

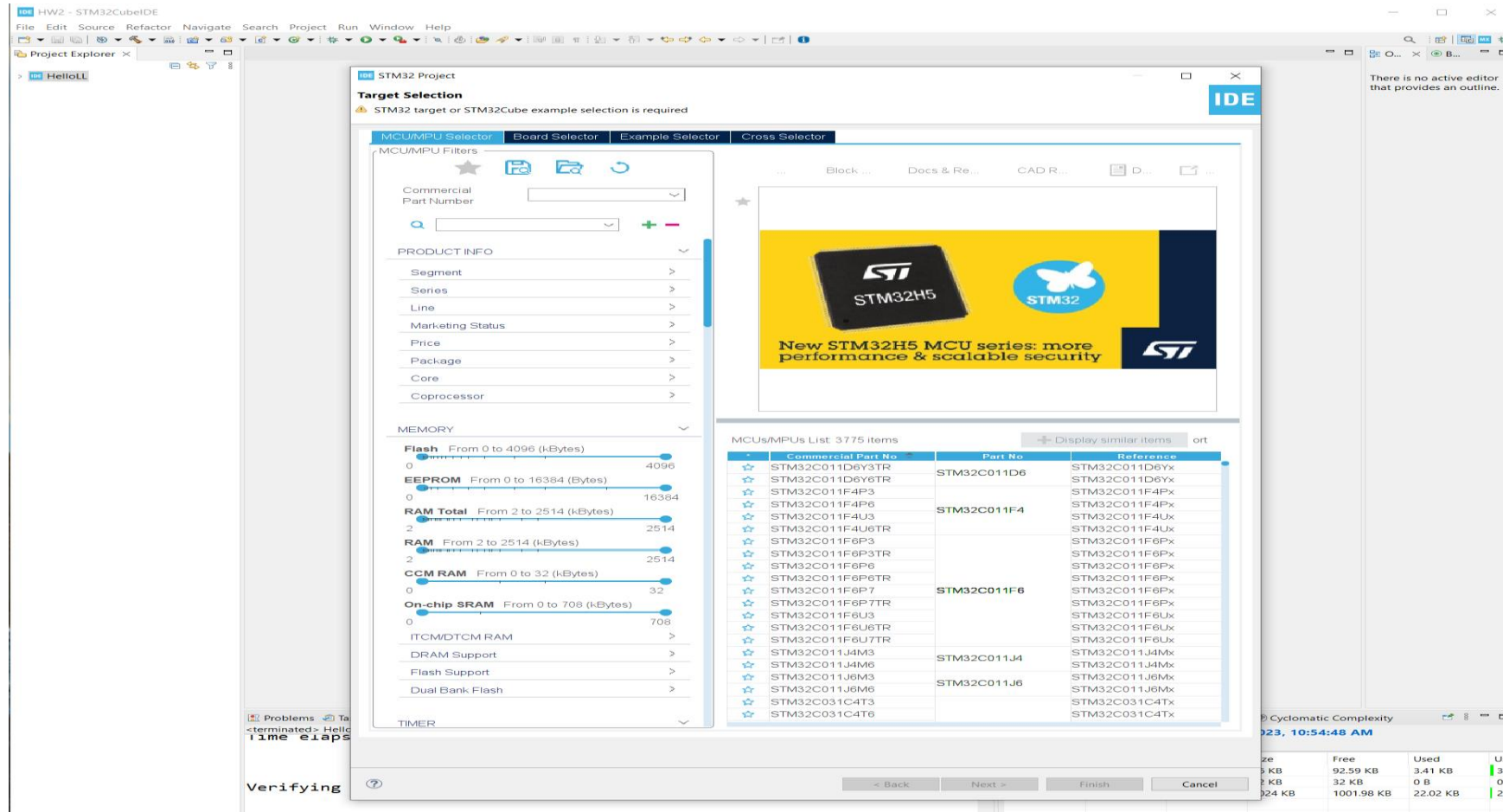
# UCSD Embedded C Assignment 6

By

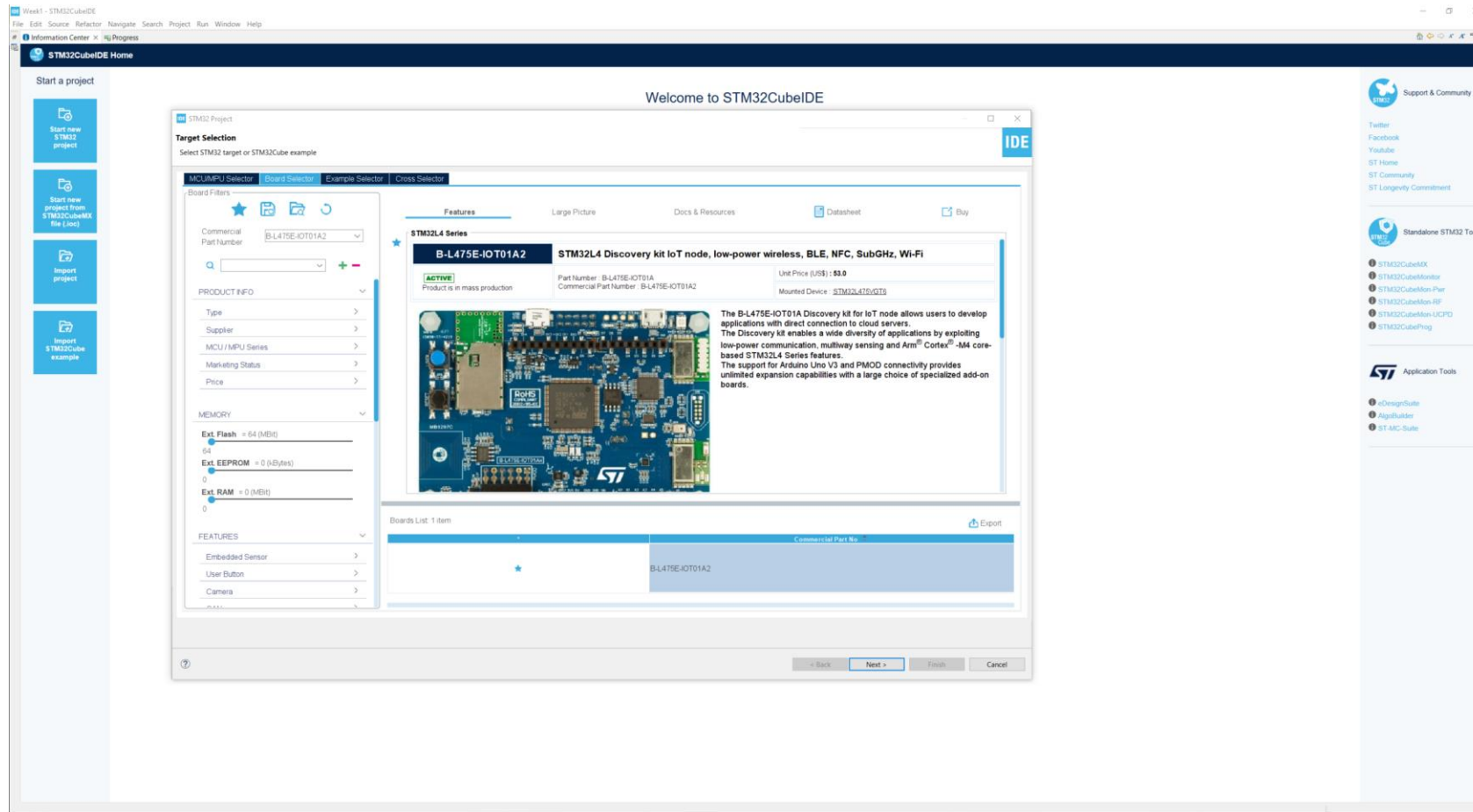
Hsuankai Chang

