**SPI: Serial Peripheral Interface (**One unique benefit of SPI is the fact that data can be transferred without interruption.**)**

SPI, I2C, and UART are quite a bit slower than protocols like USB, ethernet, Bluetooth, and Wi-Fi, but they’re a lot simpler and use less hardware and system resources. SPI, I2C, and UART are ideal for communication between microcontrollers and between microcontrollers and sensors where large amounts of high-speed data don’t need to be transferred.

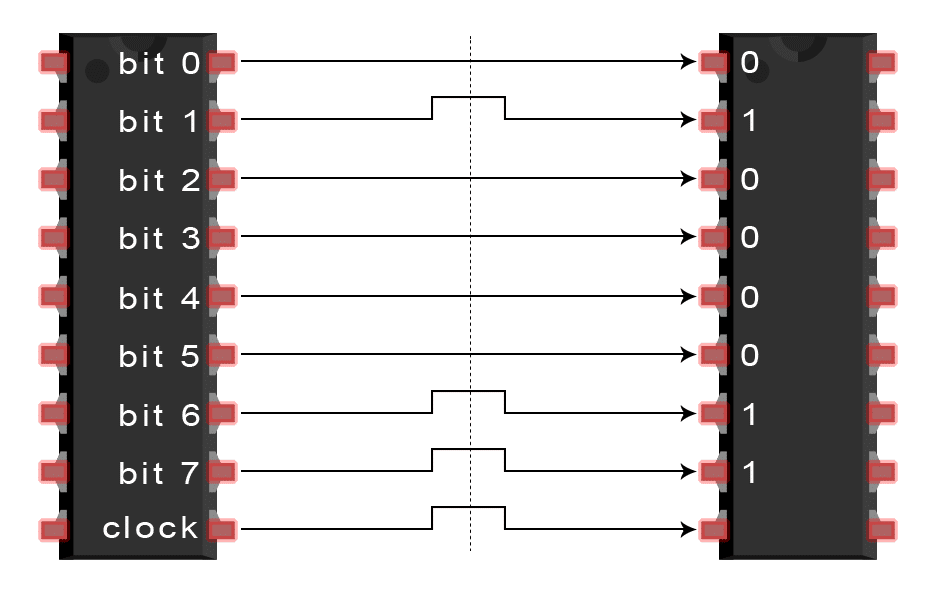


Fig: Parallel Com

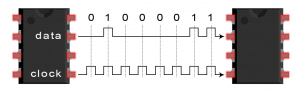


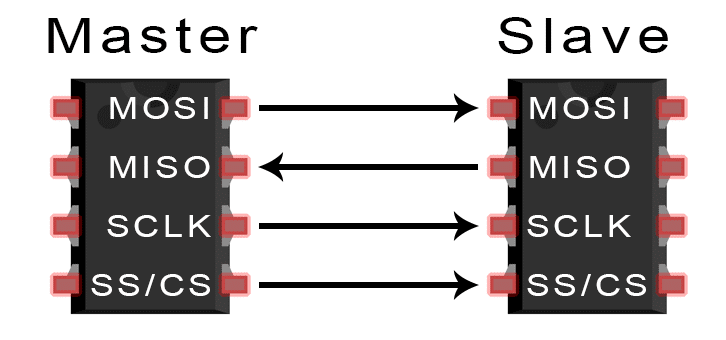
Fig: Serial Com

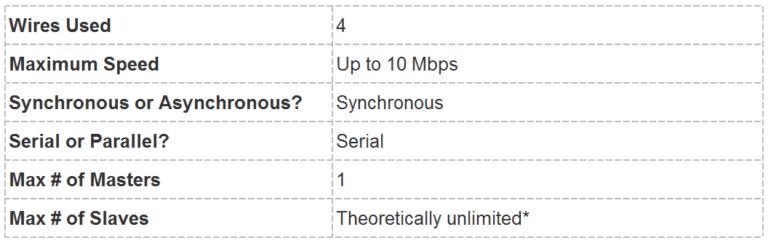
The bits (0 OR 1) of data can be transmitted either in parallel or serial form.

