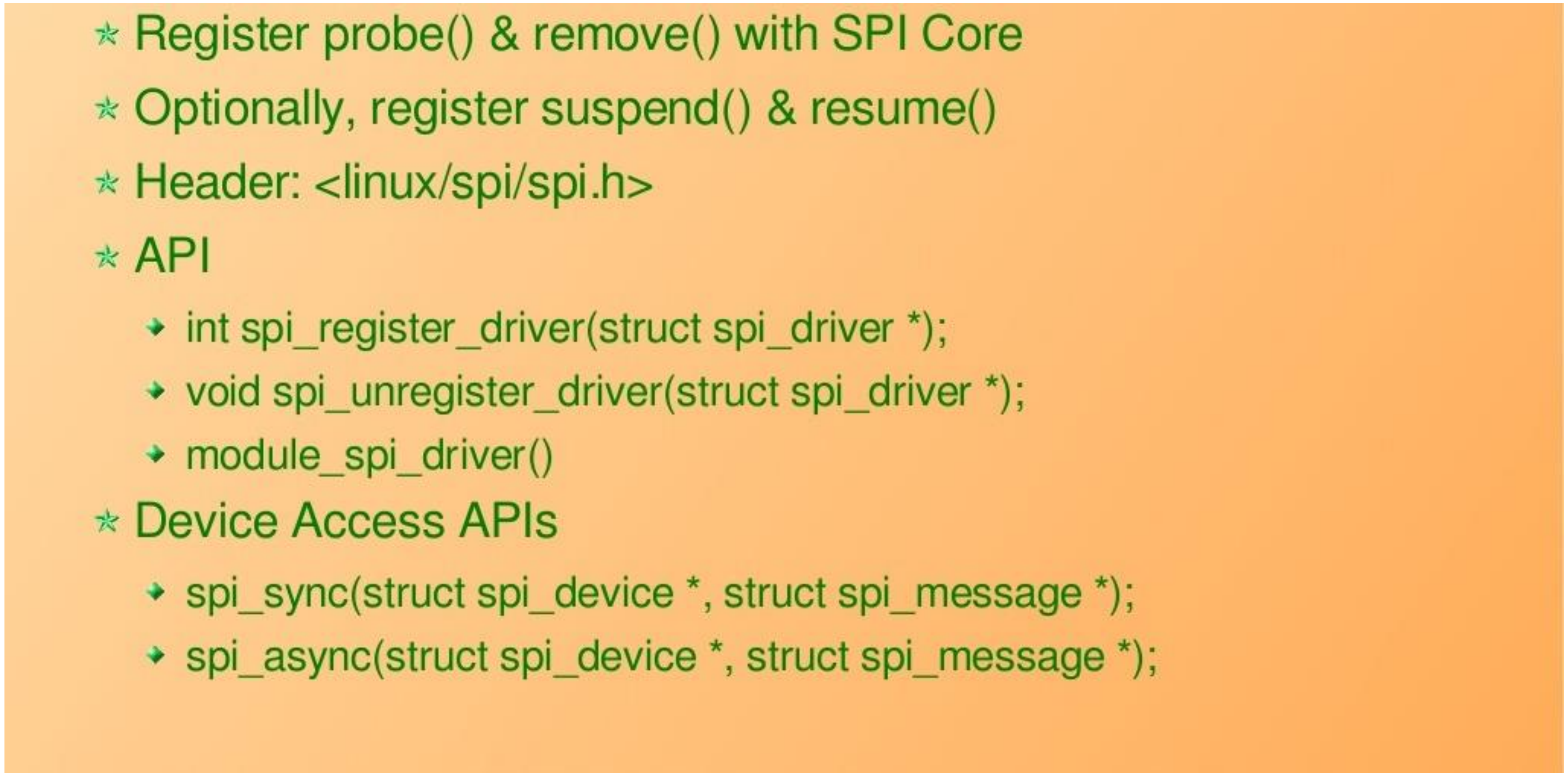
SPI Client Driver



- Host side protocol driver
- struct spi_driver
  - probe – Binds the driver to SPI device
  - remove – unbinds the driver from the SPI device
  - id_table – List of SPI devices supported by this driver
  - driver – name of this driver. This will be used to bind with SPI slaves

