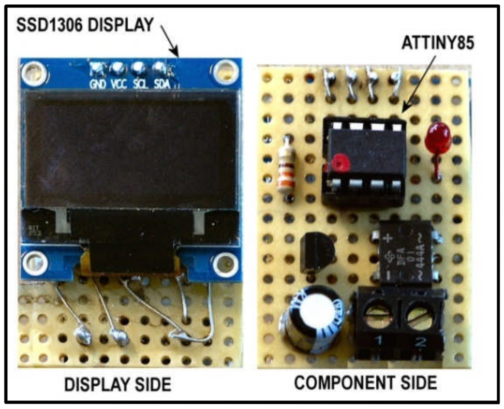
Animated Window Signs for HO Scale Buildings

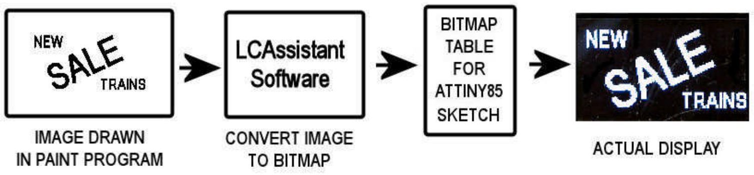
By Fred Miller, MMR

Author's HO Scale Building Signs

The recent availability of very inexpensive small electronic displays has made it possible to build animated displays suitable for model store front window signs. Using only a few electronic components a display circuit board can be built for less than $10. The SSD1306 0.96" by 0.48" OLED display is just the right size for an HO scale store front sign. The displays are available with either white or light blue text or images and can be controlled with micro-controllers such as the Arduino series. Pre-coded software libraries to control the display are readily available making any other programming to be minimal. See Appendix C for a sample one screen program. After developing the software in the traditional manner on an Arduino-UNO, the author chose to move the software to an ATTINY85 microcontroller to enable a very small electronics circuit board which would fit easily inside an HO building.



These small displays can handle up to a 128 x 64 pixel two-color image (white or light blue on black.) Microsoft's Paint program or one of the various graphics programs such as Paint Shop Pro can be used to create the image. The black pixels from the drawing are transcribed into white pixels on the display. A free program called LCD Assistant can be used to convert the image into a data bitmap required by the software.



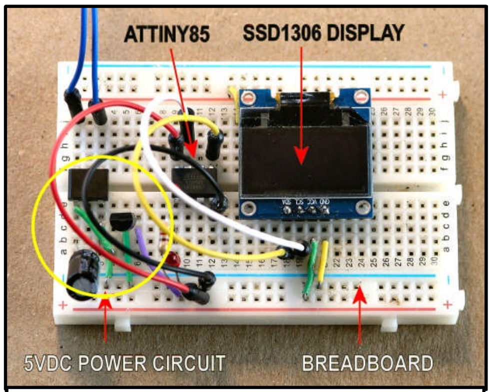
Several different software libraries available for use on an Arduino-UNO or other large boards can develop three different font sizes directly as well as displaying bitmap image data. The library the author used was developed for operation on the smaller ATTINY85 microcontroller but because of memory size restrictions it does not have as rich a command set. It supports only one small text font and no drawing or scrolling features. However, it works very nicely for displaying images up to a full 128x64 pixels. As many as 6 full screen bitmaps can be retained in the ATTINY85 memory. The program (or sketch as it is called in the ARDUINO world) running in the ATTINY85 can display any one of the images. The author's above pictured store window project displays a random selection of images each for a random number of seconds.

Developing the Software

As noted, the ATTINY85 micro-controller chip is used for controlling the SSD1306 Display board. The ATTINY85 is a small 8-pin chip which takes the same software (sketch) as the bigger brother Arduino boards but of course it has less memory and fewer control pins. It uses a 5VDC power supply which can be constructed with a few electronic components to accept a broad range of input voltages (6-12VDC or AC). However the ATTINY85 does not have the circuitry to enable programming directly from the PC based Arduino IDE (Integrated Development Environment) as do the Arduino-UNO and other larger boards. Instead a special programming board is needed to program the ATTINY85. Appendix A describes the steps necessary to prepare the PC Arduino IDE to program and download sketches to an ATTINY85.

