

Ph. 480-503-4295 | NOPP@FocusLCDs.com TFT | DLED | GRAPHIC | CHARACTER | UWVD | SEGMENT | CUSTOM

Application Note FAN3207

Driving a 128x64 Graphic LCD with a Microcontroller: A Programming Guide

This application note will present the project for driving the G126FLGFGS164T33XAR Graphic LCD Module with an STM32F411RE-Nucleo microcontroller development board.



Contents

Introduction	4
Hardware Requirements	5
Development Tools	7
STM32CubeMX and STM32CubeIDE	7
STM32F411RE-Nucleo Project Development	8
Start a New Project in STM32CubeMX	8
Peripheral Configuration	9
Project Management	12
STM32CubeIDE	14
Add Files	15
Modify the Code	15
Change the Optimization Settings	16
Summary	18
Appendix	18
LCD Handling Precautions	19
Disclaimer	19





List of Figures



Driving a Graphic LCD with a Microcontroller Board

In this application the hardware used, and firmware developed to drive a graphic LCD will be presented. The graphic LCD in this application note is the G126FLGFGS164T33XAR from Focus LCDs. The STM32F411RE-Nucleo from ST Microelectronics will be the microcontroller board for this project. The Firmware will be developed for the Nucleo-F411 board.



Figure 1: G126FLGFGS164T33XAR 128 x 64 Graphic LCD

Introduction

Driving the G126FLGFGS164T33XAR Graphic with images will require a microcontroller development board and firmware. The development board chosen for this project is the STM32F411RE-Nucleo. This application note will show the hardware and firmware development with the STM32F411RE-Nucleo. The board has a 3.3V supply rail with enough current to power the LCD and SPI interface. Though not required, a separate 3.3V power supply will power the backlight to ensure the required amount of current is provided.

Contact Focus LCDs for more information on the graphic LCD and firmware.

The G126FLGFGS164T33XAR display used in this application is a 3.175" graphic LCD with a 128 x 64 monochrome dot resolution from Focus LCDs. This display is interfaced using the SPI protocol with 14 through hole pins.



The main features of the G126FLGFGS164T33XAR are:

- 3.175" diagonal display, 128 x 64 dot resolution
- Monochrome color
- SPI Interface
- Transflective, FSTN Positive
- RGB LED Backlight
- ST7565P Display Controller
- Typical Operating Voltage: 3.3V

Hardware Requirements

The development board that will drive the display is STM32F411RE-Nucleo board as shown below.



Figure 2: STM32F411RE-Nucleo Development Board

In addition to the development board, a piece of prototyping PCB is used to connect the display to header pins. The display is soldered to the prototype PCB along with the header pins. This allows standard Female-to-Male jumper wires to be connected between the dev board and display.





Figure 3: G126FLGFGS164T33XAR with Adapter

In the table below the connections from the display and prototype board to the STM32F411RE Nucleo are listed.

G126FL	GFGS164T33XAF	R Display	STM32F411RE-Nucleo					
Pin	Signal	Function	Pin	Connector	Port	Signal		
2	SCL	Serial Clock	25	CN10	PB10	SCK		
3	SI	Serial Data	37	CN7	PC3	MOSI		
4	VDD	3.3V Power	5	CN7	-	VDD		
5	A0	Register Select	38	CN7	PCO	A0		
6	/RESET	Reset Signal	36	CN7	PC1	NRST		
7	/CS	Chip Select	16	CN10	PB12	NCS		
8	VSS	Ground	19	CN7	-	GND		

The image below shows the physical connections between the display and board.



Figure 4: Display Connected to Nucleo Board



Development Tools STM32CubeMX and STM32CubeIDE

Developing applications will typically use STM32CubeMX for setting up peripheral initialization and STM32CubeIDE (integrated development environment) to code the application side. These tools can be downloaded from the ST website. They do require an account to be created before downloading the tools.

Here is the STM32CubeMX Development Tool:



Figure 5: STM32CubeMX Initialization Code Generator

The STM32CubeMX tool will be used to configure the peripherals that will be used to drive the graphic LCD. In this application the general-purpose IO and the SPI serial port will be configured. In addition, a USART will be configured in case the end user would like to use a serial port for debugging.





