life.augmented

Board Bring Up Introduction

Emil Damkjaer Petersen

Agenda

1 Introduction

2 Hardware Selection

3 Board Bring-Up

life.augmented

Introduction

Goal of this presentation

- Provide information to help developers with selecting hardware for graphcis applications
  - Preleminary 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 neccesary parts for an GUI solution and ensuring the drivers work

Introduction

Introduction

Further reading

- More help and informaion 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

Introduction

TouchGFX Development

Main Activies

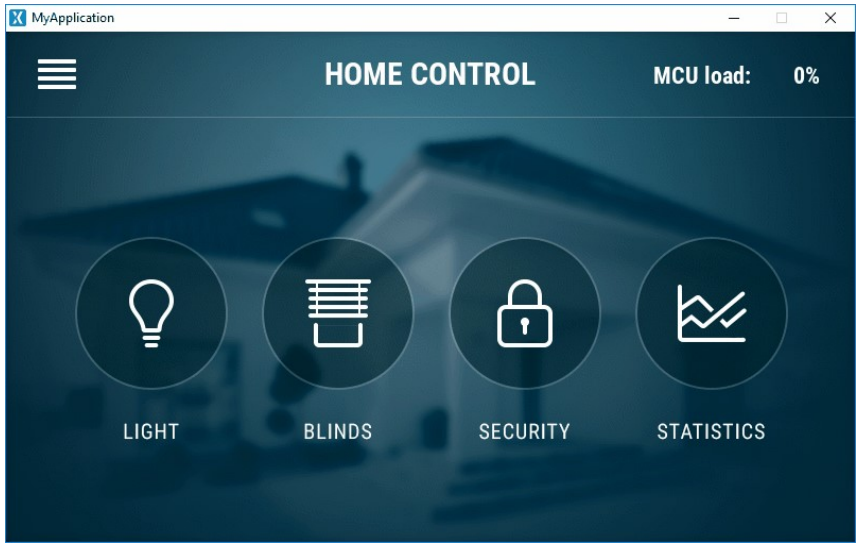
Main Component

This presentation will start out with discussing the Main Activty, 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 compnents, as a part of the Board Initialization Code, Main Component.

Hardware Selection

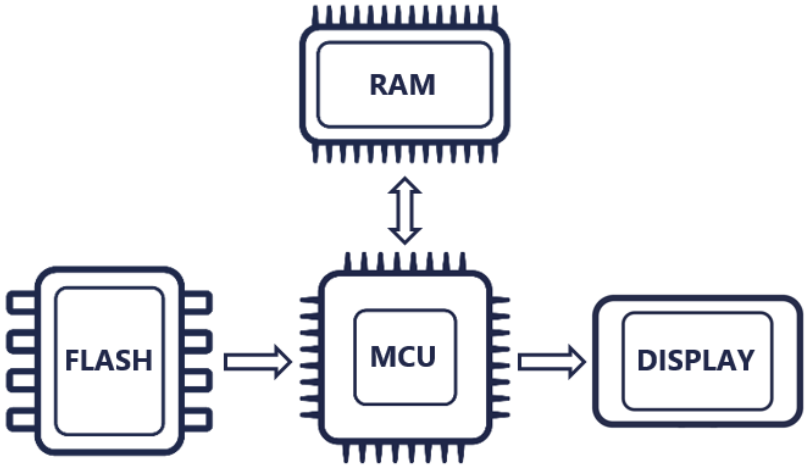
Hardware Selection

- 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



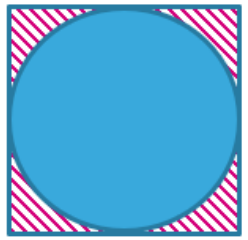
- More information can be found in the documentation:

  Hardware Selection

life.augmented

Hardware Selection

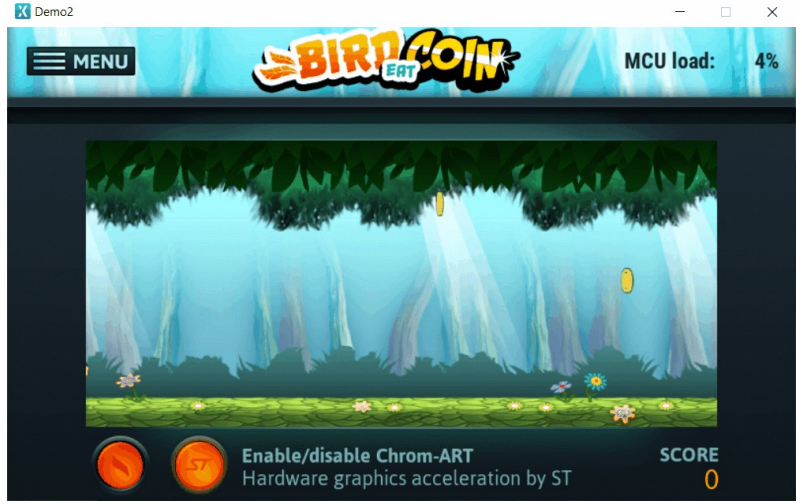
- MCU
  - Frequency
  - Embedded Hardware Acceleration
    - Chrom-ART
    - JPEG Accelerator
    - Chrom-GRC
  - Internal Memory
    - Ram
    - Flash
  - LCD Controller
  - Packages & I/O
  - Memory Interfacing



Saved Memory

MCU PORTFOLIO FOR GRAPHICS

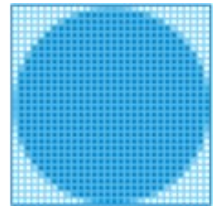
STM32 SERIES	FREQUENCY	HARDWARE ACCELERATION	DISPLAY INTERFACES	SUPPORTED RESOLUTIONS
STM32G0 (CM0+)	64 MHz		SPI	Up to 320*240
STM32F7 (CM7)	216 MHz	Chrom-ART ™ Hardware JPEG Codec	Parallel LCD TFT MIPI-DSI	Up to 1024*768



life.augmented

Hardware Selection

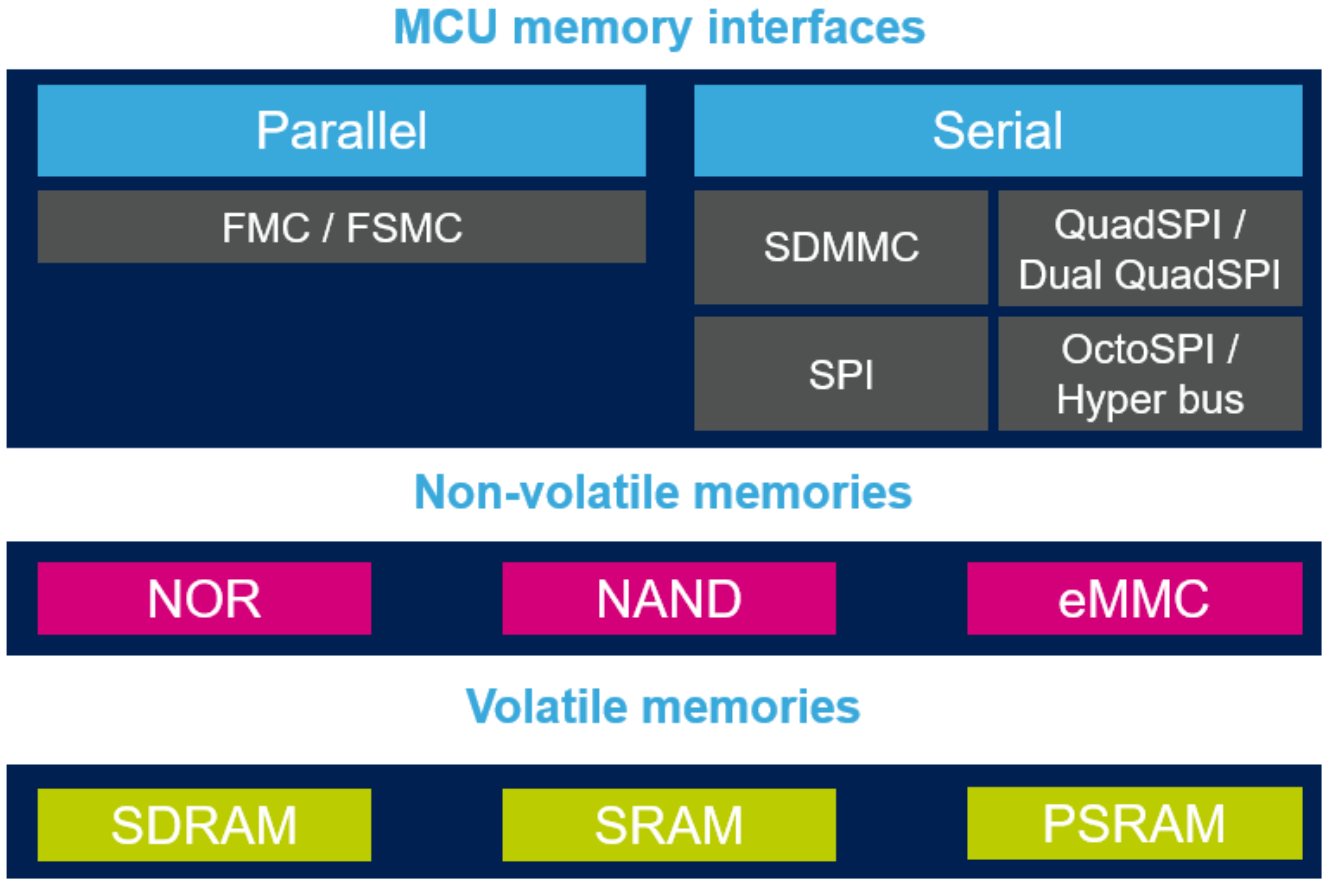
- Display
  - Different Types of Display
    - LCD-TFT
    - MIP
    - ePaper/eInk
  - Display Interface
  - Brightness and Backlight
  - Viewing Position and Color Inversion
  - Pixel density
  - Displays with RAM
  - Environment



