|  |  |  |
| --- | --- | --- |
|  | MECHATRONICS DEPARTMENT  TE2024 – MICROCONTROLLER LABORATORY  *LAB #2* – *µVision 5 IDE* | |
| Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  ID: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  ID: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

**Instructions:**

1. Follow sections A, B and C (in that order).
2. When writing the report, after the cover, the elements in the following sections must appear in this order:
   1. C.1.18
   2. C.1.21
   3. C.1.22
   4. C.2.5
   5. C.2.6
   6. C.2.7
3. Remember that those points are results. Those results are the culmination of the work you do on the laboratory. Before each one of them you must answer as concise as possible the following (do not place these questions in the report):
   1. What were you trying to accomplish?
   2. Which obstacles you found in the way?
   3. How did you sort them?
   4. How your final program works?
   5. What results does it throw?
   6. How do you interpret the obtained results?
   7. What can you conclude of the overall process?
4. After you finish writing the report, name it as TE2024**<X>**.**<Y>**\_E**<Z>**\_L**<W>**.docx.

<X>: Bachelor ID: ISD, IMT

<Y>: Group number (not required for IMT. In this case, please ignore the dot).

<Z>: Team number

<W>: Lab number

1. Create a folder named **Projects\** and move inside your Keil-MDK project for each one of the final systems.
2. Inside a **.zip file,** include both the report and the **Projects\** folder.
3. Upload it to the assigned place indicated by your instructor.
4. APPLICATIONS SETUP

**Objective:** Install on at least one personal computer the required software for both theory and lab classes.

**Instructions:** Read and execute the steps in the following subsections.

A.1 – Keil MDK

1. Run the **Keil-MDK installer** file as *Administrator*.

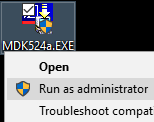


Figure A.1.1. Opening the Keil-MDK Installer

1. Click on *Next* to skip the **Welcome window**.

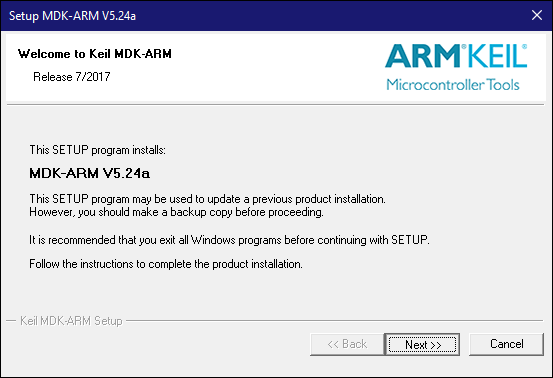


Figure A.1.2. Keil-MDK Installer Welcome window

1. Accept the **License Agreement** and click on *Next*.

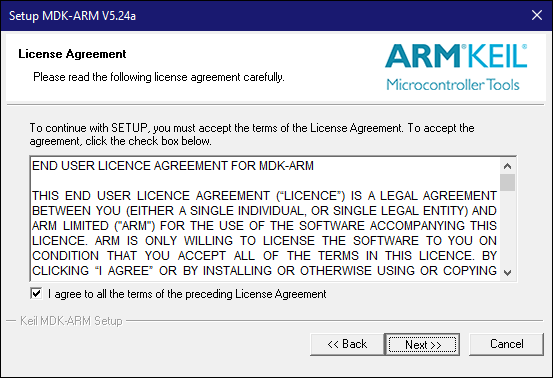


Figure A.1.3. Keil-MDK License Agreement window

1. Enter a location (preferably a folder named *\TE2023\Keil\_v5* at the top of the C: drive) where to install the IDE. Click on *Next*.

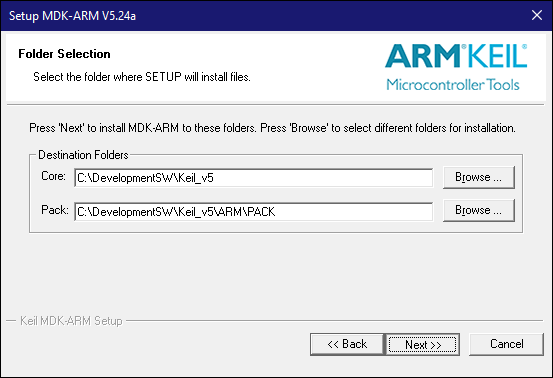


Figure A.1.4. Keil-MDK installation location

1. Enter contact information and then click on *Next*.

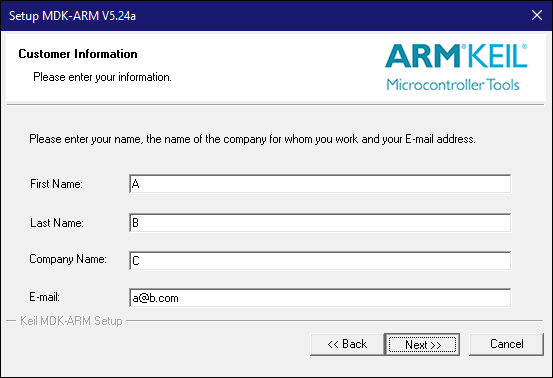


Figure A.1.5. Keil-MDK customer information

1. Wait until the installation is over.

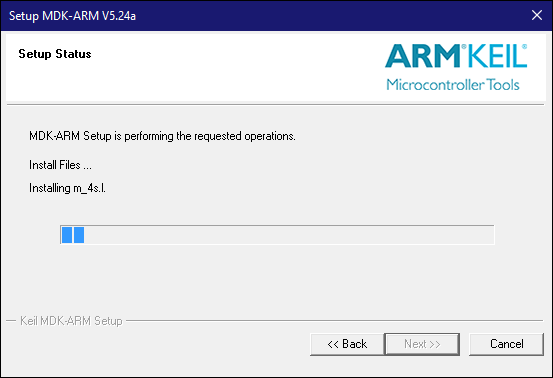


Figure A.1.6. Keil-MDK Installation

1. When prompted about the **ULINK****drivers**, install none of them.

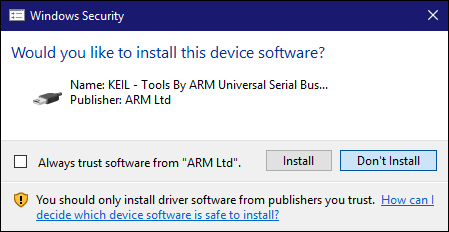


Figure A.1.7. ULINK Drivers Installation

1. Untick all the options once the installation is over. Click on *Finish*.

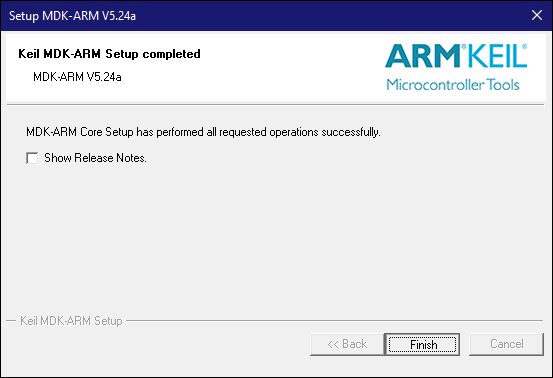


Figure A.1.8. Finishing Keil-MDK Installation

1. At this point, you should have an active Internet connection. When the Keil Pack Installer window prompts, click on *OK*. Wait until the program loads all available packages*.*

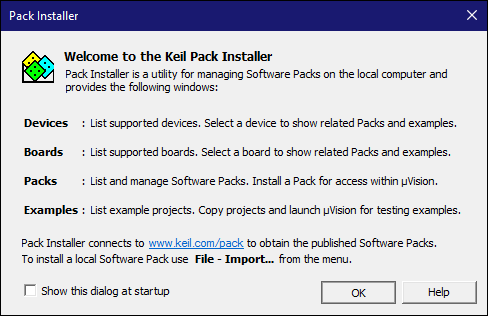


Figure A.1.9. Keil-MDK Pack Installer Welcome window

1. Once all the packages are loaded, in the left window click in *STMicroelectronics 🡪 STM32F1 Series 🡪 STM32F103.*

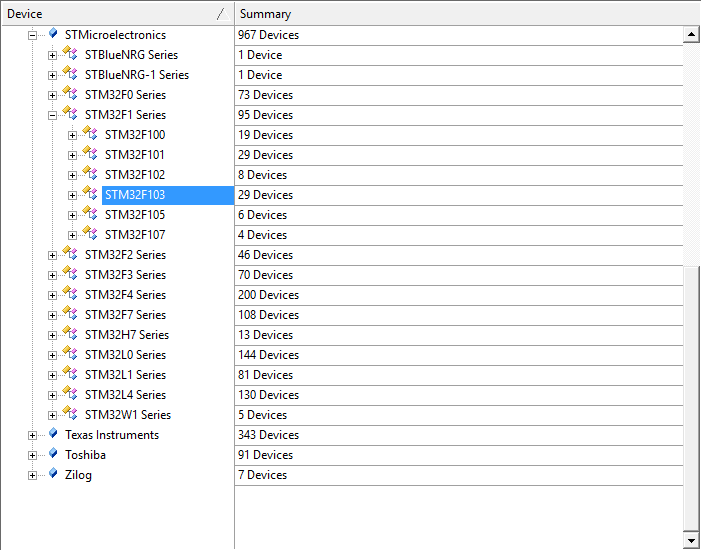


Figure A.1.10. Keil-MDK Pack Installer Available Packages

1. On the right window click on *Keil::STM32F1xx\_DFP*, *ARM::CMSIS*, *Keil::ARM\_Compiler*, *Keil::MDK-Middleware* in order to install them. Once done, you should have something similar to Figure A.1.11. Finally, close the window.

