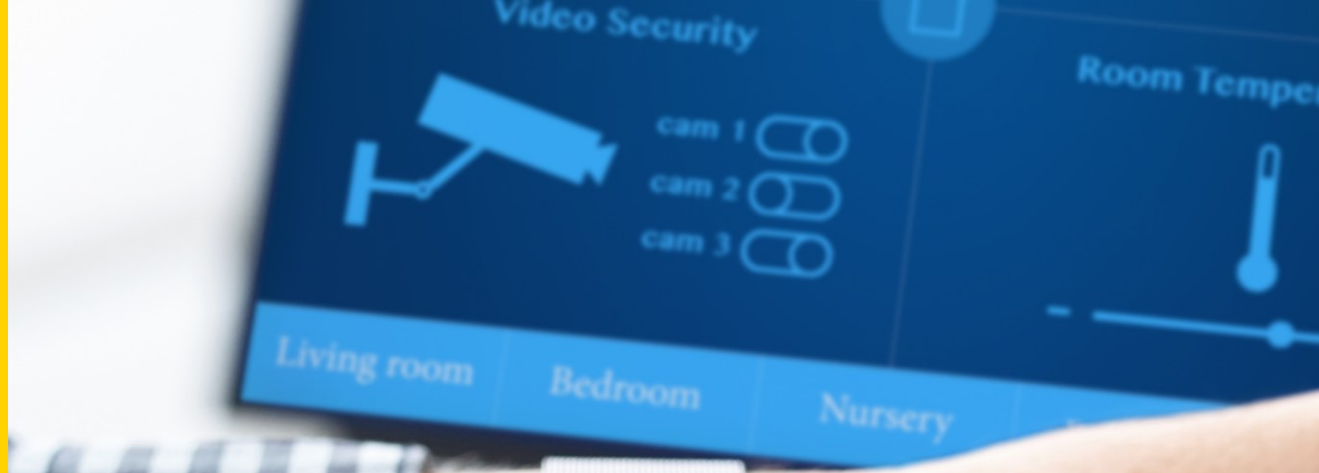




life.augmented



# 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

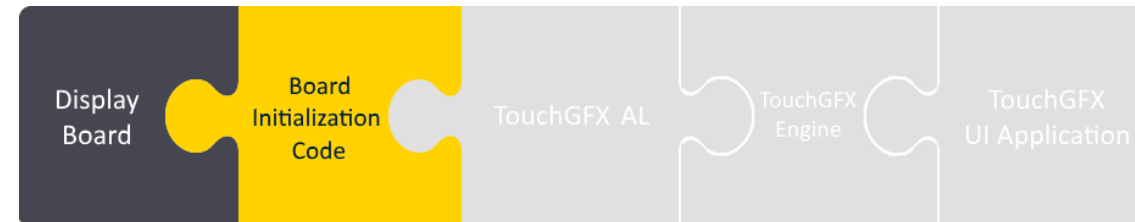
- More help and information can be found at the TouchGFX documentation site:  
<https://support.touchgfx.com/>
- Slides in this presentation will refer to relevant documentation pages. Links will be in the lower right-hand corner of the slides
- The presentation will general be related to the two sections: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

## TouchGFX Development

### Main Activities



### Main Component

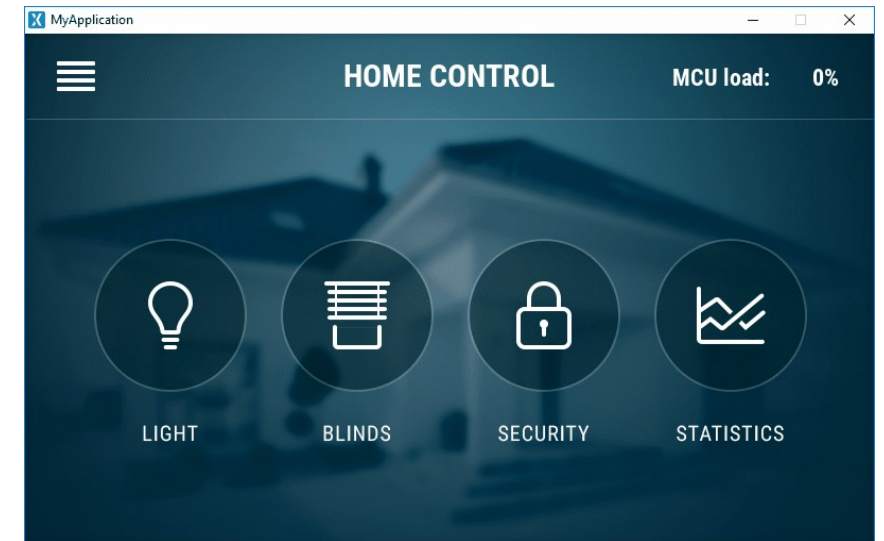


This presentation will start out with discussing the Main Activity, Hardware Selection, before it moves on to the activity Board Bring Up. Thereby we will learn how to select the hardware components needed for the Display Board, before moving on to the initialization of the selected hardware components,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

# Hardware Selection

# Hardware Selection

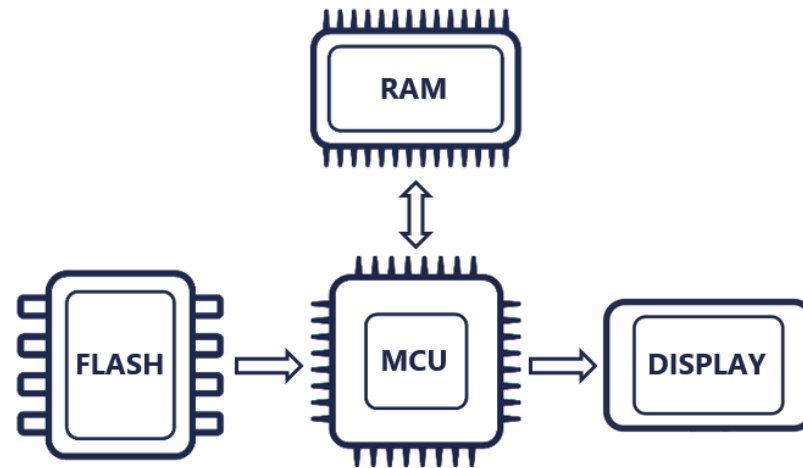
- Preliminary Considerations
  - Application design have impact on the hardware
    - Display Resolution
    - Color Depth
    - Animations
    - Complexity
    - Size
    - Touch
  - Mechanical Design Requirements
    - Size
    - Environment





# 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:

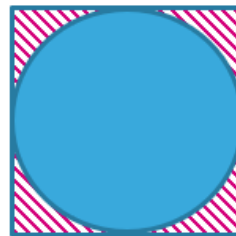
## Hardware Selection



# Hardware Selection






- MCU

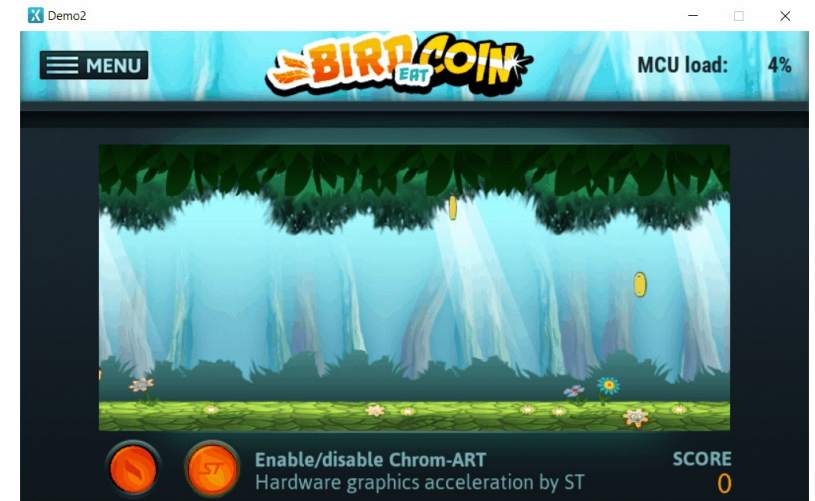
- Frequency
- Embedded Hardware Acceleration
  - Chrom-ART
  - JPEG Accelerator
  - Chrom-GRC
- Internal Memory
  - Ram
  - Flash
- LCD Controller
- Packages & I/O
- Memory Interfacing