**In parallel communication**, the bits of data are sent all at the same time, each through a separate wire. The following diagram shows the parallel transmission of the letter “C” in binary (01100011)

**In serial communication**, the bits are sent one by one through a single wire. The following diagram shows the serial transmission of the letter “C” in binary (01000011):

**Devices communicating via SPI are in a master-slave relationship. The master is the controlling device (usually a microcontroller).** **One master can control more than one slave.**

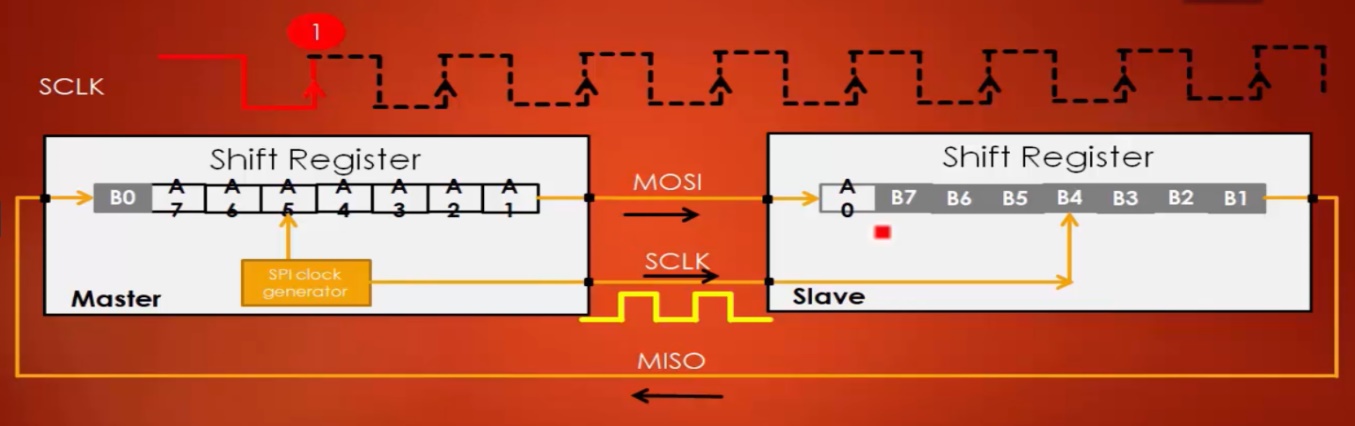




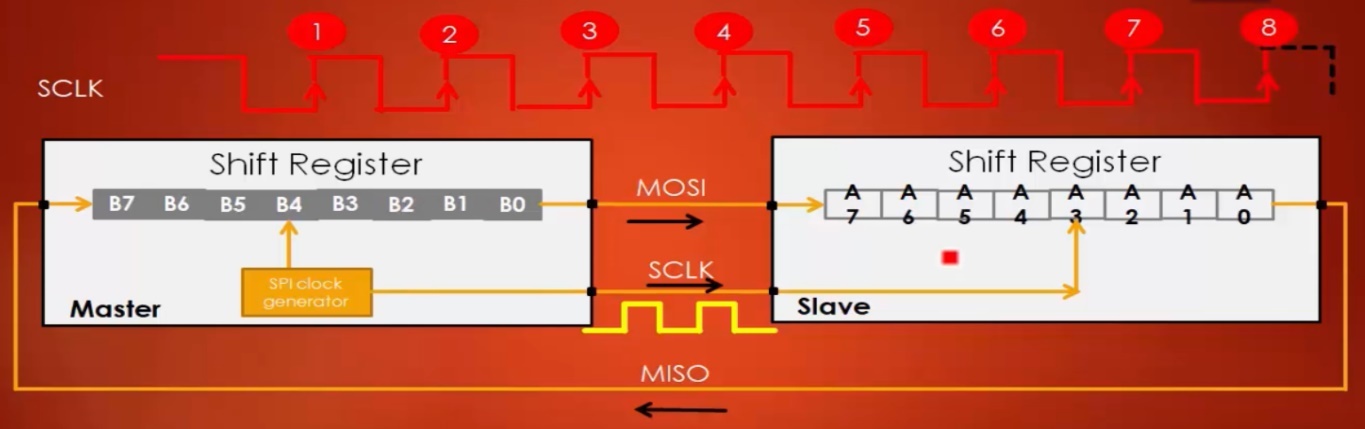
* **MOSI (Master Output/Slave Input) – Line for the master to send data to the slave.**
* **MISO (Master Input/Slave Output)** – Line for the slave to send data to the master.
* **SCLK (Clock)** – Line for the clock signal.
* **SS/CS (Slave Select/Chip Select)** – Line for the master to select which slave to send data to.