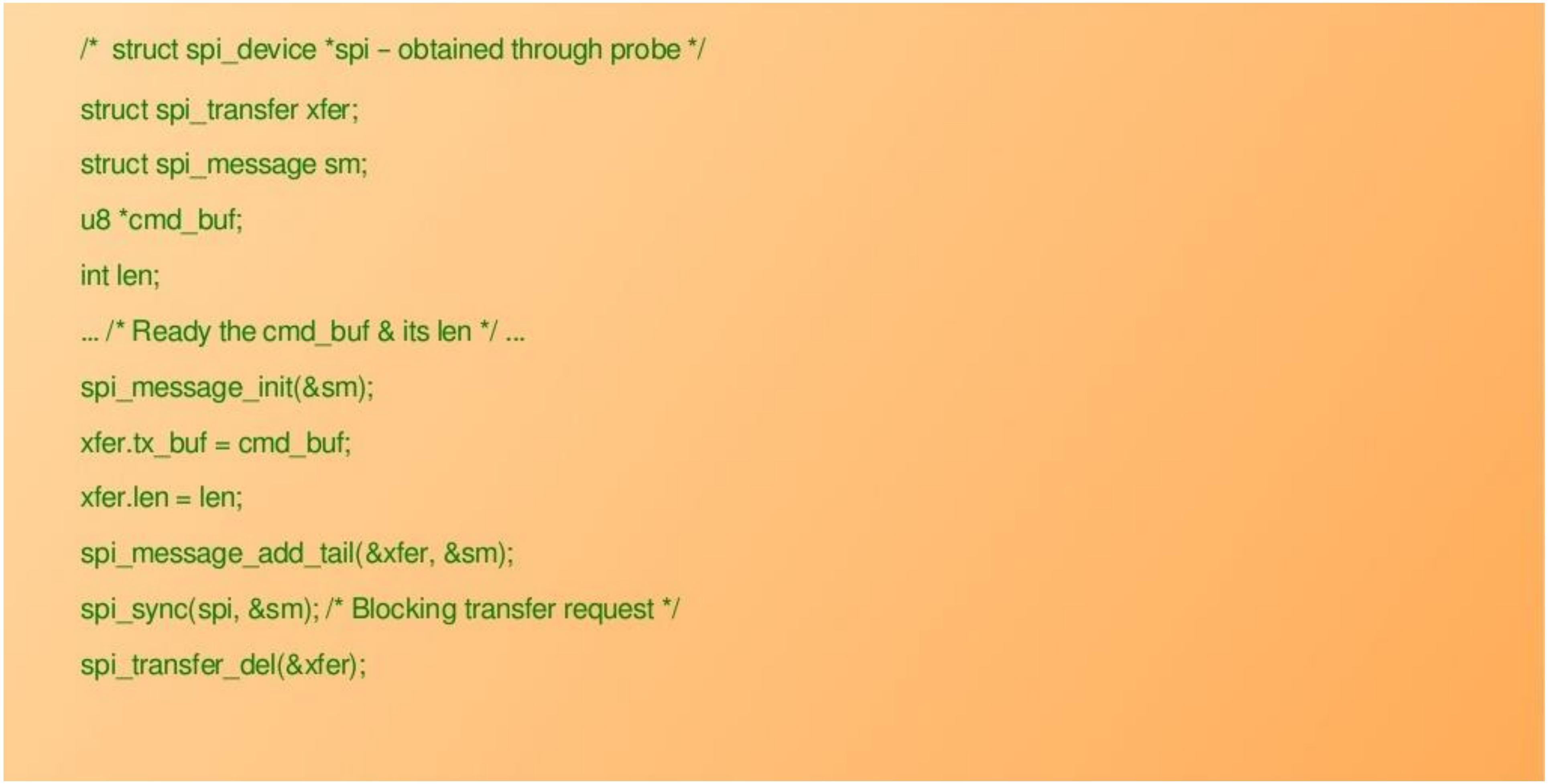
Proprietary & Confidential. © Aricent 2019

SPI Client Driver….



- Register probe() & remove() with SPI Core
- Optionally, register suspend() & resume()
- Header: <linux/spi/spi.h>
- API
  - int spi_register_driver(struct spi_driver *);
  - void spi_unregister_driver(struct spi_driver *);
  - module_spi_driver()
- Device Access APIs
  - spi_sync(struct spi_device *, struct spi_message *);
  - spi_async(struct spi_device *, struct spi_message *);

