Tutorial
Coding, Programming and Debugging STM32F767ZI

TECHNISCHE UNIVERSITÄT DARMSTADT

Aufgabe 1 Topic

This Tutorial is about a toolchain (IDE) for coding, programming and debugging of the STM32F767ZI.

Aufgabe 2 Requirements

In this tutorial two programs are needed. First of all `STM32CubeMX` (`http://www.st.com/en/development-tools/stm32cubemx.html`), a Code-Generator from ST. As an IDE `TrueStudio` (`https://atollic.com/truestudio/`) is used. Both programs are available on Linux and Windows and should be installed in default-configuration.

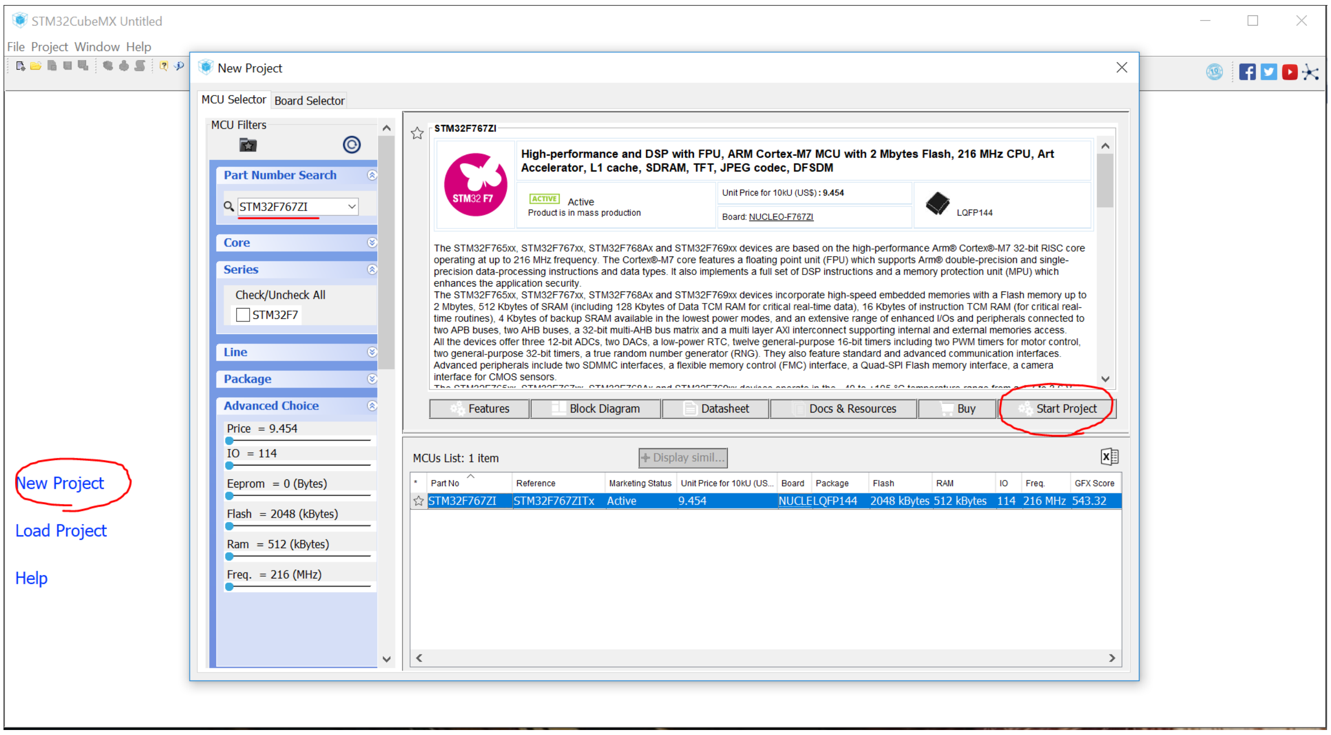
Aufgabe 3 Code-Generation with STM32CubeMX

For an intuitive way of configuring the periphery and the clocks ST provides a code-generator named `STM32CubeMX`.

1st Step: Start the program.

2nd Step: Click on `New Project`.

3rd Step: Select the device (STM32F767ZI) by searching for the part number and click `Start Project`.