Next is the STM32CubeIDE Development Environment:

wonapace_1.14.1 * 31W02CubeDE									-	
Eile Edit Source Refactor Navigate Search Project E	un Window Help 💄 Hello Tony									and a family
	• 🖻 • 🕲 • 🕸 • 🛈 • 💁 • 🥭 🗸	· 토 비 비 입 ·	1-000-	10 x						
💫 Project Explorer 🗙 👘 🖏 🍞 💈 🛄 🗖							- 0	E Outline X	Build Targe	0
There are no projects in your workspee. To add a project To add								outline.	ve editor that pro	vides an
	🖹 Problems 🗙 🤌 Tasks 📮 Console 🛽	Properties	<u>8</u> • •		Build Analyzer	🗙 📠 Static Stac	k Analyzer	Ocyclomatic	Complexity	0
	Problems X Tasks Console [Ottems	Properties	₹ § =		Build Analyzer	× 🛋 Static Stac	k Analyzer I	D Cyclomatic (Complexity	
	Problems X Tasks Console (Otems Description	Properties Resource	₽ath	Lo	Build Analyzer	× 🛋 Static Stac	k Analyzer I	⊕ Cyclomatic (Complexity	
	Problems X Tasks Console [Otterns Description	Properties. Resource	₽ath	Lo	Build Analyzer	× 🛋 Static Stac	k Analyzer I	∂ Cyclomatic (Complexity	
	Problems X (2) Tasks (2) Console (1) 0 tense Description	Properties Resource	تې § ده Path	Lo	Build Analyzer Memory Regions Region	× a Static Stac Mernory Details Start address	k Analyzer I	Cyclomatic (Size	Complexity	
	E Problems X Tasks C Console 0.tems Description	Properties Resource	تې § ه Path	Lo	Build Analyzer Memory Regions Region	× 🛋 Static Stac Memory Details Start address	k Analyzer (End addre	Cyclomatic (Size)	Complexity Free	■ ■
	Problems X 2 Tasks Console () Ottoms Description	Properties Resource	Path	Lo	Build Analyzer Memory Regions Region	× 🛋 Static Stac Memory Details Start address	k Analyzer (End øddre:	Cyclomatic (s Size	Complexity Free	
	Problems X Tests Console Citems Description	Properties Resource	₽ath	Lo	Build Analyzer Memory Regions Region	× 🛋 Static Stac Memory Details Start address	k Analyzer 1 End addres	Cyclomatic (s Size	Complexity	U

Figure 6: STM32CubeIDE - Integrated Development Environment

After installation, STM32CubeMX or STM32CubeIDE might request additional downloads depending on the device series integrated into the project. When developing the firmware, an additional download for the F4 series of devices was required for both tools.

STM32F411RE-Nucleo Project Development

In this section, the initial project generation will be presented. The steps to create a new project, what files are to be added, initial modifications, building, and testing the firmware will be shown.

Start a New Project in STM32CubeMX

Open STM32CubeMX and select "Start My project from ST Board" from the available selections under the "New Project". This will bring up the board selection dialog. In the "Commercial Part Number" drop down, type in "Nucleo-F411RE" This will then show the board in the "Board List" in the lower half of the window. Click on the board image shown and a detailed view of the board will be displayed in the upper half of the window.



Commercial	~	Features STM32F4 Series	Large Picture	Docs & Resources	E Datashee	n 📑 Buy	Start Project
Q V	+-	NUCLEO-	F411RE STM3	2 Nucleo-64 developn ho connectivity	nent board with STM	132F411RE MCU, suppo	orts Arduino and ST
DDODUCT INFO		ACTIVE	PattNu	mber: NUCLEO-F411RE		Unit Price (US\$): 13.0	
PRODUCT INFO		Product is in mass	production Comme	ercial Part Number : NUCLEO-F	F411RE	Nounted Device : STM32F411RE	<u>T6</u>
MCU / MPU Series Marketing Status Price MEMORY Ext. Flash = 0 (MSH) 0 Ext. EEPROM = 0 (MSH) 0	> > - -				combinations of performs \$TM32 microcontroller. F significantly reduces pow The ARDUINO [®] Uno V3 of the easy expansion of the platform with a wide choic the easy expansion of the res STM32 Nucleo-64 to integrates the ST-UINI do software libraries and exp	nce and power consumption or the compatible boards, the or consumption in Rum mode connectivity support and the is functionality of the STM22 e of specialized shelds. and does not require any se buggerprogrammer, and comes with the STM22 imples available with the STM	I features, provided by the external SMPS ST morpho headers allow Nucleo open development iparate probe as it comprehensive free M32Cube MCU Package.
Ext. RAM = 0 (MBit)	_	Boards List: 1 item					đEx
0		• Overview X	Commercial Part No	e ty	npe X Mart	keting Statue 🗙 Unit Price	(US\$) × Mounted Devic
FEATURES	~	o LAND	UCLEOLEANDE	Nucleo,64	Active	13.0	

Figure 7: Board Selection Window

Once the board has been selected, click on "Start Project" and select "No" in the initialize default peripherals dialog.



