



# 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

<b>ACTIVITY 1 – COMPILATION APPROACH.....</b>	<b>4</b>
1.1- MAKE FILE .....	4
1.2- STARTUP CODE.....	5
1.3- LINKER FILE.....	8
1.4- DEBUGGING TECHNIQUES.....	9
 <b>ACTIVITY 2 – IMPLEMENTATION OF PROTOCOLS USING STM IDE.....</b>	 <b>10</b>
2.1 GPIO .....	10
2.2 EXTI .....	12
2.3 ADC.....	14
2.4 SPI.....	16
2.5 UART .....	19

## Activity 1 – COMPILATION APPROACH

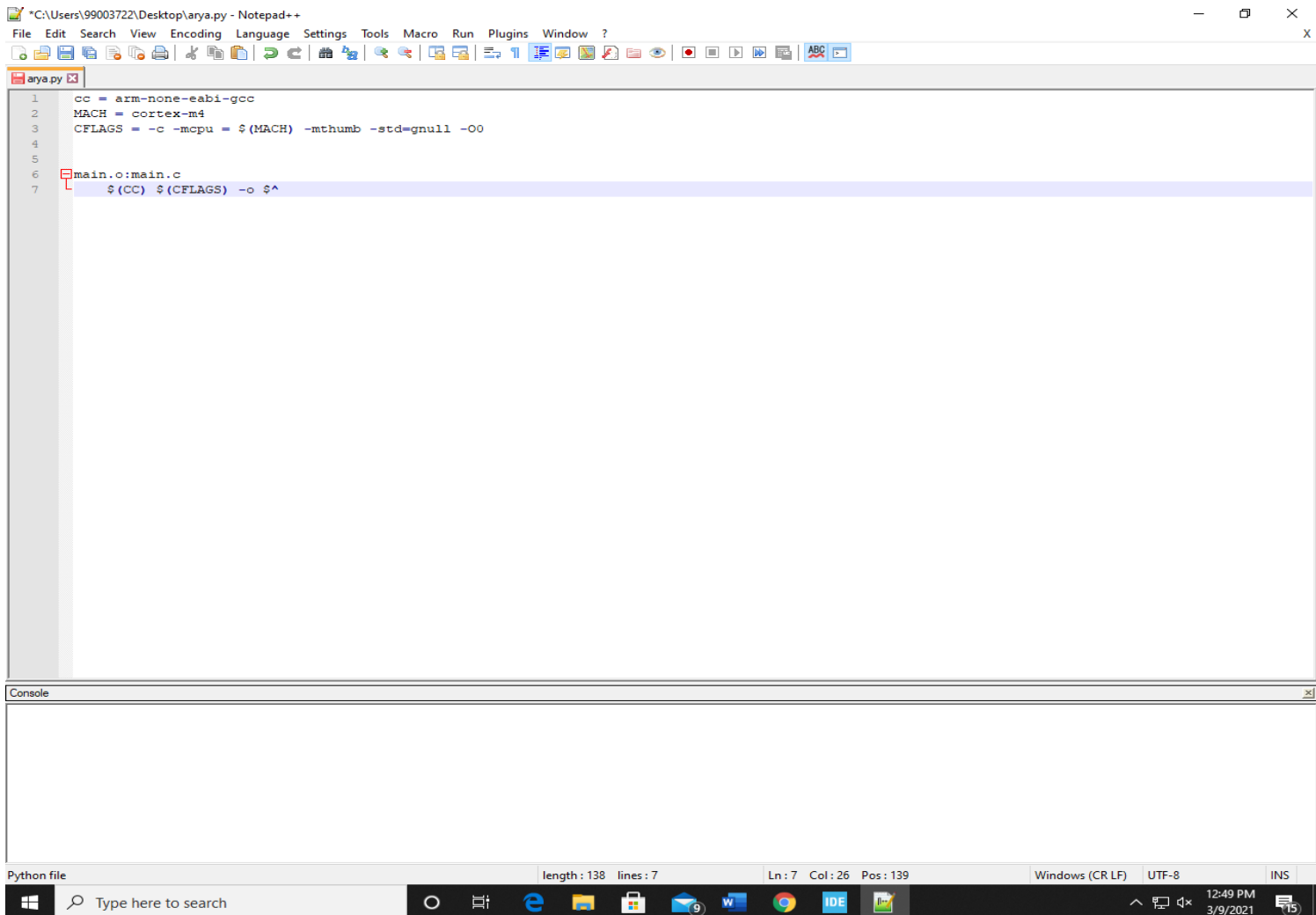
This process of the sample program for ARM Cortex Mx processor-based boards.

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:

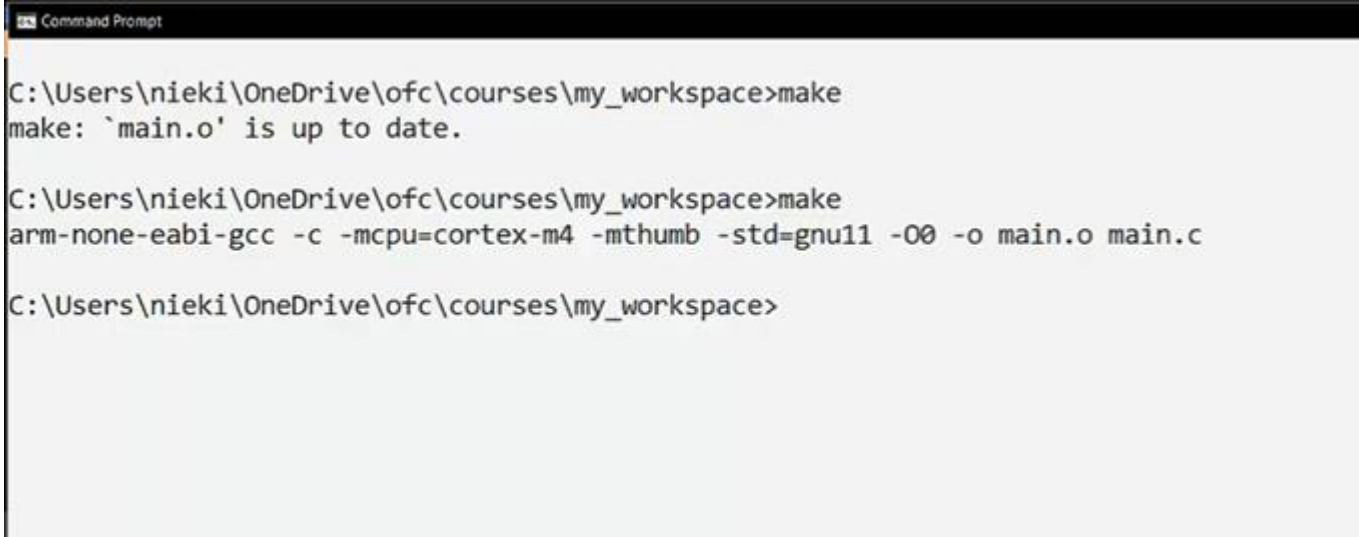


```
1 cc = arm-none-eabi-gcc
2 MACH = cortex-m4
3 CFLAGS = -c -mcpu = $(MACH) -mthumb -std=gnu11 -O0
4
5
6 main.o:main.c
7     $(CC) $(CFLAGS) -o $^
```

The screenshot shows a Notepad++ window titled '\*C:\Users\99003722\Desktop\arya.py - Notepad++'. The menu bar includes File, Edit, Search, View, Encoding, Language, Settings, Tools, Macro, Run, Plugins, Window, and ?. The toolbar contains various icons for file operations and editing. The text area shows the Makefile content for 'arya.py'. The status bar at the bottom indicates 'Python file', 'length: 138 lines: 7', 'Ln: 7 Col: 26 Pos: 139', 'Windows (CR LF)', 'UTF-8', and 'INS'. The Windows taskbar is visible at the bottom with the search bar and several application icons.