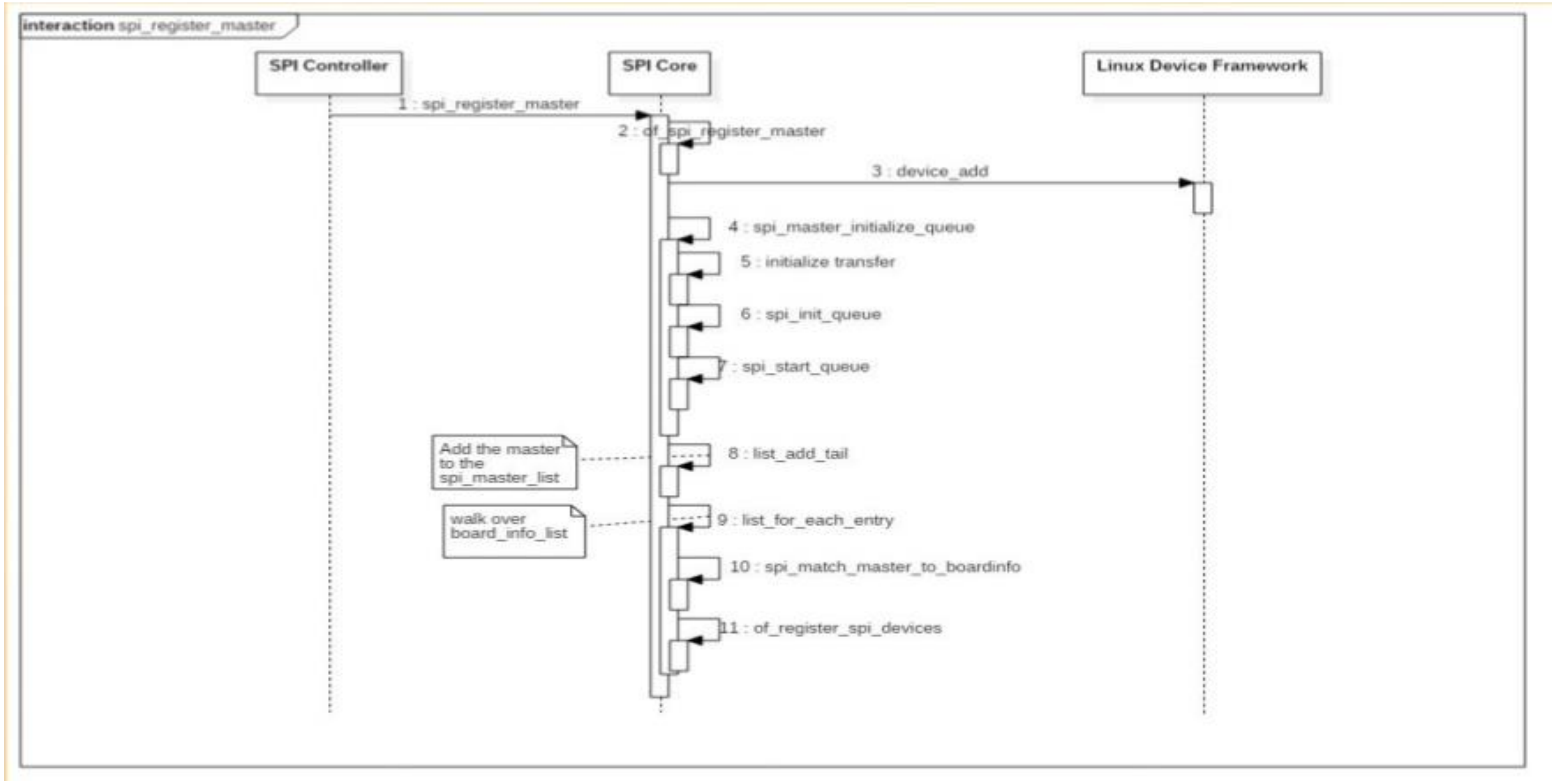
SPI Device Access



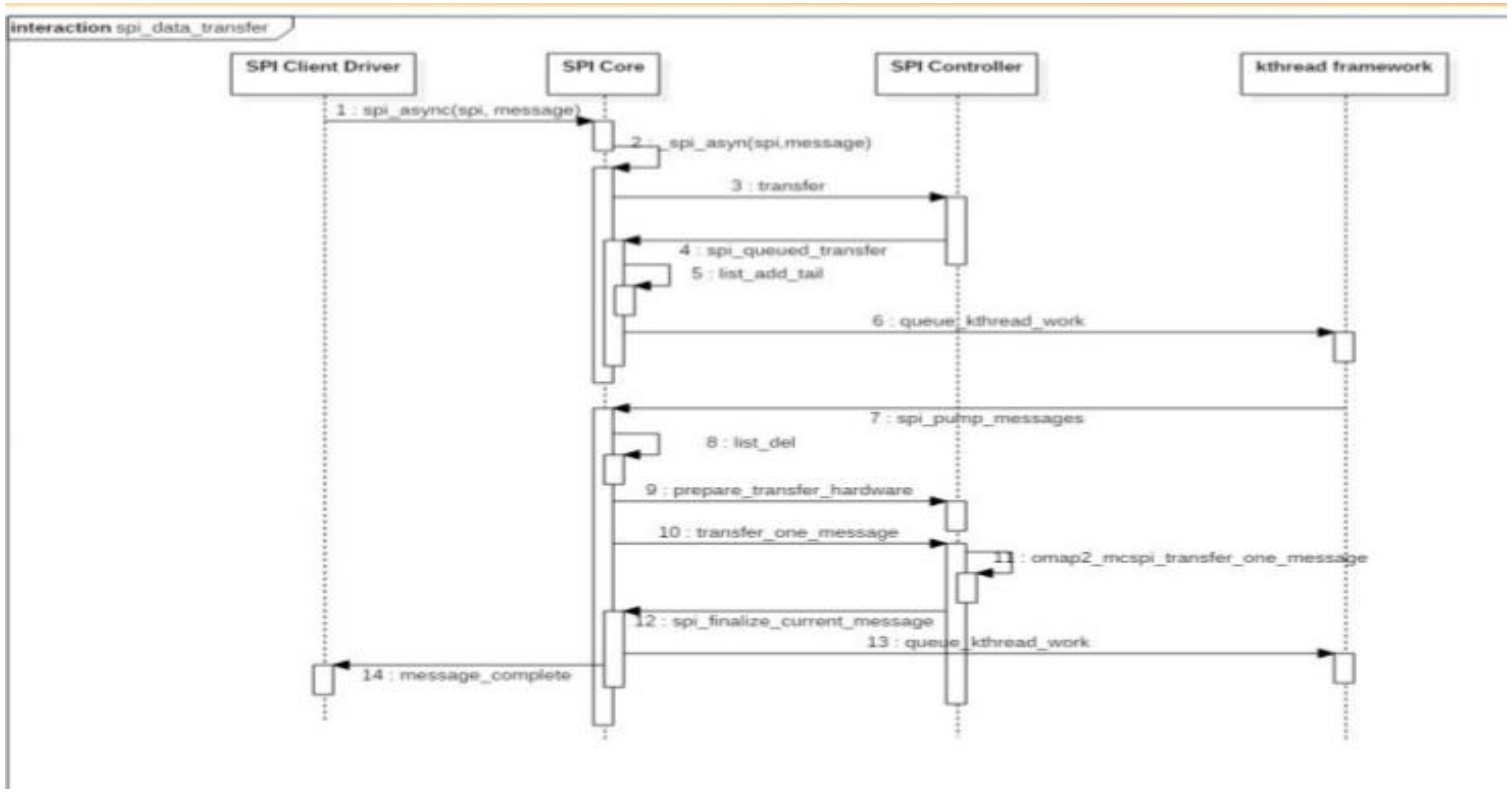
```
/*  struct spi_device *spi – obtained through probe */
struct spi_transfer xfer;
struct spi_message sm;
u8 *cmd_buf;
int len;
... /* Ready the cmd_buf & its len */ ...
spi_message_init(&sm);
xfer.tx_buf = cmd_buf;
xfer.len = len;
spi_message_add_tail(&xfer, &sm);
spi_sync(spi, &sm); /* Blocking transfer request */
spi_transfer_del(&xfer);
```

