MPLAB First Steps

Installation

Install MPLAB X IDE

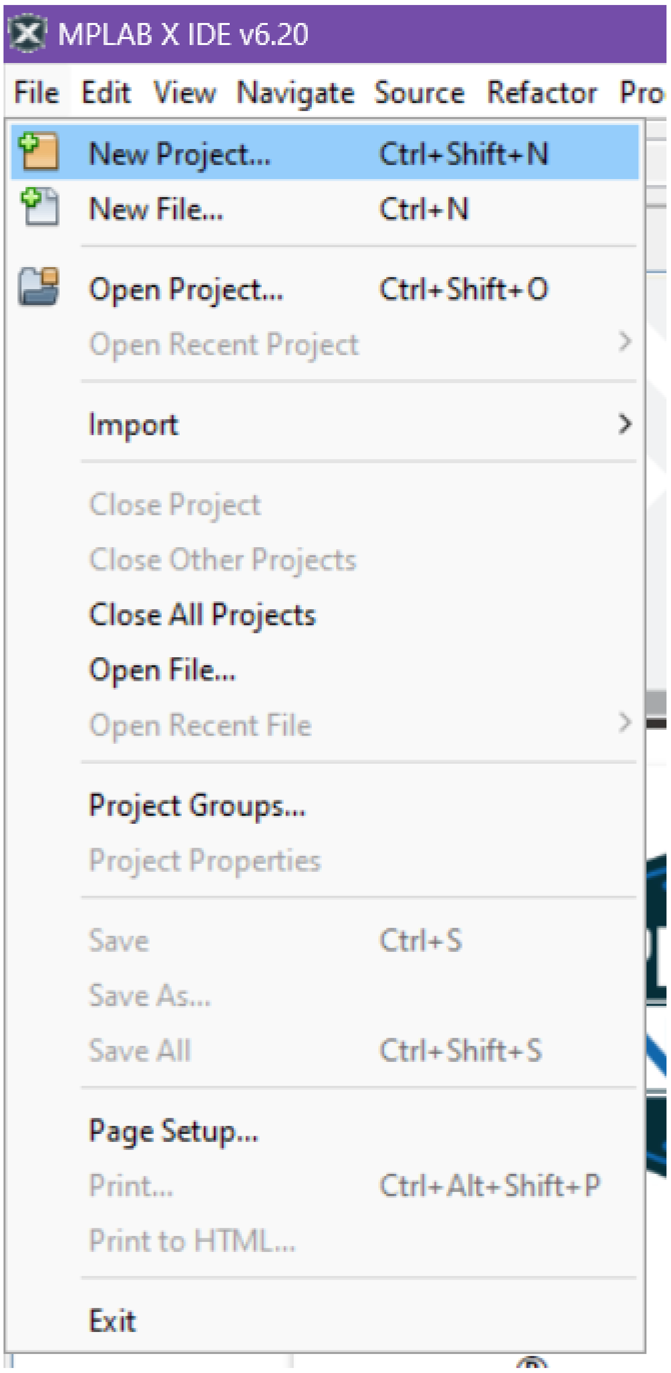
Install MPLAB X IDE through the official Microchip site: MPLAB X IDE

Install XC32 Compiler

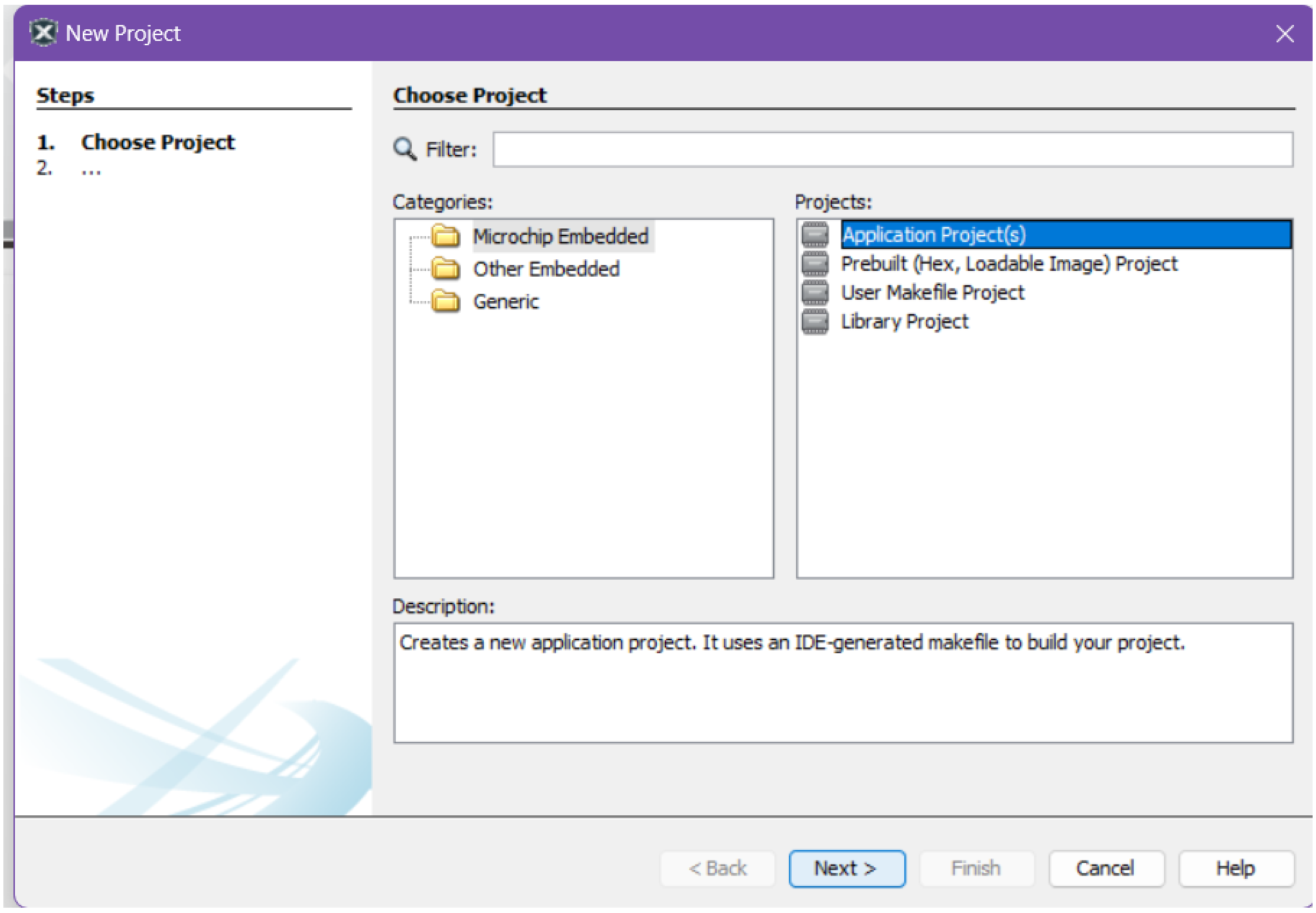
Install the XC32 COmpilers from Microchip: MPLAB xC32 Compilers

