



Board Bring Up Introduction

Emil Damkjaer Petersen

Agenda

1 Introduction

2 Hardware Selection

3 Board Bring-Up

Goal of this presentation

- Provide information to help developers with selecting hardware for graphics applications
 - Preliminary considerations for how the application can have an impact on the needed hardware
 - Information on the hardware components that an embedded GUI solution consist of
- Guide developers in the Board Bring Up phase, thereby helping with setting up the necessary parts for an GUI solution and ensuring the drivers work

Introduction

Further reading

- More help and information can be found at the TouchGFX documentation site:
<https://support.touchgfx.com/>
- Slides in this presentation will refer to relevant documentation pages. Links will be in the lower right-hand corner of the slides
- The presentation will generally be related to the two sections:
Hardware Selection & Board Bring Up

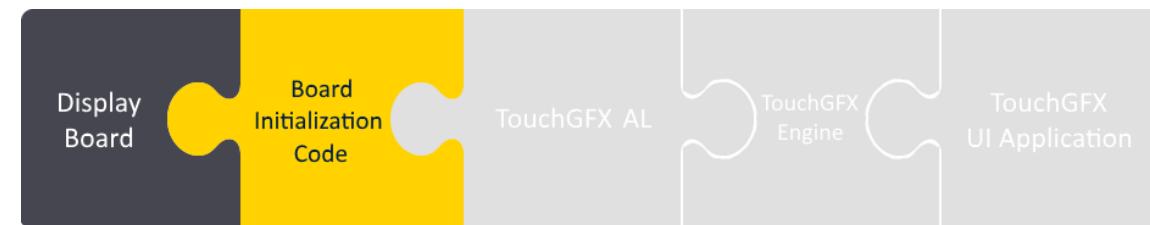
Introduction

TouchGFX Development

Main Activities



Main Component



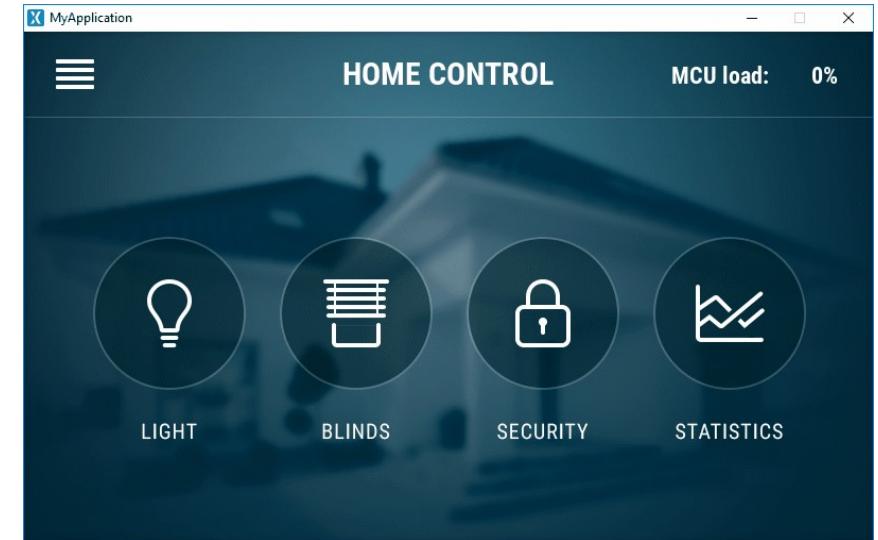
This presentation will start out with discussing the Main Activity, Hardware Selection, before it moves on to the activity Board Bring Up. Thereby we will learn how to select the hardware components needed for the Display Board, before moving on to the initialization of the selected hardware components, as a part of the Board Initialization Code, Main Component.

Hardware Selection

Hardware Selection

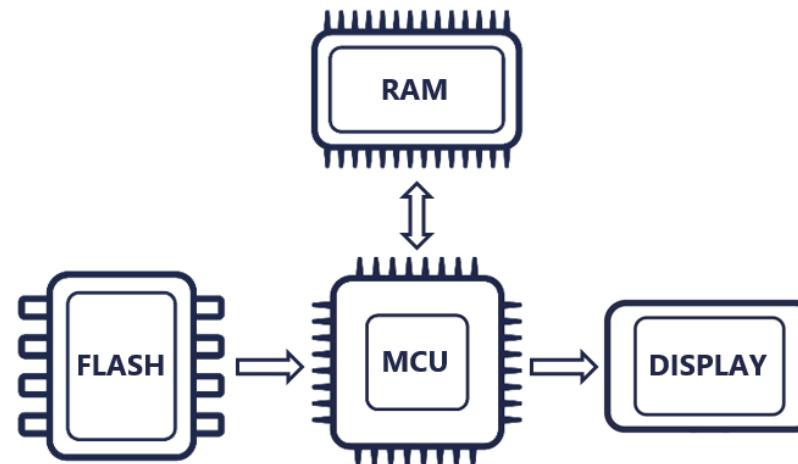
- Preliminary Considerations

- Application design have impact on the hardware
 - Display Resolution
 - Color Depth
 - Animations
 - Complexity
 - Size
 - Touch
- Mechanical Design Requirements
 - Size
 - Environment



Hardware Selection

- The basic hardware can be divided into 4 parts
 - MCU
 - Heavy Lifting
 - RAM
 - Framebuffer
 - Flash
 - Images, Fonts, Texts
 - Display

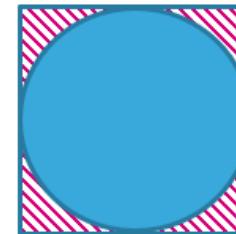


- More information can be found in the documentation:



Hardware Selection

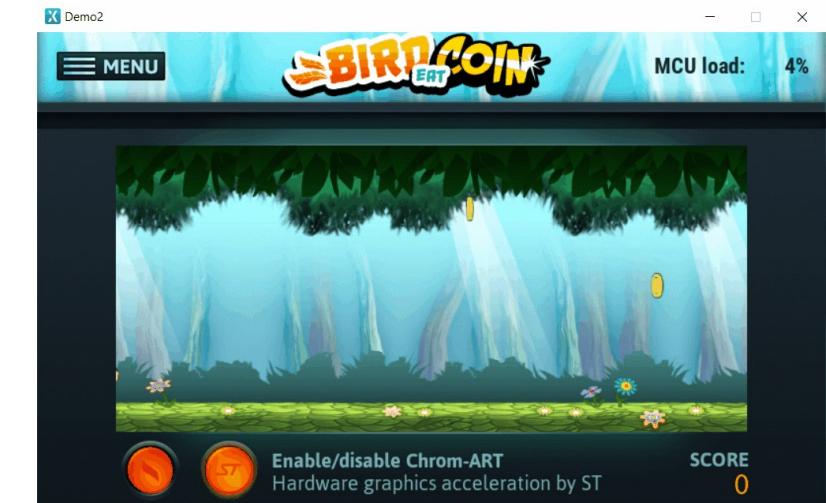
- MCU
 - Frequency
 - Embedded Hardware Acceleration
 - Chrom-ART
 - JPEG Accelerator
 - Chrom-GRC
 - Internal Memory
 - Ram
 - Flash
 - LCD Controller
 - Packages & I/O
 - Memory Interfacing



 Saved Memory

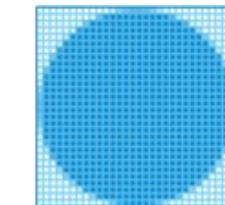
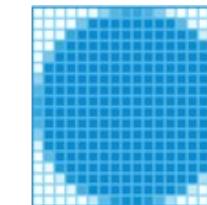
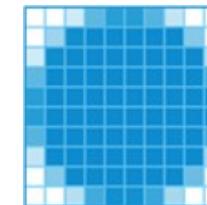
MCU PORTFOLIO FOR GRAPHICS

	STM32 SERIES	FREQUENCY	HARDWARE ACCELERATION	DISPLAY INTERFACES	SUPPORTED RESOLUTIONS
	STM32G0 (CM0+)	64 MHz		SPI	Up to 320*240
	STM32F7 (CM7)	216 MHz	Chrom-ART™ Hardware JPEG Codec	Parallel LCD TFT MIPI-DSI	Up to 1024*768



