Learning Report – Embedded C

L&T Technology Services

Document History

Ver. Rel. No.	Release Date	Prepared. By	Reviewed By	Approved By	Remarks/Revision Details
	09/03/2021	Aryan Verma			

Contents

Activity 1 – COMPILATION APROACH

This process of the sample program for ARM Cortex Mx processor-based boards.
Following are the compilation stages of a C program:

1. Preprocessor stage
2. Processor stage
3. Compilation stage
4. Assembly stage
5. Linking stage

1.1- MAKE FILE

Below is the make file for the sample program:

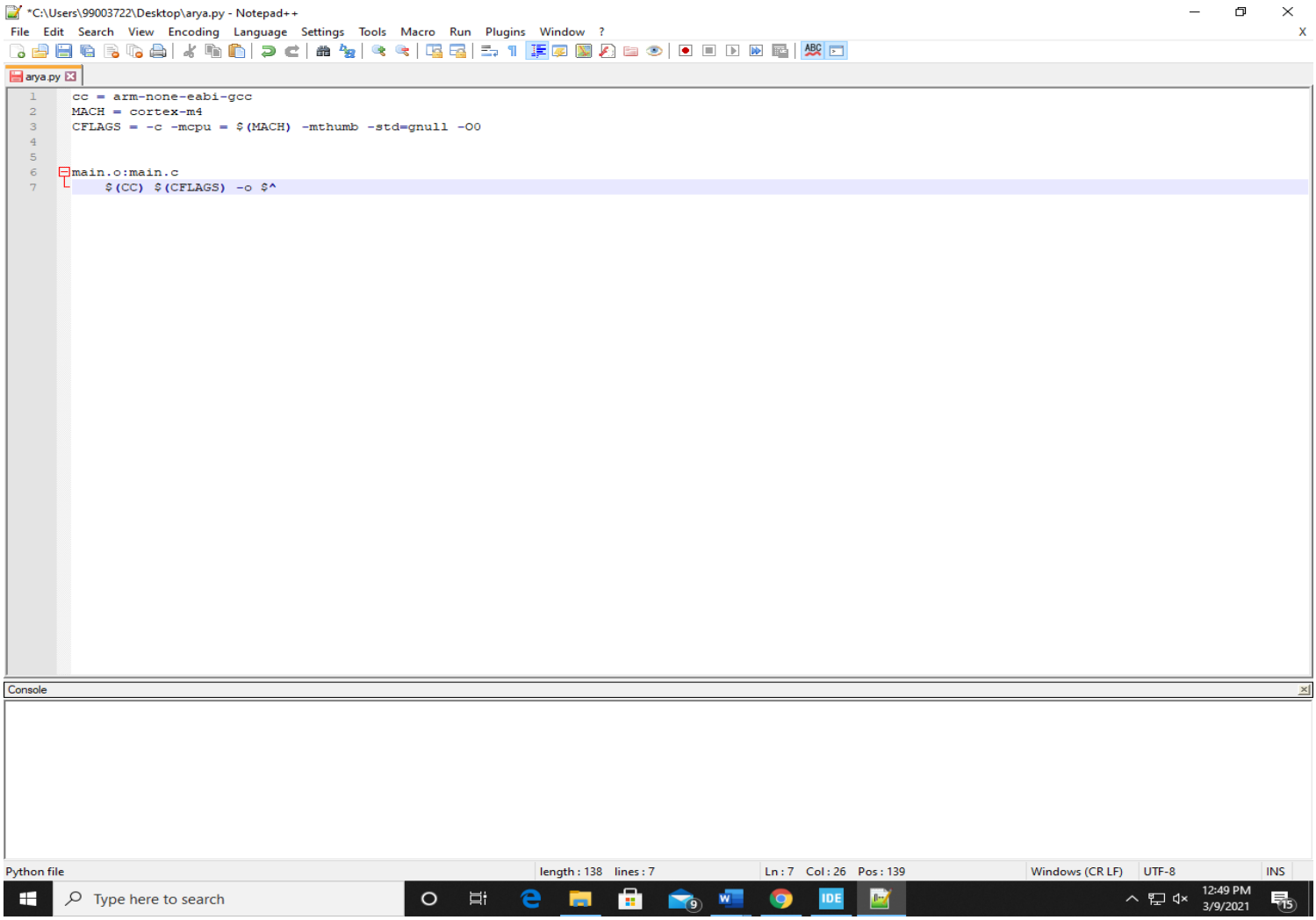


Fig 1.1.1 make file

The command to run this make file in the command prompt is:

```
Command Prompt

C:\Users\nieki\OneDrive\ofc\courses\my_workspace>make
make: `main.o' is up to date.

C:\Users\nieki\OneDrive\ofc\courses\my_workspace>make
arm-none-eabi-gcc -c -mcpu=cortex-m4 -mthumb -std=gnu11 -O0 -o main.o main.c

C:\Users\nieki\OneDrive\ofc\courses\my_workspace>
```

Fig 1.1.2 Make command

1.2- STARTUP CODE

- The startup file is responsible for setting up the right environments to run the code in main.c file.
- Some part of the startup code is target (processor) dependent.
- Startup code take care of the implementation of the vector table
- Role of startup file:

1. Create an MCU specific vector table for microcontroller.
2. To write a startup code which initializes .data and .bss section in SRAM.
3. Call main ()

```c

#include<stdint.h>

#define SRAM_START  0x20000000U
#define SRAM_SIZE   (128U * 1024U) //128KB
#define SRAM_END    ((SRAM_START) + (SRAM_SIZE))

#define STACK_START   SRAM_END

/* function prototypes of STM32F407x system exception and IRQ handlers */

void Reset_Handler(void);

void NMI_Handler                    (void) __attribute__ ((weak, alias("Default_Handler")));
void HardFault_Handler              (void) __attribute__ ((weak, alias("Default_Handler")));
void MemManage_Handler              (void) __attribute__ ((weak, alias("Default_Handler")));
void BusFault_Handler               (void) __attribute__ ((weak, alias("Default_Handler")));
void UsageFault_Handler             (void) __attribute__ ((weak, alias("Default_Handler")));
void SVC_Handler                    (void) __attribute__ ((weak, alias("Default_Handler")));
void DebugMon_Handler               (void) __attribute__ ((weak, alias("Default_Handler")));
void PendSV_Handler                 (void) __attribute__ ((weak, alias("Default_Handler")));
void SysTick_Handler                (void) __attribute__ ((weak, alias("Default_Handler")));
void WWDG_IRQHandler                (void) __attribute__ ((weak, alias("Default_Handler")));
void PVD_IRQHandler                 (void) __attribute__ ((weak, alias("Default_Handler")));
void TAMP_STAMP_IRQHandler          (void) __attribute__ ((weak, alias("Default_Handler")));
void RTC_WKUP_IRQHandler            (void) __attribute__ ((weak, alias("Default_Handler")));
void RCC_IRQHandler                 (void) __attribute__ ((weak, alias("Default_Handler")));
void EXTI0_IRQHandler               (void) __attribute__ ((weak, alias("Default_Handler")));
void EXTI1_IRQHandler               (void) __attribute__ ((weak, alias("Default_Handler")));
void EXTI2_IRQHandler               (void) __attribute__ ((weak, alias("Default_Handler")));
void EXTI3_IRQHandler               (void) __attribute__ ((weak, alias("Default_Handler")));
void EXTI4_IRQHandler               (void) __attribute__ ((weak, alias("Default_Handler")));
```

```c
void TIM1_BRK_TIM9_IRQHandler       (void) __attribute__ ((weak, alias("Default_Handler")));
void TIM1_UP_TIM10_IRQHandler       (void) __attribute__ ((weak, alias("Default_Handler")));
void TIM1_TRG_COM_TIM11_IRQHandler  (void) __attribute__ ((weak, alias("Default_Handler")));
void TIM1_CC_IRQHandler             (void) __attribute__ ((weak, alias("Default_Handler")));
void TIM2_IRQHandler                (void) __attribute__ ((weak, alias("Default_Handler")));
void TIM3_IRQHandler                (void) __attribute__ ((weak, alias("Default_Handler")));
void TIM4_IRQHandler                (void) __attribute__ ((weak, alias("Default_Handler")));
void I2C1_EV_IRQHandler             (void) __attribute__ ((weak, alias("Default_Handler")));
void I2C1_ER_IRQHandler             (void) __attribute__ ((weak, alias("Default_Handler")));
void I2C2_EV_IRQHandler             (void) __attribute__ ((weak, alias("Default_Handler")));
void I2C2_ER_IRQHandler             (void) __attribute__ ((weak, alias("Default_Handler")));
void SPI1_IRQHandler                (void) __attribute__ ((weak, alias("Default_Handler")));
void SPI2_IRQHandler                (void) __attribute__ ((weak, alias("Default_Handler")));
void USART1_IRQHandler              (void) __attribute__ ((weak, alias("Default_Handler")));
void USART2_IRQHandler              (void) __attribute__ ((weak, alias("Default_Handler")));
void USART3_IRQHandler              (void) __attribute__ ((weak, alias("Default_Handler")));
void EXTI15_10_IRQHandler           (void) __attribute__ ((weak, alias("Default_Handler")));
void RTC_Alarm_IRQHandler           (void) __attribute__ ((weak, alias("Default_Handler")));
void OTG_FS_WKUP_IRQHandler         (void) __attribute__ ((weak, alias("Default_Handler")));
void TIM8_BRK_TIM12_IRQHandler      (void) __attribute__ ((weak, alias("Default_Handler")));
void TIM8_UP_TIM13_IRQHandler       (void) __attribute__ ((weak, alias("Default_Handler")));
void TIM8_TRG_COM_TIM14_IRQHandler  (void) __attribute__ ((weak, alias("Default_Handler")));
void TIM8_CC_IRQHandler             (void) __attribute__ ((weak, alias("Default_Handler")));
void DMA1_Stream7_IRQHandler        (void) __attribute__ ((weak, alias("Default_Handler")));
void FSMC_IRQHandler                (void) __attribute__ ((weak, alias("Default_Handler")));
void SDIO_IRQHandler                (void) __attribute__ ((weak, alias("Default_Handler")));
void TIM5_IRQHandler                (void) __attribute__ ((weak, alias("Default_Handler")));
void SPI3_IRQHandler                (void) __attribute__ ((weak, alias("Default_Handler")));
void UART4_IRQHandler               (void) __attribute__ ((weak, alias("Default_Handler")));
void UART5_IRQHandler               (void) __attribute__ ((weak, alias("Default_Handler")));
void TIM6_DAC_IRQHandler            (void) __attribute__ ((weak, alias("Default_Handler")));
void TIM7_IRQHandler                (void) __attribute__ ((weak, alias("Default_Handler")));
```

```
79  void DMA2_Stream1_IRQHandler        (void) __attribute__ ((weak, alias("Default_Handler")));
80  void DMA2_Stream2_IRQHandler        (void) __attribute__ ((weak, alias("Default_Handler")));
81  void DMA2_Stream3_IRQHandler        (void) __attribute__ ((weak, alias("Default_Handler")));
82  void DMA2_Stream4_IRQHandler        (void) __attribute__ ((weak, alias("Default_Handler")));
83  void ETH_IRQHandler                 (void) __attribute__ ((weak, alias("Default_Handler")));
84  void ETH_WKUP_IRQHandler            (void) __attribute__ ((weak, alias("Default_Handler")));
85  void CAN2_TX_IRQHandler             (void) __attribute__ ((weak, alias("Default_Handler")));
86  void CAN2_RX0_IRQHandler            (void) __attribute__ ((weak, alias("Default_Handler")));
87  void CAN2_RX1_IRQHandler            (void) __attribute__ ((weak, alias("Default_Handler")));
88  void CAN2_SCE_IRQHandler            (void) __attribute__ ((weak, alias("Default_Handler")));
89  void OTG_FS_IRQHandler              (void) __attribute__ ((weak, alias("Default_Handler")));
90  void DMA2_Stream5_IRQHandler        (void) __attribute__ ((weak, alias("Default_Handler")));
91  void DMA2_Stream6_IRQHandler        (void) __attribute__ ((weak, alias("Default_Handler")));
92  void DMA2_Stream7_IRQHandler        (void) __attribute__ ((weak, alias("Default_Handler")));
93  void USART6_IRQHandler              (void) __attribute__ ((weak, alias("Default_Handler")));
94  void I2C3_EV_IRQHandler             (void) __attribute__ ((weak, alias("Default_Handler")));
95  void I2C3_ER_IRQHandler             (void) __attribute__ ((weak, alias("Default_Handler")));
96  void OTG_HS_EP1_OUT_IRQHandler      (void) __attribute__ ((weak, alias("Default_Handler")));
97  void OTG_HS_EP1_IN_IRQHandler       (void) __attribute__ ((weak, alias("Default_Handler")));
98  void OTG_HS_WKUP_IRQHandler         (void) __attribute__ ((weak, alias("Default_Handler")));
99  void OTG_HS_IRQHandler              (void) __attribute__ ((weak, alias("Default_Handler")));
100 void DCMI_IRQHandler                (void) __attribute__ ((weak, alias("Default_Handler")));
101 void CRYP_IRQHandler                (void) __attribute__ ((weak, alias("Default_Handler")));
102 void HASH_RNG_IRQHandler            (void) __attribute__ ((weak, alias("Default_Handler")));
103 void FPU_IRQHandler                 (void) __attribute__ ((weak, alias("Default_Handler")));
104
```

Fig 1.2.1 Startup code

In user defined function we use variable attribute for the define function to store inside the variable:

- Weak: Lets programmer override already defined weak function (dummy function) with the same function name.
- Alias: Lets programmer give any alias name for same function.

