Implementation of use cases using GPDMA with different STM32 peripherals : PSSI and DCMI on STM32U5

Summer Internship

Supervised by:
Mme.Kaouther DJEMEL

Elaborated by:
Jaafer Hosni

Table of Contents

01 Company Presentation

02 Tools & Concepts

03 DCMI Use Case Implementation

04 PSSI Use Case Implementation

05 Project Deployment

06 Conclusion

01

Company Presentation

Introduction to the company

Key facts

- Global Semiconductor Manufacturer
- leading supplier of semiconductors for the automotive industry
- Diverse Product Portfolio

Services & Products

- Semiconductor Components
- Microcontrollers and Processors
- Micro-Electro-Mechanical Systems
- Analog and Mixed-Signal ICs

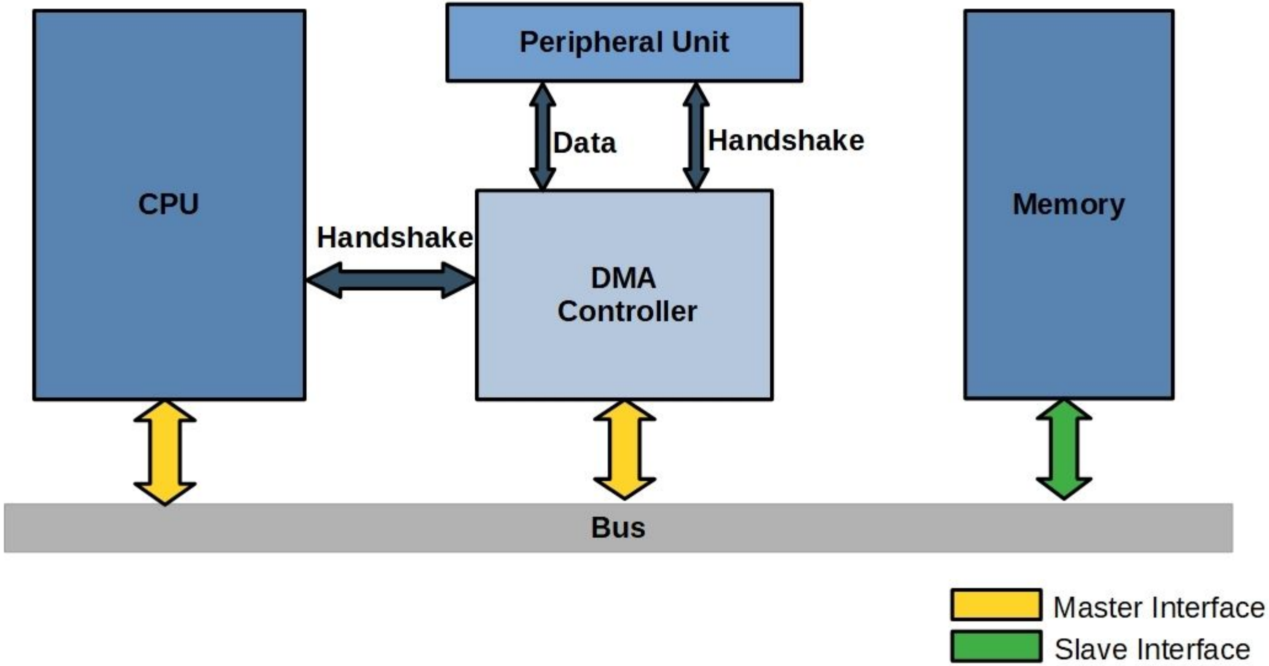
Goals

- Innovation and Technology Leadership
- Product Diversification
- Sustainability and Environmental Responsibility
- Financial Performance

02

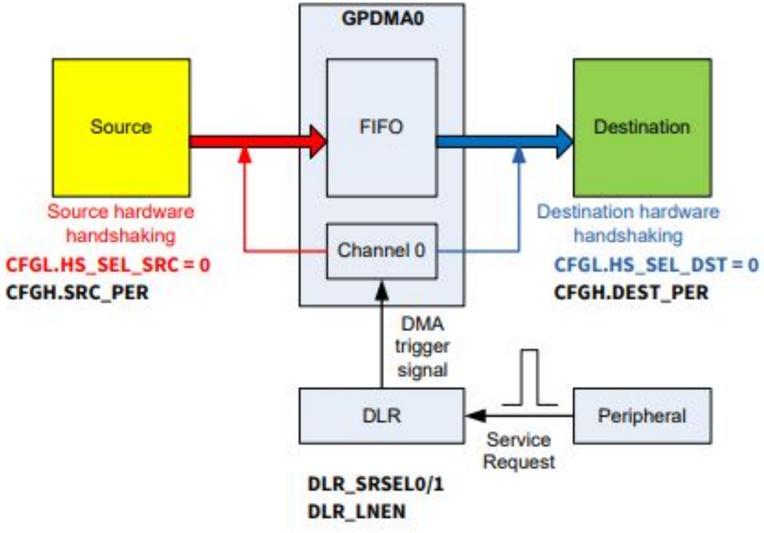
Tools & Concepts

General-Purpose Direct Memory Access

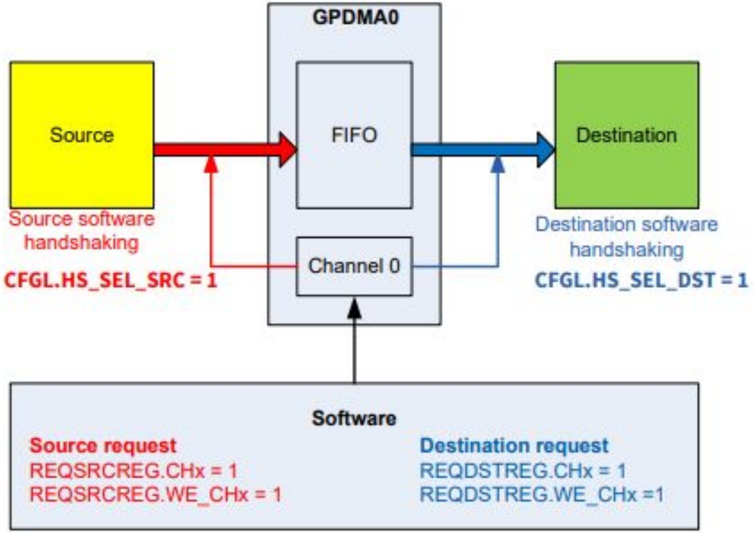


GPDMA Architecture Diagram

General-Purpose Direct Memory Access



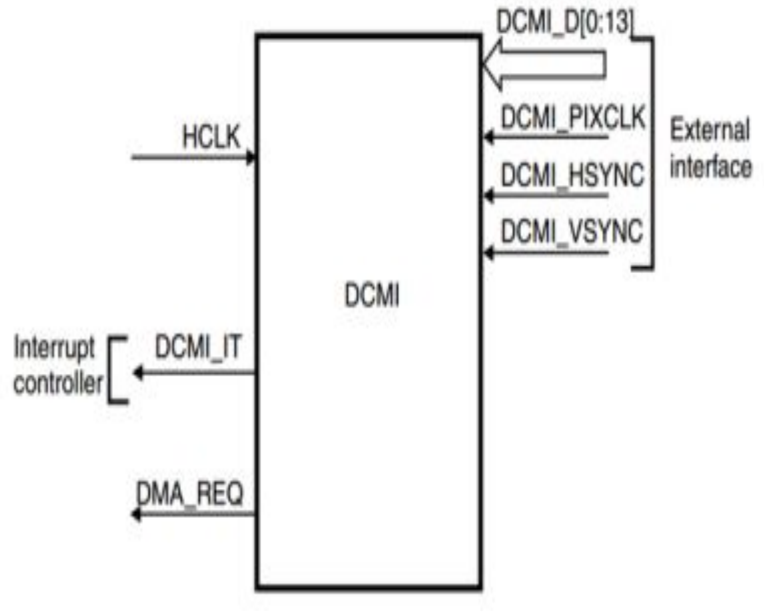
Hardware Handshake



Software Handshake

Digital Camera Memory Interface

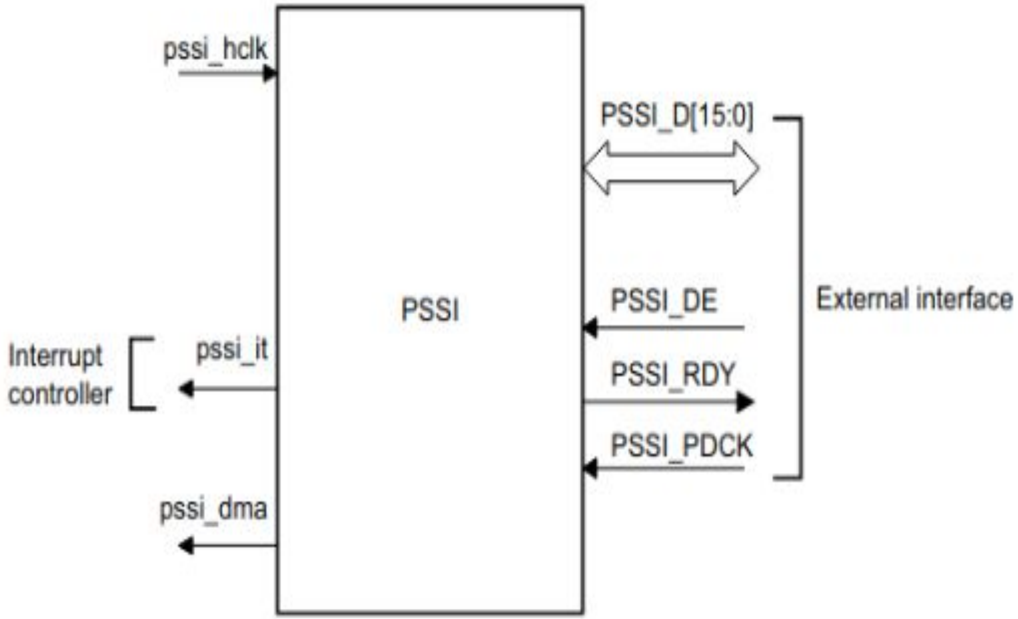
- Simplifies the integration of cameras into embedded system
- Serves as a bridge between the microcontroller and the image sensor
- Enables the capture and transmission of digital image data
- Handling tasks such as data synchronization, pixel data retrieval, and control signals



DCMI Architecture Diagram

Parallel Synchronous Slave Interface

- A communication protocol and hardware interface
- Responds to commands and data requests from a master device
- Facilitates synchronized data exchange between the slave and the master in a parallel fashion
- Maintains data integrity in real-time or high performance scenarios

