FocusLCDs.com
LCDs MADE SIMPLE®
Ph. 480-503-4295 | NOPP@FocusLCD.com

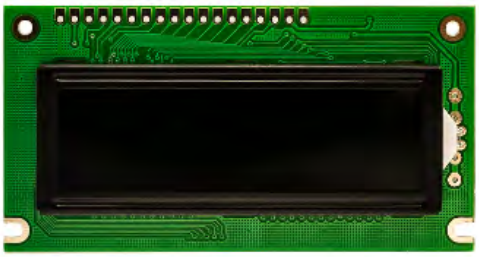
TFT | CHARACTER | UWVD | FSC | SEGMENT | CUSTOM | REPLACEMENT

Application Note FAN3201

Characters and Fonts on a Graphic LCD

In this application note we will discuss how to power and print text in different fonts on a 122x32 graphic display. This LCD is graphical which means it can display pixels as well as characters. The graphic display chosen in this project has a white LED backlight and will display dark gray pixels with a variable contrast. Graphic displays are great for creating a variety of images and characters because they allow for different pixel assignments.

Characters and Fonts on a Graphic LCD



Introduction

In this simple display project, we will be using a 122x32 mono-graphical display and an Arduino UNO microcontroller to feature the applications of the G123AXGFGSW6WN55AAC display module. As for any display, be sure the check the data sheet to determine the pin out, voltage ratings and driver of the module. Below is an overview of the display used in this project. (datasheet)

- 18 pins, 84x44mm
- 122x32 display, FSTN
- SBN1661G controller
- White LED side light
- Transmissive (negative)
- 8-bit parallel data input
- 5V power supply

What You'll Need

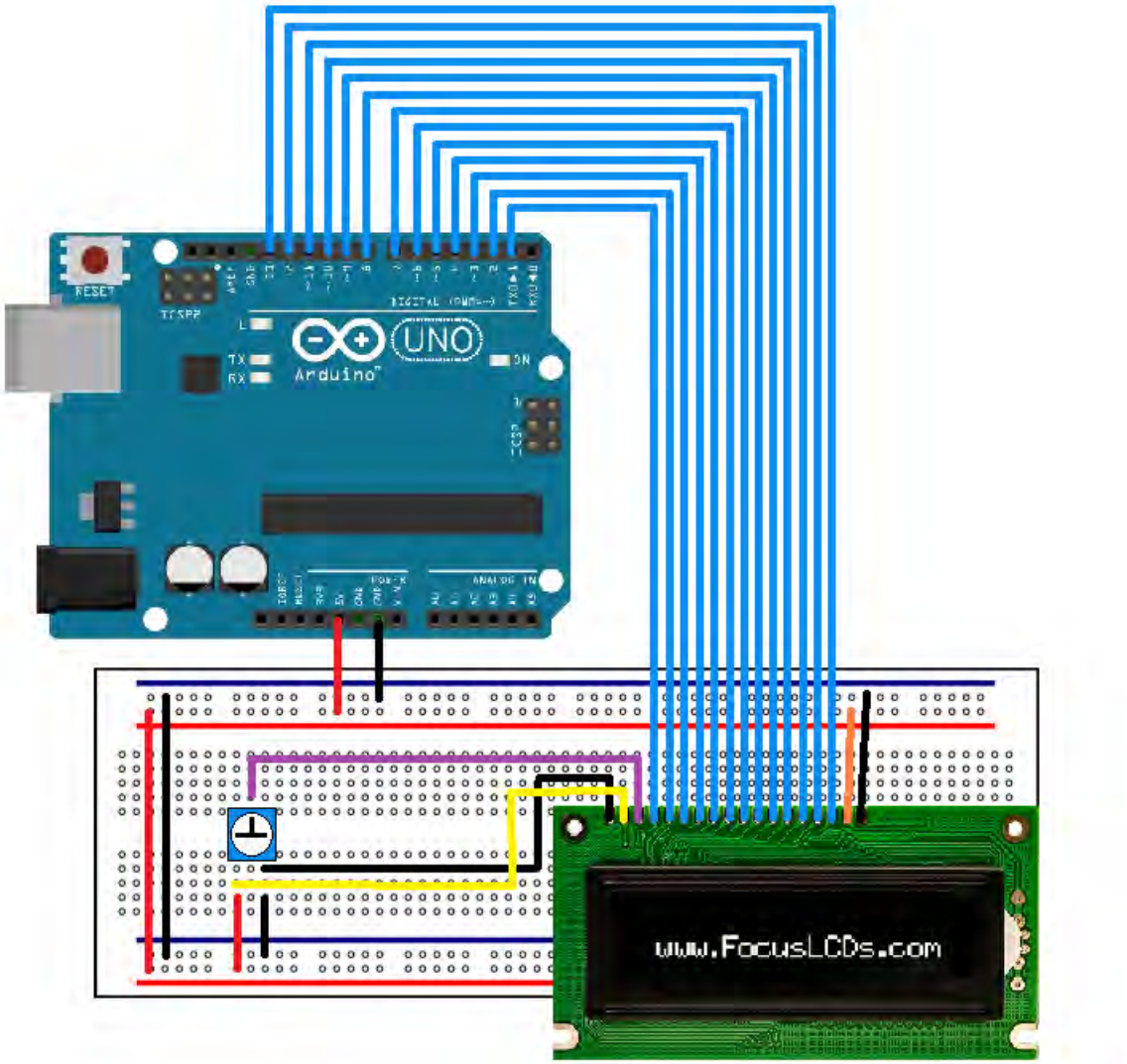
1. First and foremost you will need to Download the Arduino IDE software if you have not already. Alternatively, you can use the cloud-based version: "Ardunio Web Editor". We will come back to this program after we have the display wired up.

