Lab 2: Application of a Digital Latch

ECE 3720

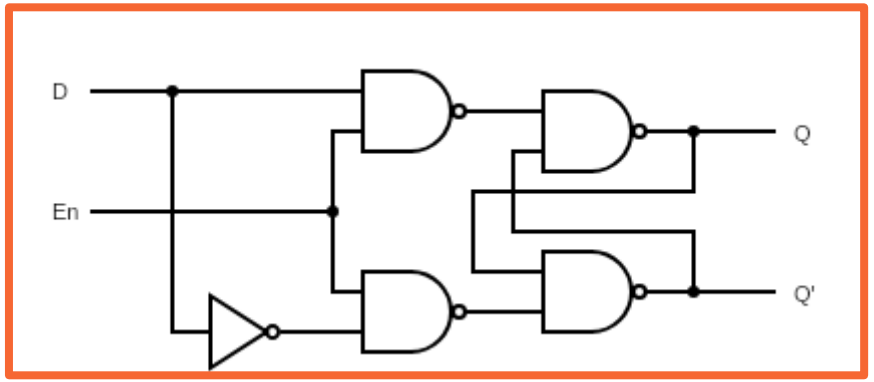
Preview

The microcontroller will read two digital inputs and output those same values to a latch. When a button is pressed, the latch outputs will be updated to mirror the inputs. The states of the latch's inputs and outputs will be indicated by LEDs.

Latches vs Flip-Flops

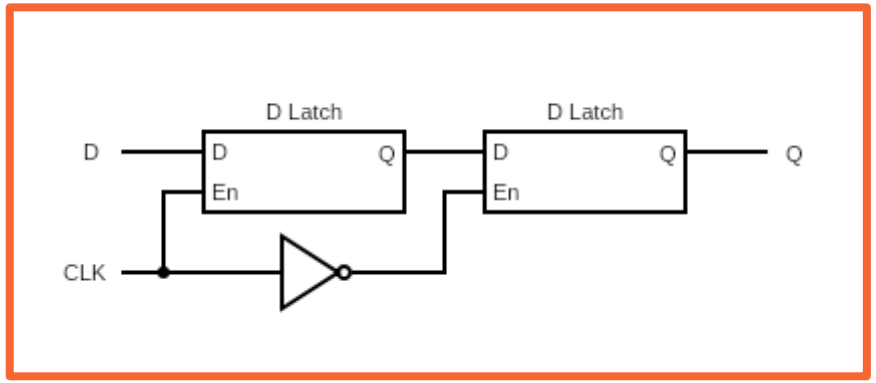
- Both latches and flip-flops are storage elements for holding a binary state.
- Latches are *level-sensitive*, while flip-flops are *edge-sensitive*.

D Latch



- While En is HIGH, Q mirrors D.
- When En is LOW, Q remains constant.
- **This is what will be used in this lab**.

D Flip-Flop



- Q updates to match D on rising edge of CLK.
- Consists of two D latches
- Useful for sequential circuits, since it will only update at the times determined by the clock.

SN74LS373

- Datasheet available here
- Notice that this datasheet covers multiple devices, including both latches and flip-flops.
- Use the datasheet to learn the behavior and pinout of the SN74LS373
  - Notice that the pins are arranged in a Q-D-D-Q-Q… pattern

> The eight latches of the 'LS373 and 'S373 are transparent D-type latches, meaning that while the enable (C or CLK) input is high, the Q outputs follow the data (D) inputs. When C or CLK is taken low, the output is latched at the level of the data that was set up.

SN74LS373 datasheet, pg. 1

We use this

Output updates as long as C is HIGH

Not this

Output updates on rising edge of CLK

Function Tables

'LS373, 'S373 (each latch)

INPUTS			OUTPUT
$\overline{OC}$	C	D	Q
L	H	H	H
L	H	L	L
L	L	X	Q_0
H	X	X	Z

'LS374, 'S374 (each latch)

INPUTS			OUTPUT
$\overline{OC}$	CLK	D	Q
L	↑	H	H
L	↑	L	L
L	L	X	Q_0
H	X	X	Z

SN74LS373 datasheet, pg. 3

Registers

- Every lab in this class will involve setting the values of the PIC32's registers to achieve the desired behavior.
  - In particular, the I/O registers detailed in the following slides will be some of the first things to consider for each lab.
  - Register names refer to memory locations, defined through plib.h.
- Xbits functionality
  - Type "bits." after a register name in MPLAB to see a list of individual bits from that register.
  - This is often preferable, since it improves readability and lets you address individual bits, leaving the others unchanged.

```
TRISBbits.
    TRISB0   unsigned
    TRISB1   unsigned
    TRISB10  unsigned
    TRISB11  unsigned
    TRISB12  unsigned
    TRISB13  unsigned
    TRISB14  unsigned
    TRISB15  unsigned
    TRISB2   unsigned
```

11.1 Parallel I/O (PIO) Ports

All port pins have 10 registers directly associated with their operation as digital I/O. The data direction register (TRISx) determines whether the pin is an input or an output. If the data direction bit is a '1', then the pin is an input. All port pins are defined as inputs after a Reset. Reads from the latch (LATx) read the latch. Writes to the latch write the latch. Reads from the port (PORTx) read the port pins, while writes to the port pins write the latch.

PIC32 datasheet, pg. 144

Data Direction

- Use TRISx registers to designate pins as inputs or outputs.
  - where x is A, B, or C
  - TRIS is short for “Tri-State”
- Set bitwise, 1 for input, 0 for output
  - (“1nput or 0utput”)

```
TRISB = 0b00001111;      // Set B0-B3 as inputs and B4-B7 as outputs
TRISB = 0x0F;            // Equivalent
TRISB = 15;              // Equivalent

TRISBbits.TRISB2 = 1;    // Set just B2 as an input
TRISBbits.TRISB2 = 0;    // Set just B2 as an output
```

11.1 Parallel I/O (PIO) Ports

All port pins have 10 registers directly associated with their operation as digital I/O. The data direction register (TRISx) determines whether the pin is an input or an output. If the data direction bit is a ‘1’, then the pin is an input. All port pins are defined as inputs after a Reset. Reads from the latch (LATx) read the latch. Writes to the latch write the latch. Reads from the port (PORTx) read the port pins, while writes to the port pins write the latch.