Your First MPLAB Project

- Create a New Project



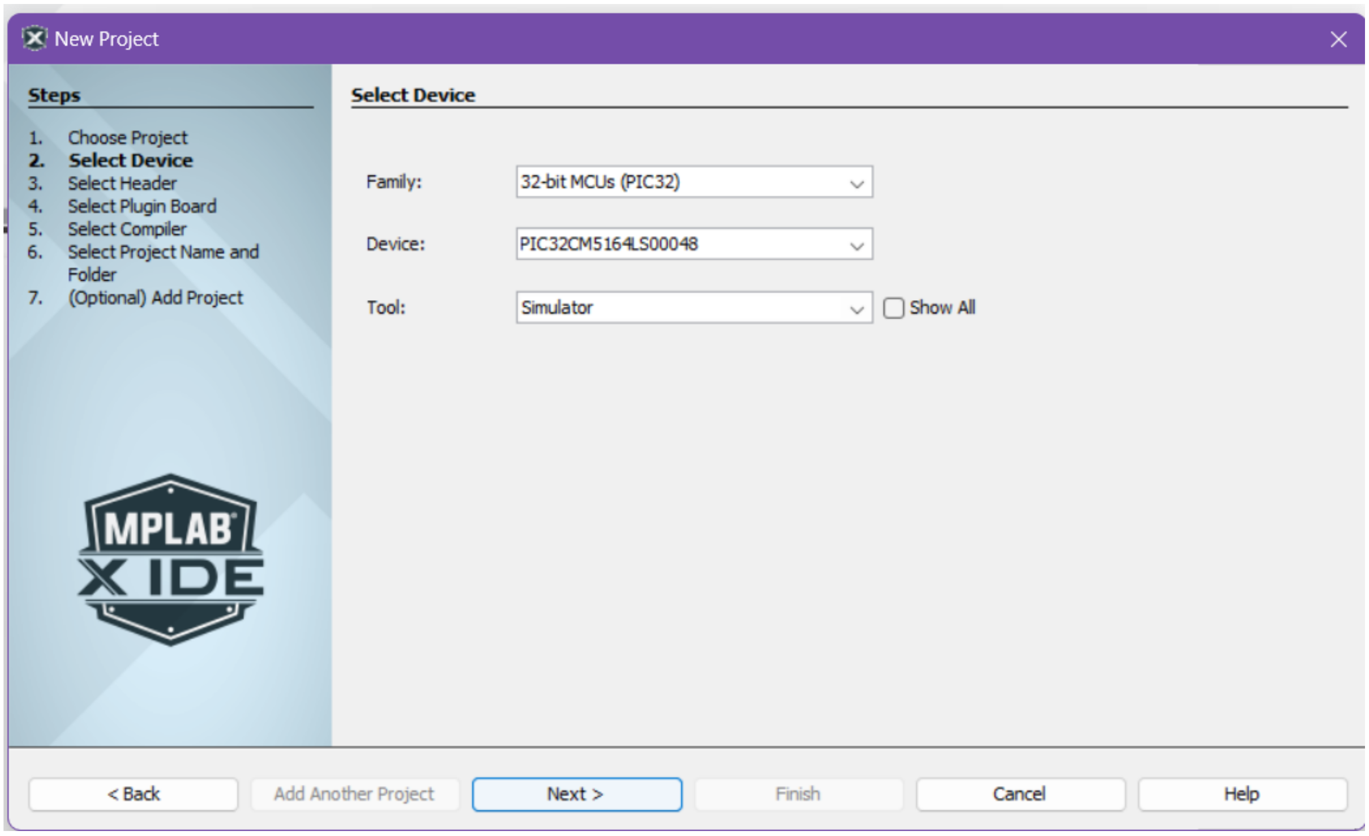
- Choose `Application Project(s)` then click `Next >`