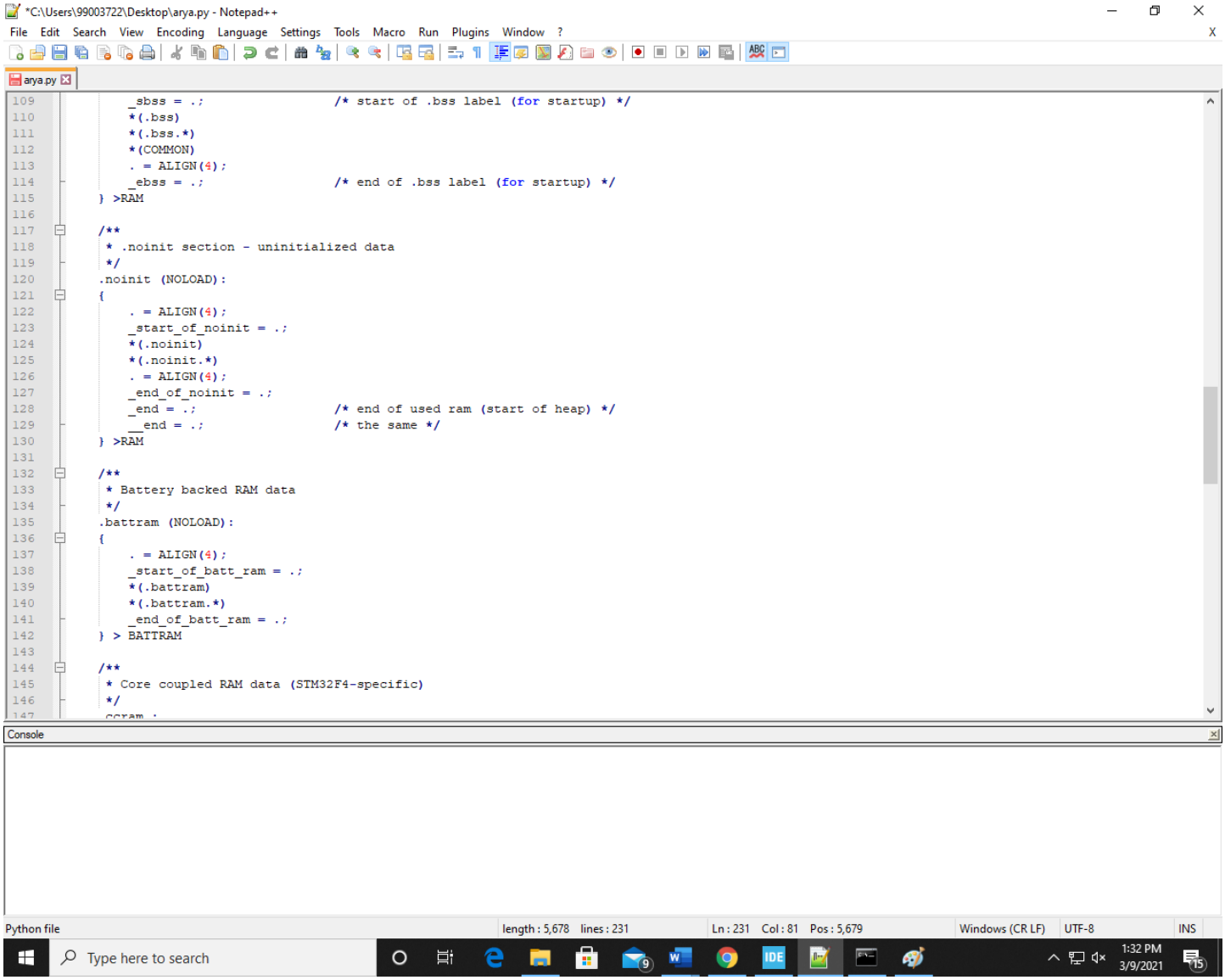
The startup.o file generated is of elf executable format, various sections of which are shown below:

```
C:\Windows\System32\cmd.exe

C:\Users\Training\STM32CubeIDE\workspace_1.5.0\new\Src>make -f Makefile.mak
arm-none-eabi-gcc -c -mcpu=cortex-m4 -mthumb -std=gnu11 -Wall -O0  -o stm32_startup.o stm32_startup.c

C:\Users\Training\STM32CubeIDE\workspace_1.5.0\new\Src>arm-none-eabi-objdump.exe -h stm32_startup.o

stm32_startup.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000090  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  000000c4  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  000000c4  2**0
                  ALLOC
  3 .isr_vector   00000188  00000000  00000000  000000c4  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, DATA
  4 .comment      0000004e  00000000  00000000  0000024c  2**0
                  CONTENTS, READONLY
  5 .ARM.attributes 0000002e  00000000  00000000  0000029a  2**0
                  CONTENTS, READONLY

C:\Users\Training\STM32CubeIDE\workspace_1.5.0\new\Src>
```

Fig 1.2.2: Startup command

1.3- LINKER SCRIPT

- Linker script is written using the GNU linker command script.
- Linkers take one or more object files or libraries as input and combines them to create a single executable file as output.
- Linker file is the file which tell about how the text of the whole program will be merged.
- Linker scripts decide how different sections of object file should be merged to create an output file.
- Reset handler is the entry point to the application
- Entry command is used to set the “Entry point address” information in the header of final elf file generated.



```
C:\Windows\System32\cmd.exe
C:\Users\Training\Documents\Embedded C\STM project\Activity1>arm-none-eabi-gcc -nostdlib -T stm32_ls.ld *.o -o final.elf
c:/program files (x86)/gnu arm embedded toolchain/10 2020-q4-major/bin/../lib/gcc/arm-none-eabi/10.2.1/../../../../arm-none-eabi/bin/ld.exe: stm3
2_startup.o: in function `Reset_Handler':
stm32_startup.c:(.text+0x80): undefined reference to `_la_data'
collect2.exe: error: ld returned 1 exit status

C:\Users\Training\Documents\Embedded C\STM project\Activity1>arm-none-eabi-gcc -nostdlib -T stm32_ls.ld *.o -o final.elf

C:\Users\Training\Documents\Embedded C\STM project\Activity1>dir
 Volume in drive C has no label.
 Volume Serial Number is F8FC-05CA

 Directory of C:\Users\Training\Documents\Embedded C\STM project\Activity1

23-02-2021  14:29    <DIR>          .
23-02-2021  14:29    <DIR>          ..
23-02-2021  14:29           135,752 final.elf
23-02-2021  14:24             2,034 main.c
23-02-2021  14:24             1,940 main.o
23-02-2021  11:35               220 Makefile.mak
23-02-2021  14:28               662 stm32_ls.ld
23-02-2021  14:27            12,847 stm32_startup.c
23-02-2021  14:24             5,688 stm32_startup.o
               7 File(s)        159,143 bytes
               2 Dir(s)  125,771,227,136 bytes free

C:\Users\Training\Documents\Embedded C\STM project\Activity1>
```

Fig 1.3.1 command to generate final. Elf file

1.4- DEBUGGING TECHNIQUES

Debug Level can be set among four levels:

• None: Level 0 produces no debug information at all;

• Minimal (-g1): Level 1 produces minimal information, enough for making back traces in parts of the program for which no debug is planned. This includes descriptions of functions and external variables, and line number tables, but no information about local variables.

• Default (-g): Produce debugging information in the operating system's native format (stabs, COFF, XCOFF, or DWARF). GDB can work with this debugging information.

• Maximal (-g3): Level 3 includes extra information, such as all the macro definitions present in the program. Some debuggers support macro expansion when -g3 is used.

Activity 2 – IMPLEMENTATION OF PROTOCOLS USING STM IDE

Implementation of protocols for STM32F407VG microcontroller featuring ARM32 bit ARM-cortex - M4 with FPU core using HAL library.

2.1 GPIO:

Toggling LED at pin PD12 at GREEN_LED_GPIO_PORT. Serial wire is enabled at pin PA13.