Part Number Search: STM32F767ZI

Price = 9.454
IO = 114
Eeprom = 0 (Bytes)
Flash = 2048 (kBytes)
Ram = 512 (kBytes)
Freq. = 216 (MHz)

STM32F767ZI

High-performance and DSP with FPU, ARM Cortex-M7 MCU with 2 Mbytes Flash, 216 MHz CPU, Art Accelerator, L1 cache, SDRAM, TFT, JPEG codec, DFSDM

ACTIVE Active
Product is in mass production

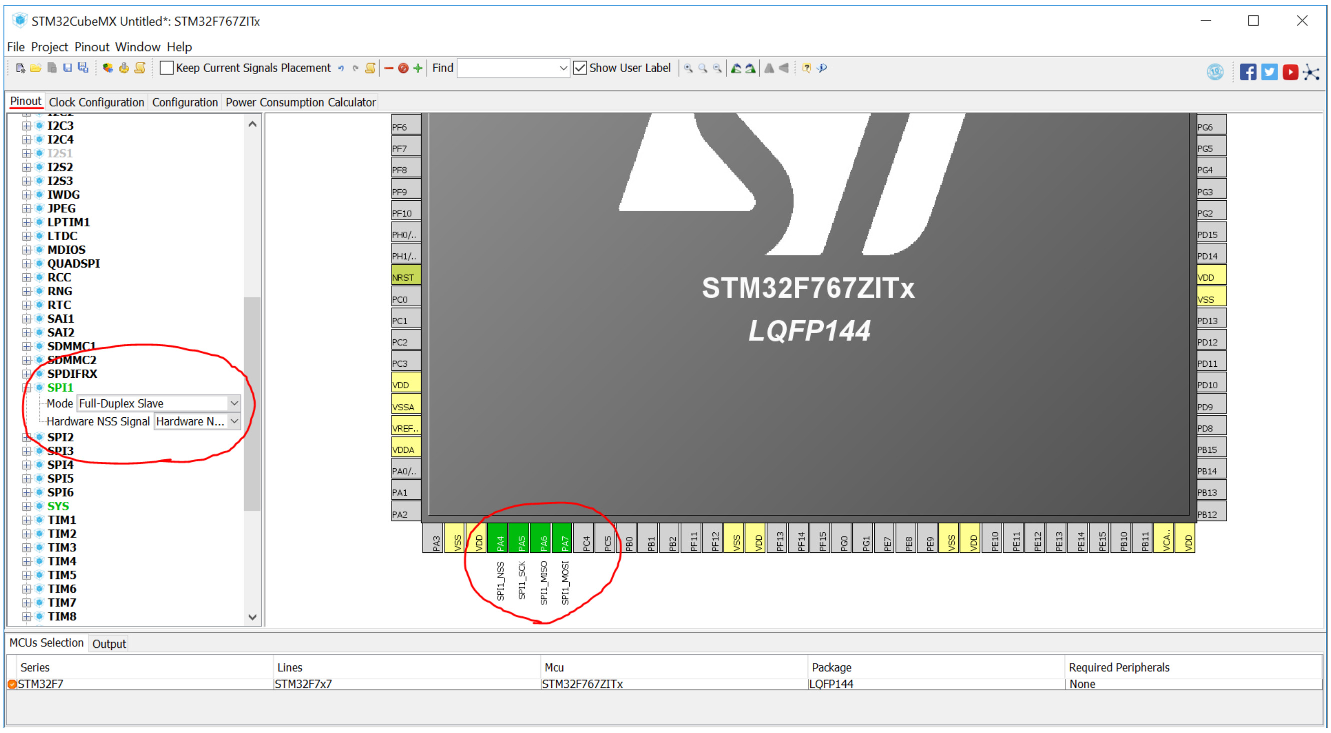
Unit Price for 10kU (US$) : 9.454
Board: NUCLEO-F767ZI
LQFP144

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 frequency. The Cortex®-M7 core features a floating point unit (FPU) which supports Arm® double-precision and single-precision data-processing instructions and data types. It also implements a full set of DSP instructions and a memory protection unit (MPU) which enhances the application security.
The STM32F765xx, STM32F767xx, STM32F768Ax and STM32F769xx devices incorporate high-speed embedded memories with a Flash memory up to 2 Mbytes, 512 Kbytes of SRAM (including 128 Kbytes of Data TCM RAM for critical real-time data), 16 Kbytes of instruction TCM RAM (for critical real-time routines), 4 Kbytes of backup SRAM available in the lowest power modes, and an extensive range of enhanced I/Os and peripherals connected to two APB buses, two AHB buses, a 32-bit multi-AHB bus matrix and a multi layer AXI interconnect supporting internal and external memories access.
All the devices offer three 12-bit ADCs, two DACs, a low-power RTC, twelve general-purpose 16-bit timers including two PWM timers for motor control, two general-purpose 32-bit timers, a true random number generator (RNG). They also feature standard and advanced communication interfaces.
Advanced peripherals include two SDMMC interfaces, a flexible memory control (FMC) interface, a Quad-SPI Flash memory interface, a camera interface for CMOS sensors.

