

MPLAB First Steps

Installation

Install MPLAB X IDE

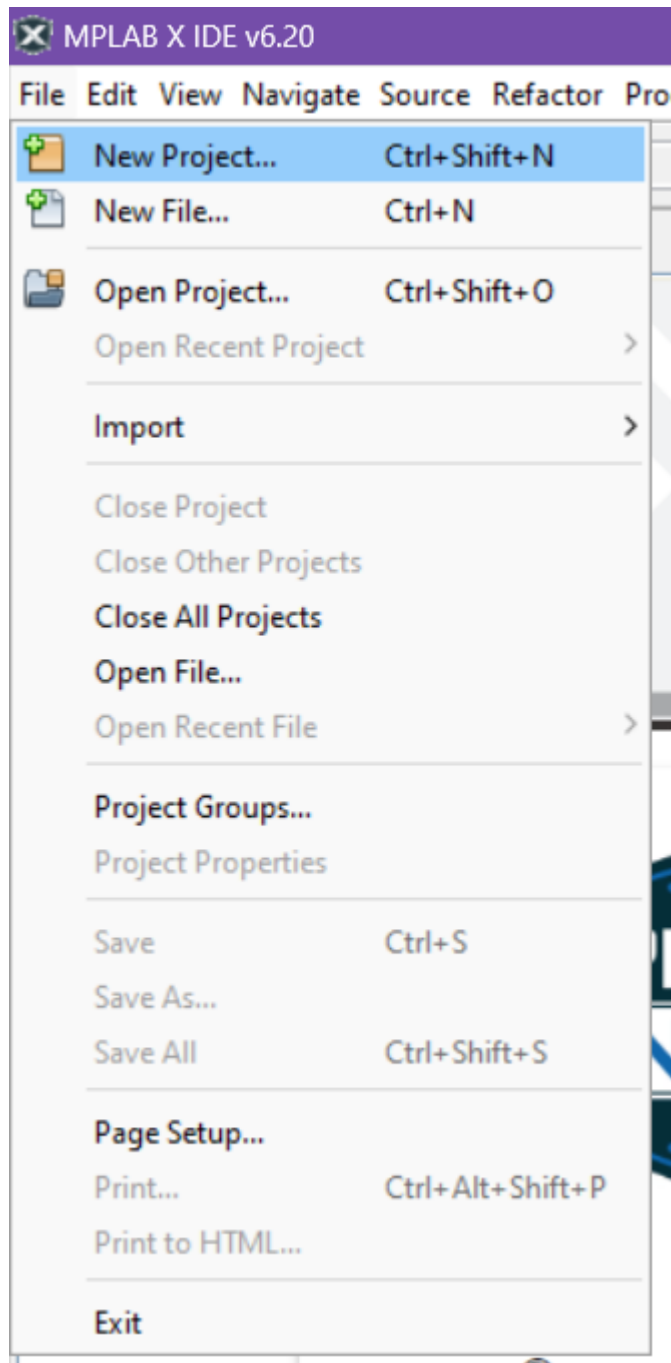
Install MPLAB X IDE through the official Microchip site: [MPLAB X IDE](#)

Install XC32 Compiler

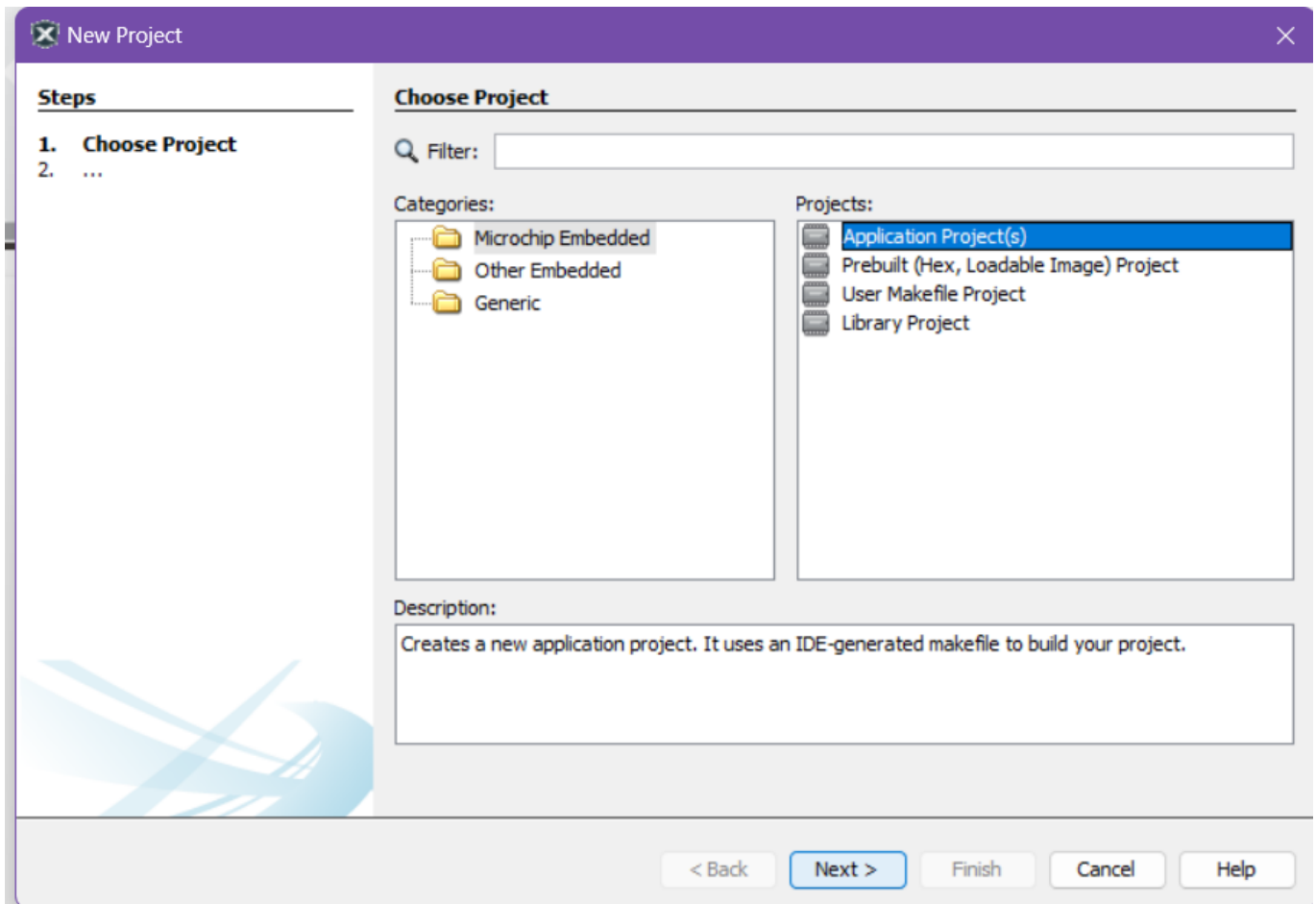
Install the XC32 COmpilers from Microchip: [MPLAB xC32 Compilers](#)

