FOCUS LCDs
LCDs MADE SIMPLE®

Ph. 480-503-4295 | NOPP@FocusLCDs.com

TFT | OLED | GRAPHIC | CHARACTER | UWVD | SEGMENT | CUSTOM

Application Note FAN4220

Working with the E70GE1-RW820-N (Part 2)

This application note will discuss the firmware requirements for driving the E70GE1-RW820-N TFT LCD Display Module.

Working with the E70GE1-RW820-N (Part 2)

This application note will discuss the firmware requirements for driving the E70GE1-RW820-N TFT LCD Display Module. An ST Nucleo-F767ZI microcontroller (MCU) development board will interface with the display. In Part 1 (FAN4219) the hardware requirements were presented, and in Part 2 (this app note) the firmware will be developed.

1. Introduction

Following FAN4219, the MCU board is connected to the display. As noted previously, the Nucleo board has the IO pins and data bandwidth for the design but lacks the internal static RAM (SRAM) for a full frame buffer. Driving the display is the main purpose of this app note and a smaller frame buffer will be utilized.

When interfacing an RGB display, timing is critical and the information from the datasheet will be required to properly drive the LCD. The pixel clock (or DCLK) will be the fastest signal sent from the MCU to the display and is normally in a few MHz to tens of MHz. The color data and synchronization signals are not as high frequency compared to the pixel clock.

The synchronization signals might need adjustment depending on defects shown on the display. These defects could be horizontal stripping, vertical image scrolling, or patterns moving in either direction. These defects are best observed while driving the LCD black or in a zebra pattern and visually inspecting the display. If any defects are present, it is up to the user to adjust the SYNC, back porch, and front porch to remove these defects.

MCU's with TFT LCD peripherals will have registers where the timing information is entered. Development tools that provide a graphical user interface for easily setting the peripheral registers make the process quicker. Tools that also generate the low-level initialization code can help reduce errors in bringing up the application. This application note will use the development tools provided by ST.

2. Hardware Connection

The schematic below shows the E70GE1-RW820-N connected to the ST Nucleo-F767ZI MCU board.

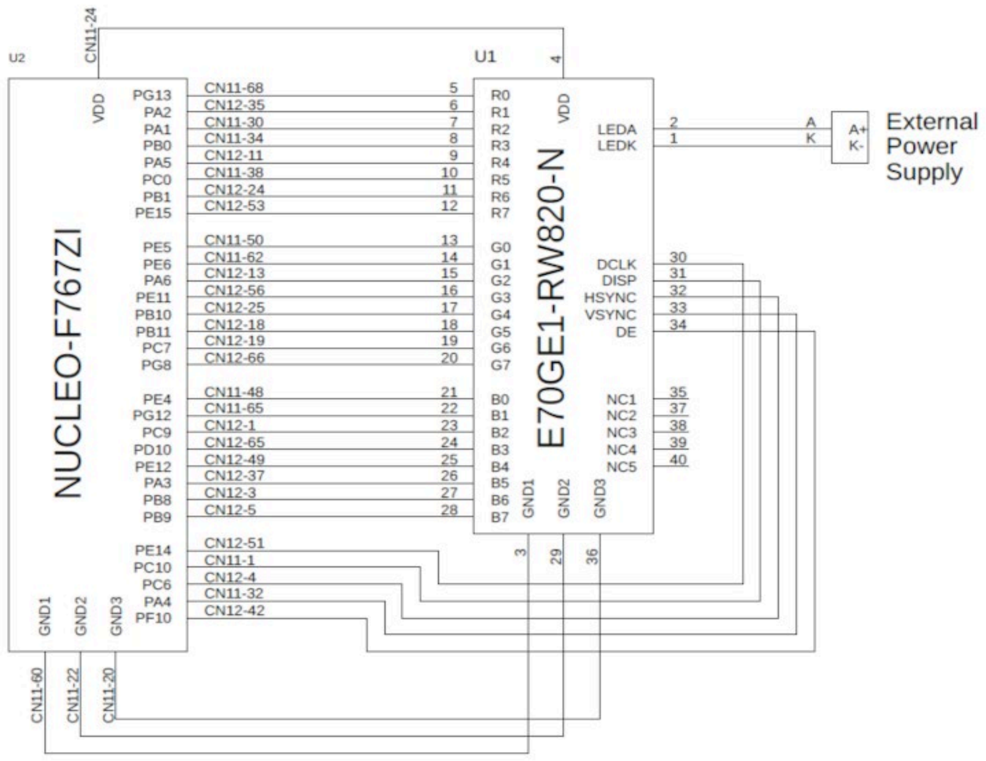


Figure 1. E70GE1-RW820-N to Nucleo Schematic

As shown, there are two different categories of pins:

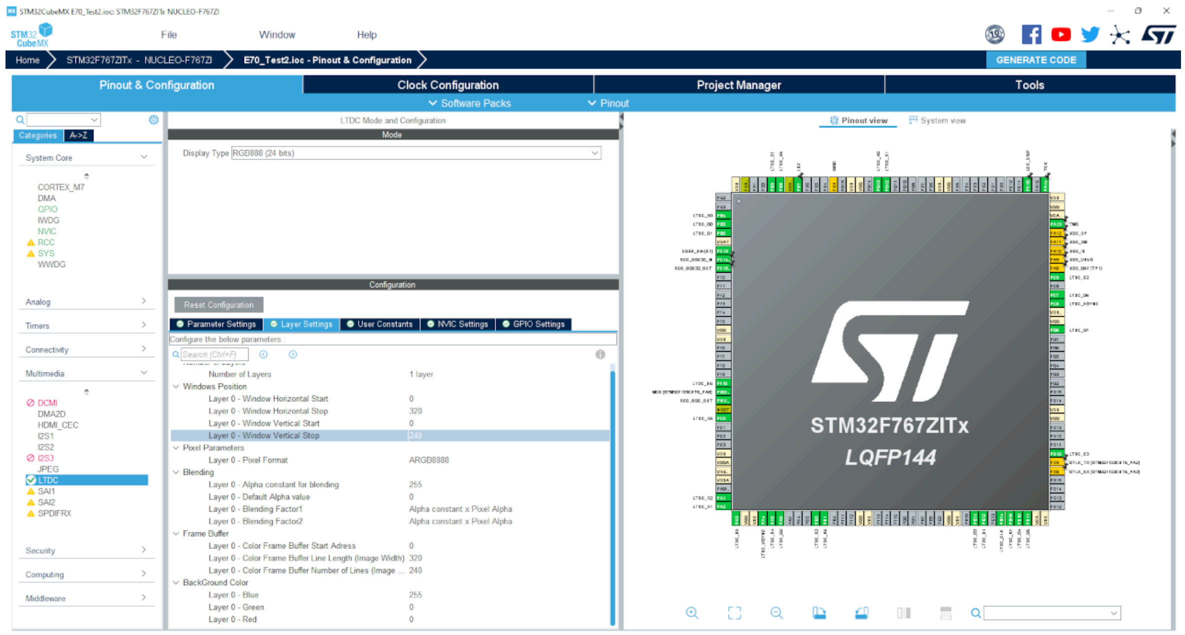
- LTDC Interface Pins
  - 24-bit RGB pins
  - Timing signals: HSYNC, VSYNC, DCLK (LCD_CLK), and DE (LCD_DE)
- Control Interface Pins
  - LCD_DISP to enable and disable the display standby mode
  - LEDA and LEDK for powering and controlling the backlight

The traces between the display and the MCU should be kept as short as possible to reduce the amount of unwanted interference. The ground plane should not have discontinuities underneath any signals to keep out induced noise. Though not strictly necessary, high-speed signal routing best practices should be utilized in the PCB layout.

3. Development Tools

Developing the application will use STM32CubeMX for setting up peripheral initialization and STM32CubeIDE (integrated development environment) to code the application side. Download these tools from the ST website and install the applications. The first time setting up a project in

STM32CubeMX will force a download of additional software components. The STM32CubeIDE will not require any additional downloads.



