DIRECT MEMORY ACCESS (DMA)

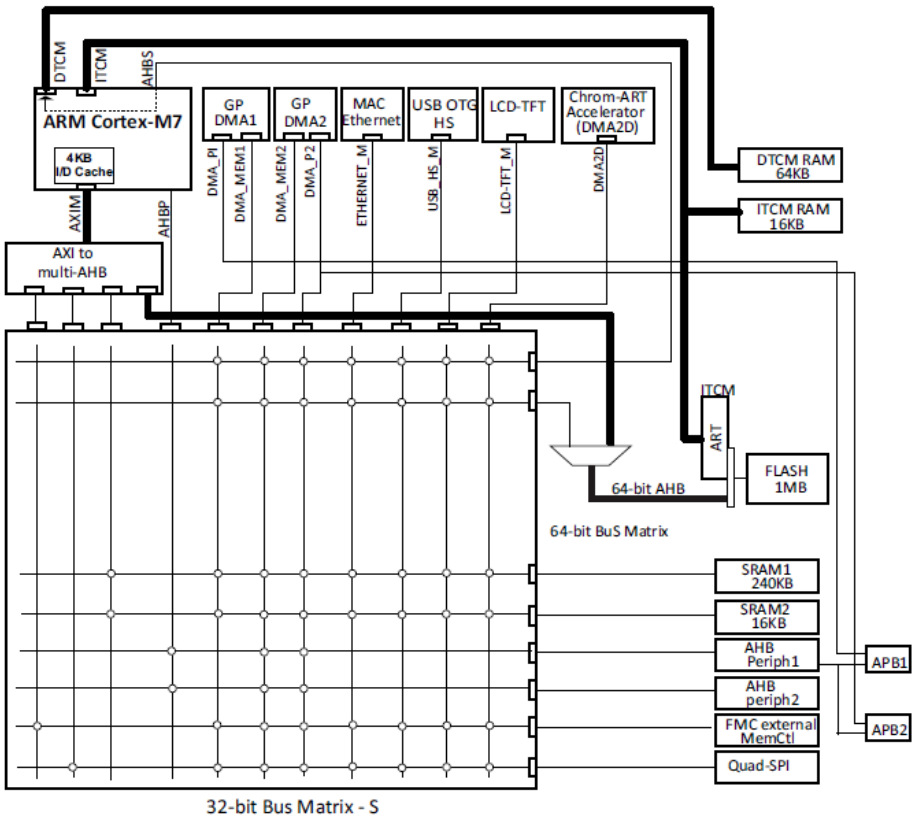
1 INTRODUCTION

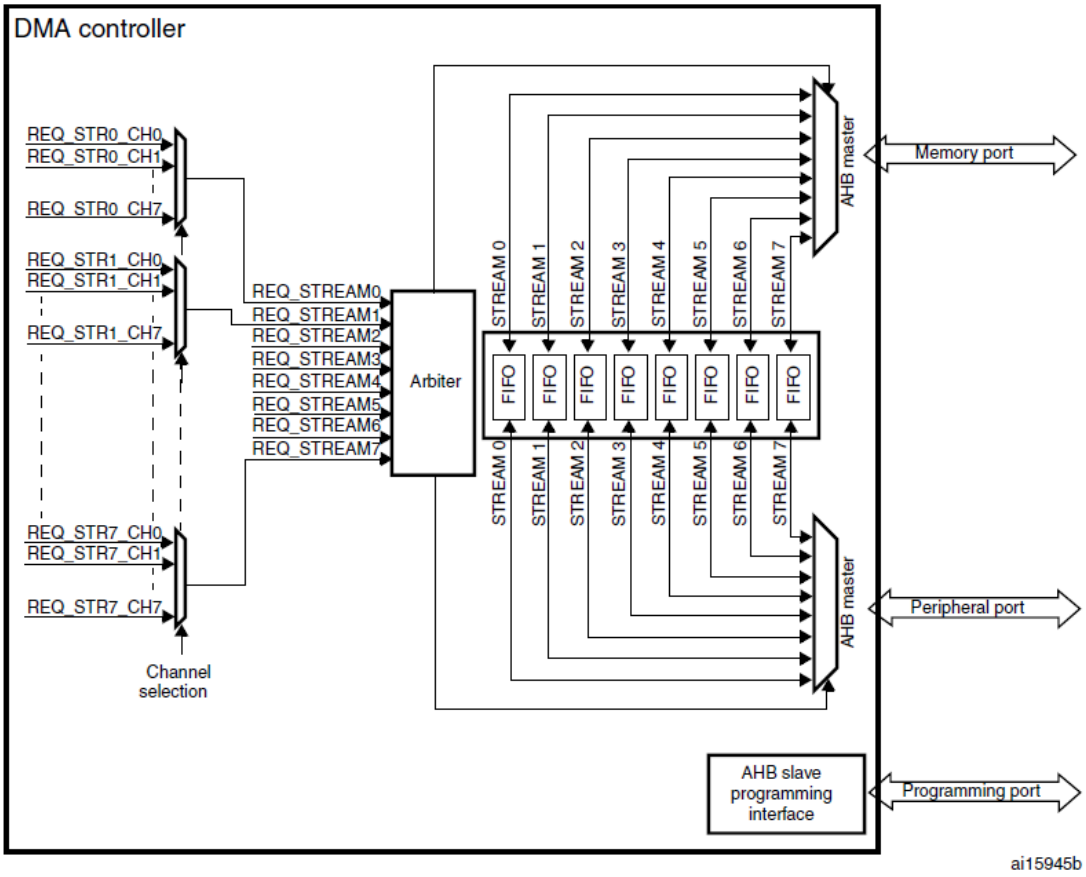
This laboratory aims to demonstrate the performance of Direct Memory Access (DMA). In this practical workshop, you will use a graphic display connected by SPI. This SPI display will be controlled with and without DMA. You will be able to analyze the difference in performance.

A DMA transfer allows data exchanges between the main memory and a peripheral (RAM to peripheral) without needing a processor except to initiate and conclude the transfer. The DMA controller takes care of all memory transfers.

The processor initiates the data transfers by setting up the DMA controller. From then on, the DMA controller takes care of the transfers alone, using a specific bus to perform the data transfers. This means the processor can perform other tasks in parallel.

