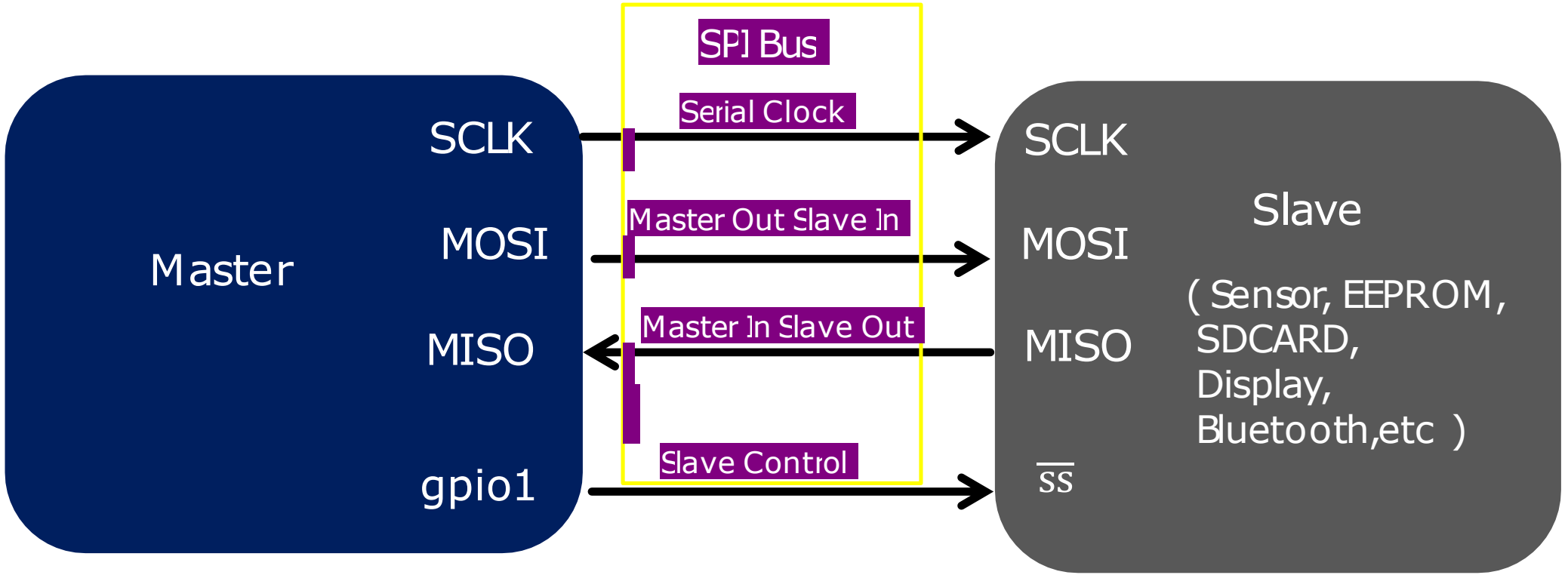
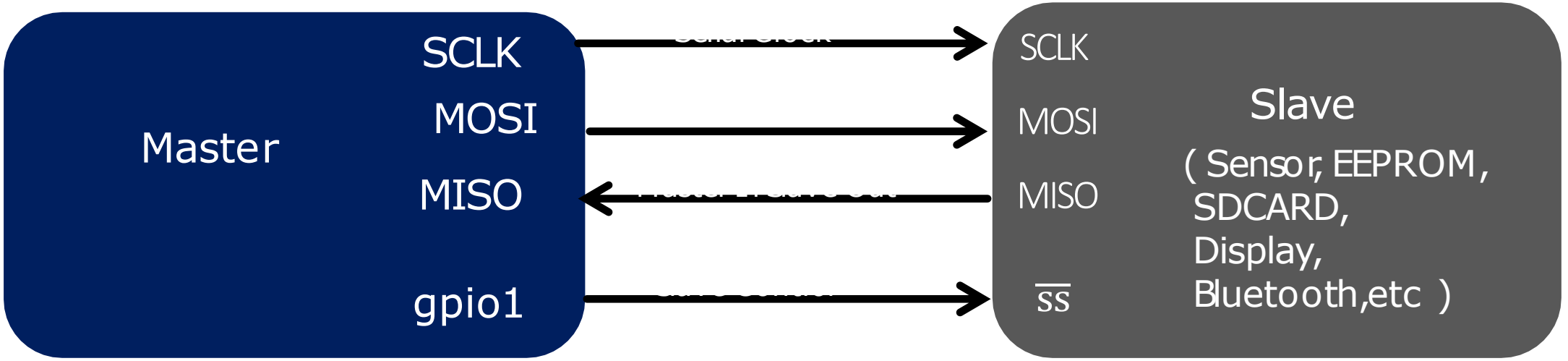
SPI
SERTIAL PERIPHERAL INTERFACE

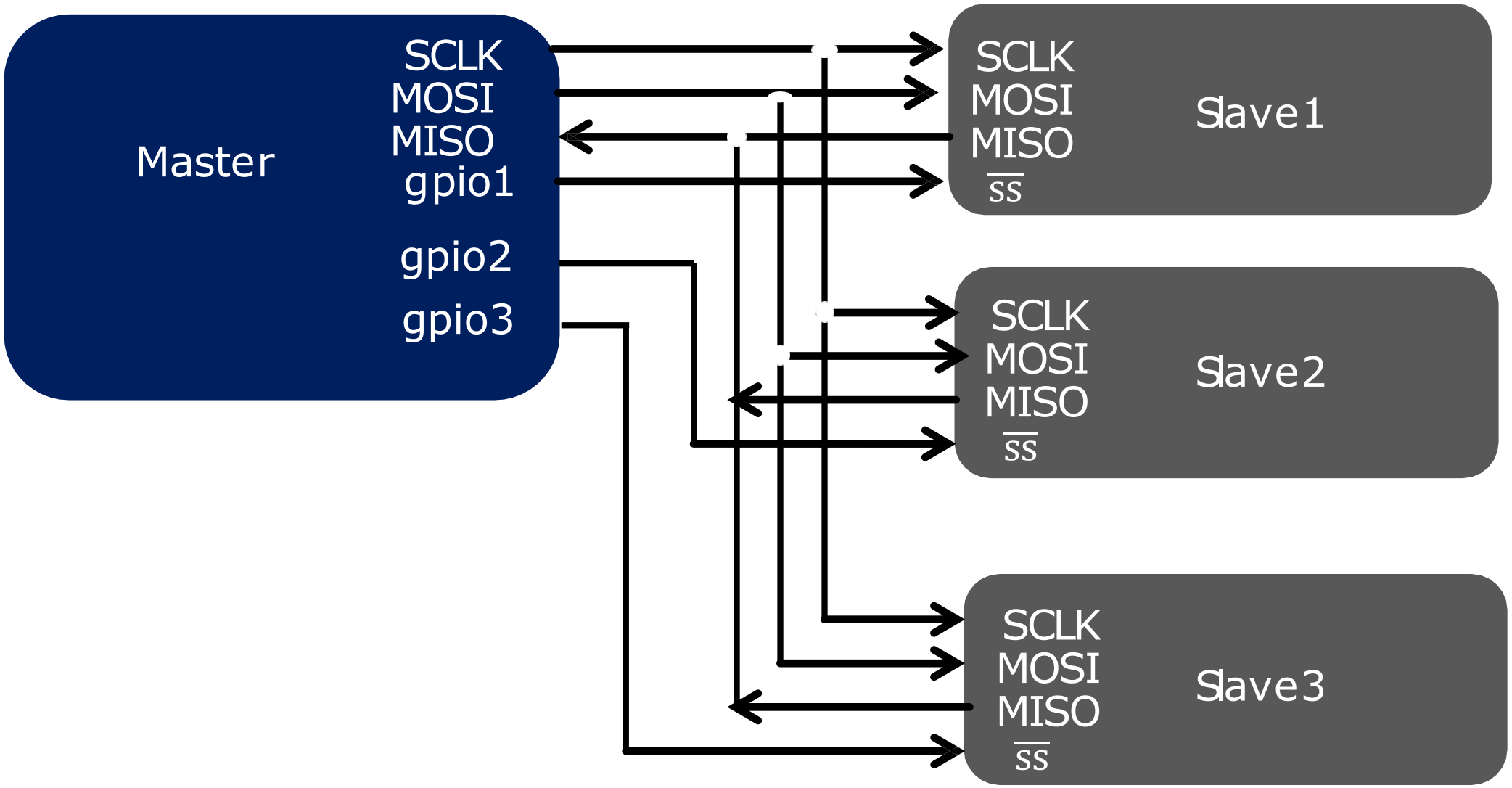
Suresha C
Assistant professor
EEE Dept



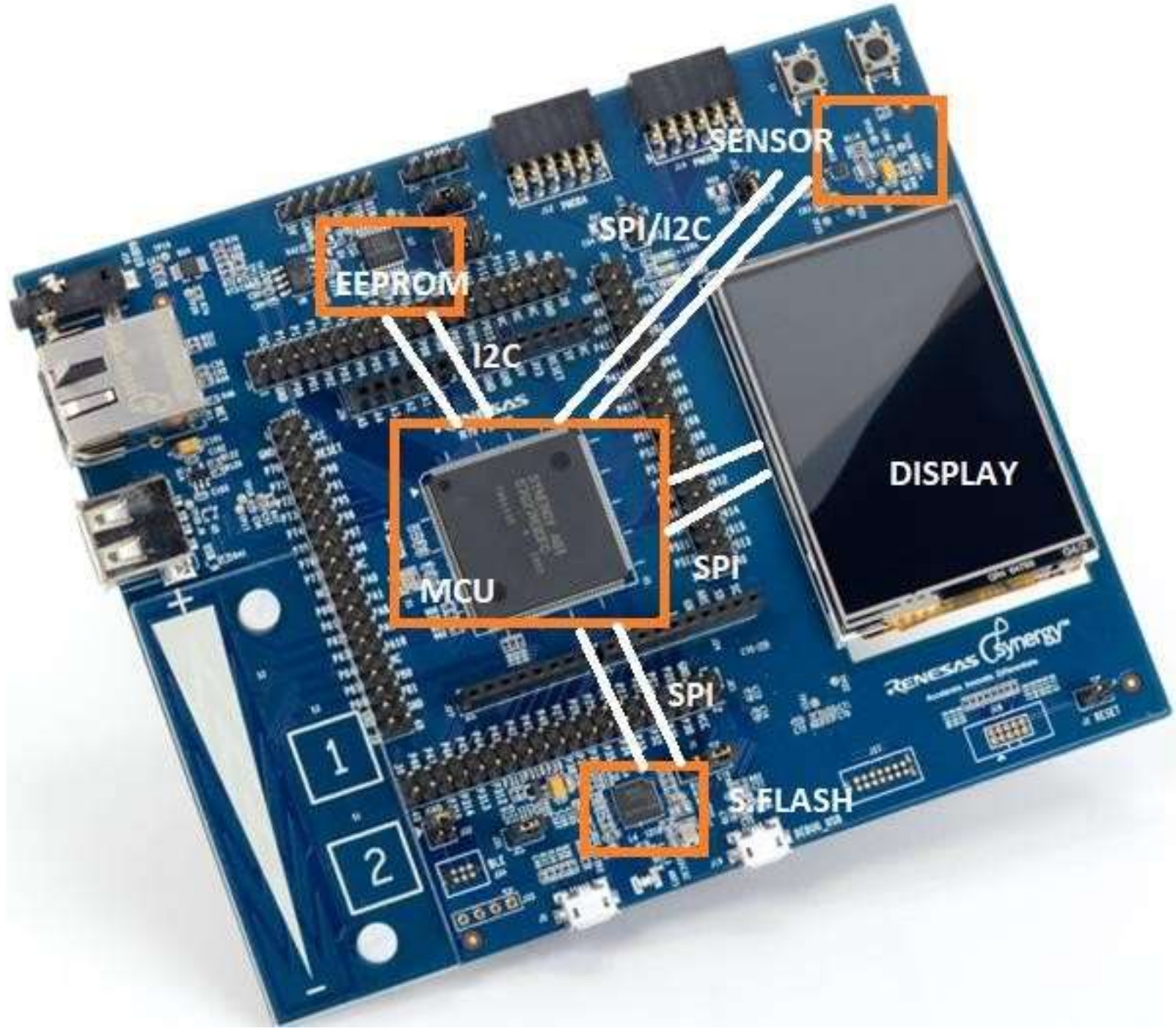
The SPI bus allows the communication between one master device and one or more slave devices.

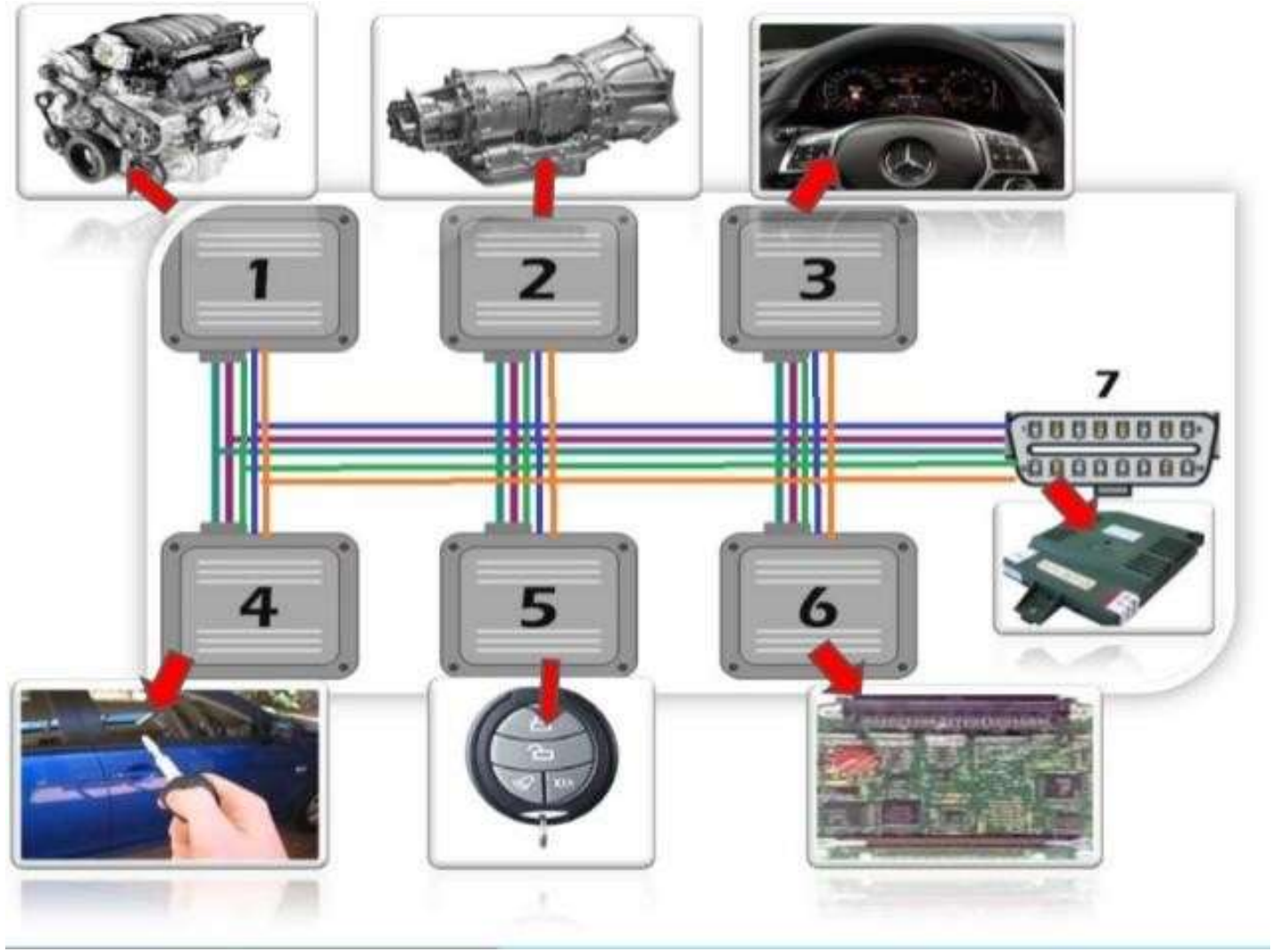


1. Four I/O pins are dedicated to SPI communication with external devices.
2. MISO: Master In / Slave Out data. In the general case, this pin is used to transmit data in slave mode and receive data in master mode
3. MOSI: Master Out / Slave In data. In the general case, this pin is used to transmit data in master mode and receive data in slave mode.
4. SCK: Serial Clock output pin for SPI master and input pin for SPI slaves.
5. NSS: Slave select pin. Depending on the SPI and NSS settings, this pin can be used to select an individual slave device for communication