LTDC Mode and Configuration

Mode

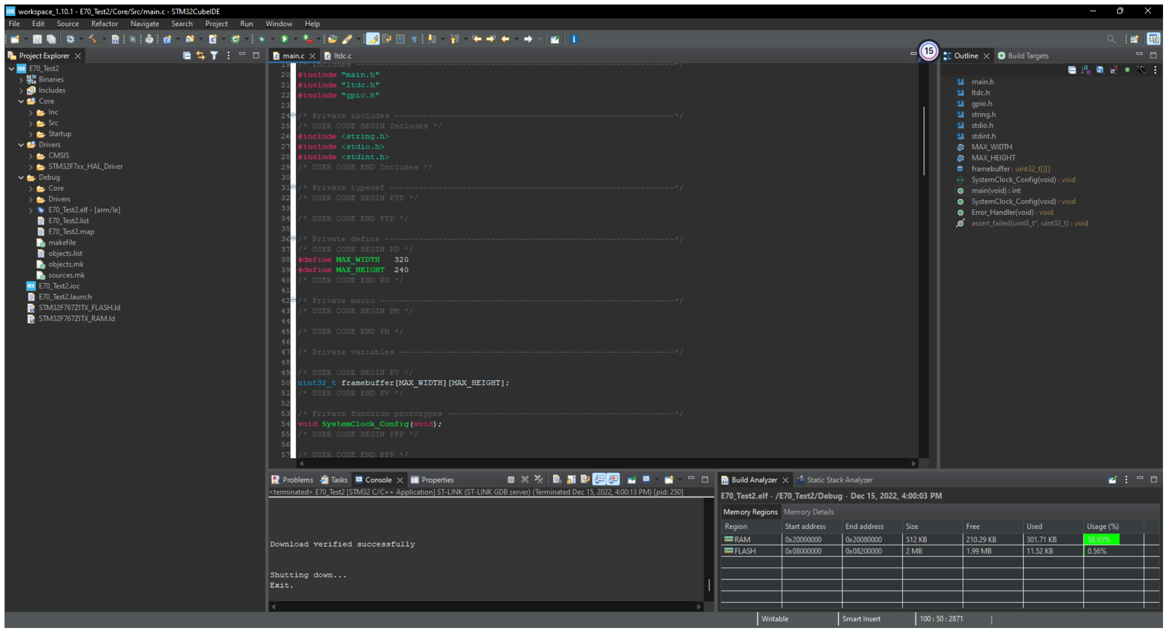
Display Type RGB888 (24 bits)

Configuration

Configure the below parameters :

Number of Layers	1 layer
Windows Position	
Layer 0 - Window Horizontal Start	0
Layer 0 - Window Horizontal Stop	320
Layer 0 - Window Vertical Start	0
Layer 0 - Window Vertical Stop	240
Pixel Parameters	
Layer 0 - Pixel Format	ARGB8888
Blending	
Layer 0 - Alpha constant for blending	255
Layer 0 - Default Alpha value	0
Layer 0 - Blending Factor1	Alpha constant x Pixel Alpha
Layer 0 - Blending Factor2	Alpha constant x Pixel Alpha
Frame Buffer	
Layer 0 - Color Frame Buffer Start Adress	0
Layer 0 - Color Frame Buffer Line Length (Image Width)	320
Layer 0 - Color Frame Buffer Number of Lines (Image ...	240
BackGround Color	
Layer 0 - Blue	255
Layer 0 - Green	0
Layer 0 - Red	0

Figure 2. STM32CubeMX Configuration Tool



workspace_1.10.1 - E70_Test2/Core/Src/main.c - STM32CubeIDE

```
#include "main.h"
#include "ltdc.h"
#include "gpio.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include <string.h>
#include <stdio.h>
#include <stdint.h>
/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
#define MAX_WIDTH   320
#define MAX_HEIGHT  240
/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/

/* USER CODE BEGIN PV */
uint32_t framebuffer[MAX_WIDTH][MAX_HEIGHT];
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */
```

```
<terminated> E70_Test2 [STM32 C/C++ Application] ST-LINK (ST-LINK GDB server) (Terminated Dec 15, 2022, 4:00:13 PM) [pid: 250]


Download verified successfully


Shutting down...
Exit.
```

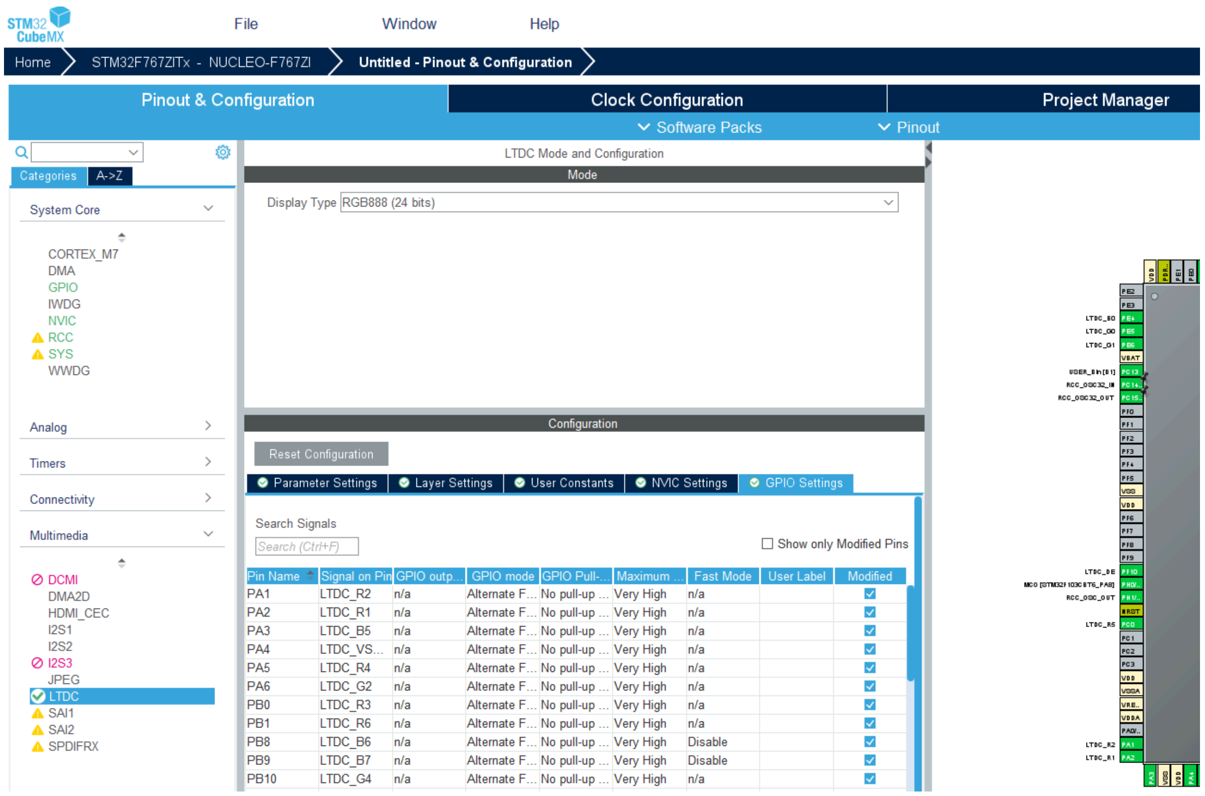
E70_Test2.elf - /E70_Test2/Debug - Dec 15, 2022, 4:00:03 PM

Region	Start address	End address	Size	Free	Used	Usage (%)
RAM	0x20000000	0x20080000	512 KB	210.29 KB	301.71 KB	58.93%
FLASH	0x08000000	0x08200000	2 MB	1.99 MB	11.52 KB	0.56%

Figure 3. STM32CubeIDE Development Tool

4. Determine Framebuffer Size and Location

The Nucleo board has an STM32F767ZI MCU with 512KB of internal static RAM (SRAM). The SRAM is divided into 128KB of tightly coupled memory for data (DTCM), 16KB of tightly coupled memory for instructions (ITCM), 4KB of backup SRAM, and 364KB of user application SRAM. Framebuffer memory requirements are calculated by width (in pixels) × height (in lines) × internal color bit depth/bits per byte. Internal color bit depth can be represented in differing bit depths depending on selections made in the STM32 development tools.

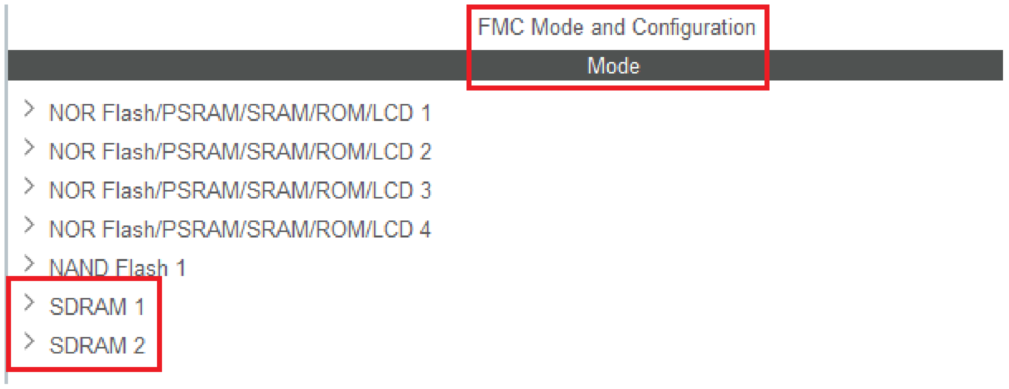


Display Type RGB888 (24 bits)