**SPI behind the scene data communication principle:**

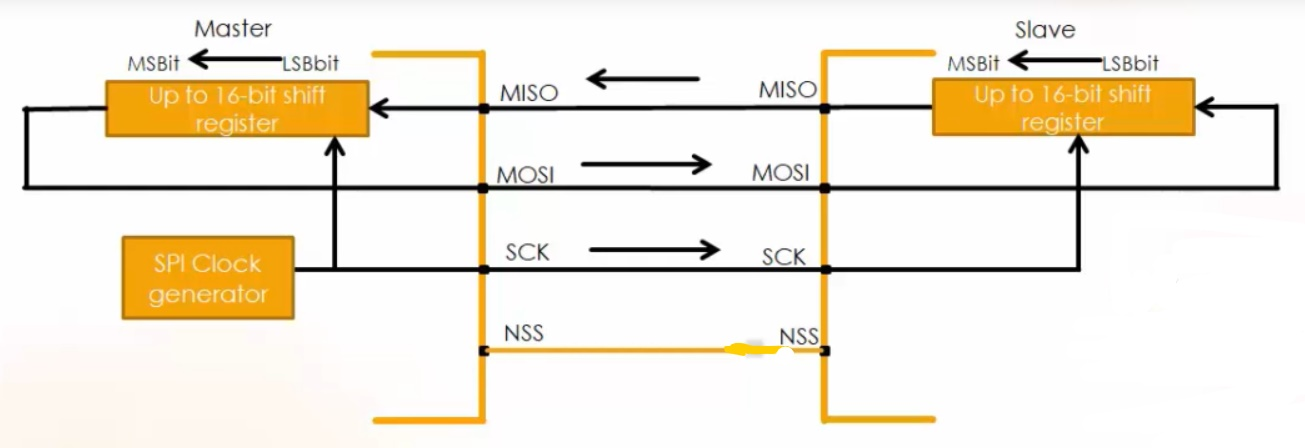
SPI hardware setup using two shift-registers.

