

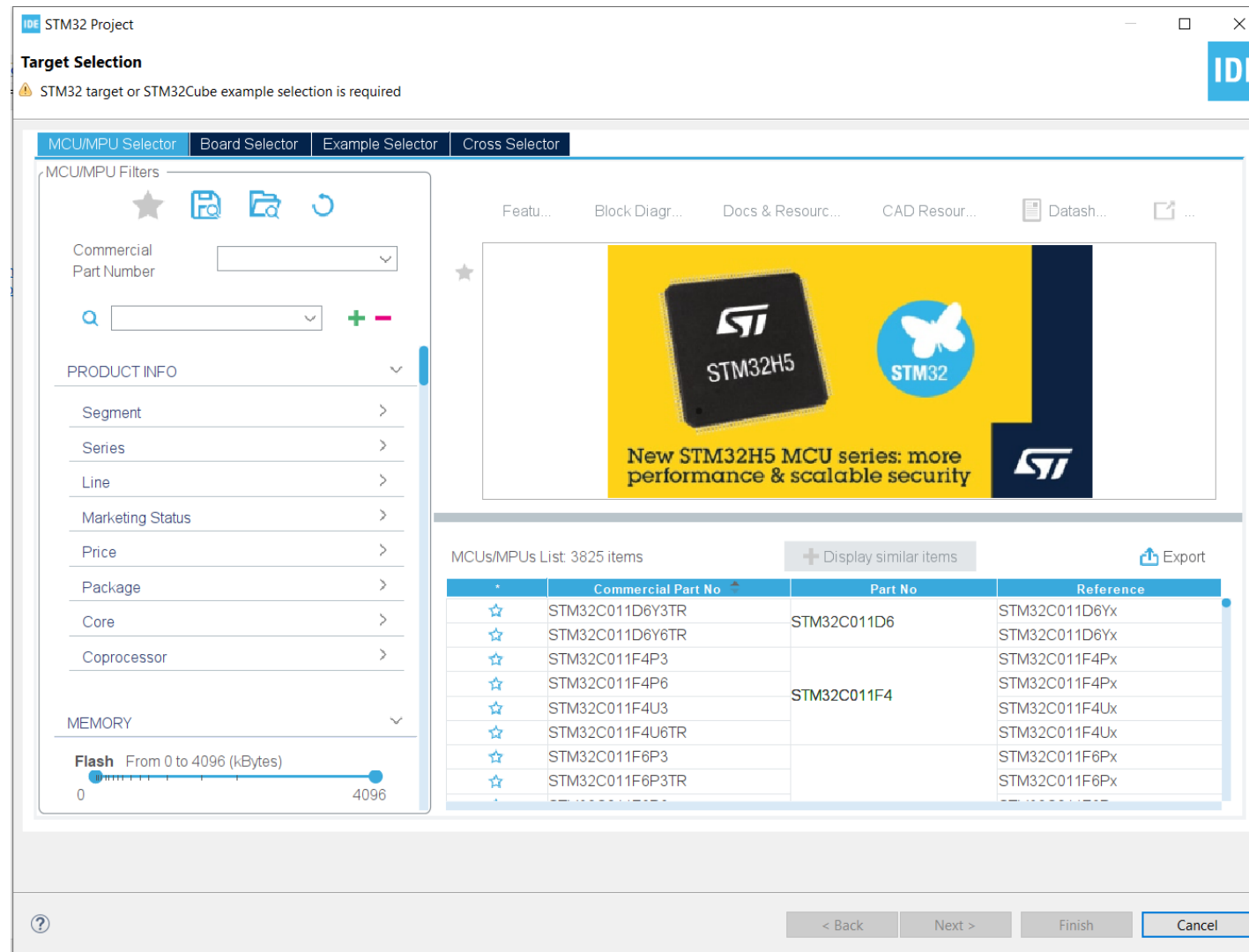
UCSD Embedded C Assignment 5

By

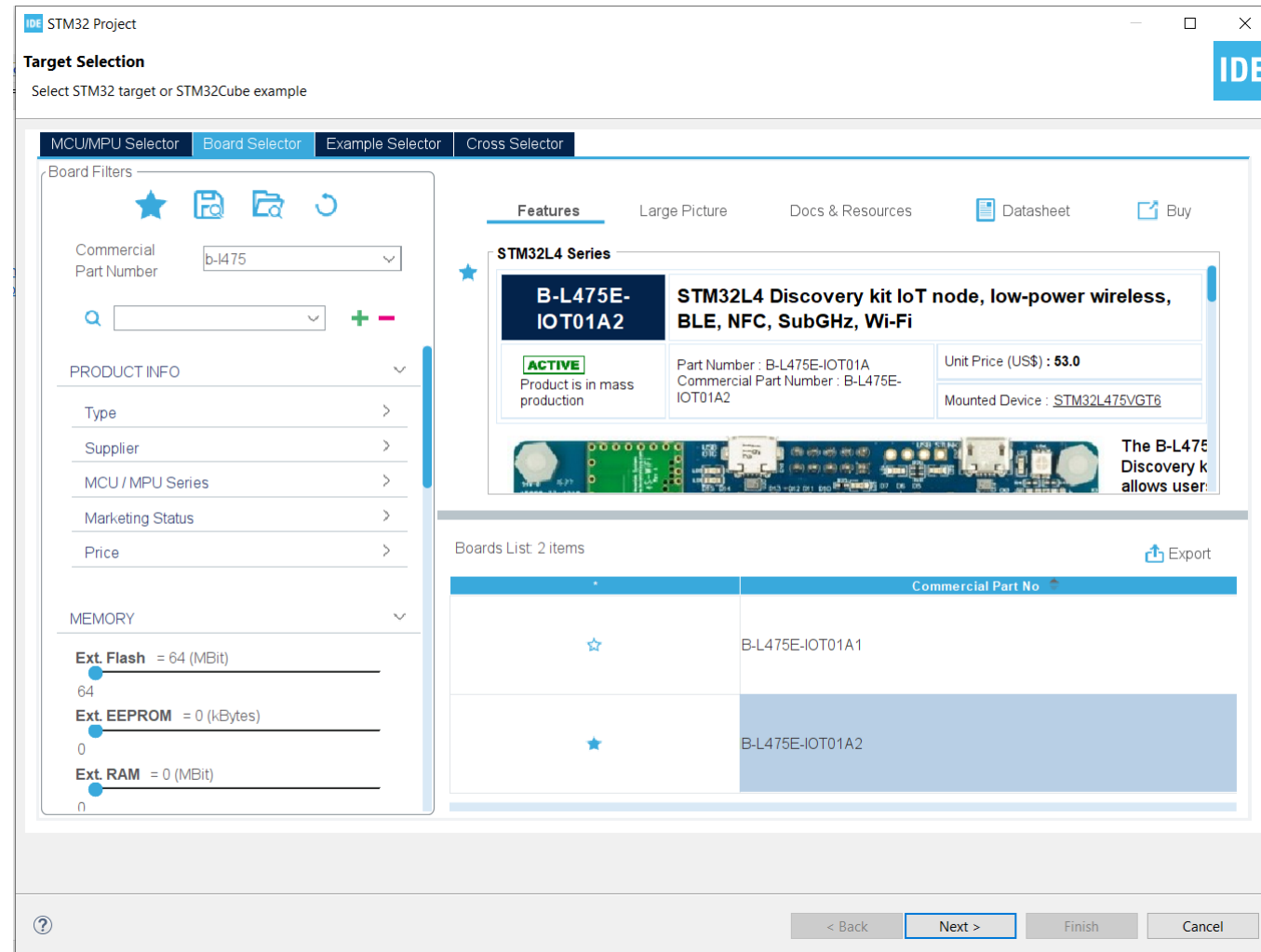
Hsuankai Chang

hsuankac@umich.edu

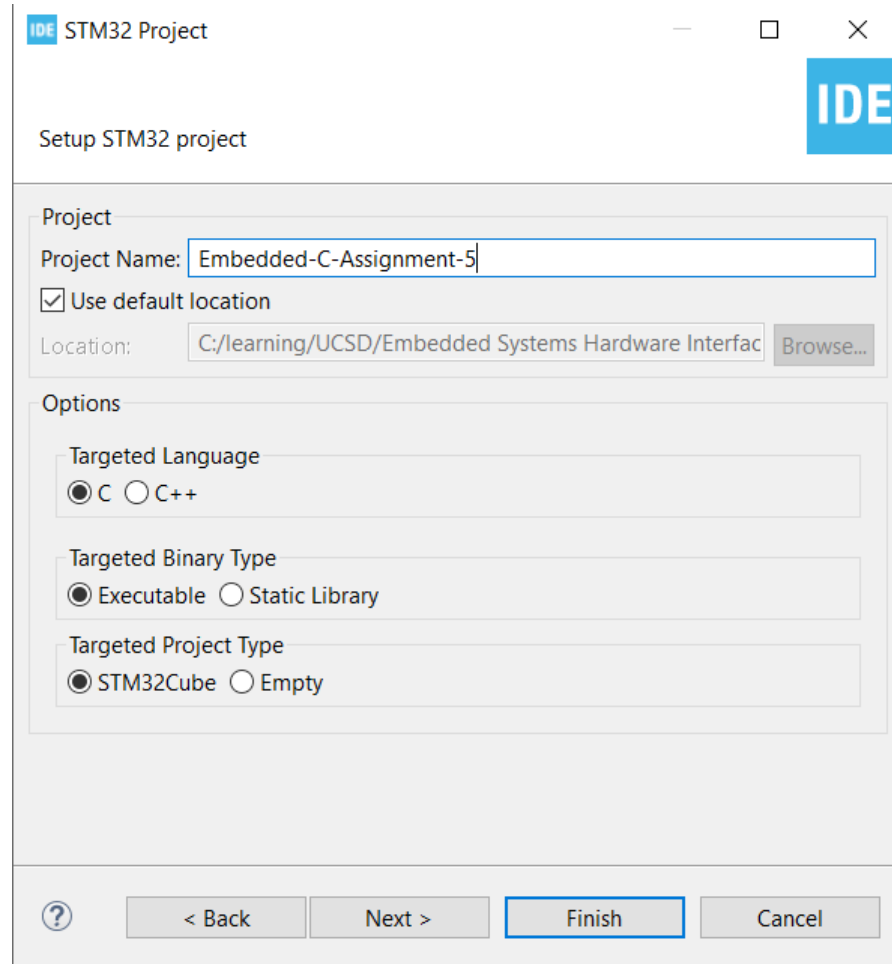
Step 1. Startup STM32CubeIDE and create new STM32 project



Step 2. Access board selector and type in the board you use, click Next



Step 3. Enter the project name then click Next



The image shows a 'Setup STM32 project' dialog box from an IDE. The window title is 'STM32 Project'. The main heading is 'Setup STM32 project'. Under the 'Project' section, the 'Project Name' field contains 'Embedded-C-Assignment-5'. The 'Use default location' checkbox is checked. The 'Location' field shows 'C:/learning/UCSD/Embedded Systems Hardware Interfac' with a 'Browse...' button. The 'Options' section has three groups: 'Targeted Language' with 'C' selected, 'Targeted Binary Type' with 'Executable' selected, and 'Targeted Project Type' with 'STM32Cube' selected. At the bottom, there are buttons for '?', '< Back', 'Next >', 'Finish' (highlighted with a blue border), and 'Cancel'.

STM32 Project

Setup STM32 project

Project

Project Name: Embedded-C-Assignment-5

☒ Use default location

Location: C:/learning/UCSD/Embedded Systems Hardware Interfac Browse...

Options

Targeted Language

☒ C ☐ C++

Targeted Binary Type

☒ Executable ☐ Static Library

Targeted Project Type

☒ STM32Cube ☐ Empty

? < Back Next > Finish Cancel

Step 4. See the firmware package name and version



The image shows a Windows-style dialog box titled "STM32 Project" with a subtitle "Firmware Library Package Setup". The subtitle is followed by the instruction "Setup STM32 target's firmware". The dialog is divided into three sections: "Target and Firmware Package", "Firmware and Software Package Repository", and "Code Generator Options". In the first section, "Target Reference" is set to "B-L475E-IOT01A2" and "Firmware Package Name and Version" is set to "STM32Cube FW_L4" with a dropdown menu showing "V1.17.2". The second section shows the "Location" as "C:\Users\hsuankai.chang\STM32Cube\Repository" and includes a link to the "Firmware Updater". The third section has three radio button options for code generation, with "Copy only the necessary library files" being selected. At the bottom, there are buttons for "?", "< Back", "Next >", "Finish", and "Cancel".