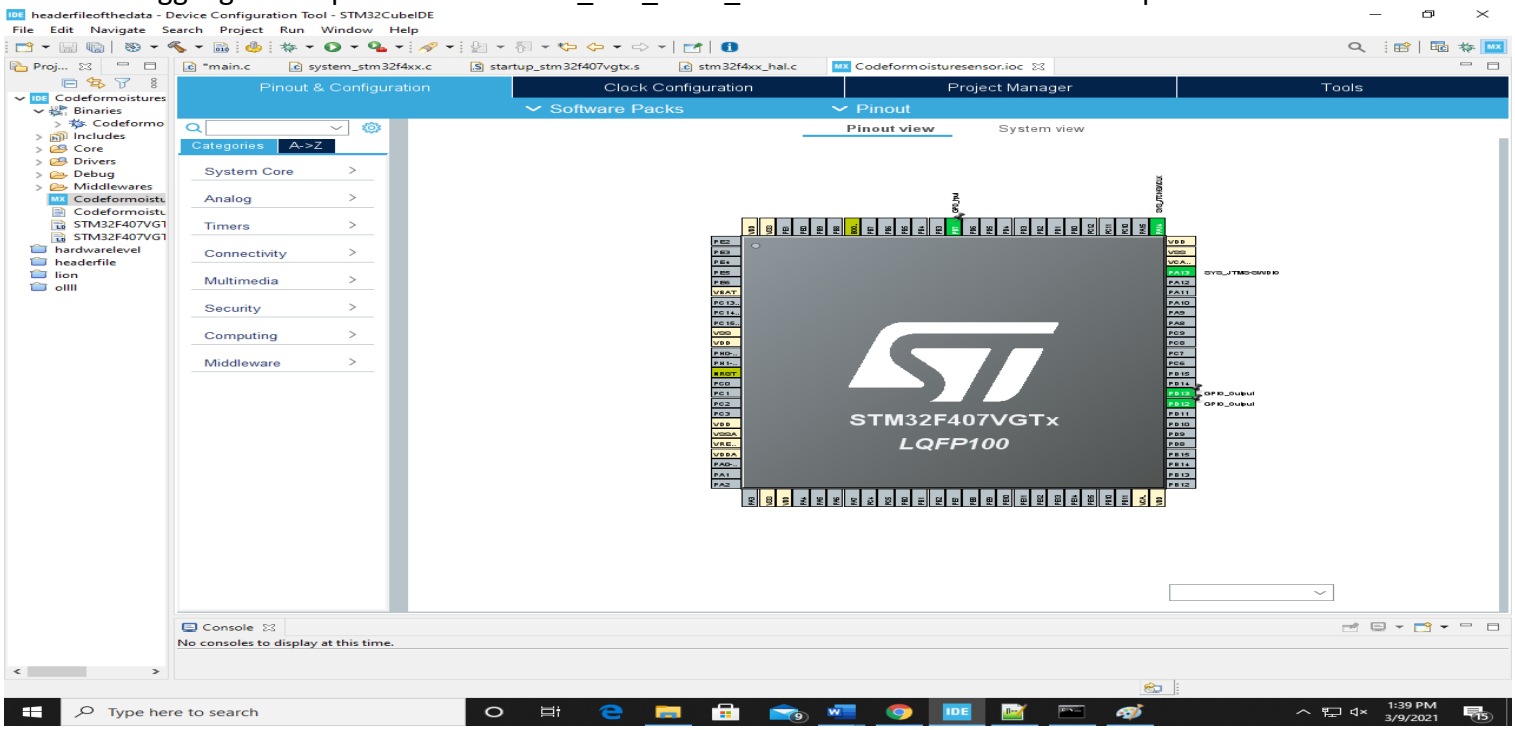


Fig: 2.1.1 GPIO pin configuration

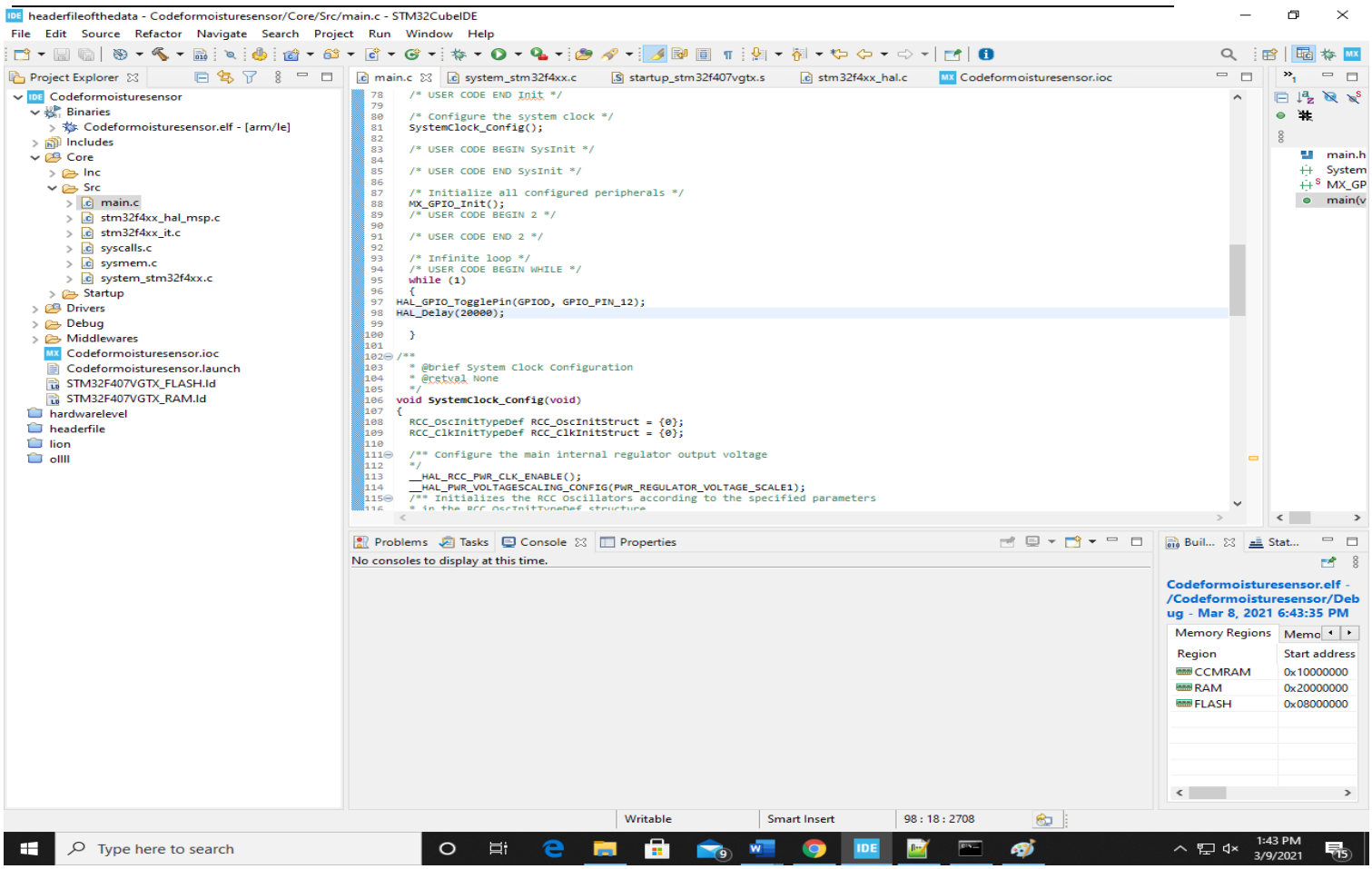


Fig: 2.1.2 GPIO configuration code

2.2 EXTI:

PA0 works as an external interrupt.
When the blue button is pressed the Green LED toggles.

