

**An introduction to**

**PIC32** MX110F016B

**Microcontroller (1)**

---

Sina Akbari

# Device Overview

- 28-PIN SPDIP
- 4GB Flexible Virtual Memory Space
- MIPS32® M4K® Processor Core
- Complies with MIPS32® Instruction Set



# Device Pinout

Pin #	Full Pin Name
1	MCLR
2	VREF+/CVREF+/AN0/C3INC/RPA0/CTED1/RA0
3	VREF-/CVREF-/AN1/RPA1/CTED2/RA1
4	PGED1/AN2/C1IND/C2INB/C3IND/RPB0/RB0
5	PGEC1/AN3/C1INC/C2INA/RPB1/CTED12/RB1
6	AN4/C1INB/C2IND/RPB2/SDA2/CTED13/RB2
7	AN5/C1INA/C2INC/RTCC/RPB3/SCL2/RB3
8	Vss
9	OSC1/CLKI/RPA2/RA2
10	OSC2/CLKO/RPA3/PMA0/RA3
11	SOSCI/RPB4/RB4
12	SOSCO/RPA4/T1CK/CTED9/PMA1/RA4
13	VDD
14	PGED3/RPB5/PMD7/RB5

Pin #	Full Pin Name
15	PGEC3/RPB6/PMD6/RB6
16	TDI/RPB7/CTED3/PMD5/INT0/RB7
17	TCK/RPB8/SCL1/CTED10/PMD4/RB8
18	TDO/RPB9/SDA1/CTED4/PMD3/RB9
19	Vss
20	VCAP
21	PGED2/RPB10/CTED11/PMD2/RB10
22	PGEC2/TMS/RPB11/PMD1/RB11
23	AN12/PMD0/RB12
24	AN11/RPB13/CTPLS/PMRD/RB13
25	CVREFOUT/AN10/C3INB/RPB14/SCK1/CTED5/PMWR/RB14
26	AN9/C3INA/RPB15/SCK2/CTED6/PMCS1/RB15
27	AVss
28	AVDD



# Guidelines For Getting Started!

- 1) Operating Conditions of the IC
- 2) MCLR pin
- 3) Decoupling Capacitors
- 4) Capacitor on Internal Voltage Regulator (**Vcap** pin)

# Operating Conditions

Typical Operating Supply Voltage: 3.25v to 3.5v.

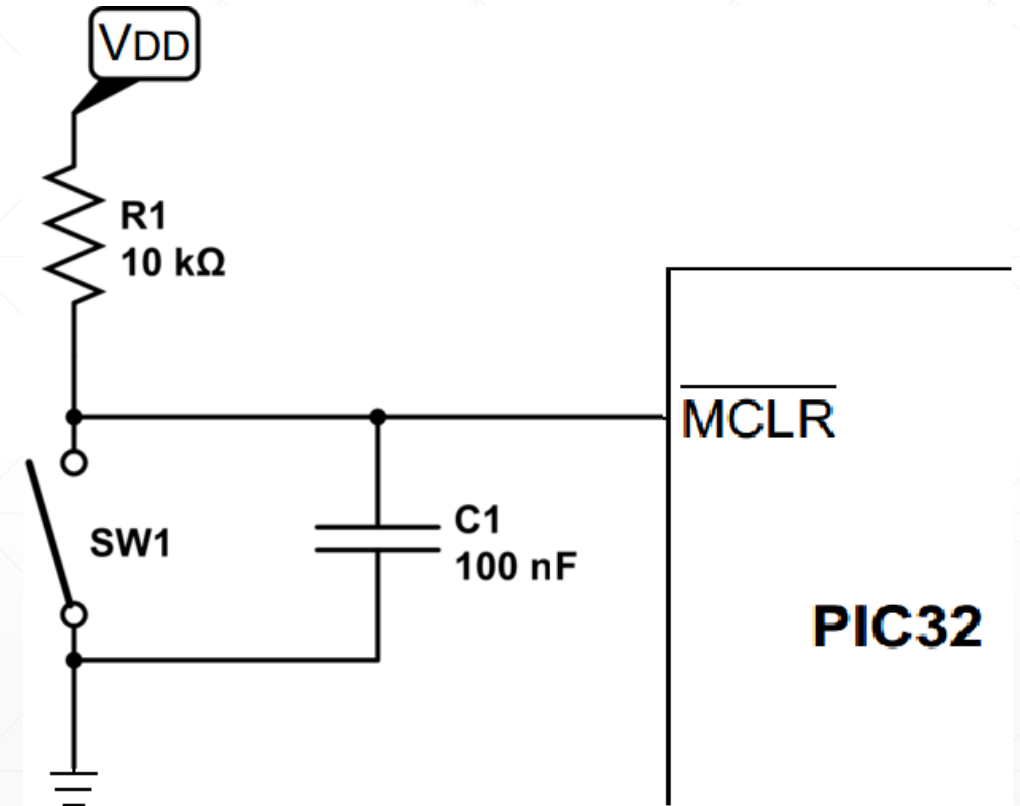
Max. current into Vdd pin: 300mA

**!! Important Note: Applying a voltage higher than 4v to AVdd with respect to AVss, will HARM the IC !!**

Unlike other common digital ICs which work at 5v, the supply voltage range of this IC is 2.3v to 3.6v.

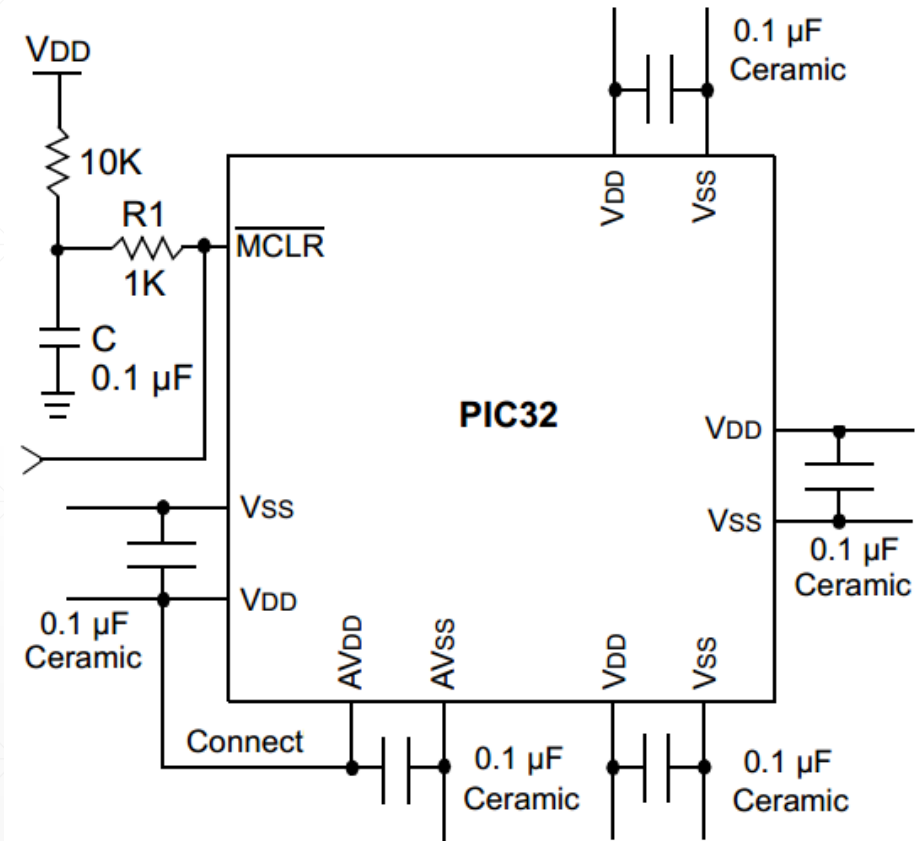
## Master Clear – $\overline{\text{MCLR}}$ (PIN #1)

- This is the typical circuitry you would use for master clear pin.
- This pin is normally pulled up via R1 resistor. (Typically 10k)
- The IC is reset when it gets low (Switch1 is closed)
- C1 is typically 100nF Ceramic Capacitor.



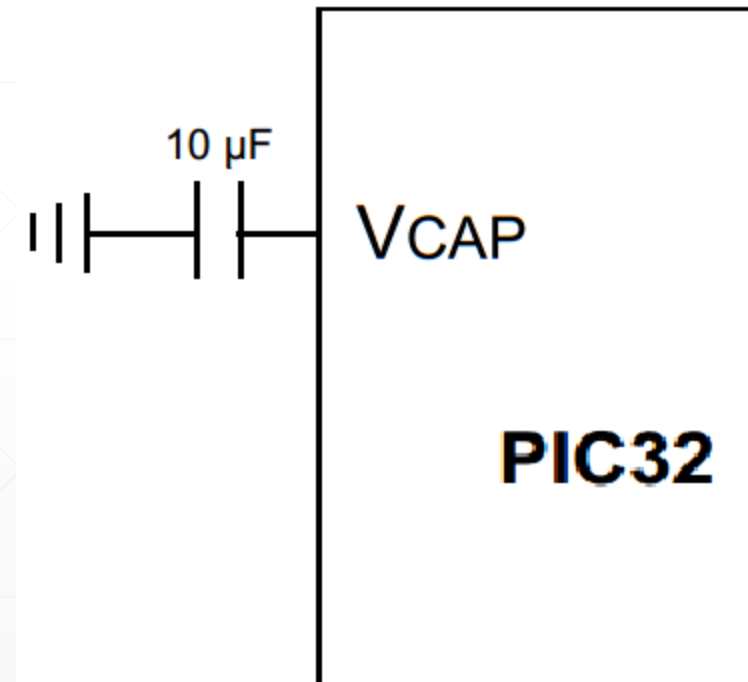
# Decoupling Capacitors

- Always **hardwire** AVdd and Vdd pins!
- Using bulk capacitors with typical values ranging from  $4.7\mu$  to  $47\mu$ , will improve power supply stability and ADC noise rejection. Although, they are **not crucial** for most applications – including your experiments during this course.