IDE STM32 Project

Firmware Library Package Setup

Setup STM32 target's firmware

Target and Firmware Package

Target Reference: B-L475E-IOT01A2

Firmware Package Name and Version: STM32Cube FW_L4 V1.17.2

Firmware and Software Package Repository

Location:
C:\Users\hsuankai.chang\STM32Cube\Repository

See ['Firmware Updater'](#) for settings related to package installation

Code Generator Options

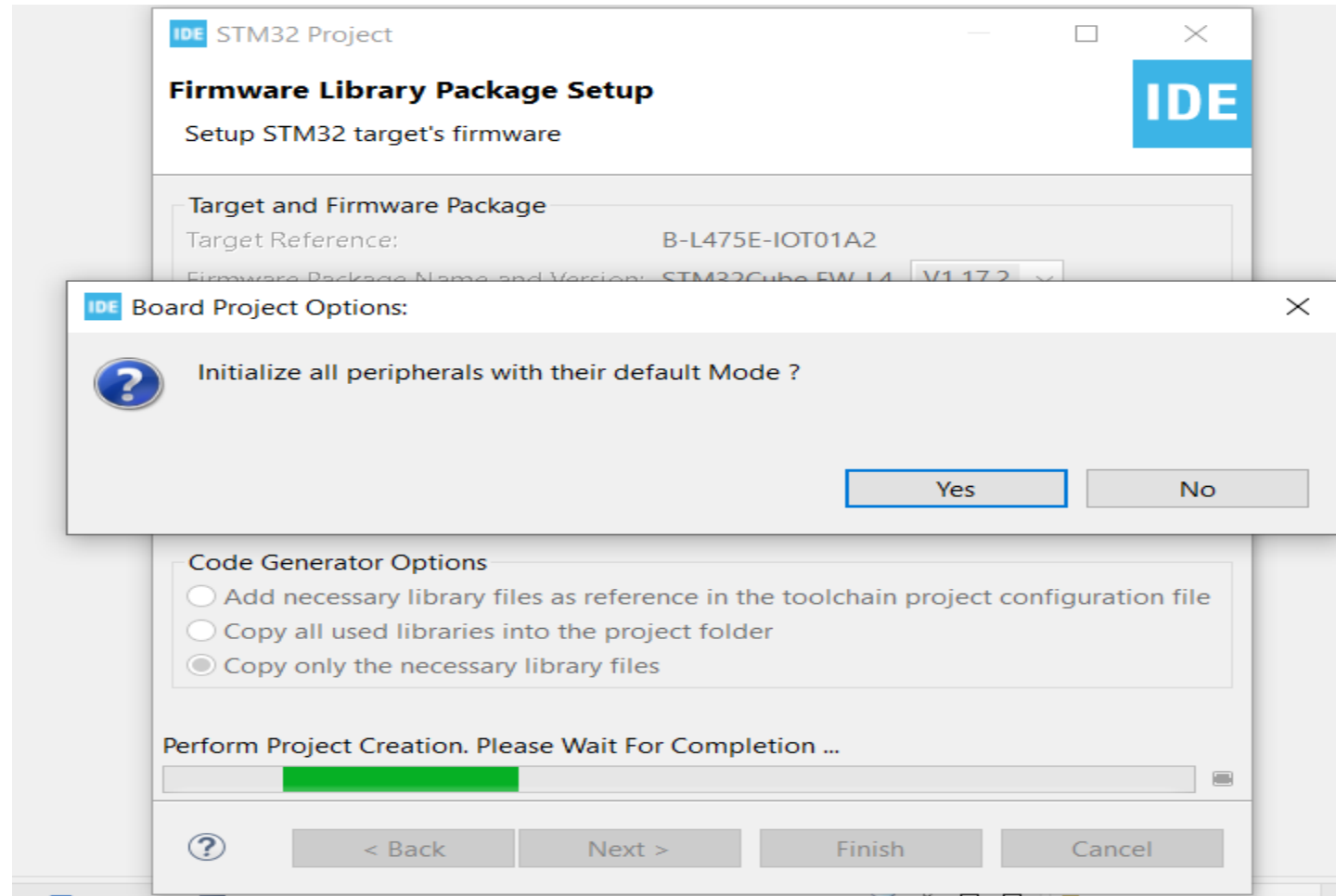
☐ Add necessary library files as reference in the toolchain project configuration file

