# Sharp Memory LCD Shield User Guide



Written by John Leung Document version 1.1 12/7/2018

## Update

1) Add support for two new Memory LCD modules LS006B7DH03 (64\*64) and LS011B7DH03(160\*68)

## **Contents at a Glance**

• Chapter 1	Introduction	1	
• Chapter 2	Setting up the hardware		
	Memory LCD	2	
	• Select 5V v.s 3V LCD voltage	3	
	• Application Processor - ESP32 WiFi/BLE SoC	4	
	Application Processor - Arduino M0 PRO	5	
Chapter 3	Setting up the development environment	6	
	• If you are using Arduino M0 PRO	7	
	• If you are using ESP32	9	
Chapter 4	Describing how hardware works	11	
Chapter 5	Describing how firmware works		
• Chapter 6	Font Creation		
Chapter 7	Image Creation	26	
Chapter 8	An estimation on power consumption	30	
• Appendix A	Schematic diagram of Memory LCD Shield	37	
	PCB layout of Memory LCD Shield	39	
Appendix B	Schematic diagram of ESP32 Application Processor	40	
	PCB layout of ESP32 Application Processor	41	

#### Chapter 1 Introduction

Sharp's Memory LCD (https://www.sharpsma.com/products?sharpCategory=Memory%20LCD) is a lightweight display with 1-bit memory in every pixel allowing high-contrast, ultrathin and at the time same delivering a relatively high frame rate (20Hz max) at merely microWatt power consumption level. An evaluation kit compatible with 3.3V Arduino platforms has been built to facilitate testing and product development for the LCDs. At time of writing, five Memory LCD models LS027B7DH01 (2.7"), LS032B7DD02 (3.2"), LS044Q7DH01(4.4"), LS006B7DH03 (0.56"), and LS011B7DH03 (1.08") have been tested with firmware designed for two Arduino boards - Arduino M0 PRO and ESP32.





#### Chapter 2 Setting up the hardware

#### **Memory LCD**

Simply stack the shield on top of any 3.3V Arduino compatible board to start development. Bonus features include a high precision shunt amplifier INA226 for measuring power consumption on  $100\Omega \pm 0.1\%$  shunt resistor. Illustration of the board is shown below. Schematic and PCB layout of the shield can be found in Appendix A of this document.



Stack Memory LCD Shield on top of any 3.3V version Arduinocompatible evaluation kit to start development.



#### Select 5V v.s 3V LCD voltage

It is important to notice that there are two operation voltages for different Memory LCD models, 5V and 3V. Table below summaries the electrical characteristics of different models.

Model	VDDA MIN.	VDDA TYP.	VDDA MAX.
LS027B7DH01 (2.7")	4.8V	5.0V	5.5V
LS032B7DD02 (3.2")	4.8V	5.0V	5.5V
LS044Q7DH01(4.4")	4.8V	5.0V	5.5V
LS006B7DH03 (0.56")	2.7V	3.0V	3.3V
LS011B7DH03 (1.08")	2.7V	3.0V	3.3V

Compatibility is enabled by a jumper resistor to select between 5V and 3.3V as the VDDA (and VDD as well) supply. Extract below shows a part of the schematic. Resistor R10 is not soldered by default. Every evaluation kit ex-factory was soldered with a 0 Ohm resistor at R13 to source 5V output from TPS60140PWP DC-DC converter as the VDDA (and VDD). If you need to use 3V Memory LCD, please swap R13 and R10 which mean to unsolder R13 and use the same resistor at R10 to complete the bridge between 3.3V supply from Arduino host board to VDDA (and VDD) of Memory LCD.





#### Application Processor - ESP32 WiFi/BLE SoC

Application processor (AP in short) is the microcontroller or System-on-Chip (SoC) to draw every pixel on screen. We are doing the job with ESP32-WROOM-32 which is an off-the-shelf Wifi and Bluetooth Low Energy (BLE) SoC. By "googling" the keyword ESP32 there are myriad links and resources available. Official web site can be found at https://www.espressif.com/en/products/hardware/modules. The first module on list of the table is the model we have chosen.

O Home	Module	Description	Chip Embedded	Dimensions (mm)	Pins	Flash (MB)	PSRAM (MB)	Antenna	Development Board
Products Company Ecosystem		ESP32-WROOM-32 contains the ESP32 SoC, flash memory, high- precision discrete components, and a PCB antenna which provides outstanding RF performance in space- constrained applications.	ESP32- D0WDQ6	18x25.5x2.8	38	4	N/A	PCB antenna	ESP32-DevKitC

There are many reasons to opt for this device including its popularity, open-source, resourceful documentation, nicely supported SDK, low cost (~US\$4), and most importantly there is an ample SRAM of 520KB together with 4MB Flash miniaturized on this 18x25mm PCB.

Gearing with 520KB SRAM is critical for our applications because a frame buffer is implemented to hold graphical contents. There are three Memory LCD models from resolution 320\*240 (4.4") to 336\*530 (3.2"). If we are storing full contents in SRAM for 1-bit pixel depth, the requirement is tabulated in table below. A divisor of 8 in arithmetic is required because we are using 1-byte to hold 8 pixels in 1-bit color depth.

Memory LCD	Resolution	SRAM (byte)
LS027B7DH01	400*240	400*200/8 = 12,000
LS044Q7DH01	320*240	320*240/8 = 9,600
LS032B7DD02	336*536	336*536/8 = 22,512
LS006B7DH03	64*64	64*64/8 = 512
LS011B7DH03	160*68	160*68/8=1,360

As indicated a SRAM as high as 22KB is required and this memory is satisfied with 520KB SRAM of ESP32 SoC.

To fit an Arduino-UNO form factor, a PCB was designed with peripheral components including voltage regulator, USB-to-UART transceiver chip (CH340C), and an automatic binary file download circuitry for kick-starting development. Schematic and PCB layout of the ESP32 Application Processor is shown in Appendix B of this document.



#### **Application Processor - Arduino M0 PRO**

In addition to ESP32 we have also tested Arduino M0 PRO as the AP. This board has been chosen because it is readily available and there is a 32KB SRAM which is more than sufficient to support the frame buffer requirement explained in last section.

Getting started guide can be found under this hyperlink.

https://www.arduino.cc/en/Guide/ArduinoM0Pro



#### Chapter 3 Setting up the development environment

Download and install Arduino IDE version 1.8.1 or later.

Locate the library folder. This information is available from File $\rightarrow$ Preferences. In my case it is under C:\Users\John\Documents\Arduino.