# Capacitor on Internal Voltage Regulator – Vcap (PIN #20)

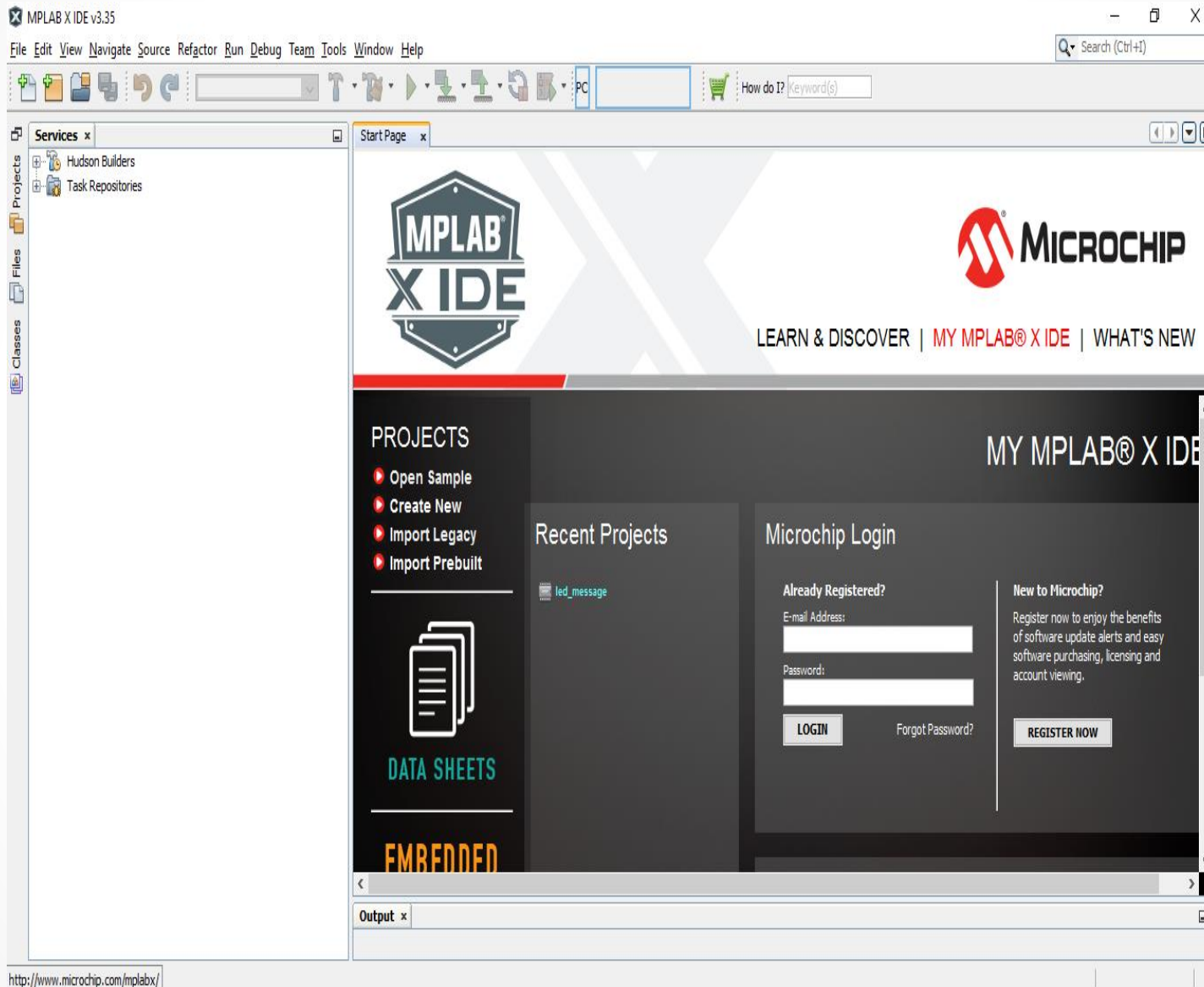
- A capacitor is required on the Vcap pin, which is used to stabilize the internal voltage regulator output.
- Typical value =  $10\mu\text{F}$
- Never connect Vcap to Avdd.





# **Getting Started with MPLAB X IDE**

---



<http://www.microchip.com/mplabx/>

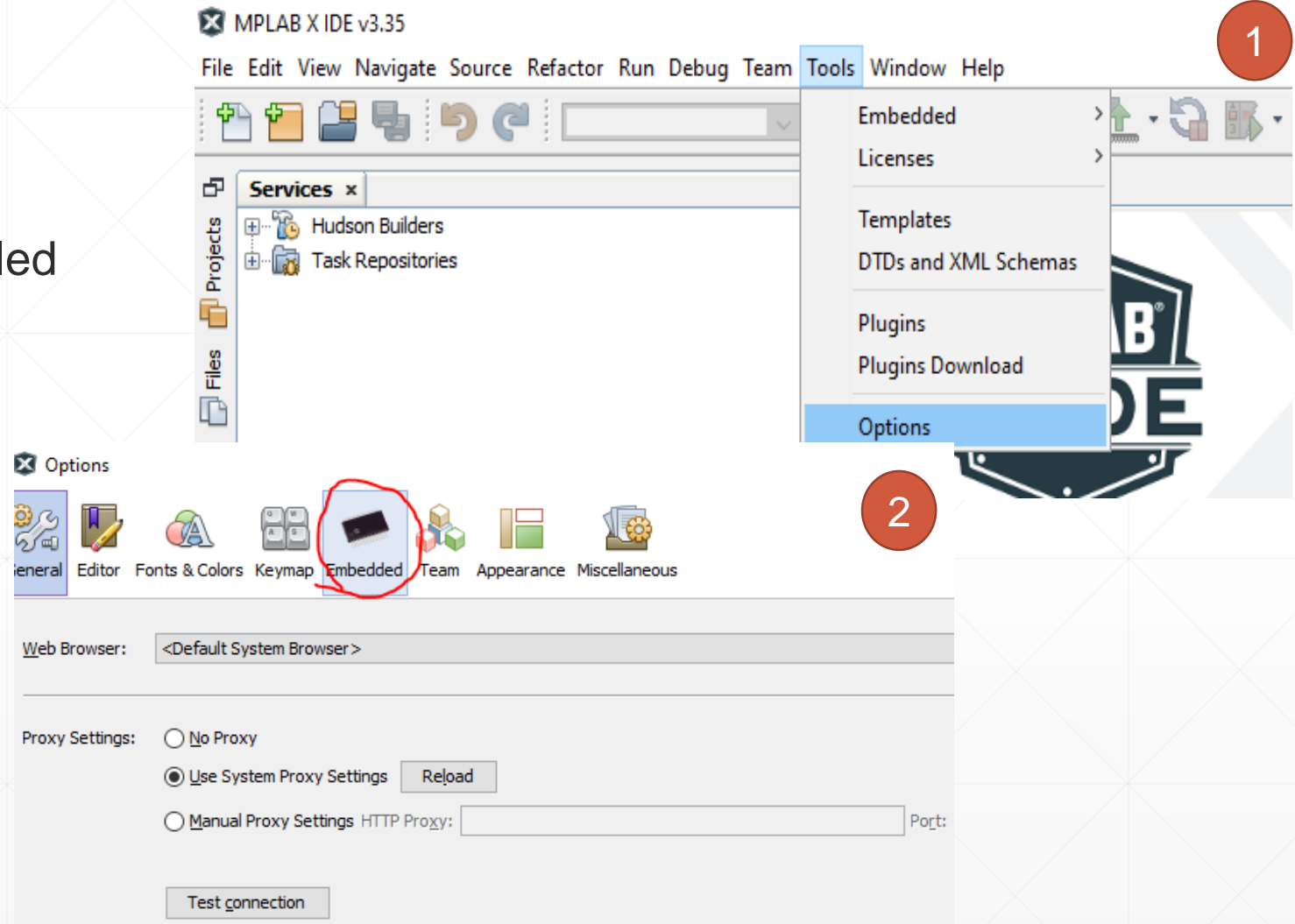
# MPLAB X IDE

Microchip's IDE for PIC microcontrollers family.