☐ Copy all used libraries into the project folder

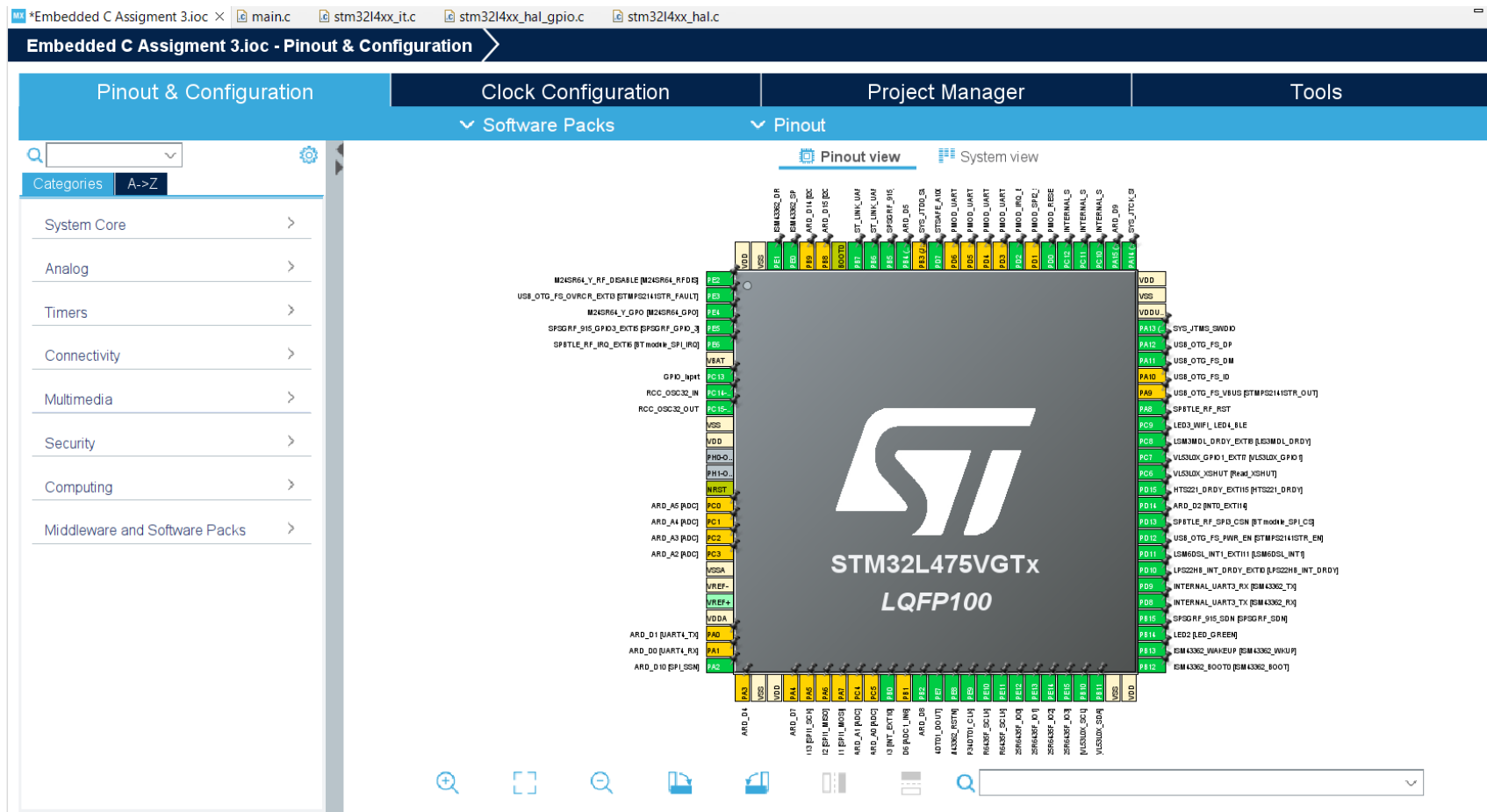
☒ Copy only the necessary library files

? < Back Next > Finish Cancel

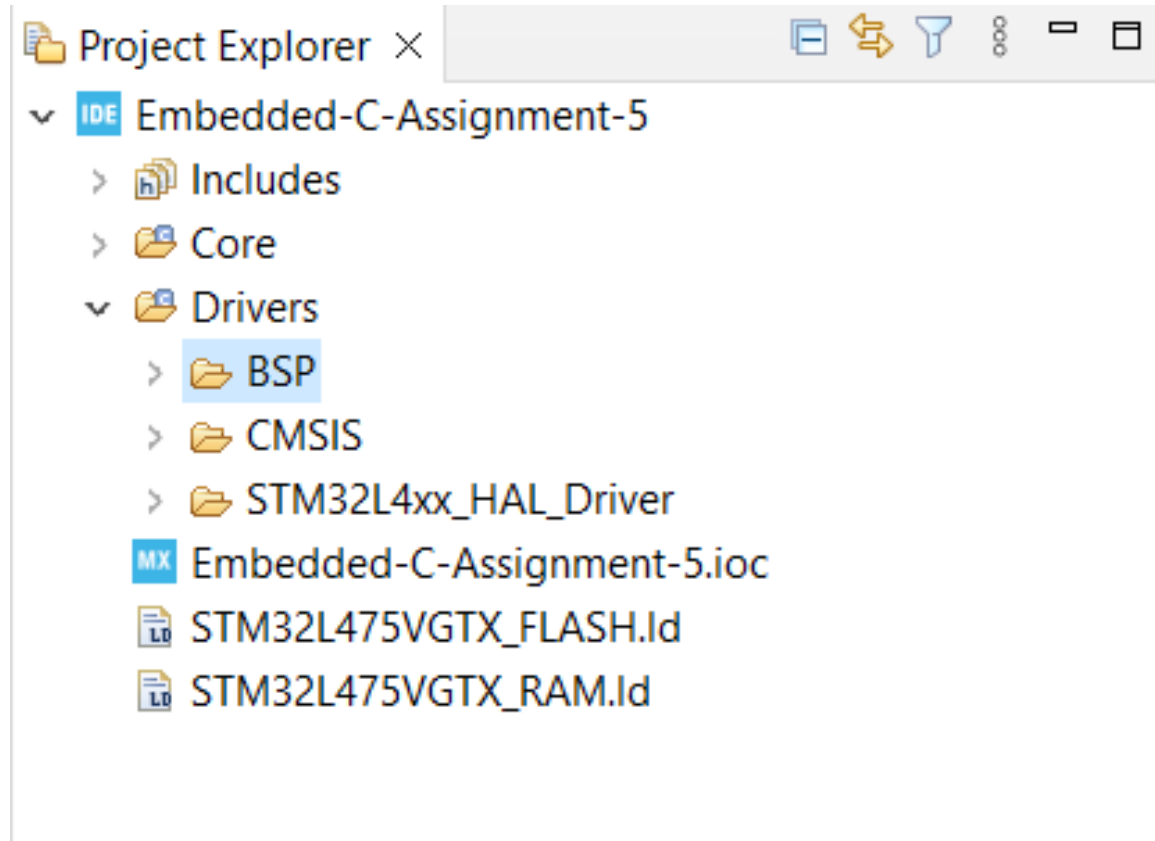
Step 5. Click yes to initialize all peripherals to default



Step 6. When in .ioc file, click Pinout & Configurations. Use the default setting then generate code



Step 7. Create BSP folder in project under drivers

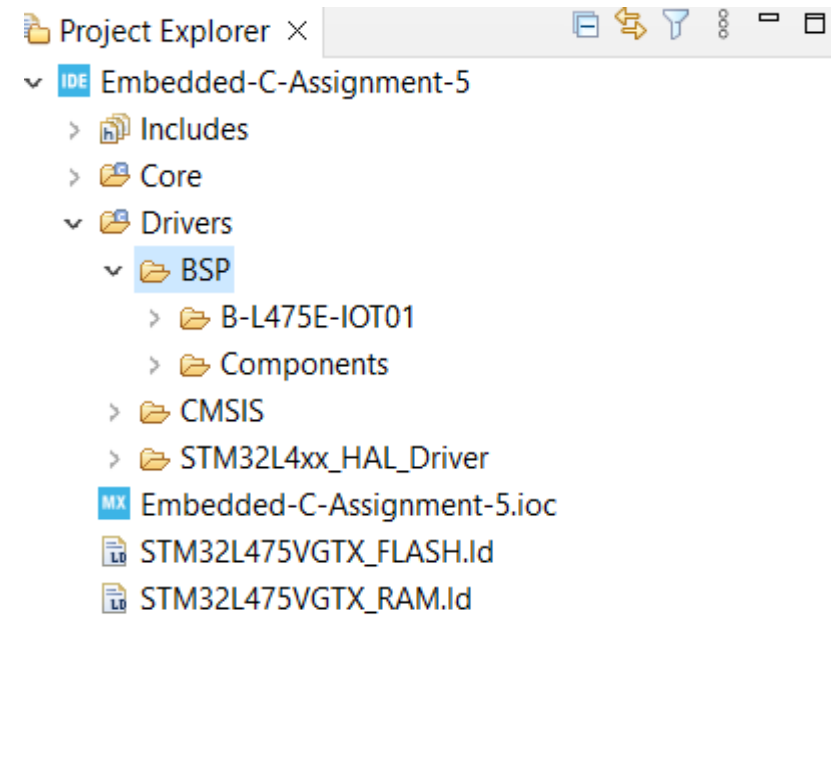


Step 8. Find the BSP location in the system, then copy B-L475E-IOT01 and Components folder into BSP folder in the project