MCUs List: 1 item

*	Part No	Reference	Marketing Status	Unit Price for 10kU (US...	Board	Package	Flash	RAM	IO	Freq.	GFX Score
☆	STM32F767ZI	STM32F767ZITx	Active	9.454	NUCLE	LQFP144	2048 kBytes	512 kBytes	114	216 MHz	543.32

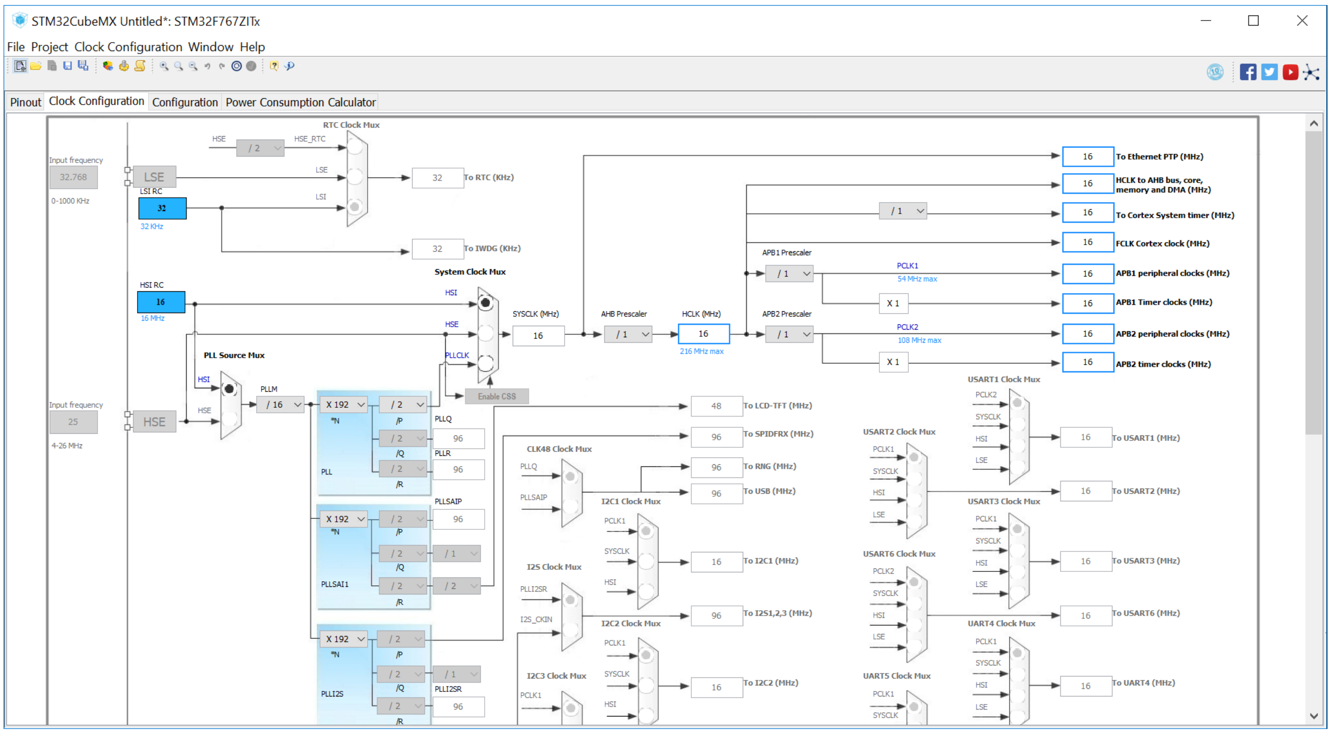
4th Step: The `Pinout`-Tab appears. Configure Pins by selecting an entry in the left column or clicking on the Pin directly.