Pin Name	Signal on Pin	GPIO outp...	GPIO mode	GPIO Pull-...	Maximum ...	Fast Mode	User Label	Modified
PA1	LTDC_R2	n/a	Alternate F...	No pull-up ...	Very High	n/a		✓
PA2	LTDC_R1	n/a	Alternate F...	No pull-up ...	Very High	n/a		✓
PA3	LTDC_B5	n/a	Alternate F...	No pull-up ...	Very High	n/a		✓
PA4	LTDC_VS...	n/a	Alternate F...	No pull-up ...	Very High	n/a		✓
PA5	LTDC_R4	n/a	Alternate F...	No pull-up ...	Very High	n/a		✓
PA6	LTDC_G2	n/a	Alternate F...	No pull-up ...	Very High	n/a		✓
PB0	LTDC_R3	n/a	Alternate F...	No pull-up ...	Very High	n/a		✓
PB1	LTDC_R6	n/a	Alternate F...	No pull-up ...	Very High	n/a		✓
PB8	LTDC_B6	n/a	Alternate F...	No pull-up ...	Very High	Disable		✓
PB9	LTDC_B7	n/a	Alternate F...	No pull-up ...	Very High	Disable		✓
PB10	LTDC_G4	n/a	Alternate F...	No pull-up ...	Very High	n/a		✓

Figure 4. LTDC Mode and Configuration

It is advisable to select an MCU with a large internal SRAM, a large internal dynamic RAM (DRAM), or an external memory controller to contain an entire frame buffer. The STM32 family of MCUs has a range of devices with flexible memory controllers (FMC). It is left to the end user to determine if external memories are applicable.


FMC Mode and Configuration

Mode

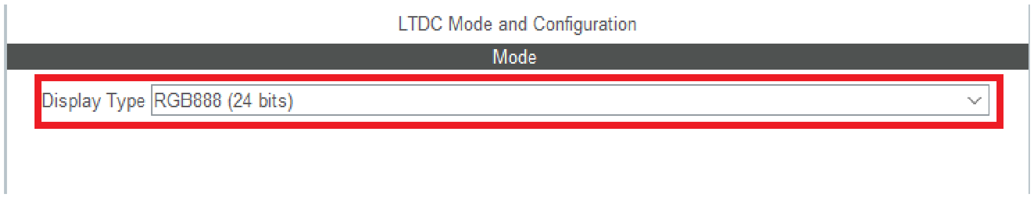
- NOR Flash/PSRAM/SRAM/ROM/LCD 1
- NOR Flash/PSRAM/SRAM/ROM/LCD 2
- NOR Flash/PSRAM/SRAM/ROM/LCD 3
- NOR Flash/PSRAM/SRAM/ROM/LCD 4
- NAND Flash 1
- SDRAM 1
- SDRAM 2

Figure 5. FMC Modes

In the case of this MCU the size and bit depth of the framebuffer has been chosen to be: (320 × 240 × 32) / 8, for a total buffer memory requirement of 307KB. The internal format of the color data is 24 bits with 8 bits of alpha information. This represents approximately 60% of the usable internal SRAM.

5. Configuration

Create a project in STM32CubeMX for the chosen MCU board. Select the LTDC peripheral in the Pinout tab and under Mode choose RGB888 (24-bit).



LTDC Mode and Configuration

Mode

Display Type RGB888 (24 bits)

Figure 6. LTDC Mode Display Type

5.1 LTDC and GPIO Settings

The user can configure all the GPIO pins related to the LTDC from this tab. If the GPIOs used do not match the display panel connection to the MCU, the user can change the GPIO alternate pin functions. This app note will use the default GPIO configuration of the LTDC.

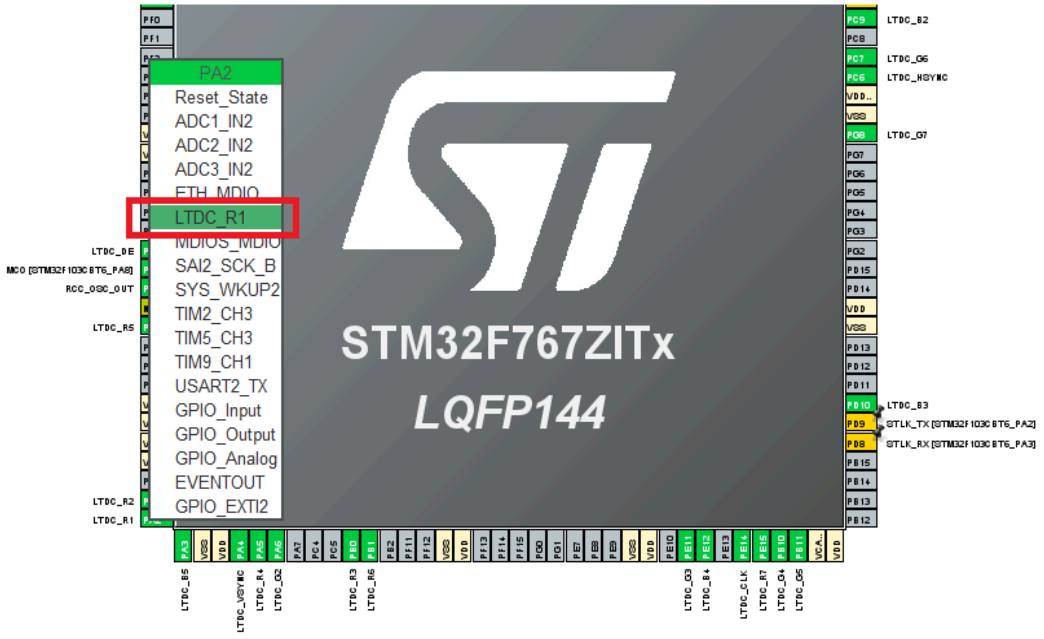
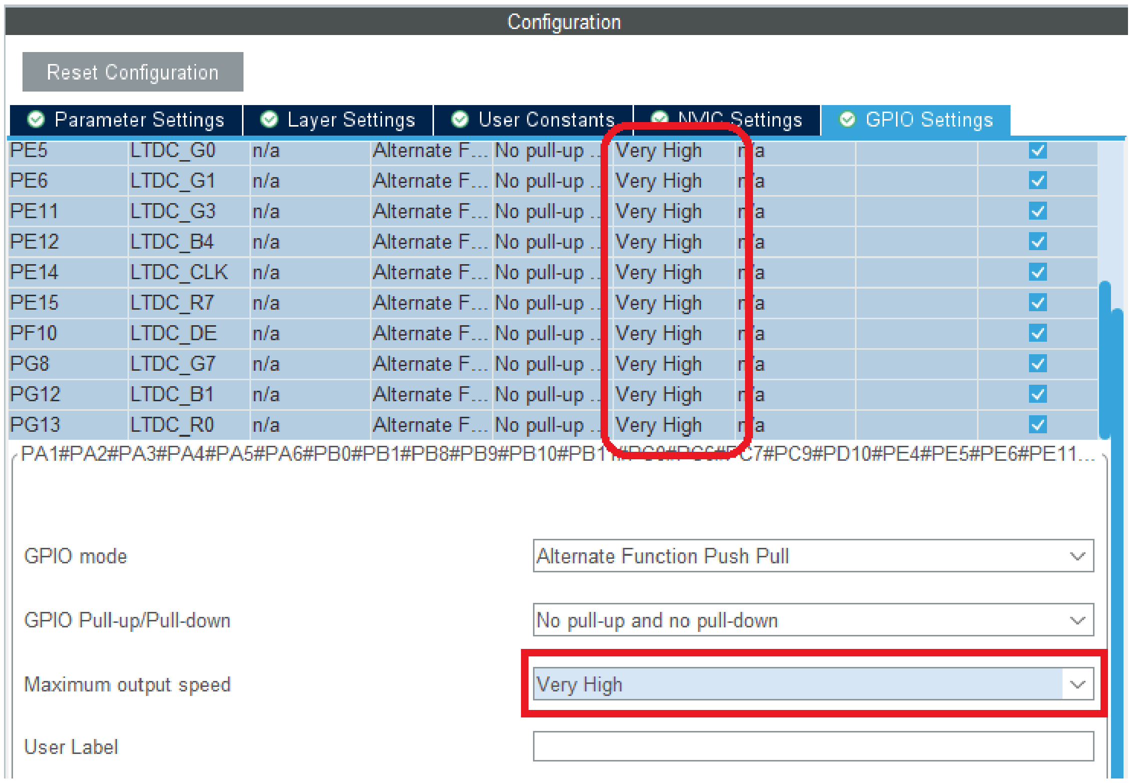


Figure 7. LTDC GPIO Pin Configuration

Once the LTDC GPIO is configured correctly the pins used are highlighted green.

To set the speed of the GPIO, select the GPIO Settings tab under Configuration. Highlight all the GPIOs and set the “Maximum Output Speed” to “Very High”.