life.augmented

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

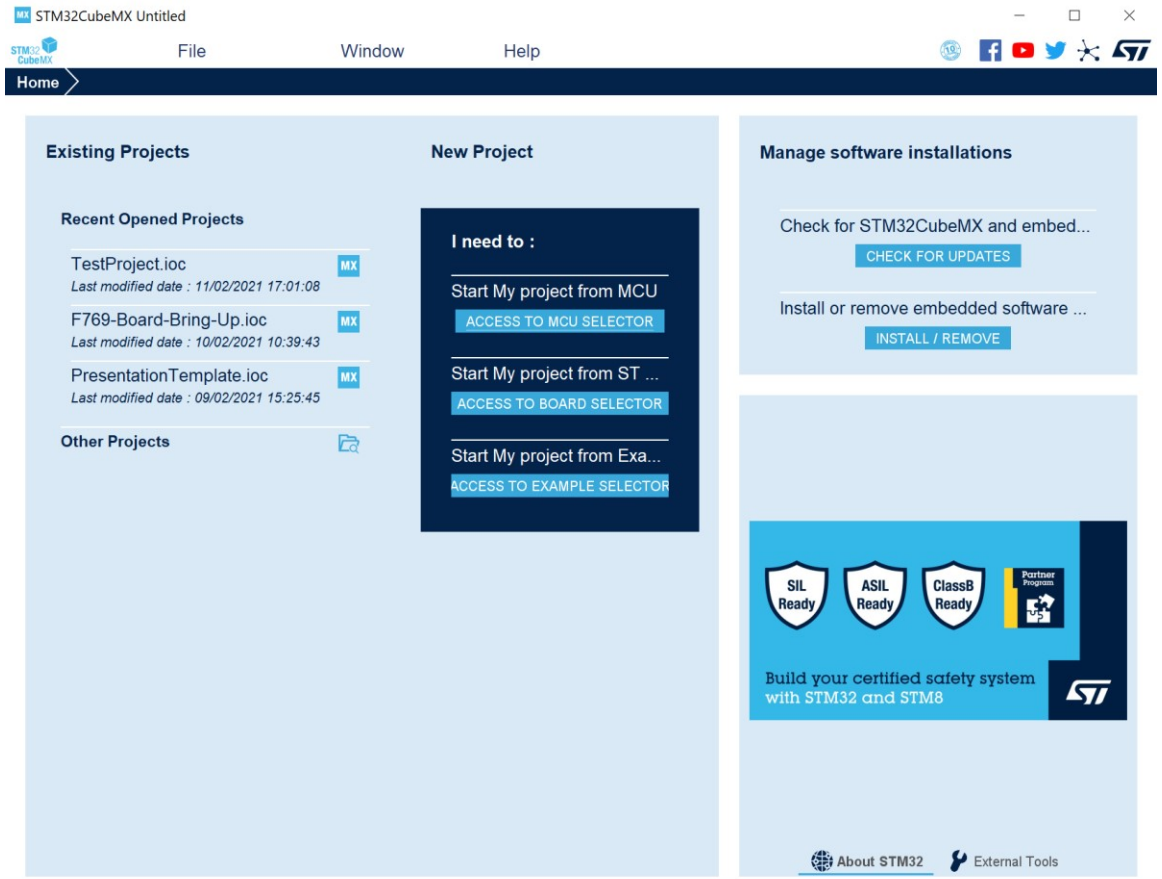
Board Bring Up

- 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

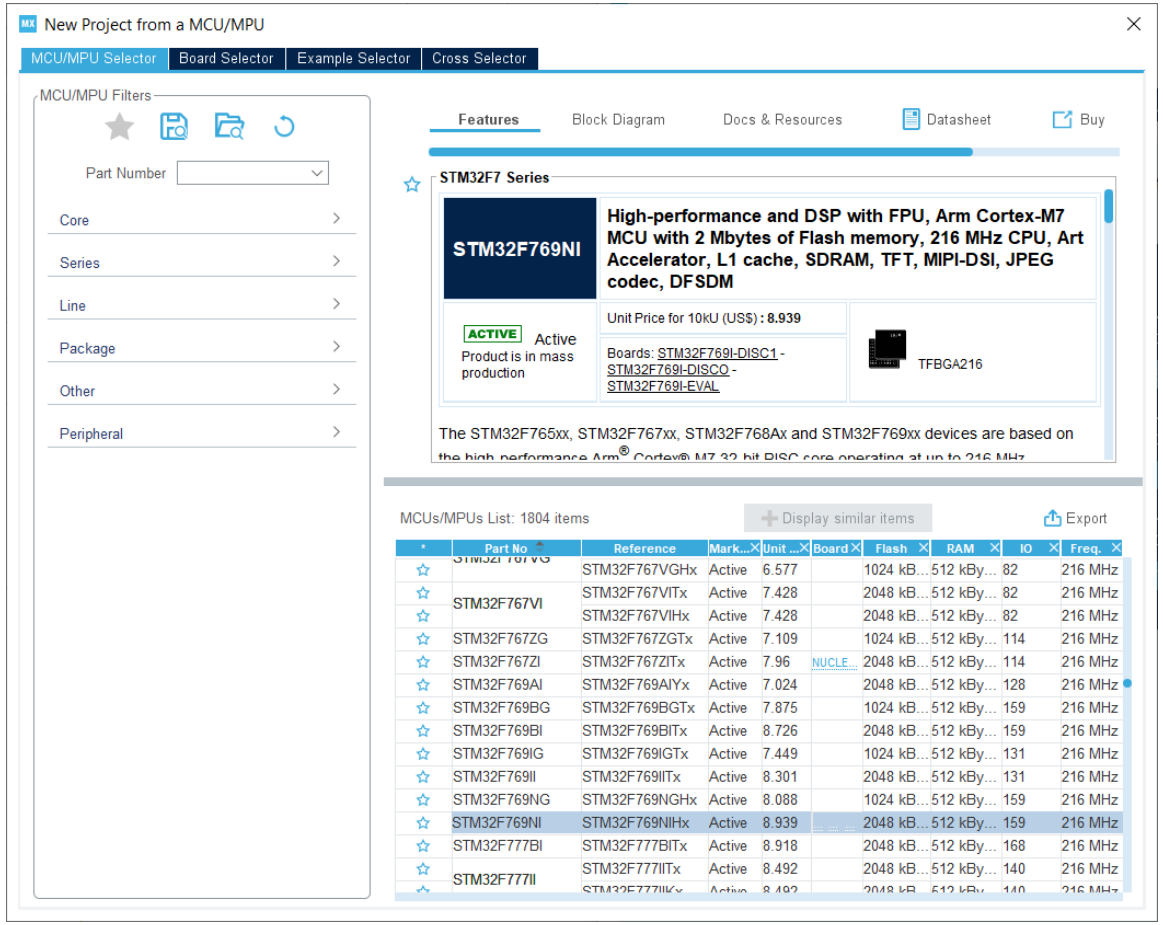
Board Bring Up

- 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



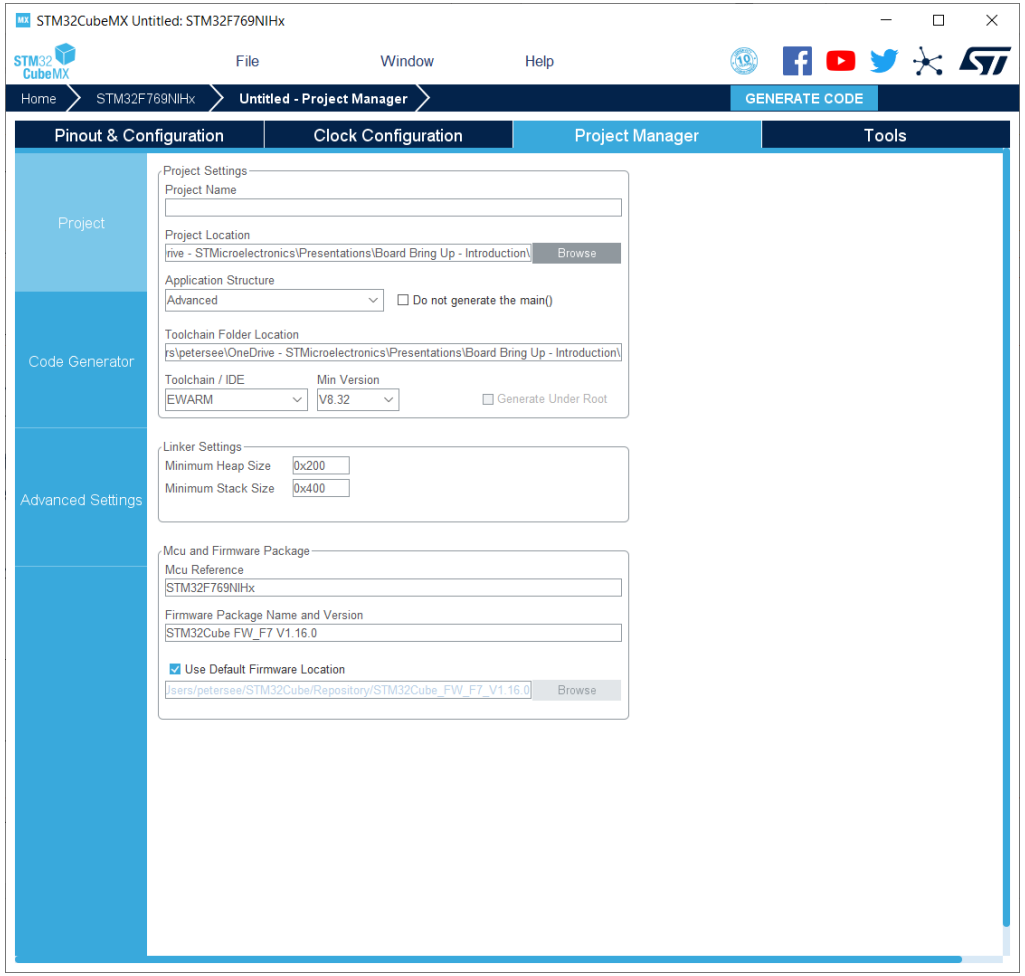
Board Bring Up

- 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



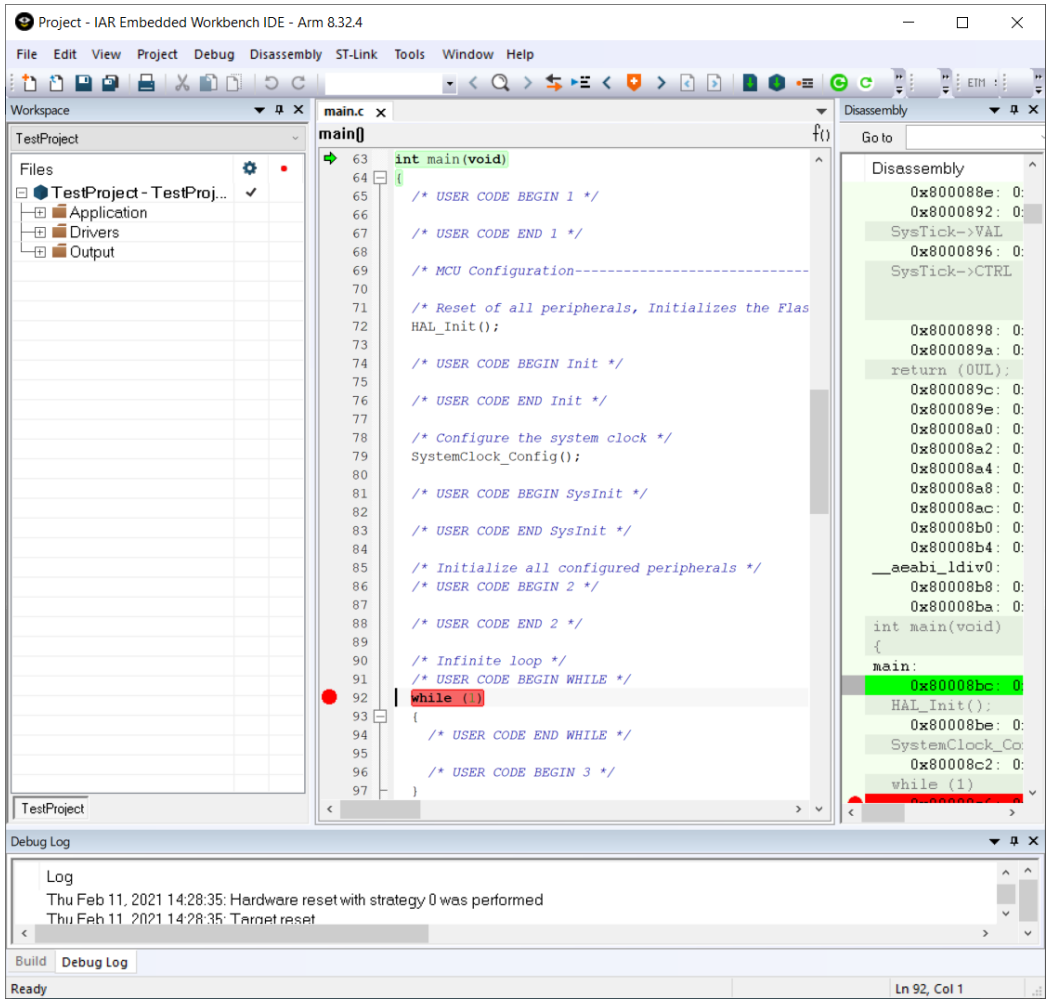