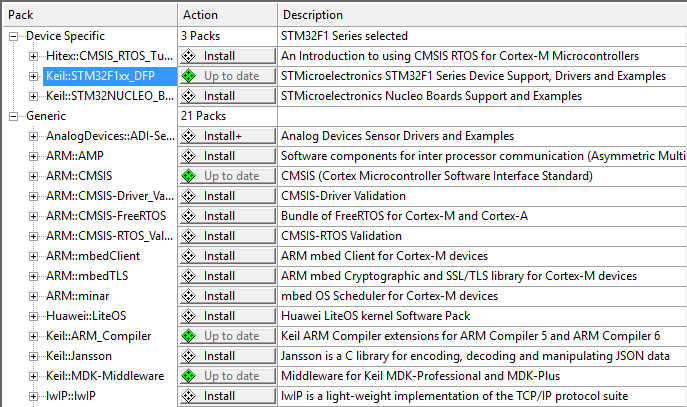


Figure A.1.11. STM32F103 Related Packages

1. Open the **Keil µVision 5** program. Set *four* in *Edit 🡪 Configuration 🡪 Editor 🡪 C/C++ Files 🡪 Tab size*. Click on *OK*. Then close the window.

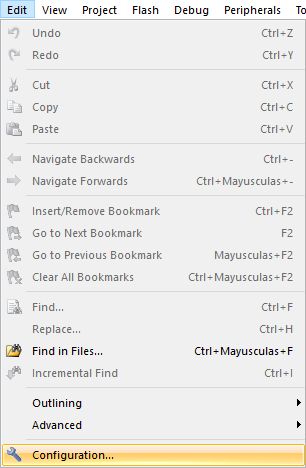


Figure A.1.12. STM32F103 Related Packages

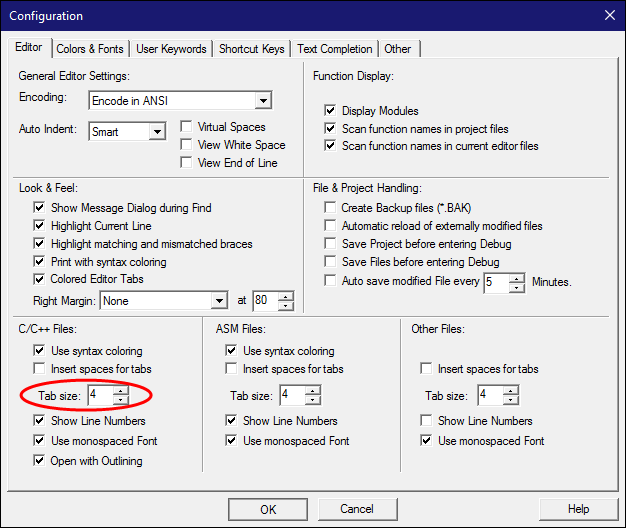


Figure A.1.13. STM32F103 Related Packages

A.2 – ST-LINK Utility and Drivers

1. Run the **STM32 ST-LINK utility** installer as administrator.

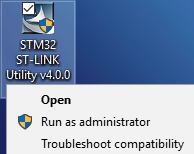


Figure A.2.1. Running STM32 ST-LINK Utility installer as Administrator

1. Click on *Next* to skip the **Welcome window**.

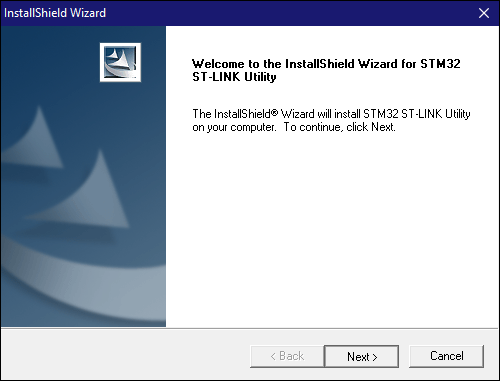


Figure A.2.2. ST-LINK Utility installer Welcome window

1. Accept the **License Agreement** and click on *Next*.

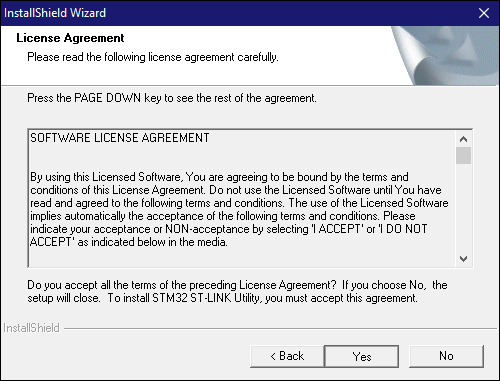


Figure A.2.3. ST-LINK Utility installer License Agreement

1. Enter a location (preferably in a folder named *TE2023* at the top of the *C:* drive) where to install the IDE. Click on *Next*.

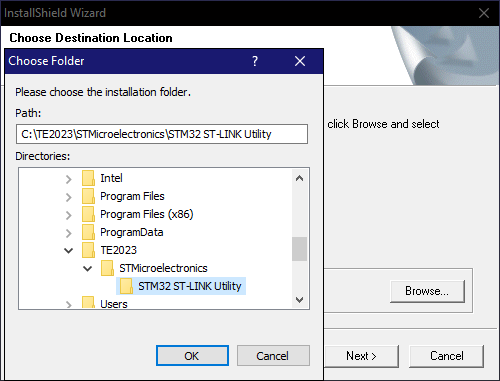


Figure A.2.4. ST-LINK Utility installation location

1. Wait until the installation is over.

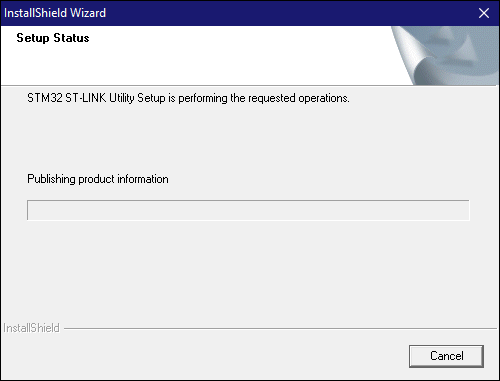


Figure A.2.5. ST-LINK Utility installation

1. When prompted by the driver installer, click on *Next*. When **Windows** asks for authorization to the driver installation, allow it.