**\*NOTE:** Make sure you've installed MPLAB X IDE and XC32 compiler before the next lab session.

# Getting Started with MPLAB X IDE – 1

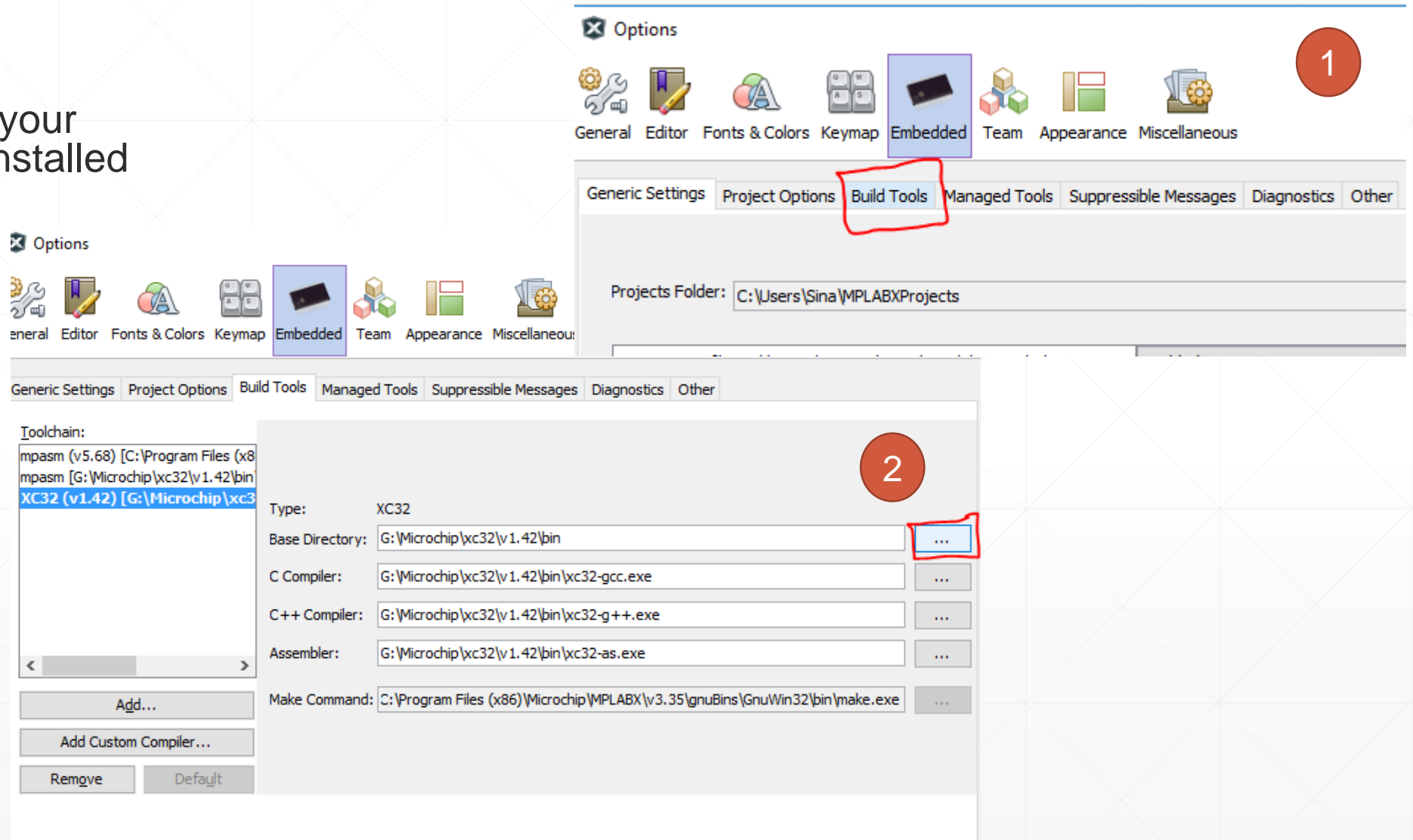
- Checking whether your XC32 compiler is installed correctly
- Tools > Options > Embedded tab



# Getting Started with MPLAB X IDE – 2

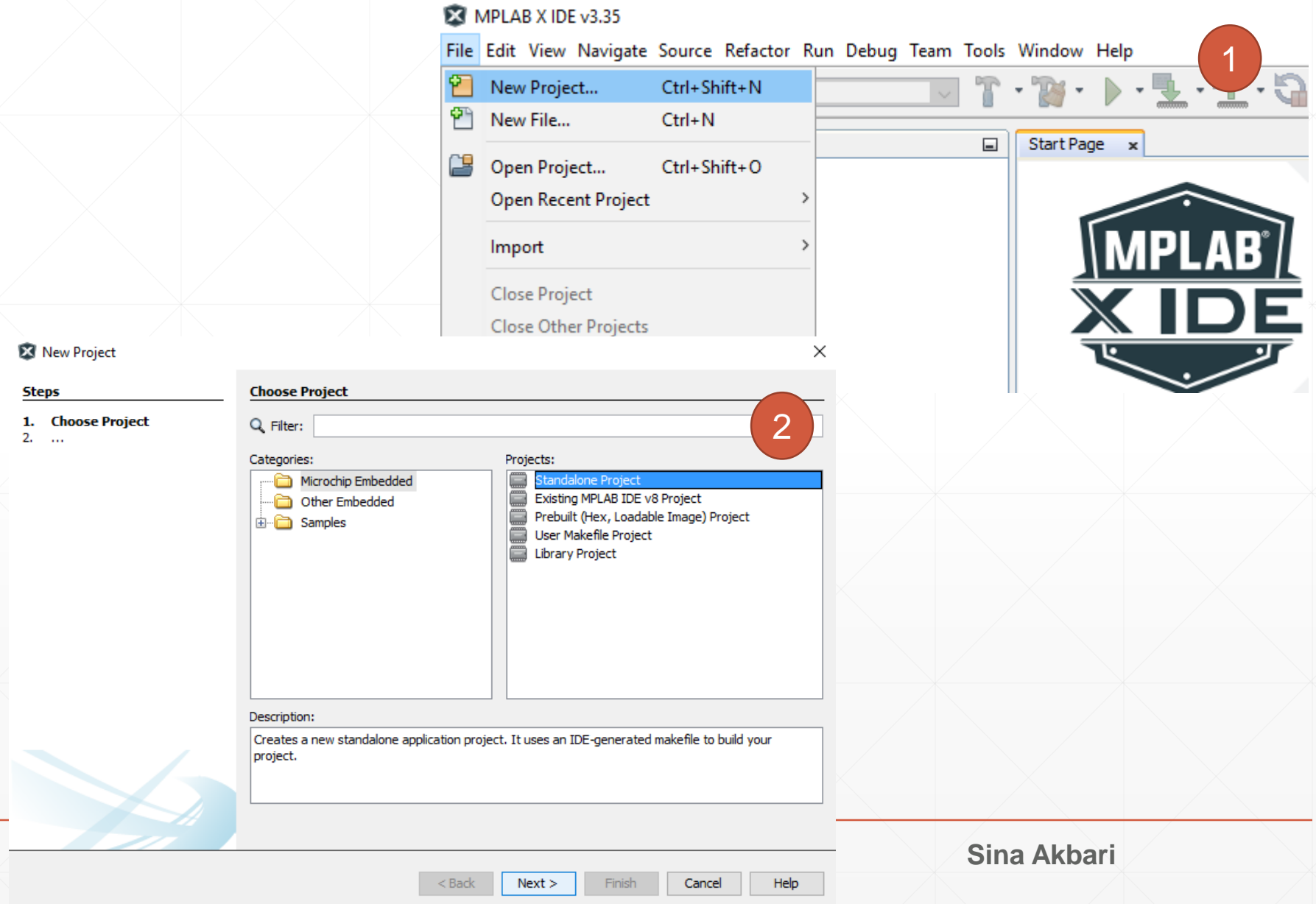
- Checking whether your XC32 compiler is installed correctly

- Build Tools tab > select the base directory where your compiler has been installed



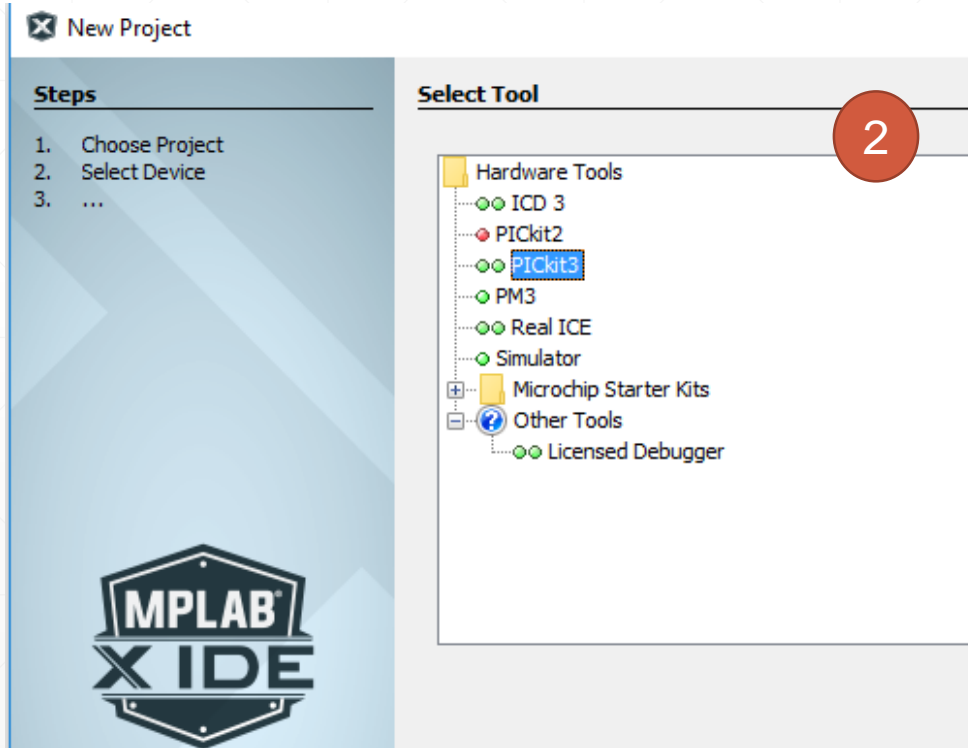
# Getting Started with MPLAB X IDE – 3

- It's time to open a new project!
- Select Standalone Project and click on Next>