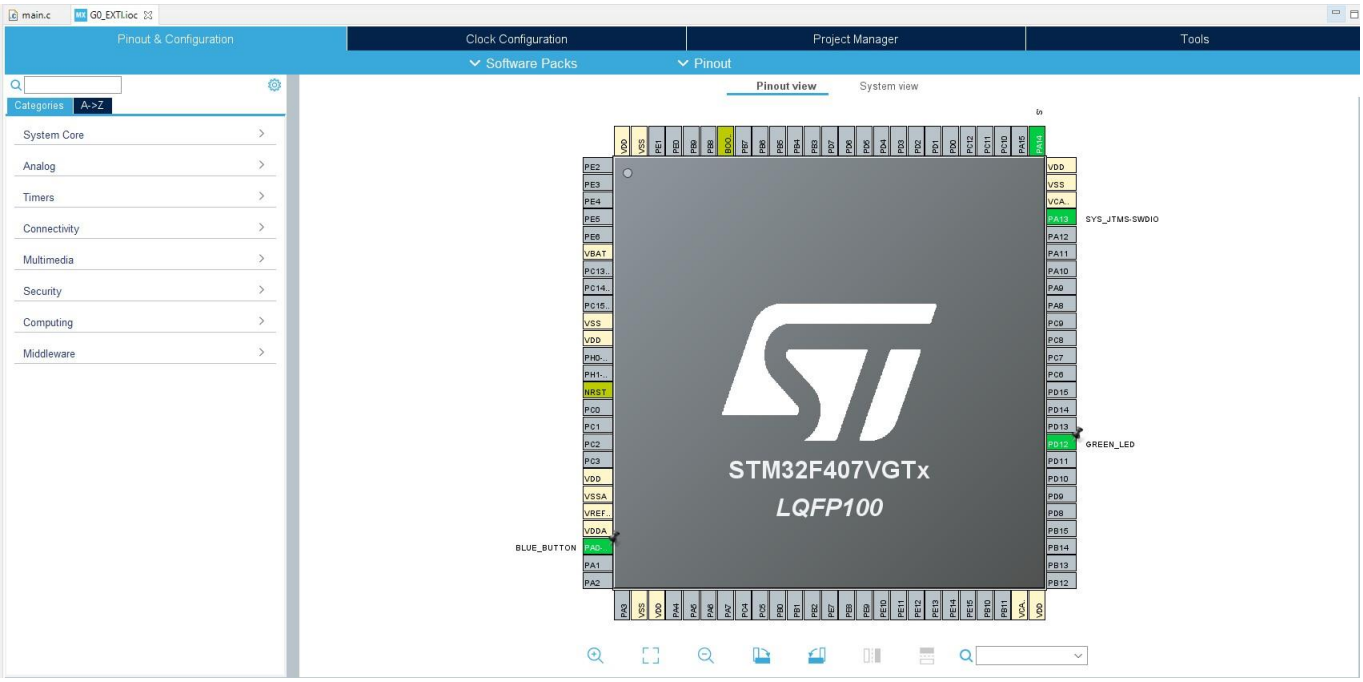


Fig: 2.2.1 EXTI pin configuration

In the main.c file a flag is initialized and if the flag == 1, the condition under the if loop executed to toggle the LED at PD12.

