Version Number:

Team Members :

Team No:

Module: Model Based System Engineering



./



Learning Report

–

Embedded C: Hardware + Programming + Testing

(99003779)

**Document History**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ver. Rel. No.** | **Release Date** | **Prepared. By** | **Reviewed By** | **Approved By** | **Remarks/Revision Details** |
|  | 09/03/2021 | Neha Tabassum |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Contents**

Introduction to the software: STM32CUBEIDE

**ACTIVITY 1** – COMPILATION APPROACH…………………………………………………………………………………………………………….. 4

1.1- MAKE FILE ................................................................................................. 4

1.2- STARTUP CODE……………………………………………………………………………………………………………………… 5

1.3- LINKER FILE……………………………………………………………………………………………………………………………. 8 1.4- DEBUGGING TECHNIQUES……………………………………………………………………………………………………………… 9

**ACTIVITY 2** – IMPLEMENTATION OF PROTOCOLS USING STM IDE ........................................................................ 10

2.1 GPIO……………………………………………………………………………………………………………………………………………………10

2.2 EXTI……………………………………………………………………………………………………………………………………………………. .12 2.3 ADC……………………………………………………………………………………………………………………………………………………...14 2.4 SPI………………………………………………………………………………………………………………………………………………………..16 2.5 UART……………………………………………………………………………………………………………………………………………………19

**ACTIVITY 3** – PROJECT ON BCM MODULE .......................................................................................................... 21

**INTRODUCTION TO THE SOFTWARE : STM32CUBEIDE**

STM32CubeIDE is a multi-OS development tool, which is part of the STM32Cube system program.

STM32CubeIDE is a state-of-the-art C / C ++ development platform with boundary modification, code production, code integration, and debugging features for STM32 microcontrollers and microprocessors. Based on the Eclipse® / CDT framework and the GCC development tools, as well as the GDB debugging. Allows integration of hundreds of existing plugins that complete the Eclipse® IDE features.

STM32CubeIDE combines STM32 configuration and project creation functionality from STM32CubeMX to provide the same tools experience and save installation time and development time. After the selection of the STM32 MCU or blank MPU, or a customized microcontroller or microprocessor from the board selection or model selection, the project is created and a code is generated. At any time during development, the user can revert to implementation and configuration or middleware and update the startup code without affecting the user code.

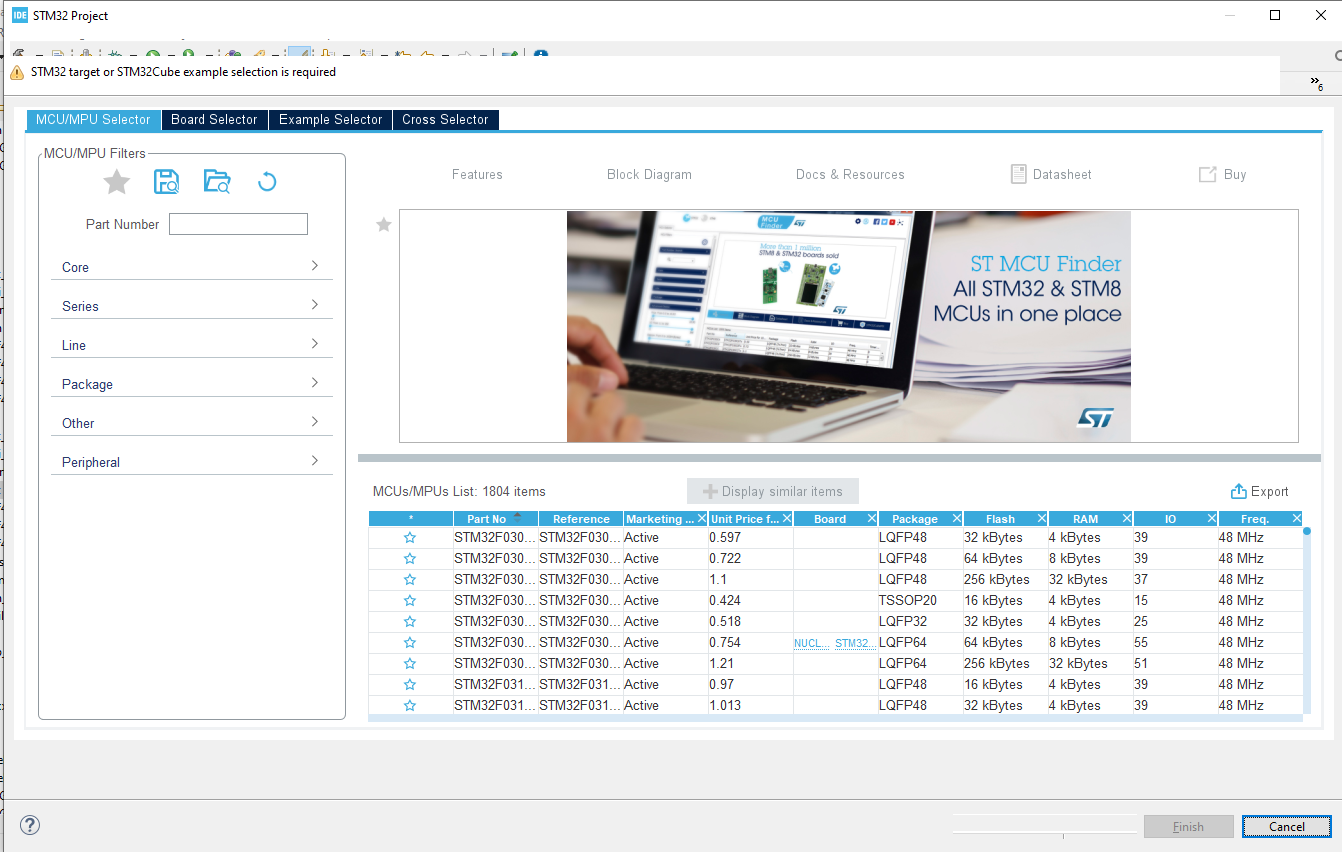
STM32CubeIDE includes build and layout analysts that provide the user with useful information about project status and memory requirements.

STM32CubeIDE also includes advanced and advanced debugging features including basic CPU registers, memory, and road-related registers, as well as a flexible live clock, Serial Wire Viewer interface, or error analyzer.

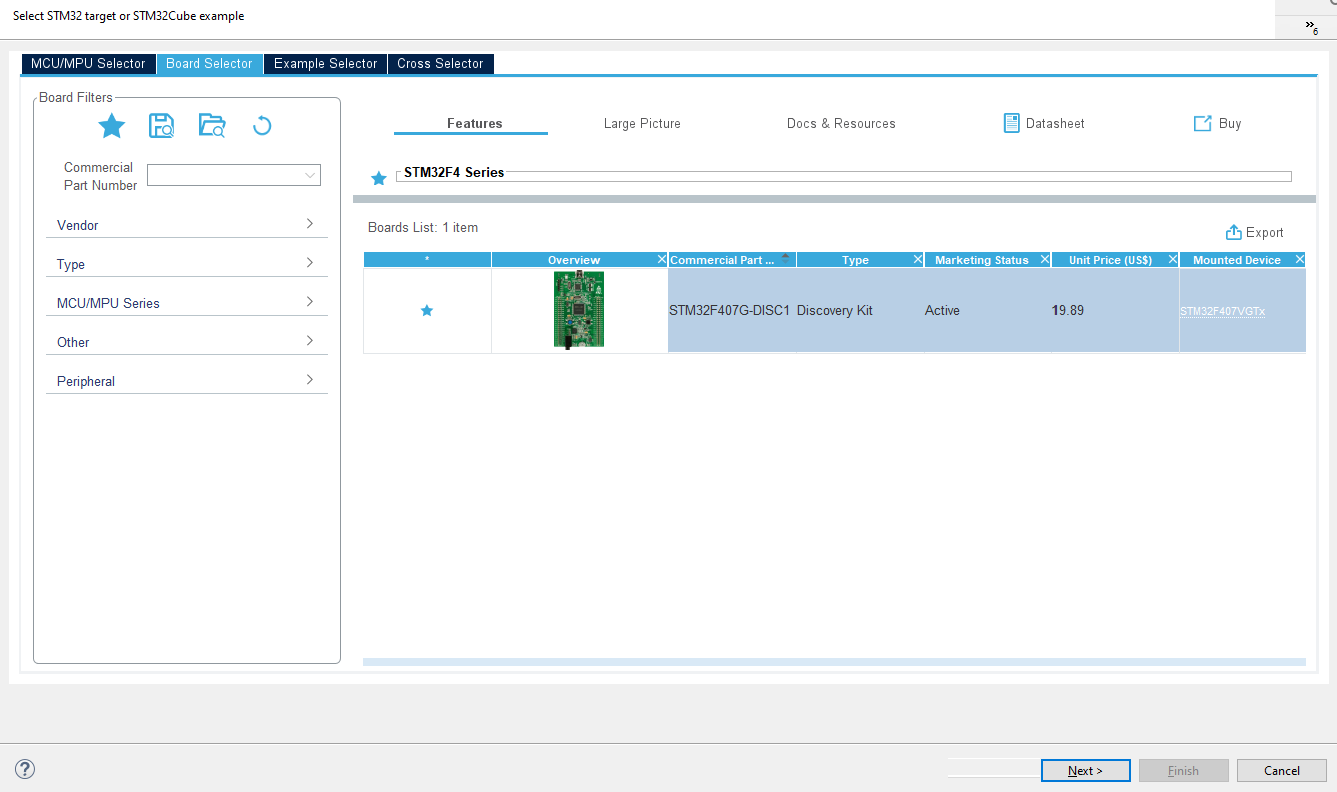
**GETTING STARTED WITH THE SOFTWARE:**

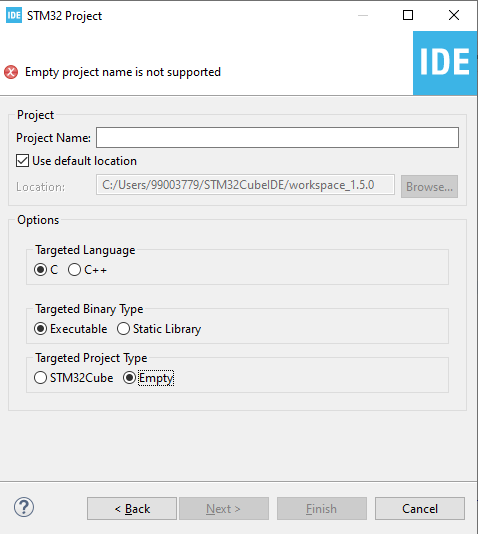
Start the software:

1. Goto file 🡪 New🡪 STM32 Project



1. Select the board and star it for frequent uses and start with the project .





Type the name of the project and start with the coding.