[hsuankac@umich.edu](mailto:hsuankac@umich.edu)

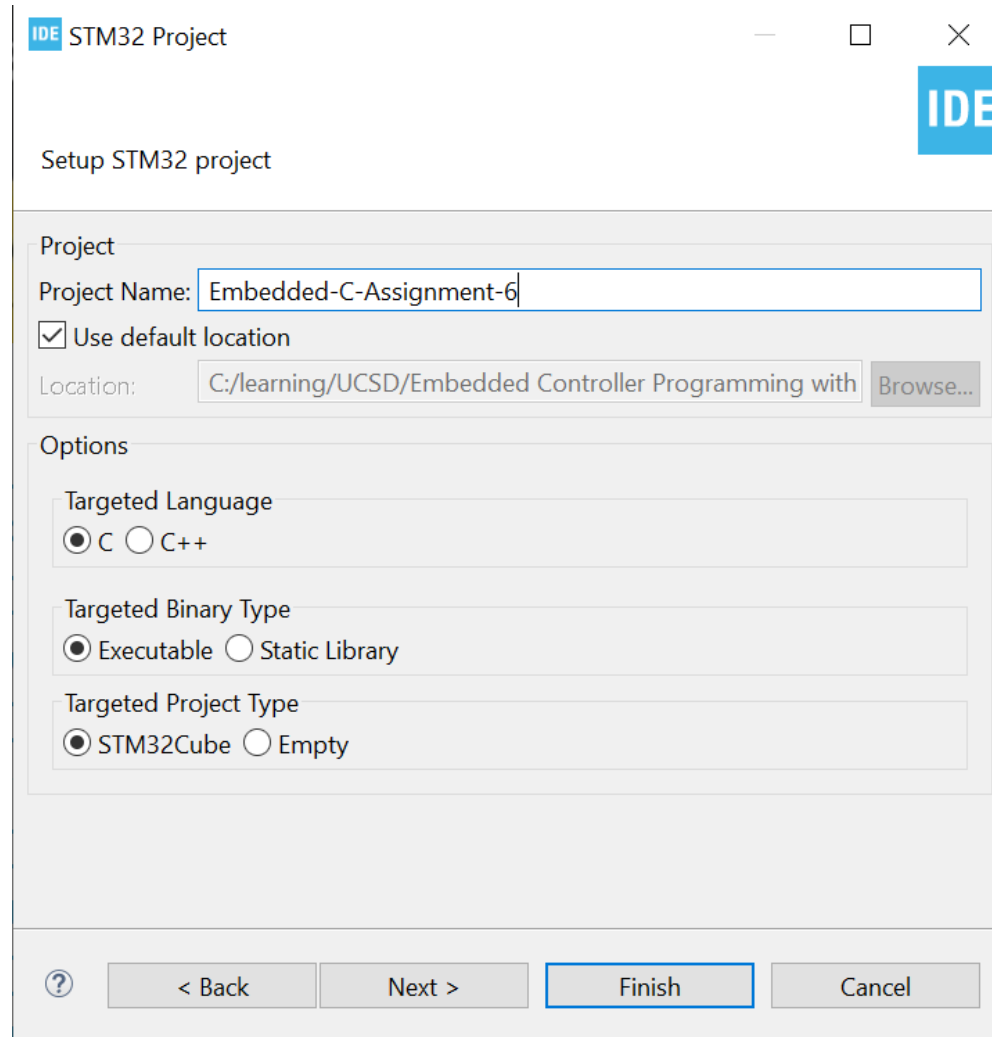
# Step 1. Startup STM32CubeIDE and create new STM32 project



