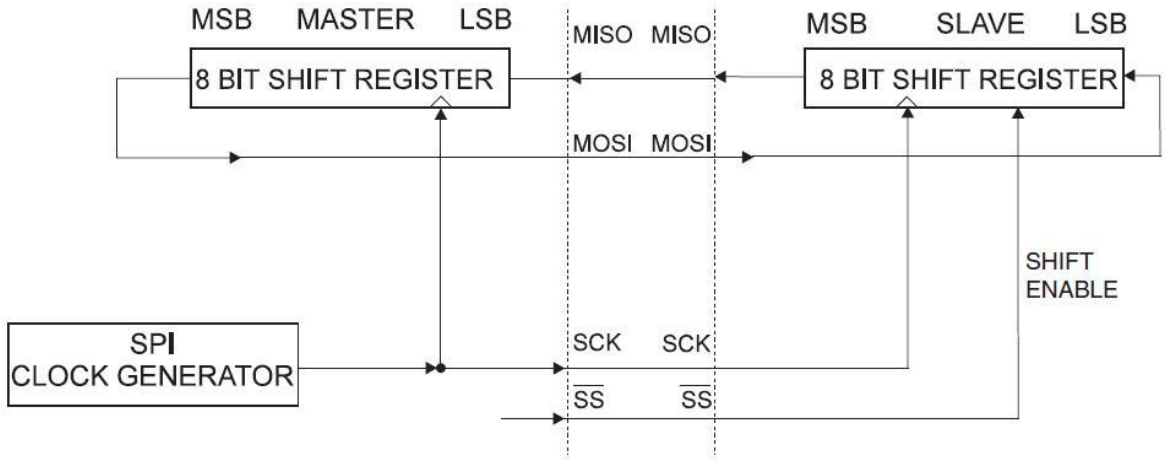
SPI Interface

Reference:

1. Reference System Design
2. Adafruit Motor Shield - Part 1



Consider the following C code example from Section 18.2 Overview from the datasheet.

Code Example 1

```
void SPI_MasterInit(void)
{
  /* Set MOSI and SCK output, all others input are low (default) */
  DDR_SPI = (1<<DD_MOSI)|(1<<DD_SCK);
  /* Enable SPI, Master, set clock rate fck/16 */
  SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0);
}

void SPI_MasterTransmit(char cData)
{
  /* Start transmission */
  SPDR = cData;
  /* Wait for transmission complete */
  while(!(SPSR & (1<<SPIF)))
  ;
}
```

1. What ATmega328P Port pins and Arduino Digital pins correspond to DD_MOSI and DD_SCK? (Hint: look down the Atmega 328P and Arduino columns in Table 2.1 "Microcontroller Resource Map - Motors" of the "Reference System Design" pdf document)

   ANSWER:

	ATmega328P	Arduino	
DD_MOSI	PB3 (MOSI, OC2A)	J3-4	Digital Pin 11
DD_SCK	PB5 (SCK)	J3-6	Digital Pin 13

2. The Adafruit motor shield SPI names are similar to the Atmel names, where DD_MOSI functionally is equivalent to DIR_SER (Digital Input Register Serial data input) and DD_SCK is functionally equivalent to DIR_CLK. What ATmega328P Port pins and Arduino Digital pins correspond to SER and CLK? (Hint: look down the Motor Shield column in Table 2.1 "Microcontroller Resource Map - Motors" of the "Reference System Design" pdf document)

   ANSWER:

	ATmega328P	Arduino	
DIR_SER	PB0 (ICP1)	J3-1	Digital Pin 8
DIR_CLK	PD4 (T0)	J1-5	Digital Pin 4

Adafruit implments the SPI interface in software. In otherwords it does not use the hardware SPI subsystem of the ATmega328P microcontroller. This was done in order to not lose the Timer 2's Output Compare register A output (OC2A) shared with the SPI Master Out Slave In (MOSI) pin. Adafruit uses the OC2A to generate a Pulse Width Modulation (PWM) output to control the speed of one of the H-Bridge outputs.