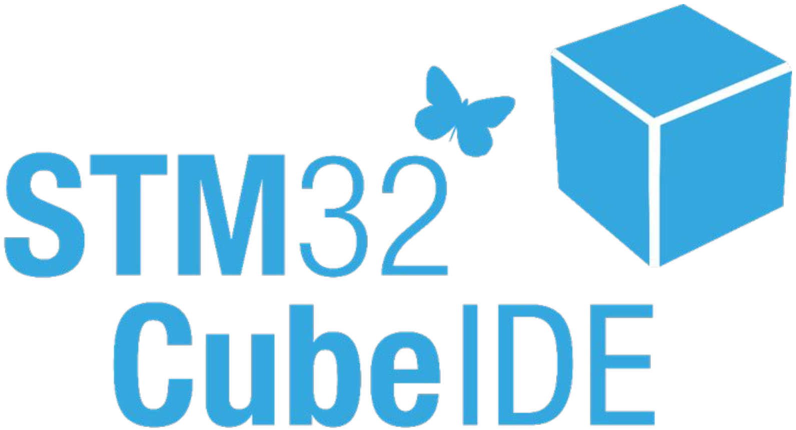


PSSI Architecture Diagram

ARM Cortex-M Series

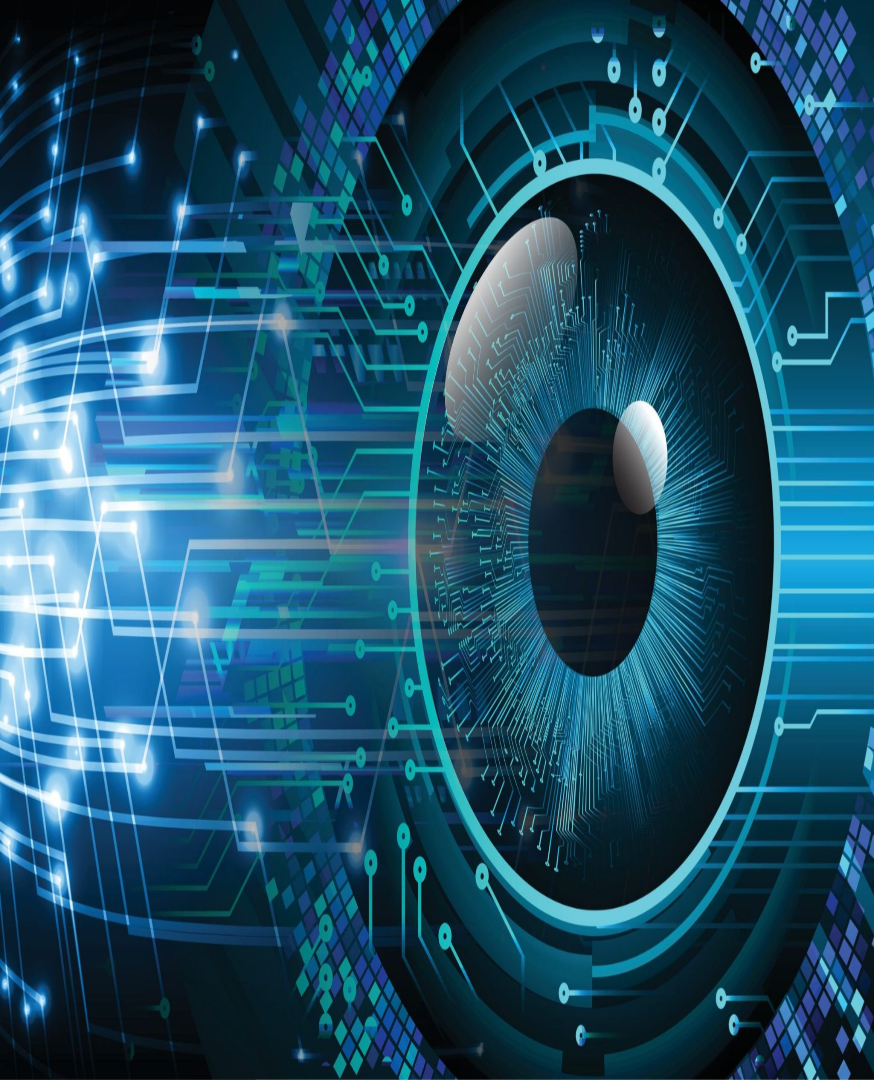
- Highly efficient and power-optimized microcontrollers
- Ideal for a wide range of embedded applications, from IoT devices to automotive systems
- Supported by a robust ecosystem and a comprehensive set of development tools like CMSIS
- Includes a range of integrated peripherals

Development environment



03

DCMI Use Case Implementation



Project Objectives

- Demonstrate STM32U5's real-time image acquisition and display capabilities by capturing RGB data in RGB565 format with QVGA resolution

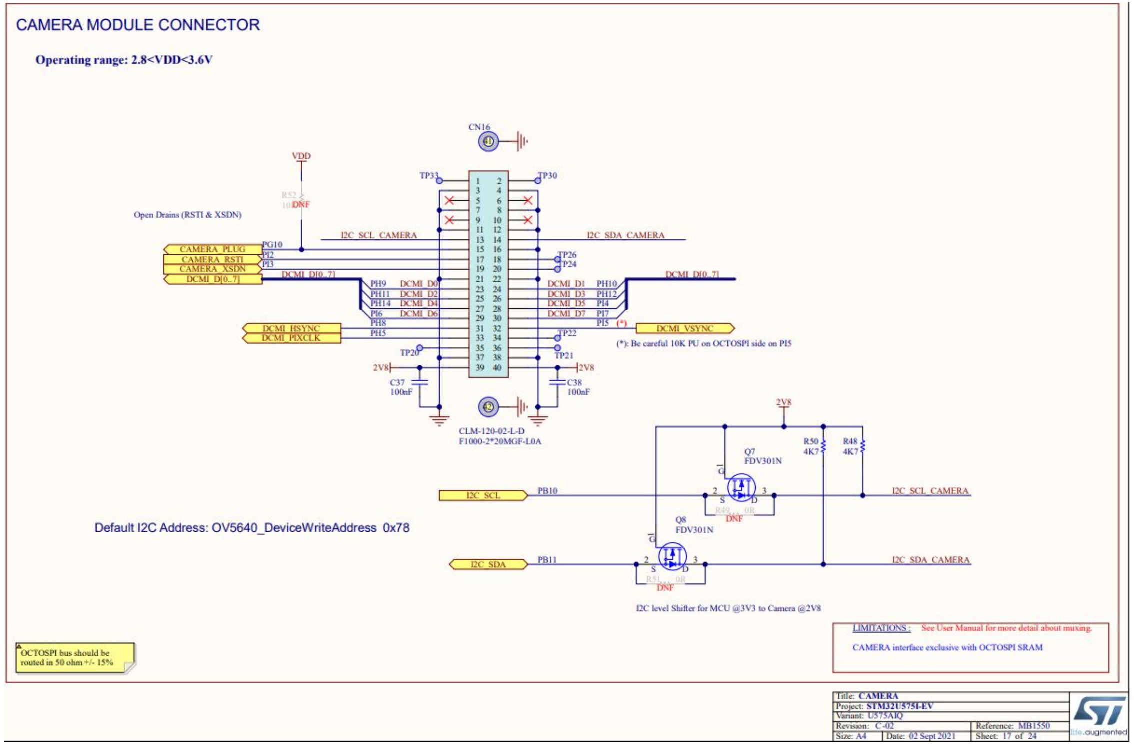
- Showcase the adaptability of STM32U5 to various data formats(YCrCb and Y-Only) as well as recording data in SDRAM for diverse application suitability

- Highlighting applicability for video processing and compression applications

STM32U575I_EV

- Features a powerful Arm Cortex-M33 core running at up to 480 MHz, providing high-performance computing capabilities
- Offers a large amount of flash memory and RAM, which makes it ideal for image processing applications
- This board is a great choice for projects that require a Digital Camera Interface to interact with image sensors.

MB1379A Camera Module



CAMERA MODULE CONNECTOR

Operating range: 2.8<VDD<3.6V

Open Drains (RSTI & XSDN)

(*): Be careful 10K PU on OCTOSPI side on PI5

Default I2C Address: OV5640_DeviceWriteAddress 0x78

I2C level Shifter for MCU @3V3 to Camera @2V8

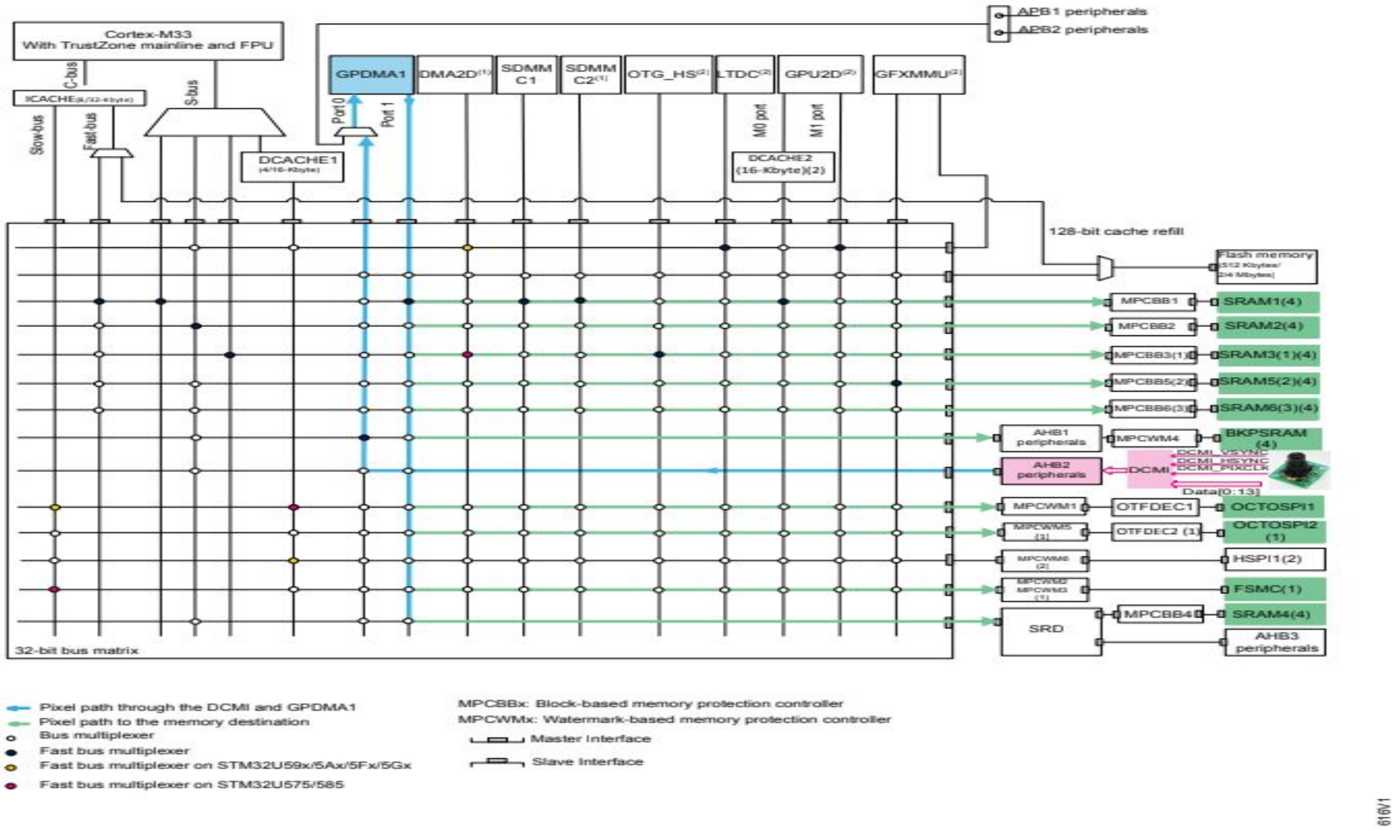
LIMITATIONS : See User Manual for more detail about muxing.