PIC32 datasheet, pg. 144

Read / Write

- Read from PORTx
  - When a pin is designated as an input, reading the corresponding bit of PORTx will return the value on that pin.

```
int x;
x = PORTAbits.RA0;      // Read the value of A0 into variable x
```

- Write to LATx
  - When a pin is designated as an output, writing to the corresponding bit of LATx will output that value on the pin.

```
LATCbits.LATC0 = 1;     //Write a 1 (HIGH)to C0

int y = 12;
LATB = y;           // Write the value of variable y to Latch B
                    //(consider how many bits/pins this requires)

LATCbits.LATC0 = PORTAbits.RA0; // What does this do?
```

11.1 Parallel I/O (PIO) Ports

All port pins have 10 registers directly associated with their operation as digital I/O. The data direction register (TRISx) determines whether the pin is an input or an output. If the data direction bit is a '1', then the pin is an input. All port pins are defined as inputs after a Reset. Reads from the latch (LATx) read the latch. Writes to the latch write the latch. Reads from the port (PORTx) read the port pins, while writes to the port pins write the latch.

PIC32 datasheet, pg. 144

Analog Select

- Use ANSELx to set pins to analog or digital mode.
- 0 for digital, 1 for analog
  - We will typically use digital.
- Use reference manual to determine which pins are analog capable.

```
ANSELB = 0;              // Set all B pins to digital
ANSELBbits.ANSB0 = 1;    // Set B0 to analog mode
```

11.1.2 CONFIGURING ANALOG AND DIGITAL PORT PINS

The ANSELx register controls the operation of the analog port pins. The port pins that are to function as analog inputs must have their corresponding ANSEL and TRIS bits set. In order to use port pins for I/O functionality with digital modules, such as Timers, UARTs, etc., 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.

PIC32 datasheet, pg. 144

Pull-up/Pull-down Enable

- Use CNPUx/CNPDx to set the internal pull-up/pull-down resistors
- 0 for off, 1 for on
  - By default are off
- Pull-up resistors require a 5V tolerable pin. Use reference manual or pinout to determine which pins are 5V tolerable.

```
CNPUBbits.CNPUB10 = 1;   // Enable internal pull-up resistor on B11
CNPDBbits.CNPDB1  = 1;   // Enable internal pull-down resistor on B1
```

11.1.4 INPUT CHANGE NOTIFICATION

The input change notification function of the I/O ports allows the PIC32MX1XX/2XX devices to generate interrupt requests to the processor in response to a change-of-state on selected input pins. This feature can detect input change-of-states even in Sleep mode, when the clocks are disabled. Every I/O port pin can be selected (enabled) for generating an interrupt request on a change-of-state.

Five control registers are associated with the CN functionality of each I/O port. The CNENx registers contain the CN interrupt enable control bits for each of the input pins. Setting any of these bits enables a CN interrupt for the corresponding pins.

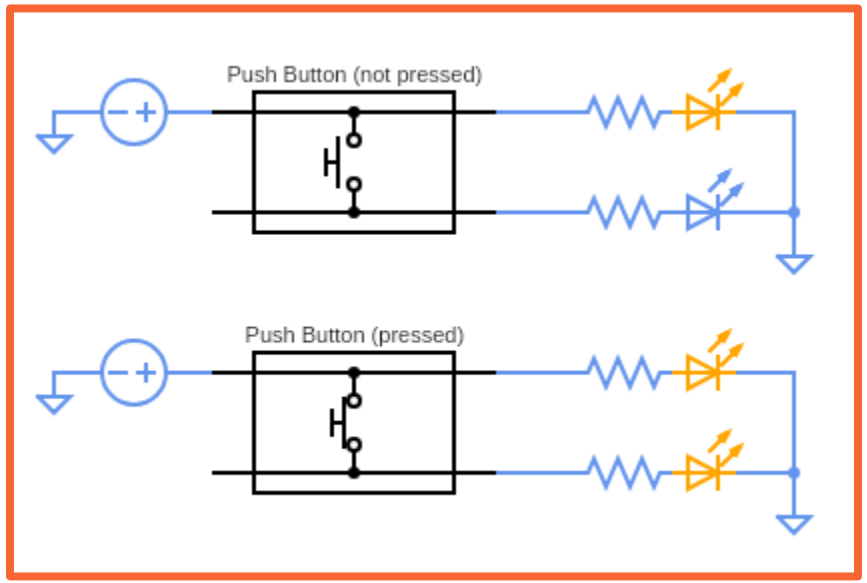
The CNSTATx register indicates whether a change occurred on the corresponding pin since the last read of the PORTx bit.

Each I/O pin also has a weak pull-up and a weak pull-down connected to it. The pull-ups act as a current source or sink source connected to the pin, and eliminate the need for external resistors when push-button or keypad devices are connected. The pull-ups and pull-downs are enabled separately using the CNPUx and the CNPDx registers, which contain the control bits for each of the pins. Setting any of the control bits enables the weak pull-ups and/or pull-downs for the corresponding pins.