Peripheral Configuration

In this section the peripherals will be configured. Start in the "Pinout & Configuration" tab, and under that select "System Core" to access the GPIO.

In the datasheet for the G126FLGFGS164T33XAR there are 3 pins that need to be configured to control the display. The A0 pin is the register select pin which determines if the SPI transfer is a command (LOW) or data (HIGH). A0 will be initialized to idle high to avoid accidentally sending a command. The next pin is RESET (NRST), which is active low. This pin will also be set to idle high to avoid accidental resets. The chip select signal (NCS) will also be configured as a GPIO to have finer control of when it gets activated. NCS is active low and like the other pins, it will idle high. These settings are shown in the image below.



STM32CubeMX stm32f411_sp	i_demo.ioc: ST	M32F411RETk NUCLE	0-F411RE													3 - 2	
CubeMX	F	ile		Window	1		Help			L Hello	Tony	(1)	f			3	~ 5
Home 🔰 STM32F411F	RETX - NUC	CLEO-F411RE	> stm3	stm32f411_spi_demo.ioc - Pinout & Configuration					C	ENERATE	CODE						
Pinout & C	onfigurati	on		Clock Configuration			Project Manager Tools				l.						
				✓ Softwa	re Packs			Pinout									
۹	٢			GPIO Mode	and Configu	ration		1			0	Pinout view	I1	System viev	v		
Categories A->Z				Con	figuration												
System Core	~	Group By Perip	herals					~									
		SPI	•	SYS	O US	SART	NV	10									
DMA	-	GPIO		Single	Mapped Sig	nals	© F	CC					8		2		
GPIO IMBO		Cassels Signals													2 4		
NVIC		Search /Ctritten					Show only Modi	fied Pins					2 2 2 2	PCI PCI PCI	100		
✓ SYS		Demon (oning)		lania	Lanua n			-		B1(Blue Pus	hilution)				V88		
WWDG		Pin Signal	D. GPIO	o GPI0 m Output	Mo pulle	. Maximu	LD2 (Gr	Modified		400_00 400_000	32,04T				PAIZ		
		PB12 n/a	Low	Output .	No pull	Low	NCS			RCC_O	0005_00 00055				PA11 PA10		
Analog	>	PC0 n/a	Low	Output .	No pull	Low	A0			- I	10 100				PAB		
		PC1 n/a	High	Output .	No pull	. Low	NRST				RET RET				826		
Timers	>			all state in the	and the pairs in		e i feierr	-		38	2,0081 x=3	OTA			967		
Connectivity	>										VDBA	STIVIS L	QFP64		115		
											PAL				210.12	1	
Multimedia	7									*0	ANT_TX IN	8 2 2 2 2 3			al activity		
Computing	>										3	3	1-1-1-1-1-1	8	2		
Middleware and Software	Dacke >										TANEN	Diffeen		048			
mouleware and SoftWare	acto /	2 Select Pins fr	om table to	configure the	em. Multiple	e selectio	n is Allowed.					19:					
									Q	23	Q		1			Q	

Figure 9: GPIO Configuration

Once the GPIO pins are configured, the SPI peripheral is next. In this application, SPI2 will be used as SPI1 is unavailable. Again, in the "Pinout & Configuration" tab, select the "Connectivity" section to gain access to SPI2 configuration. Under "SPI2 Mode & Configuration" (center section) set the Hardware NSS Signal to disable since a GPIO (NCS) will be used to select the ST7565P display controller. Next, set the basic parameters for Motorola, 8-bits, and MSB first.

