Mechatronics Workshop-in-a-Box

When you walk out of here today you will know…

- Why designing a PIC® microcontroller into a mechanical system is *beneficial to you*
- How *easy* it is to get started with Microchip and PIC® microcontrollers
- How to use Microchip's low-cost tools to *change*, *adapt* and *add features* to a design
- How to *perform basic mechatronic tasks* using a PIC® microcontroller

Agenda

- Introduction to Microchip Technology Inc.
- Mechatronics examples and benefits
- PIC® Microcontroller Basics
  - Hands-On Learning Cycles
- Labs
  - Simple I/O and Timer 0
  - Reading an analog sensor, LCD module
  - Controlling the speed of a motor
- Resources

Agenda

- **Introduction to Microchip**
- Mechatronics examples and benefits
- PIC® Microcontroller Basics
  - Hands-On Learning Cycles
- Labs
  - Simple I/O and Timer 0
  - Reading an analog sensor, LCD module
  - Controlling the speed of a motor
- Resources

Microchip Technology Inc.

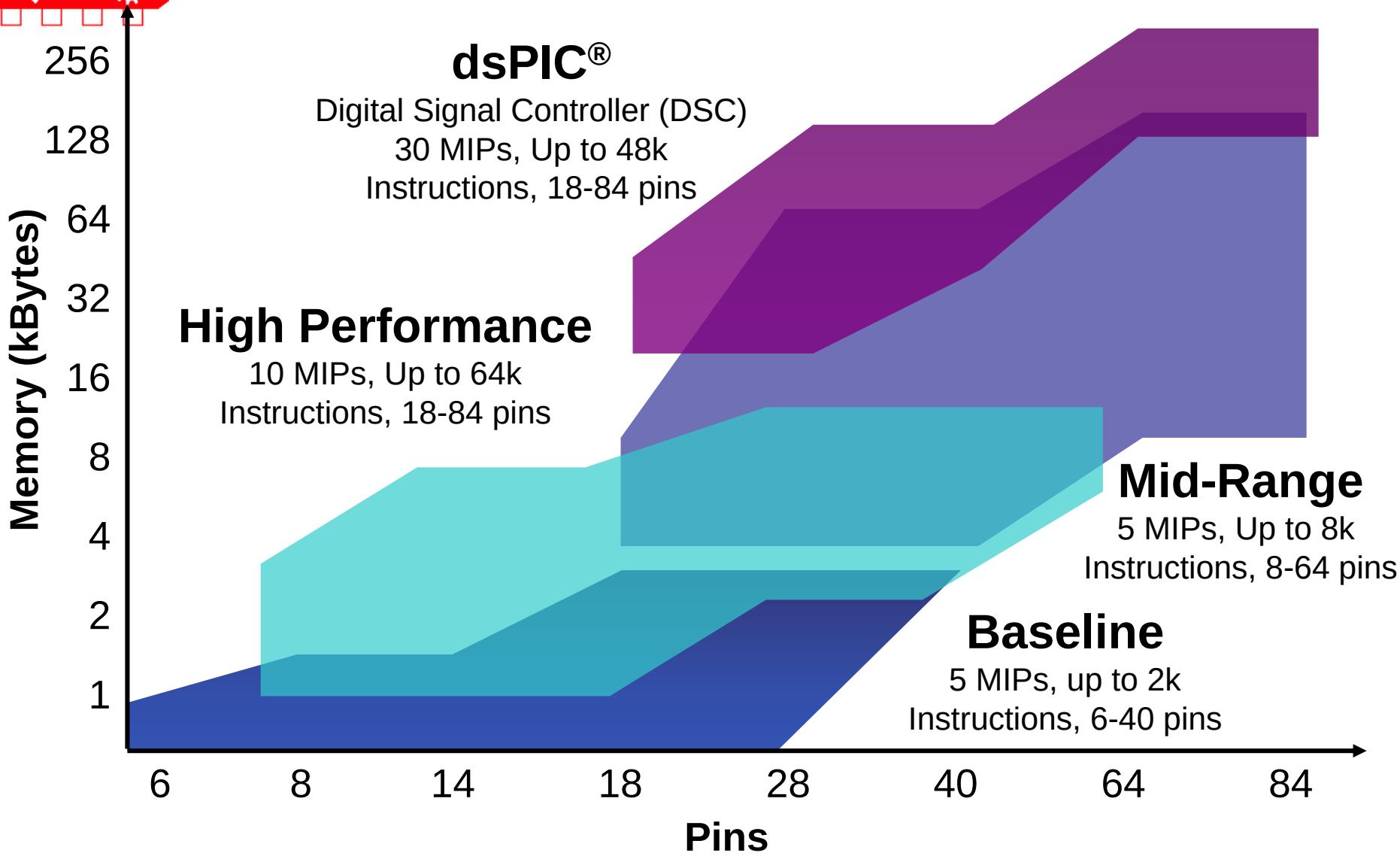
- Leading semiconductor manufacturer:
  - of high-performance, field-programmable, 8-bit & 16-bit RISC Microcontrollers
  - of Analog & Interface products
  - of related Memory products
  - for high-volume Embedded Control applications
- $847M in net sales in FY05
- More than 3,800 employees
- #1 Unit shipments of 8-bit Microcontrollers*

* Gartner Dataquest, 2003 Microcontroller Market Share & Unit Shipments, Tom Starnes, June 2003.

Microchip Delivers Solutions

- Low-risk product development
- Lower total system cost
- Faster time to market
- Dependable delivery
- High-quality devices
- Outstanding support through all phases

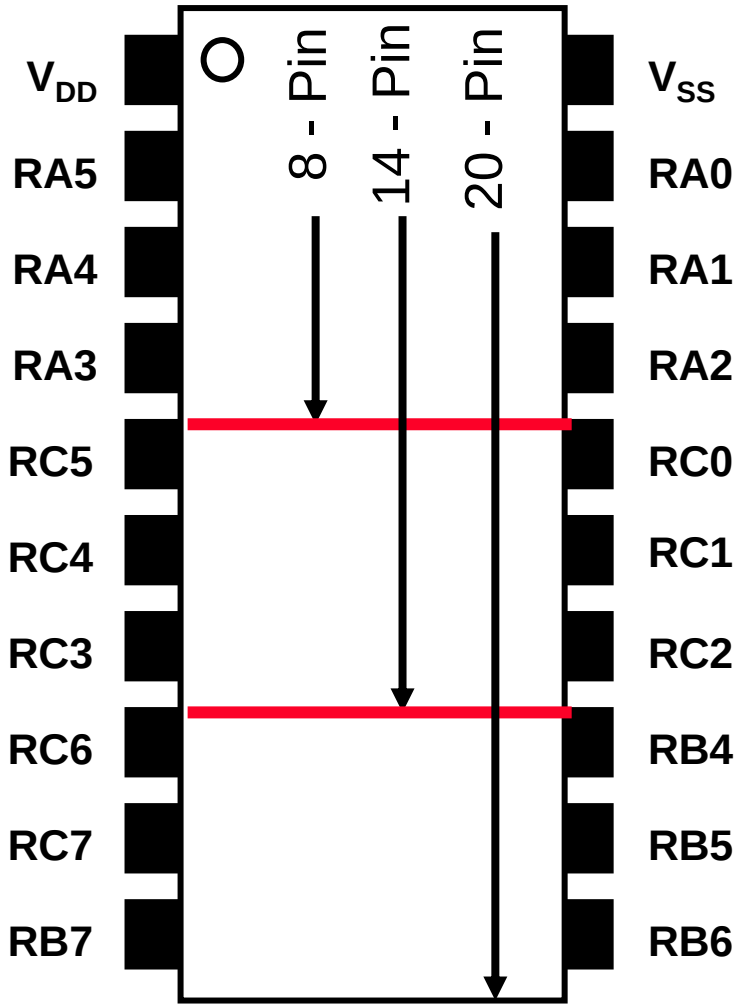
The PIC® Microcontroller Family



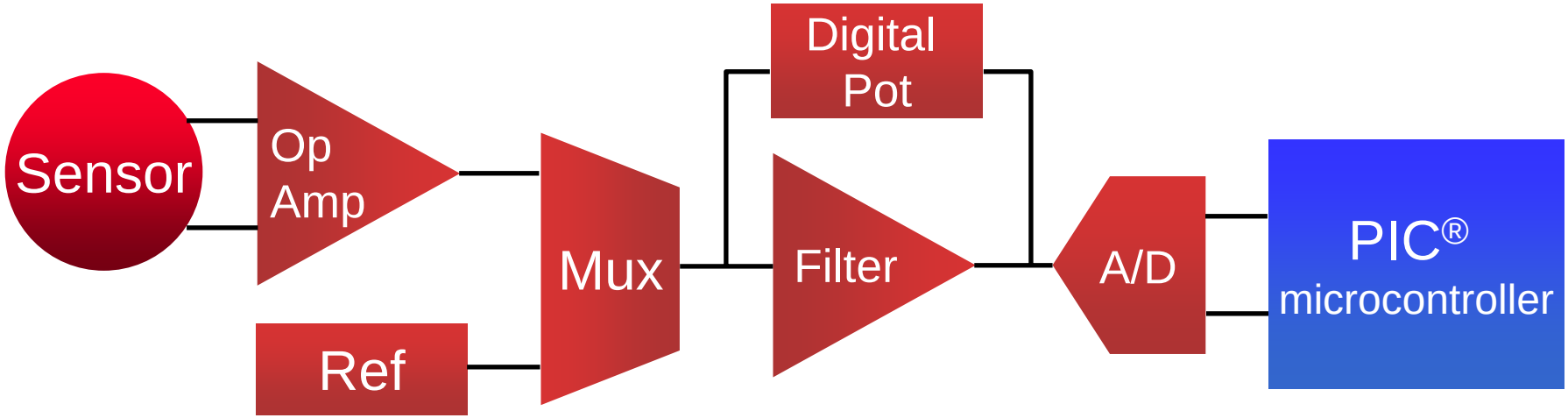
Migration between Different Products Example: Mid-Range Parts

- Pin Compatible within specific pinouts
- Code Compatible
- Peripheral Consistency

Seamless Migration across more than 200 Products



Complete Signal Chain Solutions from Microchip



- ***Performance:*** Low Power, High Precision
- ***Low Cost:*** System Cost, Low Component count
- ***Size:*** Small Packaging – SC70, SOT23, DFN

Analog & Interface Products

Thermal Management

- Temperature Sensors
- Fan Speed Controllers Fan Fault Detectors

Linear Products

- Single Supply CMOS Op Amps
- Comparators
- Linear Integrated Devices
- Programmable Gain Amplifiers

Power Management

- Linear Regulators
- Switching Regulators/Controllers
- Charge Pump DC/DC Converters
- Voltage References
- CPU/System Supervisors
- Voltage Detectors
- Power MOSFET Drivers
- Battery Management
- PWM Controller

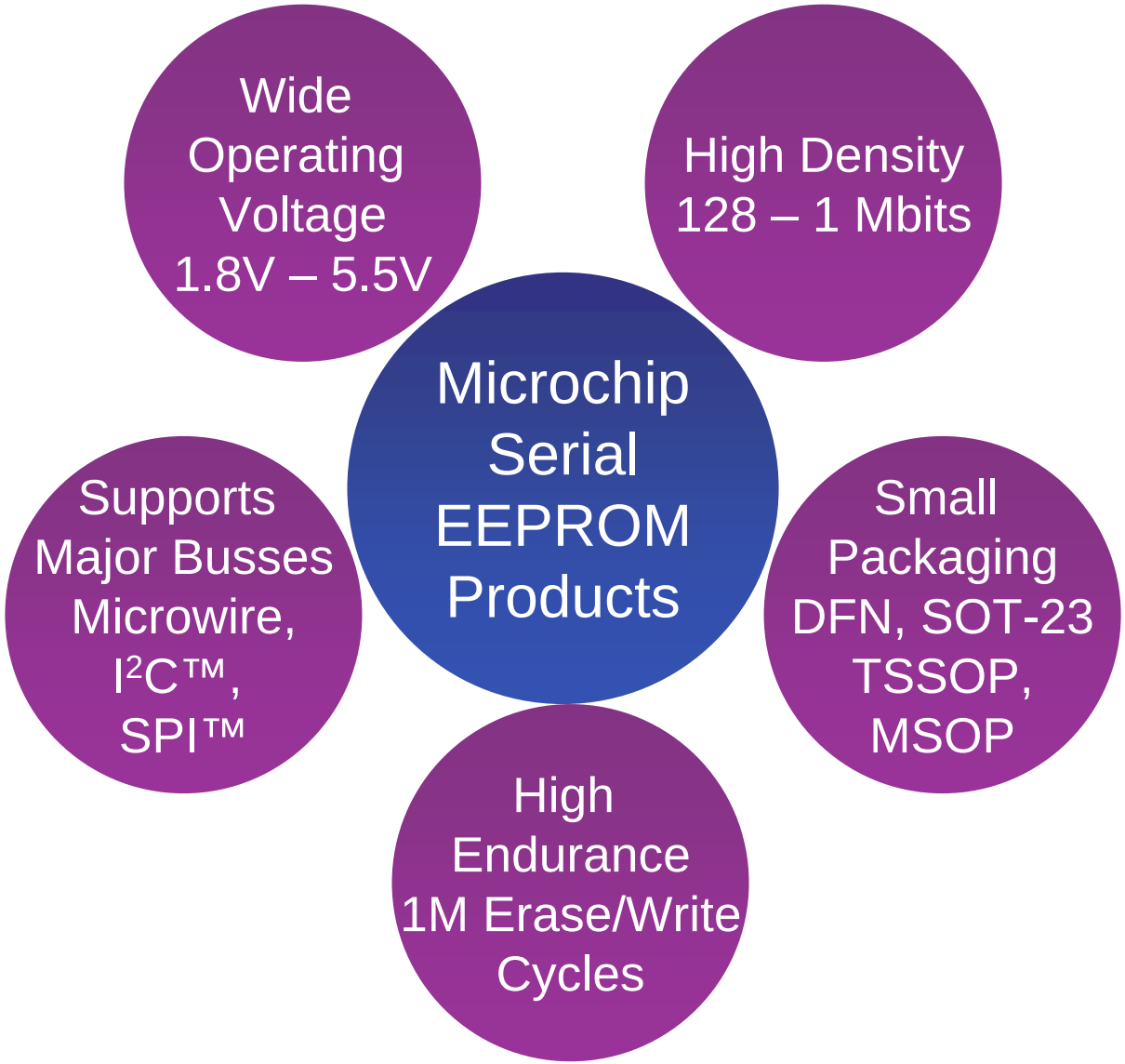
Mixed Signal

- SAR/Delta-Sigma A/D Converters
- Dual Slope A/D Converters
- Display A/D Converters
- System D/A Converters
- V/F and F/V Converters
- Digital Potentiometers

Interface Products

- CAN Peripherals
- Infrared Peripherals
- LIN Transceivers
- Serial Peripherals

Overview of Memory Products



Proven Quality Record across all Product Families

- ISO9001 Certification
- QS9000 Certification
- Quality Awards from numerous Fortune 100 customers
- PPM levels consistently below 10
- Field Failures for PIC® microcontrollers virtually non-existent

Agenda

- Introduction to Microchip
- **Mechatronics examples and benefits**
- PIC® Microcontroller Basics
  - Hands-On Learning Cycles
- Labs
  - Simple I/O and Timer 0
  - Reading an analog sensor, LCD module
  - Controlling the speed of a motor
- Resources

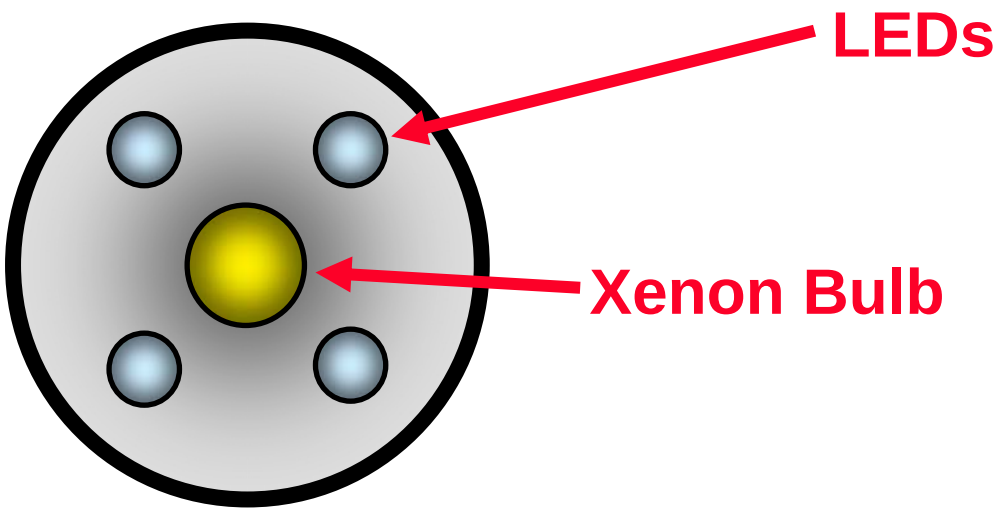
Mechatronics is:

- *Implementing* electronic controls in a mechanical system
- *Enhancing* existing mechanical designs with intelligent controls
- *Replacing* mechanical components with an electronic solution

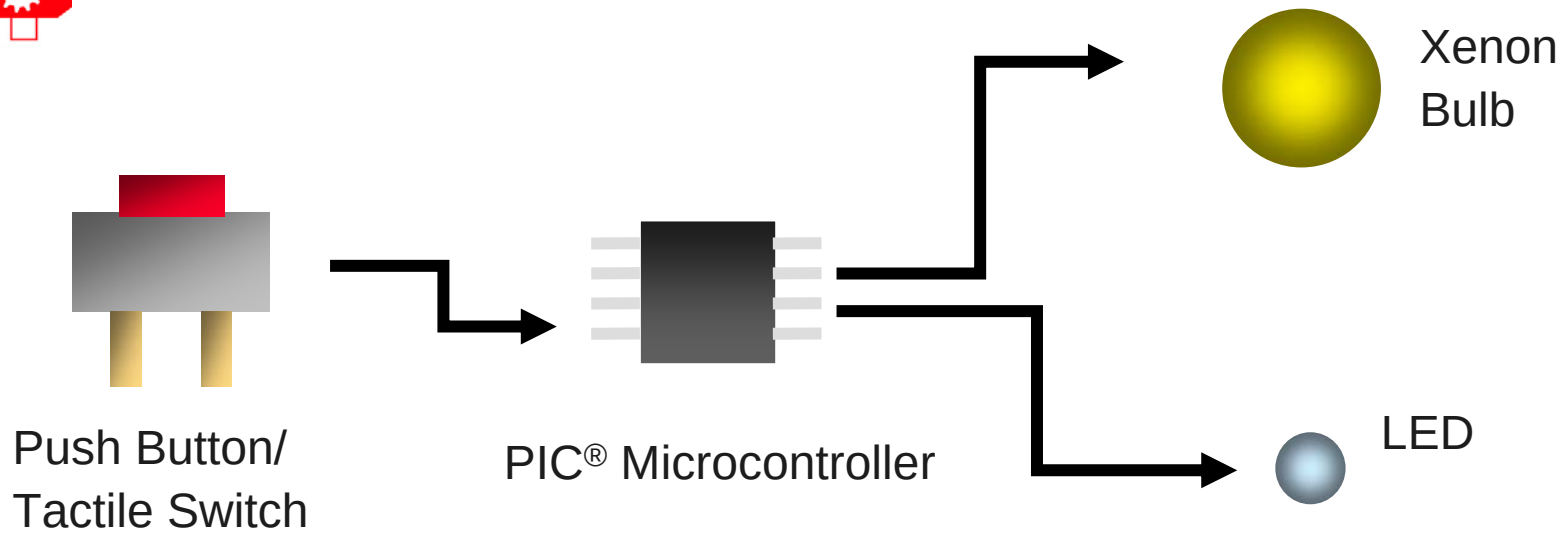
A perfect system for a PIC® microcontroller!!

Example 1: Flashlight Switch

- Traditional Flashlight: On or Off
- Hybrid Flashlights: Mode selector



Intelligent Switch



Modes:

- Xenon bulb on
- LEDs on (dim)
- LEDs on (bright)
- Strobe LEDs
- Flashlight off

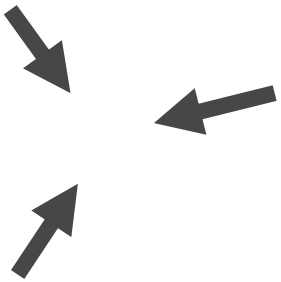
Switch Comparison

	Mechanical	Intelligent
Hybrid Control	No	Yes
Current Control	No	Yes
Programmable	No	Yes
Low Battery Detect	No	Yes
Strobe Capable	No	Yes
Adaptable	**NO!**	**YES!**

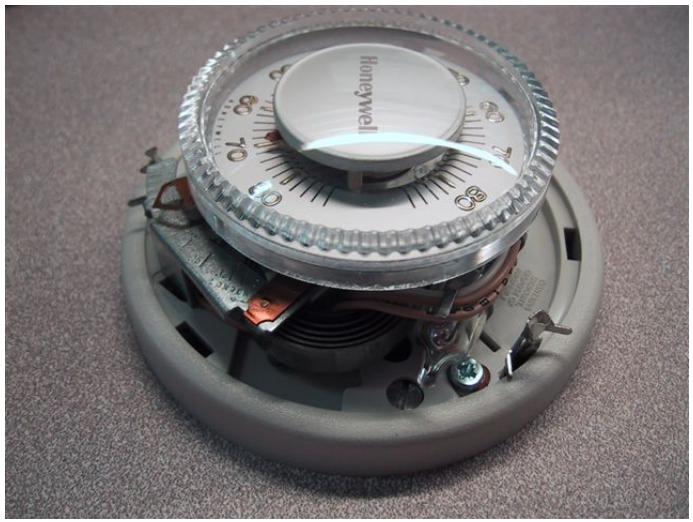
Possible Solution: PIC10F20x

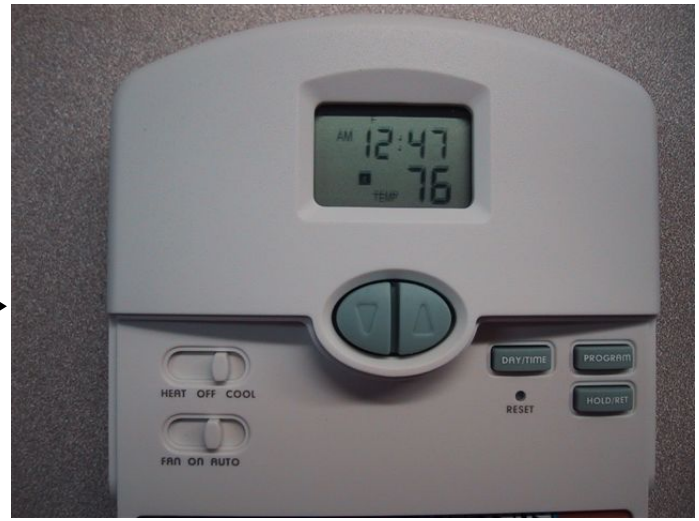
- SOT-23 package
- 6 pins:
  - 3 Input/Output pins
  - 1 Input only
  - 1 Power, 1 Ground
- 4 MHz internal oscillator
  - ± 2% accuracy over voltage and temperature
- one analog comparator
  - internal voltage reference
- Ultra-low sleep current
- 8-bit timer
- 2V to 5V operating range

Actual Size

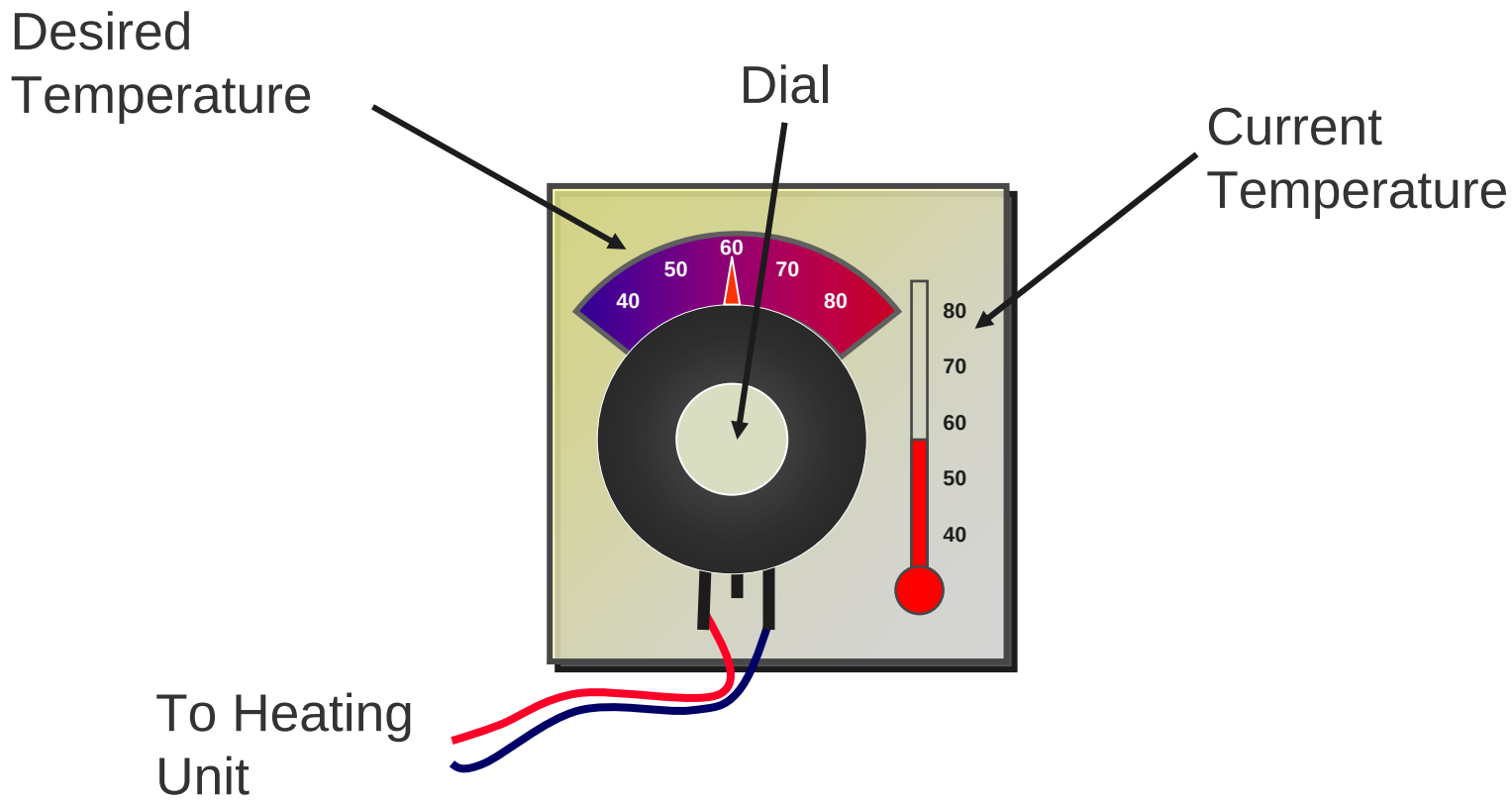


Example 2: Household Thermostat

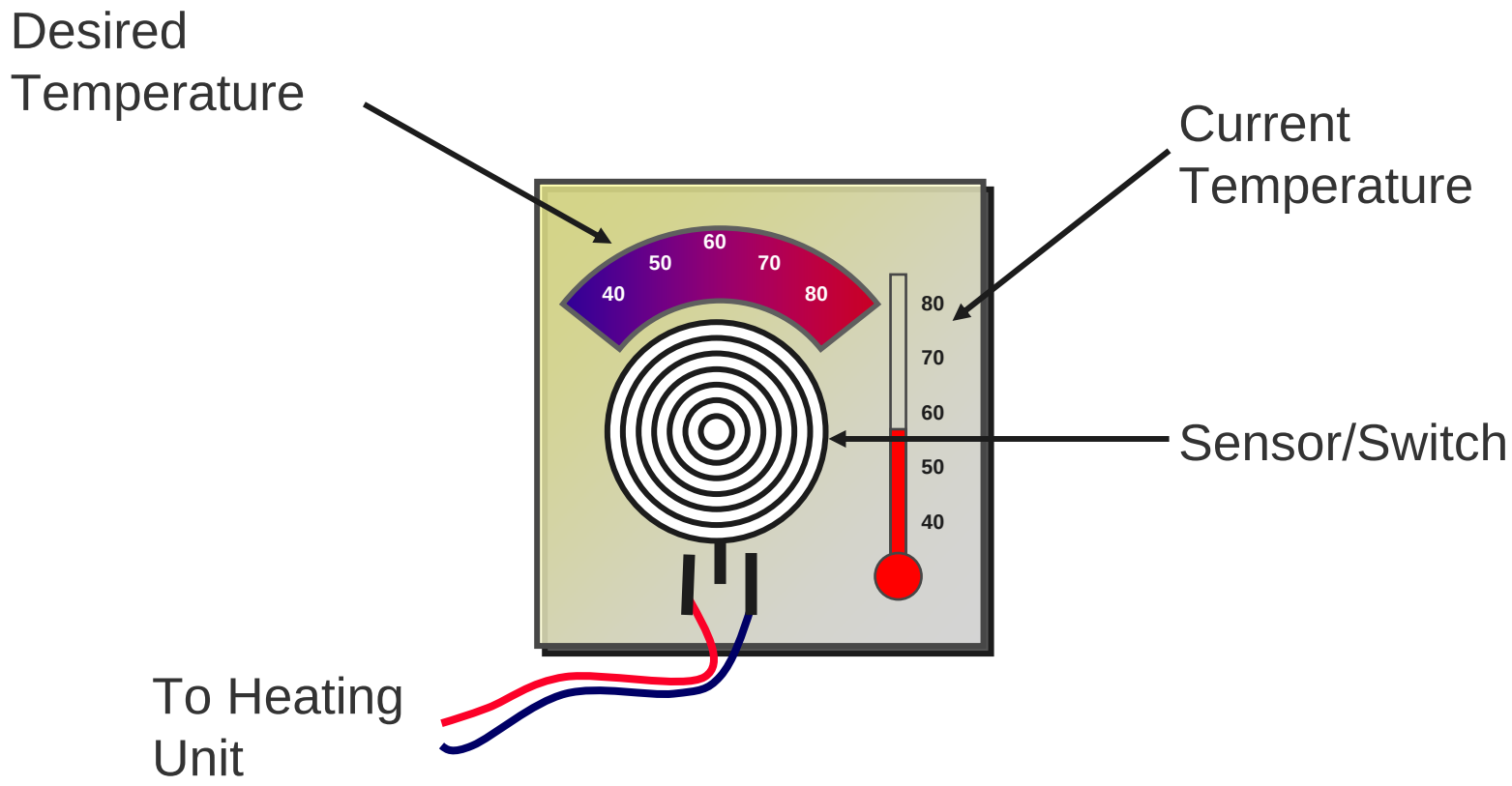




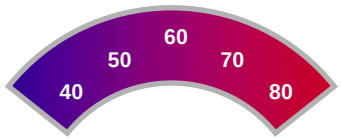
Example Application: Thermostat



Example Application: Thermostat

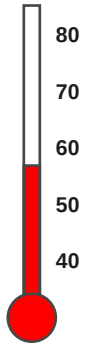


Breakdown of Thermostat

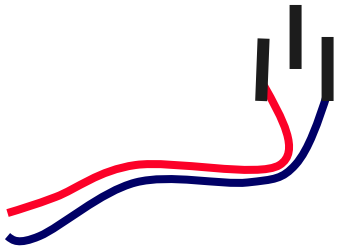


User Feedback

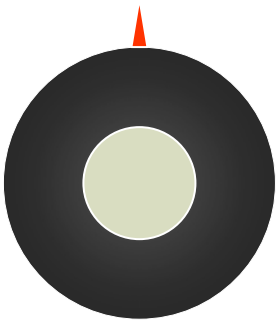
Temperature Sensor



User Feedback



Switch to turn on Heater



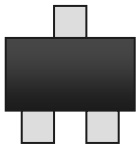
User Input

Conversion to Mechatronic Design

User Feedback

Tactile Switches

User Input

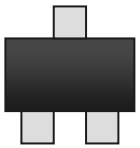


Small IC

Temperature Sensor

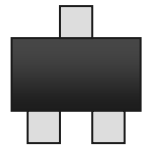
PIC® Microcontroller

Control



MOSFET

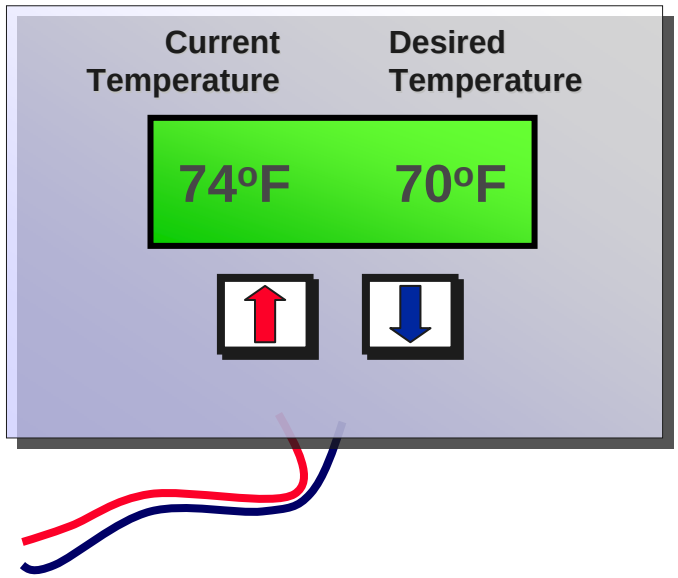
Switch to turn on Heater



Regulator

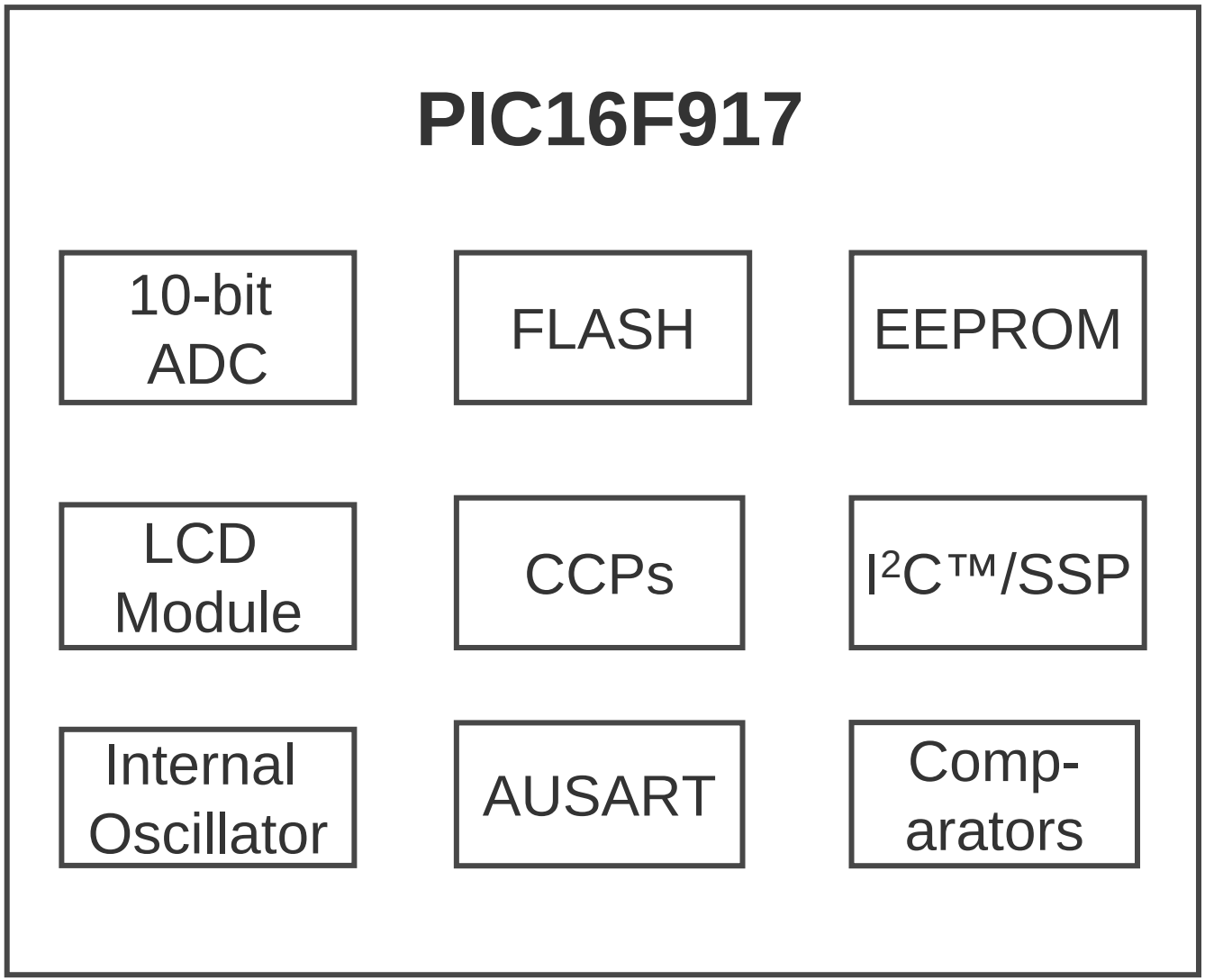
Power

Finished Design

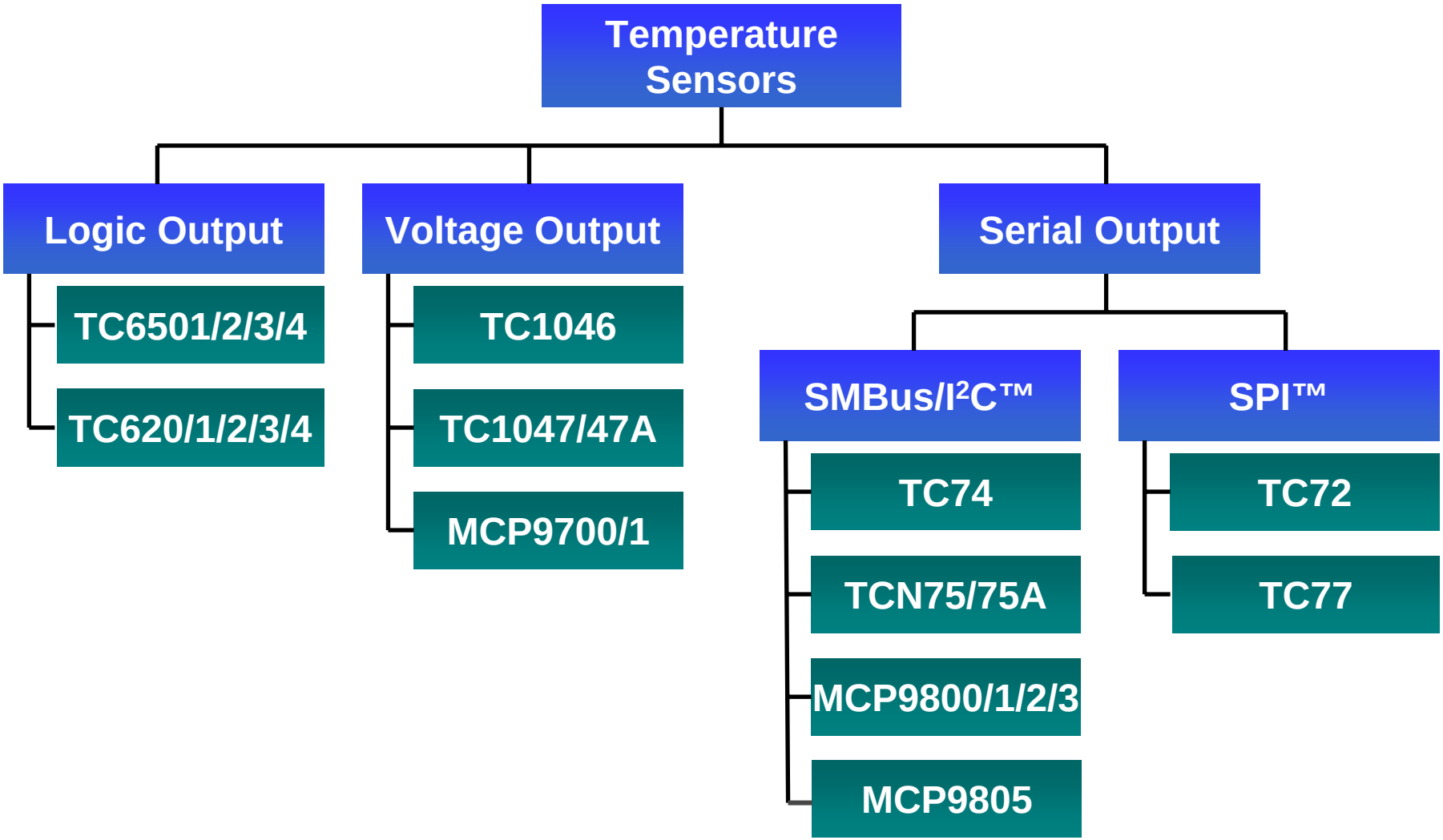


- Higher resolution and accuracy
- Reduces household heating costs
- Self calibrating
- Flexible Design
- Environmentally Friendly