 Saved Memory

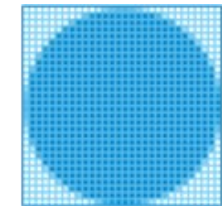
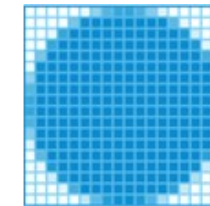
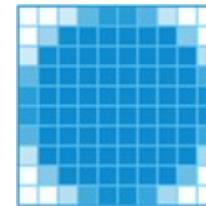
## MCU PORTFOLIO FOR GRAPHICS

 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

## MCU memory interfaces

Parallel	Serial	
FMC / FSMC	SDMMC	QuadSPI / Dual QuadSPI
	SPI	OctoSPI / Hyper bus

## Non-volatile memories



## Volatile memories



# 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



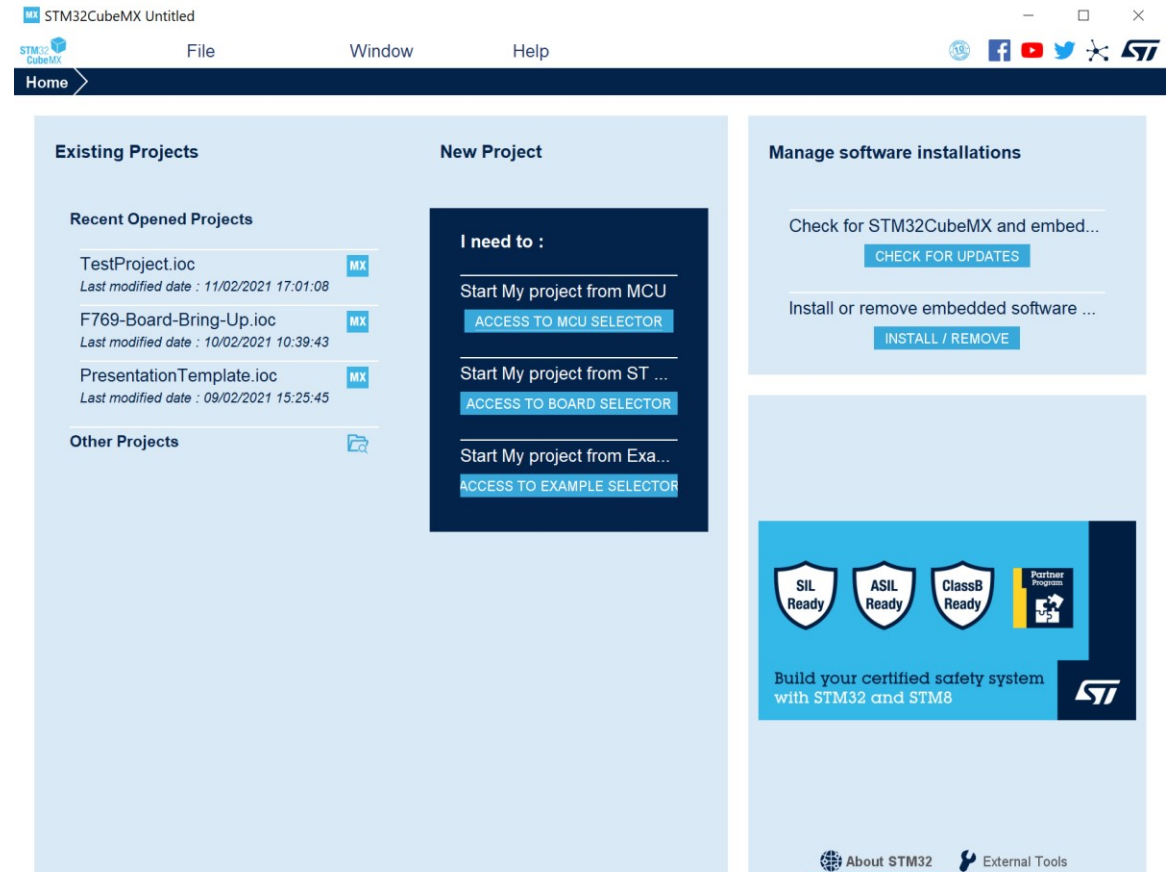
# Board Bring Up

- 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

MCU/MPU Filters

Part Number

Core >

Series >

Line >

Package >

Other >

Peripheral >

Features | Block Diagram | Docs & Resources | Datasheet | Buy

STM32F7 Series

★ **STM32F769NI** 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

**ACTIVE** Active Product is in mass production

Unit Price for 10kU (US\$): 8.939

Boards: [STM32F769I-DISC1](#) - [STM32F769I-DISCO](#) - [STM32F769I-EVAL](#)

TFBGA216

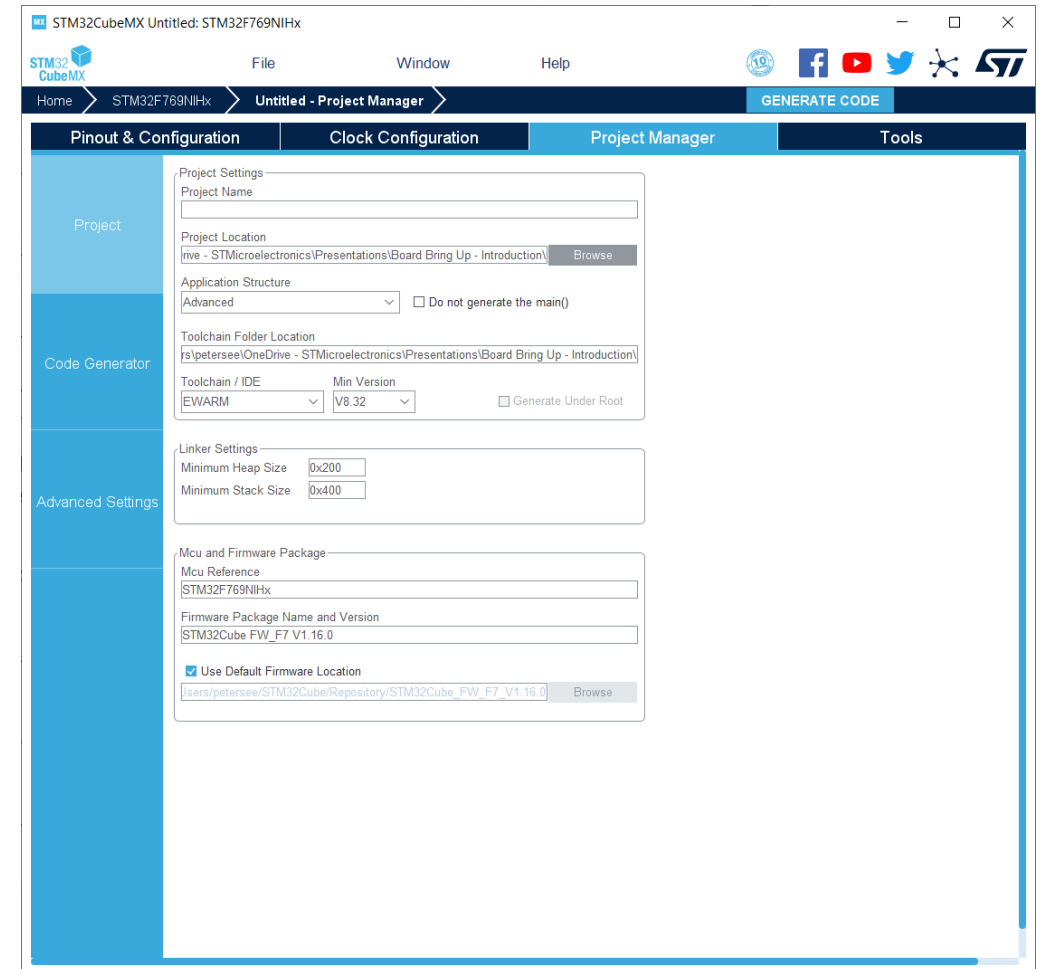
The STM32F765xx, STM32F767xx, STM32F768Ax and STM32F769xx devices are based on the high performance Arm® Cortex® M7 32-bit RISC core operating at up to 216 MHz