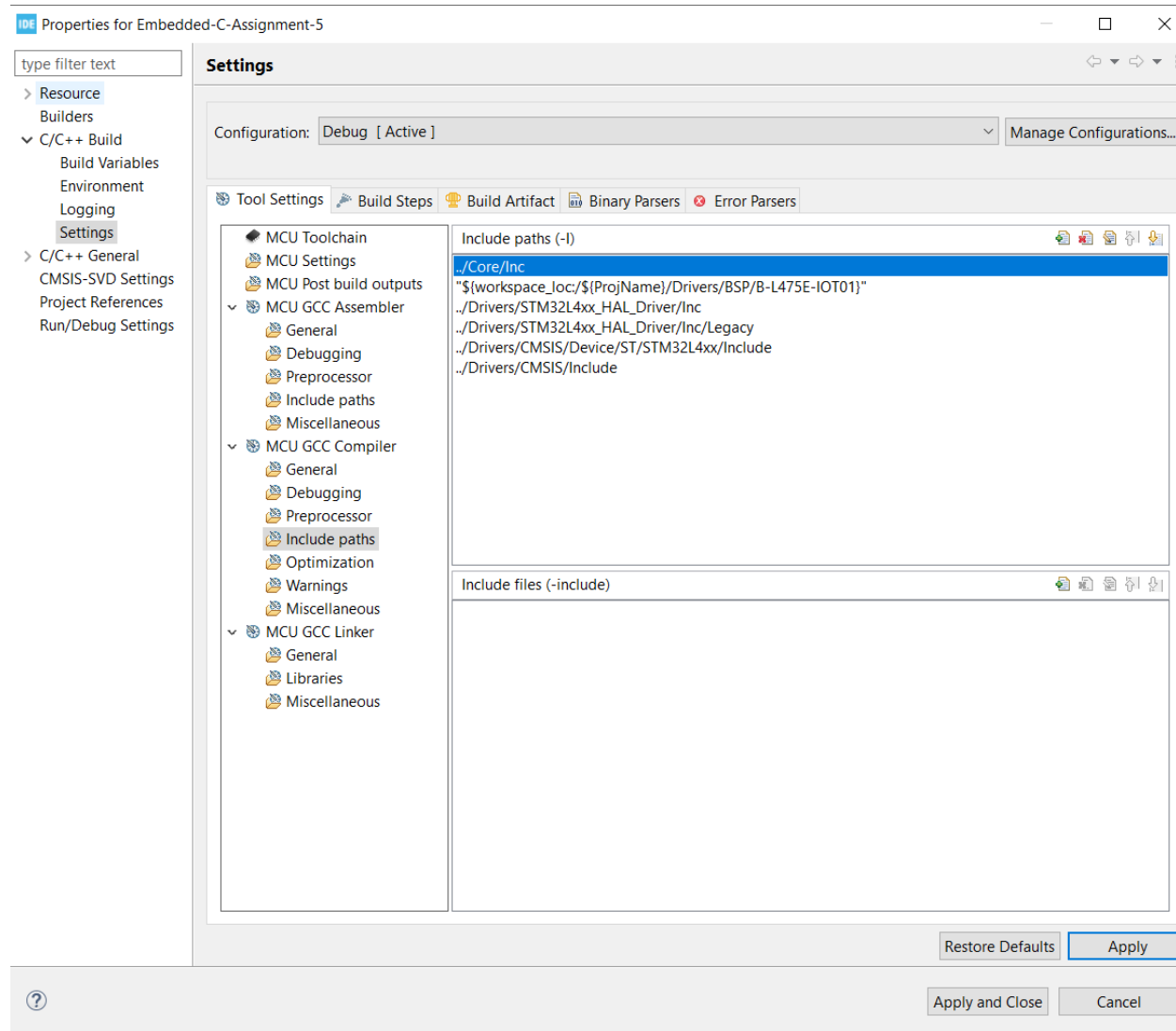
STM32Cube > Repository > STM32Cube_FW_L4_V1.17.2 > Drivers > BSP

Search BSP

Name	Date modified	Type	Size
Adafruit_Shield	2/27/2023 10:24 PM	File folder	
B-L455I-IOT01	2/27/2023 10:24 PM	File folder	
B-L475E-IOT01	2/27/2023 10:24 PM	File folder	
Components	2/27/2023 10:24 PM	File folder	
STM32L4P5G-Discovery	2/27/2023 10:24 PM	File folder	
STM32L4R9I_EVAL	2/27/2023 10:24 PM	File folder	
STM32L4R9I-Discovery	2/27/2023 10:24 PM	File folder	
STM32L4xx_Nucleo	2/27/2023 10:24 PM	File folder	
STM32L4xx_Nucleo_32	2/27/2023 10:24 PM	File folder	
STM32L4xx_Nucleo_144	2/27/2023 10:24 PM	File folder	
STM32L476G_EVAL	2/27/2023 10:24 PM	File folder	
STM32L476G-Discovery	2/27/2023 10:24 PM	File folder	
STM32L496G-Discovery	2/27/2023 10:24 PM	File folder	