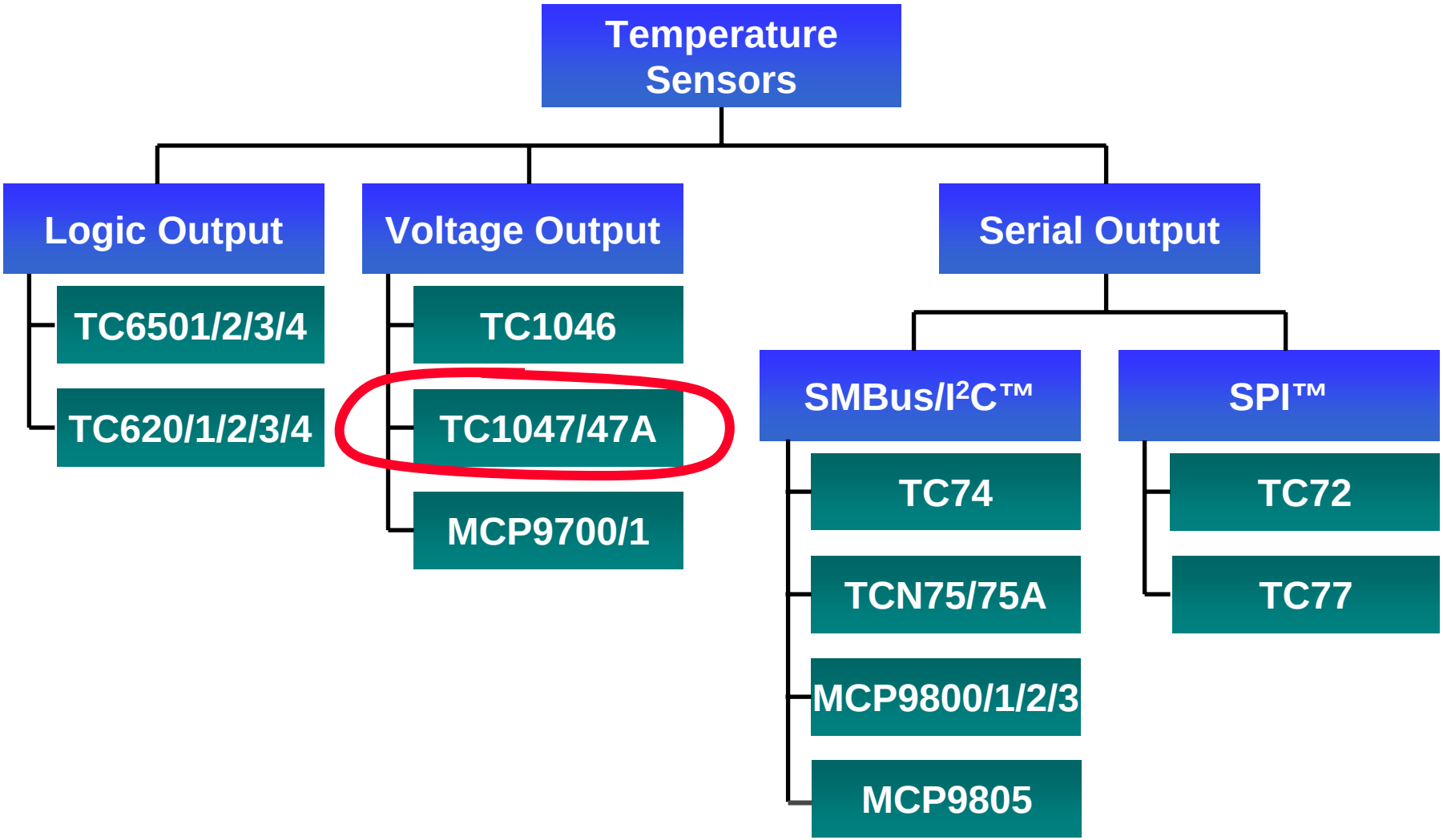
Possible Solution: PIC16F917



Temperature Sensing Options from Microchip

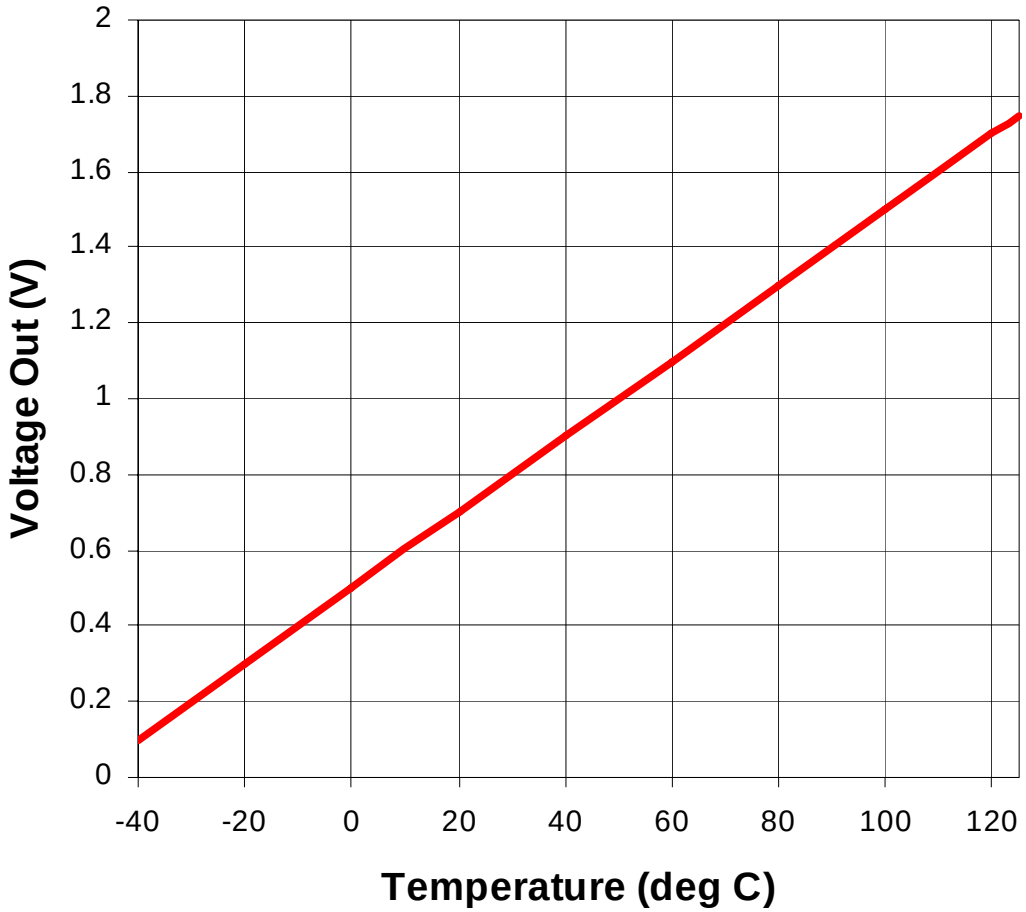


Temperature Sensing Options from Microchip

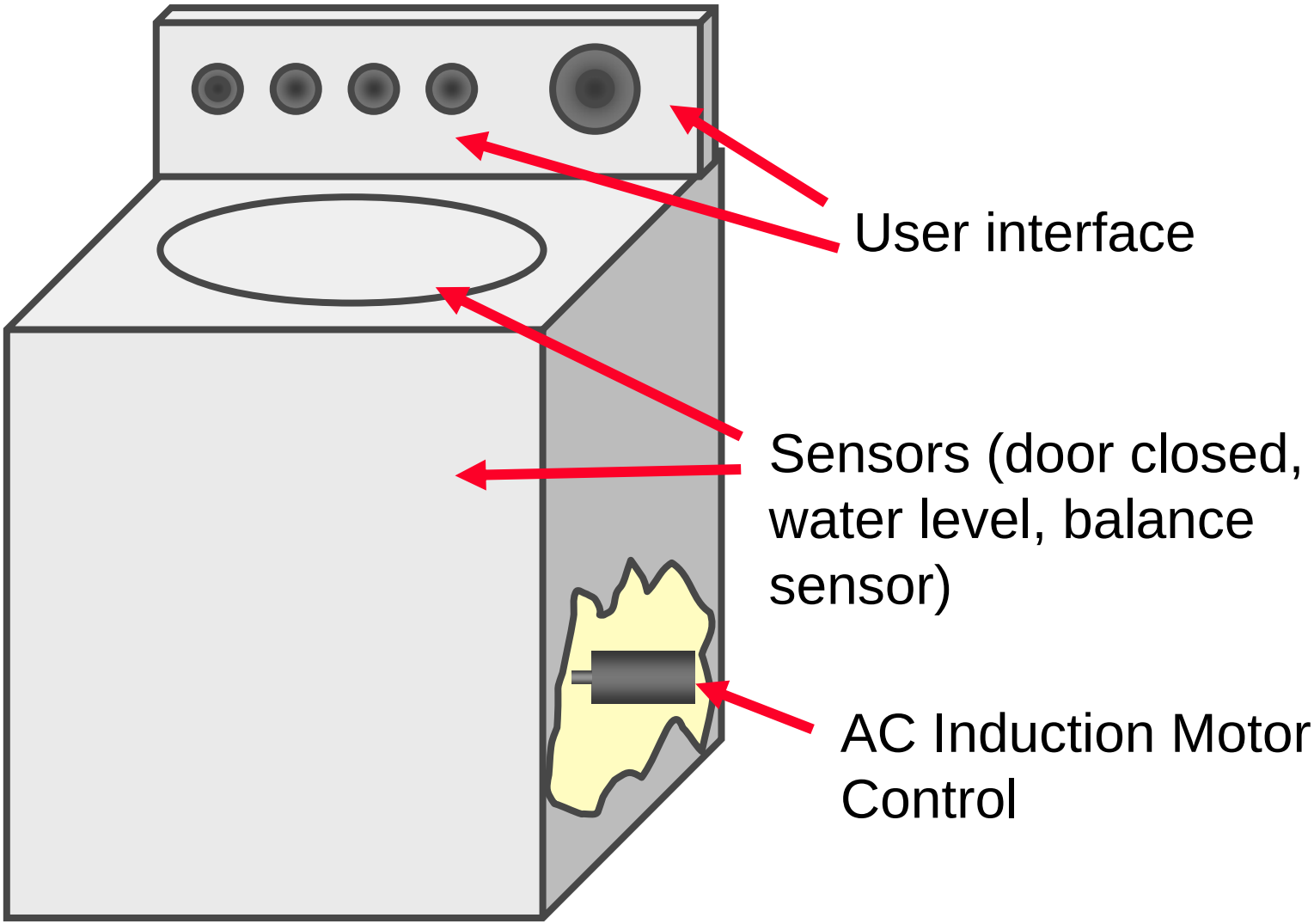


Linear Output
TC1047A Temperature Sensor

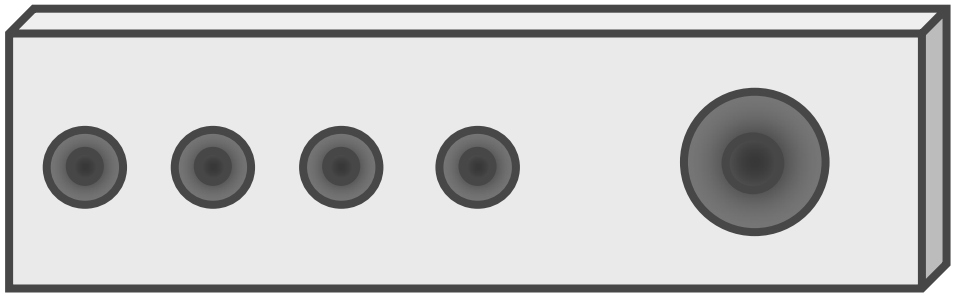
- Linear Slope
  - 10 mV/ºC
- -40 to 125ºC Temperature Range
- ±2ºC Accurate
- Low Current Consumption
- Small Package
  - SOT-23



Example 3: Washing Machine



User Interface Improvements



- Use standard dials and knobs
- Same knobs used on different models
- Reprogramming microcontroller gives knobs different functionality
- Switches less prone to wear compared to mechanical washing machine

Benefits of Modern Sensors

- Turbidity sensor*: How much dirt is in the water?
- Door position sensor**: Is the door ajar?
- Balance sensor**: Is the machine unbalanced?
- Water Level sensor*: Determined by size of load.

Benefits:

* More efficient washing
  - Energy efficient
  - Water efficient

** Make washing safer (for people and the machine)

AC Induction Motor Control

Benefits to Microcontroller-based speed control:

- Smaller motor
- Standardized motor
- Higher efficiency
- Consistency over varying load conditions
- Precision speed-control (more speeds possible)

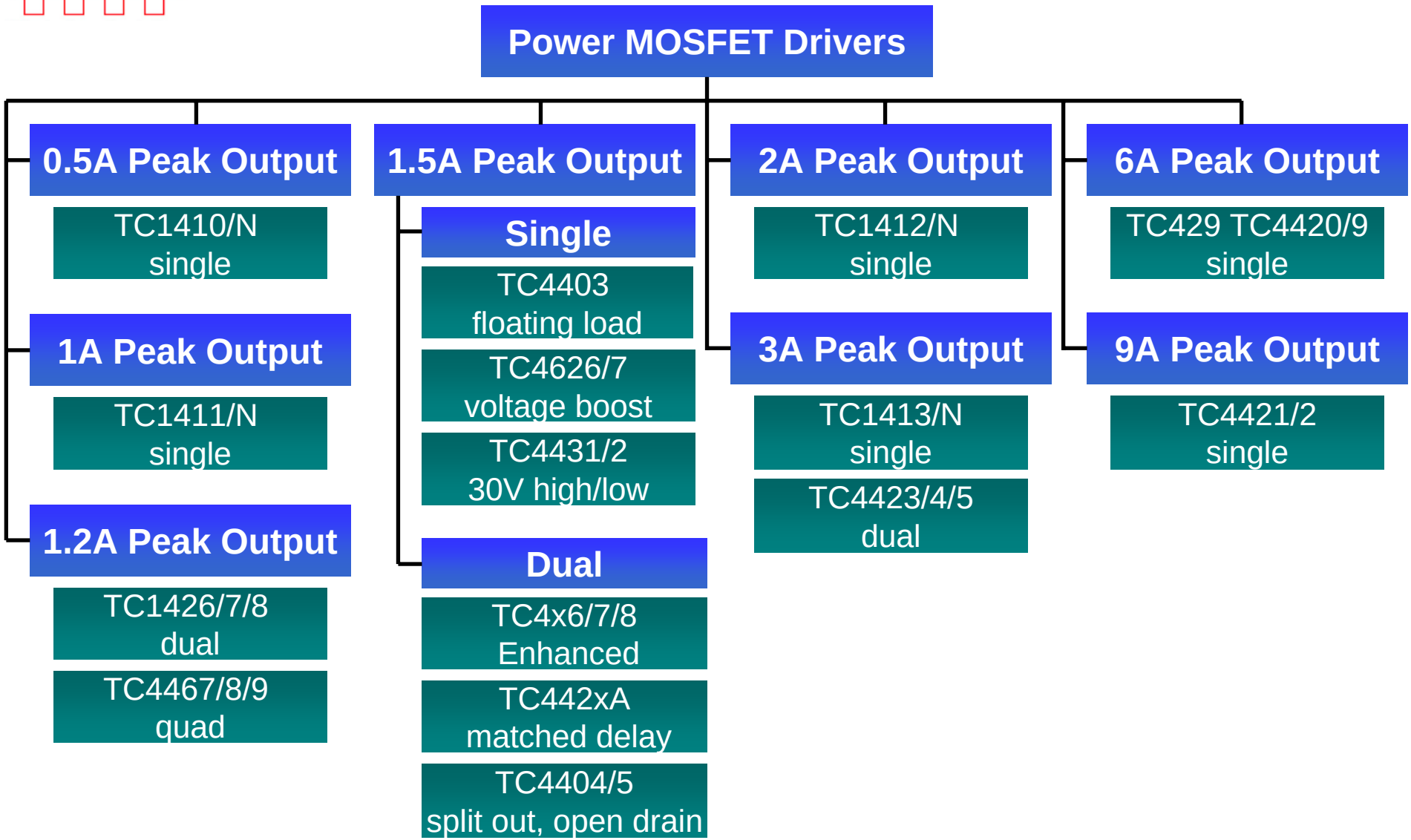
Possible Solution: PIC18F4431

- PWM Output: 8 independent channels
  - Up to 14-bit resolution
  - Edge and center-aligned
  - Programmable dead-time
- 3-ch Quadrature Encoder Interface
- 10-bit High-Speed A/D Converter

Created from the ground up

for precision motor control!

Microchip MOSFET Driver Solutions



Mechatronics Review

- Mechatronics is _____.
- What are the benefits to mechatronics?
  - How does the consumer benefit?
  - How does the manufacturer benefit?
- What is the world's smallest microcontroller?
- What microcontroller can directly drive a LCD?

Agenda

- Introduction to Microchip
- Mechatronics examples and benefits
- **PIC® Microcontroller Basics**
  - Hands-On Learning Cycles
- Labs
  - Simple I/O and Timer 0
  - Reading an analog sensor, LCD module
  - Controlling the speed of a motor
- Resources

Objectives of PIC® Microcontroller Basics

We will learn how to:

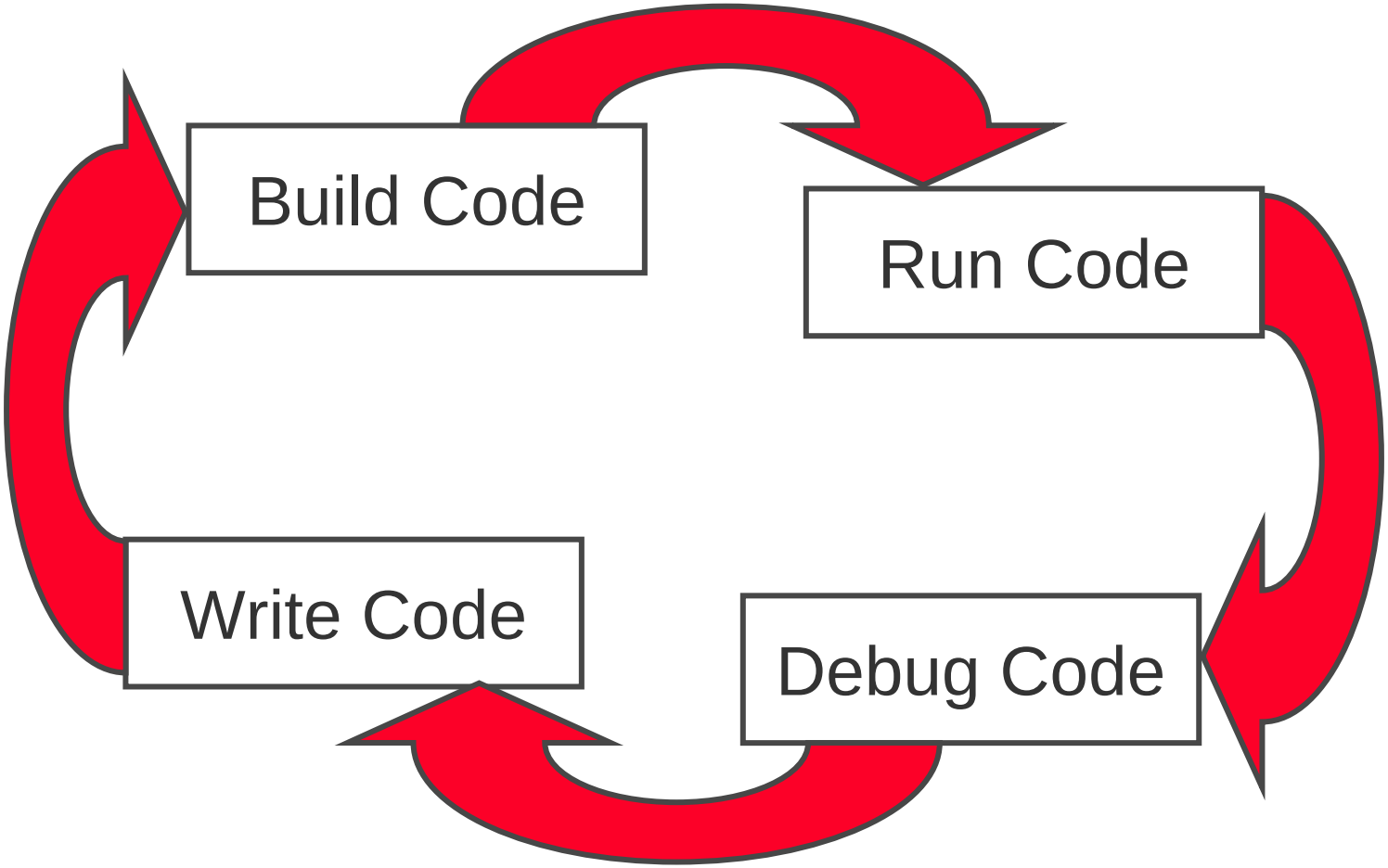
- Create code
- Compile
- Test and debug
- Use MPLAB® Integrated Development Environment (IDE)

. . . all with a focus on learning about PIC microcontroller architecture and terms

What is MPLAB® IDE?

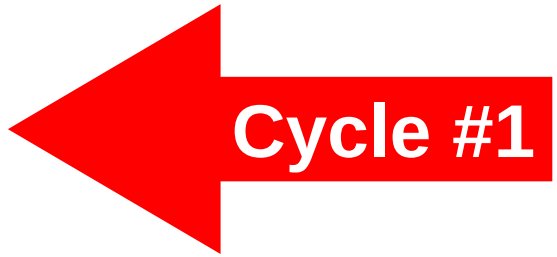
- MPLAB® IDE is the name of Microchip's free **I**ntegrated **D**evelopment **E**nvironment
- Software that runs on your PC
- Complete development environment for your embedded system
- Our basic tool for the rest of this class

Embedded Software Design Cycle

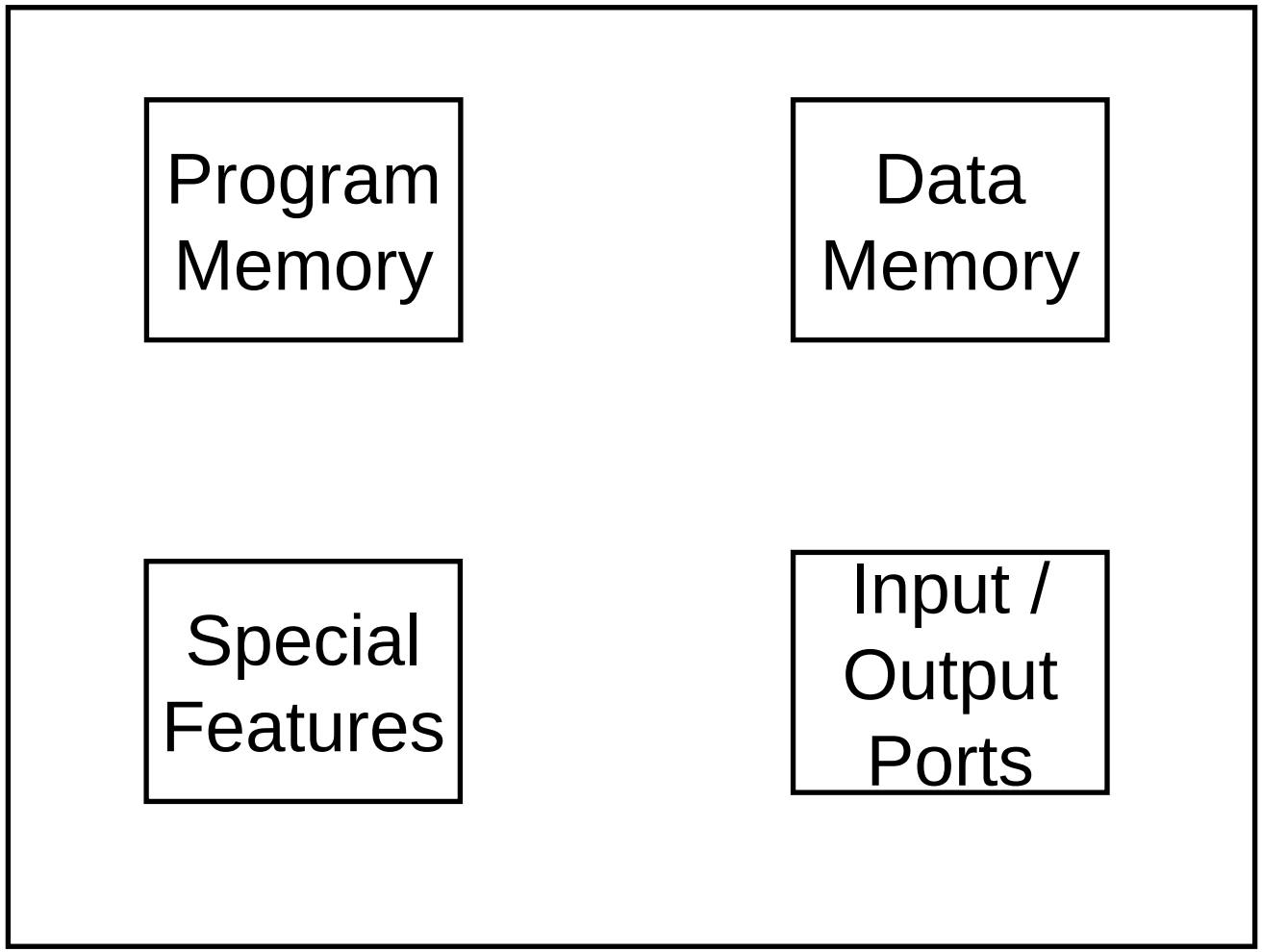


Agenda

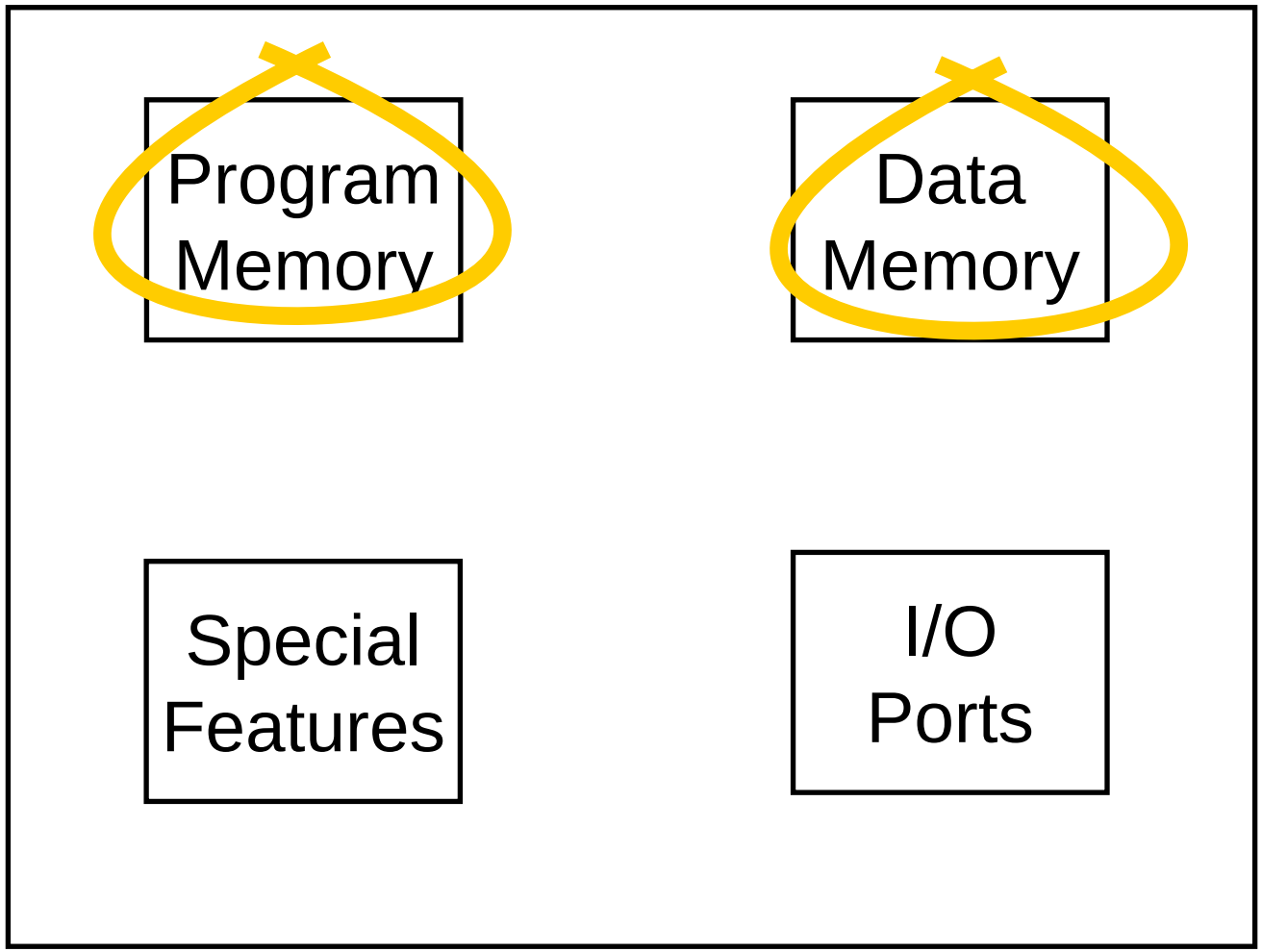
- Introduction to Microchip
- Mechatronics examples and benefits
- PIC® Microcontroller Basics
  - **Hands-On Learning Cycles**
- Labs
  - Simple I/O and Timer 0
  - Reading an analog sensor, LCD module
  - Controlling the speed of a motor
- Resources



