INTERFACING

SPI

AMIT

Serial Peripheral Interface:

Specs of SPI:

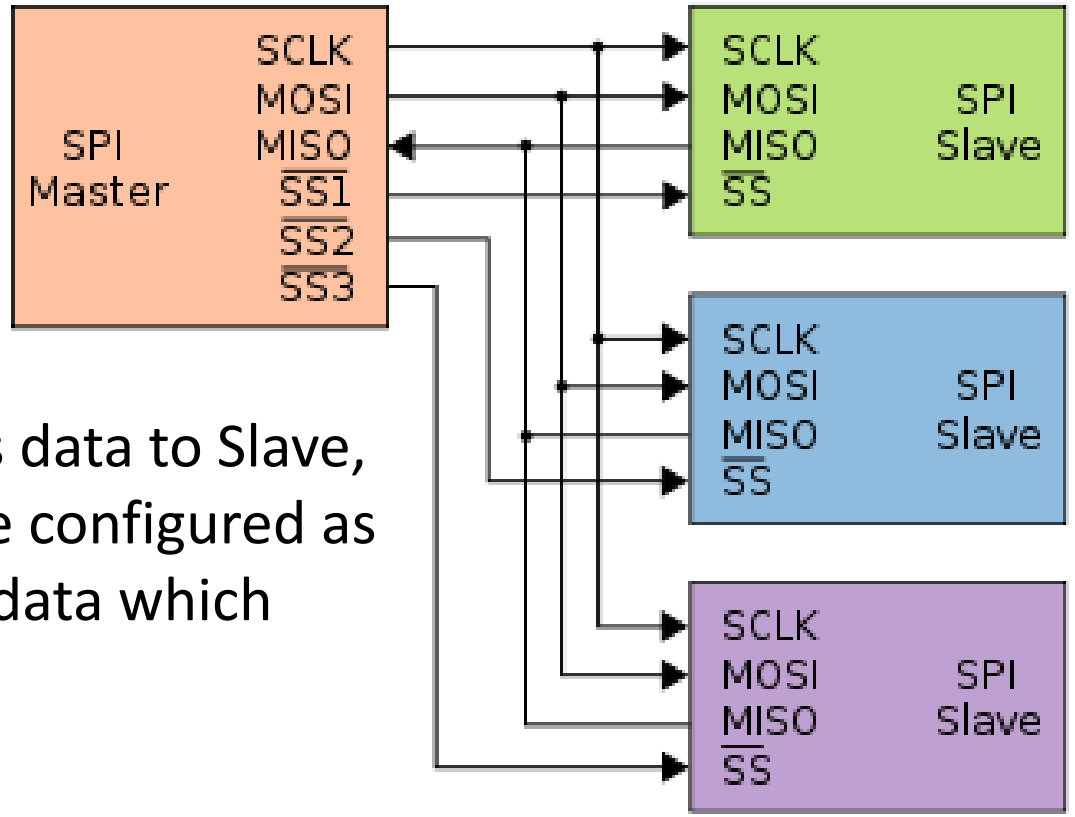
- The Serial Peripheral Interface (SPI) allows high-speed synchronous data transfer between the ATmega32 and peripheral devices or between several AVR devices. The ATmega32 SPI includes the following features:
  - Wired & Serial.
  - Single Master Multi Slave “**SMMS**”.
  - Synchronies & Full Duplex.
  - Throughput = 100 %.
  - TTL.

AMIT

Serial Peripheral Interface:

Hardware Connection of SPI:

- There are four pins for SPI connection, every pin has a specific function, like:



- MOSI:

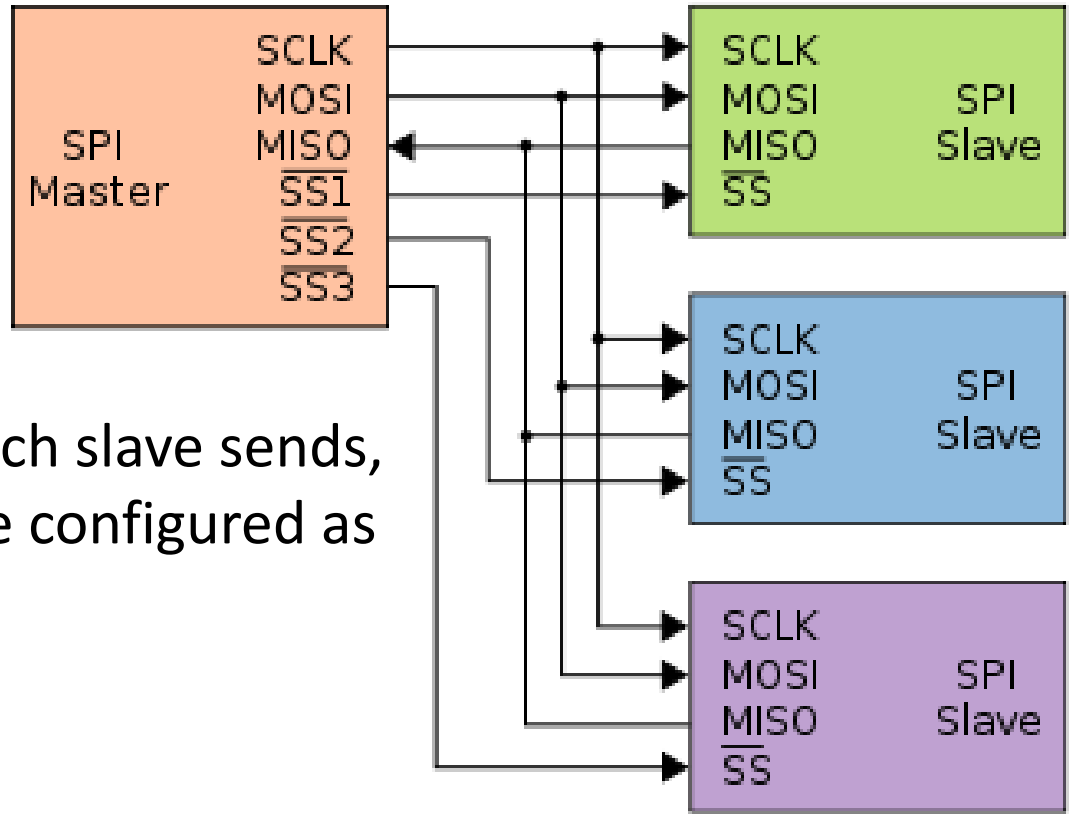
The pin is configured as output, if the mode of Node is a Master, the master uses it to write and send its data to Slave, if the mode of Node is Slave, it must be configured as input and the slave uses it to read the data which master sends.

AMIT

Serial Peripheral Interface:

Hardware Connection of SPI:

- There are four pins for SPI connection, every pin has a specific function, like:



- MISO:

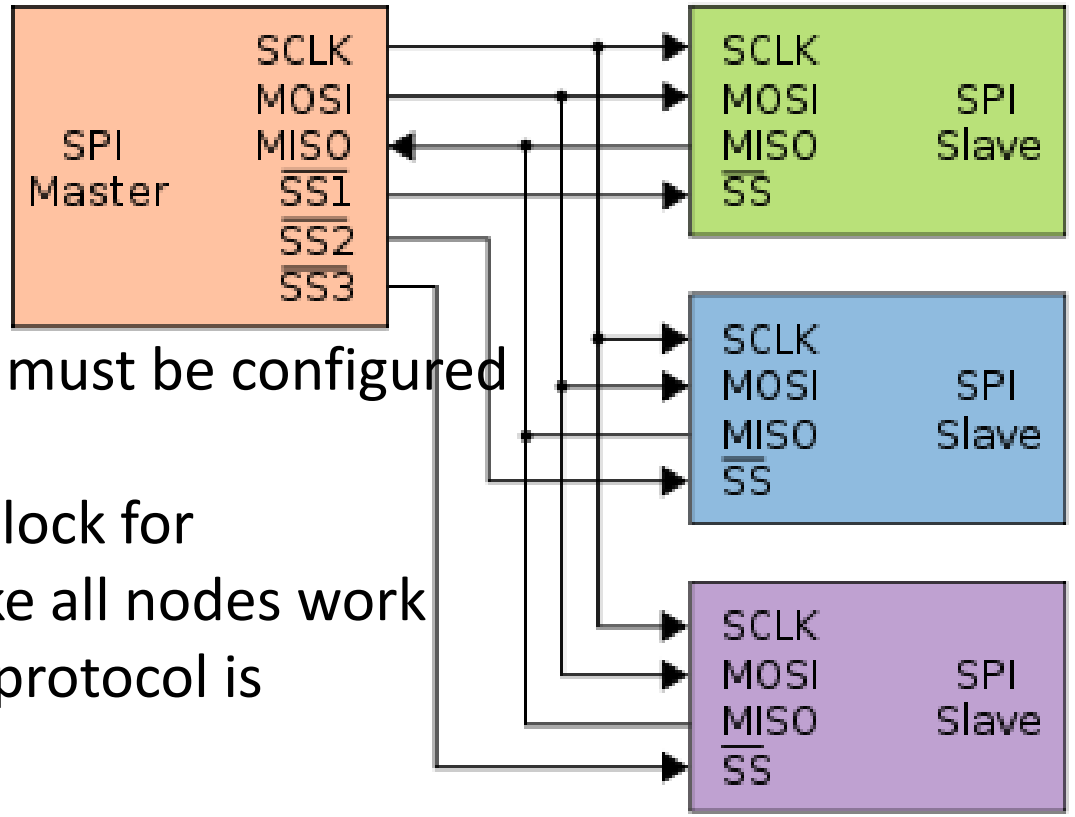
The pin is configured as input, if the mode of Node is a Master, the master uses it to read the data which slave sends, if the mode of Node is Slave, it must be configured as output and the slave uses to write and send its data to Master.