# 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 title is 'Setup STM32 project'. The 'Project' section contains a 'Project Name' field with the text 'Embedded-C-Assignment-6' and a checked 'Use default location' checkbox. The 'Location' field shows the path 'C:/learning/UCSD/Embedded Controller Programming with' and a 'Browse...' button. The 'Options' section has three groups of radio buttons: '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-6

☒ Use default location

Location: C:/learning/UCSD/Embedded Controller Programming with 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 also includes 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, the "Target Reference" is "B-L475E-IOT01A2" and the "Firmware Package Name and Version" is "STM32Cube FW\_L4 V1.17.2", with the version part highlighted by a blue selection box. The second section shows the "Location" as "C:\Users\hsuankai.chang\STM32Cube\Repository" and includes a link to the "Firmware Updater". The third section contains three radio button options for code generation, with the last option, "Copy only the necessary library files", being selected. At the bottom, there are buttons for "?", "< Back", "Next >", "Finish" (which is highlighted with a blue border), 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

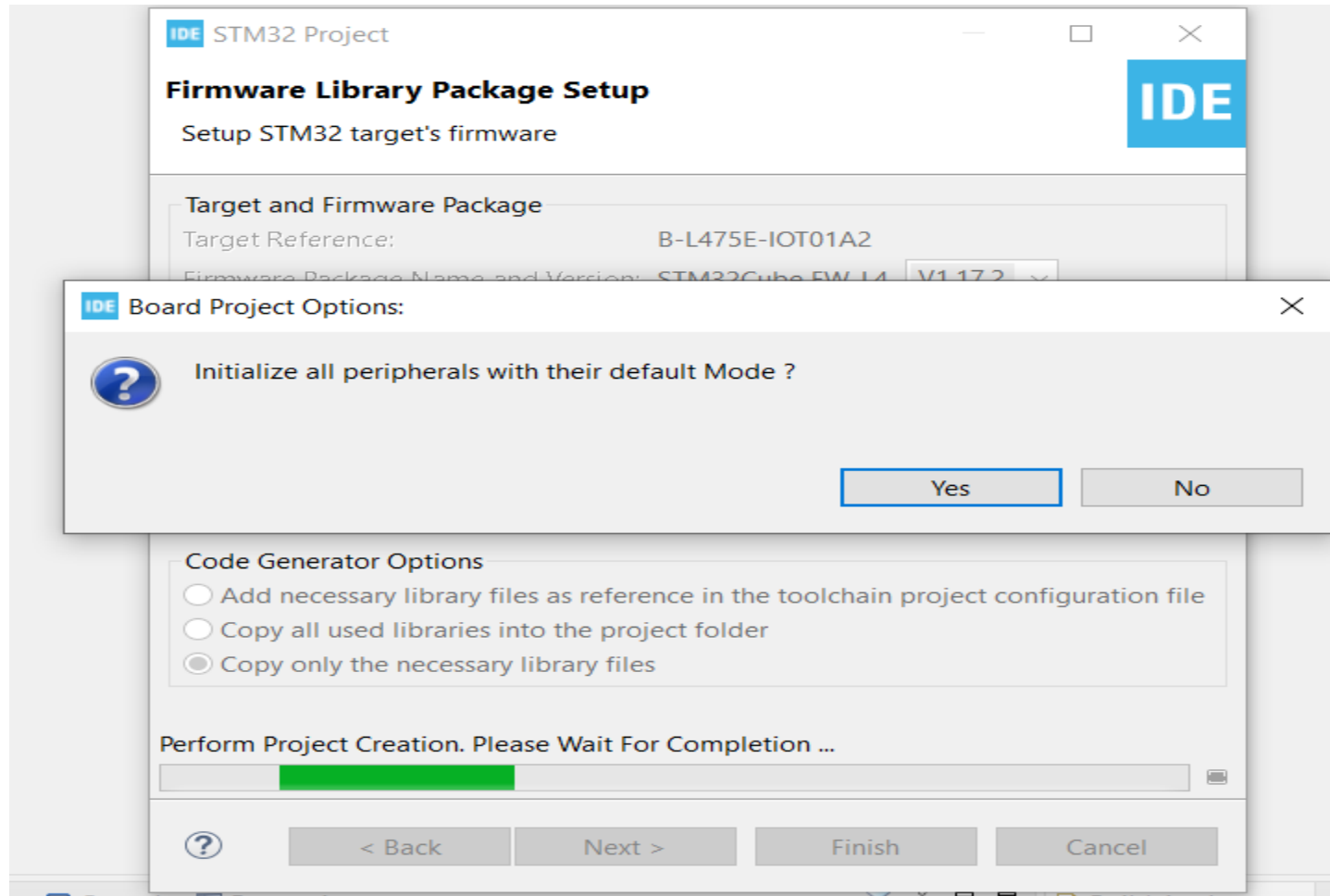
☐ Add necessary library files as reference in the toolchain project configuration file