STM32CubeMX stm32f411	spi_demo.ioc: STM	32F411RETx NUCLEO-F411RE						- 🗆 X
	File	е	Window	Help	L Hello Tony	1	f 🖸 X 🗘	اک ⊁ 🍪
lome > STM32F41	1RETx - NUCL	EO-F411RE 🔪 stm:	32f411_spi_demo.ioc - F	Pinout & Configuration >		GENERATE CODE		
Pinout &	Configuration	1 1	Clock Configurati	ion	Project Manager		Tools	
×	۵		SPI2 Mode and Configurati	on		🔯 Pinout view	System view	
Custom Care		Mode Transmit Only Ma	ister	~				
System Core		Hardware NSS Signal D	isable	~				
Analog	>	-						
Timers	>						i i i i i i i i i i i i i i i i i i i	
Connectivity						100 100 100 100	101 101 101 101 101 101 101 101 101 101	
Connectivity			Configuration		81(Blue PushBuiller)		100	
1201		Reset Configuration			REC_000_00R		PAI2	
A 12C2					800_000_H		PAIL	
12C3		NVIC Settings Parameter S	OMA Settings	GPIO Settings			240	
SDIO O ODH		Configura the below paramet	lore :	C COSCI CONSIGNES	AD BRIT		203	
SPI2		O Reamb (Ct/45)			972.WO	C2	PC0	
OPIS		Control (Carry)	U	- V	le la	STM32	F411RETX RES	
SPI4		Dasic Parameters Erame Format	Motorola		No. of the second se	LQ	FP64 9814	
USART1		Data Size	8 Bite		,	A3	100	
✓ USART2		First Bit	MSB First		WGART_TX			
USART6		Clock Parameters	inoto i not			NA VICE	PER PER	
USB_OTG_FS		Prescaler (for Ba	ud Rate) 4			of the second	8	
		Baud Rate	10.5 MBits/			ā 6	10	
		Clock Polarity (C	POL) High			5		
Multimedia	>	Clock Phase (CF	PHA) 2 Edge					
0		Advanced Parameters		- 1				
Computing	,	CRC Calculation	Disabled		0 53 6		-	-
Middleware and Softwar	Par >	NSS Signal Type	Software		Q LI G	2 🕒		Q

Figure 10: SPI Configuration





The clock parameters are critical to the correct communication between the display and dev board. Set the Baud Rate prescaler to 4 so that the serial clock is within the ST7565P clock limitations. Looking at the ST7565P datasheet for the SPI clock timing diagram, it is shown that the clock idles high, and that the data is read on the second edge. Set Clock Polarity to High and Clock Phase to 2 Edge.



Figure 11: SPI Clock and Data Timing

Configuring USART2 is straight forward as the default parameters require no changes. See the image below for configuration. Though not used in this project, the USART is included for debugging.

STM32Cub	eMX stm32f411_spi_dem	o.ioc: STM	132F411RETx NUCLEO-F411RE								- 🗆 X
STM32		Fi	le	Window	Help		L Hello Tony	(19)	f 🕒 🛛	0	> 🕂 🟹
Home >	STM32F411RETx	- NUC	LEO-F411RE 🔪 stm3	2f411_spi_demo.ioc ·	Pinout & Configur	ation >			GENERATE C	ODE	
	Pinout & Config	guratio	n	Clock Configura	ition		Project Manager			ools	
Q	~	٢	L	ISART2 Mode and Configu	Iration	1		Pinout view	System view		
Categories	A->Z			Mode							8
Sustan	Care		Mode Asynchronous			~					1
System	oure	<u> </u>	Hardware Flow Control (i	RS232) Disable		~					
Analog		>									
Timara		s.							346 X		
Timers	_	<u>^</u>									
Connecti	vity	×.		Conferentian		_	21.0				
-				Comgenation			806_08632_8	ţ.		100	
1201			Reset Configuration				ADD_0000.00 P10	ţ.		455	
12C3			NVIC Settings	OMA Settings	GPIO Sett	ings	#DC_050C_047			130	
SDIO			Parameter Si	attings	User Constants		A0 PC0				
Ø SPI1			Configure the below paramet	ers :			PC2			19	
SPI3			Q Search (Ctrl+F) (Ctrl+F)	0		0	012,000 100	STM32	411RETx	17	
SPI4			Basic Parameters				1000	LQ	FP64		
SPIS	174	_	Baud Rate Word Length	115200 Bi 8 Bite (inc	ts/s Judina Paritu)		PAL	1		10	
S USAR	रा2		Parity	None	ioung rany)		USANT_TX 7A3			108	
	170	- 1	Stop Bits	1							
USB	OIG_FS		 Advanced Parameters 	12000000				C Land	8		
			Data Direction	Receive a	nd Transmit			2 d 9			
Multimed	lia	>	Over Sampling	16 Sampa	15						
Computir	19	>				- 1	0 11 0	12	<0 m	-	
Middlewa	are and Software Pac	>					य 13 य	<u> </u>			4

Figure 12: USART2 Configuration

Finally, the clock will be configured. The default settings for the clock peripheral are good for this application and will not be changed from the default values. The image below shows the clock configuration.