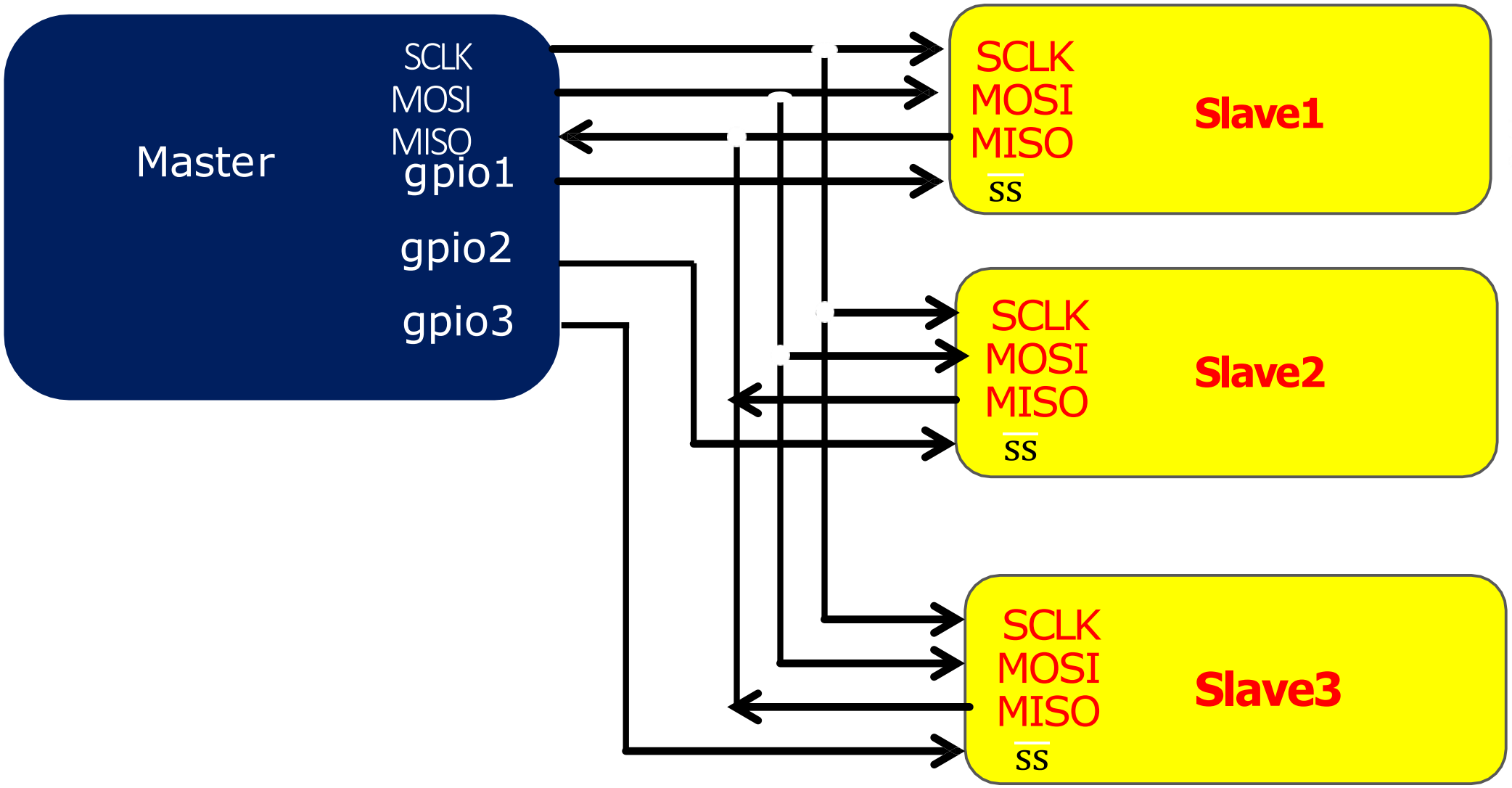

Master
SCLK
MOSI
MISO
gpio1
gpio2
gpio3
SCLK
MOSI
MISO
$\overline{ss}$
Slave1
SCLK
MOSI
MISO
$\overline{ss}$
Slave2
SCLK
MOSI
MISO
$\overline{ss}$
Slave3

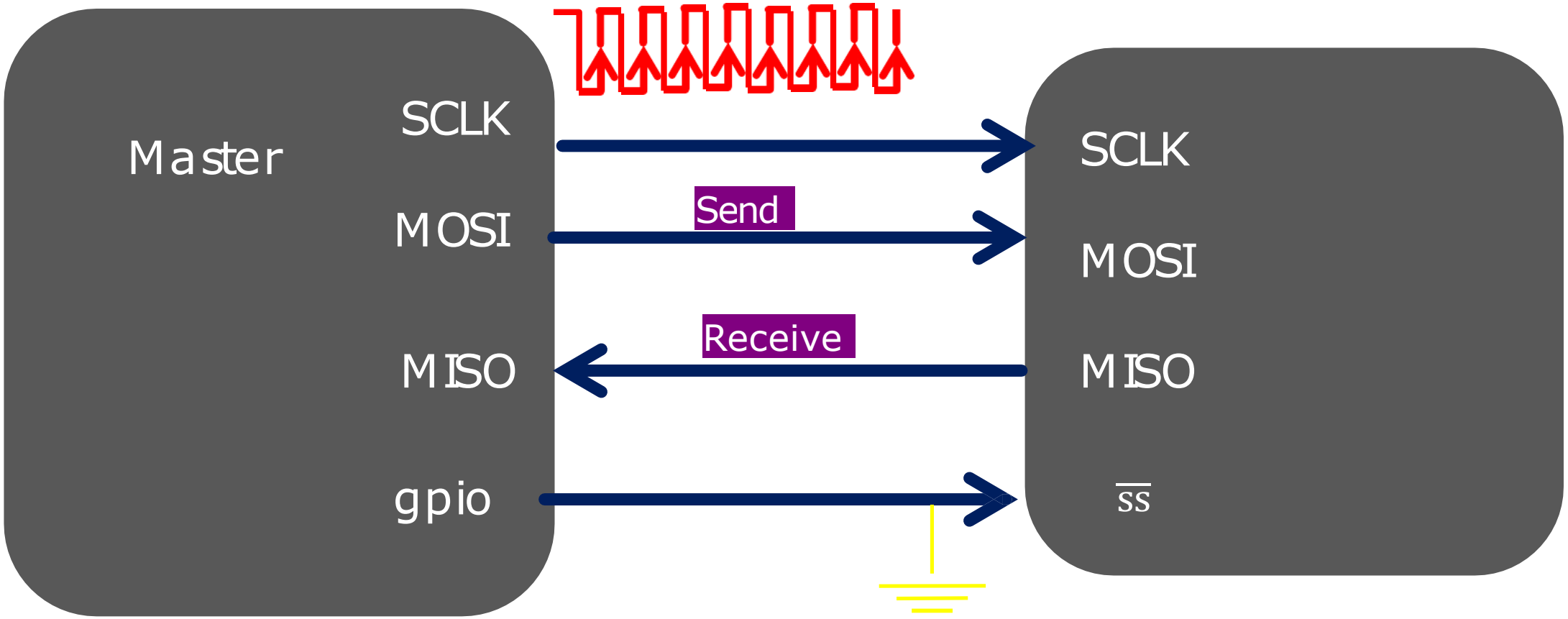
Protocol	Type	Max distance(ft.)	Max Speed (bps)	Typical usage
USB 3.0	dual simplex serial	9 (typical) (up to 49 with 5 hubs)	5 G	Mass storage, video
USB 2.0	half duplex serial	16 (98 ft. with 5 hubs)	1.5M, 12M, 480M	Keyboard, mouse, drive, speakers, printer, camera
Ethernet	serial	1600	10G	network communications
I2C	synchronous serial	18	3.4 M in High-speed mode.	Microcontroller communications
RS-232	asynchronous serial	50–100	20k	Modem, mouse, instrumentation
RS-485	asynchronous serial	4000	10M	Data acquisition and control systemsSPI
SPI	synchronous serial	10	fPCLK/2	


SENSOR
SPI/I2C
EEPROM
I2C
DISPLAY
MCU
SPI
SPI
S.FLASH



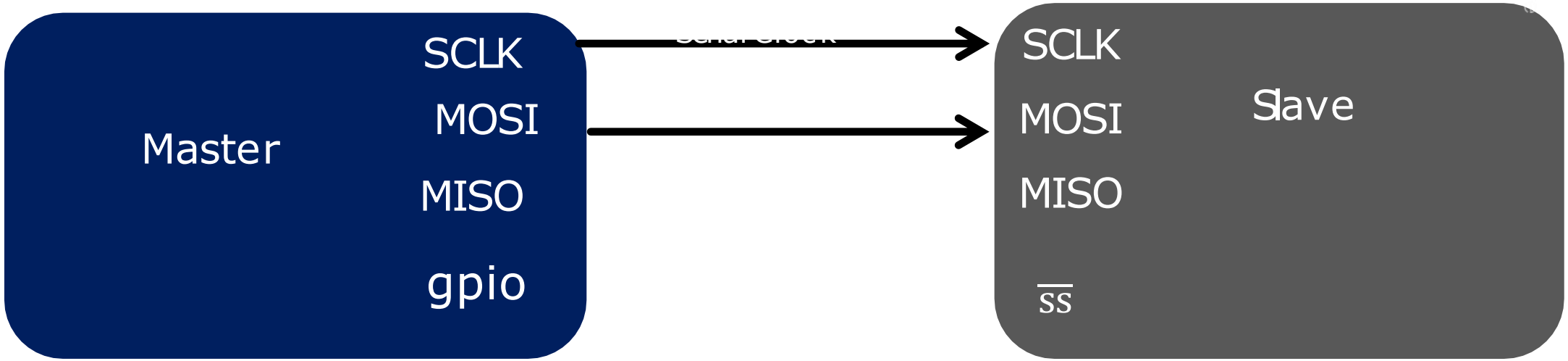
You can use CAN, ETHERNET , RS485, RS232 or combination of them , when you have to cover larger distances and want to achieve better quality of service.


Master
SCLK
MOSI
MISO
gpio1
gpio2
gpio3
SCLK
MOSI
MISO
SS
Slave1
SCLK
MOSI
MISO
SS
Slave2
SCLK
MOSI
MISO
SS
Slave3

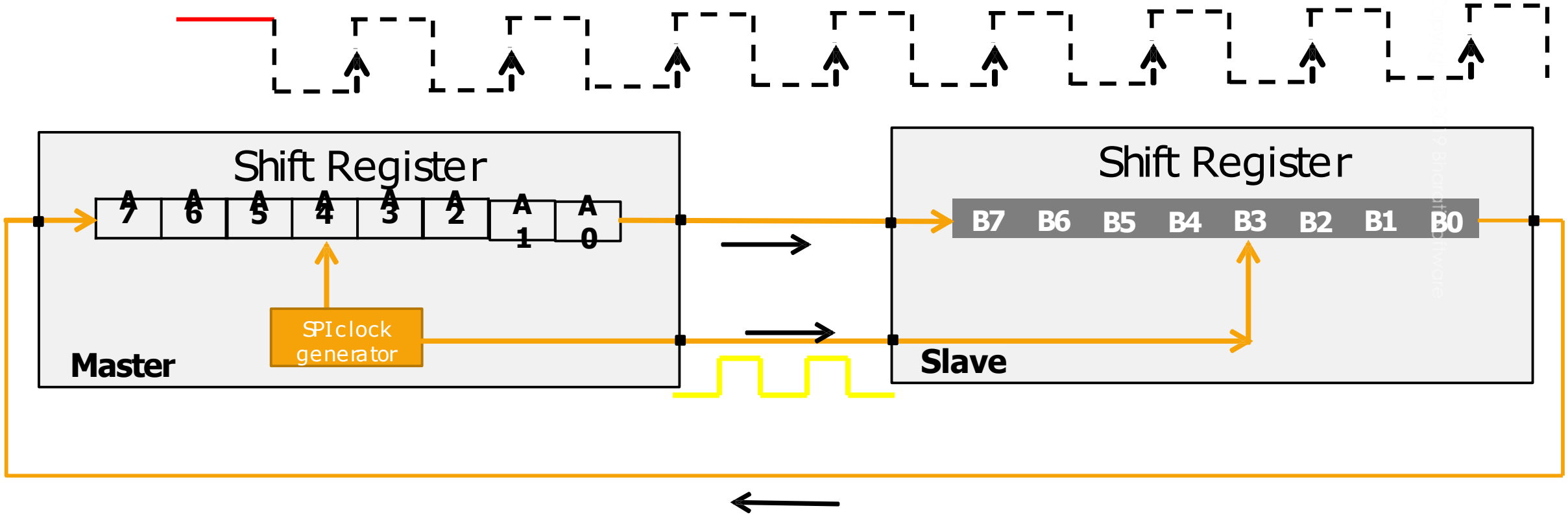

Master
SCLK
MOSI
MISO
gpio
Send
Receive
SCLK
MOSI
MISO
$\overline{ss}$

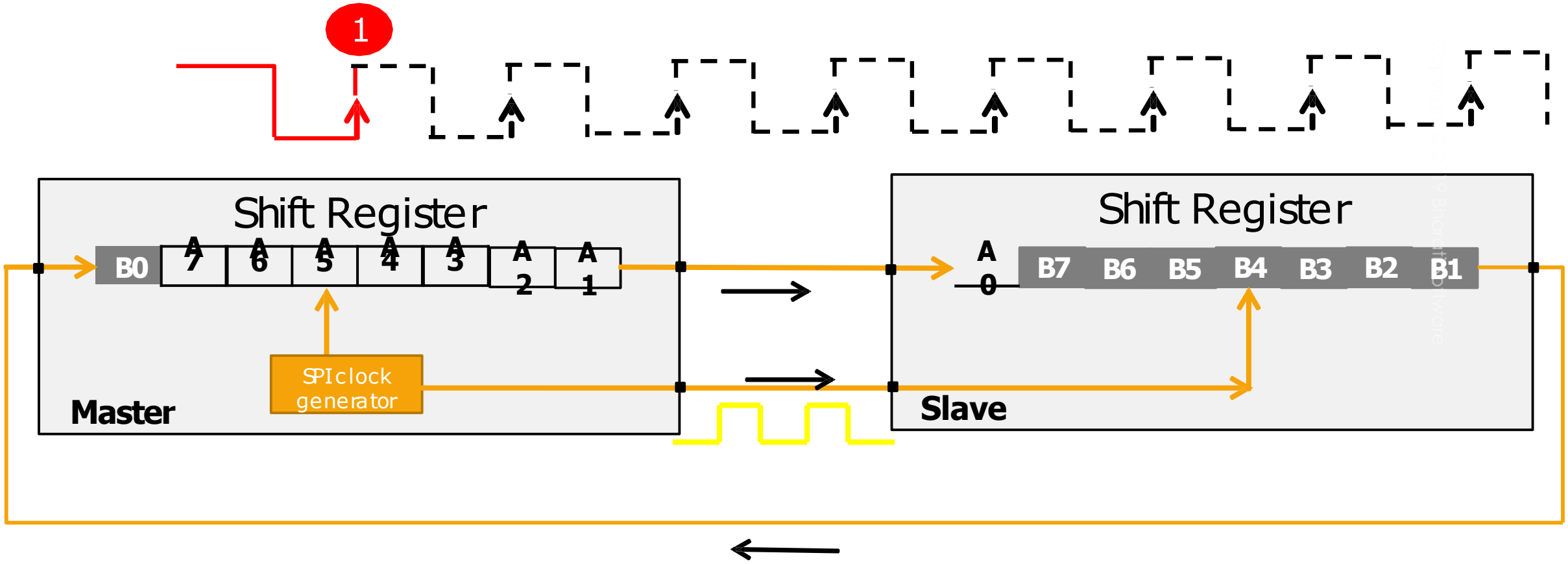
MINIMAL SPI BUS

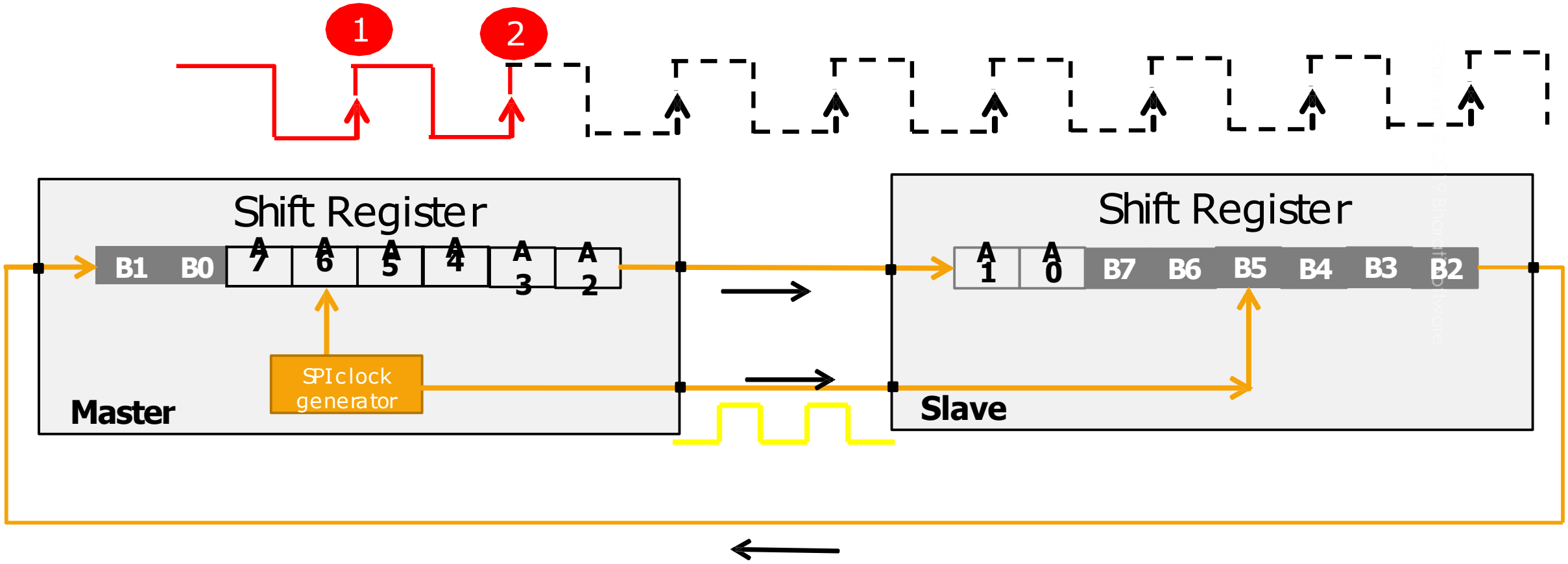
The SPI bus allows the communication between one master device and one or more slave devices. In some applications SPI bus may consists of just two wires - one for the clock signal and the other for synchronous data transfer. Other signals can be added depending on the data exchange between SPI nodes and their slave select signal management.

