MPLAB First Steps

Installation

Install MPLAB X IDE

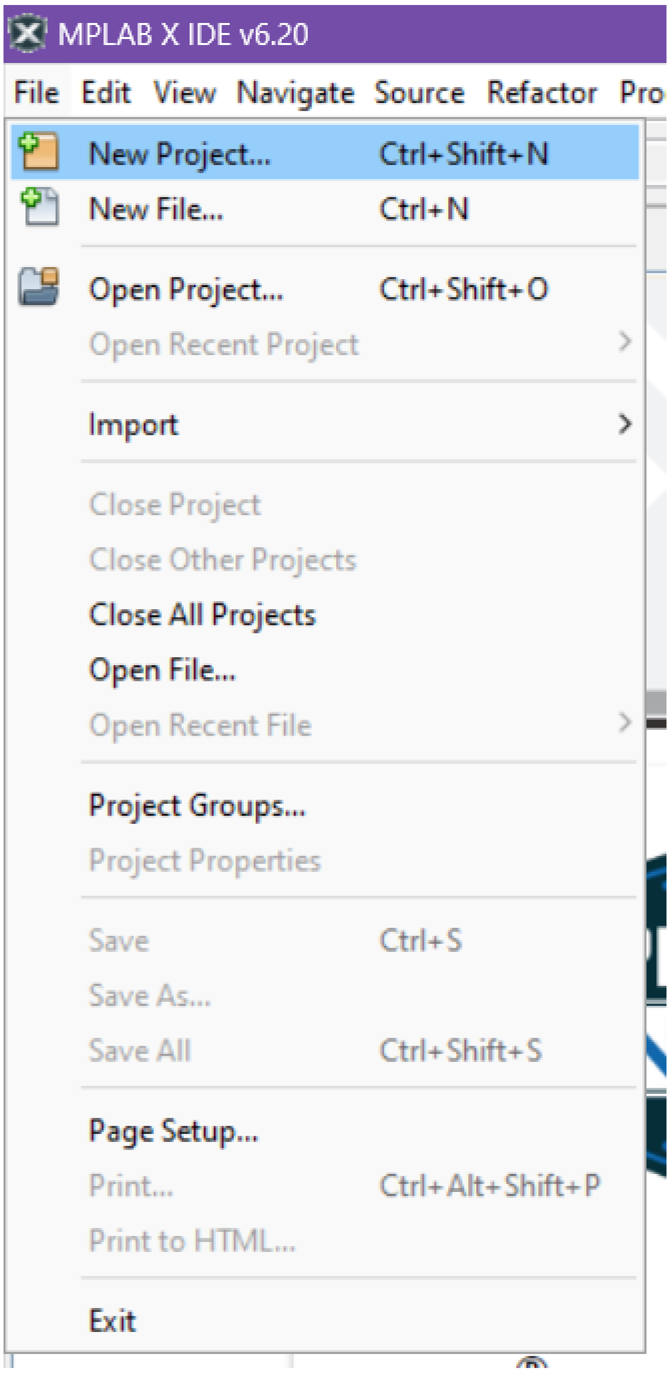
Install MPLAB X IDE through the official Microchip site: MPLAB X IDE

Install XC32 Compiler

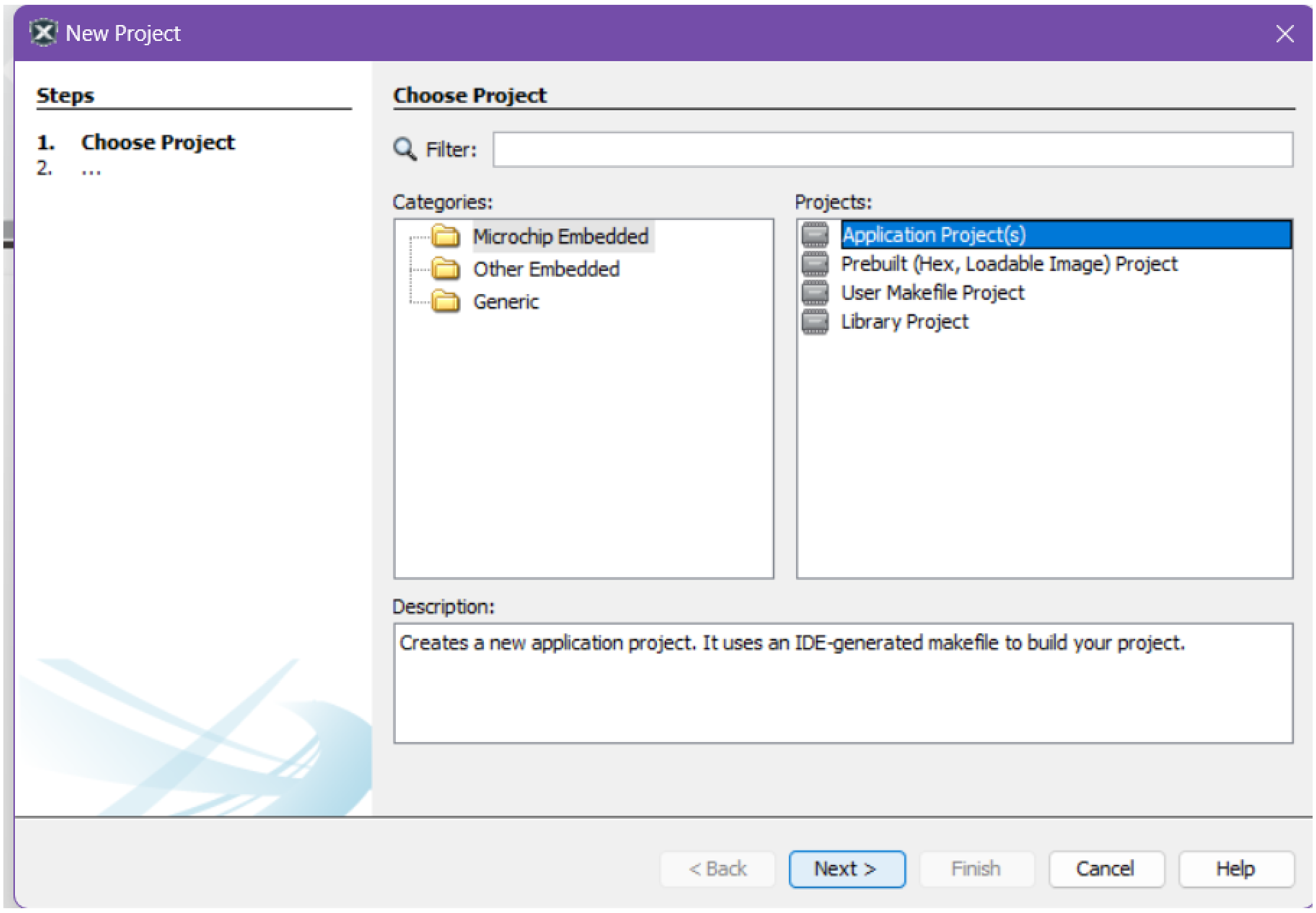
Install the XC32 COmpilers from Microchip: MPLAB xC32 Compilers

Your First Bare-Metal Project

- Create a New Project



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


New Project

Steps

1. Choose Project
2. ...

Choose Project

Filter:

Categories:
- Microchip Embedded
- Other Embedded
- Generic

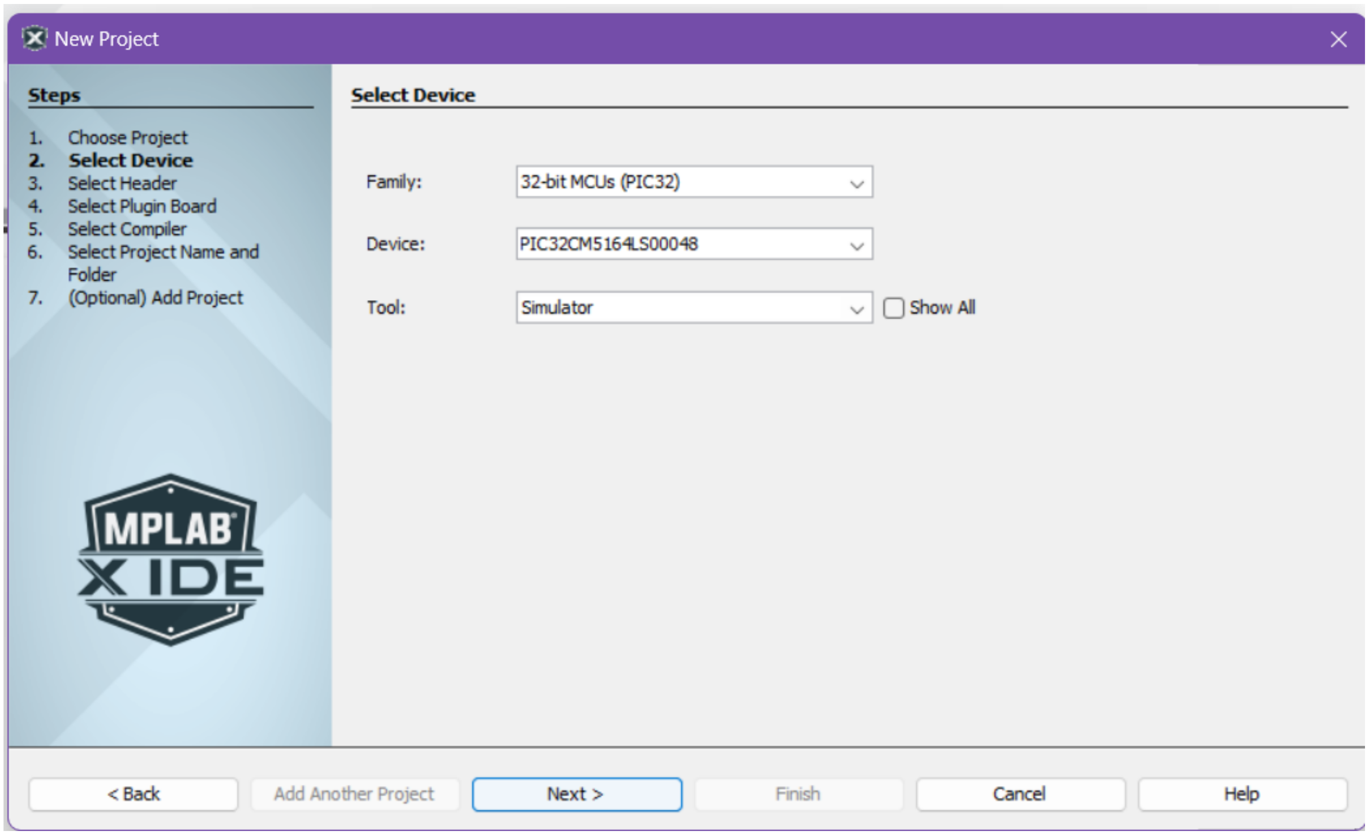
Projects:
- Application Project(s)
- Prebuilt (Hex, Loadable Image) Project
- User Makefile Project
- Library Project

Description:

Creates a new application project. It uses an IDE-generated makefile to build your project.

- 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



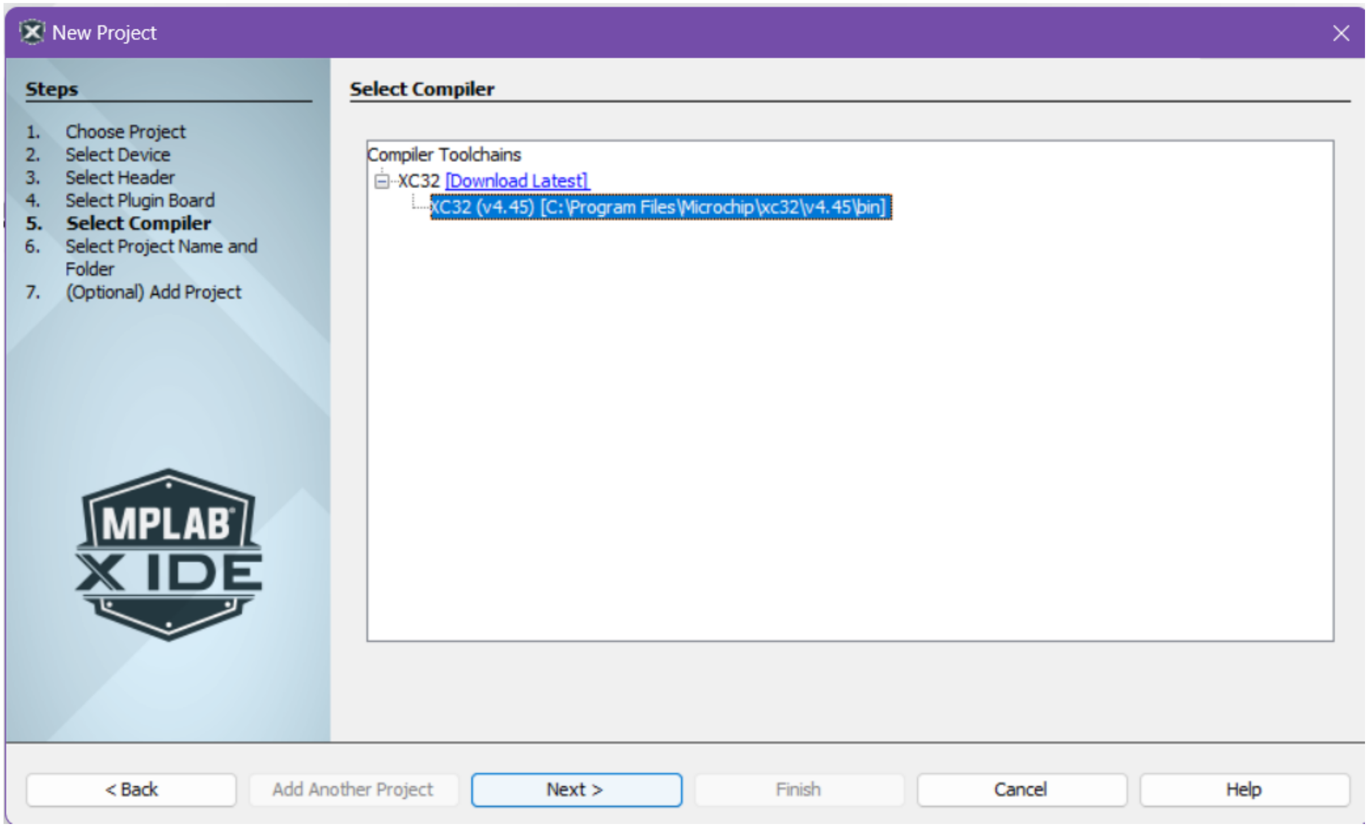
Select Device

Family: 32-bit MCUs (PIC32)