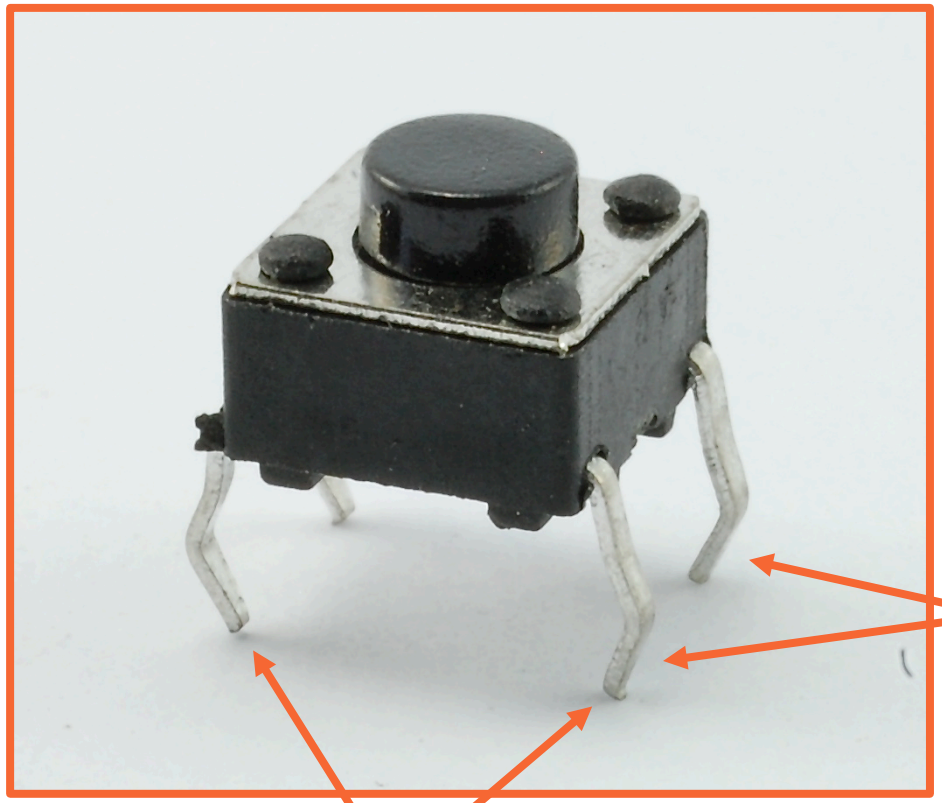
PIC32 datasheet, pg. 144

Push Buttons

Illustration of Internal Connection



Notice that the top LED in this setup is always connected to the voltage source, but the bottom is only connected when the button is pressed.



Connected when button is pressed

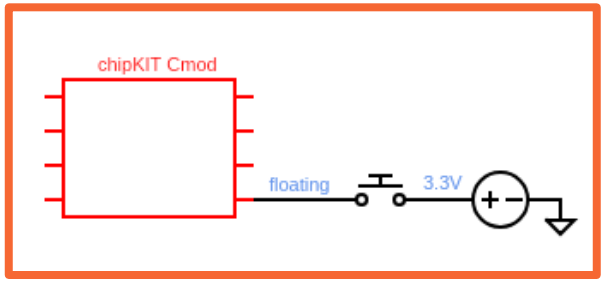
Always connected

Pull-Up and Pull-Down Resistors

- If a microcontroller's input pin is left floating, it may not read the correct digital value.
- Pull-up and pull-down resistors should be used to ensure consistent behavior.

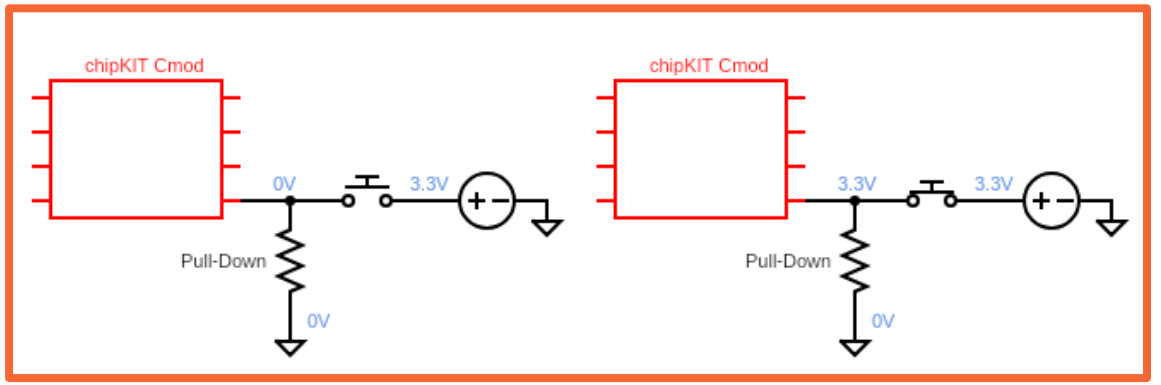
Floating (incorrect)