AMIT

Serial Peripheral Interface:

Hardware Connection of SPI:

- There are four pins for SPI connection, every pin has a specific function, like:



- SCLK:

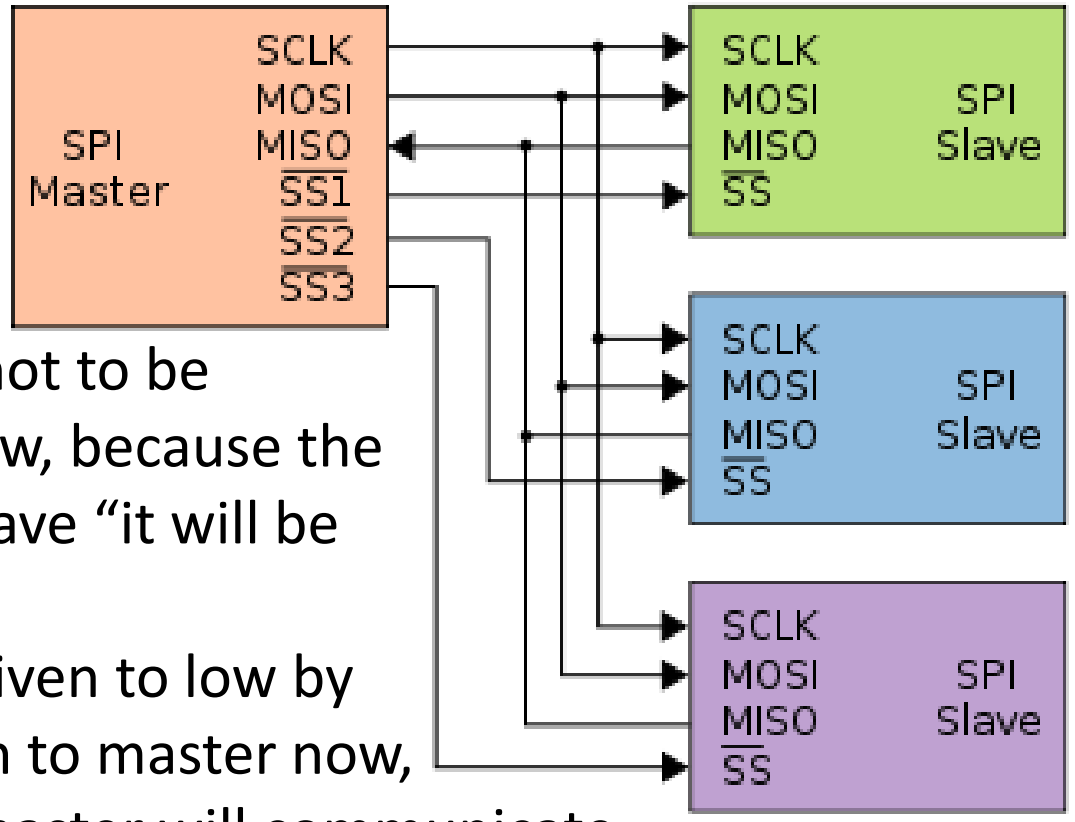
The pin is configured as output, if the mode of Node is a Master, and it must be configured as input, if the mode of Node is Slave.
the master slave uses it to provide its clock for all nodes exist into the network to make all nodes work with the same speed of master, so SPI protocol is a synchronies protocol.

AMIT

Serial Peripheral Interface:

Hardware Connection of SPI:

- There are four pins for SPI connection, every pin has a specific function, like:



- SS:

This pin is useless if the node is configured as a master, and take care not to be configured as an input and driven to low, because the low signal will convert the master to slave “it will be declared later”.