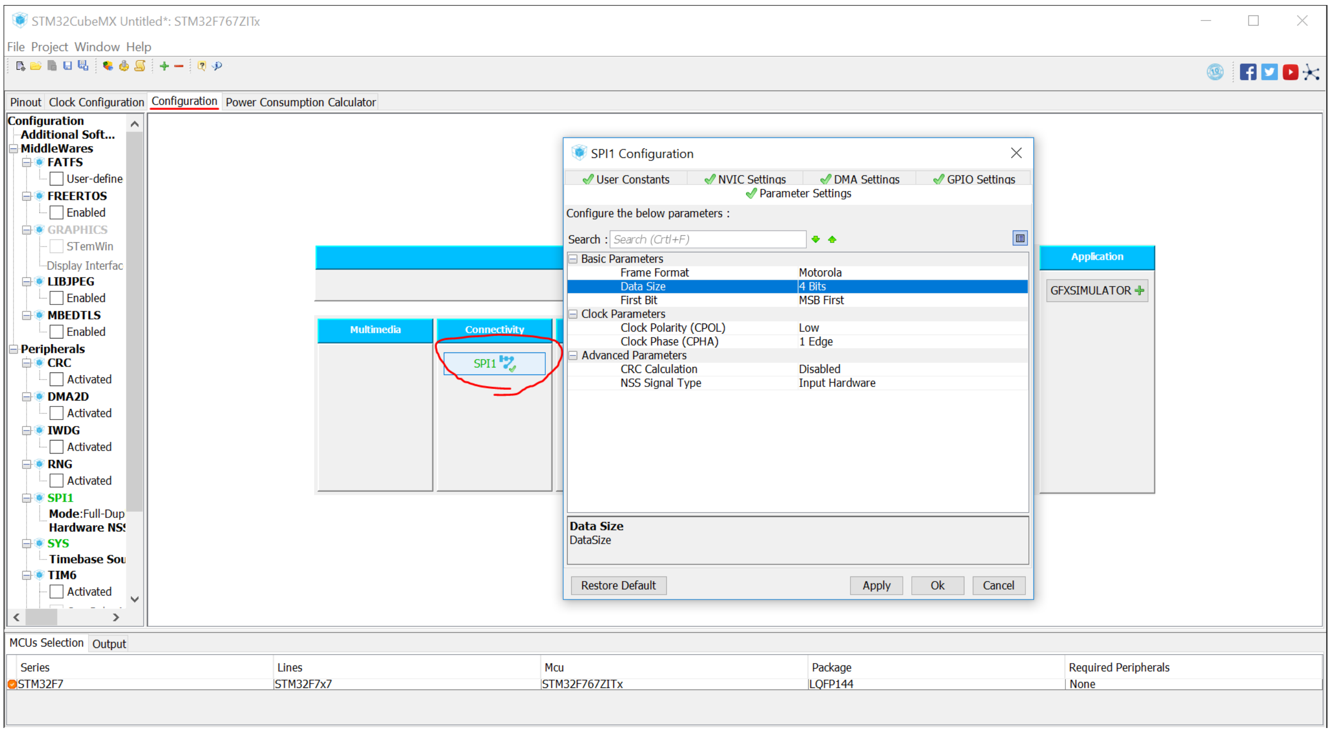
SPI1
- Mode: Full-Duplex Slave
- Hardware NSS Signal: Hardware N...

Series	Lines	Mcu	Package	Required Peripherals
STM32F7	STM32F7x7	STM32F767ZITx	LQFP144	None

5th Step: Adjust Clocks in the `Clock Configuration`-Tab.



6th Step: In the Configuration-Tab it is possible to configure periphery in more detail. For example setting the SPI-Slave's Frame Format.



SPI1 Configuration

User Constants | NVIC Settings | DMA Settings | GPIO Settings | Parameter Settings

Configure the below parameters :

Search : Search (Crtl+F)