Figure 13: Clock Configuration

Project Management

The next section to configure is the "Project Manager" tab. Under the tab will be 3 sections on the left-hand side of the window. Make sure "Project" is selected. Here is where the project settings will be entered. Provide a name for the project and then select the toolchain from the drop-down menu. Keep "Generate Under Root" box checked.

STM32CubeMX stm32f411_spi_de	emo.ioc*: STM32F411RETx NUCLEO-F411RE					- 🗆 X
	File V	Vindow	Help	L Hello Tony	🥸 F 🕒 🛛	0 🍑 🔆 🖅
Home STM32F411RE	Tx - NUCLEO-F411RE > stm32f4	11_spi_demo.ioc - Proj	ect Manager >		GENERATE CC	DE
Pinout & Con	figuration C	Clock Configuration		Project Manager	Т	pols
Project	Project Settings Project Name Project Location	stm32f411_sp	_demo	1.34.1	Browse]
Code Generator	Application Structure Toolchain Folder Location Toolchain / IDE	Advanced C1Usersttony	a\STM32CubeIDE\workspace	1 14 1\stm32411_spi_demo\ erate Under Root	○ □ Do not generate the main()	
Advanced Setting	s Minimum Stack Size	0×200 0×400				
	Thread-safe Settings Cortex-M4NS Enable multi-threaded support Thread-safe Locking Strategy	ort Default – Map	ping suitable strategy depend	ing on RTOS selection.	~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~	
	Mcu and Firmware Package — Mcu Reference Firmware Package Name and V	STM32F411RE	ETx W_F4 V1.28.0]
	Use Default Firmware Locat	C:/Users/tony	s/STM32Cube/Repository/STI	//32Cube_FW_F4_V1.28.0	Browse	

Figure 14: Project Settings





In the next section "Code Generation", make sure the radio button "Copy only the necessary files" is selected and that the check boxes "Generate peripheral code initialization as a pair of '.c/.h' files", "Keep User Code when re-generation", and "Delete previous generated files when not re-generated".

STM32CubeMX stm32f411_spi_t	demo.ioc*: STM32F411RETx NUCLE	EO-F411RE		M	- 🗆 ×
STM32	File	Window	Help	L Hello Tony	🐵 🖪 😐 X 🗘 🍏 🔆 ភ
Home > STM32F411RE	ETx - NUCLEO-F411RE	> stm32f411_spi_demo.ioc	- Project Manager >		GENERATE CODE
Pinout & Co	nfiguration	Clock Configura	ation	Project Manager	Tools
Project	 STM32Cube MC ○ Copy all used ⑧ Copy only the ○ Add necessa 	U packages and embedded software d libraries into the project folder e necessary library files ny library files as reference in the too	packs		
Code Generato	Generated hies -	ipheral initialization as a pair of '.c/.h ously generated files when re-genera	' files per peripheral ting		
Advanced Setting	gs	usly generated files when not re-gene	erated		
	, HAL Settings—	ins as analog (to optimize the power Assert	consumption)		
	/User Actions	neration			Browse
	After Code Gene	eration			Browse
	, Template Setting Select a templa	gs te to generate customized code	Settings	ч	

Figure 15: Code Generator Settings

The final step in setting up the project is to click on "GENERATE CODE".

CubeMX	E	ile	Window	Help	L Hello Tony	🐵 <mark>F 🖻 🗴 </mark>
Home 🔰 STM32F4	11RETx - NUC	CLEO-F411RE	stm32f411_spi_demo.ioc - Pine	out & Configuration $>$		GENERATE CODE
Pinout &	Configuratio	on	Clock Configuration		Project Manager	Tools
λ[]	0		SPI2 Mode and Configuration	1	🗇 Pir	nout view 🔢 System view
Categories A->Z			Mode			
System Core	>	Mode Transmit (Only Master	~		
		Hardware NSS S	ignal Disable	~		
Analog	<u> </u>					
Timers	>					₿ Ĕ

Figure 16: Generate Code





Once the code generation is completed the following dialog box will allow the project to be opened within the STM32CubeIDE.



