

First Experiments

1. "Hello World"
2. "Blink with software delay"





Contents

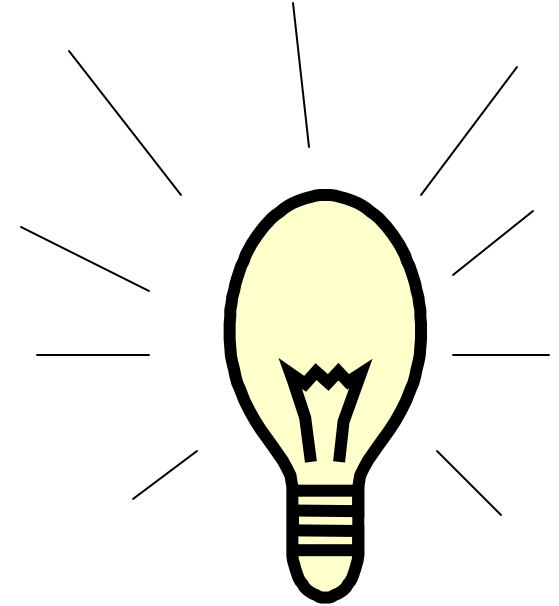
- Directions and descriptions for beginner 16 bit Experimenter starting experiments
 - Each experiment builds on another
- Recommend review of “Quick Start Guide” (available on www.kibacorp.com) to better understand tools installation
- Addendums
 - MPLAB Debugger





Experiment #1 Hello World

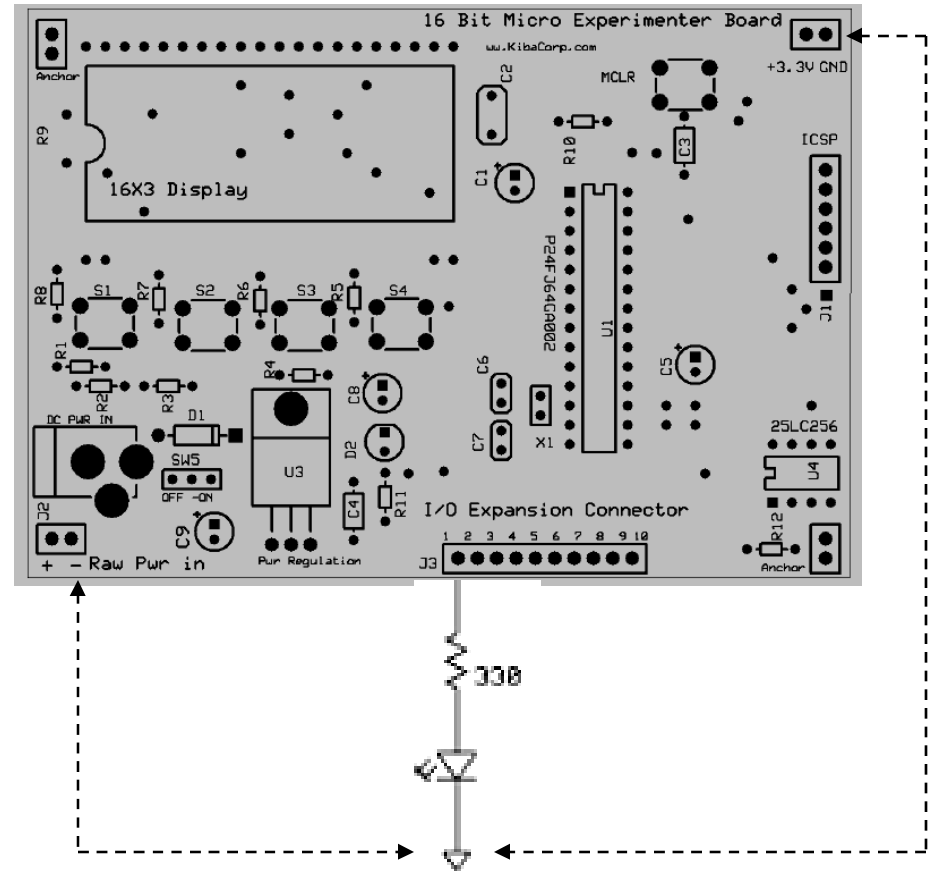
- “Hello World” is typically the first program anyone writes when faced with learning a new programming language or paradigm
- Because of the inherent limitations associated with microcontroller and their lack of a “Standard I/O” to support a screen display our “Hello World” will be limited to turning on an LED





Hello World Hardware

- This lesson shows how to turn on a LED using Experimenter
- Let's start with hardware
 - Connect an LED with 330 ohm resistor to I/O Expansion Pin 1
 - Any I/O pin can sink/source up to 25 ma --the resistor is used to limited current
 - The Expansion I/O pins can be configured for input or output.
- On start-up, the default is input. The software will configure it for an output