Sketches are developed in the PC Arduino IDE in the usual fashion. Modelers not familiar with the Arduino can find many tutorials and help documents on the Internet. (See Appendix B). A copy of a simple sketch is shown in the Appendix C. The sketches are saved in the Arduino files section of the PC running the IDE. The Support Library which is used for this project is further described in Appendix A. This simple sketch makes use of this library as well as an image bitmap created in the manner described in Appendix A. The bitmap could be embedded in the sketch but a file saved in the same PC folder as the sketch is more convenient. When building the sketch, the ATTINY85 characteristics need to be selected in the IDE 'Tools' menu. After testing for appropriate links, spelling and punctuation in the sketch using the 'compile' feature of the Arduino IDE, the sketch can be downloaded to the ATTINY85 using the AVR Tiny Programmer.

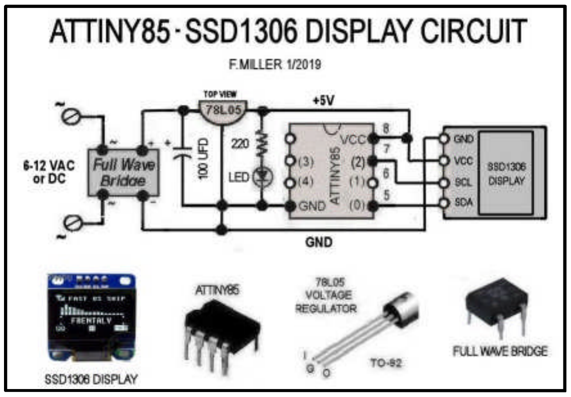
It is worthwhile to initially test the downloaded sketch with the programmed ATTINY85 plugged into a Breadboard. Jumpers from the ATTYINY85 to the SSD1306 Display module as well as a +5VDC power supply will validate the operation of the sketch. If a 5VDC supply is not available the power supply from the project can be built right on the breadboard using a 6-12VDC or AC input.



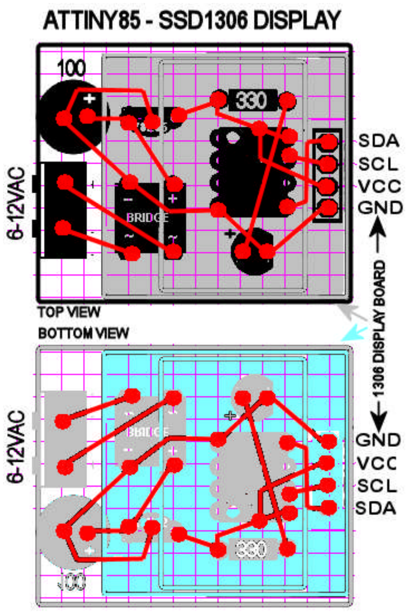
ATTINY85 and Display on Breadboard

Building the circuit board

The Display control circuit consists of 3 components (full-wave Bridge, 5V regulator, 100ufd capacitor) in addition to the ATTINY85 and the SSD1306 Display module. An LED and dropping resistor is an optional addition but is useful to indicate that power has been provided to the board. Appendix B lists the parts used for the circuit.

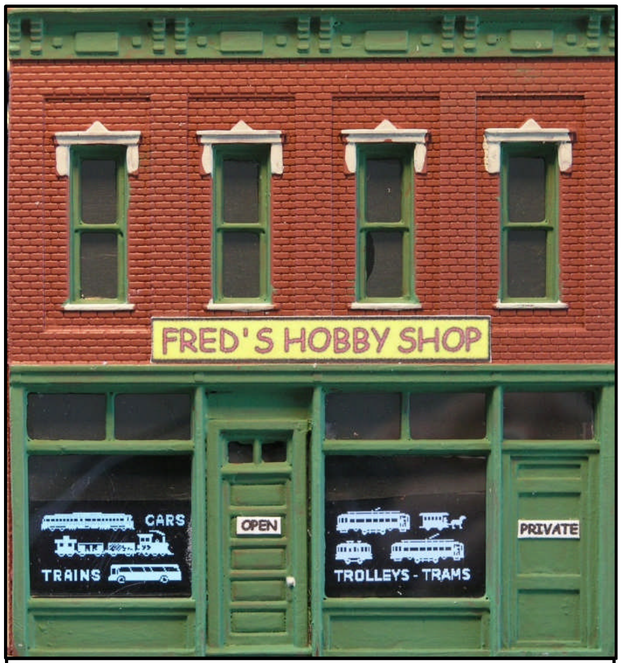


The display control board is built on a small piece of 'perf board' cut to match the display module. The author's favored construction technique is to draw out the component layout (using graphic software such as Paint Shop Pro) and graphically connect the components on the top (component) side of the board. He then graphically flips the drawing to show the connections to be soldered on the wiring side of the board.