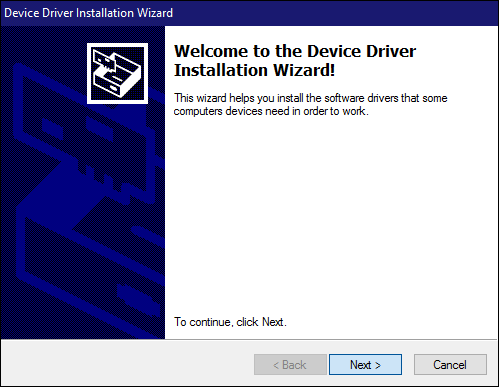


Figure A.2.6. ST-LINK Driver installation Welcome window

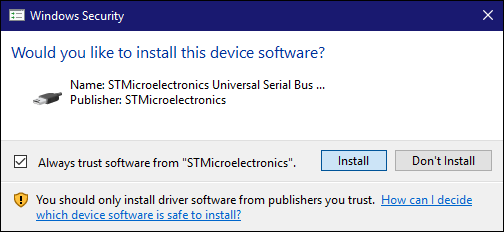


Figure A.2.7. ST-LINK Driver setup authorization

1. Wait until the driver is installed, then click on *Finish*.



Figure A.2.8. ST-LINK Driver installation Success window

1. Click on *Finish* to exit the **ST-LINK Utility Installation wizard**.

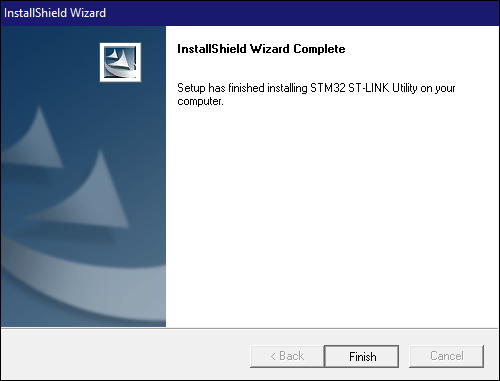


Figure A.2.9. ST-LINK Utility installation Success window

A.3 – Text Editor Installation

1. Execute **Sublime Text 2 installer** as *Administrator.*

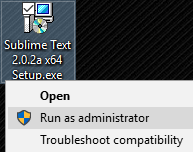


Figure A.3.1. Running Sublime Text 2 installer

1. In the **Welcome window** click on *Next*.

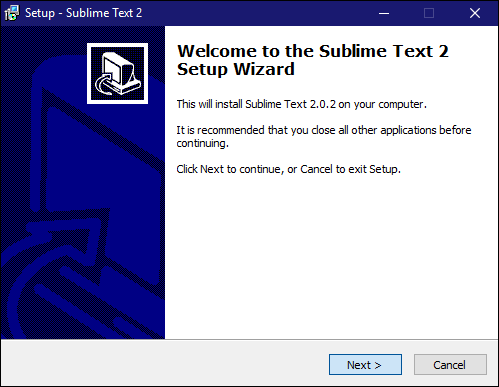


Figure A.3.2. Sublime Text 2 Installer Welcome window

1. In the **Installer Directory Window,** select the default route.

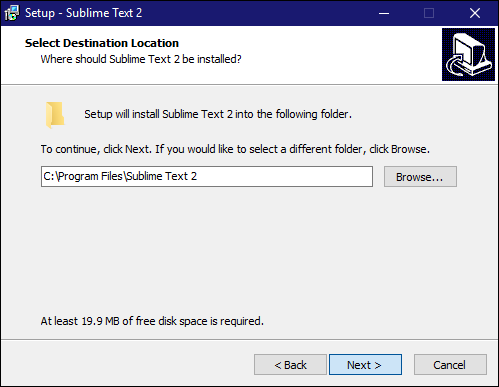


Figure A.3.3. Sublime Text 2 installation Location window

1. Tick the *Add to explorer context menu* option.

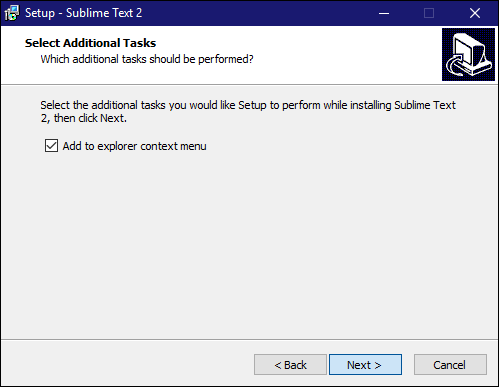


Figure A.3.4. Sublime Text 2 installation Additional Tasks window

1. Once the installer is ready to install, click on the *Install* button.

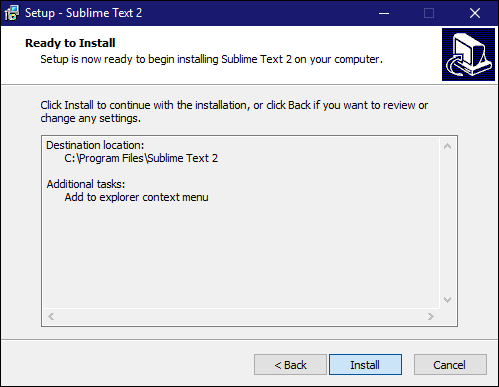


Figure A.3.5. Sublime Text 2 installation Ready to Install window

1. Wait until the installation is over. Then click on the *Finish* button.

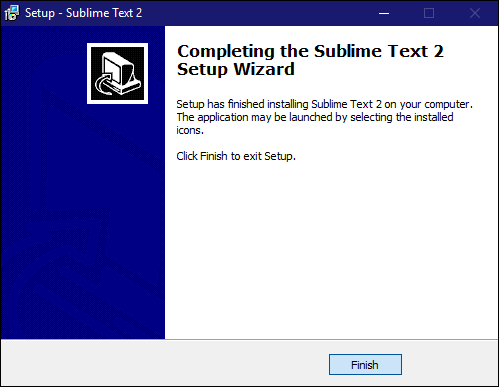


Figure A.3.5. Sublime Text 2 installation Success window

1. Open the **Sublime Text 2** program. Go to *Preferences 🡪 Color Scheme* and select the option of your preference. You can disable the recent file history by setting *Preferences 🡪 Settings-Default 🡪 "hot\_exit"* and *Preferences 🡪 Settings-Default 🡪 "remember\_open\_files"* to *false*.