Pin has indeterminate value until button is pressed



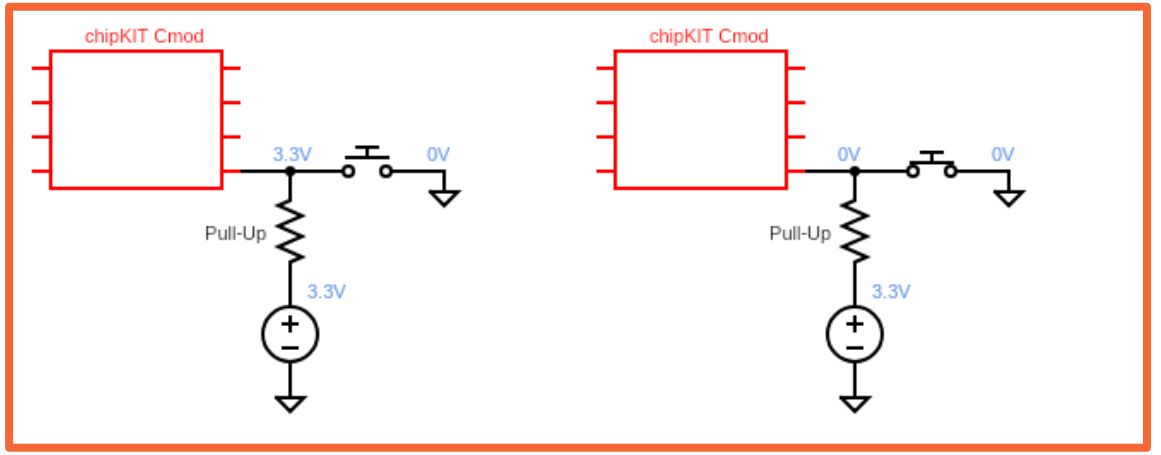
Pull-down

Pin is pulled LOW until button is pressed



Pull-up

Pin is pulled HIGH until button is pressed

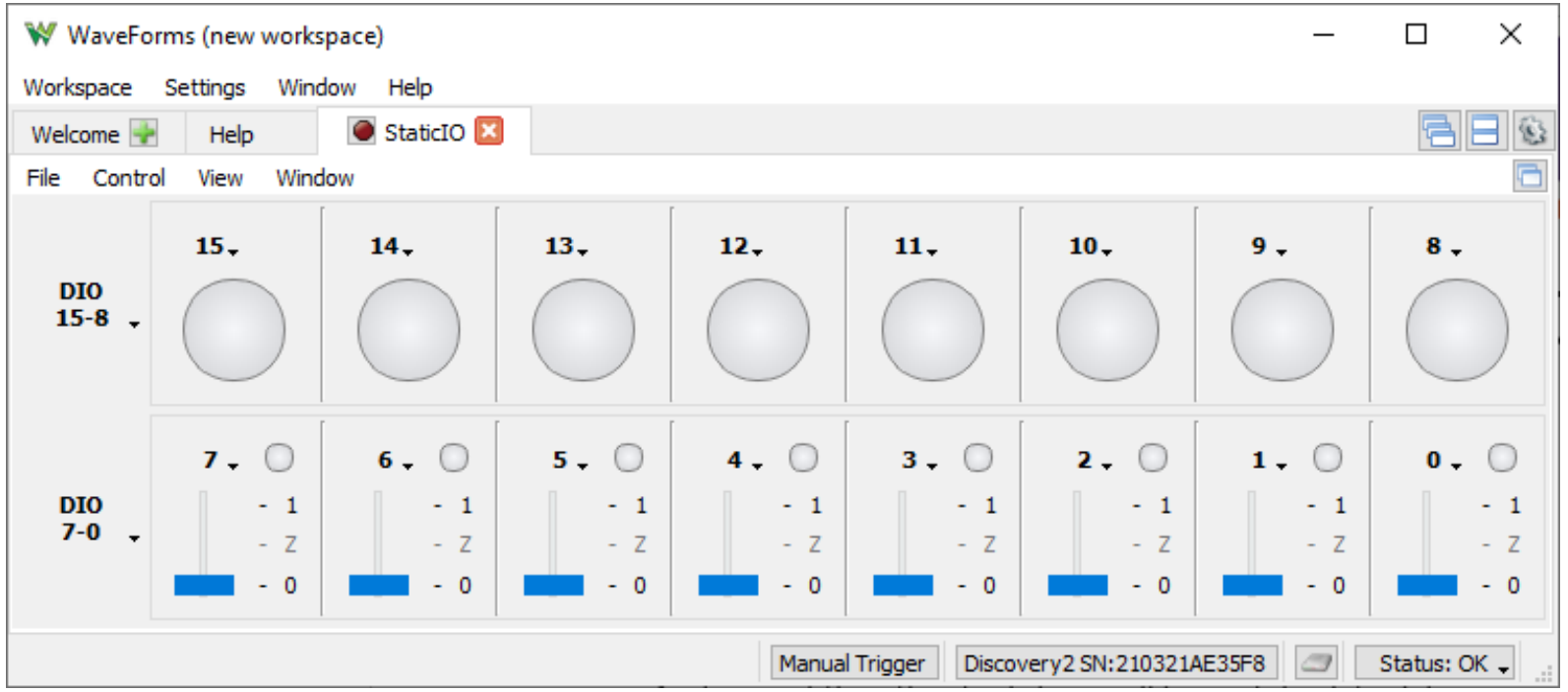


Active-Low vs Active-High

- An active-high signal is “on” when its voltage is HIGH.
- An active-low signal is “on” when its voltage is LOW.
  - e.g., the Output Control (/OC) pin of the SN74LS373
- **In this lab, we want the button’s output to be ACTIVE-LOW.**
  - **This means the MC will read LOW when the button is pressed, and HIGH otherwise.**
  - Reference the diagrams in the previous slide to determine how to wire the button to achieve this.
  - Since the button will be used to update the latch, consider what this means for the program you will write.

DIO (Digital Input/Output)

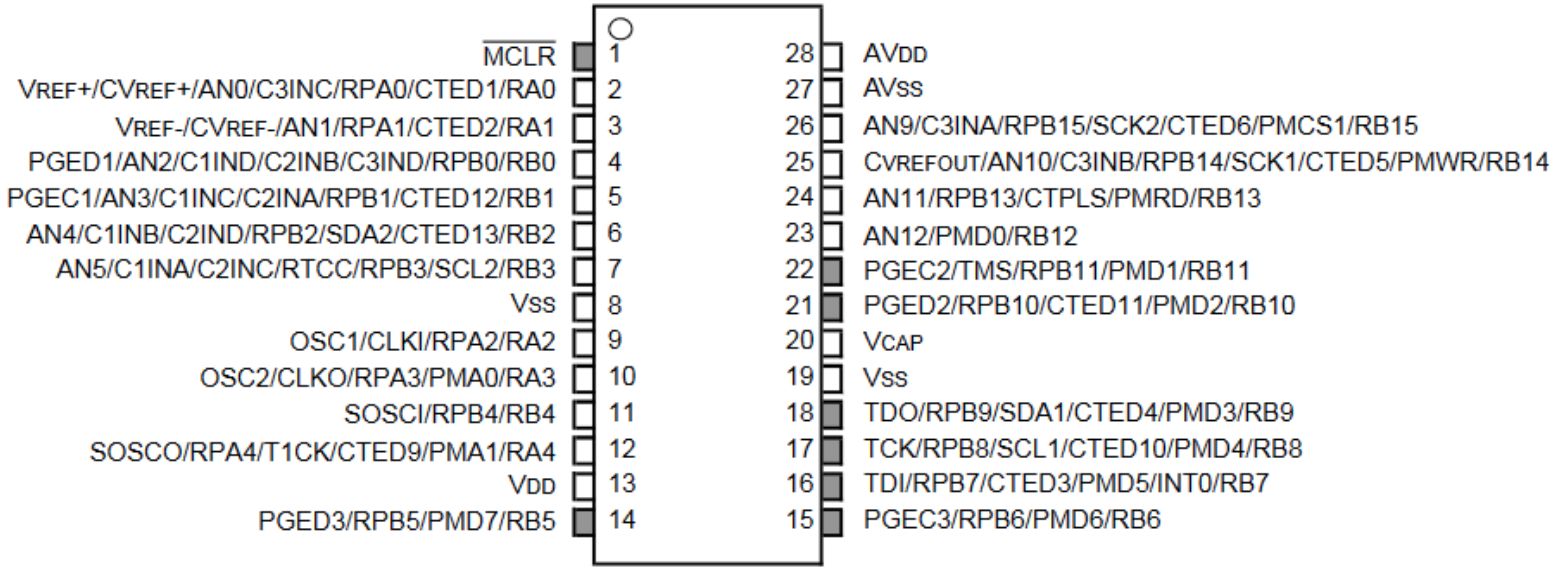
- Open the *WaveForms* application and select *StaticIO.*
- Click the drop-down arrow(s) and select push/pull switch.
- The DIO pins are labeled 0-15 on the AD2's case.