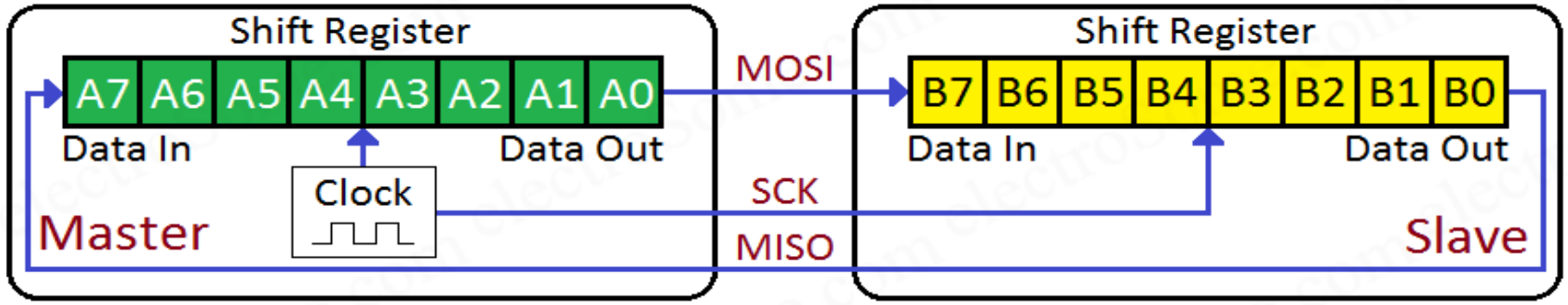
In slave mode, this pin must be driven to low by master to indicate the salve must listen to master now, so it is used to select which slave the master will communicate.

AMIT

Serial Peripheral Interface:

Data Frame of SPI:

- The transferring into SPI depends on exchanging data between Master and slave, it means when the master generates its clock, the master and slave at the same time will write their data, master will its bit on MOSI, and the slave will its bit on MISO and so on until the data of slave becomes into the master and the data of master becomes into the slave, as the following figure, and if it does not work, use the link below:



https://electrosome.com/wp-content/uploads/2017/04/SPI-Working-Data-Transfer.gif

Serial Peripheral Interface:

Data Frame and Throughput of SPI:

- As we mentioned, the transferring data depends on exchanging data without any extra bits like checking mechanism or start or stop bits, so the allover frame of SPI protocol is pure data, so the throughput of it is 100%, if the network of SPI connection is normal not Daisy chain or Cascading connection “it will be declared later”.

AMIT

Serial Peripheral Interface:

Register Description of SPI:

- SPI Control Register "**SPCR**":
  - **Bit 7** – SPIE: SPI Interrupt Enable:
    This bit causes the SPI interrupt to be executed if SPIF bit in the SPSR Register is set and the if the global interrupt enable bit in SREG is set.

SPI Control Register – SPCR

Bit	7	6	5	4	3	2	1	0
	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

  - **Bit 6** – SPE: SPI Enable:
    When the SPE bit is written to one, the SPI is enabled. This bit must be set to enable any SPI operations.

AMIT

Serial Peripheral Interface:

Register Description of SPI:

- SPI Control Register "**SPCR**":
  - **Bit 5** – DORD: Data Order:
    When the DORD bit is written to one, the LSB of the data word is transmitted first.
    When the DORD bit is written to zero, the MSB of the data word is transmitted first.
  - **Bit 4** – MSTR: Master/Slave Select:
    This bit selects Master SPI mode when written to one, and Slave SPI mode when written logic zero.
    If SS is configured as an input and is driven low while MSTR is set, MSTR will be cleared, and SPIF in SPSR will become set.
    The user will then have to set MSTR to re-enable SPI Master mode.

AMIT

Serial Peripheral Interface:

Register Description of SPI:

- SPI Control Register "**SPCR**":
  - **Bit 3** – CPOL: Clock Polarity:
    When this bit is written to one, SCK is high when idle. When CPOL is written to zero, SCK is low when idle.

  - **Bit 2** – CPHA: Clock Phase:
    The settings of the Clock Phase bit (CPHA) determine if data is sampled on the leading (first) or trailing (last) edge of SCK.

AMIT

Serial Peripheral Interface:

Register Description of SPI:

- SPI Control Register "**SPCR**":
  - According to the last two bits configuration, the leading edge varies according to CPOL bit configuration, and the sampling at first or setup at first varies according to the CPHA configuration, so if:

    CPOL = 0, CPHA = 0:

    the leading edge is Rising edge and SPI will sample at it,
    the trailing edge is Falling edge and SPI will setup at it.

    CPOL = 1, CPHA = 0:

    the leading edge is Falling edge and SPI will sample at it,
    the trailing edge is Rising edge and SPI will setup at it.