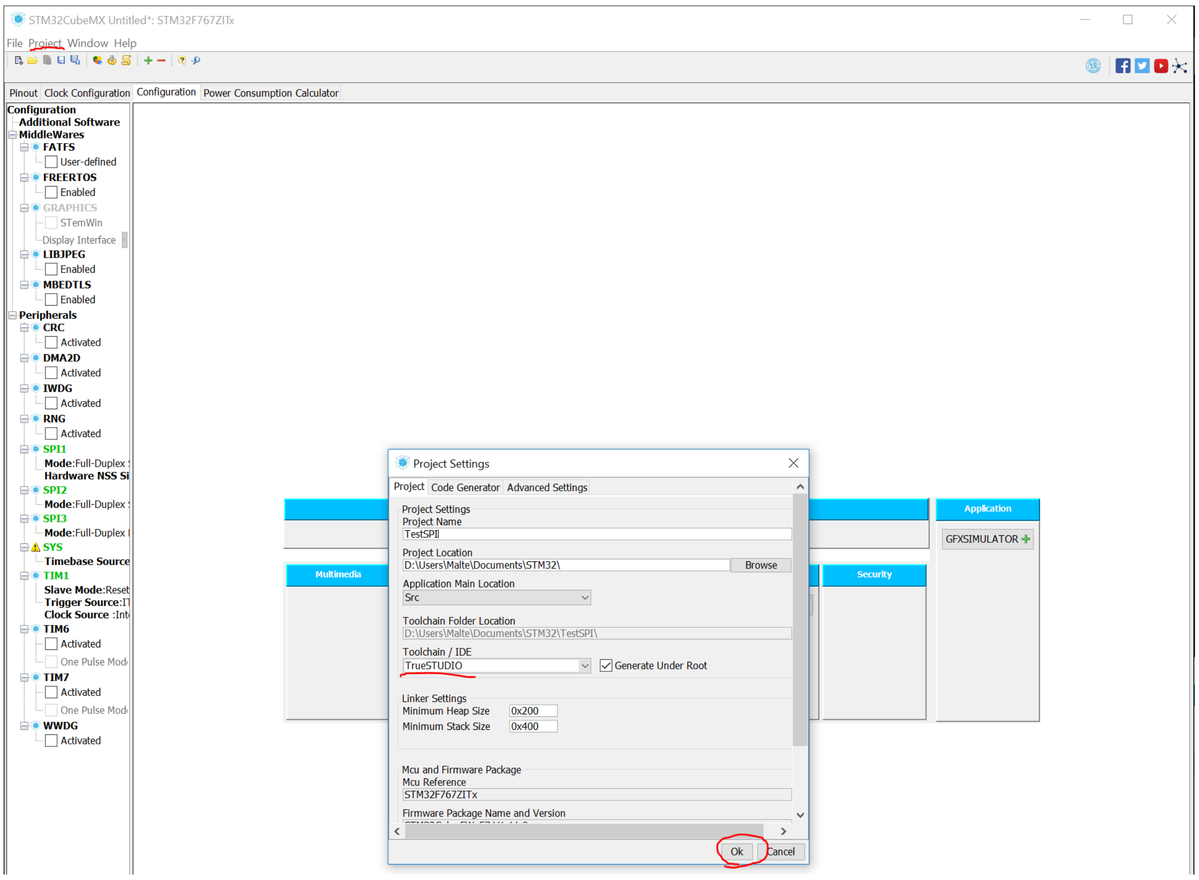
Basic Parameters	
Frame Format	Motorola
Data Size	4 Bits
First Bit	MSB First
Clock Parameters	
Clock Polarity (CPOL)	Low
Clock Phase (CPHA)	1 Edge
Advanced Parameters	
CRC Calculation	Disabled
NSS Signal Type	Input Hardware

Data Size
DataSize

Restore Default | Apply | Ok | Cancel

Series	Lines	Mcu	Package	Required Peripherals
STM32F7	STM32F7x7	STM32F767ZITx	LQFP144	None

7th Step: When the configuration is done, go to Project → Generate Code. Type in a Project Name and select as Toolchain/IDE the program TrueSTUDIO. After clicking Ok the code-generation begins. After the generation succeeded, click Open Project to start TrueSTUDIO.