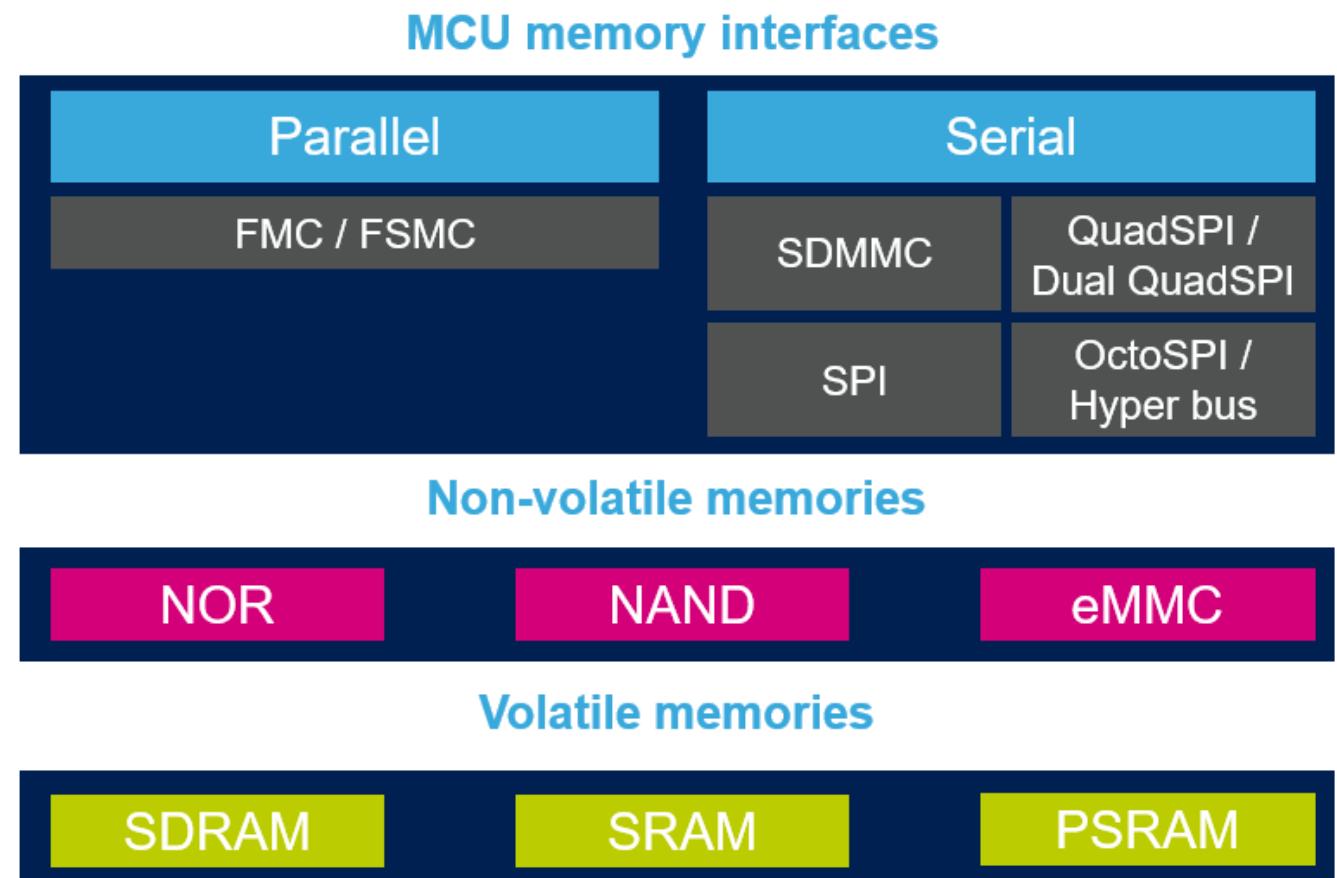
Hardware Selection

- Display
 - Different Types of Display
 - LCD-TFT
 - MIP
 - ePaper/eInk
 - Display Interface
 - Brightness and Backlight
 - Viewing Position and Color Inversion
 - Pixel density
 - Displays with RAM
 - Environment



Hardware Selection

- External Memory
 - MCU memory interface
 - Non-volatile Memories
 - NOR Flash
 - NAND Flash
 - eMMC
 - Volatile Memories
 - SDRAM
 - SRAM
 - PSRAM



Board Bring Up

Board Bring Up

- Introduction
 - Tools of the trade
 - STM32CubeMX
 - STM32Cube Firmware Package
 - Vendor datasheets
 - Vendor driver code
 - Verification of Functionality
 - Abstraction Layer
 - Test Code



Board Bring Up

- Introduction
 - How To Guide
 - Motivation
 - Goal
 - Prerequisites
 - Do

Step	Content
Create Project	Create an empty project in CubeMX
CPU Running	Ensure that the MCU is running at the desired speed
Framebuffer in internal RAM	Allocate a framebuffer in internal RAM and transmit it to the display
External RAM	Enable the external RAM
Framebuffer in external RAM	Move the framebuffer to external RAM and transmit it to the display
External addressable flash	Enable external memory-mapped flash
External block mode flash	Enable external block-mode flash
Hardware acceleration	Enable the Chrom-ART graphics accelerator
Touch controller	Setup communication to the touch controller
Physical buttons	Configure access to physical buttons
Flash loader	Develop a way to write data to the external flash

Board Bring Up

• 1. Create Project

- Motivation

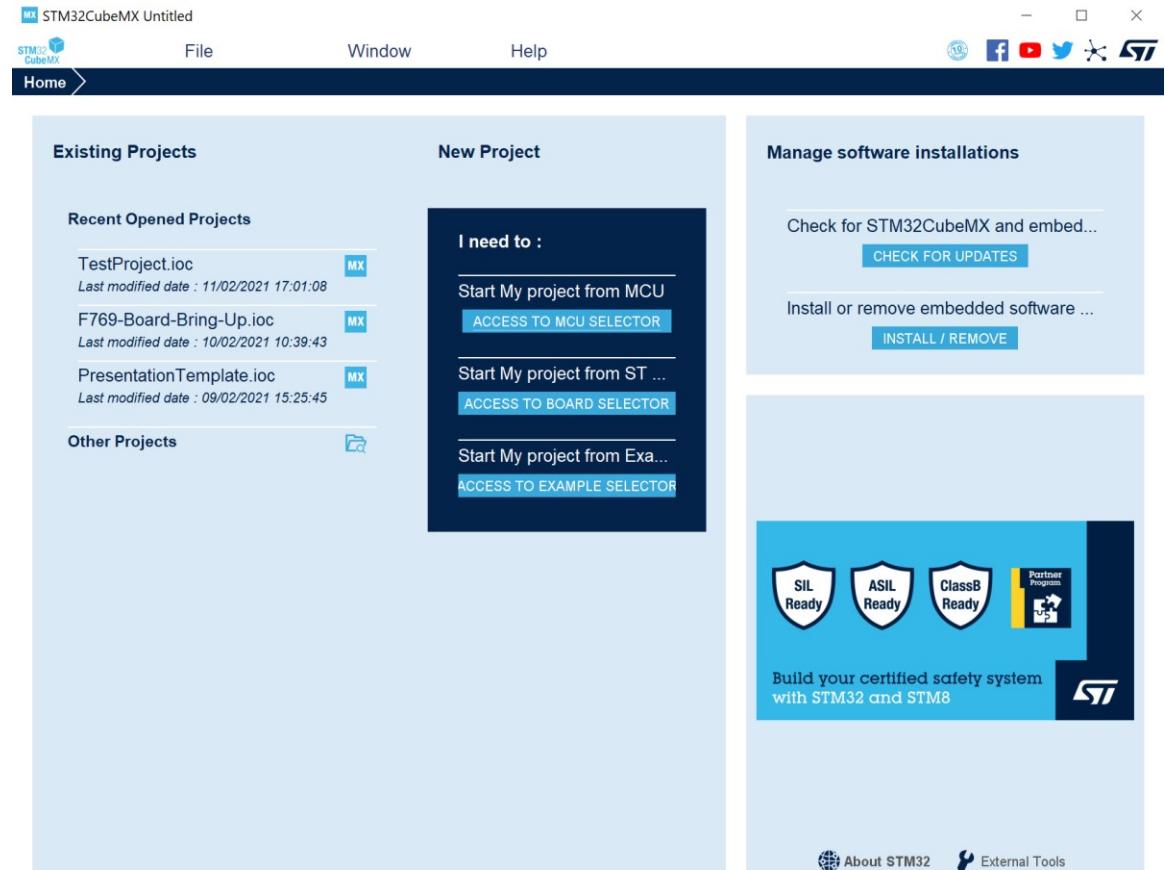
- Have a project in the chosen IDE and STM32CubeMX to be the basis for board bring up

- Goal

- Working STM32CubeMX project
- Able to debug code on the MCU

- Prerequisites

- STM32 Based Board
- Programming/Debugging interface
- STM32CubeMX
- IDE Installed



Board Bring Up

• 1. Create Project

- Do
 - Create a project in STM32CubeMX based on the MCU on the Board
 - Select “Advance” under application structure, and don’t select “Do not generate the main()”
 - Select Toolchain/IDE
 - Generate Code and when done, open project
 - Compile and debug from IDE