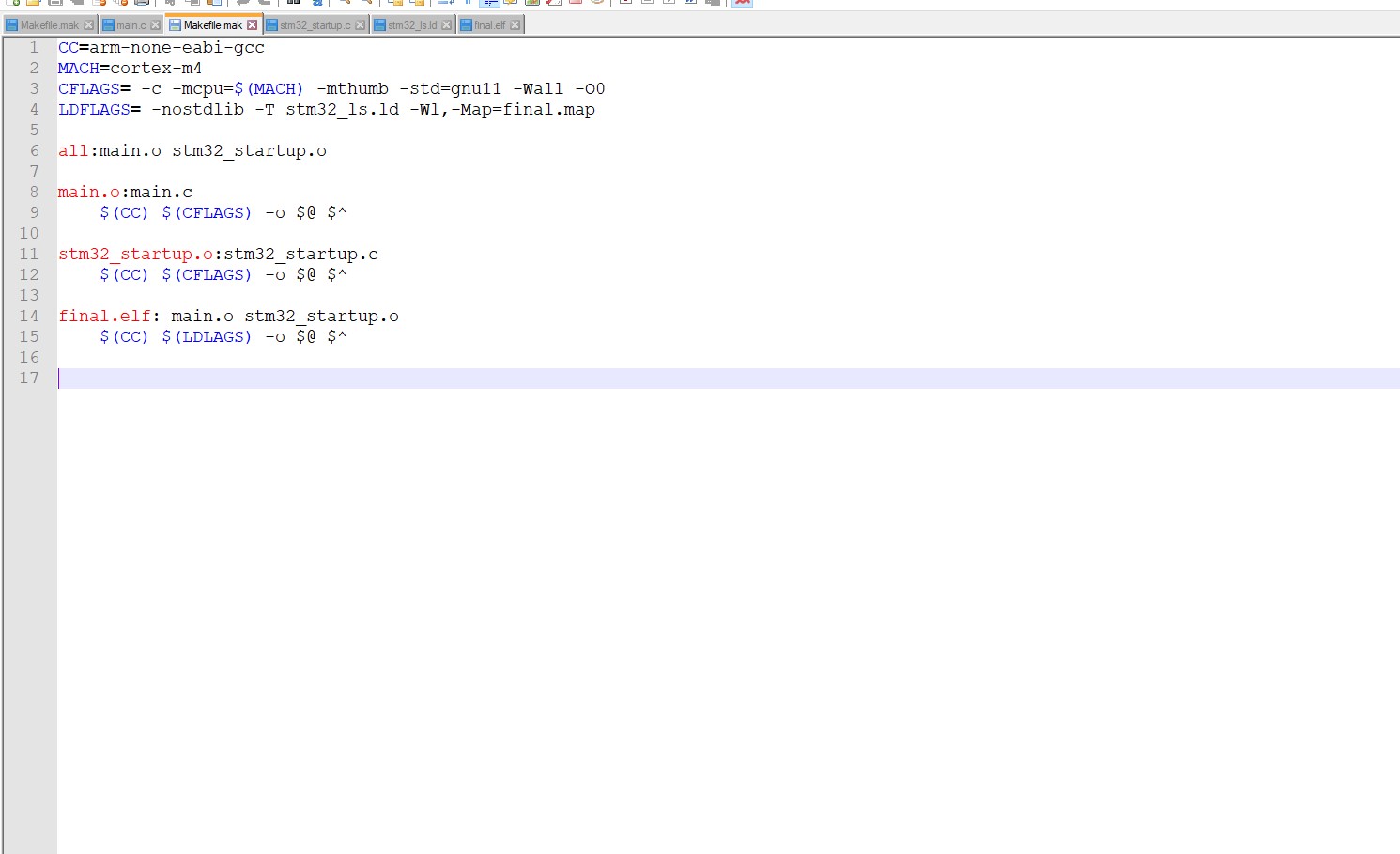
# Activity 1 – COMPILATION APROACH

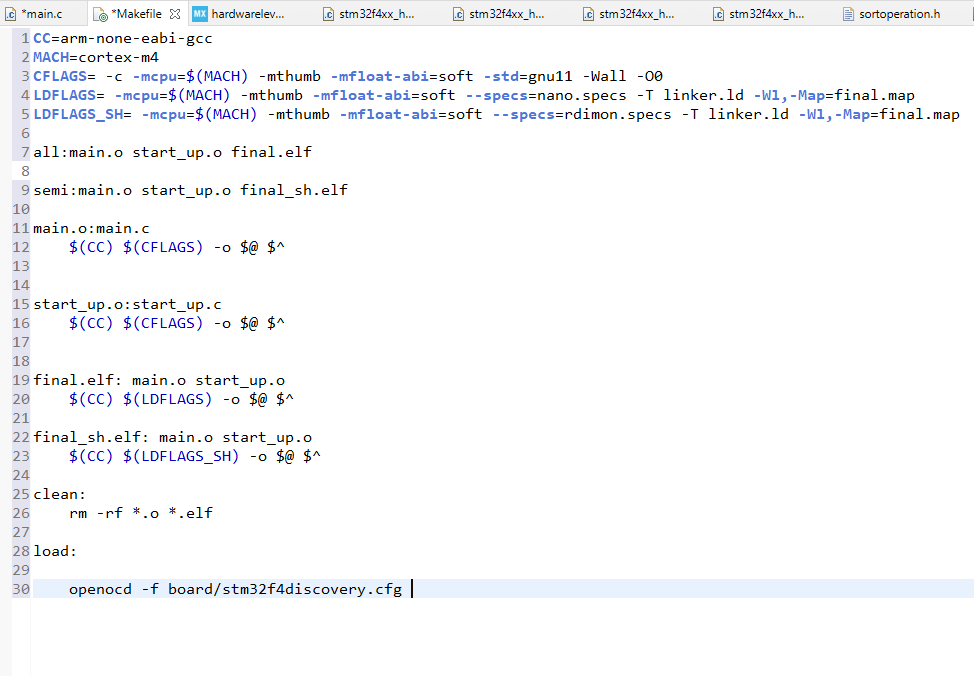
This is a complete process for integrating the ARM Cortex Mx processor sample system. The following are the integration stages of plan C:

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

**1.1- MAKE FILE**

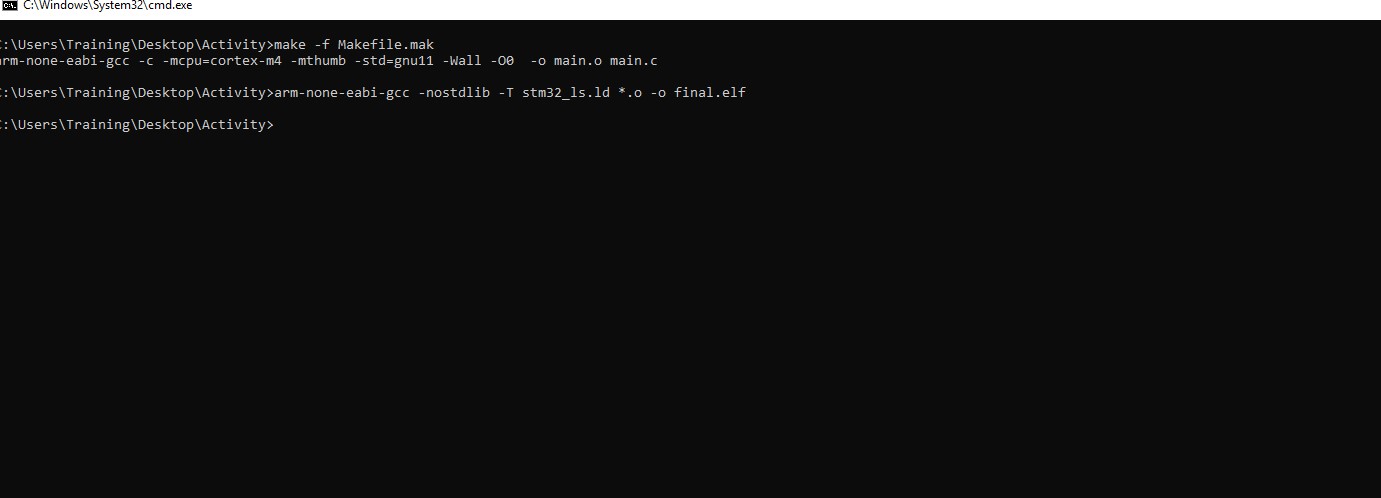
Below is the make file for the sample program:





**Fig 1. Make file**

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



**Fig 2 Make command**

* -mcpu = cortex-m4 used to select our cortex-m4 processor used
* -mthumb is used to generate output code in ARM state
* main.o is a redirect file
* main.c dependence

## 1.2- STARTUP CODE

• The start up file is responsible for setting the correct code usage fields in the main.c file.