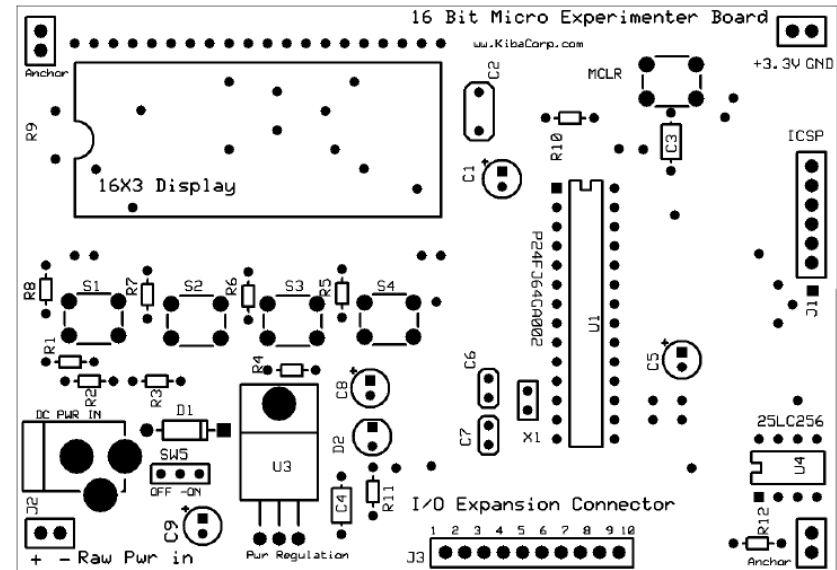


Alternative Ground Connections

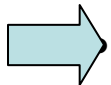


Many possible configurations for PIN 1?

- Pin 1 can have many configurations
- Normal PIC24F Chip level Pin assignments
 - Analog:
 - AN14 analog ADC input channel 14
 - Comparator input (C1IN)
 - Digital
 - SDA2 (I2C data in)
 - CN16 –change detect 16
 - **Simple Digital Port I/O Port B pin 2 (RB2)**
- Peripheral programmable Assigned using RP2
 - Redirect in/out of any other on chip digital peripherals (UART, SPI, CCP) to this pin (more on that later)



J3 I/O Expansion Bus		
J3 I/O bus PIN	Normal Assigned PIN	Programmable PIN
PIN 1	AN4,C1IN-,SDA2,CN6,RB2	RP2
PIN 2	AN5,CIN+,SCL2,CN7,RB3	RP3
PIN 3	SDA1,CN27,RB5	RP5
PIN 4	INT0,CN23,RB7	RP7
PIN 5	SCL1/CN22/RB8	RP8
PIN 6	SDA1/CN21/RB9	RP9
PIN 7	AN12/CN14,RB12	RP12
PIN 8	AN11/CN13,RB13	RP13
PIN 9	AN10,CVREF,RTCC,CN12,RB14	RP14
PIN 10	AN9,CN11,RB15	RP15



We are interested in using RB2 for "Hello World" as a digital output



Rules for Pin Priorities

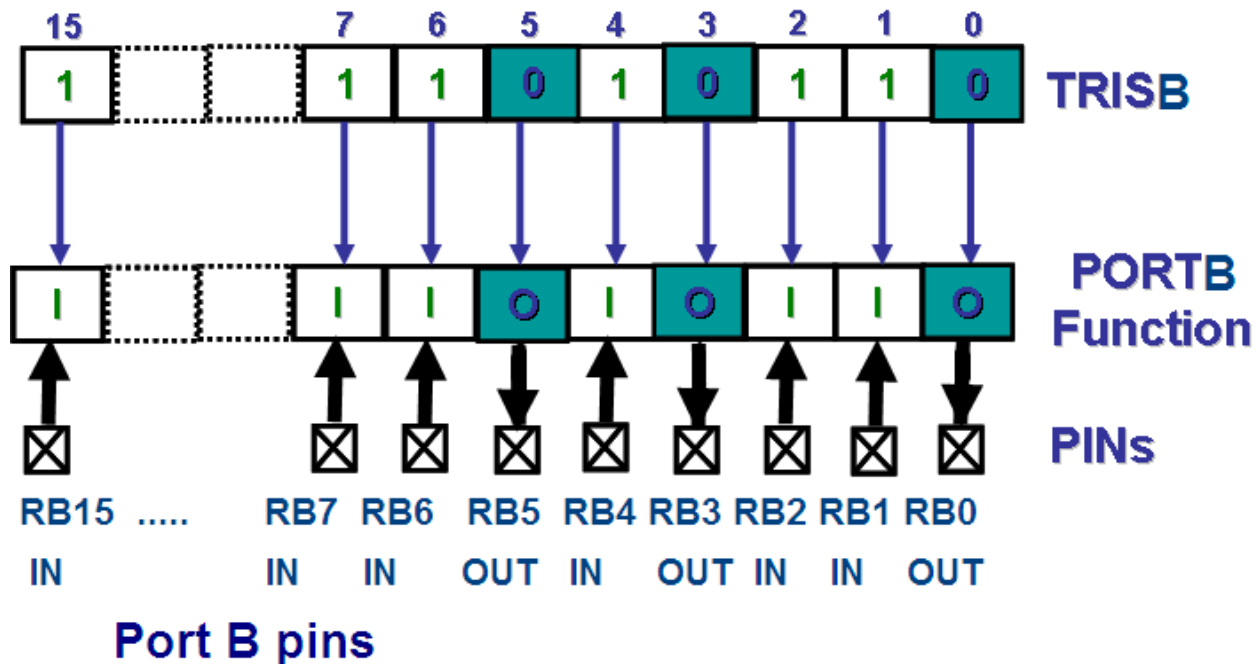
- Both analog and PPS are high priority but require a SW configuration
 - Next priority is fixed digital peripherals—again they need to be SW configured
 - Digital I/O is enabled during power up –*but is configured as input*
 - *-simple to convert to output*
- **Peripheral priorities:**
 1. **Analog Functions** ANx, Vref+/-
 2. **PPS Outputs** UART TX, SDO, OC
 3. **PPS Inputs** UART RX, SDI, IC
 4. **Fixed Digital Peripheral Outputs** I²C™, CN, I/O Ports
 5. **Fixed Digital Peripheral Inputs** I²C, CN, I/O Ports