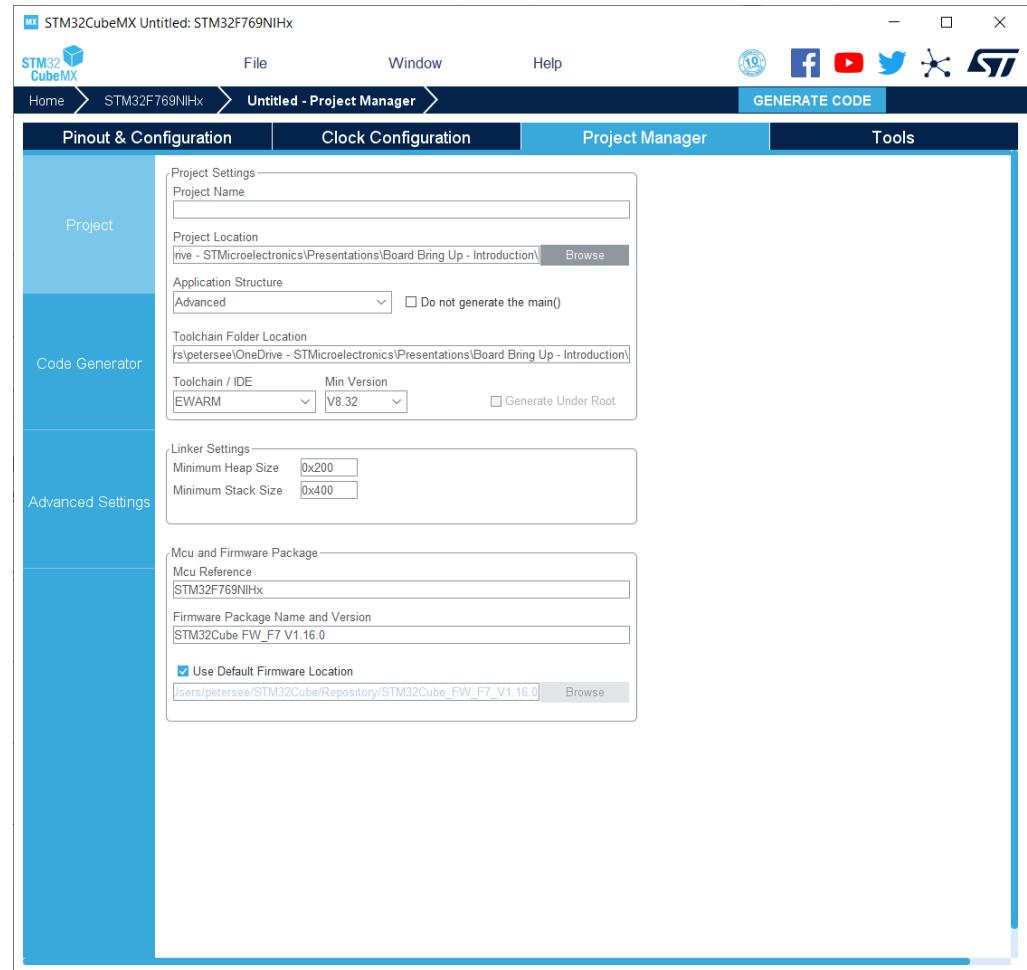
The screenshot shows the STM32CubeMX software interface for selecting an MCU. The top navigation bar includes tabs for 'MCU/MPU Selector' (which is active), 'Board Selector', 'Example Selector', and 'Cross Selector'. Below the navigation is a 'MCU/MPU Filters' section with dropdown menus for 'Core', 'Series', 'Line', 'Package', 'Other', and 'Peripheral'. To the right of the filters is a detailed product card for the 'STM32F769NI'. The card highlights it as a 'High-performance and DSP with FPU, Arm Cortex-M7 MCU with 2 Mbytes of Flash memory, 216 MHz CPU, Art Accelerator, L1 cache, SDRAM, TFT, MIPI-DSI, JPEG codec, DFSDM'. It shows an 'ACTIVE' status, a unit price of \$8.939, and mentions boards like STM32F769I-DISCO, STM32F769I-EVAL, and STM32F769I-EVAL. A small image of the TFBGA216 package is also shown. At the bottom of the card, it notes that the device is based on the high-performance Arm® Cortex® M7 32-bit RISC core operating at up to 216 MHz. Below the product card is a table titled 'MCUs/MPUs List: 1804 items' with columns for Part No., Reference, Mark..., Unit..., Board, Flash, RAM, IO, and Freq. The table lists various STM32F769 models with their respective specifications.

*	Part No.	Reference	Mark...	Unit...	Board	Flash	RAM	IO	Freq.
★	STM32F767VG	STM32F767VGHx	Active	6.577		1024 kB...512 kB...	82	216 MHz	
★	STM32F767VI	STM32F767VITx	Active	7.428		2048 kB...512 kB...	82	216 MHz	
★	STM32F767VII	STM32F767VIIHx	Active	7.428		2048 kB...512 kB...	82	216 MHz	
★	STM32F767ZG	STM32F767ZGTx	Active	7.109		1024 kB...512 kB...	114	216 MHz	
★	STM32F767ZI	STM32F767ZITx	Active	7.96		2048 kB...512 kB...	114	216 MHz	
★	STM32F769AI	STM32F769AIYx	Active	7.024		2048 kB...512 kB...	128	216 MHz	
★	STM32F769BG	STM32F769BGTx	Active	7.875		1024 kB...512 kB...	159	216 MHz	
★	STM32F769BI	STM32F769BITx	Active	8.726		2048 kB...512 kB...	159	216 MHz	
★	STM32F769IG	STM32F769IGTx	Active	7.449		1024 kB...512 kB...	131	216 MHz	
★	STM32F769II	STM32F769IITx	Active	8.301		2048 kB...512 kB...	131	216 MHz	
★	STM32F769NG	STM32F769NGHx	Active	8.088		1024 kB...512 kB...	159	216 MHz	
★	STM32F769NI	STM32F769NIHx	Active	8.939		2048 kB...512 kB...	159	216 MHz	
★	STM32F777BI	STM32F777BITx	Active	8.918		2048 kB...512 kB...	168	216 MHz	
★	STM32F777II	STM32F777IITx	Active	8.492		2048 kB...512 kB...	140	216 MHz	
★	STM32F777III	STM32F777IIIf	Active	8.102		2048 kB...512 kB...	140	216 MHz	

Board Bring Up

• 1. Create Project

- Do
 - Create a project in STM32CubeMX based on the MCU on the Board
 - Select “Advance” under application structure, and don’t select “Do not generate the main()”
 - Select Toolchain /IDE
 - Generate Code and when done, open project
 - Compile and debug from IDE



Board Bring Up

- 1. Create Project

- Do
 - Create a project in STM32CubeMX based on the MCU on the Board
 - Select “Advance” under application structure, and don’t select “Do not generate the main()”
 - Select Toolchain /IDE
 - Generate Code and when done, open project
 - Compile and debug from IDE

The screenshot shows the IAR Embedded Workbench IDE interface. The workspace contains a project named "TestProject" with files for Application, Drivers, and Output. The main.c file is open, displaying the following C code:

```
int main(void)
{
    /* USER CODE BEGIN 1 */
    /* USER CODE END 1 */

    /* MCU Configuration-----*/

    /* Reset of all peripherals, Initializes the Flash
HAL_Init();

    /* USER CODE BEGIN Init */
    /* USER CODE END Init */

    /* Configure the system clock */
SystemClock_Config();

    /* USER CODE BEGIN SysInit */
    /* USER CODE END SysInit */

    /* Initialize all configured peripherals */
    /* USER CODE BEGIN 2 */
    /* USER CODE END 2 */

    /* Infinite loop */
    /* USER CODE BEGIN WHILE */
while (1)
{
    /* USER CODE END WHILE */

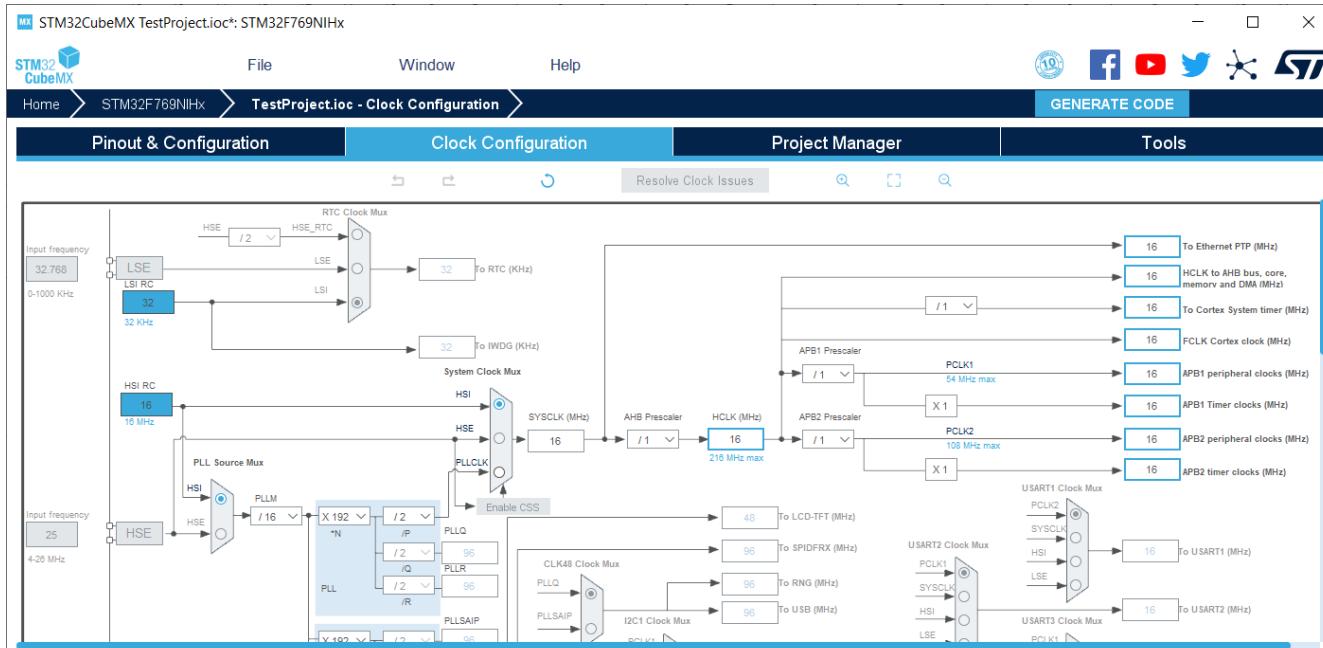
    /* USER CODE BEGIN 3 */
}
}
```

The Disassembly tab shows the assembly code generated from the C code. The Debug Log tab displays a log message indicating a hardware reset was performed on February 11, 2021, at 14:28:35.