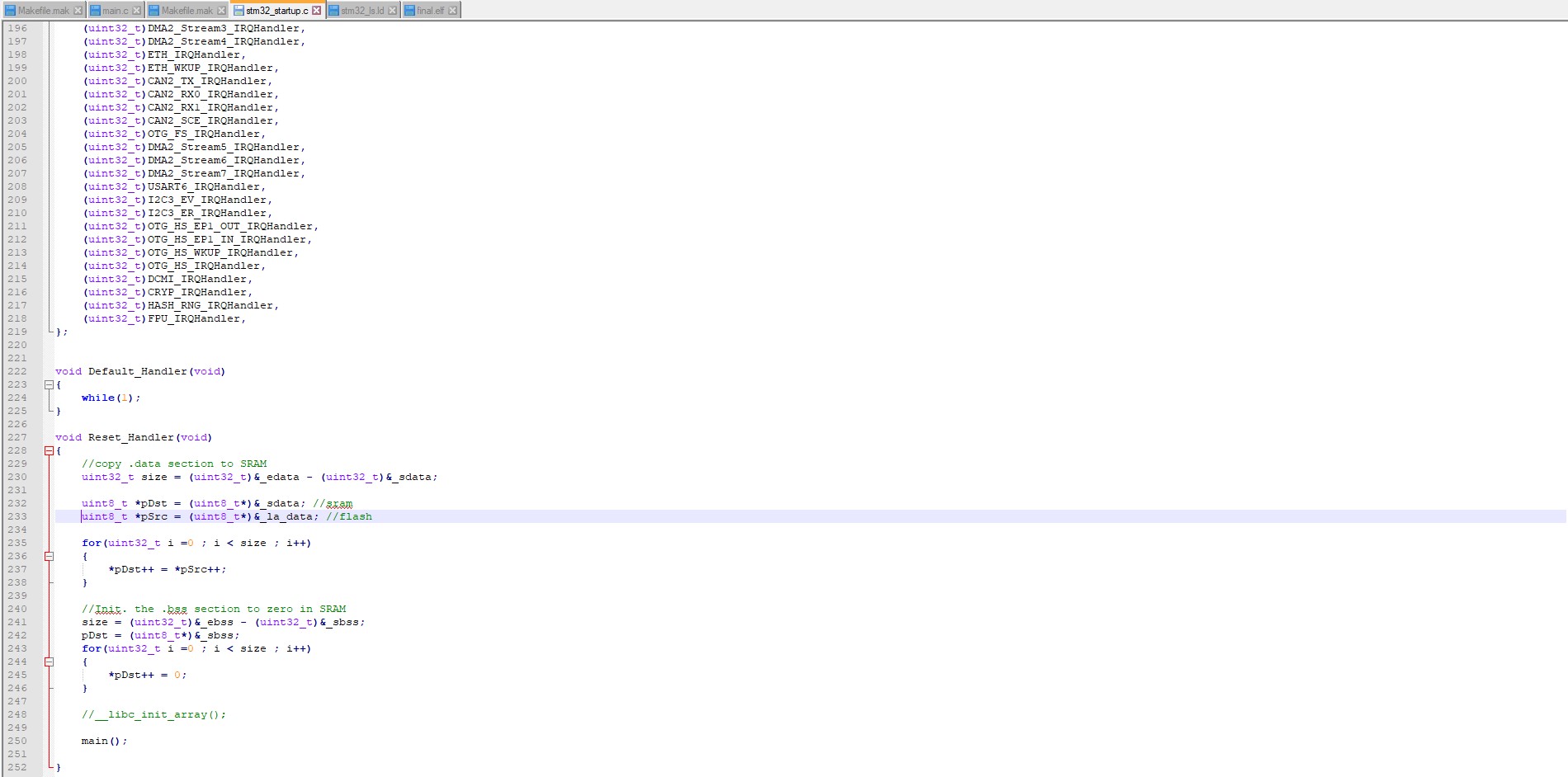
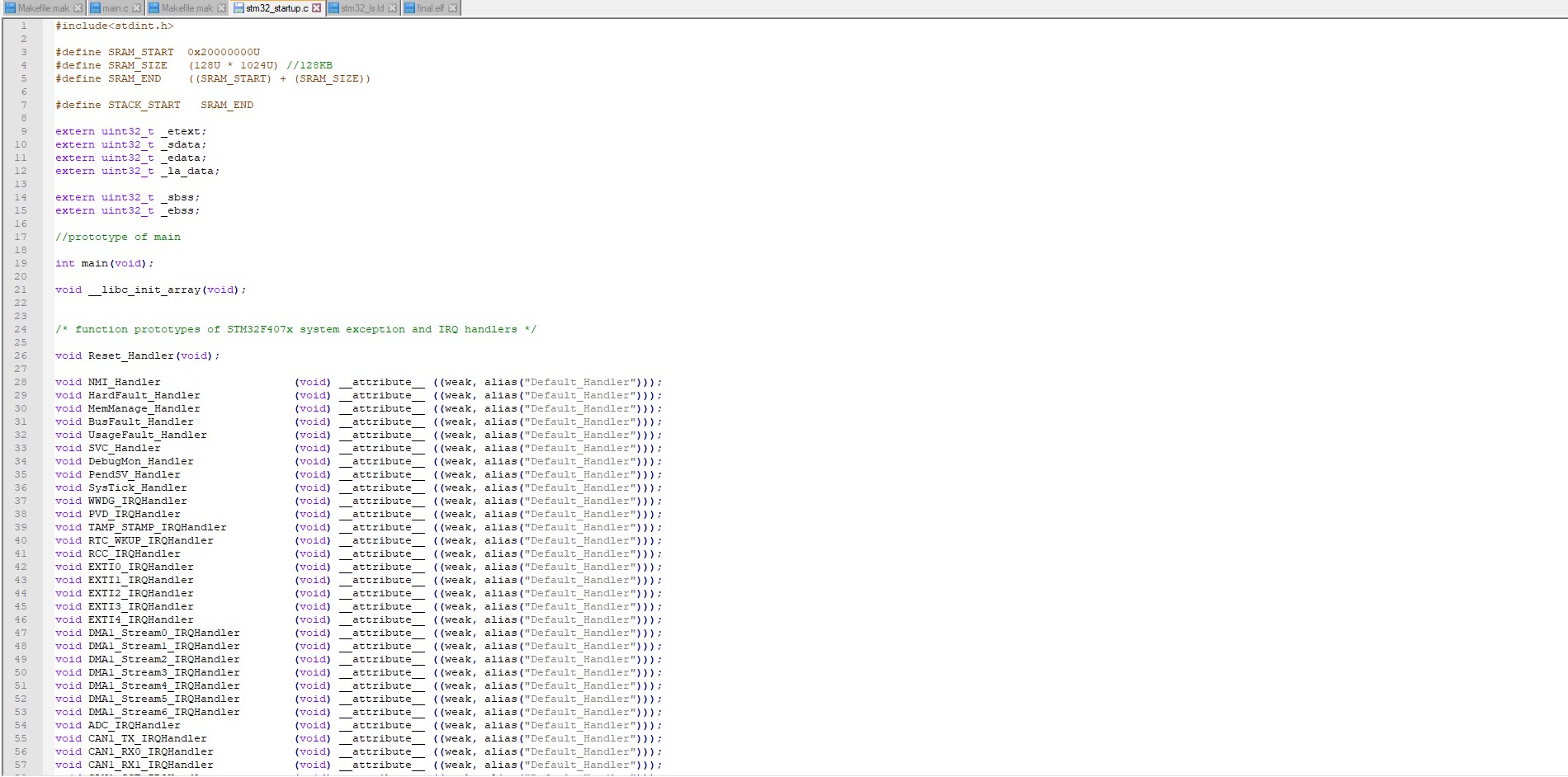
• Part of the start code depends on the direction (processed)

• Role of startup file:

1. Create a special MCU microcontroller vector table.

2. Write the startup code that starts the data section and .bss in SRAM.

3. Call the main ()



**Fig 3. Startup code**

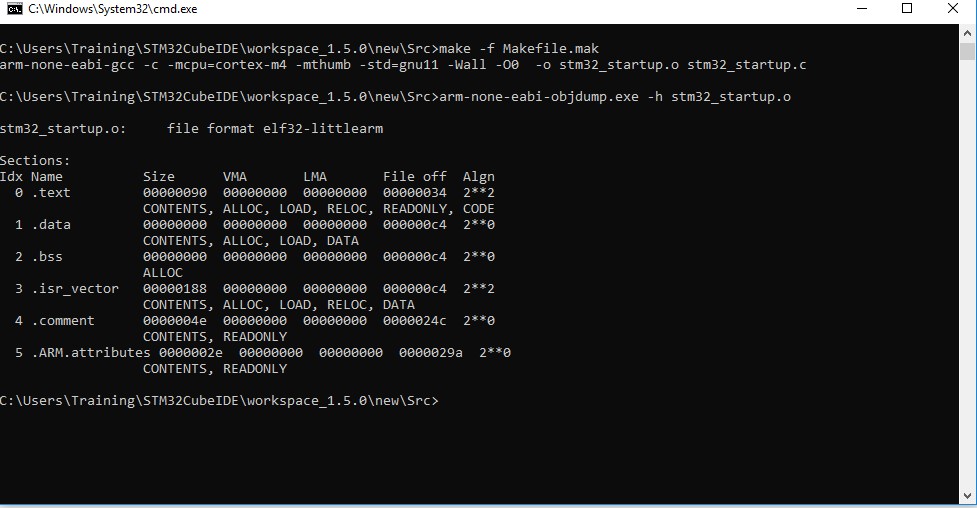
In the startup code we use the attributes to keep the variable in the user-defined function.

Job qualifications:

• Weakness: Allows the app to override a previously defined weak function (dummy function) with the same function name.

• Alias: Lets programmer provides any alias name for the same function.

A startup.o file made in usable format, the various components of which are shown below:



**Fig 4: Startup command**

## 1.3- LINKER SCRIPT

• Linkers take one or more object files or libraries as input and combine them to create a single usable file as output.

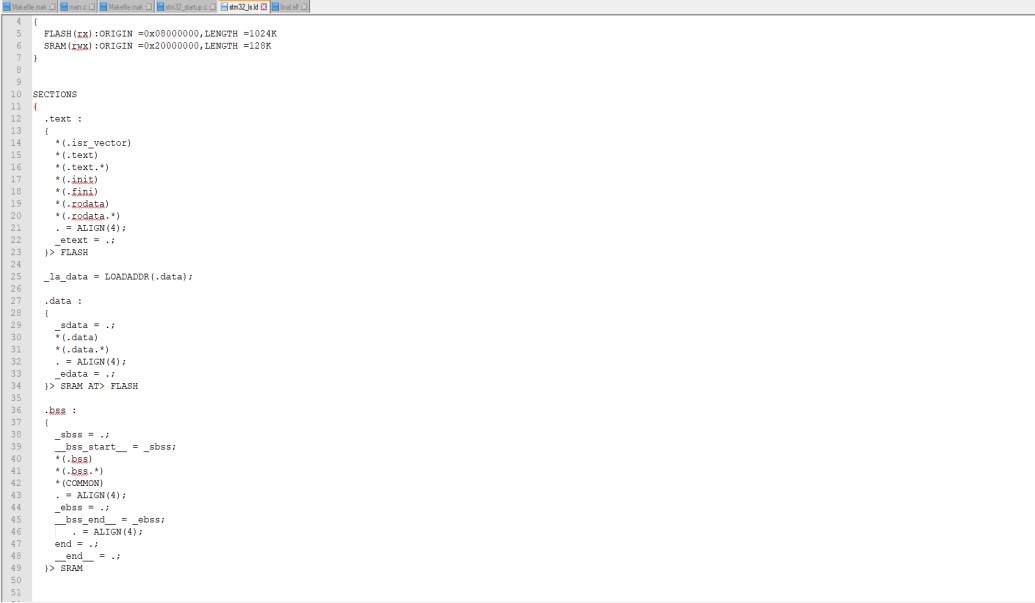
• Linker scripts determine how different categories of object files should be compiled to create an output file.

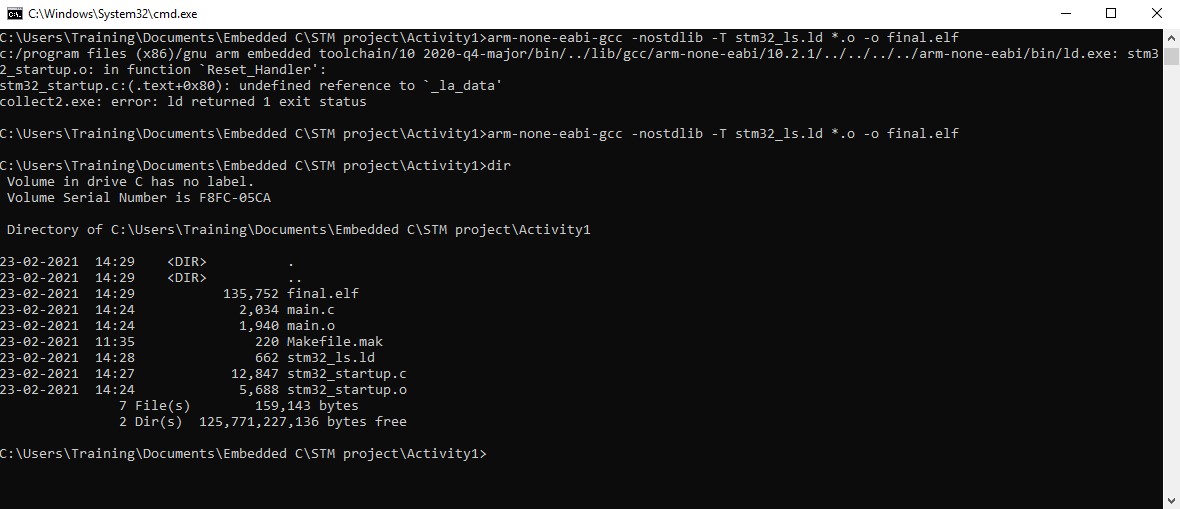
• Reset holder is the login of the application

• The login command is used to set the "login address address" information in the last elf file file created.

Syntax: Login (brand\_sign)

Login (Reset\_Manager)





**Fig 5: command to generate final.elf file**

## 1.4- DEBUGGING TECHNIQUES

• STM32F407VG embedded in chip debugger to fix code error.

• OCD ON-Chip Debugger aims to provide bug fixes, program settings and test parameters for embedded targeted devices.

• OCD is a free and resource-based application that allows you to edit, correct, and analyze your plans using GDB.