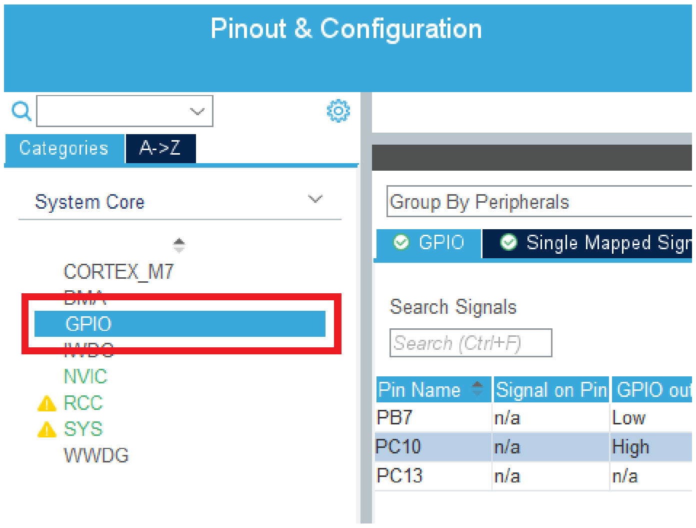
PE5	LTDC_G0	n/a	Alternate F...	No pull-up ...	Very High	n/a		✓
PE6	LTDC_G1	n/a	Alternate F...	No pull-up ...	Very High	n/a		✓
PE11	LTDC_G3	n/a	Alternate F...	No pull-up ...	Very High	n/a		✓
PE12	LTDC_B4	n/a	Alternate F...	No pull-up ...	Very High	n/a		✓
PE14	LTDC_CLK	n/a	Alternate F...	No pull-up ...	Very High	n/a		✓
PE15	LTDC_R7	n/a	Alternate F...	No pull-up ...	Very High	n/a		✓
PF10	LTDC_DE	n/a	Alternate F...	No pull-up ...	Very High	n/a		✓
PG8	LTDC_G7	n/a	Alternate F...	No pull-up ...	Very High	n/a		✓
PG12	LTDC_B1	n/a	Alternate F...	No pull-up ...	Very High	n/a		✓
PG13	LTDC_R0	n/a	Alternate F...	No pull-up ...	Very High	n/a		✓

PA1#PA2#PA3#PA4#PA5#PA6#PB0#PB1#PB8#PB9#PB10#PB11#PC0#PC6#PC7#PC9#PD10#PE4#PE5#PE6#PE11...

GPIO mode	Alternate Function Push Pull
GPIO Pull-up/Pull-down	No pull-up and no pull-down
Maximum output speed	Very High
User Label	

Figure 8. LTDC GPIO Settings

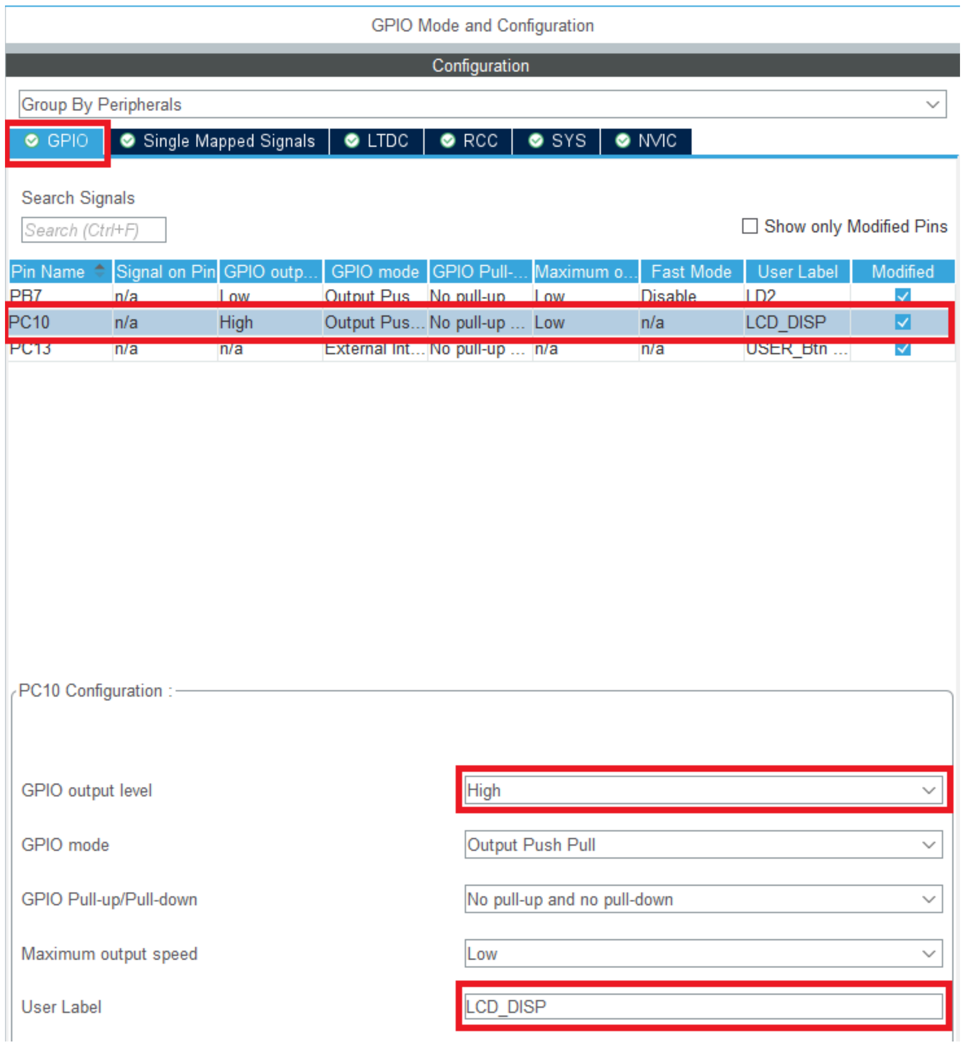
Once all the RGB interface pins are configured, the user must configure display-specific pins from the GPIO section of the Pinout tab.



Pin Name	Signal on Pin	GPIO out
PB7	n/a	Low
PC10	n/a	High
PC13	n/a	n/a

Figure 9. Display-Specific GPIO

PC10 on the MCU board is used for controlling the LCD_DISP functionality. The "GPIO Output Level" on PC10 should be set to High. Under User Label name the pin LCD_DISP. No other pins are needed for this display.


GPIO Mode and Configuration

Configuration

Group By Peripherals

GPIO | Single Mapped Signals | LTDC | RCC | SYS | NVIC

Search Signals

Search (Ctrl+F)

☐ Show only Modified Pins

Pin Name	Signal on Pin	GPIO outp...	GPIO mode	GPIO Pull-...	Maximum o...	Fast Mode	User Label	Modified
PB7	n/a	Low	Output Pus...	No pull-up ...	Low	Disable	LD2	✓
PC10	n/a	High	Output Pus...	No pull-up ...	Low	n/a	LCD_DISP	✓
PC13	n/a	n/a	External Int...	No pull-up ...	n/a	n/a	USER_Btn ...	✓

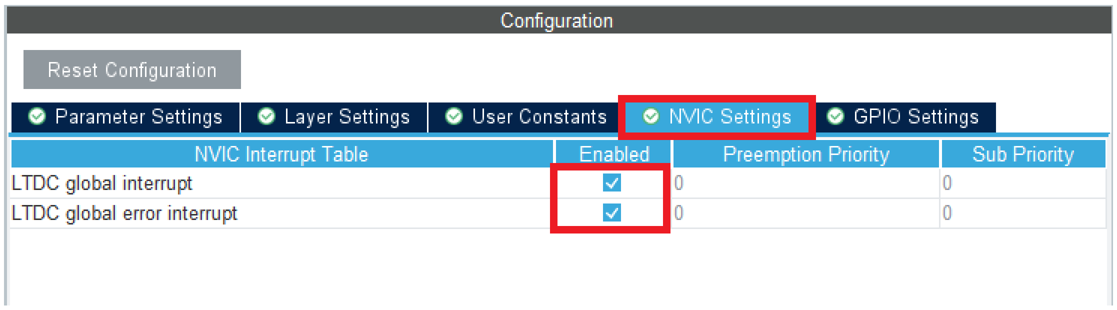
PC10 Configuration :

GPIO output level	High
GPIO mode	Output Push Pull
GPIO Pull-up/Pull-down	No pull-up and no pull-down
Maximum output speed	Low
User Label	LCD_DISP

Figure 10. PC10 Configuration

5.2 LTDC Interrupts

Global LTDC interrupts need to be enabled for proper operation of the peripheral. To enable the interrupts, under LTDC Mode and Configuration select the NVIC Settings tab. Check the two boxes for LTDC global interrupts as shown in the image below.