Board Bring Up

• 2. CPU Running

- Motivation
 - Ensure that the MCU core, Internal RAM and flash is running a desired clock speed
- Goal
 - Set up clock to ensure correct speed
- Prerequisites
 - Information about clock on hardware
- Do
 - Setup clock in STM32CubeMX under the Clock Configuration
 - Check speed by debugging
- Note

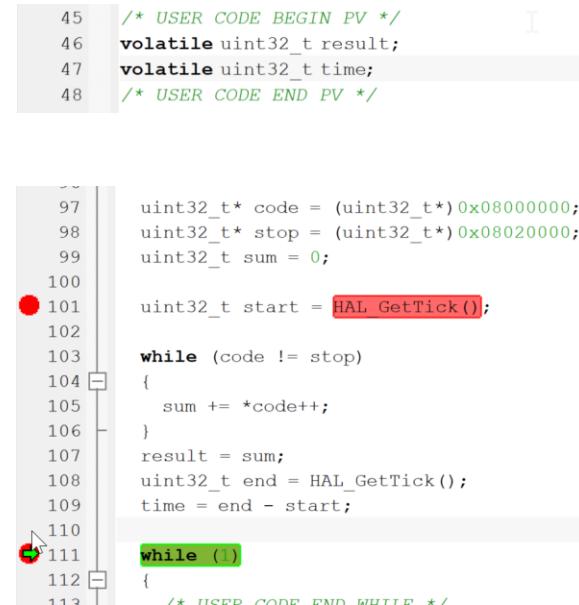


```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
HAL_Delay(5000);
while (1)
{
    /* USER CODE END WHILE */
```

Board Bring Up

- 2. CPU Running

- Motivation
 - Ensure that the MCU core, Internal RAM and flash is running a desired clock speed
- Goal
 - Set up clock to ensure correct speed
- Prerequisites
 - Information about clock on hardware
- Do
 - Setup clock in STM32CubeMX under the Clock Configuration
 - Check speed by debugging
- Note



```
45  /* USER CODE BEGIN PV */
46  volatile uint32_t result;
47  volatile uint32_t time;
48  /* USER CODE END PV */

97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
```

```
    uint32_t* code = (uint32_t*) 0x08000000;
    uint32_t* stop = (uint32_t*) 0x08020000;
    uint32_t sum = 0;

    uint32_t start = HAL_GetTick();

    while (code != stop)
    {
        sum += *code++;
    }
    result = sum;
    uint32_t end = HAL_GetTick();
    time = end - start;

    while (1)
    {
        /* USER CODE END PV */
    }
}
```

Board Bring Up

- 3. Framebuffer in internal RAM
 - Motivation
 - Setting up an internal framebuffer and the display, enables us to have simple graphic on the display
 - Goal
 - Having internal RAM allocated for the framebuffer, and show it's content on the display
 - Prerequisites
 - Information about the Display
 - Information about connection between the MCU and Display



Board Bring Up

- 3. Framebuffer in internal RAM
 - Do
 - Parallel RGB Displays
 - Setup framebuffer in internal RAM
 - Configure the GPIO connections to the display
 - Configure the LTDC controller
 - Configure the LTDC pixel clock
 - Setting the framebuffer address
 - Check the framerate

The screenshot shows a software interface for configuring an LTDC controller. At the top, there is a code editor window titled "main.c" containing the following code:

```
uint16_t framebuffer[480*272]; //16 bpp framebuffer
```

Below the code editor is a "LTDC Mode and Configuration" panel. It includes a dropdown for "Display Type" set to "RGB888 (24 bits)".

The main configuration area is titled "Configuration". It contains several tabs: "Parameter Settings", "Layer Settings", "User Constants", "NVC Settings", and "GPIO Settings".

On the left side of the interface is a navigation tree under the heading "Categories A-Z". The "LTDC" category is selected, showing its sub-components: PE4, P13, PJ14, PJ15, PG12, PK4, PK5, PK6, PK7, PJ14, PJ15, PJ7, PJ8, PJ9, PJ10, PJ11, PK0, PK1, PK2, PJ10, PJ15, PJ0, PJ1, PJ2, PJ3, PJ4, PJ5, PJ6, and PJ9.

At the bottom of the interface is a table titled "Search Signals" with a search bar. The table has columns for "Pin Name", "Signal on Pin", "GPIO output level", "GPIO mode", "GPIO Pull-up/Pull-down", "Maximum output speed", "User Label", and "Modified". The table lists numerous pins, each with a "Modified" checkbox checked.

[How To: 3. Framebuffer in internal RAM](#)

Board Bring Up

- 3. Framebuffer in internal RAM
 - Do
 - Parallel RGB Displays
 - Setup framebuffer in internal RAM
 - Configure the GPIO connections to the display
 - Configure the LTDC controller
 - Configure the LTDC pixel clock
 - Setting the framebuffer address
 - Check the framerate

LTDC Mode and Configuration

Mode

Display Type: RGB888 (24 bits)

Configuration

Reset Configuration

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

Configure the below parameters :

Search (Ctrl+F)

Number of Layers: 1 layer

Windows Position:

- Layer 0 - Window Horizontal Start: 0
- Layer 0 - Window Horizontal Stop: 480
- Layer 0 - Window Vertical Start: 0
- Layer 0 - Window Vertical Stop: 272

Pixel Parameters:

- Layer 0 - Pixel Format: RGB565

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:

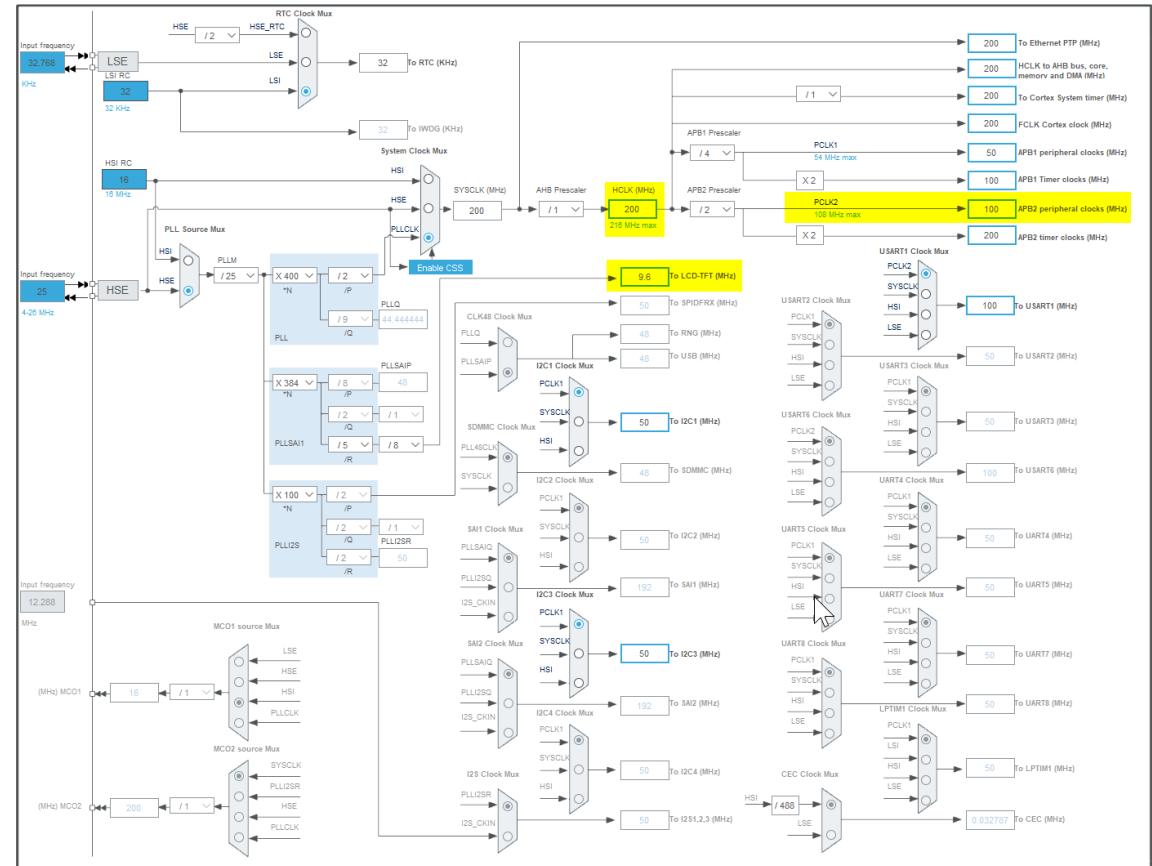
- Layer 0 - Color Frame Buffer Start Address: 0xC0000000
- Layer 0 - Color Frame Buffer Line Length (Image Width): 480
- Layer 0 - Color Frame Buffer Number of Lines (Image Height): 272

BackGround Color:

- Layer 0 - Blue: 0
- Layer 0 - Green: 0
- Layer 0 - Red: 0

Board Bring Up

- 3. Framebuffer in internal RAM
 - Do
 - Parallel RGB Displays
 - Setup framebuffer in internal RAM
 - Configure the GPIO connections to the display
 - Configure the LTDC controller
 - Configure the LTDC pixel clock
 - Setting the framebuffer address
 - Check the framerate



- 3. Framebuffer in internal RAM
 - Do
 - Parallel RGB Displays
 - Setup framebuffer in internal RAM
 - Configure the GPIO connections to the display
 - Configure the LTDC controller
 - Configure the LTDC pixel clock
 - Setting the framebuffer address
 - Check the framerate