Your First Bare-Metal Project

- Create a New Project

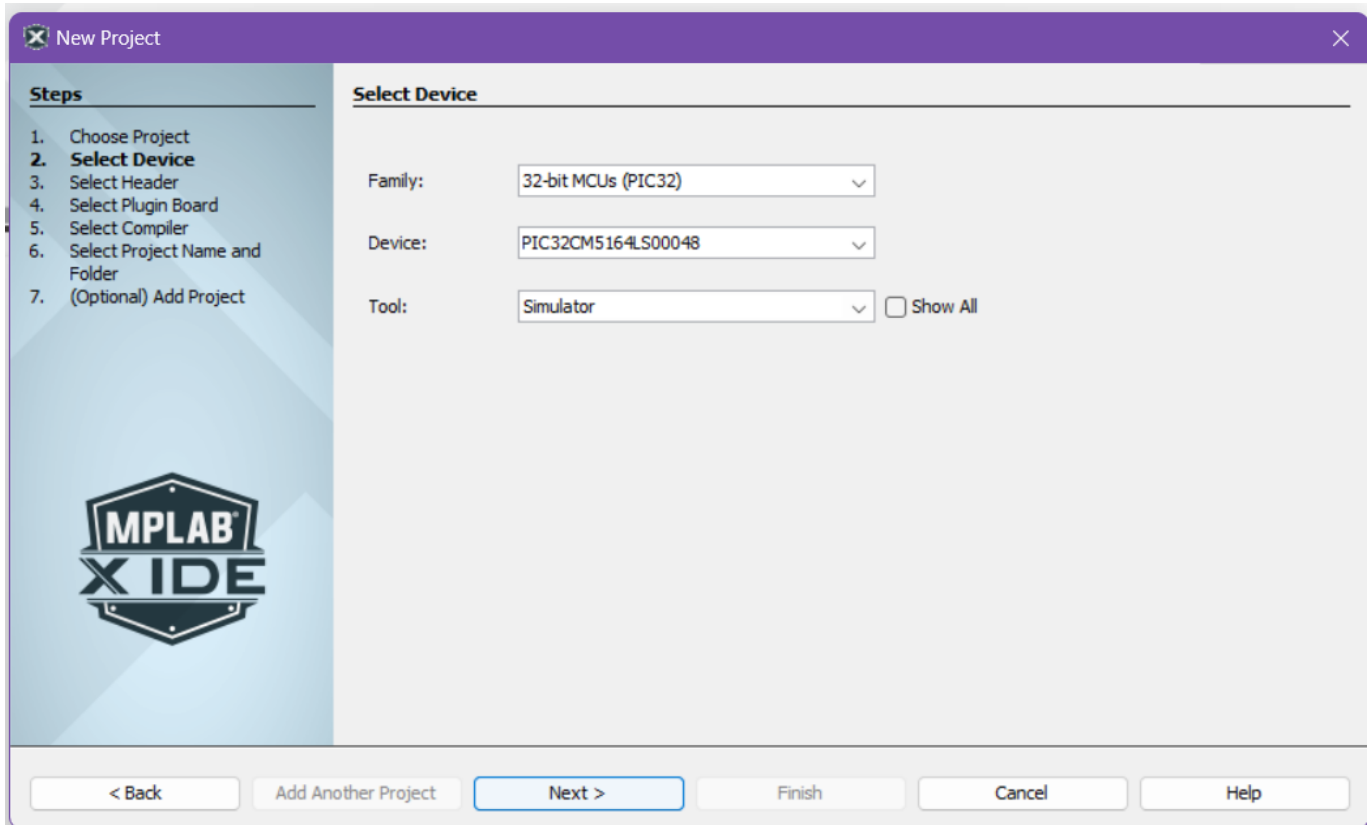


- Choose **Application Project(s)** then click **Next >**

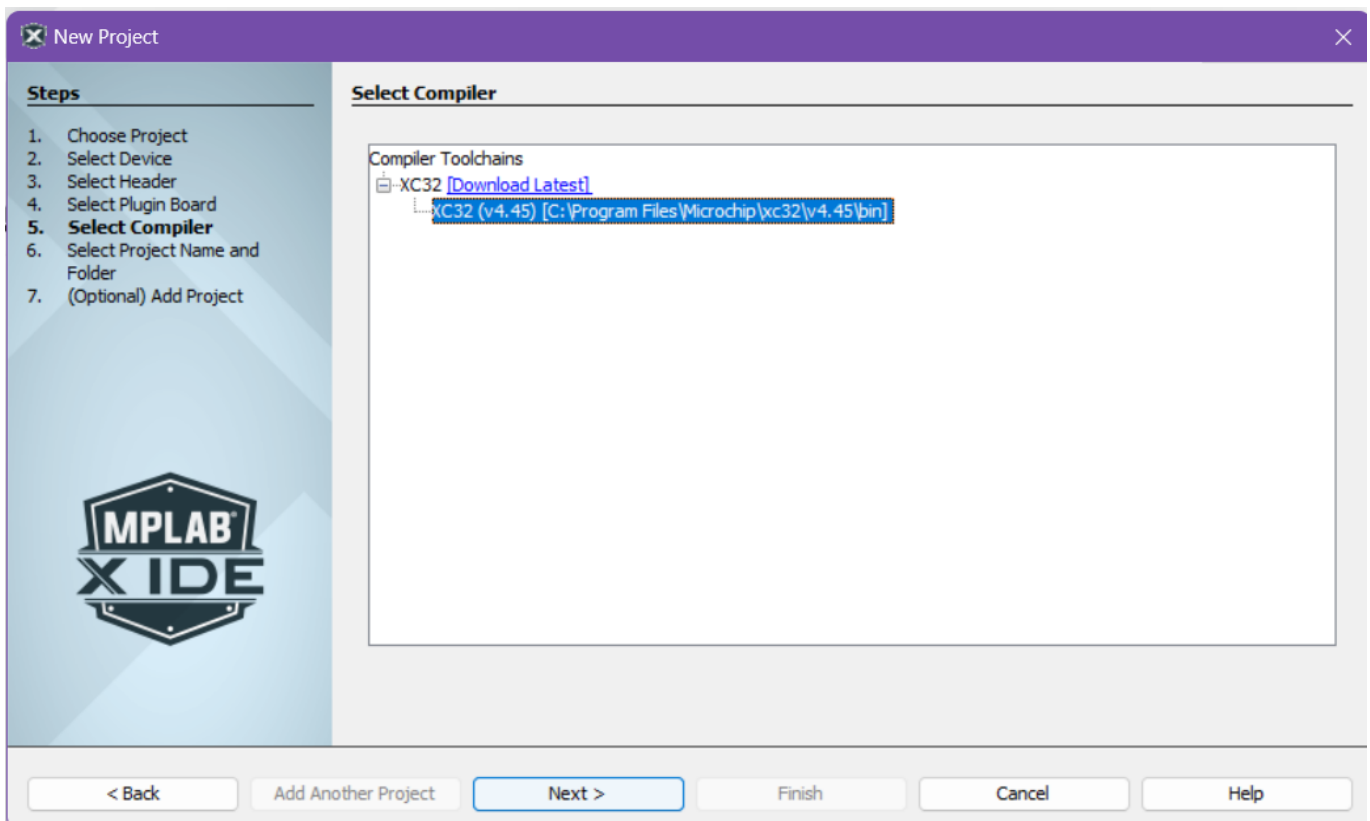


- Choose the following for 2. **Select Device**
 - **Device:** PIC32CM5164LS00048
 - **Tool:** Simulator

note: you can always change the tool later, will be useful once we move from using the simulator to actual hardware



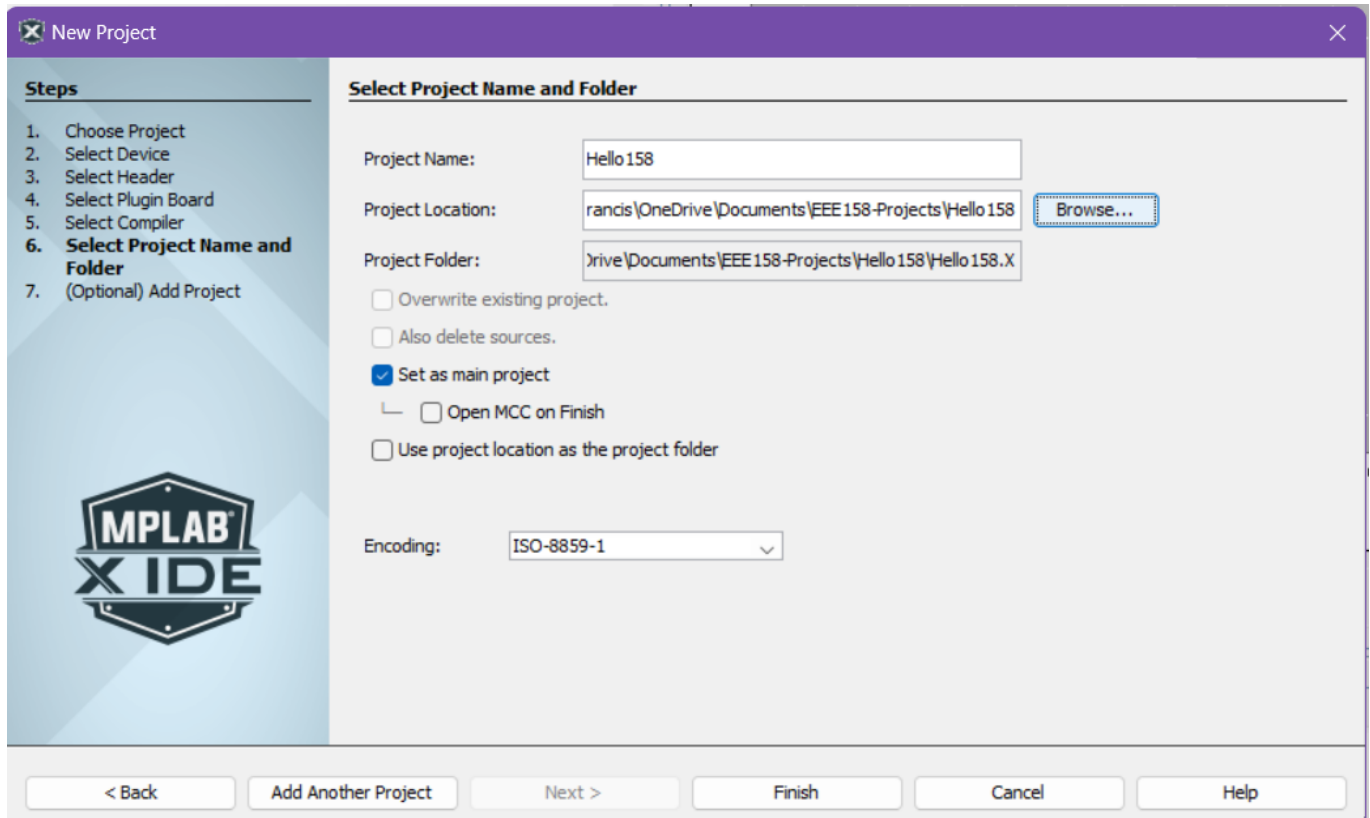
- Choose a installed XC32 Compiler
 - *note: if you do not see options for compilers, make you have properly installed the XC32 Compilers from the [Install XC32 Compilers Step](#)*



- Select where to save your Project on your local machine then click **Finish**

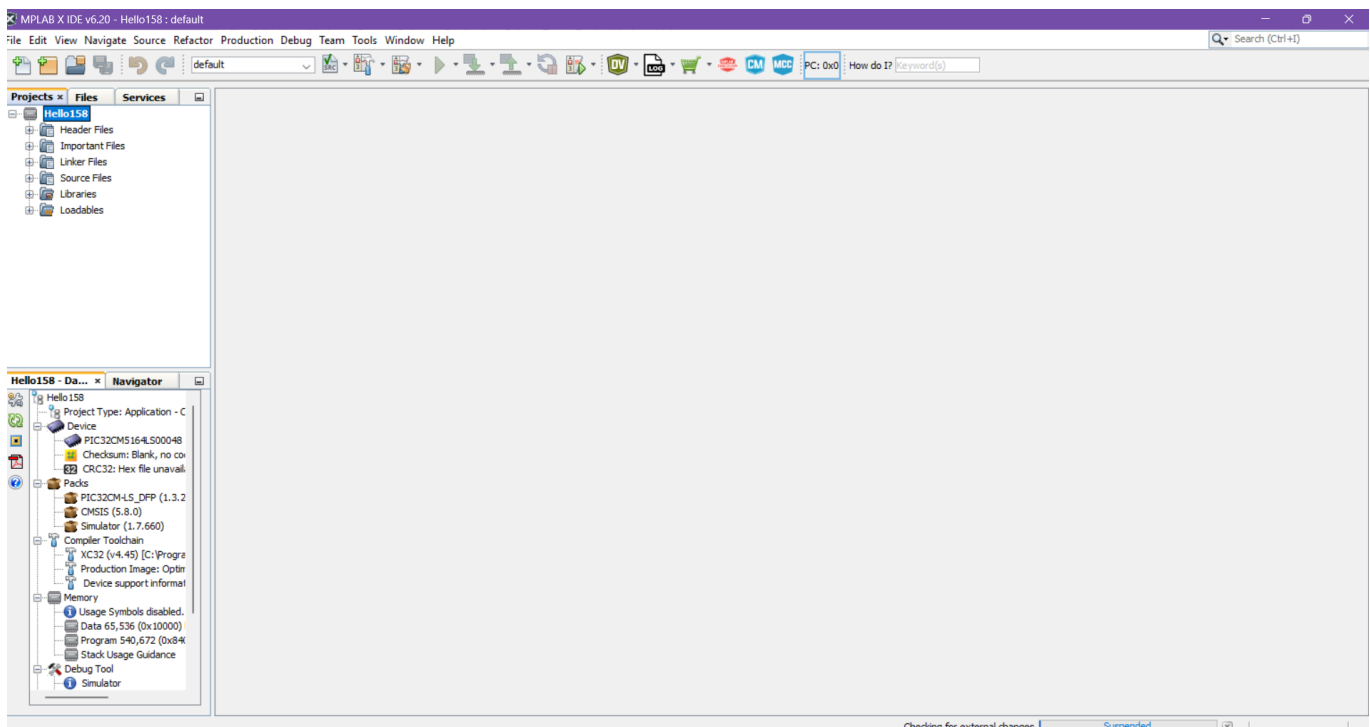
note: We suggest creating a new folder within another folder when you create a project as another folder will be generated in the same directory as the path you choose here (i.e. if you plan on storing all your Projects in a

