Lab 10 - Serial Peripheral Interface/ LCD Display

In this lab you are going to learn about how to use LCD displays and practice reading some datasheets.

Part 1 – Connecting the Chips

In your parser project add a new tab named "LCD.h"

Copy this code into that tab

```
uint8_t LCDS301C31TR_map [16] = { 0x3f };
class LCDS301C31TR
{
    void lcd_out(uint32_t value)
    {
      digitalWrite(LCDS301C31TR_cs,LOW);
      SPI.transfer((uint8_t)( (value >>  0) & 0xff ));
      SPI.transfer((uint8_t)( (value >>  8) & 0xff ));
      SPI.transfer((uint8_t)( (value >>  16) & 0xff ));
      digitalWrite(LCDS301C31TR_cs,HIGH);
      }
    uint8_t LCDS301C31TR_cs;
  public:
    uint32_t lcd_value;
    LCDS301C31TR(uint8_t cs)
    {
      pinMode(cs, OUTPUT);
      LCDS301C31TR_cs = cs;
      SPI.begin();
    }
    void lcd_present()
    {
      static uint8_t inverted = 0;
      if( inverted )
      {
        lcd_out( ~lcd_value );
        inverted = 0;
      }
      else
      {
        lcd_out( lcd_value );
        inverted = 1;
      }
    }

    uint8_t lcd_control( char* command )
    {
      if( strncmp( command, "lcd.", 4 ) == 0 ){
        char *sub_command;
        sub_command = &command[4];

        if( strcmp( sub_command, "inc" ) == 0 ) {
          if( lcd_value == 0 )
            lcd_value = 1;
          else
            lcd_value <<= 1;
          lcd_out( lcd_value );
        }
        else
        {
          return 0; //return false if lcd command not found
        }
        return 1; //return true if lcd command and found
      }
      else
        return 0; //return false if not lcd command
    }

    void lcd_display_hex(uint16_t value) {
      lcd_value = (
              ((uint32_t)LCDS301C31TR_map[(value >> 8) & 0x0f]) << 16 |
              ((uint32_t)LCDS301C31TR_map[(value >> 4) & 0x0f]) <<  8 |
              ((uint32_t)LCDS301C31TR_map[(value >> 0) & 0x0f])
            );
      lcd_present();
    }
    void lcd_display_base10(uint16_t value) {
      uint16_t v100 = value / 100;
      uint16_t v10 = (value - (v100 * 100)) / 10;
      uint16_t v1 =  (value - (v100 * 100) - (v10 * 10));

      lcd_value = (
              ((uint32_t)LCDS301C31TR_map[v100]) << 16 |
              ((uint32_t)LCDS301C31TR_map[v10]) <<  8 |
              ((uint32_t)LCDS301C31TR_map[v1])
            );
      lcd_present();
    }

};
```
At the top of your parser code

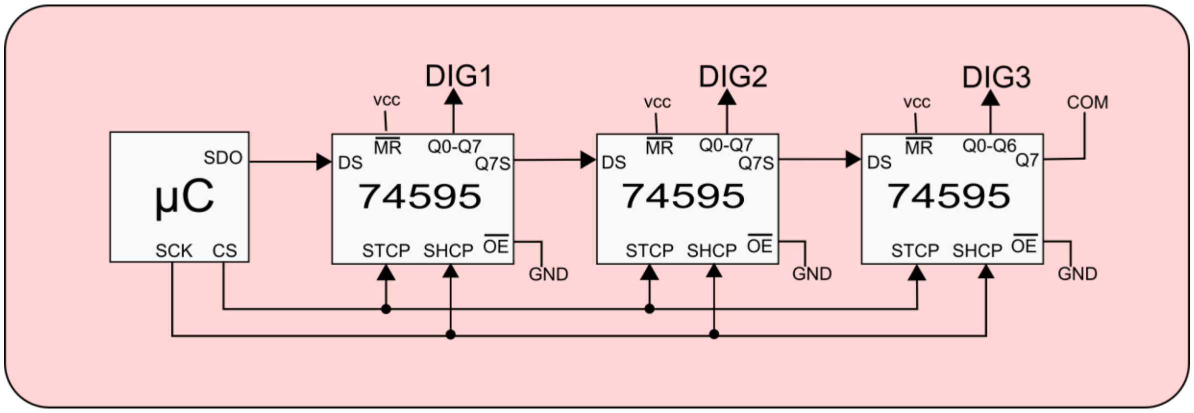
```
#include <SPI.h> //use the SPI library
#include "LCD.h" //needed because it is .h
LCDS301C31TR LCD(9); //create instance of class from LCD.h
                     //with pin 9 as chip select
```
Add a call to LCD.lcd_present() in the loop function so that it is called every cycle
May need to add delay (10)

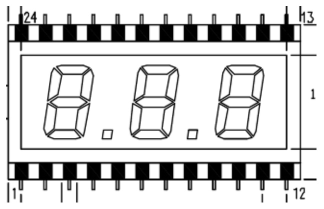
Finally add the following to the commands in your parser

```
    else if (LCD.lcd_control(command)); //just one line adds the commands
                                        //contained in lcd_control
```

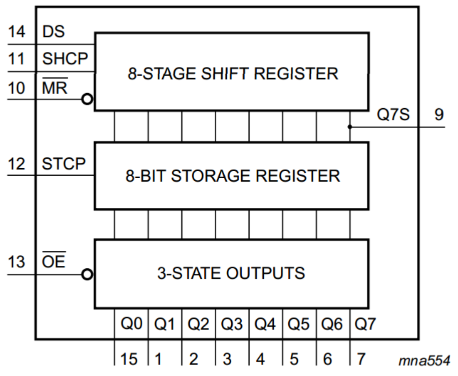
Connect the Shift Registers

Looking at the provided diagram and the datasheets connect the shift registers and LCD to the Microcontroller. With 0 connected to a 1 to b ...