Architecture: Overview



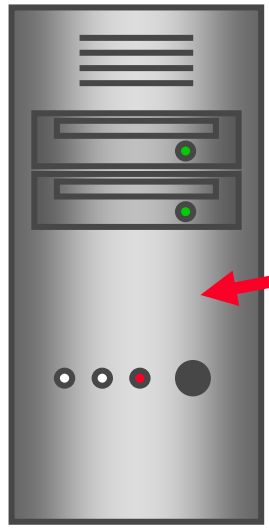
Architecture: Overview



Architecture: PIC16F917

Program Memory

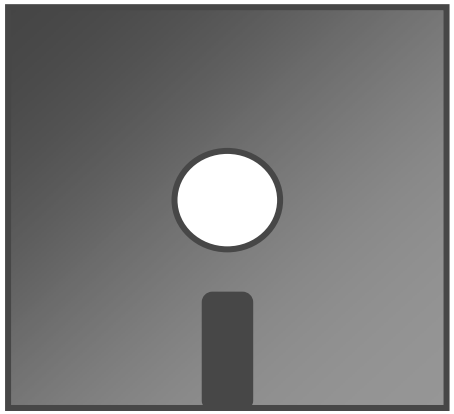
- 14 bits wide
- 8192 words



Hard Drive

Architecture: PIC16F917

Floppy Disk

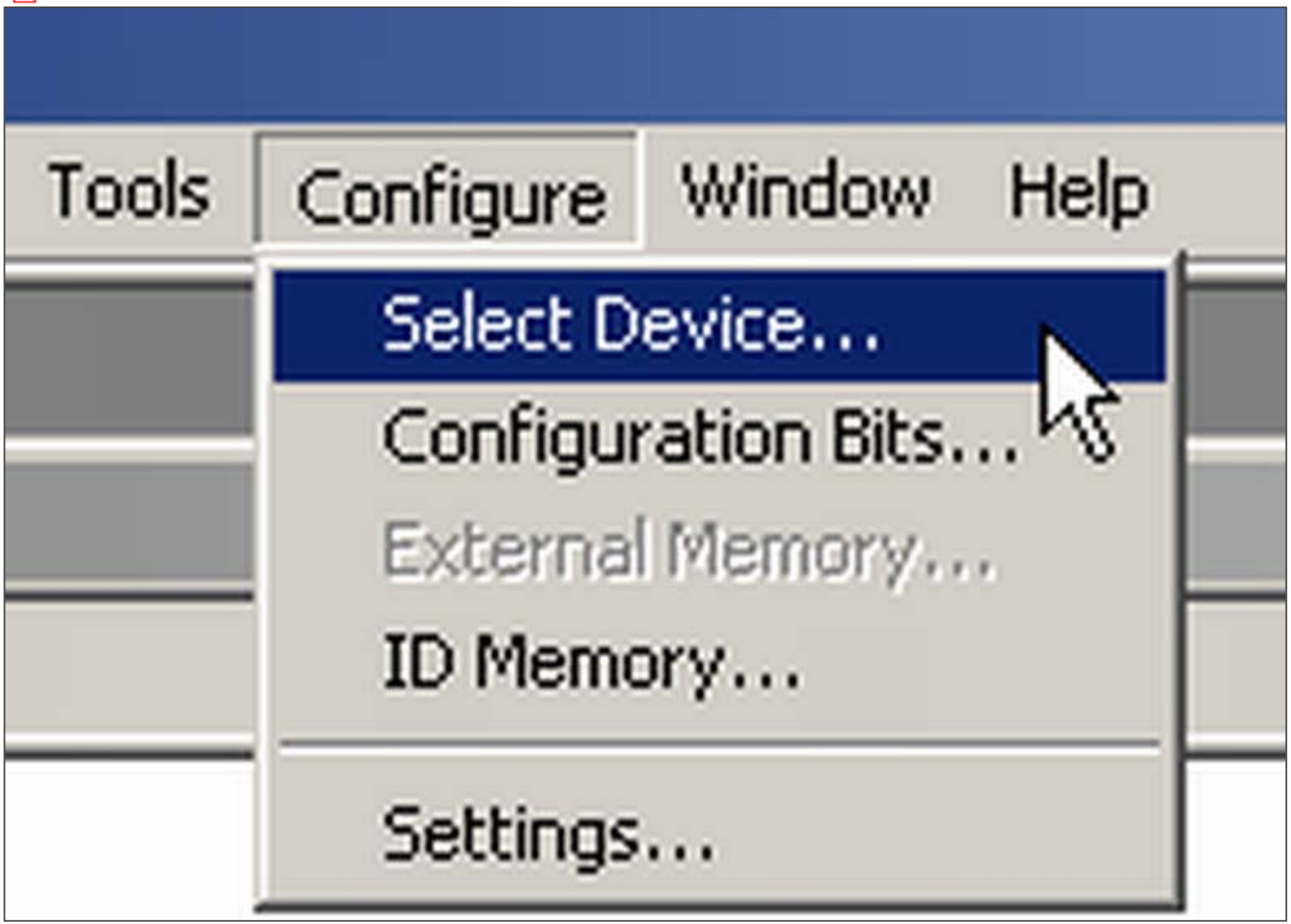


Data Memory

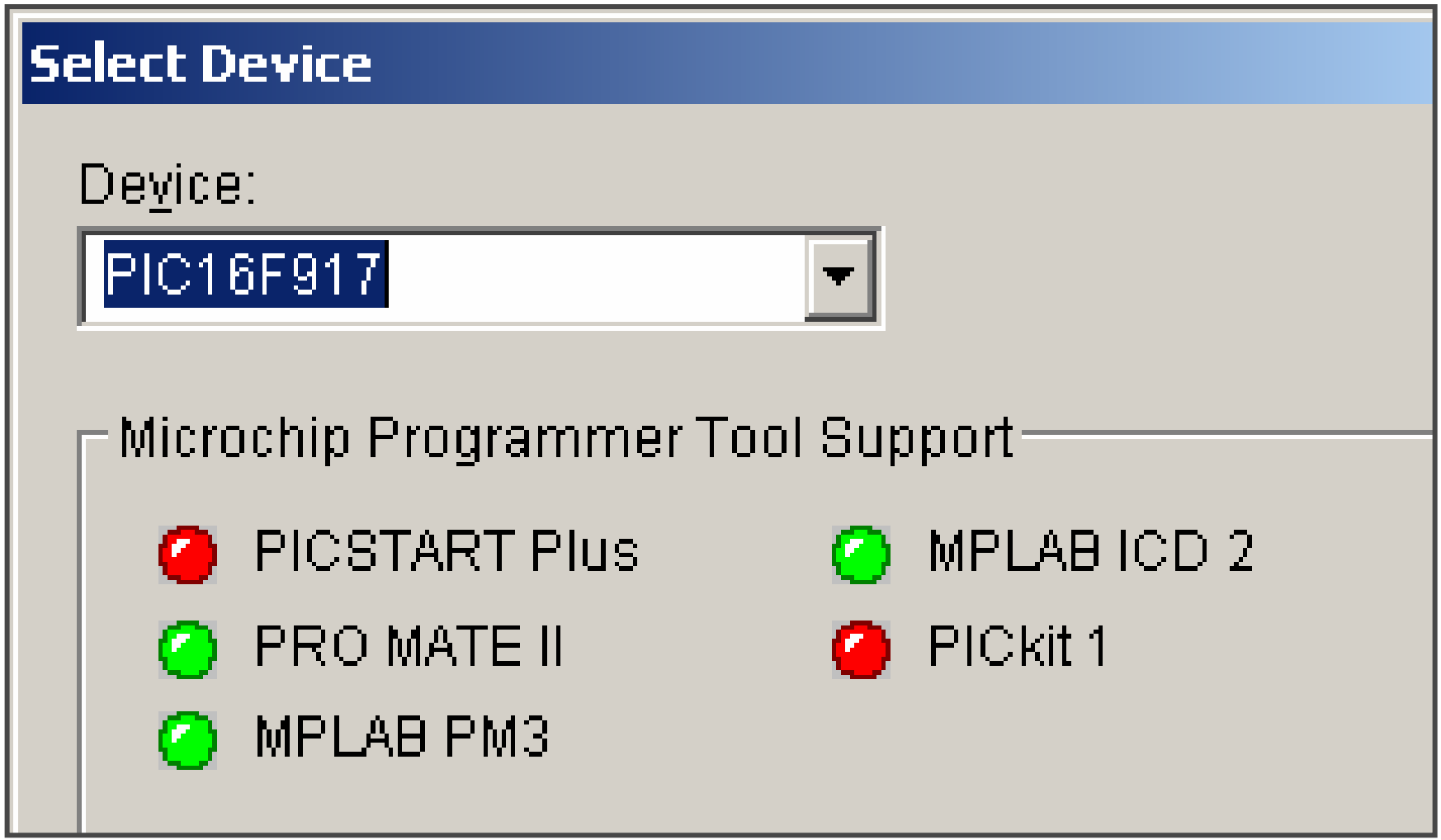
- 8 bits wide
- 512 bytes

Hands-on with MPLAB® IDE

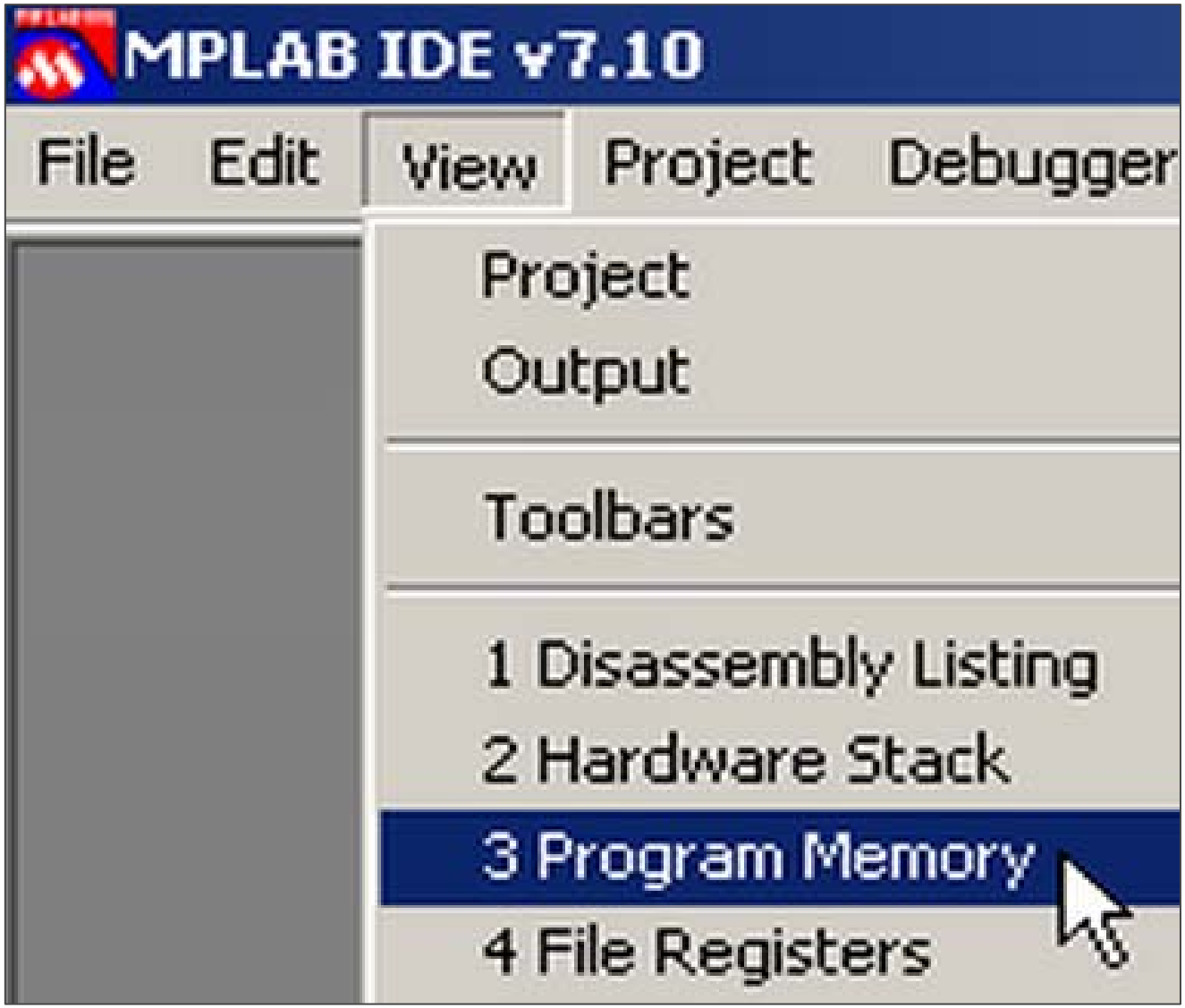
Configure – Select Device



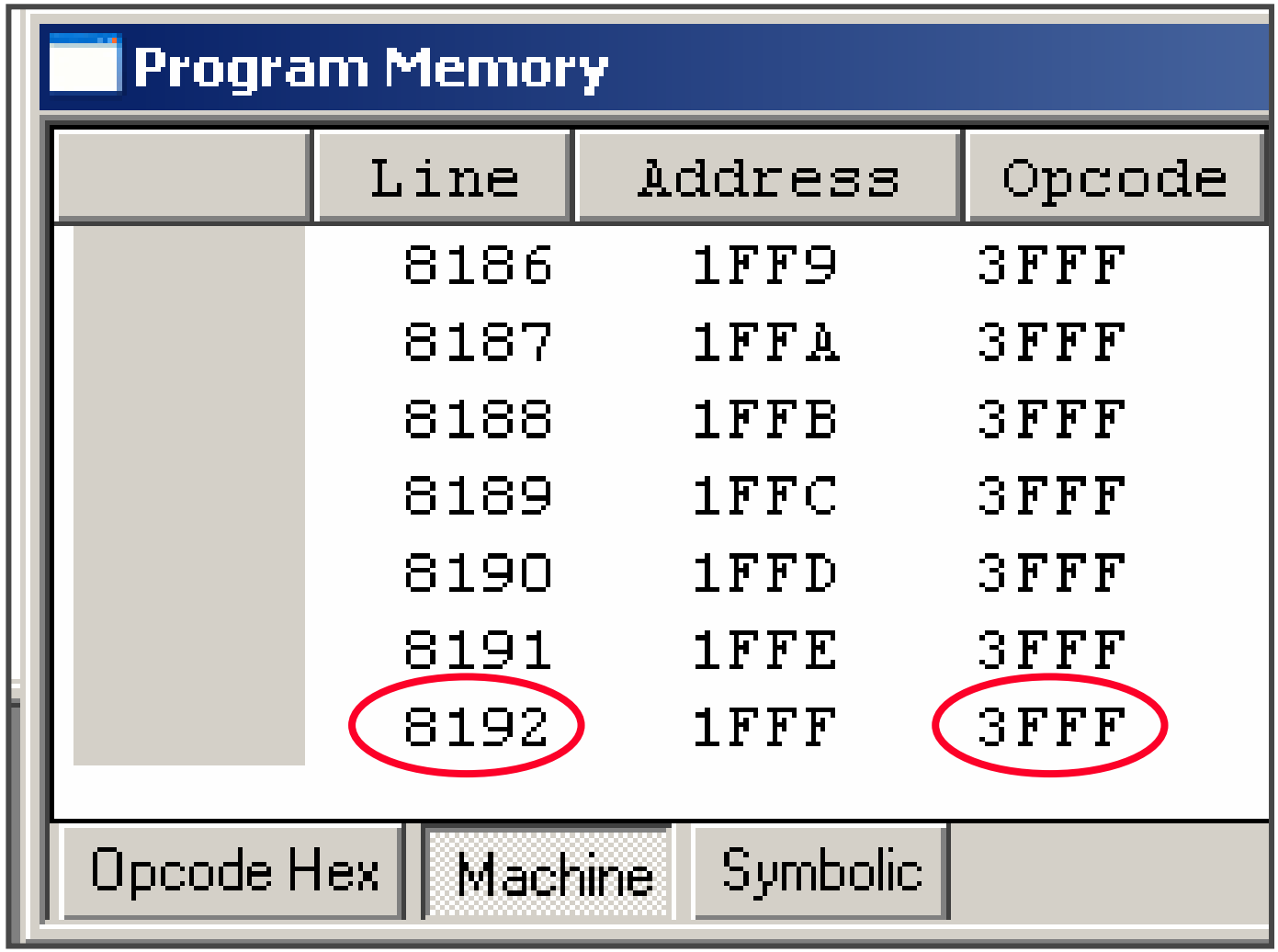
Selecting PIC16F917



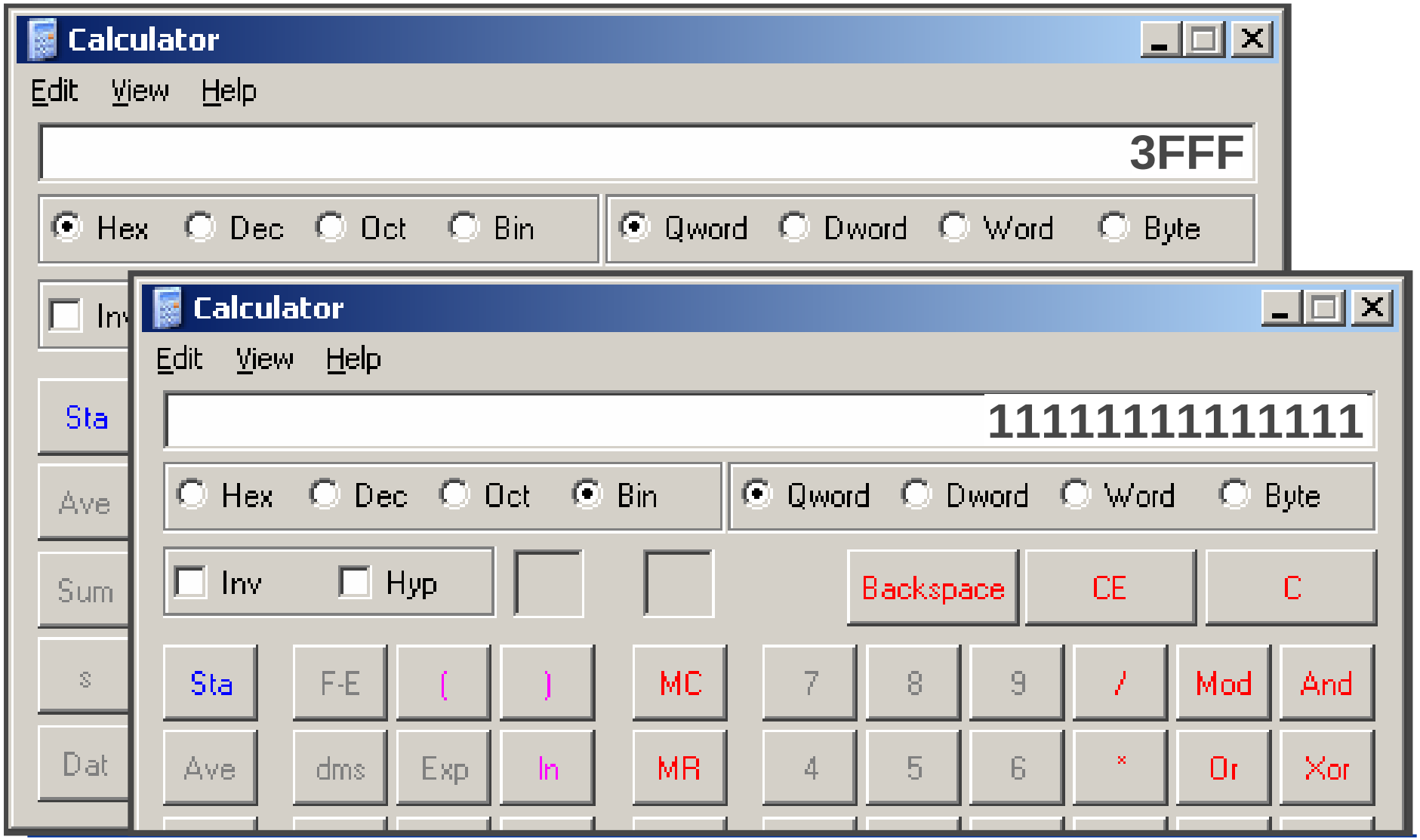
Program Memory View



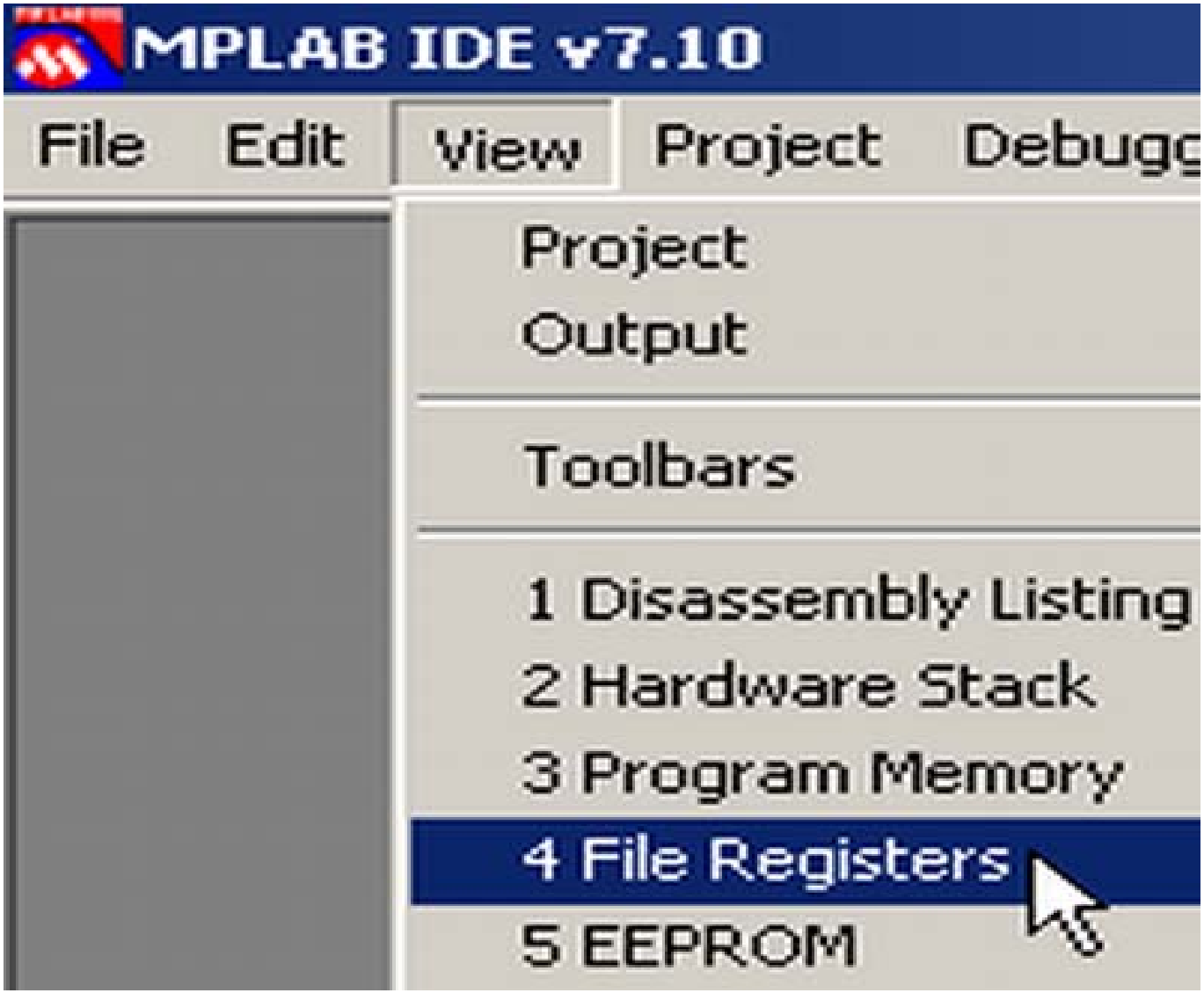
Program Memory View



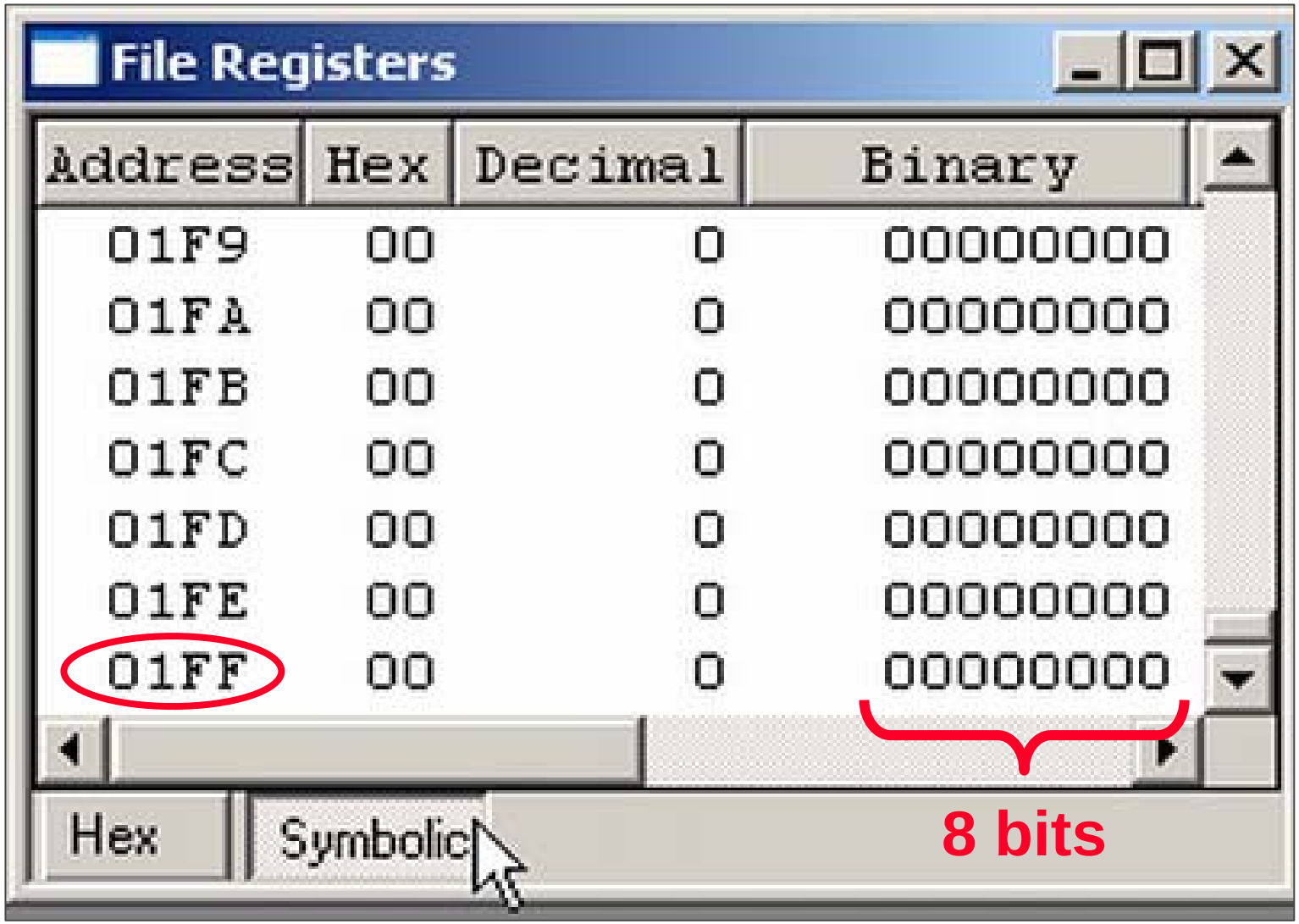
Hexadecimal to Binary



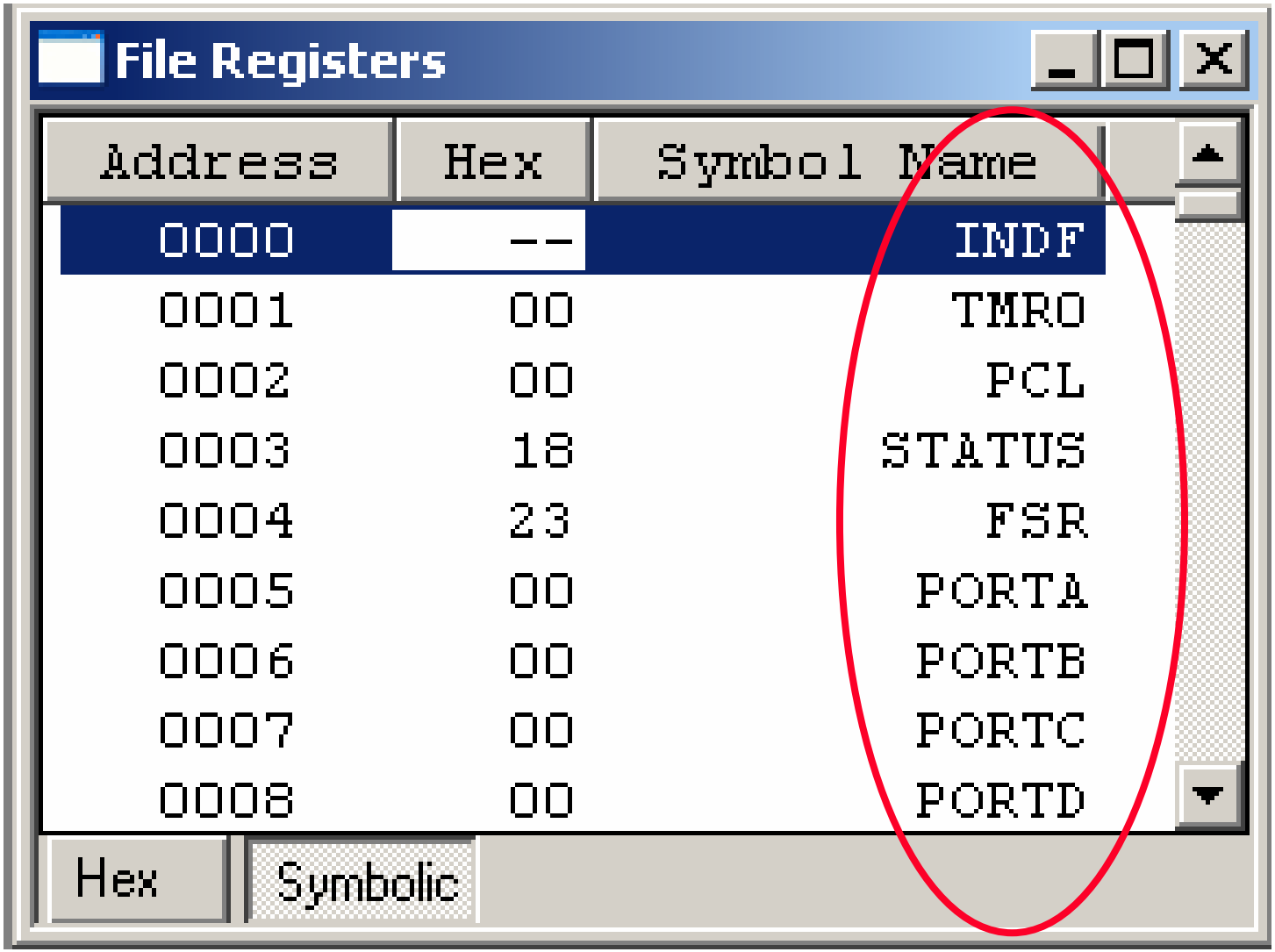
Data Memory (File) View



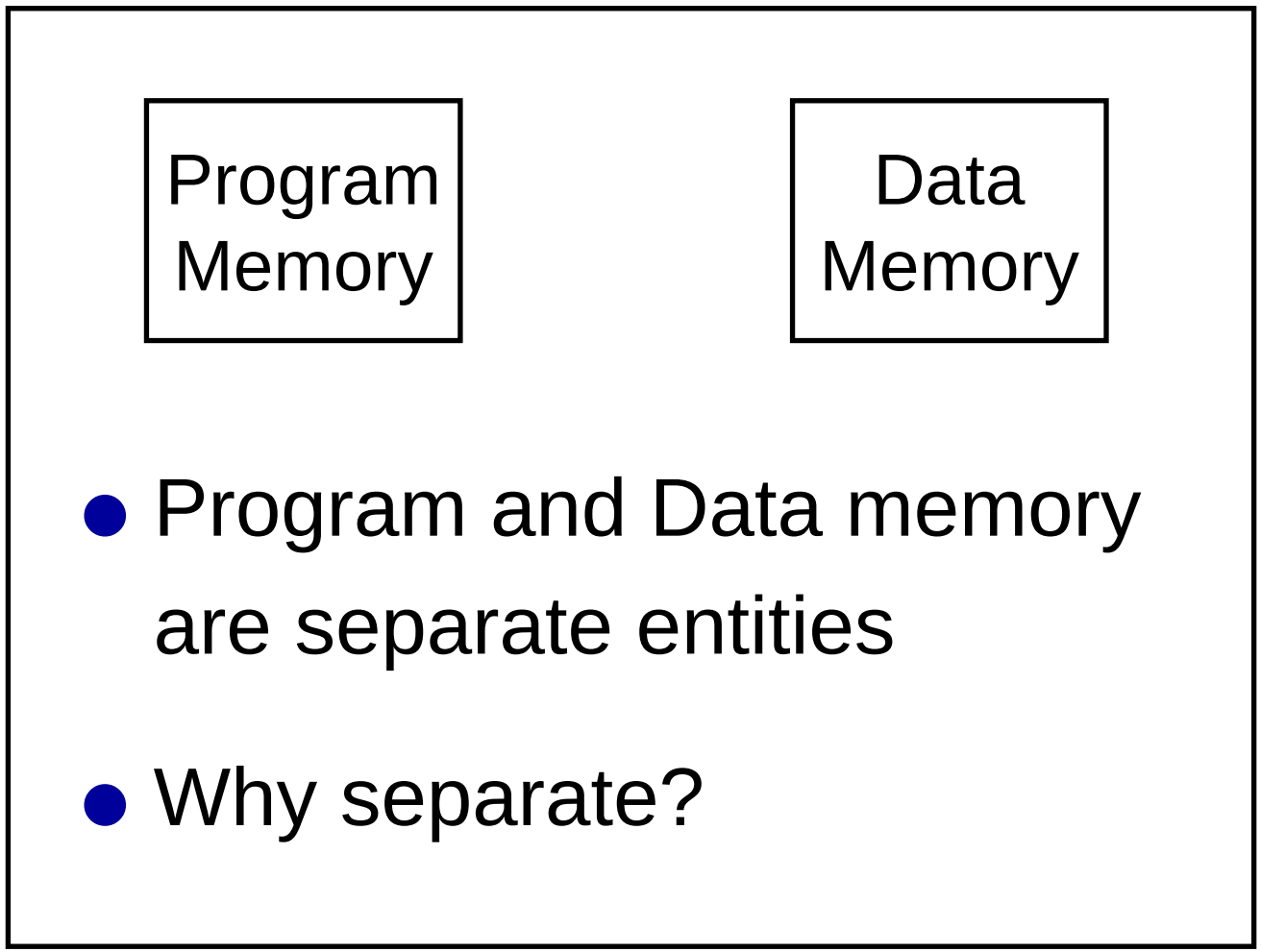
Data Memory (File) View



Special Function Registers



Separate Program & Data



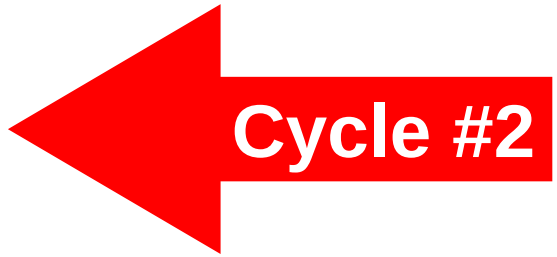
- Program and Data memory are separate entities
- Why separate?

Cycle #1: What Did I Learn?

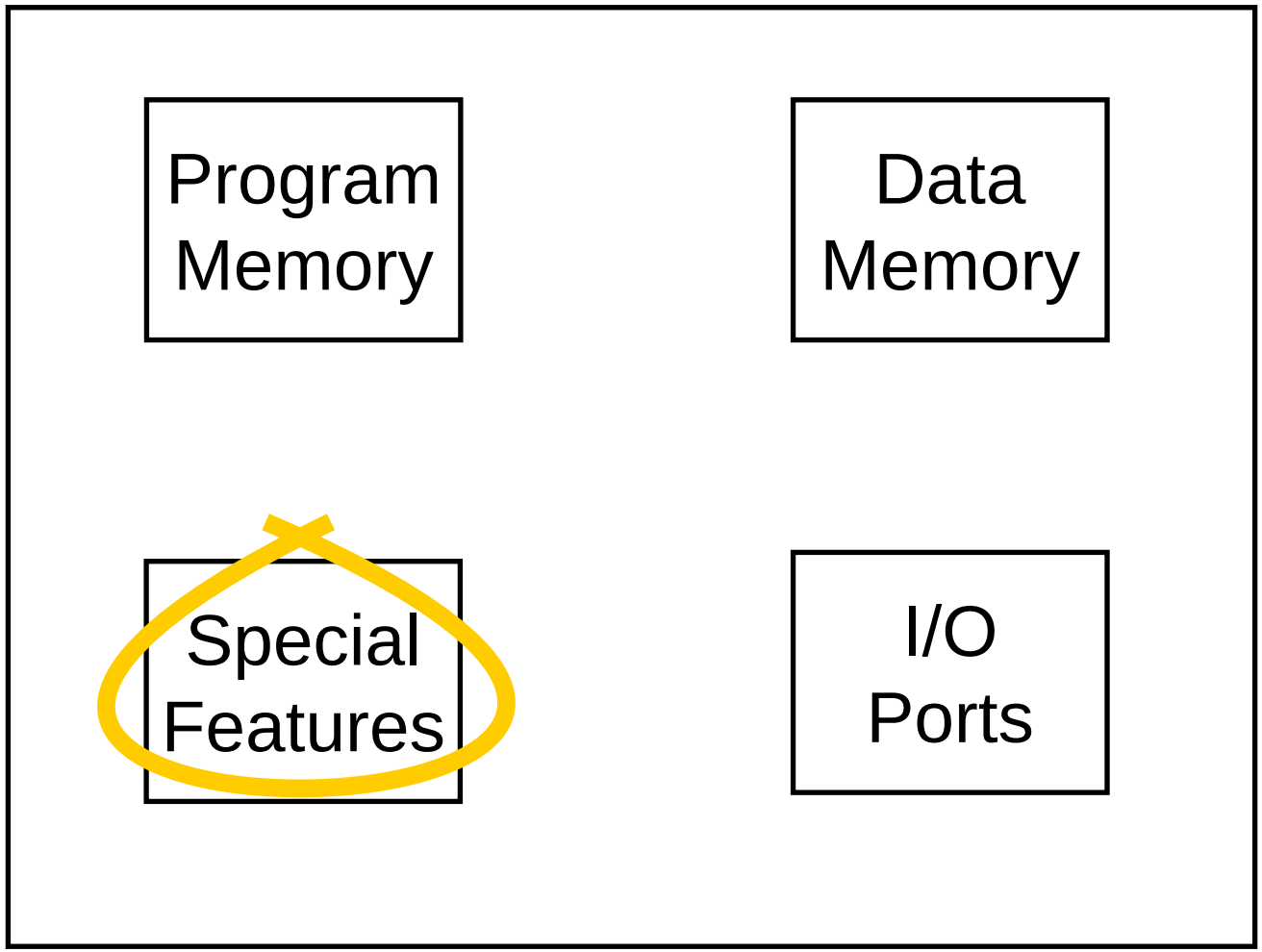
- I learned how to select my target device
- Program memory is ___ bits by ___ words
- Data memory is ___ bits by ___ words
- A “file” register is just another name for ___
- Program and data memory are separate
- Certain file registers are “special function”

Agenda

- Introduction to Microchip
- Mechatronics examples and benefits
- PIC® Microcontroller Basics
  - **Hands-On Learning Cycles**
- Labs
  - Simple I/O and Timer 0
  - Reading an analog sensor, LCD module
  - Controlling the speed of a motor
- Resources



Architecture: Overview



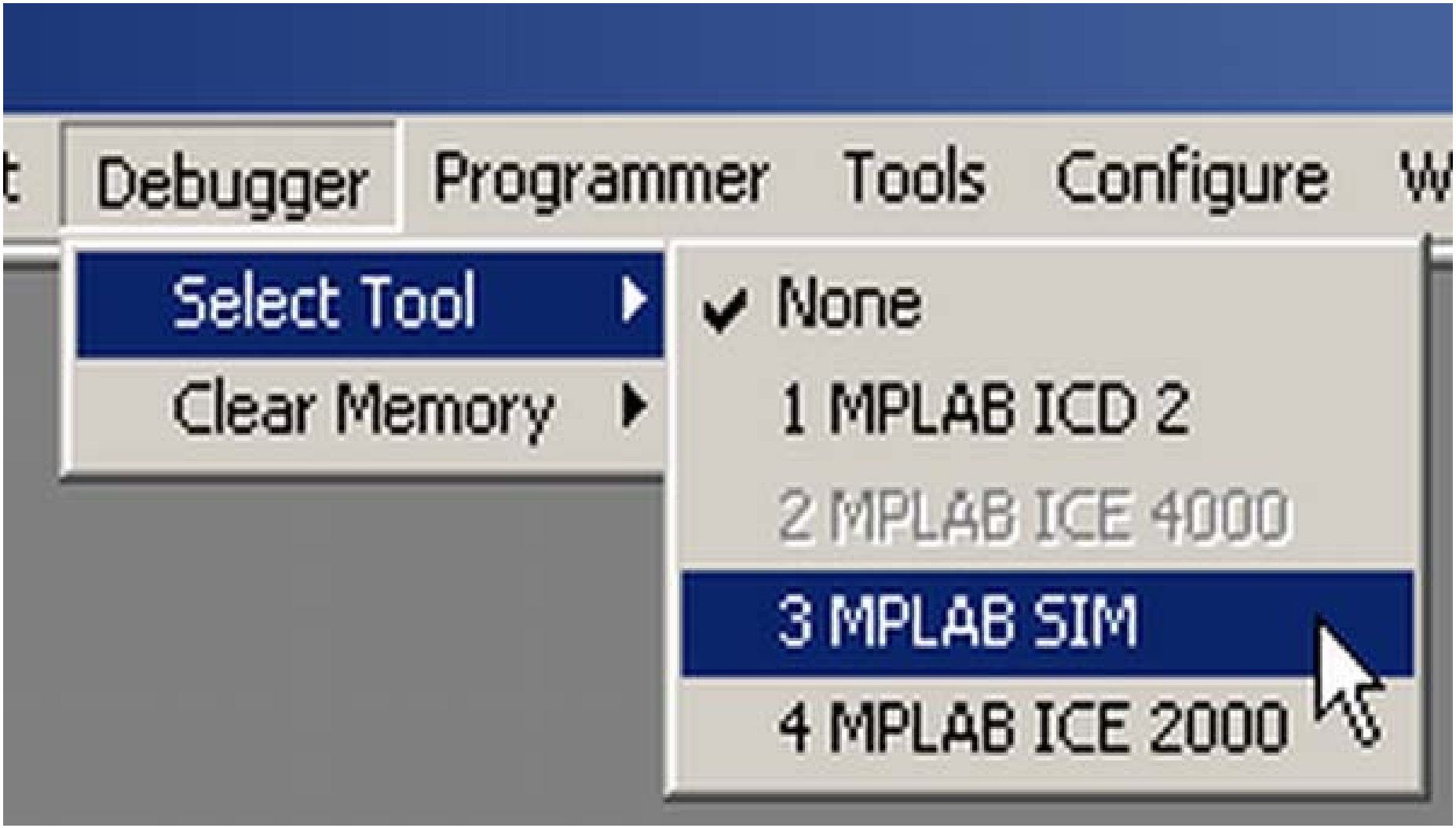
Architecture: W Register

Special Features

- **W** register is an 8-bit temporary working register
- **W** register is used in many instructions

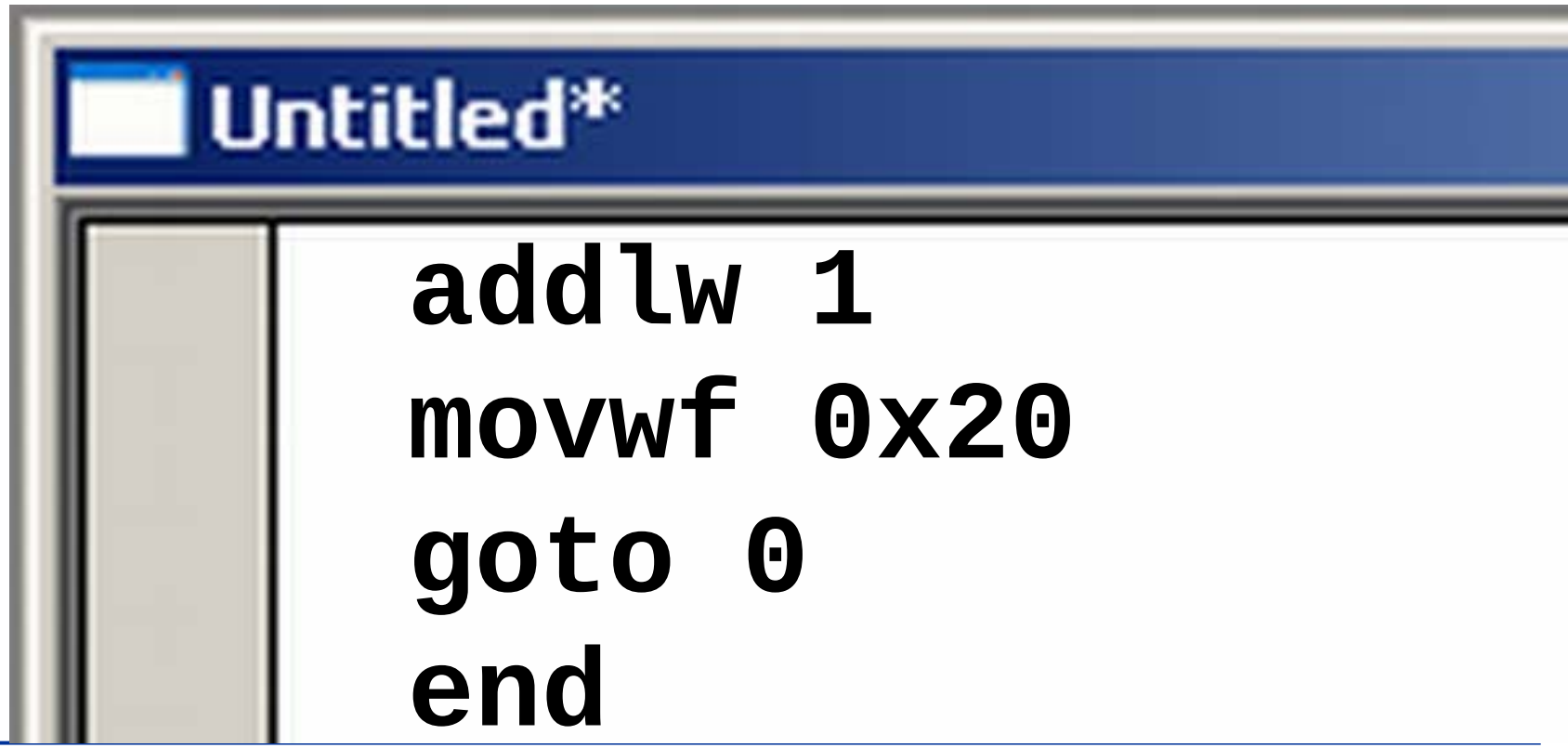
Hands-on with MPLAB® IDE

MPLAB® Simulator



Creating My First Program

- Click on **`File – New`**
- Enter the four lines of code below:



Creating Your First Program

- Click on **File – Save As…**

  c:\h1.asm

- **Project – Quickbuild h1.asm**

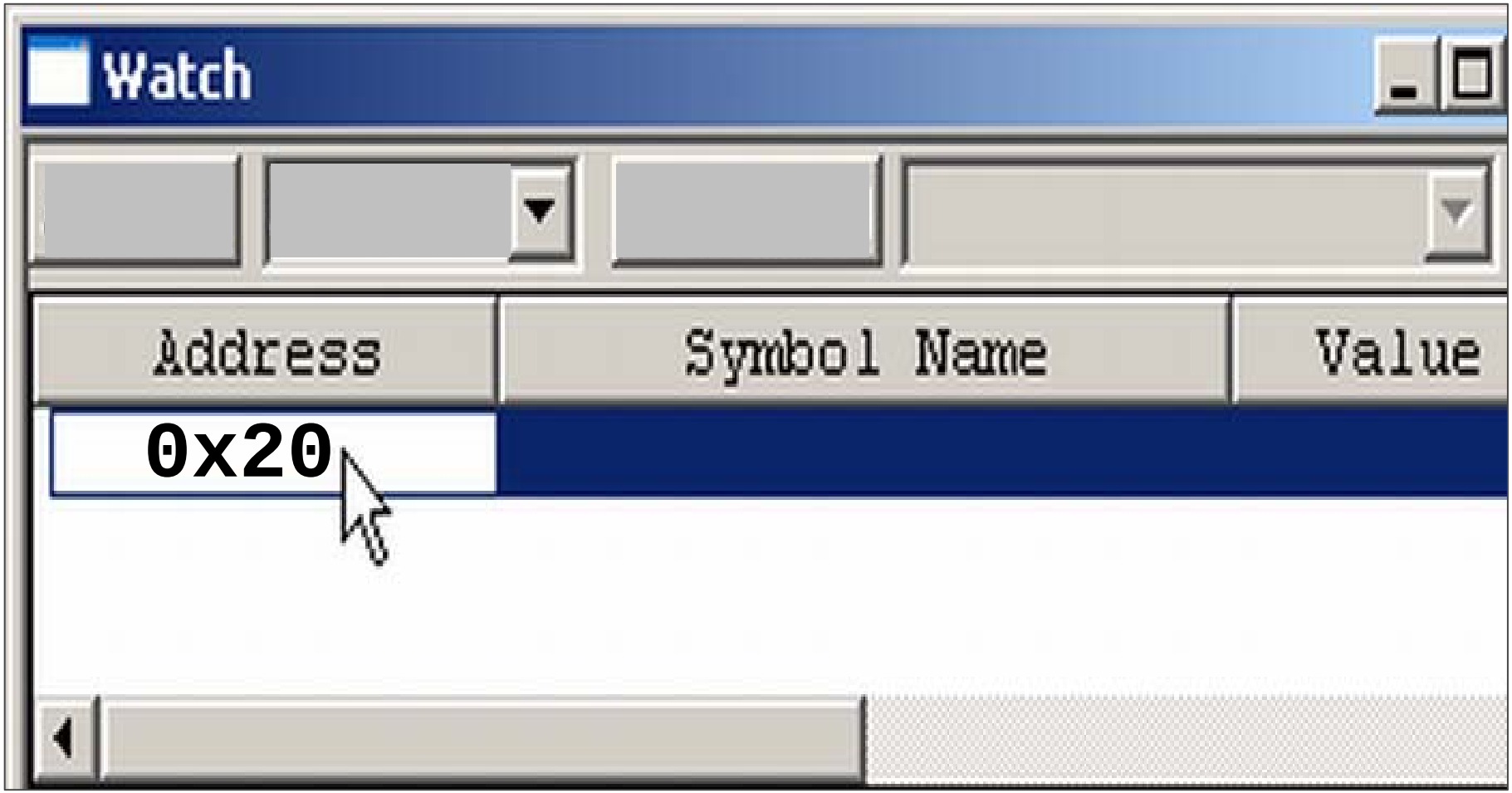
- Look in the Output Window for

  “BUILD SUCCEEDED”

Running Your First Program

Adding a Watch Window

- Click on **`View – Watch`**

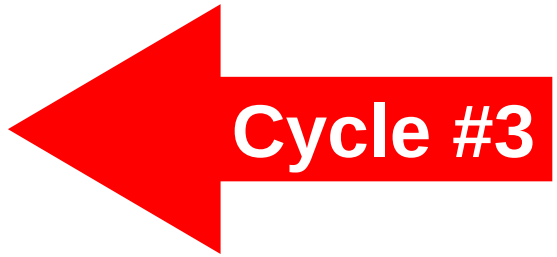


Cycle #2: What Did I Learn?