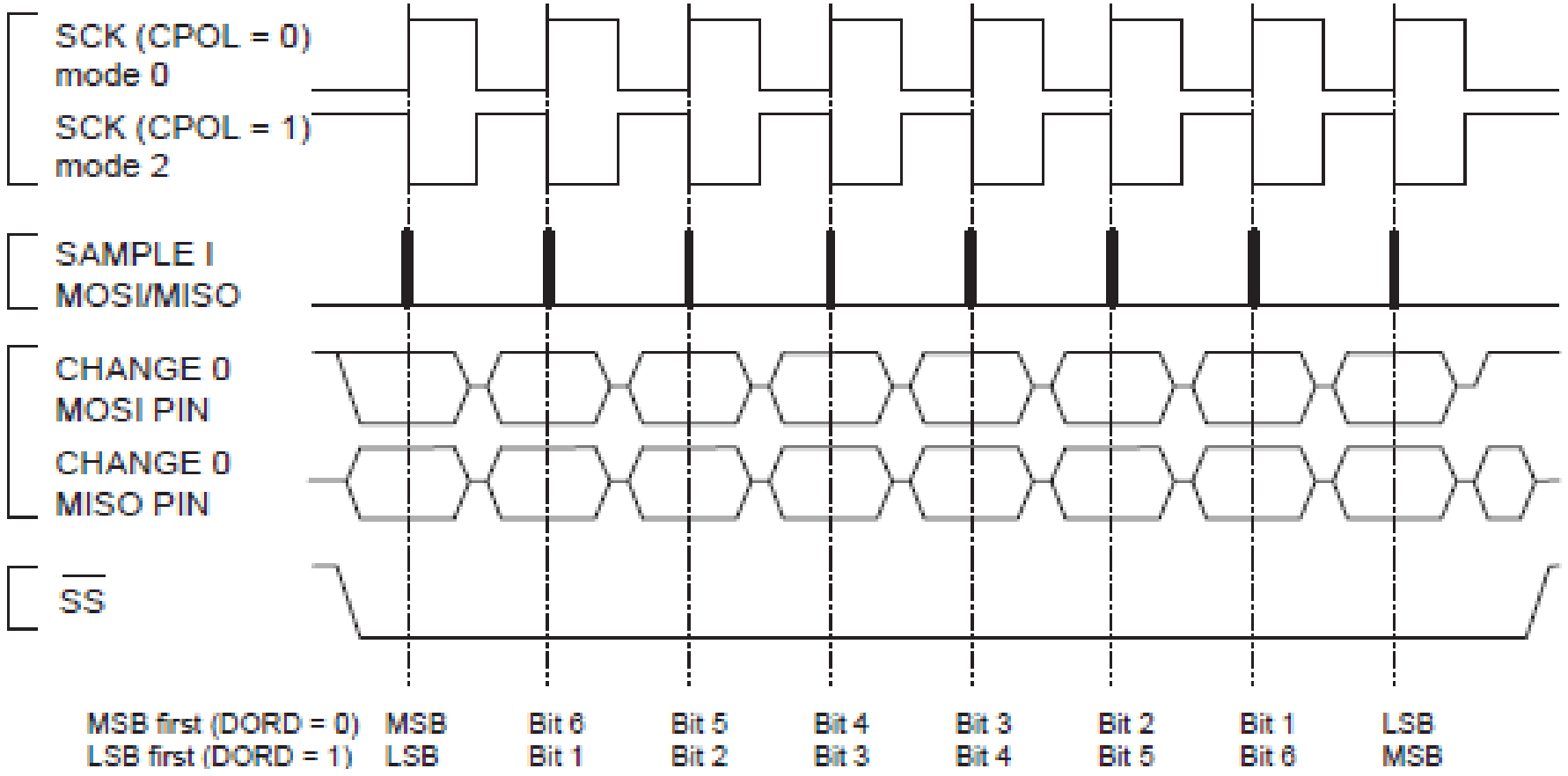
    Look at the figure exists in the next page:

AMIT

Serial Peripheral Interface:

Register Description of SPI:

- SPI Control Register "**SPCR**":



AMIT

Serial Peripheral Interface:

Register Description of SPI:

- SPI Control Register "**SPCR**":
  - According to the last two bits configuration, the leading edge varies according to CPOL bit configuration, and the sampling at first or setup at first varies according to the CPHA configuration, so if:

    CPOL = 0, CPHA = 1:

    the leading edge is Rising edge and SPI will setup at it,
    the trailing edge is Falling edge and SPI will sample at it.

    CPOL = 1, CPHA = 1:

    the leading edge is Falling edge and SPI will setup at it,
    the trailing edge is Rising edge and SPI will sample at it.