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



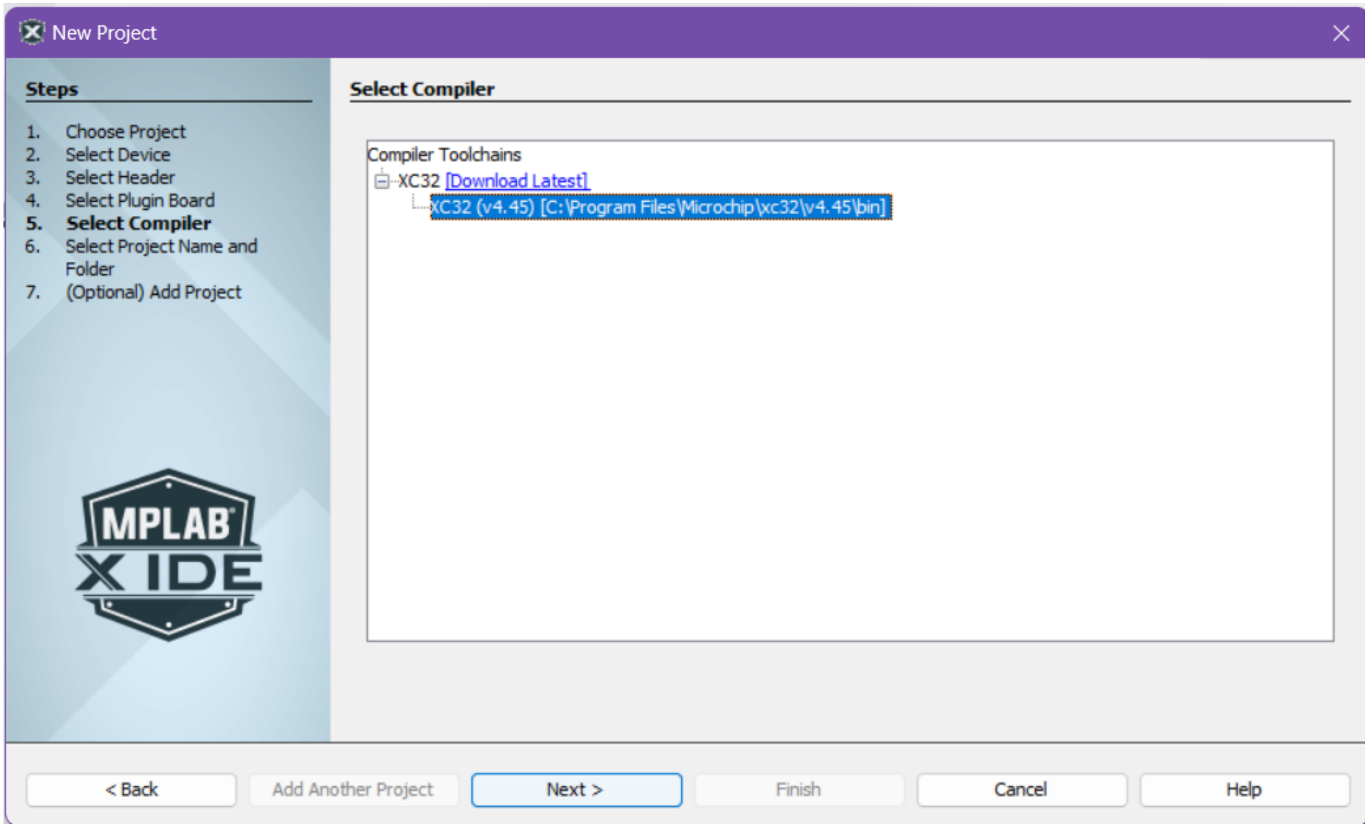
Select Device

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*

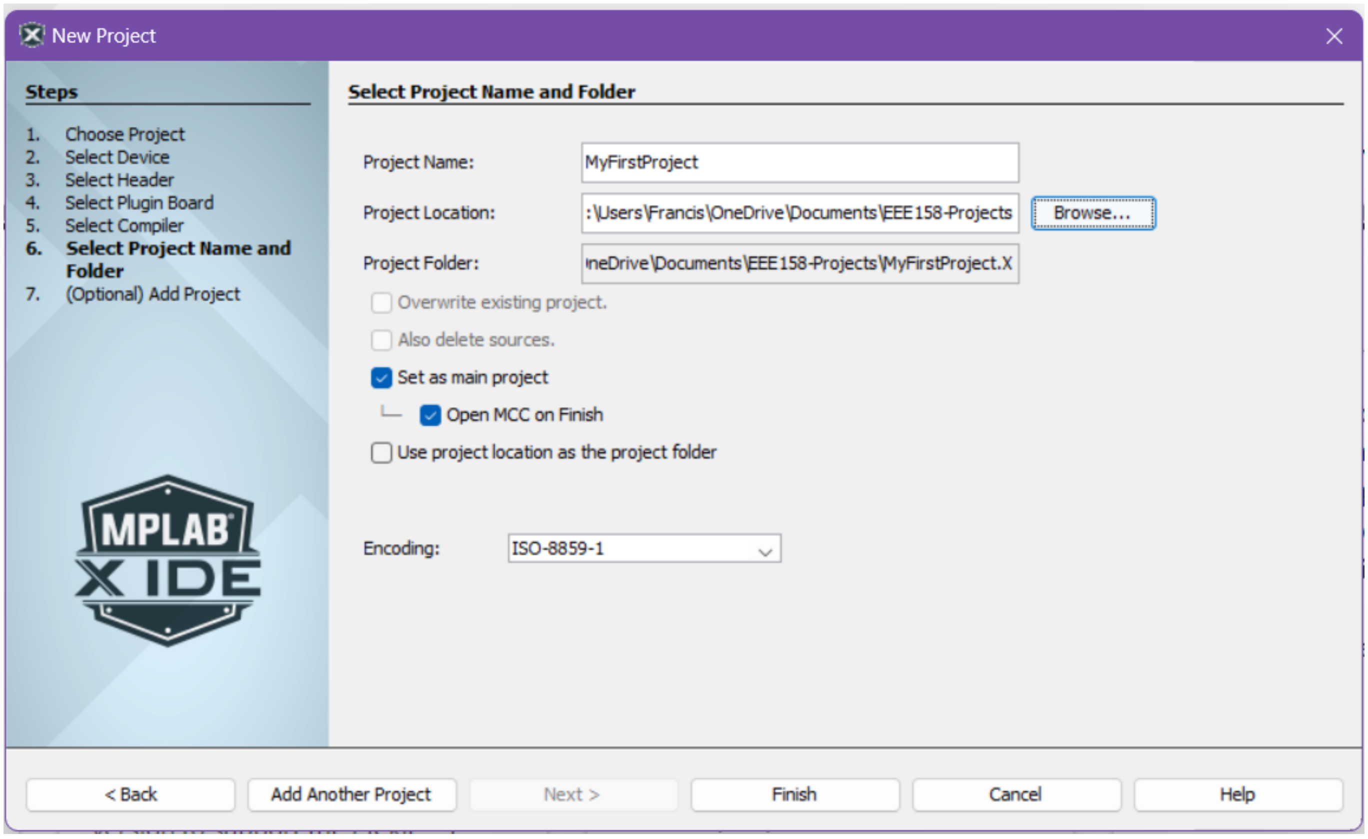


Select Compiler