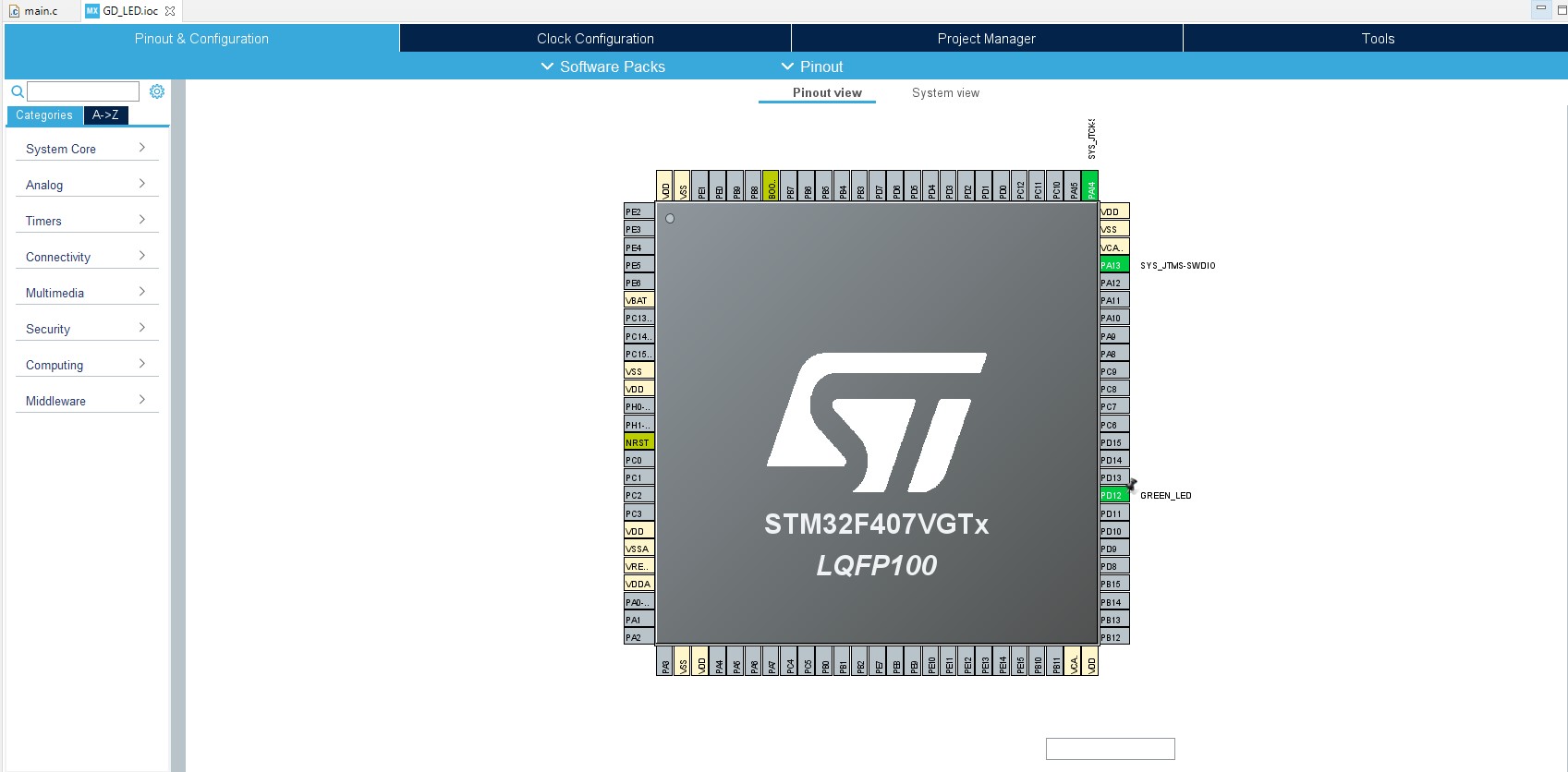
• Supports various target boards based on different processor properties.

# Activity 2 – IMPLEMENTATION OF PROTOCOLS USING STM IDE

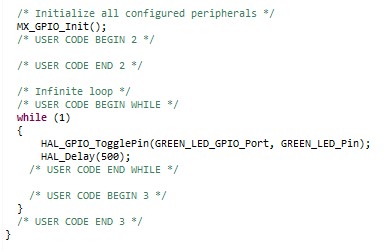
Protocol implementation for STM32F407VG microcontroller featuring ARM32 bit ARM – cortex M4 having FPU core using HAL Library

## 2.1 GPIO:

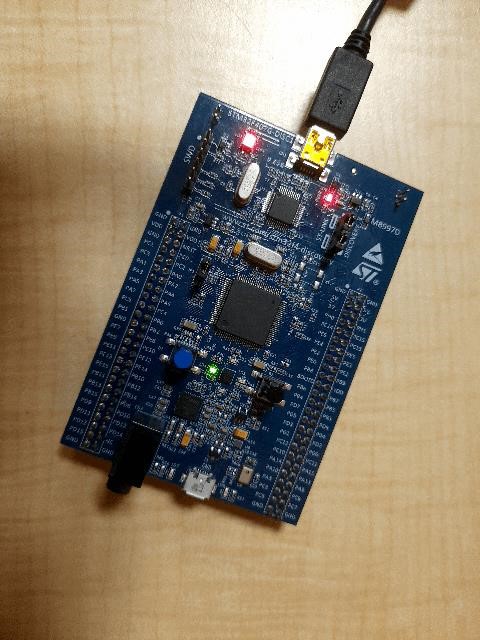
Toggling LED at pin PD12 at GREEN\_LED\_GPIO\_PORT. Serial wire is enabled at pin PA13.



**Fig: 6: GPIO pin configuration**



**Fig: 7 : GPIO configuration code**

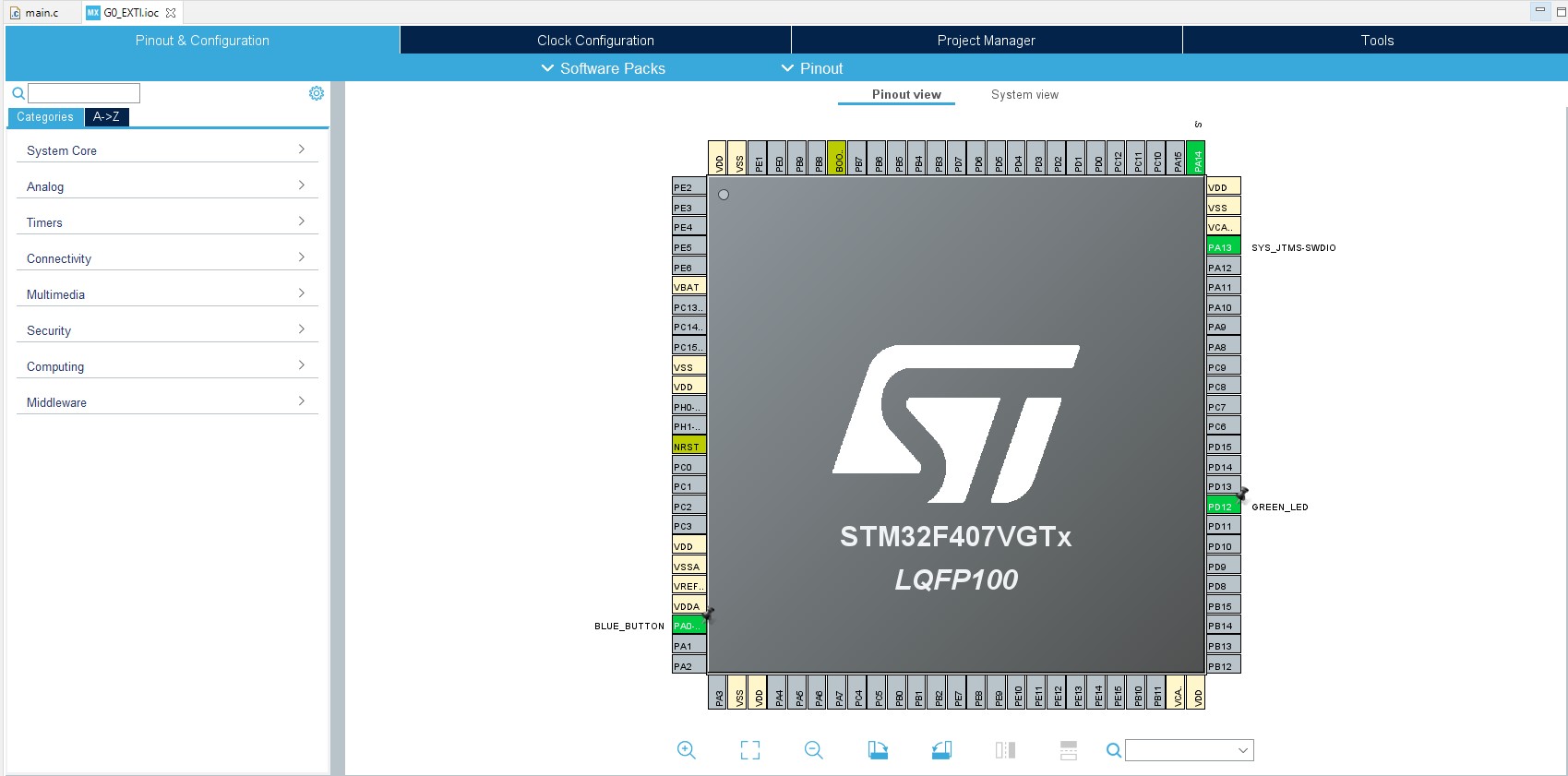


**Fig: 8: LED toggling**

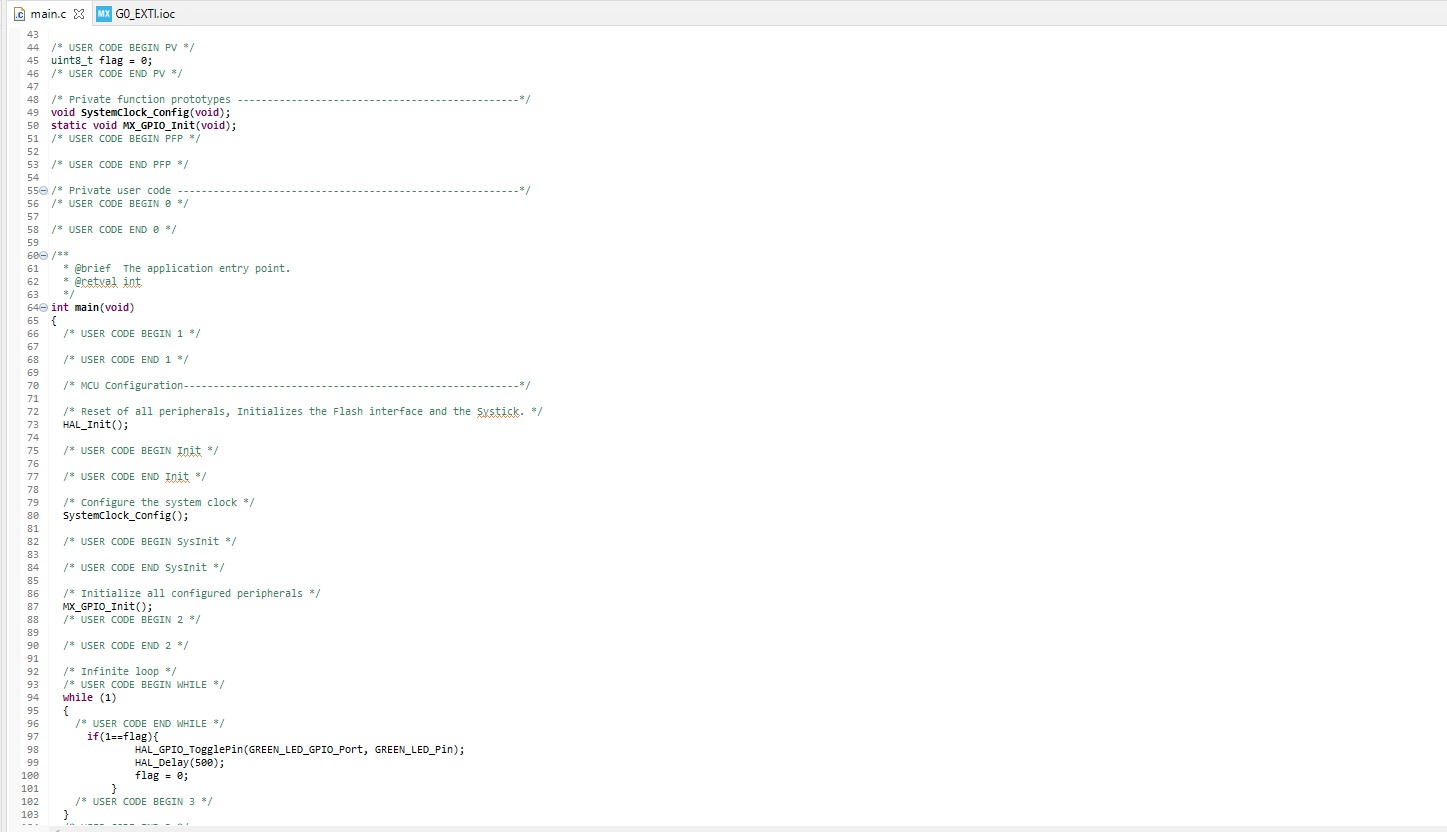
## 2.2 EXTI:

The blue button on the PA0 acts as an external distraction.