Setting PORTB Pins as Input or Output

- Example sets entire port at once
- Use Special Function register TRISB
 - If particular bit position is 1 then corresponding Port B pin is a input
 - If particular bit position is 0 then corresponding Port B pin is an output
 - C Syntax

```
_TRISB = 0x80D6;
```





Reading and Writing whole words to PORTB Pins

- Use Special Function Register PORTB to read/write

- C Syntax

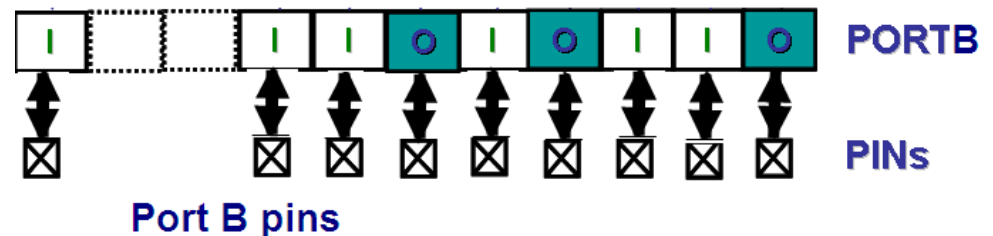
```
int X = 0;  
PORTB = x; //write  
X = PORTB; //read
```

- Use Special Function Register LATB to do a “latched” write

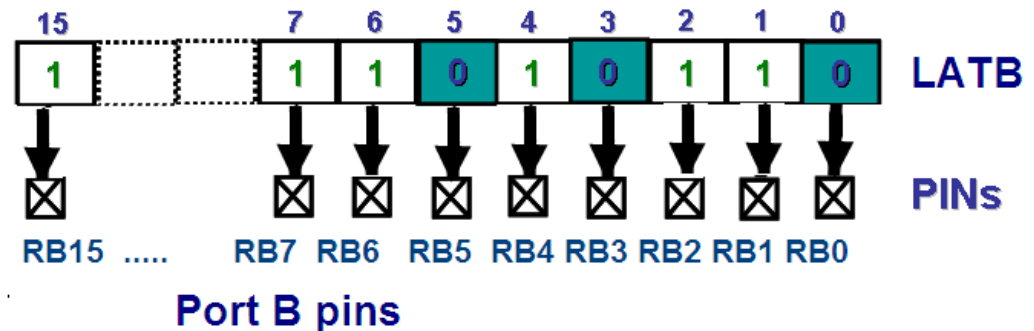
- C syntax

```
int X = 0x80D6;  
LATB = x; //write
```