Board Bring Up

- 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



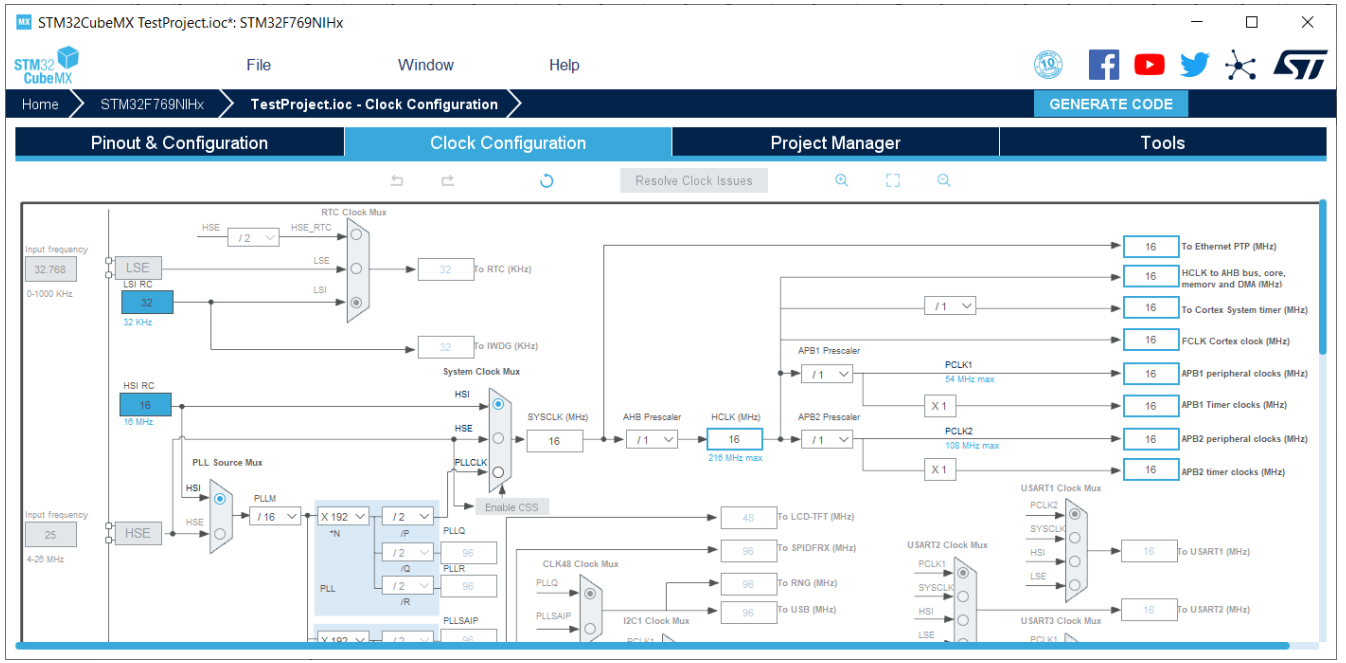
Board Bring Up

- 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



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




```
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

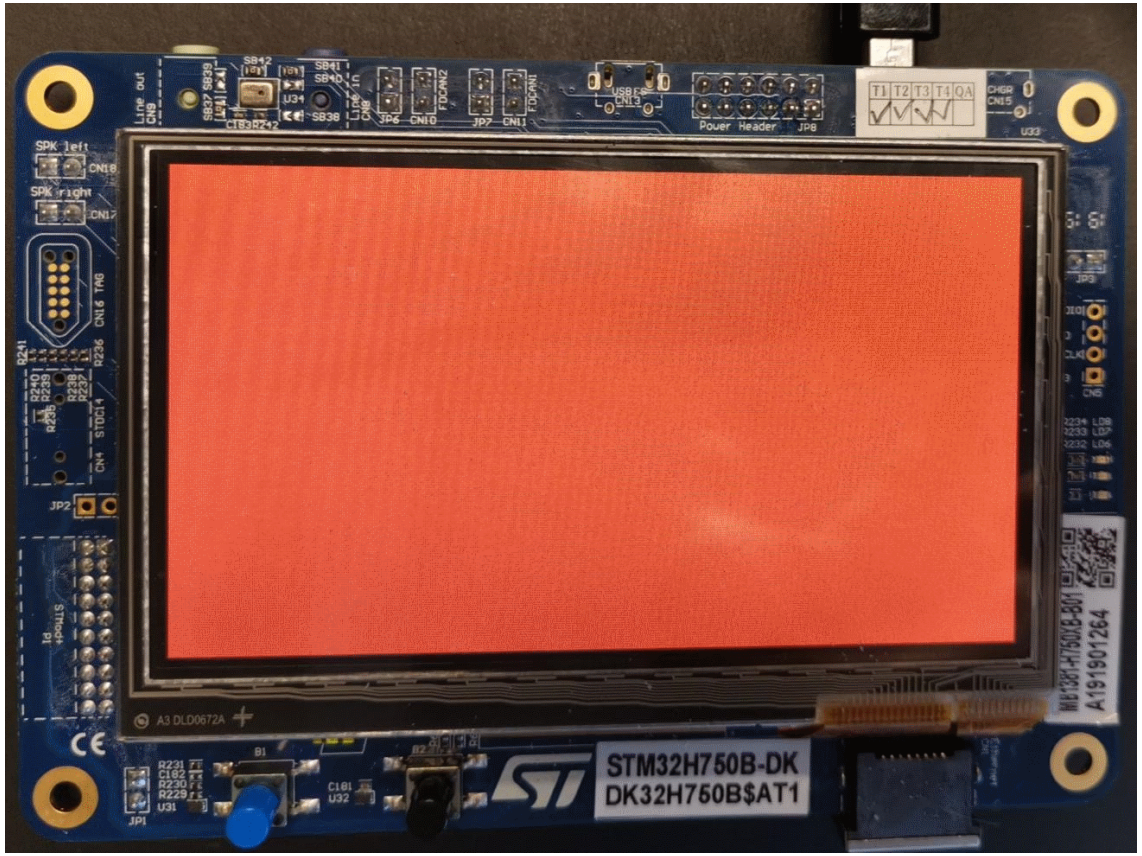
```
45  /* USER CODE BEGIN PV */
46  volatile uint32_t result;
47  volatile uint32_t time;
48  /* USER CODE END PV */
```