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 ()



```

1  #include<stdint.h>
2
3
4  #define SRAM_START    0x20000000U
5  #define SRAM_SIZE     (128U * 1024U) //128KB
6  #define SRAM_END      ((SRAM_START) + (SRAM_SIZE))
7
8  #define STACK_START    SRAM_END
9
10 /* function prototypes of STM32F407x system exception and IRQ handlers */
11
12 void Reset_Handler(void);
13
14 void NMI_Handler          (void) __attribute__((weak, alias("Default_Handler")));
15 void HardFault_Handler    (void) __attribute__((weak, alias("Default_Handler")));
16 void MemManage_Handler    (void) __attribute__((weak, alias("Default_Handler")));
17 void BusFault_Handler     (void) __attribute__((weak, alias("Default_Handler")));
18 void UsageFault_Handler   (void) __attribute__((weak, alias("Default_Handler")));
19 void SVC_Handler          (void) __attribute__((weak, alias("Default_Handler")));
20 void DebugMon_Handler     (void) __attribute__((weak, alias("Default_Handler")));
21 void PendSV_Handler       (void) __attribute__((weak, alias("Default_Handler")));
22 void SysTick_Handler      (void) __attribute__((weak, alias("Default_Handler")));
23 void WWDG_IRQHandler       (void) __attribute__((weak, alias("Default_Handler")));
24 void PVD_IRQHandler        (void) __attribute__((weak, alias("Default_Handler")));
25 void TAMP_STAMP_IRQHandler (void) __attribute__((weak, alias("Default_Handler")));
26 void RTC_WKUP_IRQHandler   (void) __attribute__((weak, alias("Default_Handler")));
27 void RCC_IRQHandler        (void) __attribute__((weak, alias("Default_Handler")));
28 void EXTI0_IRQHandler      (void) __attribute__((weak, alias("Default_Handler")));
29 void EXTI1_IRQHandler      (void) __attribute__((weak, alias("Default_Handler")));
30 void EXTI2_IRQHandler      (void) __attribute__((weak, alias("Default_Handler")));
31 void EXTI3_IRQHandler      (void) __attribute__((weak, alias("Default_Handler")));
32 void EXTI4_IRQHandler      (void) __attribute__((weak, alias("Default_Handler")));
33
34 void TIM1_BRK_TIM9_IRQHandler (void) __attribute__((weak, alias("Default_Handler")));
35 void TIM1_UP_TIM10_IRQHandler (void) __attribute__((weak, alias("Default_Handler")));
36 void TIM1_TRG_COM_TIM11_IRQHandler (void) __attribute__((weak, alias("Default_Handler")));
37 void TIM1_CC_IRQHandler       (void) __attribute__((weak, alias("Default_Handler")));
38 void TIM2_IRQHandler          (void) __attribute__((weak, alias("Default_Handler")));
39 void TIM3_IRQHandler          (void) __attribute__((weak, alias("Default_Handler")));
40 void TIM4_IRQHandler          (void) __attribute__((weak, alias("Default_Handler")));
41 void I2C1_EV_IRQHandler       (void) __attribute__((weak, alias("Default_Handler")));
42 void I2C1_ER_IRQHandler       (void) __attribute__((weak, alias("Default_Handler")));
43 void I2C2_EV_IRQHandler       (void) __attribute__((weak, alias("Default_Handler")));
44 void I2C2_ER_IRQHandler       (void) __attribute__((weak, alias("Default_Handler")));
45 void SPI1_IRQHandler          (void) __attribute__((weak, alias("Default_Handler")));
46 void SPI2_IRQHandler          (void) __attribute__((weak, alias("Default_Handler")));
47 void USART1_IRQHandler        (void) __attribute__((weak, alias("Default_Handler")));
48 void USART2_IRQHandler        (void) __attribute__((weak, alias("Default_Handler")));
49 void USART3_IRQHandler        (void) __attribute__((weak, alias("Default_Handler")));
50 void EXTI15_10_IRQHandler     (void) __attribute__((weak, alias("Default_Handler")));
51 void RTC_Alarm_IRQHandler     (void) __attribute__((weak, alias("Default_Handler")));
52 void OTG_FS_WKUP_IRQHandler   (void) __attribute__((weak, alias("Default_Handler")));
53 void TIM8_BRK_TIM12_IRQHandler (void) __attribute__((weak, alias("Default_Handler")));
54 void TIM8_UP_TIM13_IRQHandler (void) __attribute__((weak, alias("Default_Handler")));
55 void TIM8_TRG_COM_TIM14_IRQHandler (void) __attribute__((weak, alias("Default_Handler")));
56 void TIM8_CC_IRQHandler       (void) __attribute__((weak, alias("Default_Handler")));
57 void DMA1_Stream7_IRQHandler   (void) __attribute__((weak, alias("Default_Handler")));
58 void FSMC_IRQHandler          (void) __attribute__((weak, alias("Default_Handler")));
59 void SDIO_IRQHandler          (void) __attribute__((weak, alias("Default_Handler")));
60 void TIM5_IRQHandler          (void) __attribute__((weak, alias("Default_Handler")));
61 void SPI3_IRQHandler          (void) __attribute__((weak, alias("Default_Handler")));
62 void UART4_IRQHandler         (void) __attribute__((weak, alias("Default_Handler")));
63 void UART5_IRQHandler         (void) __attribute__((weak, alias("Default_Handler")));
64 void TIM6_DAC_IRQHandler      (void) __attribute__((weak, alias("Default_Handler")));
65 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

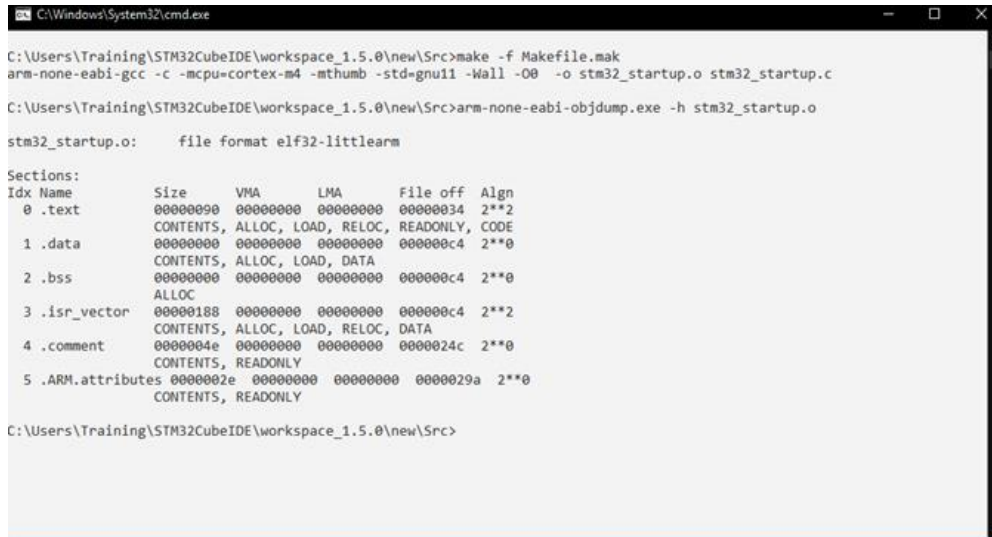
```