reading and writing to pins through PORTB

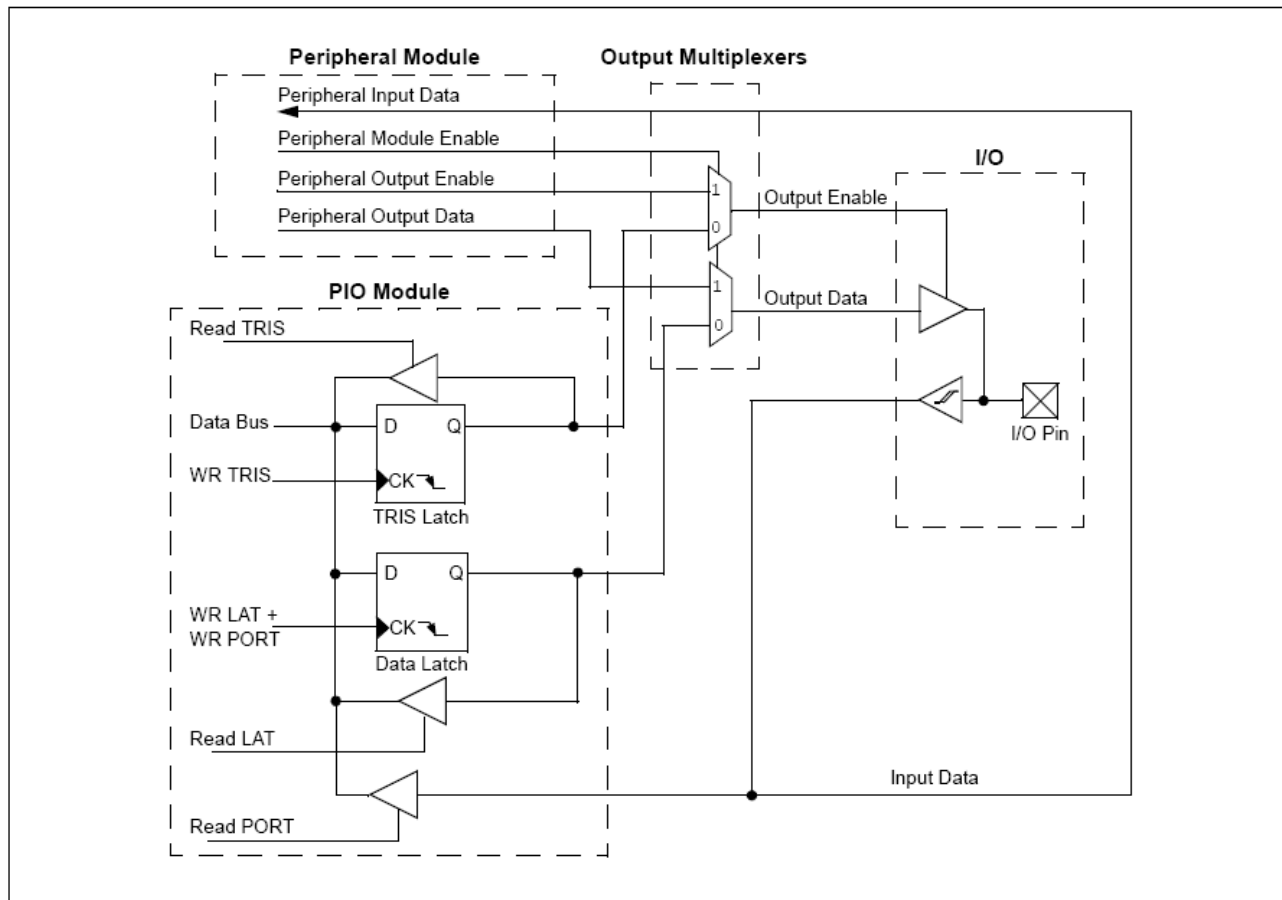


Writing latched outputs to PORTB Pins





I/O PIN Detailed Block Diagram-taken from Microchip PIC24f datasheet





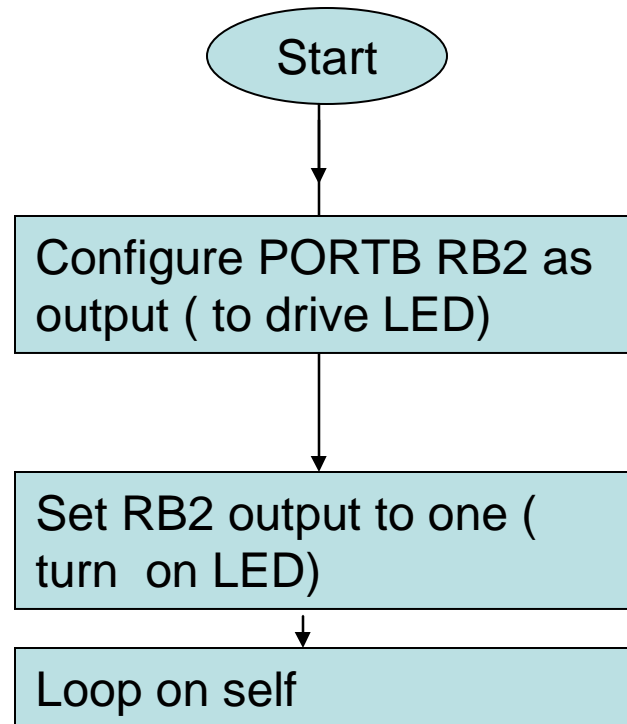
Hello World I/O Configuration

- RB2 is pin 3 of PORTB
- We wished to only control it and not the other PORTB pins
 - On power up all PORTB are inputs
 - Need to set RB2 to output
 - Need to write to RB2
 - 1 will turn on led, 0 will turn off led
- C allows us to control individual pins with following syntax

```
TRISBbits.TRISB2 =0; //set RB2 for output  
LATBbits.LATB2 =1; //turn on led  
LATBbits.LTAB2=0 ; //turn off led
```



Hello World Program Structure





Hello World Code

```
main.c
1  /*
2  First Experiment Part 1 -- Helloworld on 16-bit Micro Experimentor Board
3  turns on led connected to 330 resistor to I/O expansion port pin 1  at RB2
4  */
5
6  #include "p24fj64ga002.h"
7  #define iend 400
8  #define jend 1000
9
10 // PIC24F on experimenter runs 16 MIPS operation.
11
12 //config fues settings
13 _CONFIG2(0xF9C7); //demo
14 //default primary osc disabled, IOLOCK may be changed via unlock sequence, OSC pin has digital I/O function, clock switching and monitor disabled
15 //Fast Internal Oscillator enabled with PLL
16 _CONFIG1(0x3F5F); //demo default
17 //Watchdog disabled, ICSP cahnnel PGCq1/PGD1, code protect and JTAG idsabled.
18
19
20
21 //main code section
22 int main(void)
23 {
24
25 CLKDIV = 0x0000; //do not divide
26 TRISBbits.TRISB2 =0; // set RB2 tO output
27
28
29
30
31 LATBbits.LATB2 = 1; //RB2 = 1
32 while(1); //infinite loop processor reminas here.
33
34 }
35
36
```