CAMERA interface exclusive with OCTOSPI SRAM

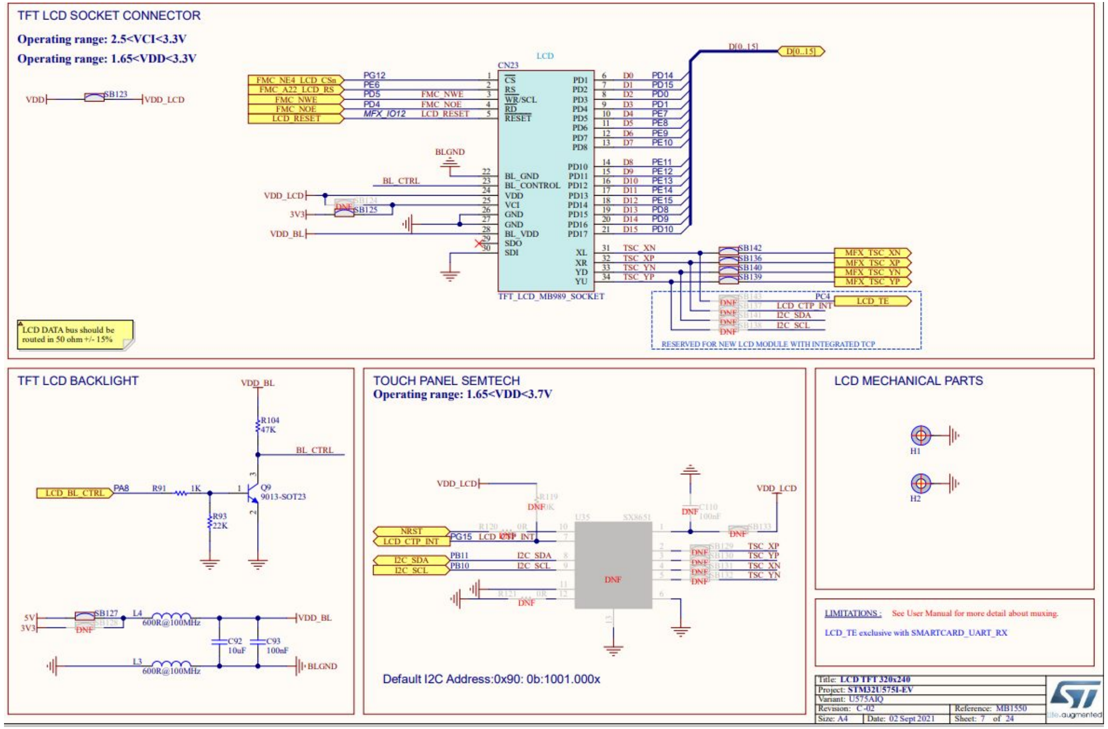
OCTOSPI bus should be routed in 50 ohm +/- 15%

Title: CAMERA		
Project: STM32U575I-EV		
Variant: U575AIQ		
Revision: C-02		Reference: MB1550
Size: A4	Date: 02 Sept 2021	Sheet: 17 of 24

MB1379A Camera Module



MB989C LCD Module



TFT LCD SOCKET CONNECTOR

Operating range: 2.5<VCI<3.3V

Operating range: 1.65<VDD<3.3V

LCD DATA bus should be routed in 50 ohm +/- 15%

RESERVED FOR NEW LCD MODULE WITH INTEGRATED TCP

TFT LCD BACKLIGHT

TOUCH PANEL SEMTECH

Operating range: 1.65<VDD<3.7V

Default I2C Address:0x90: 0b:1001.000x

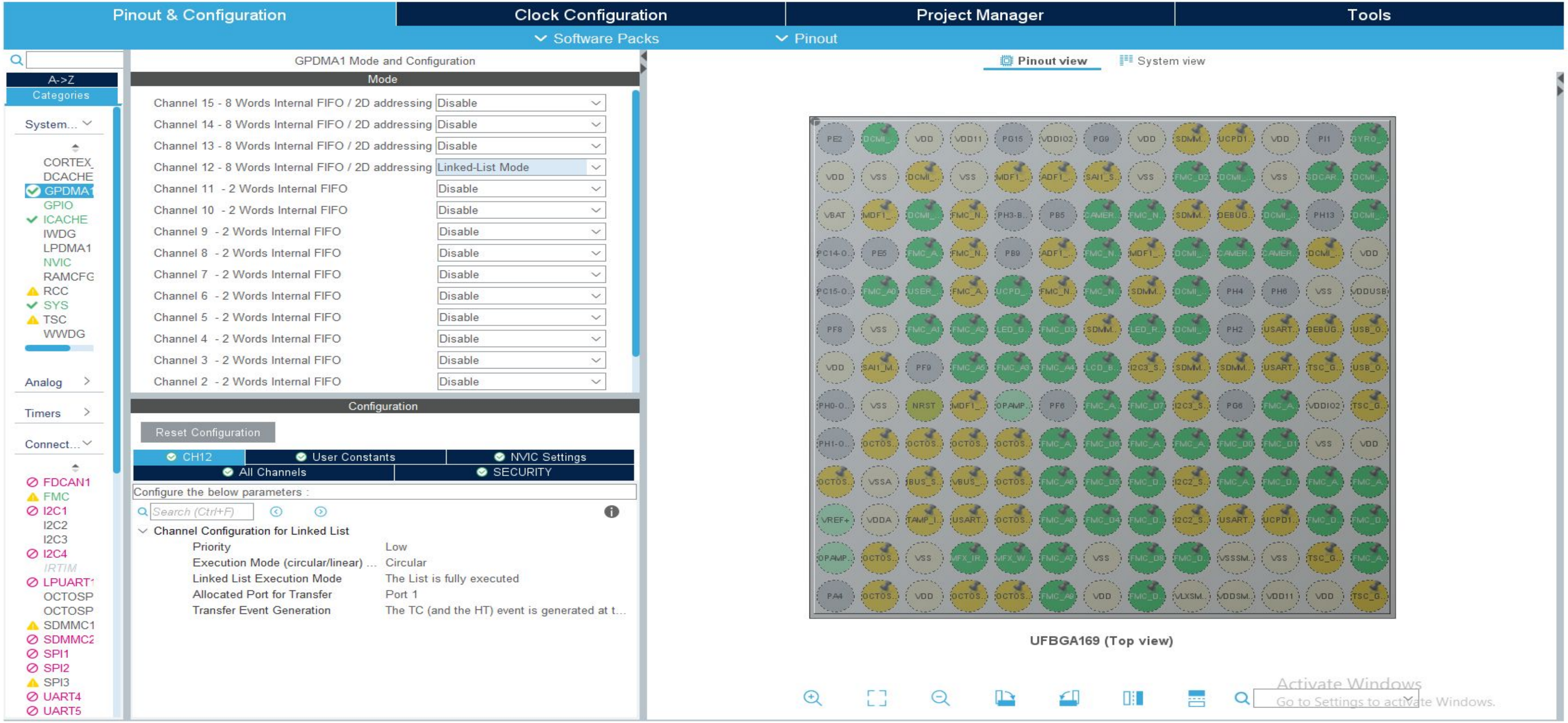
LCD MECHANICAL PARTS

LIMITATIONS : See User Manual for more detail about muxing.

LCD_TE exclusive with SMARTCARD_UART_RX

Title: **LCD TFT 320x240**			
Project: **STM32U575I-EV**			
Variant: U575AIQ			
Revision: C-02		Reference: MB1550	
Size: A4	Date: 02 Sept 2021	Sheet: 7 of 24	

GPDMA Configuration



GPDMA1 Mode and Configuration

Mode

Channel 15 - 8 Words Internal FIFO / 2D addressing	Disable
Channel 14 - 8 Words Internal FIFO / 2D addressing	Disable
Channel 13 - 8 Words Internal FIFO / 2D addressing	Disable
Channel 12 - 8 Words Internal FIFO / 2D addressing	Linked-List Mode
Channel 11 - 2 Words Internal FIFO	Disable
Channel 10 - 2 Words Internal FIFO	Disable
Channel 9 - 2 Words Internal FIFO	Disable
Channel 8 - 2 Words Internal FIFO	Disable
Channel 7 - 2 Words Internal FIFO	Disable
Channel 6 - 2 Words Internal FIFO	Disable
Channel 5 - 2 Words Internal FIFO	Disable
Channel 4 - 2 Words Internal FIFO	Disable
Channel 3 - 2 Words Internal FIFO	Disable
Channel 2 - 2 Words Internal FIFO	Disable

Configuration

Reset Configuration

CH12 | User Constants | NVIC Settings | All Channels | SECURITY

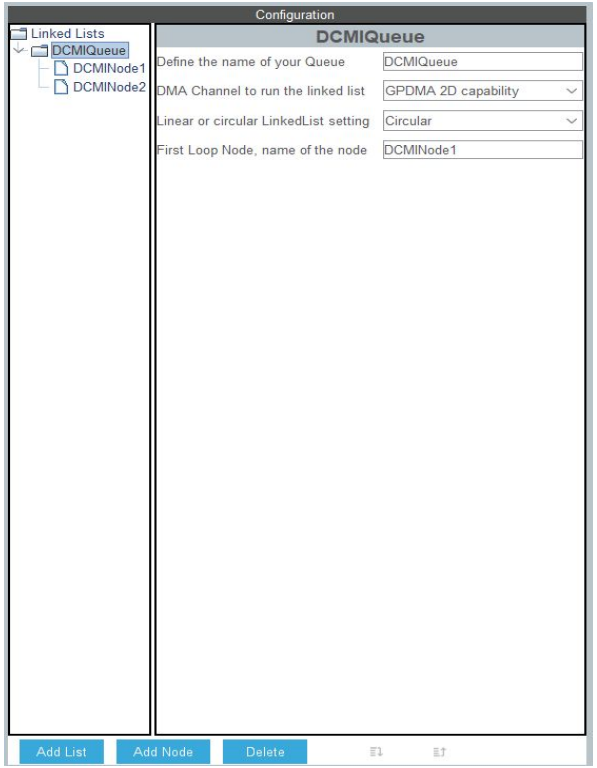
Configure the below parameters :

Search (Ctrl+F)

Channel Configuration for Linked List

Priority	Low
Execution Mode (circular/linear) ...	Circular
Linked List Execution Mode	The List is fully executed
Allocated Port for Transfer	Port 1
Transfer Event Generation	The TC (and the HT) event is generated at t...