```
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 {
```

life.augmented

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

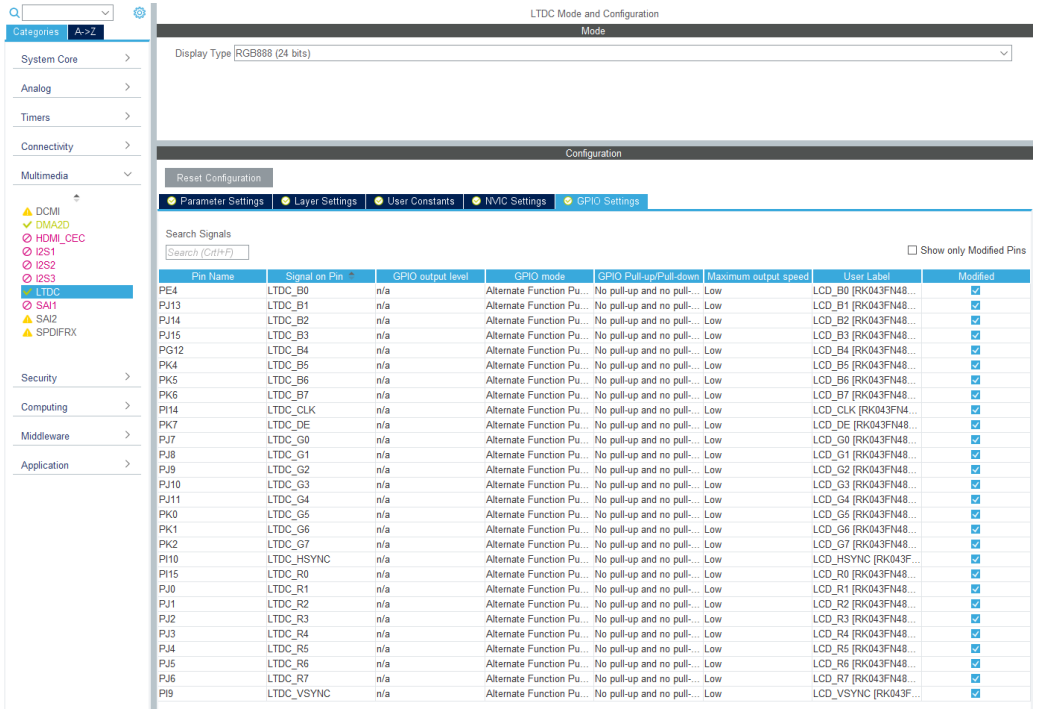


Board Bring Up

- 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

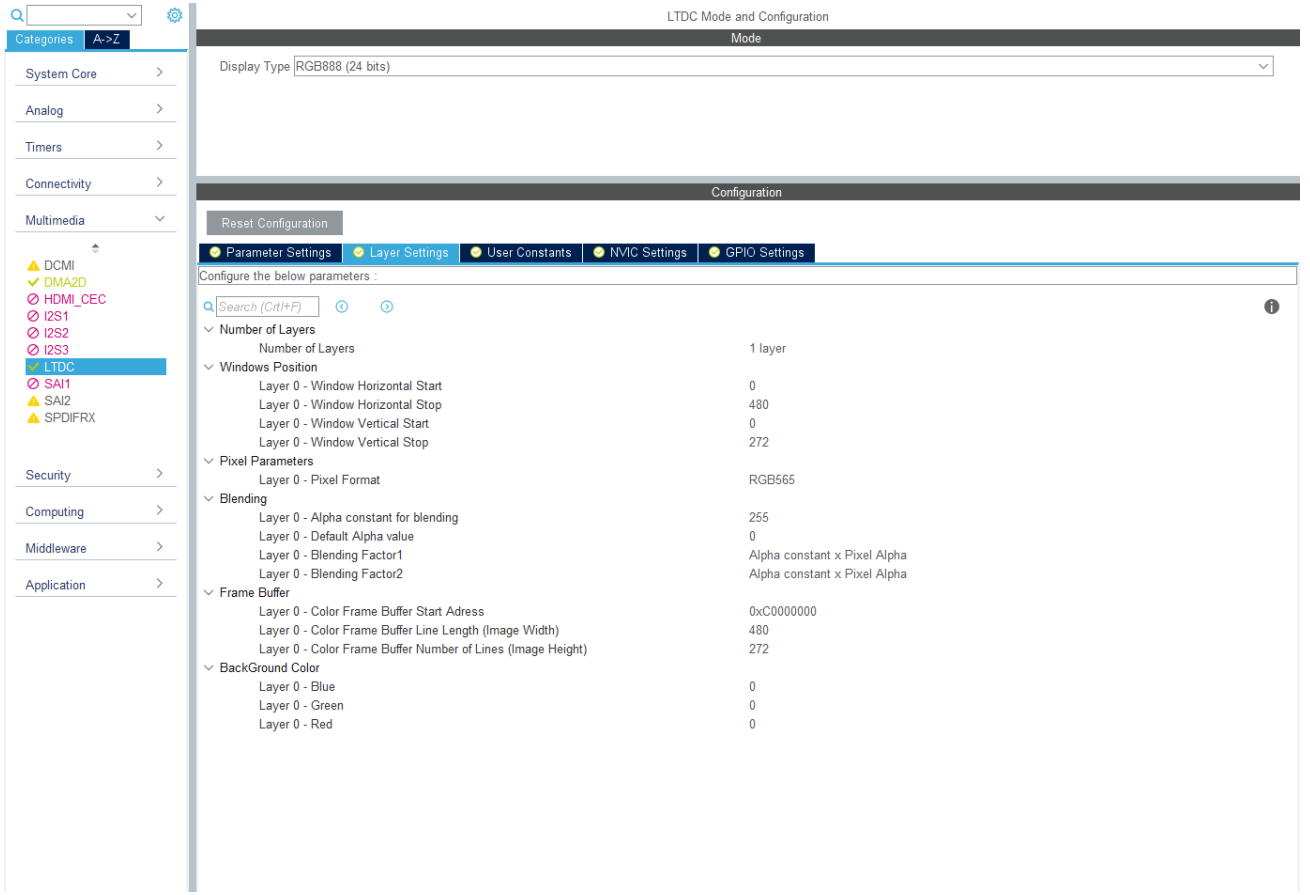
main.c