Device: PIC32CM5164LS00048

Tool: Simulator ☐ Show All

- 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*

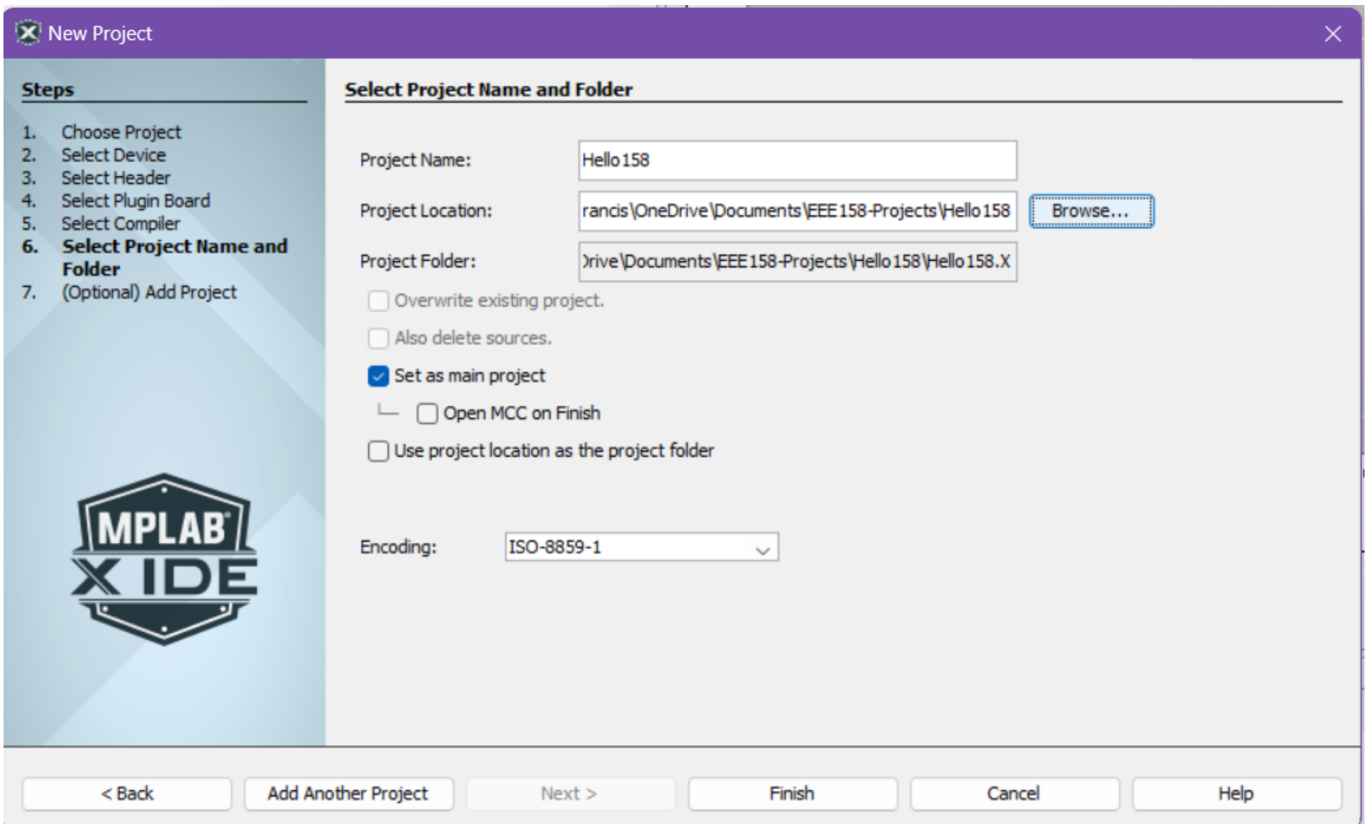


Select Compiler

Compiler Toolchains
XC32 [Download Latest]
XC32 (v4.45) [C:\Program Files\Microchip\xc32\v4.45\bin]

- 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)*



New Project

Steps

1. Choose Project
2. Select Device
3. Select Header
4. Select Plugin Board
5. Select Compiler
6. **Select Project Name and Folder**
7. (Optional) Add Project

Select Project Name and Folder

Project Name: Hello158

Project Location: rancis\OneDrive\Documents\EEE158-Projects\Hello158

Project Folder:)rive\Documents\EEE158-Projects\Hello158\Hello158.X

☐ Overwrite existing project.

☐ Also delete sources.

☑ Set as main project

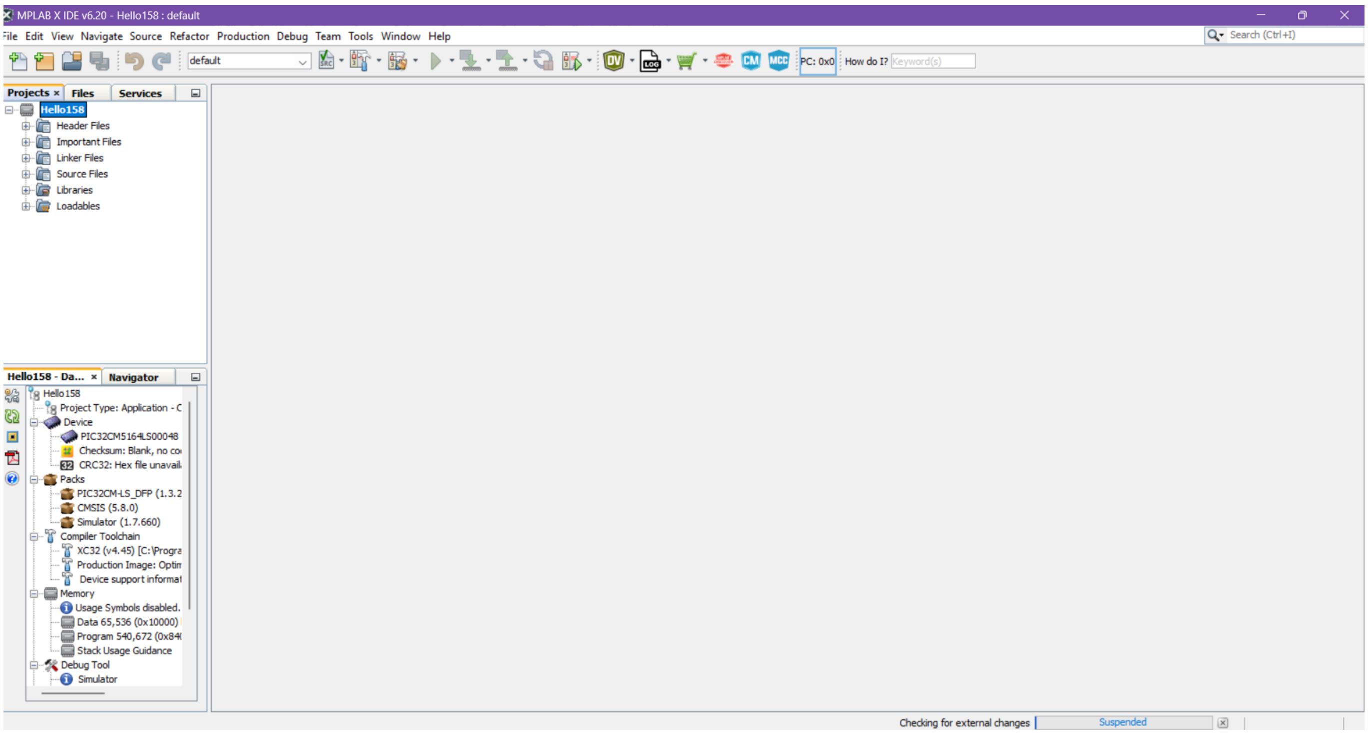
☐ Open MCC on Finish

☐ Use project location as the project folder

Encoding: ISO-8859-1

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

- Your First Project should now initialize

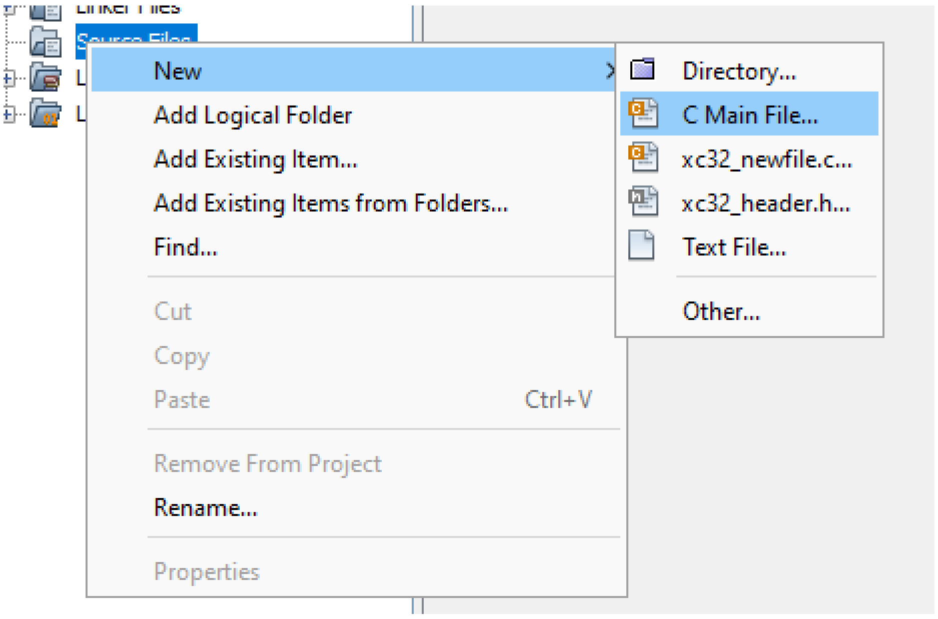


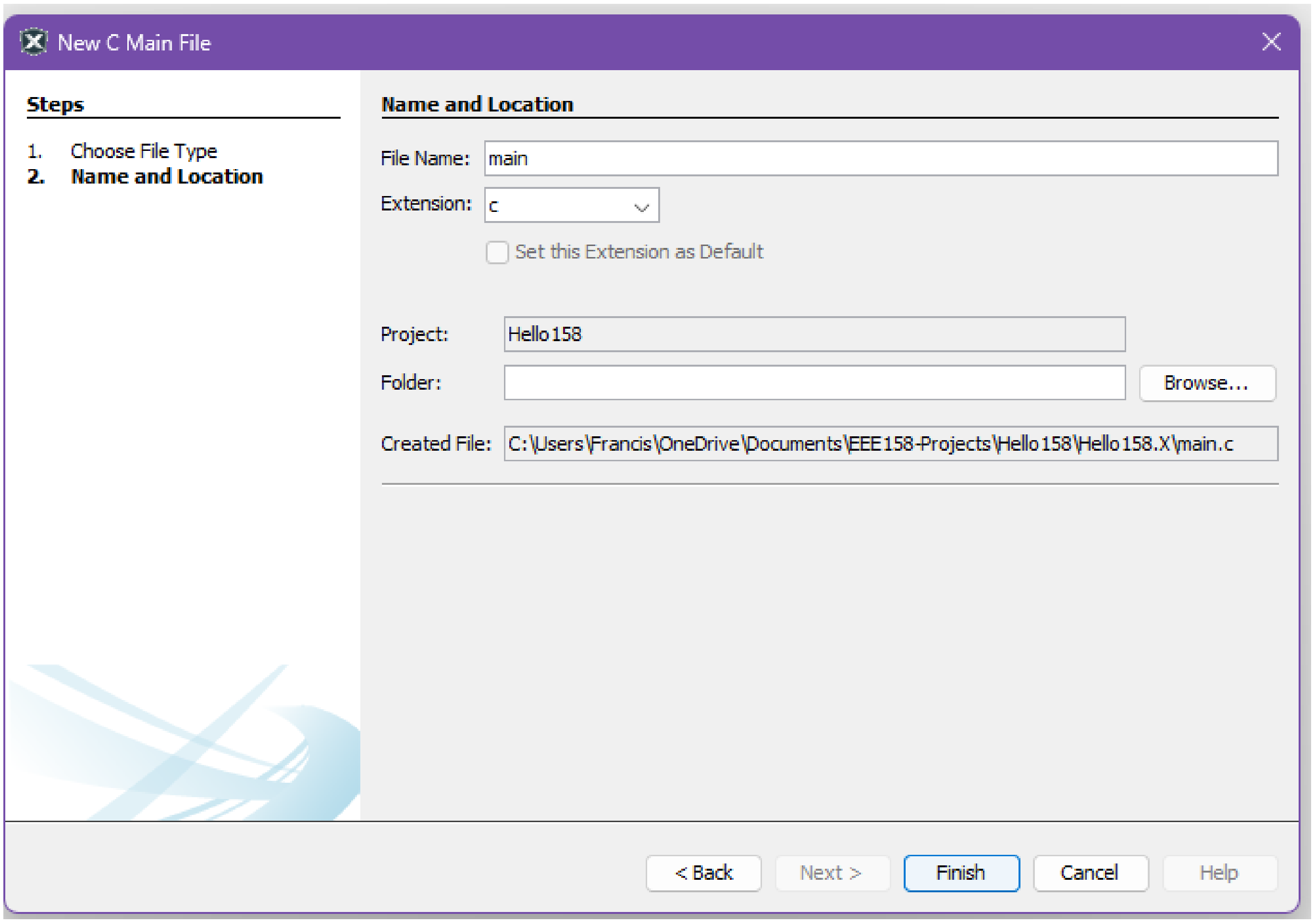
Let's now proceed to creating a `main.c` file so we can start coding our application.