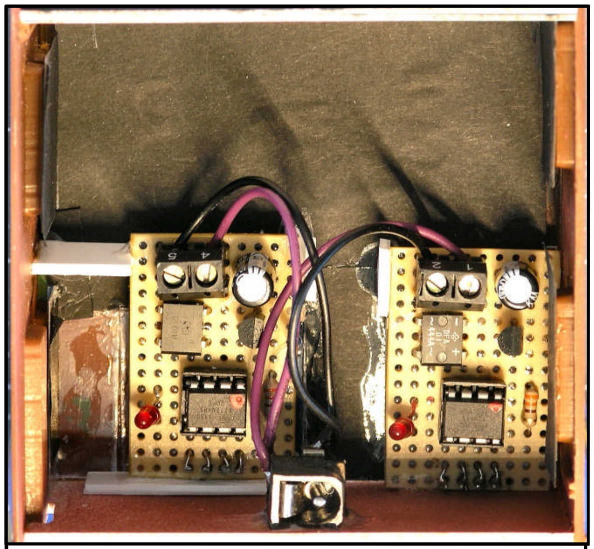


The illustration shows how the SSD1306 Display module (shown in blue) is connected over the wiring side of the board. It is good practice to test the constructed board (before connecting the display or plugging in the ATTINY85) by first applying power and measuring for correct +5VDC voltage. If the optional LED was included it should light at this point. Then plug in the ATTINY85 but connect the display board using jumper wires. If the operation of the display looks correct it is useful to coat the wiring side of my board with 5-min Epoxy to secure, insulate and protect the wiring. The Display module can then be soldered to the wires provided on the control board.

For those modelers that do not want to get involved with programming the ATTINY, the author offers to program a modeler's provided chip with provided 128x64 B&W .bmp images (maximum 6) without charge except for mailing expenses. In addition, copies of referenced files are available by email.

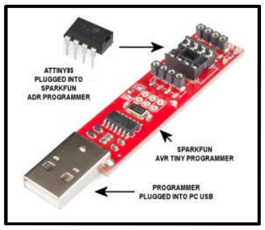


FRONT VIEW OF WINDOWS PROJECT



INSIDE VIEW OF WINDOWS PROJECT

Appendix A
Preparing to program the ATTINY85 for the SSD1306 Display

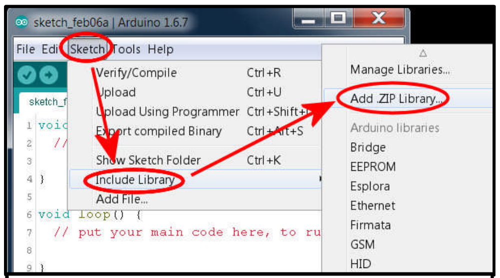


There are four modifications to the Arduino IDE (Integrated Programming Environment) to enable programming and downloading a sketch to an ATTINY85 chip. Circuits and instructions for using an Arduino-UNO to program the chips are published on the Internet. However using the Sparkfun Tiny Programmer board (PGM-11801) is a simple, ready-made process. Further help to accomplish many of the following steps is available on the Arduino support site.

Assuming the free Ardinuino IDE is loaded and operating on the PC:

1. The IDE must have knowledge of the ATTINY. All micro-controllers are defined to the IDE using "board cores". The IDE already contains cores for the popular Arduino boards. A useful core for the ATTINY line can be found at https://github.com/SpenceKonde/ATTinyCore.
   Download the .zip, extract, and place in the hardware folder inside the IDE's sketchbook folder (if there is no hardware folder, create it). You can find the location of the sketchbook folder in the Arduino IDE at File > Preferences -> Sketchbook location. Startup the Arduino IDE and the 'ATTINY Series X5' should show in the 'Boards' command of the 'Tools' pulldown menu. Series