Configuration

Reset Configuration

Parameter Settings | Layer Settings | User Constants | NVIC Settings | GPIO Settings

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
LTDC global interrupt	✓	0	0
LTDC global error interrupt	✓	0	0

Figure 11. LTDC Interrupt Settings

5.3 Clock and Timing Configuration

5.3.1 System Clock Configuration

The system clock configuration is setup with the following parameters:

- Use of external HSE oscillator with the main PLL used as the system source clock
- HCLK set to 200MHz (core MCU and LTDC peripheral clock)

To configure the system clock, select the Clock Configuration tab and set it as shown in the image below.

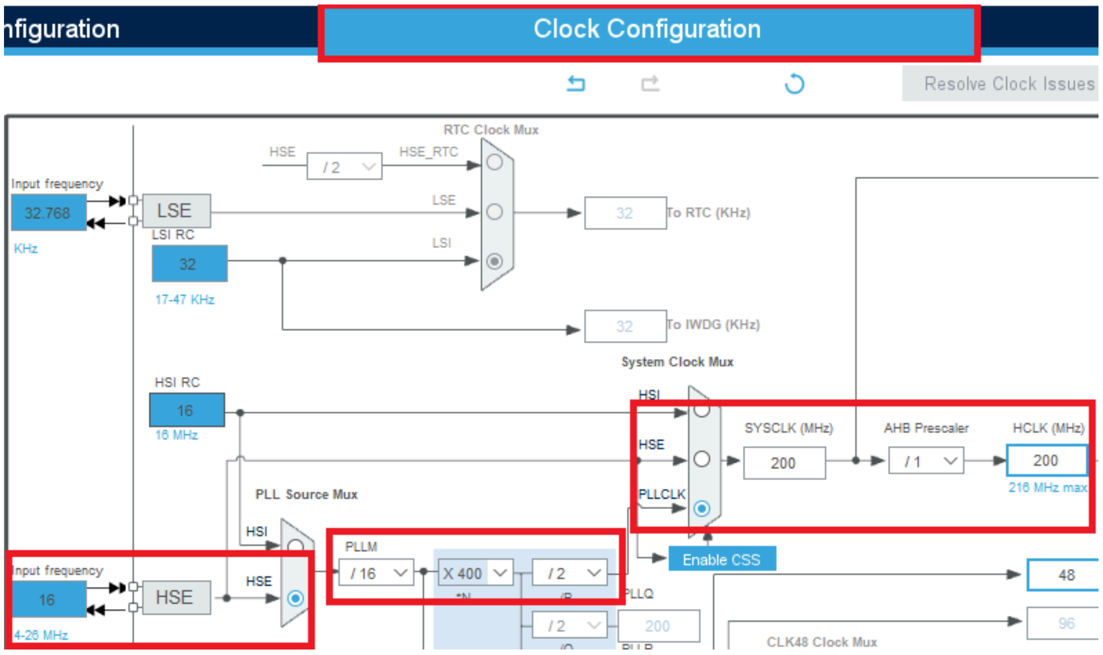
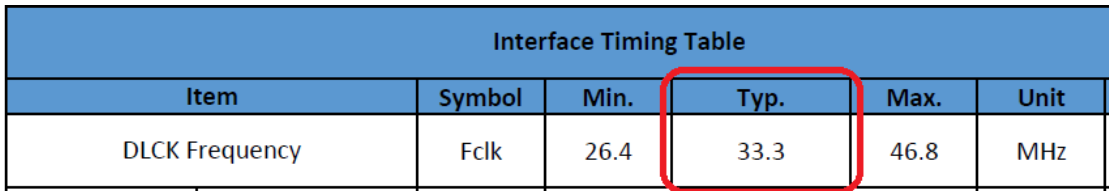


Figure 12. System Clock Settings

5.3.2 Pixel Clock Configuration

The Pixel clock, or DCLK, is provided in the display datasheet.



Interface Timing Table					
Item	Symbol	Min.	Typ.	Max.	Unit
DLCK Frequency	Fclk	26.4	33.3	46.8	MHz

Figure 13. Pixel Clock (DCLK) Settings

To configure the DCLK to 33.3MHz, select the Clock Configuration tab, and set the PLLSAI 1 to the parameters shown below. Verify that "To LCD-TFT (MHz)" displays 33.333333.

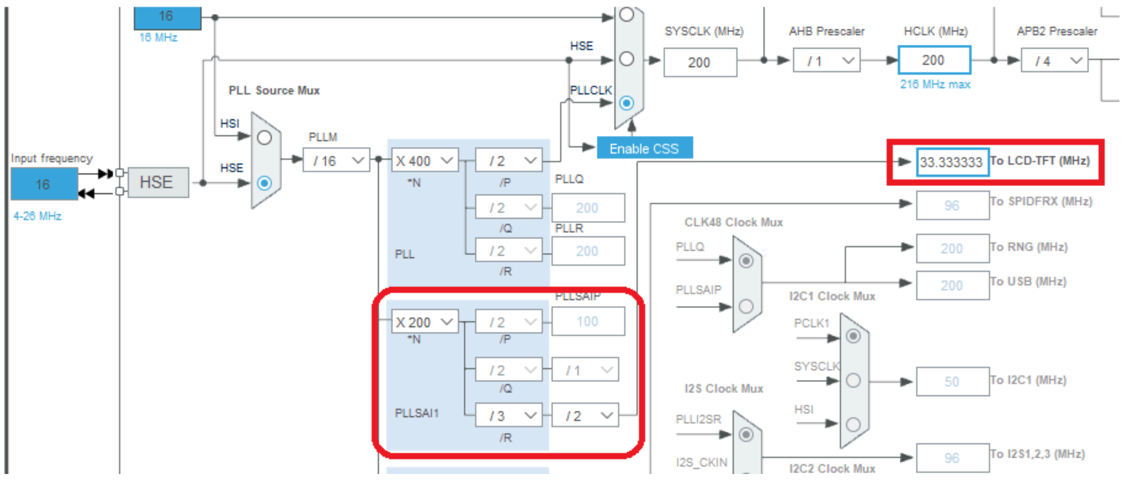
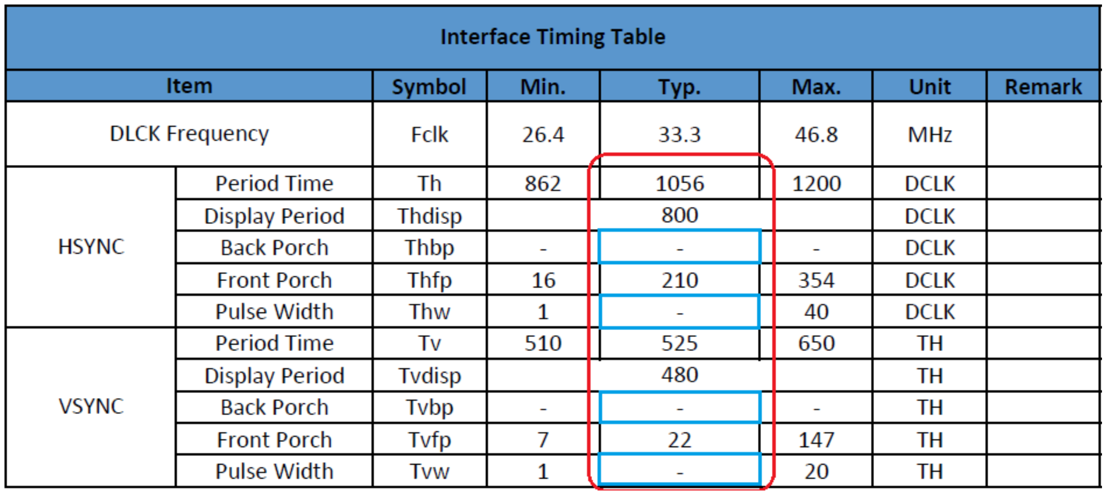


Figure 14. Pixel Clock Settings

5.3.3 Display Timing

The user must refer to the display datasheet to configure the display timing. In the Pinout & Configuration tab, under the LTDC, click the Parameter Settings tab. Fill in the timing parameters from the datasheet data.