Figure 17: Code Generation Successful

STM32CubeIDE

<pre>Ele E a guest Androy Engleste Sageh Boek Bony Worker Bing Wor</pre>	workspace_1.14.1 - stm32f411_spi_demo/Core/Src/main.c - S1	32CubelDE				-		-	
Image: International State State Image: International State <t< td=""><td>Eile Edit Source Refactor Navigate Search Project Ru</td><td>Window Help L Hello Tony</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></t<>	Eile Edit Source Refactor Navigate Search Project Ru	Window Help L Hello Tony							
<pre>bp Project Explore x</pre>	🗂 • 🗐 🕼 🕸 • 🗞 • 📾 🖉 🏷 🗙 (ð) 📷 • 6	• 🖻 • 🞯 • 🕸 • 🔕 • 💁 🛷 •	🇾 📴 👖 🖗 🕶 🖓 🕶 🗘 🕫 🤇	> • ⇔ • 🛃	0				Q 😰 🖪
<pre>1 1 / USA COE BIGN Nearer */ 2 **********************************</pre>	Project Explorer ×	e) main.c × (a) st7565.h (c) st7565.c					BE Outline	× Build Targe	ts 🗖 🗖
Problems @ Tasks @ Console X Properties Protection Ne consoles to display at this time. Build Analyzer X Basks Stack Analyzer (@ Cyclomatic Complexity Problems Build Analyzer X Basks Stack Analyzer (@ Cyclomatic Complexity Problems Properties	Imail: A line in the second	1 /* USER CODE BEGIN Header */ 2007** 3 * [file :: main.c. 4 * copyright (c) 2024 STHicreelect 4 * copyright (c) 2024 STHicreelect 4 * copyright (c) 2024 STHicreelect 4 * This software is licensed under 1 * This software is licensed * * * * * * * * * * * * * * * * * * *	ronics. Terms that can be found in the L offsame component. It is servering, it is provided AS-1	ICENSE file (5		, ,	1 ma 1 spi 1 usa 1 gpi 1 st7 1 st7 1 sos	in Plank	v² ● ₩ § oid) : void oid) : void oid inf(22,0) : void
Ne consoles to display at this time. Memory Regions' Memory Details Region Start address End address Size Free U		🖞 Problems 🧔 Tasks 📮 Console 🗙 🛅 Proper	ties 📑 🖻 🕶 🗖	Build Analyze	er 🗙 📠 Static Sta	ick Analyzer	(a) Cycloma	tic Complexity	- 0
Memory Begions Memory Details Region Start address End address Size Free U		o consoles to display at this time.							🛃 🤣 🖇
Memory Regions: Memory Details Region Start address End address Size Free U									
Region Start address End address Size Free U				Memory Region	ns Memory Details				
				Region	Start address	End addre	ess Size	Free	U
				٢					>
Writable Smart Insert 95:29:2567 🖗			Writable Smi	art Insert	95:29:2567	R			

Figure 18: IDE Project

Now that the project setup with STM32CubeMX is complete, the project can be opened in the IDE. There will be several pre-generated files including header and source files for the different peripherals. Modification will be made to the main.c file along with the addition of st7565.h and st7565.c files.





Add Files

Two files need to be added to the project: st7565.h and st7565.c. Place these files in the Inc and Src folders of the project. Then right click the top-level project file in Project Explorer and select refresh to update the project with the new files. Contact Focus LCDs to obtain the individual files or the complete set of project files.

Currently, the st7565.h file contains the command definition for the ST7565 controller chip along with display specific definitions. Function prototypes for interfacing with the SPI bus, for controlling the display features, and the graphic functions are all included in this file. The st7565.c file contains the definitions of the functions used to send commands and data over the SPI bus and controlling the display. The graphic functions have not had their code added at this time but might be included in a future release.

Modify the Code

In **main.c** a few modifications are needed to complete the application. The **#include "st7565.h"** needs to be added to the code. After the generated includes, the previous line gets placed between the USER CODE BEGIN/END includes comments as shown below.