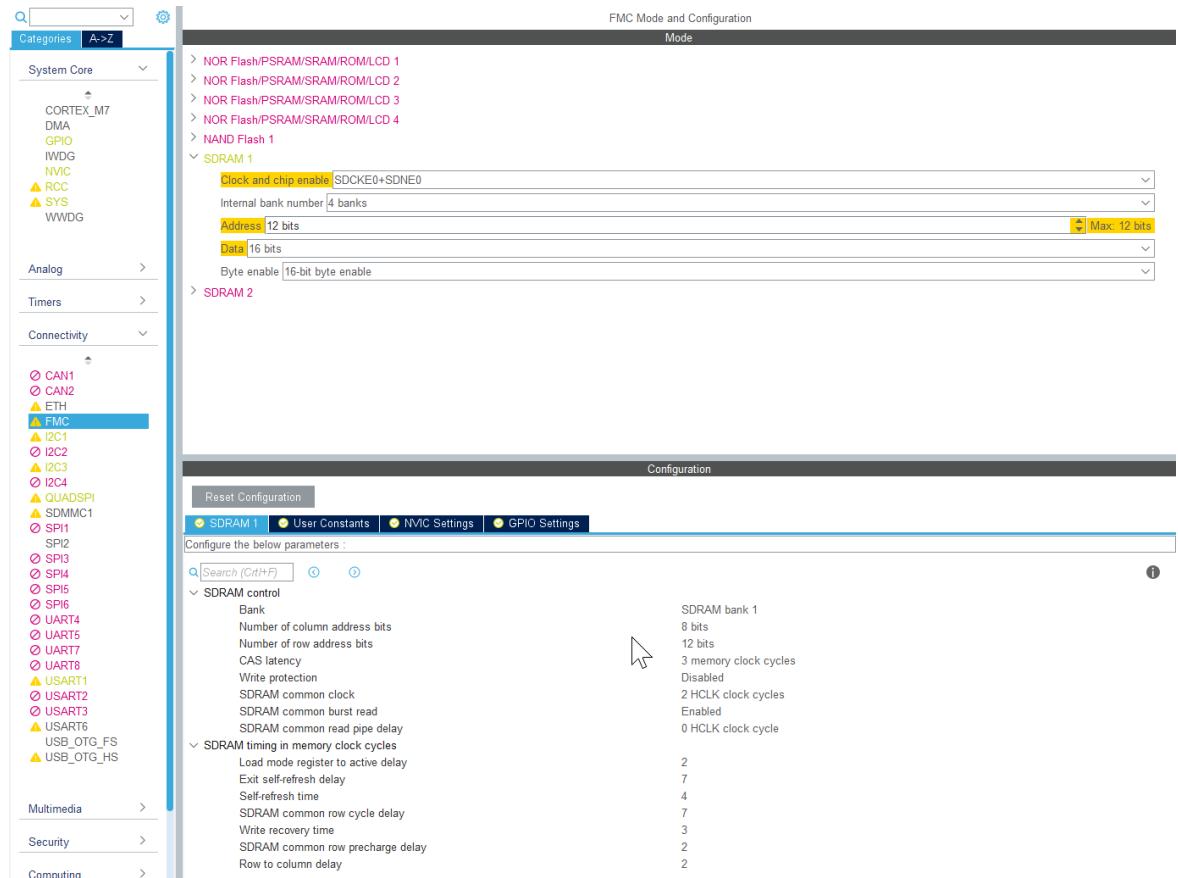
main.c

```
/* USER CODE BEGIN 2 */  
HAL_LTDC_SetAddress(&hltdc, framebuffer, LTDC_LAYER_1);  
/* USER CODE END 2 */
```

```
uint8_t r      = 0xff, g = 0x00, b = 0x00;           // Solid red  
uint16_t col = ((r>>3)<<11) | ((g>>2)<<5) | (b>>3); // Convert colors to RGB565  
// put colors into the framebuffer  
for(int i = 0; i < W*H; i++) {  
    framebuffer[i] = col;  
}
```

Board Bring Up

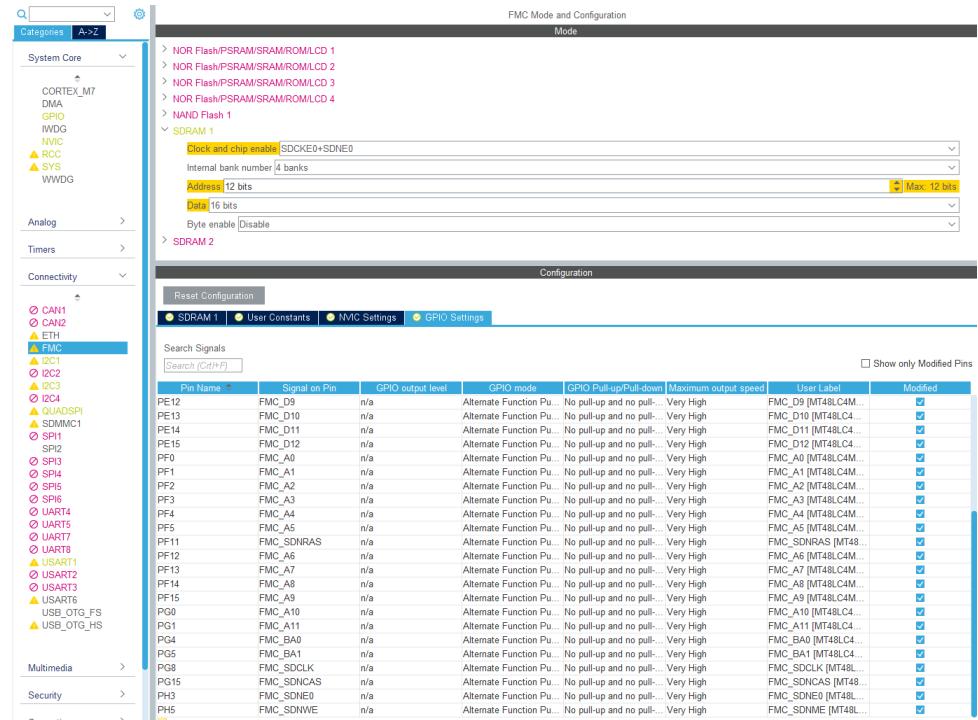
- 4. External Ram
 - Motivation
 - External RAM is often required since the framebuffer(s) does not fit in internal RAM
 - Goal
 - Enable External RAM and be able to r/w
 - Prerequisites
 - Information about the RAM
 - Information about connection between the MCU and RAM
 - Do
 - Setup in STM32CubeMX
 - Some RAMs needs extra C code



Board Bring Up

• 4. External Ram

- Motivation
 - External RAM is often required since the framebuffer(s) does not fit in internal RAM
- Goal
 - Enable External RAM and be able to r/w
- Prerequisites
 - Information about the RAM
 - Information about connection between the MCU and RAM
- Do
 - Setup in STM32CubeMX
 - Some RAMs needs extra C code



```
main.c

FMC_SDRAM_CommandTypeDef Command;

/* Step 1: configure a clock configuration enable command */
Command.CommandMode      = FMC_SDRAM_CMD_CLK_ENABLE;
Command.CommandTarget     = FMC_SDRAM_CMD_TARGET_BANK1;
Command.AutoRefreshNumber = 1;
Command.ModeRegisterDefinition = 0;

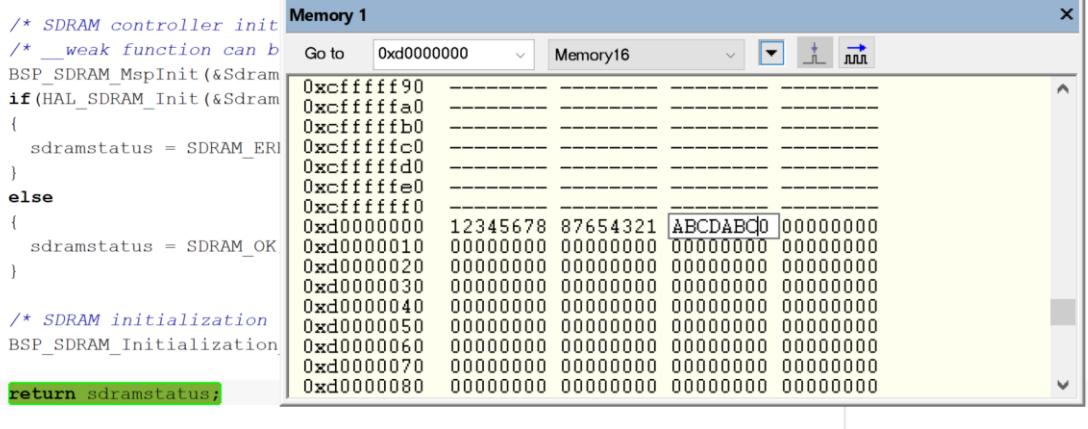
/* Send the command */
HAL_SDRAM_SendCommand(&hsdram1, &Command, SDRAM_TIMEOUT);
```

Board Bring Up

• 4. External Ram

- Motivation
 - External RAM is often required since the framebuffer(s) does not fit in internal RAM
- Goal
 - Enable External RAM and be able to r/w
- Prerequisites
 - Information about the RAM
 - Information about connection between the MCU and RAM
- Do
 - Setup in STM32CubeMX
 - Some RAMs needs extra C code

```
/* FMC SDRAM control configuration */
SdramHandle.Init.SDBank          = FMC_SDRAM_BANK2;
/* Row addressing: [7:0] */
SdramHandle.Init.ColumnBitsNumber = FMC_SDRAM_COLUMN_BITS_NUM_8;
/* Column addressing: [11:0] */
SdramHandle.Init.RowBitsNumber   = FMC_SDRAM_ROW_BITS_NUM_12;
SdramHandle.Init.MemoryDataWidth = SDRAM_MEMORY_WIDTH;
SdramHandle.Init.InternalBankNumber = FMC_SDRAM_INTERN_BANKS_NUM_4;
SdramHandle.Init.CASLatency     = SDRAM_CAS_LATENCY;
SdramHandle.Init.WriteProtection = FMC_SDRAM_WRITE_PROTECTION_DISABLE;
SdramHandle.Init.SDClockPeriod   = SDCLK_PERIOD;
SdramHandle.Init.ReadBurst      = SDRAM_READBURST;
SdramHandle.Init.ReadPipeDelay  = FMC_SDRAM_RPIPE_DELAY_1;
```