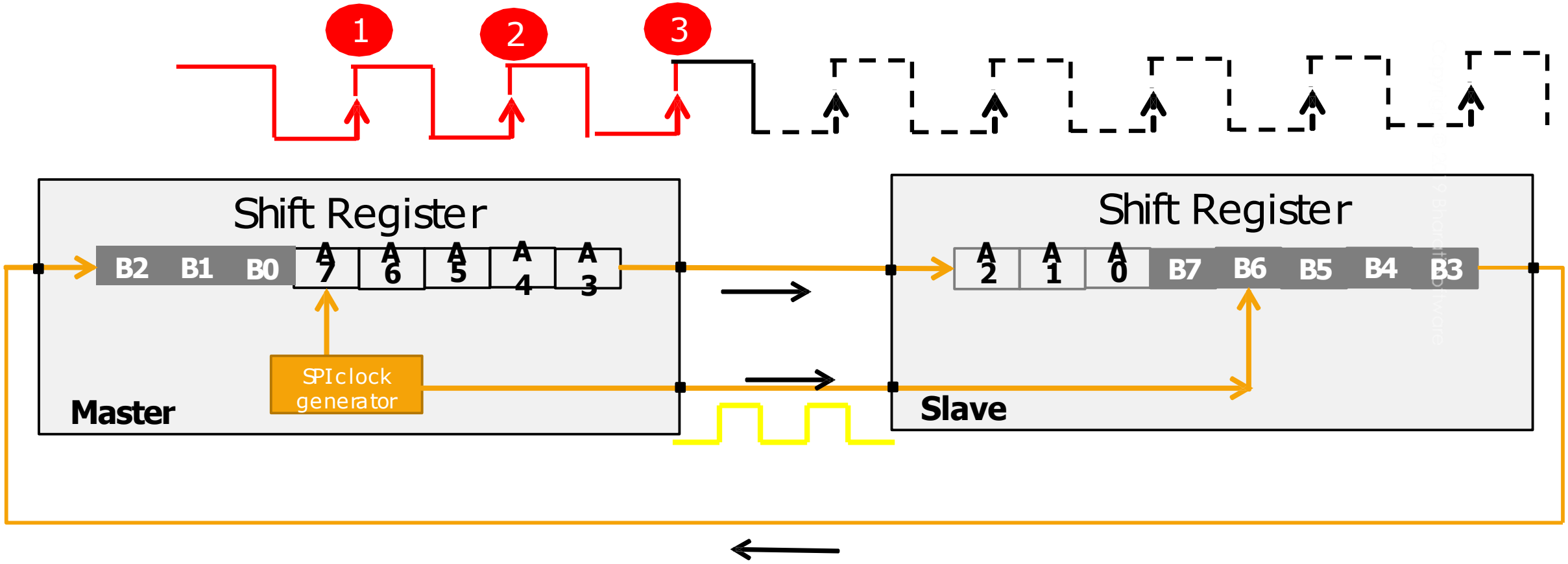


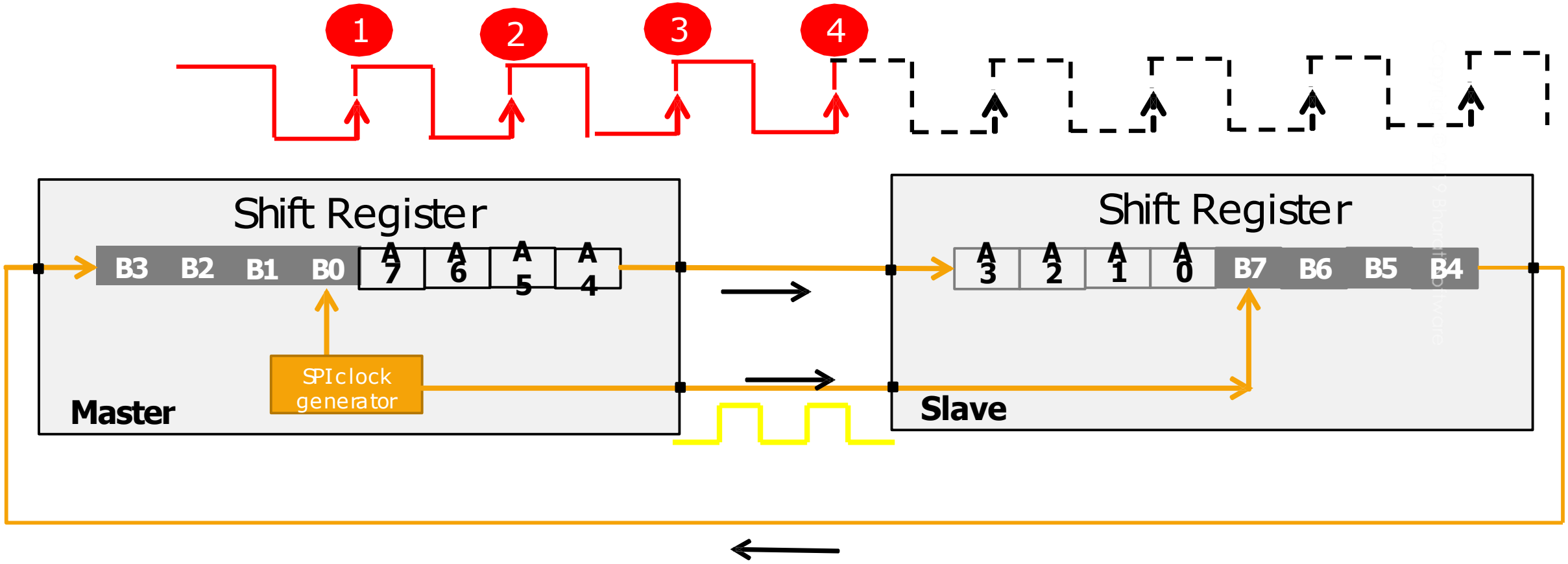
SPI Hardware :Behind the scenes

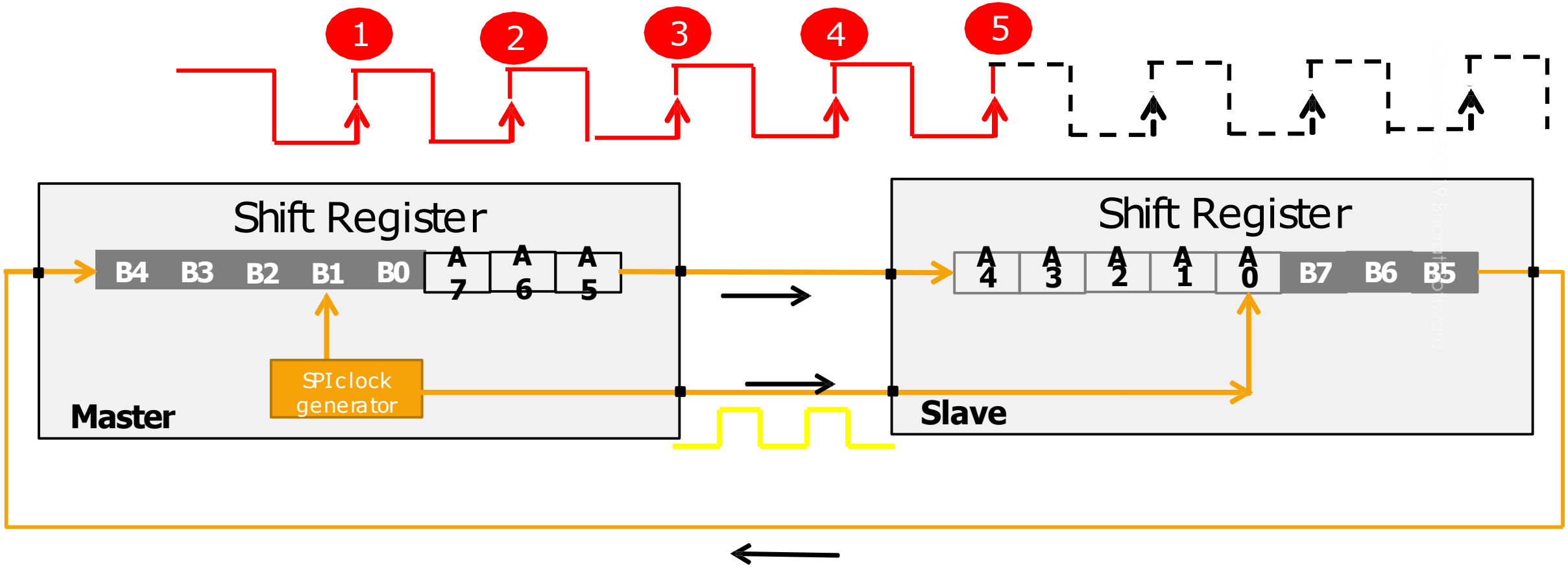

Shift Register
A7
A6
A5
A4
A3
A2
A1
A0
SPI clock generator
Master
Shift Register
B7
B6
B5
B4
B3
B2
B1
B0
Slave

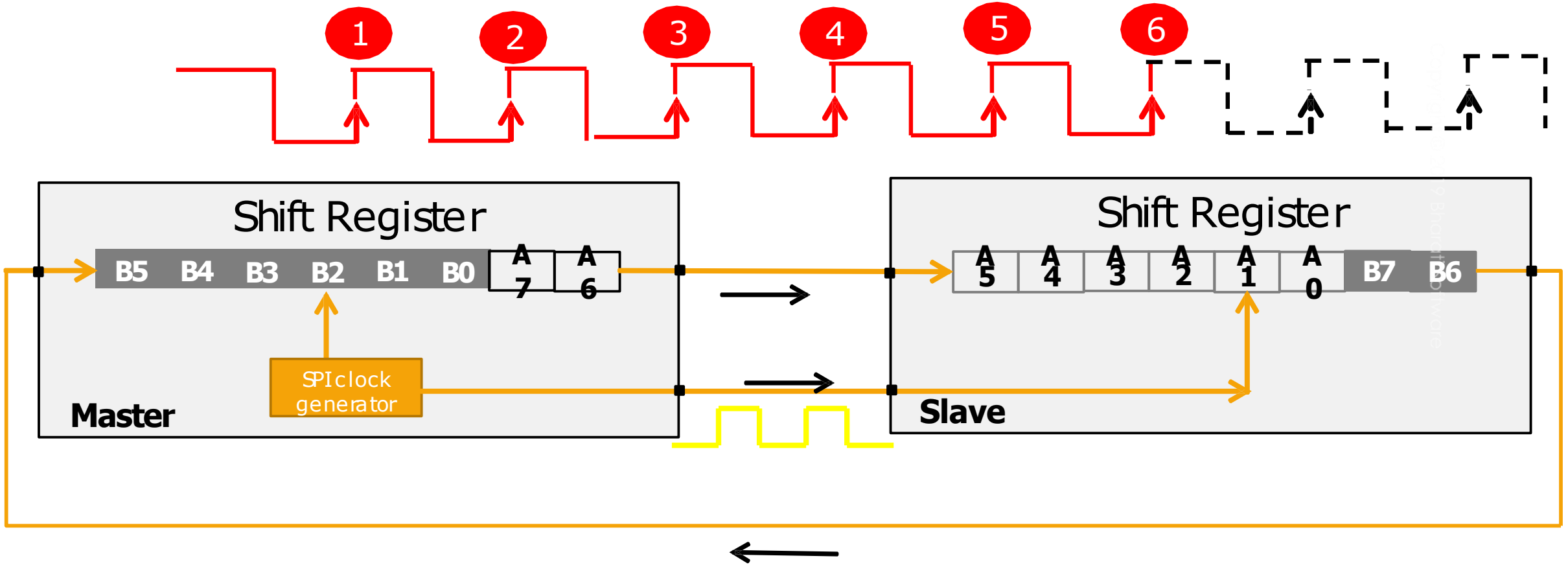

1
Shift Register
B0
A 7
A 6
A 5
A 4
A 3
A 2
A 1
SPI clock generator
Master
Shift Register
A 0
B7
B6
B5
B4
B3
B2
B1
Slave

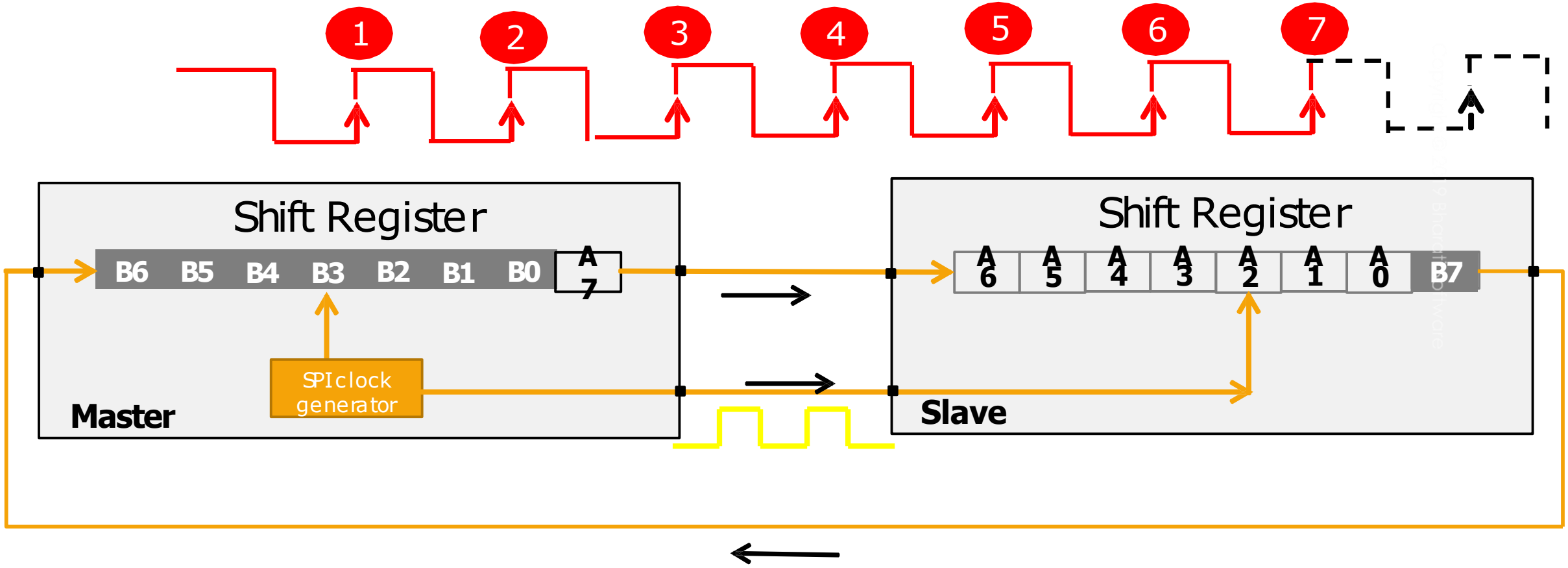

1
2
Shift Register
B1
B0
A7
A6
A5
A4
A3
A2
SPI clock generator
Master
Shift Register
A1
A0
B7
B6
B5
B4
B3
B2
Slave