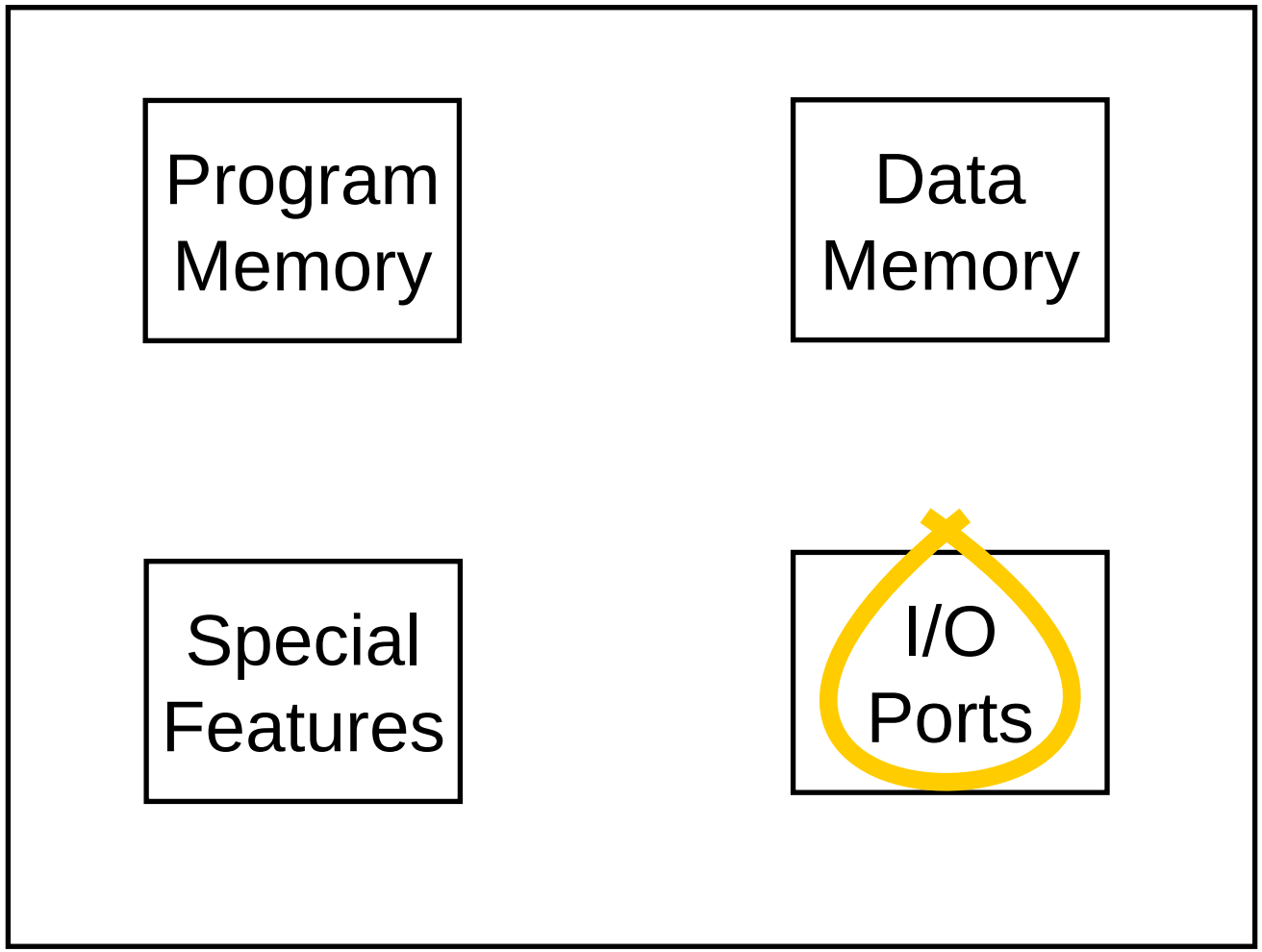
- W Register
- Create a .asm file
- Quickbuild
- MPLAB® SIM – step, animate
- Watch window
- Status bar
- Three instructions
- One assembler directive

Agenda

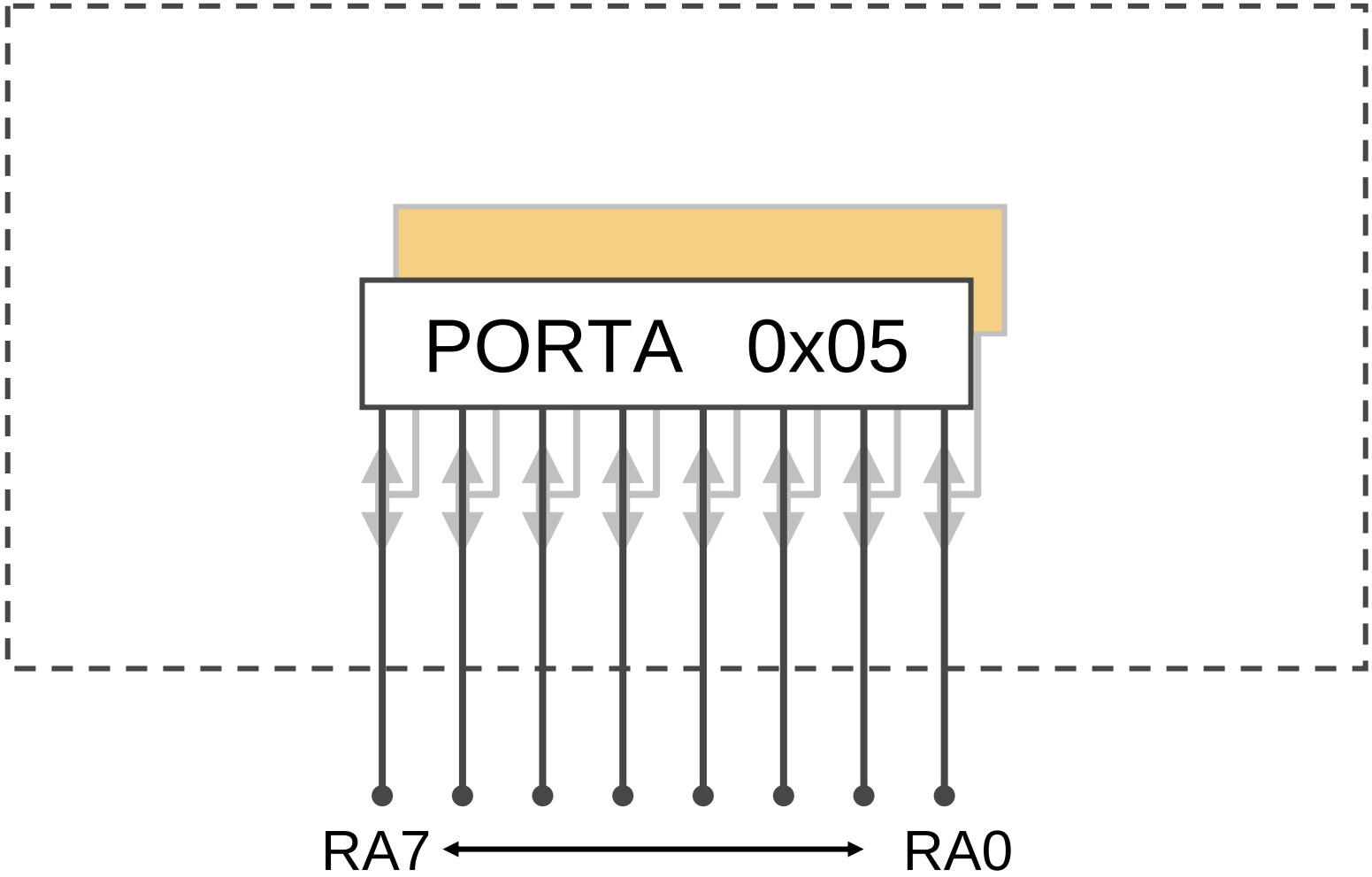
- Introduction to Microchip
- Mechatronics examples and benefits
- PIC® Microcontroller Basics
  - **Hands-On Learning Cycles**
- Labs
  - Simple I/O and Timer 0
  - Reading an analog sensor, LCD module
  - Controlling the speed of a motor
- Resources



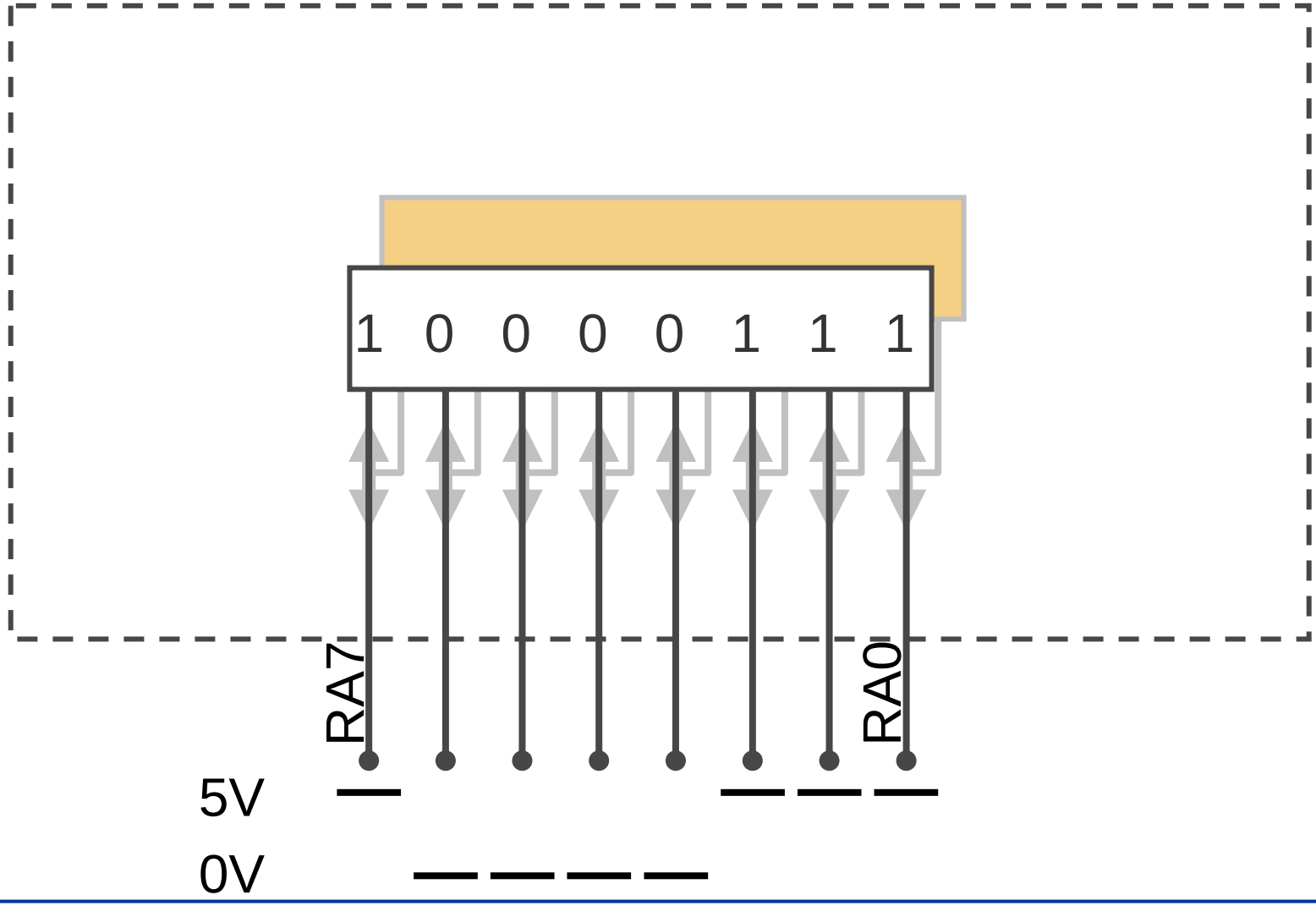
Architecture: Overview



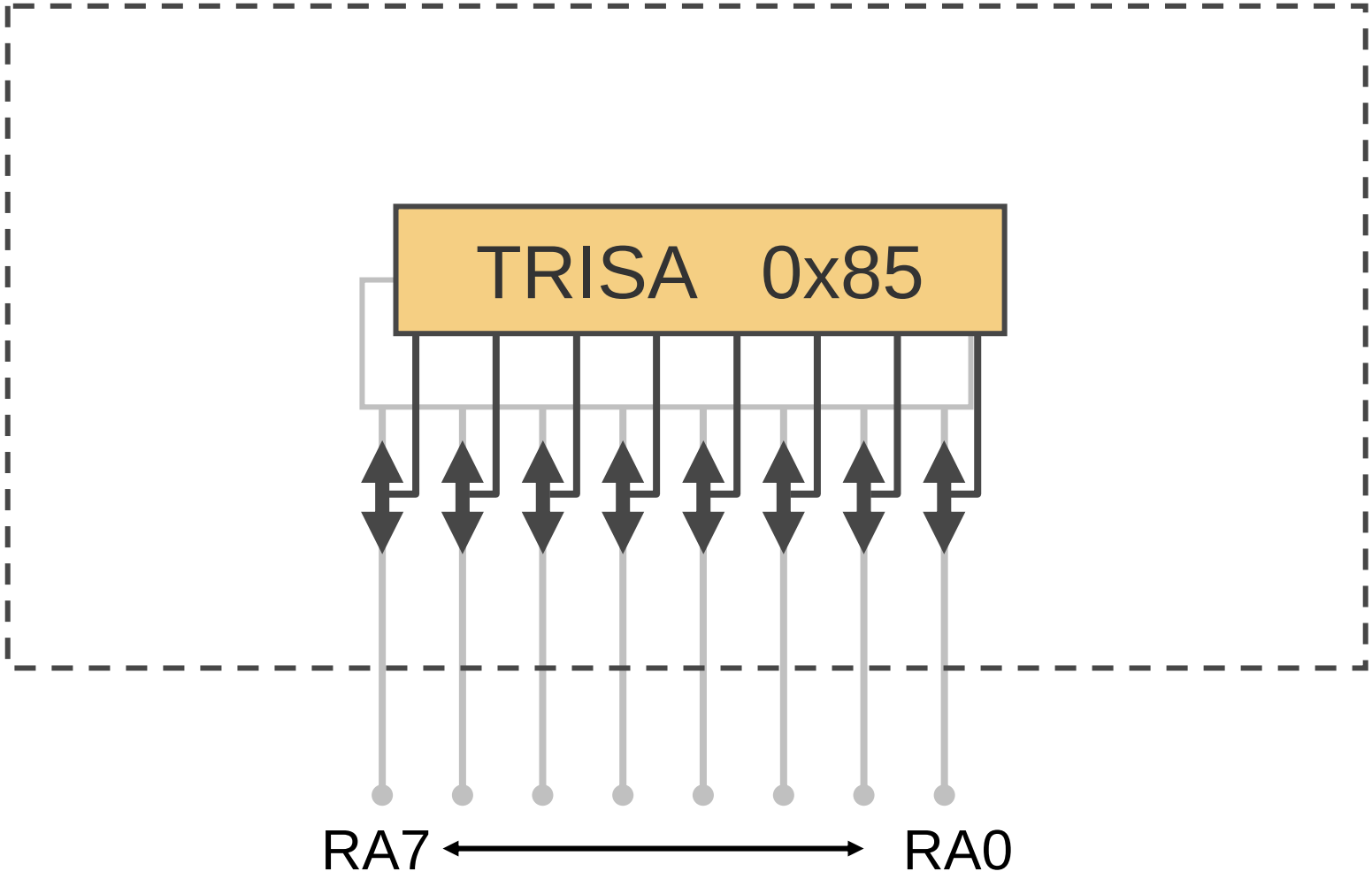
I/O Ports: PORTA Register



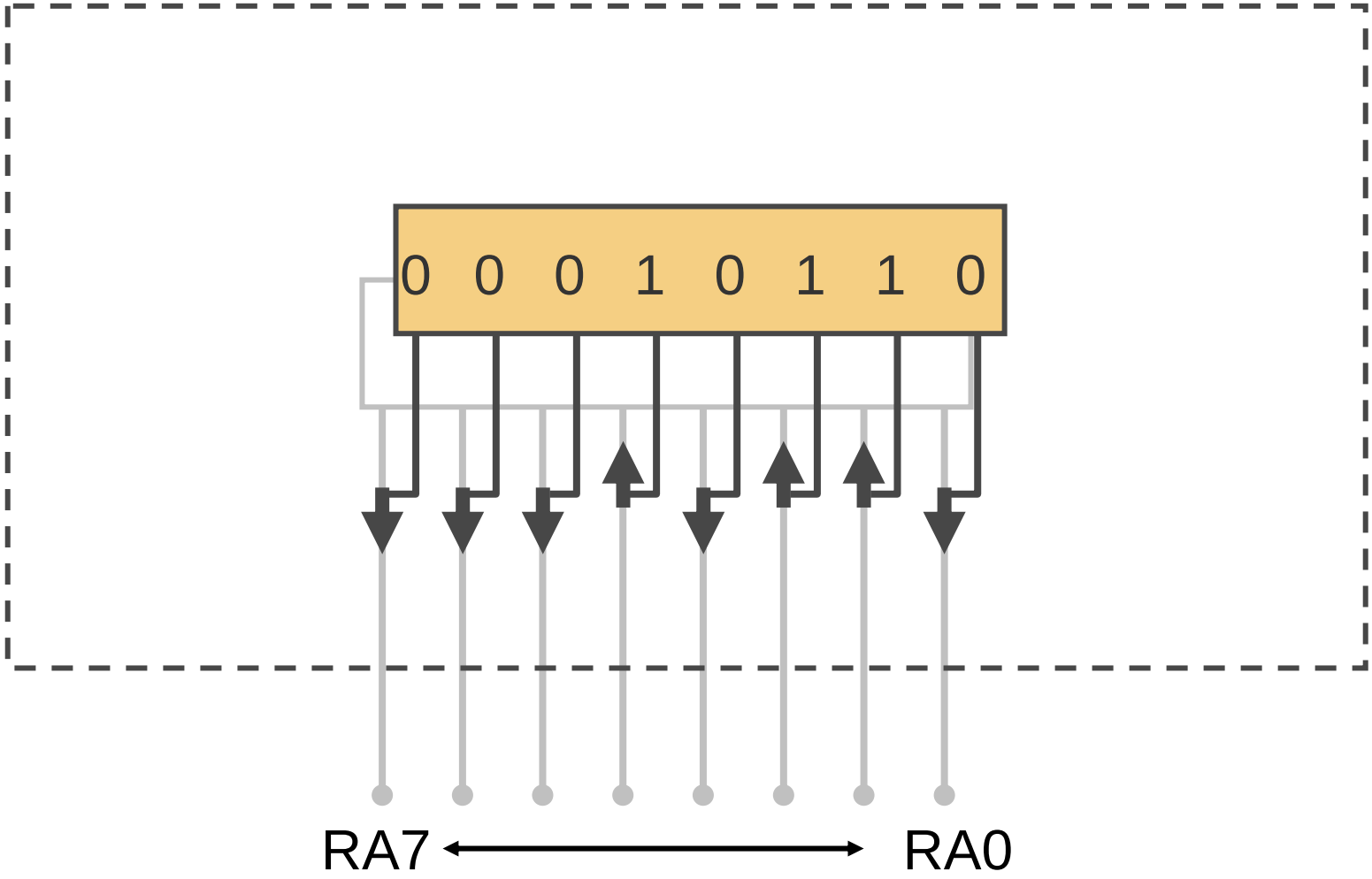
I/O Ports: PORTA Register



I/O Ports: TRISA Register



I/O Ports: TRISA Register



I/O Ports: Port D

- PORTD: 8-bit I/O port
- PORTD is actually data memory 0x08
- TRISD: 8-bit direction control register
- TRISD is actually data memory 0x88

Hands-on with MPLAB® IDE

Code – Round 3

- Change your code to

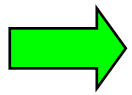
```
movlw b'00000000'
movwf 0x08
movlw b'11111111'
movwf 0x08
goto 0
```

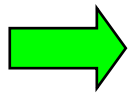
- Quickbuild

- Change the Watch address to 8

Stepping Through the Code

- Do a **`Window – Tile Horizontally`**
- Use **F7** to Single-Step through the code





```
movlw b'00000000'
movwf 0x08
movlw b'11111111'
movwf 0x08
goto 0
```

PORTD Direction Control

- Add TRISD (0x88) to your Watch
- PORTD is all inputs!
- Set TRISD to all outputs (zeros)

TRISD - Direction Control

- Add this line to the top of your code

```
clrf 0x88 ; clear TRISD
```

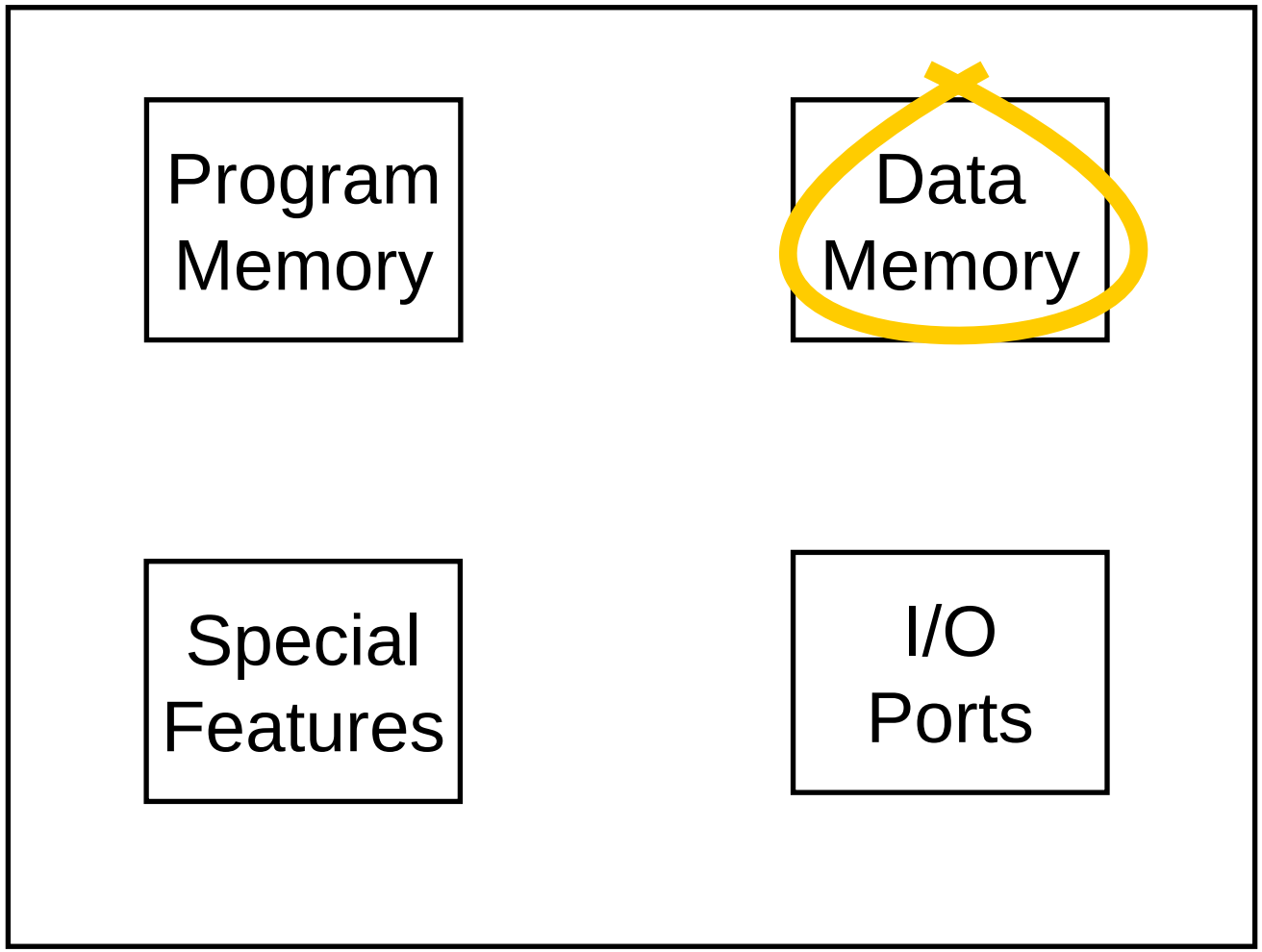
- Build, single-step

- Can you see TRISD being cleared?

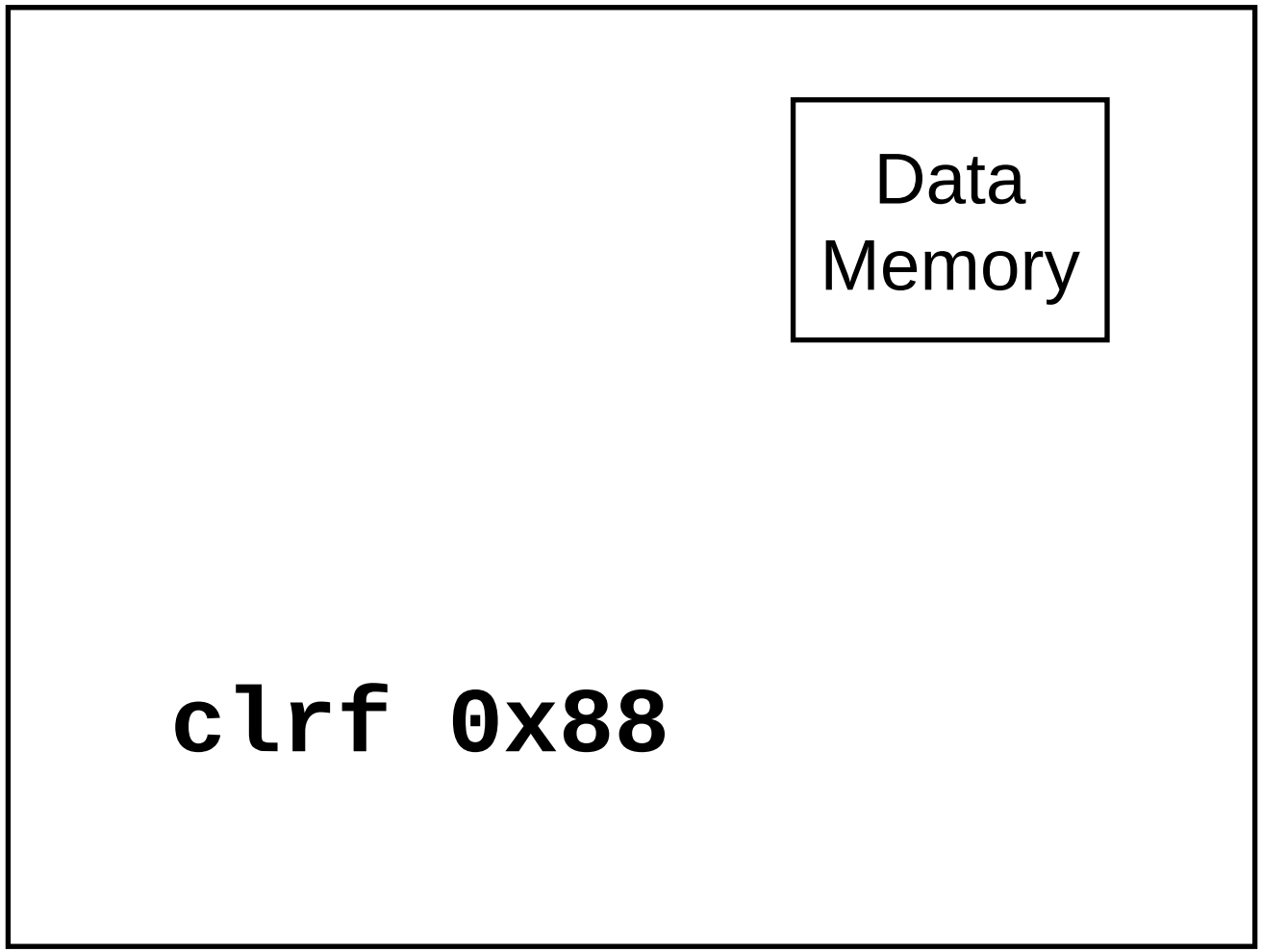
Why doesn’t TRISD clear?