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



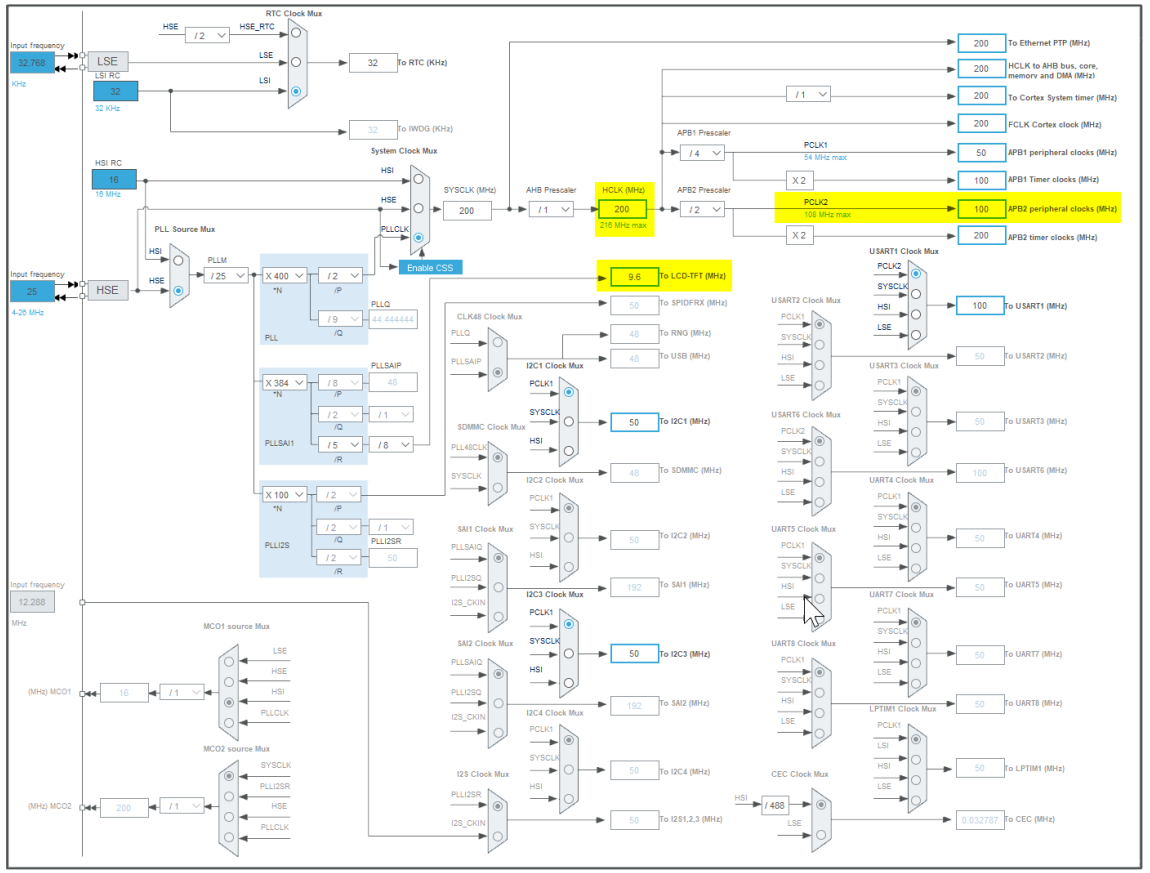
Board Bring Up

- 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



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



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

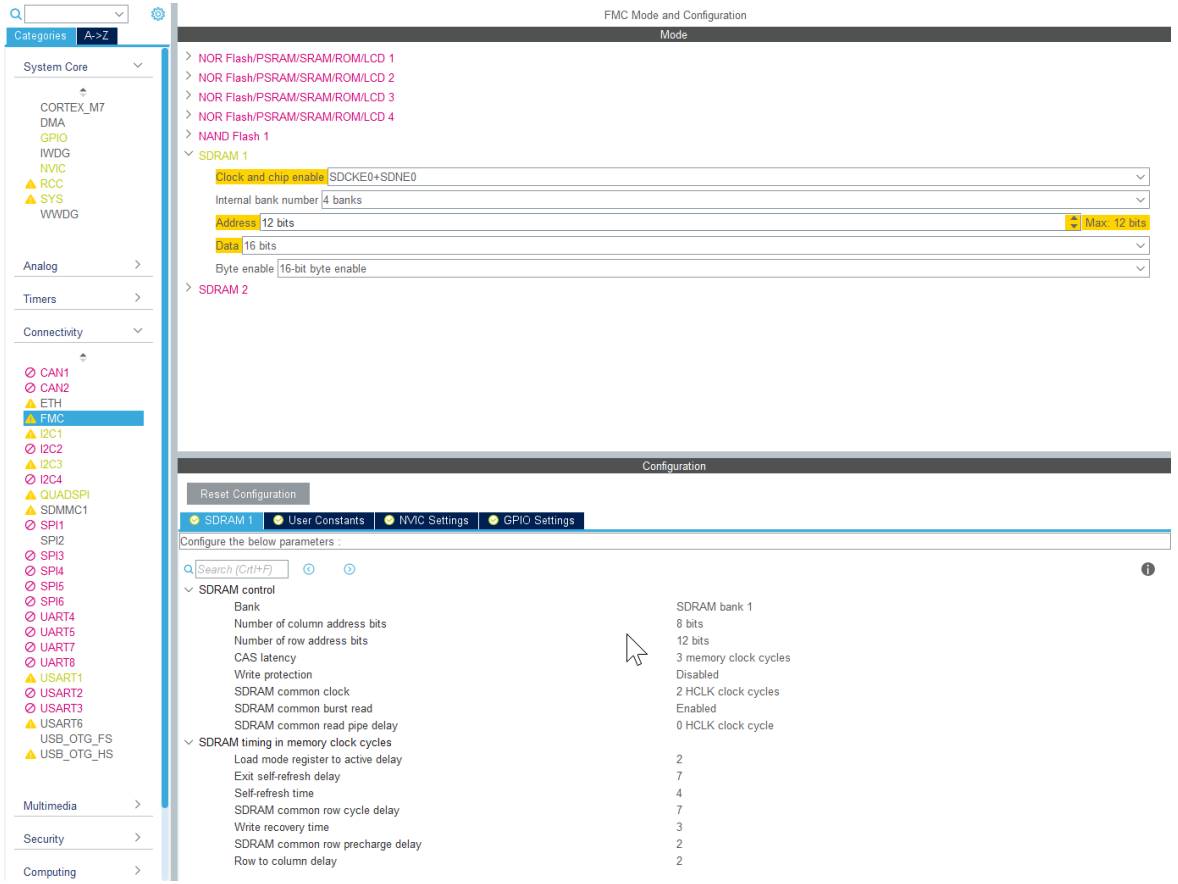
main.c