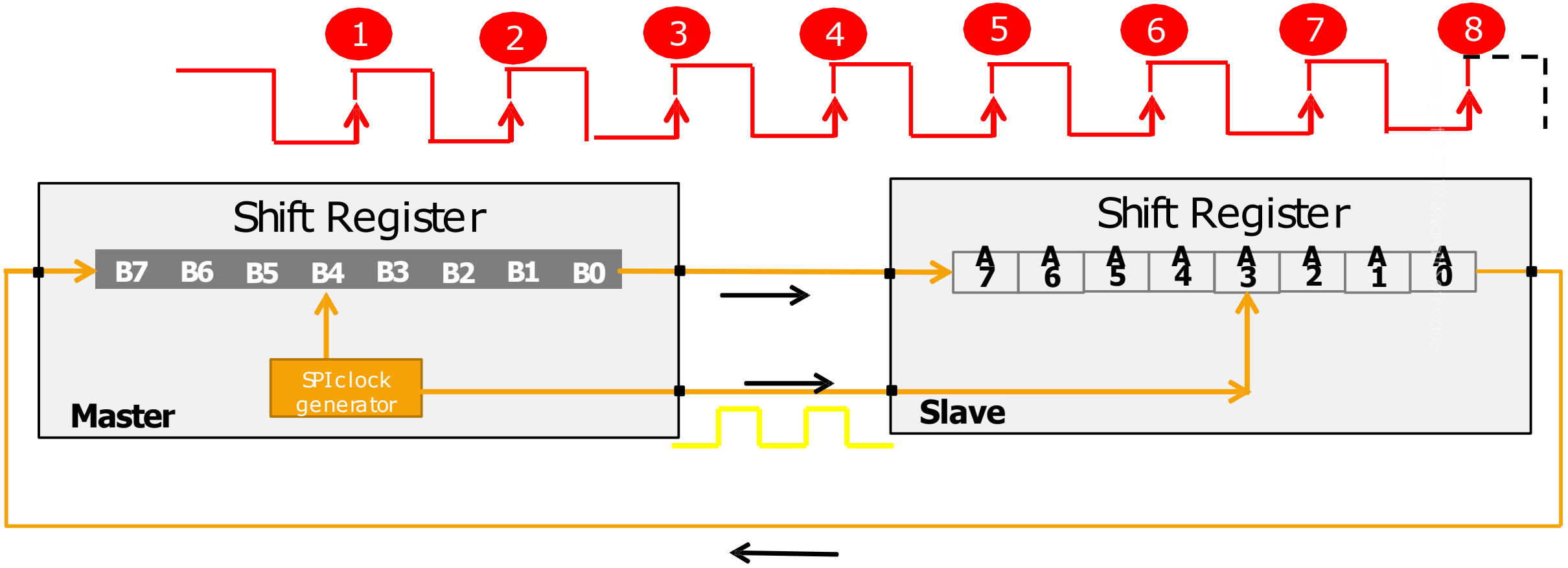

1
2
3
Shift Register
B2
B1
B0
A7
A6
A5
A4
A3
SPI clock generator
Master
Shift Register
A2
A1
A0
B7
B6
B5
B4
B3
Slave


1
2
3
4
Shift Register
B3
B2
B1
B0
A7
A6
A5
A4
SPI clock generator
Master
Shift Register
A3
A2
A1
A0
B7
B6
B5
B4
Slave


1
2
3
4
5
Shift Register
B4
B3
B2
B1
B0
A 7
A 6
A 5
SPI clock generator
Master
Shift Register
A 4
A 3
A 2
A 1
A 0
B7
B6
B5
Slave


1
2
3
4
5
6
Shift Register
B5
B4
B3
B2
B1
B0
A7
A6
SPI clock generator
Master
Shift Register
A5
A4
A3
A2
A1
A0
B7
B6
Slave

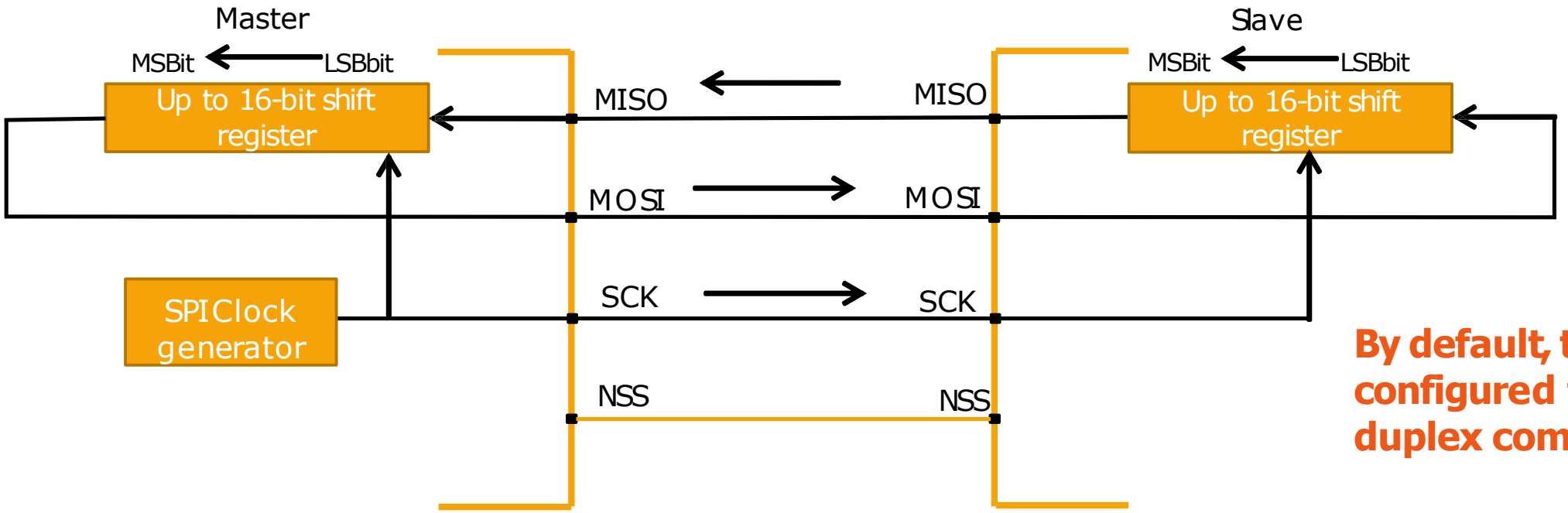

1
2
3
4
5
6
7
Shift Register
B6 B5 B4 B3 B2 B1 B0
A7
SPI clock generator
Master
Shift Register
A6 A5 A4 A3 A2 A1 A0 B7
Slave


1
2
3
4
5
6
7
8
Shift Register
B7 B6 B5 B4 B3 B2 B1 B0
SPI clock generator
Master
Shift Register
A7 A6 A5 A4 A3 A2 A1 A0
Slave

CUSTOMIZING SPI BUS : BUS CONFIGURATIONS

The SPI allows the MCU to communicate using different configurations, depending on the device targeted and the application requirements.

FULL-DUPLEX COMMUNICATION

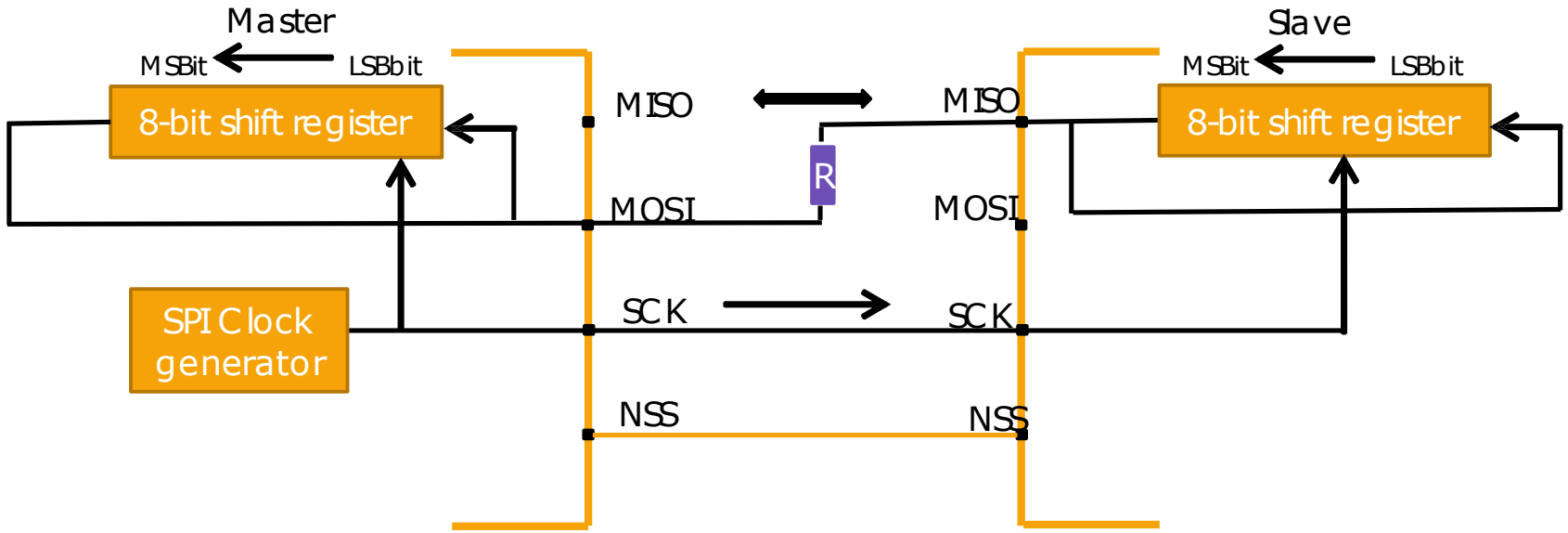


By default, the SPI is configured for full-duplex communication

In this configuration, the shift registers of the master and slave are linked using two unidirectional lines between the MOSI and the MISO pins. During SPI communication, data is shifted synchronously on the SCK clock edges provided by the master. The master transmits the data to be sent to the slave via the MOSI line and receives data from the slave via the MISO line.

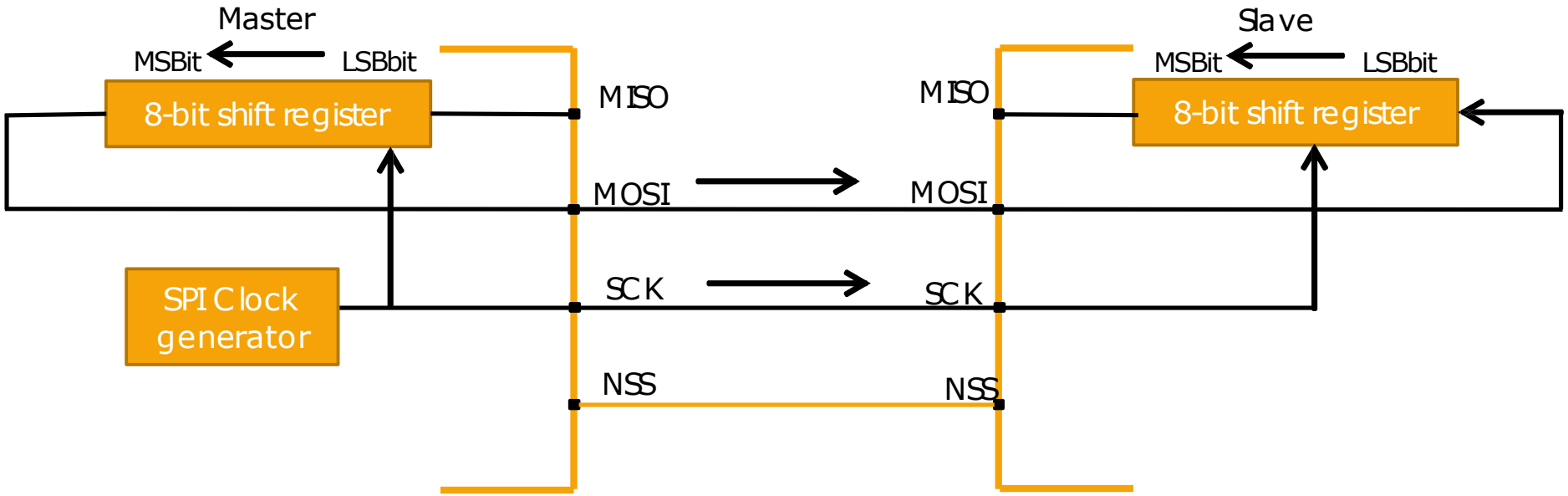
Remember that in SPI communication ,slave will not initiate data transfer unless master produces the clock.

HALF-DUPLEX COMMUNICATION



In this configuration, one single cross connection line is used to link the shift registers of the master and slave together. During this communication, the data is synchronously shifted between the shift registers on the SCK clock edge in the transfer direction selected reciprocally by both master and slave

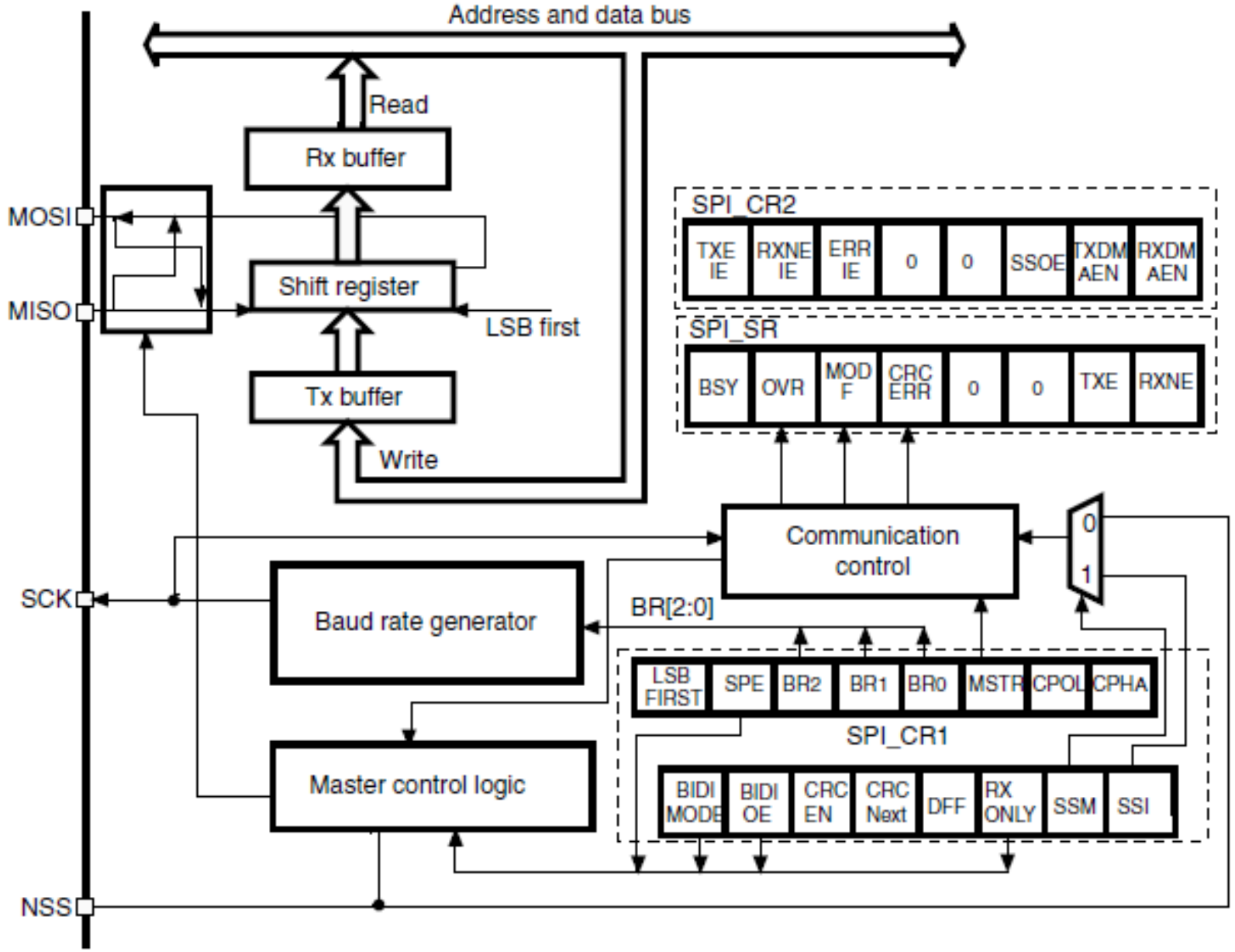
SIMPLEX COMMUNICATION



Simplex single master, single slave application (master in transmit-only/slave in receive-only mode)

Transmit-only , Receive Only mode :
The configuration settings are the same as for full- duplex. The application has to ignore the information captured on the unused input pin. This pin can be used as a standard GPIO

STM32 SPI FUNCTIONAL BLOCK DIAGRAM


Address and data bus
Read
Rx buffer
MOSI
MISO
Shift register
LSB first
Tx buffer
Write
SPI_CR2
TXE IE
RXNE IE
ERR IE
0
0
SSOE
TXDM AEN
RXDM AEN
SPI_SR
BSY
OVR
MOD F
CRC ERR
0
0
TXE
RXNE
Communication control
0
1
SCK
Baud rate generator
BR[2:0]
LSB FIRST
SPE
BR2
BR1
BR0
MSTR
CPOL
CPHA
SPI_CR1
Master control logic
BIDI MODE
BIDI OE
CRC EN
CRC Next
DFF
RX ONLY
SSM
SSI
NSS

