

Board Bring Up Introduction

Emil Damkjaer Petersen

Agenda

- 1 Introduction
- 2 Hardware Selection
- 3 Board Bring-Up



Hardware Selection:

- [Preliminary Considerations](#) - contains several pointers to considerations you should take into account before moving on finding the right hardware.
- [Hardware Components](#) - contains information on the different components that makes up a hardware solution and what impact they have on a TouchGFX application.

Introduction

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



life.augmented

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](#)



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.

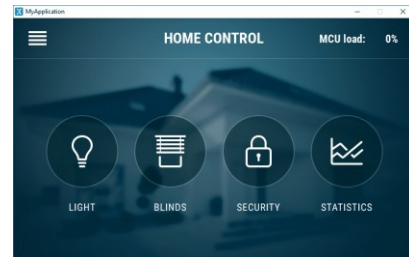
Hardware Selection



Hardware Selection

- Preliminary Considerations

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



Preliminary Considerations

8

Do you need high speed full screen transitions?

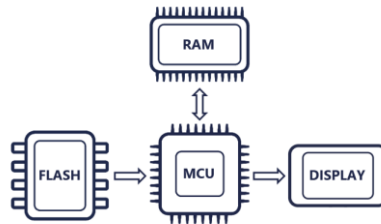
Do you need complex texture mapper animations like rotation and scaling?

Display Resolution = Framebuffer size

Presentation Embedded Graphics - Basic Concepts

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](#)



Documentation Link: [Embedded Graphics](#)

9

The Main parts

Flash

Static data Stores the image, Fonts and texts

Ram

Mainly for framebuffer, which is where the calculated image is stored, and transferd from, to the display

Display

Image from Ram (sizes, types, resolution)

MCU

Handles the heavy lifting, calculates the color, renders transfer images

Multiple set up with internal ram, flash, displays

Example: H750, external flash, ram, LTDC display

Hardware Selection

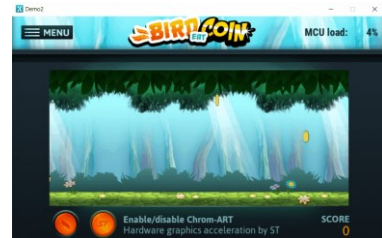
- MCU
 - Frequency
 - Embedded Hardware Acceleration
 - Chrom-ART
 - JPEG Accelerator
 - Chrom-GRC
 - Internal Memory
 - Ram
 - Flash
 - LCD Controller
 - Packages & I/O
 - Memory Interfacing



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 MPI-DSI	Up to 1024*768



Hardware Components: MCU 10



Notes:

Before board bring-up, we need HW, this section will help by understanding which hardware to choose based on the end application.

Use an STM32 DK or Eval as reference when doing the first pair of prototypes

Frequency:

Graphic Subsystem Frequency = internal busses, the frequency of the graphics accelerator and internal and external memories.

LCD Controller

High resolution = high speed interface

RGB-TFT and MPI-DSI interfaces are often used for higher resolutions, as the bandwidth is in many cases higher than SPI or parallel 8080/6800.

Memory Interfacing

STM32 offers different types of memory controllers to interface with different types of external flash and ram

LCD-TFT - thin-film-transistor, variant of LCD, common used type of display, and is also the most used on our example kits
MIP – memory in Pixels,

NOR Flash

Memory mapped mode

external flash is seen as an internal memory for read operations.

NAND Flash

high volume of graphical assets and faster write and erase operations.

cannot be configured in a memory mapped mode.

not recommended for code execution.

Using a cache in RAM is often necessary when using NAND flash.

eMMC

eMMC (Embedded Multi Media Card)

provides standard interfaces and management for the flash memory

RAM:

- Density
- Performance
- Power consumption
- Interface / pin size
- Framebuffer strategy

SRAM

SRAM typically has a faster access time compared to DRAM and is therefore more suitable for GUIs

SDRAM

physical space to store the same amount of data compared to SRAM
it requires more power compared to SRAM

PSRAM

PSRAM compared to traditional SDRAM and SRAM has the advantages of higher speed and lower power consumption.