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

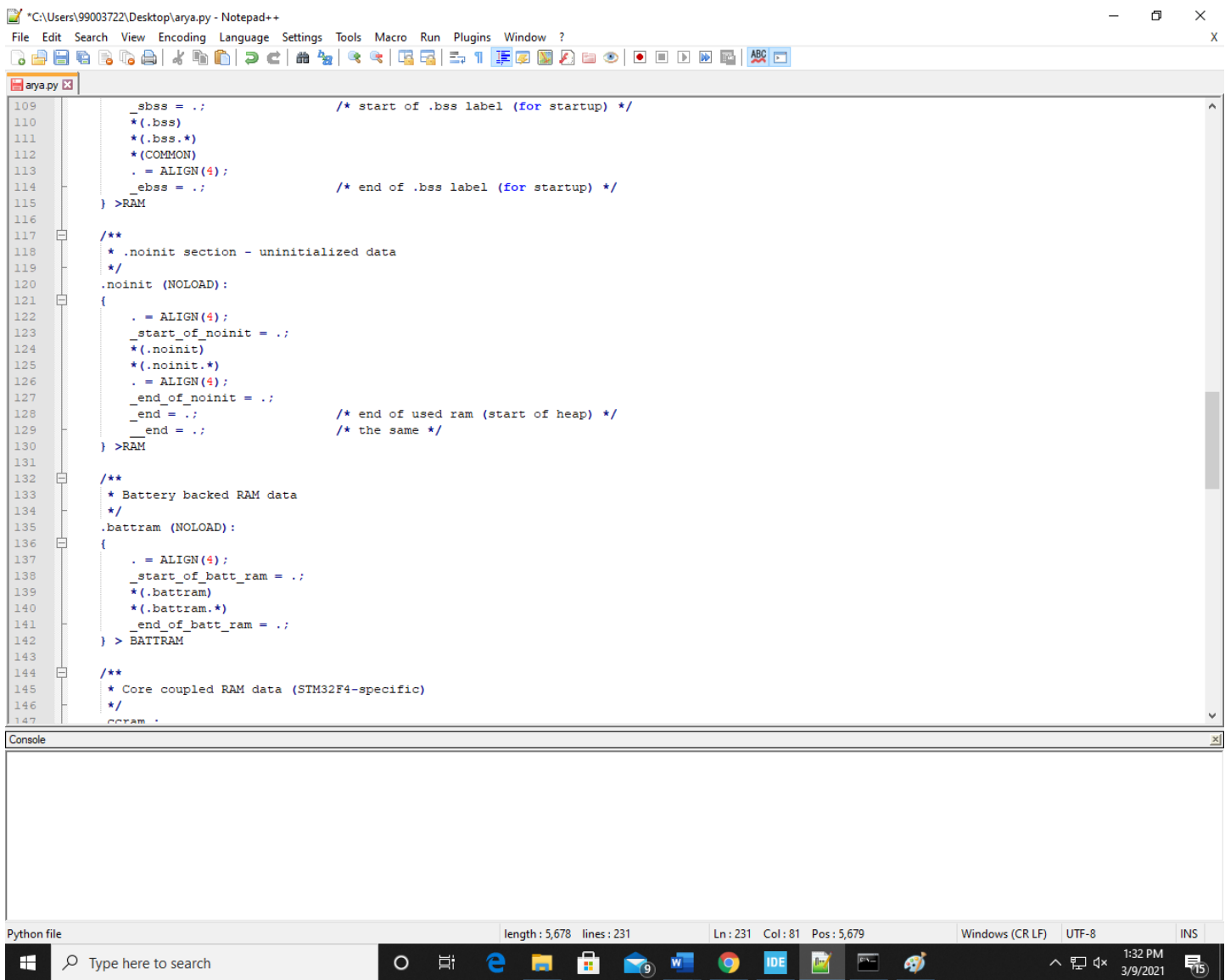
C:\Users\Training\STM32CubeIDE\workspace_1.5.0\new\Src>

```

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.



```
109  _sbss = .;                /* start of .bss label (for startup) */
110  *(.bss)
111  *(.bss.*)
112  *(COMMON)
113  . = ALIGN(4);
114  _ebss = .;                /* end of .bss label (for startup) */
115  } >RAM
116
117  /**
118   * .noinit section - uninitialized data
119   */
120  .noinit (NOLOAD):
121  {
122      . = ALIGN(4);
123      _start_of_noinit = .;
124      *(.noinit)
125      *(.noinit.*)
126      . = ALIGN(4);
127      _end_of_noinit = .;
128      _end = .;              /* end of used ram (start of heap) */
129      _end = .;              /* the same */
130  } >RAM
131
132  /**
133   * Battery backed RAM data
134   */
135  .battmem (NOLOAD):
136  {
137      . = ALIGN(4);
138      _start_of_batt_ram = .;
139      *(.battmem)
140      *(.battmem.*)
141      _end_of_batt_ram = .;
142  } > BATTMEM
143
144  /**
145   * Core coupled RAM data (STM32F4-specific)
146   */
147  .core =
```

Console

Python file length: 5,678 lines: 231 Ln: 231 Col: 81 Pos: 5,679 Windows (CR LF) UTF-8 INS

Type here to search

1:32 PM 3/9/2021



```

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:

Toggle LED at pin PD12 at GREEN\_LED\_GPIO\_PORT. Serial wire is enabled at pin PA13.