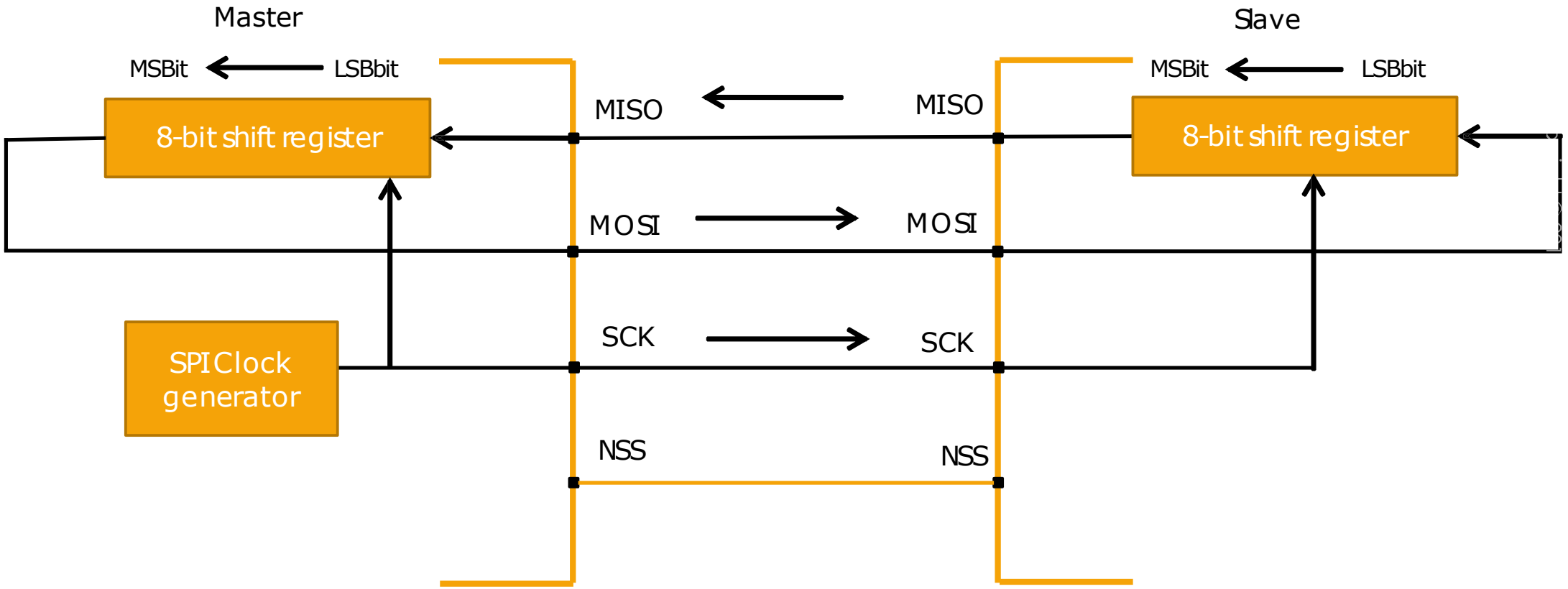
SLAVE SELECT (NSS) PIN MANAGEMENT

When a device is slave mode:

In slave mode, the NSS works as a standard "chip select" input and lets the slave communicate with the master.

When a device is master:

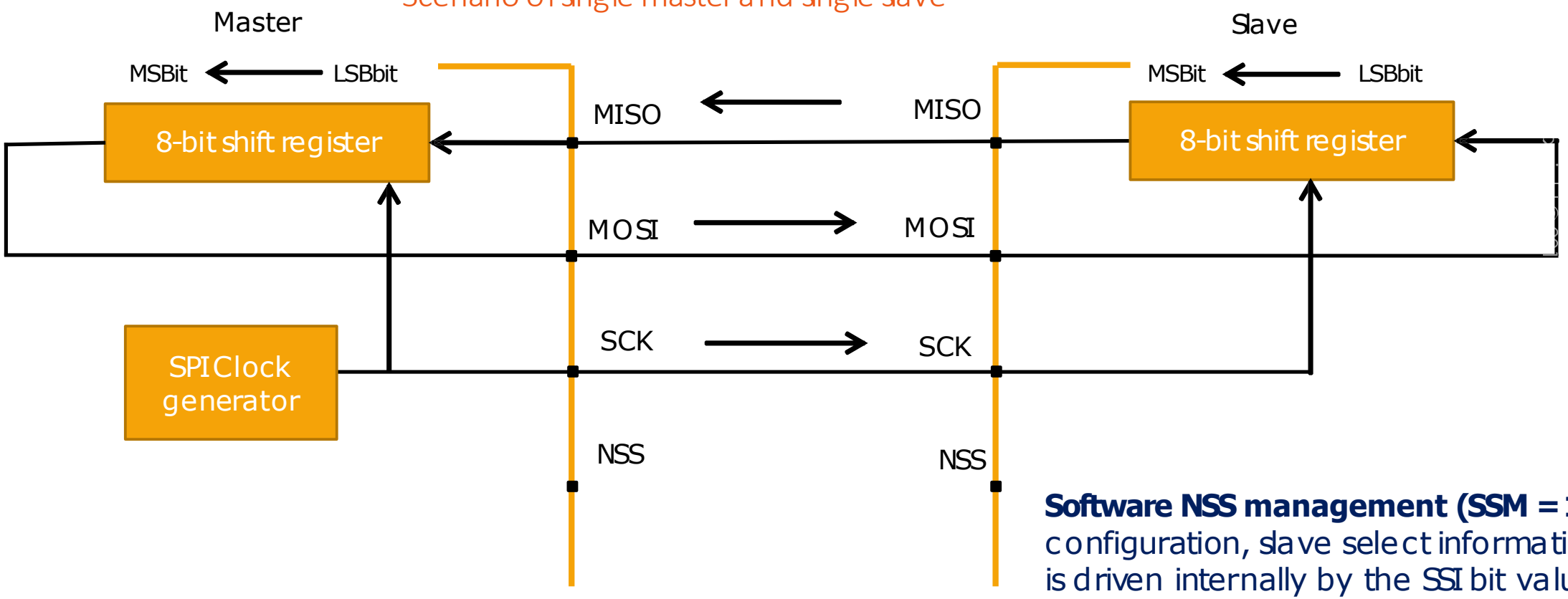
In master mode, NSS can be used either as output or input. As an input it can prevent multi-master bus collision, and as an output it can drive a slave select signal of a single slave.



Scenario of single master and single slave

Hardware slave management
Software slave management

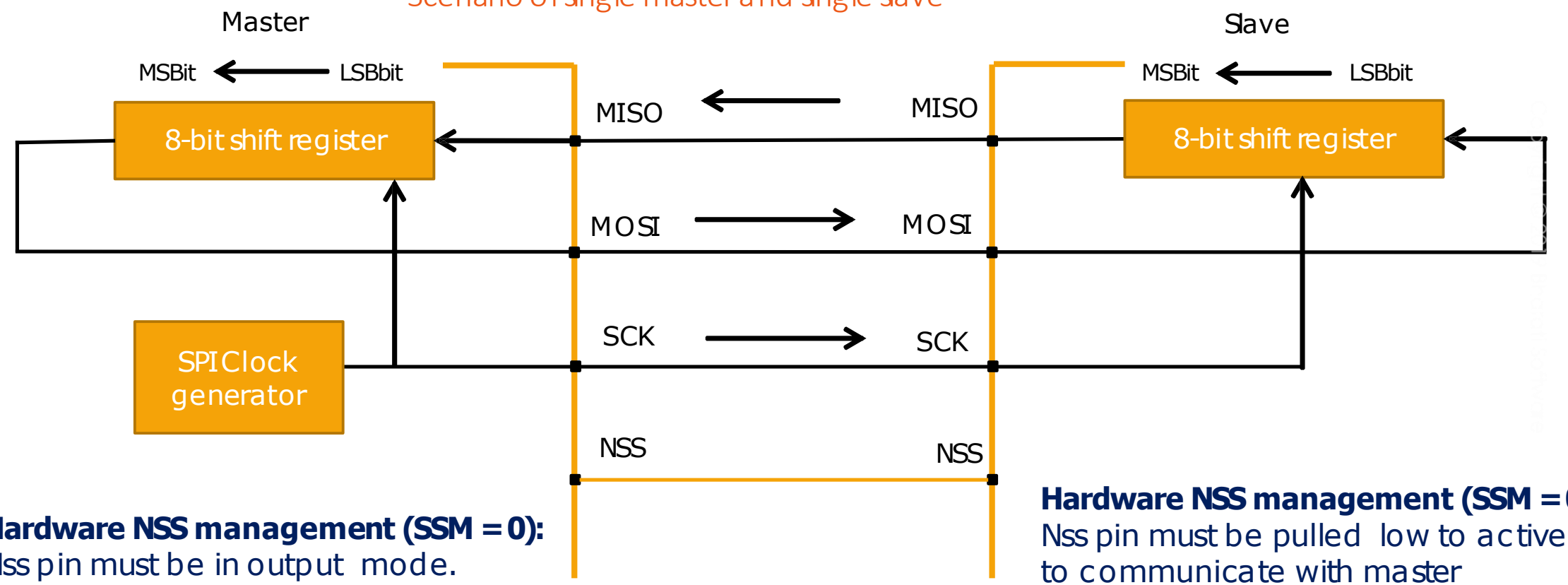
Scenario of single master and single slave



Software NSS management (SSM = 1): in this configuration, slave select information is driven internally by the SSI bit value in register SPIx_CR1. The external NSS pin is free for other application uses.

Hardware or software slave select management can be set using the SSM bit in the SPIx_CR1 register:

Scenario of single master and single slave

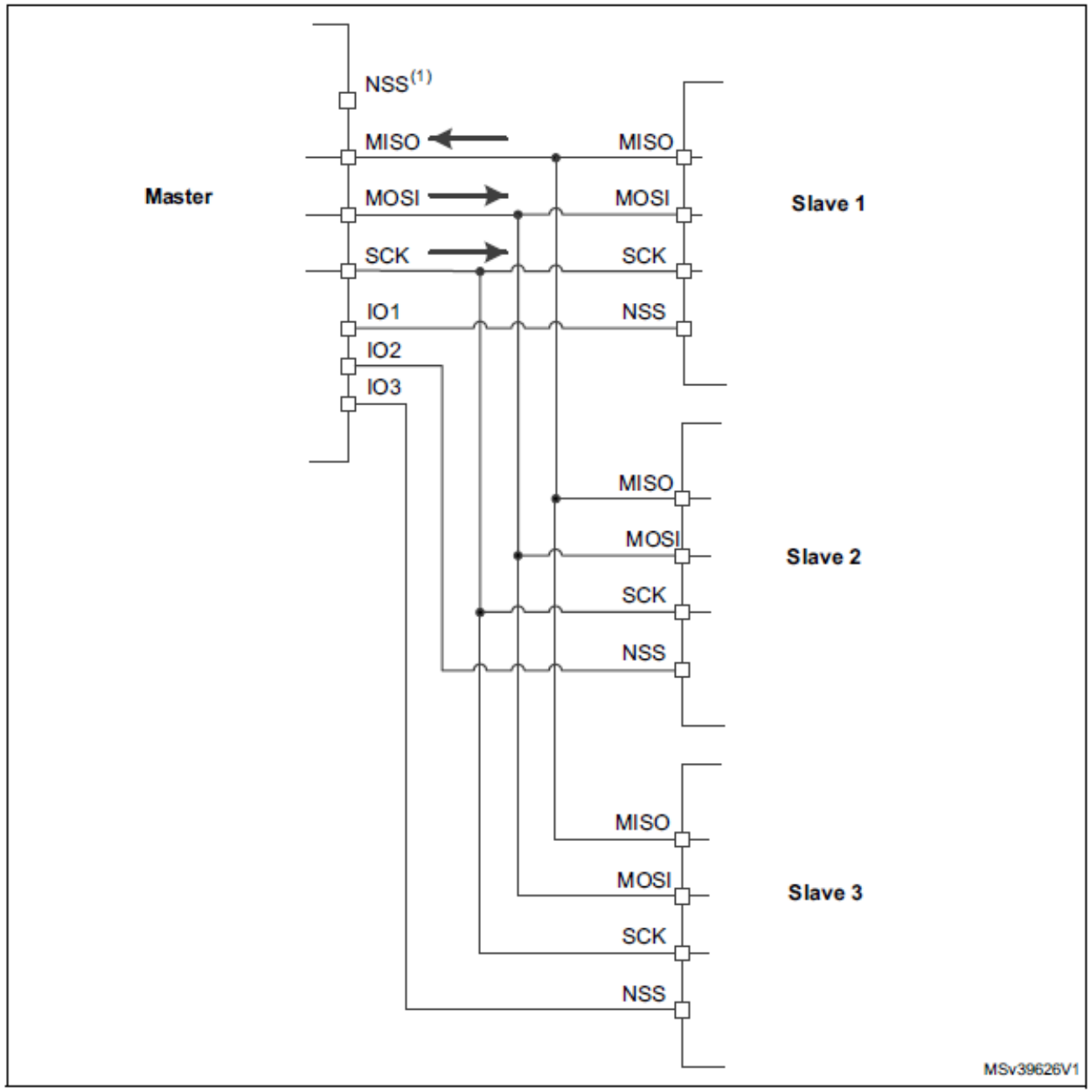


Hardware NSS management (SSM = 0):
Nss pin must be in output mode.
The NSS pin is managed by the hardware

Hardware NSS management (SSM = 0):
Nss pin must be pulled low to active salve to communicate with master

Hardware or software slave select management can be set using the SSM bit in the SPIx_CR1 register:

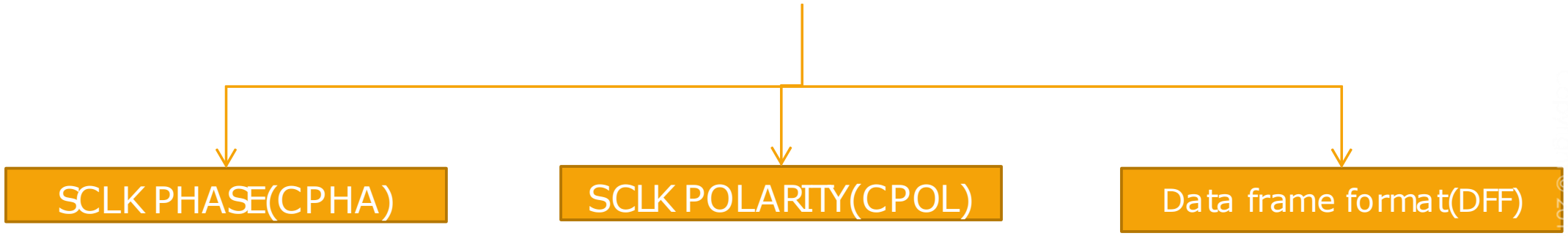
Master and three independent slaves



In this application you cannot use software slave management . You have to use hardware slave management

SPI Communication Format

SPI Communication Format



SPI communication, receive and transmit operations are performed simultaneously.

•The serial clock (SCK) synchronizes the shifting and sampling of the information on the data lines

•The communication format depends on the clock phase, the clock polarity and the data frame format. To be able to communicate together, the master and slaves devices must follow the same communication format.

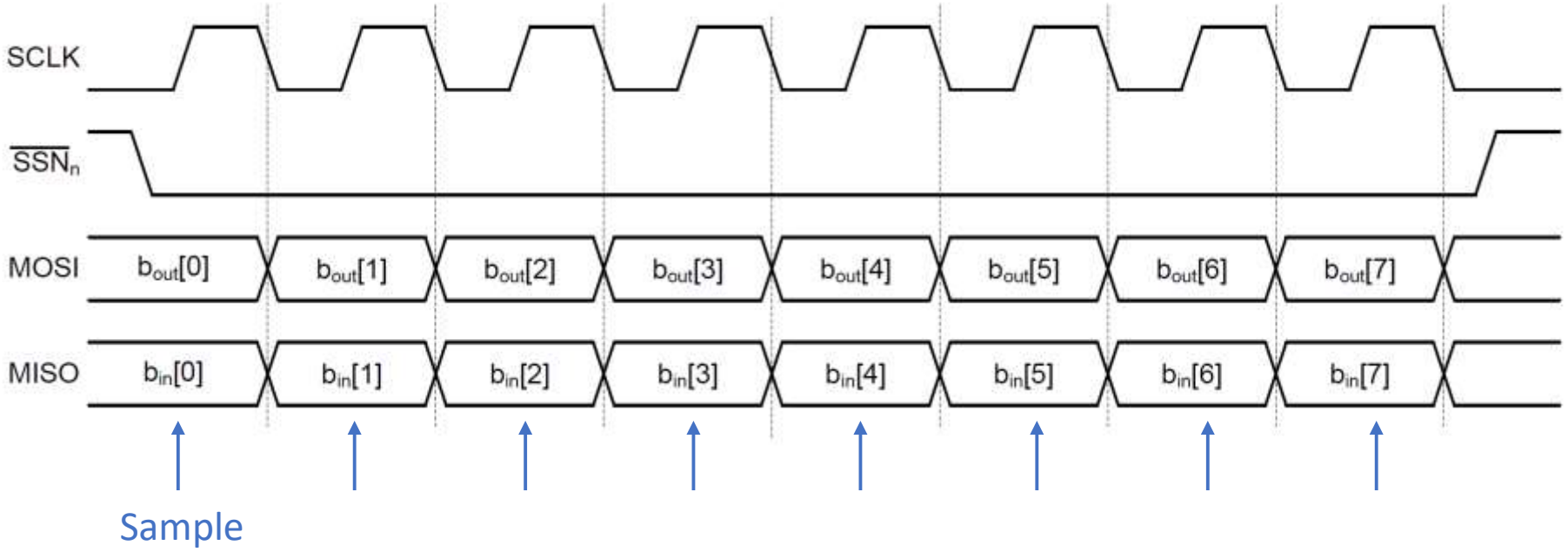
CPOL(CLOCK POLARITY)

- The CPOL (clock polarity) bit controls the idle state value of the clock when no data is being transferred
- If CPOL is reset, the SCLK pin has a low-level idle state. If CPOL is set, the SCLK pin has a high-level idle state.