The screenshot shows a memory dump window titled "Memory 1". The address bar shows "0xd0000000" and the data width is set to "Memory16". The memory dump area displays a series of memory locations starting from 0x00000000. The first few bytes are filled with the values 12345678 and 87654321, followed by a sequence of bytes labeled "ABCDABC0" and "00000000". Below this, there are several rows of memory starting at 0xd0000010, all containing the value 00000000. The dump continues with more rows of 00000000. At the bottom of the dump, the line "return sdramstatus;" is highlighted in green.

```
uint32_t *externalRAM = 0xC000000;
const uint32_t size = 1000;

//write external RAM
for(int i = 0; i < size; i++)
{
    externalRAM[i] = i;
}
```

Board Bring Up

- 5. Framebuffer in external RAM
 - Motivation
 - Getting graphics on the display from External RAM
 - Goal
 - Remove Framebuffer from Internal RAM, and have the Framebuffer in external RAM
 - Prerequisites
 - Address of the framebuffer in the external RAM
 - Do
 - Place the framebuffer in external RAM
 - Setup the display controller to read from the external RAM

main.c

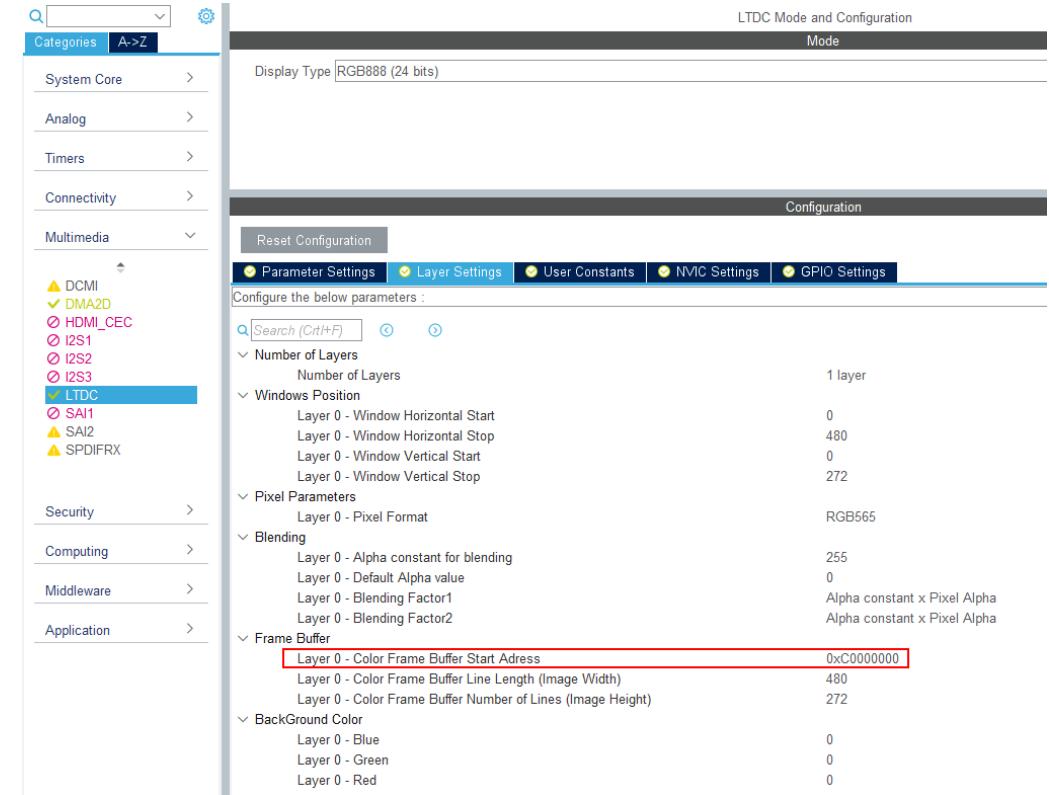
```
uint16_t* framebuffer = (uint16_t*)0xC0000000; //16 bpp framebuffer
```

main.c

```
/* USER CODE BEGIN 2 */  
HAL_LTDC_SetAddress(&hltc, framebuffer, LTDC_LAYER_1);  
/* USER CODE END 2 */
```

Board Bring Up

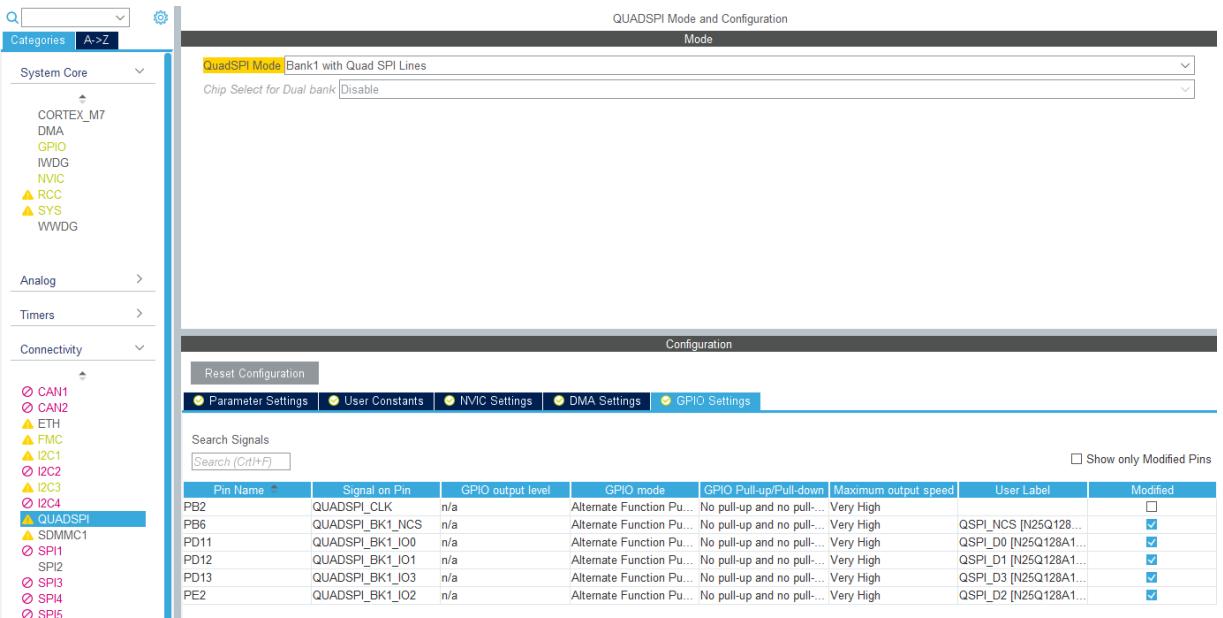
- 5. Framebuffer in external RAM
 - Motivation
 - Getting graphics on the display from External RAM
 - Goal
 - Remove Framebuffer from Internal RAM, and have the Framebuffer in external RAM
 - Prerequisites
 - Address of the framebuffer in the external RAM
 - Do
 - Place the framebuffer in external RAM
 - Setup the display controller to read from the external RAM



Board Bring Up

• 6. External addressable flash

- Motivation
 - External Flash memory is often needed in graphic application since it consist of a lot of images.
- Goal
 - Enable external flash In memory mapped mode and read data from it.
- Prerequisites
 - Information about the flash
 - Information about connection between the MCU and flash
- Do
 - Setup in CubeMX
 - Setup Memory Mapped mode in code



Board Bring Up

- 6. External addressable flash
 - Motivation
 - External Flash memory is often needed in graphic application since it consist of a lot of images.
 - Goal
 - Enable external flash In memory mapped mode and read data from it.
 - Prerequisites
 - Information about the flash
 - Information about connection between the MCU and flash
 - Do
 - Setup in CubeMX
 - Setup Memory Mapped mode in code

main.c

```
QSPI_CommandTypeDef     s_command;
QSPI_MemoryMappedTypeDef s_mem_mapped_cfg;

/* Configure the command for the read instruction */
s_command.InstructionMode    = QSPI_INSTRUCTION_1_LINE;
s_command.Instruction        = QUAD_INOUT_FAST_READ_CMD;
s_command.AddressMode        = QSPI_ADDRESS_4_LINES;
s_command.AddressSize        = QSPI_ADDRESS_24_BITS;
s_command.AlternateByteMode  = QSPI_ALTERNATE_BYTES_NONE;
s_command.DataMode           = QSPI_DATA_4_LINES;
s_command.DummyCycles        = N25Q128A_DUMMY_CYCLES_READ_QUAD;
s_command.DdrMode             = QSPI_DDR_MODE_DISABLE;
s_command.DdrHoldHalfCycle   = QSPI_DDR_HHC_ANALOG_DELAY;
s_command.SIOOMode            = QSPI_SIOO_INST_EVERY_CMD;

/* Configure the memory mapped mode */
s_mem_mapped_cfg.TimeOutActivation = QSPI_TIMEOUT_COUNTER_DISABLE;

if (HAL_QSPI_MemoryMapped(&QSPIHandle, &s_command, &s_mem_mapped_cfg) != HAL_OK)
{
    return QSPI_ERROR;
}
```