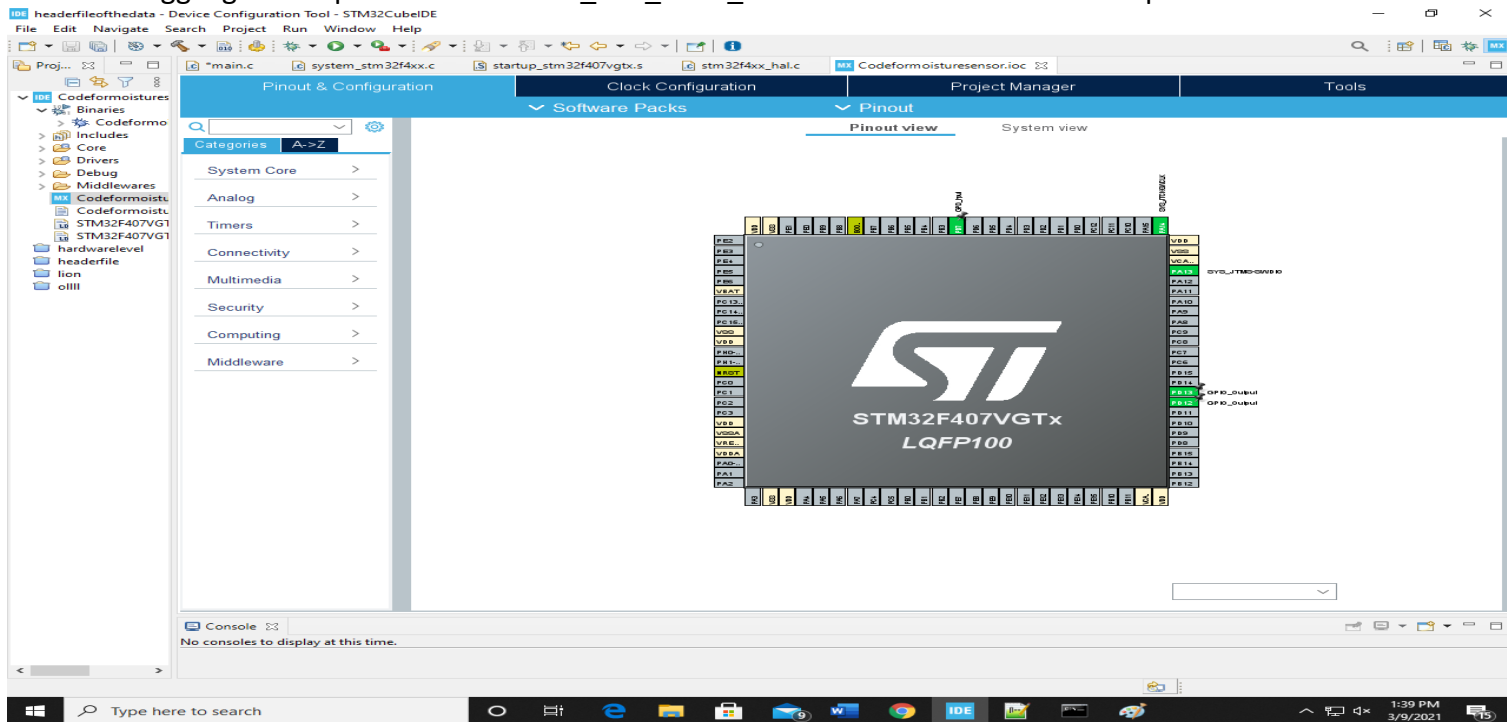


Fig: 2.1.1 GPIO pin configuration

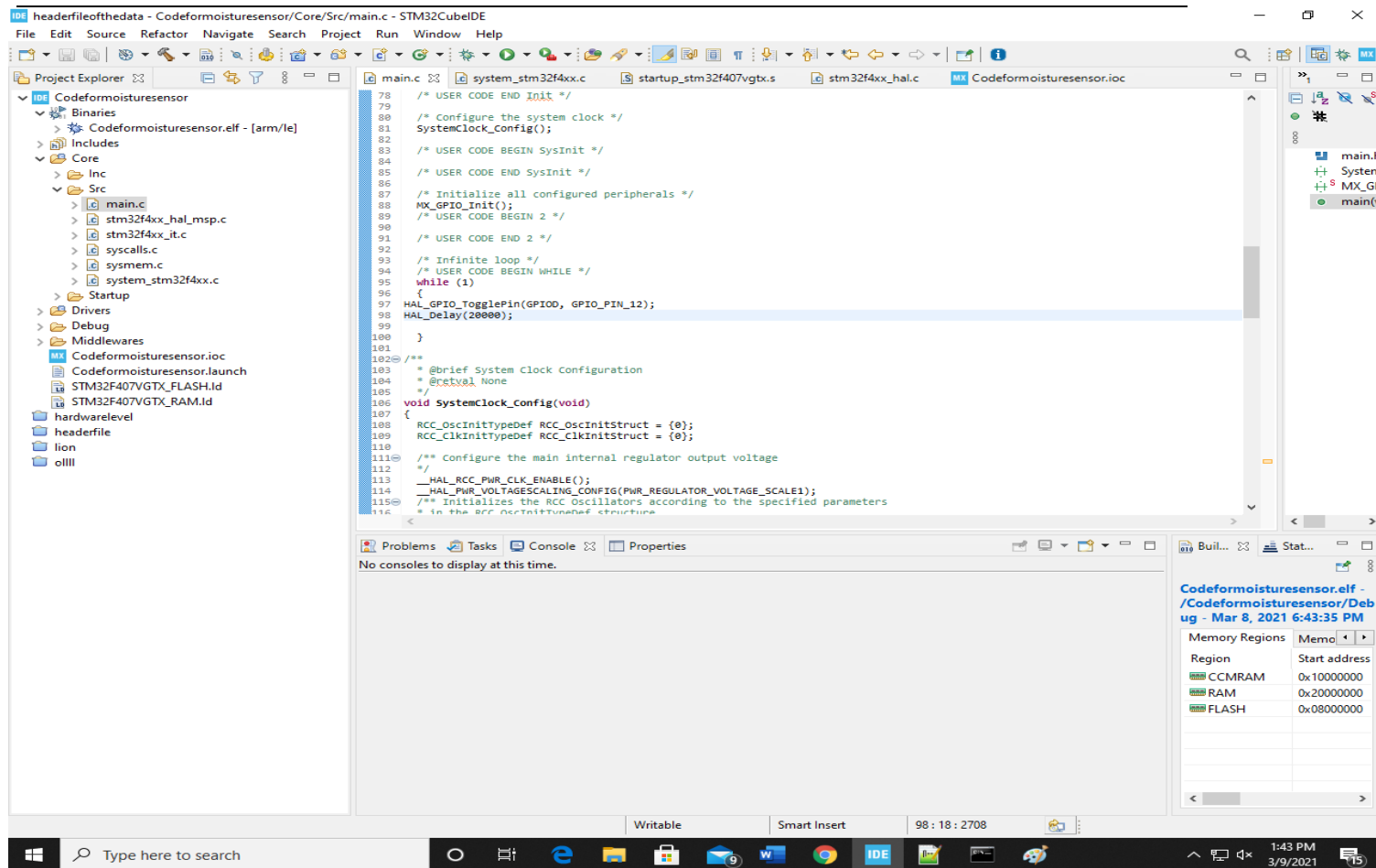


Fig: 2.1.2 GPIO configuration code

## 2.2 EXTI:

PA0 works as an external interrupt.

When the blue button is pressed the Green LED toggles.