Notes on Voltages

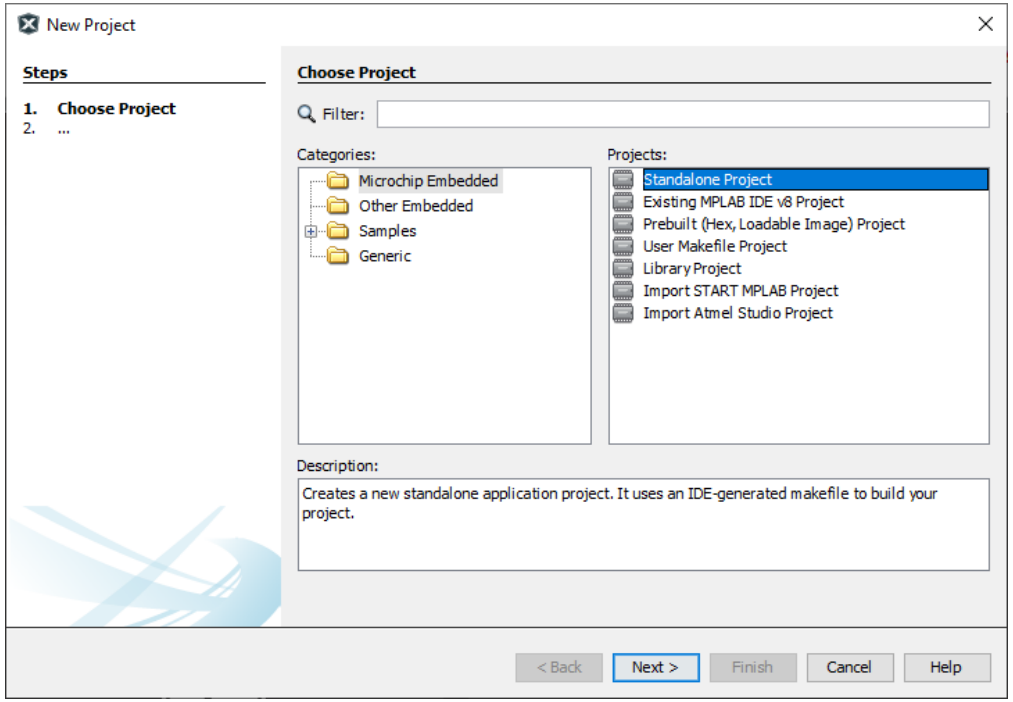
- The PIC32 runs on 3.3V, **you must set your AD2 to output 3.3V.**

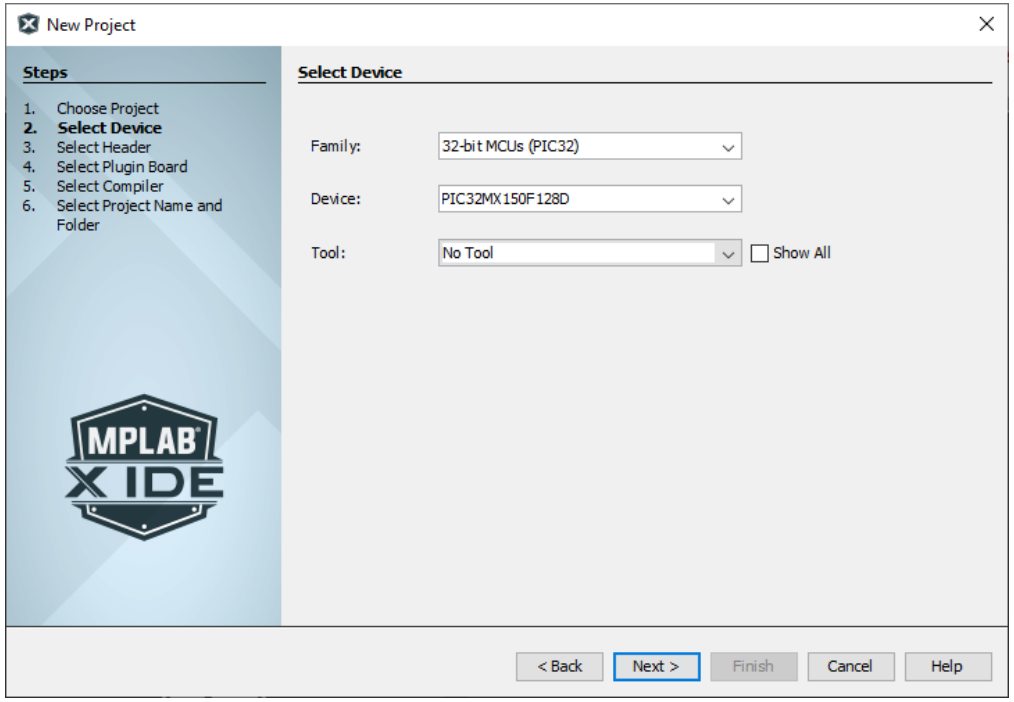
- **Only some of the PIC32's pins are 5V tolerant.**
  - Carefully read the pinout diagram in the reference manual to choose an appropriate pin.