2. The SSD1306 library needs to be installed in the 'Library' folder of the sketchbook folder. A nice library suitable for the ATTINY85 can be found at: https://github.com/anothermist/LIBRARIES/blob/master/SSD1306_minimal/FSNKTHDIDWXMIF4.zip.
   Download the library and then use the Arduino IDE to load the library from the .zip file. The 'Add .zip library' is a command in the 'Include Libraries' command of the 'sketch' pull down menu.



```
void
  //
}

void loop() {
  // put your main code here, to ru
}
```

Commands to add a .ZIP Library

3. Drivers must be installed in order to use the Sparkfun Tiny Programmer board. http://www.adafruit.com/downloads/usbtiny_signed_8.zip provides the drivers and installation software. The Sparkfun site for the 'PGM-11801 Tiny AVR PROGRAMMER' at: https://www.sparkfun.com/products/11801 provides additional information about Drivers for the board. The programmer board is a USB device which plugs into the PC's USB ports. The 'port' in the Arduino IDE when programming an ATTINY will be blanked out. Instead select the USBtinyISP(ATtiny) for the programmer.

A graphics program is needed to create the screen images. Most all of the available graphics programs such as PaintShopPro or Photoshop will do the job. However, the Microsoft Paint program provided as a part of Windows is capable of doing this. Remember to create a black and white (2-color) image sized at 128x64 pixels and save in monocrome .bmp format appropriate for loading into the LCD Assistant program.

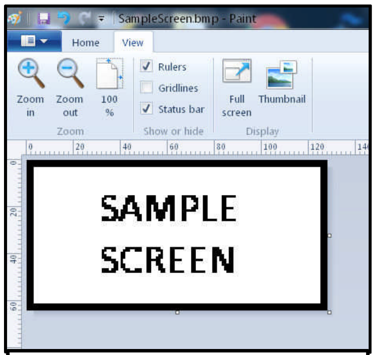
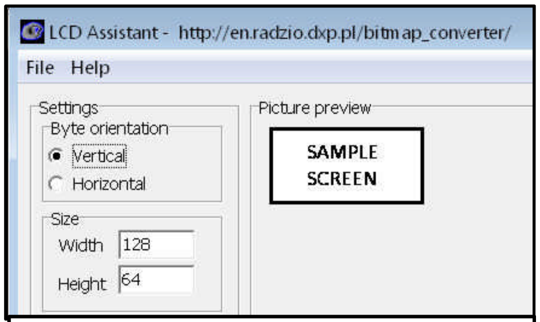


Image drawn in Paint program

A very useful tool to convert the images for SSD1306 display is a free program called LCD Assistant. It can be downloaded from:

http://en.radzio.dxp.pl/bitmap_converter/ and installs in the PC in the usual manner. A good tutorial on the use of LCDAssistant can be found at: https://www.instructables.com/id/Display-Images-on-OLED-Screen-With-Arduino-ATtiny8/


LCD Assistant - http://en.radzio.dxp.pl/bitmap_converter/

File Help

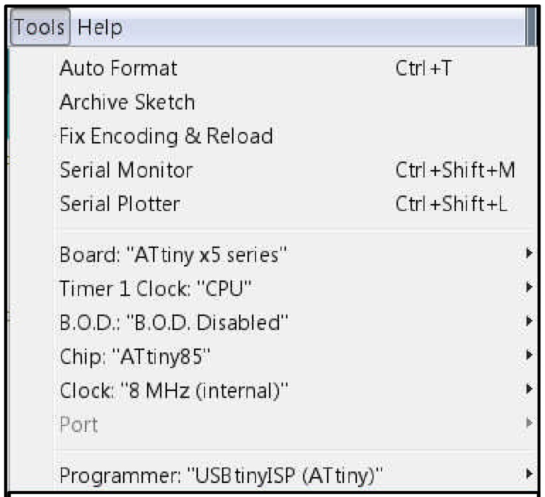
Settings
Byte orientation: Vertical (selected)
Size: Width 128, Height 64

Picture preview

Image loaded into LCD Assistant

Some Cautions

Depending upon the orientation of the Display screen within the modeled building, it may be necessary to rotate the displayed image 180 degrees. This is easily accomplished within the graphics drawing program prior to converting the image to the bitmap file with LCD Assistant.