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

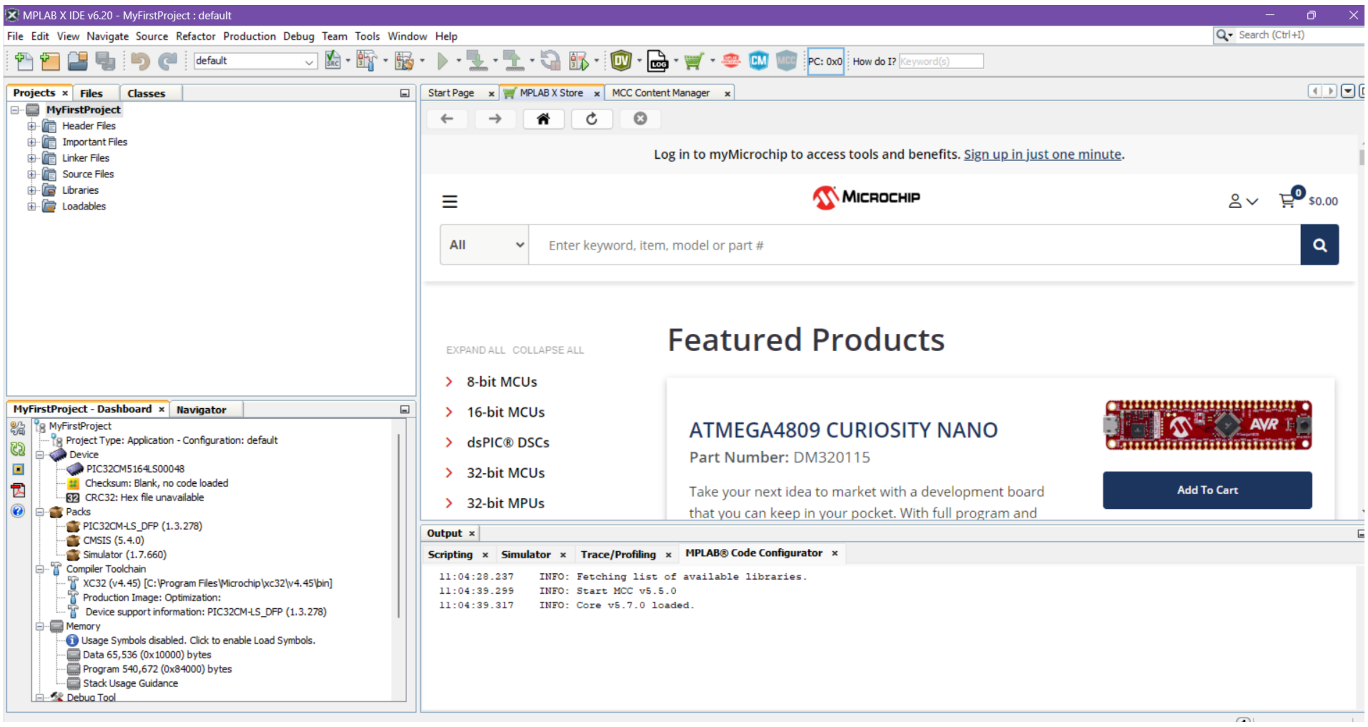
Steps

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: neDrive\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