3. What is the name assigned by Adafruit to this PWM signal? (Hint: look down the Atmega 328P and Motor Shield columns in Table 2.1 "Microcontroller Resource Map - Motors" of the "Reference System Design" pdf document)

   ANSWER: PWM2A

4. What Motor is controlled by this PWM channel? (Hint: look down the Motor Shield and the Motor Shield PWM (H-Bridge) columns in Table 2.1 "Microcontroller Resource Map - Motors" of the "Reference System Design" pdf document. Once you have located the IC number and Pin number/name assigned by Adafruit to this signal look at Figure 2.9 "Output Connector X1" in the Reference System Design document.

   ANSWER: H-Bridge output signals M1A and M1B which control Motor 1.

In Code Example 1 the clock rate was set to fck/16.

```
/* Enable SPI, Master, set clock rate fck/16 */
SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0);
```

as defined by Table 18.5 in the Serial SPI document and Section 18.5.1 "SPCR - SPI Control Register" of the ATmega datasheet.

Table 18-5. 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	$f_{osc}/64$

Given an Arduino system clock frequency of 16 MHz this would correspond to a bit rate of 1 Mbps.

5. Write the C code expression to enable the SPI, define the microcontroller as the master and set the clock divisor to to fck/128.

   ANSWER:

   /* Enable SPI, Master, set clock rate fck/16 */

   SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR1)|(1<<SPR0);

```
Looking again at Code Example 1.
/* Wait for transmission complete */
  while(!(SPSR & (1<<SPIF)))
  ;
```

This instruction looks at the SPSR

6. Where could you find this register in the ATmega datasheet?

ANSWER:

Section 18.5.2 SPSR - SPI Status Register

Bit	7	6	5	4	3	2	1	0	
0x2D (0x4D)	SPIF	WCOL	–	–	–	–	–	SPI2X	SPSR
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Does this loop construct implement polling or an interrupt?

ANSWER: The loop polls the SPI Flag bit

Here is the Adafruit software version of the `SPI_MasterTransmit` method.

Code Example 2

```
void AFMotorController::latch_tx(void) {
 uint8_t i;
```

```
 //LATCH_PORT &= ~_BV(LATCH);
 digitalWrite(MOTORLATCH, LOW);     // - Output register clock low

 //SER_PORT &= ~_BV(SER);
 digitalWrite(MOTORDATA, LOW);      // - Serial data bit = 0

 for (i=0; i<8; i++) {              // - Shift out 8-bits
    //CLK_PORT &= ~_BV(CLK);
    digitalWrite(MOTORCLK, LOW);    //    - Shift clock low

    if (latch_state & _BV(7-i)) {   //    - Is current bit of
     //SER_PORT |= _BV(SER);                latch_state == 1
     digitalWrite(MOTORDATA, HIGH); //    -   Yes, serial data bit = 1
    } else {
     //SER_PORT &= ~_BV(SER);
     digitalWrite(MOTORDATA, LOW);  //    -   No, serial data bit = 0
    }
    //CLK_PORT |= _BV(CLK);
    digitalWrite(MOTORCLK, HIGH);   //    - Shift clock high, rising edge
 }                                  //      shift bit into shift register
 //LATCH_PORT |= _BV(LATCH);
 digitalWrite(MOTORLATCH, HIGH);    // - Output register clock high,
rising
}
```

7. Does this software implementation of the SPI use polling to detect when a byte has been transmitted?

   ANSWER: No, instead it uses a `for` loop.

8. What 8-bit unsigned variable holds the data to be transmitted?

   ANSWER: `latch_state`

9. What 8-bit unsigned variable holds the bit currently being sent?

   ANSWER: variable `i`

10. What conditional expression detects the state (1 or 0) of the currently selected bit?

    ANSWER: `if (latch_state & _BV(7-i))`