folder called *MyProjects*, create a folder *MyProjects/MyFirstProject* and choose *MyFirstProject* as the place to save)



note: Unlick the `Open MCC on Finish` option as we won't be using that for now

- Your First Project should now initialize

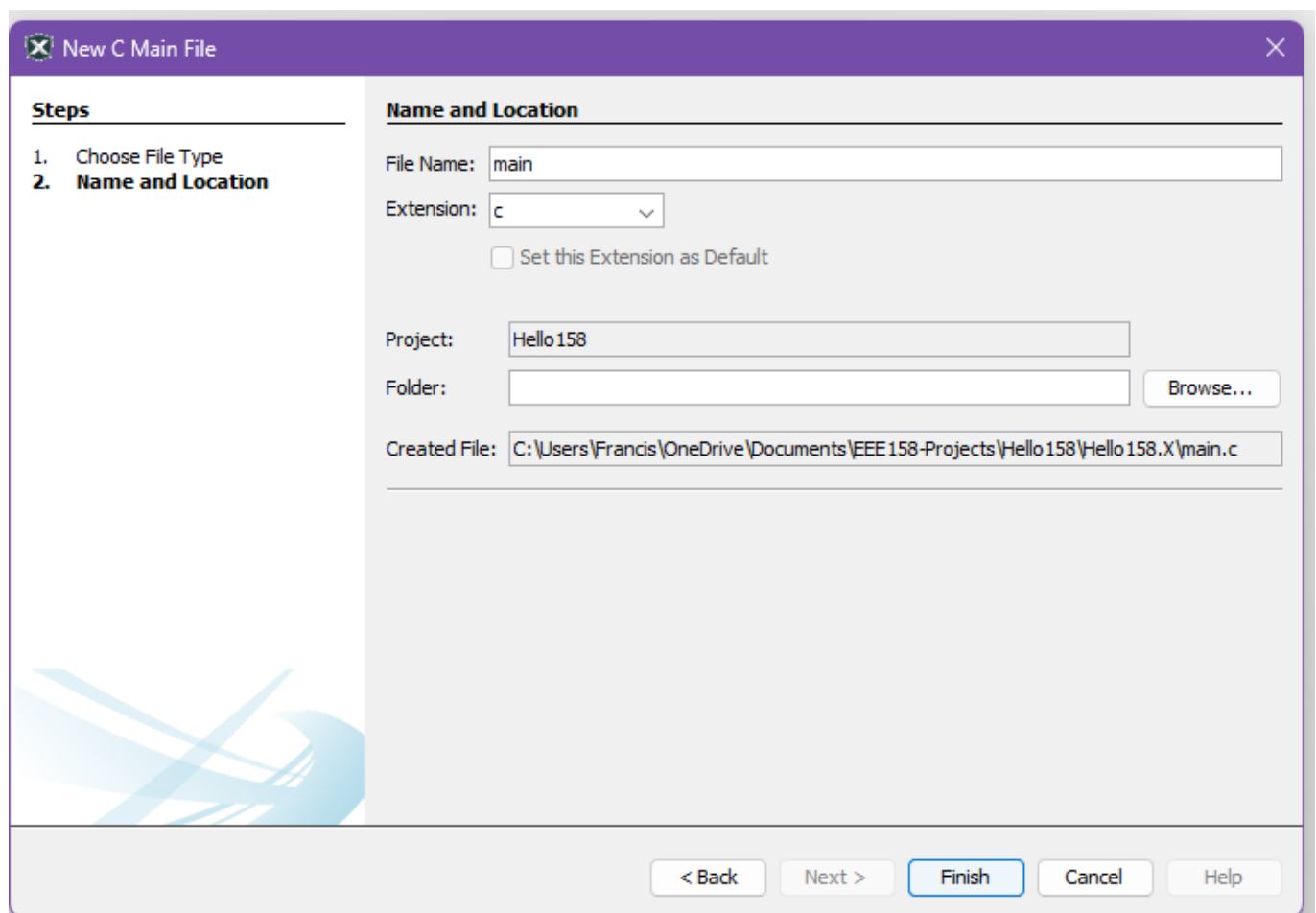
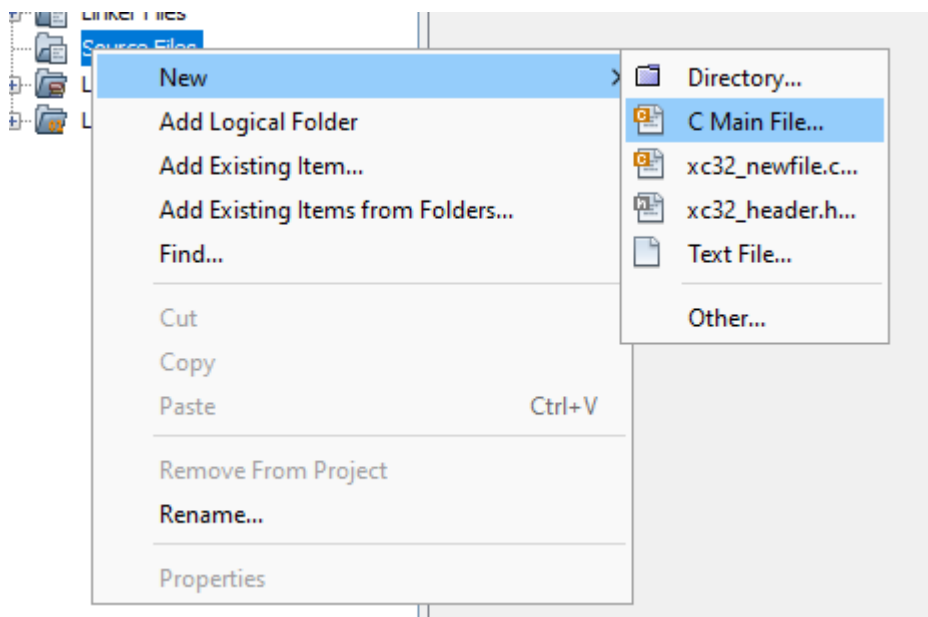


Let's now proceed to creating a `main.c` file so we can start coding our application.

Guided Exercise: "Blinky!"

Blinky!

Create a `main.c` file. Right click on the **Source Files** of the Project Tab then click **New** and **C Main File**



Proceed to input the following code into `main.c`

```
#include <stdio.h>
#include <stdlib.h>
#include <xc.h>
```

```

/*
 *
 */

int crude_ms_delay(int ms){
    int count = 0;
    unsigned int delay_count = ms * 240;

    while(count < delay_count){
        asm("nop");
        count = count + 1;
    }
    return 0;
}

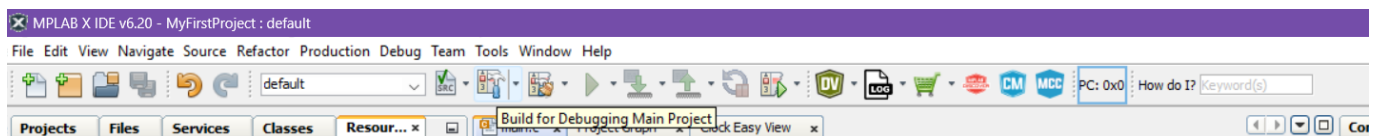
int main() {
    //Set the Data Direction for PA15 as Output
    PORT_SEC_REGS->GROUP[0].PORT_DIRSET = (1 << 15);

    //Set the Initial Output Value for PA15 as HIGH
    PORT_SEC_REGS->GROUP[0].PORT_OUTSET = (1 << 15);

    while(1){
        //Toggle the Output of PA15
        PORT_SEC_REGS->GROUP[0].PORT_OUTTGL = (1 << 15);
        crude_ms_delay(1000);
    }
    return (EXIT_SUCCESS);
}

```

Let us first try Building our Application. Press the **Build** Icon to compile your project. It's the one that looks like a hammer



If it builds, we are now ready to try and debug your application.

Simulator Debugging

note: the simulator has limited functionality in simulating the PORT peripheral

```

-----
W0106-SIM: This device only has partial support for PORT_GROUP0 peripheral. ** beta mode **
W0106-SIM: This device only has partial support for PORT_GROUP1 peripheral. ** beta mode **
W0106-SIM: This device only has partial support for EIC peripheral. ** beta mode **
W0106-SIM: This device only has partial support for TC0 peripheral. ** beta mode **
W0106-SIM: This device only has partial support for TC1 peripheral. ** beta mode **
W0106-SIM: This device only has partial support for TC2 peripheral. ** beta mode **
W0106-SIM: This device only has partial support for EVSYS peripheral. ** beta mode **
W0111-SIM: The selection is not supported: We don't simulate the clock selection
-----

```

Include a breakpoint in your program by clicking on the line number on the left side of the code editor

```

34 |         while(1){
35 |             //Toggle the Output of PA15
36 |             PORT_SEC_REGS->GROUP[0].PORT_OUTTGL = (1 << 15);
37 |             crude_ms_delay(1000);
38 |         }
39 |         return (EXIT_SUCCESS);
40 |     }

```

Click on the **Debug Main Project** button on the ribbon to begin Simulator Debugging.

Application Execution Control

These buttons on the ribbon will allow you to control the debugging of your application.