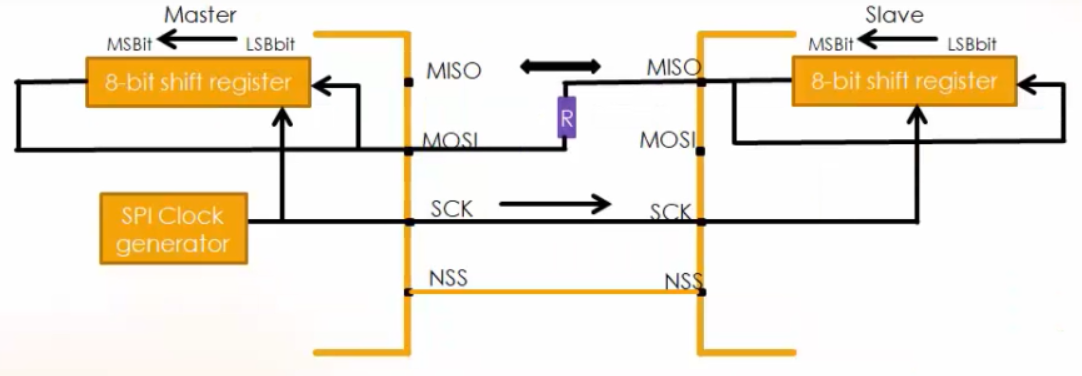
****

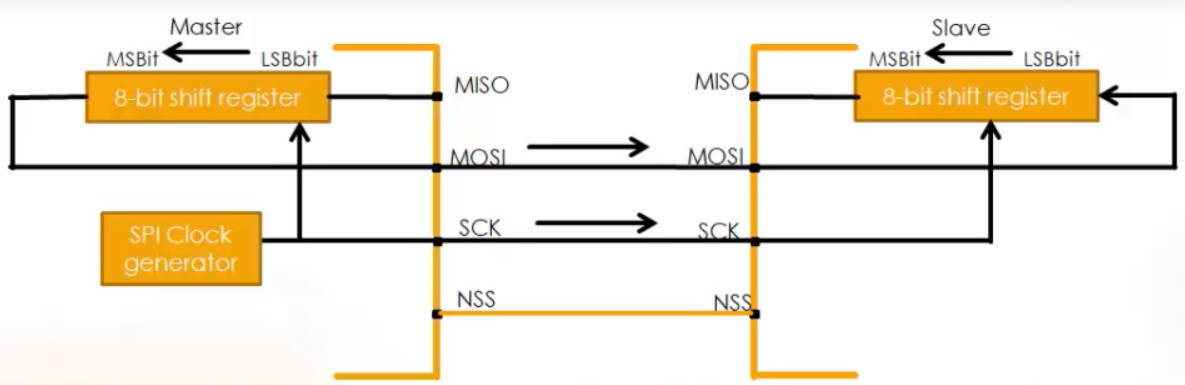
After the 8th clock cycle, the data which is there in the master shift register exchanged with a slave shift register.



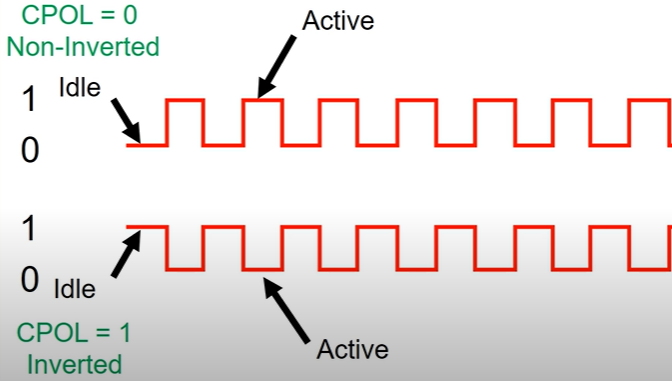
**SPI bus configuration discussion: Full duplex, Half-duplex, and Simplex:**







SPI CONFIGURATION:



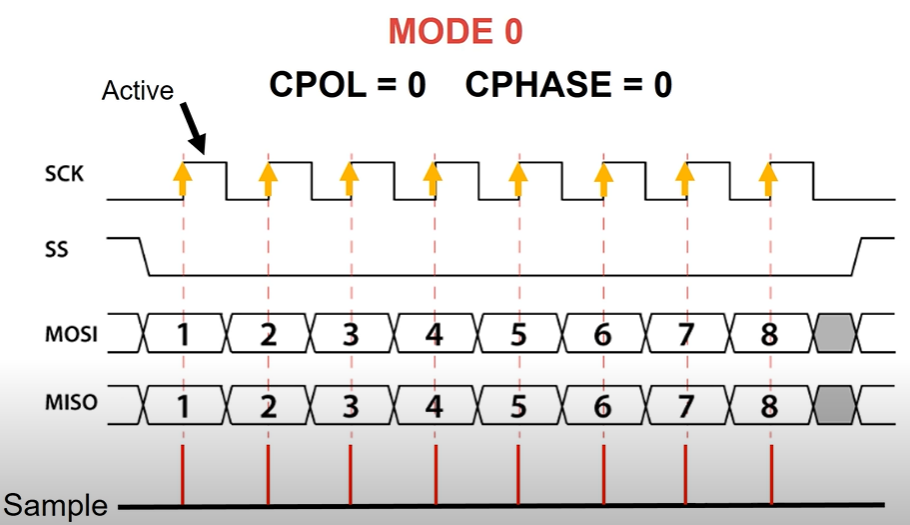
* CLOCK POLARITY(CPOL)
* CLOCK PHASE (CPHASE)

CLOCK POLARITY:

* INVERTED (when CPOL =1)
* NON-INVERTED (when CPOL=0)

CLOCK PHASE:

* When the data has to be toggled
* When the data has to be sampled

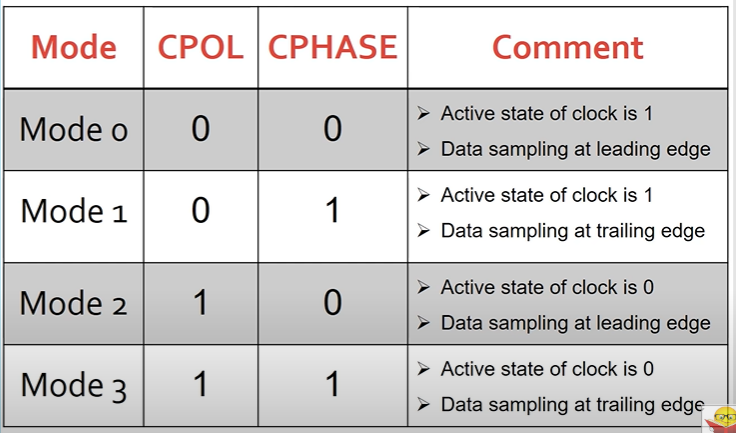


When CPHASE = 0,

* Data will be sampled at the leading edge of the clock.

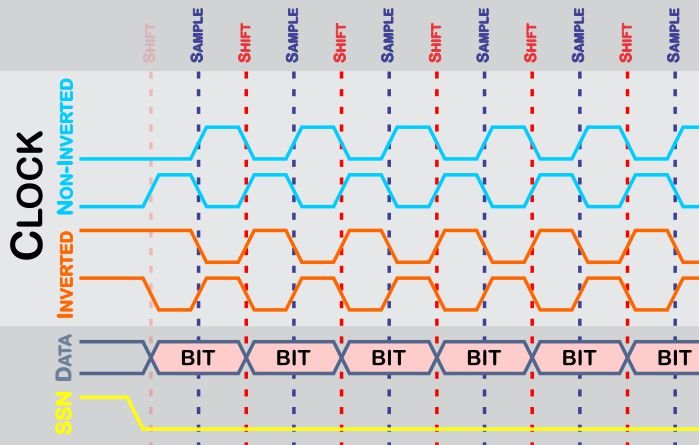
When CPHASE = 1,

* Data will be sampled at the trailing edge of the clock.

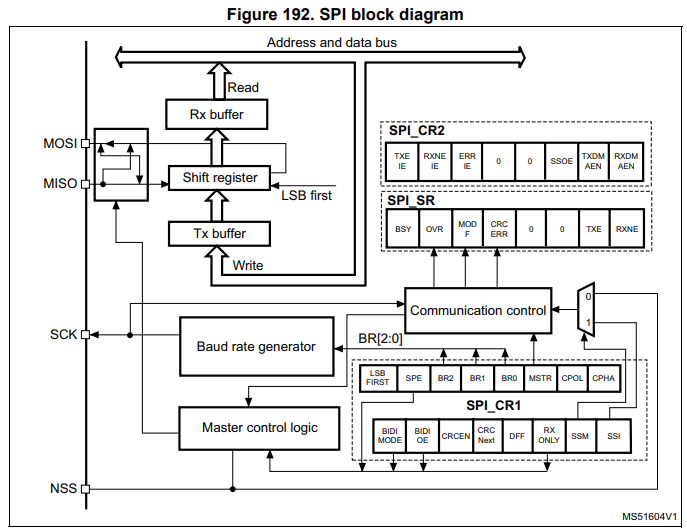


Clock transitions govern the shifting and sampling of data. SPI has [four modes (0,1,2,3)](https://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus#Clock_polarity_and_phase) that correspond to the four possible clocking configurations.

Bits that are sampled on the rising edge of the clock cycle are shifted out on the falling edge of the clock cycle, and vice versa.