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`





Name and Location

File Name: main

Extension: c

Project: Hello158

Folder:

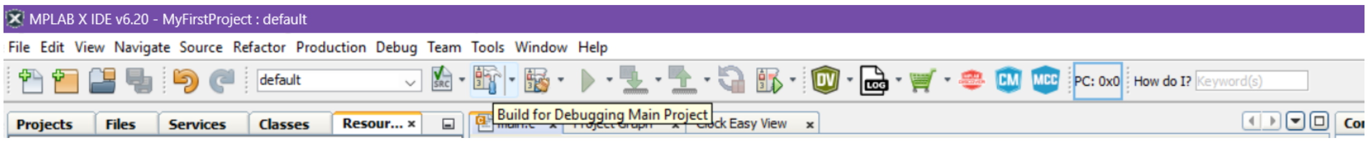
Created File: C:\Users\Francis\OneDrive\Documents\EEE158-Projects\Hello158\Hello158.X\main.c

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

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

```c
/*
 *
 */

int crude_delay(int count){
    for(int i; i < count; i++);
}

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

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


    while(1){
        //Toggle the Output of PA15
        PORT_REGS->GROUP[0].PORT_OUTTGL = (1 << 15);
        crude_delay(100);
    }
    return (EXIT_SUCCESS);
}
```

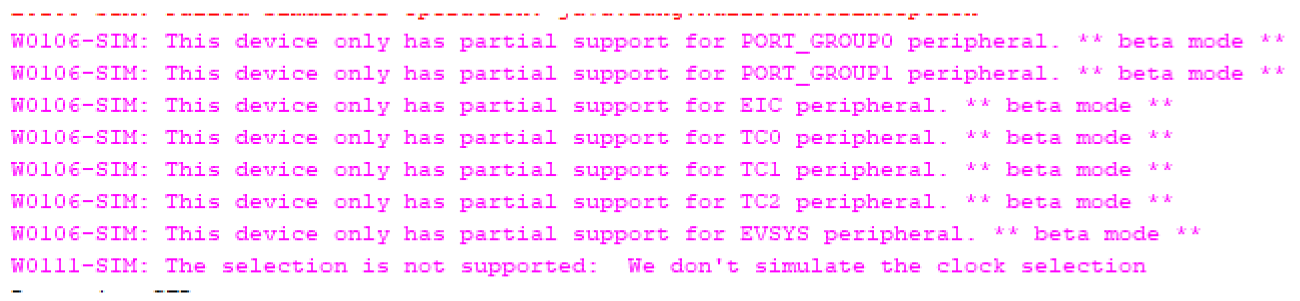
Let us first try Building our Application. Press the `Build` Icon to compile your project. It's the one that looks like a hammer


MPLAB X IDE v6.20 - MyFirstProject : default
File Edit View Navigate Source Refactor Production Debug Team Tools Window Help
default
Build for Debugging Main Project
PC: 0x0
How do I? Keyword(s)
Projects Files Services Classes Resour...
Clock Easy View

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

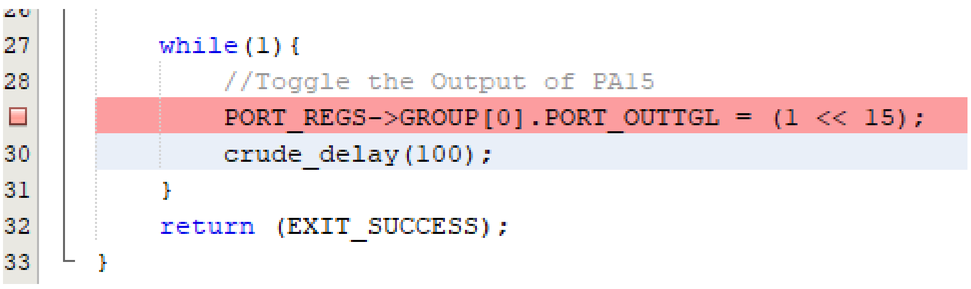
Simulator Debugging

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

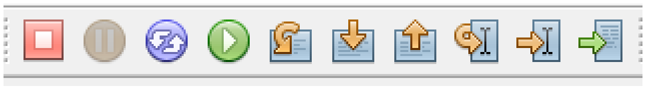


```
    while(1){
        //Toggle the Output of PA15
        PORT_REGS->GROUP[0].PORT_OUTTGL = (1 << 15);
        crude_delay(100);
    }
    return (EXIT_SUCCESS);
}
```

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

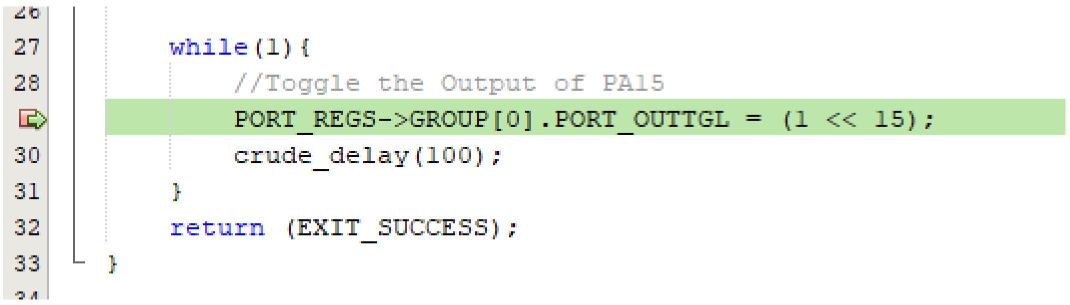
Application Execution Control

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



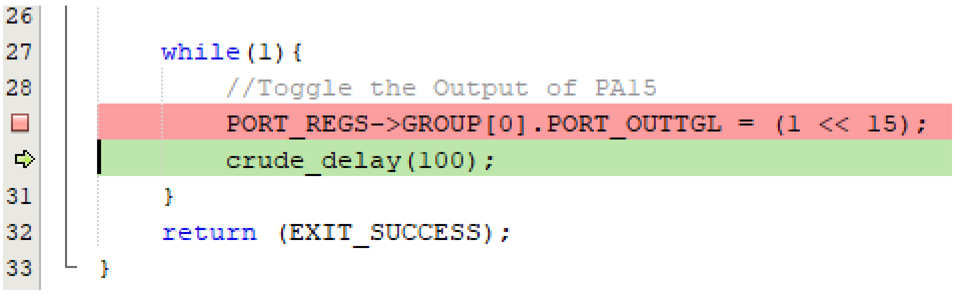
- `Finish Debugger Session` - End Debugging
- `Pause` - Pause Application Execution
- `Reset` - Reset Application Execution
- `Continue` - Resume Application Execution
- `Step Over` - Step over a line of your Application
- `Step Into` - Step into a a method in your Application
- `Step Out` - Step out 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



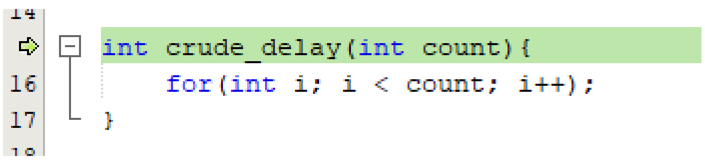
```
    while(1){
        //Toggle the Output of PA15
        PORT_REGS->GROUP[0].PORT_OUTTGL = (1 << 15);
        crude_delay(100);
    }
    return (EXIT_SUCCESS);
}
```

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



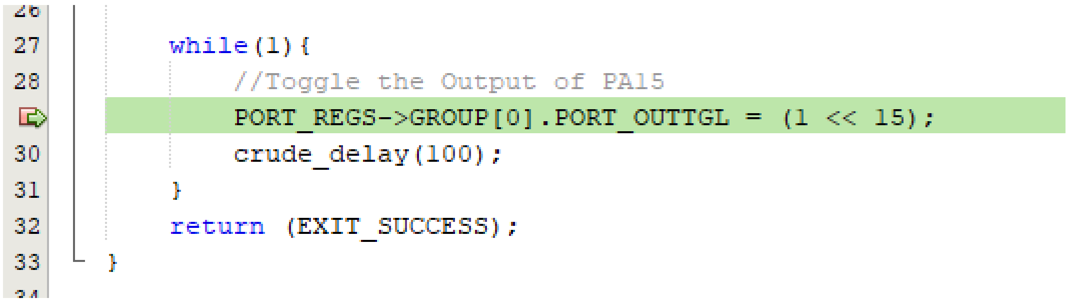
```
    while(1){
        //Toggle the Output of PA15
        PORT_REGS->GROUP[0].PORT_OUTTGL = (1 << 15);
        crude_delay(100);
    }
    return (EXIT_SUCCESS);
}
```

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



```
int crude_delay(int count){
    for(int i; i < count; i++);
}
```

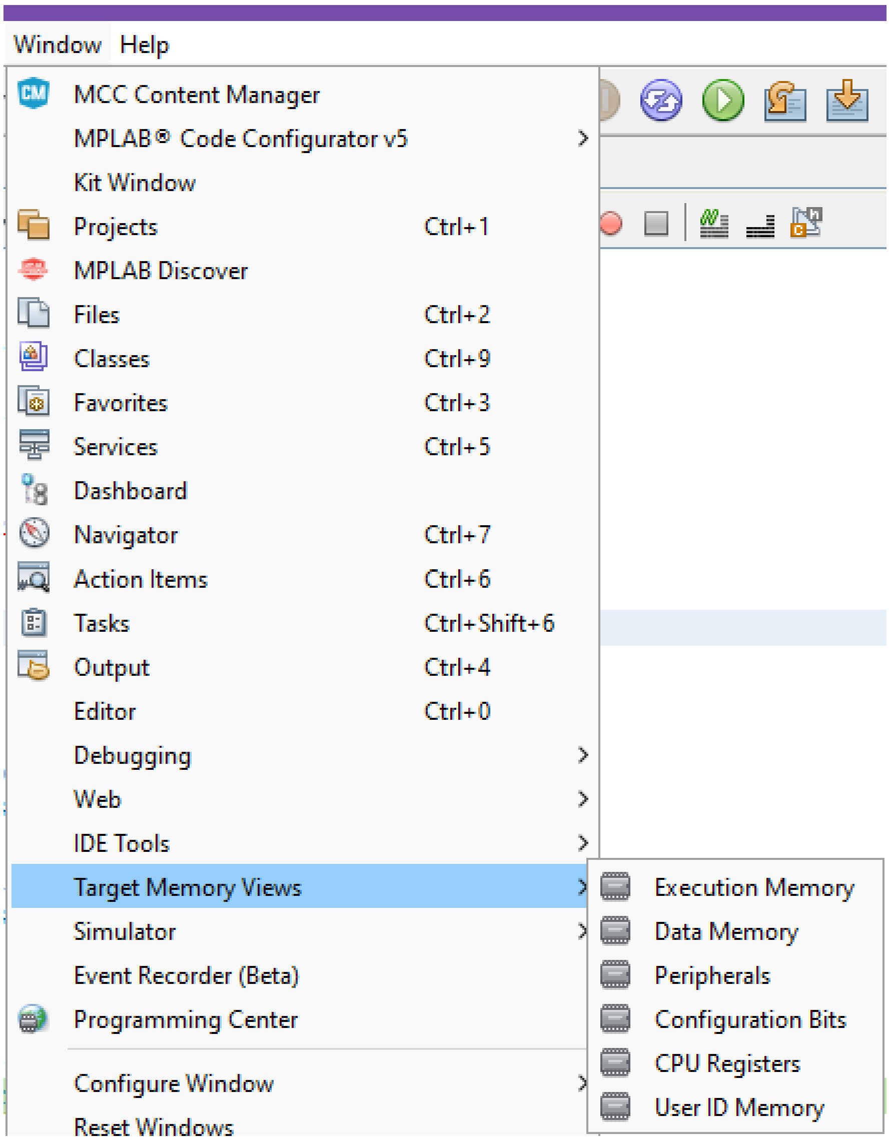
Press the Continue button and after a while, you should hit the breakpoint again