Tools Help

Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L

Board: "ATtiny x5 series"
Timer 1 Clock: "CPU"
B.O.D.: "B.O.D. Disabled"
Chip: "ATtiny85"
Clock: "8 MHz (internal)"
Port

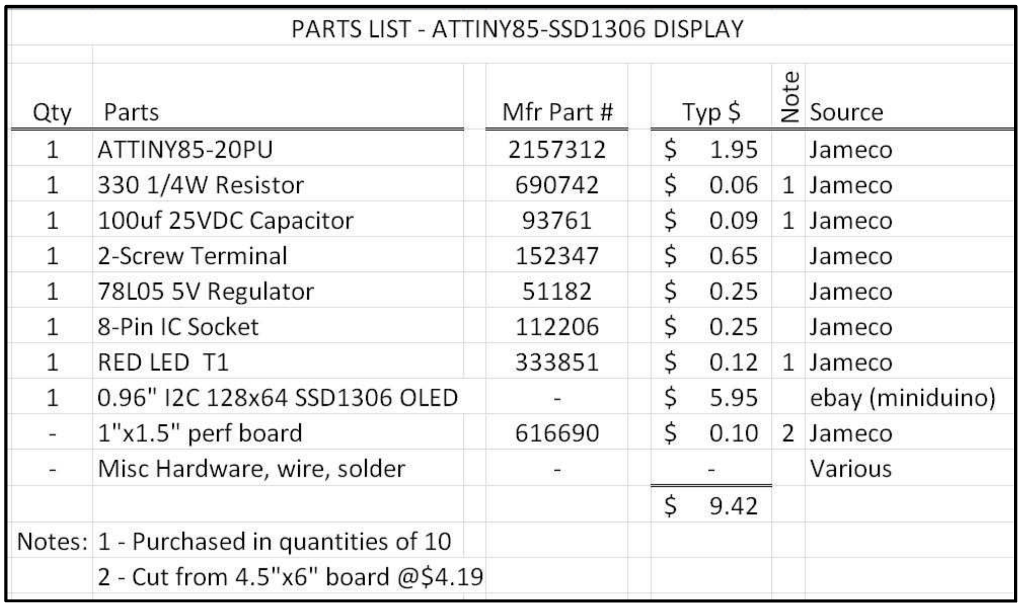
Programmer: "USBtinyISP (ATtiny)"

Commands to set ATTINY Clock Speed

New ATTINY85s may be delivered with an internal clock setting of 1 MHz. The sketch requires that the ATTINY85 be ‘programmed’ to run at the alternate 8 MHz setting. This is easily accomplished with the Arduino IDE program jwith the ATTINY85 plugged into the Sparkfun’s Tiny AVR Programmer and it, in turn, plugged into a PC’s USB port.

In the ‘Tools’ menu with all of the settings shown in the figure to the left, the ‘Burn Bootloader’ command at the bottom will set up the ATTINY85 for the internal 8 MHz clock speed.