☐ Copy all used libraries into the project folder

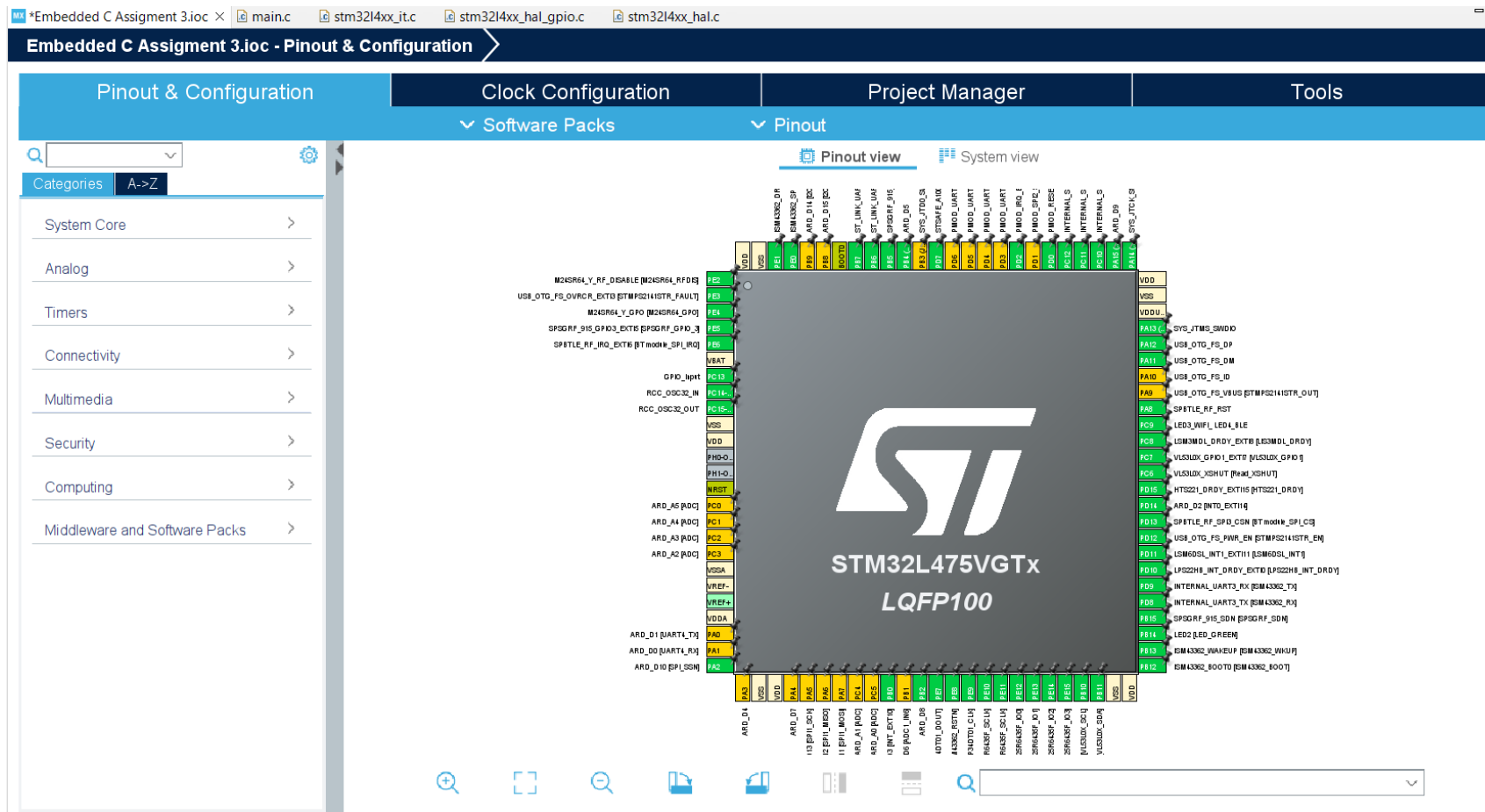
☒ Copy only the necessary library files

? < Back Next > Finish Cancel

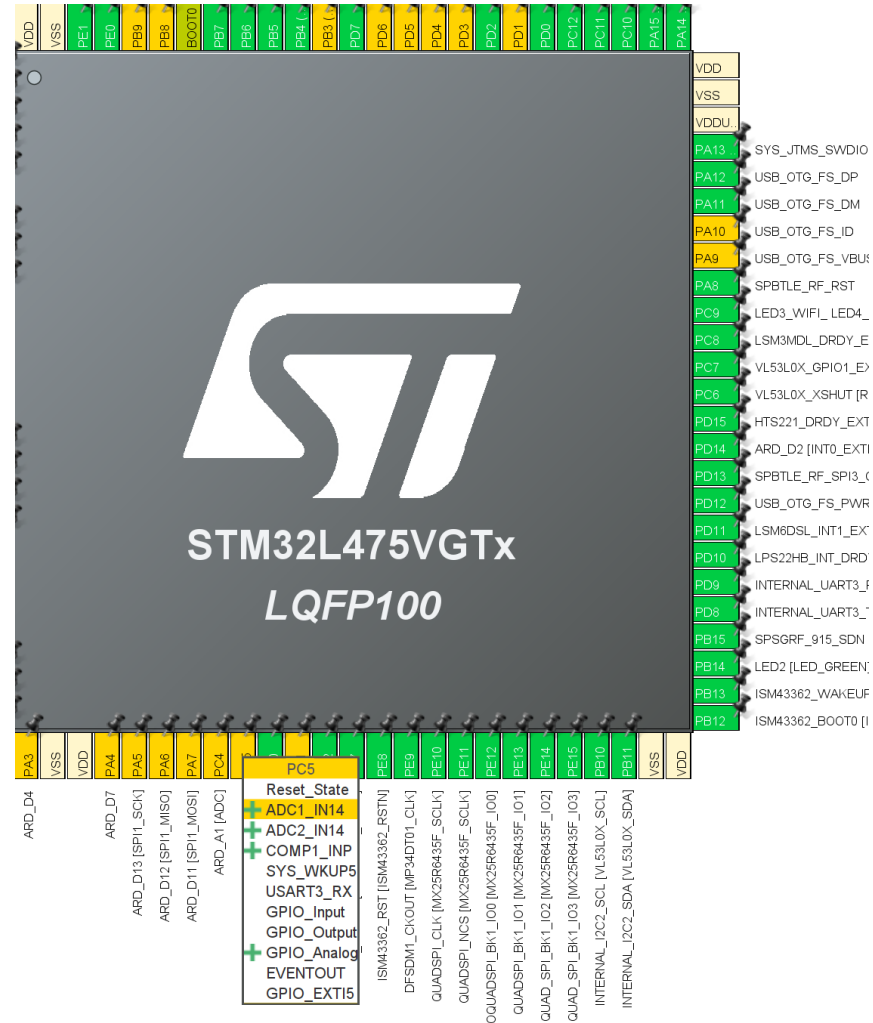
Step 5. Click yes to initialize all peripherals to default