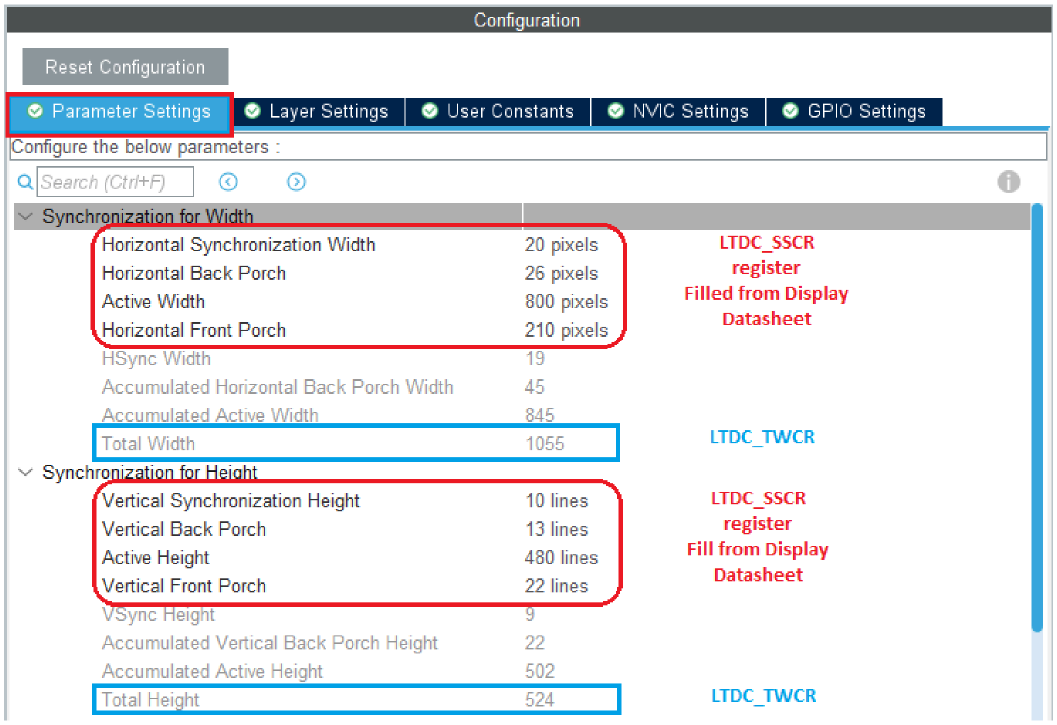


Interface Timing Table							
Item		Symbol	Min.	Typ.	Max.	Unit	Remark
DLCK Frequency		Fclk	26.4	33.3	46.8	MHz	
HSYNC	Period Time	Th	862	1056	1200	DCLK	
	Display Period	Thdisp		800		DCLK	
	Back Porch	Thbp	-	-	-	DCLK	
	Front Porch	Thfp	16	210	354	DCLK	
	Pulse Width	Thw	1	-	40	DCLK	
VSYNC	Period Time	Tv	510	525	650	TH	
	Display Period	Tvdisp		480		TH	
	Back Porch	Tvbp	-	-	-	TH	
	Front Porch	Tvfp	7	22	147	TH	
	Pulse Width	Tvw	1	-	20	TH	

Figure 15. Datasheet Interface Timing Table

Some of the parameters on the table do not have typical values. Specifically, the pulse width and back porch on both horizontal and vertical. This application set the horizontal pulse width at 20 DCLKs and the vertical at 10 TH. There are no values given for the back porch on either axis. The back porch was calculated by taking the Period Time – (Display Period + Front Porch + Pulse Width). This resulted in 26 DCLKs for the horizontal back porch and 13 lines for the vertical back porch.

The figure below shows the settings for width and height.



Synchronization for Width

Horizontal Synchronization Width	20 pixels
Horizontal Back Porch	26 pixels
Active Width	800 pixels
Horizontal Front Porch	210 pixels
HSync Width	19
Accumulated Horizontal Back Porch Width	45
Accumulated Active Width	845
Total Width	1055

Synchronization for Height

Vertical Synchronization Height	10 lines
Vertical Back Porch	13 lines
Active Height	480 lines
Vertical Front Porch	22 lines
VSync Height	9
Accumulated Vertical Back Porch Height	22
Accumulated Active Height	502
Total Height	524

LTDC_SSCR register Filled from Display Datasheet

LTDC_TWCR

LTDC_SSCR register Fill from Display Datasheet

LTDC_TWCR

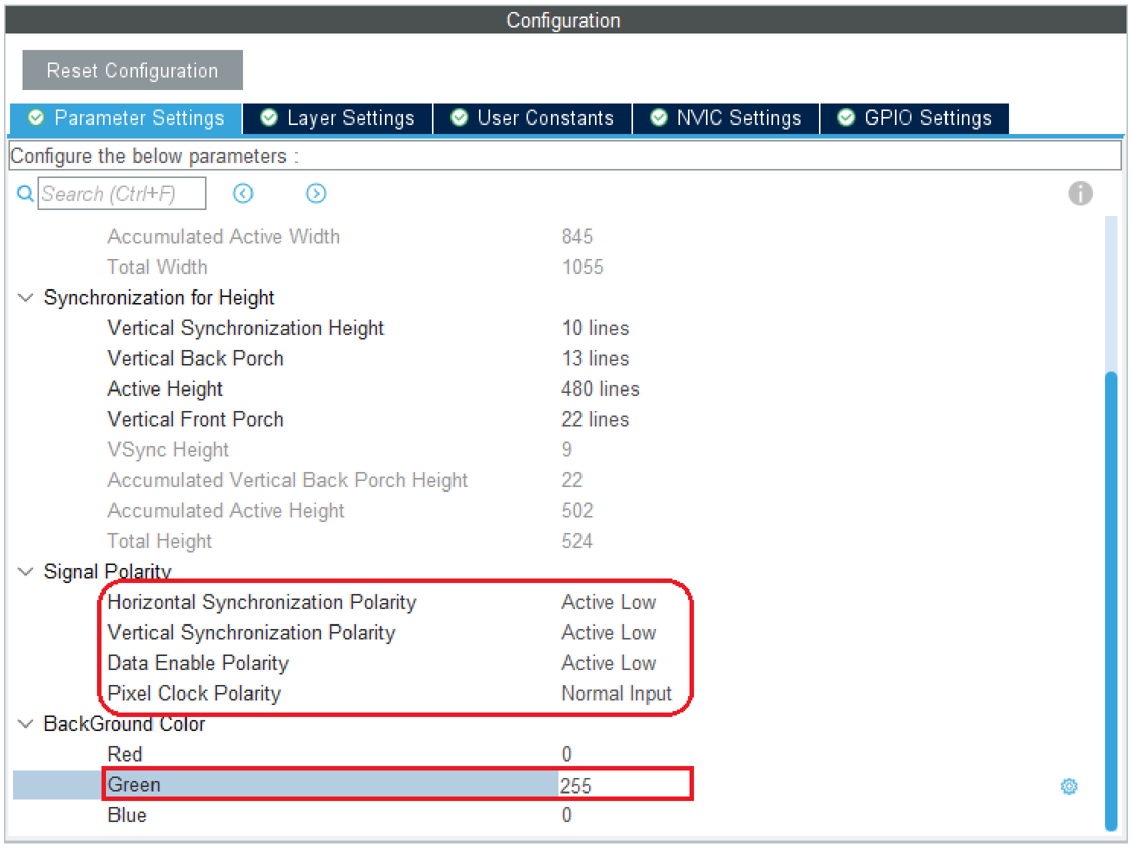
Figure 16. Width and Height Settings

- Horizontal Synchronization Width is 20 pixels
- Horizontal Back Porch is 26 pixels
- Active Width is 800 pixels
- Horizontal Front Porch is 210 pixels
- Vertical Synchronization Height is 10 lines
- Vertical Back Porch is 13 lines
- Active Height is 480 lines
- Vertical Front Porch is 22 lines

The total width is 1055 (1056 from datasheet -1 due to MX tool calculation) and the total height is 524 (525 from datasheet -1 due to MX tool calculation).

5.4 Signal Polarity and Background Color

Referring to the display datasheet, the HSYNC and VSYNC must be active low, and the display enable (DE) signal must be active high. Since the DE signal is inverted in the output it must be set to active low. DCLK must be set to active high. See the image below.