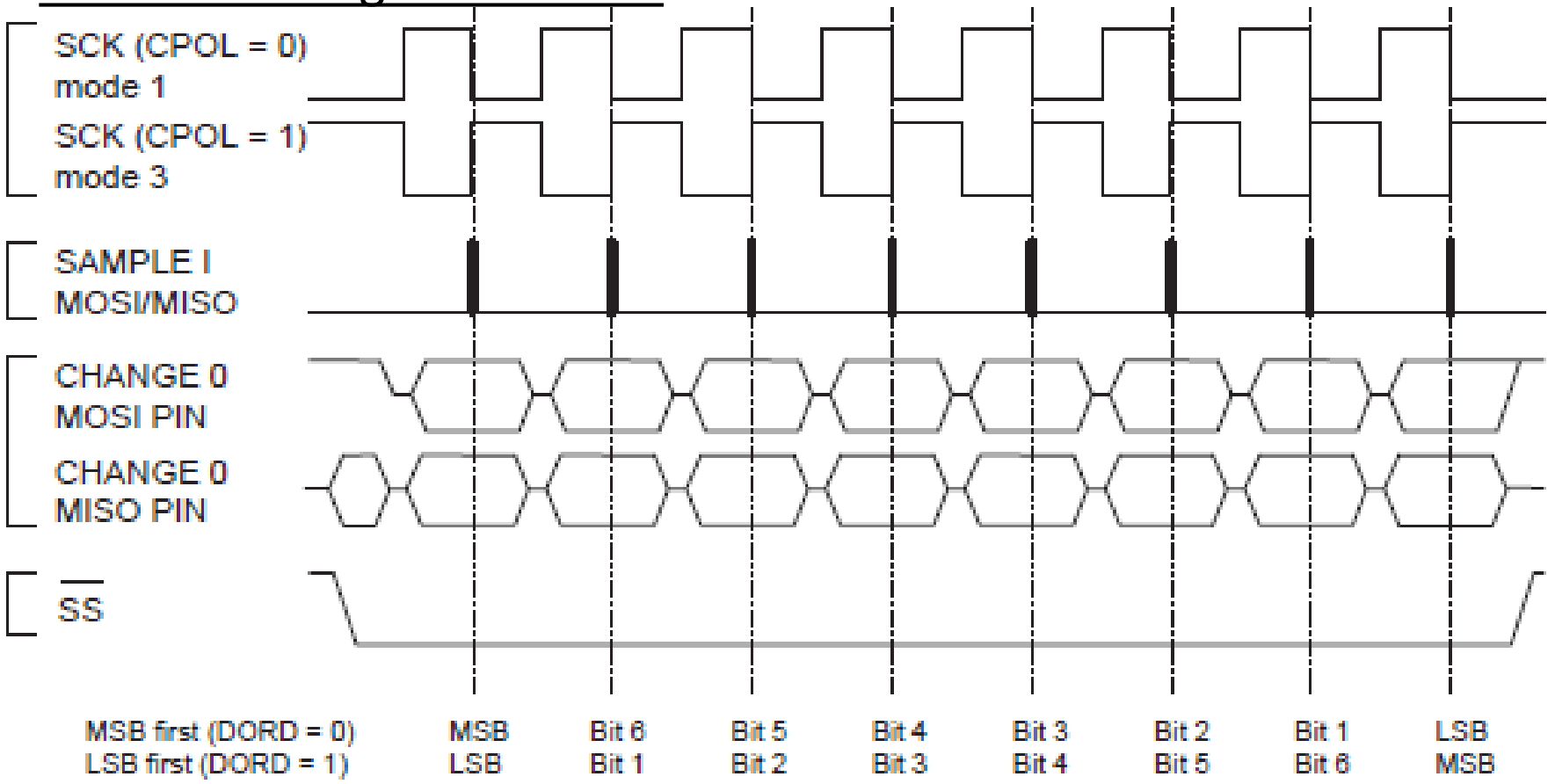
    Look at the figure exists in the next page:

AMIT

Serial Peripheral Interface:

Register Description of SPI:

- SPI Control Register "**SPCR**":



AMIT

Serial Peripheral Interface:

Register Description of SPI:

- SPI Control Register "**SPCR**":
  - **Bits 1, 0** – SPR1, SPR0: SPI Clock Rate Select 1 and 0:

    These two bits control the SCK rate of the device configured as a Master.
    SPR1 and SPR0 have no effect on the Slave.
    SPI can be run at a double speed mode, if the SPI2X bit exists into SPSR register is set.
    The relationship between SCK and the Oscillator Clock frequency F_CPU is shown in the following table:

Table 58. Relationship Between SCK and the Oscillator Frequency

SPI2X	SPR1	SPR0	SCK Frequency
0	0	0	$f_{osc}/4$
0	0	1	$f_{osc}/16$
0	1	0	$f_{osc}/64$
0	1	1	$f_{osc}/128$
1	0	0	$f_{osc}/2$
1	0	1	$f_{osc}/8$
1	1	0	$f_{osc}/32$
1	1	1	$f_{osc}/64$

AMIT

Serial Peripheral Interface:

Register Description of SPI:

- SPI Status Register "**SPSR**":
  - **Bit 7** – SPIF: SPI Interrupt Flag:

    When a serial transfer is complete, the SPIF Flag is set.