Preferences	×			
Settings Network				
Sketchbook location:				
C: \Users\John\Documents\Arduino	Browse			
Editor language: System Default (requires restart of a	Arduino)			
Editor font size: 14				
Interface scale: 🔽 Automatic 100 🚾 % (requires restart of Arduino)				
Show verbose output during: 🖵 compilation 🖵 upload				
Compiler warnings: None				
✓ Display line numbers				
Enable Code Folding				
✓ Verify code after upload				
Use external editor				
Check for updates on startup				
✓ Update sketch files to new extension on save (.pde -> .ino)				
Save when verifying or uploading				

By this time you should have obtained a copy of the Memory LCD driver from us. Extract the library and manually copy the folders to C:\Users\John\Documents\Arduino. *Your path may be different with your user name.* There are two libraries required, one for Memory LCD and the second for INA226 as shown below with highlights.

Documents library	Arrang
Name ^	Date modified
鷆 Adafruit_INA219-master	1/17/2018 11:54 AM
퉬 Adafruit_NeoPixel	10/25/2017 6:04 AM
퉬 ArduinoDueHiFi-master	8/18/2016 10:43 AM
퉬 Arduino-INA226-master	7/31/2017 1:55 AM
퉬 ArduinoJson	4/17/2018 11:15 AM
퉬 AutoAnalogAudio-master	12/5/2016 8:04 AM
퉬 Blynk	10/25/2017 6:03 AM
Division BlynkESP8266_Lib	4/9/2017 8:23 PM
퉬 DirectIO-master	12/8/2016 1:07 PM
3 MemoryLCD	5/7/2018 6:03 PM
퉬 Ra8876_Lite	3/14/2018 6:22 PM

Restart Arduino IDE.

#### If you are using Arduino M0 PRO

M0 PRO is natively supported in Arduino IDE so the only step is to Select Arduino M0 or Arduino M0 Pro (Programming Port) from Tools-Board.

If Arduino M0/M0 Pro is not available, you will have to install it from Boards Manager.



Connect USB port of Arduino M0 (or debug port of Arduino M0 PRO) to your PC.

From Device Manager you will see an enumerated port for M0/M0 PRO like screen shot below.



Make sure it is the same port number you have selected under Tools→Port for binary code download.

Memory LCD shield is compatible with five LCD models. The "switch" to control is located at the header file with #define as below. Select one of them for your Memory LCD model. You may open this .h file with your favorite text editing program. A Windows **Notepad** will be sufficient. My personal preference is Notepad++. **Don't forget to save the file**.

C Mem	oryLCD.h ×	
та	#тистипе	Limage.n
20		
21	* @note	Define any model below and recompile
22		LS027B7DH01 = 2.7" Memory LCD with 400*240 pixels
23		LS032B7DD02 = 3.16" Memory LCD with 336*536 pixels
24		LS044Q7DH01 = 4.4" Memory LCD with 320*240 pixels
25		LS006B7DH03 = 0.56" Memory LCD with 64*64 pixels, 3V input voltage
26		LS011B7DH03 = 1.08" Memory LCD with 160*68, 3V input voltage(br>)
27		
28	//#define	LS027B7DH01
29	//#define	LS032B7DD02
30	#define	LS044Q7DH01
31	//#define	LS006B7DH03
32	//#define	LS011B7DH03
33		

Finally click Sketch→Upload.



After download successful you will see a message from the console window



#### If you are using ESP32

Install ESP32 Arduino Core from https://github.com/espressif/arduino-esp32.

The installation procedure is available in full details from the download link above.

ESP32 AP board is using a USB-UART bridge CH340C to link up with a PC. Download its driver from http://www.wch.cn/download/CH341SER\_ZIP.html and get it installed.

#### CH341SER.ZIP

5用范围	版本	上传时间	资料大小	
	3.4	2016-09-27	198KB	◆ 下载

After download and installation, connect the USB port to PC and make sure there is a new COM PORT enumerated in Device Manager. In my case it is COM55. It could be different in your environment.

Unplug it from PC and stack Memory LCD Shield on it.





Install Memory LCD to the Omron dual contact connector onboard  $\rightarrow$  close the zip lock. Reconnect USB cable.



Restart Arduino IDE. Browse through Arduino Examples from File→Examples→MemoryLCD→HelloWorld.

There are four examples at time of writing this document.

Make sure you have uncomment the LCD model from MemoryLCD.h header file for what you are using.

<pre>19 #INCLOVE LIMAGE.N 20 /** 21  * @note Define any model below and re</pre>	
	compile
22 * LS027B7DH01 = 2.7" Memory L	CD with 400*240 pixels
23 * LS032B7DD02 = 3.16" Memory	LCD with 336*536 pixels
24 * LS044Q7DH01 = 4.4" Memory L	CD with 320*240 pixels
25 * LS006B7DH03 = 0.56" Memory	LCD with 64*64 pixels, 3V input volt
26 * LS011B7DH03 = 1.08" Memory	LCD with 160*68, 3V input voltage(br
30 #define LS044Q7DH01	
31 //#define LS006B7DH03	
32 //#define LS011B7DH03	

🔯 HelloWorld   Arduino 1.8.:	1		
File Edit Sketch Tools Hel	p		
New Ctrl+N			
Open Ctrl+O			
Open Recent	ded_MT_Bold55h.c	Consolas24h.	c SimHei_35h.c
Sketchbook 🕨			
Examples 🕨	A		with Chann Manage
Close Ctrl+W	Ethernet	• merr	t with sharp memor
Save Ctrl+S	Firmata	• 11 1	functions to draw
Save As Ctrl+Shift+S	LiquidCrystal	eith	ner of the two tac
Page Setup Ctrl+Shift+P	Steener	pryl	.CD.h by uncomment
Print Ctrl+P	Temboo	el.	
Desferences ChildCommo	TET		
Preferences Cur+Comma	WiFi	•	
Quit Ctrl+Q			
10 *	Examples for ESP32 De	ev Module	
11 * @note Programm	ArdunoUTA		
12 * Doto:	EERDOM		
12 4/	ESP32		
10 */	ESP32 BLE Arduino		
14 #include MemoryLU	ESPmDNS	•	
15	HTTPClient	•	
16 #if defined (ESP32	Preferences	•	
17 const int SW3 = 35	SD(esp32)	•	
18 const int SW2 = 34	SD_MMC	•	
19 const int BUZZ = 2	SimpleBLE	•	
20 #elif defined (_VA	SPIFFS	•	
21 const int SW3 = 6:	Update	•	
22 const int SW2 = 5:	WiFi	•	
23 compt int BU77 = 5	WiFiClientSecure	•	
20 const int bozz = 1	Examples from Custom	Libraries	
24 #endii	Adafruit INA219	•	
25	Adafruit NeoPixel	•	
26 extern const BFC_F	Arduino-INA226-maste	r ▶ Ld5t	bh;
27 extern const BFC_F	ArduinoDueHiFi-master	•	
28 extern const tImag	ArduinoJson	•	
29 extern const tImag	Blynk	•	
30	Directio-master		in our l
31 ///@note Font: Sim	Page 76 Lite	B	oodPressure_GUI
32 const uint16_t hel	SdEat		rstPivel , C
33 ///@note Font: Sim	Time	H	elloWorld
34 const uint16 t nol	Tiny COM	- TD	
05	ugfx-arduino-master	•	
	INCOMPATIBLE	•	
	$\nabla$		

For ESP32 AP, from Tools→Boards Manager, select ESP32 Dev Module. Please also make sure parameters are selected according to screen shot below.