```
/* 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;
}
```

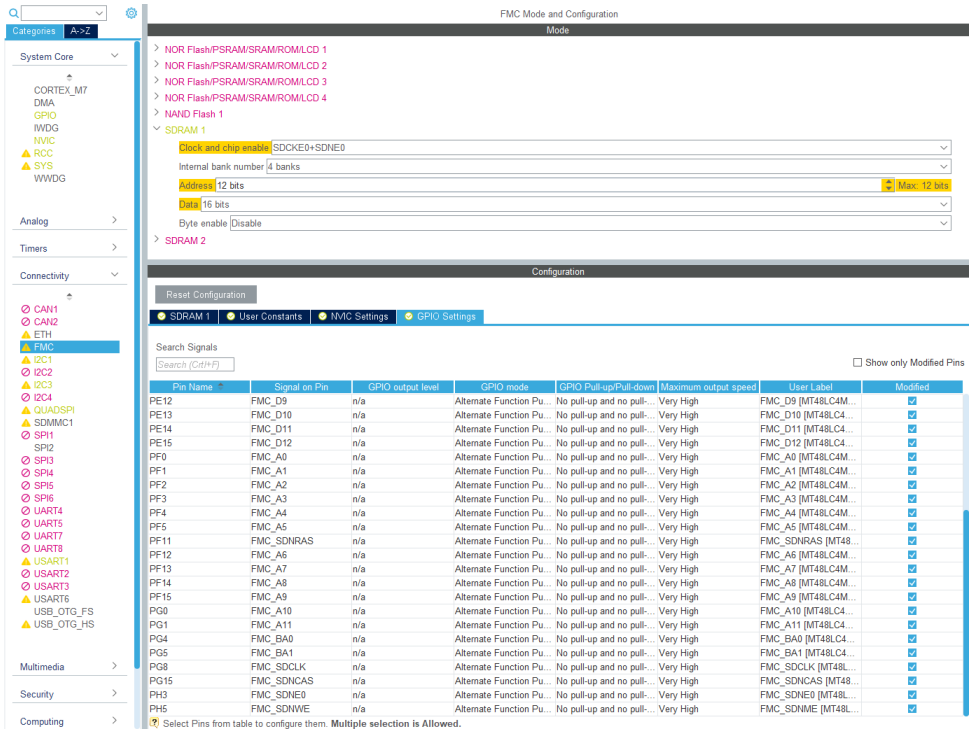
Board Bring Up

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

/* SDRAM controller init
/* __weak function can b
BSP_SDRAM_MspInit(&Sdram
if(HAL_SDRAM_Init(&Sdram
{
  sdramstatus = SDRAM_ERI
}
else
{
  sdramstatus = SDRAM_OK
}

/* SDRAM initialization
BSP_SDRAM_Initialization

return sdramstatus;
}
```

Memory 1

Go to 0xd0000000 Memory16

0xcfffff90	--------	--------	--------	--------
0xcfffffa0	--------	--------	--------	--------
0xcfffffb0	--------	--------	--------	--------
0xcfffffc0	--------	--------	--------	--------
0xcfffffd0	--------	--------	--------	--------
0xcfffffe0	--------	--------	--------	--------
0xcffffff0	--------	--------	--------	--------
0xd0000000	12345678	87654321	ABCDABC0	00000000
0xd0000010	00000000	00000000	00000000	00000000
0xd0000020	00000000	00000000	00000000	00000000
0xd0000030	00000000	00000000	00000000	00000000
0xd0000040	00000000	00000000	00000000	00000000
0xd0000050	00000000	00000000	00000000	00000000
0xd0000060	00000000	00000000	00000000	00000000
0xd0000070	00000000	00000000	00000000	00000000
0xd0000080	00000000	00000000	00000000	00000000

```
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

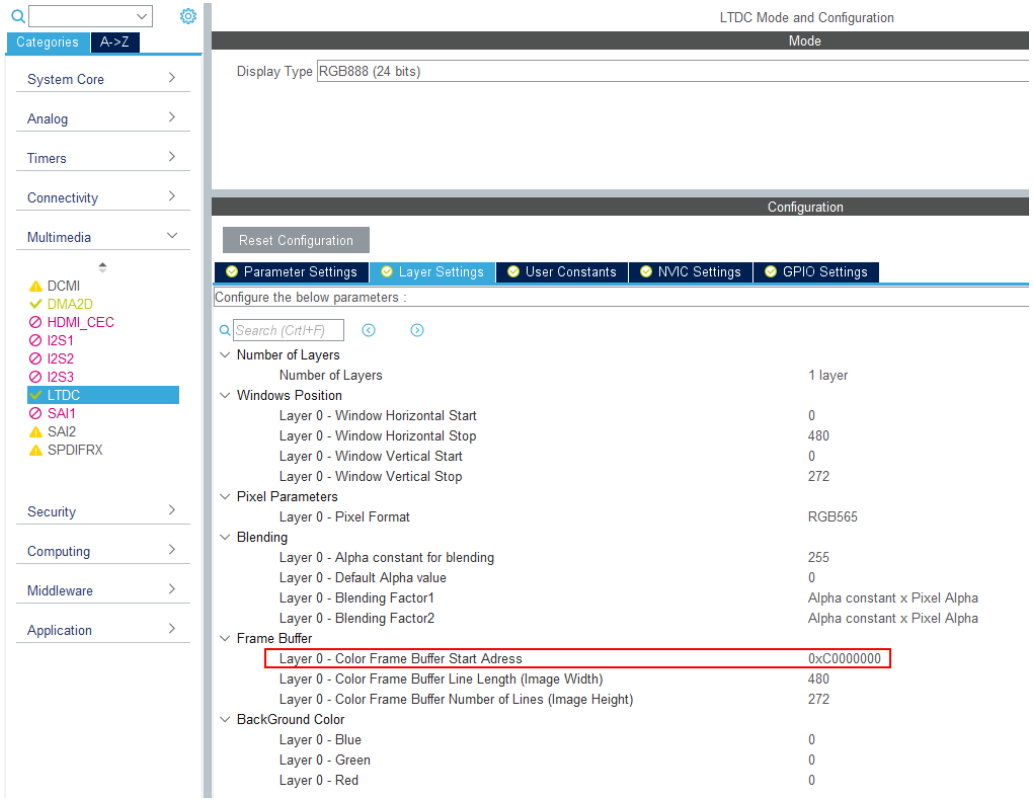
```
uint16_t* framebuffer = (uint16_t*)0xC0000000;  //16 bpp framebuffer
```

main.c

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

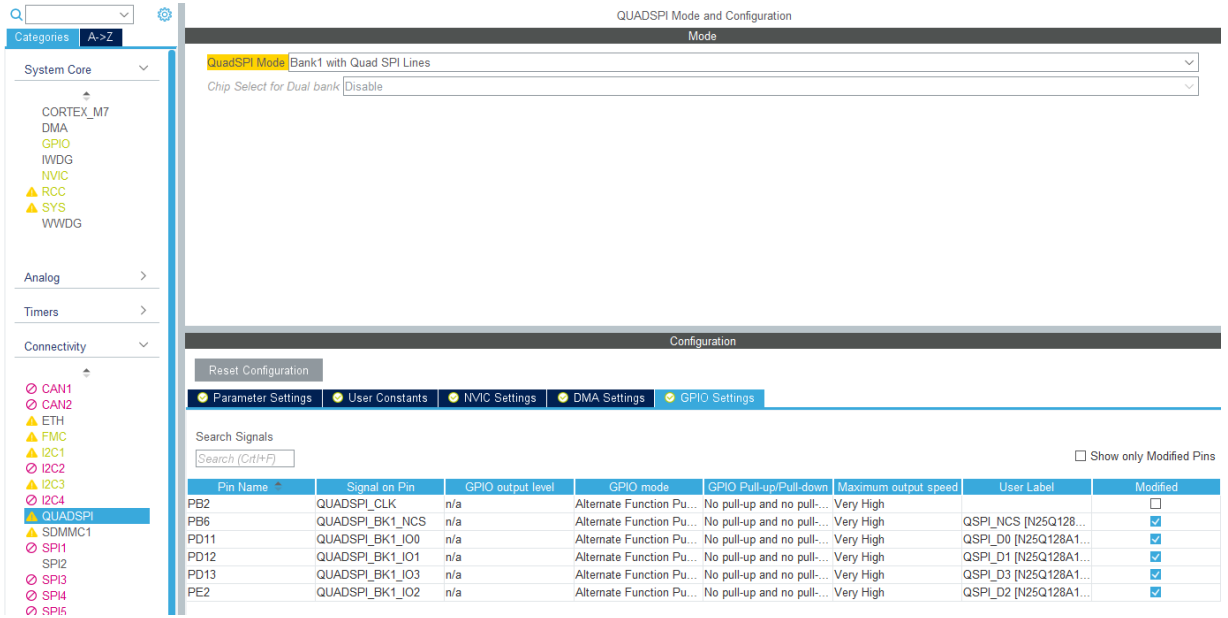
Board Bring Up

- 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



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



life.augmented

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;
}
```