Step 9. Add the include path



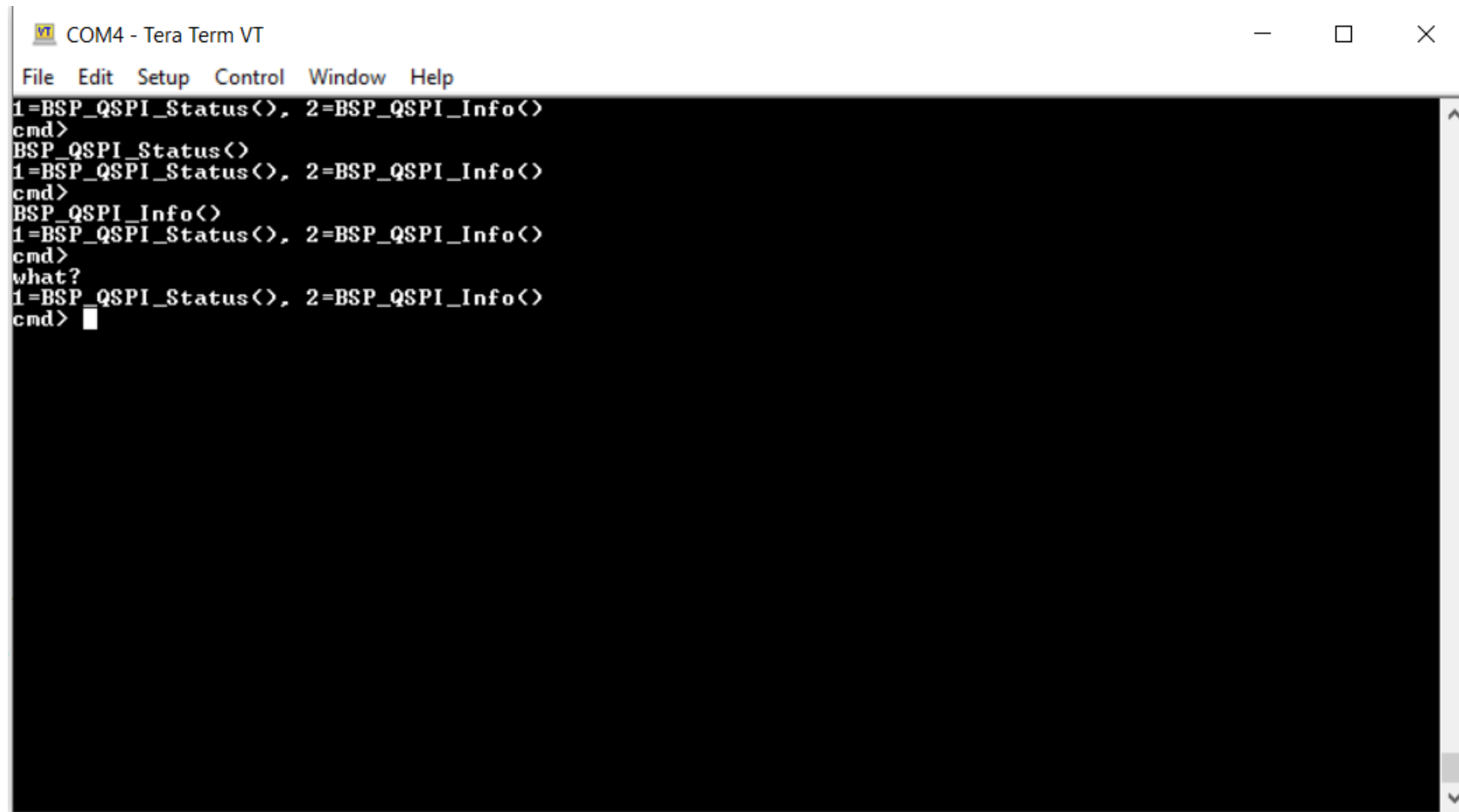
Step 10. User Story 1. CLI. Create a CLI (Command Line Interface) on UART1

```
*main.c x stm32l475e_iot01.h
129 /* Infinite loop */
130 /* USER CODE BEGIN WHILE */
131 while (1)
132 {
133     /* USER CODE END WHILE */
134
135     /* USER CODE BEGIN 3 */
136     // Issue command prompt
137     char *prompt = "1=BSP_QSPI_Status(), 2=BSP_QSPI_Info()\n\r cmd> ";
138     HAL_UART_Transmit(&huart1, (uint8_t*) prompt, strlen(prompt), 1000);
139     // Wait for a single number entry
140     char ch;
141     HAL_UART_Receive(&huart1, (uint8_t*)&ch, 1, HAL_MAX_DELAY);
142     char *msg = "what?";
143     switch(ch)
144     {
145     case '1':
146     {
147         msg = "\r\nBSP_QSPI_Status()\r\n";
148         HAL_UART_Transmit(&huart1, (uint8_t*) msg, strlen(msg), 1000);
149         do_qspi_status();
150         break;
151     }
152     case '2':
153     {
154         msg = "\r\nBSP_QSPI_Info()\r\n";
155         HAL_UART_Transmit(&huart1, (uint8_t*) msg, strlen(msg), 1000);
156         do_qspi_info();
157         break;
158     }
159     default:
160     {
161         msg = "\r\nwhat?\r\n";
162         HAL_UART_Transmit(&huart1, (uint8_t*) msg, strlen(msg), 1000);
163     }
164     }
165 }
166 /* USER CODE END 3 */
```

```
17 //
18 /* USER CODE END Header */
19 /* Includes -----
20 #include "main.h"
21
22 /* Private includes -----
23 /* USER CODE BEGIN Includes */
24 #include "stm32l475e_iot01.h"
25 #include "stm32l475e_iot01_qspi.h"
26 #include <string.h>
27 #include <stdio.h>
28 /* USER CODE END Includes */
29
```

```
78 // USER CODE BEGIN 0
79 void do_qspi_status()
80 {
81
82 }
83
84 void do_qspi_info()
85 {
86
87 }
88 /* USER CODE END 0 */
89
```