Board Bring Up



life.augmented

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



STM32CubeMX is a graphical tool that allows a very easy configuration of STM32 microcontrollers and microprocessors, as well as the generation of the corresponding initialization 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

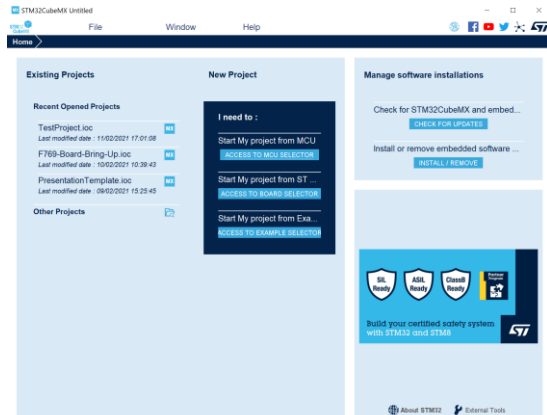


Motivation = Why

Goal = what expect to be done when finished

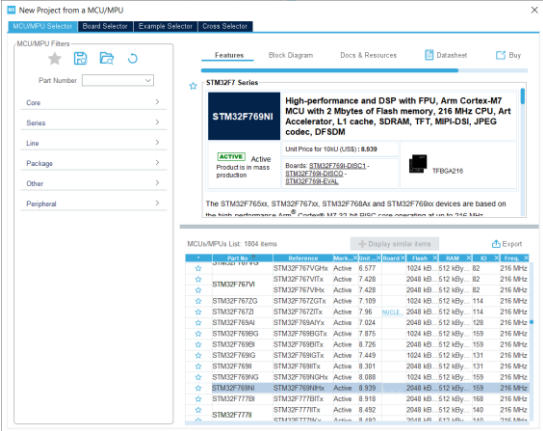
Board Bring Up

- 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

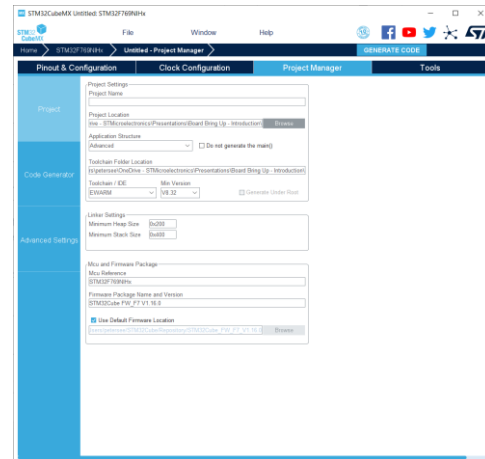
- 1. Create Project
 - Do
 - Create a project in STM32CubeMX based on the MCU on the Board
 - Select “Advance” under application structure, and don’t select “Do not generate the main()”
 - Select Toolchain/IDE
 - Generate Code and when done, open project
 - Compile and debug from IDE



How To: 1. Create Project

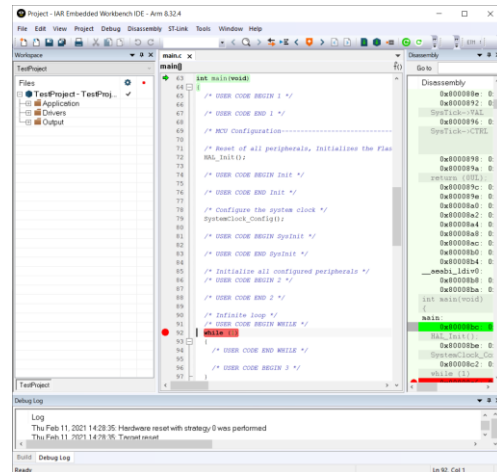
Board Bring Up

- 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



life.ougmented

How To: 1. Create Project

19

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(2000);  
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