life.augmented

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

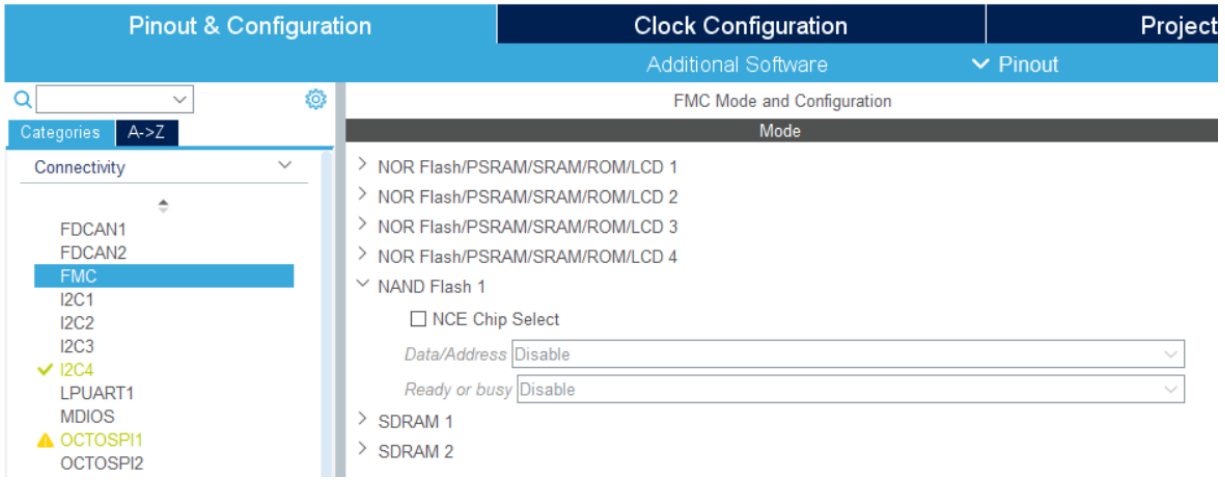
```
volatile uint32_t *externalFlash = 0x90000000;
const uint32_t size = 1000;
volatile uint32_t result = 0;

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

Board Bring Up

- 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

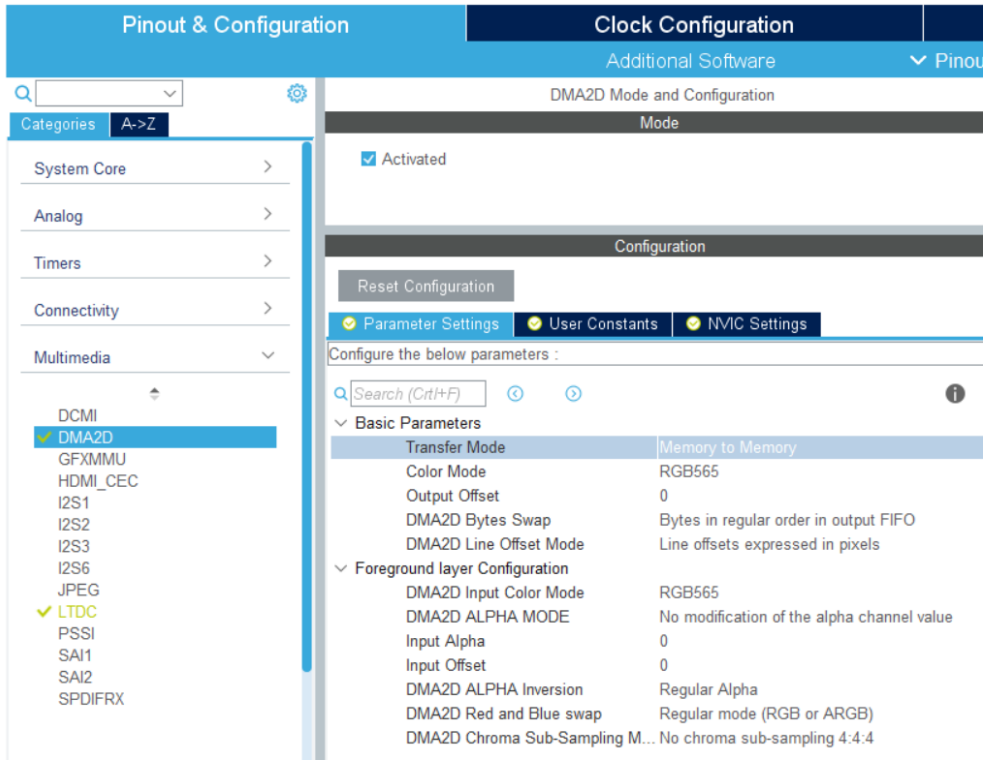
- 7. External flash in block mode
  - Do
    - Configure the NAND flash via STM32CubeMX via the FMC
    - Set the QSPI flash, similar to memory-mapped
      - Including GPIO
    - Add code to read from an address of the flash




```
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

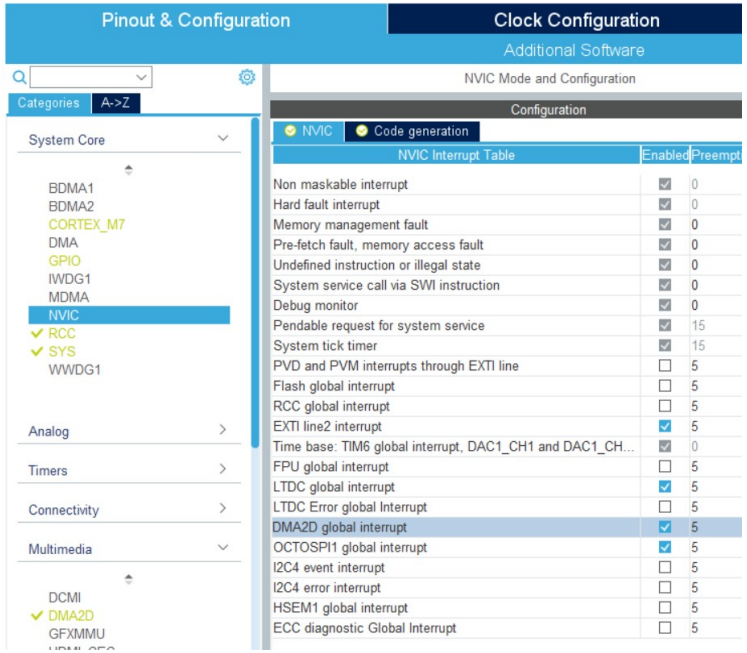
- 8. Hardware acceleration
  - Motivation
    - The Chrom-ART (DMA2D), has the possibility to drastically improving the graphical performance by transferring data from memory into the framebuffer
  - Goal
    - Enable Chrom-ART and read and write data using it
  - Prerequisites
    - MCU with Chrom-ART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