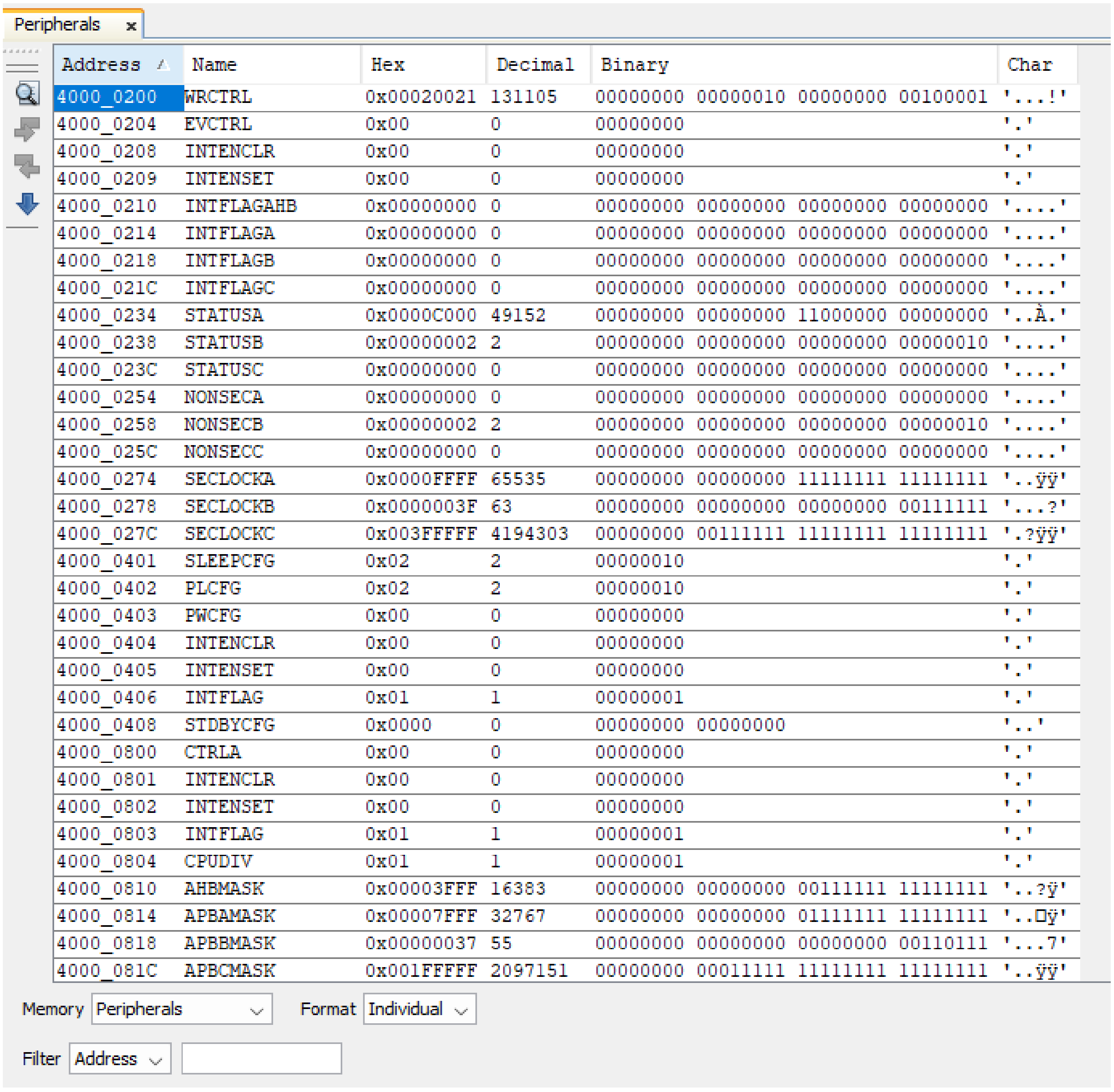
```
    while(1){
        //Toggle the Output of PA15
        PORT_REGS->GROUP[0].PORT_OUTTGL = (1 << 15);
        crude_delay(100);
    }
    return (EXIT_SUCCESS);
}
```

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.

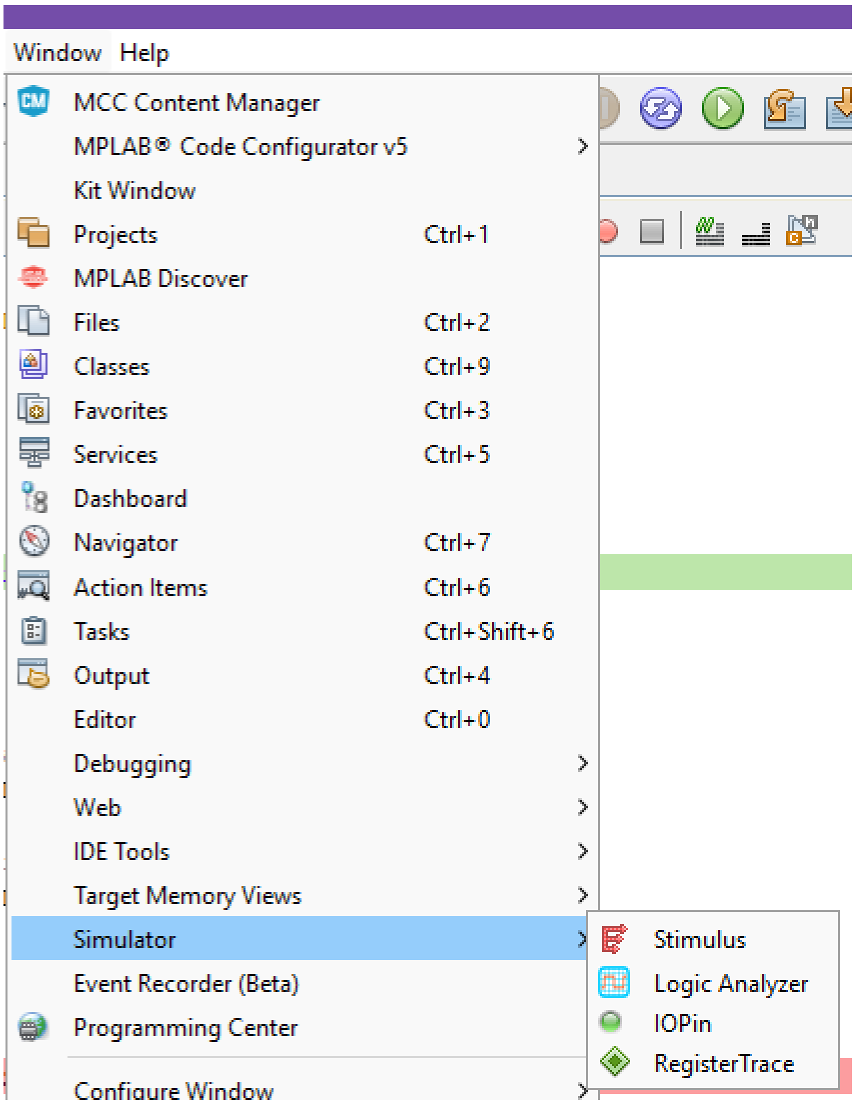


Address	Name	Hex	Decimal	Binary	Char
4000_0200	WRCTRL	0x00020021	131105	00000000 00000010 00000000 00100001	'...!'
4000_0204	EVCTRL	0x00	0	00000000	'.'
4000_0208	INTENCLR	0x00	0	00000000	'.'
4000_0209	INTENSET	0x00	0	00000000	'.'
4000_0210	INTFLAGAHB	0x00000000	0	00000000 00000000 00000000 00000000	'....'
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	'..À.'
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	0x003FFFFF	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	0x00003FFF	16383	00000000 00000000 00111111 11111111	'..?ÿ'
4000_0814	APBAMASK	0x00007FFF	32767	00000000 00000000 01111111 11111111	'..□ÿ'
4000_0818	APBBMASK	0x00000037	55	00000000 00000000 00000000 00110111	'...7'
4000_081C	APBCMASK	0x001FFFFF	2097151	00000000 00011111 11111111 11111111	'..ÿÿ'

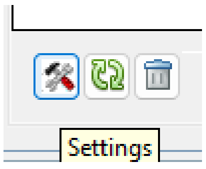
This allows us to directly check the contents of the registers and is useful for debugging.

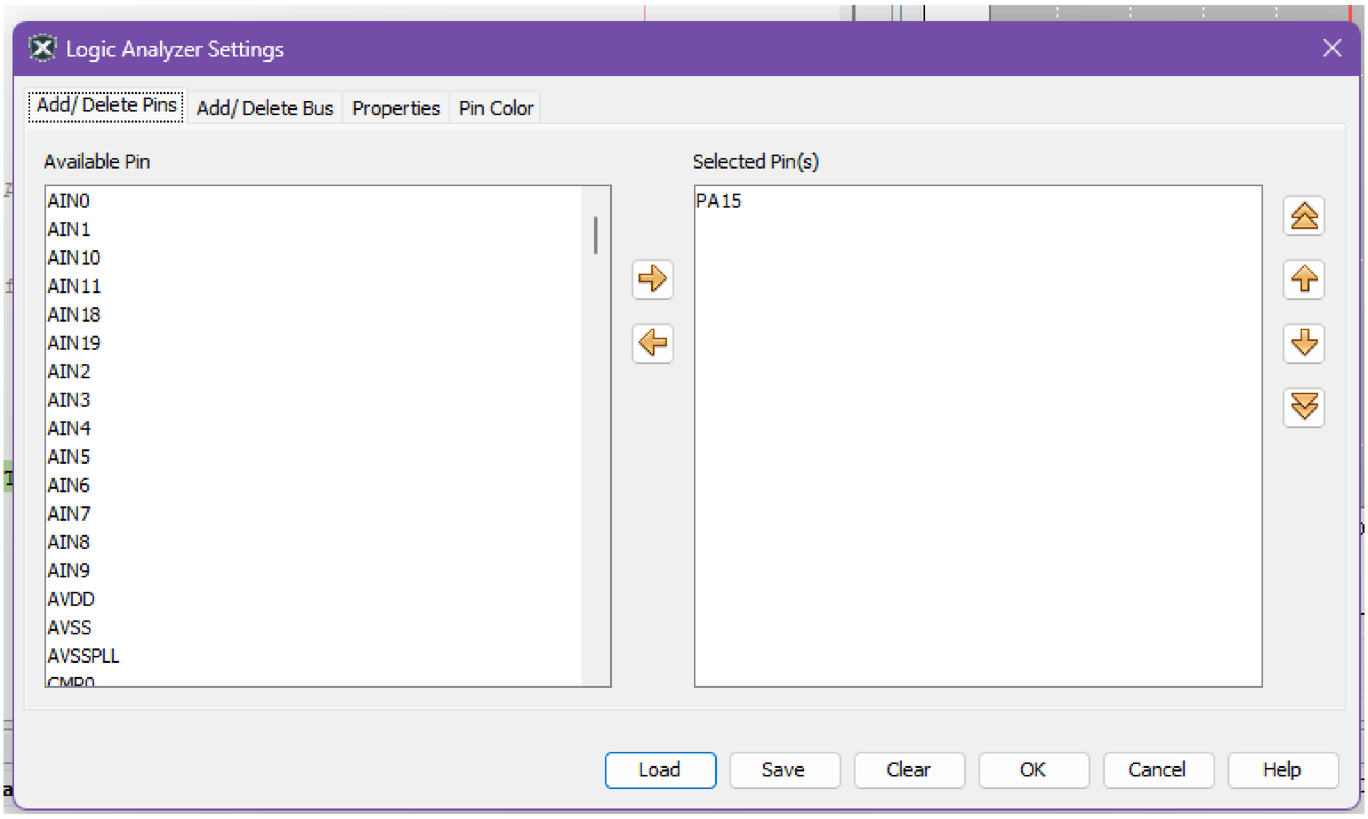
Logic Analyzer

You can open up the logic analyzer by going to 'Window' -> 'Simulator' -> 'Logic Analyzer'