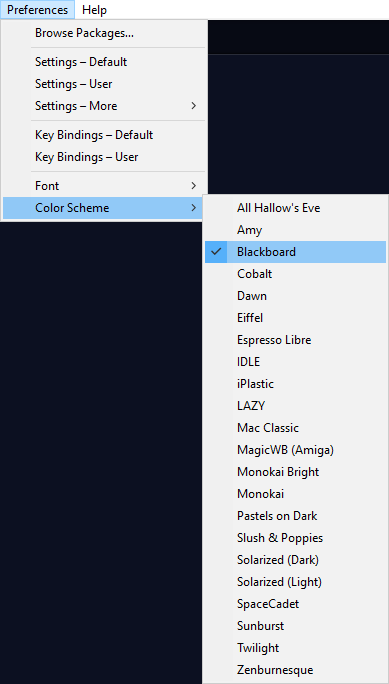


Figure A.3.6. Sublime Text 2 Color Schemes

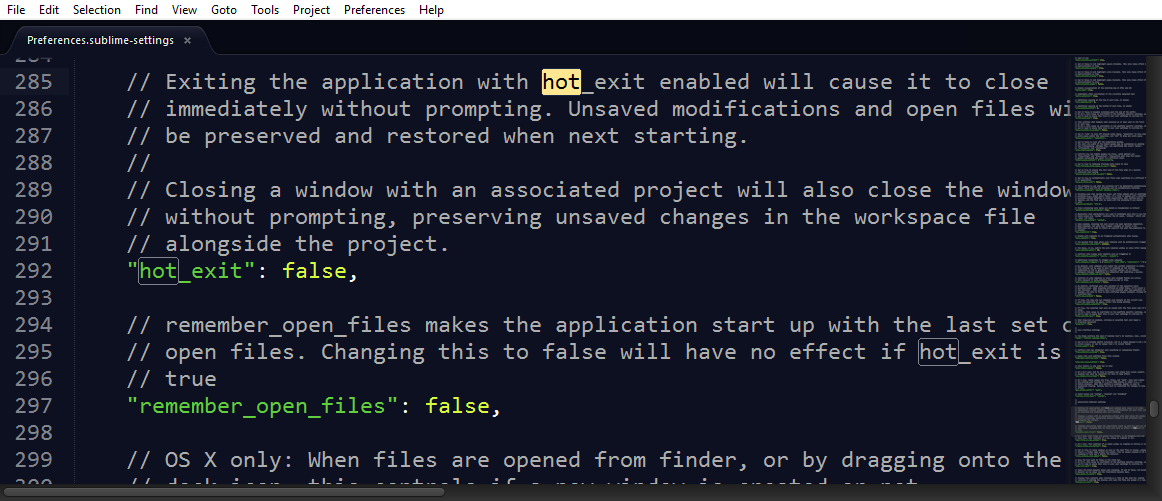


Figure A.3.7. Sublime Text 2 Settings

1. KEIL-MDK PROJECTS

**Objective:** Learn how to create Keil-MDK projects that will allow the microcontroller programming.

**Instructions:** Read and follow the steps in the following subsections.

B.1 – Project Creation

1. Open the **Keil-MDK** program. Go to *Project 🡪 New µVision Project.*

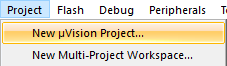


Figure B.1.1. Creating a new project

1. Create a new folder named *F103\_Template* and on it create the new project with the name *F103\_Template*.

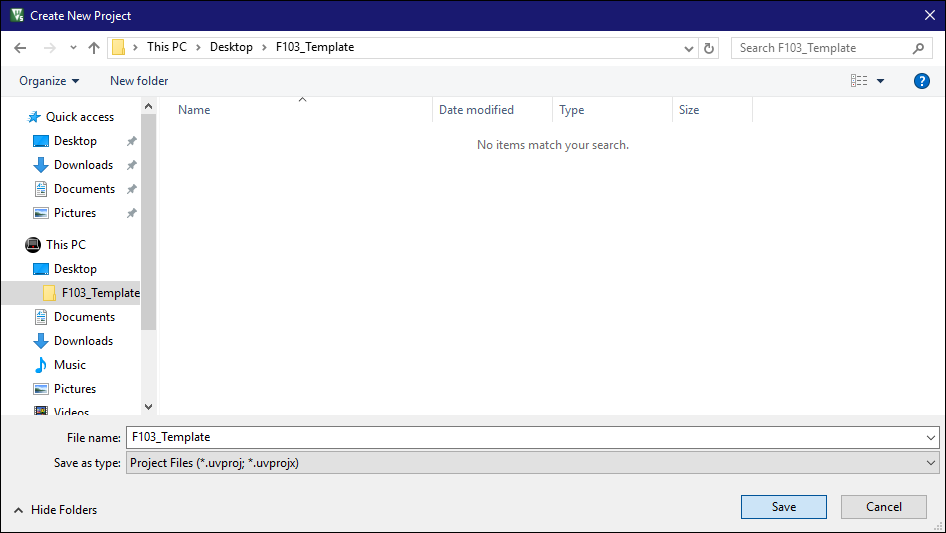


Figure B.1.2. Location of the new project

1. Select the following device: *STMicroelectronics 🡪 STM32F1 Series 🡪 STM32F103 🡪 STM32F103C8*.

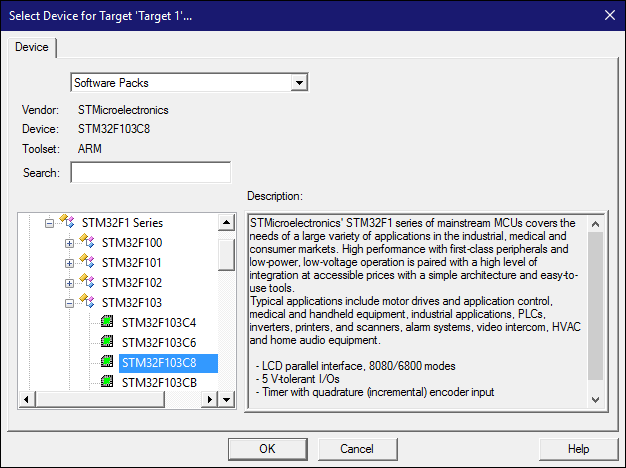


Figure B.1.3. Selecting target of new project

1. Tick the options *CMSIS 🡪 CORE* and *Device 🡪 Startup*.

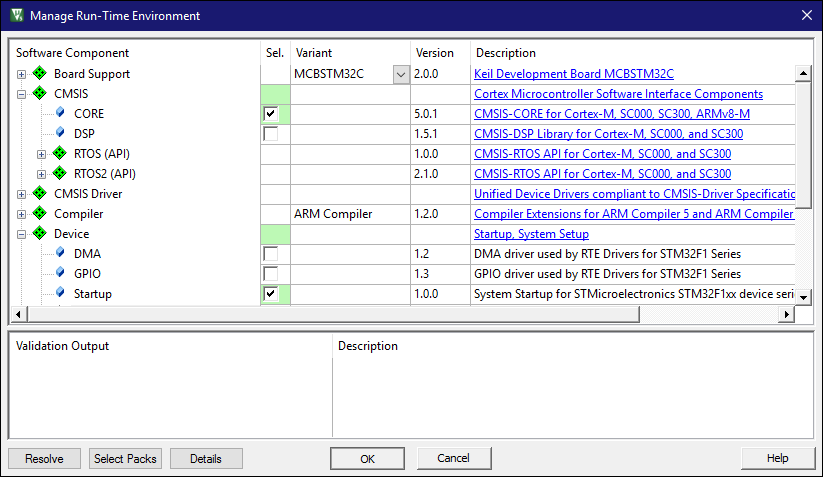


Figure B.1.4. Adding startup libraries to new project

1. Left-click and then release **Target 1**. Change its name to *F103\_Template*.

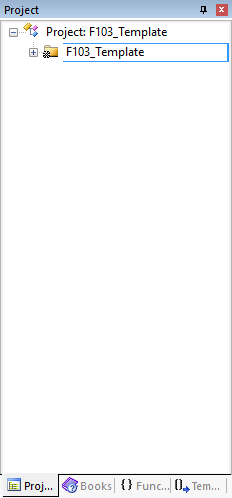


Figure B.1.5. Renaming the new project

1. Expand the project’s root folder and rename the folder **Source Group 1** to *Source*. Then close the Keil-MDK program.

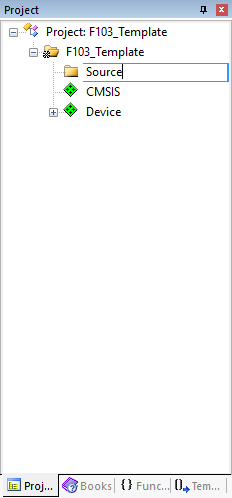


Figure B.1.6. Renaming Source Group

1. Make sure you can view file extensions by ticking *File name extensions* option in the *View* tab at the top of the file explorer.

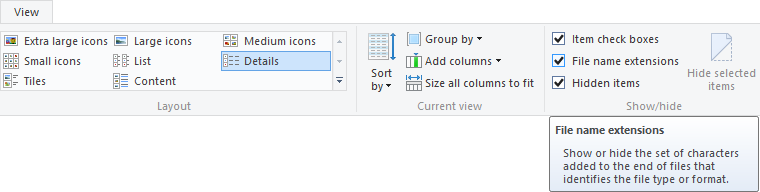


Figure B.1.7. Enabling file name extensions in Windows Explorer

1. Delete the **DebugConfig\**, **Listings\**, **Objects\** folders and the **F103\_Template.uvguix.<USER>** file. Then create a *Libraries\*, an *Output\* and *Simulation\* folders. Additionally, create a file named *main.c*. For this last file right-click on the background of the explorer window and then follow *New 🡪 Text Document*. Then rename the file to *main.c*. Open the file with the text editor. Substitute the content of Listing B.1.1 into this recently created file. Make sure the last line has an *<ENTER>*. If you installed the Sublime Text 2 editor, you can open the file by right clicking it and selecting *Open with Sublime Text 2*.

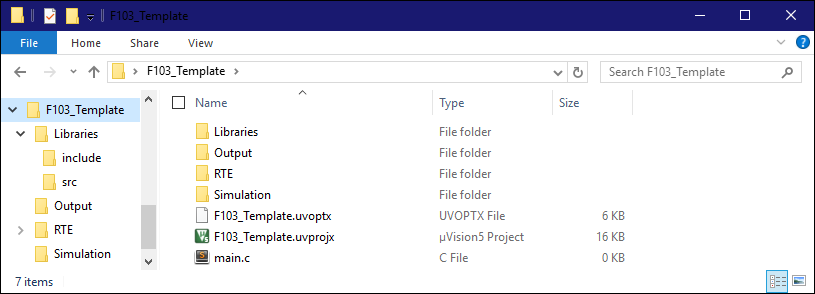


Figure B.1.8. Updated contents of new project folder

Listing B.1.1. Content of **main.c** file

|  |
| --- |
| #include <stm32f10x.h>  #include "delay.h"  int main( )  {  // Initialization code    while(1)  {    // Recurrent code    }  } |



Figure B.1.9. Editing **main.c** file with text editor

1. In the **Libraries\** folder, create two folders named *src\* and *include\*.

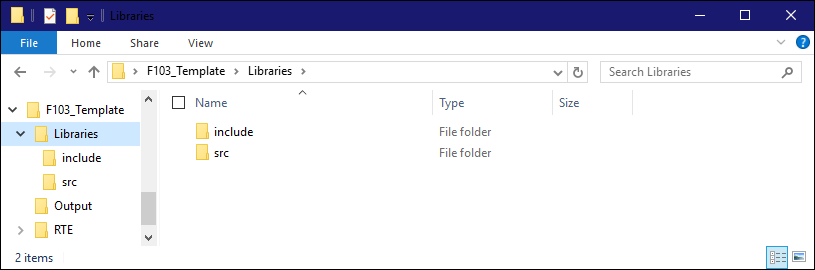


Figure B.1.10. **Libraries\** folder contents

1. Inside the **src\** folder, create two files named *\_delay\_ms.s* and *\_delay\_us.s*. Open both files with the text editor and then replace them with the content of listings B.1.2 and B.1.3. Make sure the first line has a *<TAB>* and that the last line has an *<ENTER>*.