Proprietary & Confidential. © Aricent 2019

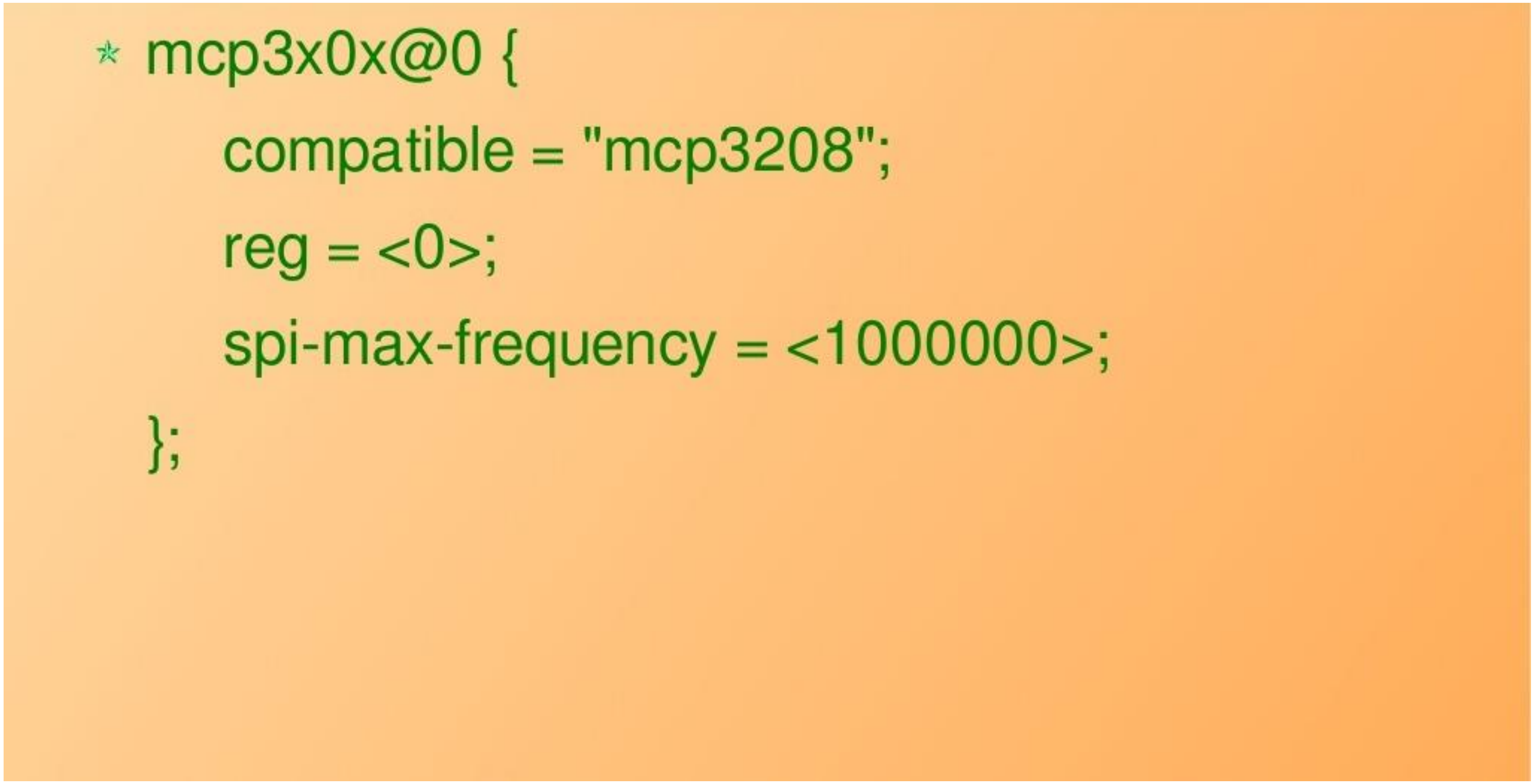
SPI Master Registration Flow


interaction spi_register_master
SPI Controller
SPI Core
Linux Device Framework
1 : spi_register_master
2 : of_spi_register_master
3 : device_add
4 : spi_master_initialize_queue
5 : initialize transfer
6 : spi_init_queue
7 : spi_start_queue
Add the master to the spi_master_list
8 : list_add_tail
walk over board_info_list
9 : list_for_each_entry
10 : spi_match_master_to_boardinfo
11 : of_register_spi_devices