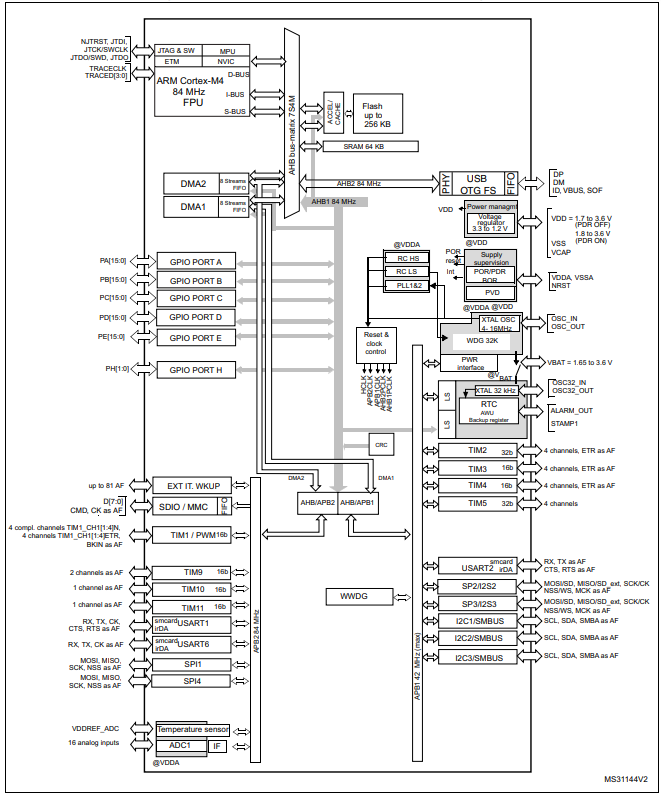
SPI FUNCTION BLOCK OF STM32F4:

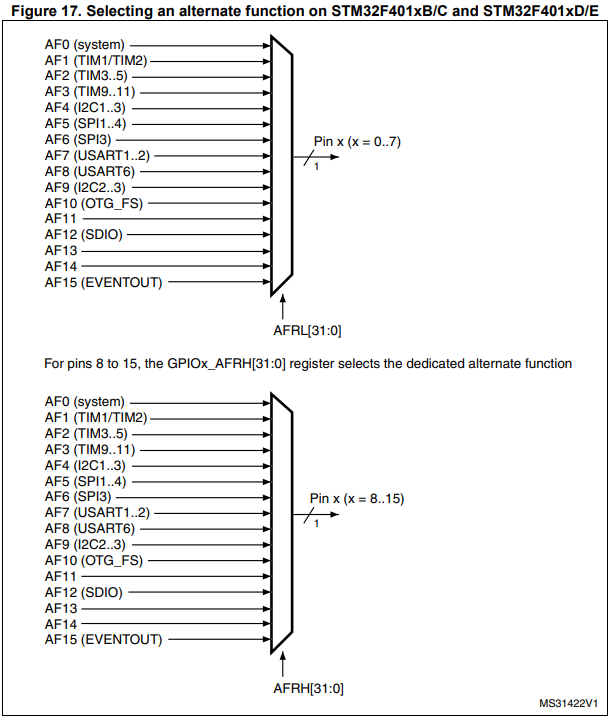


From SPI block diagram we see that it has two control register (SPI\_CR1, SPI\_CR2) and one status register (SPI\_SR).

<https://www.youtube.com/watch?v=3UWMAJzSWeE>

Figure 3. STM32F401xB/STM32F401xC block diagram

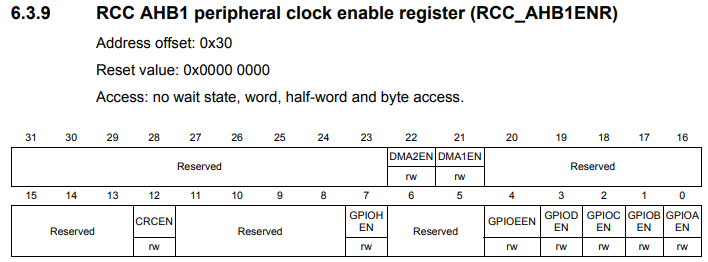




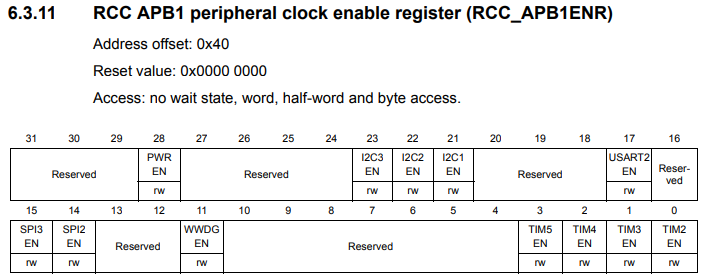
First of all, we need to see which buses connected with SPI.