- **Finish Debugger Session** - End Debugging
- **Pause** - Pause Application Execution
- **Reset** - Reset Application Execution
- **Continue** - Resume Application Execution
- **Step Over** - Step over a line of your Application
- **Step Into** - Step into a a method in your Application
- **Step Out** - Step out of a method in your application
- **Run to Cursor**
- **Set PC at Cursor**
- **Focus Cursor at PC**

Your debugger should stop at the break point we set earlier

```

33 |
34 |         while(1){
35 |             //Toggle the Output of PA15
36 |             PORT_SEC_REGS->GROUP[0].PORT_OUTTGL = (1 << 15);
37 |             crude_ms_delay(1000);
38 |         }
39 |         return (EXIT_SUCCESS);
40 |     }
41 |

```

Press the **Step Over** Button and you should go into the next line of the Application code


```

9      while(count < delay_count){
10          asm("nop");
11          count = count + 1;
12      }
13      return 0;
14  }
15
16  int main() {
17      //Set the Data Direction for PA15 as Output
18      PORT_SEC_REGS->GROUP[0].PORT_DIRSET = (1 << 15);
19
20      //Set the Initial Output Value for PA15 as HIGH
21      PORT_SEC_REGS->GROUP[0].PORT_OUTSET = (1 << 15);
22
23      while(1){
24          //Toggle the Output of PA15
25          PORT_SEC_REGS->GROUP[0].PORT_OUTTGL = (1 << 15);
26          crude_ms_delay(1000);
27      }
28      return (EXIT_SUCCESS);
29  }

```

Now press the **Step Into** Button and you should go into the `crude_ms_delay` method

```

14
15  int crude_ms_delay(int ms){
16      int count = 0;
17      unsigned int delay_count = ms * 240;
18
19      while(count < delay_count){
20          asm("nop");
21          count = count + 1;
22      }
23      return 0;
24  }
25
26

```

Press the **Continue** button and after a while, you should hit the breakpoint again

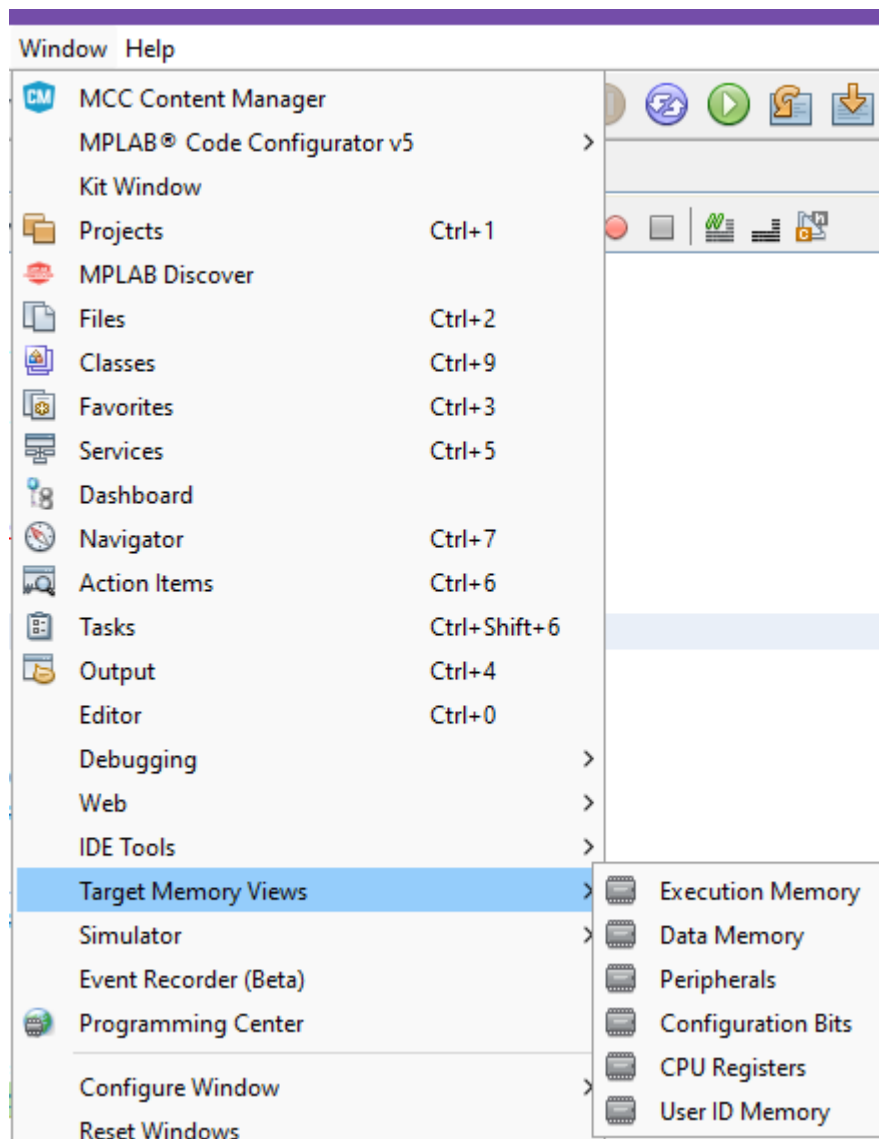
```

33
34      while(1){
35          //Toggle the Output of PA15
36          PORT_SEC_REGS->GROUP[0].PORT_OUTTGL = (1 << 15);
37          crude_ms_delay(1000);
38      }
39      return (EXIT_SUCCESS);
40  }
41

```

Target Memory Views

Target memory views allows us to look at the contents of the memory of our device under test (DUT). You can open this by navigating in the ribbon. **Window -> Target Memory Views**



Let's try and open up the **Peripherals** Memory View.

Peripherals

Address	Name	Hex	Decimal	Binary	Char
4000_0200	WRCTRL	0x00020021	131105	00000000 00000010 00000000 00100001	'...!'
4000_0204	EVCTRL	0x00	0	00000000	'.'
4000_0208	INTENCLR	0x00	0	00000000	'.'
4000_0209	INTENSET	0x00	0	00000000	'.'
4000_0210	INTFLAGAHB	0x00000000	0	00000000 00000000 00000000 00000000	'....'
4000_0214	INTFLAGA	0x00000000	0	00000000 00000000 00000000 00000000	'....'
4000_0218	INTFLAGB	0x00000000	0	00000000 00000000 00000000 00000000	'....'
4000_021C	INTFLAGC	0x00000000	0	00000000 00000000 00000000 00000000	'....'
4000_0234	STATUSA	0x0000C000	49152	00000000 00000000 11000000 00000000	'..Ä.'
4000_0238	STATUSB	0x00000002	2	00000000 00000000 00000000 00000010	'....'
4000_023C	STATUSC	0x00000000	0	00000000 00000000 00000000 00000000	'....'
4000_0254	NONSECA	0x00000000	0	00000000 00000000 00000000 00000000	'....'
4000_0258	NONSECB	0x00000002	2	00000000 00000000 00000000 00000010	'....'
4000_025C	NONSECC	0x00000000	0	00000000 00000000 00000000 00000000	'....'
4000_0274	SECLOCKA	0x0000FFFF	65535	00000000 00000000 11111111 11111111	'..ÿÿ'
4000_0278	SECLOCKB	0x0000003F	63	00000000 00000000 00000000 00111111	'...?'
4000_027C	SECLOCKC	0x003FFFFFFF	4194303	00000000 00111111 11111111 11111111	'..?ÿÿ'
4000_0401	SLEEPCFG	0x02	2	00000010	'.'
4000_0402	PLCFG	0x02	2	00000010	'.'
4000_0403	PWCFG	0x00	0	00000000	'.'
4000_0404	INTENCLR	0x00	0	00000000	'.'
4000_0405	INTENSET	0x00	0	00000000	'.'
4000_0406	INTFLAG	0x01	1	00000001	'.'
4000_0408	STDBYCFG	0x0000	0	00000000 00000000	'..'
4000_0800	CTRLA	0x00	0	00000000	'.'
4000_0801	INTENCLR	0x00	0	00000000	'.'
4000_0802	INTENSET	0x00	0	00000000	'.'
4000_0803	INTFLAG	0x01	1	00000001	'.'
4000_0804	CPUDIV	0x01	1	00000001	'.'
4000_0810	AHBMASK	0x000003FFF	16383	00000000 00000000 00111111 11111111	'..?ÿ'
4000_0814	APBAMASK	0x000007FFF	32767	00000000 00000000 01111111 11111111	'..ÿÿ'
4000_0818	APBBMASK	0x000000037	55	00000000 00000000 00000000 00110111	'...7'
4000_081C	APBCMASK	0x001FFFFFFF	2097151	00000000 00011111 11111111 11111111	'..ÿÿ'

Memory

Peripherals

Format

Individual

Filter

Address

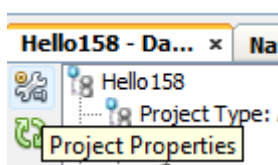
This allows us to directly check the contents of the registers and is useful for debugging.

Hardware Debugging

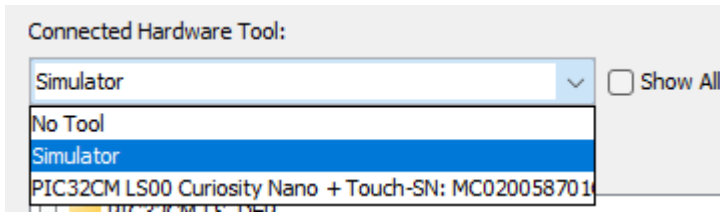
Hardware debugging is similar to Simulator debugging and you can perform any of the Application Execution Controls and Memory View functionality.