💿 H	elloWorld   Ard	uino 1.8.1				
File	Edit Sketch T	Fools Help				
Ø	0 🖬 E	Auto Format Archive Sketch	Ctrl+T			
н	elloWorld	Fix Encoding & Reload		s24h.c	SimHei_35h.c	cat_400x246.c
1	/%%	Serial Monitor Serial Plotter	Ctrl+Shift+M Ctrl+Shift+L			
3	* @brief -	WiFi101 Firmware Updater		API fun	ith Sharp Memor ctions to draw	y LCD. pixel, line, rect
4 5 6	* *	Blynk: Check for updates Blynk: Example Builder Blynk: Run USB script		either moryLCD. del.	of the two tac h by uncomment	t switches SW2 an the model to tes
7 8 9	* * *	Board: "ESP32 Dev Module" Flash Mode: "DIO" Flash Size: "4MB (32Mb)"		WifInfo Arduino 4D Svs	▲	
10 11 12	* * @note *	Flash Frequency: "80MHz" Upload Speed: "115200" Core Debug Level: "None" Port: "COM18"		Digistu Intel G Arduine	mp Oak urie (32-bit) Boards o/Genuino 101	
13	*/	Get Board Info		E9P92	Ardeino Dev Module	
14 15	#include "	Programmer: "AVRISP mkII" Burn Bootloader	•	ESP32 SparkF	Pico Kit un ESP32 Thing	
16	#if defined	(ESP32)		u-blox	NINA-W10 series (ES	P32)

The last step is to click Upload from Sketch→Upload



From Arduino's message window please make program compile OK with a Done uploading message.



Now if you are lucky, some texts will be visible. Free feel to click on any of the keys to browse through the demo.



#### Chapter 4 Describing how hardware works

Full schematics of Memory LCD Shield and ESP32 Application Processor are printed on the Appendix section of this document. A simplified block diagram showing different functional blocks is summarized below. ESP32 is shown in this block diagram for illustration purpose.



#### **Describing how firmware works**

Two data update modes are available, namely the 1-line mode and multiple-lines mode. Timing diagrams are available from LCD's datasheet with extracts shown below.

6-5-1 Data update mode (1 line) Updates data of only one specified line. (M0="H", M2="L") scs #SCS Mode select period (3ck+5ckD MY) Gate line address select period ! (8ok) Date transferperiod r16 ck) Datewrite period (400ok) M0: Mode flag. Set for "H". Data update mode (Memory internal data update) When "L", display mode (maintain memory internal data). M1: Frame inversion flag. When "H", outputs VCOM="H", and when "L", outputs VCOM="L". When EXTMODE="H", it can be "H" or "L". M2: All clear flag. Refer to 6-5-4) All Clear Mode to execute clear. DUMMY DATA: Dummy data. It can be "H" or "L" ("L" is recommended.) 6-5-2 Data Update Mode (Multiple Lines)

Updates arbitrary multiple lines data. (M0="H", M2="L")



In the firmware these update modes are implemented by two local functions:

static void GFXDisplayUpdateLine(uint16\_t line, uint8\_t \*buf); static void GFXDisplayUpdateBlock(uint16\_t start\_line, uint16\_t end\_line, uint8\_t \*buf);

Frame buffer is declared as a double array to store all pixels of the LCD as below:

```
uint8_t frameBuffer[GFX_FB_CANVAS_H][GFX_FB_CANVAS_W];
//GFX_FB_CANVAS_H = Vertical resolution of the LCD
//GFX_FB_CANVAS_W = Horizontal resolution of the LCD /8
```

With a slightly better camera and a macro lens, we were able to visualize every pixel on a 4.4" Memory LCD as illustrated below . The outline in red color is a label for the first 8 pixels.



On system startup, all bytes in frameBuffer[][] array are initialized to 0xFF meaning all pixels set to white color. This is the first element frameBuffer[0][0] to store color content of the first 8 pixels in horizontal direction.

Now, suppose we need to set the odd pixels to black for the first 8 pixels (1,3,5,7), we may call the API function GFXDisplayPutPixel(x,y,color) with code snippet show below:

What's happening in the for-loop above is that, every time GFXDisplayPutPixel() is called, say GFXDisplayPutPixel(0,0,BLACK); it is the bit position at (0,0) set BLACK with the rest bit positions unchanged. To describe it in full details, memory content of the frame buffer array tabulated against each function call is shown on next page.

Function call	Frame buffer array for 4.4" LCD as an example. An underline at the bit-position shows the pixel to write.
GFXDisplayPutPixel(0,0,BLACK)	frameBuffer[0,0] = 0b0111 1111;
	frameBuffer[n,0] = 0b1111 1111; n=1-39
GFXDisplayPutPixel(1,0,WHITE)	frameBuffer[0,0] = 0b0 <u>1</u> 11 1111;
	frameBuffer[n,0] = 0b1111 1111; n=1-39
GFXDisplayPutPixel(2,0,BLACK)	frameBuffer[0,0] = 0b01 <u>0</u> 1 1111;
	frameBuffer[n,0] = 0b1111 1111; n=1-39
GFXDisplayPutPixel(3,0,WHITE)	frameBuffer[0,0] = 0b010 <u>1</u> 1111;
	frameBuffer[n,0] = 0b1111 1111; n=1-39
GFXDisplayPutPixel(4,0,BLACK)	frameBuffer[0,0] = 0b0101 <u>0</u> 111;
	frameBuffer[n,0] = 0b1111 1111; n=1-39
GFXDisplayPutPixel(5,0,WHITE)	frameBuffer[0,0] = 0b0101 0 <u>1</u> 11;
	frameBuffer[n,0] = 0b1111 1111; n=1-39
GFXDisplayPutPixel(6,0,BLACK)	frameBuffer[0,0] = 0b0101 01 <u>0</u> 1;
	frameBuffer[n,0] = 0b1111 1111; n=1-39
GFXDisplayPutPixel(7,0,WHITE)	frameBuffer[0,0] = 0b0101 010 <u>1</u> ;
	frameBuffer[n,0] = 0b1111 1111; n=1-39

In essence, what the function GFXDisplayPutPixel(x,y,color) doing is to set /clear the required bit position at (x,y) in frame buffer while keeping the rest bit positions unchanged. After bit set/clear is finished, the whole line will be updated by the local function GFXDisplayUpdateLine() for the complete horizontal line.