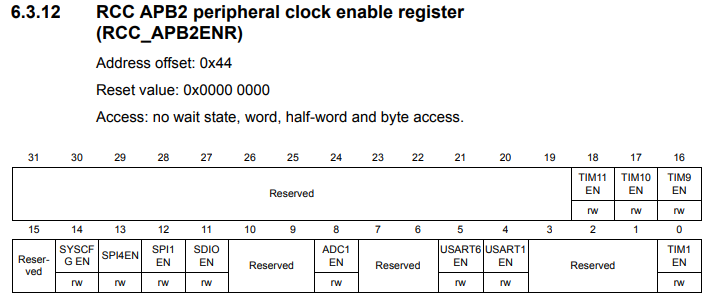
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| SPI | PORT | GPIO | MAIN BUS | SUB\_BUS | ALTERNET FUNC |
| SPI\_1 | PORT\_A | PA5, PA6, PC7 | AHB1 | APB2 | AF5(AFRL) |
| SPI\_2 | PORT\_C, B | PC2, PC3, PB10 | AHB1 | APB1 | AF5(AFRL, AFRH) |
| SPI\_3 | PORT\_C | PC12, PC11, PC10 | AHB1 | APB1 | AF6(AFRH) |
| SPI\_4 | PORT\_E | 2,5,6 | AHB1 | APB2 | AF5(AFRL) |
| SPI\_5 | PORT\_F | 7,8,9 | AHB1 | APB2 | AF5(AFRL, AFRH) |
| SPI\_6 | PORT\_G | PG12, PG13, PG14 | AHB1 | APB2 | AF5(AFRH) |

**STEP1:** from block diagram we saw that all GPIO port connected with AHB1 bus. For that reason, first of all need to enable clock of SPI GPIO PORT.

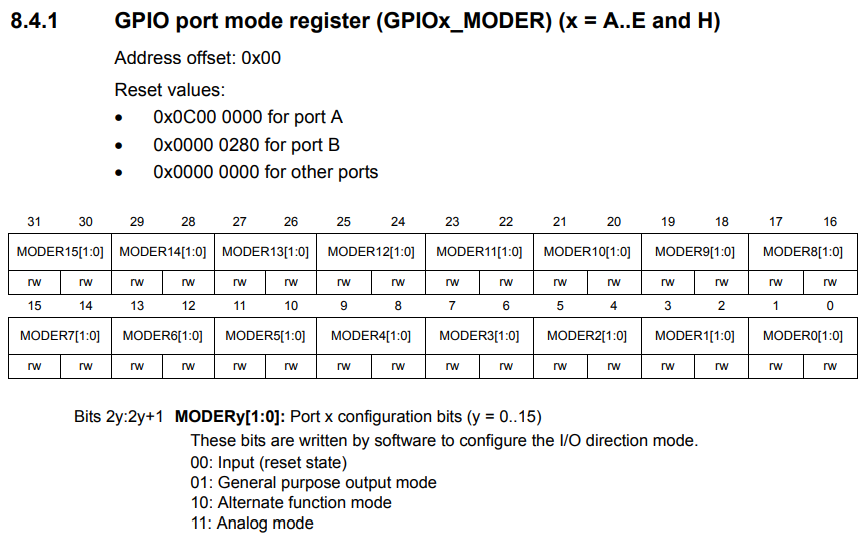


**STEP\_2:**  from block diagram we saw that SPI\_1 and SPI\_4 connected with APB2 bus, SPI\_2 and SPI\_3 connected with APB1 bus. that’s why now need to enable clock of SPI port.