When the blue button is pressed Green LED on the PD12 pin.

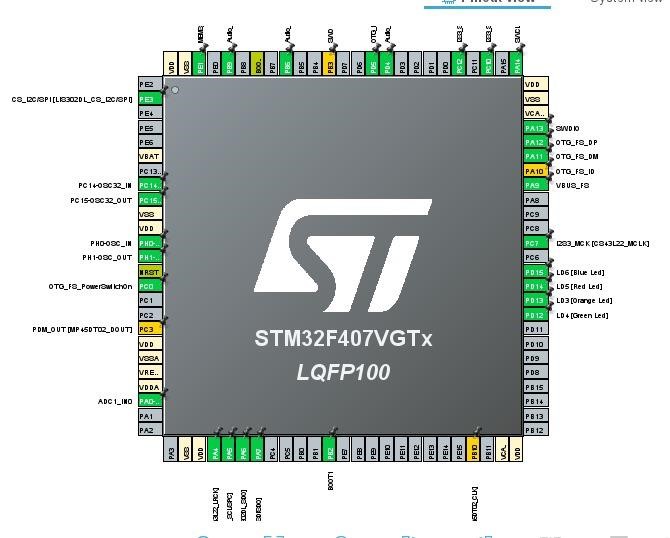


**Fig 9: EXTI pin configuration**

In the main.c file the flag is started and if the flag == 1, the status below the if loop is made to convert the LED to PD12.

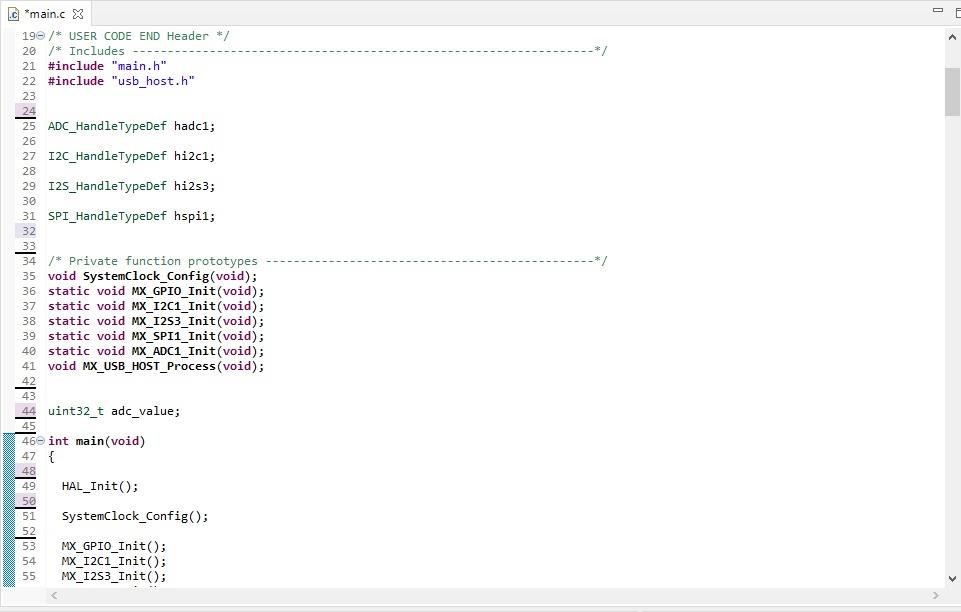
**Fig. 10: EXTI configuration code**

## 2.3 ADC

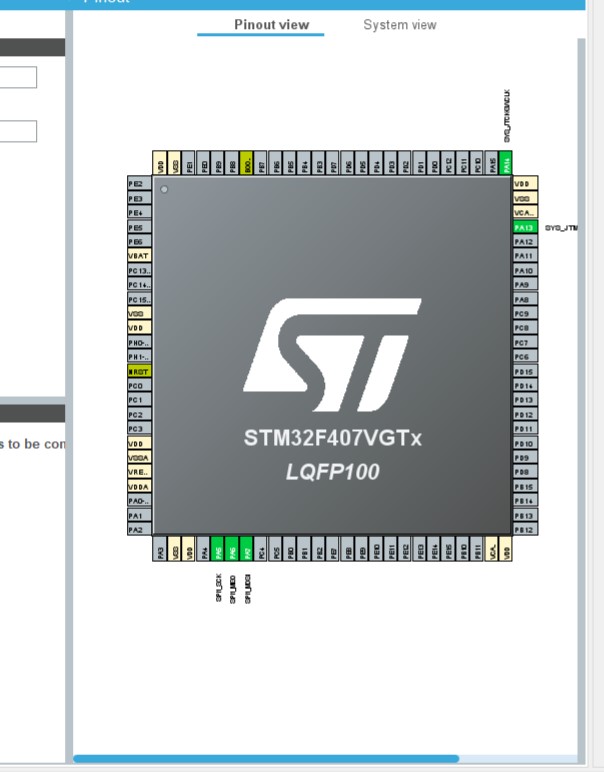


**Fig. 11: ADC pin configuration**

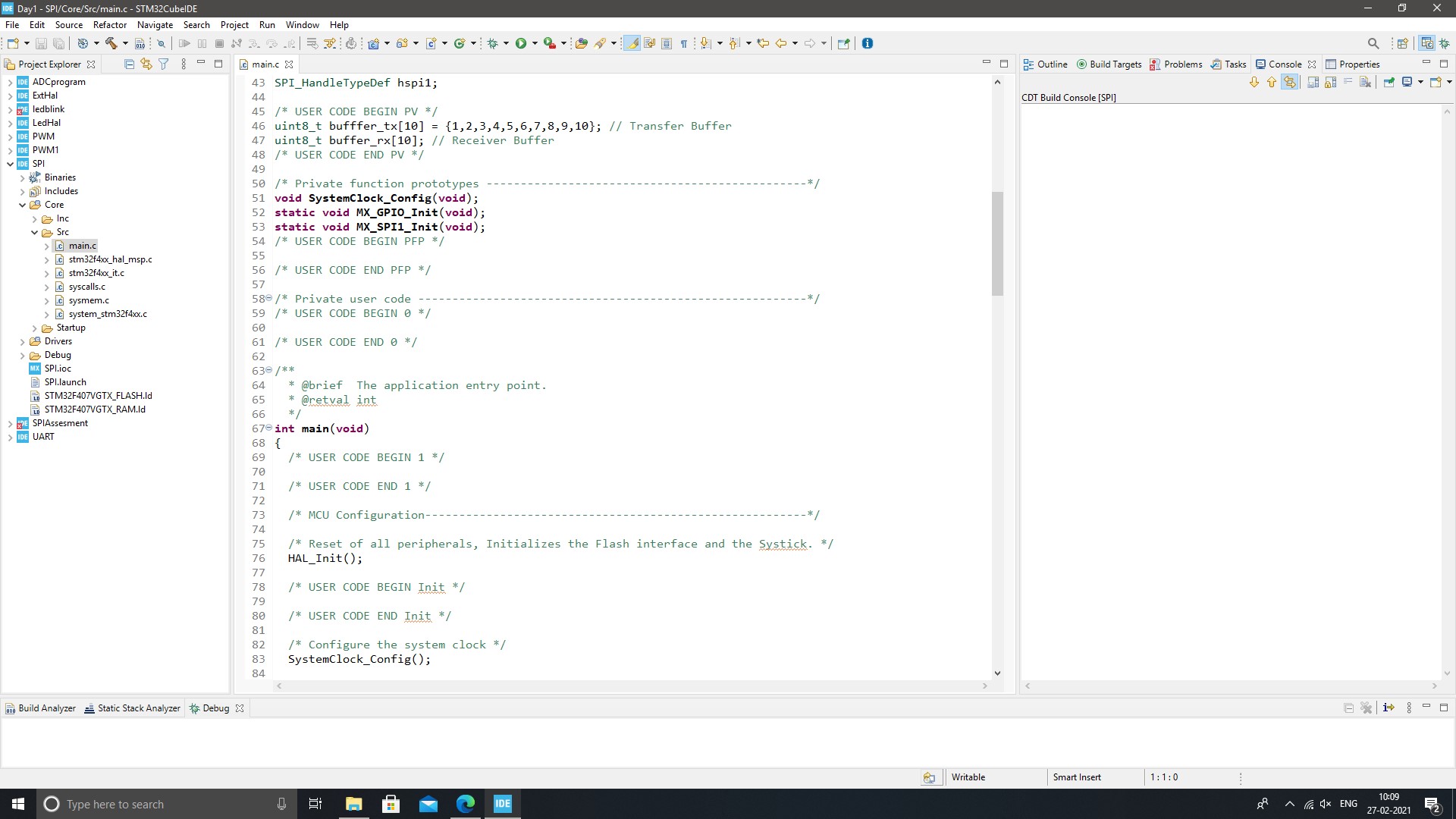
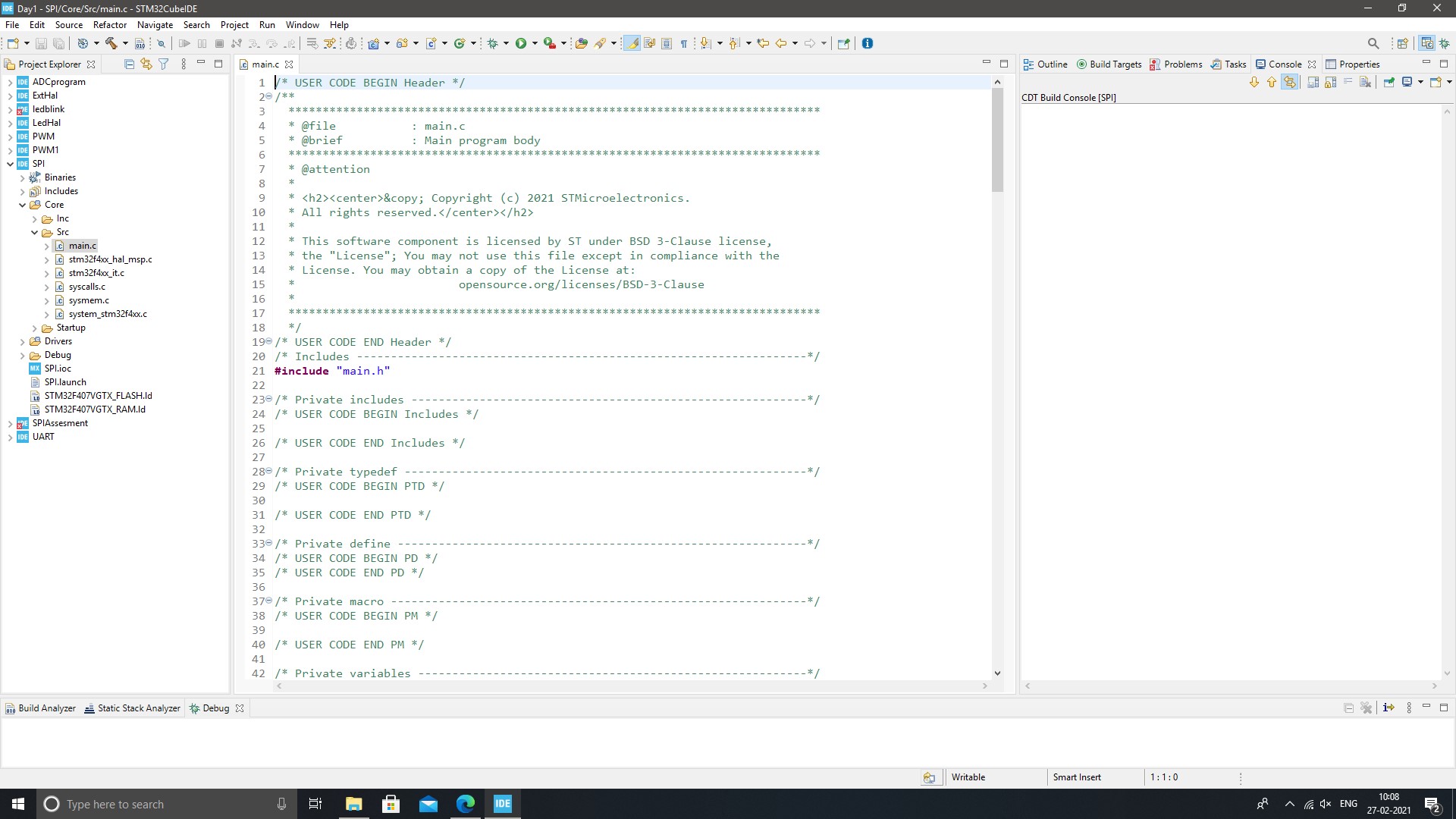
**Fig.12: configuration code**