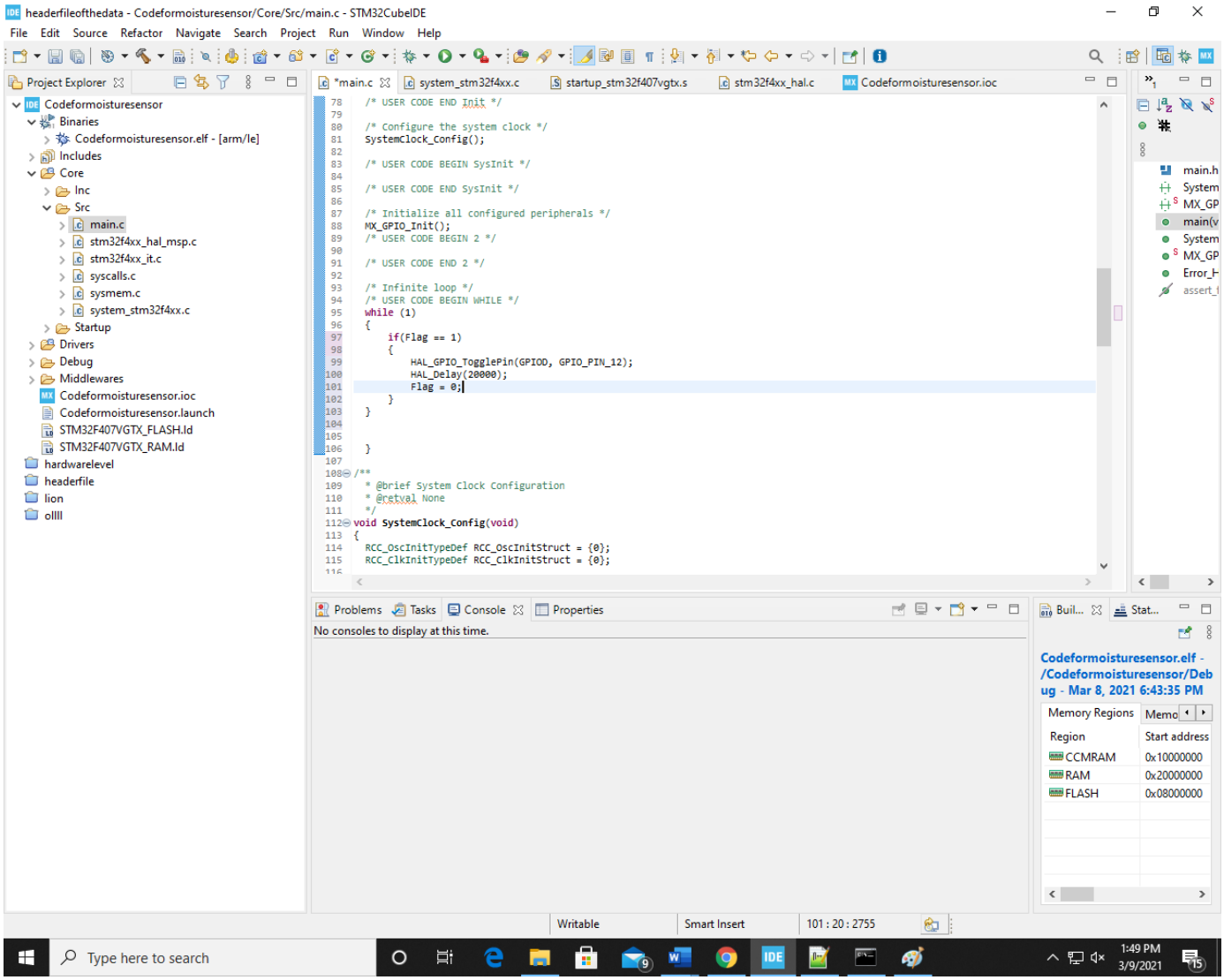


Fig: 2.2.1 EXTI configuration code

2.3 ADC

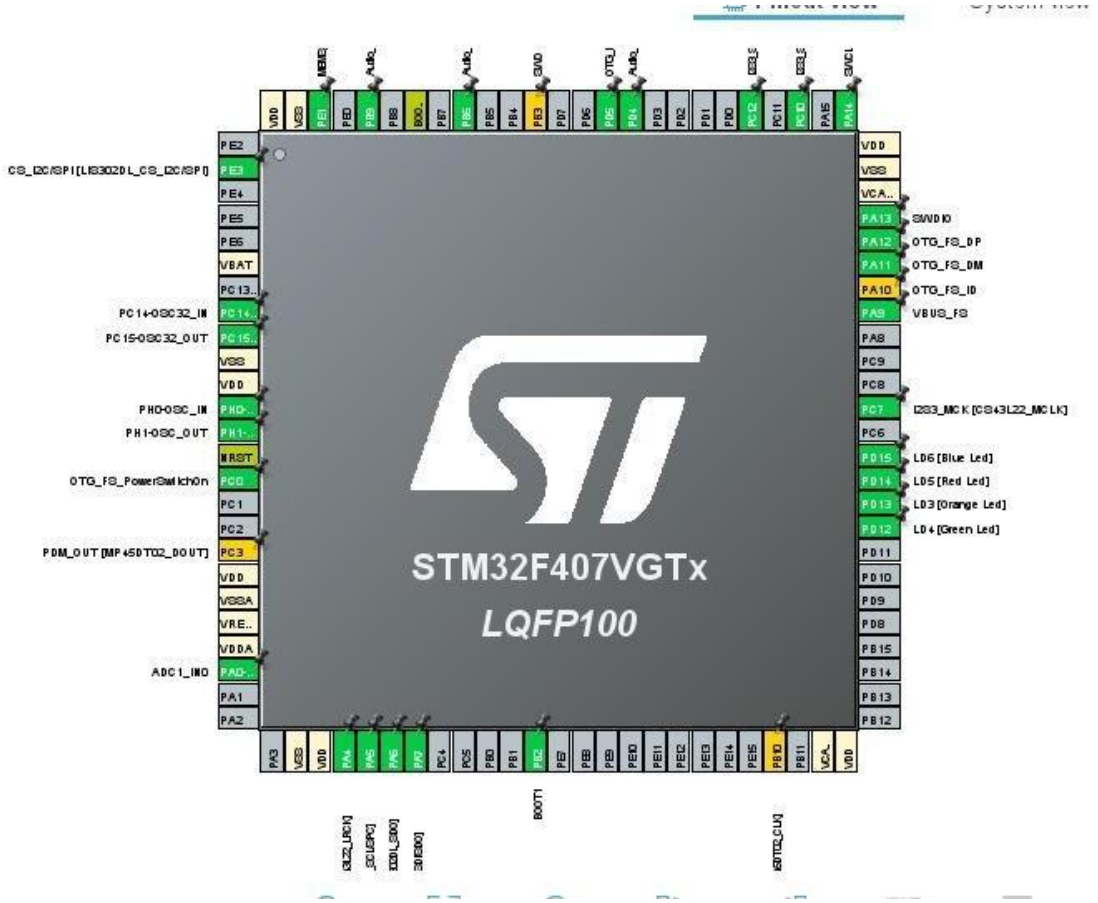