The figures below show the block diagram of the DMA controller integrated into the STM32F746 processor. This processor offers two separate DMA channels.





DMA schematic

Since the DMA offers many possibilities, it can be quite complex to configure it correctly. To simplify this operation, we will use free software from STMicroelectronics called STM32CubeMX. This software will generate the initialization code automatically.

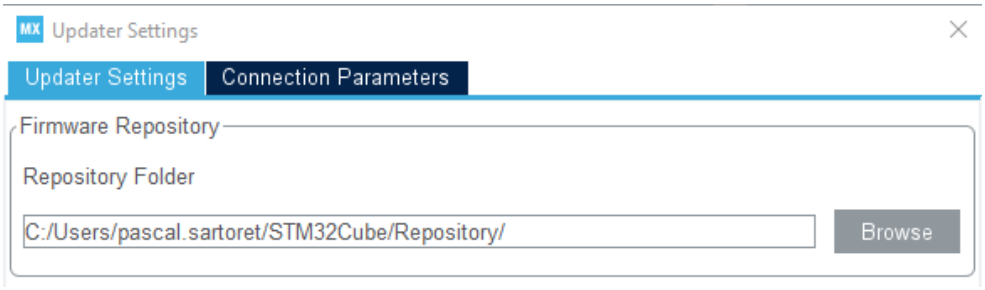
2 SPI AND DMA INITIALISATION

2.1 STM32CubeMX

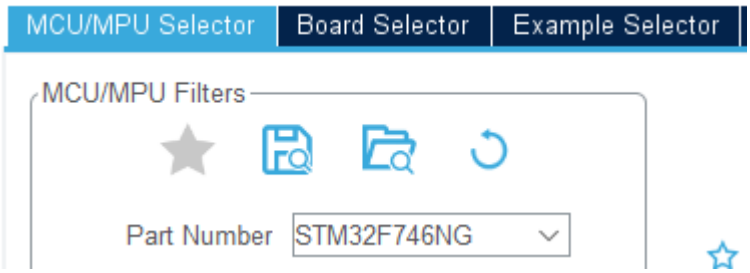
ST offers a tool to configure an STM32 processor with ease. This graphical interface software allows you to generate a part of the code or a full project automatically. This software helps you generate your application code and initialize all the peripherals.