Step 11. User Story 1: Build and run the code, test is successful

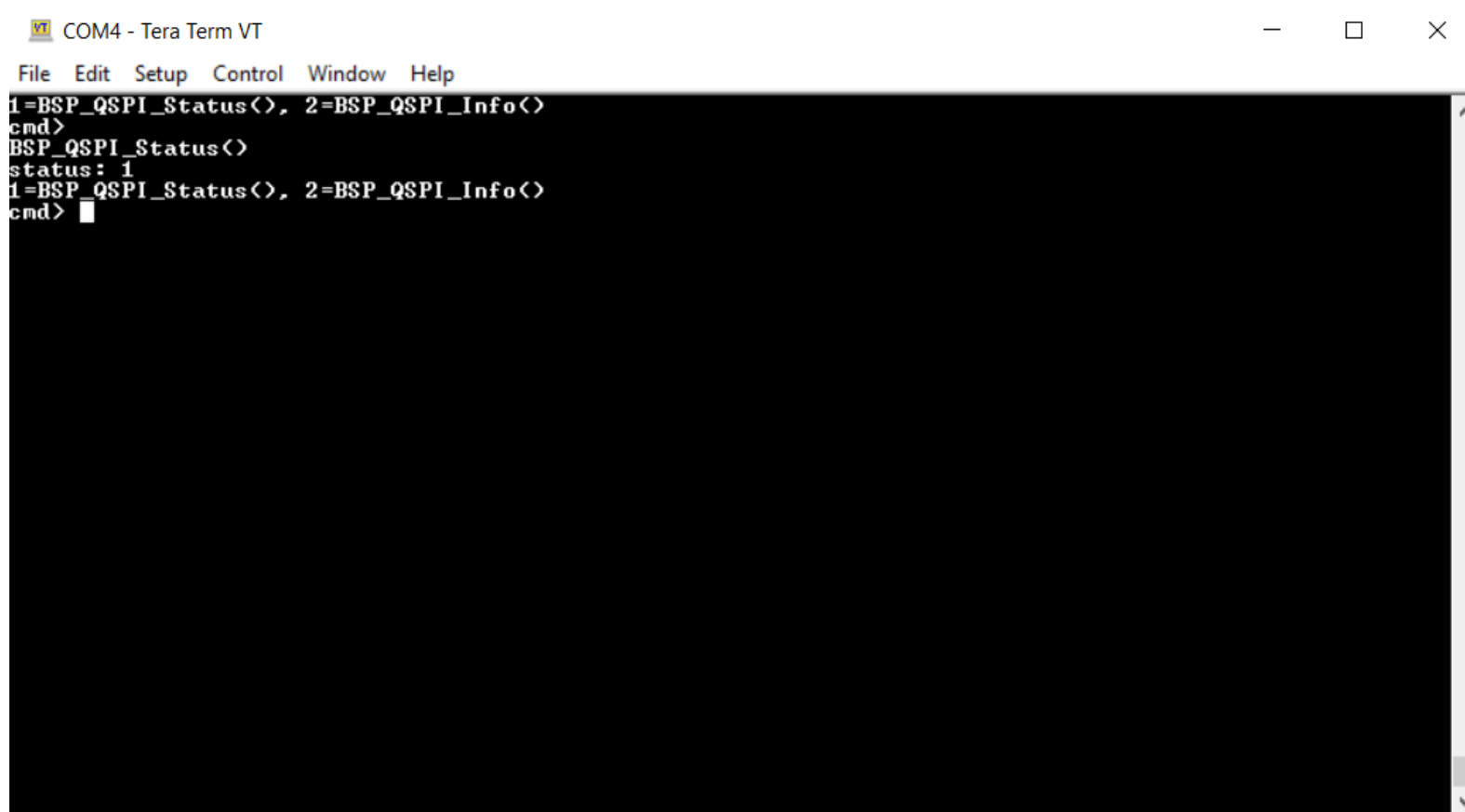


```
COM4 - Tera Term VT
File Edit Setup Control Window Help
1=BSP_QSPI_Status(), 2=BSP_QSPI_Info()
cmd>
BSP_QSPI_Status()
1=BSP_QSPI_Status(), 2=BSP_QSPI_Info()
cmd>
BSP_QSPI_Info()
1=BSP_QSPI_Status(), 2=BSP_QSPI_Info()
cmd>
what?
1=BSP_QSPI_Status(), 2=BSP_QSPI_Info()
cmd> 
```

Step 12. User Story 2. BSP_QSPI_GetStatus(). When the user selects 1, display the results of calling BSP_QSPI_GetStatus() on the console.

```
79 void do_qspi_status()
80 {
81     uint8_t status = BSP_QSPI_GetStatus();
82     char buf[100];
83     snprintf(buf, sizeof(buf), "status: %d\r\n", status);
84
85     HAL_UART_Transmit(&huart1, (uint8_t*)buf, strlen(buf), 1000);
86 }
```

Step 13. User Story 2: Build and run the code, test is successful



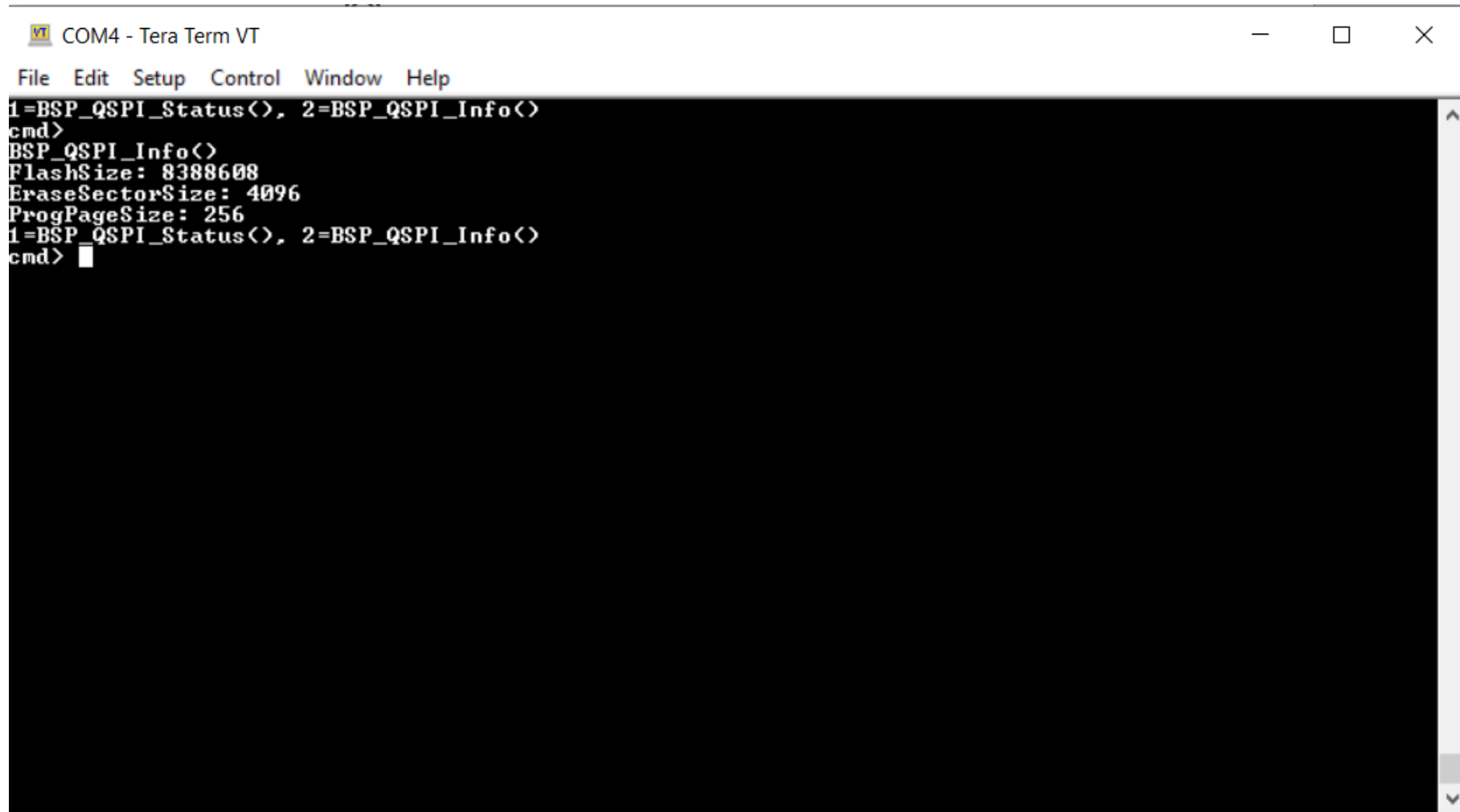
A screenshot of a Tera Term VT window titled "COM4 - Tera Term VT". The window has a menu bar with "File", "Edit", "Setup", "Control", "Window", and "Help". The main area is black with white text. The text shows a command prompt session where the command `1=BSP_QSPI_Status(), 2=BSP_QSPI_Info()` is entered, followed by `cmd>`. The output shows `BSP_QSPI_Status()` and `status: 1`. The command is then repeated, and the prompt `cmd>` is shown with a cursor.

```
COM4 - Tera Term VT
File Edit Setup Control Window Help
1=BSP_QSPI_Status(), 2=BSP_QSPI_Info()
cmd>
BSP_QSPI_Status()
status: 1
1=BSP_QSPI_Status(), 2=BSP_QSPI_Info()
cmd> █
```