- 6. External addressable flash
 - Motivation
 - External Flash memory is often needed in graphic application since it consist of a lot of images.
 - Goal
 - Enable external flash In memory mapped mode and read data from it.
 - Prerequisites
 - Information about the flash
 - Information about connection between the MCU and flash
 - Do
 - Setup in CubeMX
 - Setup Memory Mapped mode in code

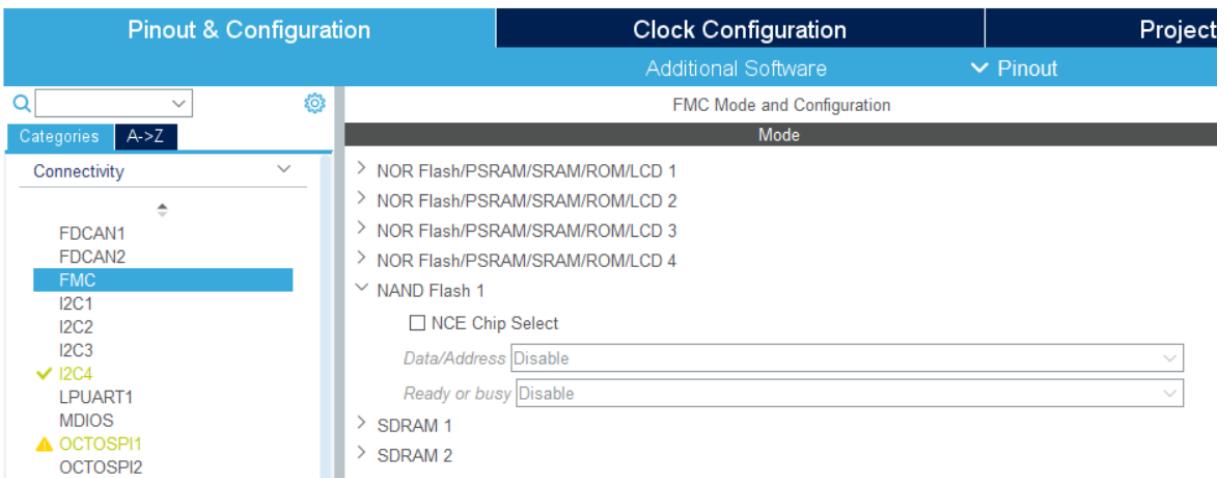
```
volatile uint32_t *externalFlash = 0x90000000;
const uint32_t size = 1000;
volatile uint32_t result = 0;

//read external Flash
for(int i = 0; i < size; i++)
{
    result += externalFlash[i];
}
```

- 7. External flash in block mode
 - Motivation
 - Non-Memory-Mapped Flash memory requires a driver to work with TouchGFX
 - Goal
 - A driver that can read from a location in the flash memory and store it in an array.
 - Prerequisites
 - Information about the flash
 - Information about connection between the MCU and flash
 - The flash speed

Board Bring Up

- 7. External flash in block mode
 - Do
 - Configure the NAND flash via STM32CubeMX via the FMC
 - Set the QSPI flash, similar to memory-mapped
 - Including GPIO
 - Add code to read from an address of the flash



```
void readNonaddressableFlash(uint32_t from, uint8_t *into, uint32_t n)
{
    ...
}

uint8_t bytes[1000];

//read external Flash
readNonaddressableFlash(0xab001212, bytes, 1000);
```

Board Bring Up

• 8. Hardware acceleration

• Motivation

- The Chrom-ART (DMA2D), has the possibility to drastically improving the graphical performance by transferring data from memory into the framebuffer

• Goal

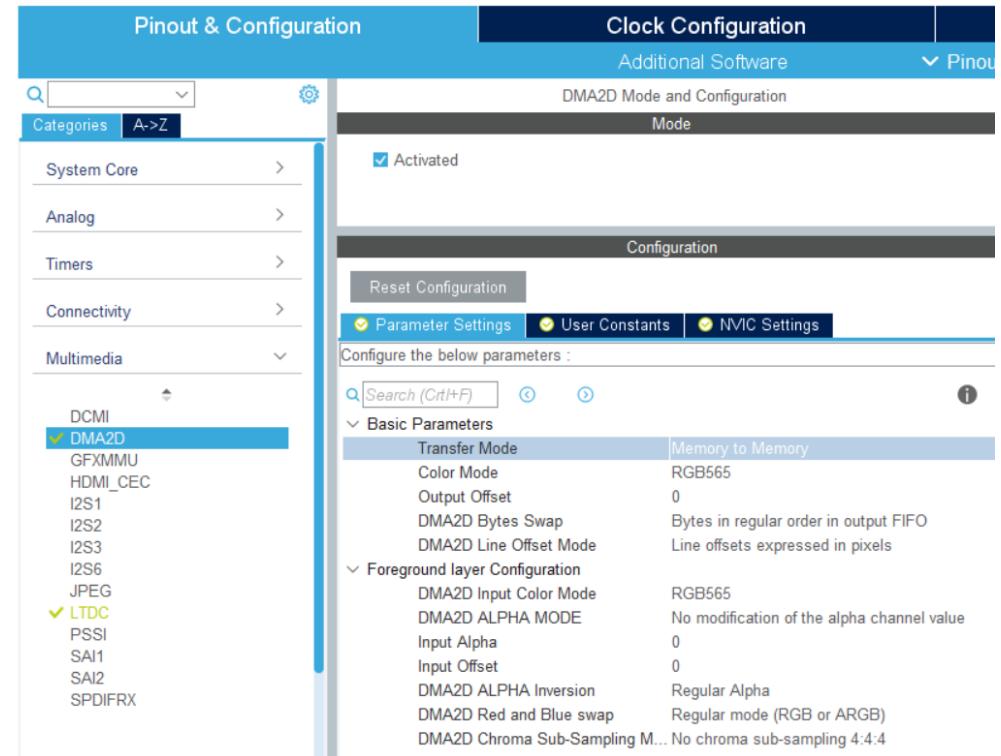
- Enable Chrom-ART and read and write data using it

• Prerequisites

- MCU with Chrom-ART

• Do

- Setup Chrom-ART in STM32CubeMX
- Add code to fil a specific color in a rectangle in target memory



Board Bring Up

• 8. Hardware acceleration

- Motivation
 - The Chrom-ART (DMA2D), has the possibility to drastically improving the graphical performance by transferring data from memory into the framebuffer
- Goal
 - Enable Chrom-ART and read and write data using it
- Prerequisites
 - MCU with Chrom-ART
- Do
 - Setup Chrom-ART in STM32CubeMX
 - Add code to fil a specific color in a rectangle in target memory

Pinout & Configuration		Clock Configuration	
Additional Software		NVIC Mode and Configuration	
Configuration		Configuration	
NVIC	Code generation	NVIC Interrupt Table	Enabled Preempt
NVIC Interrupt Table		Enabled Preempt	
Non maskable interrupt	<input checked="" type="checkbox"/>	0	
Hard fault interrupt	<input checked="" type="checkbox"/>	0	
Memory management fault	<input checked="" type="checkbox"/>	0	
Pre-fetch fault, memory access fault	<input checked="" type="checkbox"/>	0	
Undefined instruction or illegal state	<input checked="" type="checkbox"/>	0	
System service call via SWI instruction	<input checked="" type="checkbox"/>	0	
Debug monitor	<input checked="" type="checkbox"/>	0	
Pendable request for system service	<input checked="" type="checkbox"/>	15	
System tick timer	<input checked="" type="checkbox"/>	15	
PVD and PVM interrupts through EXTI line	<input type="checkbox"/>	5	
Flash global interrupt	<input type="checkbox"/>	5	
RCC global interrupt	<input type="checkbox"/>	5	
EXTI line2 interrupt	<input checked="" type="checkbox"/>	5	
Time base: TIM6 global interrupt, DAC1_CH1 and DAC1_CH...	<input checked="" type="checkbox"/>	0	
FPU global interrupt	<input type="checkbox"/>	5	
LTDC global interrupt	<input checked="" type="checkbox"/>	5	
LTDC Error global Interrupt	<input type="checkbox"/>	5	
DMA2D global interrupt	<input checked="" type="checkbox"/>	5	
OCTOSPI1 global interrupt	<input checked="" type="checkbox"/>	5	
I2C4 event interrupt	<input type="checkbox"/>	5	
I2C4 error interrupt	<input type="checkbox"/>	5	
HSEM1 global interrupt	<input type="checkbox"/>	5	
ECC diagnostic Global Interrupt	<input type="checkbox"/>	5	

Pinout & Configuration		Clock Configuration	
Additional Software		Pinout	
NVIC Mode and Configuration		Project	
Configuration		Configuration	
NVIC	Code generation	Enabled interrupt table	Select for i... Generate IRQ handler Call HAL handler
Enabled interrupt table		Select for i... Generate IRQ handler Call HAL handler	
Memory management fault	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Pre-fetch fault, memory access fault	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Undefined instruction or illegal state	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
System service call via SWI instruc...	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Debug monitor	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Pendable request for system service	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
System tick timer	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
EXTI line2 interrupt	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Time base: TIM6 global interrupt, D...	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
LTDC global interrupt	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
DMA2D global interrupt	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
OCTOSPI1 global interrupt	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Board Bring Up