Project Settings

Project | Code Generator | Advanced Settings

Project Settings
Project Name
TestSPI

Project Location
D:\Users\Malte\Documents\STM32\ Browse

Application Main Location
Src

Toolchain Folder Location
D:\Users\Malte\Documents\STM32\TestSPI\

Toolchain / IDE
TrueSTUDIO ✓ Generate Under Root

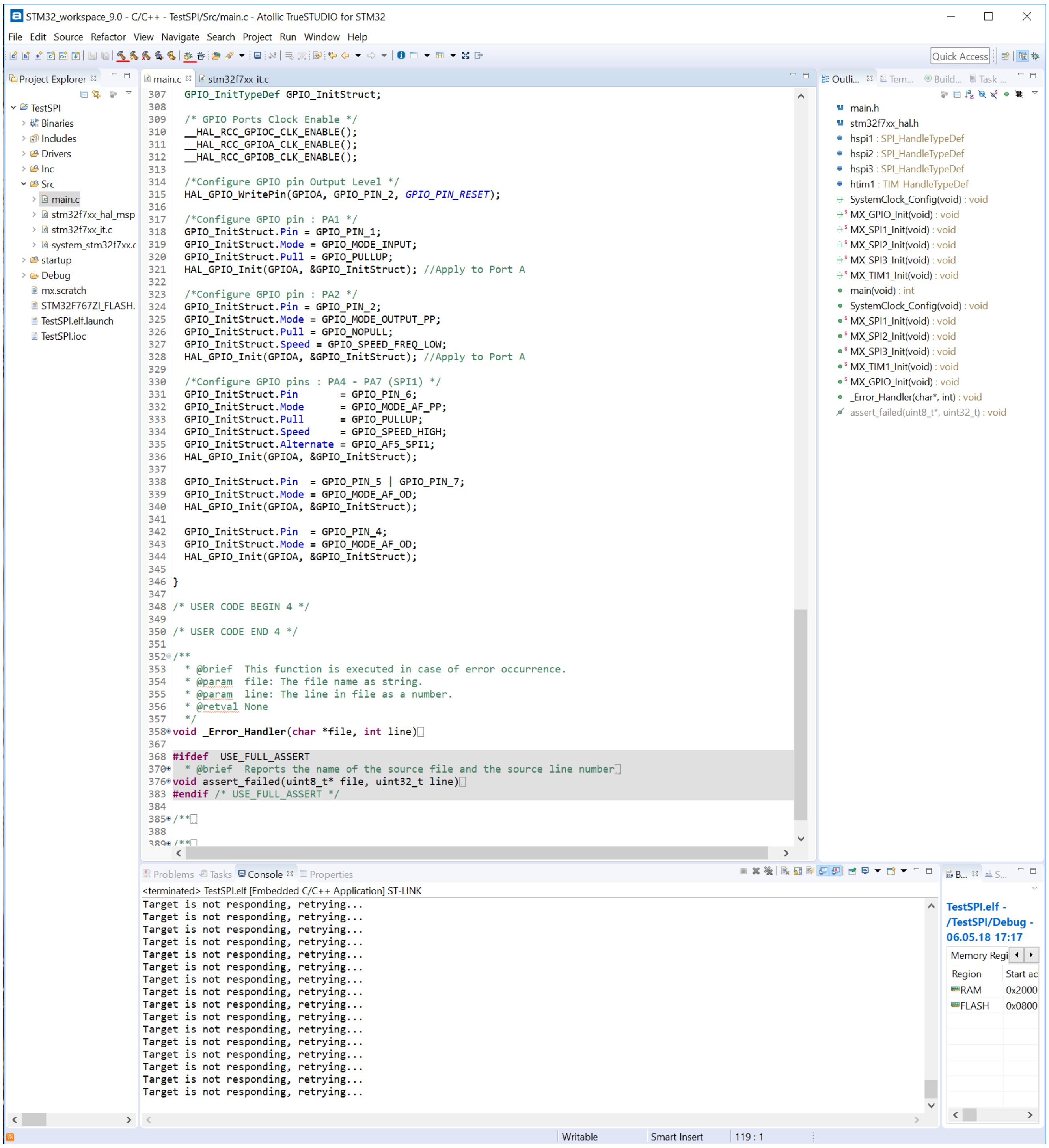
Linker Settings
Minimum Heap Size 0x200
Minimum Stack Size 0x400

Mcu and Firmware Package
Mcu Reference
STM32F767ZITx

Firmware Package Name and Version

Ok | Cancel

8th Step: In **TrueSTUDIO** the coding takes place similar to other IDEs. `STM32CubeMX` already created a folder structure including important c and h-files for using the periphery like the SPI-Slave.


STM32_workspace_9.0 - C/C++ - TestSPI/Src/main.c - Atollic TrueSTUDIO for STM32
File Edit Source Refactor View Navigate Search Project Run Window Help
Quick Access
Project Explorer
TestSPI
Binaries
Includes
Drivers
Inc
Src
main.c
stm32f7xx_hal_msp.
stm32f7xx_it.c
system_stm32f7xx.c
startup
Debug
mx.scratch
STM32F767ZI_FLASH.l
TestSPI.elf.launch
TestSPI.ioc
main.c
stm32f7xx_it.c
Outli...
Tem...
Build...
Task ...
main.h
stm32f7xx_hal.h
hspi1 : SPI_HandleTypeDef
hspi2 : SPI_HandleTypeDef
hspi3 : SPI_HandleTypeDef
htim1 : TIM_HandleTypeDef
SystemClock_Config(void) : void
MX_GPIO_Init(void) : void
MX_SPI1_Init(void) : void
MX_SPI2_Init(void) : void
MX_SPI3_Init(void) : void
MX_TIM1_Init(void) : void
main(void) : int
SystemClock_Config(void) : void
MX_SPI1_Init(void) : void
MX_SPI2_Init(void) : void
MX_SPI3_Init(void) : void
MX_TIM1_Init(void) : void
MX_GPIO_Init(void) : void
_Error_Handler(char*, int) : void
assert_failed(uint8_t*, uint32_t) : void
Problems
Tasks
Console
Properties
B...
S...
TestSPI.elf - /TestSPI/Debug - 06.05.18 17:17
Memory Regi
Writable
Smart Insert
119 : 1