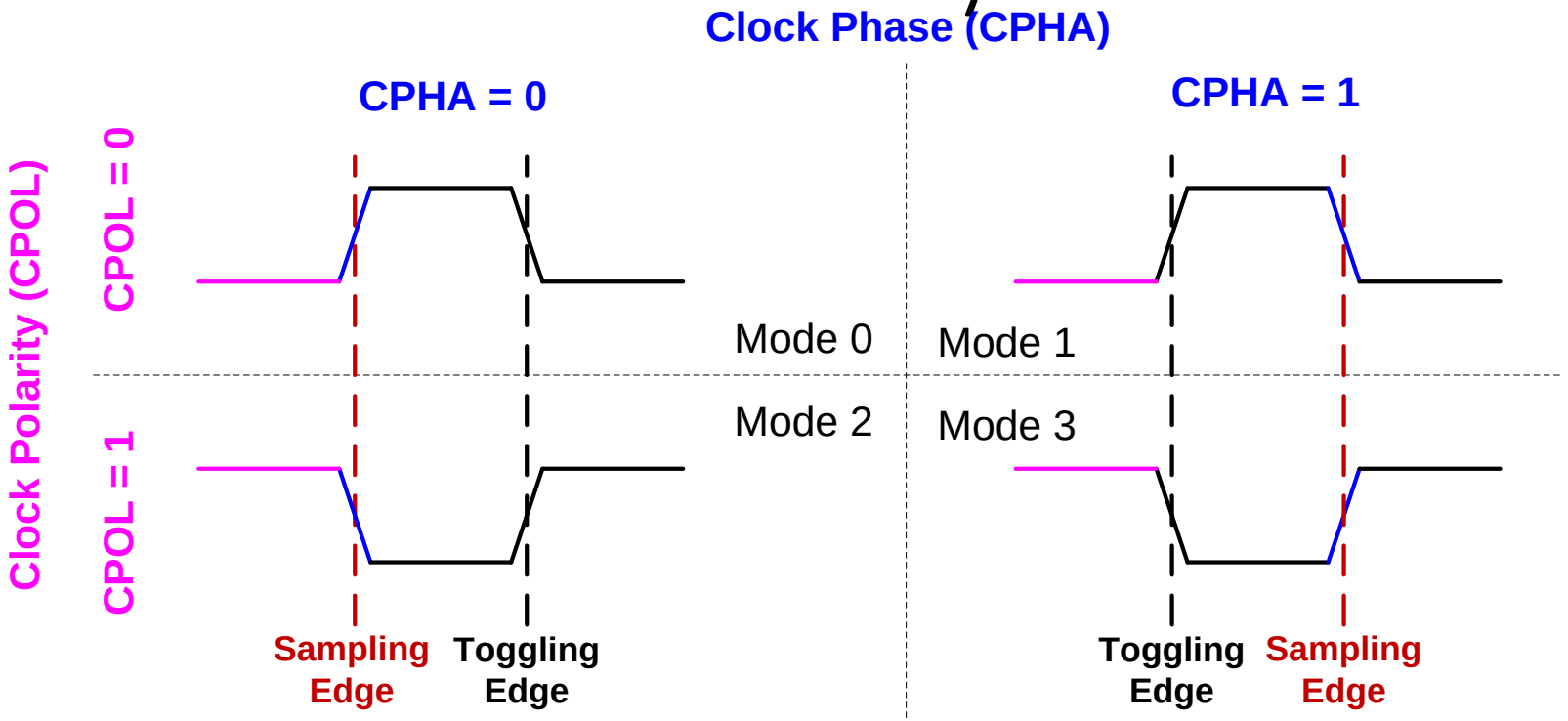
CPOL =0

CPOL =1

SPI Timing Diagram

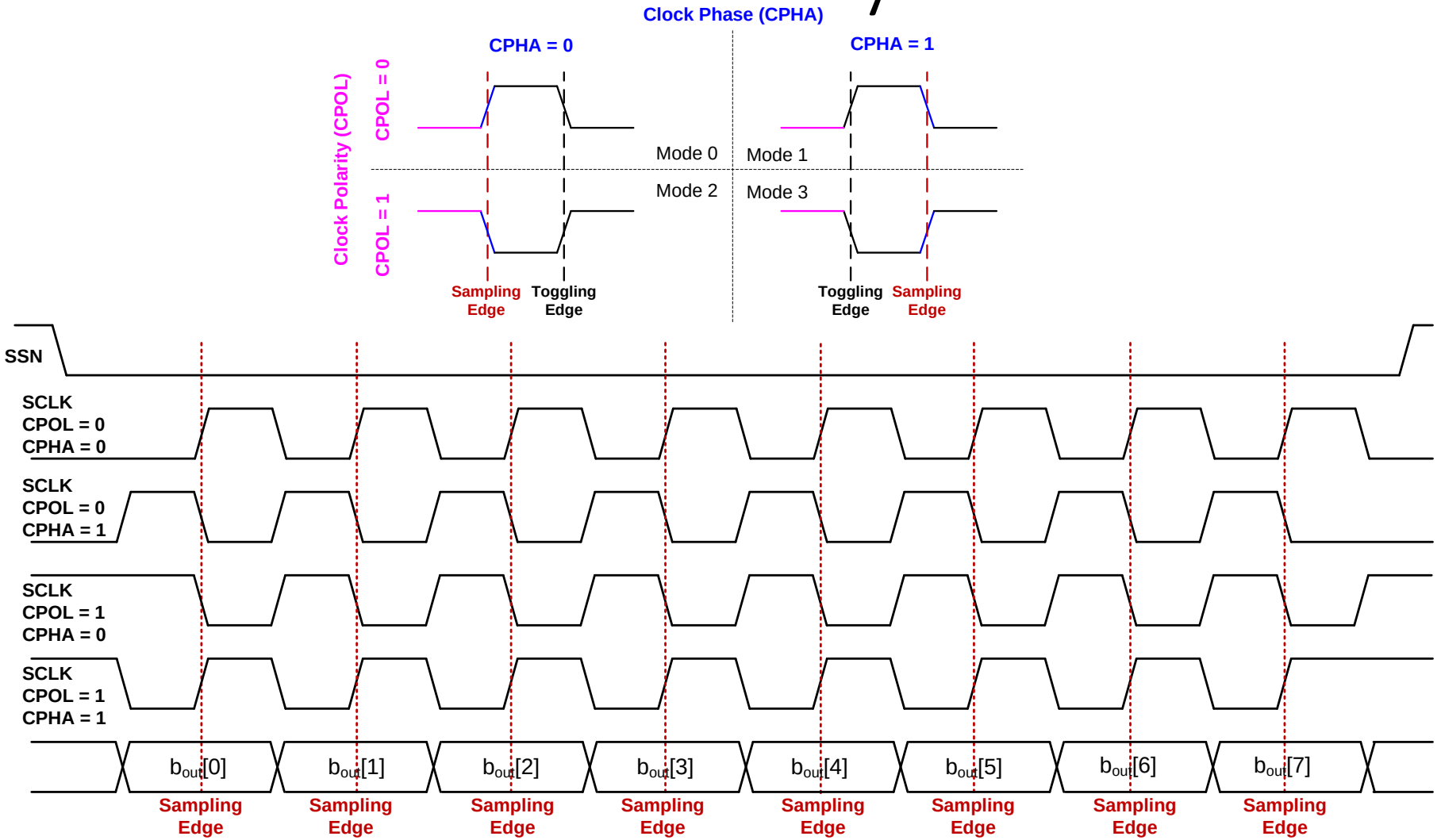


SPI Clock Phase and Polarity



- Combination of CPOL and CPHA determines the clock edge for transmitting and receiving.
- CPOL = 0 ⟶ SCLK is pushed to low during idle. Otherwise, pulled to high during idle.
- CPHA = 0 ⟶ the first clock transition (either rising or falling) is the first data capture edge. Otherwise, the second clock transition is the first data capture edge.

SPI Clock Phase and Polarity



CPHA(CLOCK PHASE)

- CPHA controls at which clock edge of the SCLK(1st or 2nd) the data should be sampled by the slave.
- The combination of CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edge.

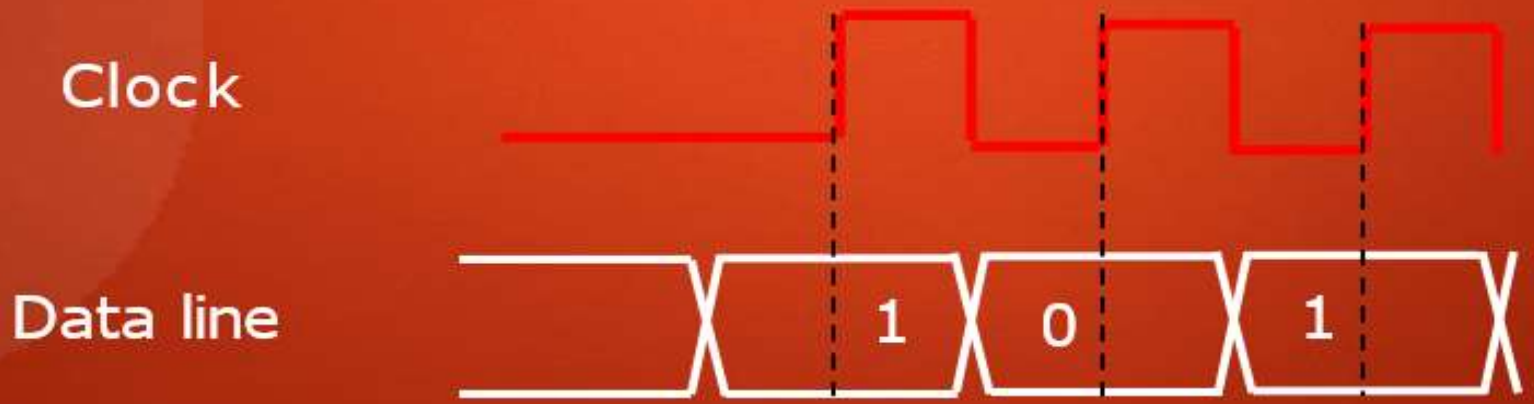
CPHASE=1

- Data will be sampled on the *trailing edge* of the clock.

- If **CPHASE=0**
- Data will be sampled on the *leading edge* of the clock.

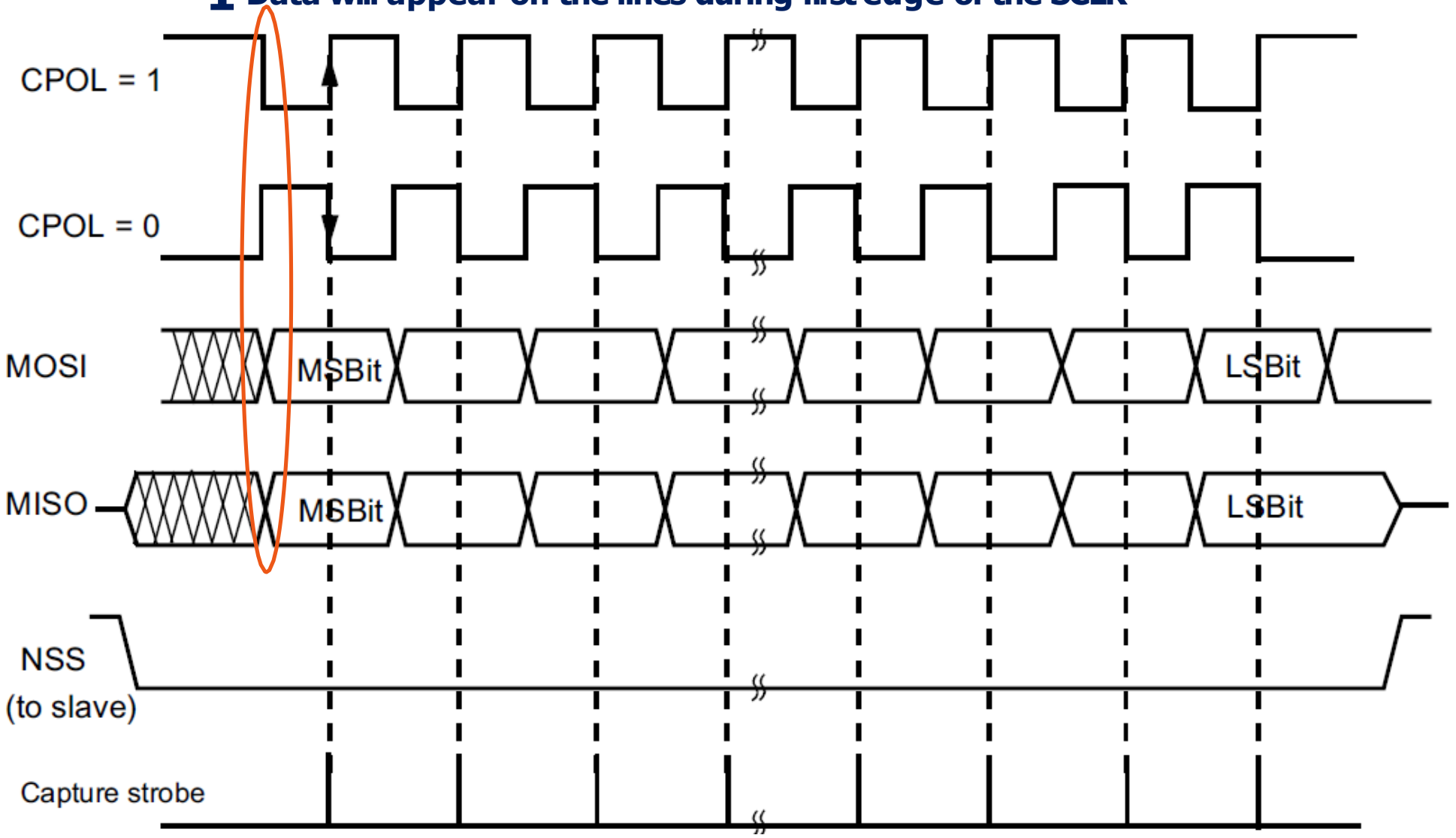
Data toggling means, data transition to the next b

Data sampling means, sampling the data line to capture the data.