Fig: 2.3.1 ADC pin configuration

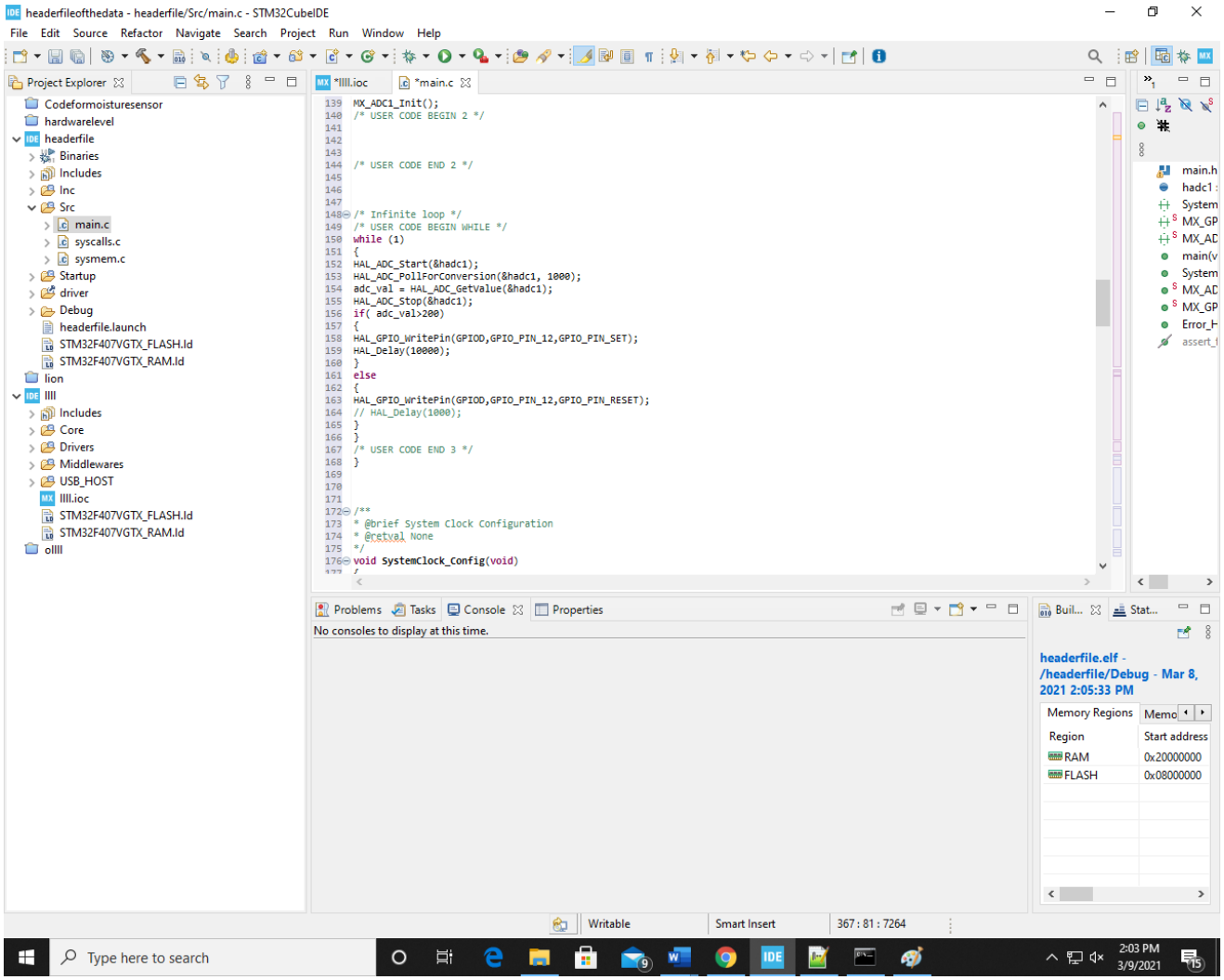
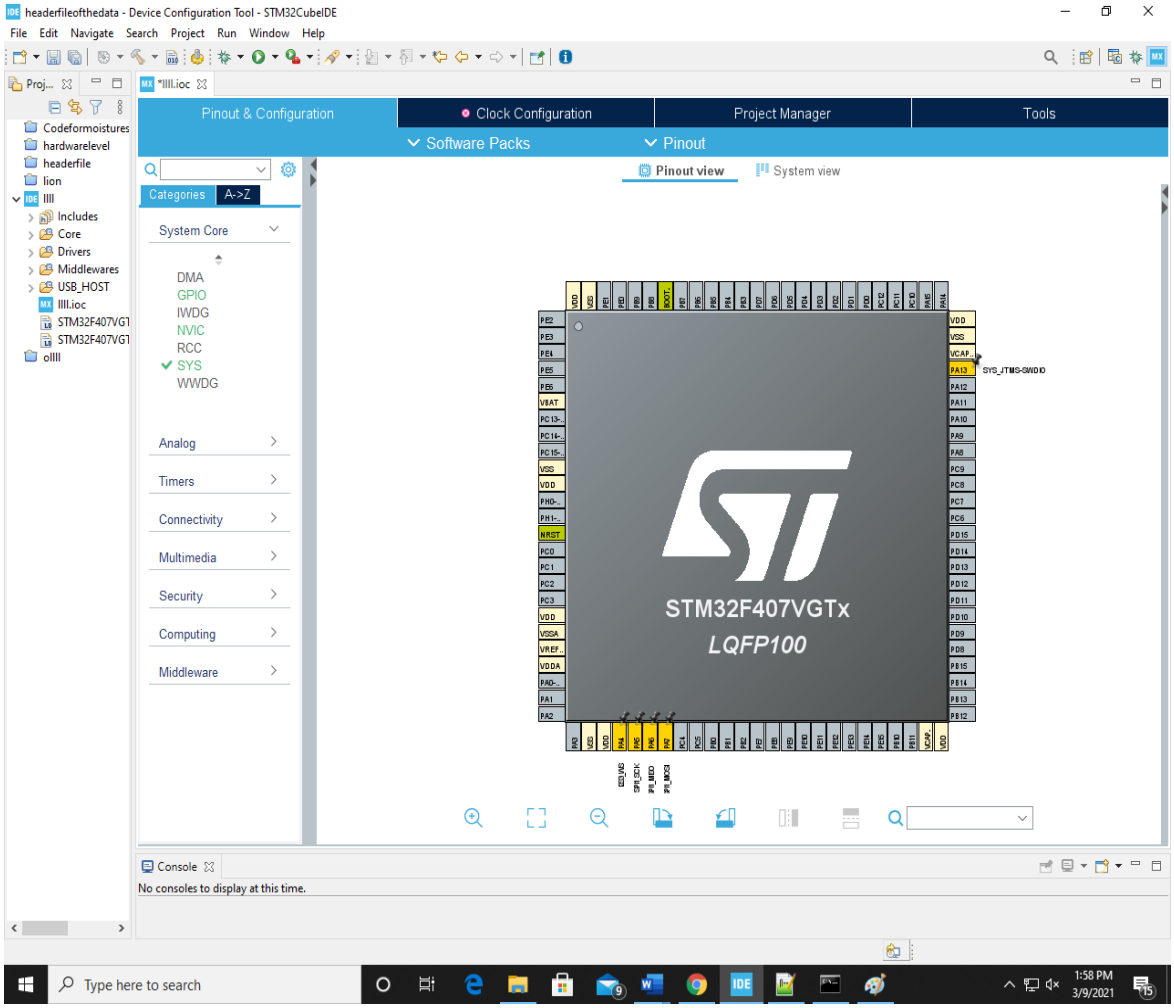