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
```

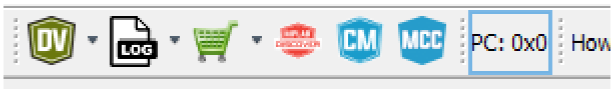
Log in to myMicrochip to access tools and benefits. Sign up in just one minute.

Featured Products

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

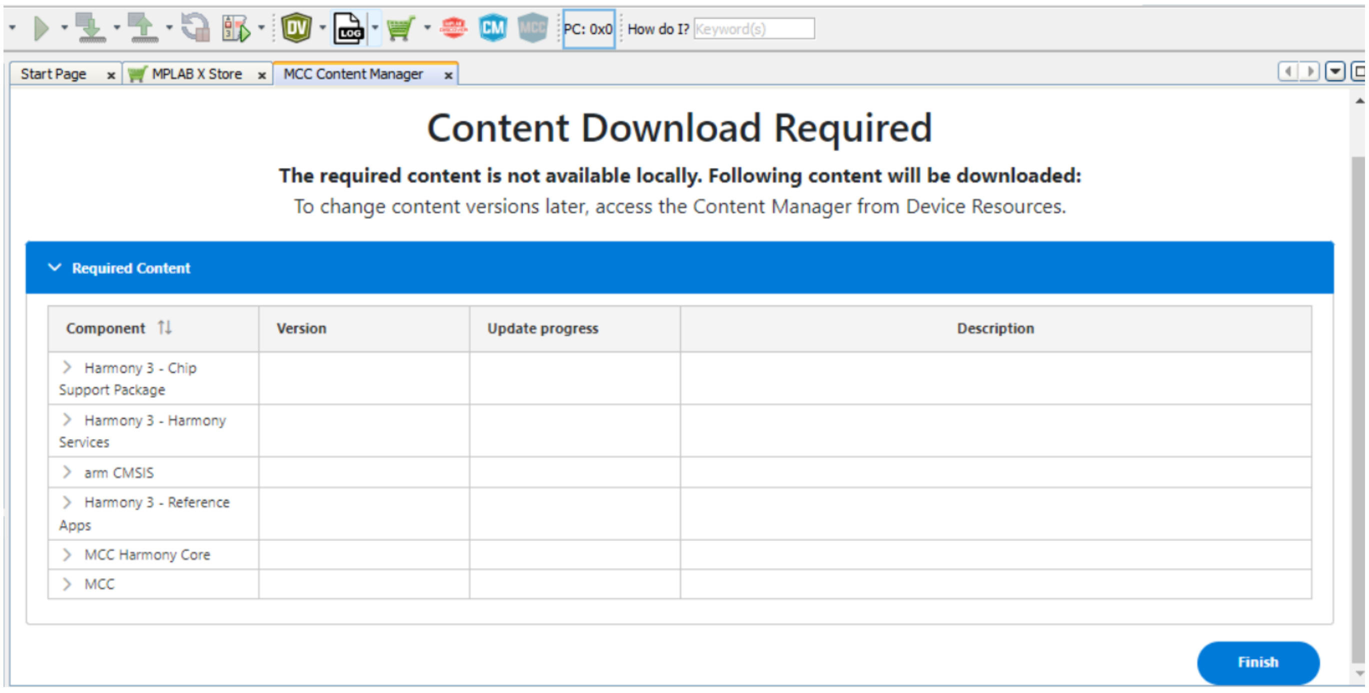
```
11:04:28.237    INFO: Fetching list of available libraries.
11:04:39.299    INFO: Start MCC v5.5.0
11:04:39.317    INFO: Core v5.7.0 loaded.
```

- 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.

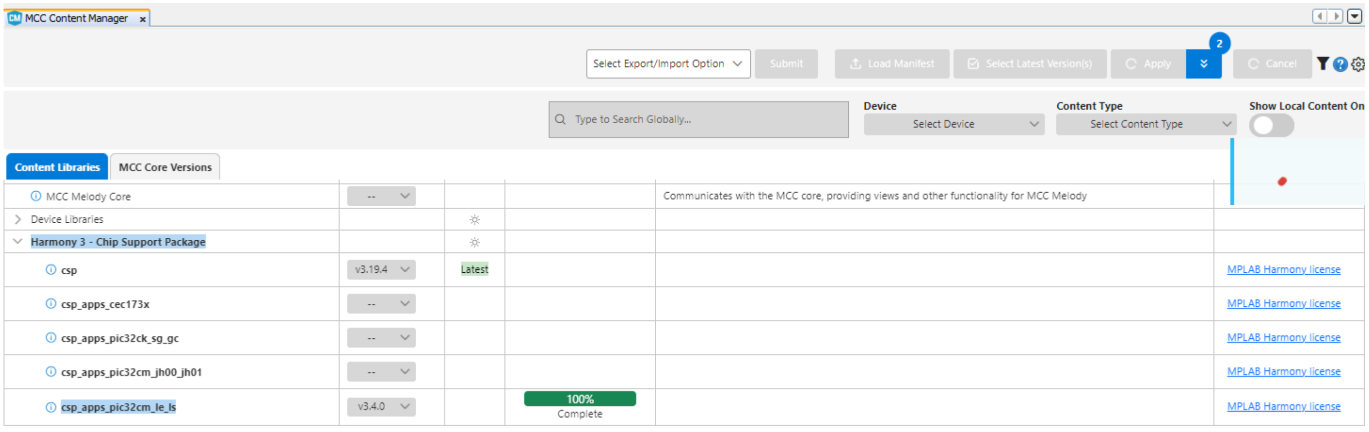
Required Content

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.

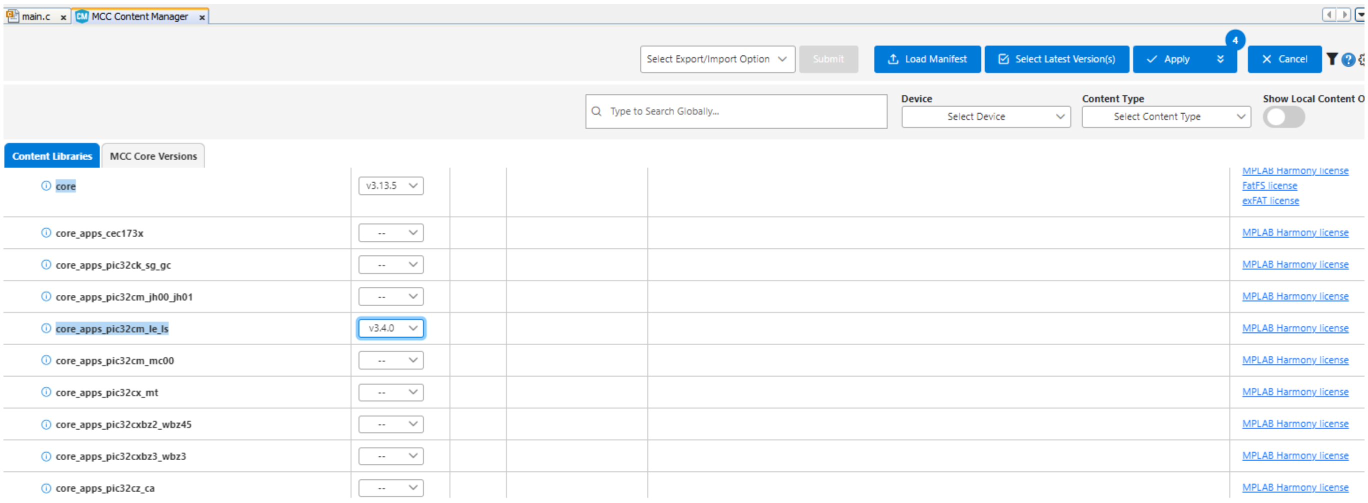
Board Packages

- 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`



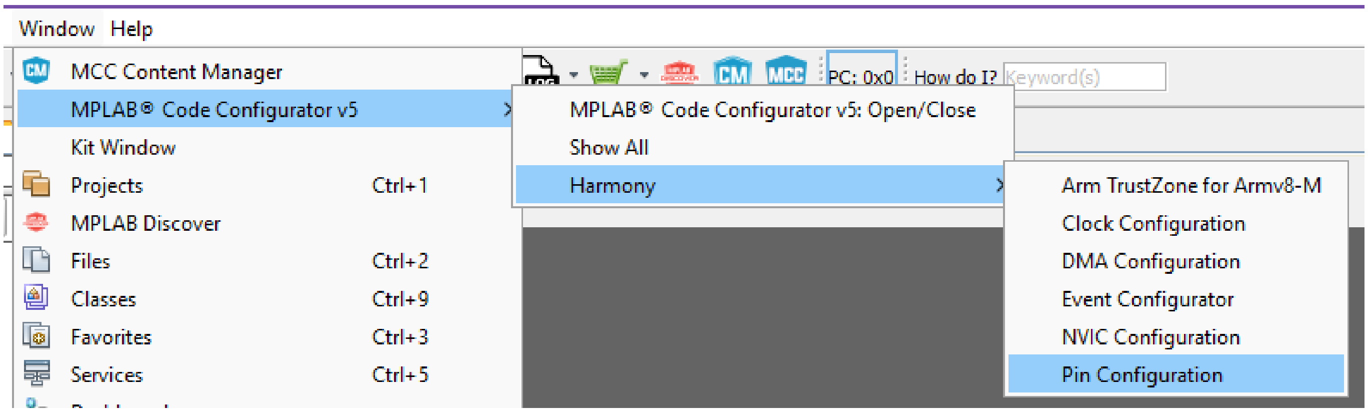
core	v3.13.5	MPLAB Harmony license FatFS license exFAT license