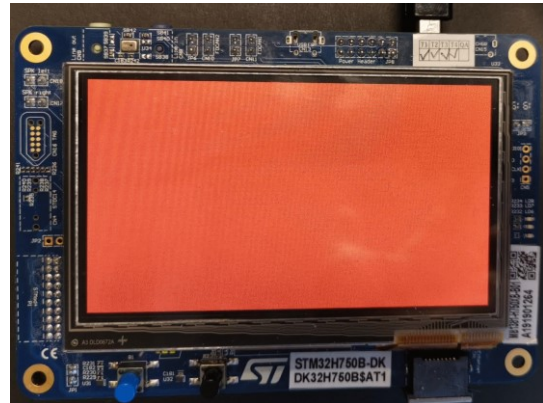
 Disable data cache on F7 and H7

```
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*)0x00000000;
98  uint32_t* stop = (uint32_t*)0x00200000;
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      /* loop loop loop ... */
114  }
```

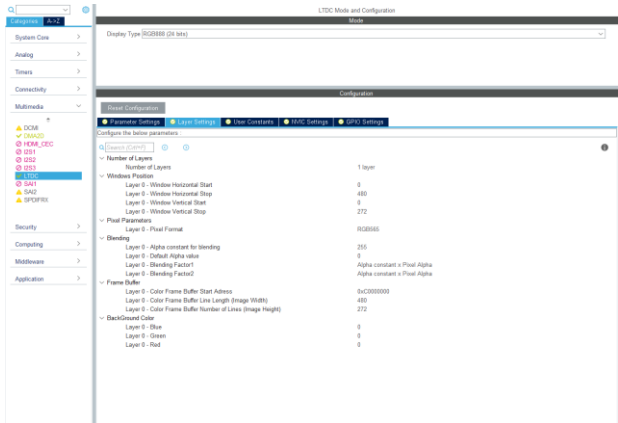
Board Bring Up

- 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

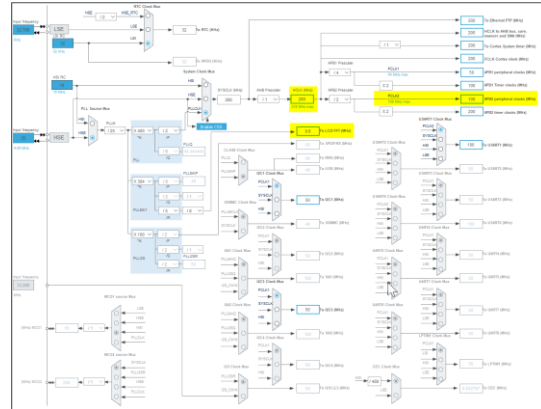
- 3. Framebuffer in internal RAM
 - Do
 - Parallel RGB Displays
 - Setup framebuffer in internal RAM
 - Configure the GPIO connections to the display
 - Configure the LTDC controller
 - Configure the LTDC pixel clock
 - Setting the framebuffer address
 - Check the framerate



life.augmented

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

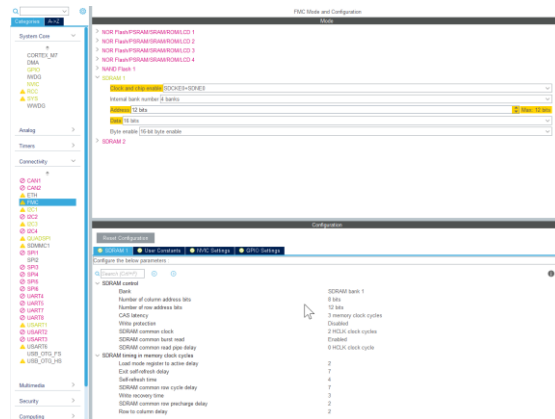
```
main.c
```

```
/* 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 < WH; 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



life.augmented

How To: 4. External RAM

27

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



```
pinmux
{
    EMC_SRAM0_CMD0DEF = Command;

    /* Step 1: configure a clock configuration table command */
    Command.CommandMode = FMC_SRAM_CMD_CLK_ENABLE;
    Command.CommandGroup = FMC_SRAM_CMD_ACCESS_ENABLE;
    Command.AutoRegisterDefinition = 1;
}