MCUs/MPUs List: 1804 items

	Part No.	Reference	Mark...	Unit...	Board...	Flash	RAM	IO	Freq.
☆	STM32F767VG	STM32F767VGHx	Active	6.577		1024 kB...	512 kBy...	82	216 MHz
☆	STM32F767VI	STM32F767VITx	Active	7.428		2048 kB...	512 kBy...	82	216 MHz
☆	STM32F767VH	STM32F767VIHx	Active	7.428		2048 kB...	512 kBy...	82	216 MHz
☆	STM32F767ZG	STM32F767ZGTx	Active	7.109		1024 kB...	512 kBy...	114	216 MHz
☆	STM32F767ZI	STM32F767ZITx	Active	7.96	NUCLE...	2048 kB...	512 kBy...	114	216 MHz
☆	STM32F769AI	STM32F769AIx	Active	7.024		2048 kB...	512 kBy...	128	216 MHz
☆	STM32F769BG	STM32F769BGTx	Active	7.875		1024 kB...	512 kBy...	159	216 MHz
☆	STM32F769BI	STM32F769BITx	Active	8.726		2048 kB...	512 kBy...	159	216 MHz
☆	STM32F769IG	STM32F769IGTx	Active	7.449		1024 kB...	512 kBy...	131	216 MHz
☆	STM32F769II	STM32F769IITx	Active	8.301		2048 kB...	512 kBy...	131	216 MHz
☆	STM32F769NG	STM32F769NGHx	Active	8.088		1024 kB...	512 kBy...	159	216 MHz
☆	STM32F769NI	STM32F769NIHx	Active	8.939		2048 kB...	512 kBy...	159	216 MHz
☆	STM32F777BI	STM32F777BITx	Active	8.918		2048 kB...	512 kBy...	168	216 MHz
☆	STM32F777IT	STM32F777ITx	Active	8.492		2048 kB...	512 kBy...	140	216 MHz
☆	STM32F777II	STM32F777IITx	Active	8.492		2048 kB...	512 kBy...	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 displays the IAR Embedded Workbench IDE interface. The main window shows the source code for `main.c` with the following content:

```
main()
{
    63 int main(void)
    64 {
    65     /* USER CODE BEGIN 1 */
    66
    67     /* USER CODE END 1 */
    68
    69     /* MCU Configuration-----
    70
    71     /* Reset of all peripherals, Initializes the Flash
    72     HAL_Init();
    73
    74     /* USER CODE BEGIN Init */
    75
    76     /* USER CODE END Init */
    77
    78     /* Configure the system clock */
    79     SystemClock_Config();
    80
    81     /* USER CODE BEGIN SysInit */
    82
    83     /* USER CODE END SysInit */
    84
    85     /* Initialize all configured peripherals */
    86     /* USER CODE BEGIN 2 */
    87
    88     /* USER CODE END 2 */
    89
    90     /* Infinite loop */
    91     /* USER CODE BEGIN WHILE */
    92     while (1)
    93     {
    94         /* USER CODE END WHILE */
    95
    96         /* USER CODE BEGIN 3 */
    97     }
}
```

The disassembly window on the right shows the corresponding assembly instructions, including `HAL_Init()` and the `while (1)` loop. The debug log at the bottom indicates a hardware reset with strategy 0 was performed on Thu Feb 11, 2021 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