- 8. Hardware acceleration
 - Motivation
 - The Chrom-ART (DMA2D), has the possibility to drastically improving the graphical performance by transferring data from memory into the framebuffer
 - Goal
 - Enable Chrom-ART and read and write data using it
 - Prerequisites
 - MCU with Chrom-ART
 - Do
 - Setup Chrom-ART in STM32CubeMX
 - Add code to fil a specific color in a rectangle in target memory

```
main.c

#include "stm32f7xx_hal.h"
#include "stm32f7xx_hal_dma2d.h"
...
uint32_t color = 0xF800; //Red in RGB565

hdma2d.Init.Mode = DMA2D_R2M;
hdma2d.Init.ColorMode = DMA2D_RGB565;

MODIFY_REG(hdma2d.Instance->CR, DMA2D_CR_MODE, DMA2D_R2M);
MODIFY_REG(hdma2d.Instance->OPFCCR, DMA2D_OPFCCR_CM, DMA2D_RGB565);
MODIFY_REG(hdma2d.Instance->OOR, DMA2D_OOR_LO, displayWidth - rectangleWidth);

hdma2d.LayerCfg[1].InputColorMode = CM_RGB565;
hdma2d.LayerCfg[1].InputOffset = 0;

HAL_DMA2D_ConfigLayer(&hdma2d, 1);

HAL_DMA2D_Start_IT(&hdma2d, color, (unsigned int)dstPtr, rectangleWidth, rectangleHeight);
```

```
hdma2d.XferCpltCallback = DMA2D_XferCpltCallback;
```

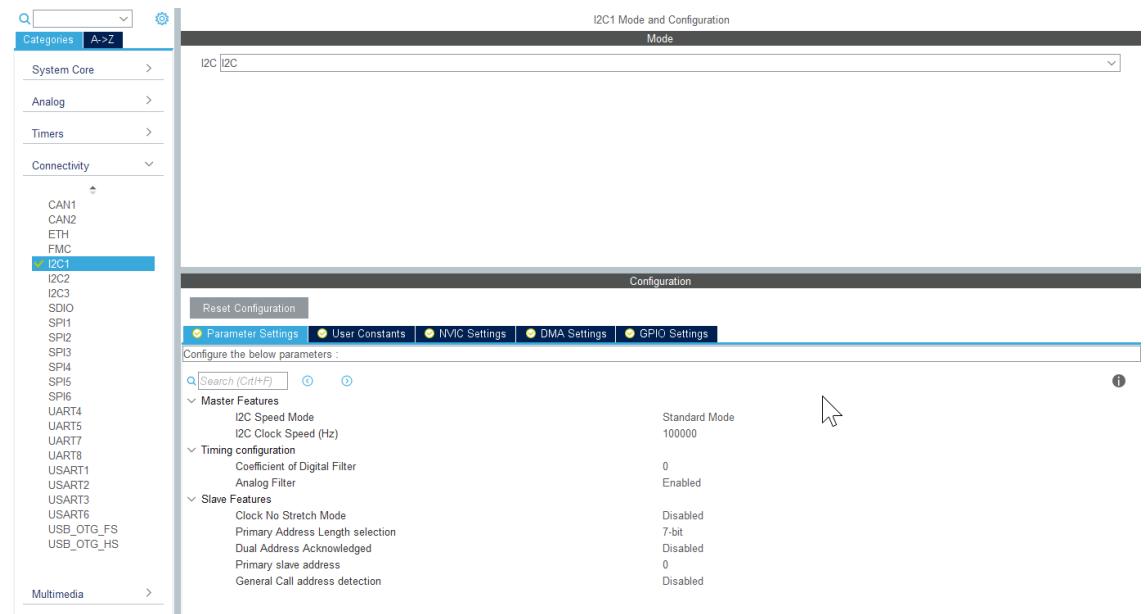
```
extern "C" {
    static void DMA2D_XferCpltCallback(DMA2D_HandleTypeDef* handle)
    {
        //Ensure that you this callback is called
    }
}
```

```
HAL_DMA2D_Start_IT(&hdma2d,
                    (unsigned int)srcPtr,
                    (unsigned int)dstPtr,
                    displayWidth - nrOfPixels);
```

Board Bring Up

• 9. Touch Controller

- Motivation
 - Setting up the Touchcontroller, enables the users to interact with the application
- Goal
 - Touch coordinates can be read from the touch controller
- Prerequisites
 - Display with a touch controller
 - Drivers to read from the touch controller
- Do
 - Setup the communication with the touch controller
- Driver code for touch controller communication



```
main.c

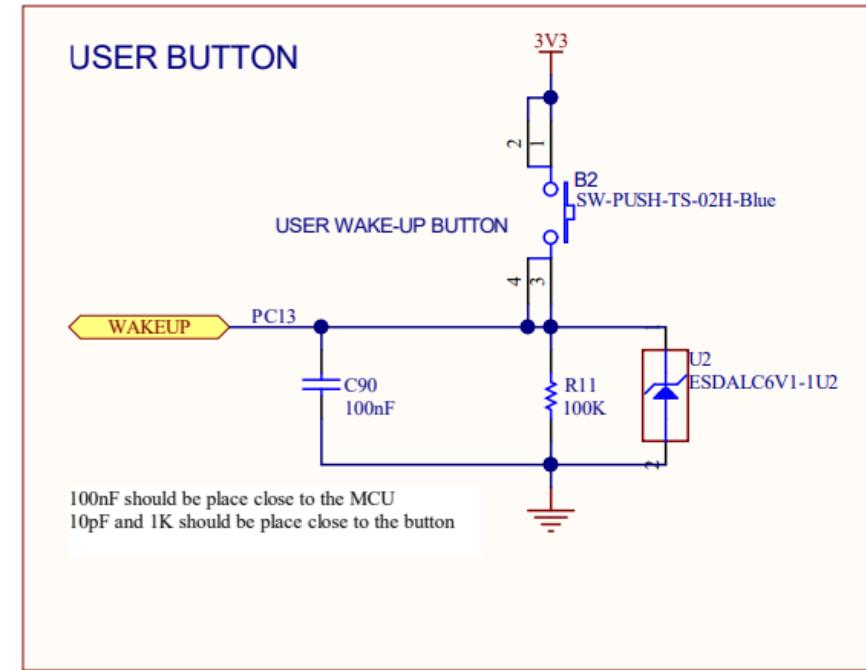
uint16_t x;
uint16_t y;

TouchControllerState state;
if (myTouchController_GetState(&state))
{
    x = state.touchY;
    y = state.touchX;
    //break point here
}
```

Board Bring Up

• 10. Physical Buttons

- Motivation
 - Physical buttons can function as external events to interact with the application, or as an alternative to touch
- Goal
 - Setup the application to receive input from a button
- Prerequisites
 - Physical button connected to MCU
- Do
 - Setup the GPIO for the button in STMCubeMX32
 - Add code to react on the GPIO



• 10. Physical Buttons

- Motivation

- Physical buttons can function as an external events to interact with the application, or as an alternative to touch

- Goal

- Setup the application to receive input from a button

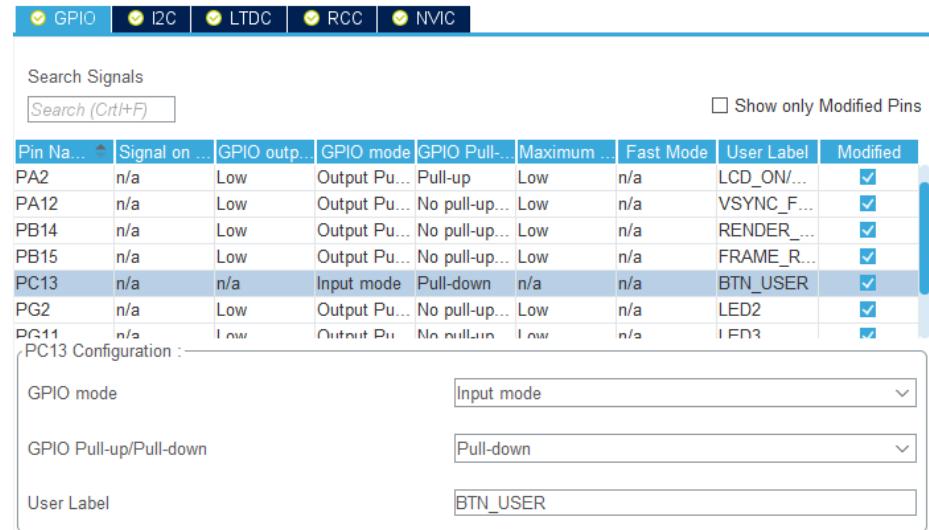
- Prerequisites

- Physical button connected to MCU

- Do

- Setup the GPIO for the button in STMCubeMX32

- Add code to react on the GPIO



```
main.c

uint8_t key;
if (HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13) != GPIO_PIN_RESET)
{
    key = 1;
}
```

- 11. Flash Loader
 - Motivation
 - Enable the solution to write data to the external flash when programming
 - Goal
 - Select a mechanism for loading data to the external flash
 - Prerequisites
 - Information about the flash
 - Information about connection between the MCU and flash
 - Do
 - Flash loader for STM32CubeProgrammer
 - Proprietary application-based solution

Where to go next?

- More information can be found in documentation:

[Hardware Selection & Board Bring Up](#)

- To get started with the Abstraction Layer Development, read the

[TouchGFX AL Development Introduction](#)

- Or watch the presentation

[Abstraction Layer Development - Introduction](#)

Thank you

© STMicroelectronics - All rights reserved.

ST logo is a trademark or a registered trademark of STMicroelectronics International NV or its affiliates in the EU and/or other countries.

For additional information about ST trademarks, please refer to www.st.com/trademarks.

All other product or service names are the property of their respective owners.