Hello World Experiment

1. Wire led/resistor as shown to pin 1 of experimenter
2. Hook up experimenter to PICKIT2
3. Open project Helloworld.mcp
4. Build code/download to PICKIT2 as programmer
5. Verify LED is lit.
6. Examine source code
7. Reconfigure MPLAB for debugger

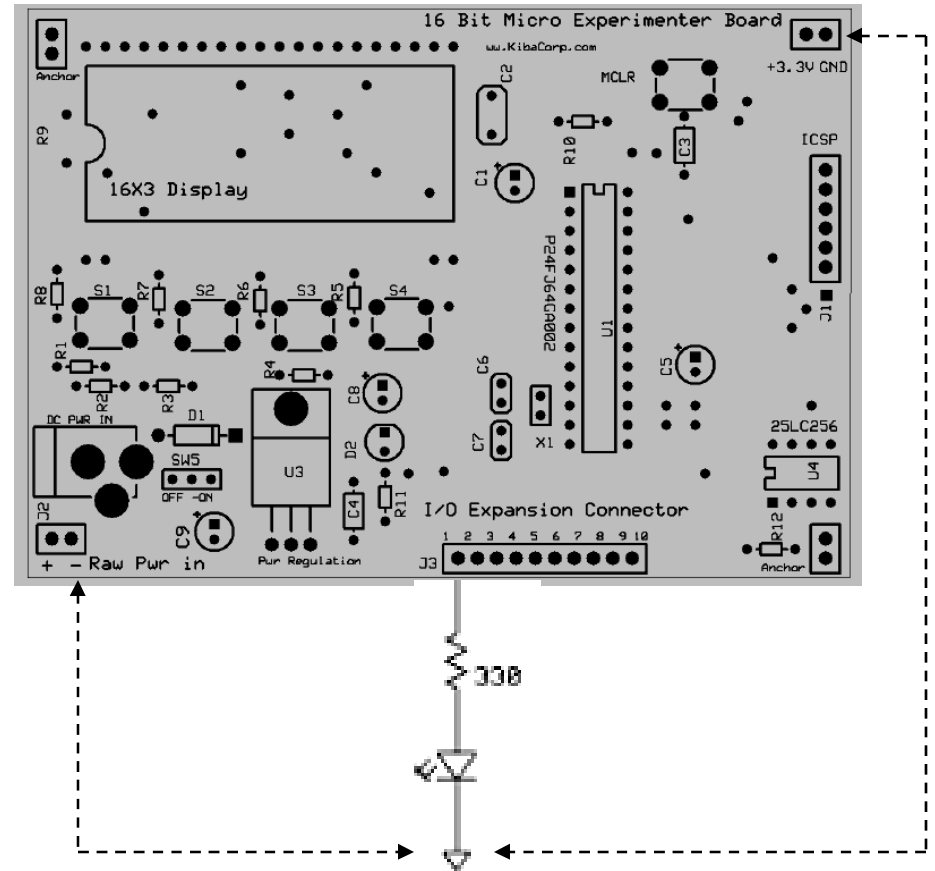
Select simulator and rebuild – animate, watch, break, single step

Hookup and select PICKIT2 rebuild/program –watch, break, single step



BLINK Hardware

- This lesson shows how to blink an LED using Experimenter
- Let's start with hardware
 - Connect an LED with 330 ohm resistor to I/O Expansion Pin 1
 - Any I/O pin can sink/source up to 25 ma --the resistor is used to limited current
 - The Expansion I/O pins can be configured for input or output.
- On start-up, the default is input. The software will configure it for an output



Alternative Ground Connections



BLINK

- This lesson shows how to turn on a LED, and make it blink.
- While this might seem a trivial change from previous lesson, it gives a context to explore the use of delay function in software.
- Uses same hardware set and software I/O configuration as “Hello World”





Why do we need a delay?

- With the PICmicro running at 16MIPS, each instruction takes only to nanoseconds to execute.
- If we want to blink the led and see it blink it will be necessary to slow down the main program loop.
- If we don't include the delay, the LED will blink so fast that it will appear to be on constantly
- The delay routine consists of a software function that does a loop in a loop to “eat” up processing cycles.