DCMI Channel Configuration



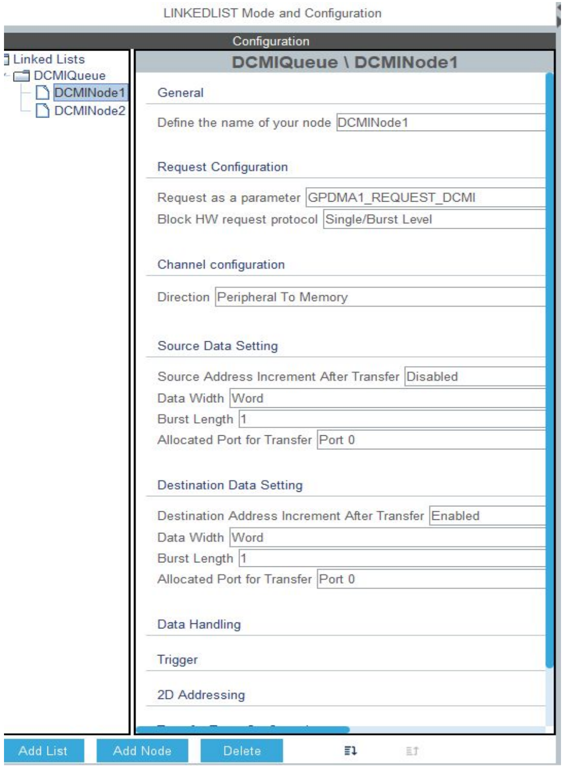
DCMIQueue

Define the name of your Queue: DCMIQueue

DMA Channel to run the linked list: GPDMA 2D capability

Linear or circular LinkedList setting: Circular

First Loop Node, name of the node: DCMINode1



DCMIQueue \ DCMINode1

General

Define the name of your node: DCMINode1

Request Configuration

Request as a parameter: GPDMA1_REQUEST_DCMI

Block HW request protocol: Single/Burst Level

Channel configuration

Direction: Peripheral To Memory

Source Data Setting

Source Address Increment After Transfer: Disabled

Data Width: Word

Burst Length: 1

Allocated Port for Transfer: Port 0

Destination Data Setting

Destination Address Increment After Transfer: Enabled

Data Width: Word

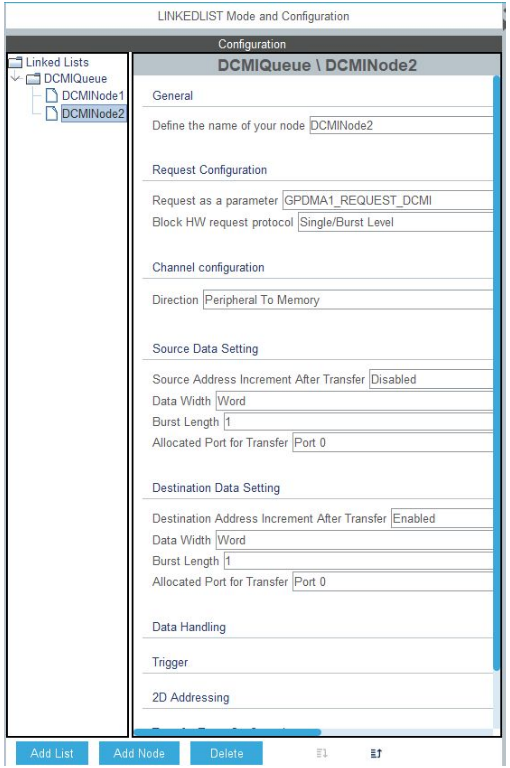
Burst Length: 1

Allocated Port for Transfer: Port 0

Data Handling

Trigger

2D Addressing



DCMIQueue \ DCMINode2

General

Define the name of your node: DCMINode2

Request Configuration

Request as a parameter: GPDMA1_REQUEST_DCMI

Block HW request protocol: Single/Burst Level

Channel configuration

Direction: Peripheral To Memory

Source Data Setting

Source Address Increment After Transfer: Disabled

Data Width: Word

Burst Length: 1

Allocated Port for Transfer: Port 0

Destination Data Setting

Destination Address Increment After Transfer: Enabled

Data Width: Word

Burst Length: 1

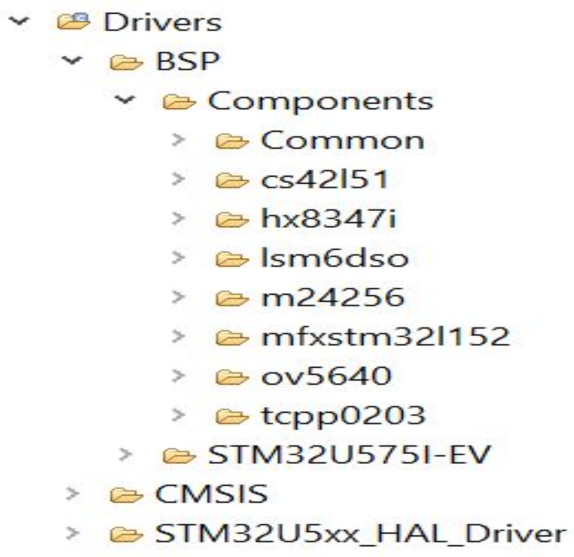
Allocated Port for Transfer: Port 0

Data Handling

Trigger

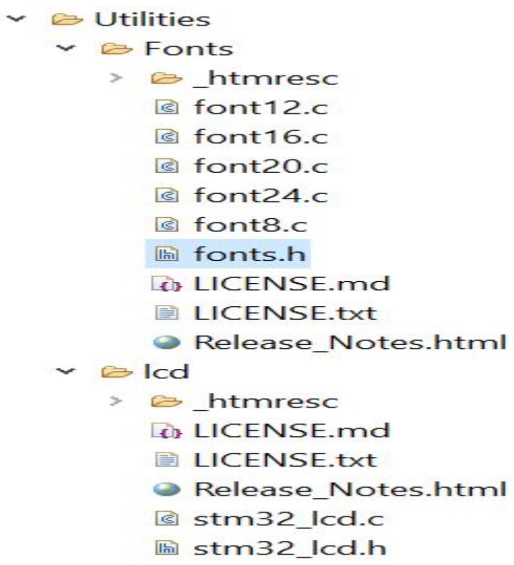
2D Addressing

Additional Software Packages



```
Drivers
  BSP
    Components
      Common
      cs42l51
      hx8347i
      lsm6dso
      m24256
      mfxstm32l152
      ov5640
      tcpp0203
    STM32U575I-EV
  CMSIS
  STM32U5xx_HAL_Driver
```

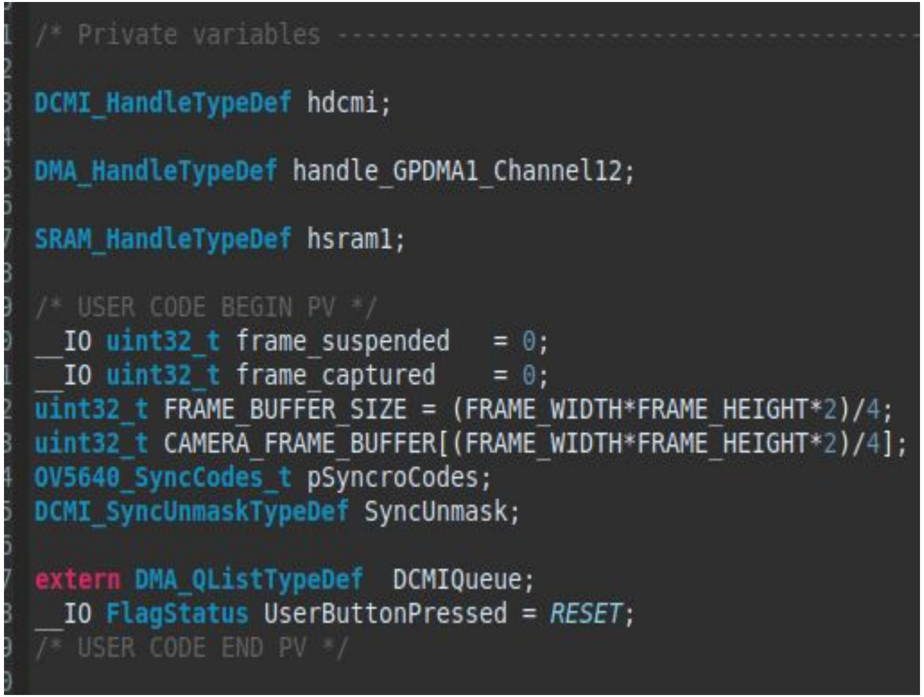
Board Support Package



```
Utilities
  Fonts
    _htmresc
    font12.c
    font16.c
    font20.c
    font24.c
    font8.c
    fonts.h
    LICENSE.md
    LICENSE.txt
    Release_Notes.html
  lcd
    _htmresc
    LICENSE.md
    LICENSE.txt
    Release_Notes.html
    stm32_lcd.c
    stm32_lcd.h
```

Utilities Package

Global Variables & HAL functions



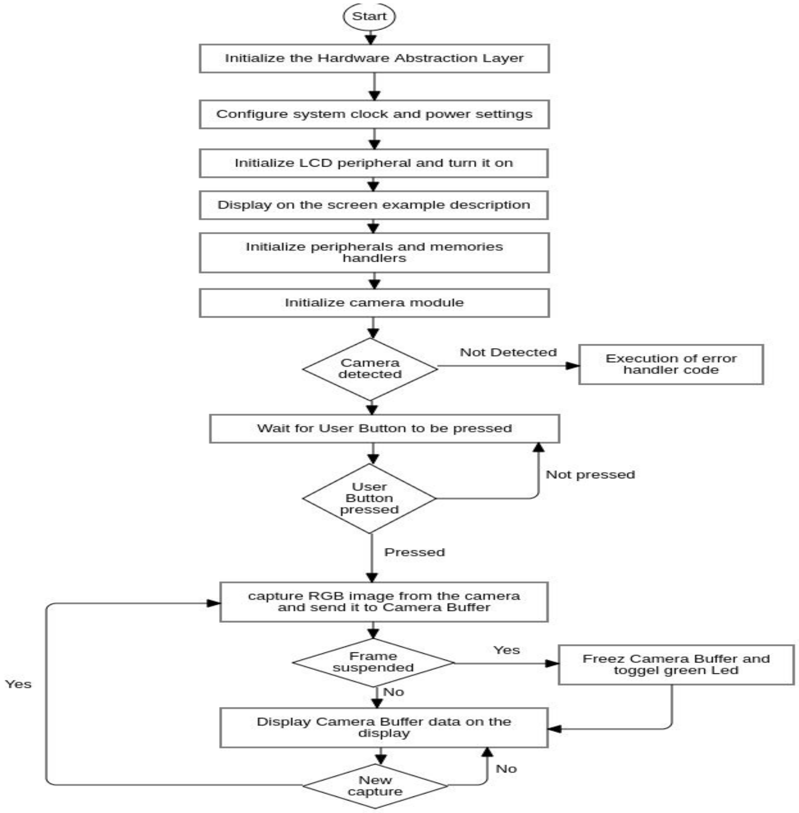
```
/* Private variables ---------------------------------------------

DCMI_HandleTypeDef hdcmi;

DMA_HandleTypeDef handle_GPDMA1_Channel12;

SRAM_HandleTypeDef hsram1;

/* USER CODE BEGIN PV */
__IO uint32_t frame_suspended   = 0;
__IO uint32_t frame_captured    = 0;
uint32_t FRAME_BUFFER_SIZE = (FRAME_WIDTH*FRAME_HEIGHT*2)/4;
uint32_t CAMERA_FRAME_BUFFER[(FRAME_WIDTH*FRAME_HEIGHT*2)/4];
OV5640_SyncCodes_t pSyncroCodes;
DCMI_SyncUnmaskTypeDef SyncUnmask;

extern DMA_QListTypeDef  DCMIQueue;
__IO FlagStatus UserButtonPressed = RESET;
/* USER CODE END PV */
```