Creating a new project

1. Open MPLAB X
2. Select *File > New Project* (or click )
3. Select *Standalone Project* and click Next
4. Select *32-bit MCUs*
5. Select *PIC323MX150F128B* and click Next

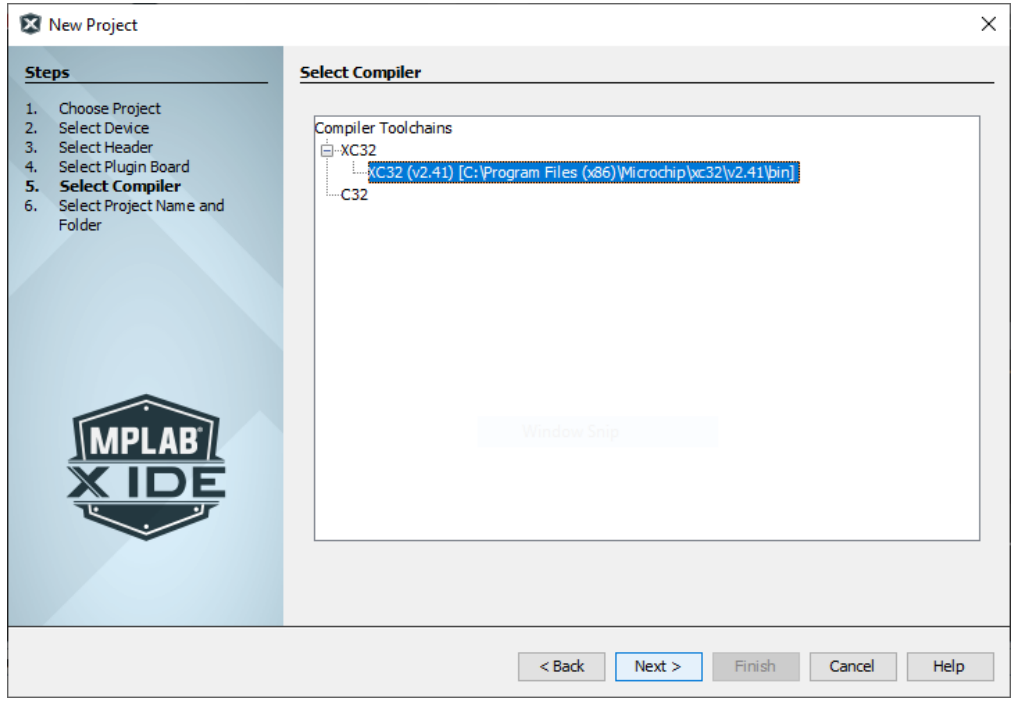




Creating a new project

6. Select the XC32 compiler and click Next

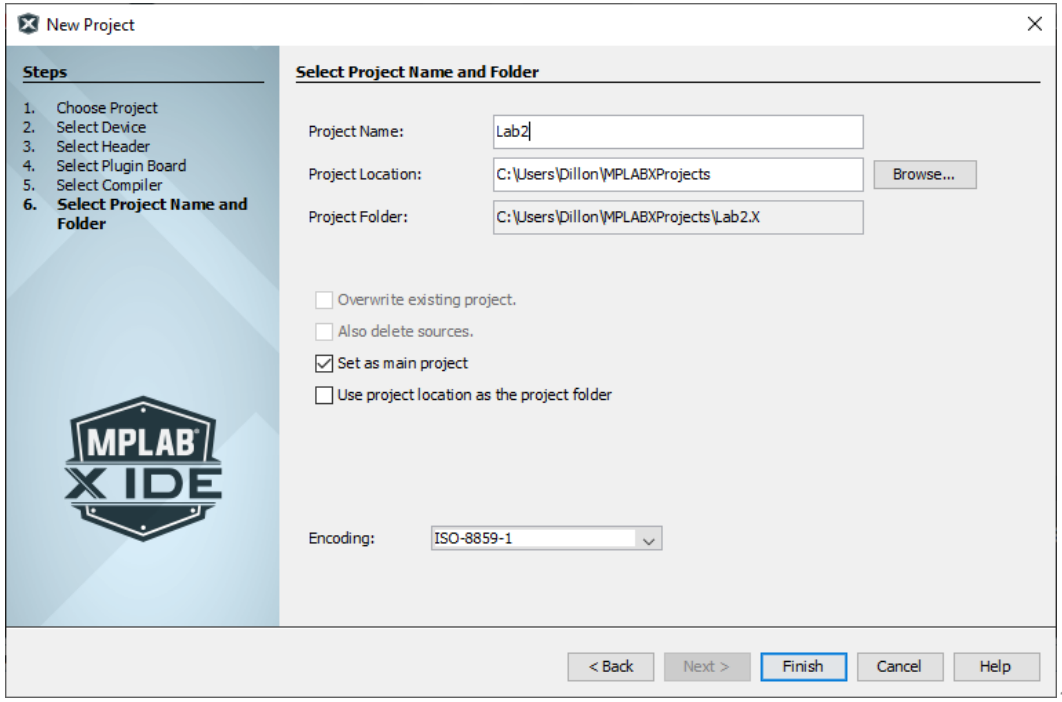
(If you have multiple compilers, select the highest version number)