/* Send the command */
FMC_SRAM_CmdAccess(EMC_SRAM0_CMD0DEF, 0, Command, 0x00000000);
```

How To: 4. External RAM

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.ColumnBitNumber = FMC_SDRAM_COLUMN_BITS_NUM_8;
/* Column addressing: [11:0] */
SDramHandle.Init.RowBitNumber     = FMC_SDRAM_ROW_BITS_NUM_12;
SDramHandle.Init.MemoryDataWidth  = SDRAM_MEMORY_WIDTH;
SDramHandle.Init.InternalBankNumber = FMC_SDRAM_INTERNAL_BANKS_NUM_4;
SDramHandle.Init.CASLatency       = SDRAM_CAS_LATENCY;
SDramHandle.Init.WriteProtection  = FMC_SDRAM_WRITE_PROTECTION_DISABLE;
SDramHandle.Init.ClockPeriod      = SDLOCK_PERIOD;
SDramHandle.Init.ReadBurst        = SDRAM_READBURST;
SDramHandle.Init.ReadPipeDelay    = FMC_SDRAM_READPIPE_DELAY_1;
    
```

```

/* SDRAM controller initialization
 * weak function can be overridden
 */
BSP_SDRAM_MemInit(4*SDramHandle.Init.MemoryDataWidth)
{
    if (HAL_SDRAM_Init(&SDramHandle) != HAL_OK)
    {
        sdramstatus = SDRAM_ERR;
    }
    else
    {
        sdramstatus = SDRAM_OK;
    }
}

/* SDRAM initialization
BSP_SDRAM_Initiation
return sdramstatus;
    
```

```

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

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



file.ougmented

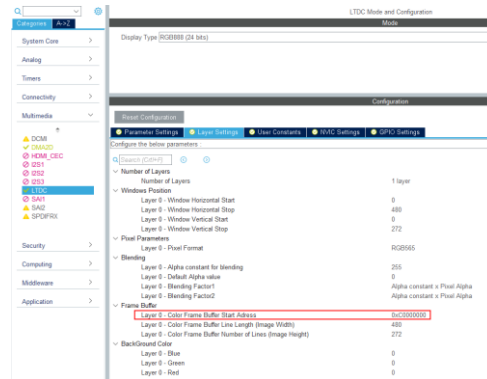
- 5. Framebuffer in external RAM
 - Motivation
 - Getting graphics on the display from External RAM
 - Goal
 - Remove Framebuffer from Internal RAM, and have the Framebuffer in external RAM
 - Prerequisites
 - Address of the framebuffer in the external RAM
 - Do
 - Place the framebuffer in external RAM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



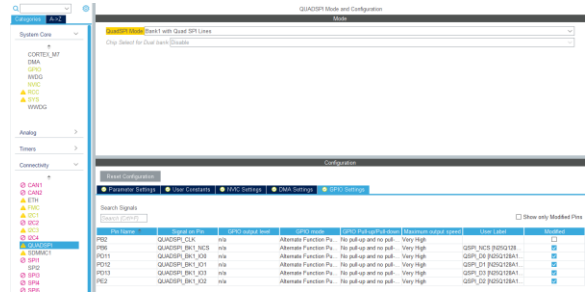
l1te.ougmented

How To: 5. Framebuffer in external RAM

31

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.ougmented

[How To: 6. External addressable flash](#)

32

• 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



life.ougmented

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_3_LINE;
s_command.Instruction     = QUAD_IHOUT_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     = H2SQ128A_DUMMY_CYCLES_READ_QUAD;
s_command.DdrMode         = QSPI_DDR_MODE_DISABLE;
s_command.DdrHoldingCycle = 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



life.augmented

Setup Memory Mapped mode in code

```
volatile uint32_t *externalFlash = 0x00000000;
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

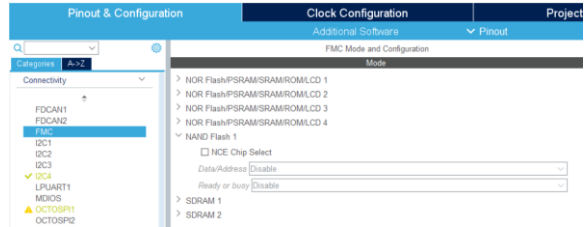


Board Bring Up

- 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);
```

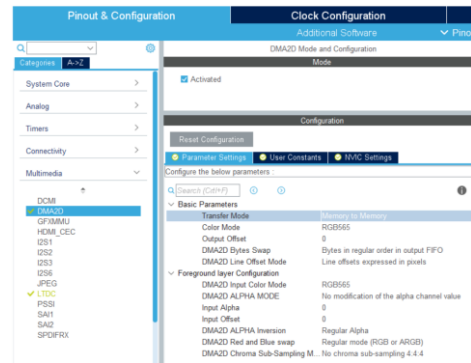


Board Bring Up

- 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



life.ougmented



How To: 8. Hardware acceleration

37

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



life.ougmented



How To: 8. Hardware acceleration

- 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



life.augmented