# Getting Started with MPLAB X IDE – 4

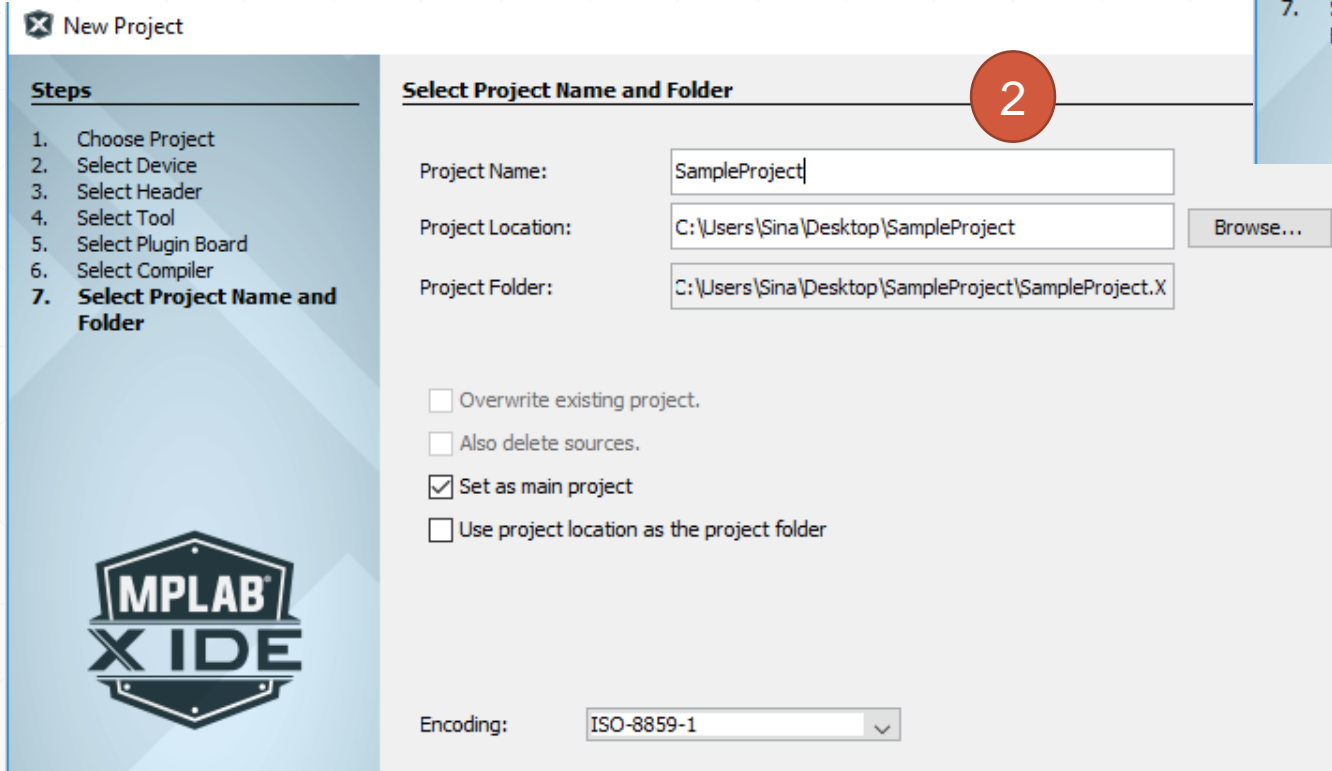
- Under 32-bit MCUs Family, select PIC32MX110F016B and click Next>



- Select PICkit3 as your programming tool.

# Getting Started with MPLAB X IDE – 5

- Select XC32 as your compiler toolchain.



The image shows the 'New Project' dialog box in MPLAB X IDE. It is divided into two main sections. The top section, labeled 'Steps', lists seven steps: 1. Choose Project, 2. Select Device, 3. Select Header, 4. Select Tool, 5. Select Plugin Board, 6. **Select Compiler**, and 7. **Select Project Name and Folder**. The bottom section, labeled 'Select Project Name and Folder', contains fields for 'Project Name' (SampleProject), 'Project Location' (C:\Users\Sina\Desktop\SampleProject), and 'Project Folder' (C:\Users\Sina\Desktop\SampleProject\SampleProject.X). There are also checkboxes for 'Overwrite existing project.', 'Also delete sources.', 'Set as main project' (checked), and 'Use project location as the project folder'. An 'Encoding' dropdown is set to 'ISO-8859-1'. A 'Browse...' button is next to the 'Project Location' field. The MPLAB X IDE logo is in the bottom left corner.

**New Project**

**Steps**

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

**Select Project Name and Folder**

Project Name: SampleProject

Project Location: C:\Users\Sina\Desktop\SampleProject Browse...

Project Folder: C:\Users\Sina\Desktop\SampleProject\SampleProject.X

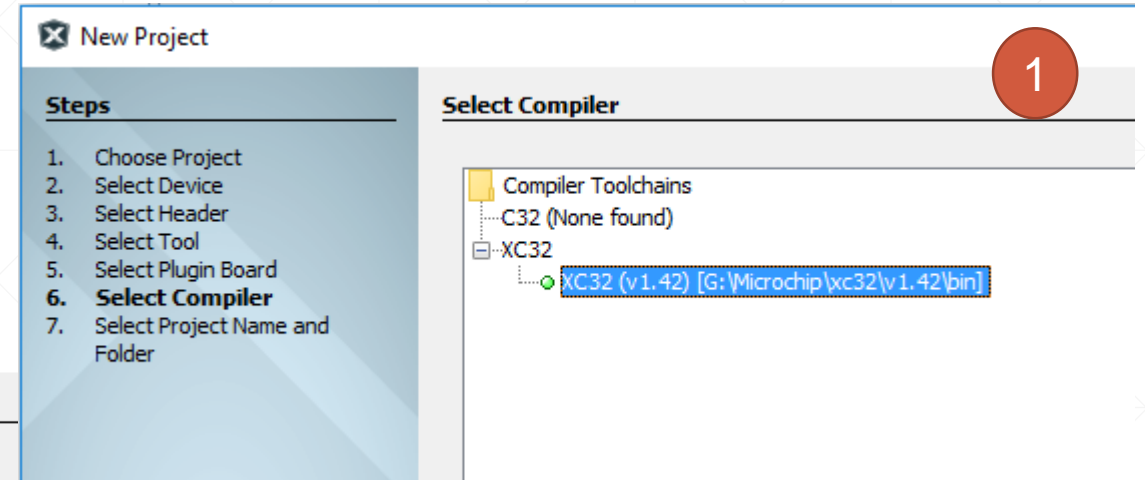
☐ Overwrite existing project.

☐ Also delete sources.

☒ Set as main project

☐ Use project location as the project folder

Encoding: ISO-8859-1



The image shows the 'New Project' dialog box in MPLAB X IDE, specifically the 'Select Compiler' step. It displays a tree view of 'Compiler Toolchains'. Under 'C32 (None found)', there is a sub-entry 'XC32'. Under 'XC32', there is a specific toolchain 'XC32 (v1.42) [G:\Microchip\xc32\v1.42\bin]' which is highlighted with a blue selection box. A red circle with the number '1' is in the top right corner.