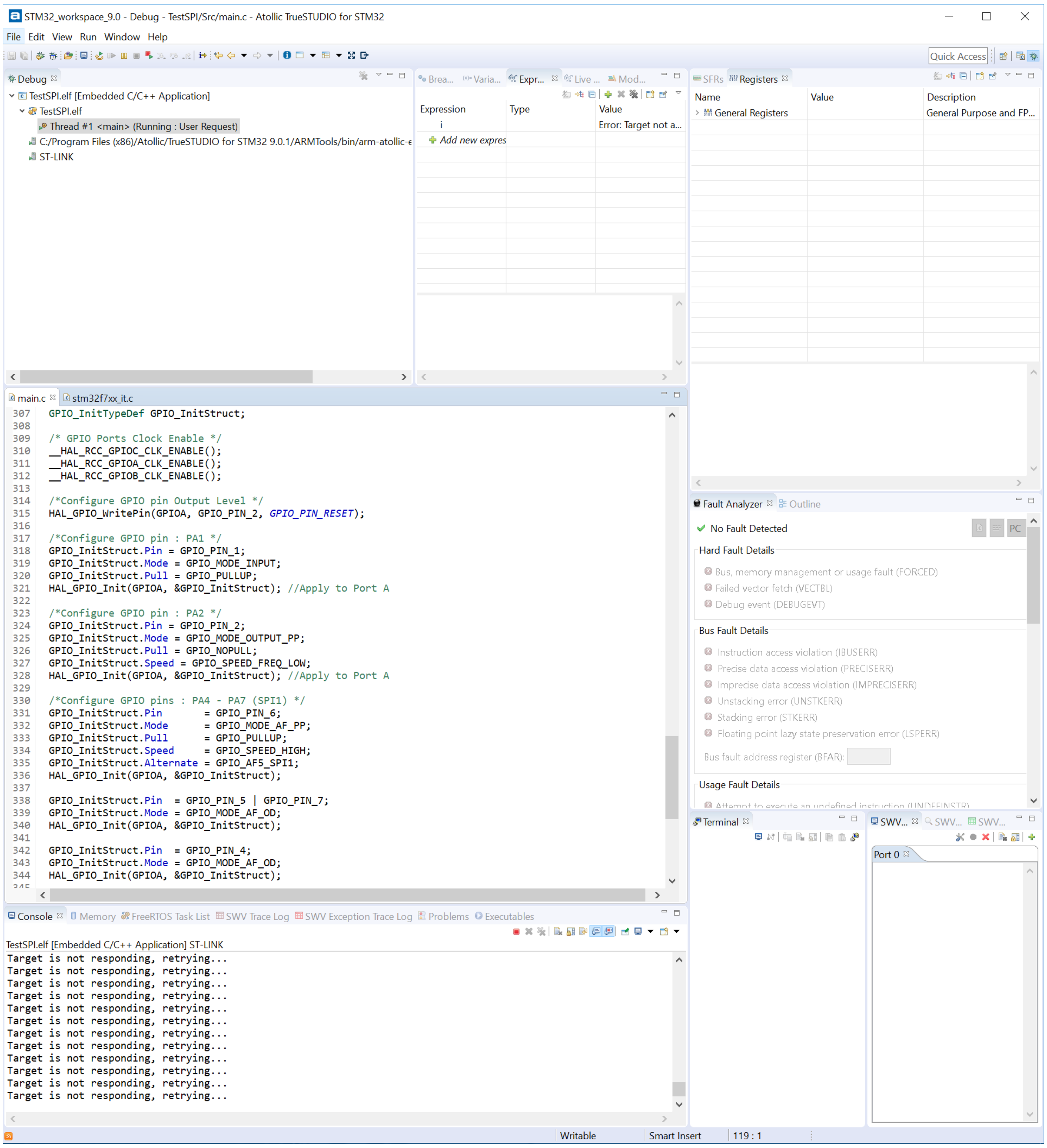
```
  GPIO_InitTypeDef GPIO_InitStruct;

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2, GPIO_PIN_RESET);

  /*Configure GPIO pin : PA1 */
  GPIO_InitStruct.Pin = GPIO_PIN_1;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_PULLUP;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct); //Apply to Port A

  /*Configure GPIO pin : PA2 */
  GPIO_InitStruct.Pin = GPIO_PIN_2;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct); //Apply to Port A

  /*Configure GPIO pins : PA4 - PA7 (SPI1) */
  GPIO_InitStruct.Pin       = GPIO_PIN_6;
  GPIO_InitStruct.Mode      = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Pull      = GPIO_PULLUP;
  GPIO_InitStruct.Speed     = GPIO_SPEED_HIGH;
  GPIO_InitStruct.Alternate = GPIO_AF5_SPI1;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  GPIO_InitStruct.Pin  = GPIO_PIN_5 | GPIO_PIN_7;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_OD;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  GPIO_InitStruct.Pin  = GPIO_PIN_4;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_OD;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @param  file: The file name as string.
  * @param  line: The line in file as a number.
  * @retval None
  */
void _Error_Handler(char *file, int line)

#ifdef  USE_FULL_ASSERT
  * @brief  Reports the name of the source file and the source line number
void assert_failed(uint8_t* file, uint32_t line)
#endif /* USE_FULL_ASSERT */

/**

/**
```

```
<terminated> TestSPI.elf [Embedded C/C++ Application] ST-LINK
Target is not responding, retrying...
Target is not responding, retrying...
Target is not responding, retrying...
Target is not responding, retrying...
Target is not responding, retrying...
Target is not responding, retrying...
Target is not responding, retrying...
Target is not responding, retrying...
Target is not responding, retrying...
Target is not responding, retrying...
Target is not responding, retrying...
Target is not responding, retrying...
Target is not responding, retrying...
Target is not responding, retrying...
Target is not responding, retrying...
Target is not responding, retrying...
```