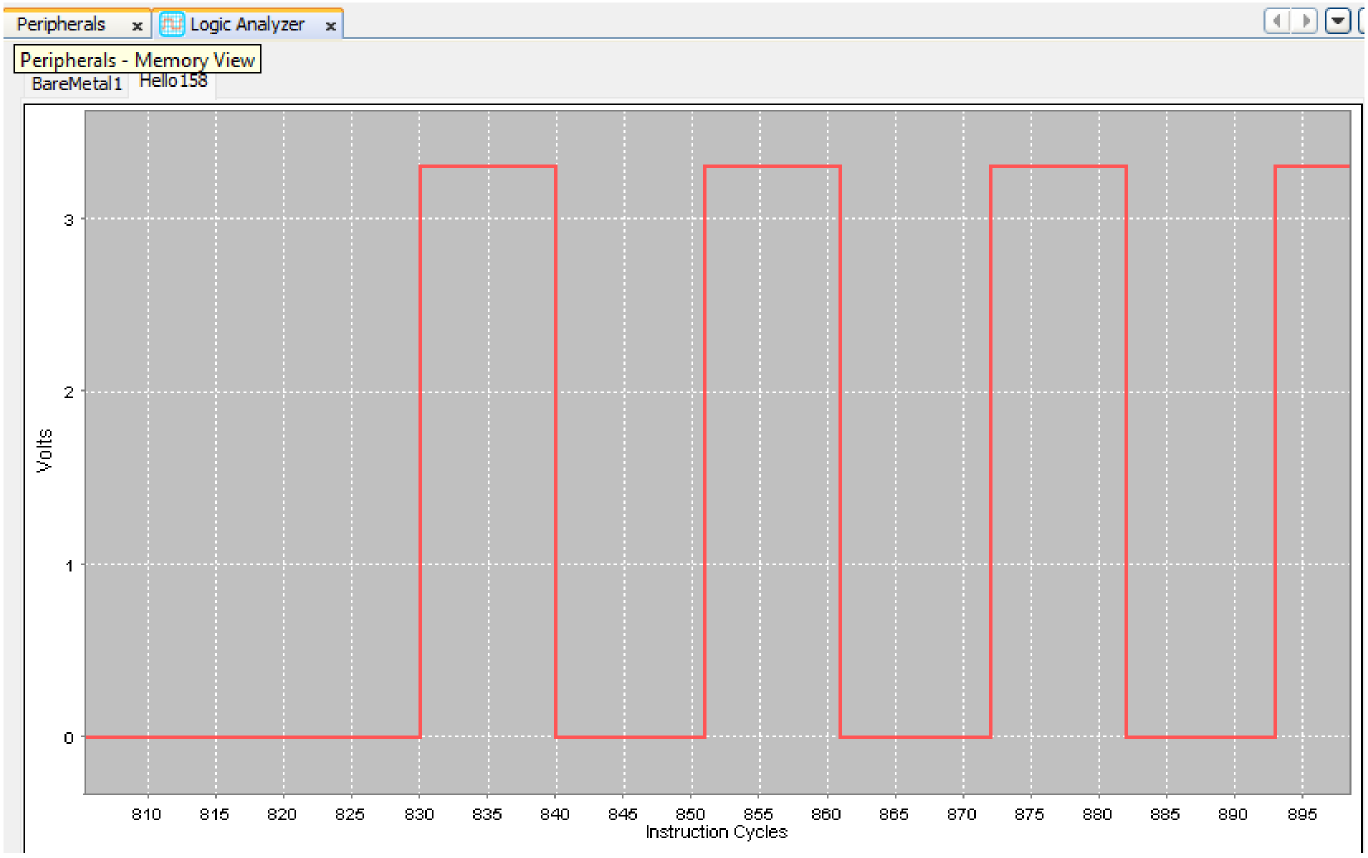


You can watch over a pin's state by adding it. Press the `Settings` icon on the logic analyzer window and add `PA15` to `Selected Pins`





You can now use the Logic Analyzer to view how `PA15` toggles between 0V and 3.3V

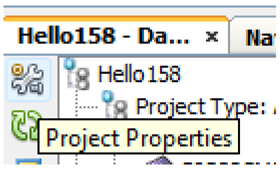


Hardware Debugging

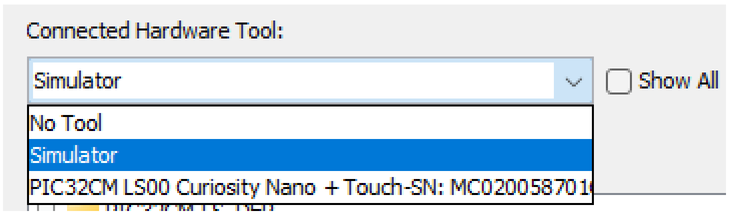
Hardware debugging is similar to Simulator debugging and you can perform any of the Application Execution Controls and Memory View functionality.

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


Connected Hardware Tool:

Simulator

Show All

No Tool

Simulator

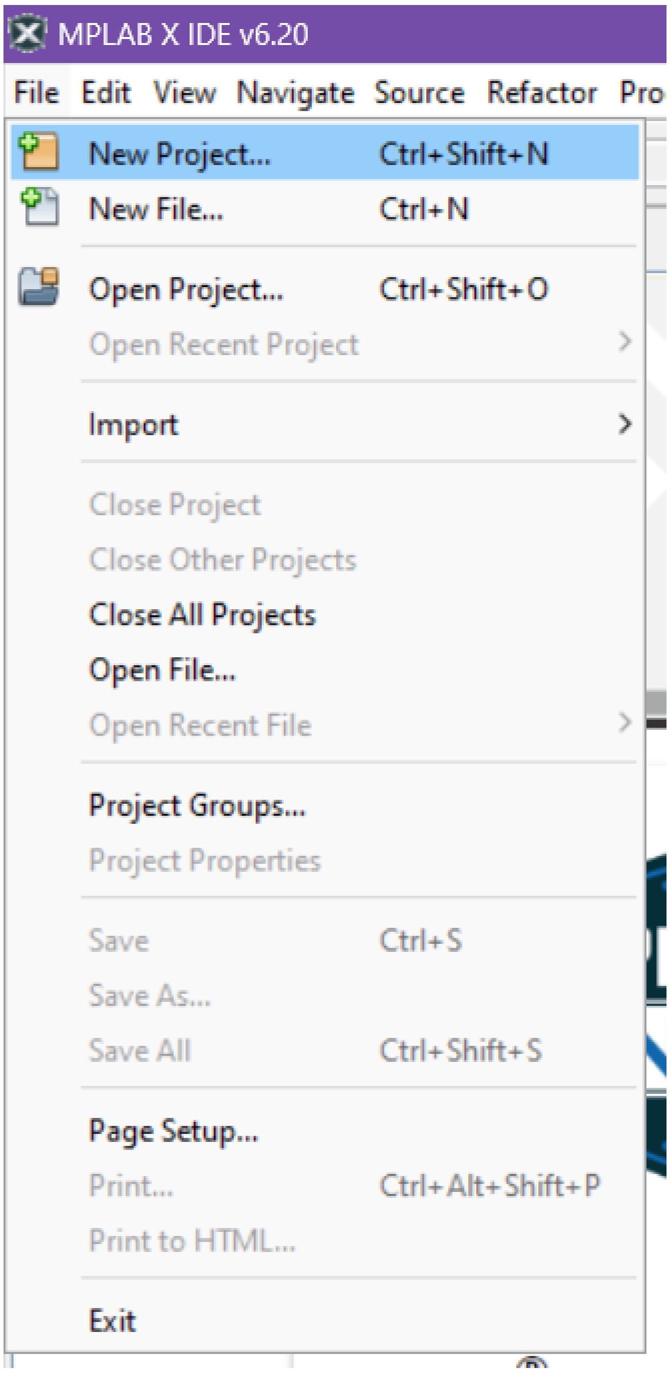
PIC32CM LS00 Curiosity Nano +Touch-SN: MC0200587010

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

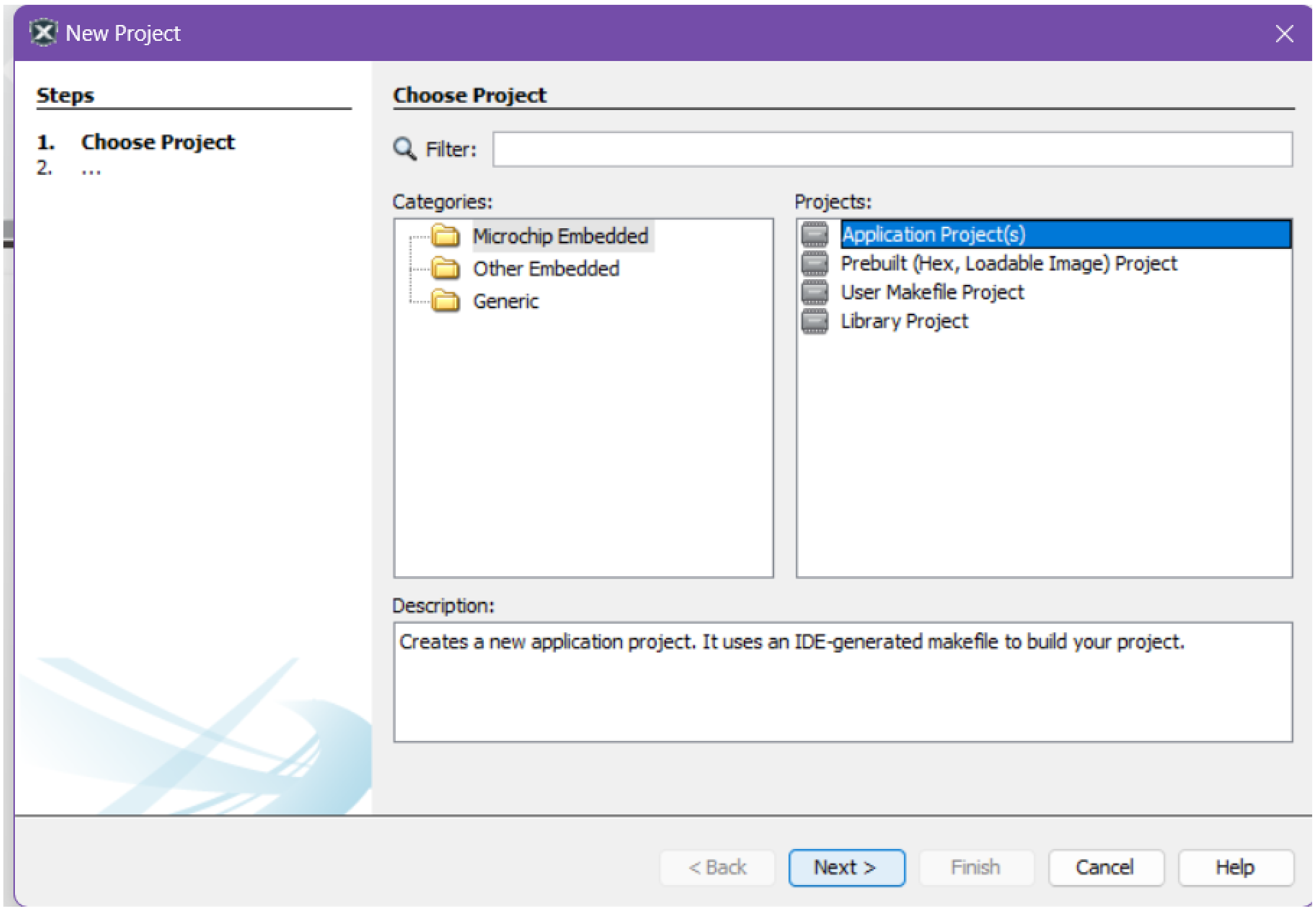
Your First MCC Project

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



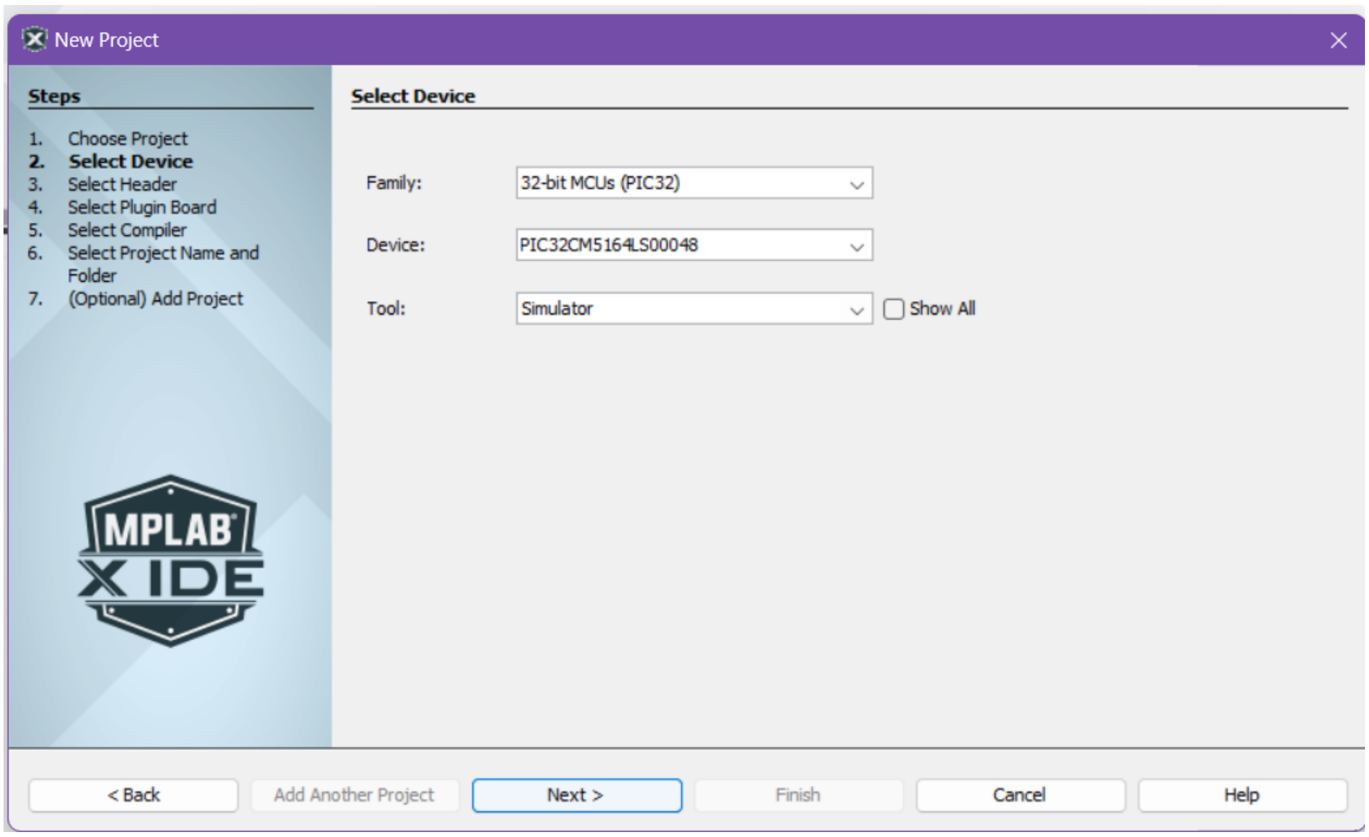
- Choose `Application Project(s)` then click `Next >`



Creates a new application project. It uses an IDE-generated makefile to build your project.