## Step 6. When in .ioc file, click Pinout & Configurations



Step 7. Observe PC5 -> ADC1\_IN14 -> ADR\_A0





## Step 8. Set ADC1, IN14 to Single-ended



## Step 9. Generate code



Step 10. In main.c, find code that initializes the ADC1

The screenshot displays an IDE with the following components:

- Editor:** Shows the file `main.c` with the following C code:
 

```

228 static void MX_ADC1_Init(void)
229 {
230
231     /* USER CODE BEGIN ADC1_Init 0 */
232
233     /* USER CODE END ADC1_Init 0 */
234
235     ADC_MultiModeTypeDef multimode = {0};
236     ADC_ChannelConfTypeDef sConfig = {0};
237
238     /* USER CODE BEGIN ADC1_Init 1 */
239
240     /* USER CODE END ADC1_Init 1 */
241
242     /** Common config
243     */
244     hadc1.Instance = ADC1;
245     hadc1.Init.ClockPrescaler = ADC_CLOCK_ASYNC_DIV1;
246     hadc1.Init.Resolution = ADC_RESOLUTION_12B;
247     hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
248     hadc1.Init.ScanConvMode = ADC_SCAN_DISABLE;
249     hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
250     hadc1.Init.LowPowerAutoWait = DISABLE;
251     hadc1.Init.ContinuousConvMode = DISABLE;
252     hadc1.Init.NbrOfConversion = 1;
253     hadc1.Init.DiscontinuousConvMode = DISABLE;
254     hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;
255     hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
256     hadc1.Init.DMAContinuousRequests = DISABLE;
257     hadc1.Init.Overrun = ADC_OVR_DATA_PRESERVED;
258     hadc1.Init.OversamplingMode = DISABLE;
259     if (HAL_ADC_Init(&hadc1) != HAL_OK)
260     {
261         Error_Handler();
262     }
263
264     /** Configure the ADC multi-mode
265     */
266     multimode.Mode = ADC_MODE_INDEPENDENT;
267     if (HAL_ADCEX_MultiModeConfigChannel(&hadc1, &multimode) != HAL_OK)
268     {
269         Error_Handler();
270     }
271
272     /** Configure Regular Channel

```
- File Explorer:** Shows a project structure with folders like `main` and `main.c`.
- Console:** Empty.
- Memory Stack Analyzer:** Shows a table with columns: Region, Start addr., End addr., Size, Free, Used, Usage (%).

## Step 11. In main.c, find code that initialize the channel IN14

```
Embedded-C-Assignment-6.ioc  main.c ×
255 hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
256 hadc1.Init.DMAContinuousRequests = DISABLE;
257 hadc1.Init.Overrun = ADC_OVR_DATA_PRESERVED;
258 hadc1.Init.OversamplingMode = DISABLE;
259 if (HAL_ADC_Init(&hadc1) != HAL_OK)
260 {
261     Error_Handler();
262 }
263
264 /** Configure the ADC multi-mode
265 */
266 multimode.Mode = ADC_MODE_INDEPENDENT;
267 if (HAL_ADCEx_MultiModeConfigChannel(&hadc1, &multimode) != HAL_OK)
268 {
269     Error_Handler();
270 }
271
272 /** Configure Regular Channel
273 */
274 sConfig.Channel = ADC_CHANNEL_14;
275 sConfig.Rank = ADC_REGULAR_RANK_1;
276 sConfig.SamplingTime = ADC_SAMPLETIME_2CYCLES_5;
277 sConfig.SingleDiff = ADC_SINGLE_ENDED;
278 sConfig.OffsetNumber = ADC_OFFSET_NONE;
279 sConfig.Offset = 0;
280 if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
281 {
282     Error_Handler();
283 }
284 /* USER CODE BEGIN ADC1_Init 2 */
285
286 /* USER CODE END ADC1_Init 2 */
287
288 }
289
290 /**
291  * @brief DFSDM1 Initialization Function
292  * @param None
293  * @retval None
294  */
295 static void MX_DFSDM1_Init(void)
296 {
297
298     /* USER CODE BEGIN DFSDM1_Init 0 */
299
```