Let’s revisit data memory…

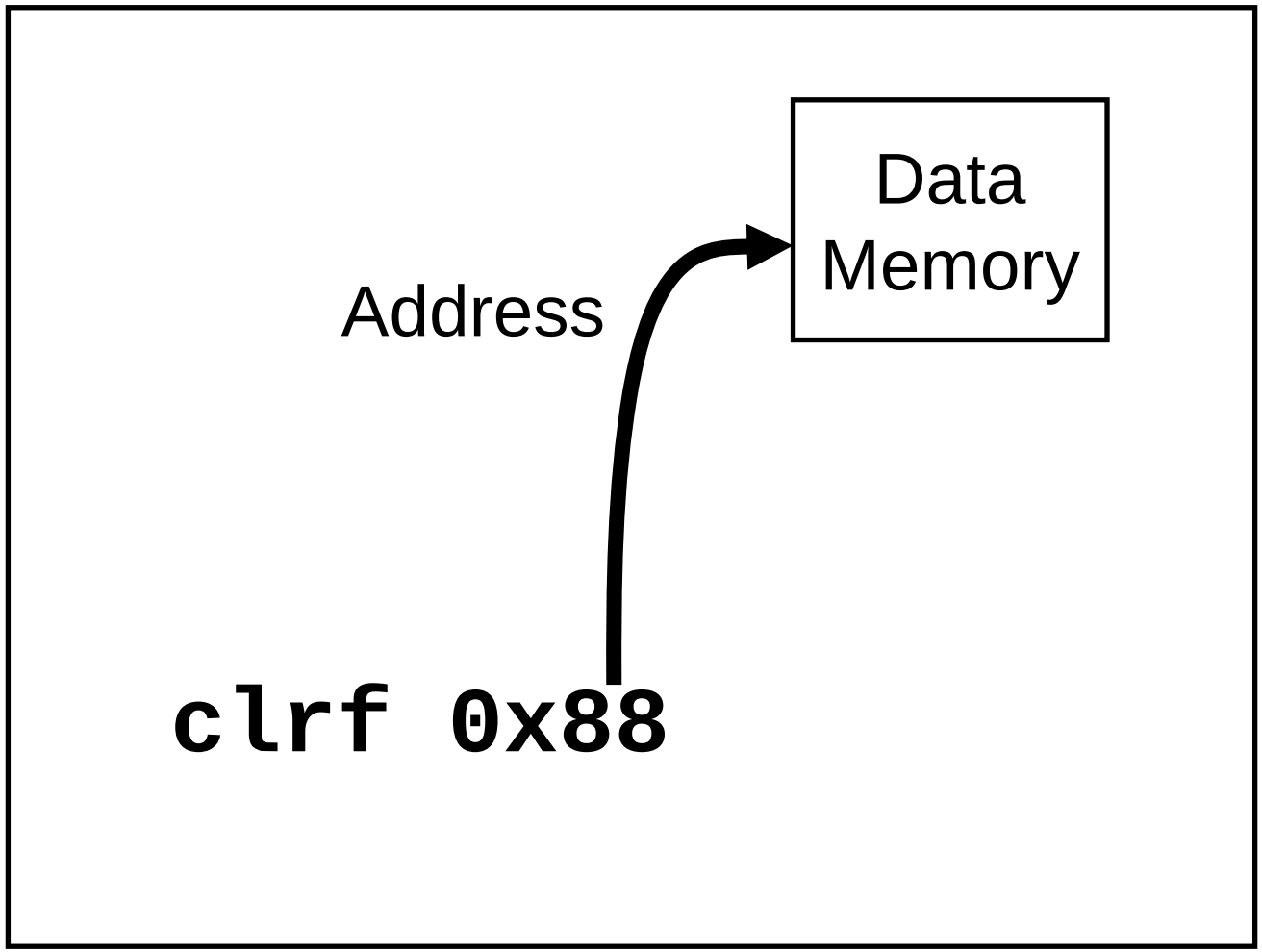
Architecture: Overview



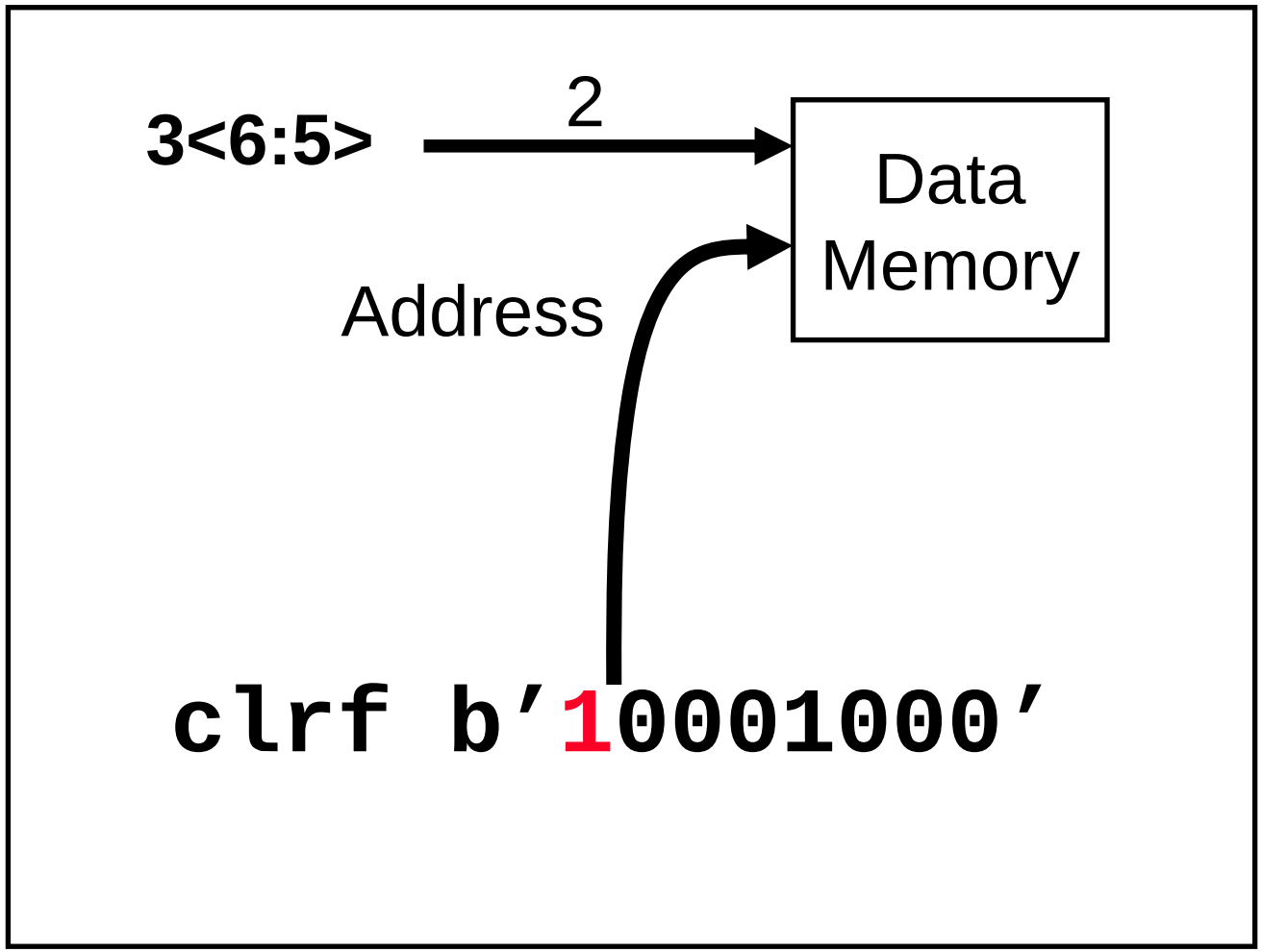
Accessing Data Memory



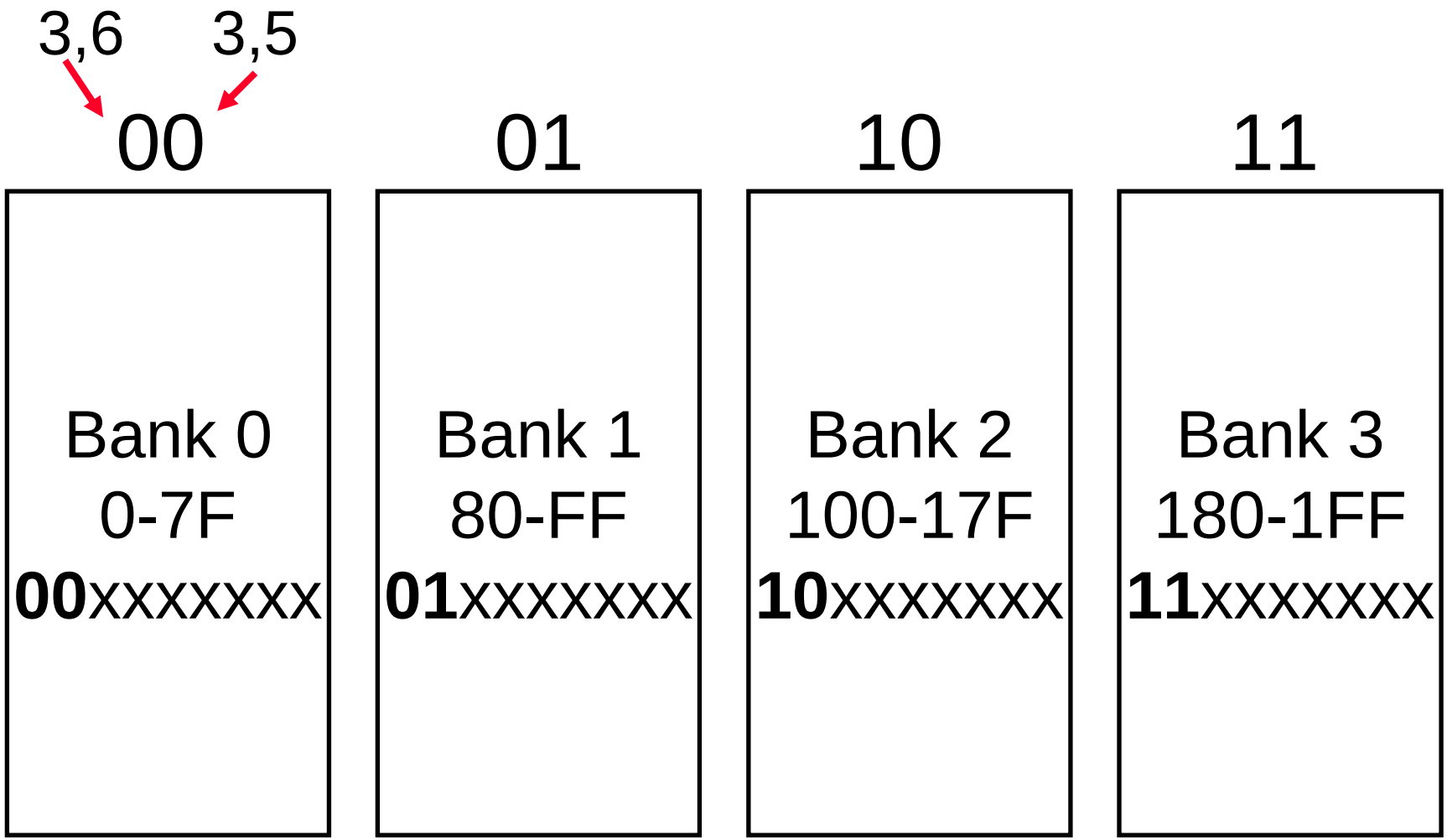
Architecture: Data Memory



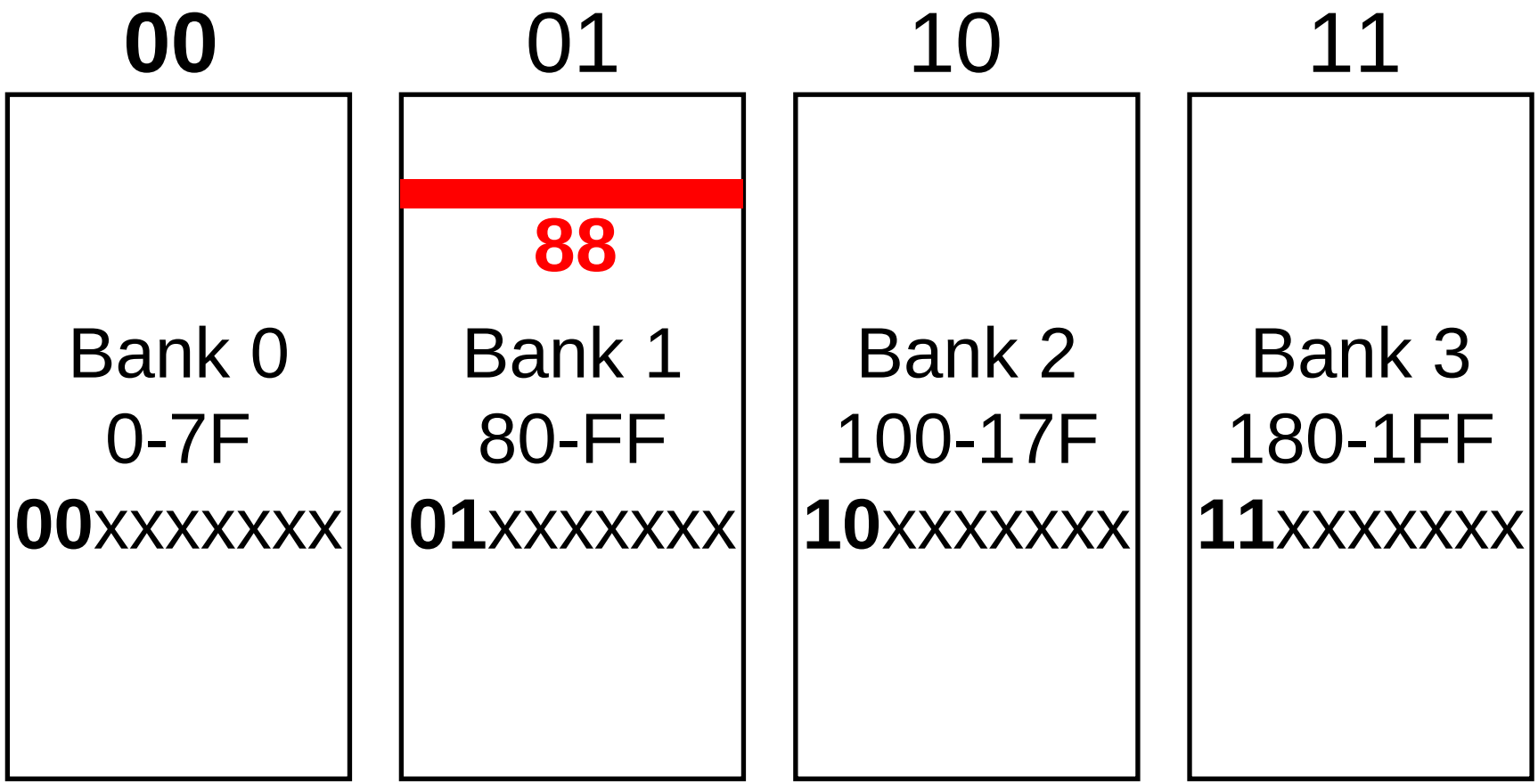
Architecture: Data Memory



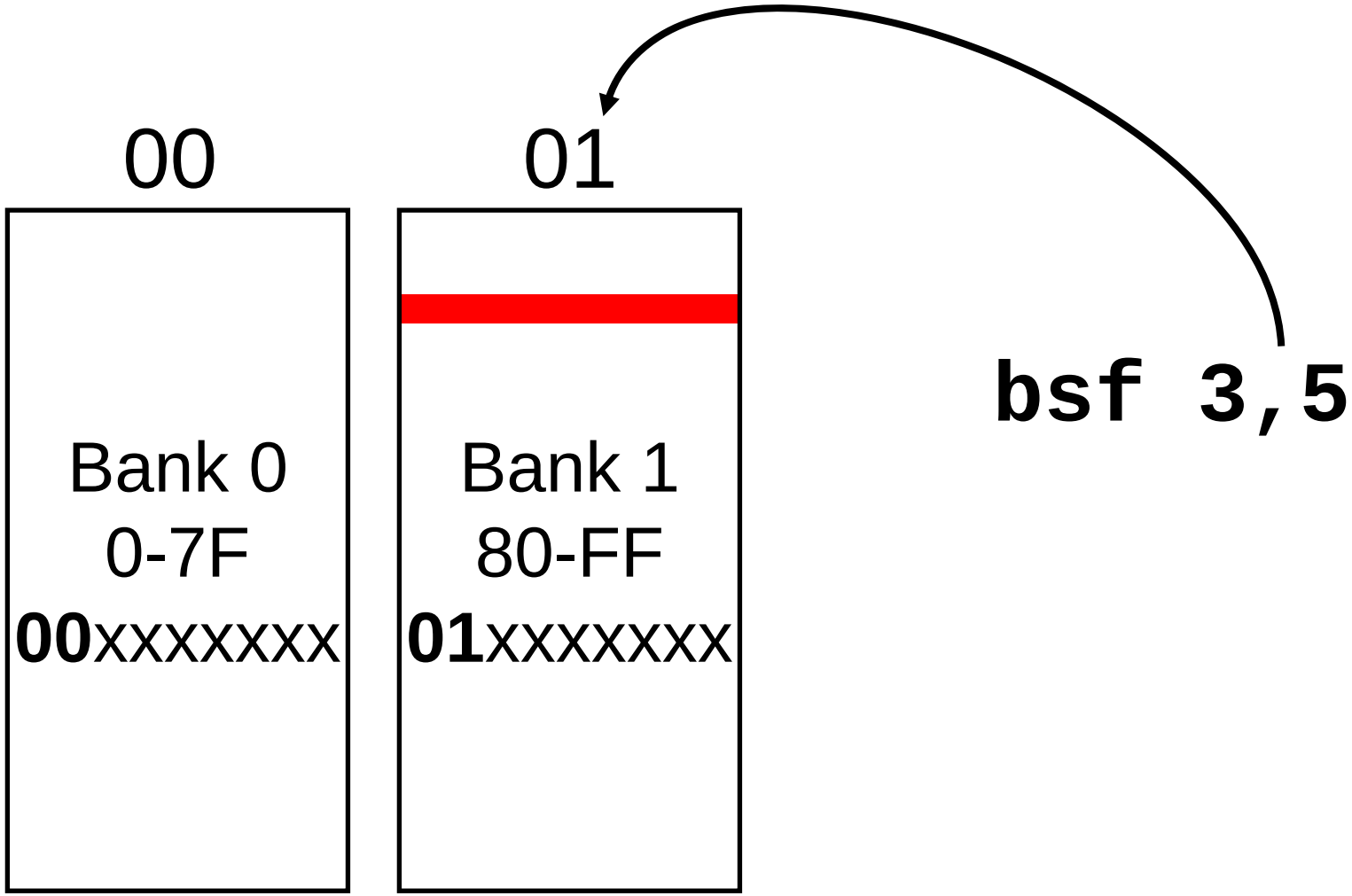
Architecture: Data Memory



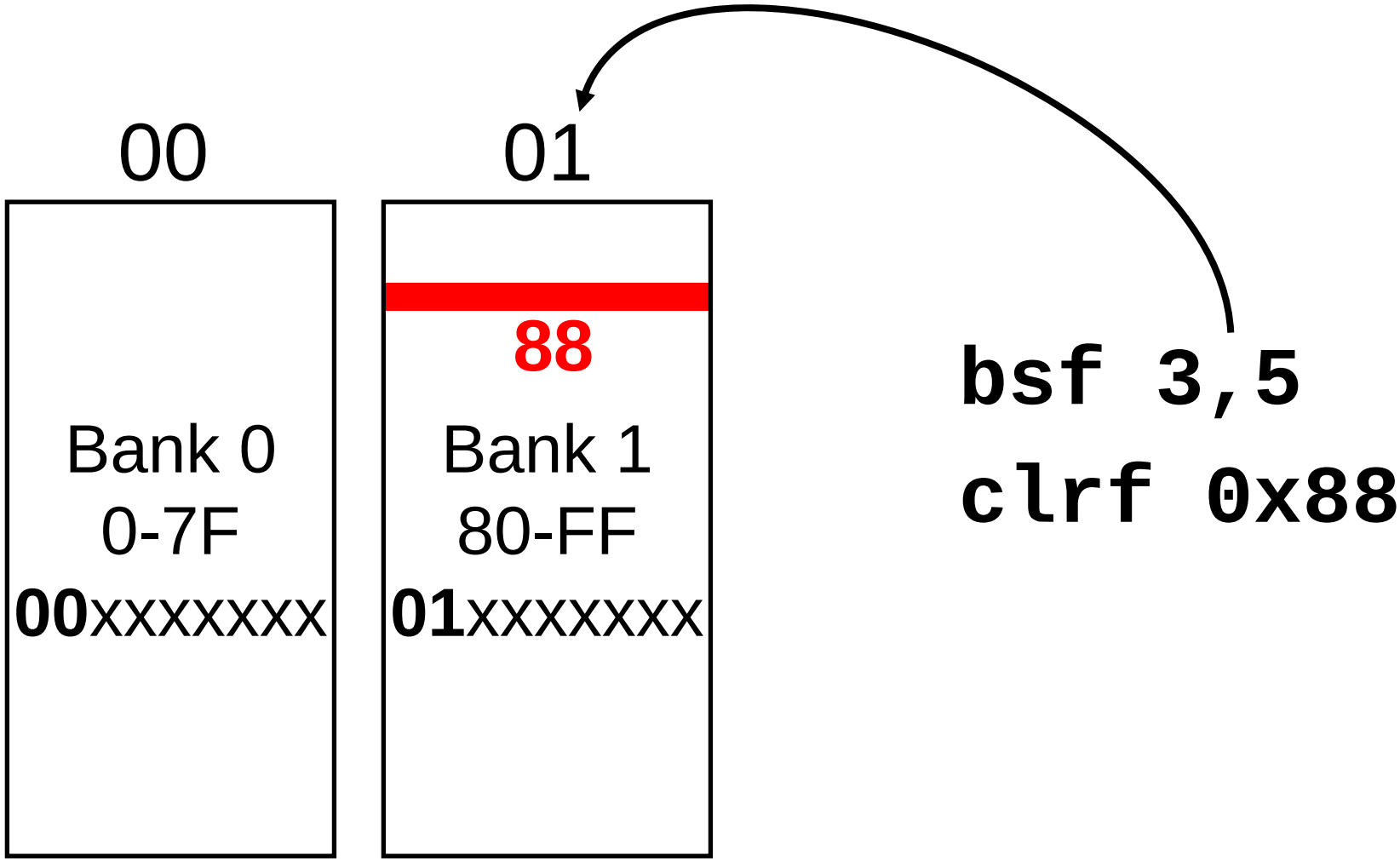
Architecture: Data Memory



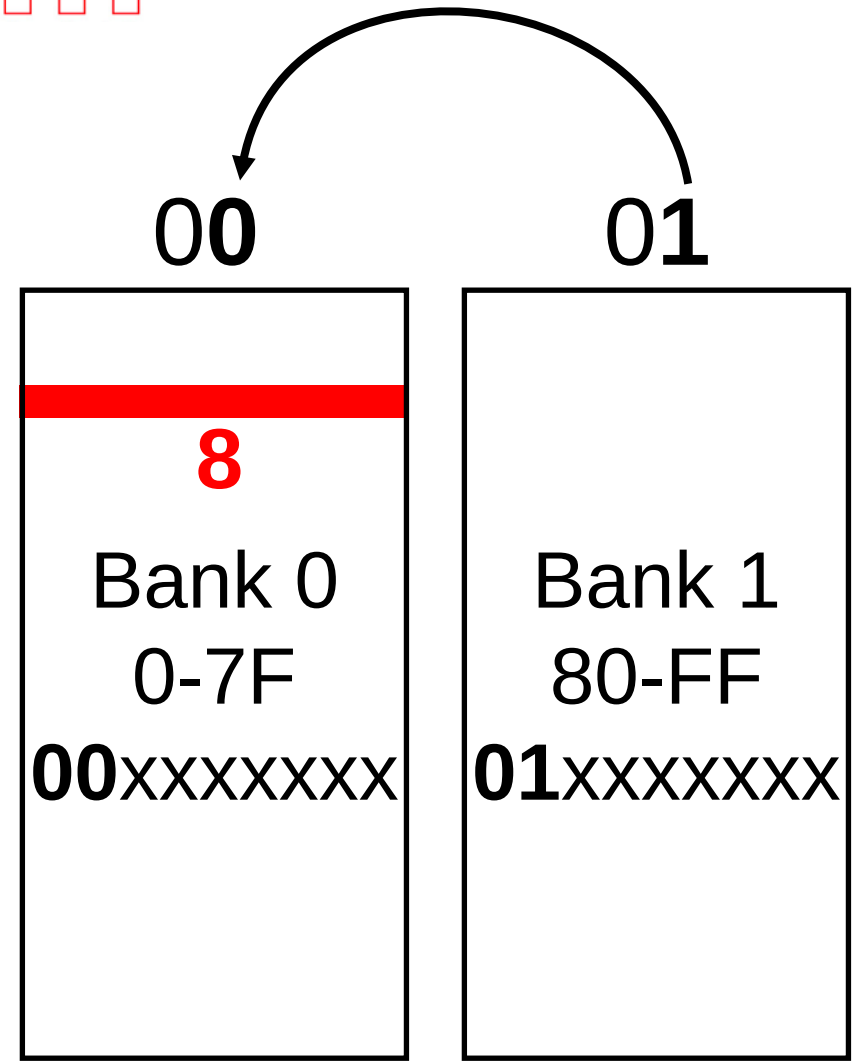
Architecture: Bank 1 Select



Architecture: TRISD



Architecture: Bank 0 Select




```
bsf 3,5
clrf 0x88
????
```

TRISD - Direction Control

- Our final code

```
bsf 3,5
clrf 0x88
bcf 3,5
movlw b'00000000'
movwf 0x08
movlw b'11111111'
movwf 0x08
goto 3
```

Symbol Name and File Register Correlation

```
bsf 3,5
clrf 8
bcf 3,5

movlw 0
movwf 8
movlw 0xff
movwf 8
goto 3
```

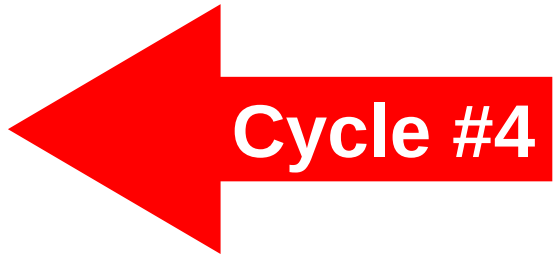
```
include <p16f917.inc>
  bsf STATUS,RP0
  clrf TRISD
  bcf  STATUS,RP0
Loop
  movlw 0
  movwf PORTD
  movlw 0xff
  movwf PORTD
  goto Loop
```

Cycle #3: What Did I Learn?

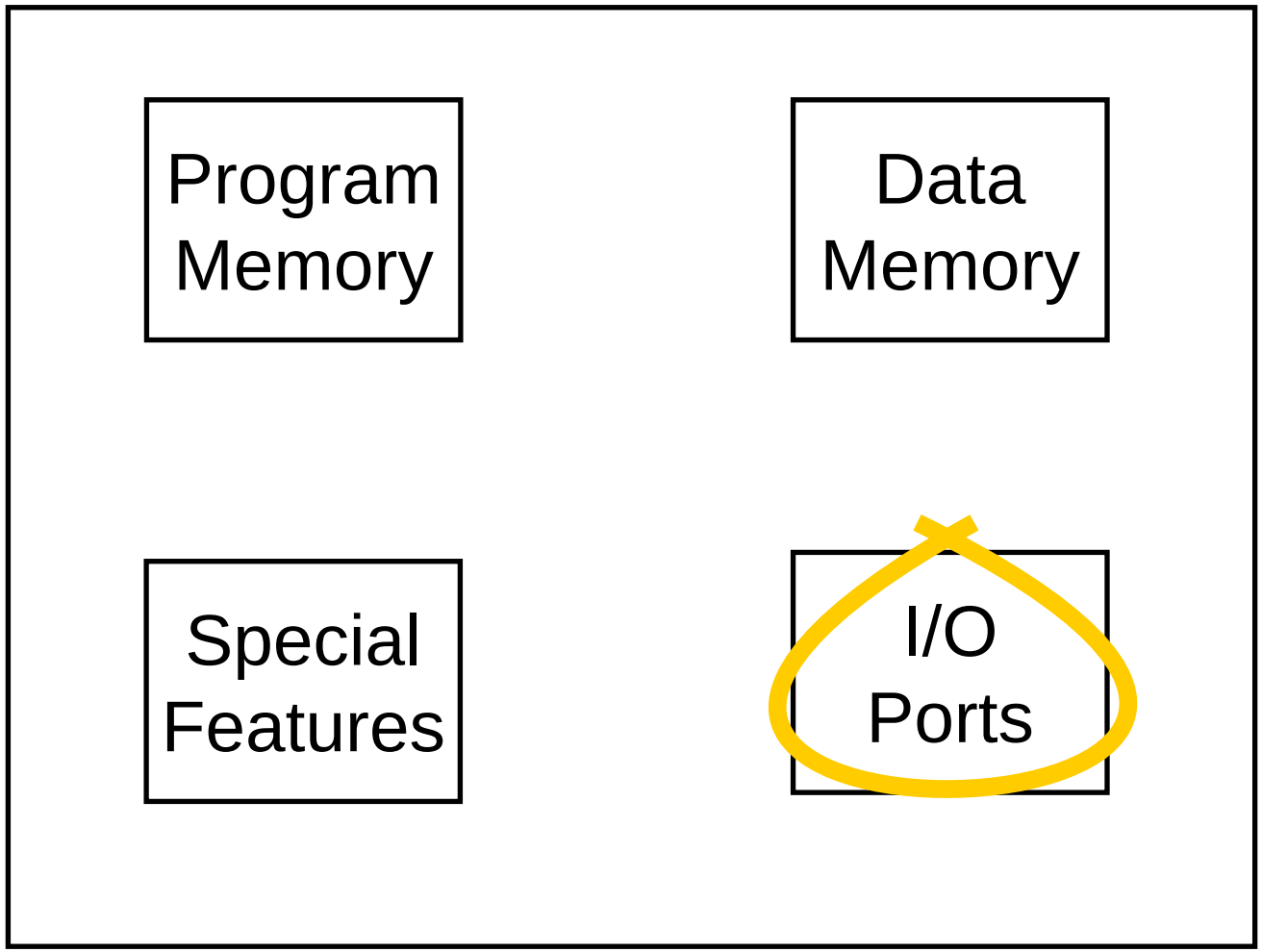
- PORTx Registers
- TRISx Registers
- Banking
- STATUS register, bits RP0 & RP1
- Symbol name and number correlation
- Four instructions
- One assembler directive

Agenda

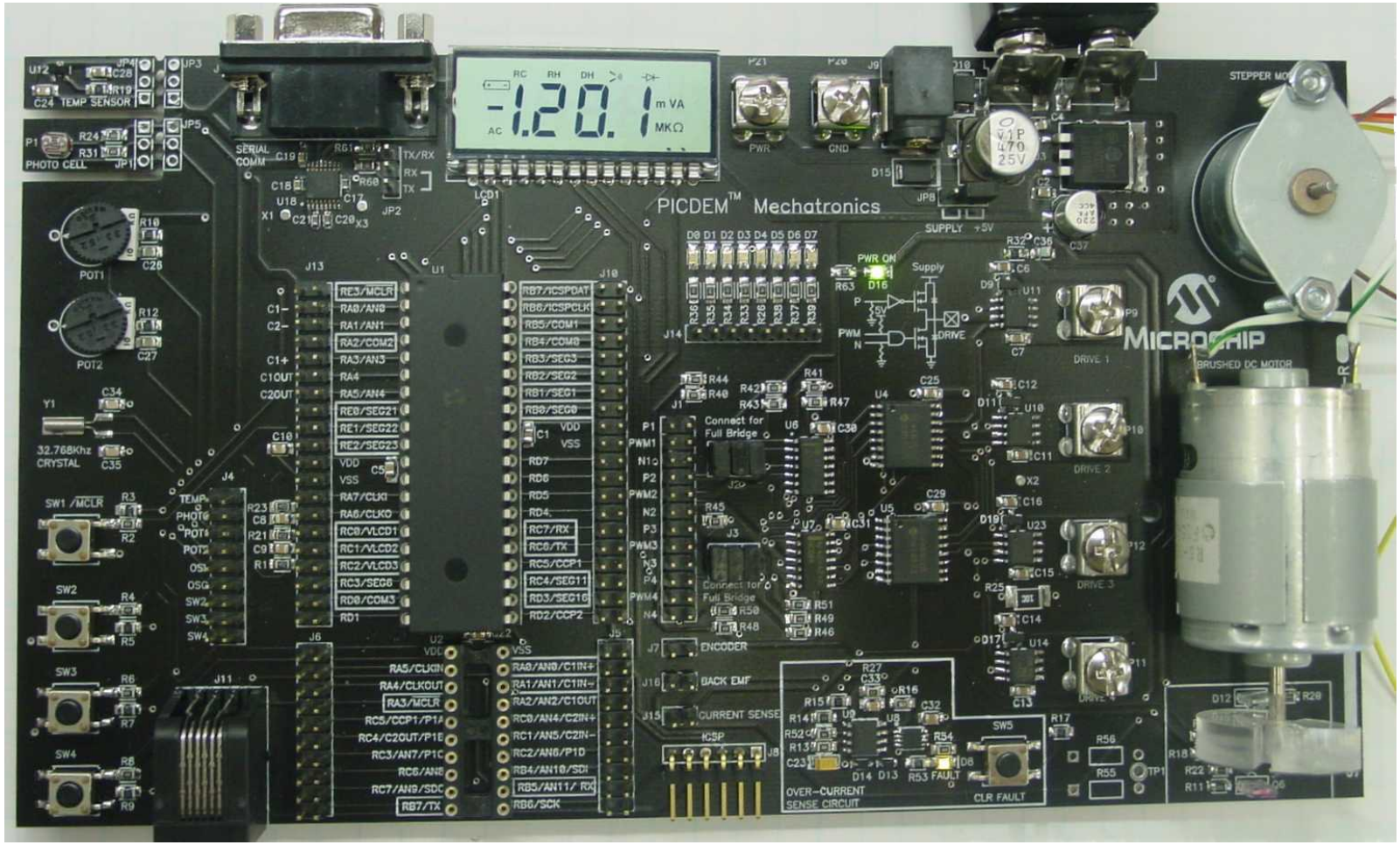
- Introduction to Microchip
- Mechatronics examples and benefits
- PIC® Microcontroller Basics
  - **Hands-On Learning Cycles**
- Labs
  - Simple I/O and Timer 0
  - Reading an analog sensor, LCD module
  - Controlling the speed of a motor
- Resources



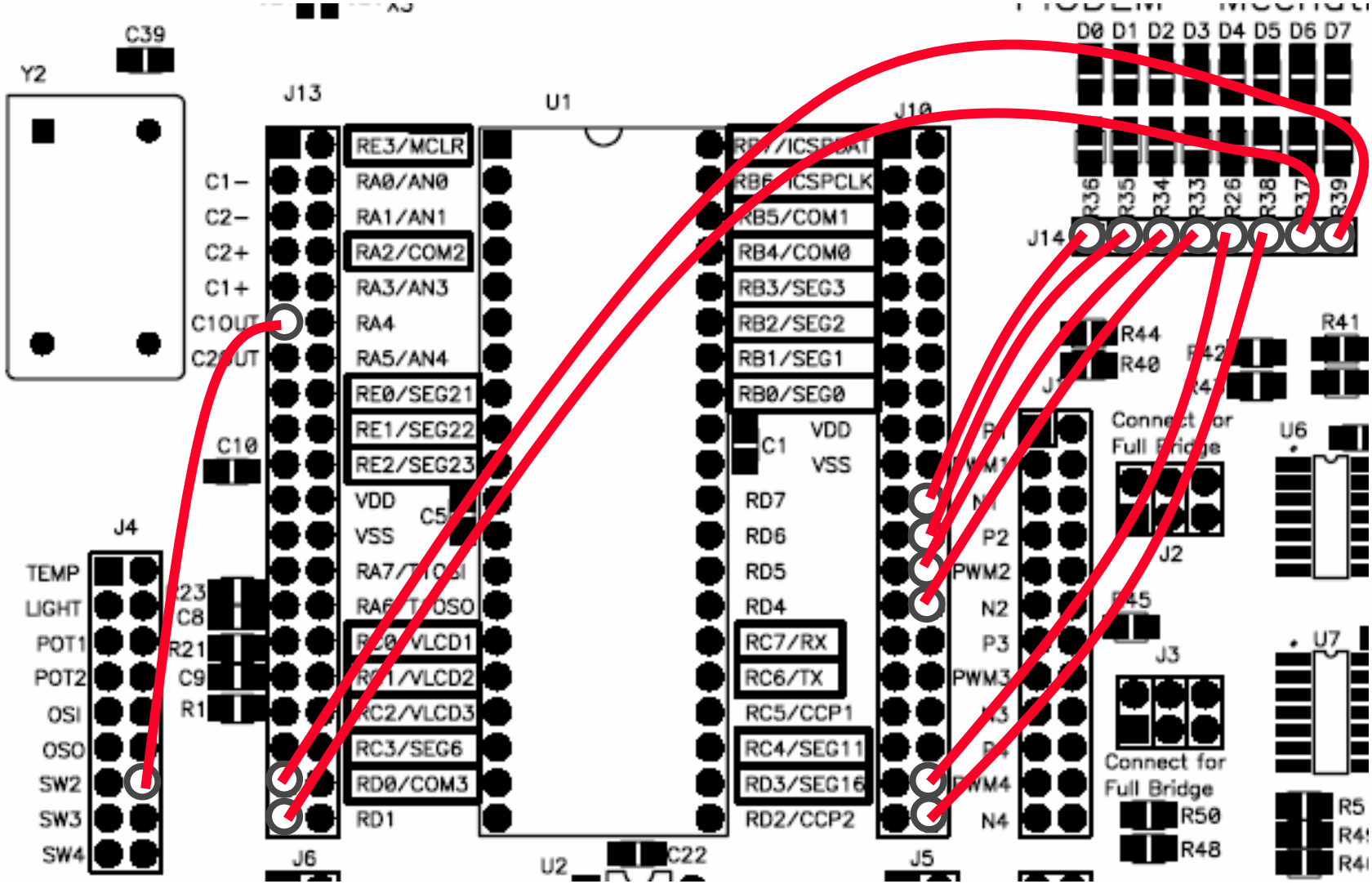
Architecture: Overview



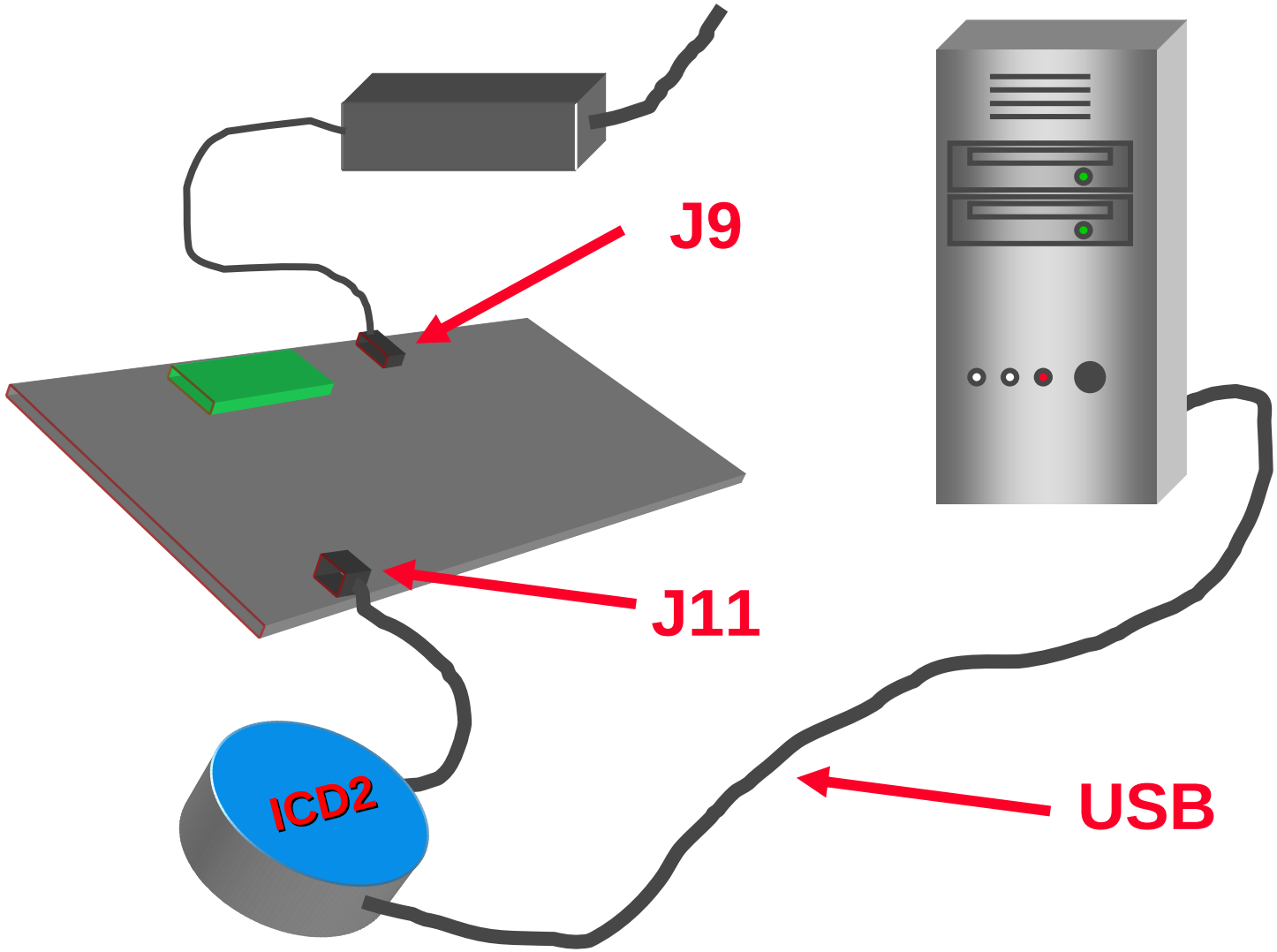
Introduction to the PICDEM™ Mechatronics Board



Connect the Hardware

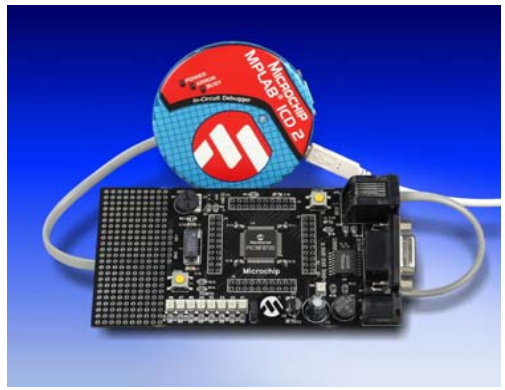


Connecting the ICD 2 and Power



Introduction to the MPLAB® ICD 2

- MPLAB ICD 2 is an In-circuit Debugger and Programmer
- Debug mode:
  - Find out why your program isn't doing what you expect it to do
  - Look at the data memory
  - Step through the code
  - Set break points
- Program mode:
  - Program a device



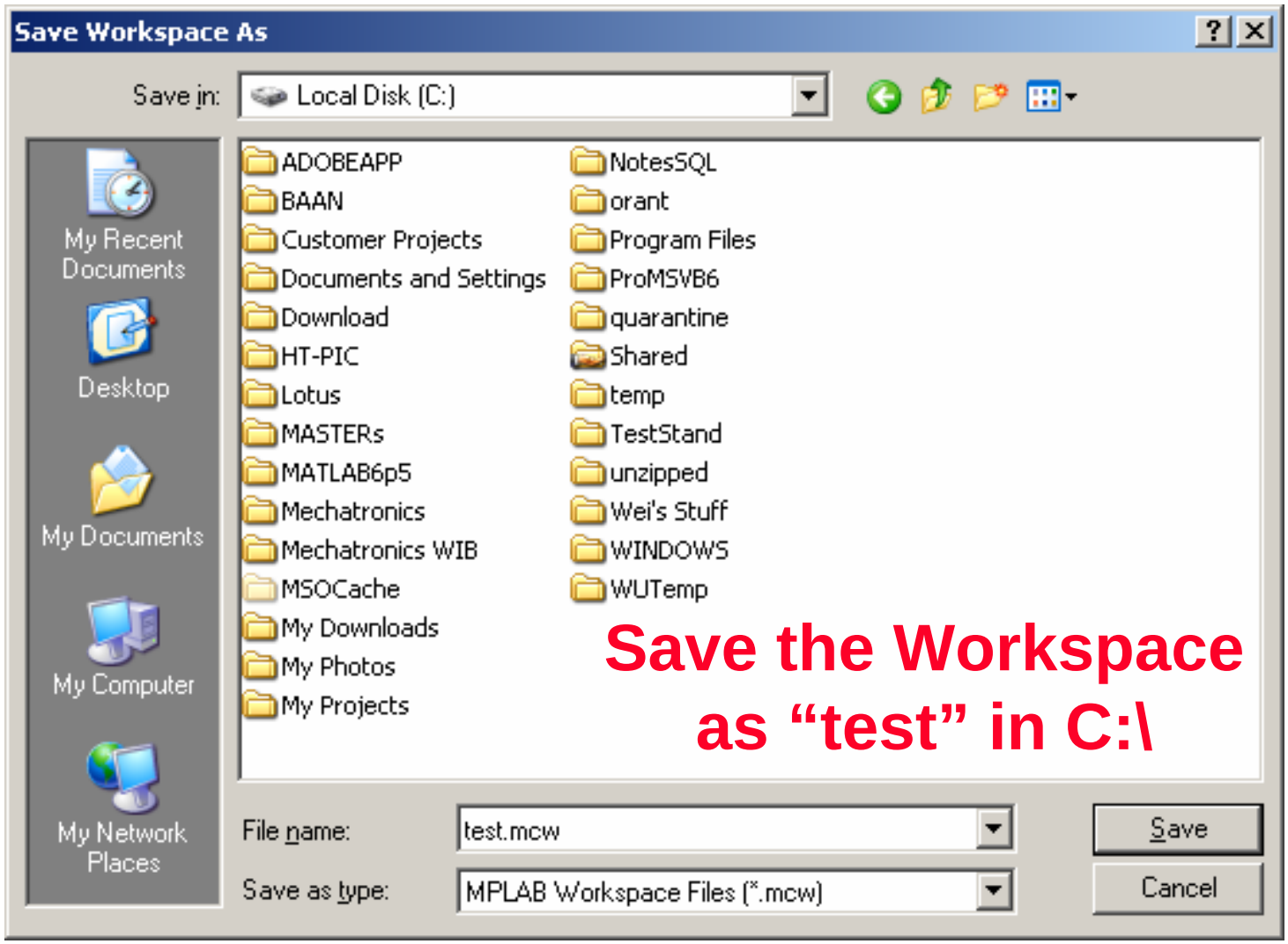
Hands-on with MPLAB® IDE

Create a Project In MPLAB IDE

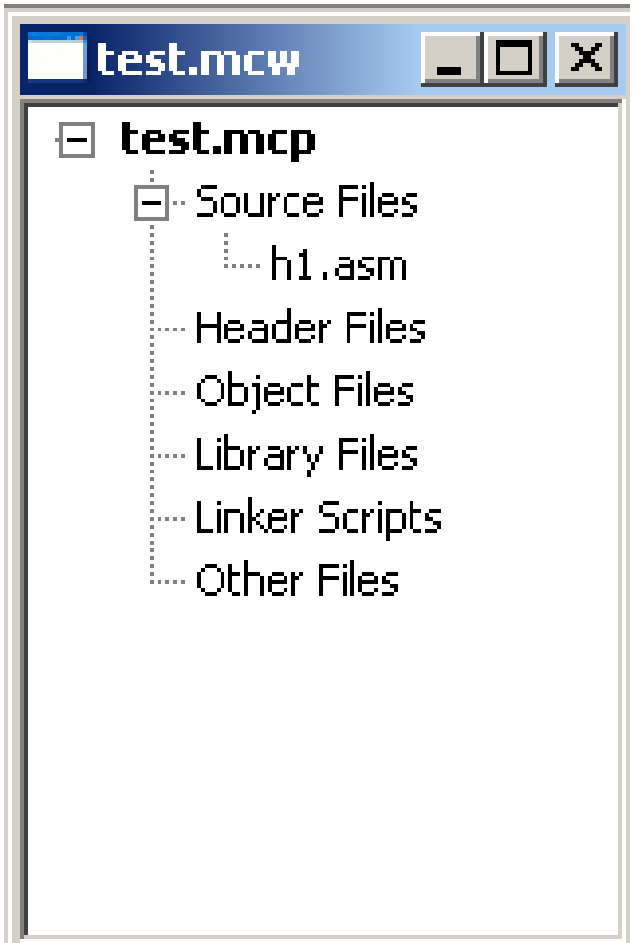
Creating a Project Continued

- Select the PIC16F917, click **Next**
- Choose the Microchip MPASM™ Toolsuite, click **Next**
- Create a project name of “test”
- Save in C:\, click **Next**
- Add h1.asm to the project, click **Next**
- Click **Finish**

Save Workspace



Opening Source Files



- The project window contains the source, include, and linker files
- Double-click on the file names to open (or focus on) the desired file
- Right-click on “Source Files” to add source files
- Remove files by right-clicking on them and choosing “Remove”

ICD 2 Setup

- Debugger - Select Tool - 1 MPLAB ICD-2
- Debugger - Connect
- Configuration - Configuration Bits…

Category	Setting
Oscillator	INTOSC
Watchdog Timer	Off
Power Up Timer	On
MCLR Pin Function Select	Normal Function
Code Protect	Off
Data EE Read Protect	Off
Brown Out Detect	BOD Enabled, SBODEN Dis
Internal-External Switch Over	Enabled
Fail Clock Monitor Enable	Enabled

Cycle #4: Write the Source Code

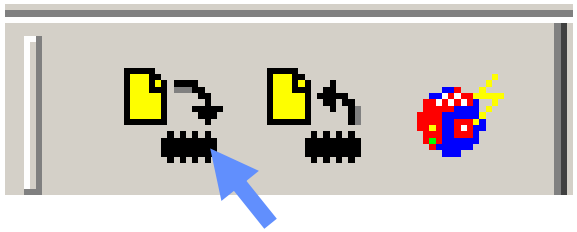
```
    include <p16f917.inc>
    bsf     STATUS,RP0
    clrf    TRISD
    bsf     TRISA,4
    bcf     STATUS,RP0
Loop
     btfsc   PORTA,4
     goto    Loop
     incf    PORTD,f
     goto    Loop
```

Building and Programming

- In a project the build command is F10.