7. Name the project

8. Select a save location you can always access

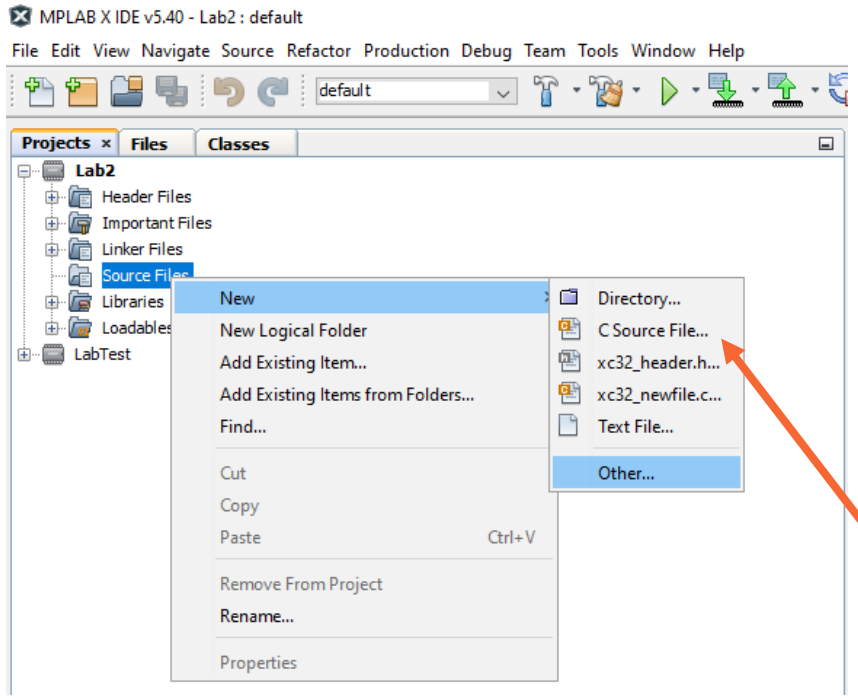
9. Click Finish



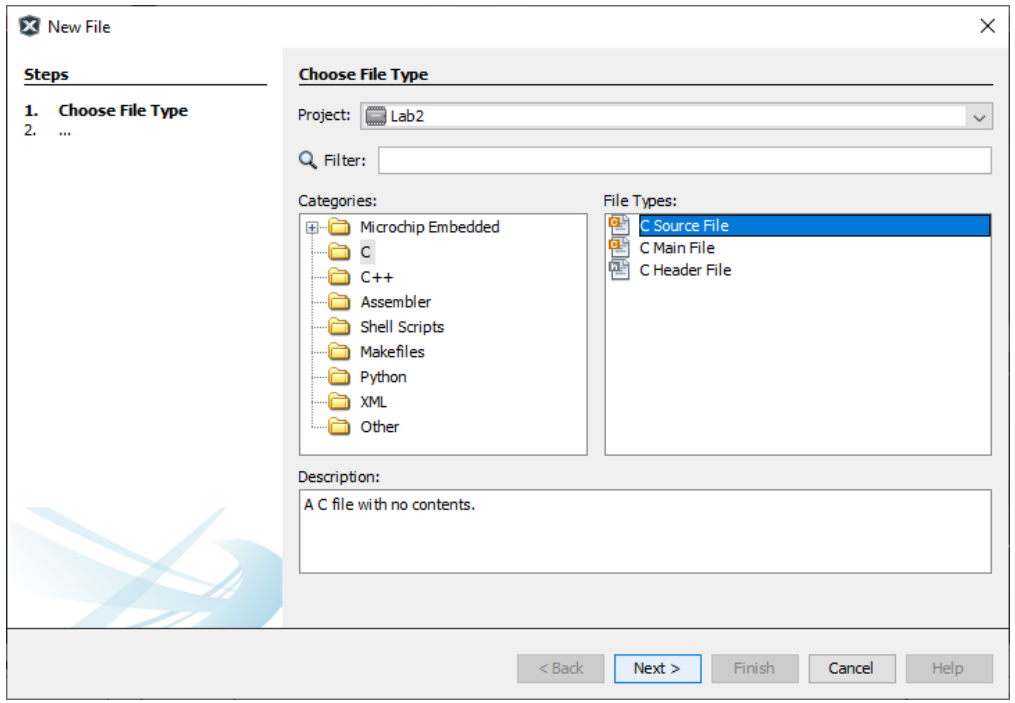
Creating a new project

10. Right-click *Source Files* under project name

11. Select *New > Other…*



12. Select *C* and *C Source File,* then click Next

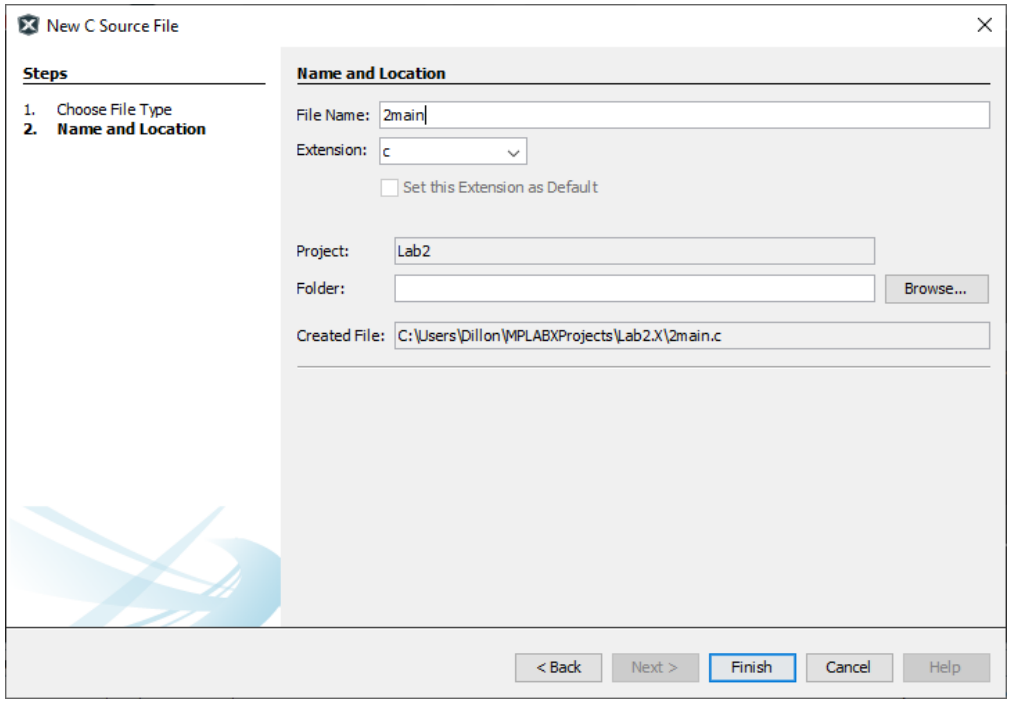


After these steps have been done once,
New > C Source File will become an option.