- **HAL_DCMI_Start_DMA**(&hdcmi, DCMI_MODE_CONTINUOUS, (uint32_t)CAMERA_FRAME_BUFFER, FRAME_BUFFER_SIZE)

- **HAL_DCMI_Suspend**(&hdcmi)

- **HAL_DCMI_Resume**(&hdcmi)

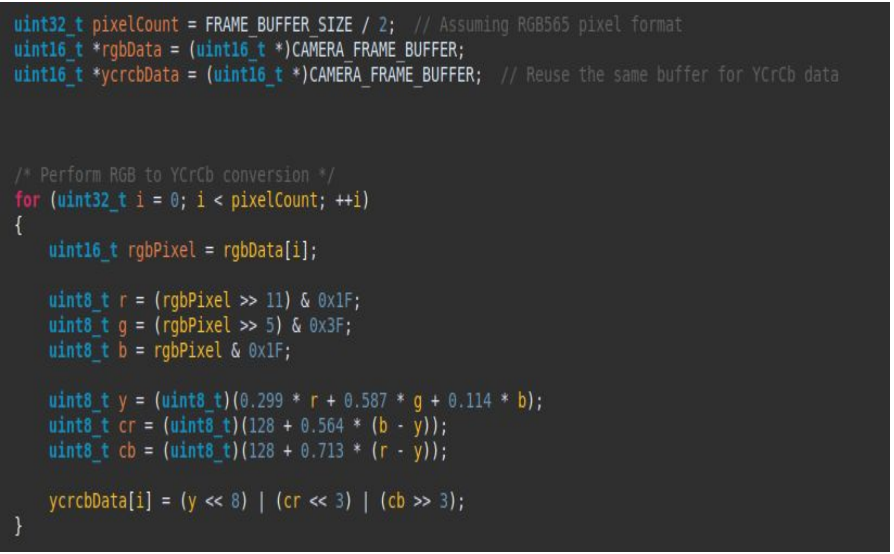
RGB Application


Start
Initialize the Hardware Abstraction Layer
Configure system clock and power settings
Initialize LCD peripheral and turn it on
Display on the screen example description
Initialize peripherals and memories handlers
Initialize camera module
Camera detected
Not Detected
Execution of error handler code
Wait for User Button to be pressed
User Button pressed
Not pressed
Pressed
capture RGB image from the camera and send it to Camera Buffer
Frame suspended
Yes
Freez Camera Buffer and toggel green Led
No
Yes
Display Camera Buffer data on the display
New capture
No

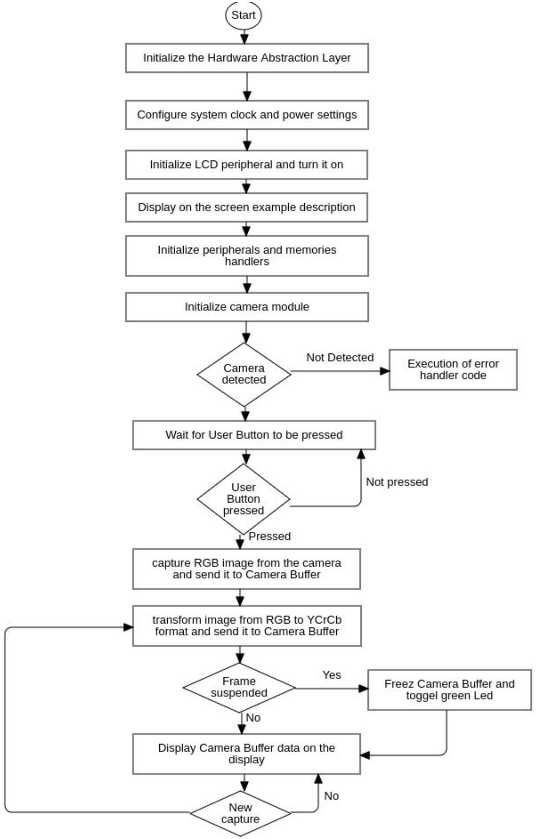
YCrCb Application

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.500 \\ 0.500 & -0.419 & -0.081 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} \qquad \begin{matrix} Y \in [0, 255] \\ C_b \in [0, 255] \\ C_r \in [0, 255] \end{matrix}$$

⇩



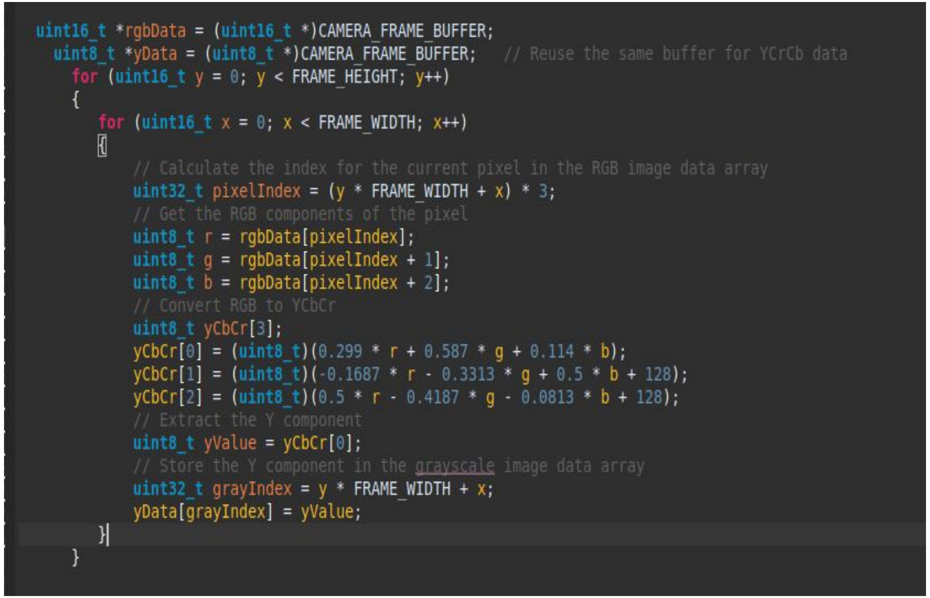
```
uint32_t pixelCount = FRAME_BUFFER_SIZE / 2;  // Assuming RGB565 pixel format
uint16_t *rgbData = (uint16_t *)CAMERA_FRAME_BUFFER;
uint16_t *ycrcbData = (uint16_t *)CAMERA_FRAME_BUFFER;  // Reuse the same buffer for YCrCb data



/* Perform RGB to YCrCb conversion */
for (uint32_t i = 0; i < pixelCount; ++i)
{
    uint16_t rgbPixel = rgbData[i];

    uint8_t r = (rgbPixel >> 11) & 0x1F;
    uint8_t g = (rgbPixel >> 5) & 0x3F;
    uint8_t b = rgbPixel & 0x1F;

    uint8_t y = (uint8_t)(0.299 * r + 0.587 * g + 0.114 * b);
    uint8_t cr = (uint8_t)(128 + 0.564 * (b - y));
    uint8_t cb = (uint8_t)(128 + 0.713 * (r - y));

    ycrcbData[i] = (y << 8) | (cr << 3) | (cb >> 3);
}
```



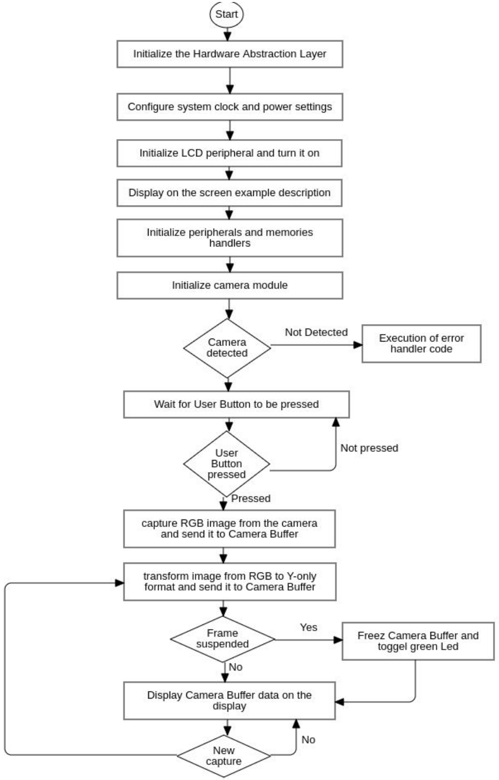
Start
Initialize the Hardware Abstraction Layer
Configure system clock and power settings
Initialize LCD peripheral and turn it on
Display on the screen example description
Initialize peripherals and memories handlers
Initialize camera module
Camera detected
Not Detected
Execution of error handler code
Wait for User Button to be pressed
User Button pressed
Not pressed
Pressed
capture RGB image from the camera and send it to Camera Buffer
transform image from RGB to YCrCb format and send it to Camera Buffer
Frame suspended
Yes
Freez Camera Buffer and toggel green Led
No
Display Camera Buffer data on the display
No
New capture

Y-Only Application

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$



```
uint16_t *rgbData = (uint16_t *)CAMERA_FRAME_BUFFER;
  uint8_t *yData = (uint8_t *)CAMERA_FRAME_BUFFER;   // Reuse the same buffer for YCrCb data
    for (uint16_t y = 0; y < FRAME_HEIGHT; y++)
    {
       for (uint16_t x = 0; x < FRAME_WIDTH; x++)
       {
           // Calculate the index for the current pixel in the RGB image data array
           uint32_t pixelIndex = (y * FRAME_WIDTH + x) * 3;
           // Get the RGB components of the pixel
           uint8_t r = rgbData[pixelIndex];
           uint8_t g = rgbData[pixelIndex + 1];
           uint8_t b = rgbData[pixelIndex + 2];
           // Convert RGB to YCbCr
           uint8_t yCbCr[3];
           yCbCr[0] = (uint8_t)(0.299 * r + 0.587 * g + 0.114 * b);
           yCbCr[1] = (uint8_t)(-0.1687 * r - 0.3313 * g + 0.5 * b + 128);
           yCbCr[2] = (uint8_t)(0.5 * r - 0.4187 * g - 0.0813 * b + 128);
           // Extract the Y component
           uint8_t yValue = yCbCr[0];
           // Store the Y component in the grayscale image data array
           uint32_t grayIndex = y * FRAME_WIDTH + x;
           yData[grayIndex] = yValue;
       }
    }
```


Start
Initialize the Hardware Abstraction Layer
Configure system clock and power settings
Initialize LCD peripheral and turn it on
Display on the screen example description
Initialize peripherals and memories handlers
Initialize camera module
Camera detected
Not Detected
Execution of error handler code
Wait for User Button to be pressed
User Button pressed
Not pressed
Pressed
capture RGB image from the camera and send it to Camera Buffer
transform image from RGB to Y-only format and send it to Camera Buffer
Frame suspended
Yes
Freez Camera Buffer and toggel green Led
No
Display Camera Buffer data on the display
New capture
No