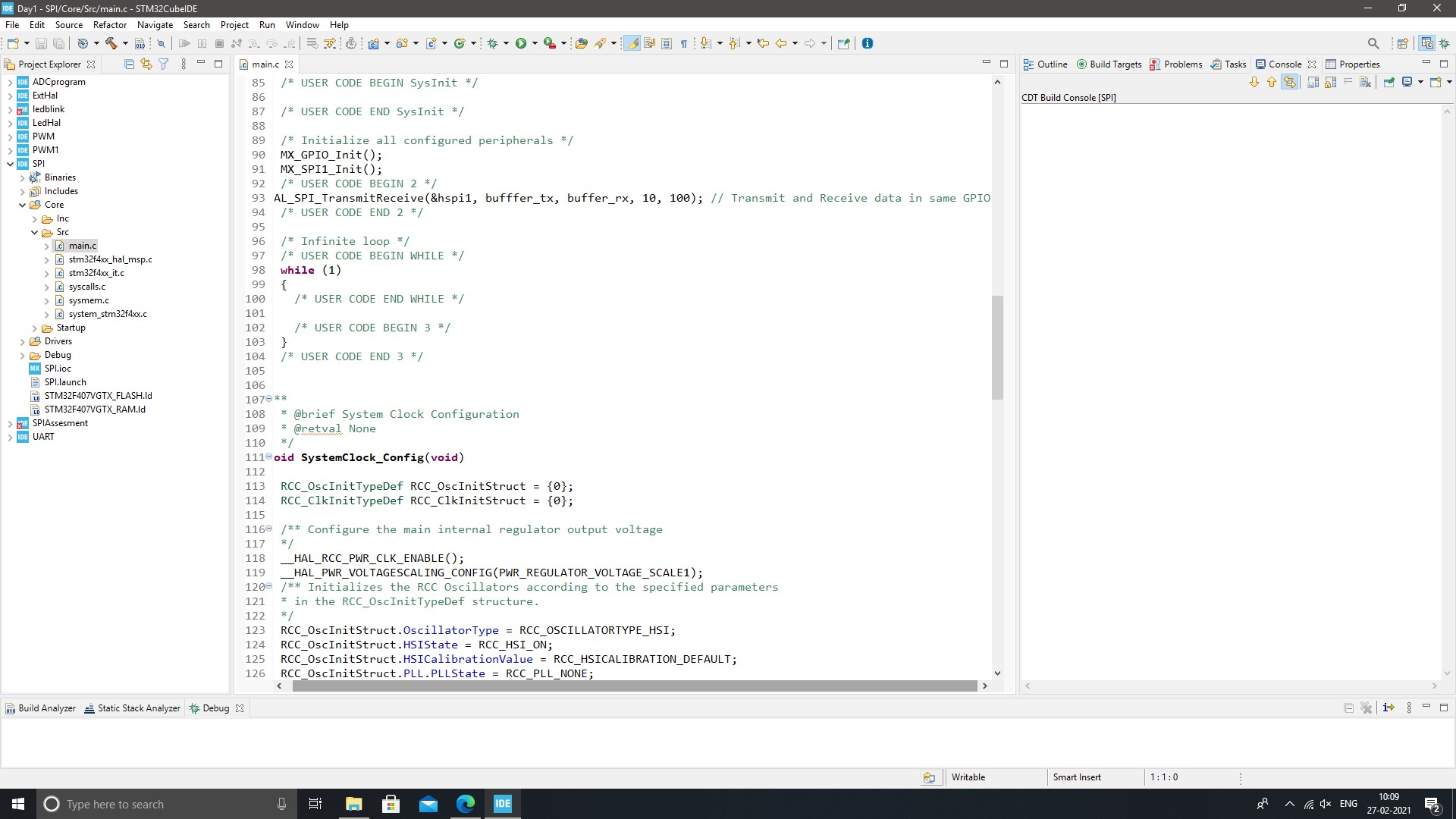


## 2.4 SPI



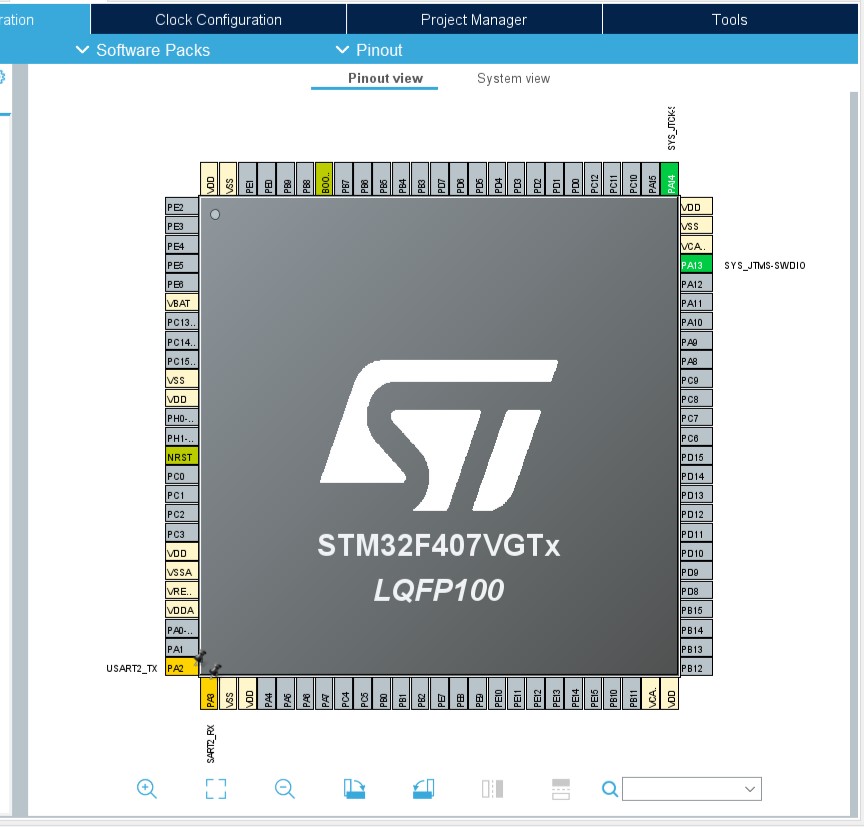
**Fig. 13: SPI Pin configuration**



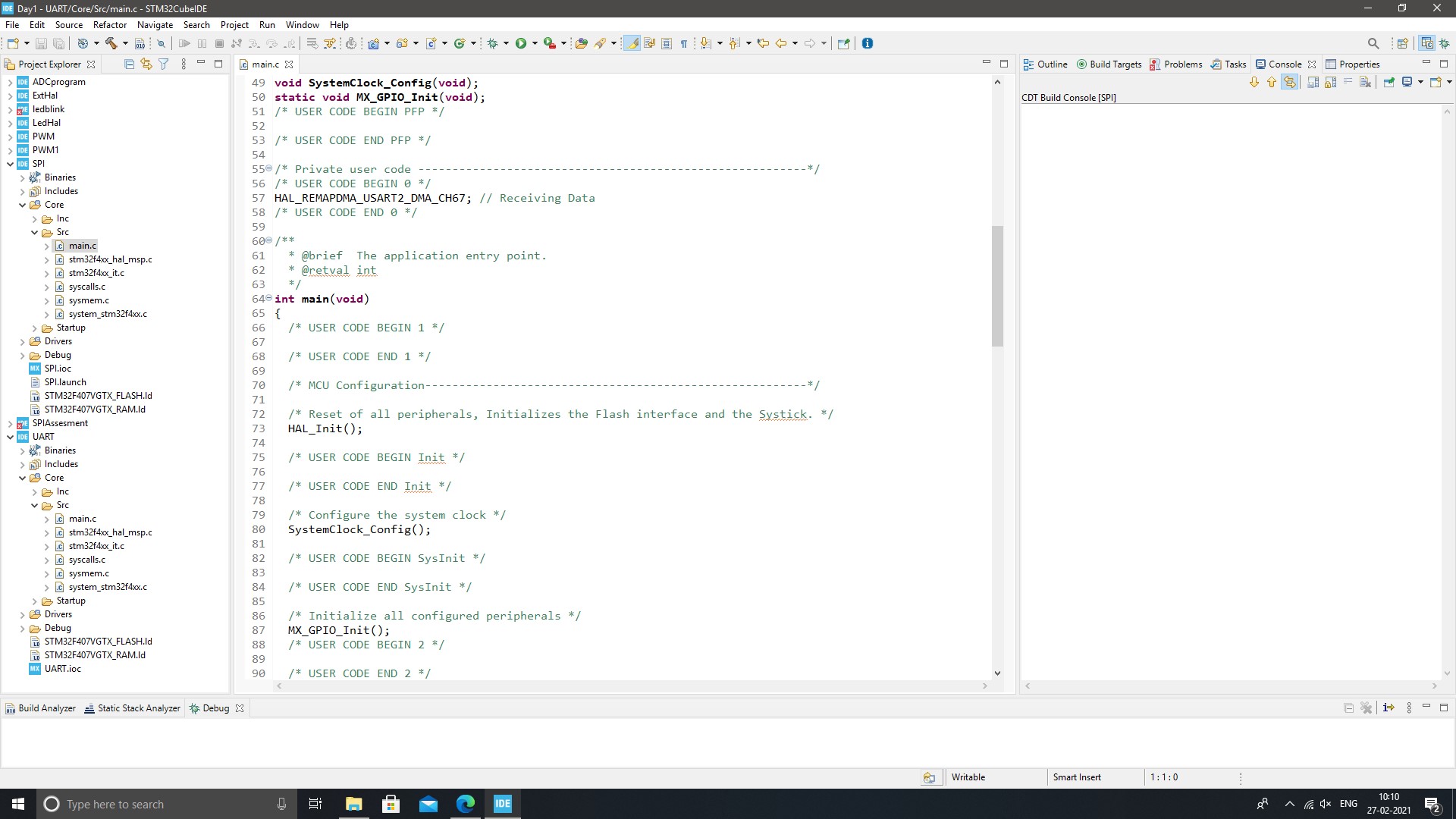
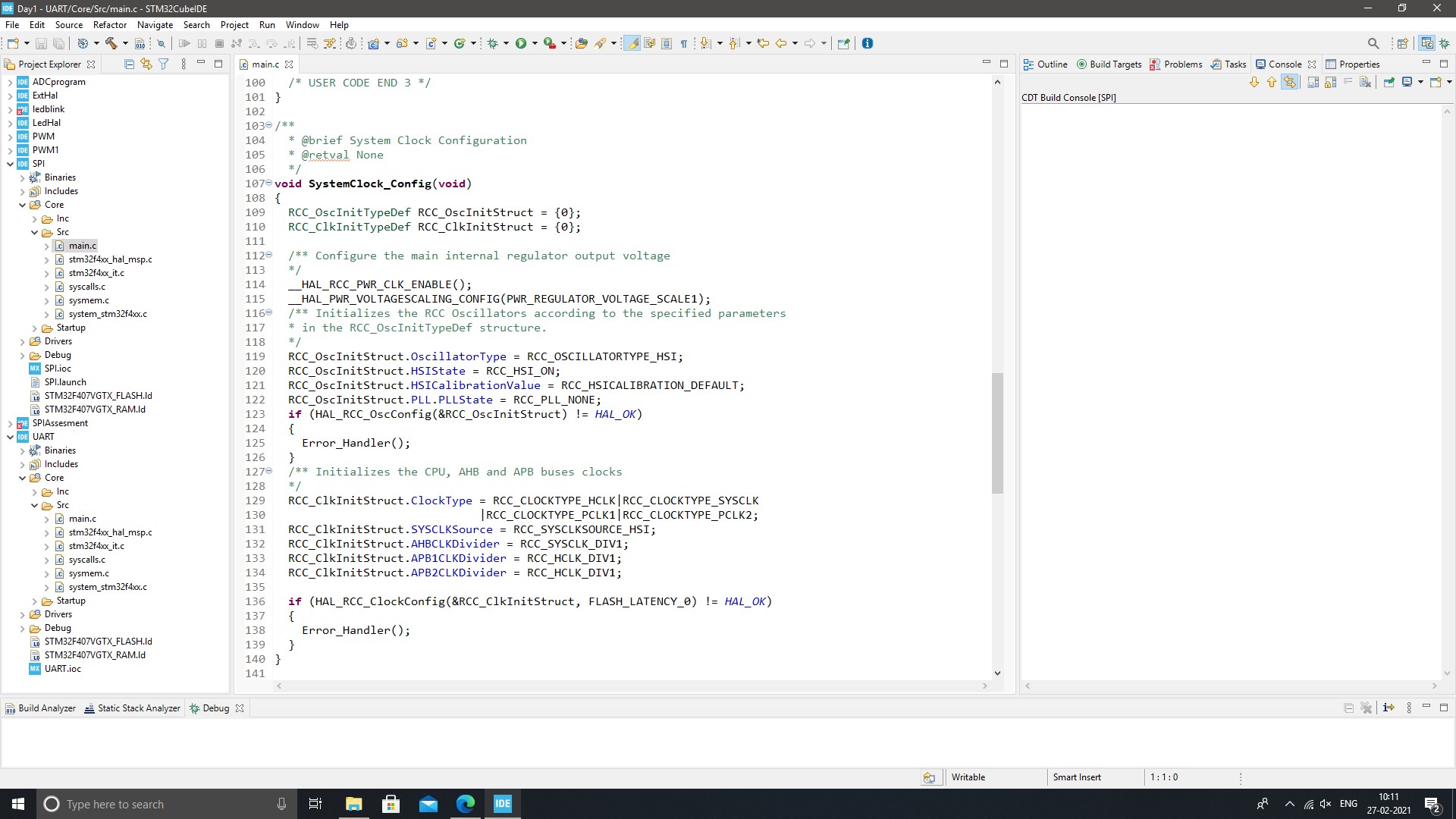


**Fig. 14: SPI configuration code**

## 2.5 UART



**Fig 15: UART Pin configuration**



**Fig. 16: UART configuration code**

**ACTIVITY 3: BODY CONTROL MODULE (BCM) :**

**Introduction**

* A control module is a computer component in a car that monitors, controls and uses electrical devices throughout the vehicle. When electric devices began to be used in cars and trucks, each tool was controlled by a different electrical component. There was a cooling module, an interior lighting module, a door lock module, and so on. The body control module combines all these different modules under a single system to work together instead of partitioning, a simple format for both production and problem solving. While there are many different types of body control modules, they are generally the same, the sensors for connecting, switching and automatic switching together in a single computer system. These components are subdivided into inputs, such as sensor data about temperature or speed, and the effects, or the way a computer control system responds to control motor performance. Inputs and outputs are further divided into analog and digital information types - analog signals used by modular continuous modules (such as oil pressure) and digital signals used for modules that can either turn on or off (such as headlights or oil indicator light).

**Features Implemented:**

In our project we have implemented six features on a microcontroller STM32f4 discovery board. In the project we integrated IR sensors, Alcohol sensor (mq7), LDR sensor, Moisture sensor, ignition sensor with microcontroller STM32f4 using GPIO protocol.