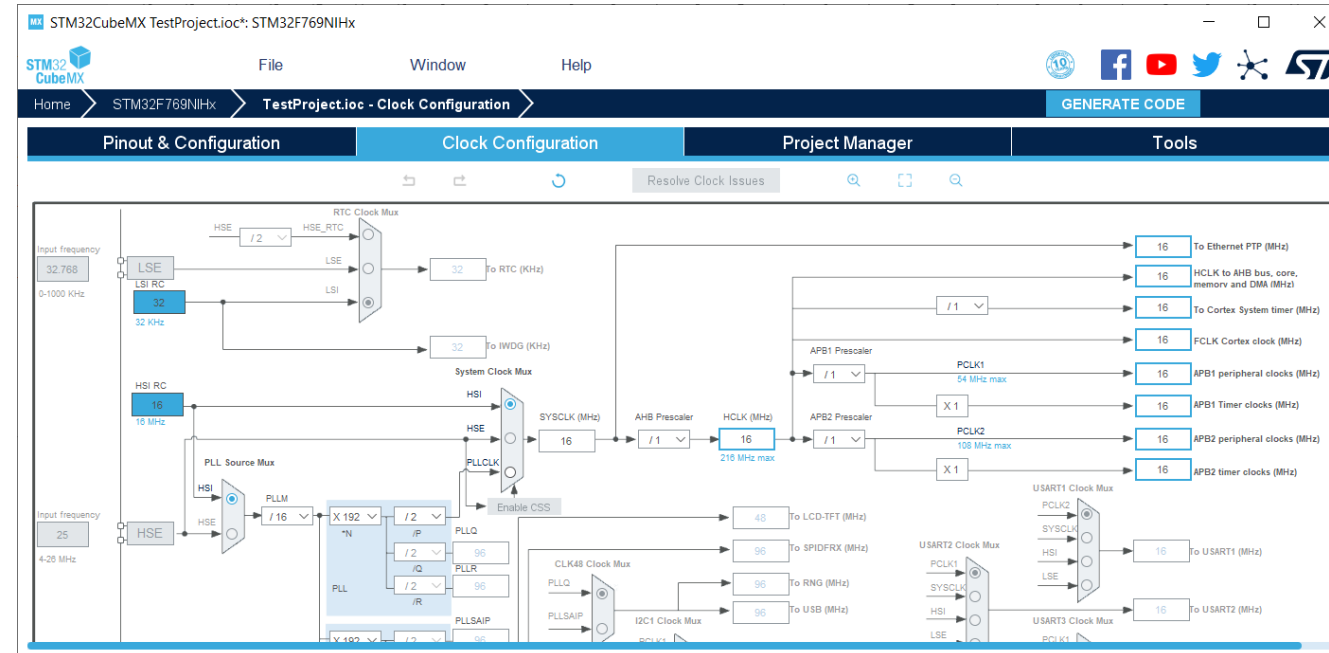
### • Do

- Setup clock in STM32CubeMX under the Clock Configuration

- Check speed by debugging

### • Note

 Disable data cache on F7 and H7  
life.augmented



```
92  /* Infinite loop */
93  /* USER CODE BEGIN WHILE */
94  HAL_Delay(5000);
95  while (1)
96  {
97      /* 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



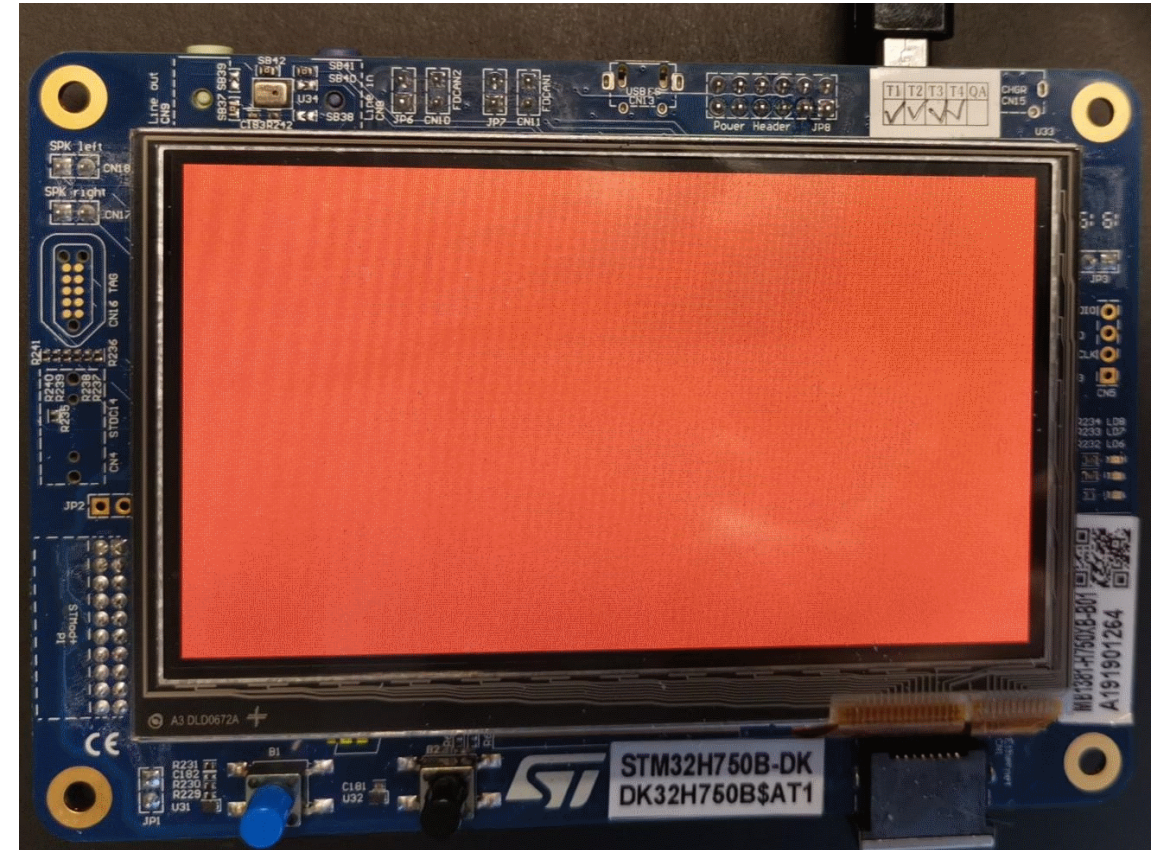
Disable data cache on F7 and H7

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

97  uint32_t* code = (uint32_t*)0x08000000;
98  uint32_t* stop = (uint32_t*)0x08020000;
99  uint32_t sum = 0;
100
101  uint32_t start = HAL_GetTick();
102
103  while (code != stop)
104  {
105      sum += *code++;
106  }
107  result = sum;
108  uint32_t end = HAL_GetTick();
109  time = end - start;
110
111  while (1)
112  {
113      /* USER CODE END WHILE */
```

# 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

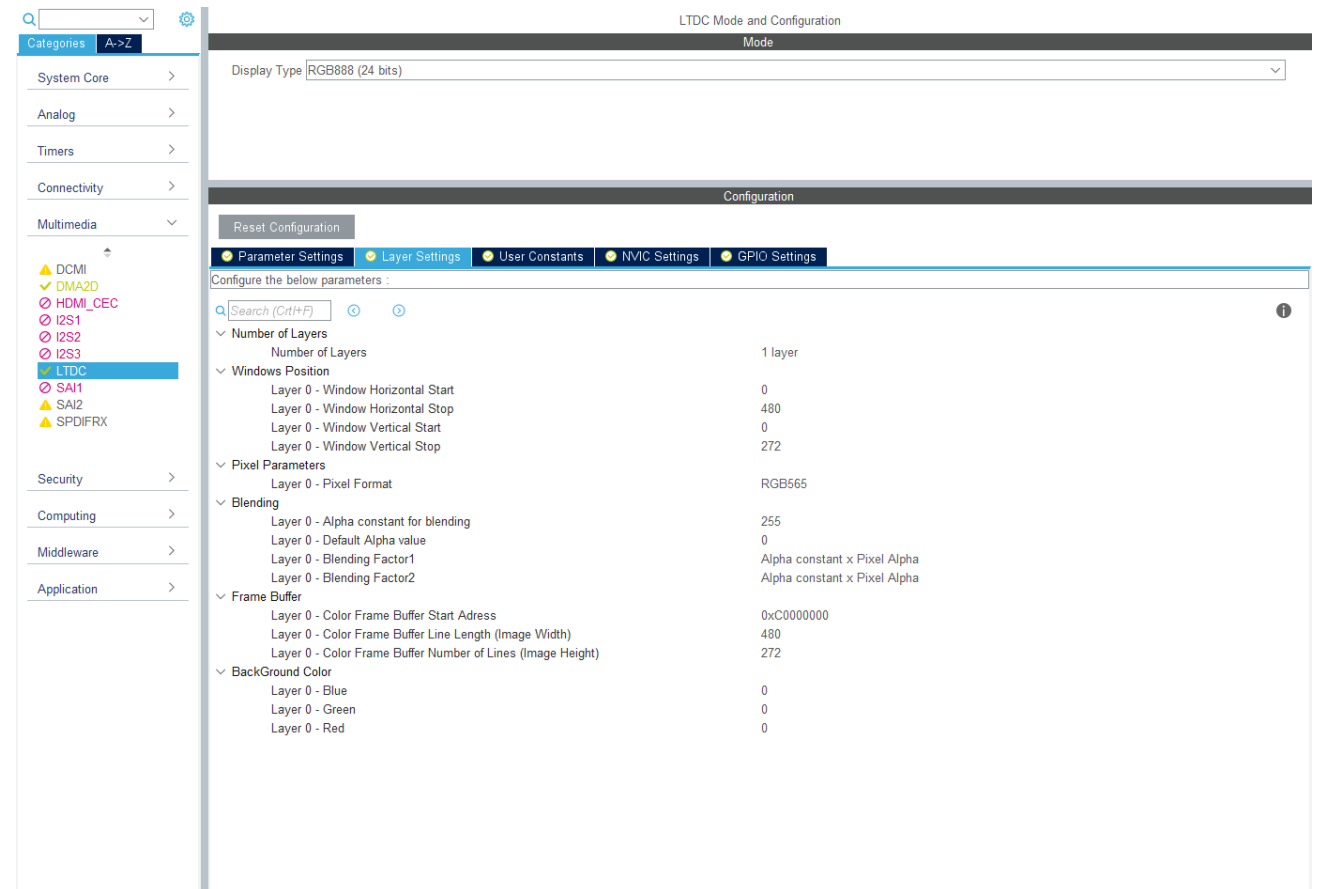
- 3. Framebuffer in internal RAM
  - Do
    - Parallel RGB Displays
      - Setup framebuffer in internal RAM
      - Configure the GPIO connections to the display
      - Configure the LTDC controller
      - Configure the LTDC pixel clock
      - Setting the framebuffer address
      - Check the framerate