To change the **Tool** from Simulator to you on-board debugger

- Plug in your Curiosity Nano board to your machine
- Click on **Project Properties**



- Change the **Connected Hardware Tool** to you on-board debugger

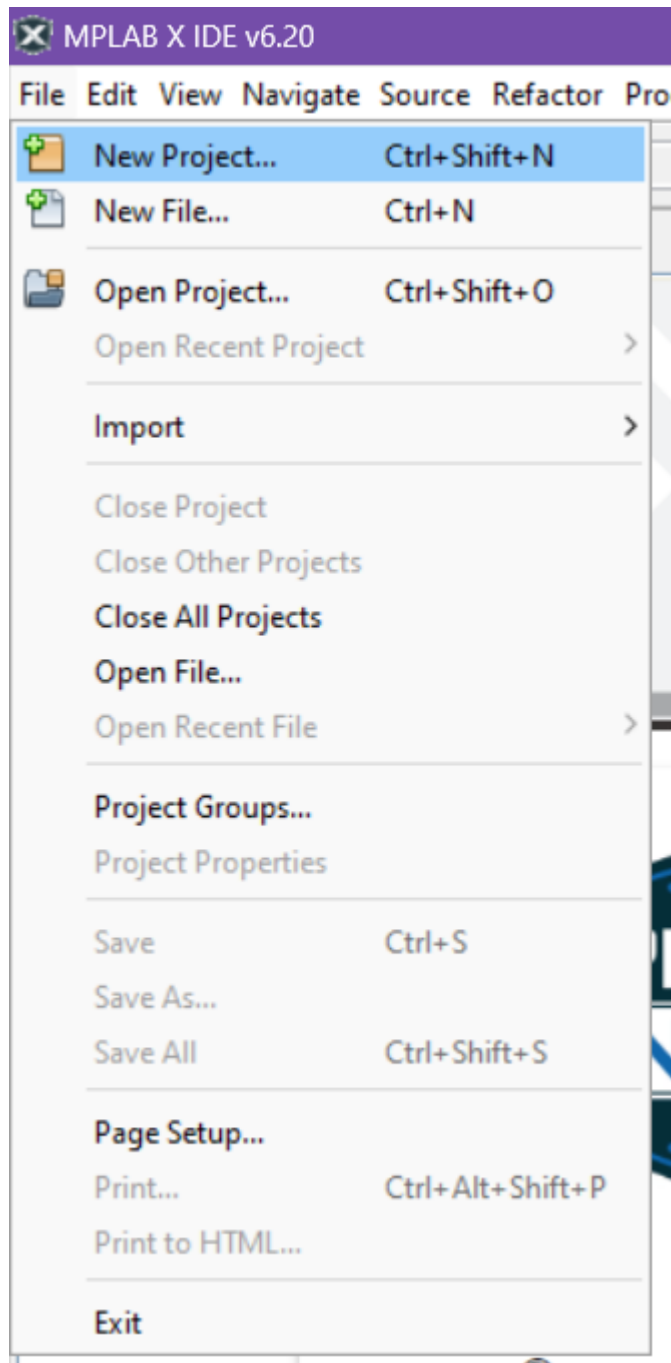


- When you perform debugging now, the program will first upload to your microcontroller and you can proceed with debugging.

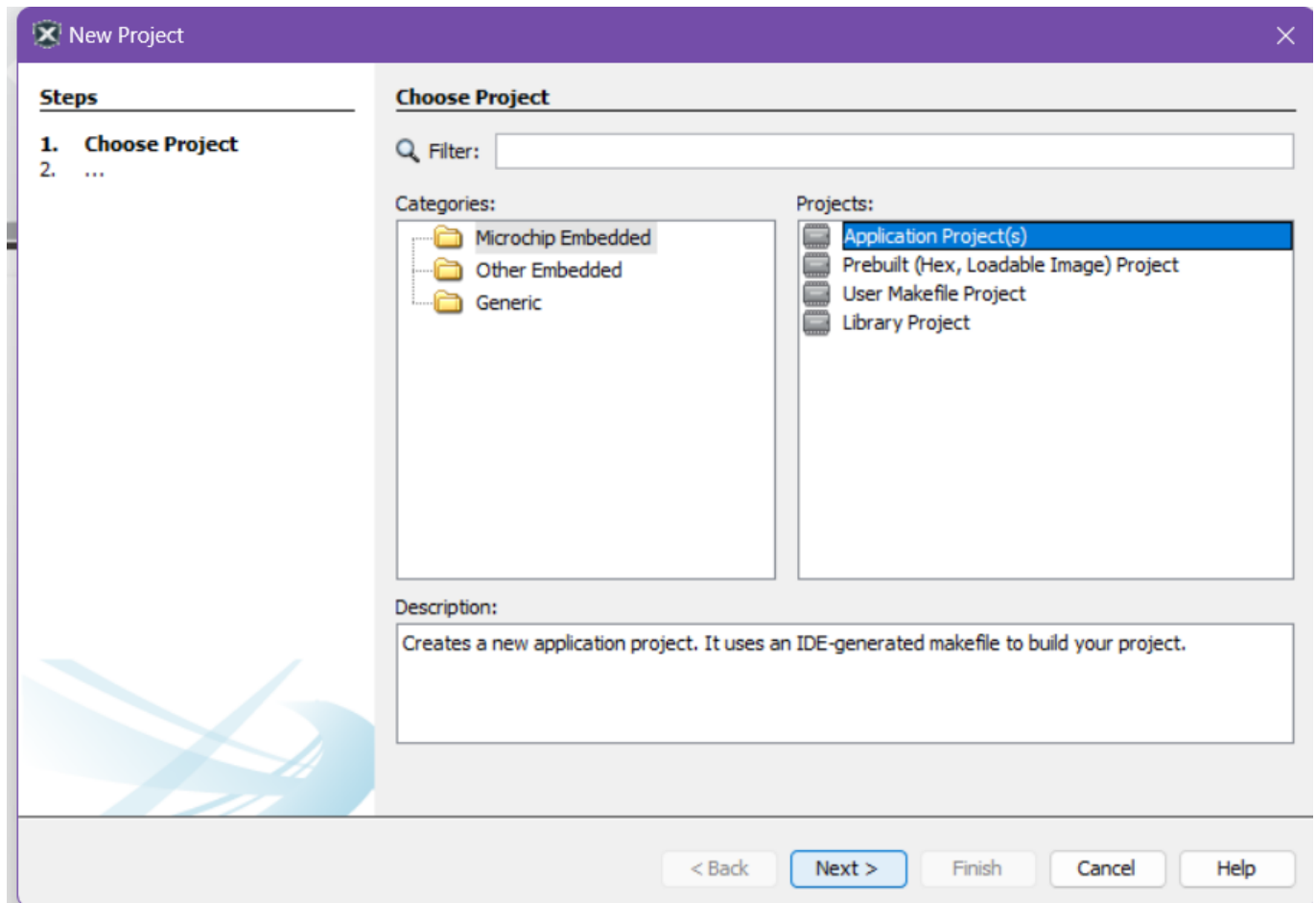
Your First MCC Project

MPLAB Code Configurator (MCC) is a tool from Microchip which allows us to easily configure our Microcontroller by automatically creating macros we can use for our code. Let's try and use it.