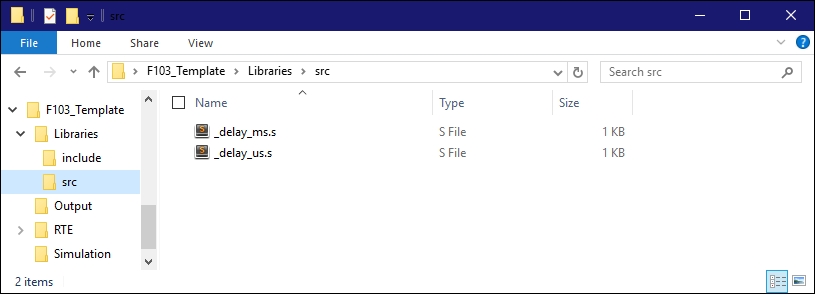


Figure B.1.11. **src\** folder contents

Listing B.1.2. Content of **\_delay\_ms.s** file

|  |
| --- |
| AREA delay, CODE  EXPORT \_delay\_ms ;make \_delay\_ms visible  ALIGN  \_delay\_ms PROC    PUSH {r4, lr}  outer MOV r1, #10282  inner SUBS r1, r1, #1  BNE inner  SUBS r0, r0, #1  BNE outer  POP {r4, pc}  ENDP  END |

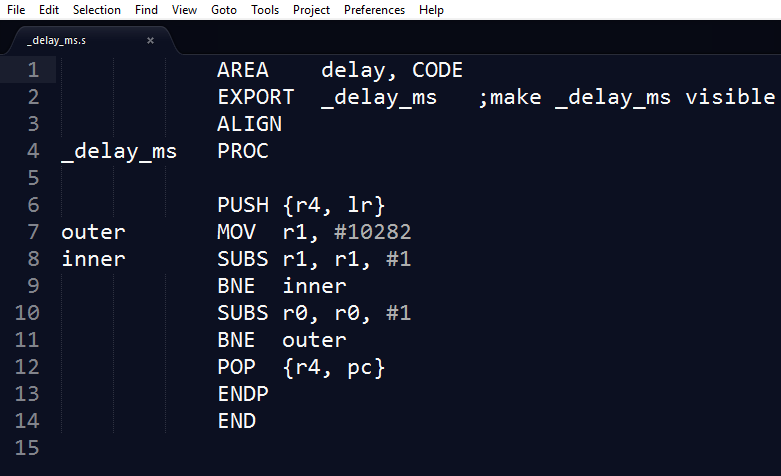


Figure B.1.12. Editing **\_delay\_ms.s** file with text editor

Listing B.1.3. Content of **\_delay\_us.s** file

|  |
| --- |
| AREA delay, CODE  EXPORT \_delay\_us ;make \_delay\_us visible  ALIGN  \_delay\_us PROC    PUSH {r4, lr}  outer MOV r1, #4  inner SUBS r1, r1, #1  BNE inner  SUBS r0, r0, #1  BNE outer  POP {r4, pc}  ENDP  END |

1. Inside the **include\** folder, create a file named *delay.h*. Open the file with the text editor. Substitute the content of Listing B.1.4 into this recently created file. Make sure the last line has an *<ENTER>*.

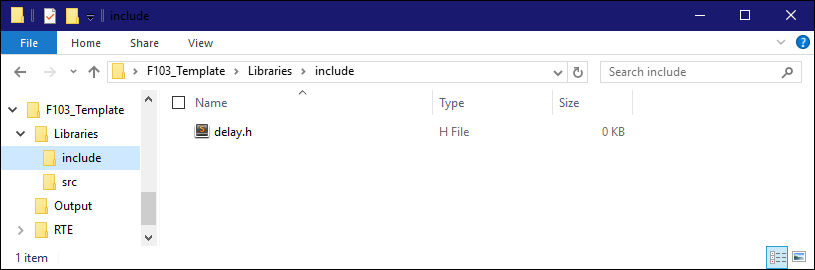


Figure B.1.13. **include\** folder contents

Listing B.1.4. Content of **delay.h** file

|  |
| --- |
| extern void \_delay\_ms(uint32\_t ms);  extern void \_delay\_us(uint32\_t ms); |

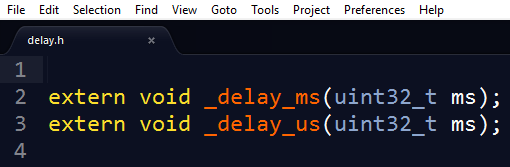


Figure B.1.14. Editing **delay.h** file with text editor

1. Open the file **F103\_Template.uvprojx**. Right-click on the project’s name and select *Options for Target ‘F103 Template’*.

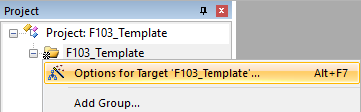


Figure B.1.15. Opening project’s options menu

1. In the **Target** 🡪 **Xtal (MHz):** set a value of *72.0*.

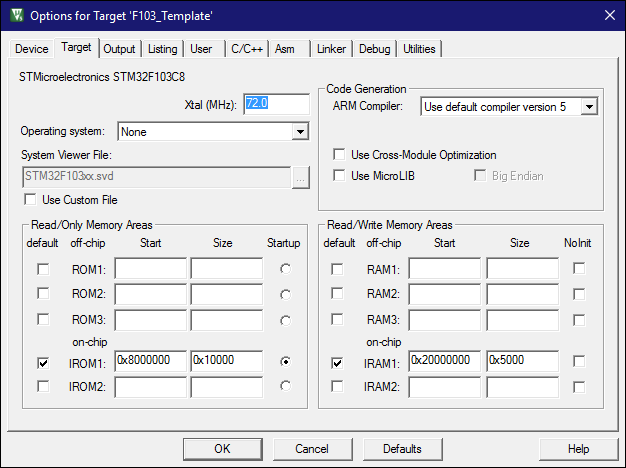


Figure B.1.16. Changing Xtal frequency

1. Click on *Output tab*. Tick the *Create HEX file option*. In the same tab click on *Select Folder for Objects* and make it point to the **Output\** folder inside the project’s root folder. Repeat for *Listing tab 🡪 Select Folder for Listings*.

|  |  |
| --- | --- |
|  |  |
| Figure B.1.17. Changing location for Objects | Figure B.1.18. Changing location for Listings |

1. In the **C/C++** **tab 🡪 Include Paths** parameter, add the *.\Libraries\include* folder.

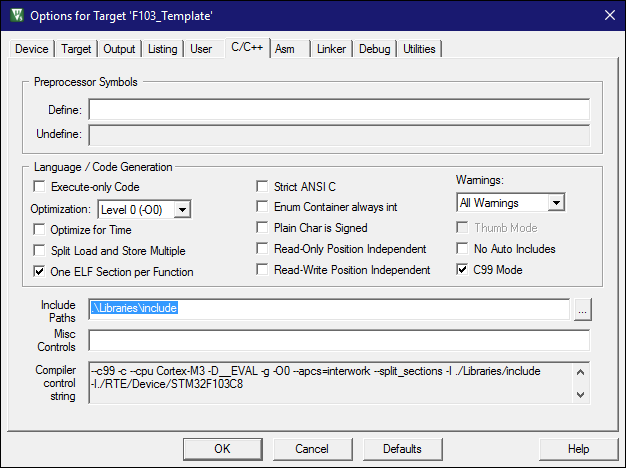


Figure B.1.19. Adding project’s include folder parameter

1. In the **Debug tab 🡪 Use,** select *ST-Link Debugger* as parameter. In the **Utilities tab,** check that the *Use Debug Driver* box is marked and then return to the **Debug tab**.

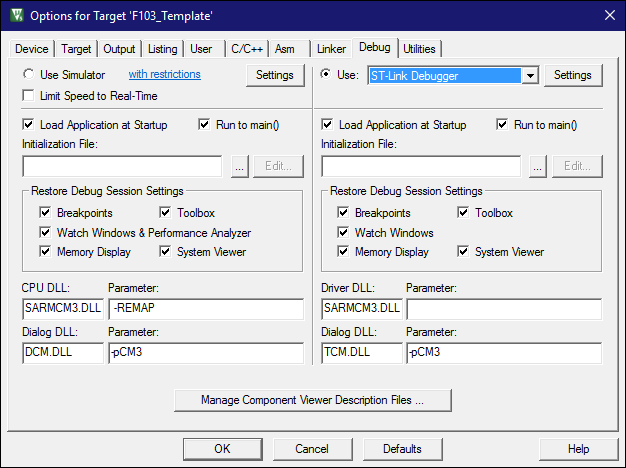


Figure B.1.20. Specifying project’s debugger

1. Click on the **Settings** button. Then go to the **Flash Download tab**.Tick the *Program, Verify* and *Reset and Run* boxes. Select the *Erase Full Chip* option. After that, click on the **Add** button. Select the *STM32F10x Flash Options* and then on then click on the **Add** button. Finally click on the **OK** button in all the open windows of the project’s options.