core_apps_cec173x	--	MPLAB Harmony license
core_apps_pic32ck_sg_gc	--	MPLAB Harmony license
core_apps_pic32cm_jh00_jh01	--	MPLAB Harmony license
core_apps_pic32cm_le_ls	v3.4.0	MPLAB Harmony license
core_apps_pic32cm_mc00	--	MPLAB Harmony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

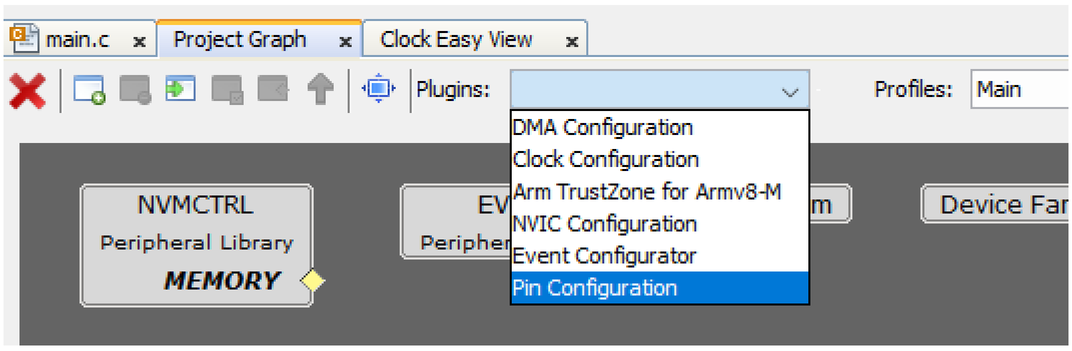
note: you can also add these packages later by opening up the `Content Manager` (CM button beside MCC) if you don't install them now

MCC Harmony Windows

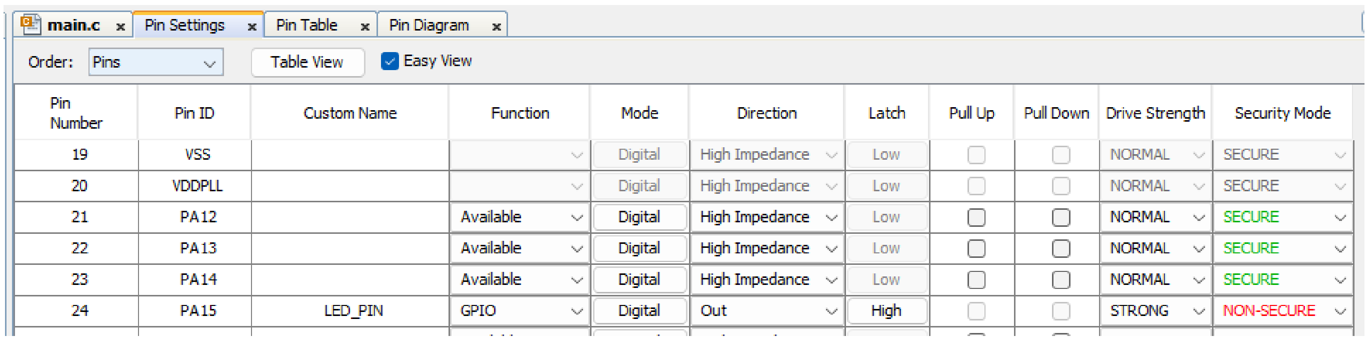
We now proceed with using MCC to easily create some configurations for our Microcontroller

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



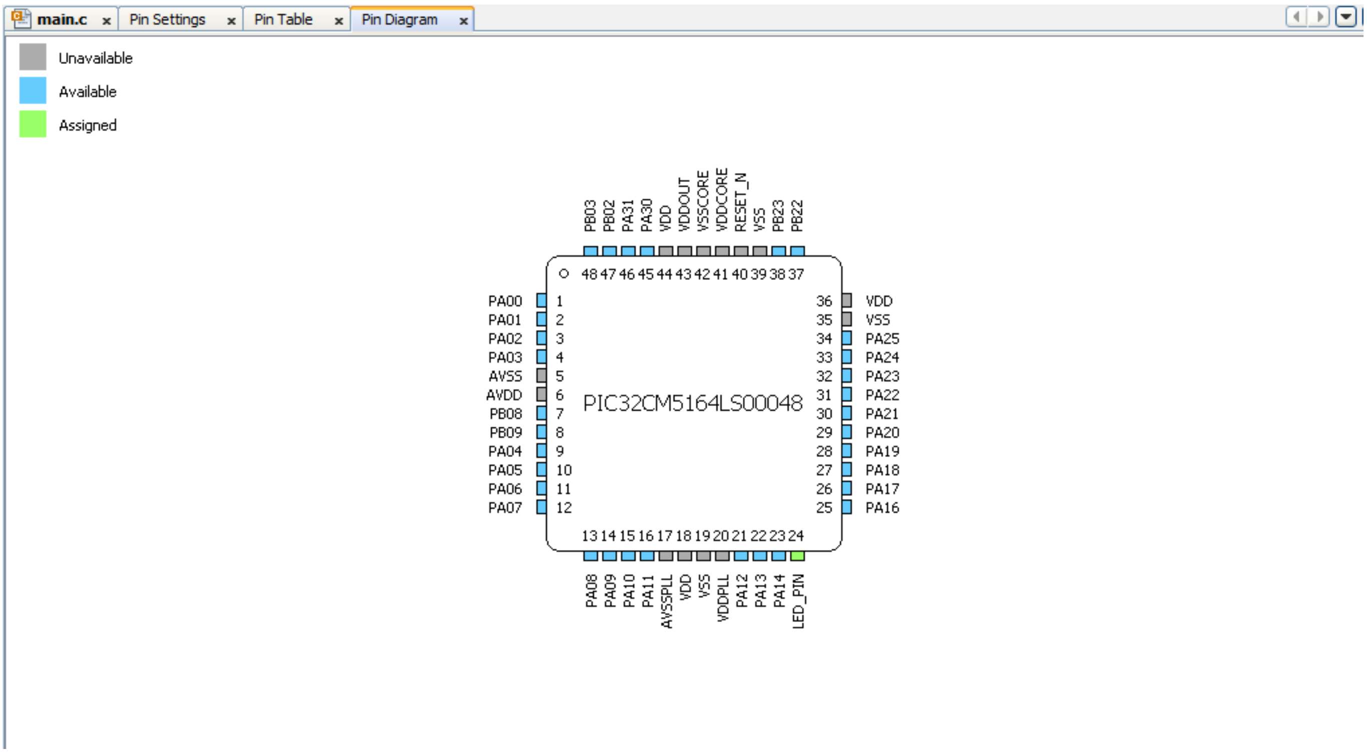


We can use a GUI to set up the pins of our Microcontroller. Proceed to set up PA15 as Output Pin as shown. Take note of the `Security Mode` being set as `NON-SECURE` as well.