Creating a new project

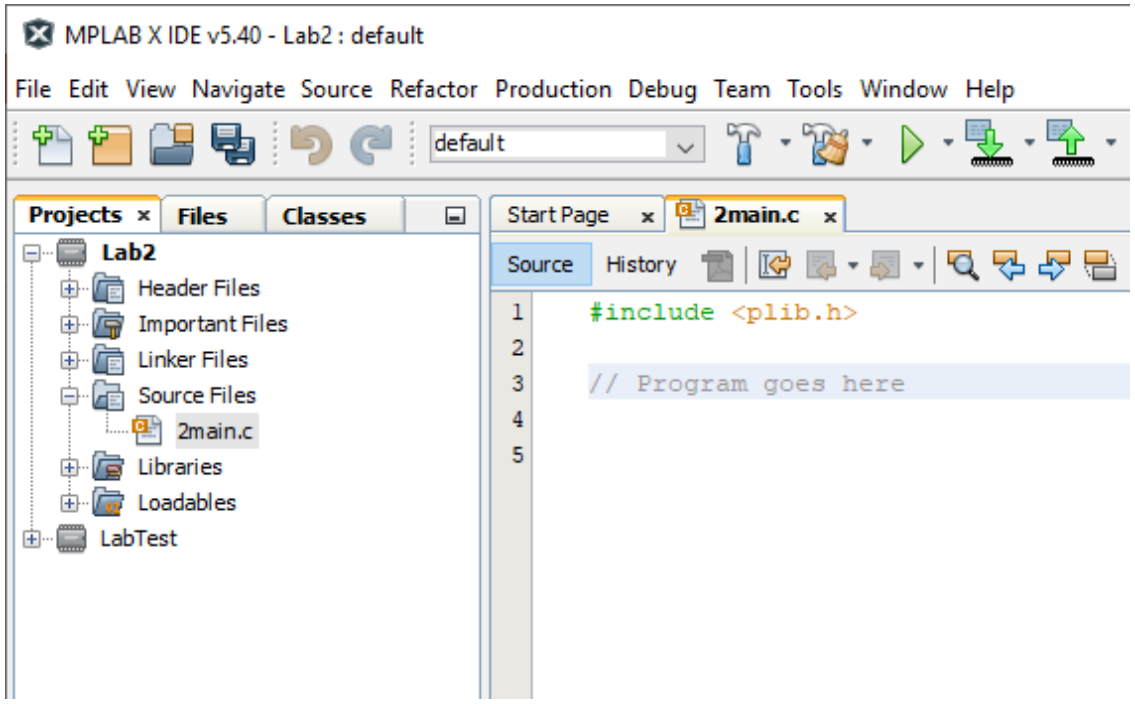
13. Name the source file and click Finish

(It is helpful to give each project's source file a different name.)



14. Make sure to include *xc.h*

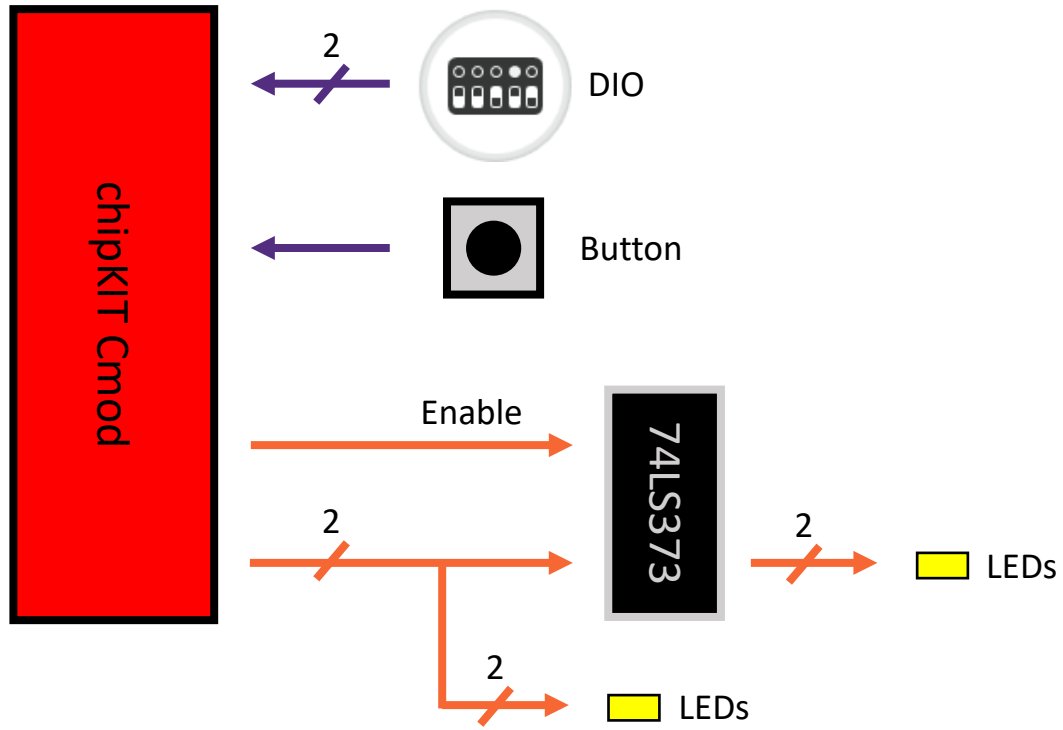
You can now write your program.



Lab Goals

- Read two inputs from DIO with the MC.
- Output those same values from the MC to both the latch and the first pair of LEDs.
- Read an active-low input signal from a button.
- When the button is pressed, output a signal from the MC to enable the latch.
- The second pair of LEDs, connected to the latch outputs, should update to match the inputs when the button is pushed.

Simple Diagram*



*Your pre-lab diagram must include the specific pins to be used, as well as resistors, connections to power and ground, etc.