  (Do this now!)

- Once the source code is built it must be programmed into the PIC16F917
- To program a device click on:



You must build and program after every change to your source code!

Watching the Source Run

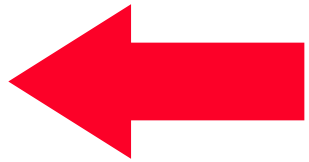
- Create a Watch window
- Watch “PORTD” and “PORTA”
- Click the Animate button
- What happens when you press SW2?

Cycle #4: What Did I Learn?

- Set up the PICDEM™ Mechatronics Board
- Configured the MPLAB ICD 2
- Created a Project
- Saved a Workspace
- Built a project
- Programmed the PIC16F917
- Ran code using the MPLAB ICD 2

Agenda

- Introduction to Microchip
- Mechatronics examples and benefits
- PIC® Microcontroller Basics
  - Hands-On Learning Cycles
- **Labs**
  - Simple I/O and Timer 0
  - Reading an analog sensor, LCD module
  - Controlling the speed of a motor
- Resources



Lab Introduction

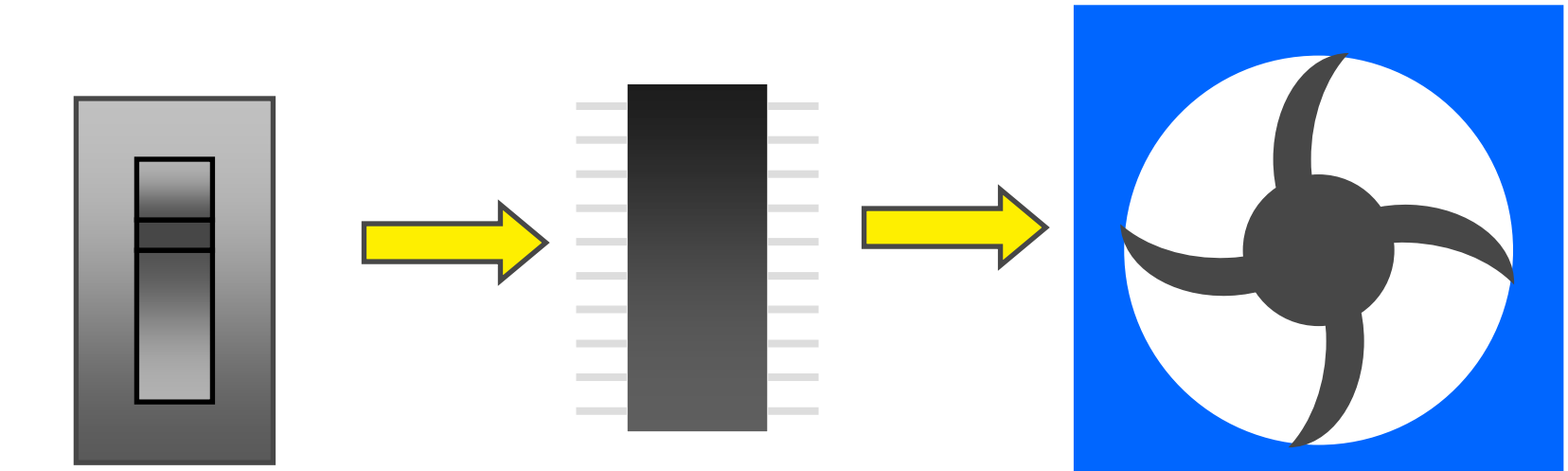
- Problem: Management wants you to design a cooling fan for an electronics bay in the new Mars Rover

Agenda

- Introduction to Microchip
- Mechatronics examples and benefits
- PIC® Microcontroller Basics
  - Hands-On Learning Cycles
- Labs
  - **Simple I/O and Timer 0**
  - Reading an analog sensor, LCD module
  - Controlling the speed of a motor
- Resources

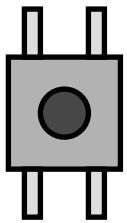
Lab 1: Design Objectives

- Read the state of a thermal breaker
- Turn on the fan when the breaker trips
- When the breaker resets, wait 2 seconds, then turn off fan

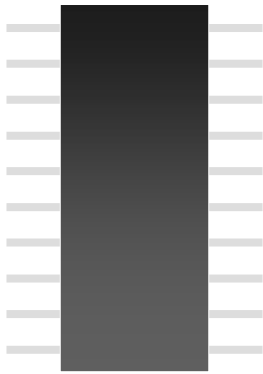


Lab 1: Hardware

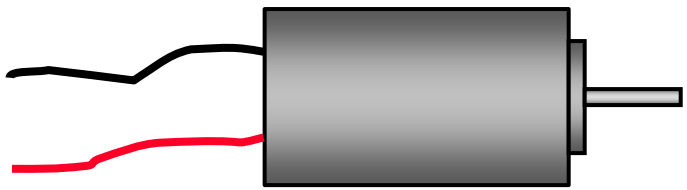
- We will use a tactile switch on the board to simulate the thermal breaker
- The Brushed DC motor will substitute for the fan
- For the PIC16F917 . . . . well we'll use that!



Tactile Switch

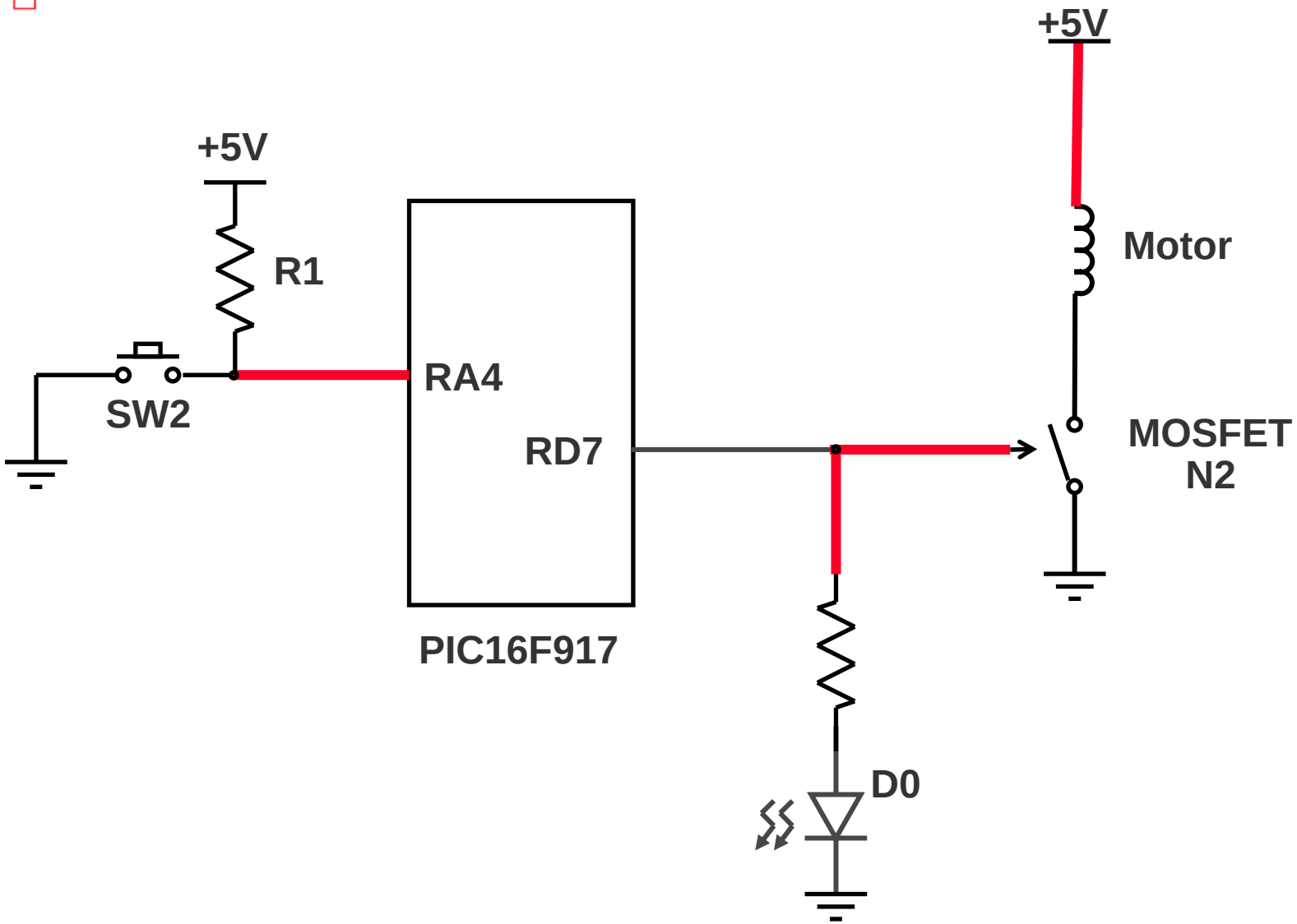


PIC16F917

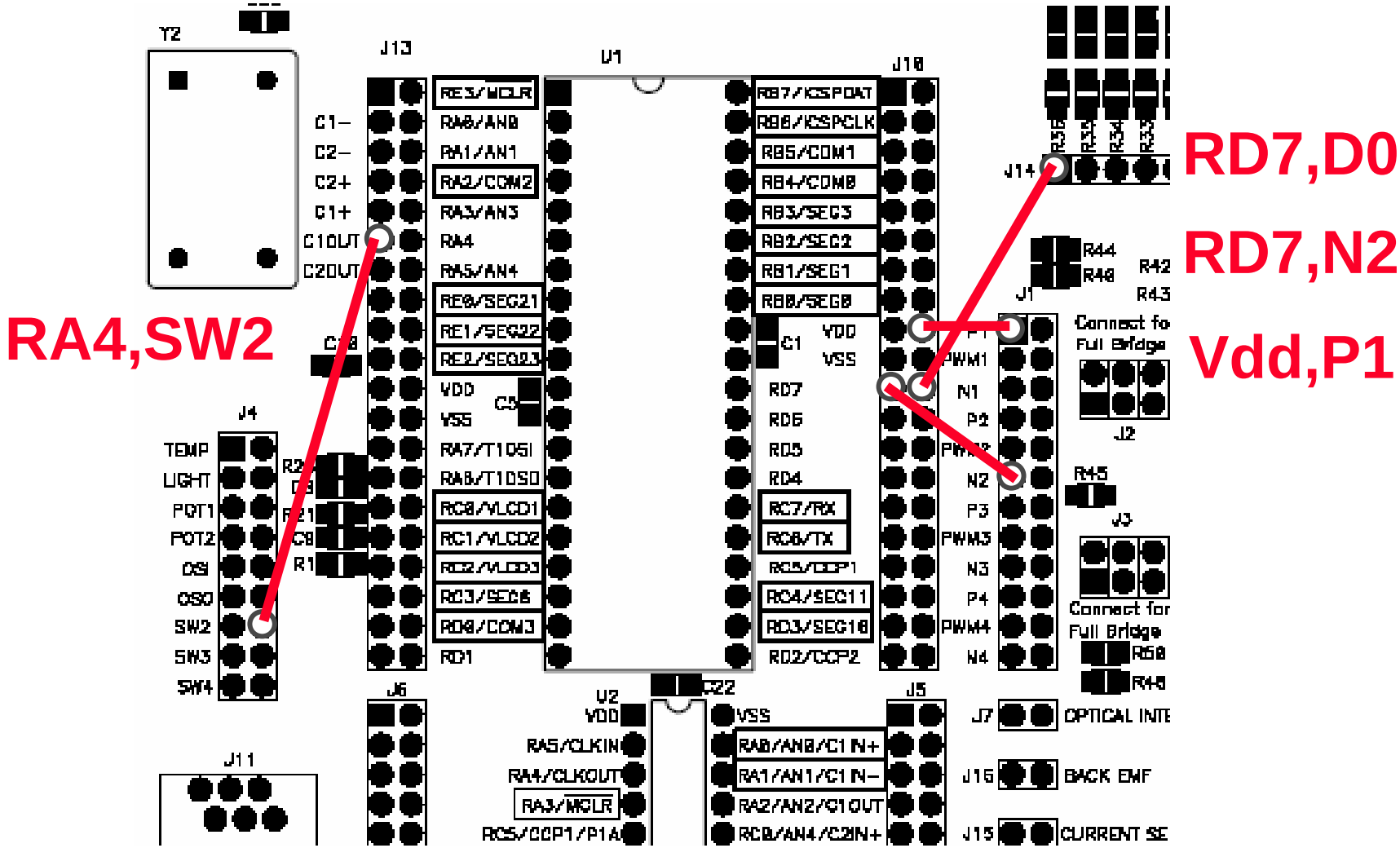


Brushed DC motor

Lab 1: Schematic

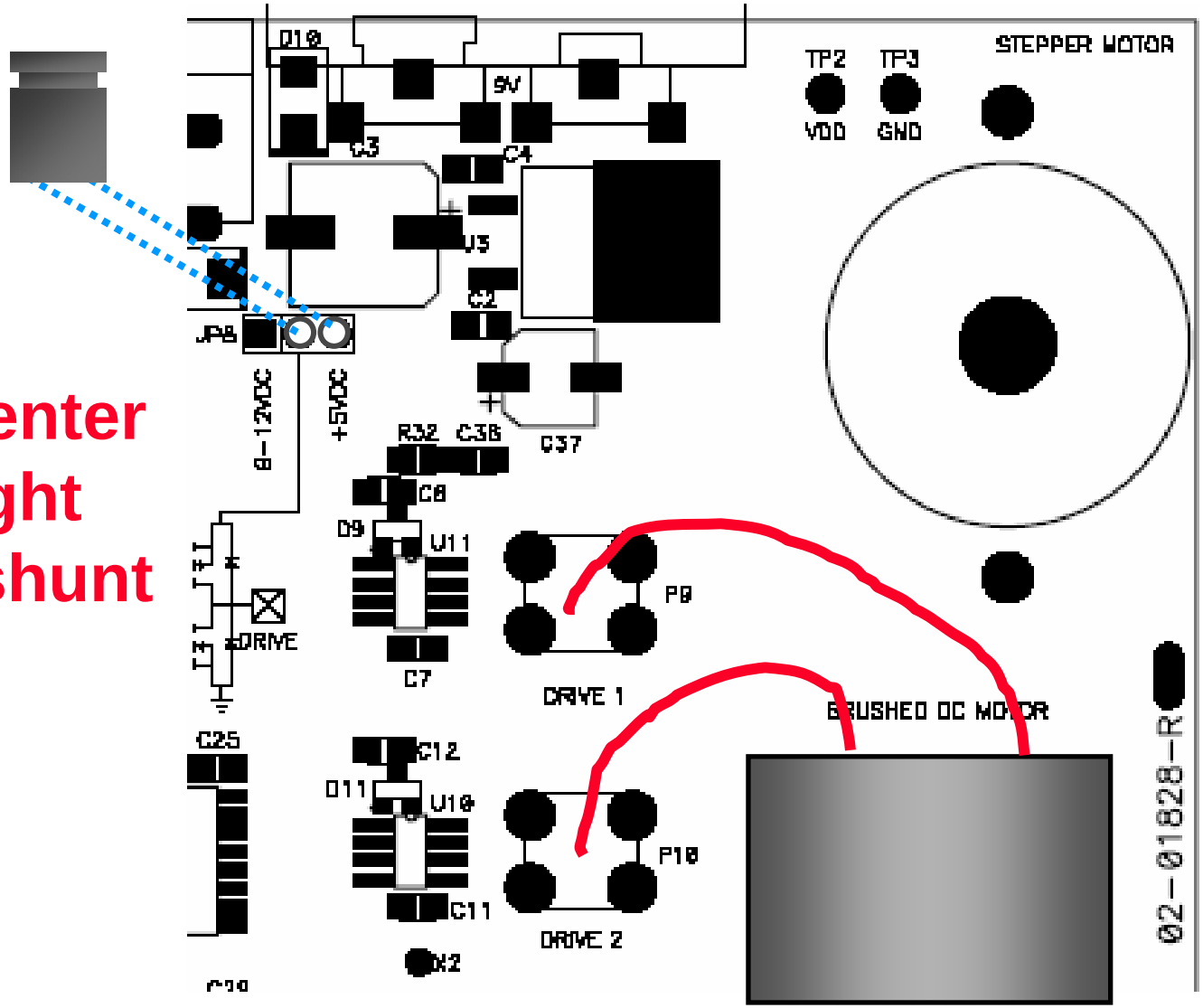


Lab 1: Connection Diagram



Lab 1: Connection Diagram

JP8: Center and Right using shunt



Timer 0

- 8-bit timer
- It continuously runs
- Overflow flag: INTCON,T0IF

Example:

Timer 0	INTCON,T0IF
11111110	0
11111111	0
00000000	1

Timer 0 Prescaler

- Timer 0 normally counts at the rate instructions are executed
- Timer 0 can be scaled using a “prescaler”
- Timer 0 will count once every ***P*** instructions, where ***P*** is the prescaler
- Prescaler: OPTION_REG<2:0>

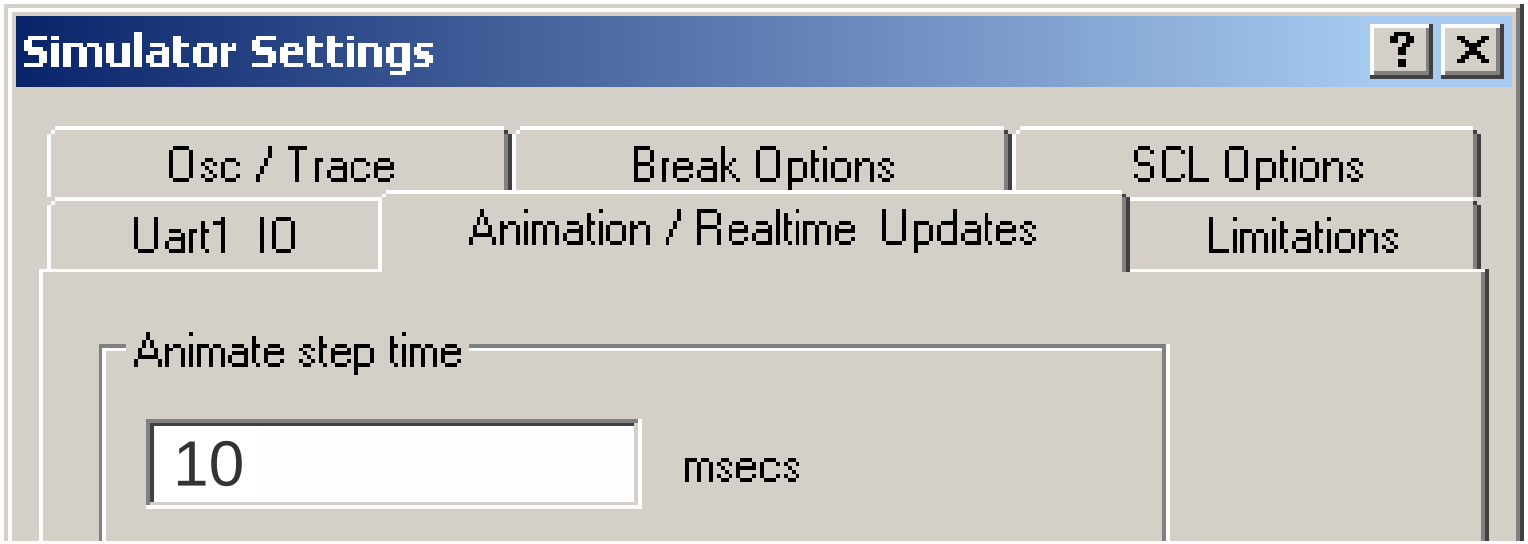
Timer 0 Hands-On

```
      include  <p16f917.inc>
      Counter  equ  0x20
Loop
      btfss    INTCON,T0IF
      goto     Loop
      bcf      INTCON,T0IF
      incf     Counter,f
      goto     Loop
      end
```

Create in h1.asm in the 'test' workspace

Set Animate Step Interval

- Choose Debugger – Select Tool – MPLAB SIM
- Choose Debugger – Settings
- Click on the Animation/Realtime Updates tab
- Change the Animate Step Time to 10 ms

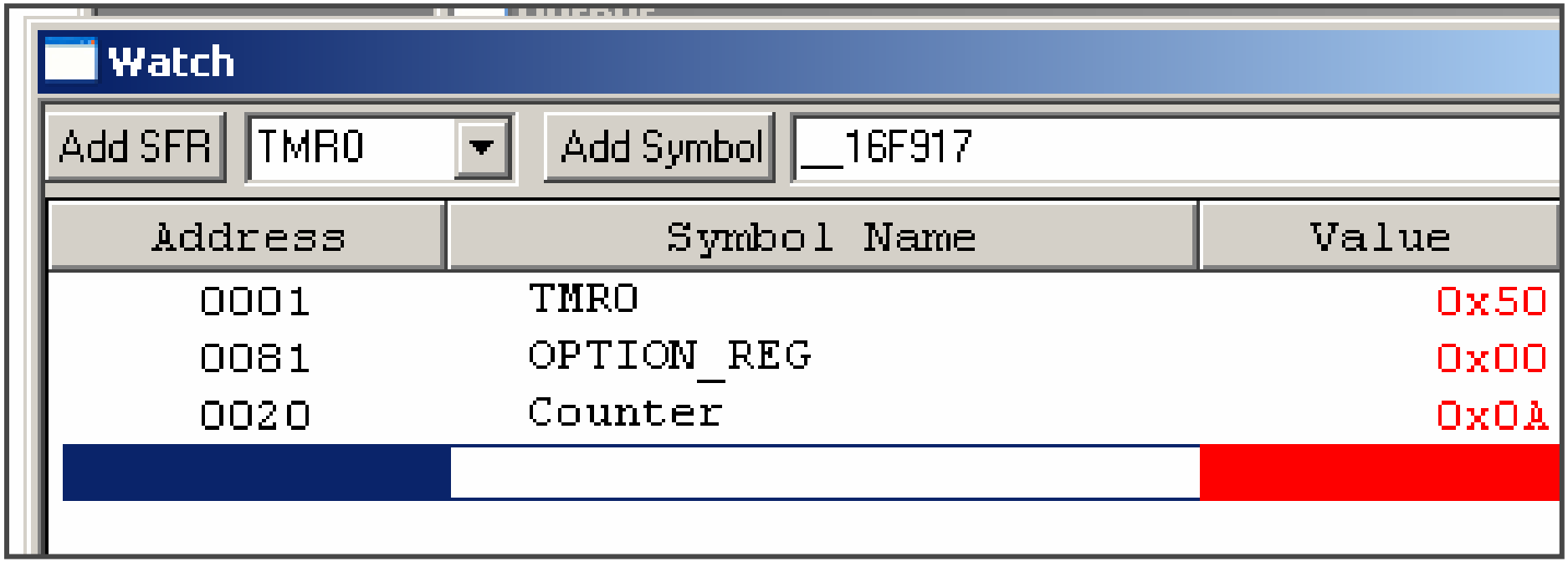


Build and Watch

- Build the project by pressing F10
- Create a Watch Window
- Watch TMR0, OPTION_REG, and Counter
- Enter “0” in the value for OPTION_REG

  (The prescaler is now 1:2, Timer 0 will count every 2 instructions)
- Choose Debugger – Animate

Watch Window



- Enter values between 0 and 7 in OPTION_REG

Relationship between Timer 0 and Instruction Cycle time

OPTION_REG bits 0 - 2	Instructions per one Timer 0 count	Rollover (4MHz)
0	2	512 µs
1	4	1.02 ms
2	8	2.05 ms
3	16	4.10 ms
4	32	8.20 ms
5	64	16.4 ms
6	128	32.8 ms
7	256	65.5 ms

Relationship between Timer 0 and Instruction Cycle time

OPTION_REG bits 0 - 2	Instructions per one Timer 0 count	Rollover (4MHz)
0	2	512 µs
1	4	1.02 ms
		2.05 ms
		4.10 ms
		8.20 ms
		16.4 ms
6	128	32.8 ms
7	256	65.5 ms

How many times must Timer 0 overflow with a prescaler of 1:256 before 2 seconds has passed?

‘decfsz’ Instruction

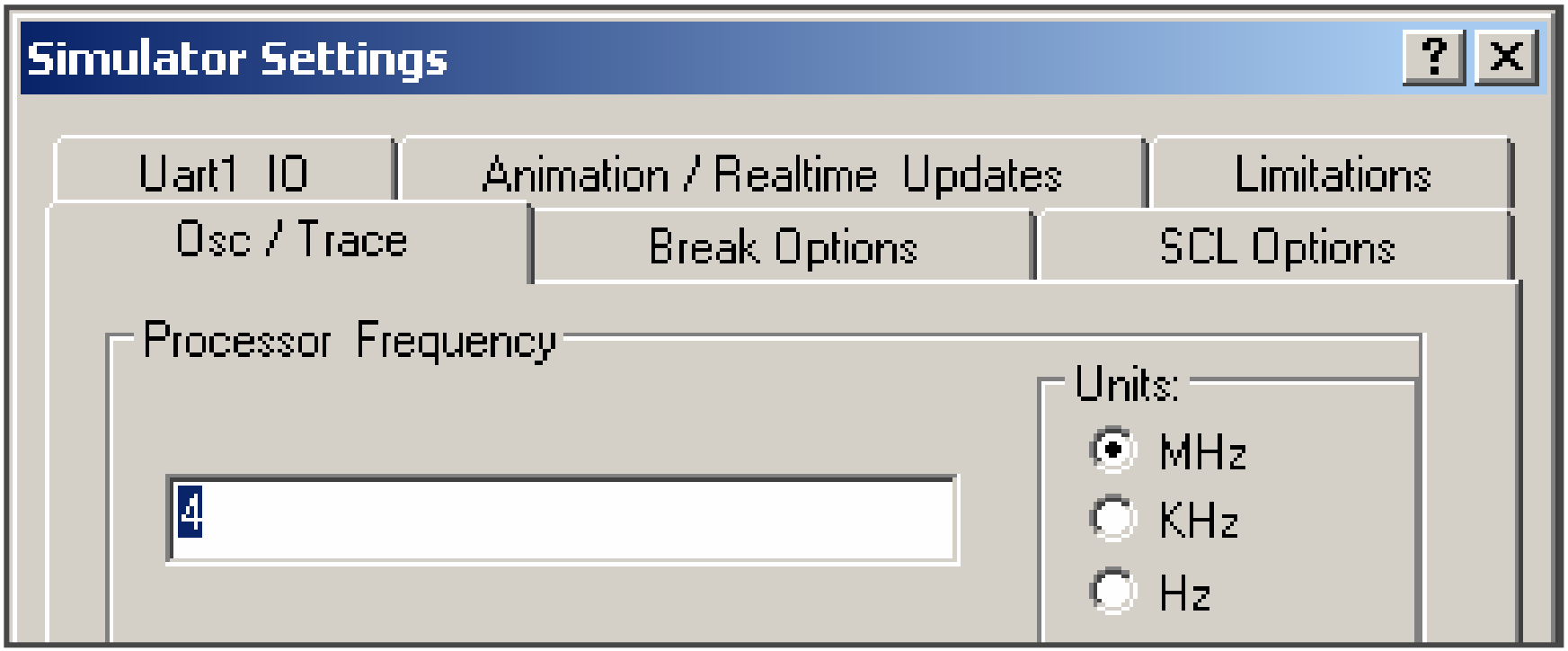
```
  clrf    TMR0
  movlw   .30
  movwf   Counter
Loop
  btfss   INTCON,T0IF
  goto    Loop
  bcf     INTCON,T0IF
  decfsz Counter,f
  goto    Loop
  end
```

Enter this code after:

Counter equ 0x20

Setup the Simulator Processor Frequency

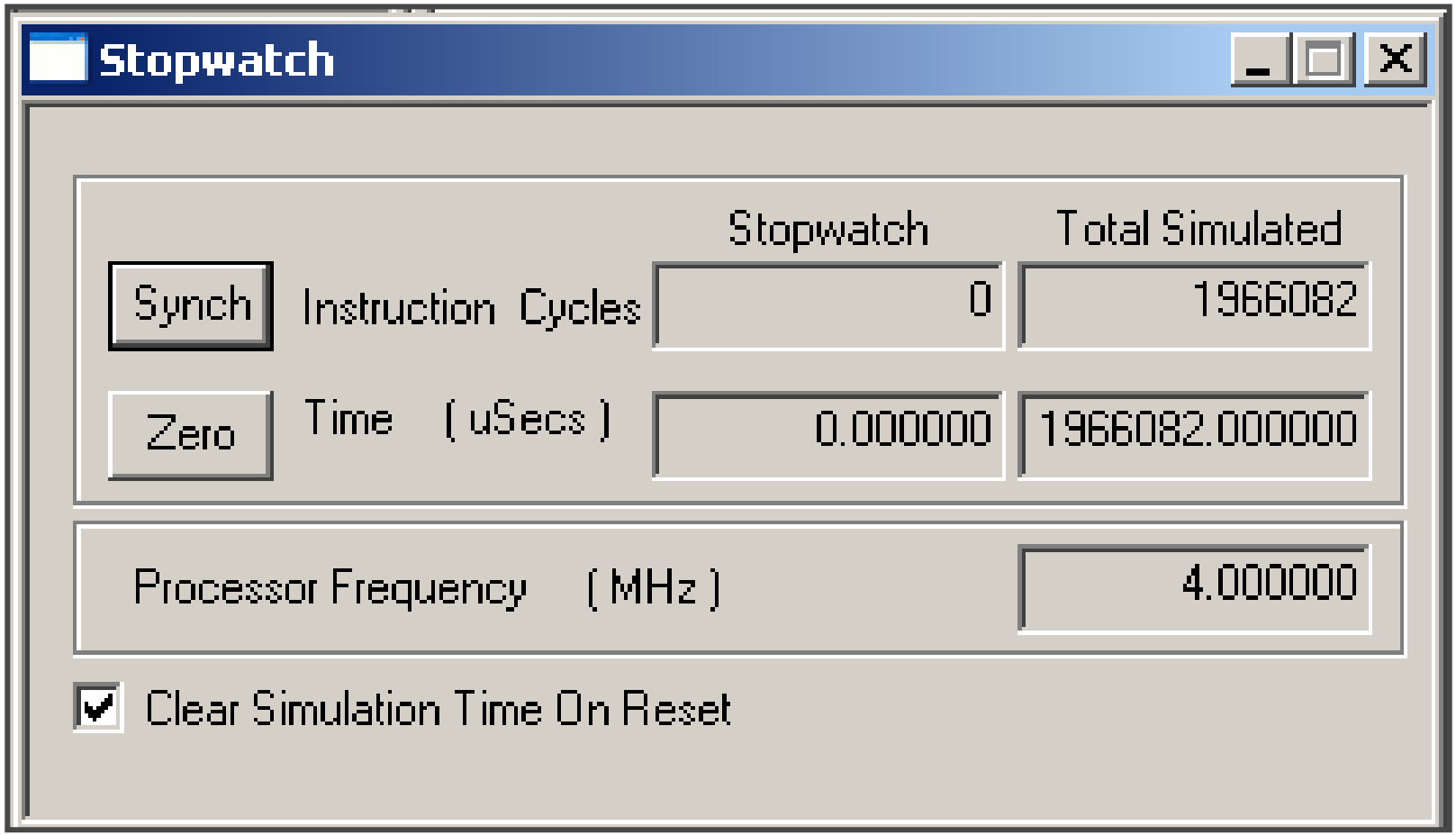
- **Debugger – Settings**
- Click on the **Osc/Trace** Tab
- Set the Processor Frequency to **4 MHz**



The MPLAB Stopwatch

- Debugger – Stopwatch
- Build the Project
- Change the OPTION_REG value to “7” in the watch window
- Run the simulator

The MPLAB Stopwatch



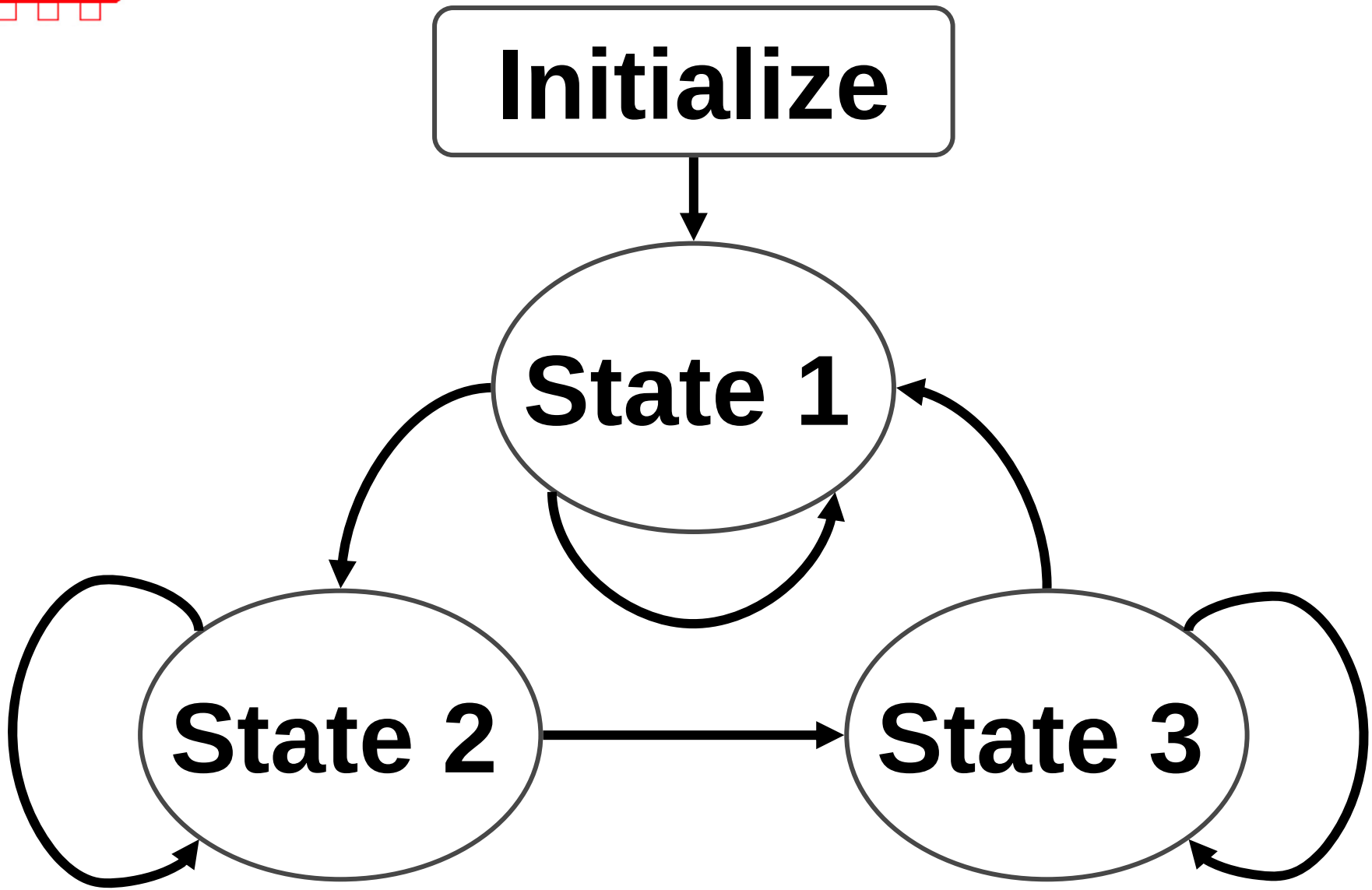
Timer 0 Review

- Timer 0 is an ____ - bit Timer.
- The prescaler for Timer 0 is in the _____ register.
- Increasing the prescaler **speeds up** or **slows down** Timer 0?
- The _____ instruction is great for counting down the number of Timer 0 overflows.