18 0	<pre>/* USER CODE END Header */ /* Includes</pre>	
20	<pre>#include "main.h"</pre>	
21	<pre>#include "spi.h"</pre>	
22	<pre>#include "usart.h"</pre>	
23	<pre>#include "gpio.h"</pre>	
24		
25	Management of the second state of the second s	
268	<pre>/* Private includes</pre>	
27	/* USER CODE BEGIN Includes */	
28	<pre>#include "st7565.h"</pre>	
	/* USER CODE END Includes */	

Figure 19: Include the st7565.h Header File

The final modification is to add the code for initializing the display and updating what is displayed on the screen.

89	/* Initialize all configured peripherals */
90	MX GPIO Init();
91	MX_SPI2_Init();
92	MX_USART2_UART_Init();
93	/* USER CODE BEGIN 2 */
95	<pre>uint8 t setContrast = 0x1C; // middle contrast/brightness</pre>
96	ST7565 Init(setContrast);
97	<pre>ST7565_UpdateDisplay();</pre>
99	/* USER CODE END 2 */

Figure 20: Display Init and Update Code





Change the Optimization Settings

The next step in creating the firmware is setting the optimization level. The typical setting is '-O2' for a good compromise between execution speed and code size. If left at '-O0' there would be no optimization for execution speed and code size. Depending on whether debugging the application code or releasing the application code the optimization settings can be adjusted. In this application it will be changed to '-O2'.

First, in Project Explorer right click on the project and select properties.

In the pop-up window, select C/C++ Build->Settings. Then in the Tool Settings tab on the right, select Optimization and set the Optimization level to -O2.



Figure 21: Optimization Levels

The final step is to build the project. This step ensures that there are no missing dependencies. If there are any errors or warnings, resolve those issues before continuing with the firmware project. In the Appendix, additional information is included to support expansion to graphical application.





The following images show a simplified, monochrome Focus LCDs logo on the graphic LCD with the various backlight colors.



Figure 22: Graphic LCD with Logo, no Backlight



Figure 23: Graphic LCD with Logo, Green Backlight



Figure 24: Graphic LCD with Logo, Blue Backlight



Figure 25: Graphic LCD with Logo, Red Backlight





Summary

In this application note it was shown how to connect and drive a graphic LCD. This application can be used as a reference to drive any of the Focus LCDs graphic display that use the ST7565 controller chip. An STM32 Nucleo board was used as the development board to drive the display. Wiring the display to the development board required a small piece of prototyping board for the connections. The tools provided by ST Microelectronics aided in the development of the application firmware by generating all the low-level peripheral code.

Appendix

Turning the code into a larger graphical application requires several modifications. Making the code modular is the priority by splitting out the functions into 3 levels. The code presented in this application note was just to show how to set up a development board, connect to the display, and drive an image to the display.

The lowest level of code above the SPI interface code would be the ST7565 header and source files. The header file should just contain the command (and associated data) defines along with the external SPI handle and function prototypes. The source file should only contain the command and data functions. This would be the only level that interfaces with the SPI peripheral. By setting up the code this way allows the ST7565 files to be used with different graphic LCDs that use the ST7565 controller chip.

The second level should be header and source files specific to the graphic display. In this case a G126FLGFGS164T33XAR graphic LCD. In the header file should be the specific definitions for the display (i.e. the resolution) and the function prototypes. The only functions that should be at this level in the source file are the display initialization, brightness setting, a way to clear the display, and a function that either takes a single data or a data array to write to the screen. These functions will have specific code for the display used in the project.

The final level will be the graphics code. Functions that provide the basic graphic features of drawing a pixel, line, rectangle, circle, and bitmap should be placed at this level. This is the level that should have the frame buffer where all the drawing functions write their data. If the lower-level functions that this layer accesses are generically named or function pointers are used, then the graphic display can be replaced easily with another graphic display.

Building the display and graphics code in this way allows the easy replacement of the LCD with another that uses the same controller chip.



LCD Handling Precautions

- Do not store the TFT-LCD module in direct sunlight, best stored in a dark place
- Do not leave it exposed to high temperature and high humidity for a long period of time
- Recommended temperature range is 0 to 35 °C, relative humidity should be less than 70%
- Stored modules away from condensation as formation of dewdrops may cause an abnormal operation or failure of the module.
- Protect the module from static discharge
- Do not press or scratch the surface and protect it from physical shock or any force

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.

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