2. Below is a list of the physical materials you will need to setup the project.

QTY	Description	Store
1	Arduino UNO (Rev 3) with USB Cable	Arduino
1	G123AXGFGSW6WN55AAC Mono Graphic Display Module	FocusLCDs
1	Solderless Breadboard	Adafruit
1	Male-to-Male Jumper Wires (Set)	Adafruit
1	10k Potentiometer (Small)	Adafruit
1	Soldering Iron	
1	Solder	

Wiring

Below is a diagram representing how the graphic LCD should be wired and connected to the Arduino. This project requires a brief amount of soldering to connect pins to the LCD module. You can choose to use an 18-pin header or to solder wires directly to the LCD.



The LCD is powered by the Arduino at 5V+ through the USB port and can be varied through a 10k potentiometer and connected to ground. The LED backlight is also powered by the 5V+ provided by the Arduino. The data pins DB0-DB7 and the control pins are all connected to digital pins 1-13 on the Arduino. Below is a description of each of the pin connections for further clarification.

Pin No.	Symbol	Description	Connection	I/O
1	VSS	Signal ground for LCM	GND pin of potentiometer	P
2	VDD	Input voltage power supply: +5V	+5V pin of potentiometer	P
3	VO	LED contrast adjustment	Output pin of potentiometer	P
4	RST	Reset signal of device	Digital pin 1 (D1) of Arduino	O
5	E1	Enable clock input, SEG (1~61)	Digital pin 2 (D2) of Arduino	I
6	E2	Enable clock input, SEG (62~122)	Digital pin 3 (D3) of Arduino	I
7	R/W	Read/write signal	Digital pin 4 (D4) of Arduino	I
8	A0	Register select input	Digital pin 5 (D5) of Arduino	I
9	DB0	Data bus	Digital pin 6 (D6) of Arduino	I/O
10	DB1	Data bus	Digital pin 7 (D7) of Arduino	I/O
11	DB2	Data bus	Digital pin 8 (D8) of Arduino	I/O
12	DB3	Data bus	Digital pin 9 (D9) of Arduino	I/O
13	DB4	Data bus	Digital pin 10 (D10) of Arduino	I/O
14	DB5	Data bus	Digital pin 11 (D11) of Arduino	I/O
15	DB6	Data bus	Digital pin 12 (D12) of Arduino	I/O
16	DB7	Data bus	Digital pin 13 (D13) of Arduino	I/O
17	LED+	LED backlight power supply (+5V)	+5V	P
18	LED-	LED backlight ground	GND	P

I: Input, O: Output, P: Power

Starting It Up

Now that the LCD is connected and powered to the Arduino it's time to turn it on. Plug the Arduino into the computer with the USB cable. The backlight of the LCD should turn on. If the backlight does not turn on check to verify that pin 17 is receiving +5V and that all connections are secure. If the backlight LED is on that's a good sign.

Now open-up the Arduino IDE software. We're going to need a library that handles the driver that is inside of the LCD. To do this go to Sketch -> Include Libraries -> Manage Libraries. A search box should pop-up to find open sources libraries for the Arduino platform. Or you can download it from GitHub. The library chosen for this project is "U8glib", (version 1.18.0) chosen specifically because it includes the information that we need for programming the SBN1661G driver. Find this library and install it.