**New Project**

**Select Compiler**

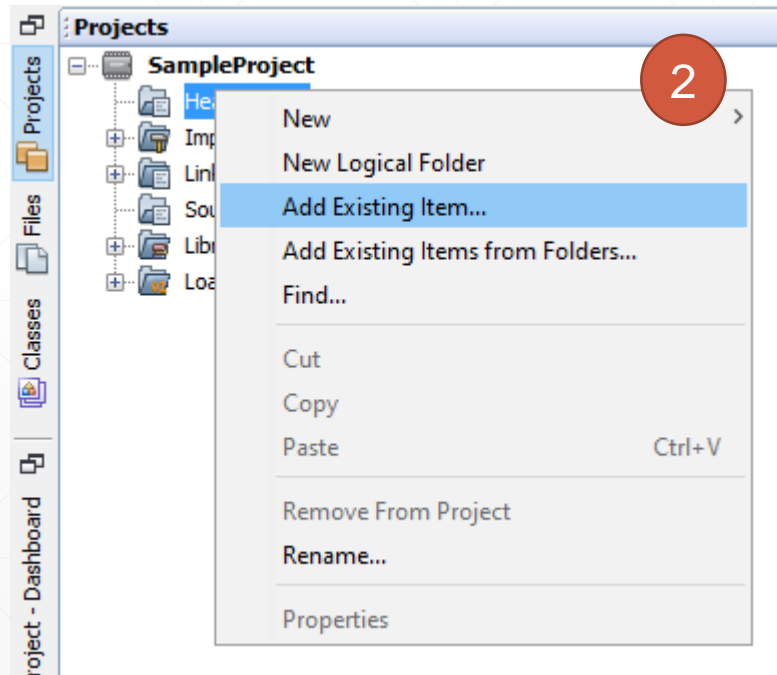
Compiler Toolchains

- C32 (None found)
- XC32
  - XC32 (v1.42) [G:\Microchip\xc32\v1.42\bin]**

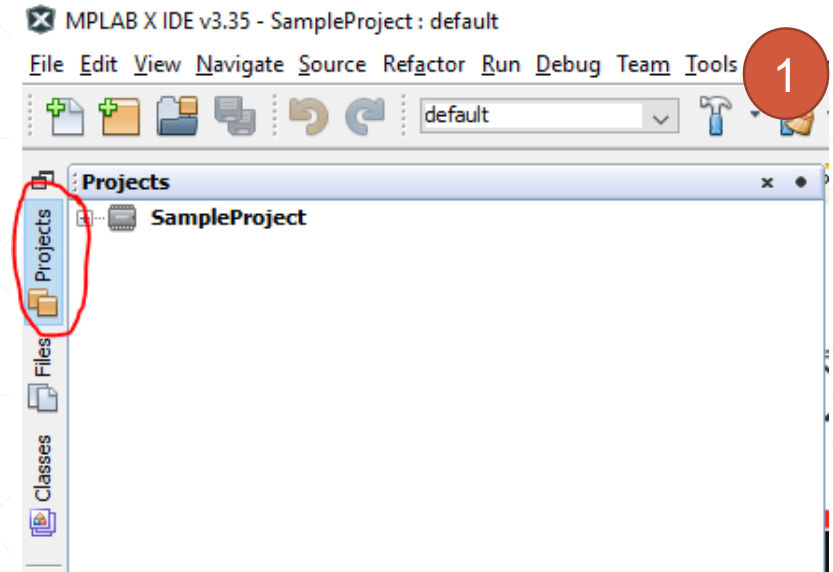
- Choose a name for your project and a location to save it.
- Click Finish!

# Getting Started with MPLAB X IDE – 6

- You can see your project here.



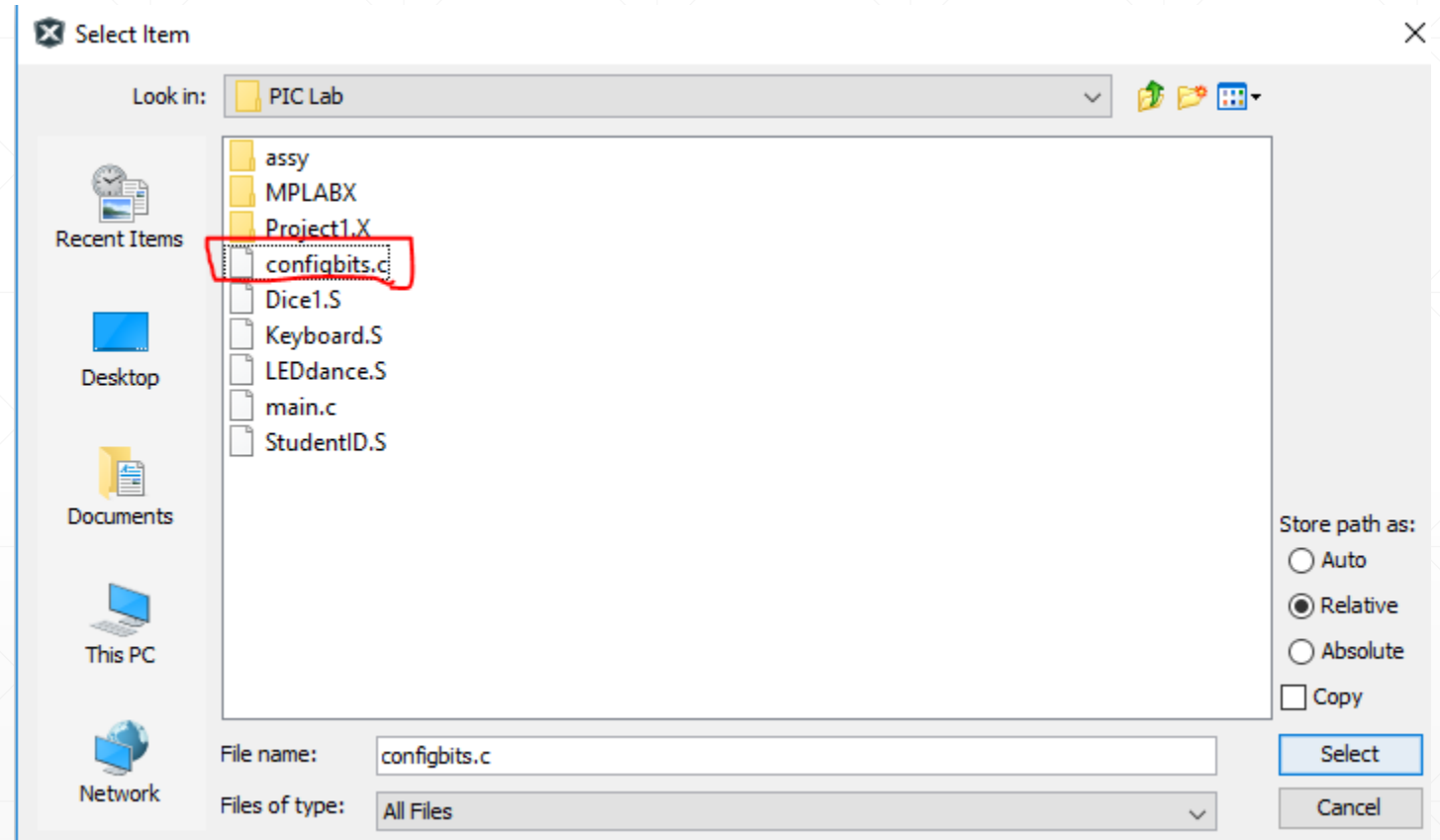
- Now you're going to add some files to your project.
- Right-click on Header Files > Add Existing Item...





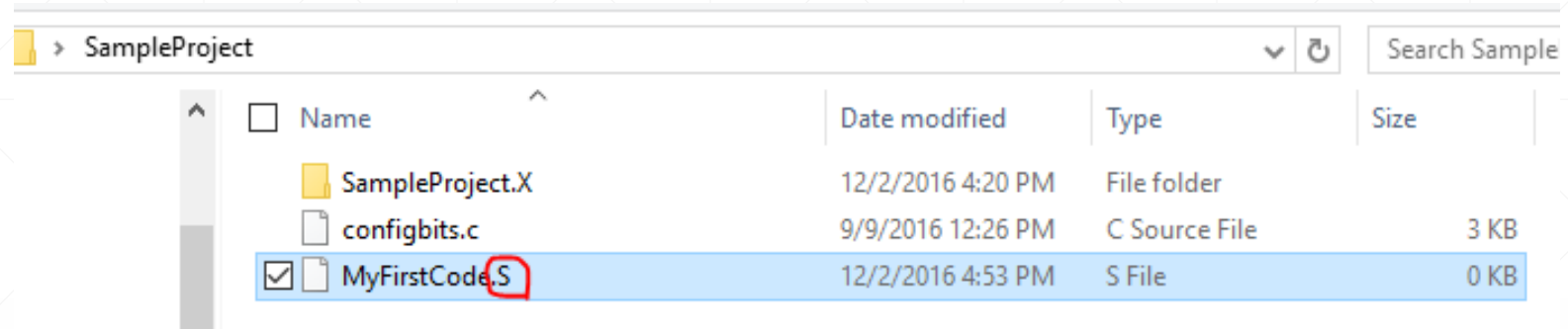
# Getting Started with MPLAB X IDE – 7

- You have been provided a file named “configbits.c” for the first lab session. Copy it to the directory where you have saved your project and then add it here as a header file. Extra information on this file will be provided later.