Interested readers may take a look at the source code for GFXDisplayPutPixel() listed below:

```
void GFXDisplayPutPixel(uint16_t x, uint16_t y, COLOR color)
{
     GFXDisplayPutPixel_FB(x, y, color);
     GFXDisplayUpdateLine(y+1, (uint8_t *)&frameBuffer[y]);
}
```

After running this code snippet the LCD is displaying something this:



To extend this concept to 2D, we may simply update a rectangular area with GFXDisplayUpdateBlock() to span it over the required top and bottom horizontal margin.

This implementation leads to a faster frame rate. Graphical interface illustrated in the Arduino Sketch BloodPressure GU.ino shows a counting blood pressure reading at the left with a stood still icon of an up-arrow at the right. Feel free to open this sketch and change the delay constant in loop() from delay(50) to delay(1), or removing it to get an impression on how fast it can go.

Another unique features of Memory LCD is that partial update is allowed as long as it spans a complete horizontal region. This means only the block occupying the font height of the SYS. pressure below is changed, whereas the top and bottom regions need not to be updated.



A coin has two sides though. At the expense of a high frame rate with higher flexibility in GUI design, the down side of this approach is that we will need more expensive microcontroller with more SRAM memory to accommodate the frame buffer.

With better GUI planning there is a compromise. Consider an example below.

If we knew the top portion is not to be changed, we may copy graphical data from flash during system startup for the top region which will not be changed in the whole course of device usage. The lower part for numbers and date-time that need continuous update may be allocated a smaller frame buffer just enough for the area. This design approach leads to a wider choice of microcontrollers with fewer SRAM memory.



right.

#### Chapter 6 Font creation

In this section we are going to introduce a commercial bitmap font creator tool (BitFontCreator) developed by a Beijing-based company called *Iseatech Software* (http://www.iseasoft.com/company/about.html). BitFontCreator converts any font from PC to C files for embedded projects.

There are three versions available namely BitFontCreator Latin, Pro, and Grayscale. The writer is using the Grayscale version but Latin version for 1-bpp font will be good enough for Memory LCD in Black/White color. Demo version is available from Iseasoft with a strikethrough on any character created. This limitation will be removed from formal version.

#### Procedures

Launch BFC, select Monochome, 1-bpp as the Font Creation Option. Click OK.

There are various encoding options available for Unicode, BIG5, and ASCII of course. If Unicode is selected, it will be possible to create C array from 16-bit characters for embedded projects. For now select 8 bit ASCII + ISO8859.

G Untitled - BitFont	Creator Grayscale		
File Edit Font Opti	ons Help		
🗋 • 🗃 🖬 🖬	🖹 1 🕂 2 a-z 3 🗐 4 🖾 3 🏟 🗄	А 🗧 🖂	
	Character Editor		Characters Table
	Font Creation Options Type of font to create Monochrome, 1 bpp Antialiased, 2 bpp Antialiased, 4 bpp	OK Cancel	
Grid Zoom	C 16 bit UNICODE 8 bit ASCII + ISO8859 C 8/16 bit SHIFT_JIS	C 8/16 bit BIG_5 C 8/16 bit GBK C 8/16 bit Hangul	

Select the font to use from your PC. Click OK.



A Character Table is updated to show the default character range to use. To save flash memory it is possible to customize the range say, from ASCII code 0x0020 to 0x007E for ASCII characters from <space> to <~>.

0020	0021	0022	0023	0024	0025	0026	0027	0028	0029	002A	002B	002C	002D	002E	002F	0030	0031	0032	0033	0034	0035	0036	0037	0
	!		#	\$	%	&	•	(	)	*	+	,	-		1	0	1	2	3	4	5	6	7	
0039	003A	003B	003C	003D	003E	003F	0040	0041	0042	0043	0044	0045	0046	0047	0048	0049	004A	004B	004C	004D	004E	004F	0050	0
9	:	;	<	=	>	3	0	А	в	с	D	Е	F	G	н	I	J	К	L	м	Ν	о	Р	
0052	0053	0054	0055	0056	0057	0058	0059	005A	005B	005C	005D	005E	005F	0060	0061	0062	0063	0064	0065	0066	0067	0068	0069	0
R	s	Т	U	V	W	Х	Y	Z	[	١.	]	۸	_	•	а	b	с	d	е	f	g	h	i	
006B	006C	006D	006E	006F	0070	0071	0072	0073	0074	0075	0076	0077	0078	0079	007A	007B	007C	007D	007E	007F	00A0	00A1	00A2	0
k	1	m	n	o	р	q	r	s	t	u	v	W	х	У	z	{		}	~			i	¢	
00A4	00A5	00A6	00A7	00A8	00A9	00AA	00AB	00AC	00AD	00AE	00AF	00B0	00B1	00B2	00B3	00B4	00B5	00B6	00B7	00B8	00B9	00BA	00BB	0
¤	¥	ł	§		©	a	~	-	-	®	-	•	±	2	3	•	μ	J	•		1	₽	»	
00BD	00BE	00BF	00C0	00C1	00C2	00C3	00C4	00C5	00C6	00C7	00C8	00C9	00CA	00CB	0000	00CD	00CE	00CF	00D0	00D1	00D2	00D3	00D4	0
1/2	%	ė	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ϊ	Ð	Ñ	ò	Ó	Ô	
00D6	00D7	00D8	00D9	00DA	00DB	00DC	00DD	00DE	00DF	00E0	00E1	00E2	00E3	00E4	00E5	00E6	00E7	00E8	00E9	00EA	00EB	00EC	00ED	00
Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	
00EF	00F0	00F1	00F2	00F3	00F4	00F5	00F6	00F7	00F8	00F9	00FA	00FB	00FC	00FD	00FE	00FF								
4	ă	ñ	à	ó	â	ã	ä	÷.	đ	h.	ú	û	ü	ú	h	Ü								

Click Font  $\rightarrow$  Edit Characters Table (or Ctrl + T). Click Disable All and followed by Enable Range. Fill in 0x0020 for the First character and 0x007E for the Last character. Click OK.