IR Sensor:-An IR sensor is an electronic device, which illuminates light to detect the surrounding environment. The IR sensor can measure the temperature of an object and detect movement. Usually, in the infrared spectrum, all substances emit a certain type of radiation. These types of radiation are invisible to our eyes, but the infrared sensor can detect these rays.

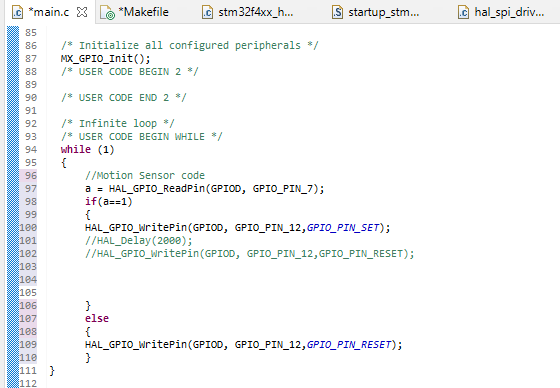
The alcohol sensor receives the attention of the liquor gas in the air and the analog voltage reading output. The sensor can operate at temperatures ranging from -10 to 50 ° C with an electrical power of less than 150 Ma to 5V. Sensitivity range ranges from 0.04 mg / L to 4 mg / L, suitable for breathalyzers.

LDR or light-resistant light resistor is also known as photo resistor, photocell, photoconductor. One type of opposition its resistance varies depending on the amount of light falling on its surface. When light falls on an opponent, the resistance changes.

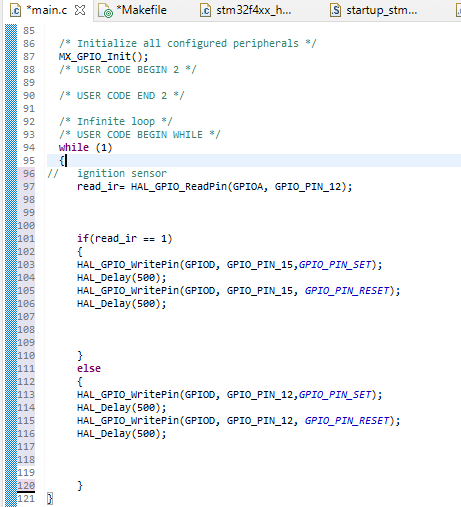
Moisture sensors measure the amount of water in the soil. Since direct gravimetric measurement of free soil moisture requires removal, drying, and weight of the sample, soil moisture sensors measure water content indirectly using other earth materials, such as electricity resistance, dielectric constant, or neutron contact, as representative of moisture content

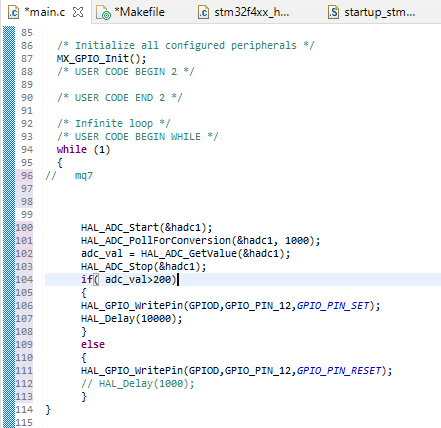
The ignition system produces sparks or burns the electrode at high temperatures to heat a mixture of gasoline gas in combustion engines fire engines, oil-fired boilers and gas, rocket engines, etc.