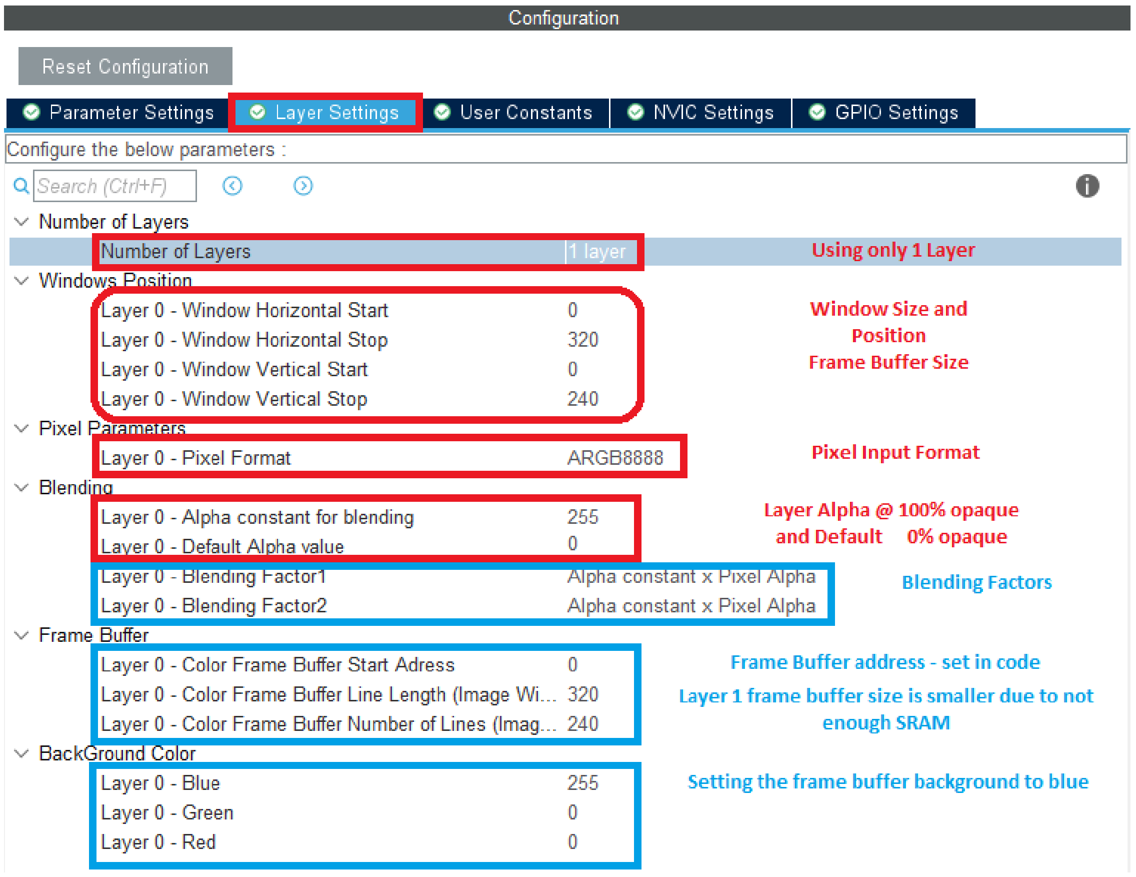
Accumulated Active Width	845
Total Width	1055
Synchronization for Height	
Vertical Synchronization Height	10 lines
Vertical Back Porch	13 lines
Active Height	480 lines
Vertical Front Porch	22 lines
VSync Height	9
Accumulated Vertical Back Porch Height	22
Accumulated Active Height	502
Total Height	524
Signal Polarity	
Horizontal Synchronization Polarity	Active Low
Vertical Synchronization Polarity	Active Low
Data Enable Polarity	Active Low
Pixel Clock Polarity	Normal Input
BackGround Color	
Red	0
Green	255
Blue	0

Figure 17. Signal Polarity and Background Color Settings

The background color has been set to 100% green. This was done for testing the display so that black or white does not display. If the screen came up black it could mean the backlight was not powered correctly. If the display came up white it could mean that the screen is not working. Both are error conditions to be fixed.

5.5 LTDC Layer Parameters

All the LTDC clock and timing parameters have been configured at this stage. Now the user must configure the layer parameters according to the display size. Frame buffer size, and color depth. Under the LTDC Configuration window, select the Layer Settings tab and follow the figure below.



Number of Layers
- Number of Layers: 1 layer — **Using only 1 Layer**

Windows Position
- Layer 0 - Window Horizontal Start: 0
- Layer 0 - Window Horizontal Stop: 320
- Layer 0 - Window Vertical Start: 0
- Layer 0 - Window Vertical Stop: 240

Window Size and Position Frame Buffer Size

Pixel Parameters
- Layer 0 - Pixel Format: ARGB8888 — **Pixel Input Format**

Blending
- Layer 0 - Alpha constant for blending: 255
- Layer 0 - Default Alpha value: 0

Layer Alpha @ 100% opaque and Default 0% opaque

- Layer 0 - Blending Factor1: Alpha constant x Pixel Alpha
- Layer 0 - Blending Factor2: Alpha constant x Pixel Alpha

Blending Factors

Frame Buffer
- Layer 0 - Color Frame Buffer Start Adress: 0
- Layer 0 - Color Frame Buffer Line Length (Image Wi...: 320
- Layer 0 - Color Frame Buffer Number of Lines (Imag...: 240

Frame Buffer address - set in code
Layer 1 frame buffer size is smaller due to not enough SRAM

BackGround Color
- Layer 0 - Blue: 255
- Layer 0 - Green: 0
- Layer 0 - Red: 0

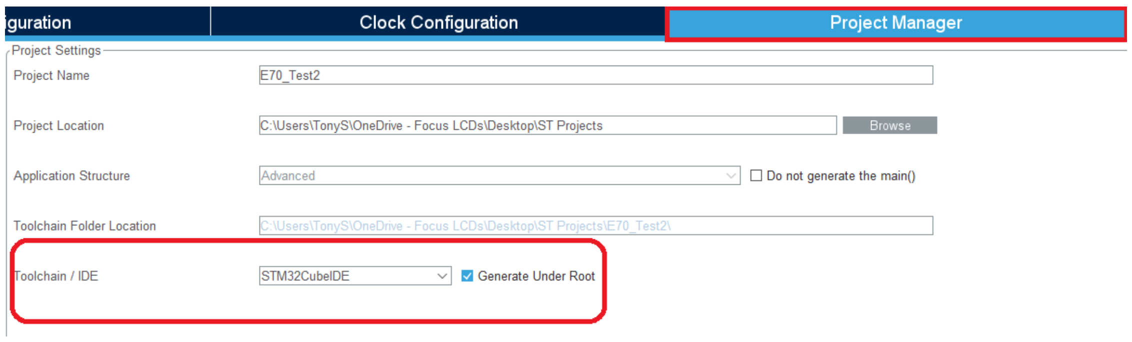
Setting the frame buffer background to blue

Figure 18. LTDC Layer Configuration

- Using only 1 layer configuration
- The window size is set to the same size as the frame buffer
- The pixel format is set to ARGB888 since there is a background and layer 1 to Alpha blend
- ARGB888 is set for 24-bit color format (bit depth)
- Setting the Alpha constant to 255 will display the background (green) with the frame buffer window (blue) over the background
- Setting the background layer color to blue is for testing that both the background and the layer are active and displaying correctly

5.6 Finalize the Configuration

There is one last setting to change before generating the source code. Under Project Manager click on the drop-down menu for Toolchain/IDE and set it for STM32CubeIDE.



Project Settings
Project Name: E70_Test2
Project Location: C:\Users\TonyS\OneDrive - Focus LCDs\Desktop\ST Projects
Application Structure: Advanced ☐ Do not generate the main()
Toolchain Folder Location: C:\Users\TonyS\OneDrive - Focus LCDs\Desktop\ST Projects\E70_Test2\
Toolchain / IDE: STM32CubeIDE ☑ Generate Under Root

Figure 19. Project Manager - Toolchain Selection

Now the Generate Code button can be clicked.

