ATmega328P SPI

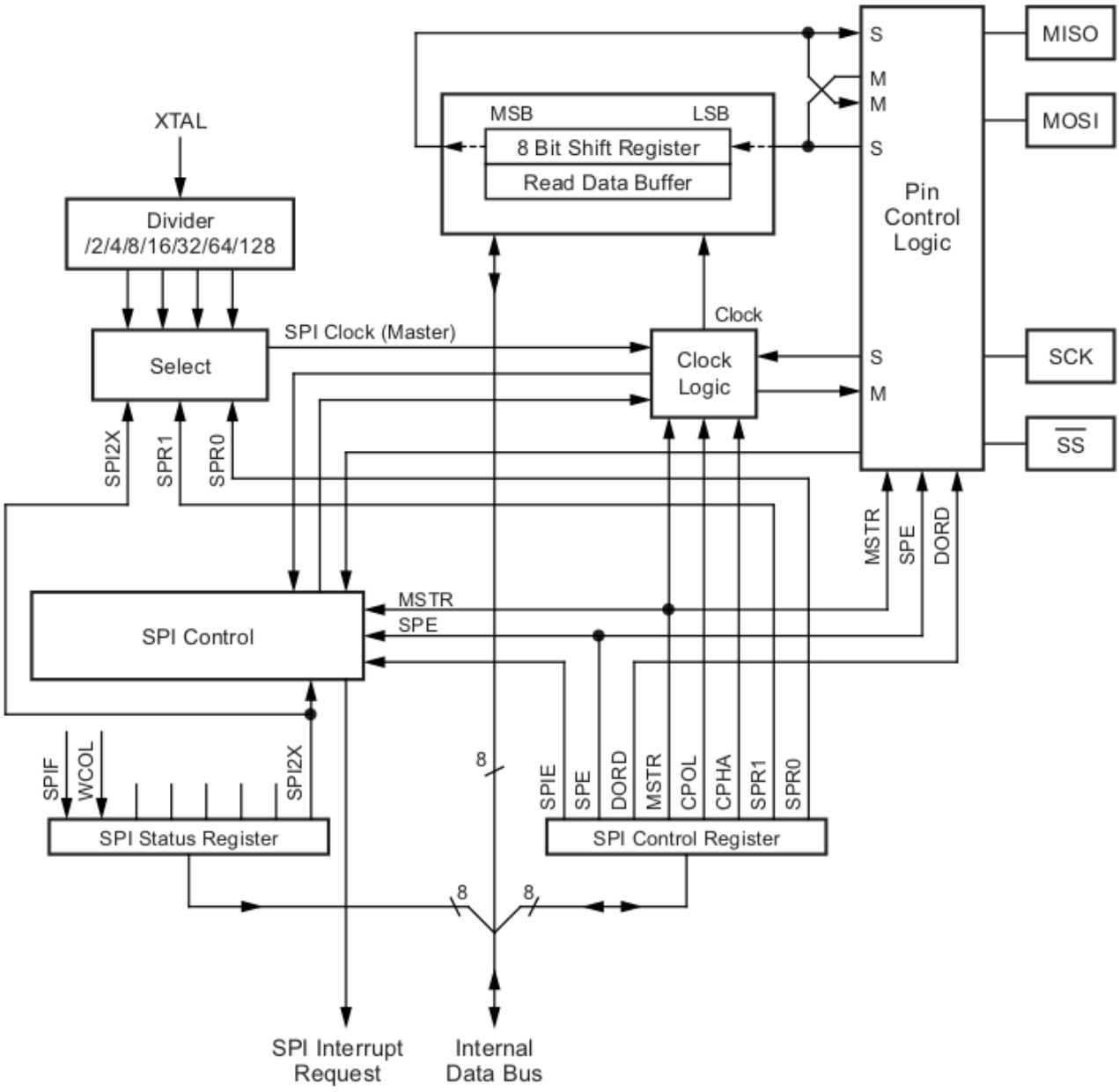
Narendiran S

July 28, 2020

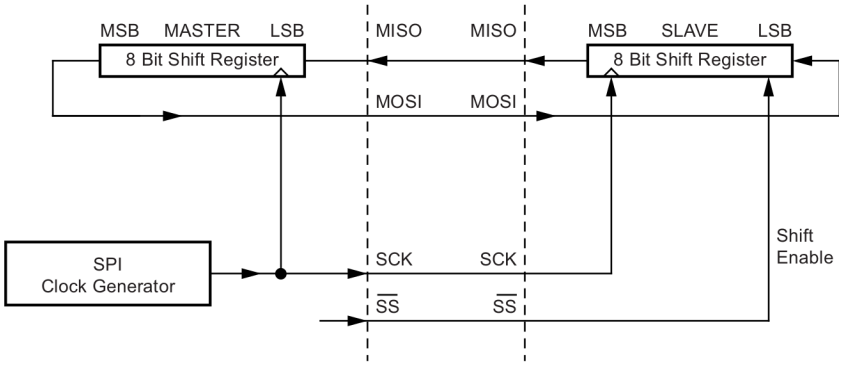
1 Features

- Full-duplex, three-wire synchronous data transfer
- LSB first or MSB first
- Seven Programmable bit rates
- high-speed synchronous data transfe

2 Block Diagram



3 SPI Master-Slave Interconnection



3.1 SPI Pins

The SPI is connected to external devices trhough four pins namely,

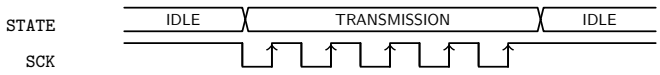
- **MISO** - Master IN / Slave OUT data - transmit data in slave mode and receive data in master mode.
- **MOSI** - Master OUT / Slave IN data - transmit data in master mode and receive data in slave mode.
- **SCK** - Serial Clock - outputs clock on SPI master mode and inputs clock on SPI slave mode.
- **NSS** - Slave Select - select the chip or the slave.

3.2 Basic Operation

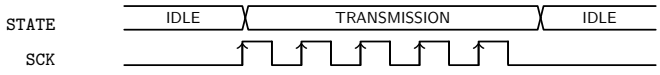
- Two shift Registers and a master clock generator.