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

Pin Name	Signal on Pin	GPIO output level	GPIO mode	GPIO Pull-up/Pull-down	Maximum output speed	User Label	Modified
PE4	LTDC_B0	n/a	Alternate Function Pu...	No pull-up and no pull...	Low	LCD_B0 [RK043FN48...	<input checked="" type="checkbox"/>
PJ13	LTDC_B1	n/a	Alternate Function Pu...	No pull-up and no pull...	Low	LCD_B1 [RK043FN48...	<input checked="" type="checkbox"/>
PJ14	LTDC_B2	n/a	Alternate Function Pu...	No pull-up and no pull...	Low	LCD_B2 [RK043FN48...	<input checked="" type="checkbox"/>
PJ15	LTDC_B3	n/a	Alternate Function Pu...	No pull-up and no pull...	Low	LCD_B3 [RK043FN48...	<input checked="" type="checkbox"/>
PG12	LTDC_B4	n/a	Alternate Function Pu...	No pull-up and no pull...	Low	LCD_B4 [RK043FN48...	<input checked="" type="checkbox"/>
PK4	LTDC_B5	n/a	Alternate Function Pu...	No pull-up and no pull...	Low	LCD_B5 [RK043FN48...	<input checked="" type="checkbox"/>
PK5	LTDC_B6	n/a	Alternate Function Pu...	No pull-up and no pull...	Low	LCD_B6 [RK043FN48...	<input checked="" type="checkbox"/>
PK6	LTDC_B7	n/a	Alternate Function Pu...	No pull-up and no pull...	Low	LCD_B7 [RK043FN48...	<input checked="" type="checkbox"/>
PH14	LTDC_CLK	n/a	Alternate Function Pu...	No pull-up and no pull...	Low	LCD_CLK [RK043FN4...	<input checked="" type="checkbox"/>
PK7	LTDC_DE	n/a	Alternate Function Pu...	No pull-up and no pull...	Low	LCD_DE [RK043FN48...	<input checked="" type="checkbox"/>
PJ7	LTDC_G0	n/a	Alternate Function Pu...	No pull-up and no pull...	Low	LCD_G0 [RK043FN48...	<input checked="" type="checkbox"/>
PJ8	LTDC_G1	n/a	Alternate Function Pu...	No pull-up and no pull...	Low	LCD_G1 [RK043FN48...	<input checked="" type="checkbox"/>
PJ9	LTDC_G2	n/a	Alternate Function Pu...	No pull-up and no pull...	Low	LCD_G2 [RK043FN48...	<input checked="" type="checkbox"/>
PJ10	LTDC_G3	n/a	Alternate Function Pu...	No pull-up and no pull...	Low	LCD_G3 [RK043FN48...	<input checked="" type="checkbox"/>
PJ11	LTDC_G4	n/a	Alternate Function Pu...	No pull-up and no pull...	Low	LCD_G4 [RK043FN48...	<input checked="" type="checkbox"/>
PK0	LTDC_G5	n/a	Alternate Function Pu...	No pull-up and no pull...	Low	LCD_G5 [RK043FN48...	<input checked="" type="checkbox"/>
PK1	LTDC_G6	n/a	Alternate Function Pu...	No pull-up and no pull...	Low	LCD_G6 [RK043FN48...	<input checked="" type="checkbox"/>
PK2	LTDC_G7	n/a	Alternate Function Pu...	No pull-up and no pull...	Low	LCD_G7 [RK043FN48...	<input checked="" type="checkbox"/>
PH10	LTDC_HSYNC	n/a	Alternate Function Pu...	No pull-up and no pull...	Low	LCD_HSYNC [RK043F...	<input checked="" type="checkbox"/>
PH5	LTDC_R0	n/a	Alternate Function Pu...	No pull-up and no pull...	Low	LCD_R0 [RK043FN48...	<input checked="" type="checkbox"/>
PJ0	LTDC_R1	n/a	Alternate Function Pu...	No pull-up and no pull...	Low	LCD_R1 [RK043FN48...	<input checked="" type="checkbox"/>
PJ1	LTDC_R2	n/a	Alternate Function Pu...	No pull-up and no pull...	Low	LCD_R2 [RK043FN48...	<input checked="" type="checkbox"/>
PJ2	LTDC_R3	n/a	Alternate Function Pu...	No pull-up and no pull...	Low	LCD_R3 [RK043FN48...	<input checked="" type="checkbox"/>
PJ3	LTDC_R4	n/a	Alternate Function Pu...	No pull-up and no pull...	Low	LCD_R4 [RK043FN48...	<input checked="" type="checkbox"/>
PJ4	LTDC_R5	n/a	Alternate Function Pu...	No pull-up and no pull...	Low	LCD_R5 [RK043FN48...	<input checked="" type="checkbox"/>
PJ5	LTDC_R6	n/a	Alternate Function Pu...	No pull-up and no pull...	Low	LCD_R6 [RK043FN48...	<input checked="" type="checkbox"/>
PJ6	LTDC_R7	n/a	Alternate Function Pu...	No pull-up and no pull...	Low	LCD_R7 [RK043FN48...	<input checked="" type="checkbox"/>
PH9	LTDC_VSYNC	n/a	Alternate Function Pu...	No pull-up and no pull...	Low	LCD_VSYNC [RK043F...	<input checked="" type="checkbox"/>

# 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 | NVMC Settings | GPIO Settings

Configure the below parameters :

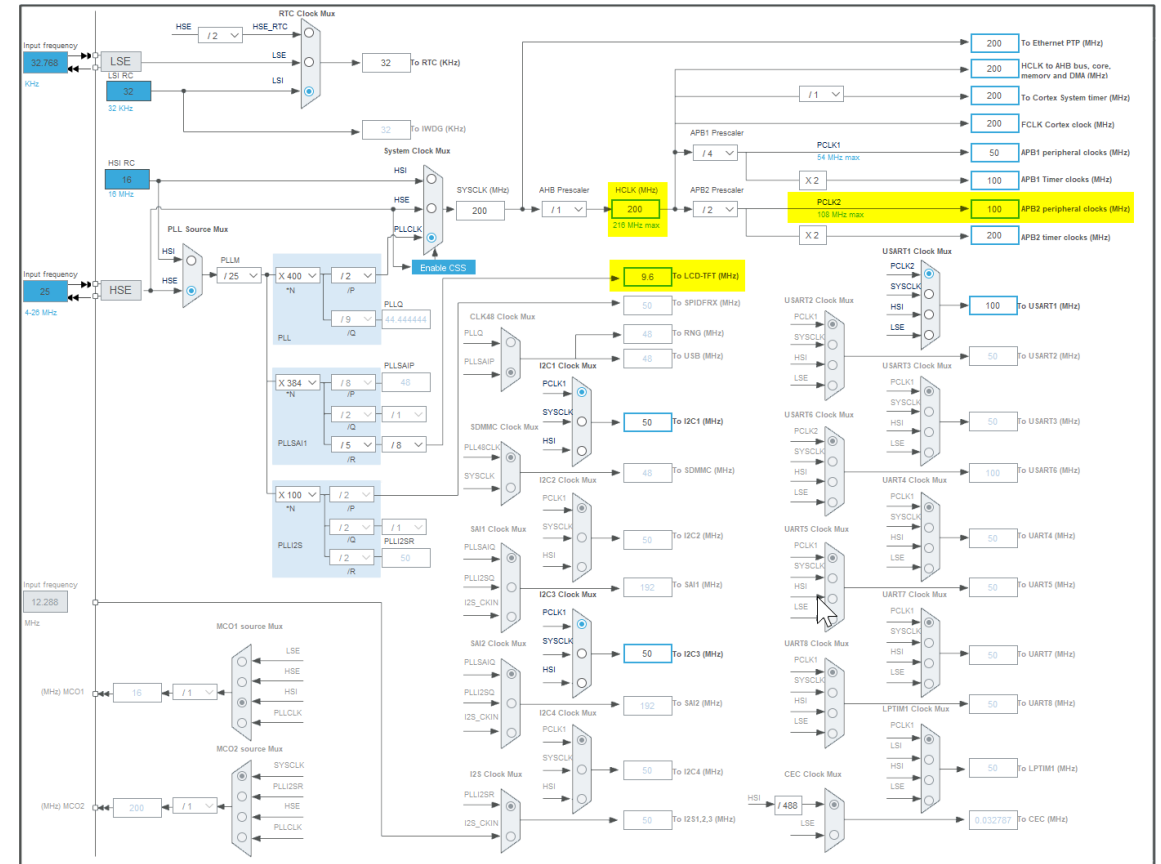
Search (Ctrl+F)

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

The screenshot displays the STM32CubeMX configuration tool in FMC Mode and Configuration. The left sidebar shows a tree view of system components, with FMC selected. The main area shows the configuration for SDRAM 1, including clock and chip enable, internal bank number (4 banks), address (12 bits), data (16 bits), and byte enable (16-bit byte enable). Below this, the Configuration section shows a table of SDRAM control parameters for SDRAM bank 1.