SPI _ASync Flow

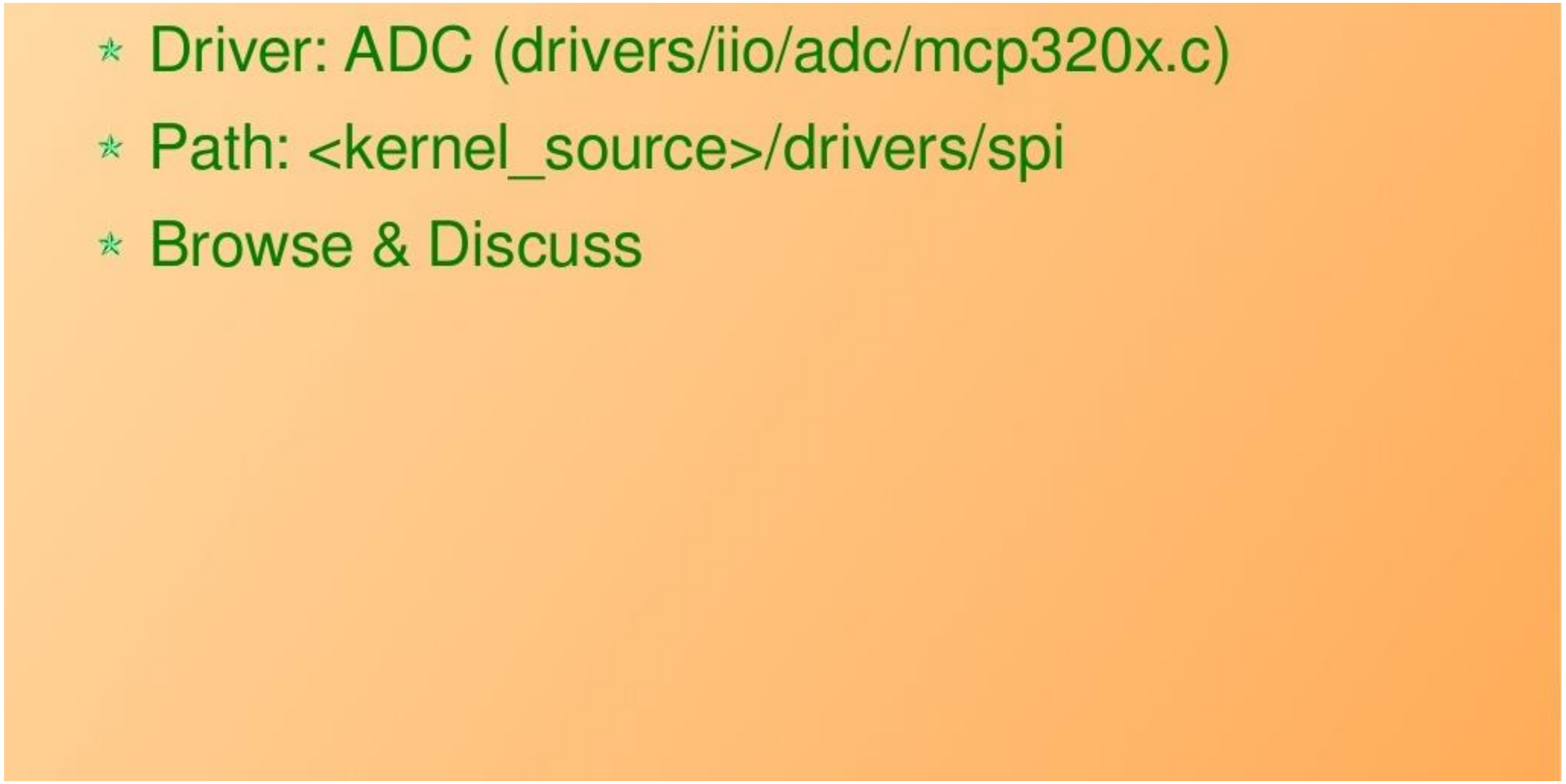


DTB changes for MCP3008



```
* mcp3x0x@0 {
      compatible = "mcp3208";
      reg = <0>;
      spi-max-frequency = <1000000>;
  };
```

SPI Driver Example



- Driver: ADC (drivers/iio/adc/mcp320x.c)
- Path: <kernel_source>/drivers/spi
- Browse & Discuss

Pros and Cons

Pros:

- Fast and easy
  - Fast for point-to-point connections
  - Easily allows streaming/Constant data inflow
  - No addressing/Simple to implement
- Everyone supports it

Cons:

- SS makes multiple slaves very complicated
- No acknowledgement ability
- No inherent arbitration
- No flow control

Proprietary & Confidential. © Aricent 2019

Uses

- Some Serial Encoders/Decoders, Converters, Serial LCDs, Sensors, etc.
- Pre-SPI serial devices

Proprietary & Confidential. © Aricent 2019

summary

- SPI – 4 wire serial bus protocol
- MOSI MISO SS SCLK wires
- Full duplex
- Multiple slaves, One master
- Best for point-to-point streaming data
- Easily Supported

Proprietary & Confidential. © Aricent 2019

What all have we learn

- What is SPI?
- Basic Serial Peripheral Interface (SPI)
- Capabilities
- Protocol
- SPI Framework
- SPI Framework Components
- SPI Client Driver
- Pro / Cons and Competitor
- Uses
- Conclusion

Proprietary & Confidential. © Aricent 2019

Any Queries?

Proprietary & Confidential. © Aricent 2019