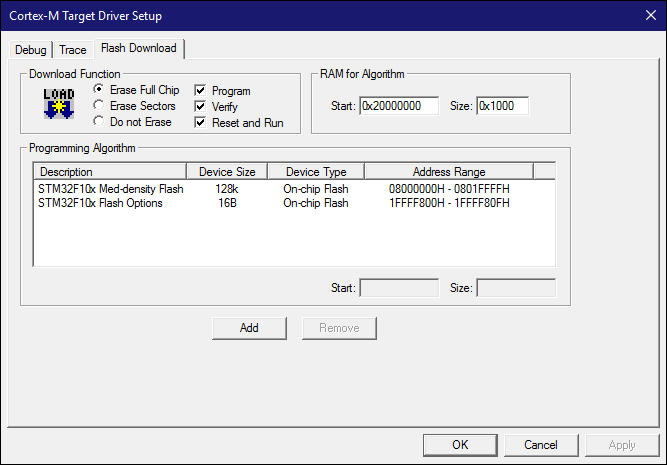


Figure B.1.21. Flash Download options

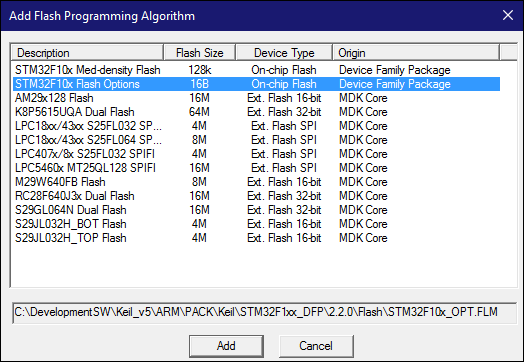


Figure B.1.22. Flash Programming algorithms

1. Right-click on the **Source** folder in the project’s tree. Select *Add Existing Files to Group ‘Source’*. Select (in this order) the **main.c** file in the project’s root folder and then all the files in the project’s **Libraries\src\** folder. The project’s tree should show a structure similar to the one in Figure B.1.24.

|  |  |
| --- | --- |
|  |  |
| Figure B.1.23. Adding files to the ‘Source’ group | Figure B.1.24. Project tree after files addition |

1. Click on the **Rebuild** button in the upper toolbar. In the **Console window** (at the bottom of the Keil program) you should get an output like the one in Figure B.1.26.

|  |  |
| --- | --- |
|  |  |
| Figure B.1.25. Rebuilding project | Figure B.1.26. Console output after rebuilding project |

1. Inside the **Simulation\** folder, create a Proteus schematic file named *F103\_Template.pdsprj* that contains the circuit in Figure B.1.27. The terminal with an arrow is found in *Right-cli*ck 🡪 *Terminals 🡪 Power*.

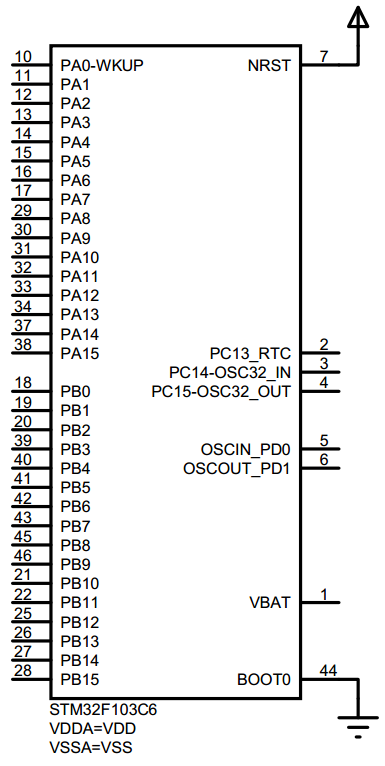


Figure B.1.27. Proteus circuit for project template

1. Make sure that the **STM32F103C6** component should has as properties the parameters depicted in Figure B.1.28. To edit the hidden pins, click on the **Hidden Pins** button. Once done, save the file and then close the program.

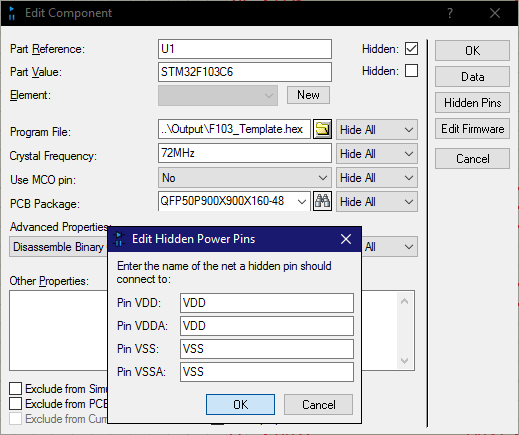


Figure B.1.28. STM32F103C6 Component Properties

1. Erase the following:

* All the contents inside the **Output\** folder.
* All the contents inside the **Simulation\**folder except the **F103\_Template.pdsprj** file.
* The **F103\_Template.uvguix.<USER>** file.

You should get a result similar like the one depicted in Figure B.1.29. Finally, compress all the contents in the folder into a file named *F103\_Template.zip*. Backup this file for future use (unless you wish to repeat all the steps of this section per each project you upload to the microcontroller).

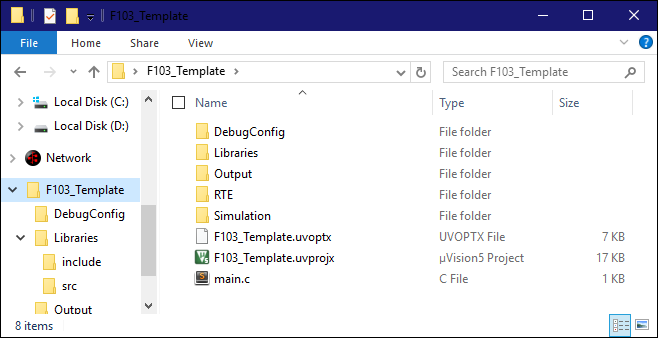


Figure B.1.29. F103\_Template folder

B.2 – Project Renaming

1. In the following steps section, the new name will be *L2*.Decompress the **F103\_Template.zip** file. Rename the folder as *L2.* Open the folder and rename the **.uvprojx** and **.uvoptx** files.

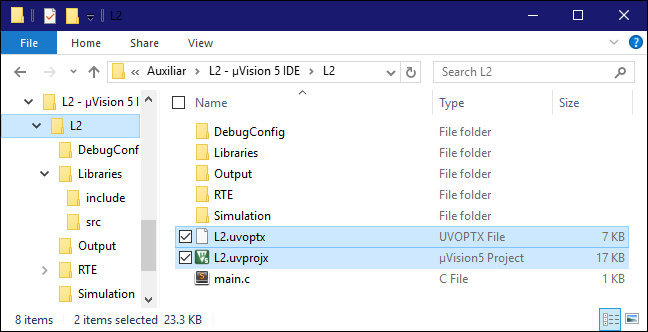


Figure B.2.1. F103\_Template folder renaming

1. Open the ***L2.uvprojx*** file and then rename the project’s folder as *L2*. To do this, remember to left-click once and then release the name aside the folder you want to rename.

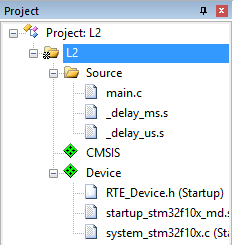


Figure B.2.2. Project folder renaming

1. Right-click on the ***L2*****folder** and then go to **Options for Target ‘L2’ 🡪 Output tab 🡪 Name of Executable** and rename it as *L2.* Click on **OK** and then rebuild the whole project.

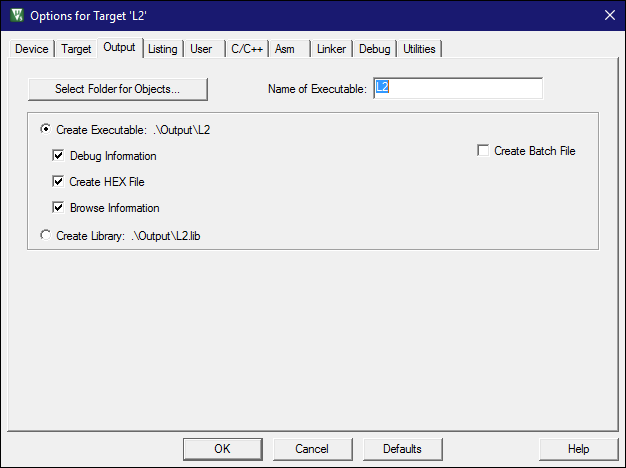


Figure B.2.3. Executable renaming

1. Go to the **Simulation** folder and rename the **F103\_Template.pdsprj** to *L2.pdsprj*.

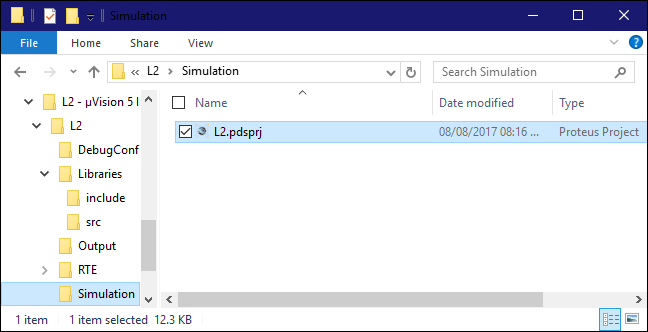


Figure B.2.4. Simulation file renaming

1. Open the simulation file and change the microcontroller program file to *..\Output\L2.hex*. Click on **OK** and finally, save the file.