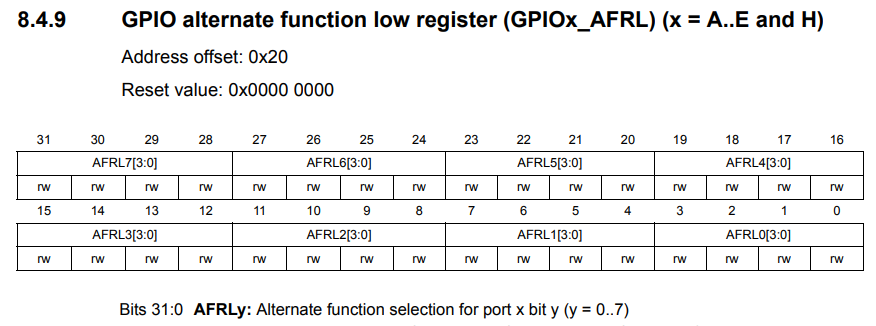


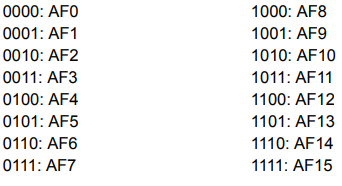


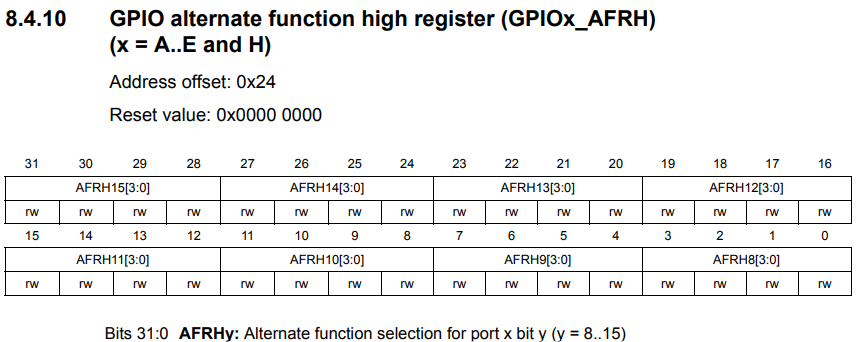
**STEP\_3: we need to setup GPIO MODE as an alternate function for SPI.**



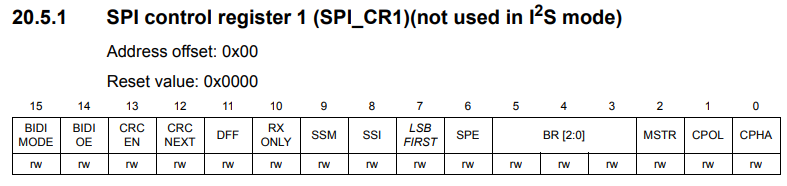
**STEP\_4:** Now need to configure GPIO ALTERNATE FUNCTION for SPI. From previous discussion we saw that AF5 and AF6 are responsible for SPI 1…4.







**STEP 5:** Now we configure SPI control register 1 (SPIx\_CR1).



**BIT: 0 (**CPHA=1) full meaning of CPHA is CLOCK PHASE

When CPHASE = 0,

* Data will be sampled at the leading edge of the clock.

When CPHASE = 1,

* **Data will be sampled at the trailing edge of the clock**.

**BIT:1** (CPOL=1) full meaning of CPOL is CLOCK POLARITY

CLOCK POLARITY:

* **INVERTED (when CPOL =1)**
* NON-INVERTED (when CPOL=0)

**BIT:2**(MSTR=1) master and slave selection

* 0: Slave configuration
* 1: Master configuration

**Note:** This bit should not be changed when communication is ongoing. It is not used in I2S mode.

**BIT:3 to 5(BR=**011**)** Baud rate control

* 000: fPCLK/2
* 001: fPCLK/4
* 010: fPCLK/8
* **011: fPCLK/16**
* 100: fPCLK/32
* 101: fPCLK/64
* 110: fPCLK/128
* 111: fPCLK/256

**BIT: 6** (SPE ) SPI enable / disable

* 0 : Peripheral disabled
* 1 : Peripheral enabled

**BIT: 7**(LSBFIRST=0) Frame formats

* **0: MSB transmitted first**
* 1: LSB transmitted first

**BIT: 8**(SS1=1) Internal slave select

This bit has an effect only when the SSM bit is set. The value of this bit is forced onto the NSS pin and the IO value of the NSS pin is ignored

**BIT:9** (SSM=1) SOFTWER SLAVE MANAGEMENT

When the SSM bit is set, the NSS pin input is replaced with the value from the SSI bit.

* 0: Software slave management disabled
* **1: Software slave management enabled**

**BIT:10**(RXONLY=0) Transmit OR receive MODE SELECTION

* **0: Full duplex (Transmit and receive)**
* 1: Output disabled (Receive-only mode)

**BIT:11** (DFF=0) Data frame format

* 0: 8-bit data frame format is selected for transmission/reception
* 1: 16-bit data frame format is selected for transmission/reception