- Initialization is done by pulling low the $\overline{SS}$ pin.
- Master generates the required clock pulses on *SCK* to interchange data.
- Using *MOSI* – Master Out Slave In – data is shifted from master to slave.
- Using *MISO* – Master In Slave Out – data is shifted from slave to master.
- After each data packet, the master will synchronize the Slave by pulling high the Slave select $\overline{SS}$ pin.

3.3 Clock Phase and Clock polarity

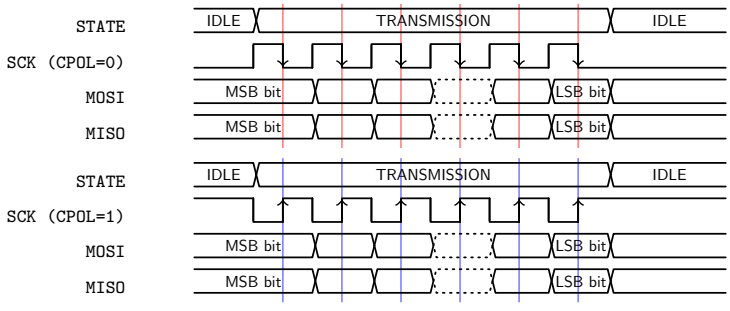
- ***CPOL*** bit controls the steady state value of *SCK* line when idle(no data is transferred).
  - ***CPOL*** = 1 : *SCK* line is high-level idle state



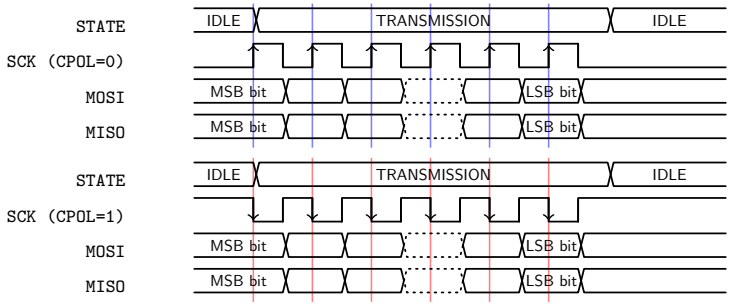
  - ***CPOL*** = 0 : *SCK* line is low-level idle state



- ***CPHA*** bit controls the capture of datas.
  - ***CPHA*** = 1 : MSB bit is captured on the **second edge** of *SCK* pin (falling edge if the ***CPOL*** bit is 0, rising edge if the ***CPOL*** bit is 1).