## Step 12. Add code to main.c to read from ARD\_A0

```
/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_DFSDM1_Init();
MX_I2C2_Init();
MX_QUADSPI_Init();
MX_SPI3_Init();
MX_USART1_UART_Init();
MX_USART3_UART_Init();
MX_USB_OTG_FS_PCD_Init();
MX_ADC1_Init();
/* USER CODE BEGIN 2 */
uint32_t adcResult;
/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    /* USER CODE END WHILE */

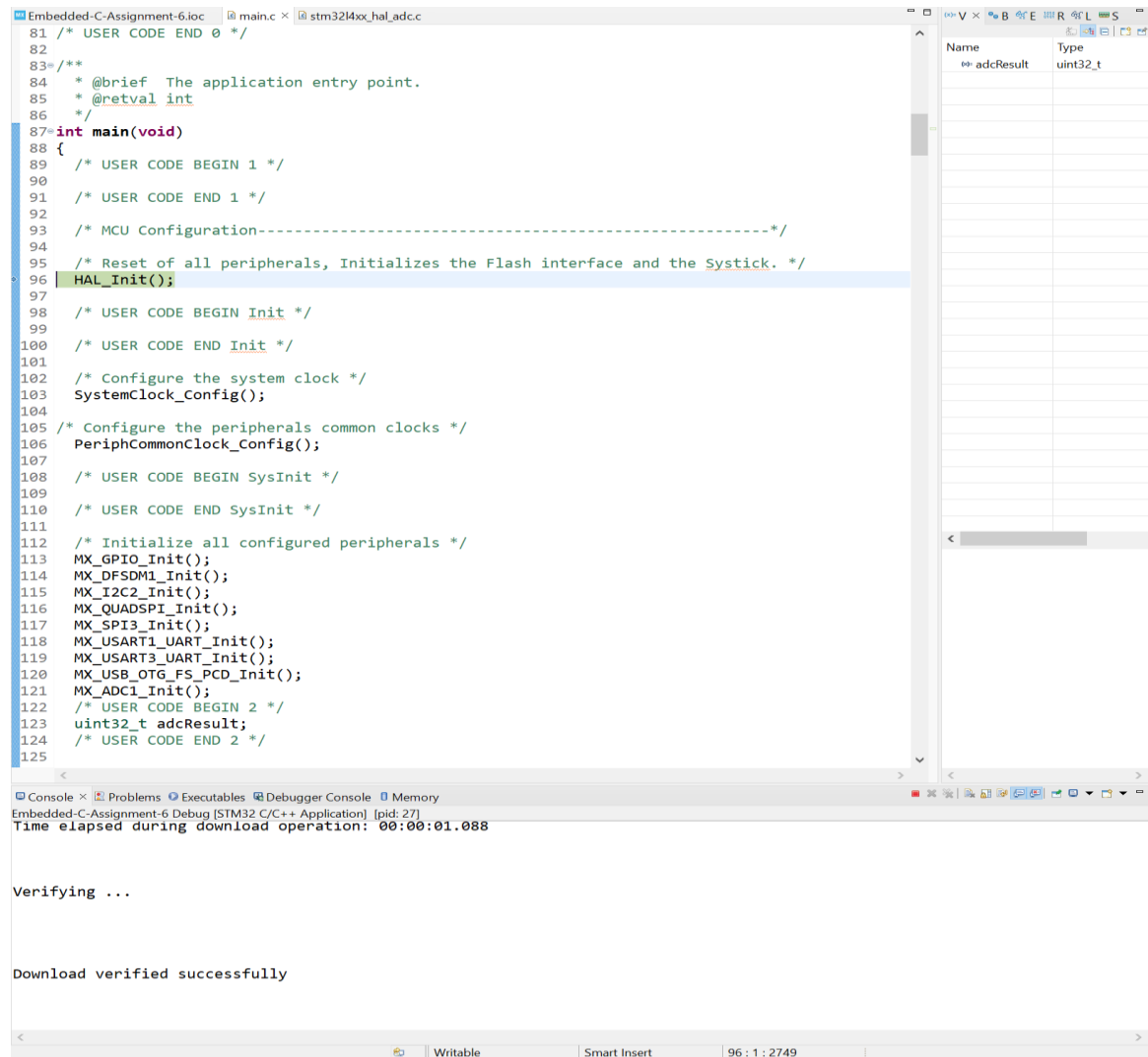
    /* USER CODE BEGIN 3 */
    HAL_ADC_Start(&hadc1);
    HAL_ADC_PollForConversion(&hadc1, 100);|
    adcResult = HAL_ADC_GetValue(&hadc1);

    printf("adcResult: %lu - 0x%lx\n", adcResult, adcResult);

    HAL_ADC_Stop(&hadc1);

    HAL_Delay(1000);
}
/* USER CODE END 3 */
}
```