- Create a New Project

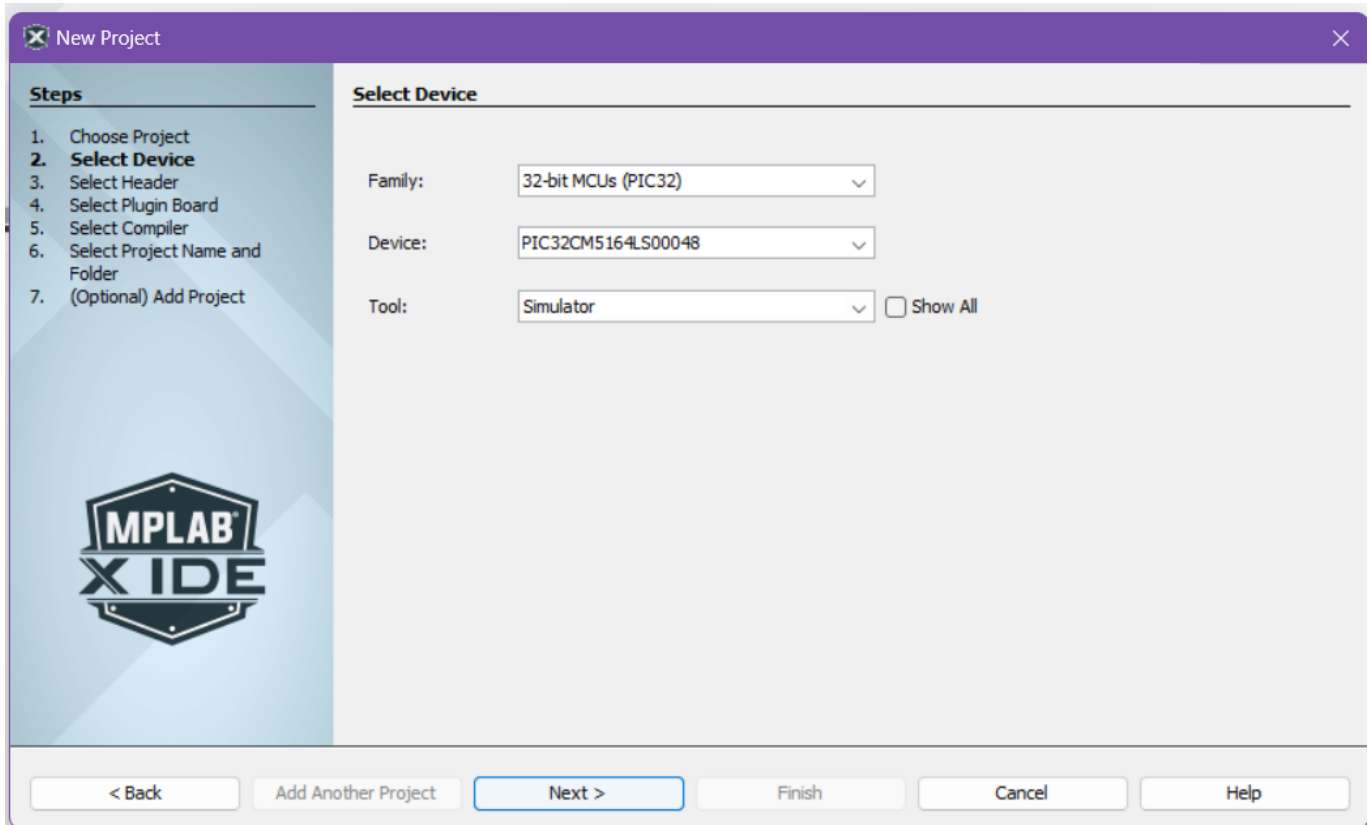


- Choose **Application Project(s)** then click **Next >**

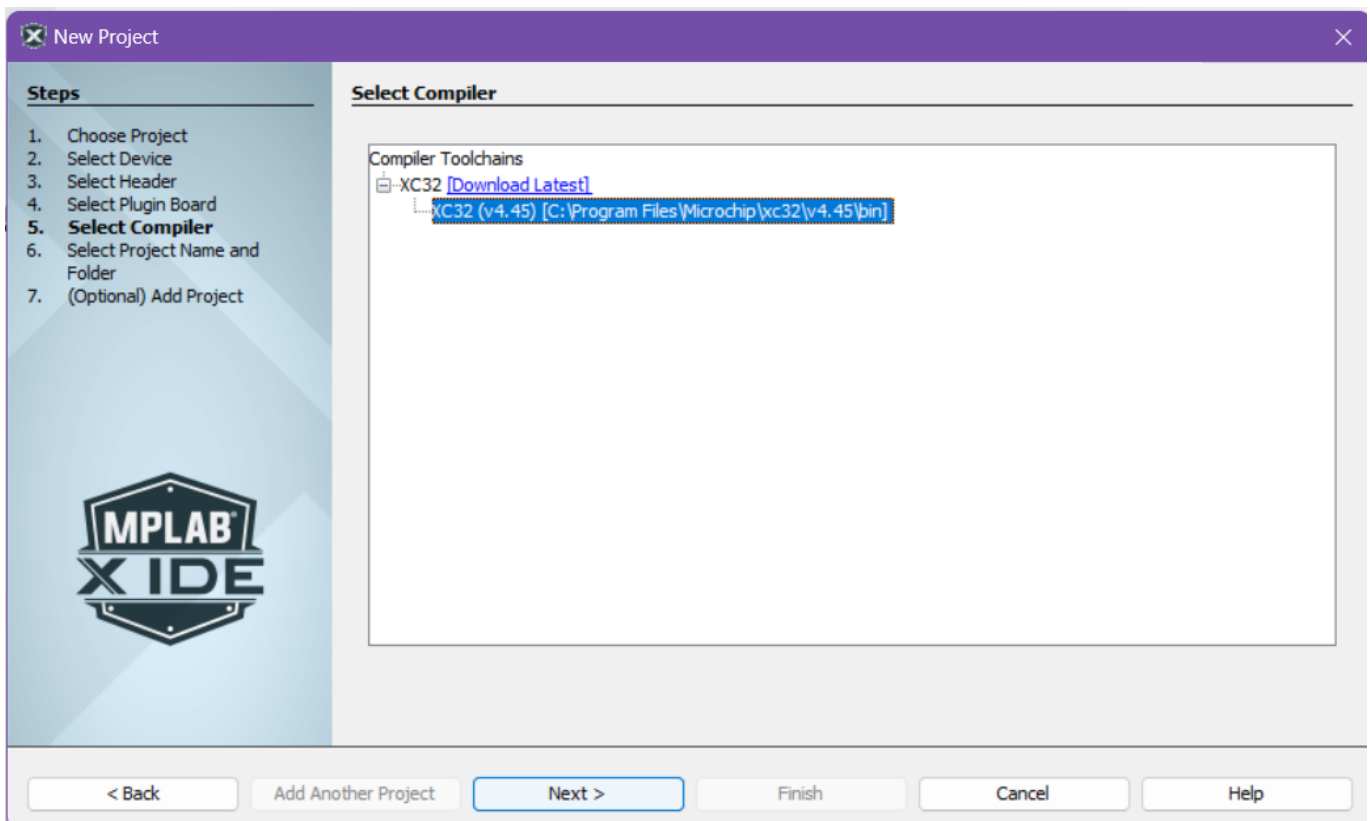


- Choose the following for 2. **Select Device**
 - **Device:** PIC32CM5164LS00048
 - **Tool:** Simulator

note: you can always change the tool later, will be useful once we move from using the simulator to actual hardware



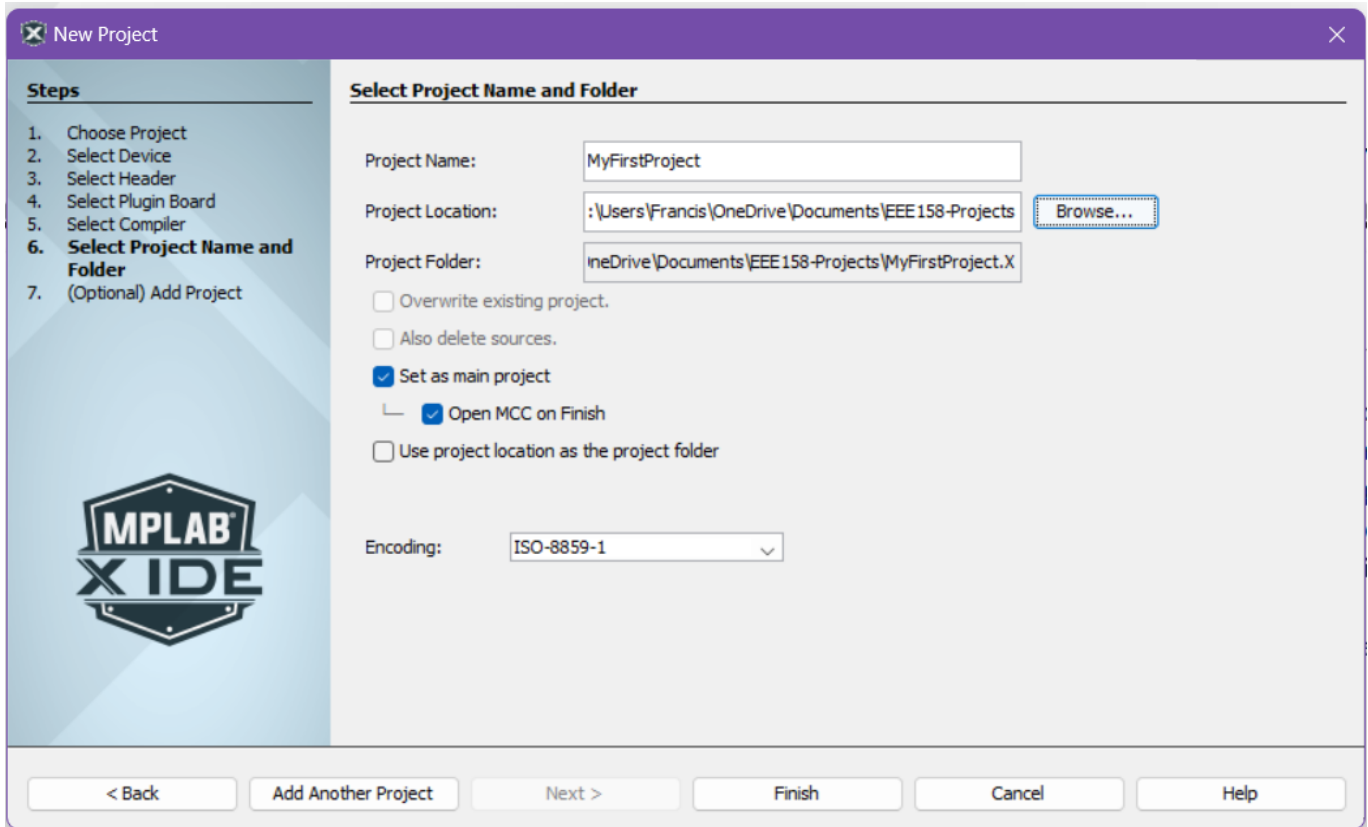
- Choose a installed XC32 Compiler
 - *note: if you do not see options for compilers, make you have properly installed the XC32 Compilers from the [Install XC32 Compilers Step](#)*



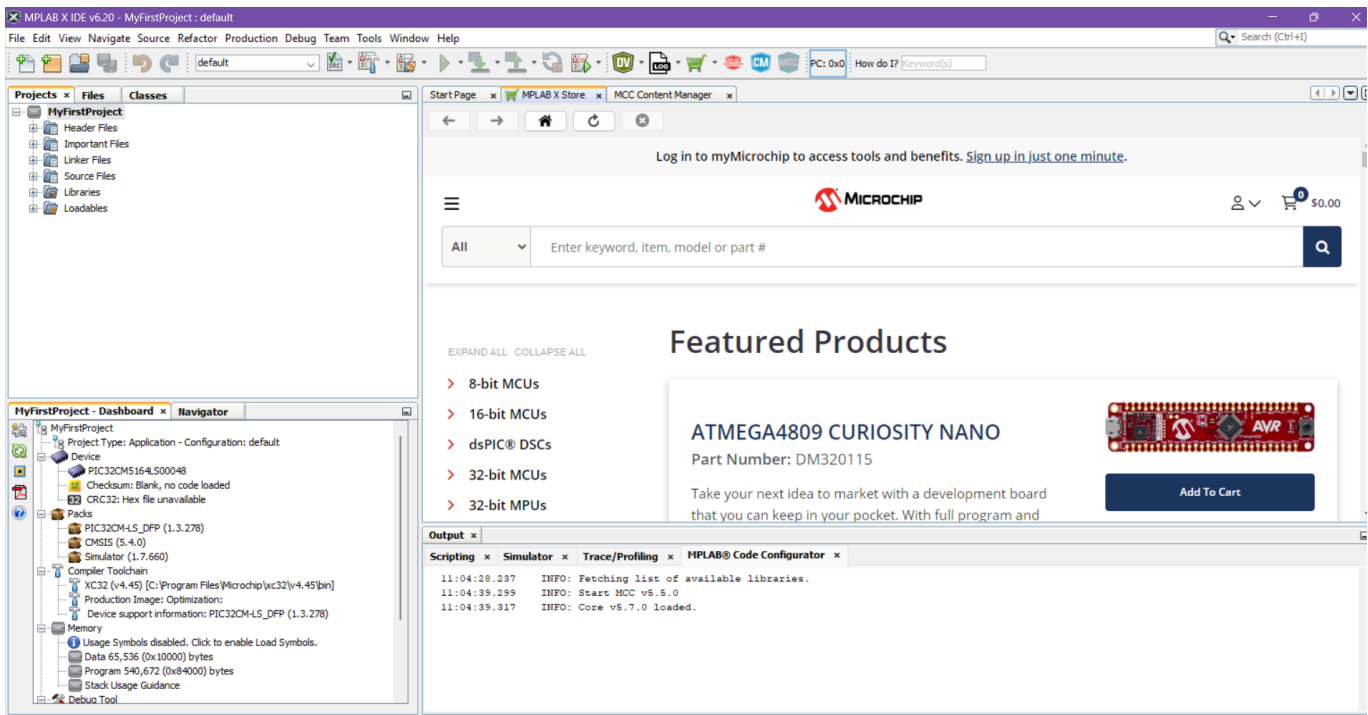
- Select where to save your Project on your local machine then click **Finish**

note: We suggest creating a new folder within another folder when you create a project as another folder will be generated in the same directory as the path you choose here (i.e. if you plan on storing all your Projects in a

folder called *MyProjects*, create a folder *MyProjects/MyFirstProject* and choose *MyFirstProject* as the place to save)