Appendix B



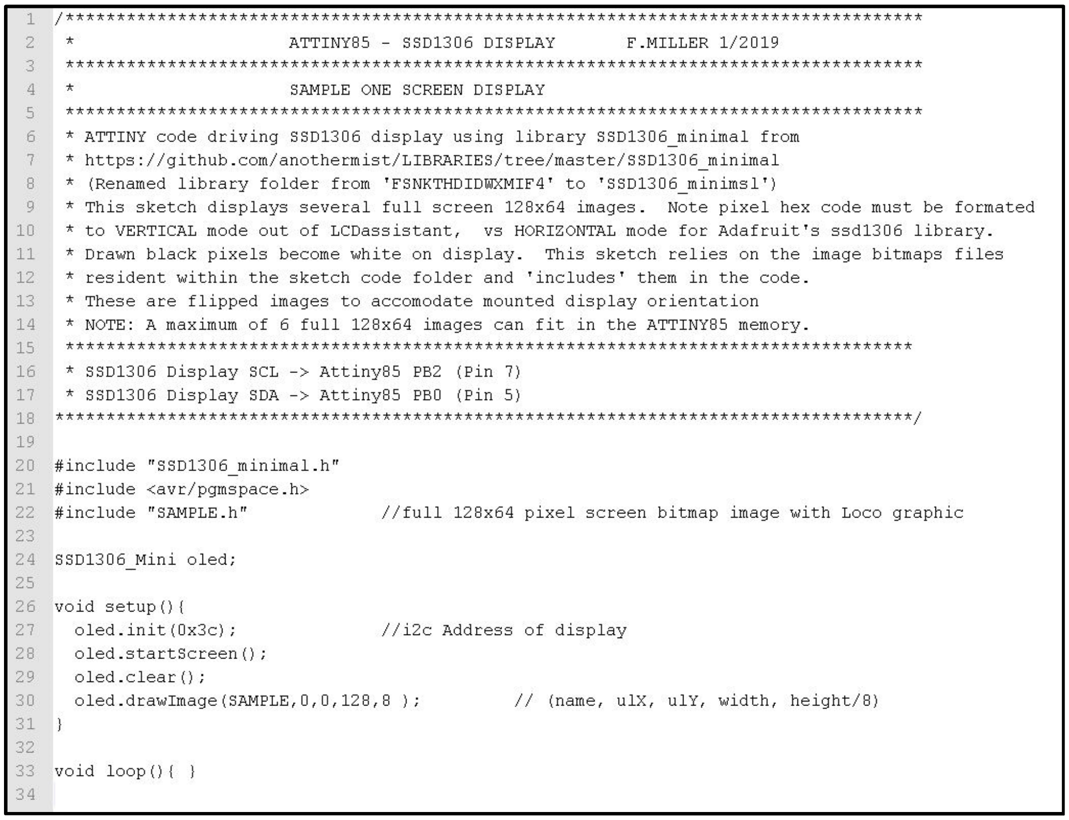
PARTS LIST - ATTINY85-SSD1306 DISPLAY					
Qty	Parts	Mfr Part #	Typ $	Note	Source
1	ATTINY85-20PU	2157312	$ 1.95		Jameco
1	330 1/4W Resistor	690742	$ 0.06	1	Jameco
1	100uf 25VDC Capacitor	93761	$ 0.09	1	Jameco
1	2-Screw Terminal	152347	$ 0.65		Jameco
1	78L05 5V Regulator	51182	$ 0.25		Jameco
1	8-Pin IC Socket	112206	$ 0.25		Jameco
1	RED LED T1	333851	$ 0.12	1	Jameco
1	0.96" I2C 128x64 SSD1306 OLED	-	$ 5.95		ebay (miniduino)
-	1"x1.5" perf board	616690	$ 0.10	2	Jameco
-	Misc Hardware, wire, solder	-	-		Various
			$ 9.42		
Notes:	1 - Purchased in quantities of 10				
	2 - Cut from 4.5"x6" board @$4.19				

References:

- Arduino website (for tutorials, etc.): https://www.arduino.cc/
- Jameco website (for parts): https://www.jameco.com
- Ebay (minidunio) website for Display: https://www.ebay.com/itm/US-0-96-I2C-IIC-Serial-128X64-LED-OLED-LCD-Display-Module-for-Arduino-White/382557870211?epid=17024015997&hash=item591239d883:g:0xMAAOSwDLVbkDje
- Sparkfun website (AVR Tiny Programmer): https://www.sparkfun.com/products/11801
- Download LCD Assistant : http://en.radzio.dxp.pl/bitmap_converter/
- Tutorial using LCD Assistant to create bitmap: https://www.instructables.com/id/Display-Images-on-OLED-Screen-With-Arduino-ATtiny8/
- Download SSD1306 library: https://github.com/anothermist/LIBRARIES/blob/master/SSD1306_minimal/FSNKTHDIDWXMIF4.zip
- Arduino Core for ATTINY chips: https://github.com/SpenceKonde/ATTinyCore
- Authors email (for further information, files, etc.): tractionfan@aol.com