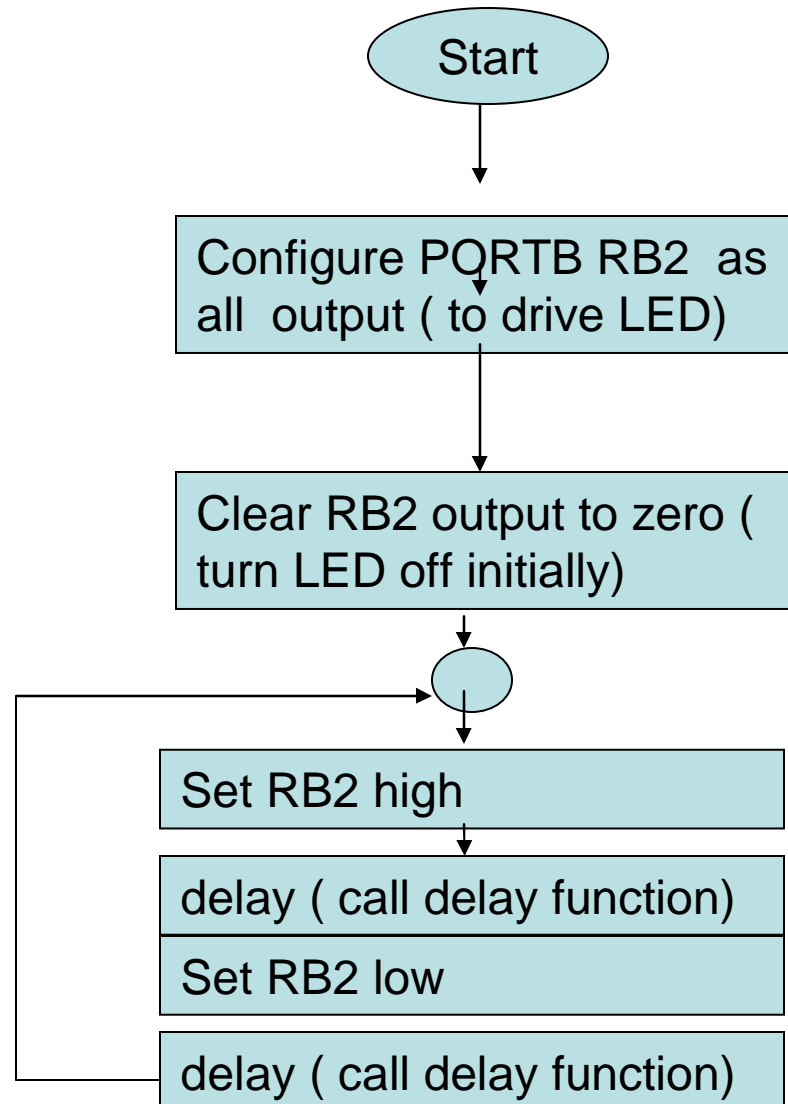
Delay Function

```
//delay function
#define iend 400
#define jend 1000

void delay(void) { // 200 msec
    int i,j;
    for (i=0; i<iend; i++)
        for (j=0;j<jend; j++);
}
```



Program Structure





Blink Code

```
main.c*
1  /*
2  First Experiment Part 1 -- Blink on 16-bit Micro Experimentor Board
3  blinks led connected to 330 resistor to I/O expansion port pin 1 at RB2
4  */
5
6  #include "p24fj64ga002.h"
7  #define iend 400
8  #define jend 1000
9
10 // PIC24F on experimenter runs 16 MIPS operation.
11
12 //config fues settings
13 _CONFIG2(0xF9C7); //demo
14 //default primary osc disabled, IOLOCK may be changed via unlock sequence, OSC pin has digital I/O function, clock switching and monitor disabled
15 //Fast Internal Oscillator enabled with PLL
16 _CONFIG1(0x3F5F); //demo default
17 //Watchdog disabled, ICSP cahnnel PGCq1/PGD1, code protect and JTAG idsabled.
18
19 //delay function
20
21 void delay(void) { // 200 msec
22     int i,j;
23     for (i=0; i<iend; i++)
24         for (j=0; j<jend; j++);
25 }
26
27 //main code section
28 int main(void)
29 {
30
31     CLKDIV = 0x0000; //do not divide
32     TRISBbits.TRISB2 = 0; // set RB2 to output
33
34     while(1)
35     {
36         LATBbits.LATB2 = 0; //RB2 = 0
37         delay(); //call dealay function
38
39         LATBbits.LATB2 = 1; //RB2 = 1
40         delay(); // call delay function
41     }
42 }
43
```



Blink Experiment

1. Wire led/resistor as shown to pin 1 of experimenter
2. Hook up experimenter to PICKIT2
3. Open project Helloworld.mcp
4. Build code/download to PICKIT2 as programmer
5. Verify LED is blinking.
6. Examine source code
7. Reconfigure MPLAB for debugger
Hookup and select PICKIT2 rebuild/program –watch,
break, single step



Addendum -Using Debugger





MPLAB Debugger

- Lots of capabilities
- Can use MPLAB Simulator for just simulations on PC
 - Can't simulate analog
- Can use PICKIT2 as debugger for real time debug
- Need to set MPLAB for debug and then rebuild and download code for debugging (this last step for PICKIT2 only)

MPLAB® IDE DEBUG

Breakpoints

- ✦ Address/Label/Line Number
- ✦ Complex condition

Complex Watch Points

- ✦ Arrays
- ✦ Structures
- ✦ Source Data Number Formats

Execution Trace

- ✦ Software (simulator)
- ✦ Hardware (ICE)