- Your First Project should now initialize

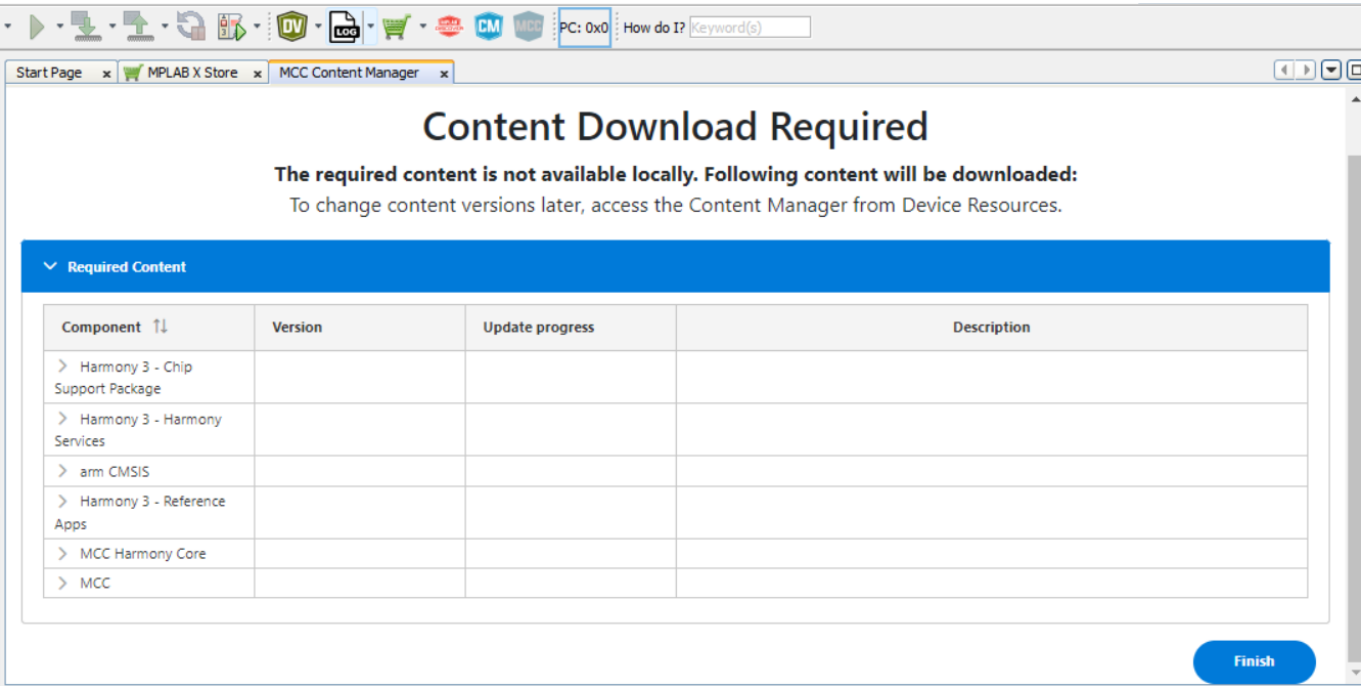


- We now proceed to setting up the MPLAB Code Configurator, It will either open up automatically or you can open it by clicking on the **MCC** icon on the top ribbon



MPLAB Code Configurator

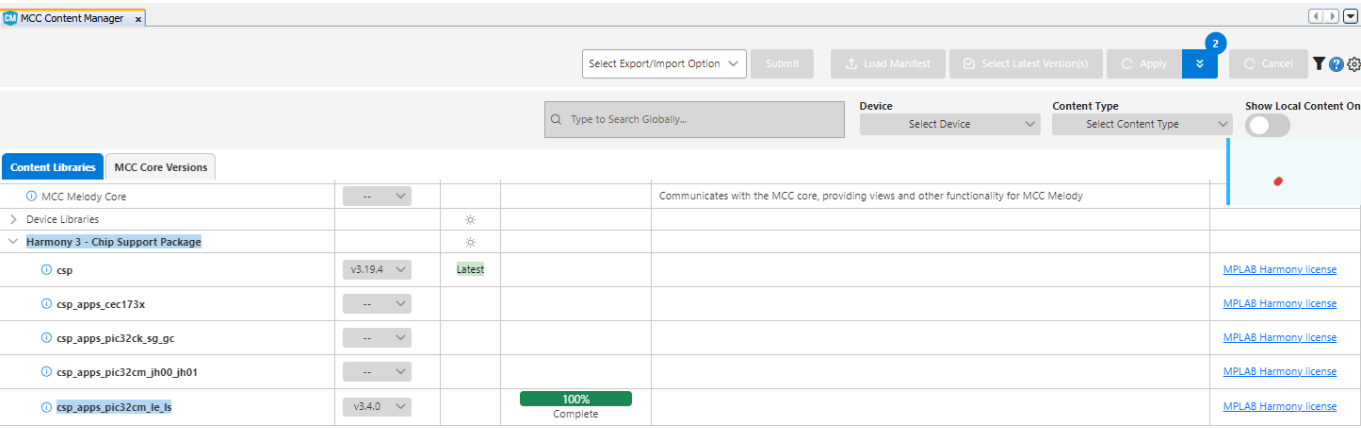
- Download content for MPLAB Code Configurator (MCC)



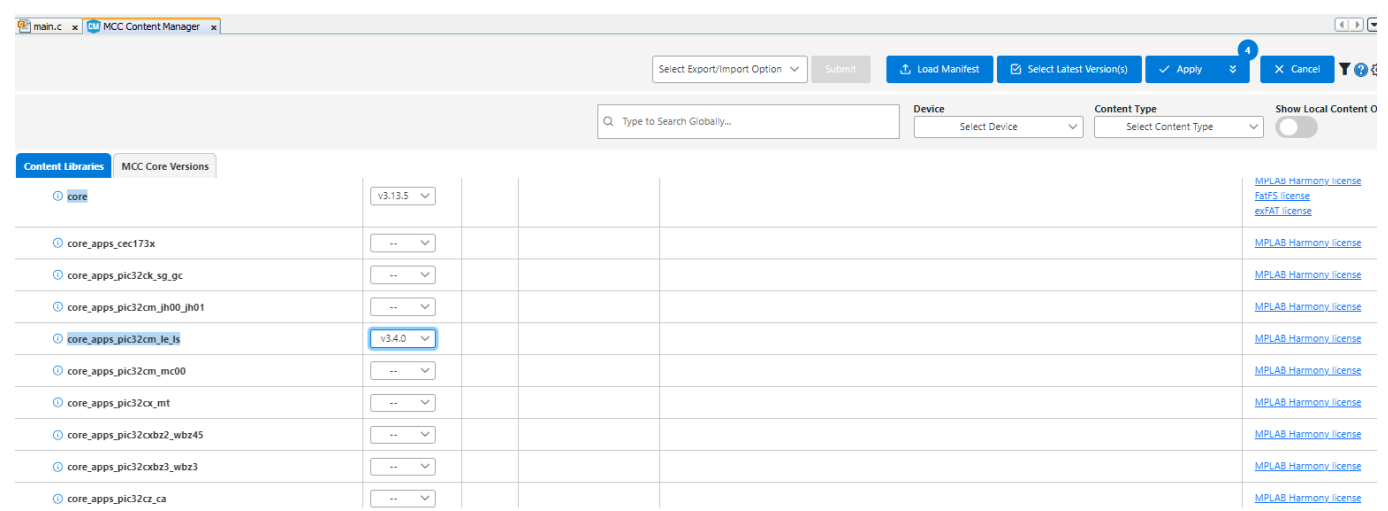
- On your first time opening, MCC will automatically ask you to download some files.

Board Packages

- Chip Support Package: proceed to also include `csp_apps_pic32cm_1e_1s` from under **Harmony 3 - Chip Support Packages**



- Core Apps Package: proceed to also include `core_apps_pic32cm_1e_1s` from under **Harmony 3 - Core**

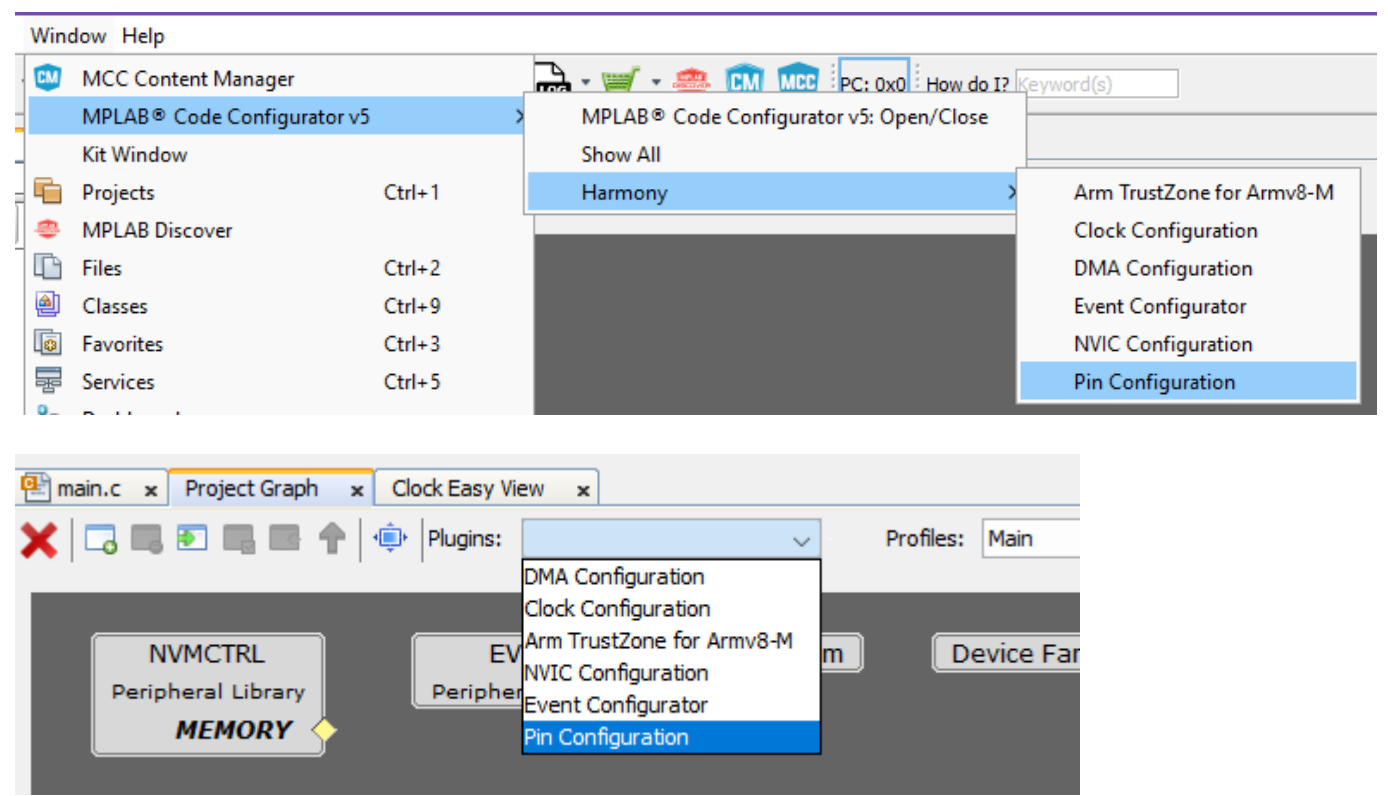


note: you can also add these packages later by opening up the **Content Manager** (CM button beside MCC) if you don't install them now