    An interrupt is generated if SPIE in SPCR is set and global interrupts are enabled.

Status Register – SPSR

Bit	7	6	5	4	3	2	1	0
	SPIF	WCOL	–	–	–	–	–	SPI2X
Read/Write	R	R	R	R	R	R	R	R/W
Initial Value	0	0	0	0	0	0	0	0

    If SS is an input and is driven low when the SPI is in Master mode, this will also set the SPIF Flag.

    SPIF is cleared by hardware when executing the corresponding interrupt handling vector.

    Alternatively, the SPIF bit is cleared by first reading the SPI Status Register with SPIF set, then accessing the SPI Data Register (SPDR).

AMIT

Serial Peripheral Interface:

Register Description of SPI:

- SPI Status Register "**SPSR**":
  - **Bit 6** – WCOL: Write Collision Flag:
    The WCOL bit is set if the SPI Data Register (SPDR) is written during a data transfer.
    The WCOL bit (and the SPIF bit) are cleared by first reading the SPI Status Register with WCOL set, and then accessing the SPI Data Register.

  - **Bit 5 : 1** – Res: Reserved Bits:
    These bits are reserved bits in the ATmega32 and will always read as zero.

AMIT

Serial Peripheral Interface:

Register Description of SPI:

- SPI Status Register "**SPSR**":
  - **Bit 0** – SPI2X: Double SPI Speed Bit:
    As we mentioned before, this bit makes SPI to run at double speed mode, so when this bit is written logic one the SPI speed (SCK Frequency) will be doubled when the SPI is in Master mode (see Table 58). This means that the minimum SCK period will be two CPU clock periods.
    When the SPI is configured as Slave, the SPI is only guaranteed to work at F_CPU/4 or lower.

AMIT

Serial Peripheral Interface:

Alternative Function of SPI:

- The SPI interface on the ATmega32 is also used for program memory and EEPROM downloading or uploading.
  As we know, our microcontroller has an in-circuit programming, so the default bootloader uses the SPI the receive the data of program, them passing it to the in-circuit programming to burn it on the flash memory or EEPROM memory if it is used as a program memory.

  For more details about SPI Serial Programming and Verification, see page 268 in the data sheet.

AMIT

Serial Peripheral Interface:

Initialization of SPI:

- First, select your
  - data order.
  - clock polarity.
  - clock phase.
  - speed of clock (normal or double).
  - mode of SPI (Master/Slave).

AMIT

Serial Peripheral Interface:

Initialization of SPI:

- Simple initialization function:

```
/* define the mode:   MSTR   or    SLAVE*/
#define SPI_MODE            MSTR
void SPI_init(void)
    {       /* normal speed*/
      SPSR = 0;
      /* master/slave, leading rising-sampling, MSB first, enable SPI*/
      #if SPI_MODE   ==  MSTR
            SPCR = (1<<MSTR) | (1<<SPR0) | (1<<SPR1) | (1<<SPE);
      #elif SPI_MODE  ==  SLAVE
             SPCR =(1<<4) | (1<<SPE);
      #endif
    }
```

AMIT

Serial Peripheral Interface:

Transceiver of SPI:

- Simple transceiver function:

```
u8 SPI_transceiver (u8  data)
   {      /* write dta in SPDR*/
     SPDR = data;
     /* wait for rising flag */
      while ( ! ((SPSR>>7) & 1)   );
      /* return the received data after exchanging data */
      return  data;
   }
```