U	V	W	Х	Y	Ζ	[	$\mathbf{N}$	]	۸	_
e	f	g	h	i	j	k	1	m	n	0
u	v	Selec	t range	e of cha	racter	5			×	
	+	First:	: 1	Dx 002	0			ок	] [	2
•	-	laet		r⊷ 007	F		0	`ancel		Ÿ
¥	1	- LGol		ux [	-				┙Г	-
μ	ŋ	•		1	Q	»	1/4	1/2	3/4	ż
Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
~				<u>&gt;.</u>		<u>^</u>				-

Now the Character Table will be updated with the range you have selected. A smaller character range just enough for application will create a smaller C array to save precious Flash space for microcontroller.

tors rut	Jie																			
0021	0022	0023	0024	0025	0026	0027	0028	0029	002A	002B	002C	002D	002E	002F	0030	0031	0032	0033	0034	0035
l		#	\$	%	&		(	)	*	+	ړ	-		1	0	1	2	3	4	5
0037	0038	0039	003A	003B	003C	003D	003E	003F	0040	0041	0042	0043	0044	0045	0046	0047	0048	0049	004A	004B
7	8	9	:	;	<	=	>	?	0	А	в	с	D	Е	F	G	н	Ι	J	к
004D	004E	004F	0050	0051	0052	0053	0054	0055	0056	0057	0058	0059	005A	005B	005C	005D	005E	005F	0060	0061
м	Ν	0	Р	Q	R	s	Т	υ	v	W	х	Y	z	[	١	]	۸	_	•	а
0063	0064	0065	0066	0067	0068	0069	006A	006B	006C	006D	006E	006F	0070	0071	0072	0073	0074	0075	0076	0077
с	d	e	f	g	h	i	j	k	1	m	n	о	р	q	r	s	t	u	v	W
0079	007A	007B	007C	007D	007E															
У	z	{		}	~															
	0021 1 0037 7 004D M 0063 C 0079 y	0021         0022           I         "           0037         0038           7         8           004D         004E           M         N           0003         0064           C         C           007         007A           y         Z	0021         0022         0023           I         "         #           0037         0038         0039           7         8         9           0044         0044         0044           M         N         O           00083         0044         0045           C         d         e           0079         007A         007B           y         Z         {	0021         0022         0023         0024           I         "         #         \$           0037         0038         0039         003A           7         8         9         :           0040         004E         004F         0050           M         N         O         P           0068         0064         0065         0066           C         d         e         f           0079         007A         007B         007C           y         Z         {         j         j	0021         0022         0023         0024         0025           I         '''         III         III         III         III           0037         0038         0039         003A         0038           7         8         9         III         IIII           0040         004F         004F         0050         0051           M         N         O         P         Q           0068         0064         0065         0066         0067           C         d         e         f         g           0074         007B         007C         007D           y         Z         {         I         j	0021         0022         0023         0024         0025         0026           I         '''         III         III         III         III         IIII         IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII	0021         0022         0023         0024         0025         0026         0027           !         '''         ##         \$\$         %         &         ''           0037         0038         0039         0034         003B         003C         003D           7         8         9         :         ;         ;         <         =           0040         004F         0050         0051         0052         0053           M         N         O         P         Q         R         S           0068         0064         0050         0067         0068         0069           c         dd         e         f         gg         h         i           0079         007A         007B         007C         007D         007E           y         z         {         j         ;         ;         .	0021         0022         0023         0024         0025         0026         0027         0028           I         III         III         IIII         IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII	0021         0022         0023         0024         0025         0026         0027         0028         0029           I         '''         ##         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$         \$\$ <th><math display="block">\begin{array}{c c c c c c c c c c c c c c c c c c c </math></th> <th>0021         0022         0023         0024         0025         0026         0027         0028         0029         0024         0028           I         ''         ##         \$\$         %         &amp;         ''         (         )         **         +           0037         0038         0039         0034         0038         0030         0030         0030         0030         0038         0037         0038         0037         0038         0037         0038         0037         0038         0037         0038         0037         0038         0038         0030         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038</th> <th><math display="block">\begin{array}{c c c c c c c c c c c c c c c c c c c </math></th> <th><math display="block">\begin{array}{c c c c c c c c c c c c c c c c c c c </math></th> <th>0021         0022         0023         0024         0025         0026         0027         0028         0029         0024         0026         0026         0026           I         "         #         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$</th> <th><math display="block">\begin{array}{c c c c c c c c c c c c c c c c c c c </math></th> <th>0022       0023       0024       0025       0026       0027       0028       0029       0024       0026       0020       0026       0020       0026       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040</th> <th>0021       0023       0024       0025       0026       0027       0028       0029       0026       0026       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040</th> <th>0021       0023       0024       0025       0026       0027       0028       0029       0026       0020       0026       0020       0026       0020       0026       0020       0021       0020       0031       0031       0033         1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1</th> <th>0022       0023       0024       0025       0026       0027       0028       0029       0026       0026       0020       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0040       0046       0040       0046       0046       0046       0046       0046       0046       0046       0046       0046       0046       0046       0046       0046       0046       0046       0066       0066       0066       0066       0066       0066       0066       0066       0066       0066       0066       0066</th> <th>0022       0023       0024       0025       0026       0027       0029       0024       0026       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040</th>	$\begin{array}{c c c c c c c c c c c c c c c c c c c $	0021         0022         0023         0024         0025         0026         0027         0028         0029         0024         0028           I         ''         ##         \$\$         %         &         ''         (         )         **         +           0037         0038         0039         0034         0038         0030         0030         0030         0030         0038         0037         0038         0037         0038         0037         0038         0037         0038         0037         0038         0037         0038         0038         0030         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038         0038	$\begin{array}{c c c c c c c c c c c c c c c c c c c $	$\begin{array}{c c c c c c c c c c c c c c c c c c c $	0021         0022         0023         0024         0025         0026         0027         0028         0029         0024         0026         0026         0026           I         "         #         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$         \$	$\begin{array}{c c c c c c c c c c c c c c c c c c c $	0022       0023       0024       0025       0026       0027       0028       0029       0024       0026       0020       0026       0020       0026       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0030       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040	0021       0023       0024       0025       0026       0027       0028       0029       0026       0026       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040	0021       0023       0024       0025       0026       0027       0028       0029       0026       0020       0026       0020       0026       0020       0026       0020       0021       0020       0031       0031       0033         1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1       1	0022       0023       0024       0025       0026       0027       0028       0029       0026       0026       0020       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0026       0040       0046       0040       0046       0046       0046       0046       0046       0046       0046       0046       0046       0046       0046       0046       0046       0046       0046       0066       0066       0066       0066       0066       0066       0066       0066       0066       0066       0066       0066	0022       0023       0024       0025       0026       0027       0029       0024       0026       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0020       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040       0040

Click on Data Format button from the left panel, make sure the options are the same as screen shot below because the firmware has been designed for these options for data alignment.



Click OK to continue.

Now you may choose to save the project for future use. Using the default filename is OK here. Click Save. The format in \*.foc is a proprietary format recognized only by BitFontCreator Suite.

It is possible to application few modifications such as margin trimming, adding or removing some pixels, etc. To suit our application better the top margin has been trimmed by 3 pixels with feature from Font  $\rightarrow$  Change Font Height.