MCC Harmony Windows

We now proceed with using MCC to easily create some configurations for our Microcontroller

Open up **MCC** and navigate to **Pin Configuration** either through the ribbon or through the Project Graph



We can use a GUI to set up the pins of our Microcontroller. Proceed to set up PA15 as Output Pin as shown. Take note of the **Security Mode** being set as **NON-SECURE** as well.

main.cPin SettingsPin TablePin Diagram

Order: PinsTable ViewEasy View

Pin Number	Pin ID	Custom Name	Function	Mode	Direction	Latch	Pull Up	Pull Down	Drive Strength	Security Mode
19	VSS			Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL	SECURE
20	VDDPLL			Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL	SECURE
21	PA12		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL	SECURE
22	PA13		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL	SECURE
23	PA14		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL	SECURE
24	PA15	LED_PIN	GPIO	Digital	Out	High	<input type="checkbox"/>	<input type="checkbox"/>	STRONG	NON-SECURE

main.cPin SettingsPin TablePin Diagram

UnavailableAvailableAssigned

PIC32CM5164LS00048

PA0012PA0111PA0210PA039PA048PA057PA066PA075PA084PA093PA102PA111PA120PA131PA140PA15242322212019181716151413

484746454443424140393837363534333231302928272625

VDDVSSPA25PA24PA23PA22PA21PA20PA19PA18PA17PA16

PA08PA09PA10PA11AVSSPLL

VDDVSSVDDPLLPA12PA13PA14LED_PIN

PB03PB02PB01PA31PA30VDDVDDOUTVSSCOREVDDCORERESET_NVSSPB23PB22

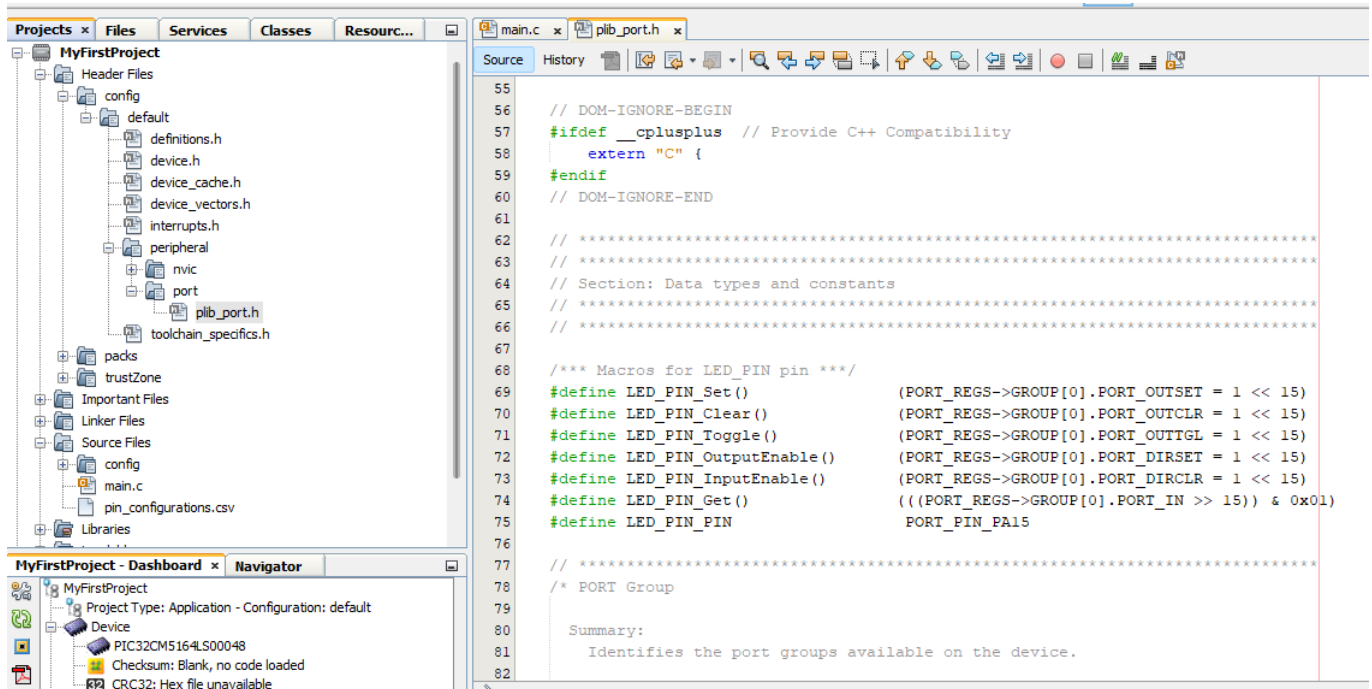
- Click on Generate to generate your configuration code

Project Resources

GenerateImport...Export

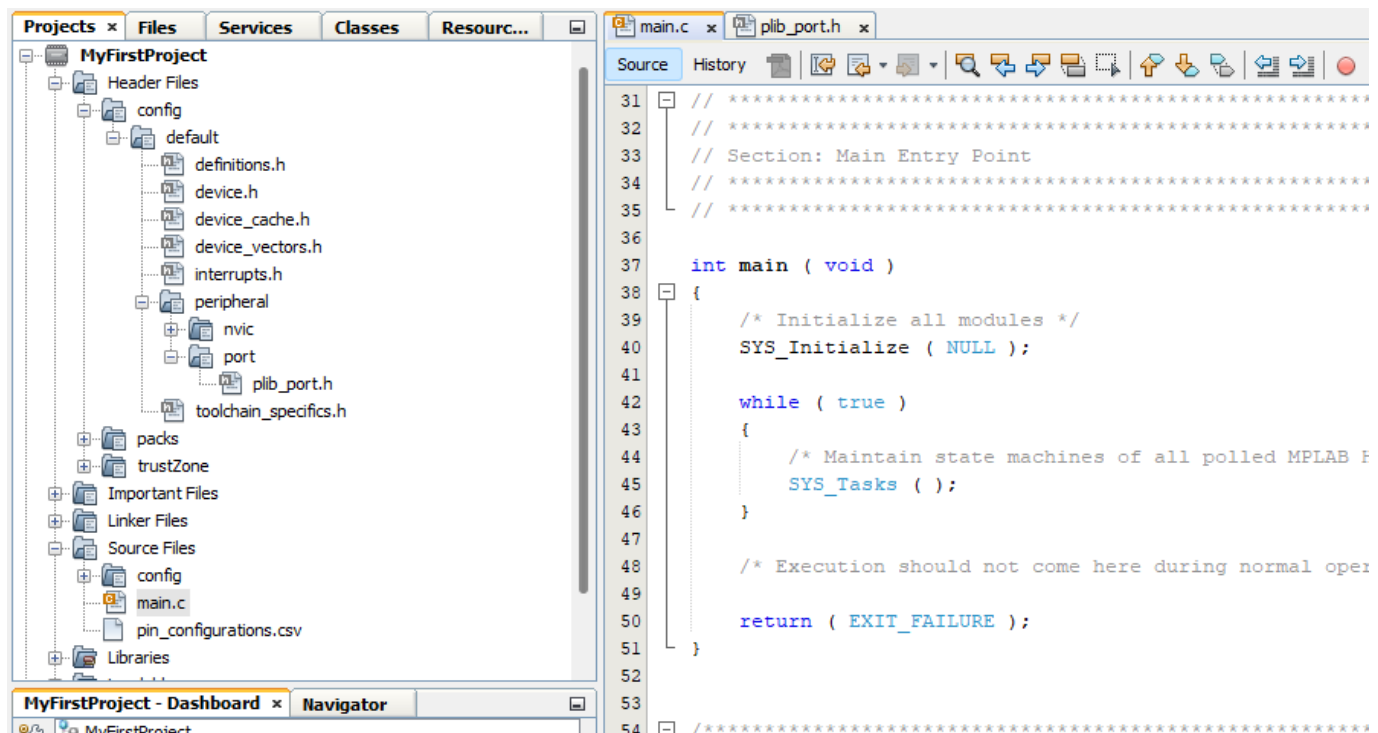
You should now see in your Project folder more files, these were auto generated by MCC. The interesting one here is the header file `plib_port.h`

19 / 21



You can use these newly declared definition in you code to control the Pin.

- Go to `main.c` under `Source Files`



Here is where you can start creating your application. Let's blink try and blink an LED by switching a pin between HIGH and LOW.

Guided Exercise: "MCC-Blinky!"

Blinky!

- in `main.c` add the function call `LED_PIN_Toggle()`; and also this `crude_delay_ms()` function before `main()`

```

int crude_ms_delay(int ms){
    int count = 0;
    unsigned int delay_count = ms * 12000;

    while(count < delay_count){
        asm("nop");
        count = count + 1;
    }
    return 0;
}

int main ( void )
{
    /* Initialize all modules */
    SYS_Initialize ( NULL );

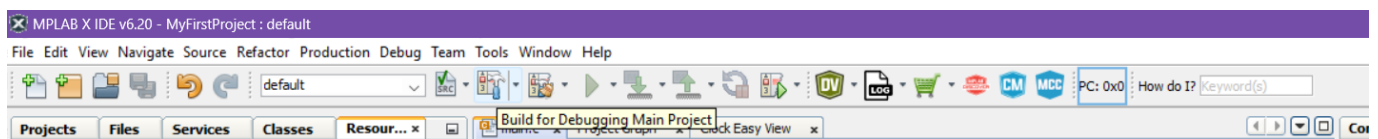
    while ( true )
    {
        LED_PIN_Toggle();
        crude_ms_delay(1000);
        /* Maintain state machines of all polled MPLAB Harmony modules. */
        SYS_Tasks ( );
    }

    /* Execution should not come here during normal operation */

    return ( EXIT_FAILURE );
}

```

Let us first try Building our Application. Press the **Build** Icon to compile your project. It's the one that looks like a hammer



If it builds, we are now ready to try and debug your application.