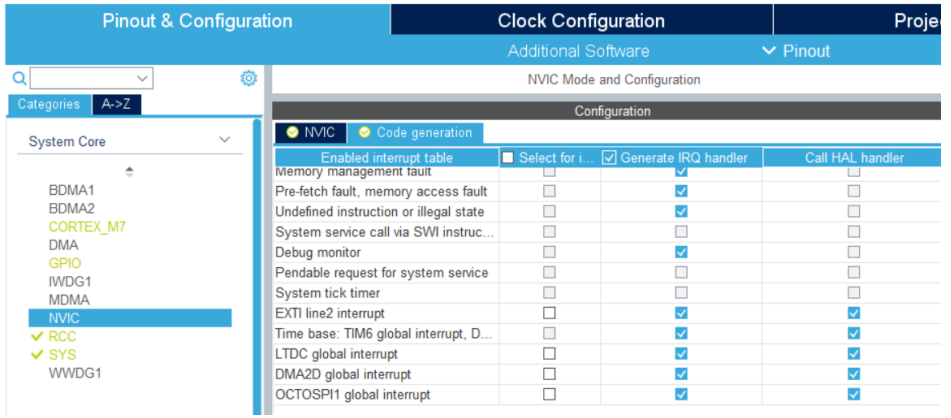


life.augmented

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





life.augmented

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

```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);
```

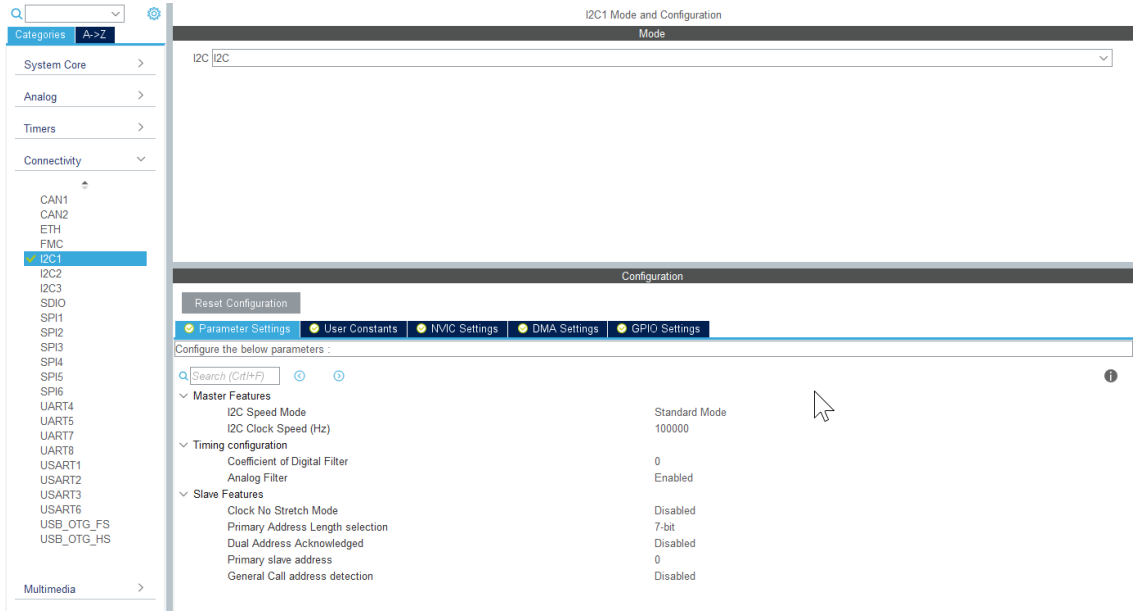
```c
    hdma2d.XferCpltCallback = DMA2D_XferCpltCallback;
```

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

```c
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

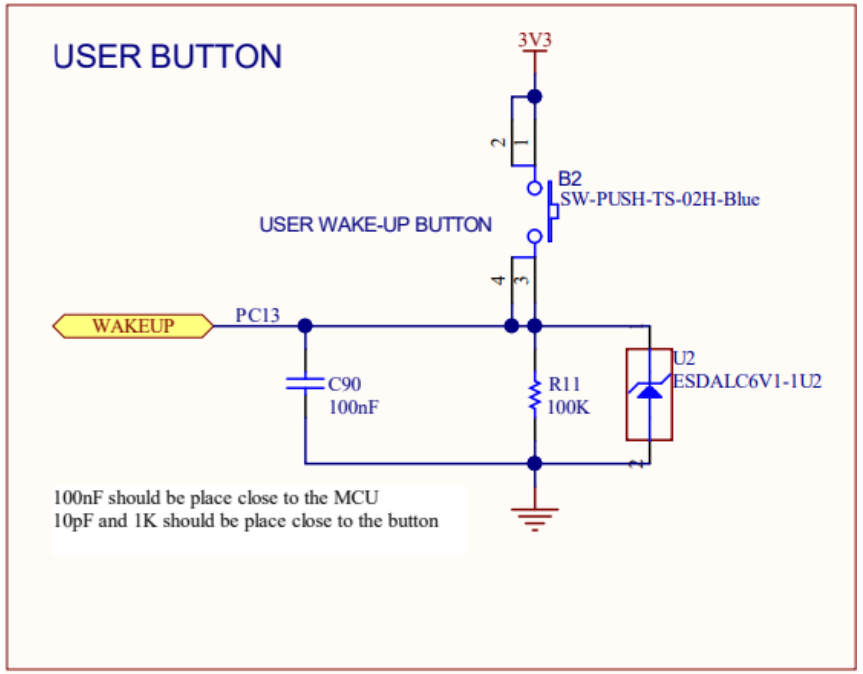


main.c

```
uint16_t x;
uint16_t y;

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

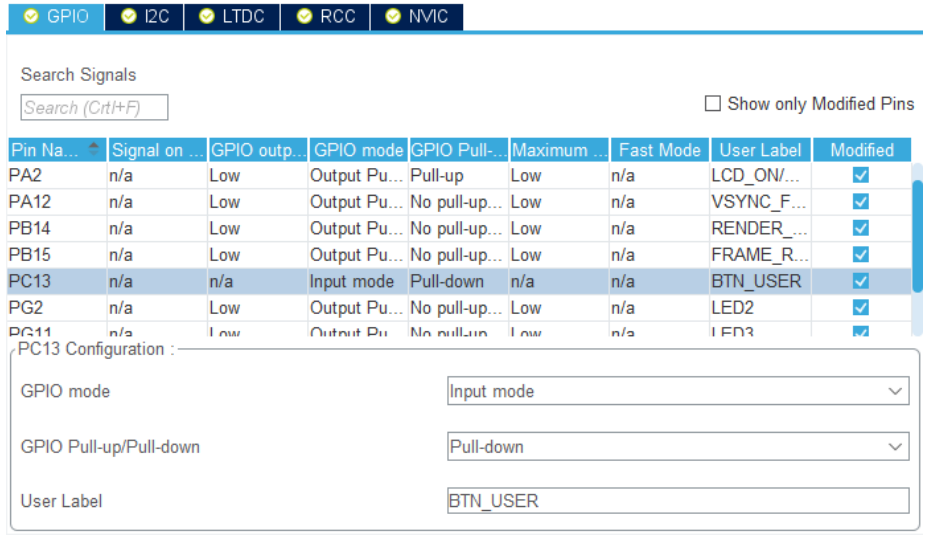
Board Bring Up

- 10. Physical Buttons
  - Motivation
    - Physical buttons can function as 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



main.c

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

Board Bring Up

- 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

life.augmented