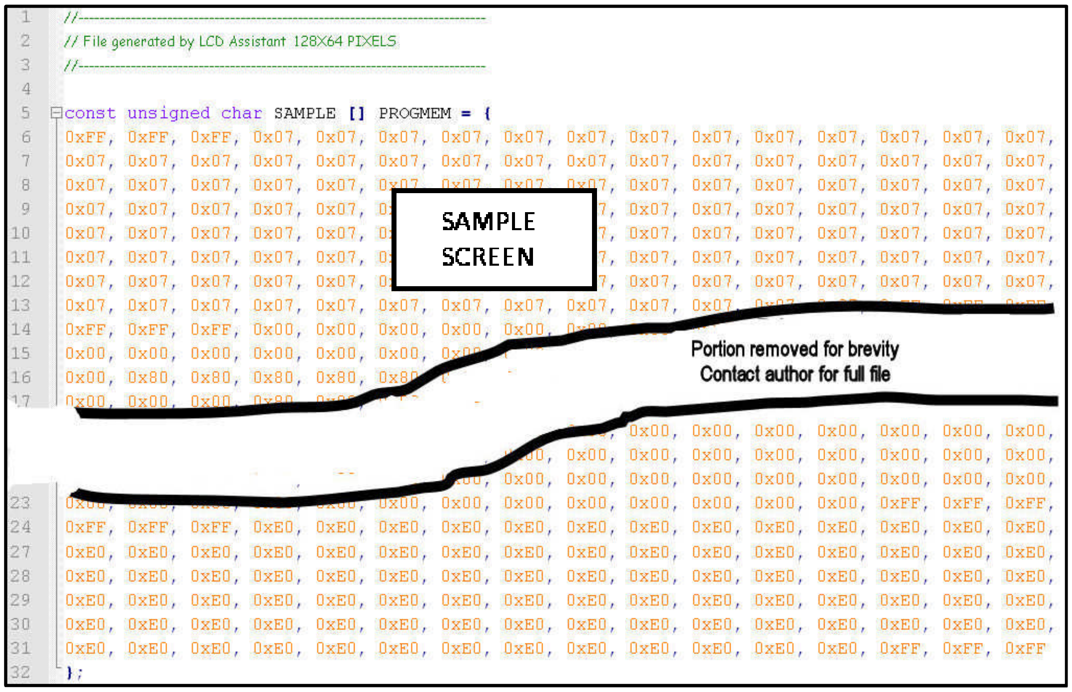
<u>Appendix C</u>

<u>Arduino sketch for the SAMPLE display:</u>



```
/*******************************************************************************
 *                      ATTINY85 - SSD1306 DISPLAY       F.MILLER 1/2019
 *******************************************************************************
 *                      SAMPLE ONE SCREEN DISPLAY
 *******************************************************************************
 * ATTINY code driving SSD1306 display using library SSD1306_minimal from
 * https://github.com/anothermist/LIBRARIES/tree/master/SSD1306_minimal
 * (Renamed library folder from 'FSNKTHDIDWXMIF4' to 'SSD1306_minimsl')
 * This sketch displays several full screen 128x64 images.  Note pixel hex code must be formated
 * to VERTICAL mode out of LCDassistant,  vs HORIZONTAL mode for Adafruit's ssd1306 library.
 * Drawn black pixels become white on display.  This sketch relies on the image bitmaps files
 * resident within the sketch code folder and 'includes' them in the code.
 * These are flipped images to accomodate mounted display orientation
 * NOTE: A maximum of 6 full 128x64 images can fit in the ATTINY85 memory.
 ******************************************************************************
 * SSD1306 Display SCL -> Attiny85 PB2 (Pin 7)
 * SSD1306 Display SDA -> Attiny85 PB0 (Pin 5)
*******************************************************************************/

#include "SSD1306_minimal.h"
#include <avr/pgmspace.h>
#include "SAMPLE.h"             //full 128x64 pixel screen bitmap image with Loco graphic

SSD1306_Mini oled;

void setup(){
  oled.init(0x3c);              //i2c Address of display
  oled.startScreen();
  oled.clear();
  oled.drawImage(SAMPLE,0,0,128,8 );         // (name, ulX, ulY, width, height/8)
}

void loop(){ }
```

<u>Bitmap for SAMPLE screen:</u>



```
//------------------------------------------------------------------------------
// File generated by LCD Assistant  128X64 PIXELS
//------------------------------------------------------------------------------

const unsigned char SAMPLE [] PROGMEM = {
0xFF, 0xFF, 0xFF, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07,
0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07,
0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07,
0x07, 0x07, 0x07, 0x07, 0x07, 0                       7, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07,
0x07, 0x07, 0x07, 0x07, 0x07, 0                       7, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07,
0x07, 0x07, 0x07, 0x07, 0x07, 0                       7, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07,
0x07, 0x07, 0x07, 0x07, 0x07, 0                       7, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07,
0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07,
0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x80, 0x80, 0x80, 0x80,
0x00, 0x00, 0x00,
                                                    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
                                              0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
                                        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
                              0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0,
0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0,
0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0,
0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0,
0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0,
0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xFF, 0xFF, 0xFF
};
```