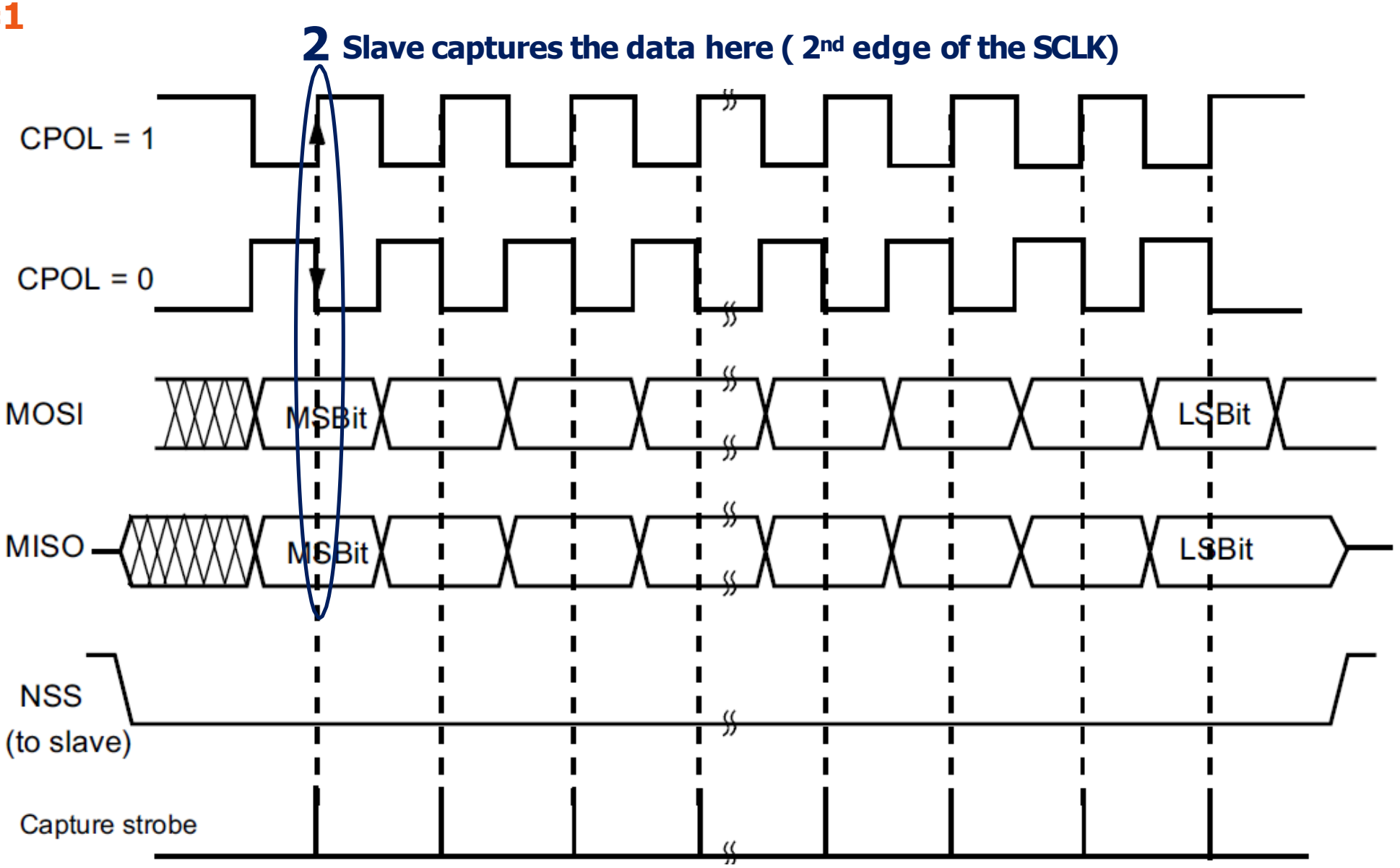


CPHA=1

1 Data will appear on the lines during first edge of the SCLK



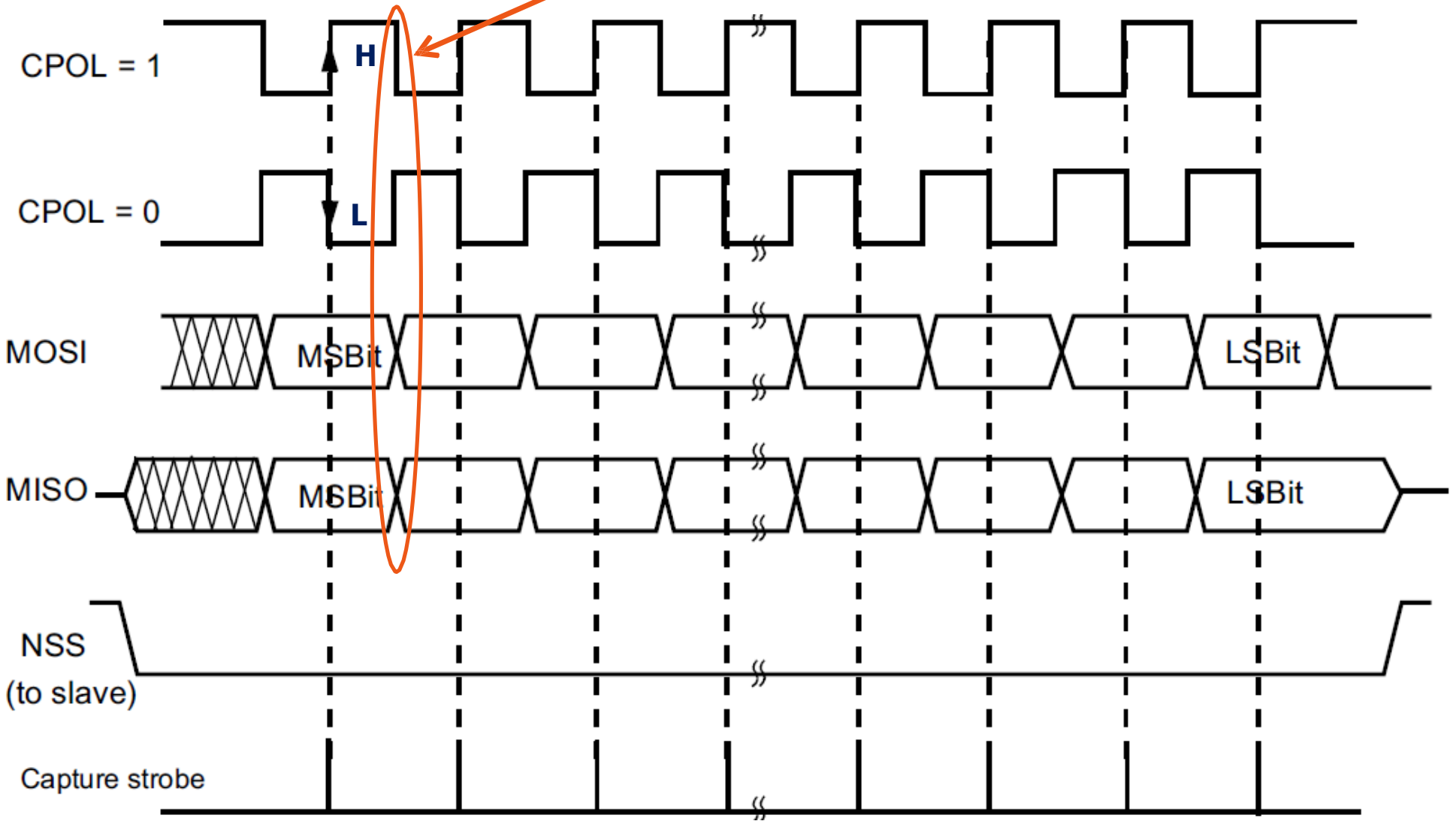
CPHA=1


2 Slave captures the data here (2nd edge of the SCLK)
CPOL = 1
CPOL = 0
MOSI
MSBit
LSBit
MISO
MSBit
LSBit
NSS
(to slave)
Capture strobe

CPHA=1

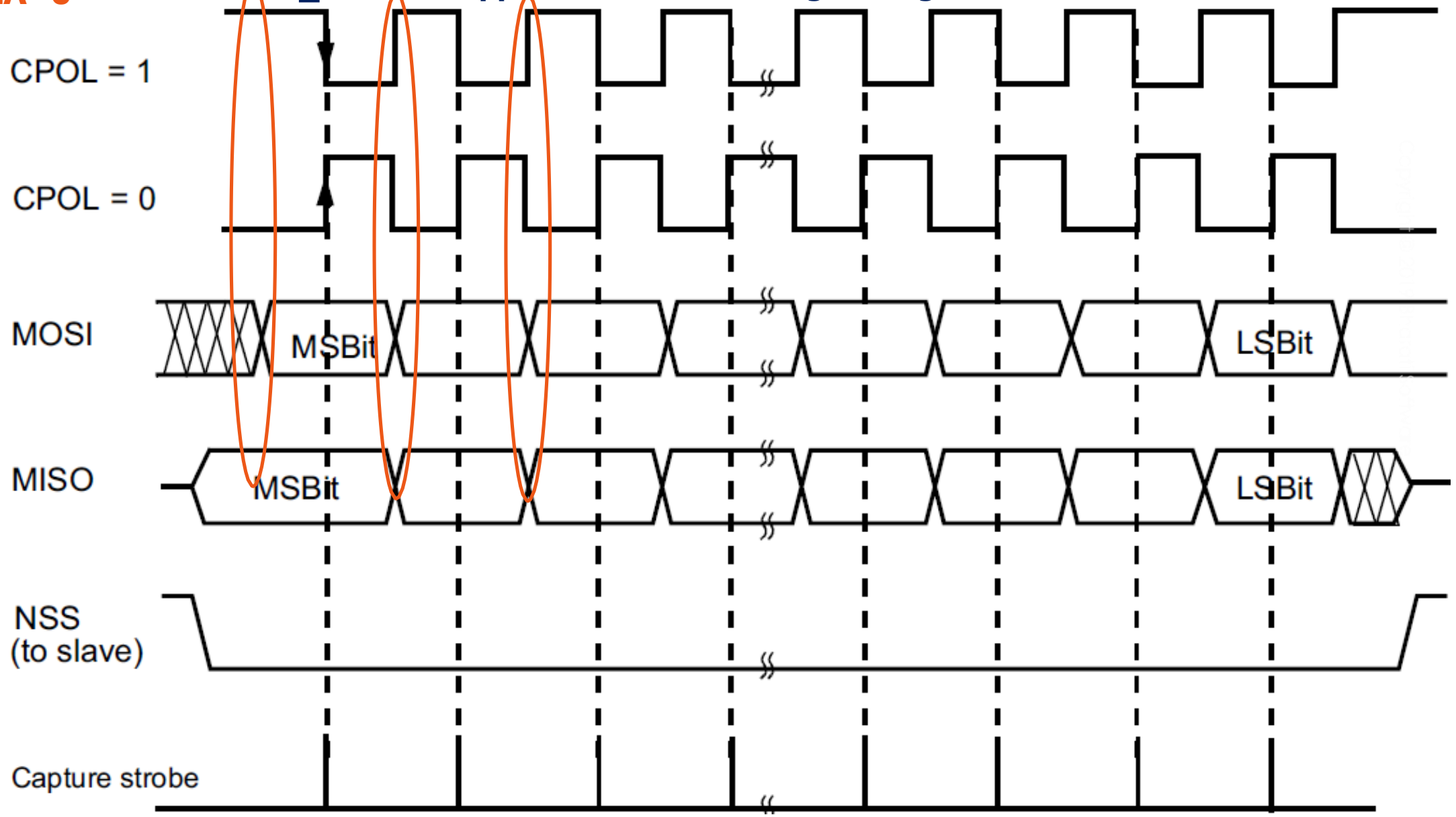
1

Data will appear on the lines during first edge of the SCLK



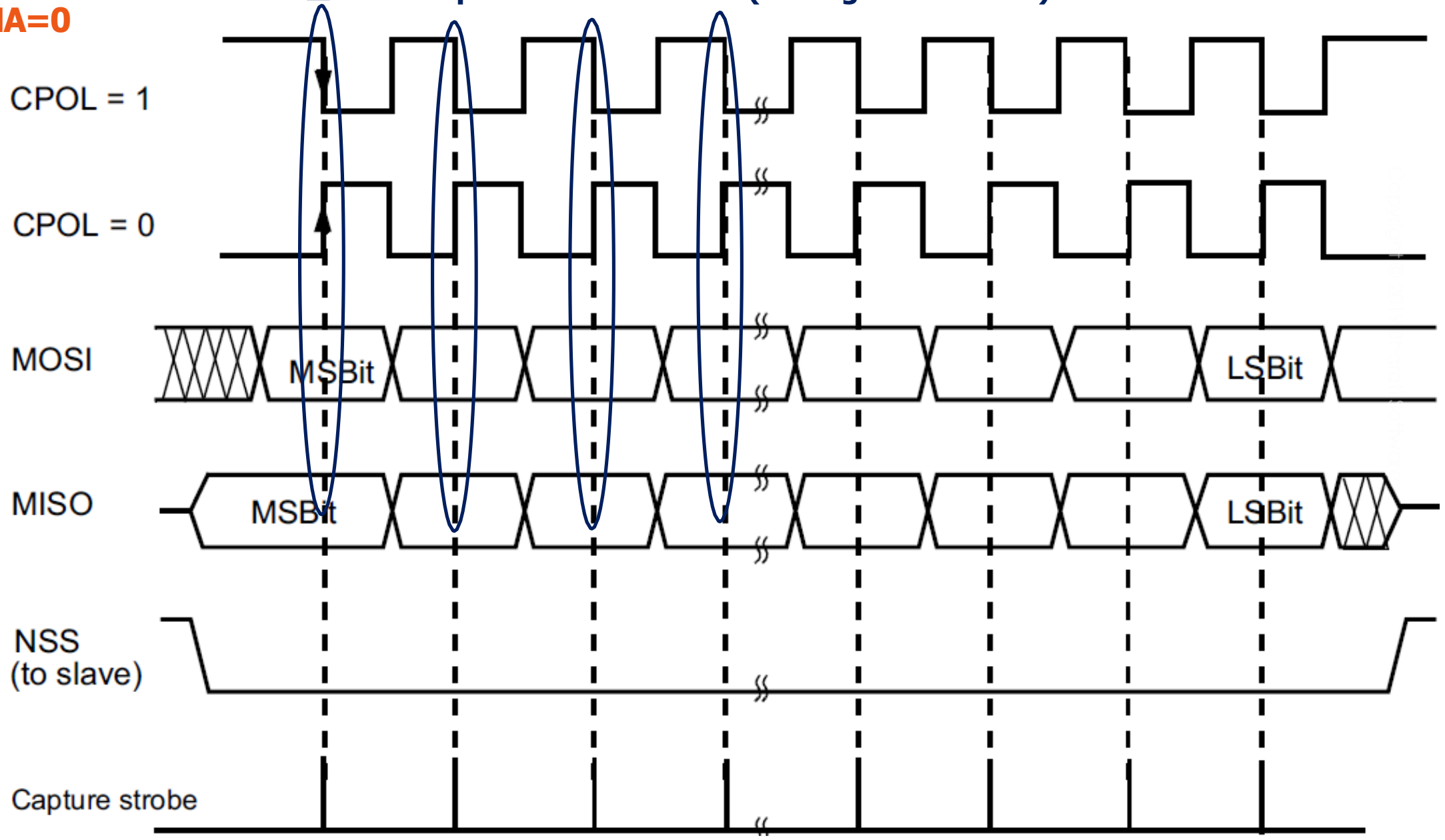
CPHA=0

1 **Data will appear on the lines during 2nd edge of the lock**



CPHA=0

2 Slave captures the data here (1st edge of the SCLK)



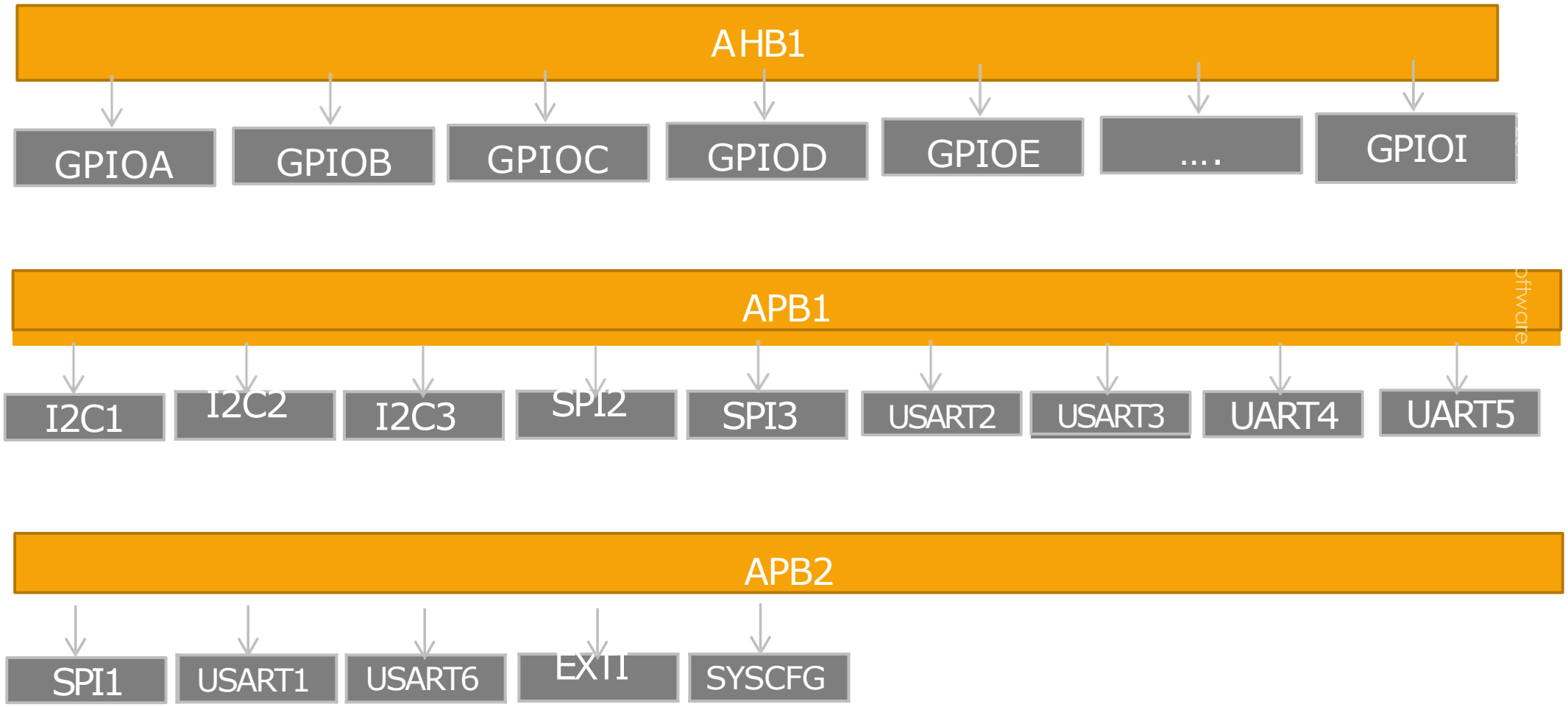
if **CPHASE=1**
Data will be sampled on the *trailing edge* of the clock.

If **CPHASE=0**
Data will be sampled on the *leading edge* of the clock.