After the library is installed it is time to test a program. U8glib has many test programs but we can start with the classic "Hello World". Once opening this program (File-> Examples->U8glib->Hello World) you will see a long list of other drivers and LCD's that this example supports. For this project we are using a 122x32 pixel module with the SBN1661G driver which is listed as follows:

```
U8GLIB_SBN1661_122X32 u8g(6,7,8,9,10,11,12,13,2,3,5,4,1);
```

HelloWorld §

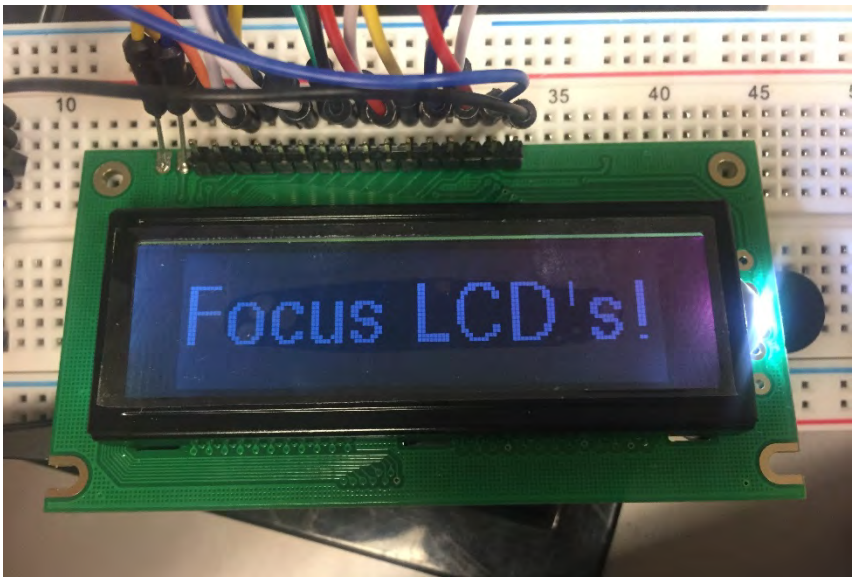
```
  LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
  CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
  ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
  ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

*/


#include "U8glib.h"


U8GLIB_SBN1661_122X32 u8g(6,7,8,9,10,11,12,13,2,3,5,4,1);



void draw(void) {
  // graphic commands to redraw the complete screen should be placed here
 // u8g.setFont(u8g_font_unifont);
  u8g.setFont(u8g_font_fur14);
  u8g.drawStr( 0, 22, "Focus LCD's!");
}

```

Once this line is uncommented, we can compile then upload. You should see Hello World! Printed across the screen.

The font can also be changed by commenting or changing the constructor:

//u8g.setFont(u8g_font_unifont);

A list of fonts and sizes that are supported by Ug8lib can be found on the GitHub library wiki. There are a variety of fonts and sizes that can now be printed on the screen. Just choose one that you like and replace the name in the “setFont” section. Graphic displays are ideal for using non-typical characters because there have a greater range of pixels. You can print a character by using the ug8.print() function.



Summary

There are various other prewritten programs in the Ug8lib that you can now try out. Be sure to include the constructor that we set up as written above. A good sketch to try out is the Menu function as graphic LCD's are often great screens for display menus. Another fun project is to display bitmaps on the screen or various fonts that are not supported by character displays. In the next project we will further discuss how to create bit maps and a menu for a larger graphical display.

DISCLAIMER

Buyers and others who are developing systems that incorporate FocusLCDs products (collectively, “Designers”) understand and agree that Designers remain responsible for using their independent analysis, evaluation and judgment in designing their applications and that Designers have full and exclusive responsibility to assure the safety of Designers' applications and compliance of their applications (and of all FocusLCDs products used in or for Designers’ applications) with all applicable regulations, laws and other applicable requirements.

Designer represents that, with respect to their applications, Designer has all the necessary expertise to create and implement safeguards that:

(1) anticipate dangerous consequences of failures

(2) monitor failures and their consequences, and

(3) lessen the likelihood of failures that might cause harm and take appropriate actions.

Designer agrees that prior to using or distributing any applications that include FocusLCDs products, Designer will thoroughly test such applications and the functionality of such FocusLCDs products as used in such applications.

[1]Arduino is an open-source development platform for easily building electronics projects that can electrically sense and control other objects. Arduino boards are primarily based on the Atmel AVR (8-bit) microcontroller (Example: Arduino UNO).