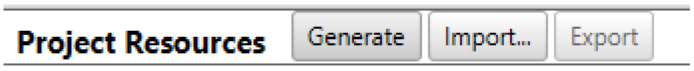
Order: Pins Table View ✓ Easy View

Pin Number	Pin ID	Custom Name	Function	Mode	Direction	Latch	Pull Up	Pull Down	Drive Strength	Security Mode
19	VSS			Digital	High Impedance	Low	☐	☐	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

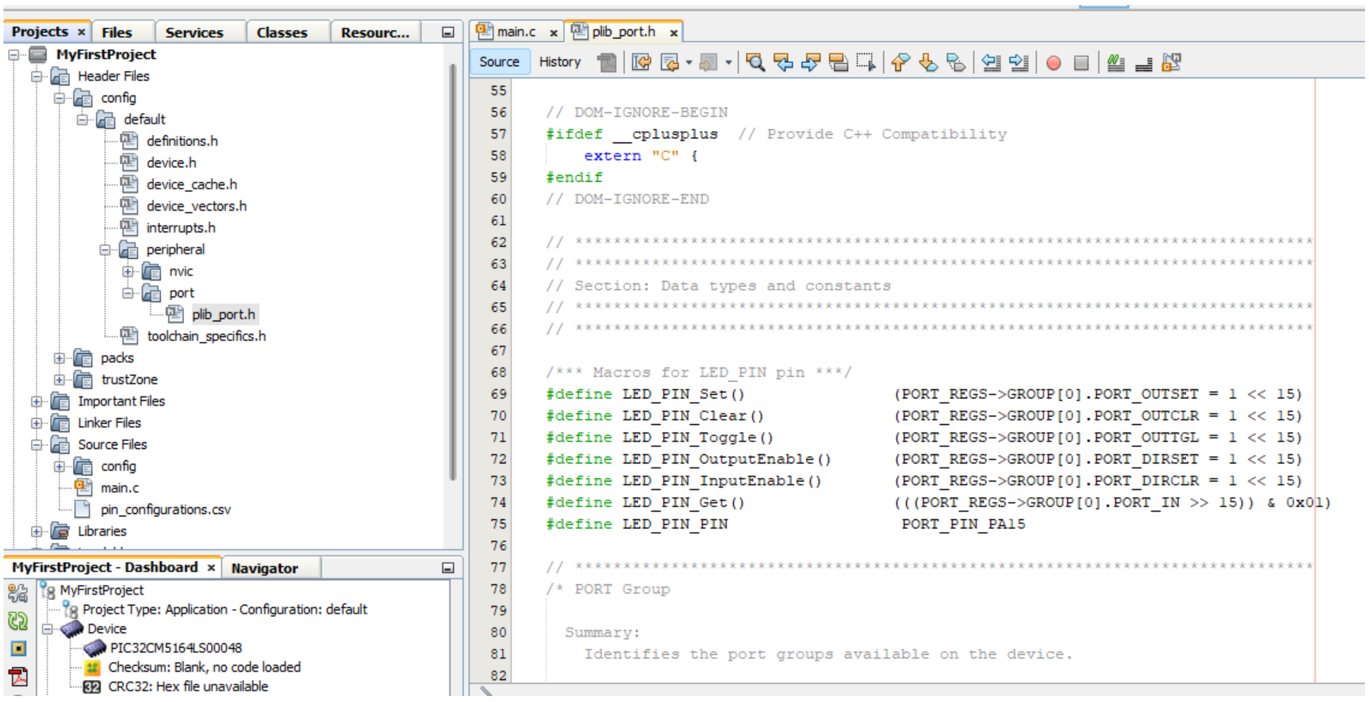

main.c
Pin Settings
Pin Table
Pin Diagram
PIC32CM5164LS00048
PB03 PB02 PA31 PA30 VDD VDDOUT VSSCORE VDDCORE RESET_N VSS PB23 PB22
PA00 PA01 PA02 PA03 AVSS AVDD PB08 PB09 PA04 PA05 PA06 PA07
VDD VSS PA25 PA24 PA23 PA22 PA21 PA20 PA19 PA18 PA17 PA16
PA08 PA09 PA10 PA11 AVSSPLL VDD VSS VDDPLL PA12 PA13 PA14 LED_PIN

⬜	Unavailable
🟦	Available
🟩	Assigned

- Click on Generate to generate your configuration code



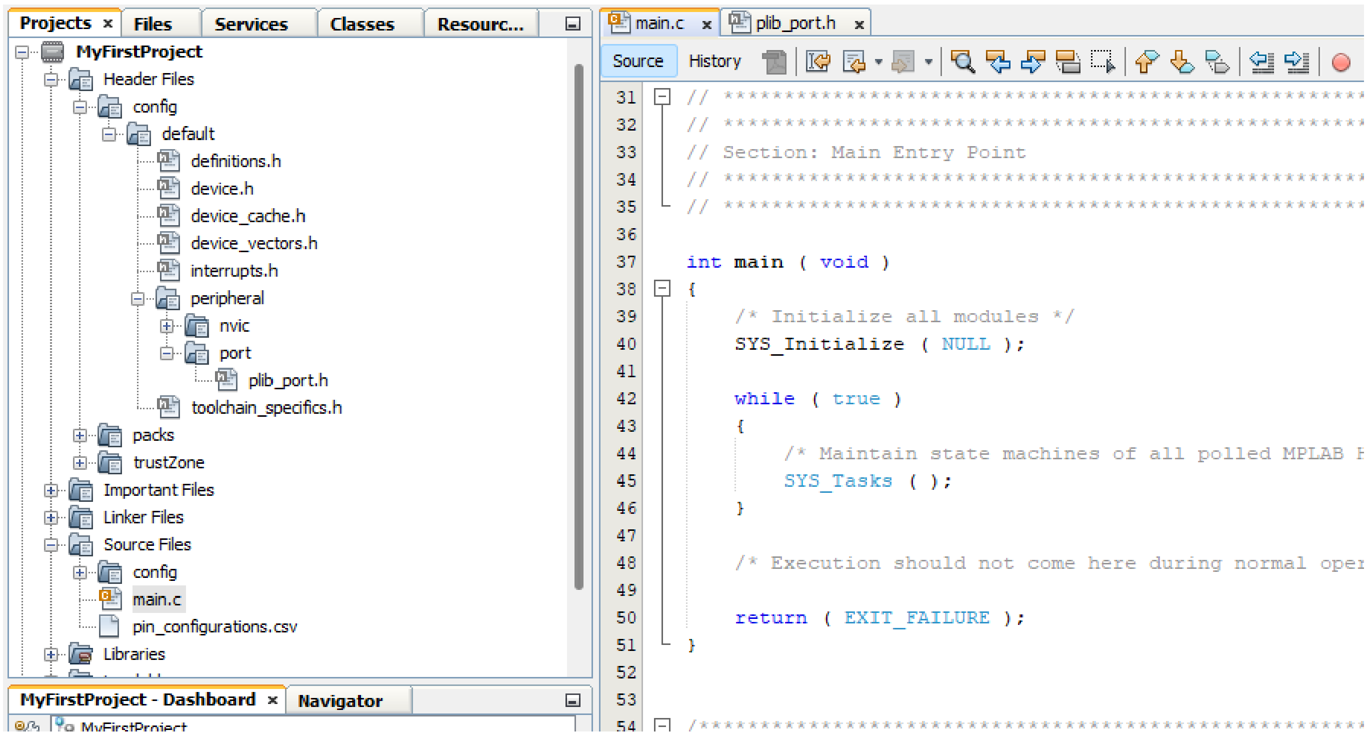
You should now see in your Project folder more files, these were auto generated by MCC. The interesting one here is the header file `plib_port.h`



```c

// 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.

```

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

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



```c
// ************************************************************************
// ************************************************************************
// 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.

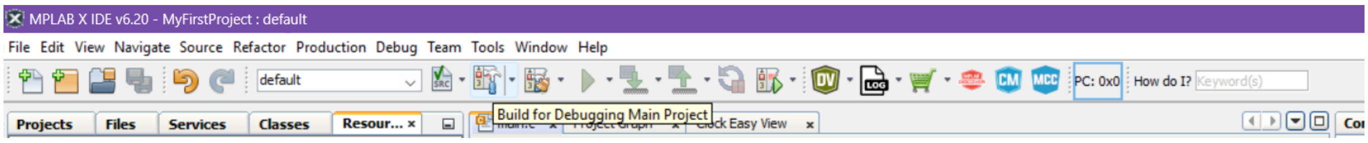
Guided Exercise: "Blinky!"

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)
Projects
Files
Services
Classes
Resour...
Build for Debugging Main Project
Clock Easy View

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

Simulator Debugging

Guided Exercise: "Hello World"

"Hello World"

Logic Analyzer