Different SPI Modes

Mode	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

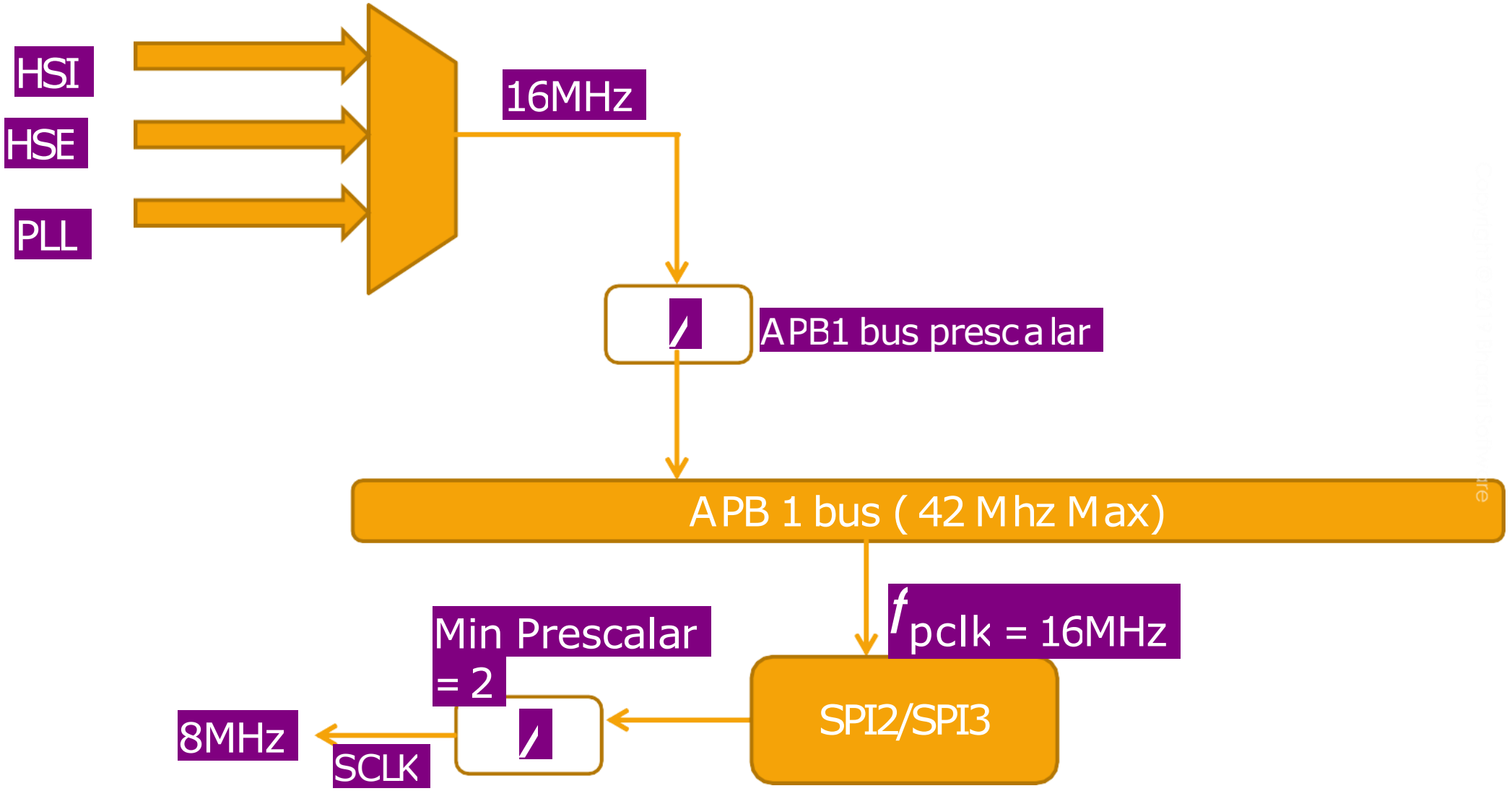
SPI peripherals of your MCU

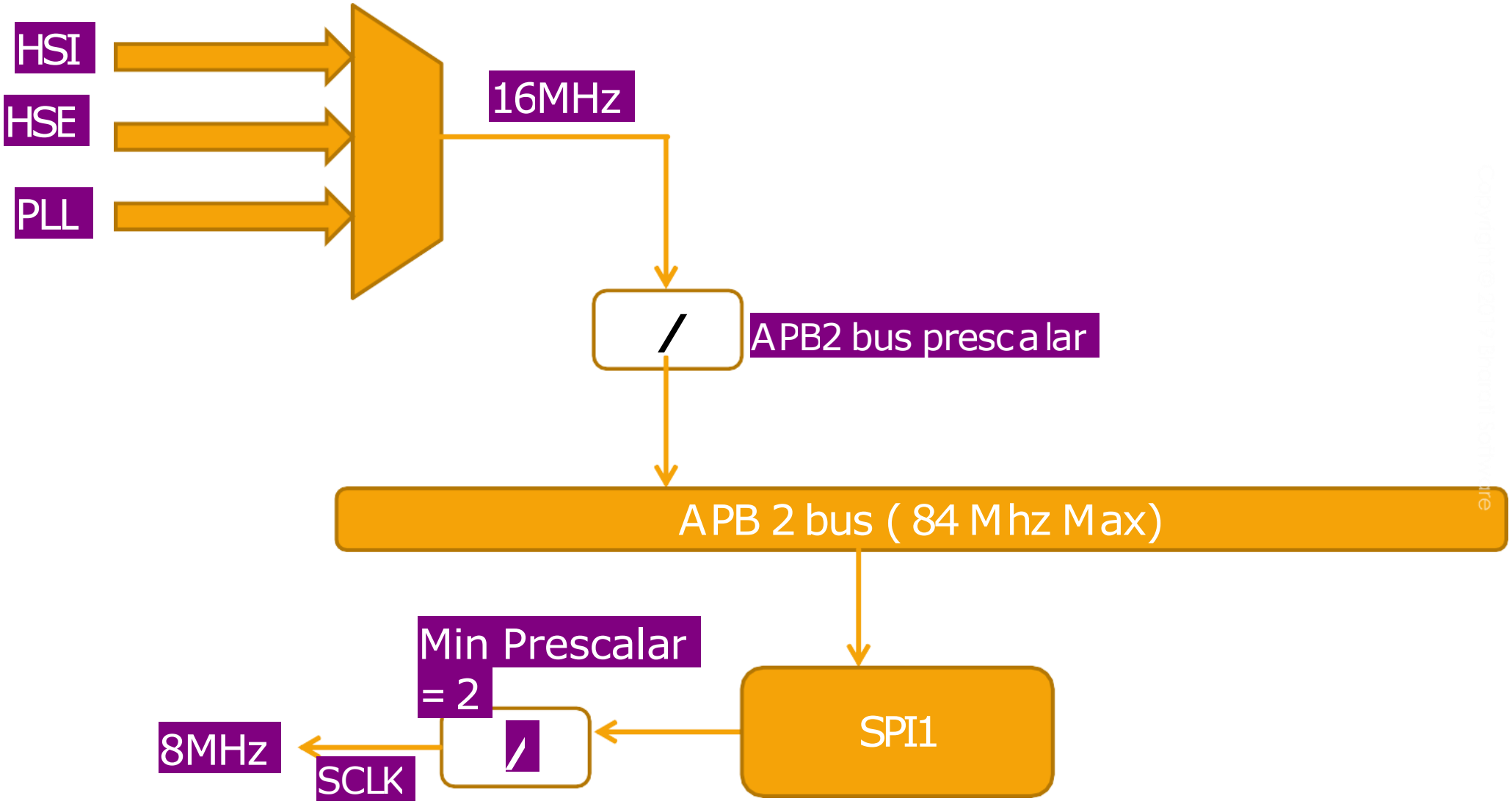


SPI serial clock (SCLK)

What is the maximum SCLK speed of SPIx peripheral which can be achieved on a given microcontorller ?

First you have to know the speed of the APBx bus on which the SPI peripheral is connected


HSI
HSE
PLL
16MHz
APB1 bus prescalar
APB 1 bus (42 Mhz Max)
fpclk = 16MHz
Min Prescalar = 2
8MHz
SCLK
SPI2/SPI3

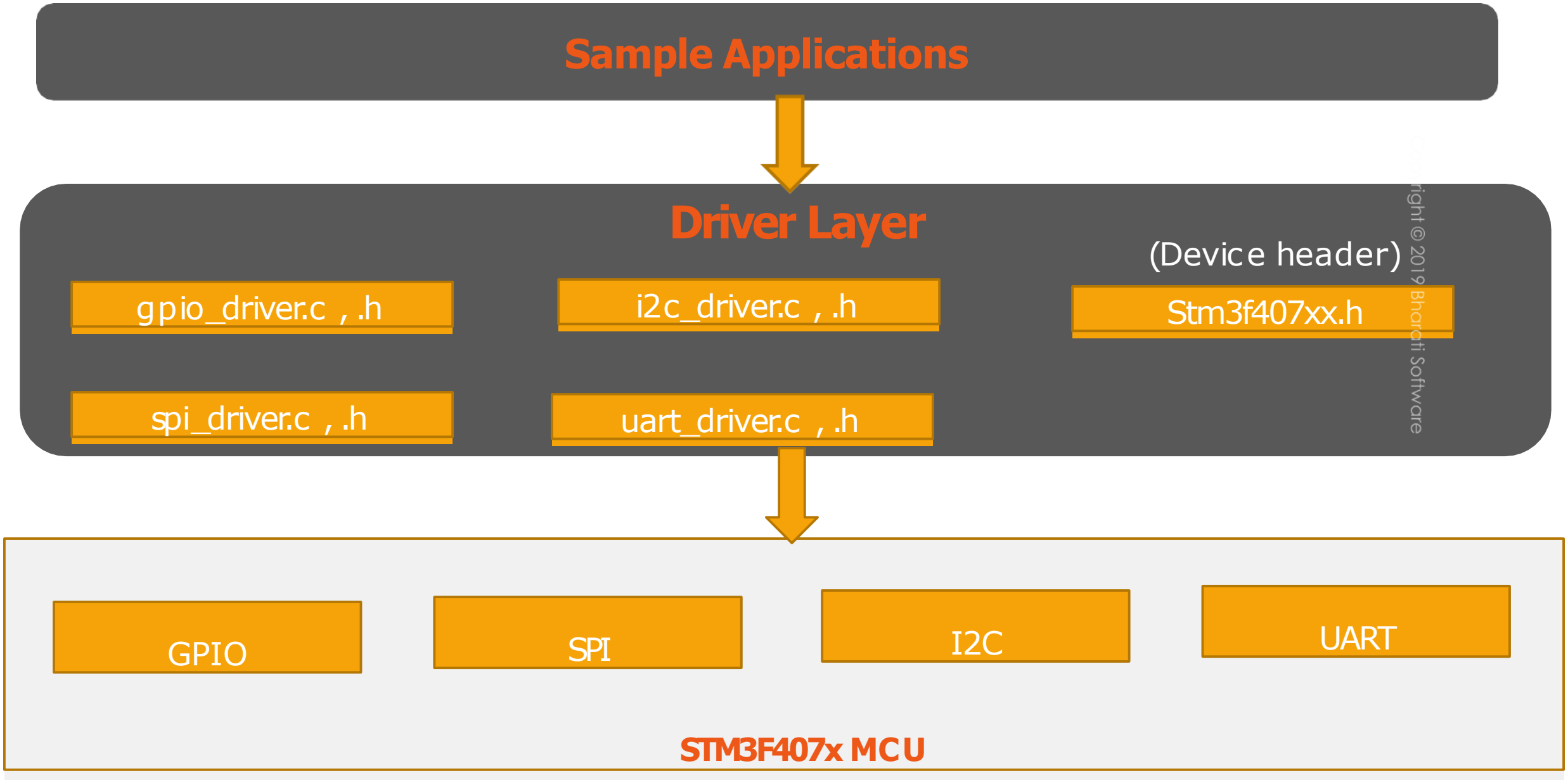

HSI
HSE
PLL
16MHz
/
APB2 bus prescalar
APB 2 bus (84 Mhz Max)
Min Prescalar
= 2
/
8MHz
SCLK
SPI1

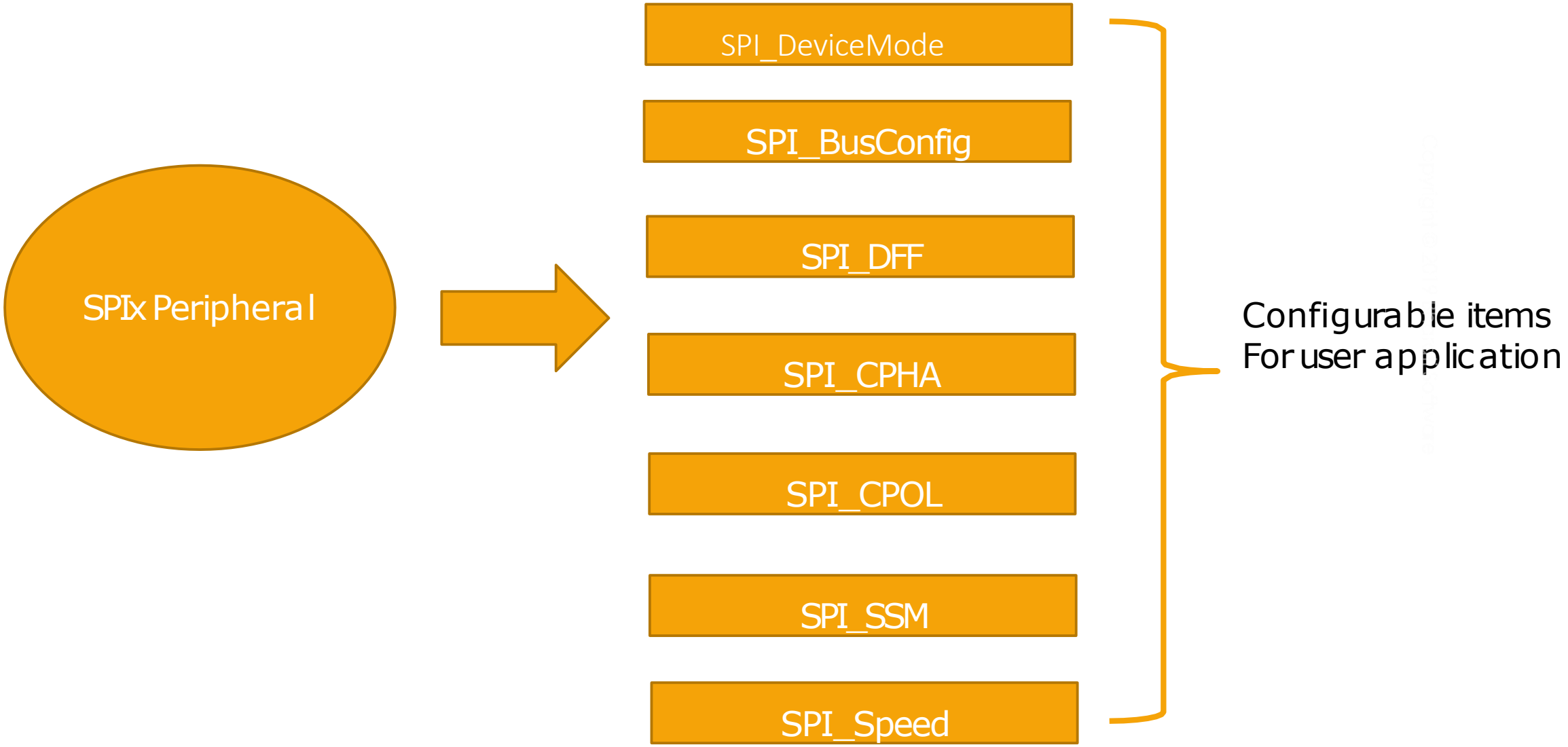
we use the internal RC oscillator of *16Mhz as our system clock* then SPI1/SPI2/SPI3 peripherals can able to produce the serial clock of maximum 8MHz.

Remember that in SPI communication, slave will not initiate data transfer unless master produces the clock.

SPI Driver Development

Driver API requirements and user configurable items




SPIx Peripheral
SPI_DeviceMode
SPI_BusConfig
SPI_DFF
SPI_CPHA
SPI_CPOL
SPI_SSM
SPI_Speed
Configurable items
For user application

SPI handle structure and configuration structure

SPI Configuration Structure

```c
/*
 *  Configuration structure for SPIx peripheral
 */
typedef struct
{
    uint8_t SPI_DeviceMode;
    uint8_t SPI_BusConfig;
    uint8_t SPI_SclkSpeed;
    uint8_t SPI_DFF;
    uint8_t SPI_CPOL;
    uint8_t SPI_CPHA;
    uint8_t SPI_SSM;
}SPI_Config_t;
```

SPI Handle Structure

```c
/*
 *Handle structure for SPIx peripheral
 */
typedef struct
{
    SPI_RegDef_t    *pSPIx;    /*!< This holds the base address of SPIx(x:0,1,2) peripheral >*/
    SPI_Config_t    SPIConfig;
}SPI_Handle_t;
```

Gyro Sensors