Start by launching STM32CubeMX (not STM32CubeIDE).

In the “Updater Settings <alt-s>”, check the path where the repository is saved. If this is in the administrator profile, change it where you want (`c:\devel\` is a good choice).



Finally, start a new project from the MCU selector.

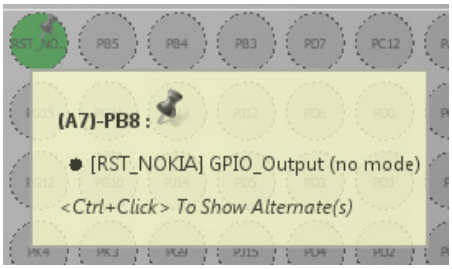


2.2 STM32CubeMX views

STM32CubeMX offers 3 main views to configure the microcontroller operation.

2.2.1 « Pinout » view

The "pinout" view lets you choose which processor pin is connected to which signal.

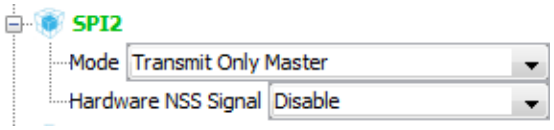


For example, we can see in the figure on the right shows that the PB8 pin is configured as an output and is named "RST_NOKIA".

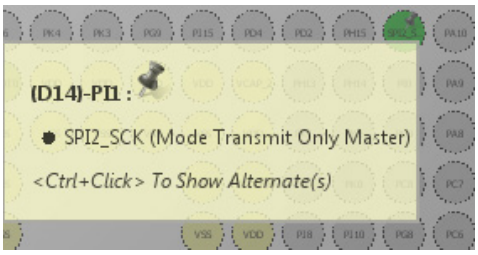
The table below describes the display signals connected to the microcontroller pins:

26 pin connector	Signal description	Signal name / User label	CPU Pin
2	SPI master out	MOSI	PB15 / MOSI SPI2
3	SPI clock	SCLK	PI1 / SCLK SPI2
4	Chip select	CS_NOKIA	PF6 (Output)
6	Reset	RST_NOKIA	PB8 (Output)
8	Data/Command	D_C_NOKIA	PB9 (Output)

The "pinout" view is also used to select the peripherals. Using CubeMX, you can configure the SPI2 module as follows:

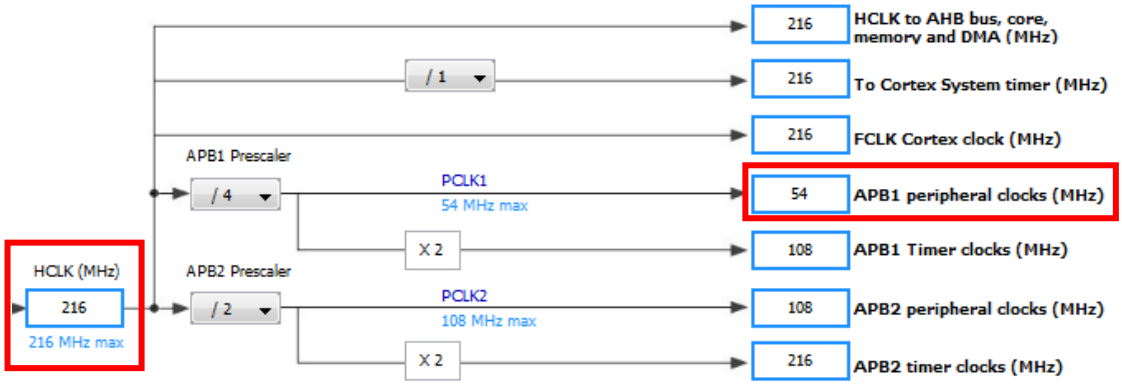


Once the module is activated, you can choose the pins on which the module is used.