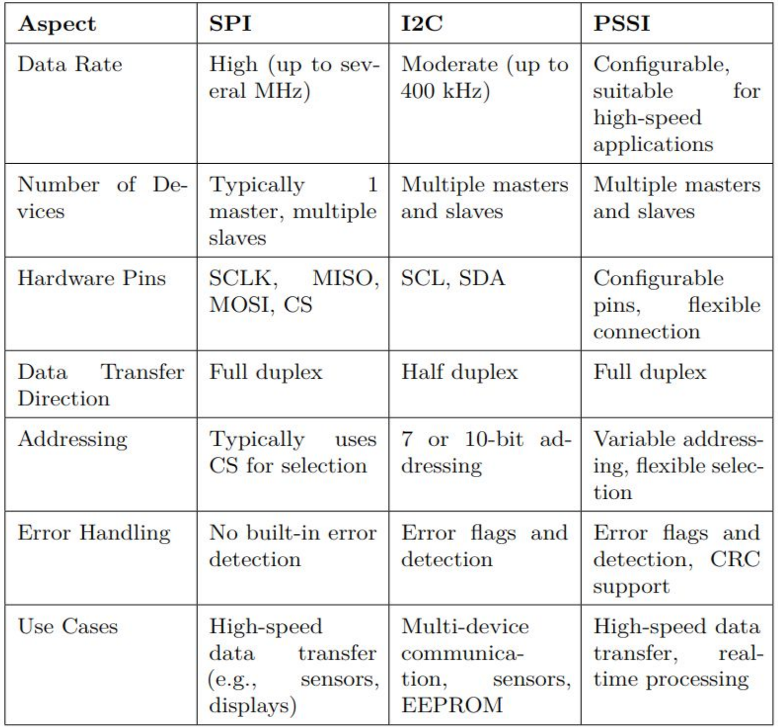
04

PSSI Use Case Implementation

Project Objectives

- Demonstrate the practical application of PSSI and GPDMA integration on the STM32U5
- Emphasize the importance of efficient master-slave communication
- Emphasize the goal of unlocking real-time data exchange capabilities for a diverse range of applications

PSSI vs SPI vs I2C



Aspect	SPI	I2C	PSSI
Data Rate	High (up to several MHz)	Moderate (up to 400 kHz)	Configurable, suitable for high-speed applications
Number of Devices	Typically 1 master, multiple slaves	Multiple masters and slaves	Multiple masters and slaves
Hardware Pins	SCLK, MISO, MOSI, CS	SCL, SDA	Configurable pins, flexible connection
Data Transfer Direction	Full duplex	Half duplex	Full duplex
Addressing	Typically uses CS for selection	7 or 10-bit addressing	Variable addressing, flexible selection
Error Handling	No built-in error detection	Error flags and detection	Error flags and detection, CRC support
Use Cases	High-speed data transfer (e.g., sensors, displays)	Multi-device communication, sensors, EEPROM	High-speed data transfer, real-time processing

- PSSI offer higher data transfer rates compared to SPI and I2C
- PSSI can transfer multiple bits of data simultaneously
- Synchronous communication can simplify timing considerations and make it easier to synchronize data between devices
- PSSI suitable for applications that require faster and more efficient data transfer

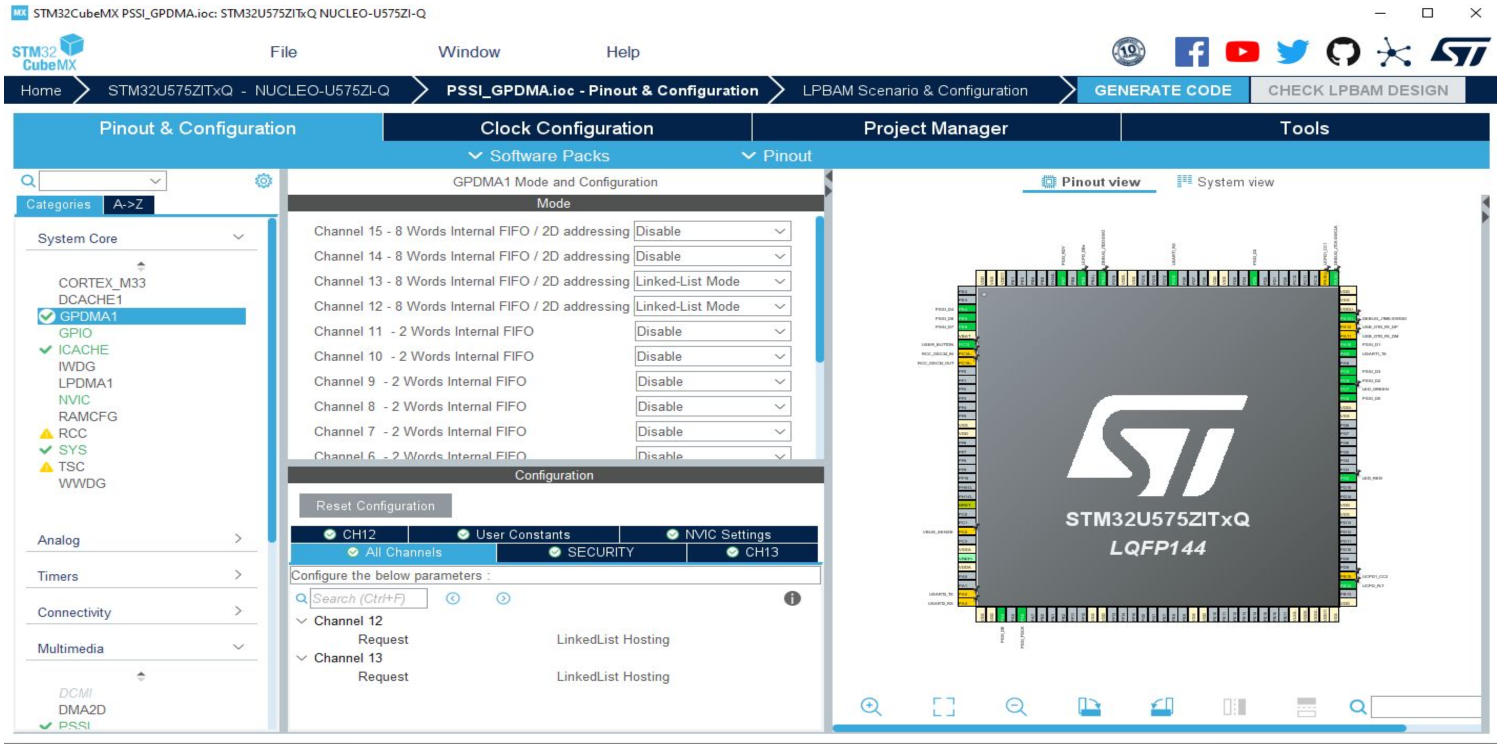
STM32U575ZI-Q-Nucleo

- Demonstrate the practical application of PSSI and GPDMA integration on the STM32U5

- Emphasize the importance of efficient master-slave communication

- Emphasize the goal of unlocking real-time data exchange capabilities for a diverse range of applications

GPDMA Configuration



GPDMA1 Mode and Configuration

Mode

Channel 15 - 8 Words Internal FIFO / 2D addressing	Disable
Channel 14 - 8 Words Internal FIFO / 2D addressing	Disable
Channel 13 - 8 Words Internal FIFO / 2D addressing	Linked-List Mode
Channel 12 - 8 Words Internal FIFO / 2D addressing	Linked-List Mode
Channel 11 - 2 Words Internal FIFO	Disable
Channel 10 - 2 Words Internal FIFO	Disable
Channel 9 - 2 Words Internal FIFO	Disable
Channel 8 - 2 Words Internal FIFO	Disable
Channel 7 - 2 Words Internal FIFO	Disable
Channel 6 - 2 Words Internal FIFO	Disable

Configuration

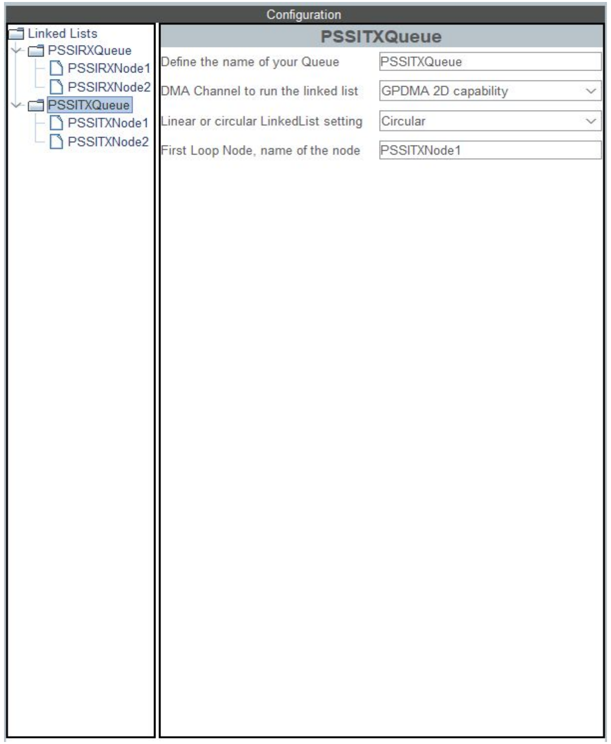
Reset Configuration

CH12 | User Constants | NVIC Settings
All Channels | SECURITY | CH13

Configure the below parameters :

Channel 12	
Request	LinkedList Hosting
Channel 13	
Request	LinkedList Hosting

Transmission Channel

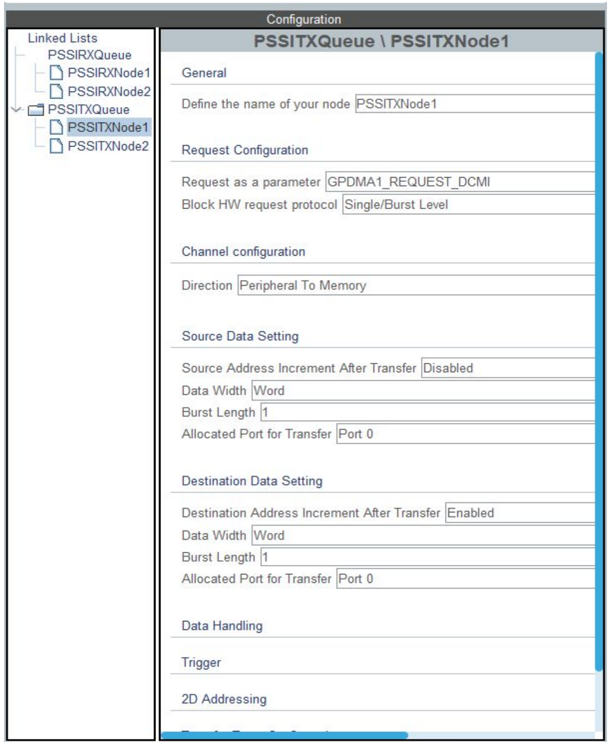


Configuration

Linked Lists
- PSSIRXQueue
  - PSSIRXNode1
  - PSSIRXNode2
- PSSITXQueue
  - PSSITXNode1
  - PSSITXNode2

