**Graphic LCD**

This section shows how to connect a serial graphic LCD to the CSM-12C32 module and provide the C-codes for initializing the SCI port and for sending graphical commands to the LCD for drawing simple graphics.

The selected serial graphic LCD is uVolt’s SGD-A available at Spark Fun Electronics as item code LCD-08428. Pictures, ordering information and web link of the datasheet are shown below. [This LCD was sold out and unavailable from Spark Fun Electronics anymore.]

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Part Number</th>
<th>Weblink for the part</th>
<th>Description</th>
<th>Unit Price</th>
</tr>
</thead>
</table>

This graphic LCD is small and is equipped with a serial interface developed by uVolt. This module is based on a popular Nokia cellular LCD with the following features.

- 9600bps 8N1 Serial Communication
- 84x48 Pixels
- 6 lines of 14 characters
- Built in fonts and line commands
- Speaker with tone generation commands
- 3V to 5V (0.9 to 1.2mA)
- 128 bytes user EEPROM
Hardware interfacing to the Freescale 9S12C32 MCU
The wiring diagram of the serial LCD to the microcontroller are as follows:

In the graphic LCD, pin 1 is Vcc, pin 2 is Rx (serial input pin), pin 3 is Tx (serial transmit pin), and pin 4 is ground. The serial communication format is 9600 bps, 8 data bits, no parity and one stop bit. The order of sending the bits is as RS232 and is illustrated in the figure below.
Software development

Controlling the graphical display is done through a list of commands as shown in the table below. Every command is one byte long plus the arguments needed for execution. For detailed information of arguments, see its datasheet.

<table>
<thead>
<tr>
<th>Command</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘B’</td>
<td>Bitmap</td>
<td>Sends to display the bitmap that is being received in Rx pin</td>
</tr>
<tr>
<td>‘C’</td>
<td>Contrast</td>
<td>Adjusts LCD contrast. SGD-A is temperature compensated, so this command normally is unnecessary.</td>
</tr>
<tr>
<td>‘D’</td>
<td>Read-Ram</td>
<td>Reads a byte of user RAM, that is 64 bytes.</td>
</tr>
<tr>
<td>‘E’</td>
<td>Write-Ram</td>
<td>Writes a byte of user RAM, that is 64 bytes.</td>
</tr>
<tr>
<td>‘L’</td>
<td>Line</td>
<td>Draws a line in the display. It starts from last coordinates of pointer (Look at command ‘P’) and ends in selected coordinates.</td>
</tr>
<tr>
<td>‘N’</td>
<td>CLS</td>
<td>Clears screen and places the pointer at (0,0) (top left corner)</td>
</tr>
<tr>
<td>‘P’</td>
<td>Locate</td>
<td>Places the pointer in the selected coordinates. It is suitable to place text or lines.</td>
</tr>
<tr>
<td>‘R’</td>
<td>Read-eeprom</td>
<td>Read a byte of user EEPROM, that is 128 bytes.</td>
</tr>
<tr>
<td>‘W’</td>
<td>Write-eeprom</td>
<td>Writes a byte of user EEPROM, that is 128 bytes.</td>
</tr>
<tr>
<td>‘S’</td>
<td>String</td>
<td>Writes a string of text in current coordinates.</td>
</tr>
<tr>
<td>‘T’</td>
<td>Tone</td>
<td>Outputs a tone of frequency and time selected.</td>
</tr>
<tr>
<td>‘F’</td>
<td>Firmware Revision</td>
<td>Displays a FW version of SGD-A.</td>
</tr>
<tr>
<td>‘X’</td>
<td>Ping</td>
<td>Outputs a ‘X’ in Tx. Useful to debug a connection and operation of display.</td>
</tr>
<tr>
<td>‘Z’</td>
<td>Sleep</td>
<td>Start low power mode.</td>
</tr>
</tbody>
</table>
The LCD screen consists of 84X48 pixels as shown below. After power on or a clear screen, the pointer is placed at (0,0). It is the pixel on top left corner. From this point, x-coordinate is pointing to the right up to 83. The y-coordinate points down to 47. The pointer is for screen control purposes. For example, the command “P,10,1” will place the pointer at new coordinates (10,1). The command “L,19,10” will draw a line from the existing position at (10,1) to the point at (19,10). Afterwards, the pointer is updated to the last coordinates (19,10). The “L” commands takes 40 ms to execute.