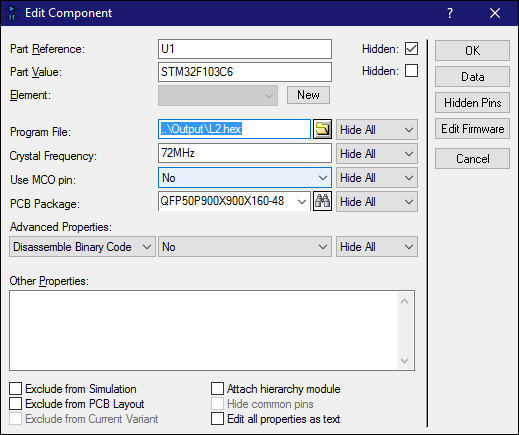


Figure B.2.5. Indicating to the microcontroller part the new .hex file

NOTE: The procedure in this sub-section should be executed each time a new program is designed for the microcontroller.

1. KEIL-MDK DEBUGGER

**Objective:** Learn how to operate the Keil-MDK debugger.

**Instructions:** Read and follow the steps in the following subsections.

**Prerequisites:** Fulfilled Sections A and B.1 of this document.

C.1 – Debugger Usage

1. According to their pinout, there exist mainly two types of **ST-LINK V2** **Debugger** in the market. The **red** one in Figure C.1.1 will be named as *Type A* and the one in **black** one in Figure C.1.2 will be named as **Type B**. The main difference is on the **SWDIO, SWIM** and **GND** pin locations. Checkout which type do you have.



Figure C.1.1. ST-LINK V2 Debugger Type A



Figure C.1.2. ST-LINK V2 Debugger Type B

1. Connect the **ST-LINK V2 Debugger** to the **STM32F103** board according to the type you have and the following figures. Then connect the debugger to the computer. If everything is correct, the computer should display a message similar to the one displayed in Figure C.1.3. In the following labs, this will be the initial basic connection. **WARNING: The debugger pinout may be different to those in the figures. Still, it should contain the same pins, so modify the connection accordingly. Ask help from your instructor should you require it. Connecting the debugger to the board in the wrong way might damage both components permanently.**

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Figure C.1.5. Type A Physical connection |
|  |
| Figure C.1.6. Type B Physical connection |
| Figure C.1.3. Type A Interface schematic | Figure C.1.4. Type B Interface schematic |

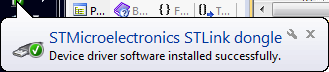


Figure C.1.7. Debugger driver successfully installed

1. Rename the **F3\_Template** project to *LAB2A*(check section B.2).Copy and paste the source code showed in Listing C.1.1 into the **main.c** file in the project then save it. The computer must have established communication with the **STM32F103** board.

Listing C.1.1. LAB #2 First Program

|  |
| --- |
| #include <stm32f10x.h>  const uint16\_t A[3][3] = { { 1,3,5 }, { 5,1,3 }, { 3,5,1 } };  const uint16\_t B[3][3] = { { 2,4,6 }, { 6,2,4 }, { 4,6,2 } };  int main( void )  {  \_\_IO uint16\_t C[3][3] = { { 0,0,0 }, { 0,0,0 }, { 0,0,0 } };  \_\_IO uint16\_t i,j,k,accum;    for( i=0; i<3; i++ )  {  for( j=0; j<3; j++ )  {  accum = 0;  for( k=0; k<3; k++ )  {  accum += A[i][k]\*B[k][j];  }  C[i][j] = accum;  }  }  for(;;);    } |

1. Go to *Project 🡪 Rebuild All Target Files* (or press the  icon in the toolbar).
2. Go to *Debug 🡪 Start Debug Session* (or press *<CTRL> + <F5>* or press the  icon in the toolbar).
3. Go to *View 🡪 Watch Windows 🡪 Watch 1*(or search for the **Watch windows** icon). Click on it.

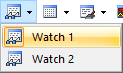


Figure C.1.8. Opening the Watch windows

1. In the cell **Enter Expression** from the **Watch 1** window, type *A*. With this, we have added the **A** matrix to the **Watch 1** window. Repeat this step for matrices B and C.
2. Right-click on each of the variables in the **Watch 1** window and uncheck the **Hexadecimal Display** option. Then expand all the variables. You should get an output similar like the one displayed in Figure C.1.9.

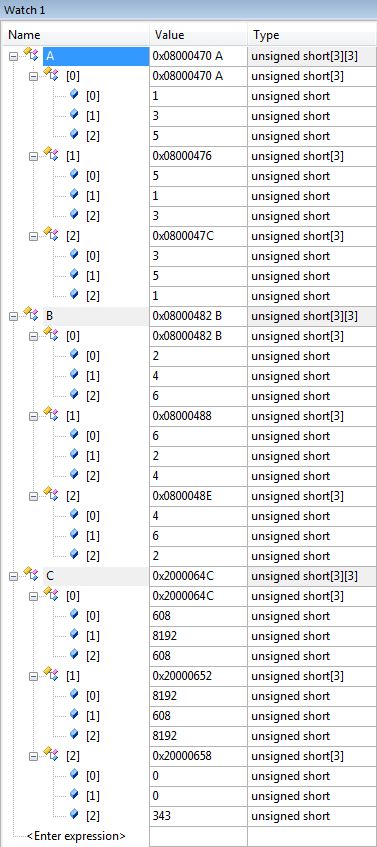


Figure C.1.9. Matrix variables in Watch 1 Window

1. Go to **line 11** in the code. Right-click on it, and select **Insert/Remove Breakpoint** (or press F9). A red dot should appear on the line number of that line, like the one in Figure C.1.11.

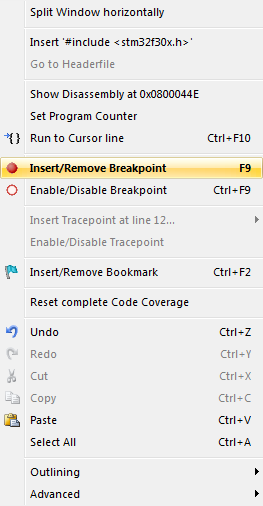


Figure C.1.10. **Insert/Remove Breakpoint** option



Figure C.1.11. Placing a breakpoint

1. Go to *Debug 🡪 Run* (or use the shortcut F5 or press the  icon in the toolbar). The program should now stop in line 11 and the values in the **Watch 1** window should be like in Figure C.1.9.
2. If the values are displayed in a matrix form, then, the outputs of **Watch 1** window are:

|  |  |  |
| --- | --- | --- |
| Matrix A | Matrix B | Matrix C |
|  |  |  |

1. Remove the breakpoint in Line 11 by repeating step 8 or by following *Debug 🡪 Kill All Breakpoints*. Place another breakpoint in Line 13. Press the *<F5>* key twice. The values of matrix C should be the following:

|  |
| --- |
| Matrix C |
|  |

1. Press the *<F5>* key two more times. The values of matrix C should be the ones depicted in Table C.1.1:

Table C.1.1. Representing matrix C after two debug steps in a table

|  |  |
| --- | --- |
| Step | Matrix C |
| 2 |  |
| 3 |  |

1. Remove the breakpoint in line 13 (right-click on the number line and select again **Insert/Remove Breakpoint**). Create a new one in line 15. Click on the *Reset* button to reset the CPU.



Figure C.1.12. Resetting the CPU

1. Press the *<F5>* key. The program should stop at line 15.
2. Add the **accum** variable into the **Watch 1** window. Disable the *Hexadecimal Display* option.
3. Press the *<F5>* key twice. In the **C** and **accum** variables from the **Watch 1** window and make it display decimal values, you should get something similar to Figure C.1.13.

