The LCD2S family of Serial LCD Displays can be configured to communicate via the I2C or SPI (mode 0) protocol. To use the SPI mode, both switches on the DIP switch must be set to the on position. This is a very simple protocol, and can easily be implemented via software - no hardware SPI peripheral is required!

In the SPI protocol, a byte is always written and read at the same time. All commands on the LCD2S are however half duplex! When writing a byte to the LCD2S, the byte read is ignored. When reading a byte from the LCD2S, we sent a dummy byte.

The SPI bus on the LCD2S has 4 lines:

- **CS**: Chip Select Input, active low. This signal is taken low for each byte written to the LCD2S.
- **CLK**: Clock Input. The device writing to the LCD2S will supply a clock signal on this line to clock bits into and out of the LCD2S.
- **SDI**: Data Input. Bits are clocked into the LCD2S on the rising edge of the CLK signal via this line.
- **SDO**: Data Output. Bits are clocked out of the LCD2S on the falling edge of the CS and CLK signal via this line.

When not reading data from the LCD2S, only 3 lines can be used, CLK, CS and SDI.

The following C code shows a simple function that writes and reads a byte to and from the LCD2S. The SPI_CS, SPI_CLK, SPI_IN and SPI_OUT defines must be set to the microcontroller port pins that are to be used. The SPI_CS, SPI_CLK, SPI_OUT ports have to be configured as outputs, and the SPI_IN port as an input before using this function!

```c
//Define all SPI Pins
#define SPI_OUT     RC4_bit     /* Define SPI SDO signal to be PIC port RC4 */
#define SPI_IN      RC5_bit     /* Define SPI SDI signal to be PIC port RC5 */
#define SPI_CLK     RC3_bit     /* Define SPI CLK signal to be PIC port RC3 */
#define SPI_CS      RB1_bit     /* Define SPI CS signal to be PIC port RB1 */

/**
 * This function writes a byte out onto the SPI OUT port, and reads a byte from
 * the SPI IN port.
 * @param c Gives the byte to write out onto the SPI port
 * @return Returns the byte read from the SPI IN port
 */
char spiPutGetByte(char c) {
    char ret;
    unsigned char mask;

    //SPI Mode 0. CS active low. Clock idle 0. Clock rising edge.
    SPI_CLK = 0;

    //Enable SPI communication. The SPI Enable signal must be pulsed low for each byte sent!
    SPI_CS = 0;

    //Ensure a minimum delay of 500ns between falling edge of SPI Enable signal
    //and rising edge of SPI Clock!
    Nop();
    mask = 0x80;                //Initialize to write and read bit 7
    ret = 0;                    //Initialize read byte with 0
    do  {
        SPI_OUT = 0;                //Clock out current bit onto SPI Out line
        if (c & mask) SPI_OUT = 1;
        SPI_CLK = 1;                //Set SPI Clock line
        if (SPI_IN) ret |= mask;    //Read current bit from SPI In line
        Nop();                      //Ensure minimum delay of 500ns between SPI Clock high and SPI Clock Low
        mask = mask << 1;
    } while (c & mask);
    return ret;
}
```
SPI_CLK = 0;                //Set SPI Clock line
mask = mask >> 1;           //Shift mask so that next bit is written and read from SPI lines
Nop();                      //Ensure minimum delay of 1000ns between bits
} while (mask != 0);

//Ensure a minimum delay of 750ns between falling edge of SPI Clock signal
//and rising edge of SPI Enable!
Nop();Nop();

//Disable SPI communication. The SPI Enable signal must be pulsed low for each byte sent!
SPI_CS = 1;
return ret;

Writing a string

The following C Source code will write "Hello" to line 1, and "World" to line two of the LCD2S display. See the LCD2S documentation for the exact command syntax!

spiPutGetByte(0xf5);   //Sync byte - each SPI command has to start with this byte!
spiPutGetByte(0x80);   //"Write parsed string command"
spiPutGetByte(0x0c);   //Clear display and go to beginning of first line.

//Write "Hello" in line one, and "World" in line two
spiPutGetByte('H');
spiPutGetByte('e');
spiPutGetByte('l');
spiPutGetByte('l');
spiPutGetByte('o');
spiPutGetByte(0x0a);   //Go to beginning of next line
spiPutGetByte('W');
spiPutGetByte('o');
spiPutGetByte('r');
spiPutGetByte('l');
spiPutGetByte('d');

Turning the backlight on and off, and changing it's brightness

The following examples show how to turn the backlight on and off, and change the brightness. The brightness can be set to 254 levels. See the LCD2S documentation for details!

spiPutGetByte(0xf5);   //Sync byte - each SPI command has to start with this byte!
spiPutGetByte(0x20);   //"Backlight Off" command
spiPutGetByte(0xf5);   //Sync byte - each SPI command has to start with this byte!
spiPutGetByte(0x28);   //"Backlight On" command
spiPutGetByte(0xf5);   //Sync byte - each SPI command has to start with this byte!
spiPutGetByte(0x81);   //"Set Backlight Brightness" command.
spiPutGetByte(150);   //Parameter for "Set Backlight Brightness" command. 0 will be off, and 254 will be
//maximum brightness.

At delivery the brightness is set to 176, which works very well for most displays.

Turning the display on and off, and changing it's contrast

The following examples show how to turn the display on and off, and change the contrast. The contrast can be set to 254 levels. See the LCD2S documentation for details!

spiPutGetByte(0xf5);   //Sync byte - each SPI command has to start with this byte!
spiPutGetByte(0x12);  //"Display Off" command
spiPutGetByte(0xf5);   //Sync byte - each SPI command has to start with this byte!
spiPutGetByte(0x1a);   //"Display On" command
spiPutGetByte(0xf5);   //Sync byte - each SPI command has to start with this byte!
spiPutGetByte(0x81);   //"Set Display Contrast" command.
spiPutGetByte(150);   //Parameter for "Set Display Contrast" command. 0 will be off, and 254 will be maximum contrast.
//Normally a value between 190 and 225 gives the best results.
spiPutGetByte(200);

At delivery the contrast is set to 208, which works very well for most viewing angles.