G Consolas24h - BitFo File Edit Font Option Change Font Height Change Font Width Change Font Ascent Resize Font	ntCreator s Help Ctrl+H Ctrl+W	Graysca IIII (w:13*)	le 4.5 4 1:18) - (	⁵ 🚮 0041 </th <th><b>₽</b> A&gt;</th> <th>⊿⊾</th> <th>4</th> <th></th> <th></th> <th></th> <th></th> <th></th> <th></th> <th></th> <th></th>	<b>₽</b> A>	⊿⊾	4								
Change to Fixed-Width Font Change to Variable-Width Font						Cha	nge	Font	: Heigh	t					×
Remove Blank Columns Center All Characters Horizontally Mirror All Characters	, ,					Fo	nt He	eight	24	pixels		C		•-14	
Edit Characters Table Font Information	Ctrl+T						Red (	duce r Clip top	p row	ignt 3	-	Add top	ont He	light 0	*
							Clip ł	bottom	n row	0	• •	Add botton	n row	0	* *
This feature will no	t shrir	ik the	for	nt							Help	Cancel		0	K

Finally, export a C file from drop down menu under

File  $\rightarrow$  Export  $\rightarrow$  General C file (\*.C).



Save the C file to any location, preferably at the same location of the Arduino project folder. It is under \BloodPressure\_GUI folder in our case.

G Save As	×
Save in: 🔑 BloodPressure_GUI 💿 🌀 🏂 📂 📰 🗸	
No items match your search.	
File name: Consolas24h Save	
Save as type: General C File (*.c) Cancel	

Similar procedures can be performed with other fonts with project folder look like this:

BloodPressure_GUI			
🗿 🗇 🕖 🗕 Docume	ents + Arduino + libraries + MemoryLCD + examples + BloodPressure_GUI	👻 🛃 Search Blood	iPressure_GUI 🛛 🙋
Organize 👻 🍳 Open	Share with 🔻 E-mail Burn New folder		8= • 🔳 🔞
Favorites	Documents library BloodPressure_GUI	Arrang	e by: Folder 🔻
Downloads	Name *	Date modified	Туре
Google Drive	Arial Rounded MT Bold55h	6/7/2018 5:30 PM	C Source File
Stopbox	Arial_Rounded_MT_Bold55h	6/7/2018 6:04 PM	BitFontCreator Docu
-	arrowDown_89x48	6/11/2018 4:05 PM	C Source File
Libraries	arrowUp_89x48	6/11/2018 2:46 PM	Bitmap image
Documents     Music	arrowUp_89x48	6/11/2018 3:49 PM	C Source File
Pictures	🙇 batteries	6/11/2018 3:04 PM	Bitmap image
Videos	attery_46x26	6/11/2018 3:08 PM	Bitmap image
_	battery_46x26	6/11/2018 3:49 PM	C Source File
P Computer	BloodPressure_GUI	6/14/2018 3:56 PM	Arduino file
Local Disk (C:)	Consolas24h	6/7/2018 1:54 PM	C Source File
Local Disk (D:)	E Consolas24h	6/7/2018 2:21 PM	BitFontCreator Docu
Cocal Disk (F.)	IoT_message	6/13/2018 12:45 PM	Bitmap image
年 Network	IoT_message	6/13/2018 12:46 PM	C Source File
	S pulseRate_icon	6/7/2018 6:28 PM	Bitmap image
	pulseRate_icon	6/11/2018 3:49 PM	C Source File
	SimHei_35h	6/12/2018 1:14 PM	C Source File

Only the C files and Arduino .ino file are useful

for programming. Those \*.foc files serve as references only.

#### Supporting Unicode

A similar procedure can be used to support Unicode (http://www.unicode.org/standard/standard.html).

Procedures in this section describe how BitFontCreator is used to convert simple greeting messages in Chinese and Japanese to C files for the Arduino Sketches BloodPressure\_GUI.ino and HelloWorld.ino.

Under File→New Font→Import An Existing System Font, select Monochrome, 1bpp with Encoding set 16 bit UNICODE. Click OK.





Pick any of the system Font with Unicode support, in our case the SimHei of Bold 26 selected.

Click OK.



It may take a while for your PC to import the map into BitFontCreator. After successful import the whole character table will be available. Browse it through the end you will see some characters in Japanese and Chinese.



Although it is possible to include the whole character range from 0x0020 to 0xFFE5, another consideration is that flash memory of microcontroller will be eaten up by the static font data if we are doing so. It is more practical to include only the characters we need to save our precious microcontroller flash area.

Bring up the Font→Edit Characters Table.



Click Disable All followed by Enable Range. Input the range to use, in our case only few characters are required for greeting in Japanese and Chinese. They are Unicode

0x3053, 0x3061, 0x306B, 0x306F, 0x3093, 0x4F60, & 0x5970.



After repeating the selection procedure for individual characters the Characters Table will be filled up with just what we need.

G 黒体35h - BitFontCreator Grayscale 4.5	
File Edit Font Options Help	
🗅 - 🚔 🖶 🐚 🖓 1 🔏 2a-z 3 🛄 4 🐯 5 🏠 🖪 🗛 🛁 🖽	
Character Editor (w:37 * h:35) - 3053 <乙 >	Characters Table
	3005         300F         3083         4F00         697D           こ         ち         に         は         ん         你         好
Grid Zoom	
Character Bitmap Data         Refresh           C Pure data         /* character = "C. "/           ● Detail info         /* character = "C. "/           /* character = "C. "/         /*           ● Detail info         /*           /* character = "C. "/         /*           ● Detail info         /*           /* character = "C. "/         /*           ● betail info         /*           /* C Language         /*           © ASM         000, b00, b00, b00, b00, b00, b00, b00,	
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,	Edit Table Help
For Help, press F1	Bpp: 1 Encoding: Unicode Monospaced font: (w:37 * h:35)

Bring up the Bitmap Data Format window to make sure settings as below. Click OK.



Now, click General C file (\*.c) from

#### File→Export→General C file (\*.c)...



Save the file, given the name as SimHei\_35h.c in our case.

To use the font created we need to "*let Arduino know*" that we have such data in our system. Now declare its existence in source code. Snippet shown below.



Similarly for fontArial\_Rounded\_MT\_Bold55h and fontConsolas24h the pattern for declaration is the same.

After announcing their existence the last step is to call the relevant functions to print the texts. Its usage can be found in the source code BloodPressure\_GUI.ino and HelloWorld.ino from the example folder.