**Code Snaps:-**

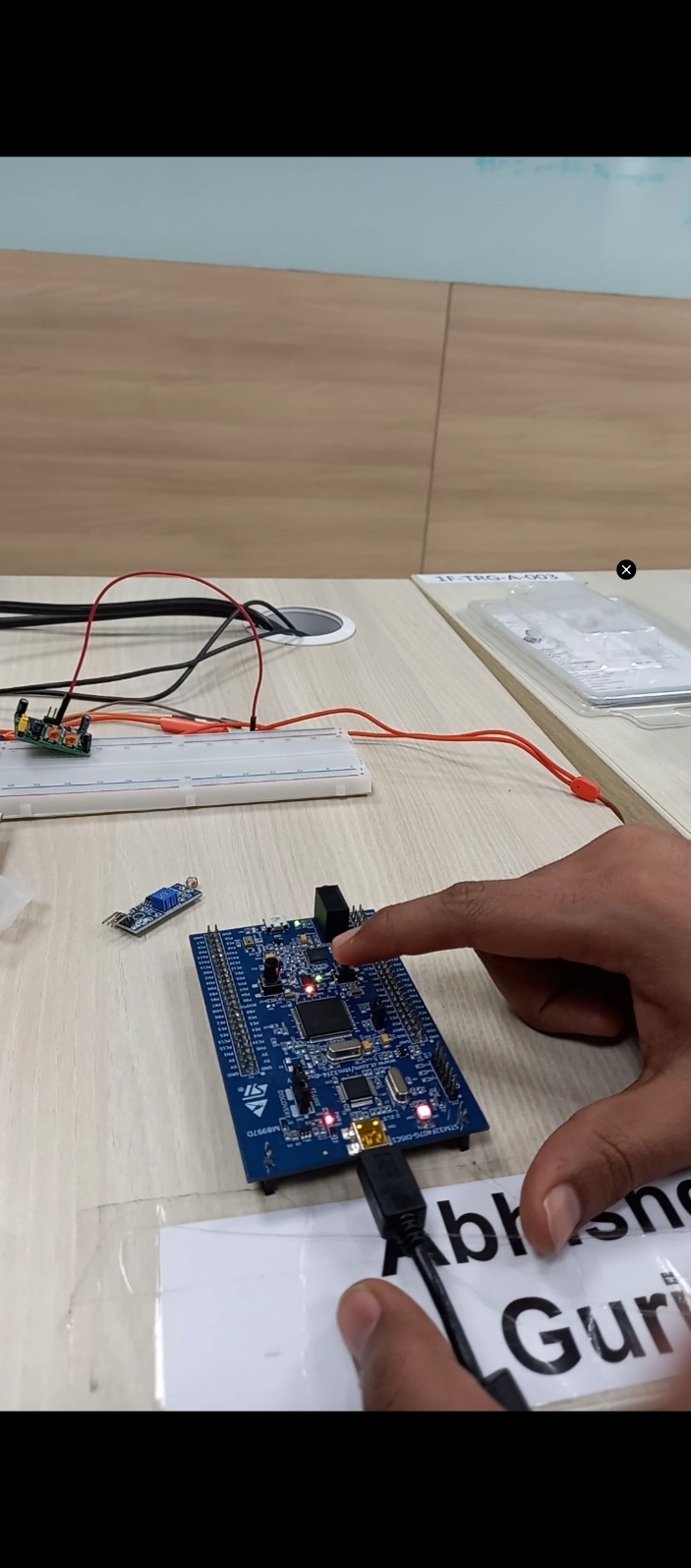
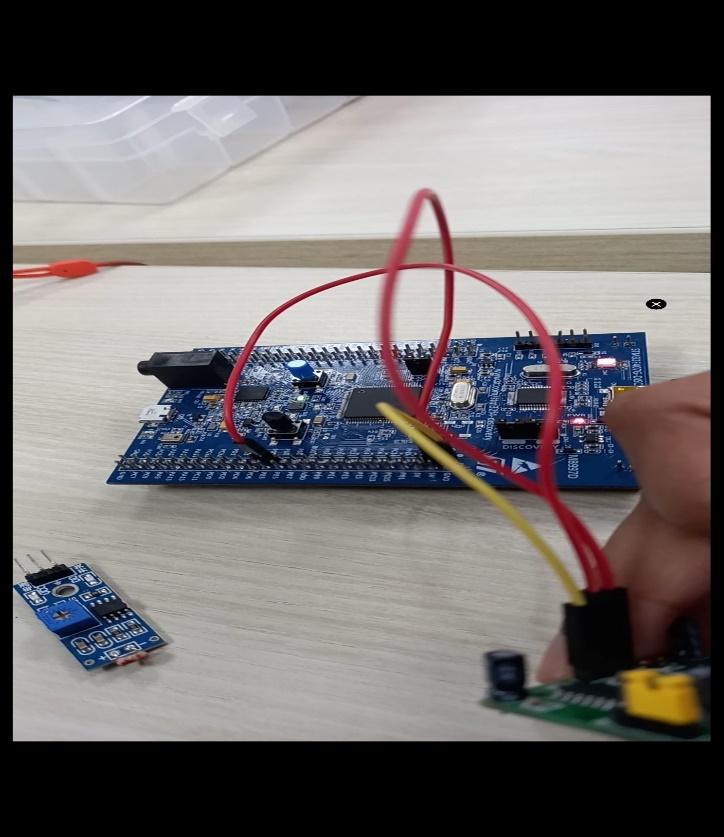


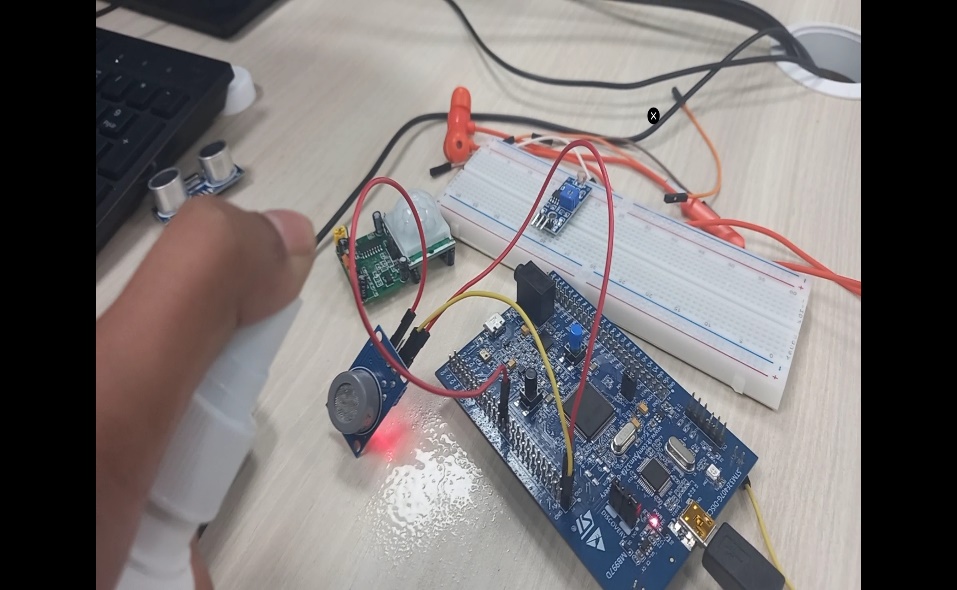
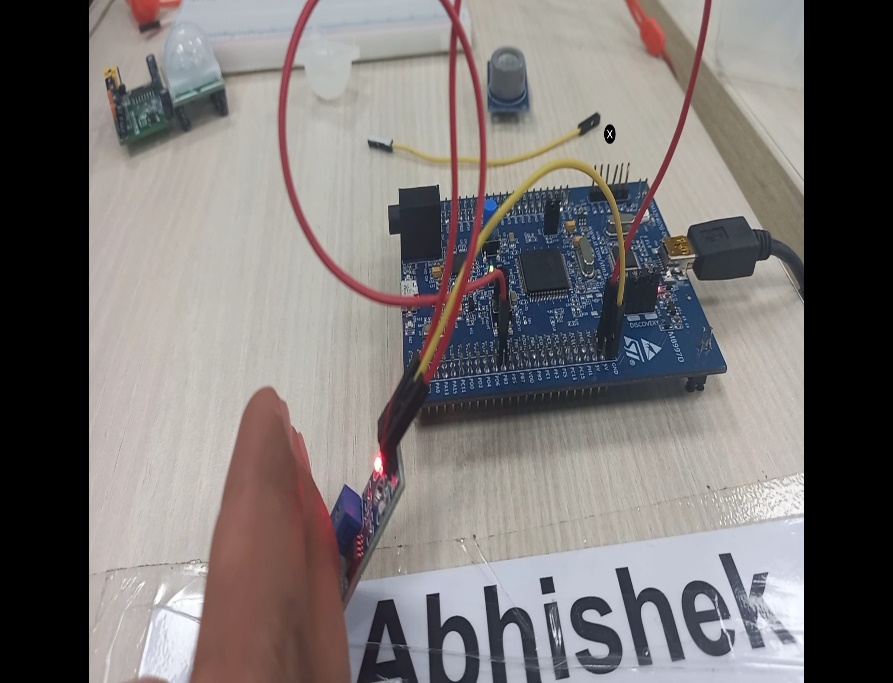




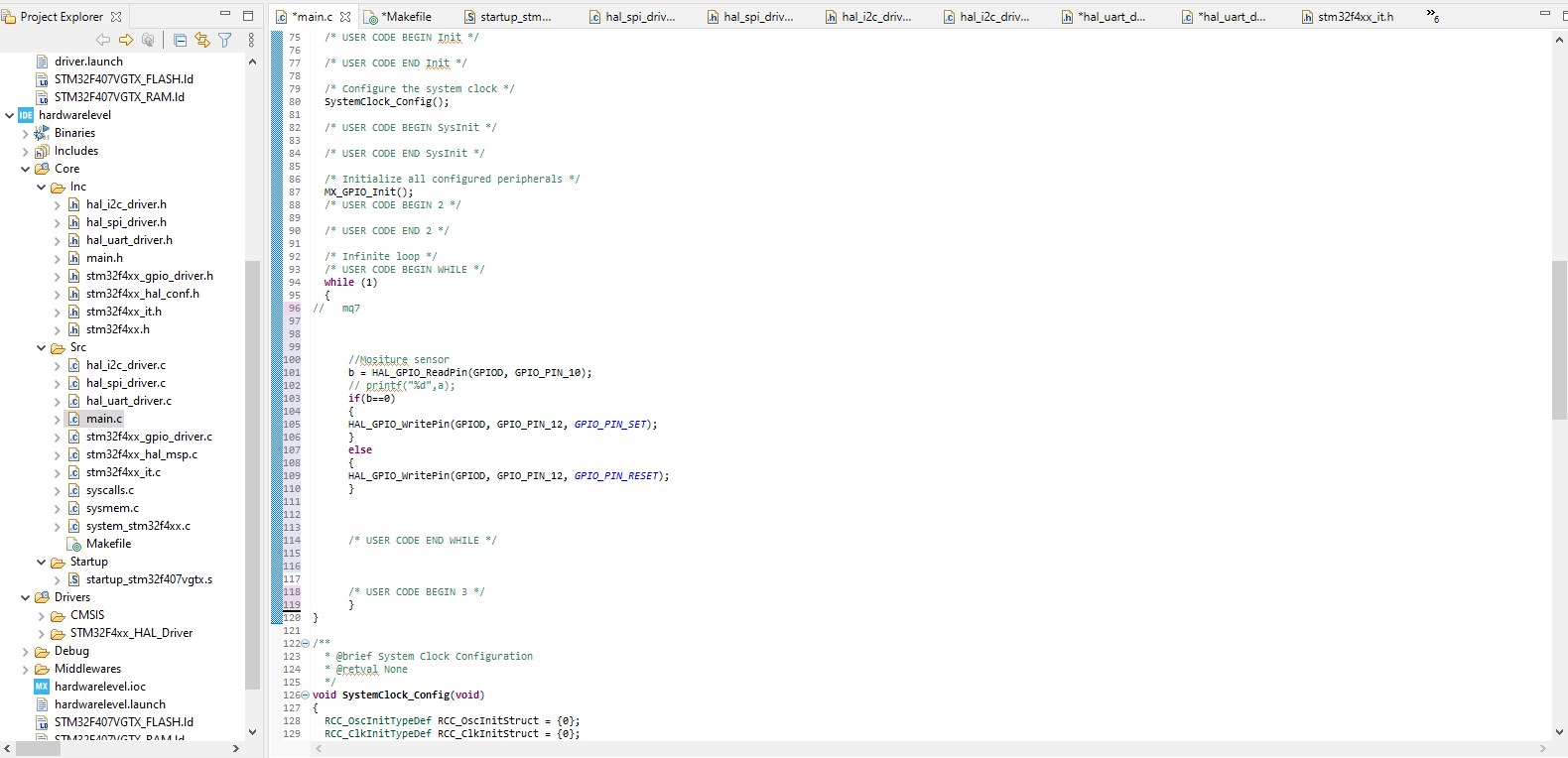


**Few Glimpses of the project:**



**Fig. 17: Project pictures**



**Fig. 18: Project Interface**

**Github Link:**

<https://github.com/99003779/Embedded-C_Official.git>