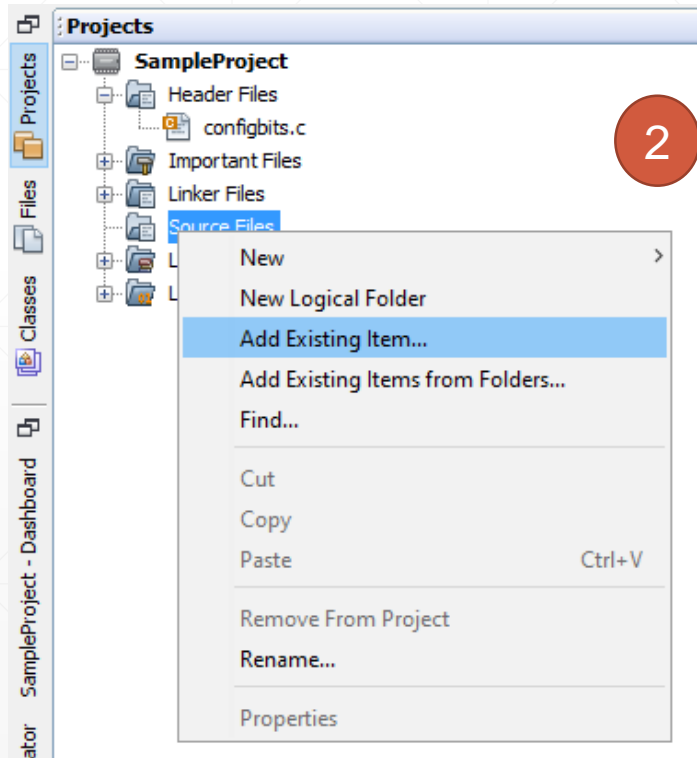
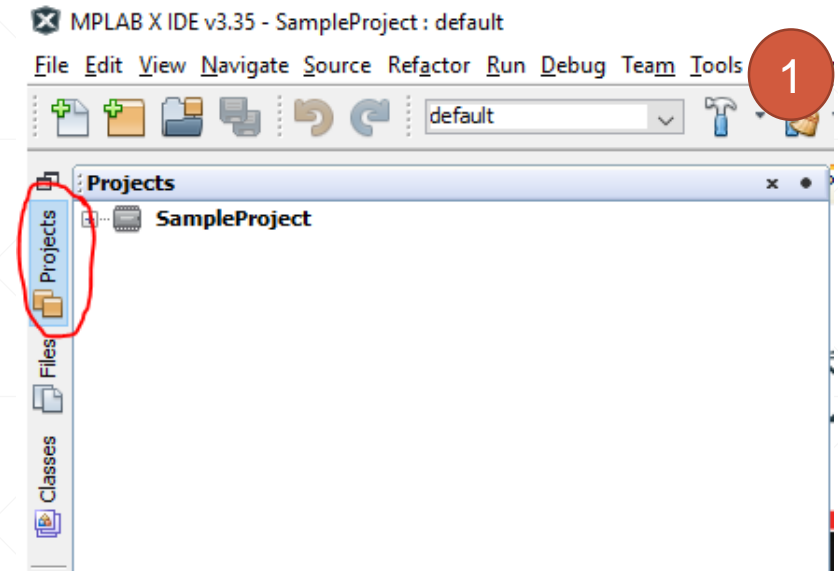
# Creating your assembly file

- Create a text file in the directory of your project, and then save it with a “.S” extension.
- Note the capital S. (not .s)
- This will be the main file in which you will write your assembly code.



# Getting Started with MPLAB X IDE – 8

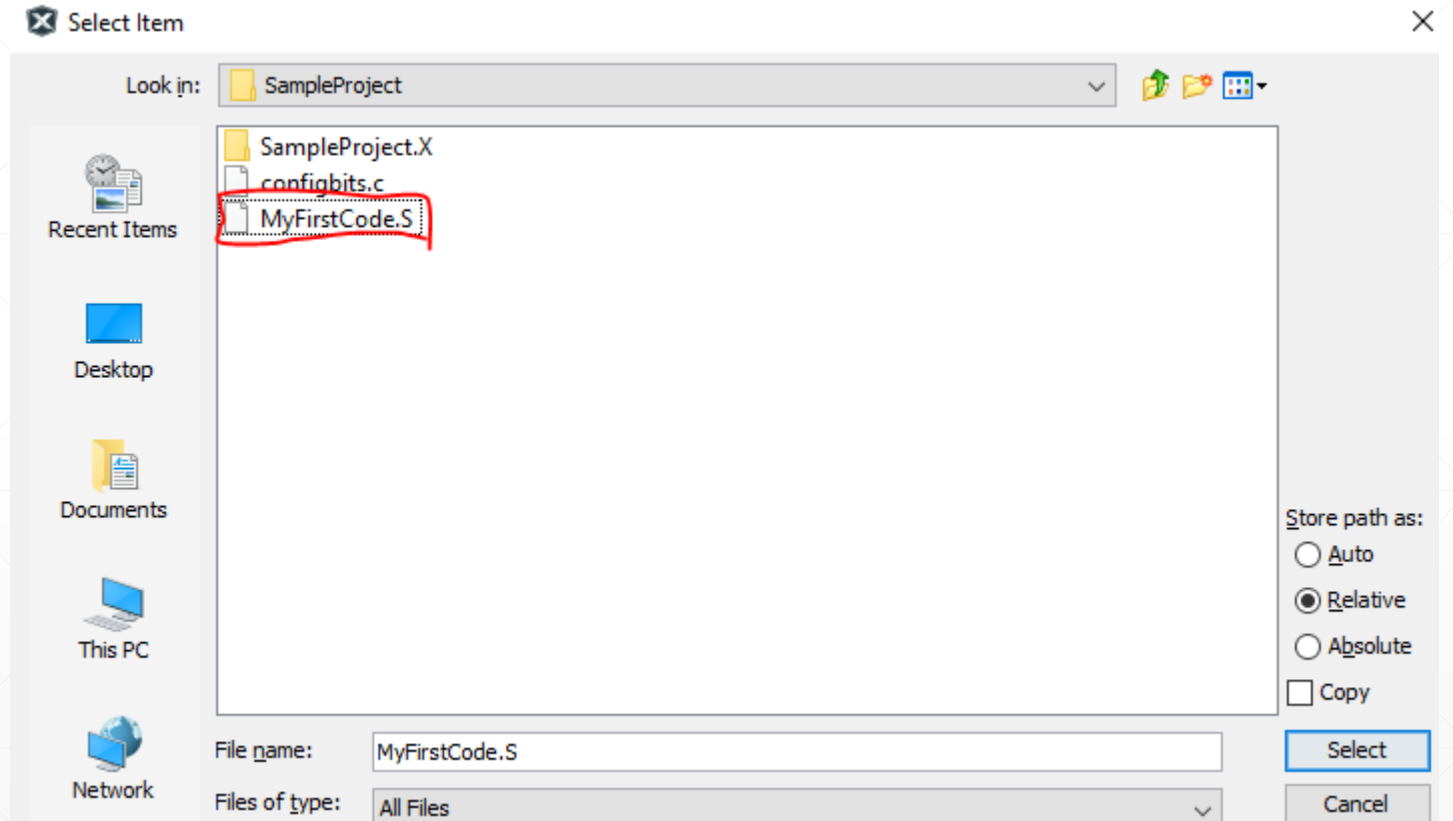
- Now go back to the MPLAB X IDE, and click on your project.



- You're going to add the file you just created, to your project.
- Right-click on **Source Files** > Add Existing Item...

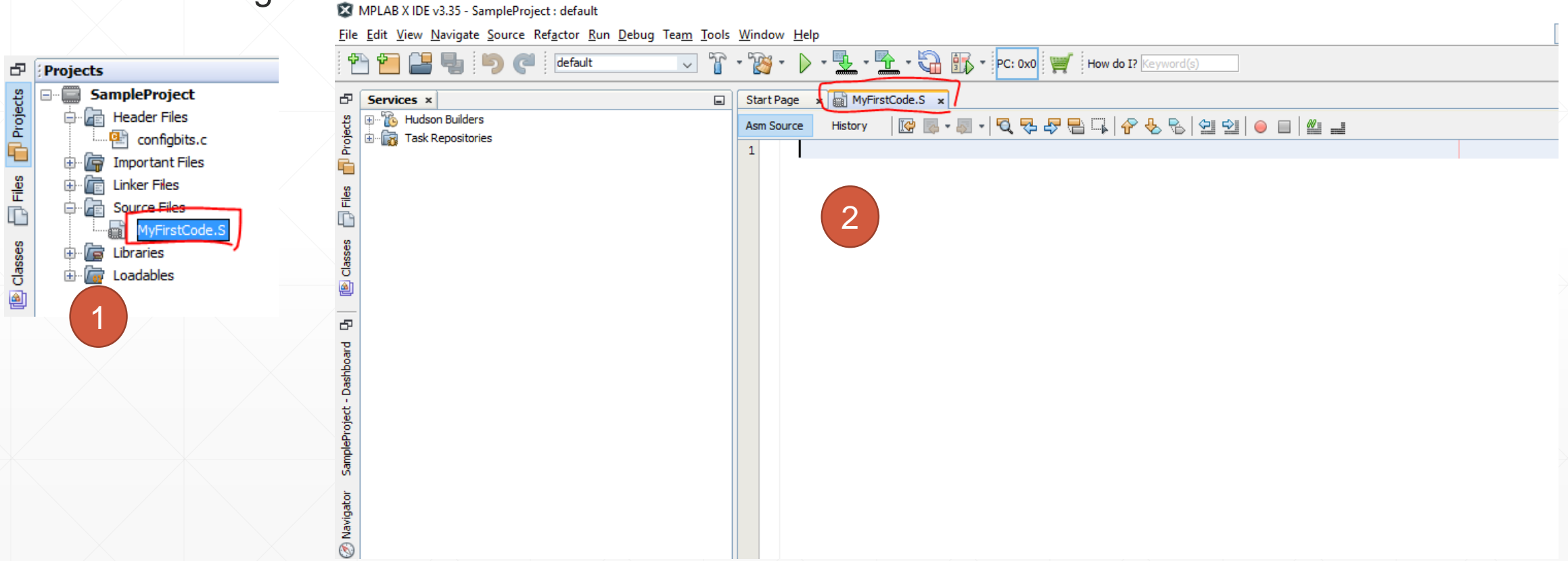
# Getting Started with MPLAB X IDE – 9

- Now add the .S file you just created.



# Getting Started with MPLAB X IDE – 10

- Now you can double-click on your assembly file to open it and start coding!