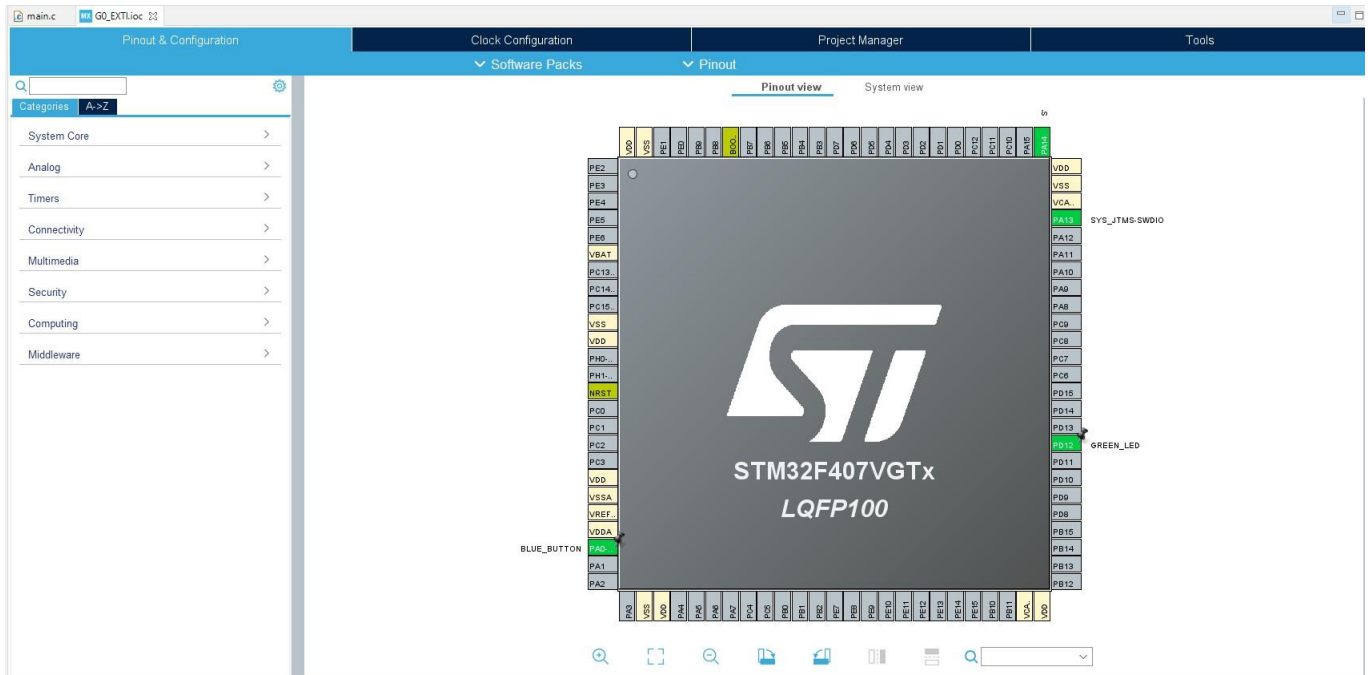


Fig: 2.2.1 EXTI pin configuration

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.