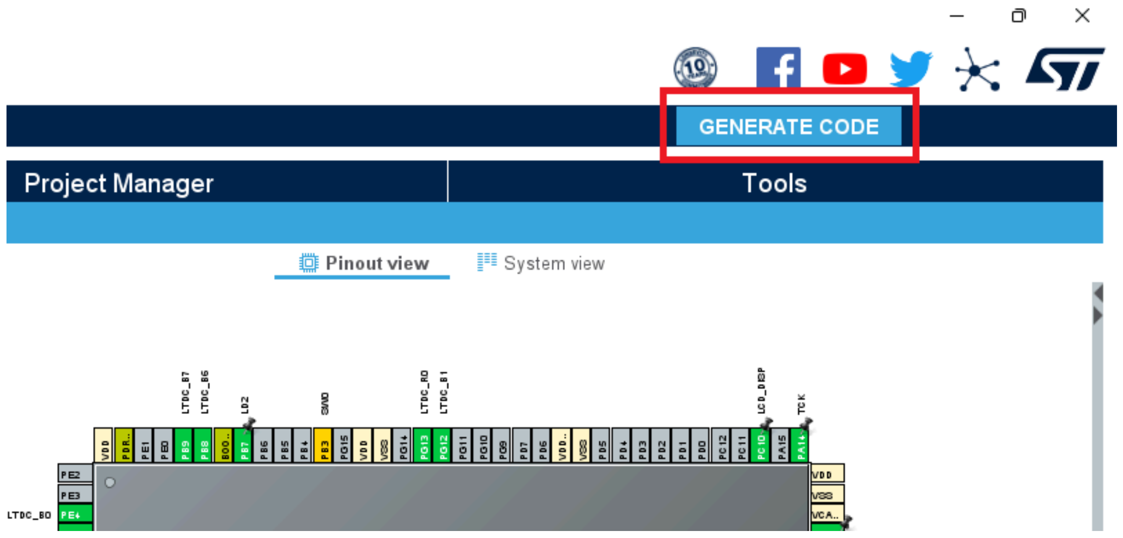
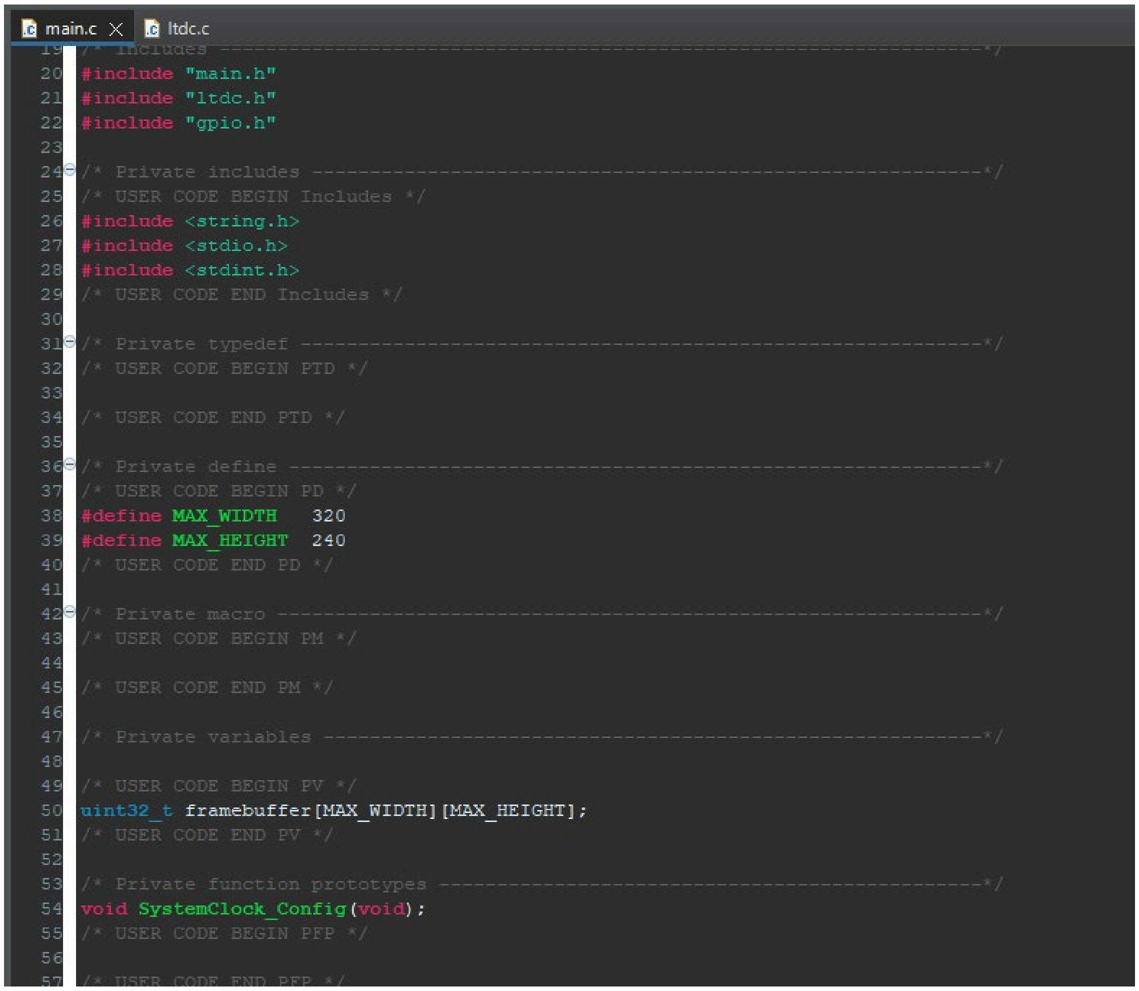


Figure 20. Code Generation

Once the process is complete the tool will ask if it should open the IDE. Select OK.

6. Firmware

Once the STM32CubeIDE (IDE) is open there is some code to add to the main.c file.



```
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "ltdc.h"
#include "gpio.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include <string.h>
#include <stdio.h>
#include <stdint.h>
/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
#define MAX_WIDTH   320
#define MAX_HEIGHT  240
/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/

/* USER CODE BEGIN PV */
uint32_t framebuffer[MAX_WIDTH][MAX_HEIGHT];
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */
```

Figure 21. Includes, Defines, and Framebuffer Code

Include the `stdint.h` header file for the `uint32_t` data type. Next, the `MAX_WIDTH` and `MAX_HEIGHT` must be defined. Now the frame buffer array can be declared:

```
uint32_t framebuffer[MAX_WIDTH][MAX_HEIGHT];
```

The frame buffer can now be filled; in this instance, the chosen color was blue. It is filled with an inner `for` loop for the height and an outer `for` loop for the width. The final piece is to set the address of the LTDC pointer to the actual frame buffer with:

```
HAL_LTDC_SetAddress(&hltdc, (int)framebuffer, LTDC_LAYER_1);
```

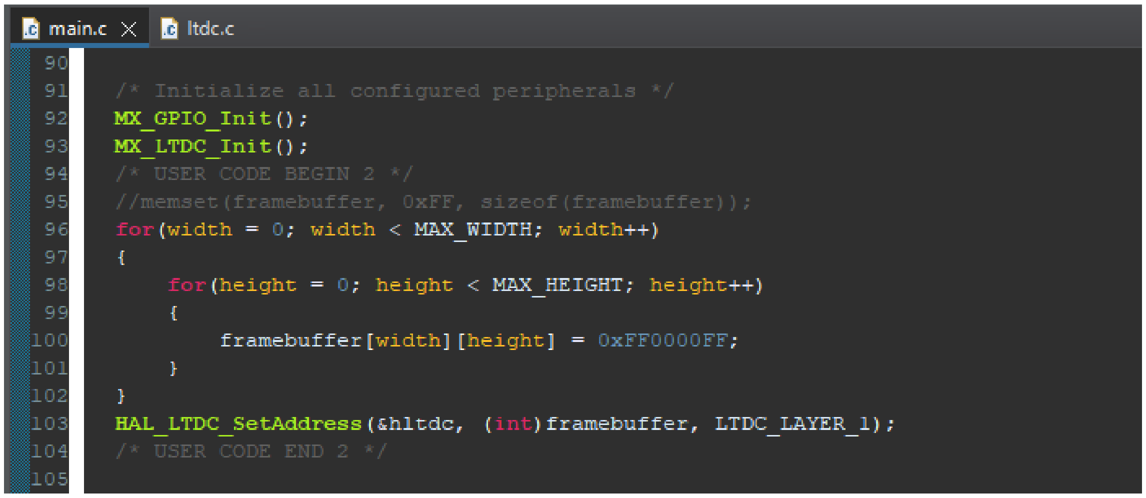

main.c ltdc.c
```
    /* Initialize all configured peripherals */
    MX_GPIO_Init();
    MX_LTDC_Init();
    /* USER CODE BEGIN 2 */
    //memset(framebuffer, 0xFF, sizeof(framebuffer));
    for(width = 0; width < MAX_WIDTH; width++)
    {
        for(height = 0; height < MAX_HEIGHT; height++)
        {
            framebuffer[width][height] = 0xFF0000FF;
        }
    }
    HAL_LTDC_SetAddress(&hltdc, (int)framebuffer, LTDC_LAYER_1);
    /* USER CODE END 2 */
```

Figure 22. Filling the Frame Buffer and Setting the Address

The finished code can be compiled and downloaded into the MCU board from the IDE.

The final display image is shown below.

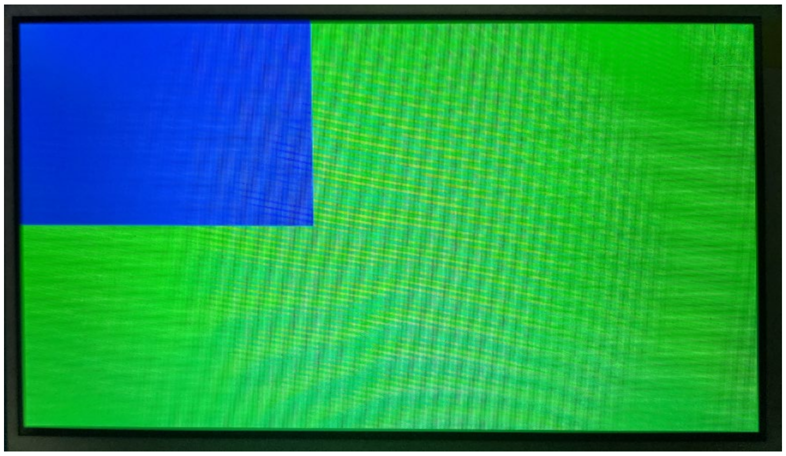


Figure 23. Image Displayed on LCD

7. Summary

In this application note, the development of firmware for driving the display was presented. The development tools included a code generator and an IDE. The LTDC peripheral of the Nucleo board was configured to drive the display with the timings provided in the datasheet. A size-limited frame buffer was defined in the configuration tool. Once all the parameters were configured the IDE was opened to add application code. Minimal code was developed for demonstration purposes only. The code was then compiled and downloaded into the Nucleo board, and the resultant display was presented.

Disclaimer