Step 14. User Story 3. BSP_QSPI_GetInfo(). When the user selects 2, display the results of calling BSP_QSPI_GetStatus() on the console.

```
88 void do_qspi_info()
89 {
90     QSPI_Info info = {0};
91     BSP_QSPI_GetInfo(&info);
92
93     char buf[100];
94     snprintf(buf, sizeof(buf), "FlashSize: %lu\r\n"
95                                "EraseSectorSize: %lu\r\n"
96                                "ProgPageSize: %lu\r\n",
97                                info.FlashSize,
98                                info.EraseSectorSize,
99                                info.ProgPageSize);
100     HAL_UART_Transmit(&huart1, (uint8_t*)buf, strlen(buf), 1000);
101 }
```

Step 15. User Story 3. Build and run the code, test is successful

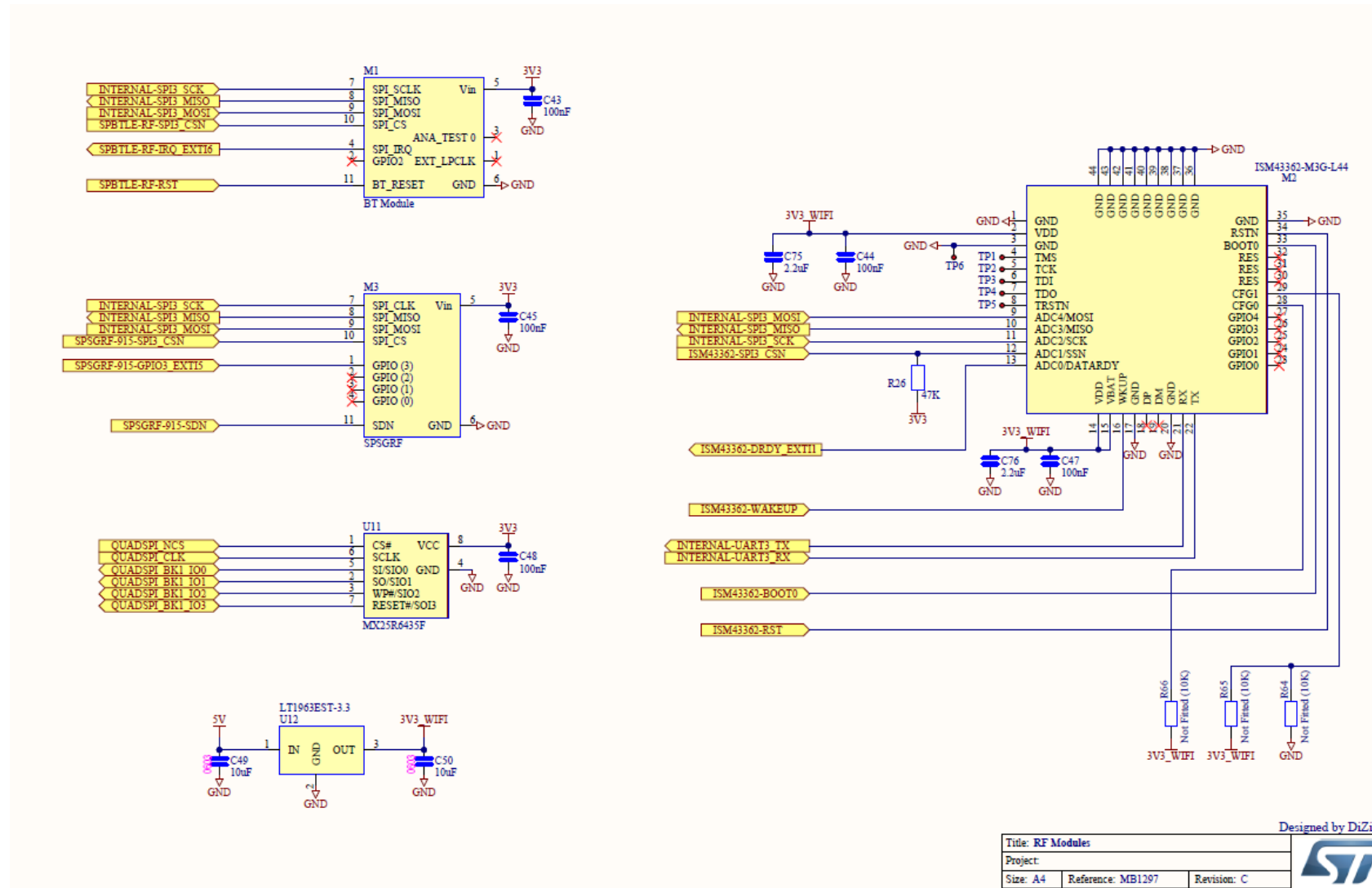


The screenshot shows a Tera Term VT terminal window titled "COM4 - Tera Term VT". The window has a menu bar with "File", "Edit", "Setup", "Control", "Window", and "Help". The terminal output is as follows:

```
1=BSP_QSPI_Status(<), 2=BSP_QSPI_Info(<)  
cmd>  
BSP_QSPI_Info(<)  
FlashSize: 8388608  
EraseSectorSize: 4096  
ProgPageSize: 256  
1=BSP_QSPI_Status(<), 2=BSP_QSPI_Info(<)  
cmd> █
```

The terminal window has a black background with white text. A vertical scrollbar is visible on the right side of the terminal area.

Appendix, schematic for the module on the board that use SPI as connection



Appendix, Ardunio connector that has SPI connection