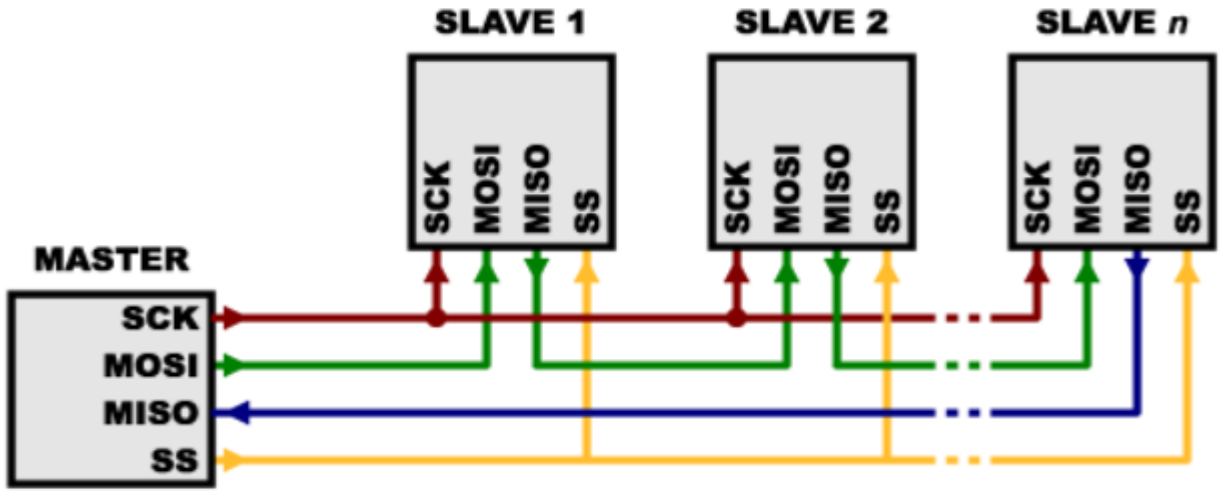
- Take care if you use this function at master mode,
  you must select the desired slave before calling the transceiver function.
- If the returned data is not important, just do not receive the return of function.

AMIT

Serial Peripheral Interface:

Daisy Chain Connection "Cascading" of SPI:

- The normal connection, the number of slaves are limited to number of DIO pins, so if the network requirement needs to exceed this number, Daisy chain connection must be used.



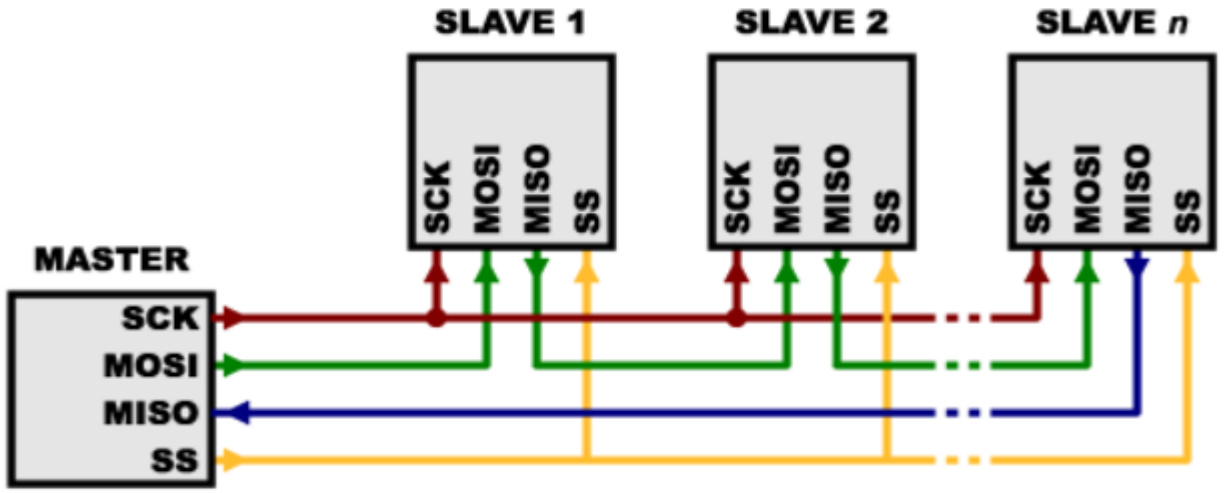
The number of slaves are unlimited, into this technique, all SS pins of slaves are driven to LOW, MOSI of master is connected to MOSI of first slave, the MISO of first slave is connected to the MOSI of second slave, the MISO of second is connected to MOSI of third slave, and so on until the last slave its MISO is connected to MISO pin to master.

AMIT

Serial Peripheral Interface:

Daisy Chain Connection “Cascading” of SPI:

- To transfer data as example “**A**” character to the second slaves, the protocol must be started, and the data of master will be transferred to first slave “**A**”, the data of first sl
will transferred to the second,
the data of second to the third,
and so on until the last salve will send its data to master, so the master must be send any Dummy data again to first slave, the data of first slaved finally reaches the second slave.



- Unfortunately, in this technique, data is not secured because it is shared with other ECUs.

Serial Peripheral Interface:

Daisy Chain Connection “Cascading” of SPI:

- The throughput will be reduced according to the place of slave.
- At the previous example, master sends two bytes to inform the intended slaves with data, so the throughput will be 50%.
- If the master wants to send a byte to the sixth slave as example, master must send this byte first then sends five dummy bytes,
  so the throughput in this case will be “1/6” 16.67 % and so on.

Guess how to implement a

Broadcast Connection???!!!!

AMIT

SPI Driver:

Time To Code

AMIT

THANK YOU!