Fig: 2.3.2 ADC configuration code

2.4 SPI



.

Fig: 2.4.1 SPI Pin configuration

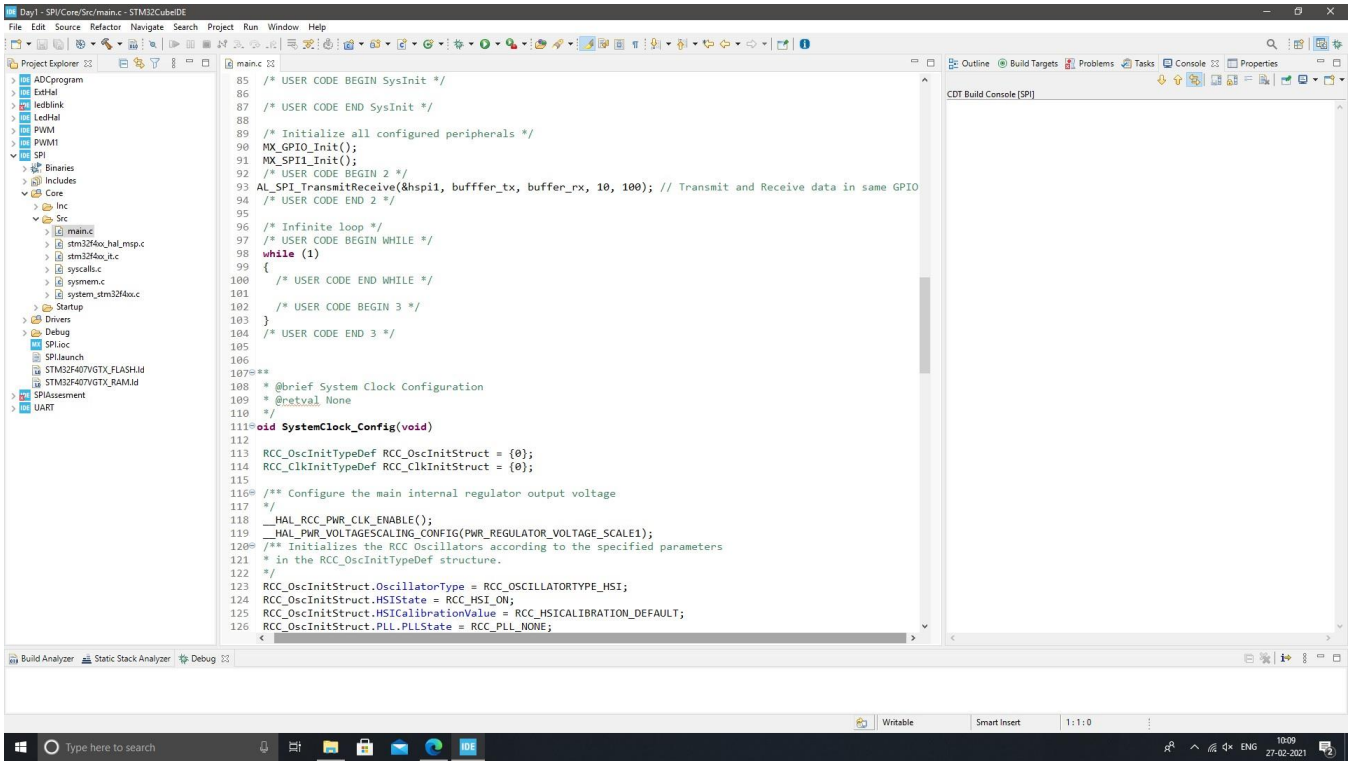


Fig: 2.4.2 SPI configuration code

2.5 UART

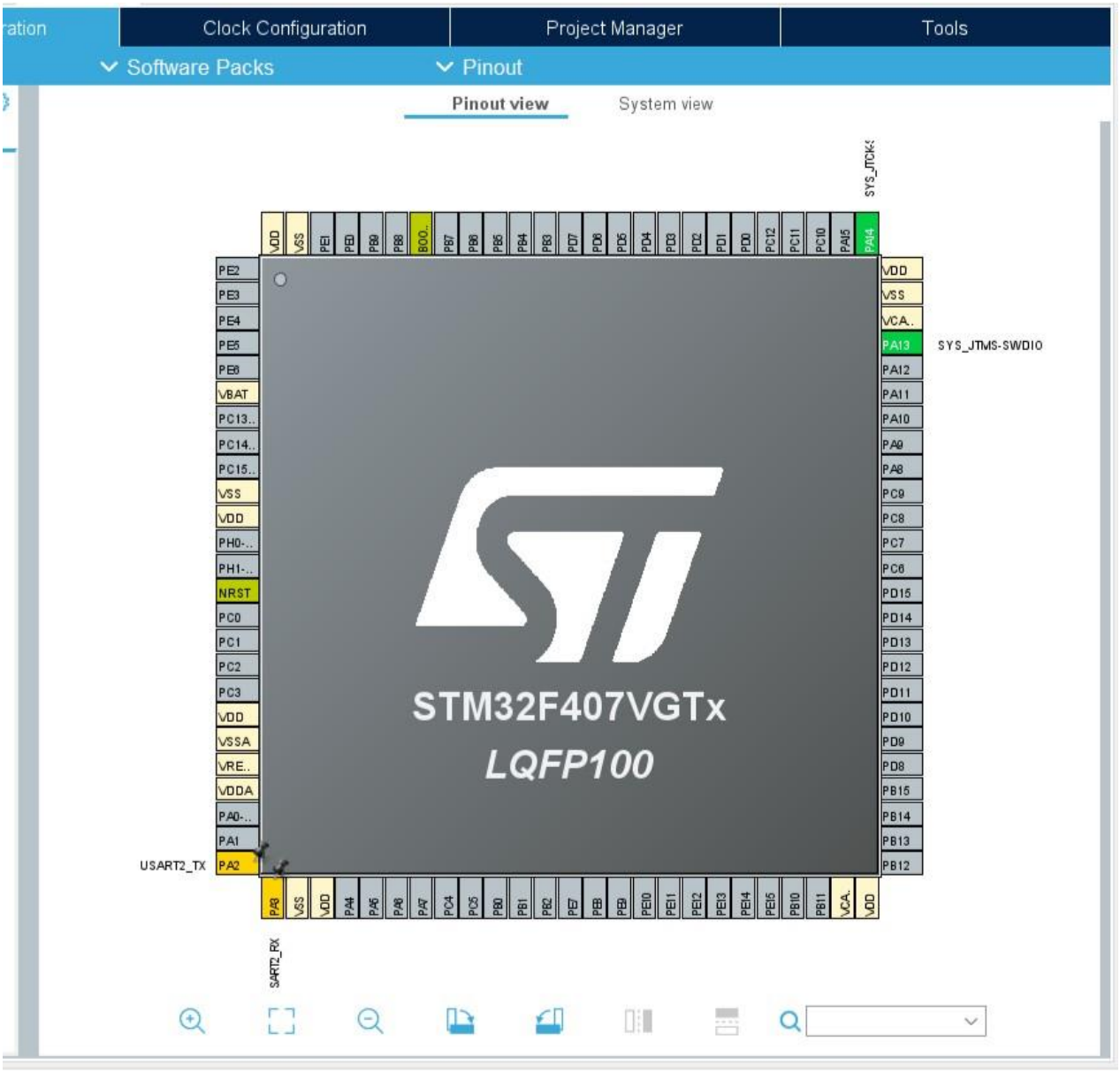


Fig: 2.5.1 UART Pin configuration

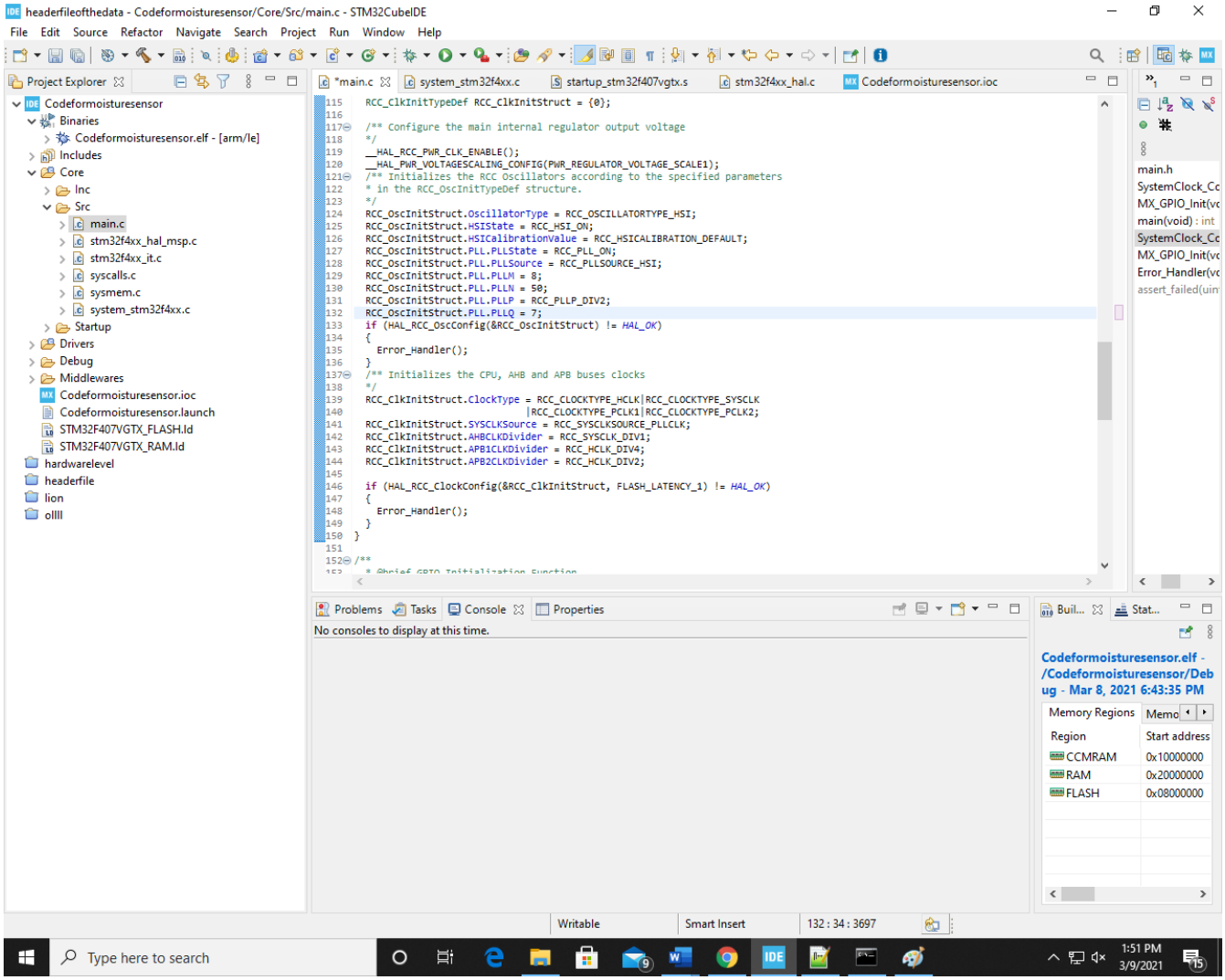


Fig: 2.5.2 UART configuration code