PSSITXQueue

Define the name of your Queue: PSSITXQueue
DMA Channel to run the linked list: GPDMA 2D capability
Linear or circular LinkedList setting: Circular
First Loop Node, name of the node: PSSITXNode1



Configuration

Linked Lists
- PSSIRXQueue
  - PSSIRXNode1
  - PSSIRXNode2
- PSSITXQueue
  - PSSITXNode1
  - PSSITXNode2

PSSITXQueue \ PSSITXNode1

General
Define the name of your node: PSSITXNode1

Request Configuration
Request as a parameter: GPDMA1_REQUEST_DCMI
Block HW request protocol: Single/Burst Level

Channel configuration
Direction: Peripheral To Memory

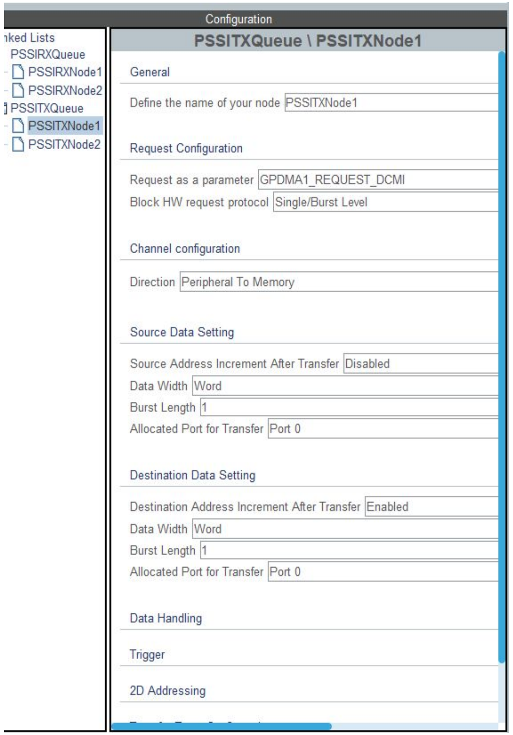
Source Data Setting
Source Address Increment After Transfer: Disabled
Data Width: Word
Burst Length: 1
Allocated Port for Transfer: Port 0

Destination Data Setting
Destination Address Increment After Transfer: Enabled
Data Width: Word
Burst Length: 1
Allocated Port for Transfer: Port 0

Data Handling

Trigger

2D Addressing



Configuration

nked Lists
- PSSIRXQueue
  - PSSIRXNode1
  - PSSIRXNode2
- PSSITXQueue
  - PSSITXNode1
  - PSSITXNode2

PSSITXQueue \ PSSITXNode1

General
Define the name of your node: PSSITXNode1

Request Configuration
Request as a parameter: GPDMA1_REQUEST_DCMI
Block HW request protocol: Single/Burst Level

Channel configuration
Direction: Peripheral To Memory

Source Data Setting
Source Address Increment After Transfer: Disabled
Data Width: Word
Burst Length: 1
Allocated Port for Transfer: Port 0

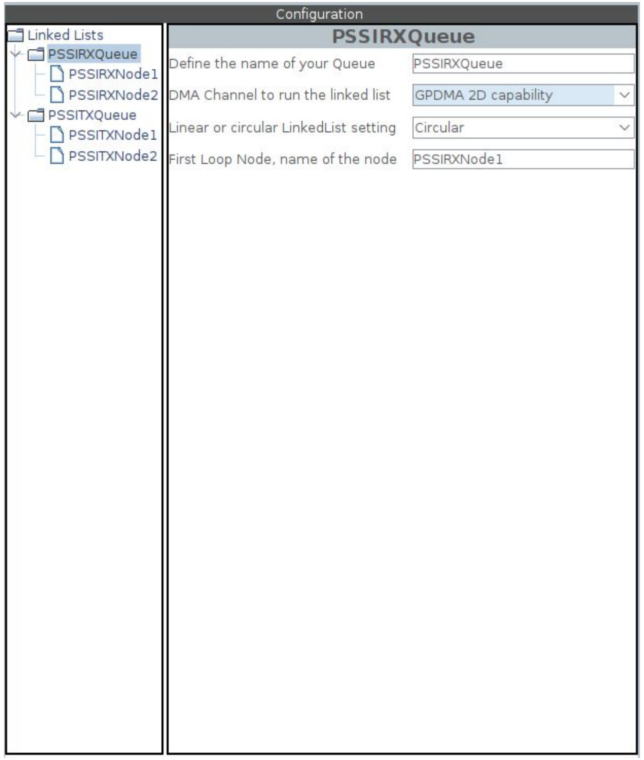
Destination Data Setting
Destination Address Increment After Transfer: Enabled
Data Width: Word
Burst Length: 1
Allocated Port for Transfer: Port 0

Data Handling

Trigger

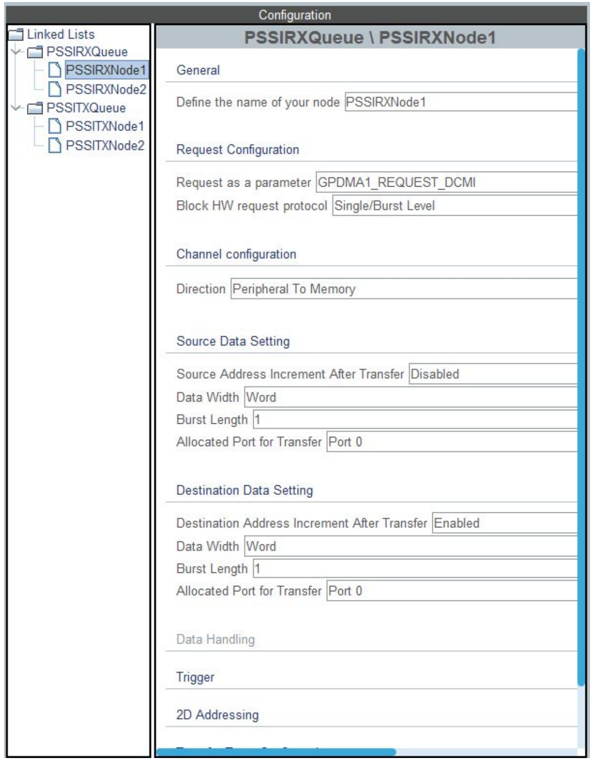
2D Addressing

Reception Channel



PSSIRXQueue

Define the name of your Queue	PSSIRXQueue
DMA Channel to run the linked list	GPDMA 2D capability
Linear or circular LinkedList setting	Circular
First Loop Node, name of the node	PSSIRXNode1



PSSIRXQueue \ PSSIRXNode1

General

Define the name of your node	PSSIRXNode1

Request Configuration

Request as a parameter	GPDMA1_REQUEST_DCMI
Block HW request protocol	Single/Burst Level

Channel configuration

Direction	Peripheral To Memory

Source Data Setting

Source Address Increment After Transfer	Disabled
Data Width	Word
Burst Length	1
Allocated Port for Transfer	Port 0

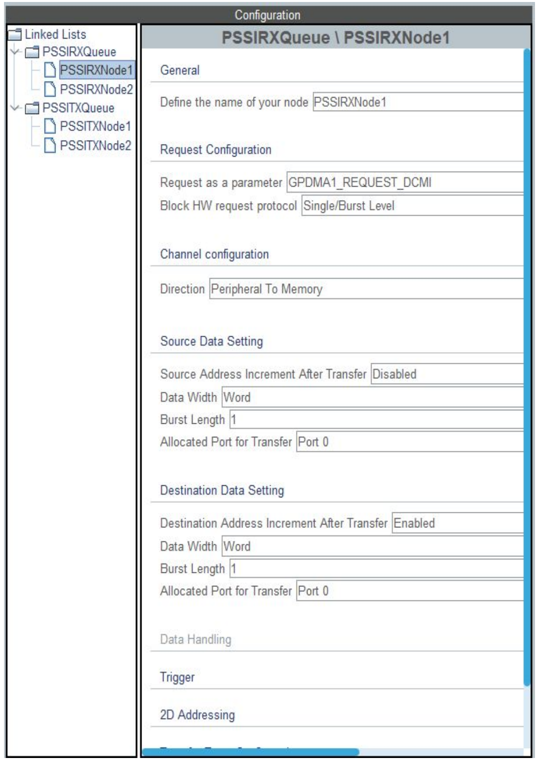
Destination Data Setting

Destination Address Increment After Transfer	Enabled
Data Width	Word
Burst Length	1
Allocated Port for Transfer	Port 0

Data Handling

Trigger

2D Addressing



PSSIRXQueue \ PSSIRXNode1

General

Define the name of your node	PSSIRXNode1

Request Configuration

Request as a parameter	GPDMA1_REQUEST_DCMI
Block HW request protocol	Single/Burst Level

Channel configuration

Direction	Peripheral To Memory

Source Data Setting

Source Address Increment After Transfer	Disabled
Data Width	Word
Burst Length	1
Allocated Port for Transfer	Port 0

Destination Data Setting

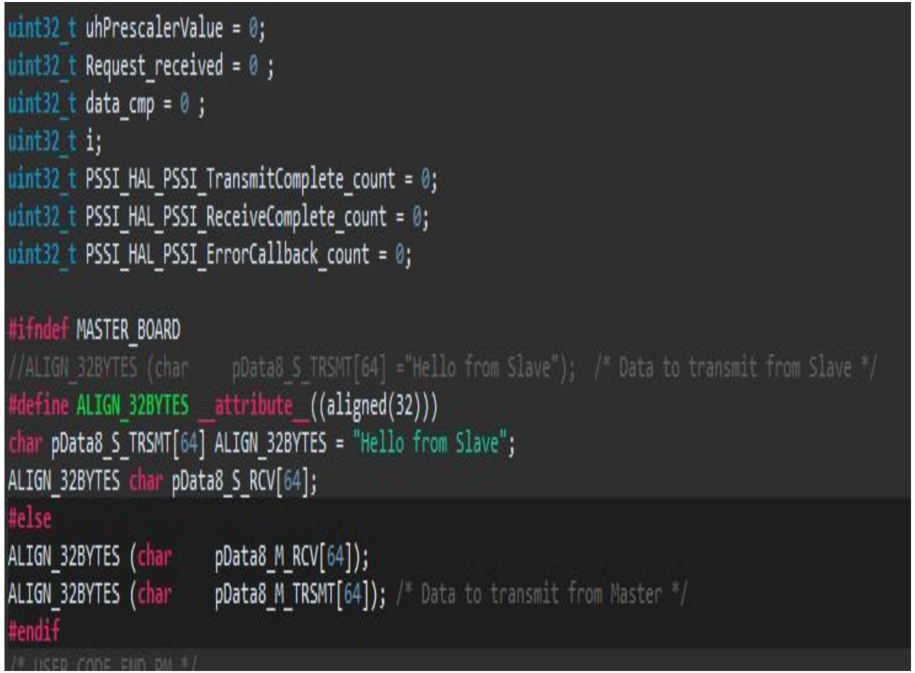
Destination Address Increment After Transfer	Enabled
Data Width	Word
Burst Length	1
Allocated Port for Transfer	Port 0