🥯 Blo	oodPressure_GUI   Arduino 1.8.1
File	Edit Sketch Tools Help
Blo	oodPressure_GUI Arial_Rounded_MT_Bold55h.c Consolas24h.c arrowDown_89x48.c arrowUp_89x48.c b
85	
86	<pre>#if defined (ESP32)</pre>
87	///@note Ref: https://github.com/espressif/arduino-esp32/blob/a4305284d085caeddd1190d141710fb6f1c6
88	<pre>ledcSetup(0, 1000, 8); //channel 0, freq=1000, resolution_bits = 8</pre>
89	#endif
90	//display Hello in Chinese and Japanese
91	<pre>uint16_t w = GFXDisplayGetWStringWidth(&amp;fontSimHei_35h, hello_chinese);</pre>
92	<pre>uint16_t h = GFXDisplayGetFontHeight(&amp;fontSimHei_35h);</pre>
93	
94	GFXDisplayPutWString( (GFXDisplayGetLCDWidth()-w)/2,
95	(GFXDisplayGetLCDHeight()-h)/3,
96	&fontSimHei_35h, hello_chinese, BLACK, WHITE);
97	
98	<pre>w = GFXDisplayGetWStringWidth(&amp;fontSimHei_35h, hello_japanese);</pre>
99	GFXDisplayPutWString((GFXDisplayGetLCDWidth()-w)/2,
100	(GFXDisplayGetLCDHeight()-h)/3 + h,
101	&fontSimHei_35h, hello_japanese, BLACK, WHITE);
102	
103	<pre>w = GFXDisplayGetStringWidth( &amp;fontConsolas24h, "Press any key to continue");</pre>
104	GFXDisplayPutString((GFXDisplayGetLCDWidth()-w)/2,
105	(GFXDisplayGetLCDHeight()-h)/3 + 2*h,
106	&fontConsolas24h, "Press any key to continue", WHITE, BLACK);
107	

#### Chapter 7 Image creation

In this section we are going to show you a free utility to create image and font source files for embedded projects, namely LCD Image Converter. This tool's home page: <a href="http://www.riuson.com/lcd-image-converter">http://www.riuson.com/lcd-image-converter</a>.

From time to time we need to display some icons such as battery level, an arrow for navigation, etc.

LCD Image Converter is an open source program that can be downloaded for free whereas BitFontCreator described in last chapter is a commercial program at an affordable cost. The purpose of using different programs to convert graphical assets (fonts and bitmaps) is to show flexibility.

There is no restriction on which tool to use as long as we are drawing pixels on LCD correctly from arrays, no matter they have been converted by a commercial program or shareware.

Here we need to convert 3 icons to C arrays for our embedded project. They can be found in the project folder of BloodPressure\_GUI.ino







Start working with File  $\rightarrow$  Open. Browse to the image to convert.



From Options→Conversion, make sure the Main Scan Direction and Line Scan Direction have been set to Top to Bottom and Forward, respectively.

Options  Preset: Monochrome  Prepare Matrix Re	▼ _:
Scanning Preprocess	ing
Top to Bottom	/* * top to bottom * forward */
Forward  Bands	<pre>for (var y = 0; y &lt; image.height; y++) {     for (var x = 0; x &lt; image.width; x++) {         image.addPoint(x, y);     }</pre>
1 px	}

Under Image tab, select **Split to rows** checkbox with **Block Size** set to 8 bit, and Byte order set to **Little-Endian**.

🤞 Options		
Preset: Monochrome		Save As R
Prepare Matrix Re	ordering Image Font Te	mplates
Common	-Data	Preview
Split to rows	Prefix:	Prefix:
Trailing bits:	0x	//
Block size:		
8 bit 💌	, Delimiter:	Delimiter:
RLE Compression	,	
from 2 block(s)		Level's replacement:
Byte order:		
Little-Endian		
C Big-Endian		•
Show Preview		

You may click the **Show Preview** button to preview the C array. Click **OK** and **Yes** to save change.

Preview						
Preview image: default						
Scale: 2						
	0xff,	0xff,	Oxff,	0xff,	Oxff,	0xfc,
	0xff,	0xff,	Oxff,	0xff,	Oxff,	Oxfc,
	0xfc,	0x00,	0x00,	0x00,	0ж00,	0x0c,
	0xfc,	0x00,	0x00,	0x00,	0x00,	0x0c,
	0xfc,	0x00,	0x00,	0x00,	0ж00,	0x0c,
	0xfc,	0x7f,	0xff,	0xff,	0xff,	0x8c,
	0xf8,	0x7f,	Oxff,	Oxff,	Oxff,	0x8c,
	0x00,	0x7f,	0xff,	0x80,	0x01,	0x8c,
	0x00,	0x7f,	Oxff,	0x80,	0x01,	0x8c,
	0x00,	0x7f,	0xff,	0x80,	0x01,	0x8c,
	0x00,	0x7f,	Oxff,	0x80,	0x01,	0x8c,

Now click Convert under File→Convert... (Ctrl+P) to export the \*.c file to the root directory of your project.



By following the same procedures two files arrowUp\_89x48.c and pulseRate\_icon.c were created. Screen shot of the sketch project folder is shown below for reference.

ies • Documents • Arduino • examples • MemoryLCD • BloodPressure_GUI							
th 🔻	Burn New folder						
	Documents library BloodPressure_GUI						
_	Name ^						
	Arial_Rounded_MT_Bold55h Arial_Rounded_MT_Bold55h arrowDup_89x48 arrowUp_89x48 arrowUp_89x48 batteries battery_46x26 battery_46x26 battery_46x26 cosolas24h Consolas24h pulseRate_icon pulseRate_icon pulseRate_icon pulseRate_icon						

Please notice that the C file arrowDown\_89x48.c was converted with the up arrow image flipped in LCD Image Converter with keyword *arrowUp\_89x48* replaced by *arrowDown\_89x48* manually in a text editor.

Launch Arduino IDE and open the sketch BloodPressure\_GUI.ino, from the IDE all C files are show in different tabs.

There is one last step. Complete the C files created with a new header #include <tImage.h> for everyone of them.

🕺 BloodPressure_GUI - battery_46x26.c   Arduino 1.8.1								
File Edit Sketch Tools	Help							
BloodPressure_GUI	Arial_Rounded_MT_Bold55h.c	Consolas24h.c	arrowDown_89x48.c	arrowUp_89x48.c	battery_46x26.c	pulseRate_icon.c		
25 uint16_t he	ight;							
26 uint8_t dataSize;								
27 } tImage;								
28 */								
29 sinclude (addine h)								
00 binclude <timage.h)< td=""></timage.h)<>								
32								
33 static const uin	t8_t image_data_battery_46x	26[156] = {						

Now in the sketch file, *let the project know* we are using those external data by declaration with extern meaning that we are declaring the data somewhere outside.



We may now display icons by calling the function GFXDisplayPutImage().