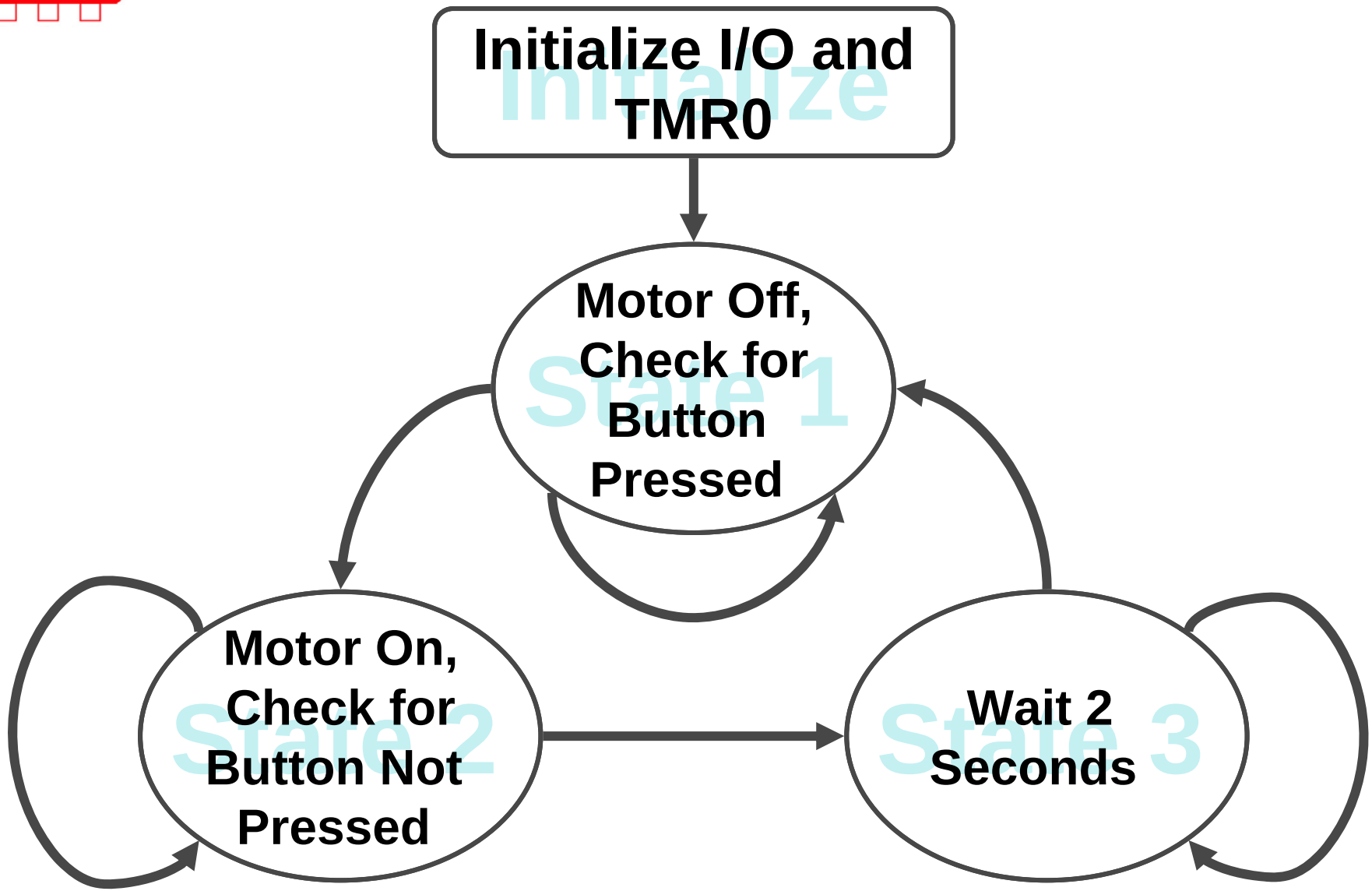
Lab 1: Design Objectives Review

- Read the state of SW2 (Thermal Breaker)
- When SW2 is pressed turn on the brushed DC motor (Fan)
- When SW2 is released, wait 2 seconds, then turn off the motor

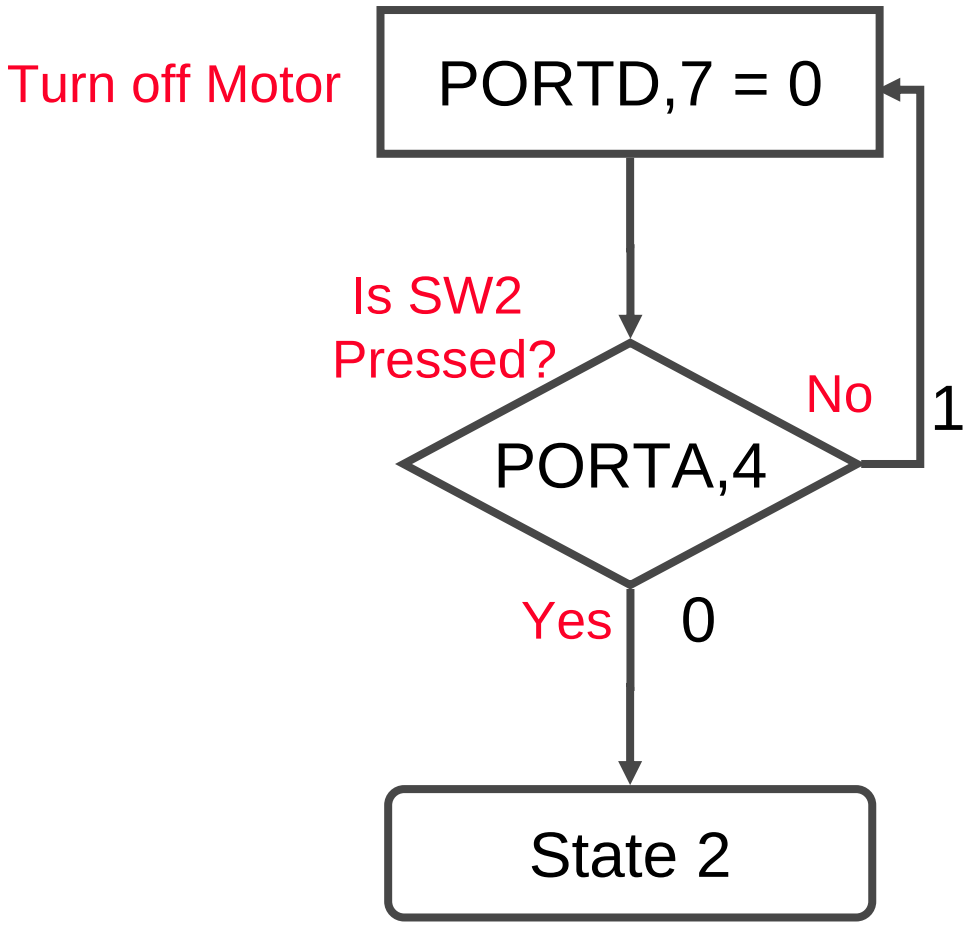
Converting Design Objectives to Program States



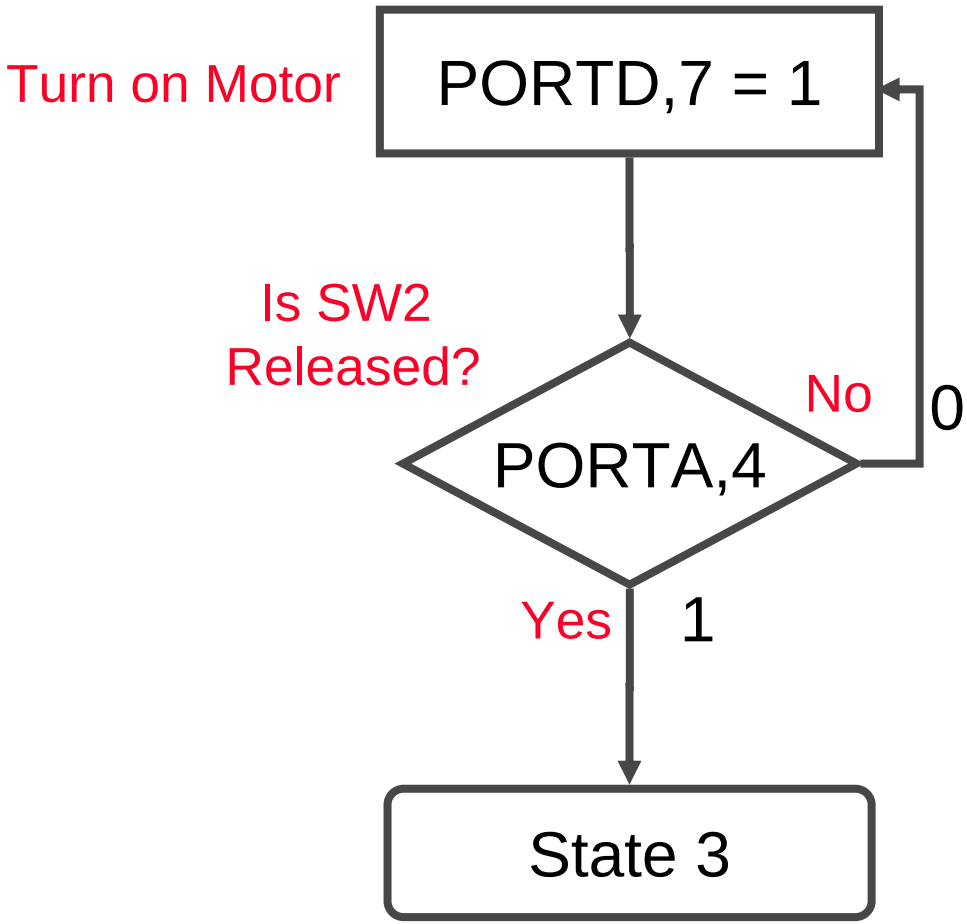
Lab 1: Your Task



State 1: Wait for Button Press



State 2: Wait for Button Release



State 3: Two Second Delay

- You just wrote this code!

Lab 1: Implement a Solution

- Open the workspace in:

  C:\Mechatronics WIB\Lab1

- Open Lab1.asm in the project folder
- All I/O ports and the OPTION_REG are initialized for you.
- Fill in the code for the three states under the corresponding labels.
- **You must press SW5 to enable the drive circuit!**

Lab 1: Review

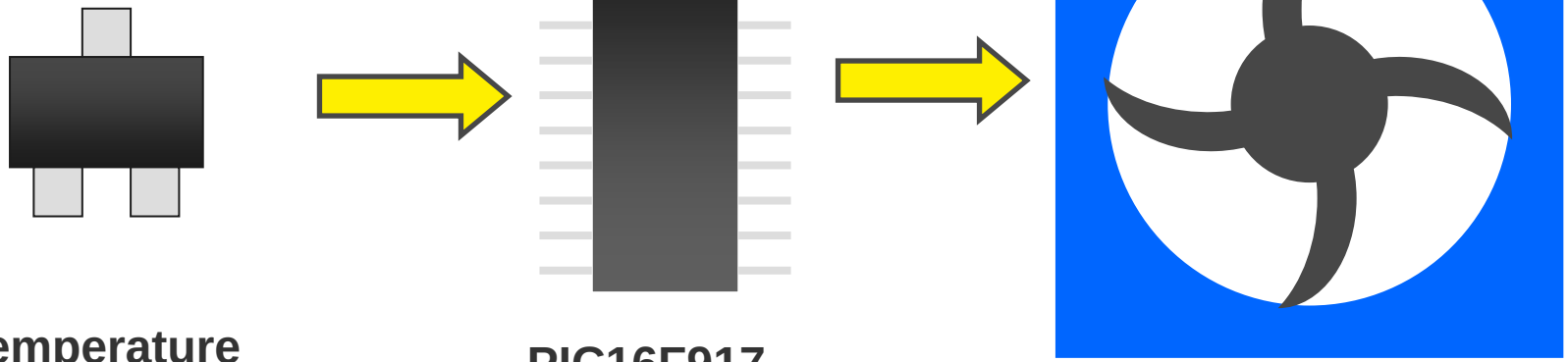
- New PIC16F917 features and registers
  - Timer 0 (TMR0)
  - OPTION_REG
- Tools
  - Stopwatch
  - Simulator: animate interval, processor frequency
- Instructions
  - decfsz
- Concepts
  - Programming with states

Agenda

- Introduction to Microchip
- Mechatronics examples and benefits
- PIC® Microcontroller Basics
  - Hands-On Learning Cycles
- Labs
  - Simple I/O and Timer 0
  - Reading an analog sensor, LCD module
  - Controlling the speed of a motor
- Resources

Lab 2: Design Objectives

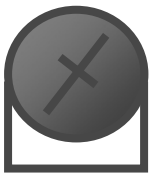
- Read an analog temperature sensor
- Turn on the fan when the temperature > 40ºC
- Turn off the fan when the temperature <= 40ºC
- Display the temperature



Lab 2: Hardware

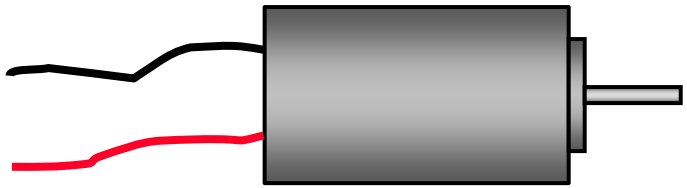
- A potentiometer on the board will be used to produce the varied output voltage
- The Brushed DC motor will substitute for the fan
- The LCD will be used to display the analog-to-digital Converter output

POT1



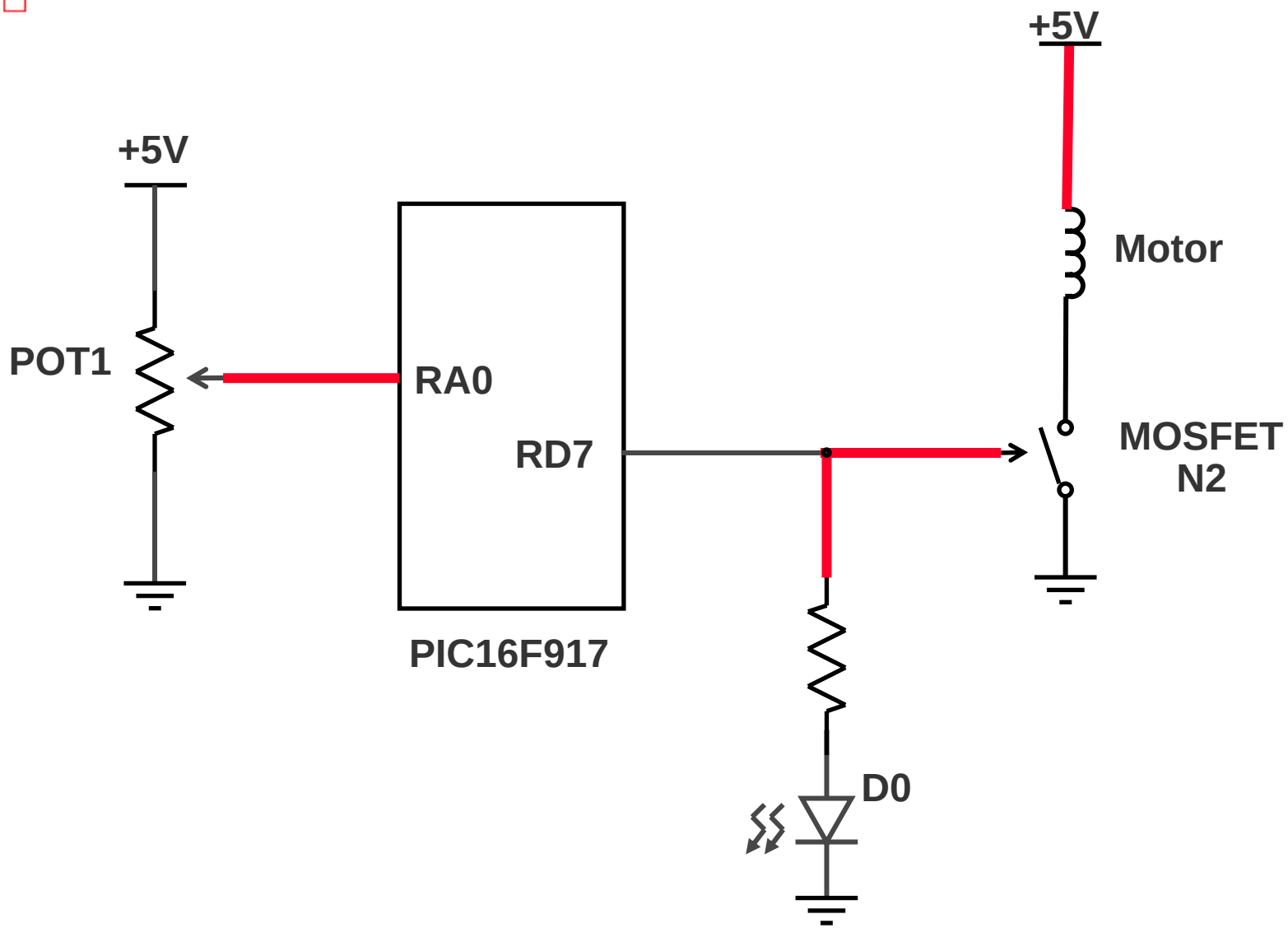
Potentiometer

LCD



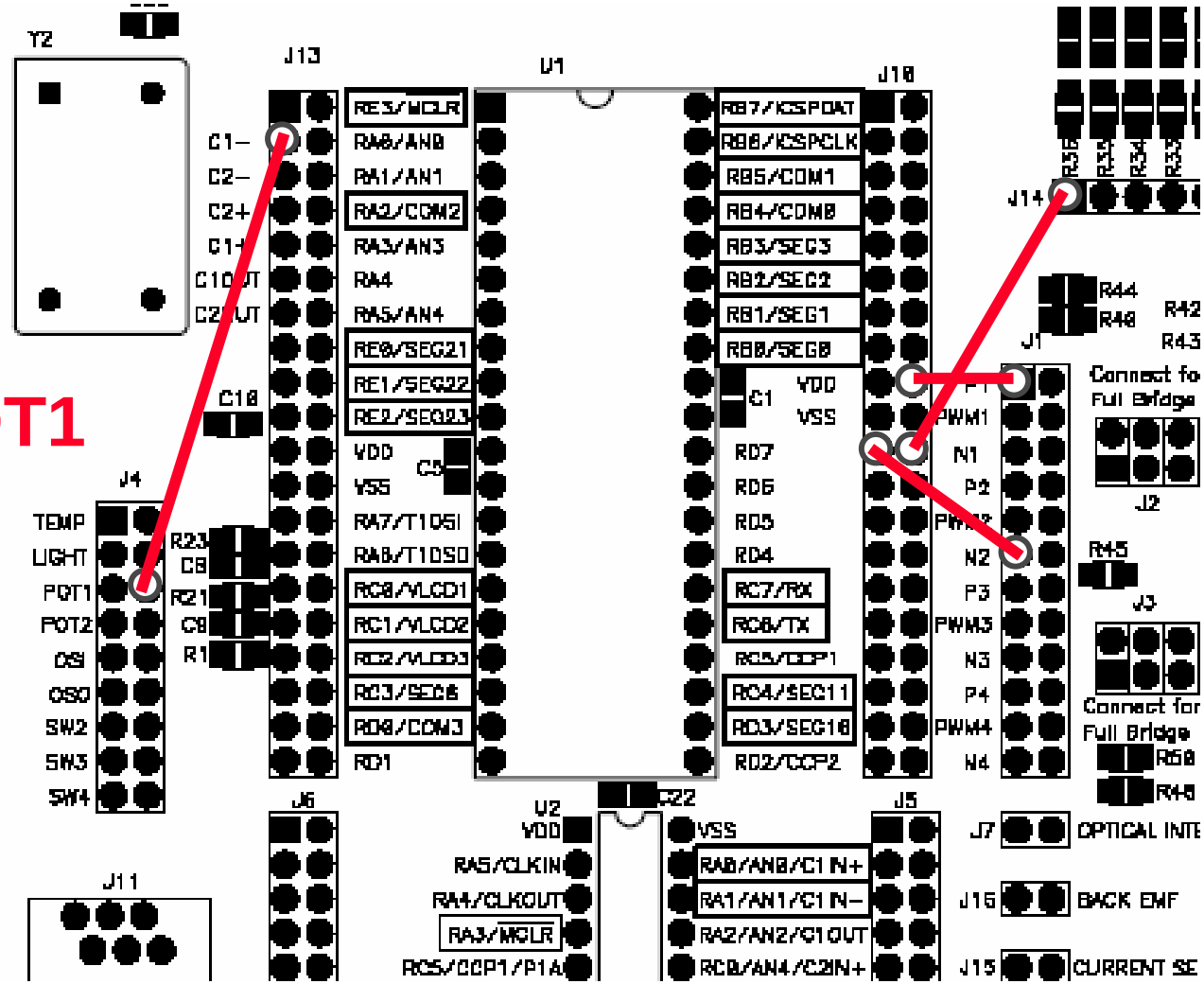
Brushed DC motor

Lab 2: Schematic



Lab 2: Connection Diagram

RA0,POT1



Analog-To-Digital Converter

- Converts an analog voltage level into a binary number
- Resolution: How many different readings are possible
  - 8-bit -> 2^8 -> 256 possible readings
  - 10-bit -> 2^{10} -> 1024 possible readings

Example: 8-bit resolution over 5 volts:

Voltage resolution = 5V/256 = 19.5 mV per count

ADC Registers

- ANSEL – Select analog input pins
- ADCON1 – Select the conversion clock
- ADCON0 – Select analog channel, turn the module on, do the conversion
- ADRESL – Low-byte of conversion value
- ADRESH – High-byte of conversion value

A/D Conversion Steps

1. Turn on the analog module
2. Set GO_DONE bit of ADCON0
3. Wait for GO_DONE bit to be cleared
4. Read A/D result in ADRESH

ADC Hands-On

```
  include <p16f917.inc>
  bsf   ADCON0,ADON
LoopOuter
  bsf   ADCON0,GO_DONE
LoopInner
  btfsc ADCON0,GO_DONE
  goto  LoopInner
  movf  ADRESH,w
  goto  LoopOuter
  end
```

Create in h1.asm in the 'test' workspace

Looking at the ADC module

- Select the MPLAB ICD 2 as the debugger
- Create a watch window
- Add ADRESH to the watch window
- Build the project
- Program the MPLAB ICD 2
- Click animate

What happens to ADRESH when you turn POT1?

ADC Module Review

- The analog to digital module converts an _____ into a _____ .
- The _____ register is used to turn the module on and start an A-to-D conversion.
- The _____ register contains the high order bits of the ADC value.
- You must wait for the _____ bit to clear before reading the ADC value.

Displaying Values on the LCD

- The LCD is driven directly from the LCD module on-board the PIC16F917
- The LCD is a 3½ digit display
- Pins that are connected to the LCD are indicated by a white box enclosing the pin designation
- LCD functions are provided in linkable files for your convenience

Linking Files in a Project

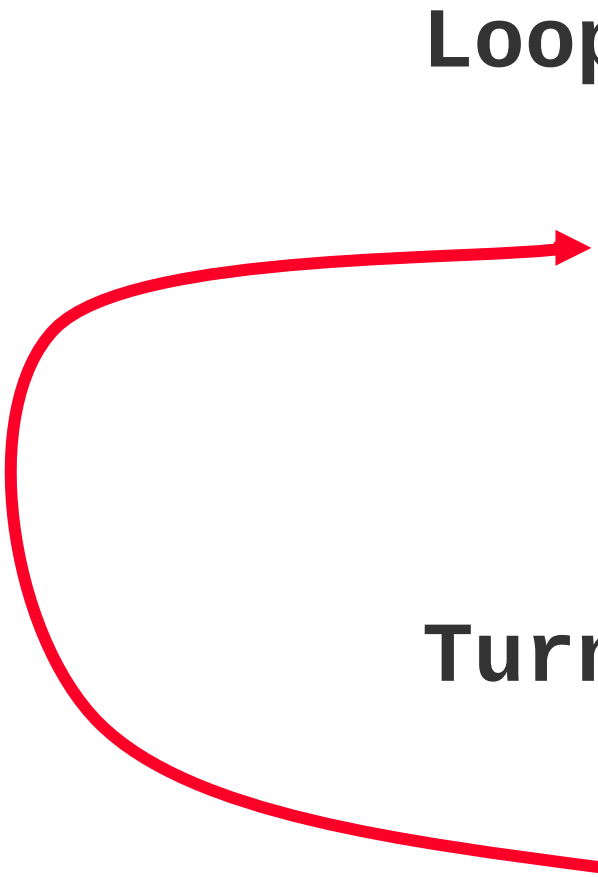
- Linking is used because:
  - It is convenient to break up lengthy code into more than one file
  - It helps make code more portable from one application to another
  - It helps make code easy for others to use

‘call’ Instruction

```
Loop
    call    TurnOnRA4
    movlw   0x78
    movwf   PORTD
    goto    Loop

TurnOnRA4
    bsf     PORTA,4
    return
```

‘call’ Instruction



```
Loop
        call    TurnOnRA4
        movlw   0x78
        movwf   PORTD
        goto    Loop

TurnOnRA4
        bsf     PORTA,4
        return
```

DisplayHex and DisplayDecimal Functions

Example 1:

```
movlw   0xFF
call    DisplayDecimal
```

Example 2:

```
movlw   0xFF
call    DisplayHex
```

LCD Hands-on

- File – Open Workspace…
  - c:\Mechatronics WIB\Lab2\Lab2.mcw
- The code from the last Hands-on is duplicated in the Lab2.asm file
- After the line:

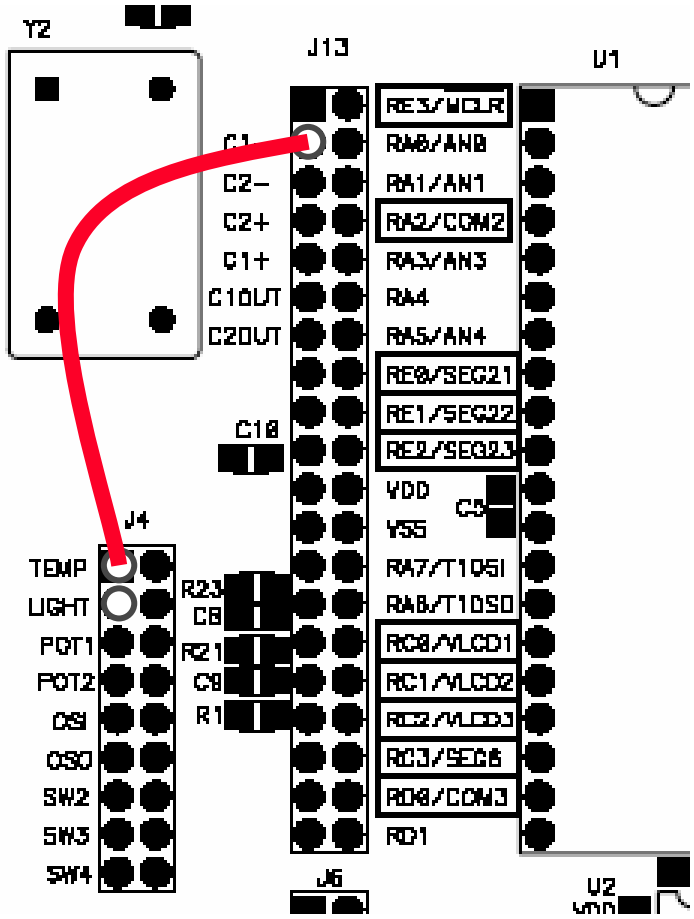
  movf ADRESH, W

  Call one of the following:

  `DisplayDecimal` or **`DisplayHex`**
- Build and animate the code using the ICD 2

Experimentation

- Light Sensor
  - Move the jumper wire from POT1 to LIGHT
  - Vary the light to the light sensor
- Temperature Sensor
  - Try moving the same wire to TEMP
  - Breath heavily on the temperature sensor



Comparing Two Numbers

- Recall these design objectives:
  - Turn the motor (fan) on when the result of the ADC (temperature) is greater than 40
  - Turn off the motor when the result of the ADC is equal to or less than 40
- This requires comparing two numbers
- Solution: Use the ‘sublw’ instruction

‘sublw’ Instruction

- Subtracts W from a literal and places the result in W
- Carry and Zero STATUS flags affected

Example: (Setpoint – Temp)

```
movlw    Temp
sublw    Setpoint
```

Relation	Result	STATUS,Z	STATUS,C
Setpoint > Temp	+	0	1
Setpoint = Temp	0	1	1
Setpoint < Temp	-	0	0

‘sublw’ Instruction

What is the result of the following operation?

```
movlw    4
sublw    5
```

Using ‘sublw’ for Greater Than

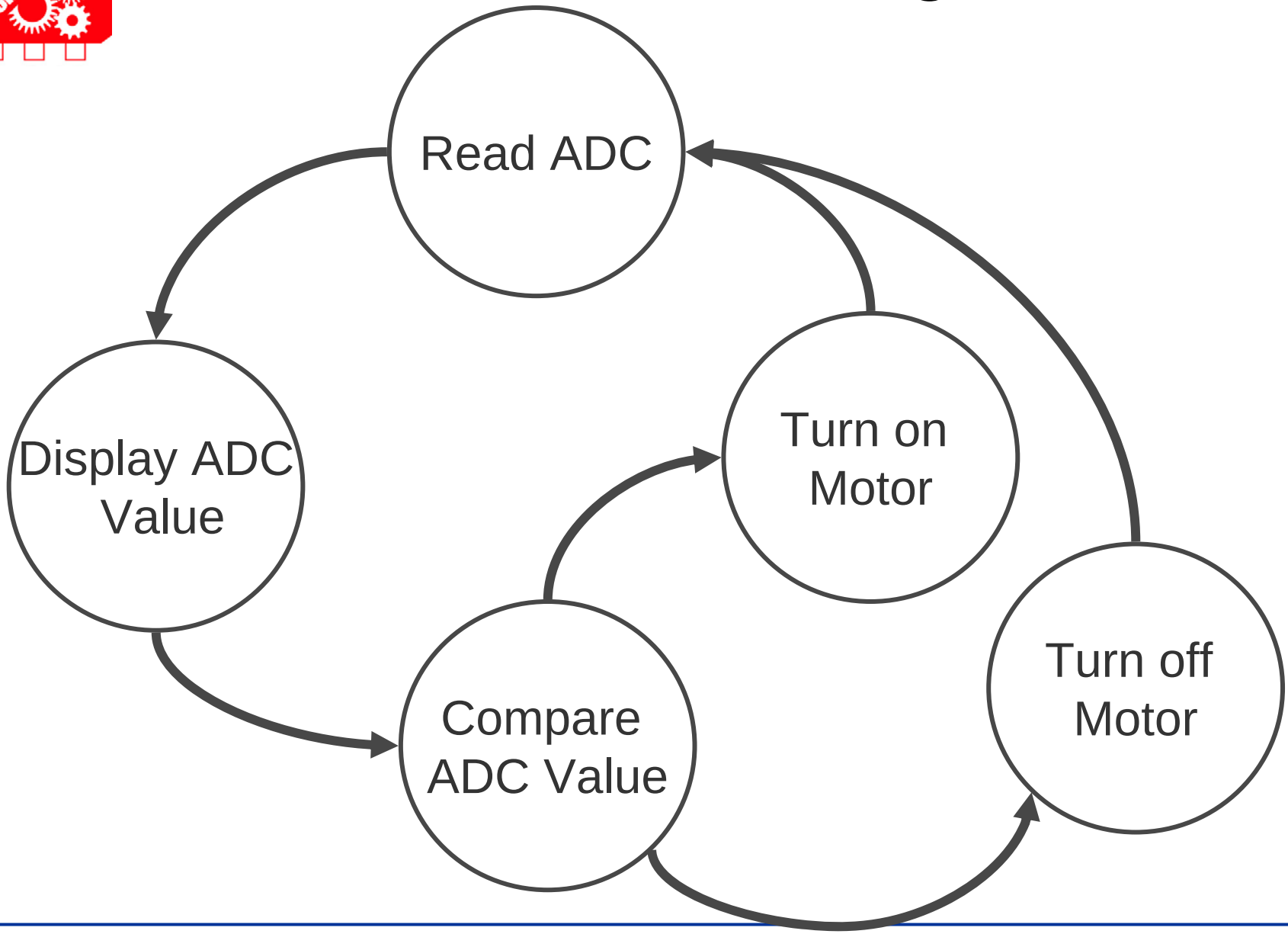
Problem:

You want to find out if the value in ADRESH is greater than 40.

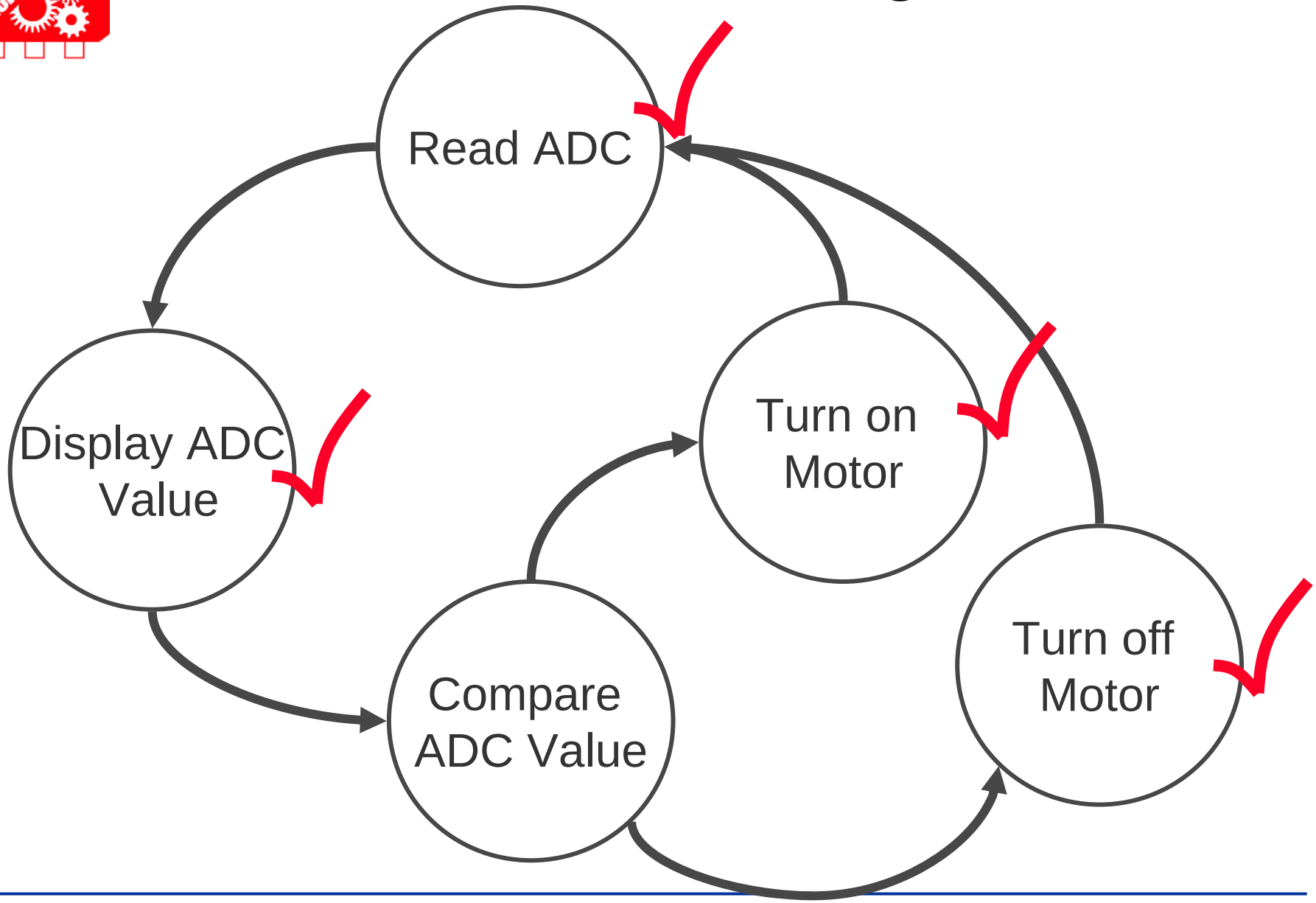
Solution:

1. Move the value in ADRESH to W
2. Subtract W from .40
3. Check the Carry flag
   - If clear, then the value is greater than 40
   - If set, then the value is less than or equal to 40

Lab 2 State Diagram



Lab 2 State Diagram



Lab 2: Implement a Solution

- Add to your existing code for Lab2
- Use “sublw” to test whether the ADC value is 40 or greater
- Check the STATUS,C flag
- If clear, turn on the motor
- If set, turn off the motor

Lab 2 Review

- New PIC16F917 Peripherals
  - Analog to Digital Conversion Module
  - LCD Module
- New Tools
  - MPLAB Linker
- New Instructions
  - **`call`** and **`return`**
  - **`sublw`**
- New Concepts
  - Comparing two numbers

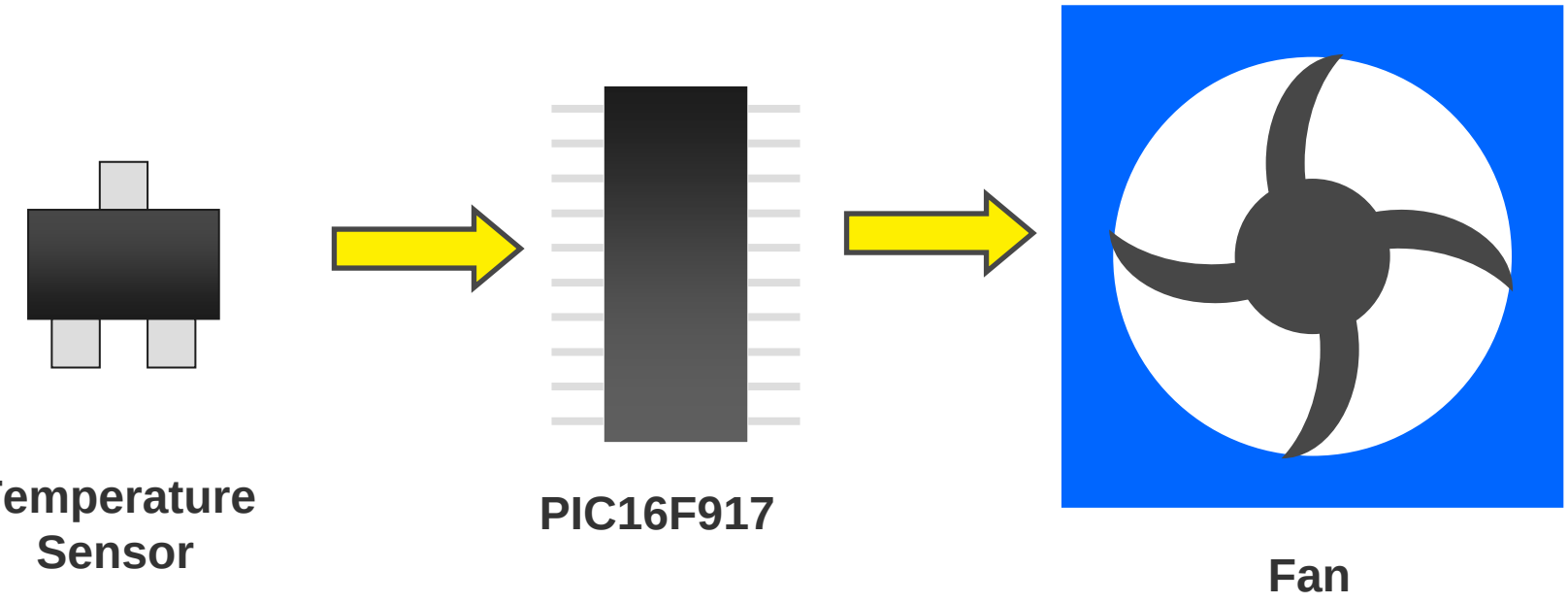
Agenda

- Introduction to Microchip
- Mechatronics examples and benefits
- PIC® Microcontroller Basics
  - Hands-On Learning Cycles
- Labs
  - Simple I/O and Timer 0
  - Reading an analog sensor, LCD module
  - **Controlling the speed of a motor**
- Resources

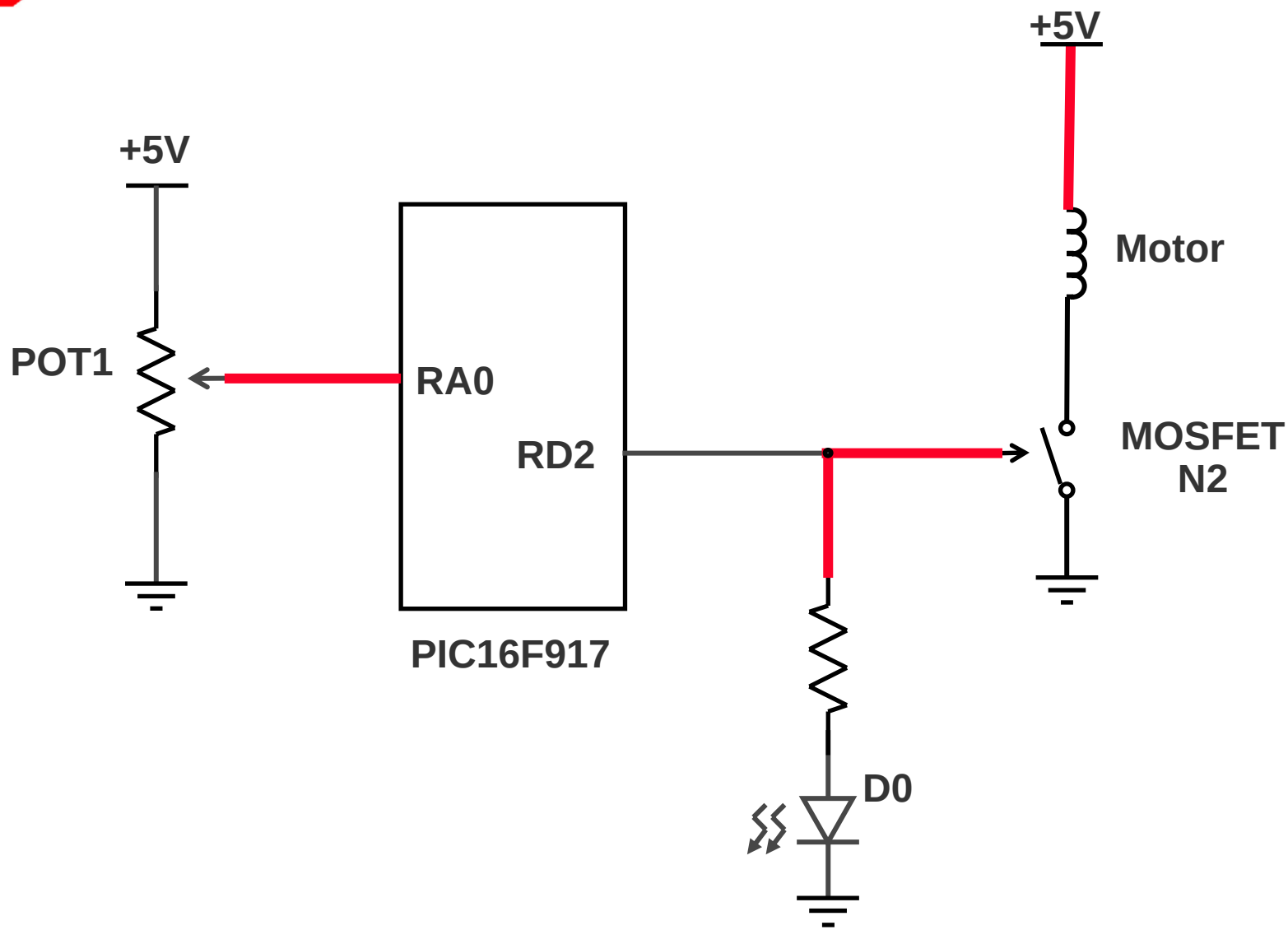
LAB 3

Lab 3: Design Objectives

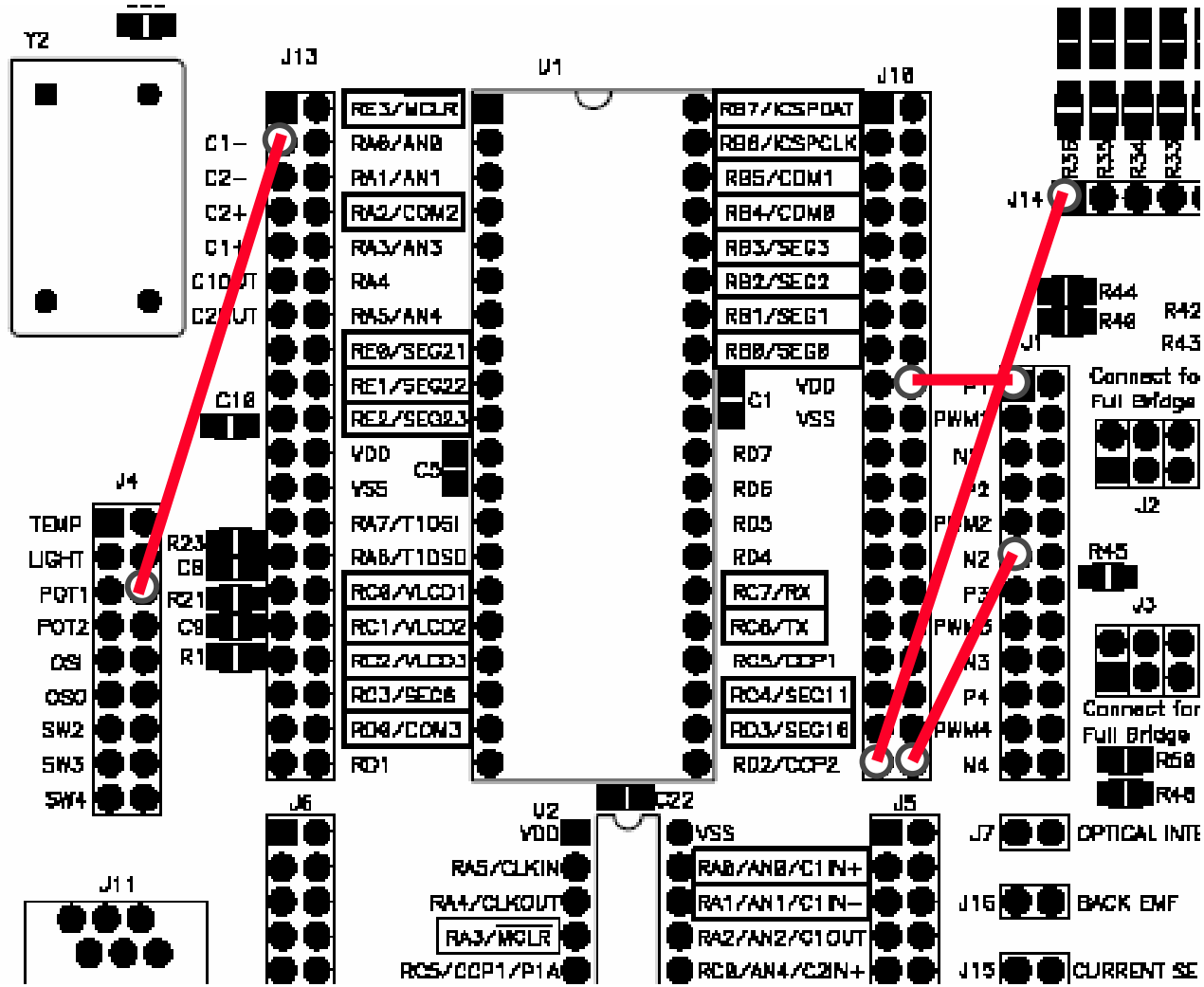
- Read an analog temperature sensor
- Vary the speed of the fan in proportion to the temperature reading