Region	Start ac
RAM	0x2000
FLASH	0x0800

9th Step: By clicking the `hammer symbol` the build starts and (eventually) errors occur.

10th Step: Connect the Nucleo-Board with the PC by an USB-cable using the micro-USB-port located on the topside of the board (USB PWR).
11th Step: By clicking the `bug symbol` the program-transfer to the Nucleo starts and the debug-window occurs.



STM32_workspace_9.0 - Debug - TestSPI/Src/main.c - Atollic TrueSTUDIO for STM32

```
TestSPI.elf [Embedded C/C++ Application]
  TestSPI.elf
    Thread #1 <main> (Running : User Request)
  C:/Program Files (x86)/Atollic/TrueSTUDIO for STM32 9.0.1/ARMTools/bin/arm-atollic-e
  ST-LINK
```

Expression	Type	Value
i		Error: Target not a...
Add new expres		

Name	Value	Description
General Registers		General Purpose and FP...

```c
    GPIO_InitTypeDef GPIO_InitStruct;

    /* GPIO Ports Clock Enable */
    __HAL_RCC_GPIOC_CLK_ENABLE();
    __HAL_RCC_GPIOA_CLK_ENABLE();
    __HAL_RCC_GPIOB_CLK_ENABLE();

    /*Configure GPIO pin Output Level */
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2, GPIO_PIN_RESET);

    /*Configure GPIO pin : PA1 */
    GPIO_InitStruct.Pin = GPIO_PIN_1;
    GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
    GPIO_InitStruct.Pull = GPIO_PULLUP;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct); //Apply to Port A

    /*Configure GPIO pin : PA2 */
    GPIO_InitStruct.Pin = GPIO_PIN_2;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct); //Apply to Port A

    /*Configure GPIO pins : PA4 - PA7 (SPI1) */
    GPIO_InitStruct.Pin       = GPIO_PIN_6;
    GPIO_InitStruct.Mode      = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Pull      = GPIO_PULLUP;
    GPIO_InitStruct.Speed     = GPIO_SPEED_HIGH;
    GPIO_InitStruct.Alternate = GPIO_AF5_SPI1;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

    GPIO_InitStruct.Pin  = GPIO_PIN_5 | GPIO_PIN_7;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_OD;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

    GPIO_InitStruct.Pin  = GPIO_PIN_4;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_OD;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
```

No Fault Detected

Hard Fault Details
- Bus, memory management or usage fault (FORCED)
- Failed vector fetch (VECTBL)
- Debug event (DEBUGEVT)

Bus Fault Details
- Instruction access violation (IBUSERR)
- Precise data access violation (PRECISERR)
- Imprecise data access violation (IMPRECISERR)
- Unstacking error (UNSTKERR)
- Stacking error (STKERR)
- Floating point lazy state preservation error (LSPERR)

Bus fault address register (BFAR):

Usage Fault Details

```
TestSPI.elf [Embedded C/C++ Application] ST-LINK
Target is not responding, retrying...
Target is not responding, retrying...
Target is not responding, retrying...
Target is not responding, retrying...
Target is not responding, retrying...
Target is not responding, retrying...
Target is not responding, retrying...
Target is not responding, retrying...
Target is not responding, retrying...
Target is not responding, retrying...
Target is not responding, retrying...
Target is not responding, retrying...
```

File Edit View Run Window Help
Quick Access
Debug
Brea... Varia... Expr... Live ... Mod...
SFRs Registers
main.c stm32f7xx_it.c
Fault Analyzer Outline
Terminal
SWV... SWV... SWV...
Port 0
Console Memory FreeRTOS Task List SWV Trace Log SWV Exception Trace Log Problems Executables
Writable Smart Insert 119 : 1