PIN NO.	1	2	3	4	5	6	7	8
SEGMENT	COM	E1	D1	C1	DP1	E2	D2	C2
PIN NO.	9	10	11	12	13	14	15	16
SEGMENT	DP2	E3	D3	C3	B3	A3	F3	G3
PIN NO.	17	18	19	20	21	22	23	24
SEGMENT	B2	A2	F2	G2	B1	A1	F1	G1



Run program and verify it and the LCD are working.

Test that the "lcd.inc" command moves the active pixel on the LCD.

Check off

Call the instructor over see your program and LCD in action.

Be prepared to show where the "lcd.inc" command came from and explain how it works.

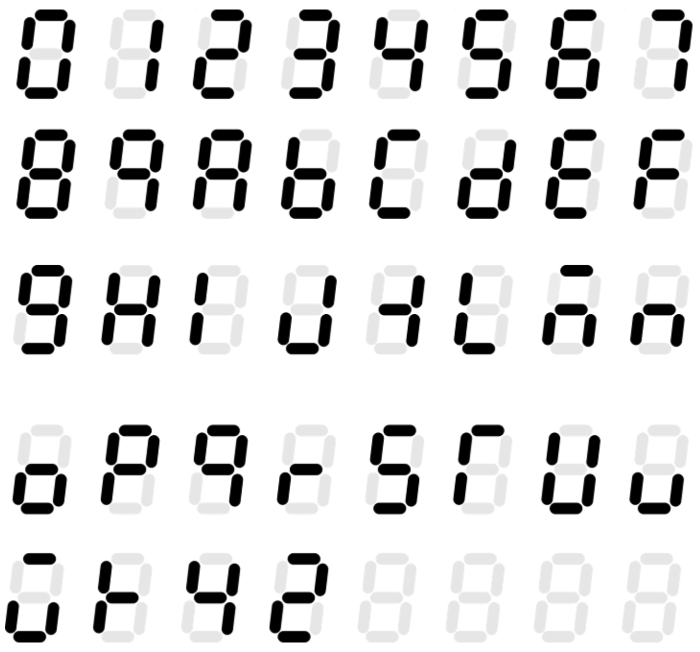
Part 2 – Character Bitmaps

Create a bitmap for each character in the provided table.

Looking at this lecture notes for this lab, for pictures and the order of bits determine Hex codes for all numbers and letters. Put them in the provided worksheet.

Value	p	g	f	e	d	c	b	a	Hex
0									
1									
2									
3									
4									
5									
6									
7									
8									
9									
A									
B									
C									
D									
E									
F									
G									
H									

Value	p	g	f	e	d	c	b	a	Hex
I									
J									
K									
L									
M									
N									
O									
P									
Q									
R									
S									
T									
U									
V									
W									
X									
Y									
Z									



Check off

Call the instructor over see your values.

Part 3 – Using the Codes

Make Array

In the lcd.h file there is a nearly empty array use the codes you just generated to fill in the array for 0-F so that the lcd_display_hex function has data to use.

Add calls to lcd_display_hex in "count up" and "count down" so that the numbers are displayed on the LCD also.

Check off

Call the instructor over see your program and LCD in action.

Part 4 – Display your Initials

Add a command called "lcd.me" to the ***lcd_control*** code

Assign the hex values for your initials directly to the variable lcd_value so that they show on the LCD

```
else if( strcmp( sub_command, "me" ) == 0 ) {
  lcd_value = LCDS301C31TR_map['J' - 'A' + 10] << 16 |
              LCDS301C31TR_map['S' - 'A' + 10] << 8 |
              LCDS301C31TR_map['C' - 'A' + 10] << 0;
}
else
```

Check off

Call the instructor over see your program in action.

Part 5 – Clear the display

Add a command called "lcd.clear"

Assign the hex values to blank out the LCD by directly writing to the variable lcd_value so that LCD will blank out when issued

```
else if( strcmp( sub_command, "clear" ) == 0 ) {
  lcd_value = 0;
}
```

Check off

Call the instructor over see your program in action.

Part 6 – Create a LCD chase pattern

Add a command called “lcd.chase”
Cause the LCD to display a chase pattern of your choice

```
        else if( strcmp( sub_command, "chase" ) == 0 ) {
            // loop (while or for)
          // in loop call lcd_present();
        }
```

Check off

Call the instructor over see your program in action.

Part 7 – Display ADC on LCD

Place in your code a call to lcd_display_hex that will continuously update the LCD with the output from an ADC connected to a pot.

Check off

Call the instructor over see your program in action.

Part 8 – Volt Meter

Make a copy of the lcd_display_hex function and modify it to output base10 instead of hex.
Replace the direct ADC value with one using base 10 converted to volts.

```
LCD.lcd_display_base10(((uint32_t)analogRead(A7)*330)/1023); //convert adc to
volts*100
LCD.lcd_value=LCD.lcd_value | 0x800000; //add the decimal point
```

Check off

Call the instructor over see your program in action.