Parameter	Value
Bank	SDRAM bank 1
Number of column address bits	8 bits
Number of row address bits	12 bits
CAS latency	3 memory clock cycles
Write protection	Disabled
SDRAM common clock	2 HCLK clock cycles
SDRAM common burst read	Enabled
SDRAM common read pipe delay	0 HCLK clock cycle
SDRAM timing in memory clock cycles	
Load mode register to active delay	2
Exit self-refresh delay	7
Self-refresh time	4
SDRAM common row cycle delay	7
Write recovery time	3
SDRAM common row precharge delay	2
Row to column delay	2

# 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

The screenshot shows the STM32CubeMX interface for configuring the FMC. The 'Mode' section is set to 'Mode' and shows configurations for NOR Flash and SDRAM. The 'Configuration' section shows a table of pin configurations for various FMC signals.

Pin Name	Signal on Pin	GPIO output level	GPIO mode	GPIO Pull-up/Pull-down	Maximum output speed	User Label	Modified
PE12	FMC_D9	n/a	Alternate Function Pu...	No pull-up and no pull...	Very High	FMC_D9 [MT48LC4M...	<input checked="" type="checkbox"/>
PE13	FMC_D10	n/a	Alternate Function Pu...	No pull-up and no pull...	Very High	FMC_D10 [MT48LC4...	<input checked="" type="checkbox"/>
PE14	FMC_D11	n/a	Alternate Function Pu...	No pull-up and no pull...	Very High	FMC_D11 [MT48LC4...	<input checked="" type="checkbox"/>
PE15	FMC_D12	n/a	Alternate Function Pu...	No pull-up and no pull...	Very High	FMC_D12 [MT48LC4...	<input checked="" type="checkbox"/>
PF0	FMC_A0	n/a	Alternate Function Pu...	No pull-up and no pull...	Very High	FMC_A0 [MT48LC4M...	<input checked="" type="checkbox"/>
PF1	FMC_A1	n/a	Alternate Function Pu...	No pull-up and no pull...	Very High	FMC_A1 [MT48LC4M...	<input checked="" type="checkbox"/>
PF2	FMC_A2	n/a	Alternate Function Pu...	No pull-up and no pull...	Very High	FMC_A2 [MT48LC4M...	<input checked="" type="checkbox"/>
PF3	FMC_A3	n/a	Alternate Function Pu...	No pull-up and no pull...	Very High	FMC_A3 [MT48LC4M...	<input checked="" type="checkbox"/>
PF4	FMC_A4	n/a	Alternate Function Pu...	No pull-up and no pull...	Very High	FMC_A4 [MT48LC4M...	<input checked="" type="checkbox"/>
PF5	FMC_A5	n/a	Alternate Function Pu...	No pull-up and no pull...	Very High	FMC_A5 [MT48LC4M...	<input checked="" type="checkbox"/>
PF11	FMC_SDNRAS	n/a	Alternate Function Pu...	No pull-up and no pull...	Very High	FMC_SDNRAS [MT48...	<input checked="" type="checkbox"/>
PF12	FMC_A6	n/a	Alternate Function Pu...	No pull-up and no pull...	Very High	FMC_A6 [MT48LC4M...	<input checked="" type="checkbox"/>
PF13	FMC_A7	n/a	Alternate Function Pu...	No pull-up and no pull...	Very High	FMC_A7 [MT48LC4M...	<input checked="" type="checkbox"/>
PF14	FMC_A8	n/a	Alternate Function Pu...	No pull-up and no pull...	Very High	FMC_A8 [MT48LC4M...	<input checked="" type="checkbox"/>
PF15	FMC_A9	n/a	Alternate Function Pu...	No pull-up and no pull...	Very High	FMC_A9 [MT48LC4M...	<input checked="" type="checkbox"/>
PG0	FMC_A10	n/a	Alternate Function Pu...	No pull-up and no pull...	Very High	FMC_A10 [MT48LC4...	<input checked="" type="checkbox"/>
PG1	FMC_A11	n/a	Alternate Function Pu...	No pull-up and no pull...	Very High	FMC_A11 [MT48LC4...	<input checked="" type="checkbox"/>
PG4	FMC_BA0	n/a	Alternate Function Pu...	No pull-up and no pull...	Very High	FMC_BA0 [MT48LC4...	<input checked="" type="checkbox"/>
PG5	FMC_BA1	n/a	Alternate Function Pu...	No pull-up and no pull...	Very High	FMC_BA1 [MT48LC4...	<input checked="" type="checkbox"/>
PG8	FMC_SDCLK	n/a	Alternate Function Pu...	No pull-up and no pull...	Very High	FMC_SDCLK [MT48L...	<input checked="" type="checkbox"/>
PG15	FMC_SDNCAS	n/a	Alternate Function Pu...	No pull-up and no pull...	Very High	FMC_SDNCAS [MT48...	<input checked="" type="checkbox"/>
PH3	FMC_SDN0	n/a	Alternate Function Pu...	No pull-up and no pull...	Very High	FMC_SDN0 [MT48L...	<input checked="" type="checkbox"/>
PH5	FMC_SDNWE	n/a	Alternate Function Pu...	No pull-up and no pull...	Very High	FMC_SDNWE [MT48L...	<input checked="" type="checkbox"/>

```
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 = SDCLOCK_PERIOD;
SdramHandle.Init.ReadBurst = SDRAM_READBURST;
SdramHandle.Init.ReadPipeDelay = FMC_SDRAM_RPIPE_DELAY_1;
```

```
/* SDRAM controller init
/* __weak function can be overridden
BSP_SDRAM_MspInit(&SdramHandle)
if(HAL_SDRAM_Init(&SdramHandle) != HAL_OK)
{
    sdramstatus = SDRAM_ERASE_ERROR;
}
else
{
    sdramstatus = SDRAM_OK;
}

/* SDRAM initialization
BSP_SDRAM_Initialization_Ext(
    SDRAM_BANK2,
    SDRAM_MEMORY_WIDTH,
    FMC_SDRAM_COLUMN_BITS_NUM_8,
    FMC_SDRAM_ROW_BITS_NUM_12,
    SDRAM_CAS_LATENCY,
    SDRAM_WRITE_PROTECTION_DISABLE,
    SDCLOCK_PERIOD,
    SDRAM_READBURST,
    SDRAM_RPIPE_DELAY_1);

return sdramstatus;
}
```