- 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



Select Device

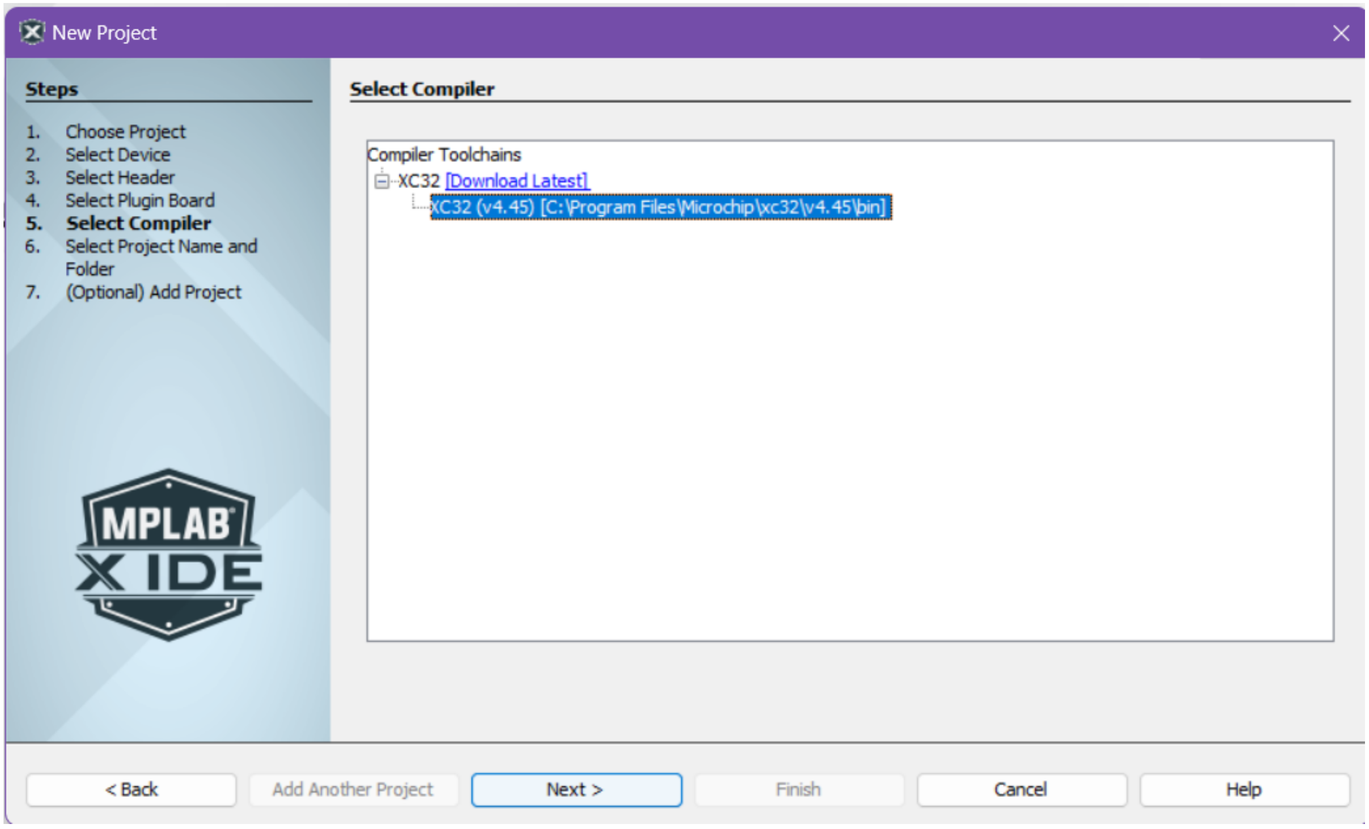
Family: 32-bit MCUs (PIC32)

Device: PIC32CM5164LS00048

Tool: Simulator

Show All

- 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 Compiler

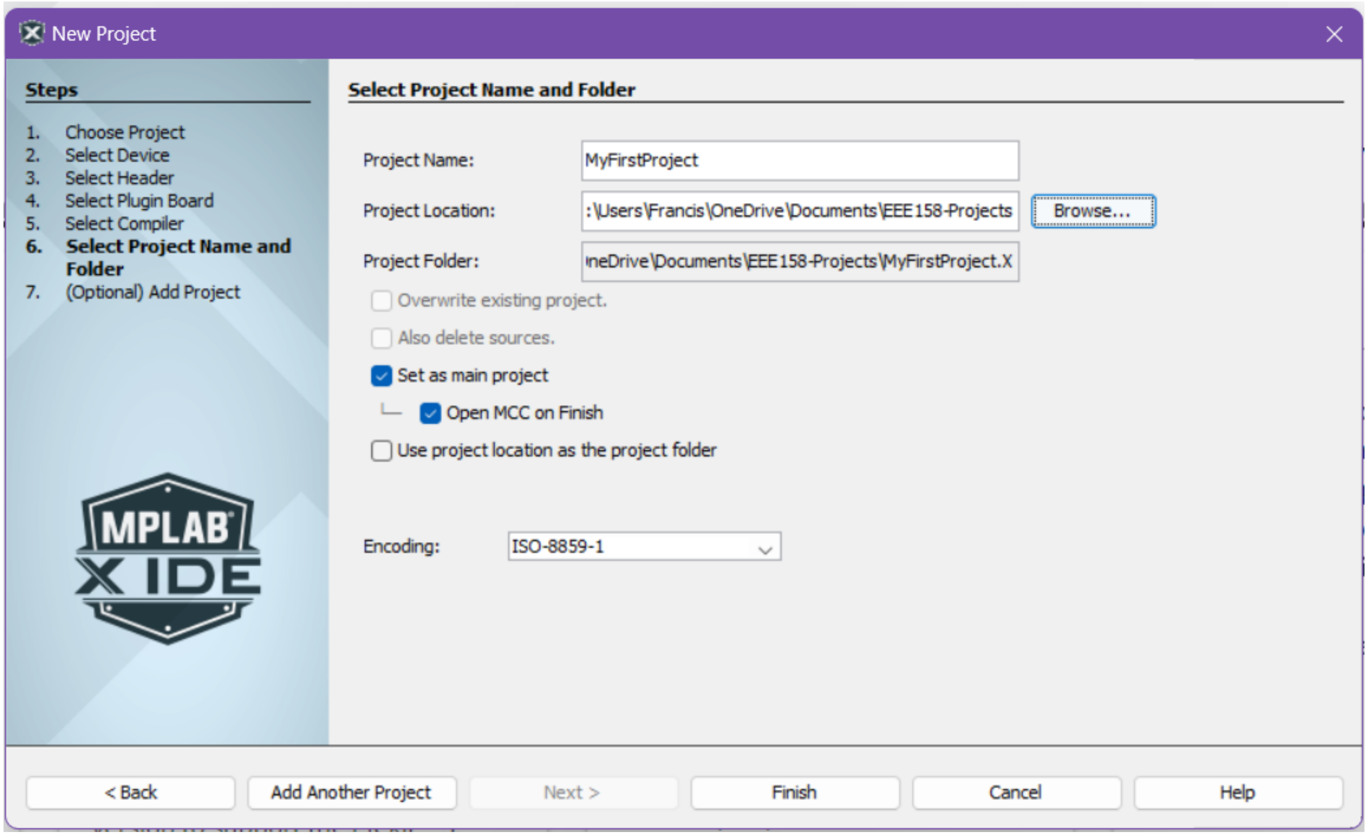
Compiler Toolchains

XC32 [Download Latest]

XC32 (v4.45) [C:\Program Files\Microchip\xc32\v4.45\bin]

- 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)*

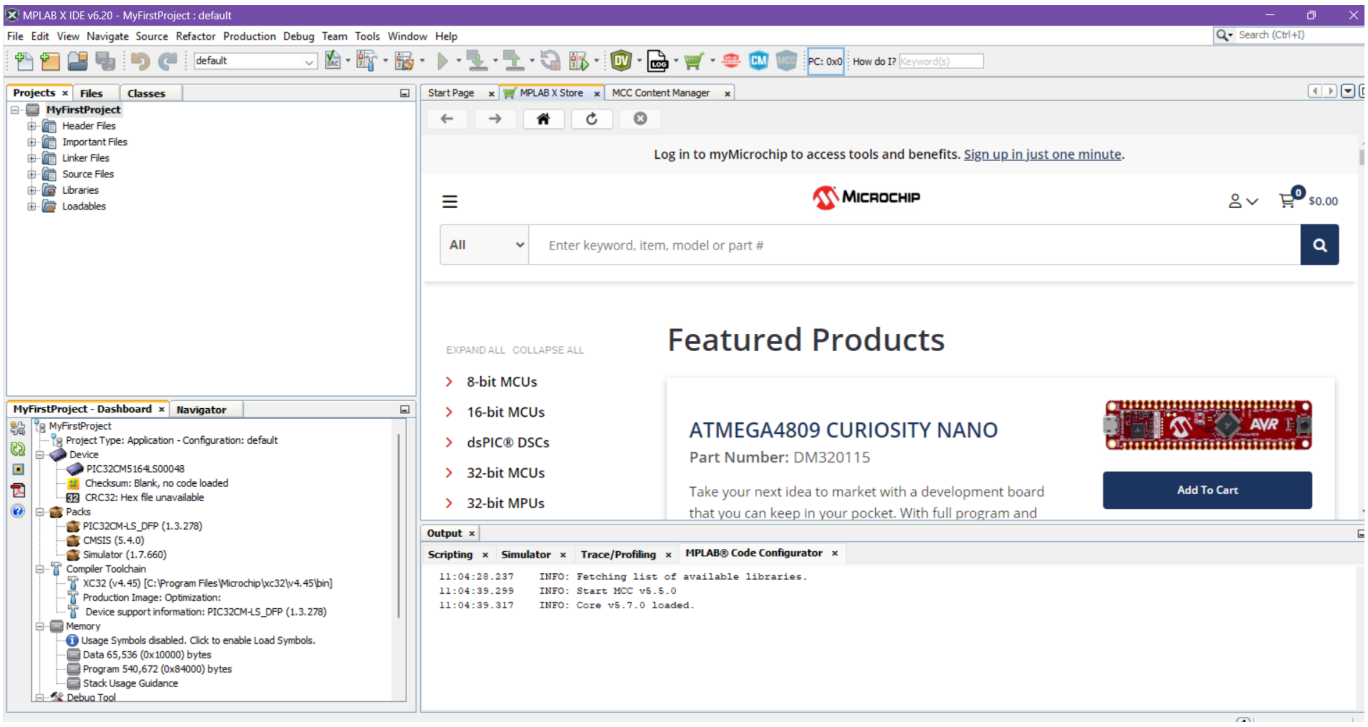

New Project

Steps
1. Choose Project
2. Select Device
3. Select Header
4. Select Plugin Board
5. Select Compiler
6. **Select Project Name and Folder**
7. (Optional) Add Project

Select Project Name and Folder

Project Name: MyFirstProject
Project Location: :\Users\Francis\OneDrive\Documents\EEE158-Projects
Project Folder: ıneDrive\Documents\EEE158-Projects\MyFirstProject.X
☐ Overwrite existing project.
☐ Also delete sources.
☑ Set as main project
☑ Open MCC on Finish
☐ Use project location as the project folder
Encoding: ISO-8859-1

- Your First Project should now initialize


MPLAB X IDE v6.20 - MyFirstProject : default

File Edit View Navigate Source Refactor Production Debug Team Tools Window Help
Projects Files Classes
Start Page MPLAB X Store MCC Content Manager

MyFirstProject
- Header Files
- Important Files
- Linker Files
- Source Files
- Libraries
- Loadables

Log in to myMicrochip to access tools and benefits. Sign up in just one minute.