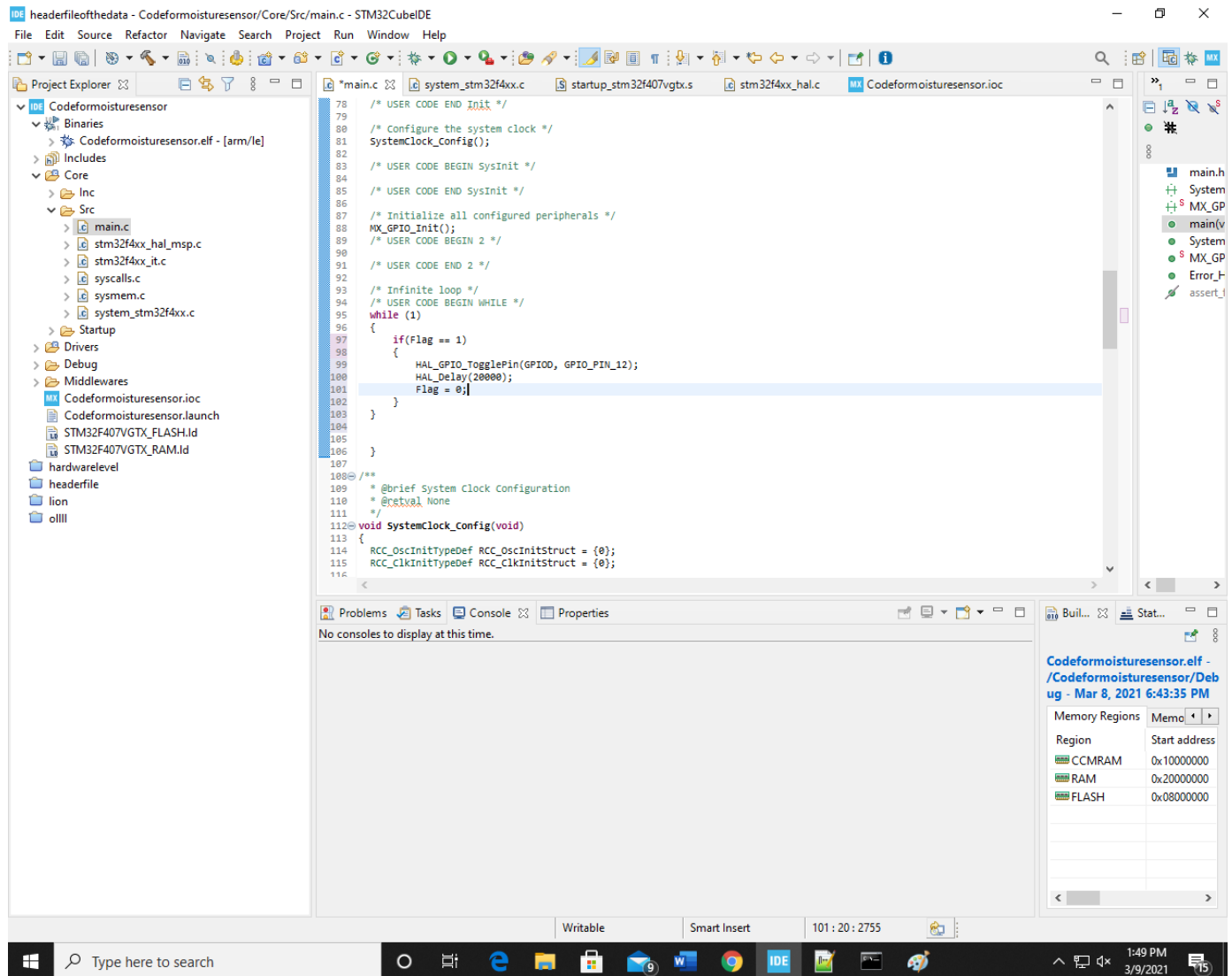


Fig: 2.2.1 EXT1 configuration code

## 2.3 ADC

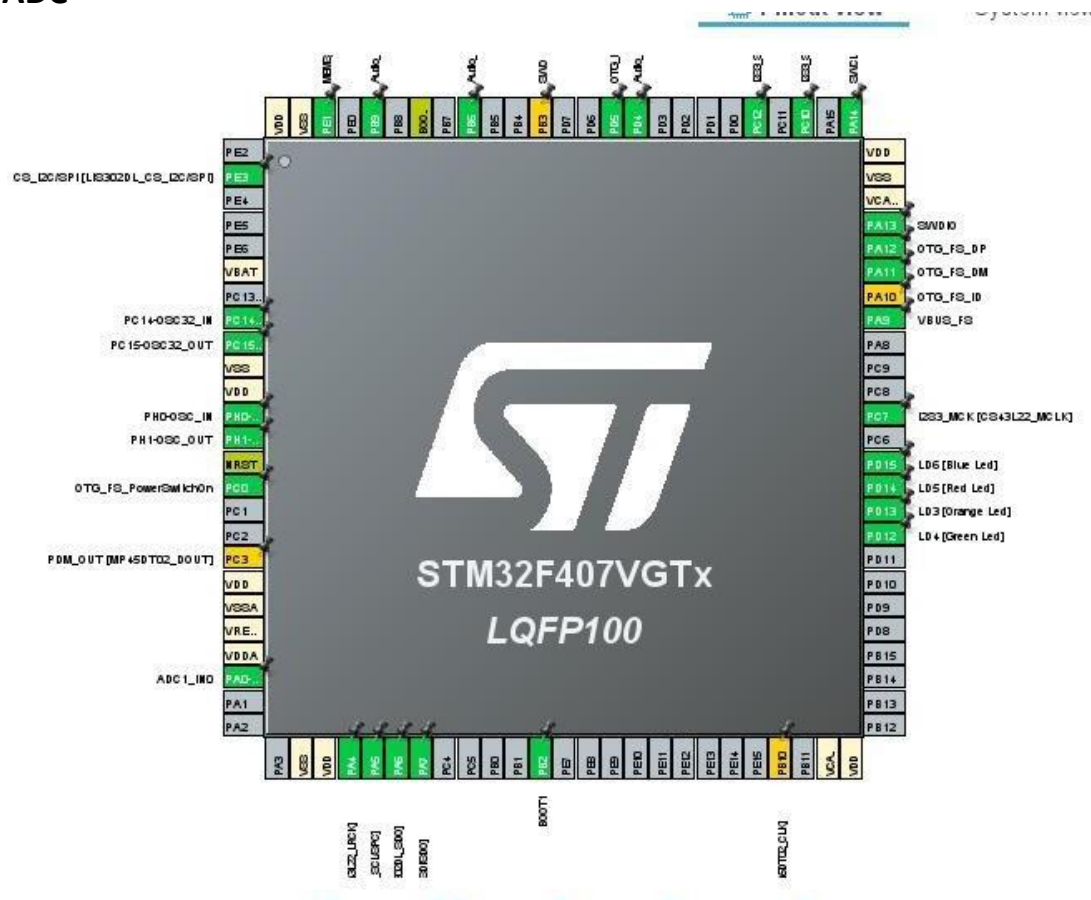


Fig: 2.3.1 ADC pin configuration

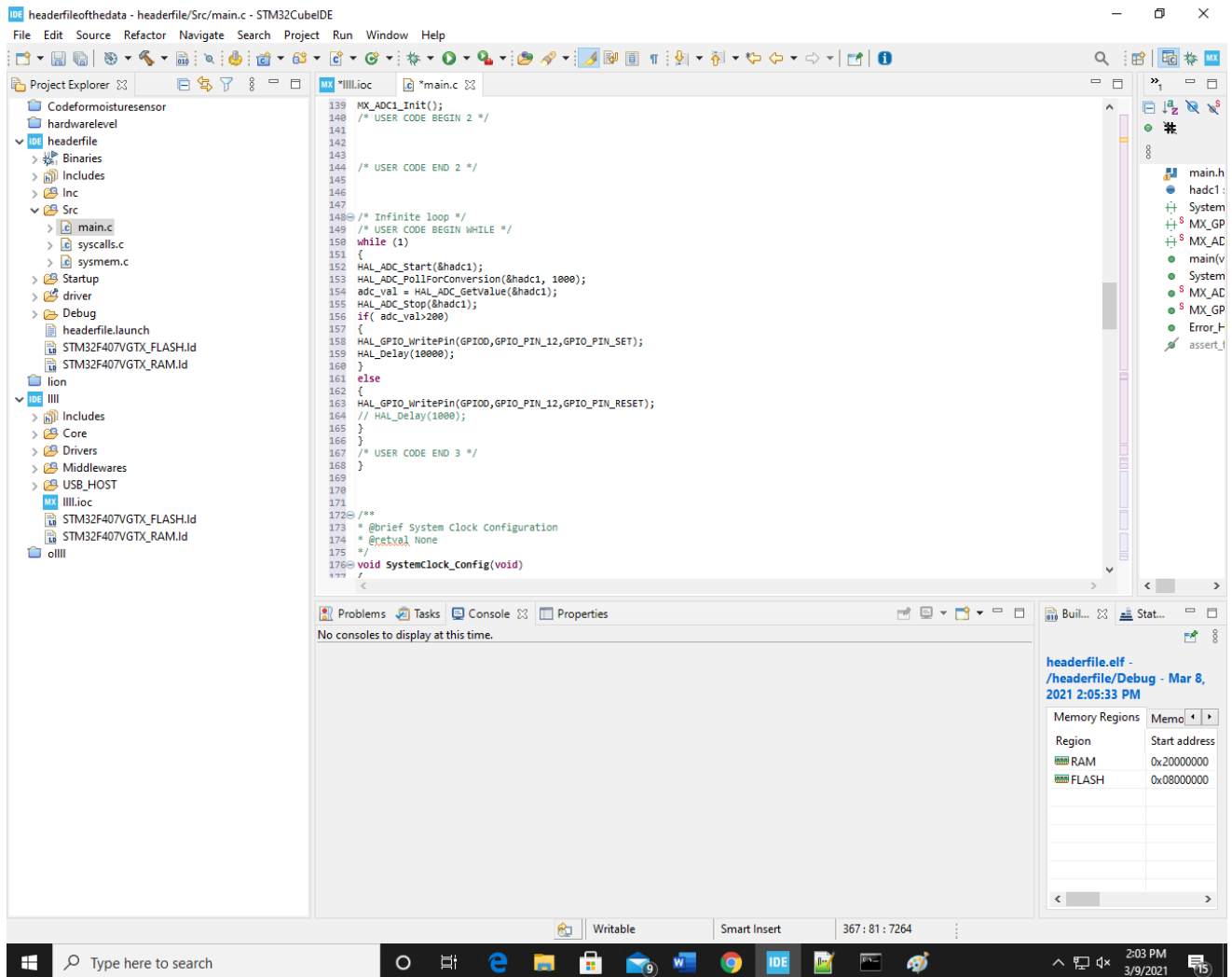


Fig: 2.3.2 ADC configuration code

## 2.4 SPI

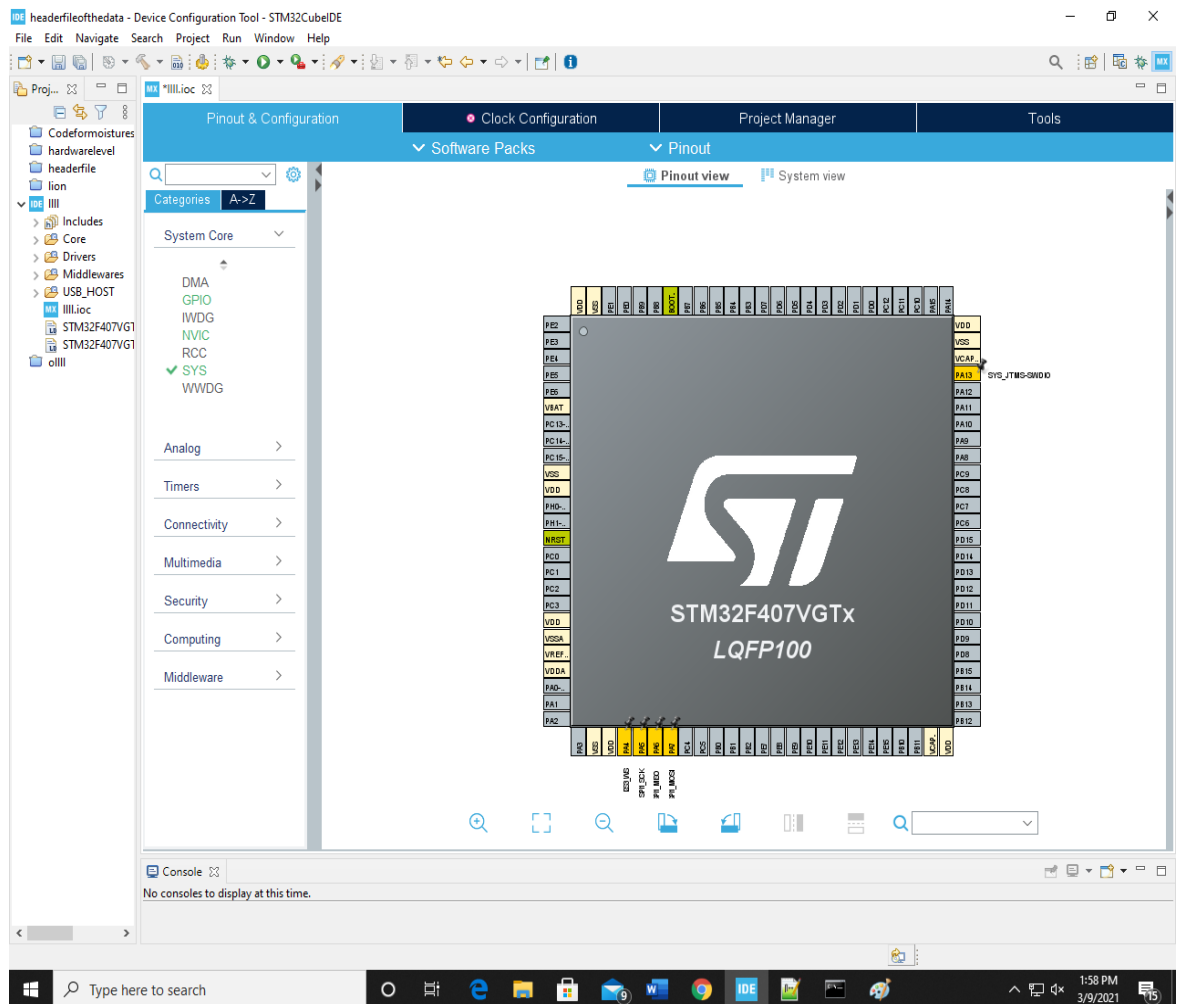
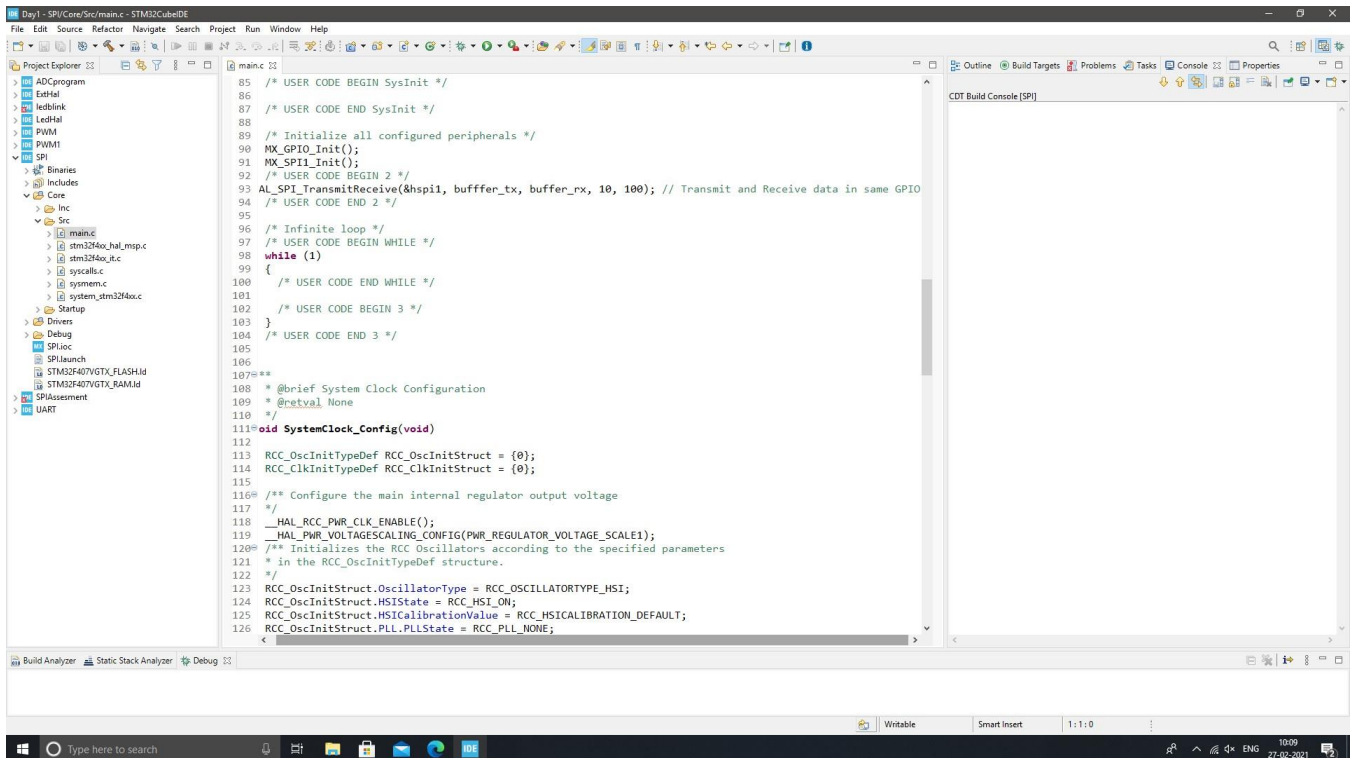


Fig: 2.4.1 SPI Pin configuration





The screenshot displays the STM32CubeIDE development environment. The Project Explorer on the left shows the project structure, including the 'main.c' file under the 'Src' folder. The main editor window shows the code for 'main.c', which includes the initialization of the SPI peripheral and the configuration of the system clock. The code is as follows:

```
85  /* USER CODE BEGIN SysInit */
86
87  /* USER CODE END SysInit */
88
89  /* Initialize all configured peripherals */
90  MX_GPIO_Init();
91  MX_SPI1_Init();
92  /* USER CODE BEGIN 2 */
93  AL_SPI_TransmitReceive(&hspi1, buffer_tx, buffer_rx, 10, 100); // Transmit and Receive data in same GPIO
94  /* USER CODE END 2 */
95
96  /* Infinite loop */
97  /* USER CODE BEGIN WHILE */
98  while (1)
99  {
100   /* USER CODE END WHILE */
101
102   /* USER CODE BEGIN 3 */
103   }
104   /* USER CODE END 3 */
105
106
107  /**
108   * @brief System Clock Configuration
109   * @retval None
110   */
111  void SystemClock_Config(void)
112  {
113    RCC_OscInitTypeDef RCC_OscInitStruct = {0};
114    RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
115
116    /** Configure the main internal regulator output voltage
117     */
118    __HAL_RCC_PWR_CLK_ENABLE();
119    __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);
120    /** Initializes the RCC Oscillators according to the specified parameters
121     * in the RCC_OscInitTypeDef structure.
122     */
123    RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
124    RCC_OscInitStruct.HSIState = RCC_HSI_ON;
125    RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
126    RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
```