2.2.2 « Clock configuration » view

The "Clock configuration" view lets you choose the frequency at which the microcontroller will operate. For this laboratory, two frequencies will be useful. The processor frequency (HCLK) and the APB1peripheral frequency for the peripheral clocks (the SPI2 peripheral in our case).



2.2.3 « Configuration » view

The "Configuration" view allows you to configure the settings for each device or peripheral.

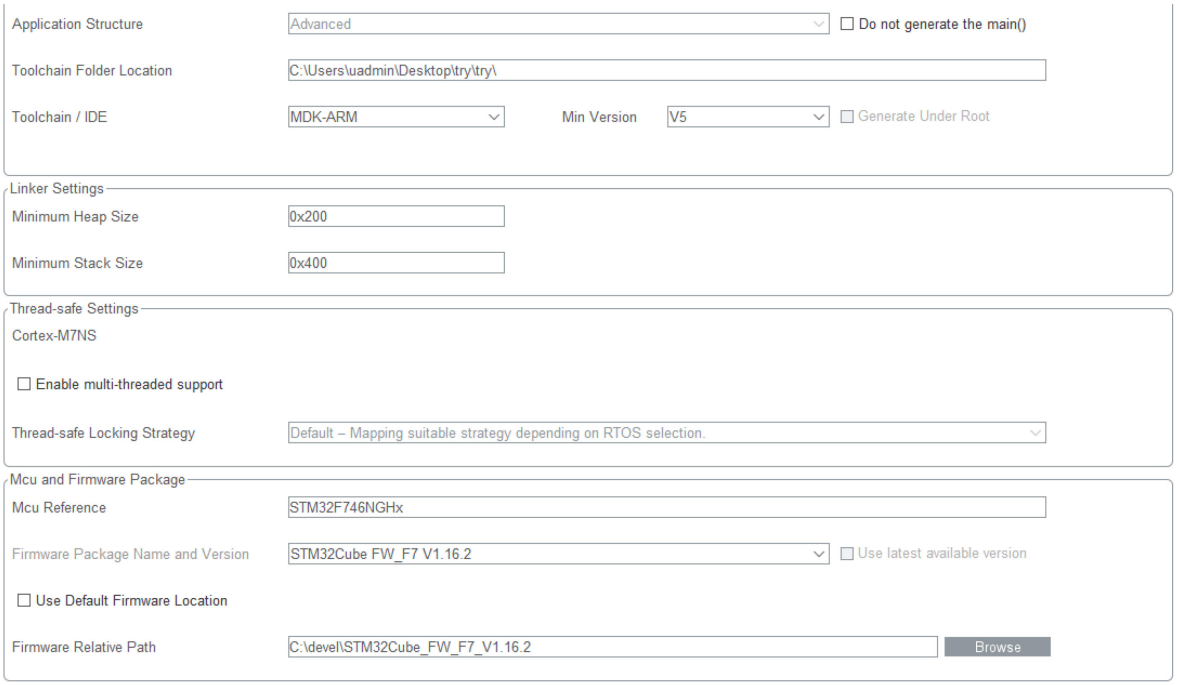
According to the display datasheet, the SPI module has to be configured as follows:

Max. SPI speed: 4MHz	CS active low
8 data bits, MSB first	Reset active low
CPOL = 0, CPHA = first edge	D_C : 0 = control D_C : 1 = data

In the same way, you can set the control signals of the LCD screen.