– ***CPHA*** = 0 : MSB bit is captured on the **first edge** of *SCK* pin (falling edge if the ***CPOL*** bit is 1, rising edge if the ***CPOL*** bit is 0).



3.4 Data Frame Format

The data can be shifted out either MSB first or LSB first.

4 Register Description

SPCR – SPI Control Register

7	6	5	4	3	2	1	0
SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0

- ***SPIE - SPI Interrupt Enable*** - Enable the SPI interrupt to be executed if ***SPIF*** bit is set in **SPSR** Register.
- ***SPE - SPI Enable*** - Enable the SPI.
- ***DORD - Data Order*** - Defines the data order being sent[1 == LSB first; 0 == MSB first]
- ***MSTR - Master/Slave Select*** - Select between Master Mode and Slave Mode[1 == Master Mode; 0 == Slave Mode]

SI2X, SPR1, SSPR0	SCK Frequency
000	$\frac{fosc}{4}$
001	$\frac{fosc}{16}$
010	$\frac{fosc}{64}$
011	$\frac{fosc}{128}$
100	$\frac{fosc}{2}$
101	$\frac{fosc}{8}$
110	$\frac{fosc}{32}$
111	$\frac{fosc}{64}$

SPSR – SPI Status Register

7	6	5	4	3	2	1	0
SPIF	WCOL	-	-	-	-	-	SPI2X

- ***SPIF - SPI Interrupt Flag*** - Denotes the end of serial transfter. A intterup its generated if ***SPIE*** bit in **SPCR** register is set.

SPDR – SPI Data Register

7	6	5	4	3	2	1	0
D7	D6	D5	D4	D3	D2	D1	D0

5 Configuring the SPI

- First, the pins *MOSI*, *MISO*, *SCK* and $\overline{SS}$ is configured to required Direction.
- Next, the $\overline{SS}$ pin is made low or high depending on the device Specs.
- The data order is selected by ***DORD*** bit in **SPCR** register.
- The Master/Slave Mode is selected by ***MSTR*** bit in **SPCR** register.
- The timing is choosen by Configuring ***CPOL*** and ***CPHL*** bit in **SPCR** register depending on the Device Specs.
- The Clock Frequency for SPI communication is choosen by Configuring the ***SPI2X***, ***SPR1*** and ***SPR*** bits of **SPCR** and **SPSR** registers.
- Interrupt is enabled by setting the ***SPIE*** bit in **SPCR** register.
- FInally, SPI is enabled by setting the **SPE** bit in **SPCR** register.
- Also, the interrupt service routing is written, when the transmission/reception completes.
- The data can be transmitted/received by writing/reading from **SPIDR** register.
- An example code is seen below,

```
// making SCK, MOSI, SS' as outptut
DDRB |= (1<<DDB2) | (1<<DDB3) | (1<<DDB5);
// making MISO as input
DDRB &= ~(1<<DDB4);

// making SCK, MOSI,  as low
PORTB &=  ~(1<<PORTB3) & ~(1<<PORTB5);
// making SS' as high
PORTB |= (1<<PORTB2);

// Select MSB first or LSB first by DORD
SPCR &= ~(1<<DORD);

// Select this as Master
SPCR |= (1<<MSTR);

// Let the clock polarity be SCK is low when idle
SPCR &= ~(1<<CPOL);

// Sampled at Rising or Falling Edge
// we choose rising edge
SPCR &= ~(1<<CPHA);

// Selecting a SCK frequnecy
// we select Fosc/4 by 000
SPSR &= ~(1<<SPI2X);
SPCR &= ~(1<<SPR1);
SPCR &= ~(1<<SPR0);
// dISBALE SPIE bit for interrupt on Serial Transfer Completion
SPCR &= ~(1<<SPIE);
// Enabling SPI
SPCR |= (1<<SPE);
```

```
uint8_t SPITransferReceive(uint8_t data_)
{
    SPDR = data_;
    // wait till serial transmission is complete by checking the SPI Interrupt Flag
    while((SPSR & (1<<SPIF)) == 0 ) {};
    // return the recieved data - can use it or ignore it
    return SPDR;
}
```