Fig: 2.4.2 SPI configuration code

## 2.5 UART

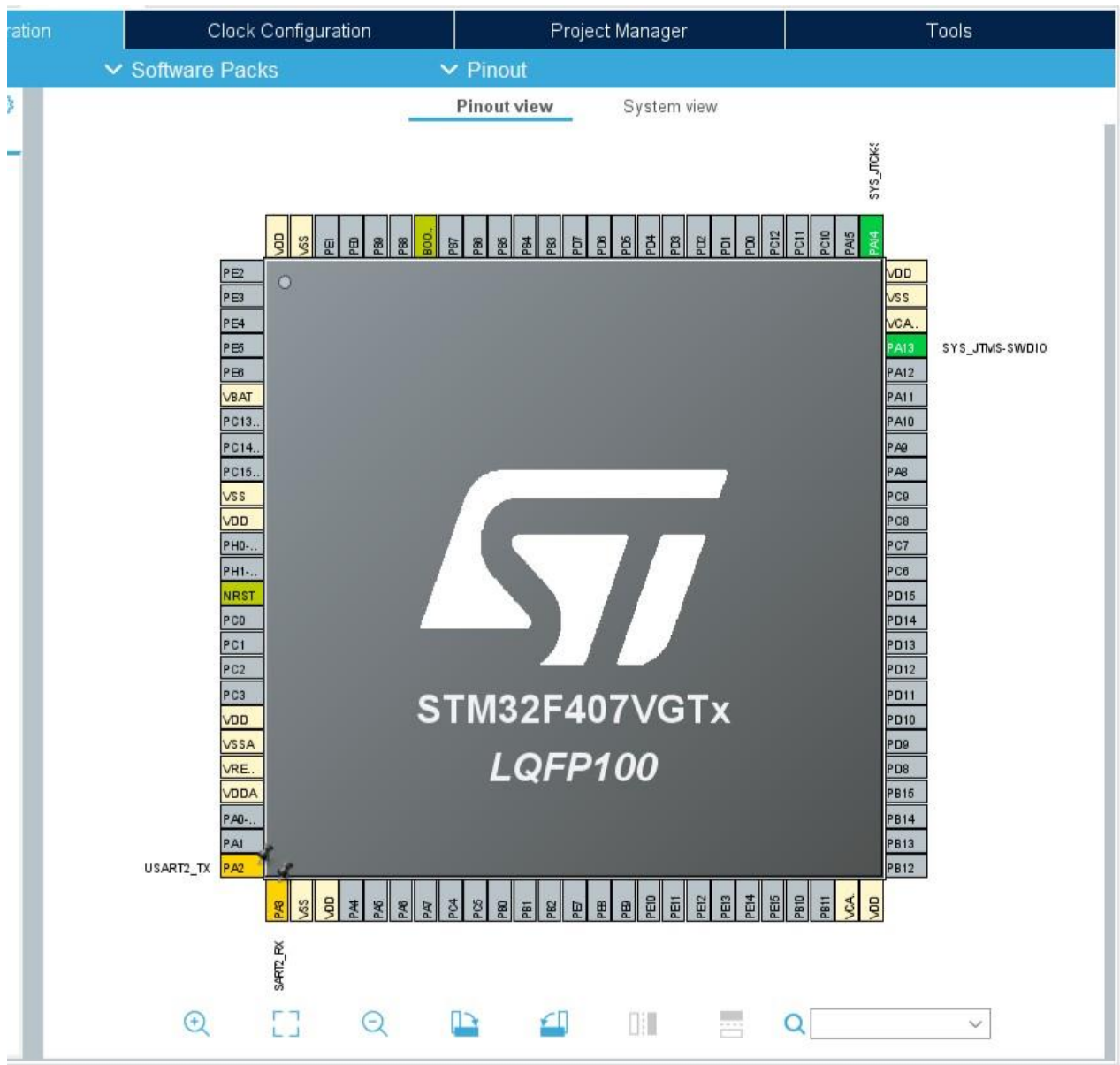


Fig: 2.5.1 UART Pin configuration

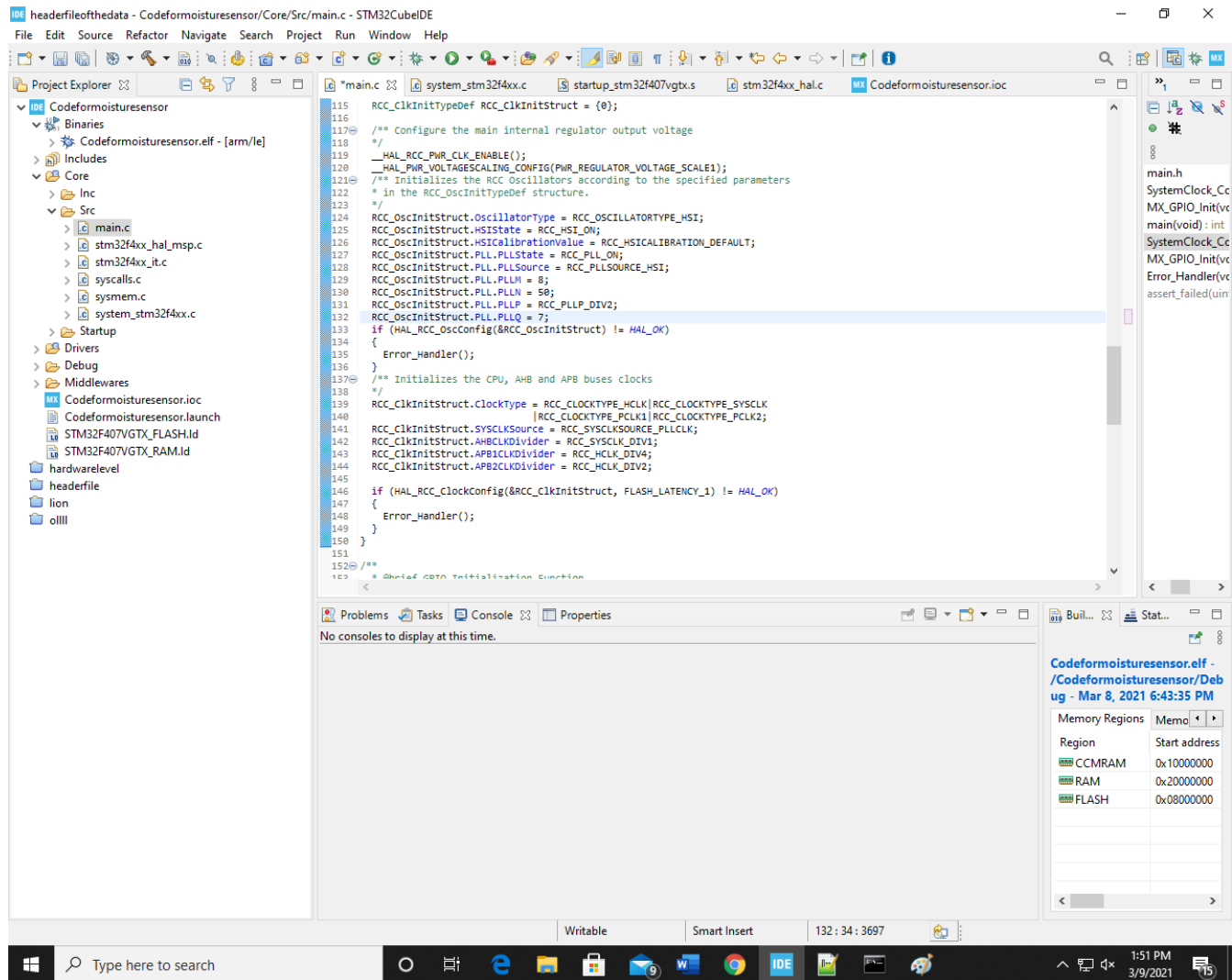


Fig: 2.5.2 UART configuration code