Featured Products

8-bit MCUs, 16-bit MCUs, dsPIC® DSCs, 32-bit MCUs, 32-bit MPUs

ATMEGA4809 CURIOSITY NANO
Part Number: DM320115
Take your next idea to market with a development board that you can keep in your pocket. With full program and

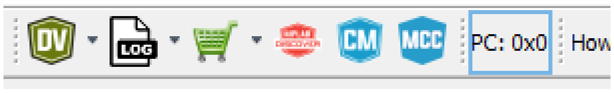
MyFirstProject - Dashboard

MyFirstProject
- Project Type: Application - Configuration: default
- Device
  - PIC32CM5164LS00048
  - Checksum: Blank, no code loaded
  - CRC32: Hex file unavailable
- Packs
  - PIC32CM-LS_DFP (1.3.278)
  - CMSIS (5.4.0)
  - Simulator (1.7.660)
- Compiler Toolchain
  - XC32 (v4.45) [C:\Program Files\Microchip\xc32\v4.45\bin]
  - Production Image: Optimization:
  - Device support information: PIC32CM-LS_DFP (1.3.278)
- Memory
  - Usage Symbols disabled. Click to enable Load Symbols.
  - Data 65,536 (0x10000) bytes
  - Program 540,672 (0x84000) bytes
  - Stack Usage Guidance
- Debug Tool

Output: Scripting, Simulator, Trace/Profiling, MPLAB® Code Configurator

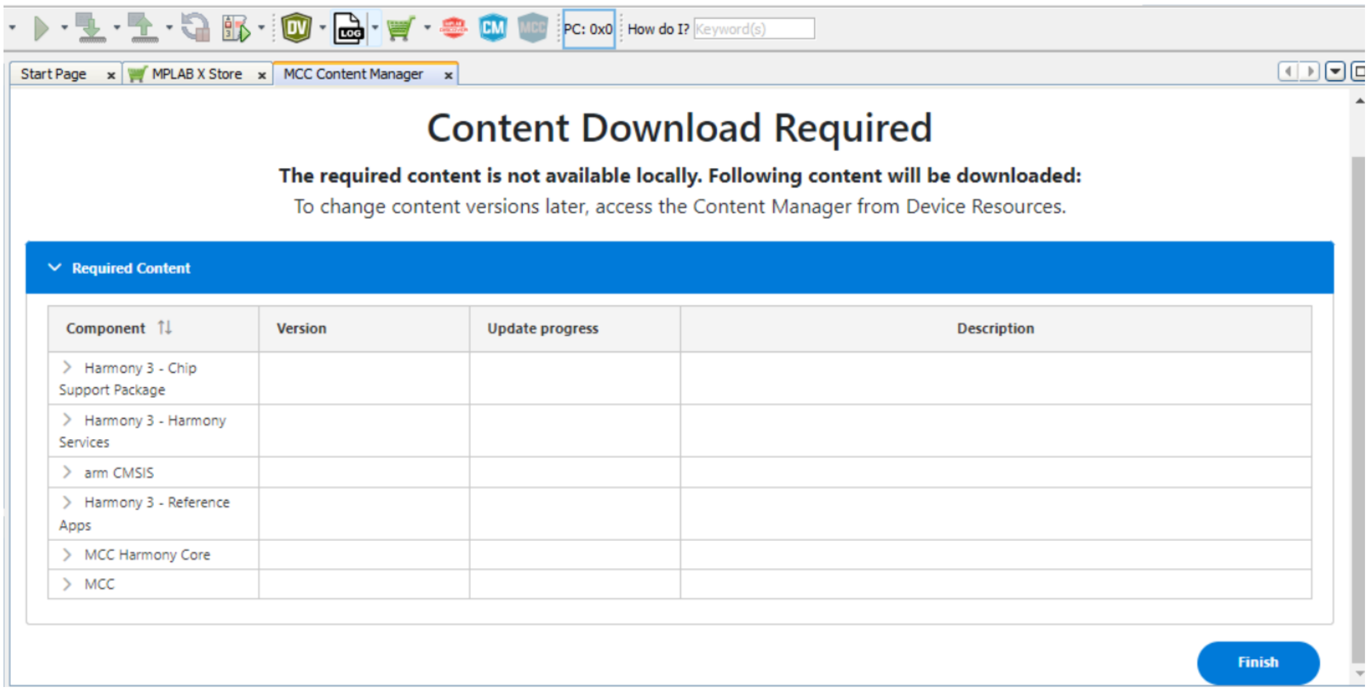
```
11:04:28.237    INFO: Fetching list of available libraries.
11:04:39.299    INFO: Start MCC v5.5.0
11:04:39.317    INFO: Core v5.7.0 loaded.
```

- 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)



Content Download Required

The required content is not available locally. Following content will be downloaded:
To change content versions later, access the Content Manager from Device Resources.

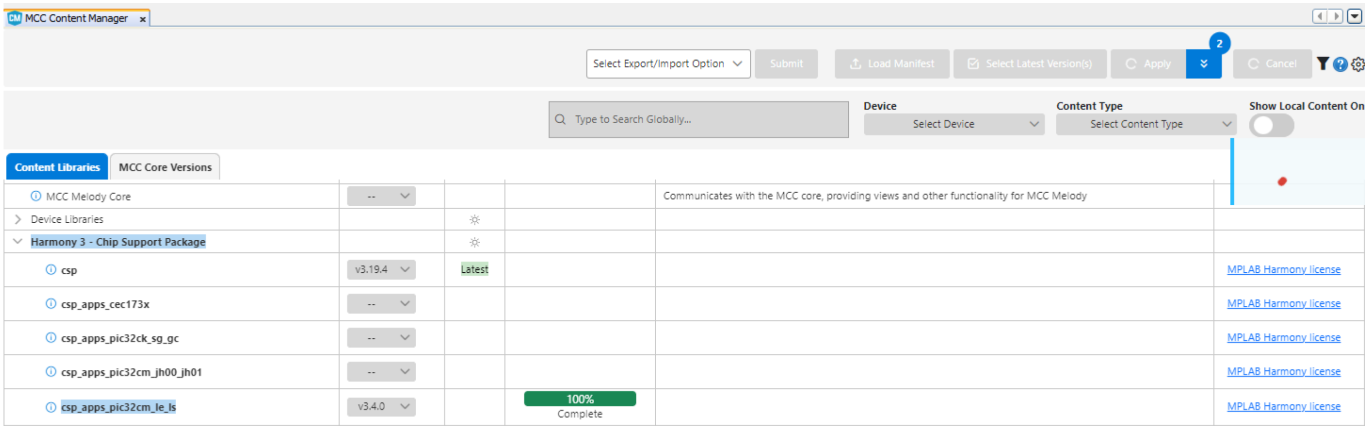
Required Content

Component	Version	Update progress	Description
Harmony 3 - Chip Support Package			
Harmony 3 - Harmony Services			
arm CMSIS			
Harmony 3 - Reference Apps			
MCC Harmony Core			
MCC			

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

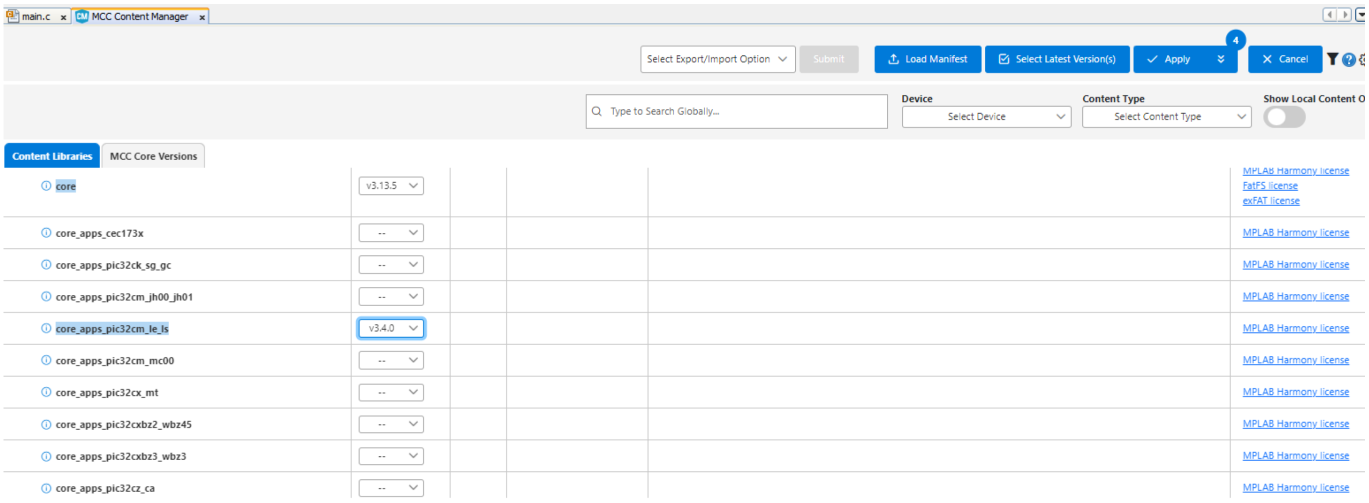
Board Packages

- Chip Support Package: proceed to also include `csp_apps_pic32cm_le_ls` from under `Harmony 3 - Chip Support Packages`



MCC Melody Core	--			Communicates with the MCC core, providing views and other functionality for MCC Melody	
Device Libraries					
Harmony 3 - Chip Support Package					
csp	v3.19.4	Latest			MPLAB Harmony license
csp_apps_cec173x	--				MPLAB Harmony license
csp_apps_pic32ck_sg_gc	--				MPLAB Harmony license
csp_apps_pic32cm_jh00_jh01	--				MPLAB Harmony license
csp_apps_pic32cm_le_ls	v3.4.0		100% Complete		MPLAB Harmony license

- Core Apps Package: proceed to also include `core_apps_pic32cm_le_ls` from under `Harmony 3 - Core`



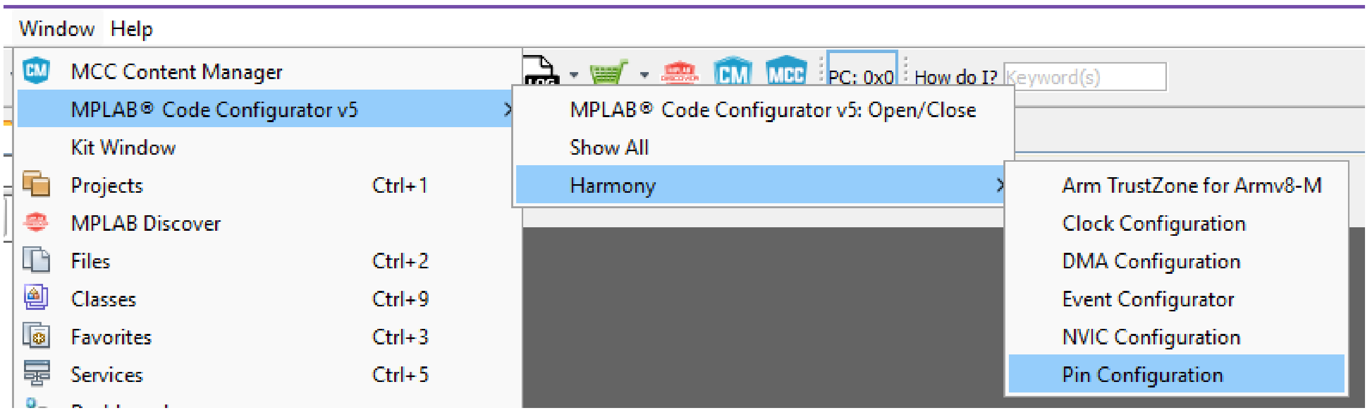
core	v3.13.5	MPLAB Harmony license FatFS license exFAT license
core_apps_cec173x	--	MPLAB Harmony license
core_apps_pic32ck_sg_gc	--	MPLAB Harmony license
core_apps_pic32cm_jh00_jh01	--	MPLAB Harmony license
core_apps_pic32cm_le_ls	v3.4.0	MPLAB Harmony license
core_apps_pic32cm_mc00	--	MPLAB Harmony license
core_apps_pic32cx_mt	--	MPLAB Harmony license
core_apps_pic32cxbz2_wbz45	--	MPLAB Harmony license
core_apps_pic32cxbz3_wbz3	--	MPLAB Harmony license
core_apps_pic32cz_ca	--	MPLAB Harmony license

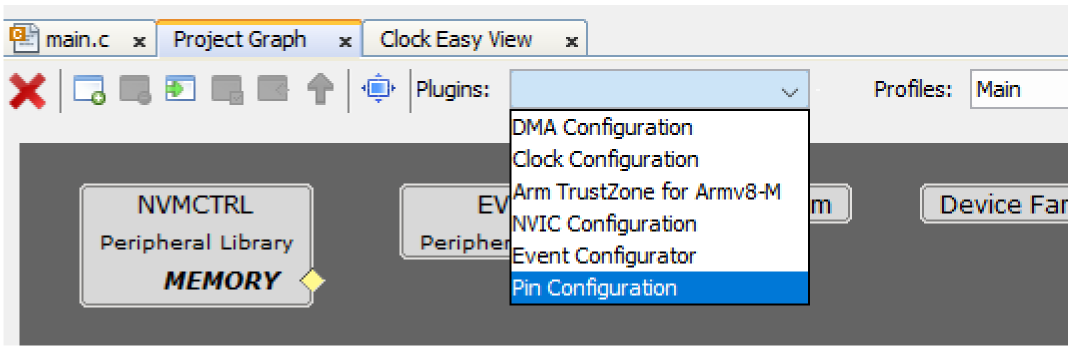
note: you can also add these packages later by opening up the `Content Manager` (CM button beside MCC) if you don't install them now

MCC Harmony Windows

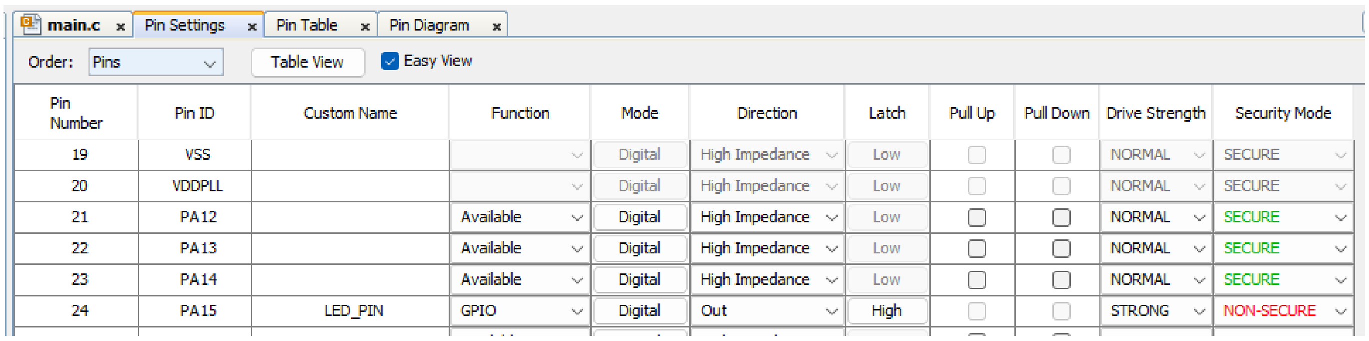
We now proceed with using MCC to easily create some configurations for our Microcontroller