Data Handling

Trigger

2D Addressing

Global Variables & HAL functions

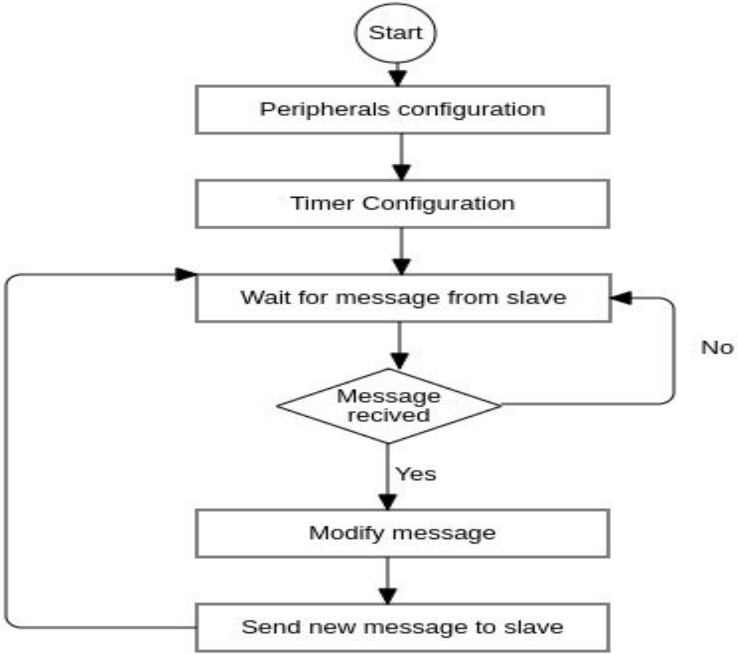


```
uint32_t uhPrescalerValue = 0;
uint32_t Request_received = 0 ;
uint32_t data_cmp = 0 ;
uint32_t i;
uint32_t PSSI_HAL_PSSI_TransmitComplete_count = 0;
uint32_t PSSI_HAL_PSSI_ReceiveComplete_count = 0;
uint32_t PSSI_HAL_PSSI_ErrorCallback_count = 0;

#ifndef MASTER_BOARD
//ALIGN_32BYTES (char     pData8_S_TRSMT[64] ="Hello from Slave");  /* Data to transmit from Slave */
#define ALIGN_32BYTES __attribute__((aligned(32)))
char pData8_S_TRSMT[64] ALIGN_32BYTES = "Hello from Slave";
ALIGN_32BYTES char pData8_S_RCV[64];
#else
ALIGN_32BYTES (char     pData8_M_RCV[64]);
ALIGN_32BYTES (char     pData8_M_TRSMT[64]); /* Data to transmit from Master */
#endif
/* USER CODE END PM */
```

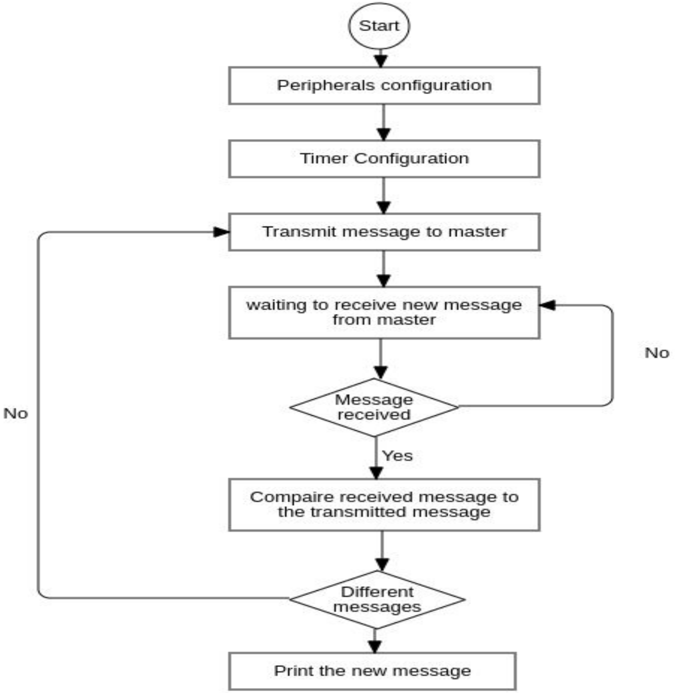
- **HAL_PSSI_Receive_DMA**(&hpssi, (uint32_t*)pData8_M_RCV , sizeof(pData8_M_RCV)/4)

- **HAL_PSSI_Transmit_DMA**(&hpssi, (uint32_t*)pData8_M_TRSMT, sizeof(pData8_M_TRSMT)/4)

Master Application


Start
Peripherals configuration
Timer Configuration
Wait for message from slave
No
Message recived
Yes
Modify message
Send new message to slave

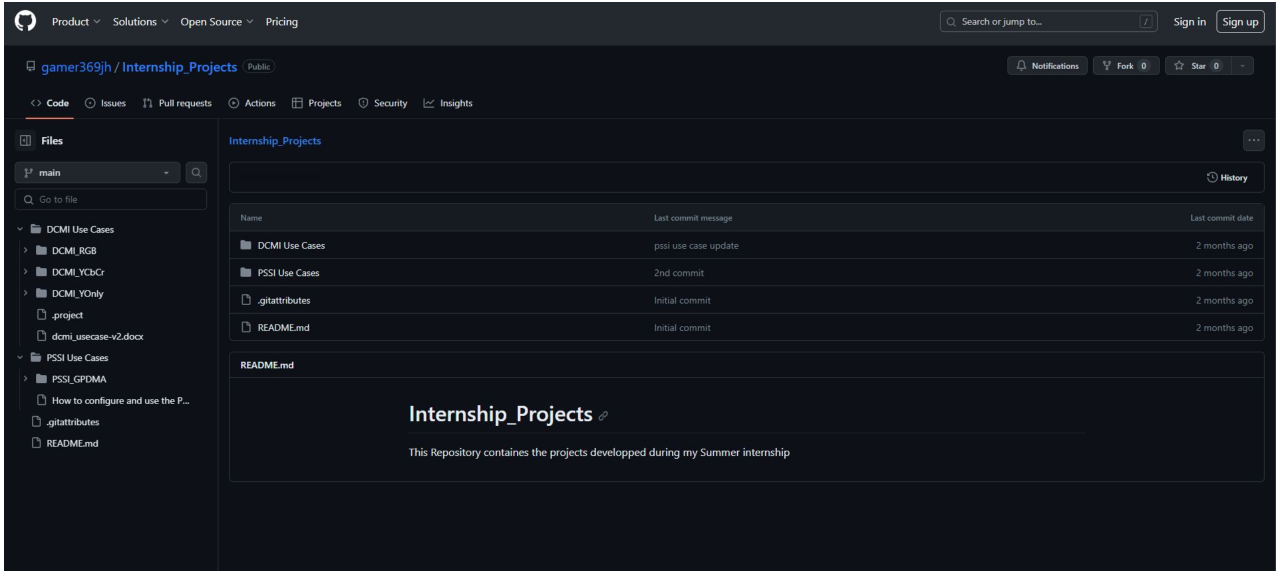
Slave Application


Start
Peripherals configuration
Timer Configuration
Transmit message to master
waiting to receive new message from master
No
Message received
No
Yes
Compaire received message to the transmitted message
Different messages
Print the new message

05

Project Deployment

Project Deployment



- DCMI Use Cases
  - DCMI_RGB
  - DCMI_YCbCr
  - DCMI_YOnly
  - .project
  - dcmi_usecase-v2.docx
- PSSI Use Cases
  - PSSI_GPDMA
  - How to configure and use the P...
- .gitattributes
- README.md

Internship_Projects

Name	Last commit message	Last commit date
DCMI Use Cases	pssi use case update	2 months ago
PSSI Use Cases	2nd commit	2 months ago
.gitattributes	Initial commit	2 months ago
README.md	Initial commit	2 months ago

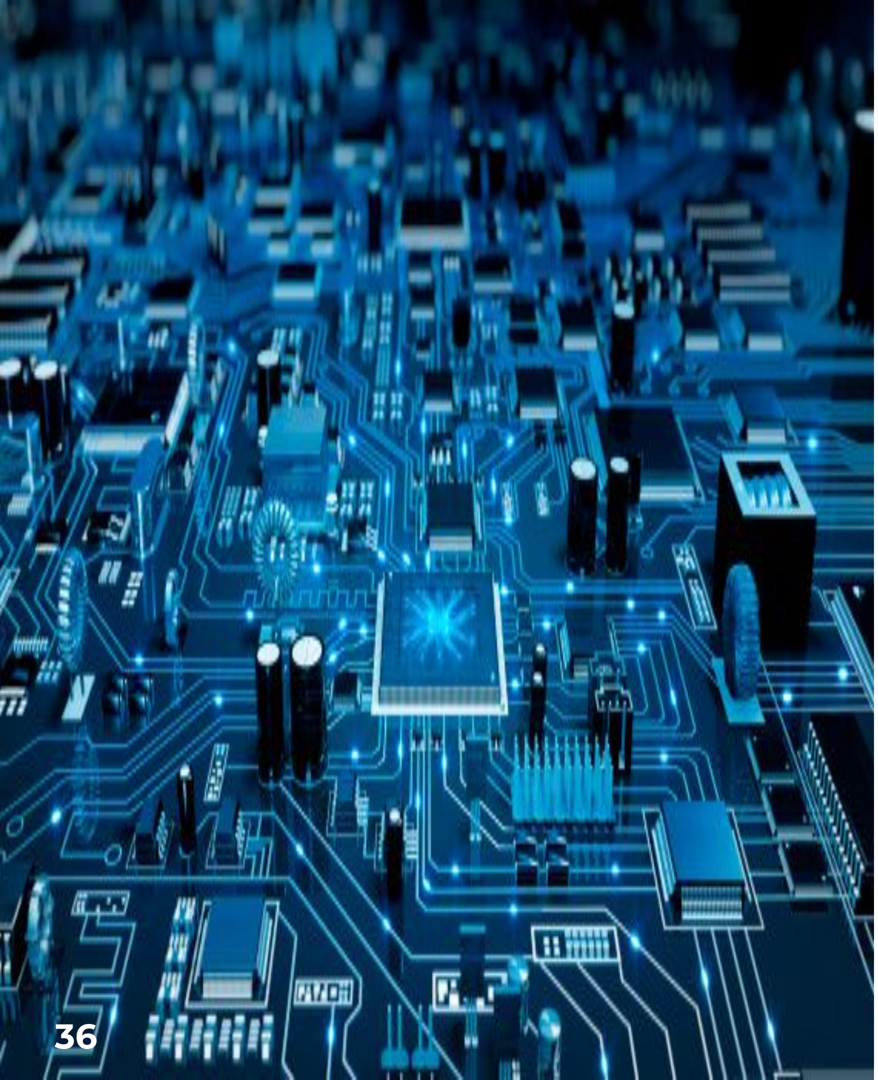
README.md

Internship_Projects

This Repository containes the projects developped during my Summer internship

06

Conclusion



CONCLUSION

- Demonstrate the impressive capabilities of the STM32U5 microcontroller
- DCMI open doors for computer vision applications like object recognition and gesture detection
- PSSI shows potential for various communication and data transfer tasks
- Showcase the importance of community engagement for further development

Thank you for your attention

Scan Me