Address	Value
0xcfffffff90	-----
0xcfffffff94	-----
0xcfffffff98	-----
0xcfffffff9c	-----
0xcfffffff9e	-----
0xcfffffff9f	-----
0xcfffffff00	-----
0xcfffffff04	-----
0xcfffffff08	-----
0xcfffffff0c	-----
0xcfffffff0e	-----
0xcfffffff10	-----
0xd0000000	12345678 87654321 ABCDABCD 00000000
0xd0000004	00000000 00000000 00000000 00000000
0xd0000008	00000000 00000000 00000000 00000000
0xd000000c	00000000 00000000 00000000 00000000
0xd0000010	00000000 00000000 00000000 00000000
0xd0000014	00000000 00000000 00000000 00000000
0xd0000018	00000000 00000000 00000000 00000000
0xd000001c	00000000 00000000 00000000 00000000
0xd0000020	00000000 00000000 00000000 00000000
0xd0000024	00000000 00000000 00000000 00000000
0xd0000028	00000000 00000000 00000000 00000000
0xd000002c	00000000 00000000 00000000 00000000
0xd0000030	00000000 00000000 00000000 00000000
0xd0000034	00000000 00000000 00000000 00000000
0xd0000038	00000000 00000000 00000000 00000000
0xd000003c	00000000 00000000 00000000 00000000
0xd0000040	00000000 00000000 00000000 00000000
0xd0000044	00000000 00000000 00000000 00000000
0xd0000048	00000000 00000000 00000000 00000000
0xd000004c	00000000 00000000 00000000 00000000
0xd0000050	00000000 00000000 00000000 00000000
0xd0000054	00000000 00000000 00000000 00000000
0xd0000058	00000000 00000000 00000000 00000000
0xd000005c	00000000 00000000 00000000 00000000
0xd0000060	00000000 00000000 00000000 00000000
0xd0000064	00000000 00000000 00000000 00000000
0xd0000068	00000000 00000000 00000000 00000000
0xd000006c	00000000 00000000 00000000 00000000
0xd0000070	00000000 00000000 00000000 00000000
0xd0000074	00000000 00000000 00000000 00000000
0xd0000078	00000000 00000000 00000000 00000000
0xd000007c	00000000 00000000 00000000 00000000
0xd0000080	00000000 00000000 00000000 00000000

```
uint32_t *externalRAM = 0xc0000000;
const uint32_t size = 1000;

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

- 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(&hltdc, 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

The screenshot shows the 'LTDC Mode and Configuration' interface. On the left is a navigation tree with categories like System Core, Analog, Timers, Connectivity, Multimedia, Security, Computing, Middleware, and Application. Under Multimedia, 'LTDC' is selected. The main panel is titled 'LTDC Mode and Configuration' and has a 'Mode' section with 'Display Type' set to 'RGB888 (24 bits)'. Below is a 'Configuration' section with a 'Reset Configuration' button and tabs for 'Parameter Settings', 'Layer Settings', 'User Constants', 'NVIC Settings', and 'GPIO Settings'. The 'Layer Settings' tab is active, showing 'Configure the below parameters :'. A search bar is present. The parameters are listed in a table:

Parameter	Value
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

- 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

QUADSPI Mode and Configuration

Mode

QuadsPI Mode Bank1 with Quad SPI Lines

Chip Select for Dual bank Disable

Configuration

Reset Configuration

Parameter Settings User Constants NVIC Settings DMA Settings **GPIO Settings**

Search Signals

Search (Ctrl+F)

Show only Modified Pins

Pin Name	Signal on Pin	GPIO output level	GPIO mode	GPIO Pull-up/Pull-down	Maximum output speed	User Label	Modified
PB2	QUADSPI_CLK	n/a	Alternate Function Pu...	No pull-up and no pull...	Very High		<input type="checkbox"/>
PB6	QUADSPI_BK1_NCS	n/a	Alternate Function Pu...	No pull-up and no pull...	Very High	QSPI_NCS [N25Q128A...	<input checked="" type="checkbox"/>
PD11	QUADSPI_BK1_IO0	n/a	Alternate Function Pu...	No pull-up and no pull...	Very High	QSPI_D0 [N25Q128A1...	<input checked="" type="checkbox"/>
PD12	QUADSPI_BK1_IO1	n/a	Alternate Function Pu...	No pull-up and no pull...	Very High	QSPI_D1 [N25Q128A1...	<input checked="" type="checkbox"/>
PD13	QUADSPI_BK1_IO3	n/a	Alternate Function Pu...	No pull-up and no pull...	Very High	QSPI_D3 [N25Q128A1...	<input checked="" type="checkbox"/>
PE2	QUADSPI_BK1_IO2	n/a	Alternate Function Pu...	No pull-up and no pull...	Very High	QSPI_D2 [N25Q128A1...	<input checked="" type="checkbox"/>



- 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;
}
```



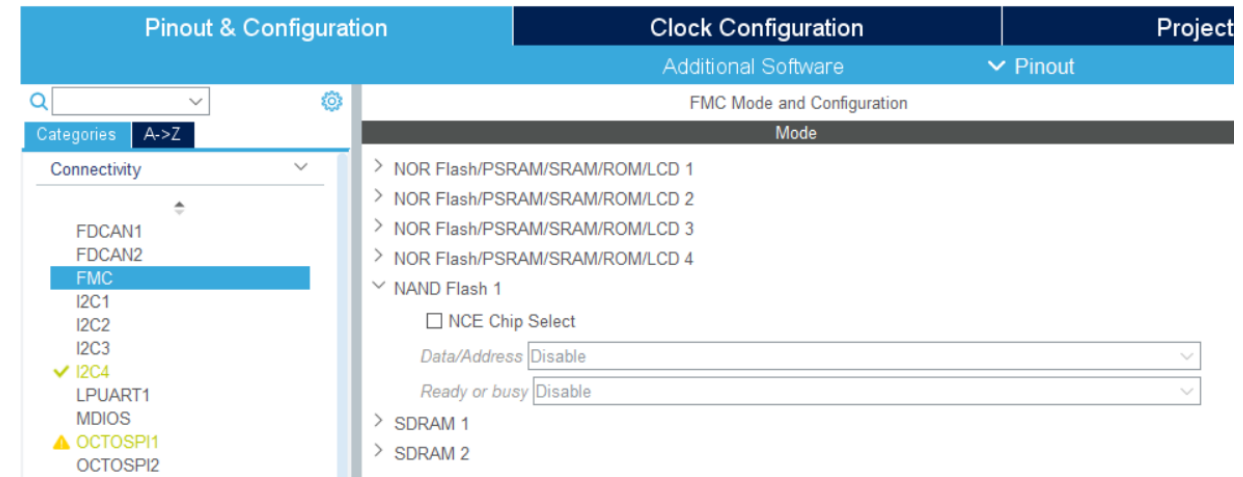
- 6. External addressable flash
  - Motivation
    - External Flash memory is often needed in graphic application since it consist of a lot of images.
  - Goal
    - Enable external flash In memory mapped mode and read data from it.
  - Prerequisites
    - Information about the flash
    - Information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