## Step 13. Build project and run in debug mode



The screenshot shows an IDE with a C source file and a debugger console. The source file, `stm32l4xx_hal_adc.c`, contains the `main` function. The code includes comments for user code sections and initialization of various peripherals. The `HAL_Init()` function call is highlighted. The debugger console at the bottom shows the output of the debug session, indicating successful download and verification.

```
81 /* USER CODE END 0 */
82
83 /**
84  * @brief The application entry point.
85  * @retval int
86  */
87 int main(void)
88 {
89     /* USER CODE BEGIN 1 */
90
91     /* USER CODE END 1 */
92
93     /* MCU Configuration-----*/
94
95     /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
96     HAL_Init();
97
98     /* USER CODE BEGIN Init */
99
100    /* USER CODE END Init */
101
102    /* Configure the system clock */
103    SystemClock_Config();
104
105    /* Configure the peripherals common clocks */
106    PeriphCommonClock_Config();
107
108    /* USER CODE BEGIN SysInit */
109
110    /* USER CODE END SysInit */
111
112    /* Initialize all configured peripherals */
113    MX_GPIO_Init();
114    MX_DFSDM1_Init();
115    MX_I2C2_Init();
116    MX_QUADSPI_Init();
117    MX_SPI3_Init();
118    MX_USART1_UART_Init();
119    MX_USART3_UART_Init();
120    MX_USB_OTG_FS_PCD_Init();
121    MX_ADC1_Init();
122    /* USER CODE BEGIN 2 */
123    uint32_t adcResult;
124    /* USER CODE END 2 */
125}
```

Console × Problems × Executables × Debugger Console × Memory  
Embedded-C-Assignment-6 Debug [STM32 C/C++ Application] [pid: 27]  
Time elapsed during download operation: 00:00:01.088

Verifying ...

Download verified successfully

Writable Smart Insert 96 : 1 : 2749

Step 14, Set the break point, then read the adcResult when nothing connected to ARD\_A0. In my case adcResult = 916

The screenshot shows an IDE with a C program for STM32 ADC configuration. The code includes initialization for various peripherals (GPIO, DFSDM, I2C2, QUADSPI, SPI3, USART1, USART3, USB OTG FS PCD, and ADC1) and a while loop that repeatedly reads the ADC value and prints it. A breakpoint is set at line 139, where the ADC is stopped. The variable window on the right shows the current value of adcResult as 1421.

```
112 /* Initialize all configured peripherals */
113 MX_GPIO_Init();
114 MX_DFSDM1_Init();
115 MX_I2C2_Init();
116 MX_QUADSPI_Init();
117 MX_SPI3_Init();
118 MX_USART1_UART_Init();
119 MX_USART3_UART_Init();
120 MX_USB_OTG_FS_PCD_Init();
121 MX_ADC1_Init();
122 /* USER CODE BEGIN 2 */
123 uint32_t adcResult;
124 /* USER CODE END 2 */
125
126 /* Infinite loop */
127 /* USER CODE BEGIN WHILE */
128 while (1)
129 {
130     /* USER CODE END WHILE */
131
132     /* USER CODE BEGIN 3 */
133     HAL_ADC_Start(&hadc1);
134     HAL_ADC_PollForConversion(&hadc1, 100);
135     adcResult = HAL_ADC_GetValue(&hadc1);
136
137     printf("adcResult: %lu - 0x%lx\n", adcResult, adcResult);
138
139     HAL_ADC_Stop(&hadc1);
140
141     HAL_Delay(1000);
142 }
143 /* USER CODE END 3 */
144 }
145
146 /**
147  * @brief System Clock Configuration
148  * @retval None
149  */
150 void SystemClock_Config(void)
151 {
152     RCC_OscInitTypeDef RCC_OscInitStruct = {0};
153     RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
154
155     /** Configure the main internal regulator output voltage
156     */
```

Name	Type	Value
adcResult	uint32_t	1421

Console x Problems x Executables x Debugger Console x Memory  
Embedded-C-Assignment-6 Debug [STM32 C/C++ Application] [pid: 27]

Step 15. Connect ADR\_A0 to GND, then `adcResult = 0`

The image shows a screenshot of an IDE with two main panels. The left panel displays C code for configuring and using an ADC on an STM32 microcontroller. The right panel shows a 'Variables' window with a table of current variable values.

**Code Snippet (Left Panel):**

```
112 /* Initialize all configured peripherals */
113 MX_GPIO_Init();
114 MX_DFSDM1_Init();
115 MX_I2C2_Init();
116 MX_QUADSPI_Init();
117 MX_SPI3_Init();
118 MX_USART1_UART_Init();
119 MX_USART3_UART_Init();
120 MX_USB_OTG_FS_PCD_Init();
121 MX_ADC1_Init();
122 /* USER CODE BEGIN 2 */
123 uint32_t adcResult;
124 /* USER CODE END 2 */
125
126 /* Infinite loop */
127 /* USER CODE BEGIN WHILE */
128 while (1)
129 {
130     /* USER CODE END WHILE */
131
132     /* USER CODE BEGIN 3 */
133     HAL_ADC_Start(&hadc1);
134     HAL_ADC_PollForConversion(&hadc1, 100);
135     adcResult = HAL_ADC_GetValue(&hadc1);
136
137     printf("adcResult: %lu - 0x%x\n", adcResult, adcResult);
138
139     HAL_ADC_Stop(&hadc1);
140
141     HAL_Delay(1000);
142 }
143 /* USER CODE END 3 */
144 }
145
146 /**
147  * @brief System Clock Configuration
148  * @retval None
149  */
150 void SystemClock_Config(void)
151 {
152     RCC_OscInitTypeDef RCC_OscInitStruct = {0};
153     RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
154
155     /** Configure the main internal regulator output voltage
156     */
```