Simulator Stimulus

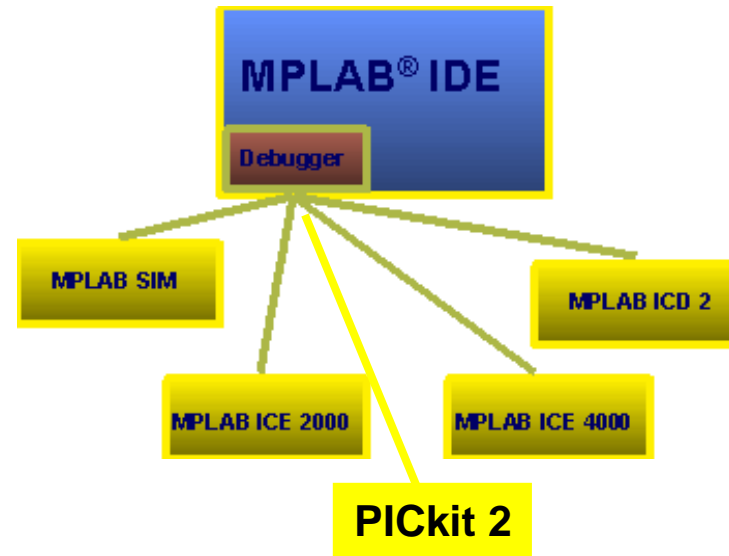
Simulator Output Log

Import/Export Data



Debugger Options

- The debugger can be a **software program** that simulates the operation of the Microcontroller for testing or it can be **special instrumentation** to analyze the program as it executes in the application.
- **MPLAB Simulator Debugger**
 - Usually a simulator runs somewhat slower than an actual Microcontroller, since the CPU in the PC is being used to simulate the operations of the Microcontroller. The speed of simulation depends upon the speed of the PC, the PC's operating system, how many other tasks are being run in the background, and the complexity of the simulation
- **PICKIT2 Hardware Debugger and Programmer**
 - A programmer simply transfers the machine code from the PC into the internal memory of the target Microcontroller. The Microcontroller can then be plugged into the application, and it will run as designed.





Select Debug-Simulator

A screenshot of the MPLAB IDE v8.00 interface. The 'Debugger' menu is open, and the 'Select Tool' option is highlighted. The 'Select Tool' submenu is visible, showing the following options: None, 1 MPLAB ICD 2, 2 MPLAB ICE-4000, 3 MPLAB SIM (selected), 4 MPLAB ICE 2000, 5 REAL ICE, 6 PICkit 2, and 7 PIC32MX Starter Kit. The main editor window displays the source code for 'Assign.c', which includes variable declarations and a while loop. The status bar at the bottom indicates the target is PIC16F887.

assign - MPLAB IDE v8.00

File Edit View Project Debugger Programmer Tools Configure Window Help

Checksum: 0x199

assign.mcw

C:\EET250

Source Files

Header Files

Object Files

Libraries

Other

Files

Assign.c

```
myke_piedko
04.09.28

*/
14 int i; // Uninitialized Variable Declaration
15 int j = 23; // Variable Declared with Initial
16 // value assignment (Initialization)
17
18 main()
19 {
20     i = 47; // The variable "i" assigned (or loaded
21 // with the constant value 47
22     i = j; // The variable "i" assigned contents of
23 // of "j".
24     i = j = 1; // "i" and "j" assigned the same value.
25
26     while(1 == 1);
27
```

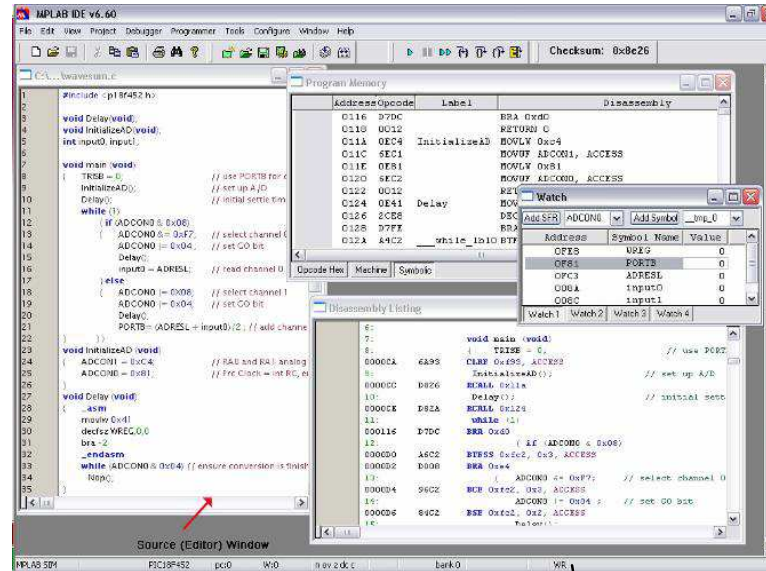
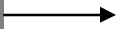
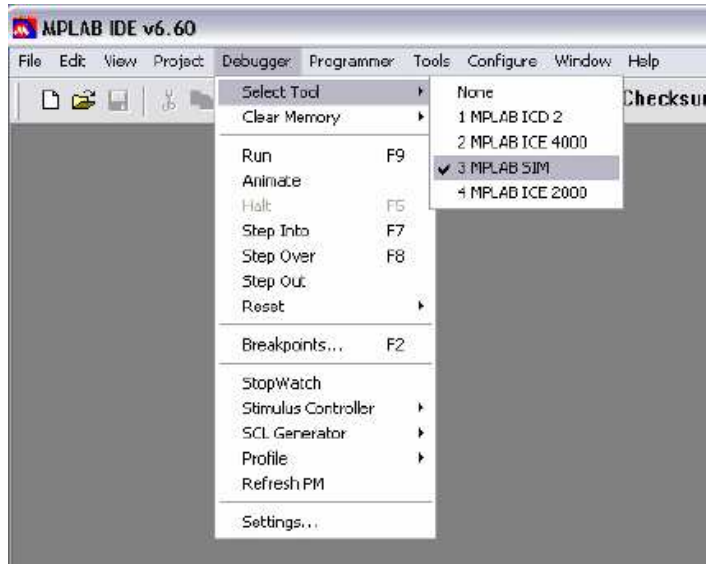
MPLAB SIM PIC16F887 pc:0x7fd W:0 Z dc c 20 MHz bank 0 Ln 7, Col 1 INS WR