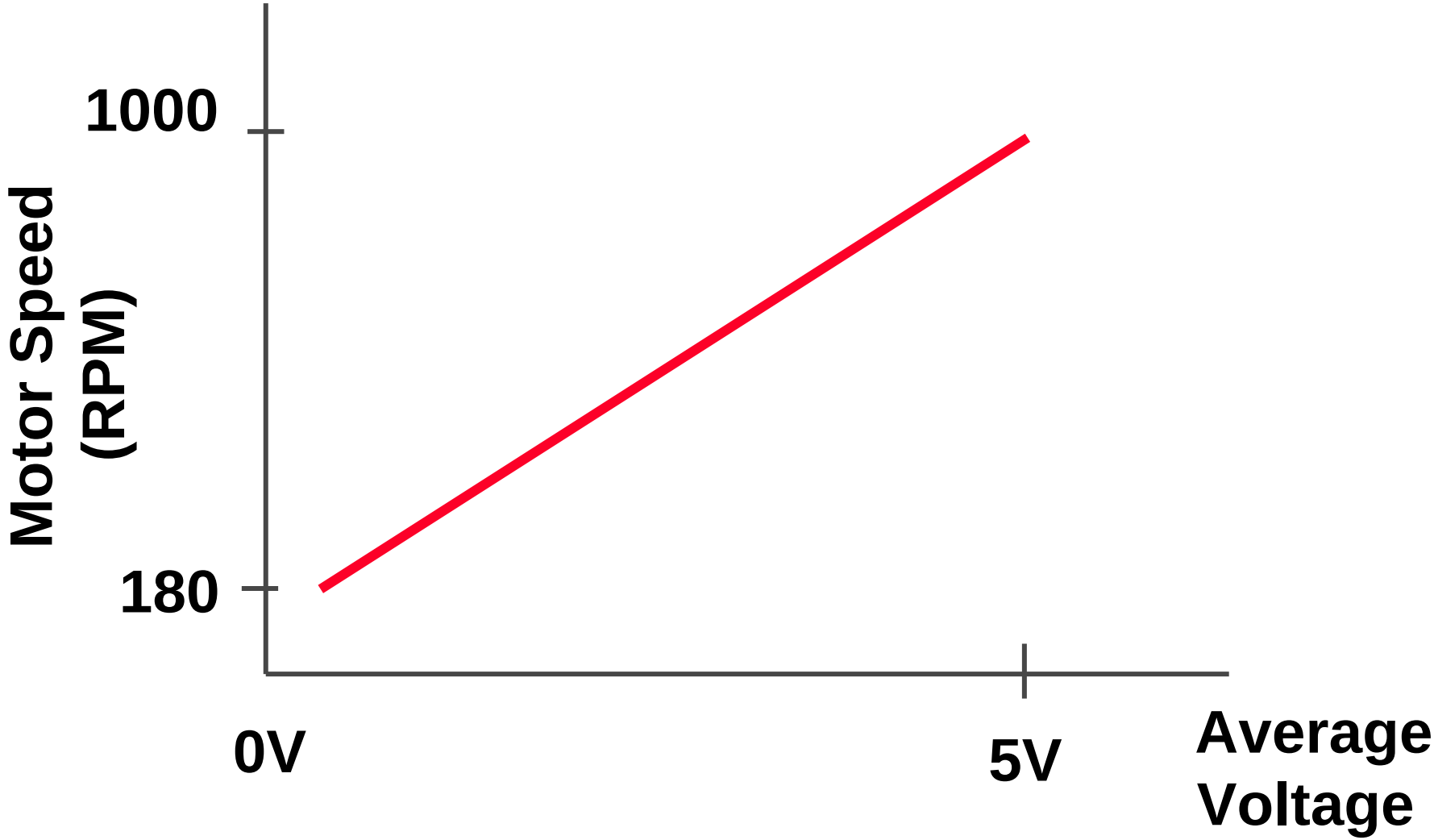
Lab 3: Schematic



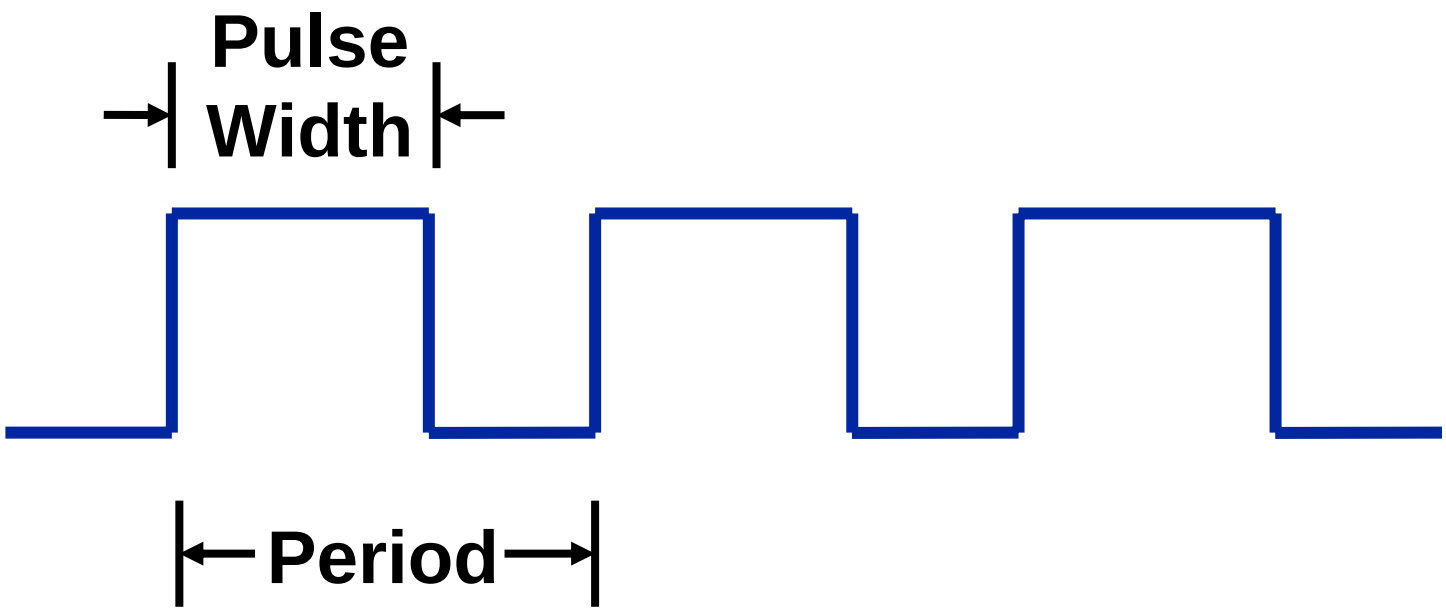
Lab 3: Connection Diagram



Brushed DC Motor Speed Control

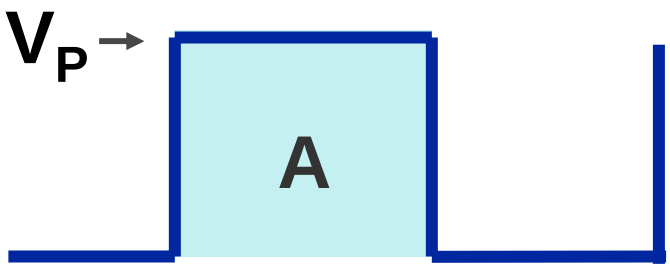


Pulse Width Modulation



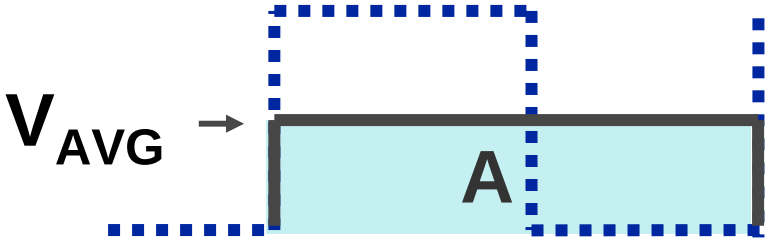
$$\text{Duty Cycle} = \frac{\text{Pulse Width}}{\text{Period}}$$

PWM and Average Voltage

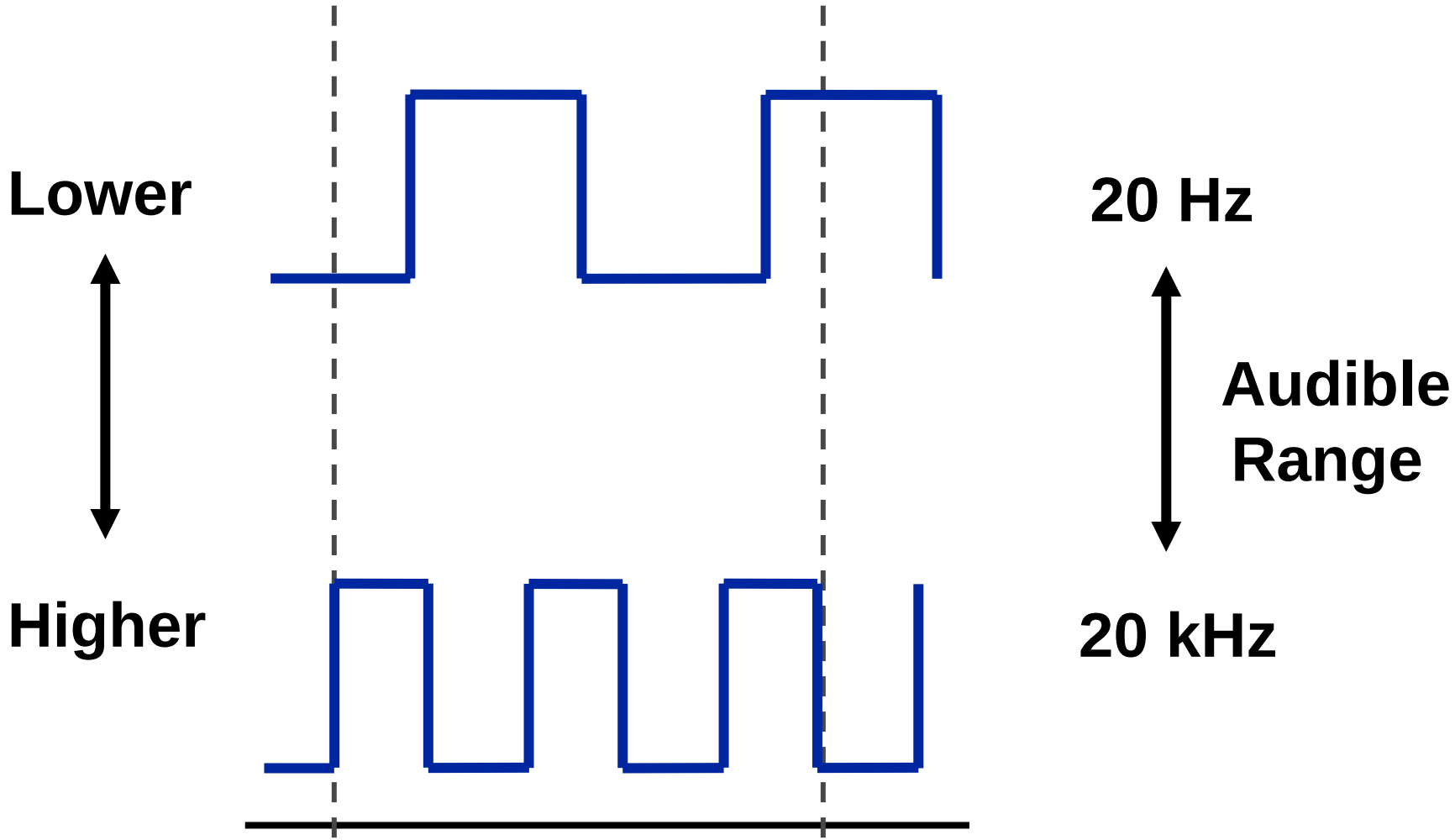


A is the area

V_{AVG} = Duty Cycle x V_P



Frequency



PWM Hands-On

Experimenting with Duty Cycle

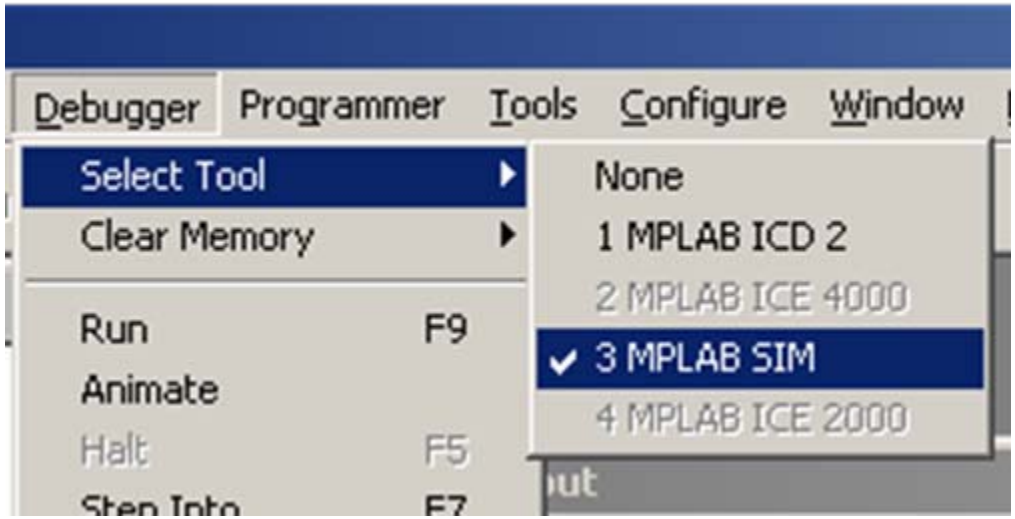
PWM Hands-On

```
    include <p16f917.inc>
    bsf     STATUS,RP0
    bcf     TRISD,2
    bcf     STATUS,RP0
PWMLoop
    bsf     PORTD,2
    nop
    nop
    .
    .
    bcf     PORTD,2
    goto    PWMLoop
    end
```

Create in h1.asm in the 'test' workspace

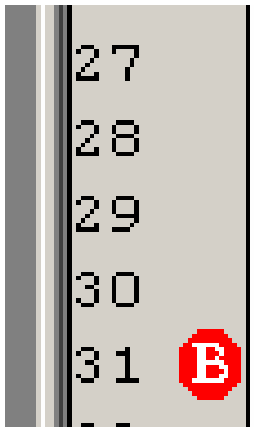
Copy these three lines and paste here 6 times

Simulating the PWM Loop



1. Choose MPLAB SIM
2. Build the project

3. Set a Breakpoint at "goto PWMLOOP"

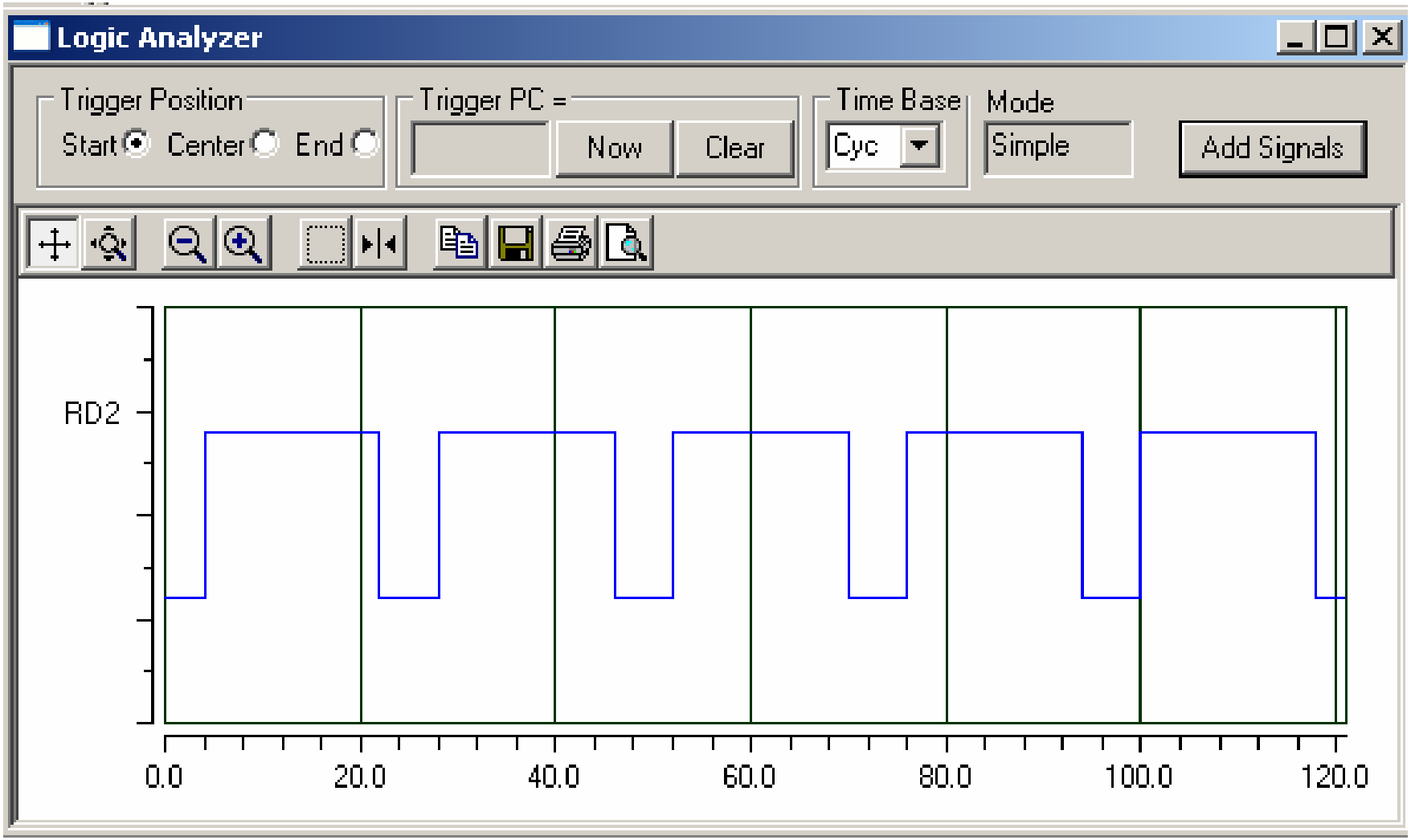



```
bcf     PORTD,2
nop
nop
bcf     PORTD,2
goto    PWMLoop
```

Set Up the Logic Analyzer

1. Choose **View - Simulator Logic Analyzer**
2. Select “Add Signals”
3. Choose “RD2” in the available Signals Box and click “Add =>”
4. Run the simulator (press F9)
5. Run again (press F9)
6. And again (press F9)

Simulate using the Logic Analyzer



Change the Duty Cycle

- Starting from the bottom change ‘bsf’ instructions to ‘bcf’ one at a time
- Build the project and simulate between changes

  Do you see the change in duty cycle?
- Remove the breakpoint
- Switch to the ICD 2 as the debugger
- Do the same experiment as above

  How does the motor behave to ‘bsf’ changes?

PWM Review

- PWM stands for _____ _____ _____.
- PWM signals have a fixed _____.
- The duty cycle is the ratio between _____ and _____.
- The human ear can hear frequencies below ____ kHz.
- Which frequency is higher (right or left)?

Source Code Generated PWM Drawbacks

- Processor intensive
- PWM Loop must be continually executed
- Little time to accomplish other tasks

Possible Solutions:

1. Slow down the PWM
2. Decrease PWM resolution
3. Use MCU with PWM peripheral ✓

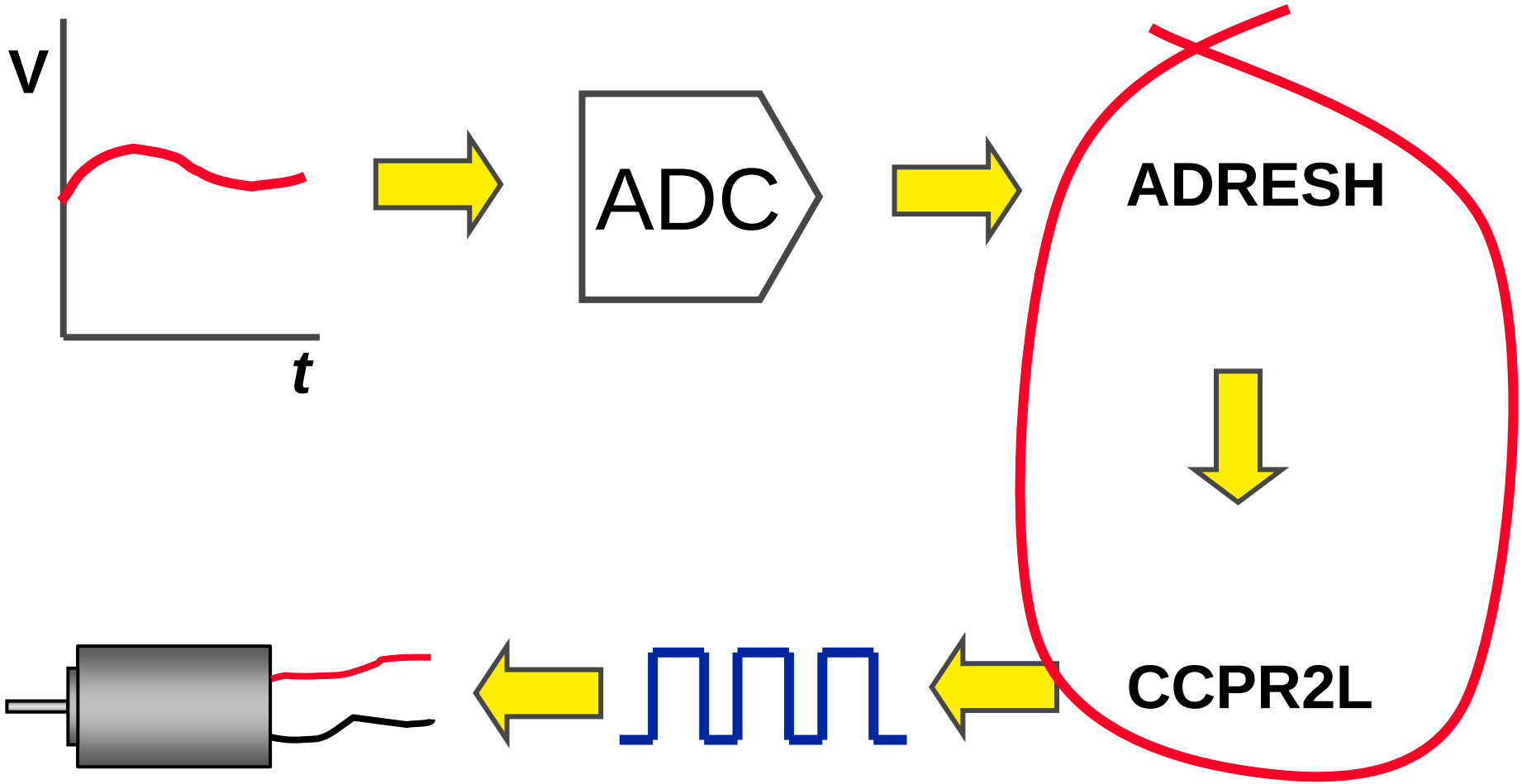
Capture, Compare, PWM (CCP) Module

- The CCP module is a hardware PWM signal generator
- Timer 2 is used to clock the PWM signal
- Up to 10-bits of resolution (1024 pulse width settings per period)
- At 8-bits of resolution the 8 MHz internal clock will output 31.2 kHz
- CCP2 is multiplexed with RD2

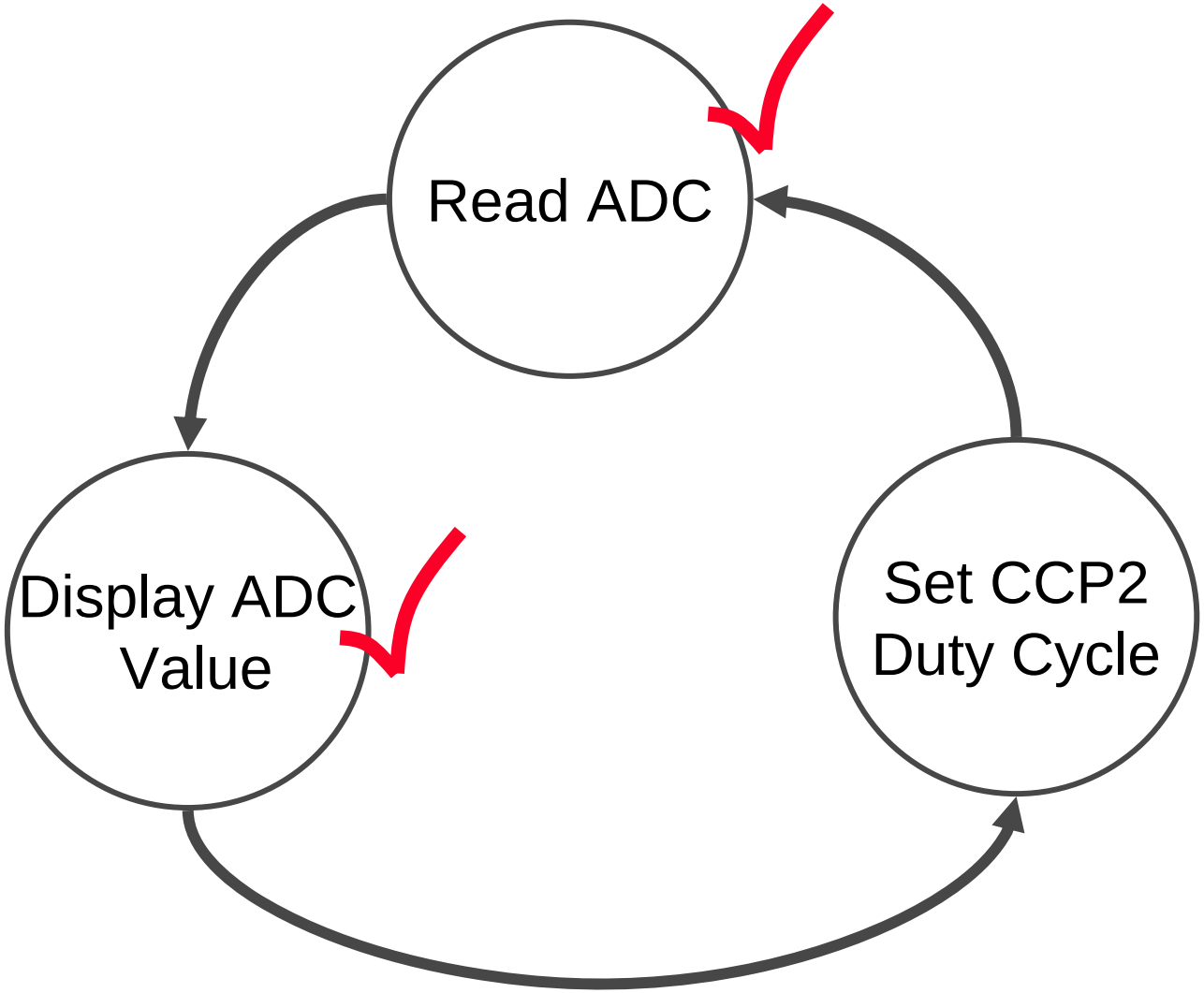
CCP2 Registers

- CCP2CON – Select PWM mode, 2 LSB of Duty Cycle
- CCPR2L – 8 MSB of Duty Cycle
- PR2 – Frequency select register
- T2CON – Turn on Timer 2
- TRISD – Make CCP2 pin an output

Lab 3: Using the CCP



Lab 3 State Diagram



Lab 3: Implement a Solution

- Open the Lab3 workspace in:

  c:\Mechatronics WIB\Lab3

- All I/O ports and modules (including the CCP) are initialized for you
- Copy the first two states in Lab 2
  - Read ADC
  - Display ADC Value
- Move result of the analog-to-digital conversion into CCPR2L

Motor Whine

Question: Why does the motor whine?

‘QuietPWM’ Function

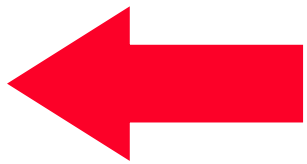
- Generates a 31.2 kHz signal using the 8 MHz internal oscillator
- Has 8-bits of resolution
- How to use:
  - Move duty cycle value in W
  - Call QuietPWM

Lab 3 Review

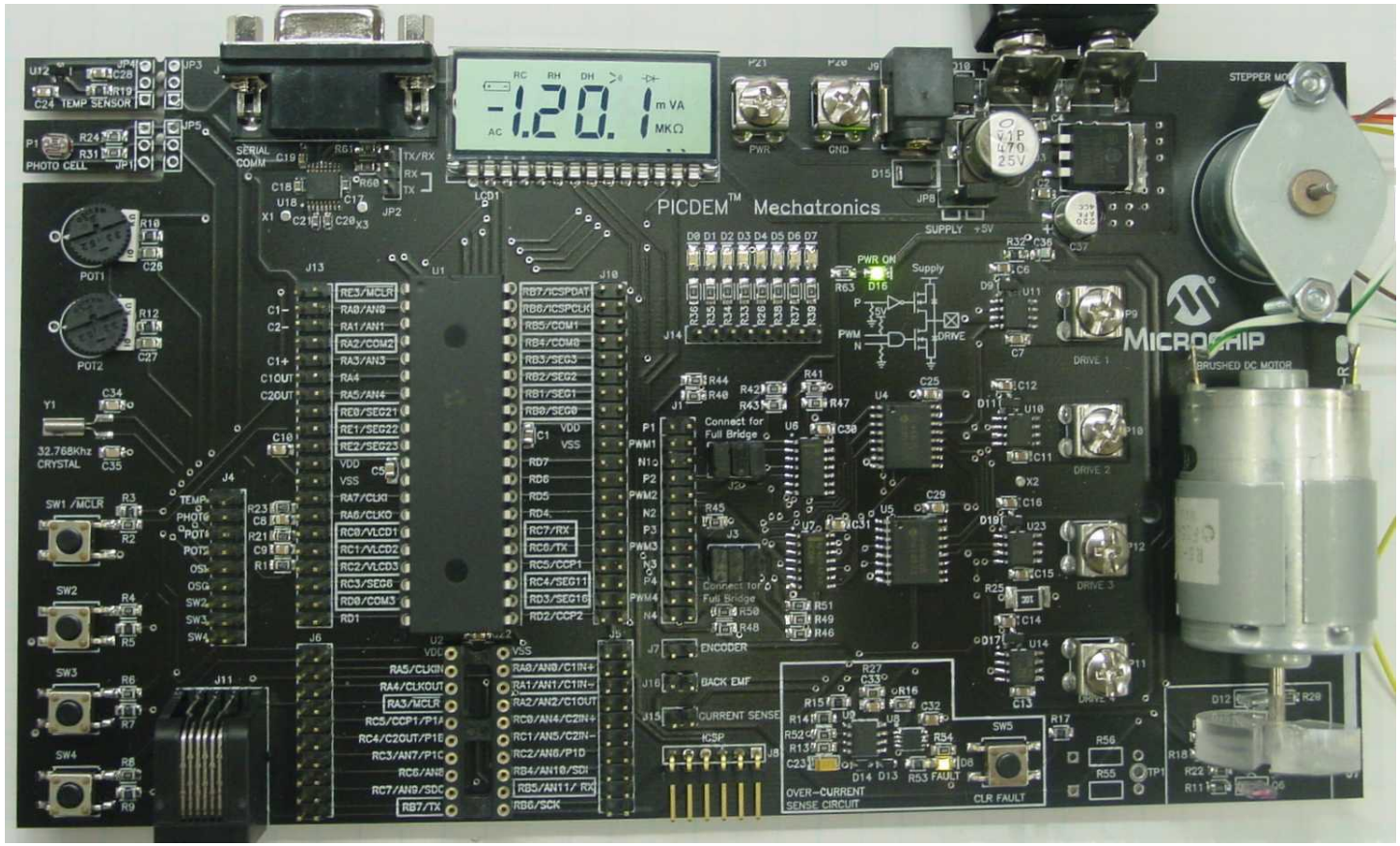
- Introduction to Pulse Width Modulation
  - Duty Cycle is linear to motor speed
  - When driving motors, modulating the PWM signal above 18 kHz is a good idea
- Development Tools
  - MPLAB Signal Analyzer
- CCP Module
  - Can generate an 8-bit 31.2 kHz PWM signal using the 8 MHz internal oscillator

Agenda

- Introduction to Microchip
- Mechatronics examples and benefits
- PIC® Microcontroller Basics
  - Hands-On Learning Cycles
- Labs
  - Simple I/O and Timer 0
  - Reading an analog sensor, LCD module
  - Controlling the speed of a motor
- **Resources**



Continue to learn with the Mechatronics Board . . .



PICDEM™ Mechatronics Projects For Self-Study

- Learn to read an analog temperature sensor and light sensor
- How to create a real-time clock
- Brushed DC motor control
- Stepper motor control
- Using the LCD module on the PIC16F917
- Using serial communication
- Other concepts: Back EMF, optical speed sensing, and current sensing

Microchip Web Resources

- Electronic Product Selector Guide
- Design Centers

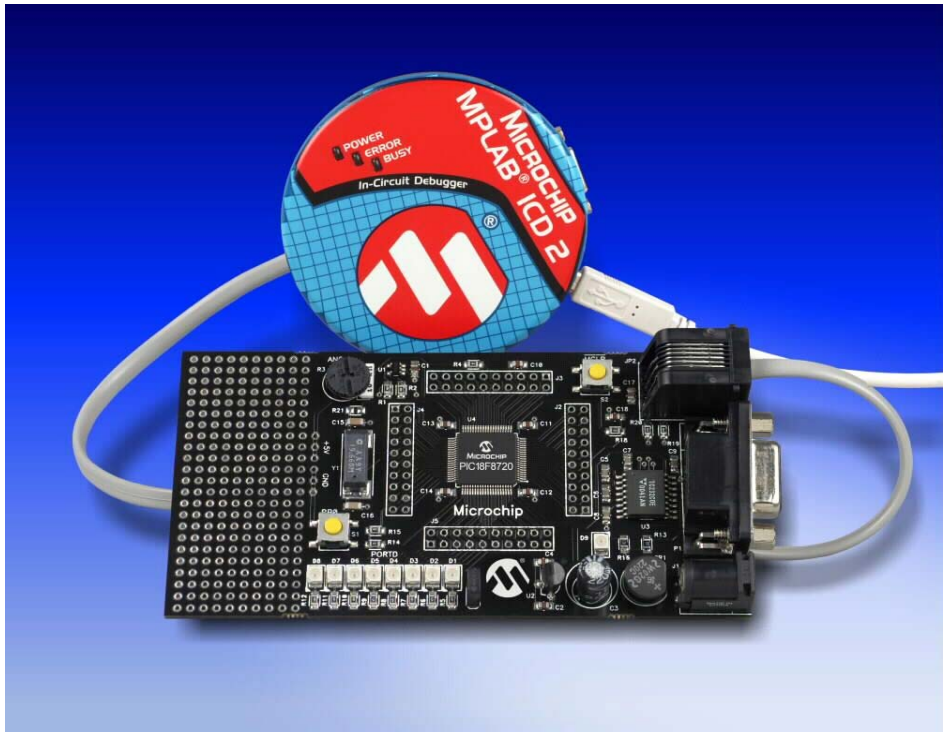
  www.microchip.com/mechatronics
- Step-by-step Tutorials
- Webinars – Online presentations
- Application Notes, Technical Briefs, Articles
- Upcoming Training and Events
- Technical Support

PICkit™ 2 Microcontroller Programmer

- Programs all Flash PIC® microcontrollers and dsPIC® Digital Signal Controllers $34.95 USD
- PICkit 2 Flash Starter Kit – Programmer bundled with small development board for $49.95 USD
- Compatible with the PICDEM Mechatronics Board

MPLAB ICD 2

- MPLAB ICD 2 Part # DV164007
  - Includes 9V power supply, serial cable, USB cable, ICD interface cable, and MPLAB ICD 2



Third Party Tools

- Third Party Tool Developers are our Partners
  - Give them access to early samples/information
  - License Hardware Design to accommodate certain Markets
- Preferred Programmer Vendor
- Top Compilers Choices
  - Hi-Tech
  - IAR
  - CCS
- Complete listing at:

  www.microchip.com/thirdparty

Summary

- Designing a PIC® microcontroller into a mechanical system is beneficial to you
- You've see how easy it is to get started with Microchip and PIC® microcontrollers
- You can now use Microchip's low-cost tools to change, adapt and add features to a design
- You can now perform basic mechatronic tasks using a PIC® microcontroller

Thank You for Coming!

Microchip appreciates your feedback on the content of this presentation. Please tell us what you liked and didn't like about the presentation.

Please send your comments to: mechatronicsWIB@microchip.com

Trademarks & Disclaimers