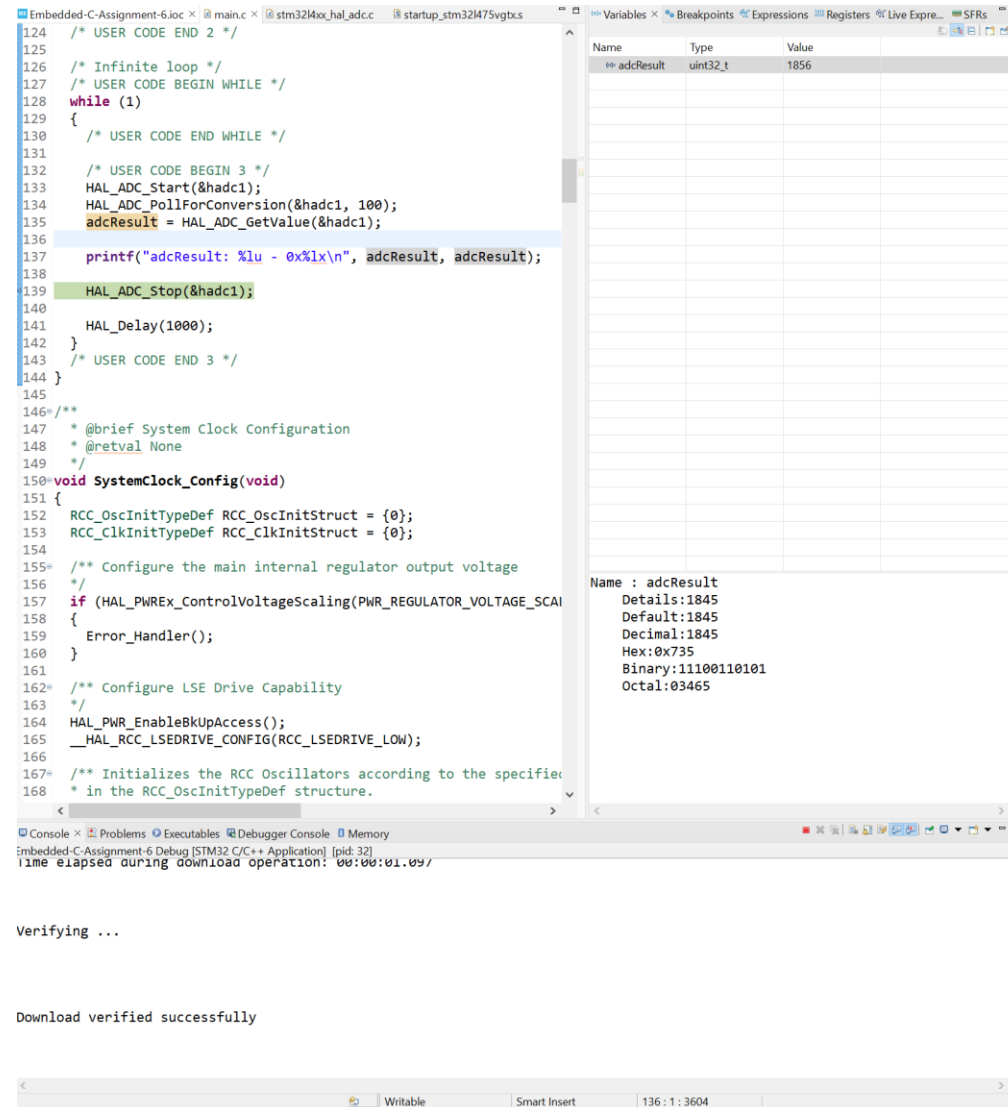
**Variables Window (Right Panel):**

Name	Type	Value
adcResult	uint32_t	0

The bottom status bar indicates the project is 'embedded-C-Assignment-6 Debug (STM32 C/C++ Application)' and the current line is 271.



Step 16. Connect ADR\_A0 to 1.5VDC. Then adcResult = 1856



```
124  /* USER CODE END 2 */
125
126  /* Infinite loop */
127  /* USER CODE BEGIN WHILE */
128  while (1)
129  {
130    /* USER CODE END WHILE */
131
132    /* USER CODE BEGIN 3 */
133    HAL_ADC_Start(&hadc1);
134    HAL_ADC_PollForConversion(&hadc1, 100);
135    adcResult = HAL_ADC_GetValue(&hadc1);
136
137    printf("adcResult: %lu - 0x%x\n", adcResult, adcResult);
138
139    HAL_ADC_Stop(&hadc1);
140
141    HAL_Delay(1000);
142  }
143  /* USER CODE END 3 */
144 }
145
146 /**
147  * @brief System Clock Configuration
148  * @retval None
149  */
150 void SystemClock_Config(void)
151 {
152     RCC_OscInitTypeDef RCC_OscInitStruct = {0};
153     RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
154
155     /** Configure the main internal regulator output voltage
156     */
157     if (HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1))
158     {
159         Error_Handler();
160     }
161
162     /** Configure LSE Drive Capability
163     */
164     HAL_PWR_EnableBkUpAccess();
165     __HAL_RCC_LSEDRIVE_CONFIG(RCC_LSEDRIVE_LOW);
166
167     /** Initializes the RCC Oscillators according to the specified values
168     * in the RCC_OscInitTypeDef structure.
169     */
170 }
```

Name	Type	Value
adcResult	uint32_t	1856

Name : adcResult  
Details:1845  
Default:1845  
Decimal:1845  
Hex:0x735  
Binary:11100110101  
Octal:03465

Console × Problems × Executables × Debugger Console × Memory  
Embedded-C-Assignment-6 Debug [STM32 C/C++ Application] [pid: 32]  
Time elapsed during download operation: 00:00:01.09/  
  
Verifying ...  
  
Download verified successfully  
  
Writable Smart Insert 136 : 1 : 3604