```
main.c
#include "stm32fxx_hal.h"
#include "stm32fxx_hal_dma2d.h"
...
uint32_t color = 0xF000; //Red in RGB565
dma2d.Instance->Mode = DMA2D_R2M;
dma2d.Instance->ColorMode = DMA2D_RGB565;
MODIFY_REG(dma2d.Instance->CR, DMA2D_CR_MODE, (DMA2D_R2M));
MODIFY_REG(dma2d.Instance->OPFCCR, DMA2D_OPFCCR_CM, (DMA2D_RGB565));
MODIFY_REG(dma2d.Instance->NORM, DMA2D_NORM_ID, (displayWidth - rectangleWidth));
dma2d.LayerCfg[0].InputColorMode = CM_RGB565;
dma2d.LayerCfg[0].InputOffset = 0;
HAL_DMA2D_ConfigLayer(&dma2d, 0);
HAL_DMA2D_Start_IT(&dma2d, color, (unsigned int)0x1000, rectangleWidth, rectangleHeight);

dma2d.XferCbCallback = DMA2D_XferCbCallback;

extern "C" {
static void DMA2D_XferCbCallback(DMA2D_HandleTypeDef* handle)
{
//Ensure that you this callback is called
}
}

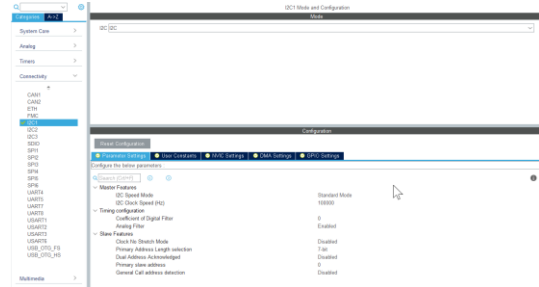
HAL_DMA2D_Start_IT(&dma2d,
(unsigned int)0xF000,
(unsigned int)0x1000,
displayWidth - mOffPixels);
```

How To: 8. Hardware acceleration

39

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



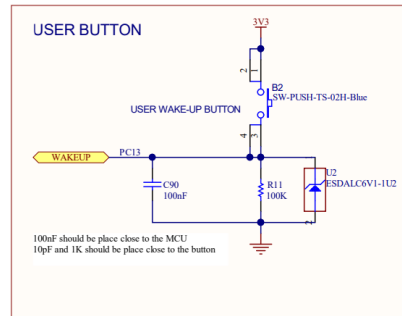
```
main.c\n\nuint16_t x;\nuint16_t y;\n\nTouchControllerState state;\nif (myTouchController_getstate(&state))\n{\n    x = state.touchX;\n    y = state.touchY;\n    //break point here\n}
```



life.augmented

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



life.augmented

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

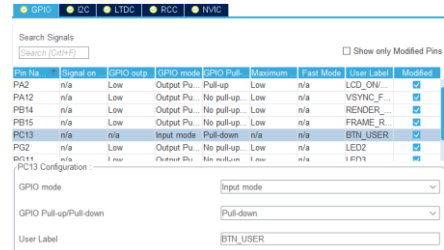
- Setup the application to receive input from a button

- Prerequisites

- Physical button connected to MCU

- Do

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



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



life.augmented

- 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



The compiler will compile the text, fonts, and images in your project and produce a binary or hex file with this data. This data is typically put into the external flash. The internal flash is then reserved for code.

The [STM32CubeProgrammer](#) comes with flash loaders for the various STM32 Evaluation kits.

The flash loader consists of two parts: Configuration of the GPIOs and flash interface that are required to communicate with the flash. The flashing algorithm that knows the sequence of commands required to write in the flash.

Another solution is to include flash loading into the application itself. The idea is that you already have the flash configuration inside your application (to be able to load from it),

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.