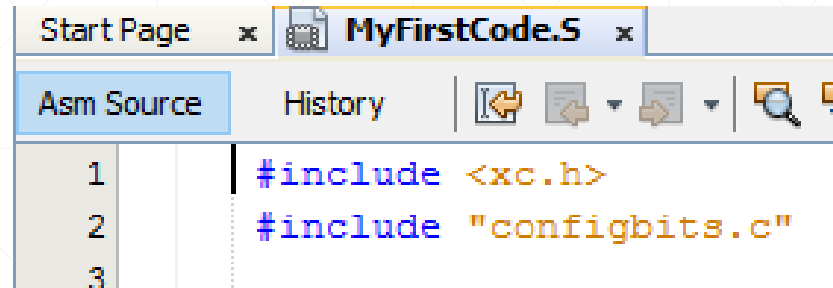


# **Write Your First Program!**

---

# First Things First!

- Include the compiler library.
- Include the header file you've added.



The screenshot shows a code editor window with the title 'MyFirstCode.S'. The editor has a menu bar with 'Start Page' and 'History'. Below the menu bar is a toolbar with icons for file operations and search. The main editing area shows assembly source code with line numbers 1, 2, and 3 on the left. The code consists of two include directives: line 1 is '#include <xc.h>' and line 2 is '#include "configbits.c"'. Line 3 is empty.

```
1  #include <xc.h>
2  #include "configbits.c"
3
```

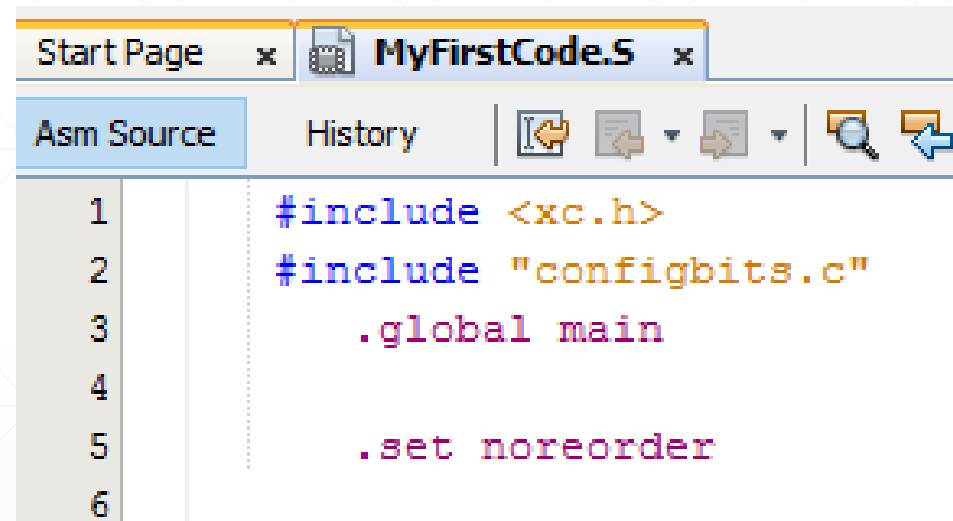
# .global, set “no reorder”

- .global :

Define all global symbols here,  
i.e. main in this program

- .set noreorder

This is important for an  
assembly programmer. This  
directive tells the assembler  
that don't optimize the order of  
the instructions as well as  
don't insert 'nop' instructions  
after jumps and branches.



The screenshot shows a code editor window titled 'MyFirstCode.S'. The editor has a menu bar with 'Start Page' and 'History', and a toolbar with icons for file operations and search. The main text area displays assembly source code with line numbers 1 through 6 on the left. The code contains the following directives:

```
1  #include <xc.h>
2  #include "configbits.c"
3  .global main
4
5  .set noreorder
6
```



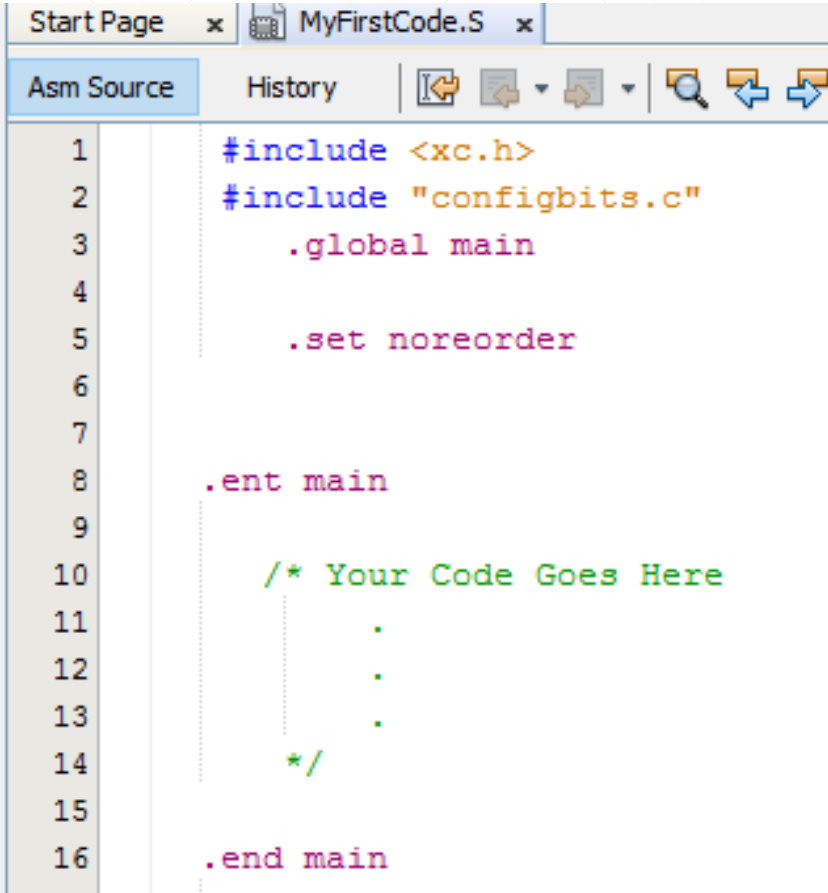
# main

- main

This is where the PIC32 start-up code will jump to after initial set-up.

It's just as in C code where we had main() function.

So your code will always begin with .ent main and end with .end main.



```
1  #include <xc.h>
2  #include "configbits.c"
3      .global main
4
5      .set noreorder
6
7
8  .ent main
9
10     /* Your Code Goes Here
11     .
12     .
13     .
14     */
15
16 .end main
--
```

# How to code!

- The instruction set for this IC is exactly the same as that of MIPS32 instruction set you've learned.
- There is only a slight difference:

When writing register names, do NOT use dollar sign(\$).

```
1  #include <xc.h>
2  #include "configbits.c"
3      .global main
4
5      .set noreorder
6
7
8  .ent main
9
10     addu t1, zero, 0x8000
11     sw t1, 0(t0)
12
13 .end main
```

# One Last Note

- Use **nop** after any jump or branch instruction.
- The PIC32 architecture is based on pipelined architecture. It's default is to always perform the instruction after jump or branch to improve throughput – you will learn more about delay branch slots during this course. So if you want your program to work properly, either take care of this instructions, or simply use a nop after any jump or branch instruction.

```
1  #include <xc.h>
2  #include "configbits.c"
3  .global main
4
5  .set noreorder
6
7
8  .ent main
9
10     main:
11     addu t1, zero, 0x8000
12     sw t1, 0(t0)
13     jmp main
14     nop
15
16 .end main
```

# I/O Ports

- PIC32MX110F016B has 2 I/O ports:

## PORTA & PORTB

- PORTA consists of 5 pins named RA0 through RA4
- PORTB consists of 16 pins named RB0 through RB15
- Each I/O pin may have other applications. For instance, RB0 may be used as PGED while programming the microcontroller.

Pin #	Full Pin Name
1	MCLR
2	VREF+/CVREF+/AN0/C3INC/RPA0/CTED1/RA0
3	VREF-/CVREF-/AN1/RPA1/CTED2/RA1
4	PGED1/AN2/C1IND/C2INB/C3IND/RPB0/RB0
5	PGEC1/AN3/C1INC/C2INA/RPB1/CTED1/RB1
6	AN4/C1INB/C2IND/RPB2/SDA2/CTED13/RB2
7	AN5/C1INA/C2INC/RTCC/RPB3/SCL2/RB3
8	Vss
9	OSC1/CLKI/RPA2/RA2
10	OSC2/CLKO/RPA3/PMA0/RA3
11	SOSCI/RPB4/RB4
12	SOSCO/RPA4/T1CK/CTED9/PMA1/RA4
13	VDD
14	PGED3/RPB5/PMD7/RB5

Pin #	Full Pin Name
15	PGEC3/RPB6/PMD6/RB6
16	TDI/RPB7/CTED3/PMD5/INT0/RB7
17	TCK/RPB8/SCL1/CTED10/PMD4/RB8
18	TDO/RPB9/SDA1/CTED4/PMD3/RB9
19	Vss
20	Vcap
21	PGED2/RPB10/CTED11/PMD2/RB10
22	PGEC2/TMS/RPB11/PMD1/RB11
23	AN12/PMD0/RB12
24	AN11/RPB13/CTPLS/PMRD/RB13
25	CVREFOUT/AN10/C3INB/RPB14/SCK1/CTED5/PMWR/RB14
26	AN9/C3INA/RPB15/SCK2/CTED6/PMCS1/RB15
27	AVss
28	AVDD

# ANSELx, TRISx, LATx and PORTx Registers

- Each port has 4 special function registers: ANSELx, TRISx, LATx and PORTx.
- For instance, we have 4 registers ANSELB, TRISB, LATB and PORTB associated to port B.
- You will have to set the first 2 registers in order to be able to use I/O ports.