Following is the code snippets for drawing a triangle on the graphical LCD.

```c
void draw_tri(void) { // triangle with vertices at (10,30), (30,10), and (30,50)
    char tri_1[]={'P',10,30,0};
    char tri_2[]={'L',30,10,0};
    char tri_3[]={'L',30,50,0};
    char tri_4[]={'L',10,30,0};

    putstring(tri_1);
    putstring(tri_2);
    putstring(tri_3);
    putstring(tri_4);
}

void putChar(unsigned char data){
    while ((SCISR1 & TDRE) == 0){
    }
    SCIDRL = data;
```
while(SCISR1&0x80==0) {
    waitms(5); // wait 5 ms
    return;
}

void putstring(char *data){
    int i;
    for(i=0;data[i]!=0;i++)
        putchar(data[i]);
    putchar(0);
    waitms(10); // wait 10 ms
    return;
}

Following is the code snippets for scrolling the word “MICROCONTROLLER” across the graphical LCD.

Void scrolling(void){
    char WORD1[] = {'S',14,'-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', 'M', 'O', 0};
    char WORD2[] = {'S',14,'-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', 'M', 'I', 'C', 'R', 'O', 'N', 0};
    char WORD3[] = {'S',14,'-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', 'M', 'I', 'C', 'R', 'O', 'N', 0};
    char WORD4[] = {'S',14,'-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', 'M', 'I', 'C', 'R', 'O', 'N', 0};
    char WORD5[] = {'S',14,'-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', 'M', 'I', 'C', 'R', 'O', 'N', 0};
    char WORD6[] = {'S',14,'-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', 'M', 'I', 'C', 'R', 'O', 'N', 0};
    char WORD7[] = {'S',14,'-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', 'M', 'I', 'C', 'R', 'O', 'N', 0};
    char WORD8[] = {'S',14,'-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', 'M', 'I', 'C', 'R', 'O', 'N', 0};
    char WORD9[] = {'S',14,'-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', 'M', 'I', 'C', 'R', 'O', 'N', 0};
    char WORD10[] = {'S',14,'-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', 'M', 'I', 'C', 'R', 'O', 'N', 0};
    char WORD11[] = {'S',14,'-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', 'M', 'I', 'C', 'R', 'O', 'N', 0};
    char WORD12[] = {'S',14,'-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', 'M', 'I', 'C', 'R', 'O', 'N', 0};
    char WORD13[] = {'S',14,'-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', 'M', 'I', 'C', 'R', 'O', 'N', 0};
    char WORD14[] = {'S',14,'-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', 'M', 'I', 'C', 'R', 'O', 'N', 0};
    char WORD15[] = {'S',14,'-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', 'M', 'I', 'C', 'R', 'O', 'N', 0};
    Putstring(WORD1);
    waitms(50);
    Putstring(clear);
    Putstring(WORD2);
    waitms(50);
    Putstring(clear);
    Putstring(WORD3);
    waitms(50);
    Putstring(clear);
    Putstring(WORD4);
    waitms(50);
    Putstring(clear);
    Putstring(WORD5);
    waitms(50);
    Putstring(clear);
    Putstring(WORD6);
    waitms(50);
    Putstring(clear);
    Putstring(WORD7);
    waitms(50);
    Putstring(clear);
    Putstring(WORD8);
    waitms(50);
    Putstring(clear);
    Putstring(WORD9);
    waitms(50);
    Putstring(clear);
Putstring(WORD10);
waitms(50);
Putstring(clear);
Putstring(WORD11);
waitms(50);
Putstring(clear);
Putstring(WORD12);
waitms(50);
Putstring(clear);
Putstring(WORD13);
waitms(50);
Putstring(clear);
Putstring(WORD14);
waitms(50);
Putstring(clear);
Putstring(WORD15);
waitms(50);
Putstring(clear);
}