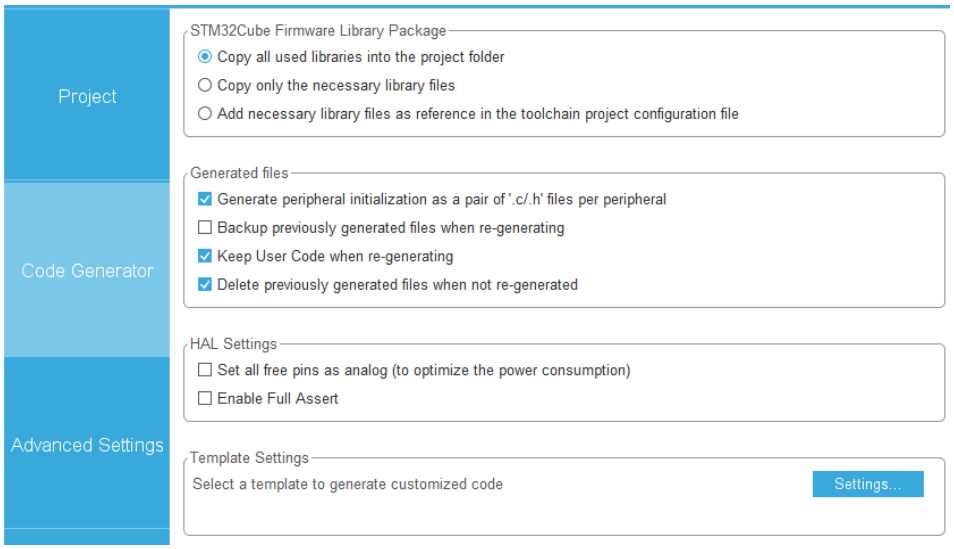
2.2.4 Project and code generation

Finally, you can generate a full project when all pins and peripherals are configured. Select a project location folder and choose MDK-ARM V5 as a toolchain. This will generate a Keil uVision project.



A uVision projet will be generated. Please use the ARM MDK version 5 and select the STM32Cube FW for CortexM7 version 1.16.2. Do not use the latest available version. The package is installed and available under the “C:\devel\” folder. Or it can be downloaded manually from https://github.com/STMicroelectronics/STM32CubeF7/tags.

In the code generator menu, choose “Generate peripheral initialization as a pair of files”.



Once the code is generated, open the generated project in uVision. Please look at the generated code and use it in your application.
Add the given files `nokia.c/.h` and `lines_functions.c` to your project.

3 PRACTICAL WORK #1

In this first task, we will use the SPI display without DMA.

The Nokia module is an LCD screen for which the `nokia.c` file offers useful functions for displaying graphics. The Nokia LCD is accessed through the SPI2 (Serial Peripheral Interface) bus. There are two distinct modes to send bytes to the LCD. The choice is made with the D/C pin (0 = control, 1 = data):

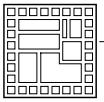
- The control mode allows you to give commands to the screen. During the initialization (`Nokia_Init()`), commands will be sent to the screen.
- The data mode allows you to send the bytes of the image (video buffer) to the screen.

Complete the code for the following functions in the `nokia.c` file:

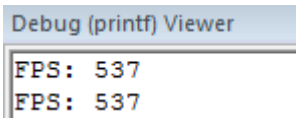
```
void NokiaControl(uint8_t controlByte);

void NokiaData(uint8_t dataByte);
```

Update your main program to call the `Nokia_Init()` function to initialize the LCD screen. Use the `Nokia_Update()` function to transmit the video buffer to the screen.



Using the `HAL_GetTick()` function and the Debug (printf) Viewer, measure and display the number of frames per second (FPS) displayed on the screen.



Please do the same speed tests with and without the cache enabled.

```
SCB_EnableICache(); // Enable Instruction cache
SCB_EnableDCache(); // Enable Data cache
```

4 PRACTICAL WORK #2

We will now do the same task with DMA activated.

Using STM32CubeMX, generate a new code/project with the DMA enabled. The `HAL_SPI_Transmit_DMA` function should be used to start a DMA transfer. Before starting a new transfer, ensure the previous one is finished using the `HAL_DMA_GetState function`.

Compare the differences in terms of performances obtained with and without DMA.Becnhmark your code using the table below and answer the following questions:

1. How could the display speed be further increased?
2. What is the theoretical maximum number of frames per second that can be displayed on the LCD monitor?
3. What must be done to reach this speed? What will be the consequences? Make measurements.

Benchmark configuration	***Measured speed [FPS]***
Without DMA, without cache	