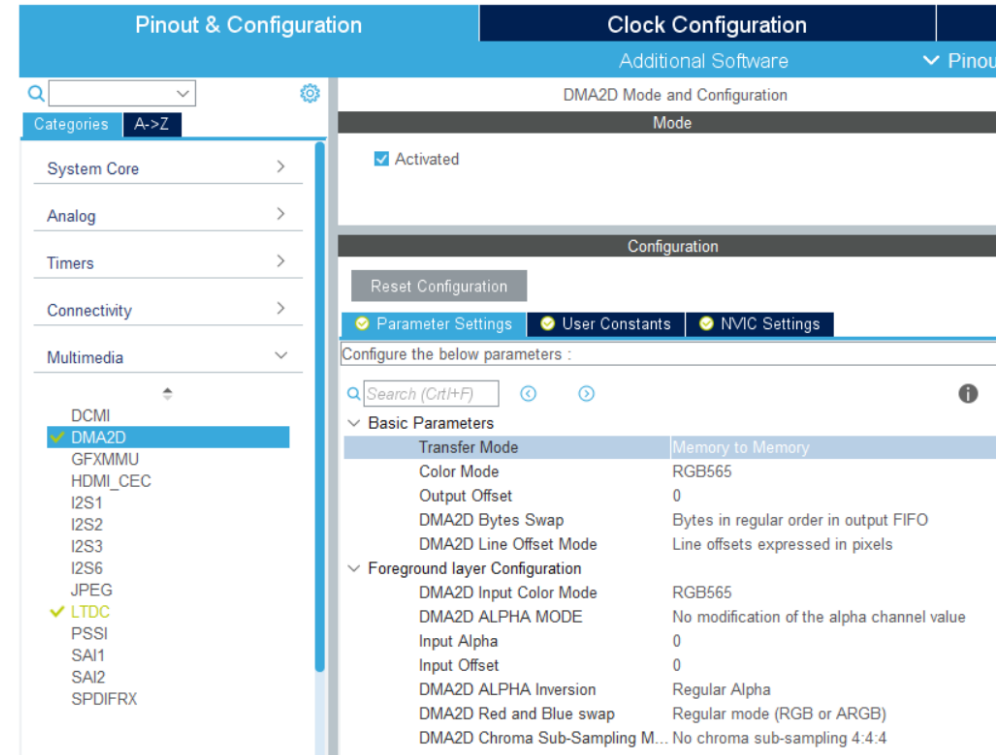
```
void readNonaddressableFlash(uint32_t from, uint8_t *into, uint32_t n)
{
    ...
}

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 fill 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

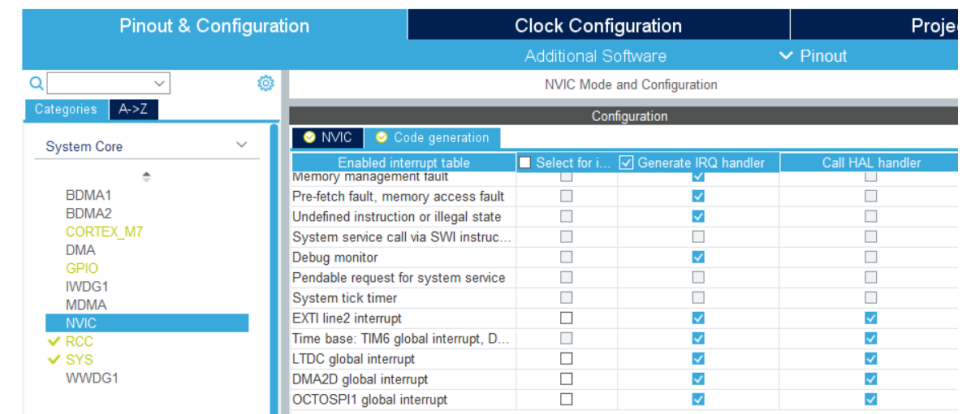
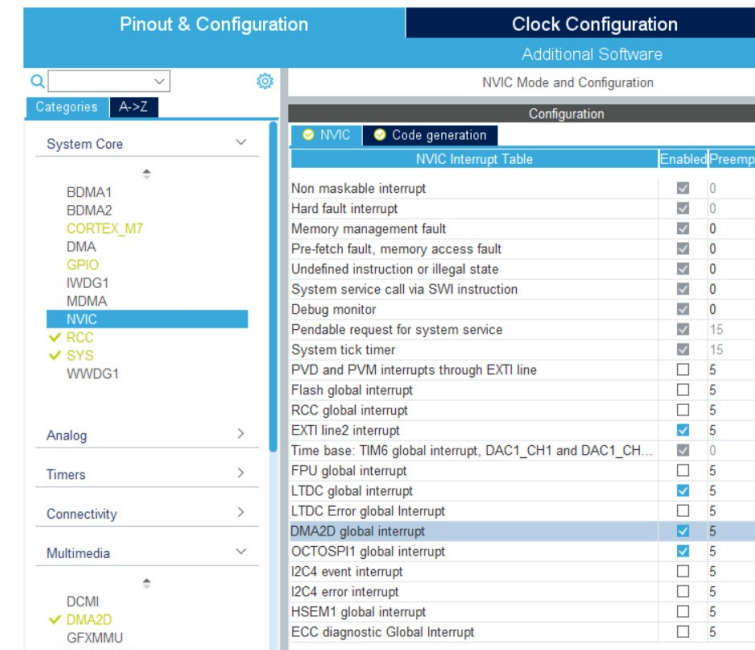
- Enable Chrom-ART and read and write data using it

### • Prerequisites

- MCU with Chrom-ART

### • Do

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



- 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 fill 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

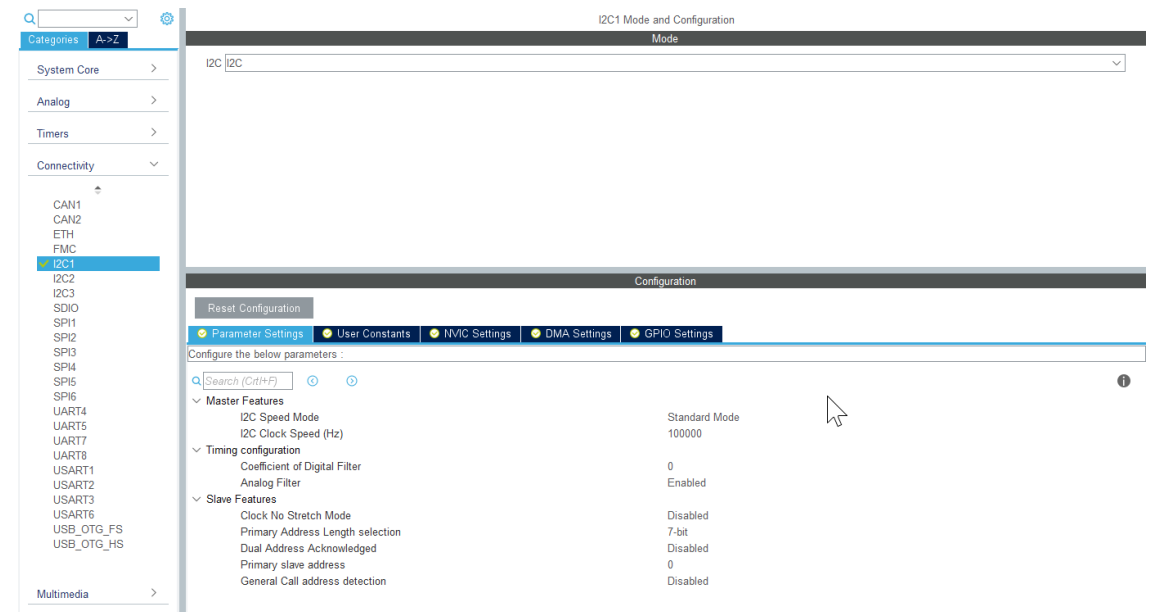
### • Prerequisites

- 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  
}
```



- 10. Physical Buttons

- Motivation

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

- Goal

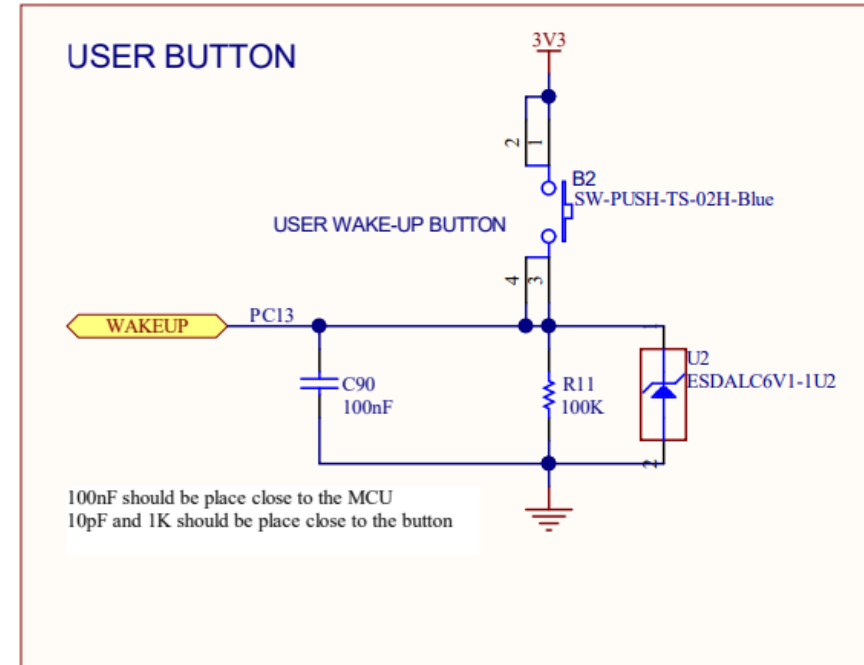
- Setup the application to receive input from a button

- Prerequisites

- Physical button connected to MCU

- Do

- Setup the GPIO for the button in STMCubeMX32
- Add code to react on the GPIO



# Board Bring Up

- 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

Pin Na...	Signal on ...	GPIO outp...	GPIO mode	GPIO Pull...	Maximum ...	Fast Mode	User Label	Modified
PA2	n/a	Low	Output Pu...	Pull-up	Low	n/a	LCD_ON/...	✓
PA12	n/a	Low	Output Pu...	No pull-up...	Low	n/a	VSYNC_F...	✓
PB14	n/a	Low	Output Pu...	No pull-up...	Low	n/a	RENDER_...	✓
PB15	n/a	Low	Output Pu...	No pull-up...	Low	n/a	FRAME_R...	✓
PC13	n/a	n/a	Input mode	Pull-down	n/a	n/a	BTN_USER	✓
PG2	n/a	Low	Output Pu...	No pull-up...	Low	n/a	LED2	✓
PC11	n/a	Low	Output Pu...	No pull-up...	Low	n/a	LED3	✓

PC13 Configuration :

GPIO mode: Input mode

GPIO Pull-up/Pull-down: Pull-down

User Label: BTN\_USER

```
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](http://www.st.com/trademarks).

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



life.augmented