#### Chapter 8 An estimation on power consumption

This section attempts to analyze the power consumption of a Memory LCD when it's content is maintained and updated. Although such data has been stated in each of the datasheets of various LCD models, a full understanding on when and how power is spent would make sense for future low power design. Block diagram below shows the functional blocks for measurement.



Photography on next page shows the set up in action.



Run this sketch from File→Examples→MemoryLCD→Energy. Press SW3 to start measurement.

DirectIO-master		
Memory LCD	•	BloodPressure_GUI
Ra8876_Lite	•	Energy
SdFat	•	FirstPixel
Time	•	HelloWorld
TinyGSM	<b>&gt;</b> ⊺	
ugfx-arduino-master	•	

Energy measurement is performed by a function void ina226\_measure(void) with listing shown below.

```
void ina226_measure(void)
{
    busVoltage_V = (float)ina.readBusVoltage(); //in V
    shuntVoltage_uV = (float)ina.readShuntVoltage()*1000000.0; //in uV
}
```

The variable busVoltage\_V returns the operation voltage of the Memory LCD. This value is measured from the output of TPS60140 DC-DC which got tethered to VDD & VDDA of Memory LCD via R13 (0  $\Omega$ ). Current is measured from a shunt resistor of R5 (100 $\Omega$ ±0.1%) in the return path of the LCD to ground. An extract of the schematic is shown below:



It has been found that the time to make measurement is critical. The power consumption during display update(dynamic) to that of display maintain(static) is different.

To tackle this problem a function dedicated for power measurement during display update is created.

Listing on next page shows the function

uint32\_t GFXDisplayTestPattern(uint8\_t pattern, void (\*pfcn)(void)).

```
uint32_t GFXDisplayTestPattern(uint8_t pattern, void (*pfcn)(void))
{
        uint32_t timing = 0;
#if defined (ARDUINO)
        uint32_t sMillis = millis();
#else
        #error Need to define the function to return millisec for other platforms
#endif
 hal_spi_start_transaction();
 hal_delayUs(3); //SCS setup time of tsSCS (refer to datasheet for timing details)
 for(uint16_t line=1; line<=DISP_VER_RESOLUTION; line++)</pre>
  {
        #ifdef LS032B7DD02
        hal_spi_write_byte(uint8_t((line<<6)|0x01));</pre>
        hal_spi_write_byte((uint8_t)(line>>2));
        #else
        hal_spi_write_byte(0x01);
        hal_spi_write_byte((uint8_t)line
        #endif
        uint32_t writePeriod = DISP_HOR_RESOLUTION>>3; //divide by 8 for 1-bit bpp
        while(writePeriod--){
        hal_spi_write_byte(pattern);
        }
        if(line==DISP_VER_RESOLUTION/2)
        if(pfcn!=NULL) {
        pfcn();
 }
        hal_spi_write_byte(0x00); //dummy byte
        hal_spi_write_byte(0x00); //dummy byte
        hal_spi_end_transaction();
#if defined (ARDUINO)
        timing = millis()-sMillis;
#else
        #error Need to define the function to return millisec for other platforms
#endif
        return timing;
}
```

There are two arguments required to run this function:

1. uint8\_t pattern : defines the horizontal bit pattern in 8-bit width so that the whole horizontal line spanning across the LCD will repeat in this pattern for the vertical direction.

2. void (\*pfcn)(void) : is a pointer to function to execute when horizontal line is updated at half the height of the LCD. This is controlled by the snippet below.

```
...
if(line==DISP_VER_RESOLUTION/2)
{
    if(pfcn!=NULL) {
        pfcn();
        }
    }
...
```

The time taken to update vertical strip pattern is returned in milliseconds.

When switch SW3 is pressed, the code will check for a simple software delay of 10ms to debounce the key and print a short message via Serial Monitor as "Energy measurement when display is updated ::". Then the function GFXDisplayTestPattern(0xF0, &ina226\_measure) will be executed with the time taken returned with variable updateTime.

Current is calculated by simple Ohm's Law:

shuntCurrent\_uA = shuntVoltage\_uV/100; //with a shunt resistor 100Ω

Power is calculated not from a direct equation P=V\*I. A time factor has been included to take account of the fact that only a finite time interval is required to display the vertical strip pattern.

```
power_uW = shuntCurrent_uA*busVoltage_V*updateTime/1000; //update time in ms
```

For full listing readers may check the source code Energy.ino Arduino Sketch.

The accuracy was verified with a DSO (Tektronix TDS1012B). An instrumentation amplifier AD620 set to a gain of x200 was used to boost the shunt voltage so that it could be measured by TDS1012B. Data comparing measurements made by the external DSO to that of INA226 shunt amplifier onboard of the Memory LCD Shield are shown below.



Don't forget the signal fed to CH1 has been amplified by x200 gain factor.

Therefore, the true value measured by DSO = 3.76V/200 = 0.0188V = 18.8mV.

This agree quite well with INA226's result.

Time taken to update the vertical strip pattern was around 58ms as measured by the DSO. This result matches very well with the parameter returned by software. But hold! How come there was a dip in the middle? This was caused by INA226's ADC sampling on VIN- and VIN+. Occurring in the middle because we have programmed the measurement to be taken when the line number was equal to DISP\_VER\_RESOLUTION/2 which is the half vertical height of the display.

Putting together the LCD voltage, shunt current, and the time taken to update,

Powerupdate = 5.01V \* 183.15uA \* 58/1000 = 53.22uW.

That's it when Memory LCD is being updated. But what about when it is not? When an image is maintained, which happens 90% of the time for modern GUI, a Memory LCD drains significant power only when the EXTCOMIN pin goes from low to high in its perpetual external COM inversion signal at 1Hz.

DSO trace below shows the same trace at CH1 with CH2 measuring EXTCOMIN.



Again, don't forget CH1 was amplified at x200 gain.

A closer look at CH1 reveals a pulse of 2.50ms peaks at around 4/200 V (20mV) when EXTCOMIN goes from low-to-high. For the rest of 997.5ms in a 1 second interval the shunt voltage was not measurable even with the help of a x200 pre-amplifier!



This is equivalent to say that, power consumption when the Memory LCD is maintaining

an image like this:



showing the vertical strip we have updated when SW3 was pressed, plus all readings covering part of the strip pattern is calculated as

 $Power_{maintain} = 5V * (20mV/100) * 2.5ms/2/1000 = 0.00125mW = 1.25uW.$ 

### Conclusion

Memory LCD is a high quality monochrome LCD delivering high frame rate, high contrast LCD, with partial update feature and the at same time consuming as small as few microWatt when display content is maintained.



Appendix A Schematic diagram of Memory LCD Shield



## PCB layout of Memory LCD Shield

Top layer



Bottom layer





Appendix B Schematic diagram of ESP32 Application Processor

## PCB layout of ESP32 Application Processor

Top layer



Bottom layer