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

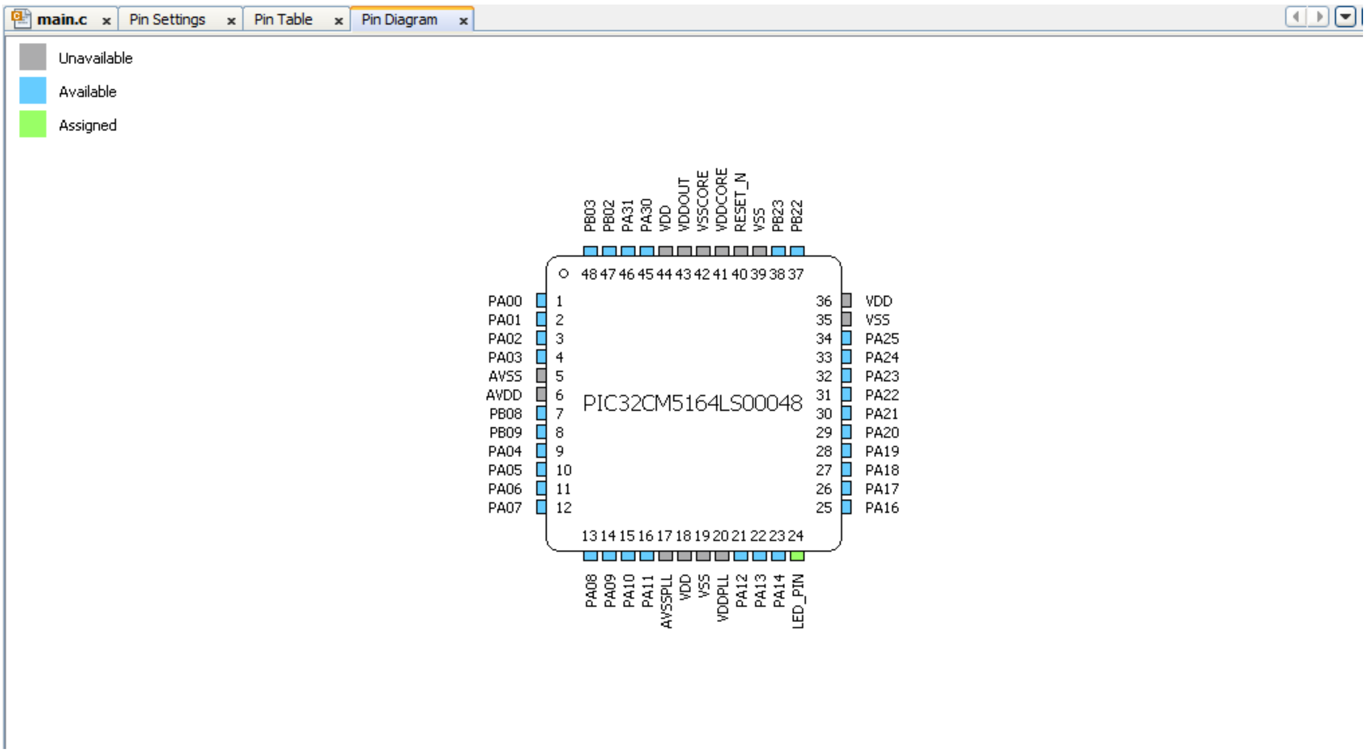




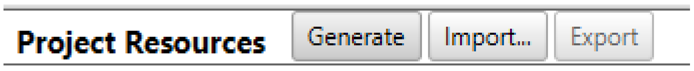
We can use a GUI to set up the pins of our Microcontroller. Proceed to set up PA15 as Output Pin as shown. Take note of the `Security Mode` being set as `NON-SECURE` as well.



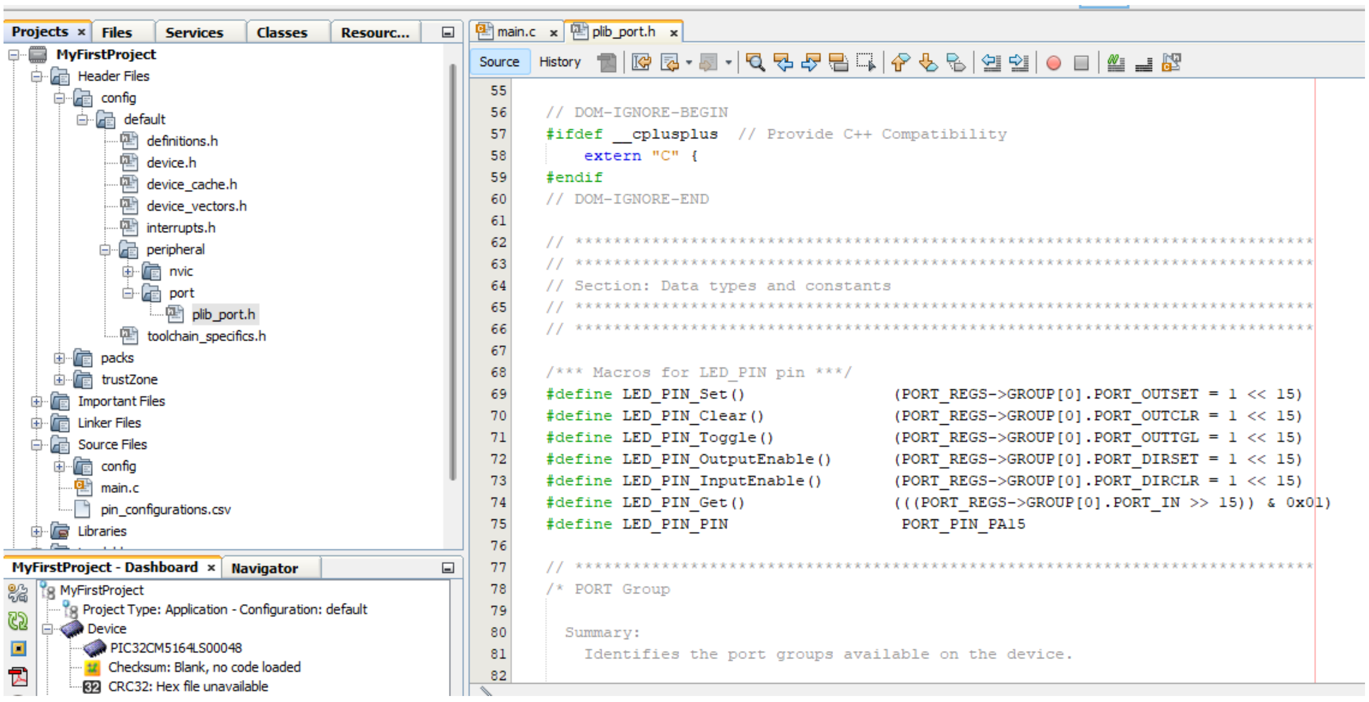
Pin Number	Pin ID	Custom Name	Function	Mode	Direction	Latch	Pull Up	Pull Down	Drive Strength	Security Mode
19	VSS			Digital	High Impedance	Low	☐	☐	NORMAL	SECURE
20	VDDPLL			Digital	High Impedance	Low	☐	☐	NORMAL	SECURE
21	PA12		Available	Digital	High Impedance	Low	☐	☐	NORMAL	SECURE
22	PA13		Available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



- Click on Generate to generate your configuration code



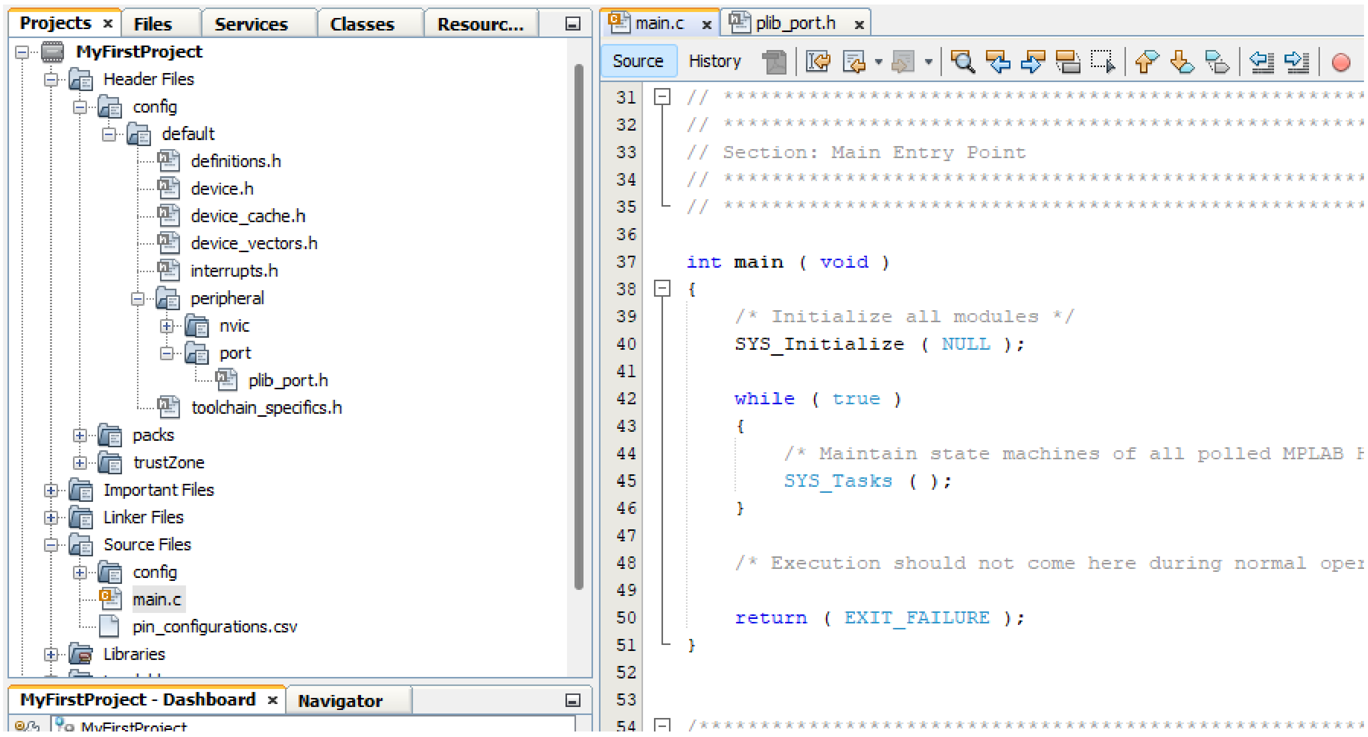
You should now see in your Project folder more files, these were auto generated by MCC. The interesting one here is the header file `plib_port.h`


```
// DOM-IGNORE-BEGIN
#ifdef __cplusplus  // Provide C++ Compatibility
    extern "C" {
#endif
// DOM-IGNORE-END

// *****************************************************************************
// *****************************************************************************
// Section: Data types and constants
// *****************************************************************************
// *****************************************************************************

/*** Macros for LED_PIN pin ***/
#define LED_PIN_Set()               (PORT_REGS->GROUP[0].PORT_OUTSET = 1 << 15)
#define LED_PIN_Clear()             (PORT_REGS->GROUP[0].PORT_OUTCLR = 1 << 15)
#define LED_PIN_Toggle()            (PORT_REGS->GROUP[0].PORT_OUTTGL = 1 << 15)
#define LED_PIN_OutputEnable()      (PORT_REGS->GROUP[0].PORT_DIRSET = 1 << 15)
#define LED_PIN_InputEnable()       (PORT_REGS->GROUP[0].PORT_DIRCLR = 1 << 15)
#define LED_PIN_Get()               (((PORT_REGS->GROUP[0].PORT_IN >> 15)) & 0x01)
#define LED_PIN_PIN                  PORT_PIN_PA15

// *****************************************************************************
/* PORT Group

  Summary:
    Identifies the port groups available on the device.
```

MyFirstProject
Project Type: Application - Configuration: default
Device
PIC32CM5164LS00048
Checksum: Blank, no code loaded
CRC32: Hex file unavailable

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

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


```
// *****************************************************************
// *****************************************************************
// Section: Main Entry Point
// *****************************************************************
// *****************************************************************

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

    while ( true )
    {
        /* Maintain state machines of all polled MPLAB H
        SYS_Tasks ( );
    }

    /* Execution should not come here during normal oper

    return ( EXIT_FAILURE );
}

/*****************************************************************
```

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.

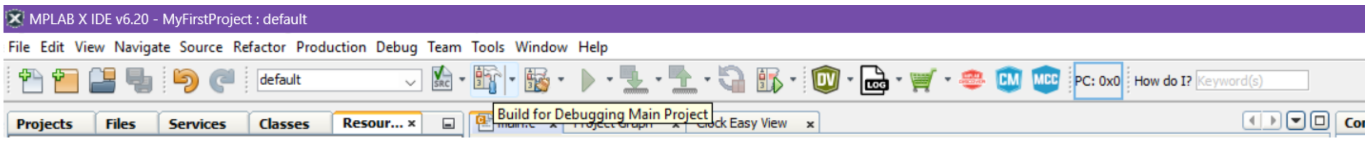
Guided Exercise: "MCC-Blinky!"

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


MPLAB X IDE v6.20 - MyFirstProject : default
File Edit View Navigate Source Refactor Production Debug Team Tools Window Help
default
PC: 0x0
How do I?
Keyword(s)
Build for Debugging Main Project
Projects
Files
Services
Classes
Resour...
Clock Easy View

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