## ANSELx – Analog Select

- The ANSELx register controls the operation of the analog port pins. In order to use port pins for I/O functionality with digital modules, the corresponding ANSELx bit must be cleared.
- The ANSELx register has a default value of 0xFFFF; therefore, all pins that share analog functions are analog (not digital) by default.
- So for example if you want to use PORTB pins as digital I/O pins, you have to clear all the bits of ANSELB register.

# TRISx

- The TRISx register determines whether an I/O pin is set as input, or output. In order to use a pin as output, you will have to clear the corresponding bit in TRISx register. Vice Versa, if you want to use a pin as input, you will have to set the corresponding bit.
- For example, suppose that you want to use RB2 as input pin and RB3 as an output. What you have to do is to set TRISB[2] and clear TRISB[3].

# LATx – Output Latch

- You use this register when you are using a pin as output. You write a data in the corresponding bit, and then it automatically is sent to the output pin.
- For example, suppose that you want to use RB3 as an output pin. You write the desired value (0 or 1) into LATB[3] and then it appears on RB3.



# PORTx

- You use this register when you are using a pin as input. In fact, you can read the input data through this register.
- For example, suppose that you want to read data from pins RB7 through RB0. What you have to do is to read the value of the bits PORTB[7:0].

# la Pseudo Instruction

- This is the only pseudo instruction you are allowed to use during this course!
- la (load address) is a pseudo code which returns the address of a register on the memory space. For instance, la t0, ANSELB will load the address of ANSELB register into t0 register.
- Exercise: Think about what you can use instead of la pseudo instruction. You may take a look at the datasheet of the IC.

# Sample Program

- Here is a sample code which is meant to read data from RB1 and RB0 continuously.
- t0 register always gets equal to the input from RB1 and RB0.

```
1  #include <xc.h>
2  #include "configbits.c"
3  .global main
4  .set noreorder
5  .ent main
6  main:
7      /*add s0, zero, zero
8      lui s0, 0xbf88 */
9      /* ANSELB = 0x0000 Assigns Port B as Digital I/O */
10     la t0, ANSELB
11     sw zero, 0(t0)
12     /* equivalent instruction for both of the above instructions is:
13     * sw zero, 0x6100(s0) */
14
15     /*TRISBbits.TRISB0 = TRISBbits.TRISB0 = 1 Assigning these pins as input,
16     * so as to read data from these pins.
17     * TRISB = 0x0003 */
18     ori t1, zero, 3
19     la t0, TRISB
20     //sw t1, 0x6110(s0)
21     sw t1, 0(t0)
22     //Now RB0 and RB1 are set as input.
23     la t1, PORTB
24     loop:
25     lw t0, 0(t1)
26     j loop
27     nop
28 .end main
```

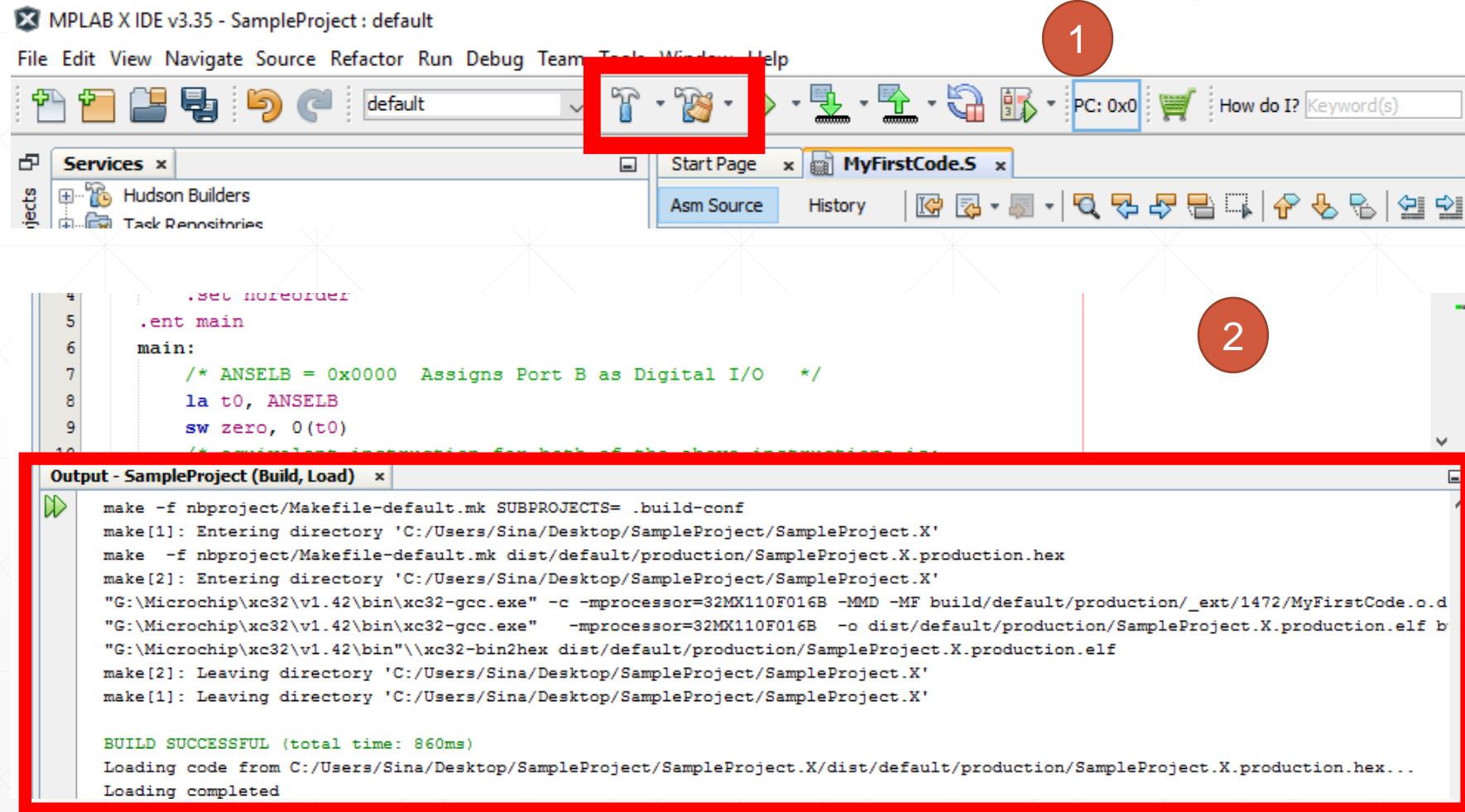
# **Programming Your Code into the Microcontroller**

---

# Getting Started with MPLAB X IDE – 11

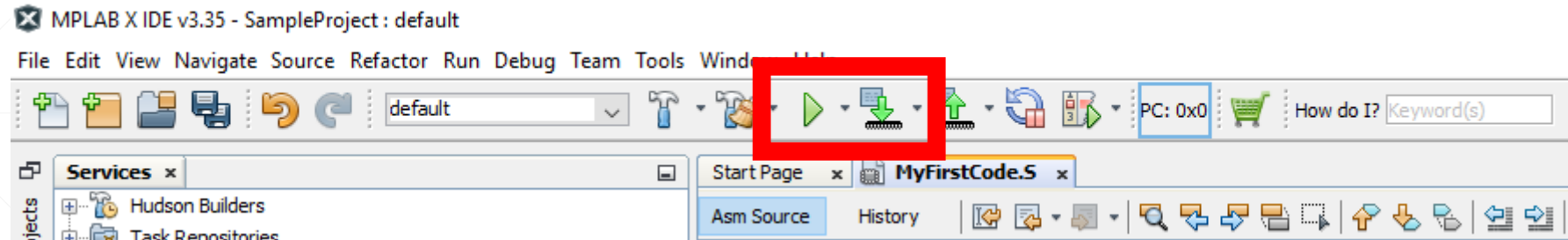
- Now that you have written your first piece of code, you can test it. Click on “Build” button. (Or clean and build for the next times)

- If your code has no syntax errors, you are going to see something like this in the output window below the MPLAB X IDE window:



# Getting Started with MPLAB X IDE – 12 : Final Move!

- After making sure that everything is ok, you're going to program your code into PIC32 micro.
- Plug the PICkit3 Programmer kit into your computer.
- Click on either “Run Main Project” or “Make and Program Device” button in order to program the device.
- Select the right Programmer Device (PICkit3) and start programming.



# Testing the circuit

- When testing your circuit, don't forget about minimum connection requirements:

1) Master Clear Pin

2) AVdd, AVss (You are recommended to connect Vdd and Vss pins to these, but not essential – note that IC gets its power supply through AVdd & AVss – not Vdd & Vss)

3) Vcap

## Last reminder!

The recommended AVdd to AVss voltage is 3.25 up to 3.5v.

**Never ever connect the IC to a voltage higher than 4v!**



That's it! Enjoy.

---