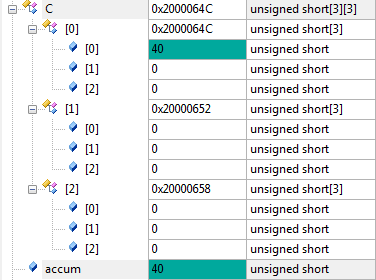


Figure C.1.13. Updating variables

1. In another representation, the last results are expressed as in Table C.1.2.

Table C.1.2. Representing variables in **Watch 1** window in a table

|  |  |  |
| --- | --- | --- |
| Step | accum | Matrix C |
| 1 | 40 |  |

1. (R) Press the <F5> key until the matrix C is completely calculated. For each step, add a row to Table C.1.2 and fill the information requested in the columns.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | Step | accum | Matrix C | | 1 | 40 |  | | 2 | 40 |  | | 3 | 28 |  | | 4 | 28 |  | | 5 | 40 |  | | 6 | 40 |  | | 7 | 40 |  | | 8 | 28 |  | | 9 | 40 |  | |

1. Remove the breakpoint in Line 15. Place a new one in Line 18. Reset the debugger. Compress the variables **A** and **B** in the **Watch 1** window (click on the – icon for each variable). Remove the variable **accum** in the **Watch 1** window (*right-click* 🡪 *Remove Watch* ***accum***). Add the variables (in this order) **i, j, k, A[i][k], B[k][j]** and **accum**. Remove on each variable the **Hexadecimal Display** option. Then press the <F5> key. The **Watch 1** window should display something like Figure C.1.14. for the involved variables:

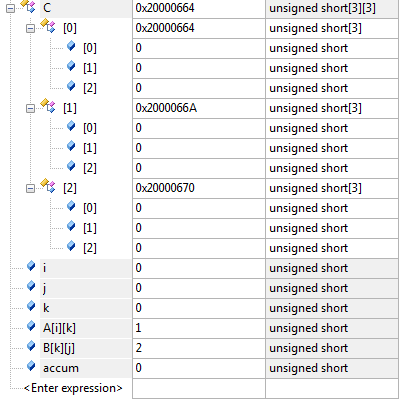


Figure C.1.14. Updating variables

1. In another representation, the last results are expressed in Table C.1.3:

Table C.1.3. Representing variables in figure C.1.10 in a table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Step | Matrix C | i | j | k | A[i][k] | B[k][j] | accum |
| 0 |  | 0 | 0 | 0 | 1 | 2 | 0 |

1. (R) Press the *<F5>* key until the matrix C is completely calculated. For each step, add a row to Table C.1.3 and fill the information requested in the columns. Copy and paste the full table in the red box.

0

0

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | 0Step | Matrix C | i | j | k | A[i][k] | B[k][j] | Accum | | 0 |  | 0 | 0 | 0 | 1 | 2 | 0 | | 1 |  | 0 | 0 | 1 | 3 | 6 | 2 | | 2 |  | 0 | 0 | 2 | 5 | 4 | 20 | | 3 |  | 0 | 1 | 0 | 1 | 4 | 0 | | 4 |  | 0 | 1 | 1 | 3 | 2 | 4 | | 5 |  | 0 | 1 | 2 | 5 | 6 | 10 | | 6 |  | 0 | 2 | 0 | 1 | 6 | 0 | | 7 |  | 0 | 2 | 1 | 3 | 4 | 6 | | 8 |  | 0 | 2 | 2 | 5 | 2 | 18 | | 9 |  | 1 | 0 | 0 | 5 | 2 | 0 | | 10 |  | 1 | 0 | 1 | 1 | 6 | 10 | | 11 |  | 1 | 0 | 2 | 3 | 4 | 16 | | 12 |  | 1 | 1 | 0 | 5 | 4 | 0 | | 13 |  | 1 | 1 | 1 | 1 | 2 | 20 | | 14 |  | 1 | 1 | 2 | 3 | 6 | 22 | | 15 |  | 1 | 2 | 0 | 5 | 6 | 0 | | 16 |  | 1 | 2 | 1 | 1 | 4 | 30 | | 17 |  | 1 | 2 | 2 | 3 | 2 | 34 | | 18 |  | 2 | 0 | 0 | 3 | 2 | 0 | | 19 |  | 2 | 0 | 1 | 5 | 6 | 6 | | 20 |  | 2 | 0 | 2 | 1 | 4 | 36 | | 21 |  | 2 | 1 | 0 | 3 | 4 | 0 | | 22 |  | 2 | 1 | 1 | 5 | 2 | 12 | | 23 |  | 2 | 1 | 2 | 1 | 6 | 22 | | 24 |  | 2 | 2 | 0 | 3 | 6 | 0 | | 25 |  | 2 | 2 | 1 | 5 | 4 | 18 | | 26 |  | 2 | 2 | 2 | 1 | 2 | 38 | | 27 |  | 3 | 3 | 3 | 6 | 8192 | 40 | |

1. (R) In the following red box, place the commented **main.c** program used in Listing C.1.1.

|  |
| --- |
| #include <stm32f10x.h>  //Se trata de un código de multiplicación de matrices que utiliza el método  //convencional para lograr esta tarea.  //para poder comprenderlo mejor, es necesario comenzar desde la linea con: "for( k=0; k<3; k++ )"  const uint16\_t A[3][3] = { { 1,3,5 }, { 5,1,3 }, { 3,5,1 } };  const uint16\_t B[3][3] = { { 2,4,6 }, { 6,2,4 }, { 4,6,2 } };  int main( void )  {  \_\_IO uint16\_t C[3][3] = { { 0,0,0 }, { 0,0,0 }, { 0,0,0 } };  \_\_IO uint16\_t i,j,k,accum;    //Se hace un ciclo que se repetirá 3 veces acumulando las columnas de C  for( i=0; i<3; i++ )  {  for( j=0; j<3; j++ ){  accum = 0;  //Se hace un ciclo que se repetirá 3 veces realizando la multiplicación que habrá en cada espacio de C  for( k=0; k<3; k++ ){  //Se realiza la multiplicación entre cada elemento de la matríz.  accum += A[i][k]\*B[k][j];  }  //Se hace multiplicando las filas de A por las columnas de B. Se van sumando los resultados  //En esta línea se cóloca en la celda respectiva el resultado de la fila A con la columna B (ya sumados)  C[i][j] = accum;  }  }  //El ciclo principal permite pasar a los resultados de las filas que deberán colocarse en C    for(;;);    } |

C.2 – Debugging Configuration registers

1. Connect the **ST-LINK Debugger** to the **STM32F103** board according to section C.1.
2. Rename the **F3\_Template** project to *LAB2B*(check section B.2).Copy and paste the source code showed in Listing C.2.1 into the **main.c** file in the project then save it. The computer must have established communication with the **STM32F103** board.

Listing C.2.1. LAB #2 Second Program

|  |
| --- |
| #include <stm32f10x.h>  #include "delay.h"  int main( )  {  RCC->APB2ENR |= RCC\_APB2ENR\_IOPCEN;  GPIOC->CRH = ( GPIOC->CRH & ~GPIO\_CRH\_CNF13 & ~GPIO\_CRH\_MODE13 ) | GPIO\_CRH\_MODE13;    while(1)  {    GPIOC->ODR ^= 1<<13;  \_delay\_ms(500);    }  } |

1. Rebuild all the target files of the project and open the debugger. Follow name the **F3\_Template** project to *LAB2B*(check section B.2).Copy and paste the source code showed in Listing C.2.1 into the **main.c** file in the project. The computer must have established communication with the **STM32F103** board.
2. Place a breakpoint in Line 8. If you follow *Peripherals 🡪 System Viewer 🡪 GPIO 🡪 GPIOC,* a window like in Figure C.2.1 appears at the right side of the application. This window shows the data in the microcontroller registers. Each time the debugger is stopped, the window contents are refreshed.

|  |  |
| --- | --- |
|  |  |
| Figure C.2.1. Accessing peripheral registers | Figure C.2.2. Peripheral registers contents |

1. (R) Debug the program. In the following red box, report the final content of the three peripheral registers involved in it.

|  |
| --- |
|  |

1. (R) In the following red box, place a screenshot of a Proteus simulation running the program in Listing C.2.1.

|  |
| --- |
|  |

1. (R) In the following red box, place the commented **main.c** program used in Listing C.2.1.

|  |
| --- |
| #include <stm32f10x.h>  #include "delay.h"  int main( )  {  // Declaración de activaciónde Clock en pines de entrada  RCC->APB2ENR |= RCC\_APB2ENR\_IOPCEN;  //Declaración de PinC13 como entrada Pull up  GPIOC->CRH = ( GPIOC->CRH & ~GPIO\_CRH\_CNF13 & ~GPIO\_CRH\_MODE13 ) | GPIO\_CRH\_MODE13;    while(1)  {  //Hacer lectura del estado del pin cada 500 milisegundos  GPIOC->ODR ^= 1<<13;  //Espera de 500 ms  \_delay\_ms(500);    }  }  } |