start C:\Documents and Se... assign - MPLAB IDE v... untitled - Paint Microsoft PowerPoint ... 9:39 AM



MPLAB SIM (non real time PC only)

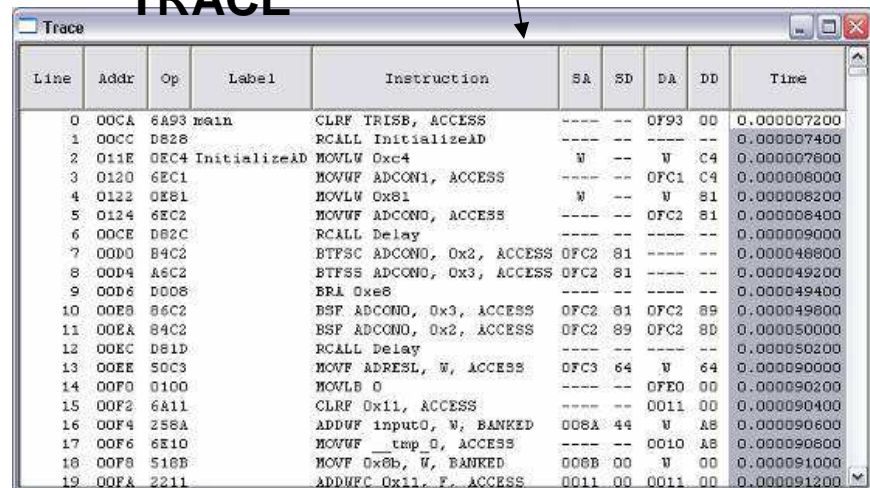
Simulated Operation for Debug



STOP WATCH



TRACE





PICKIT 2 Debug Express

- Functions as both programmer and real time debugger
- As real time debugger has all the capabilities of simulator in breakpoints, watch, single step.
 - Has limited number of breakpoints
- Lets you debug your applications in real time as in is working in your prototype

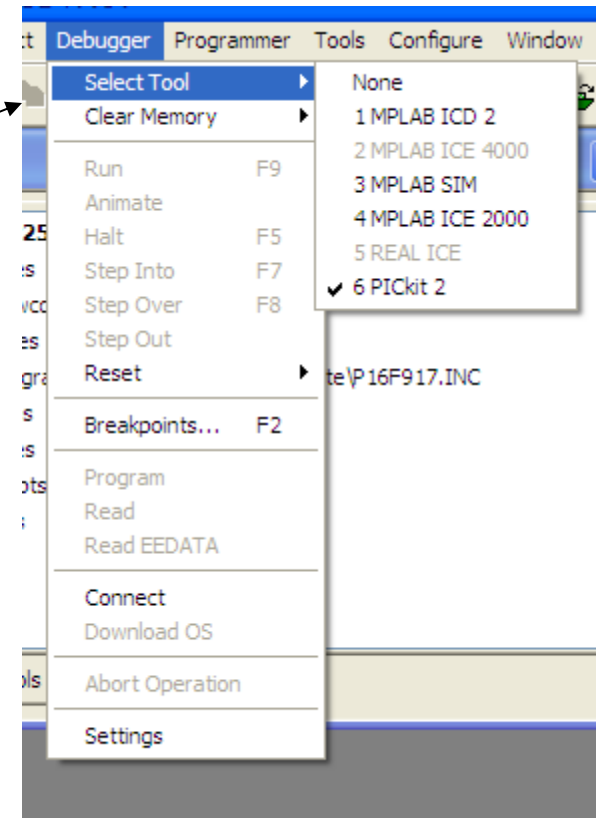




Using PICKIT2 DEBUGGER with MPLAB IDE

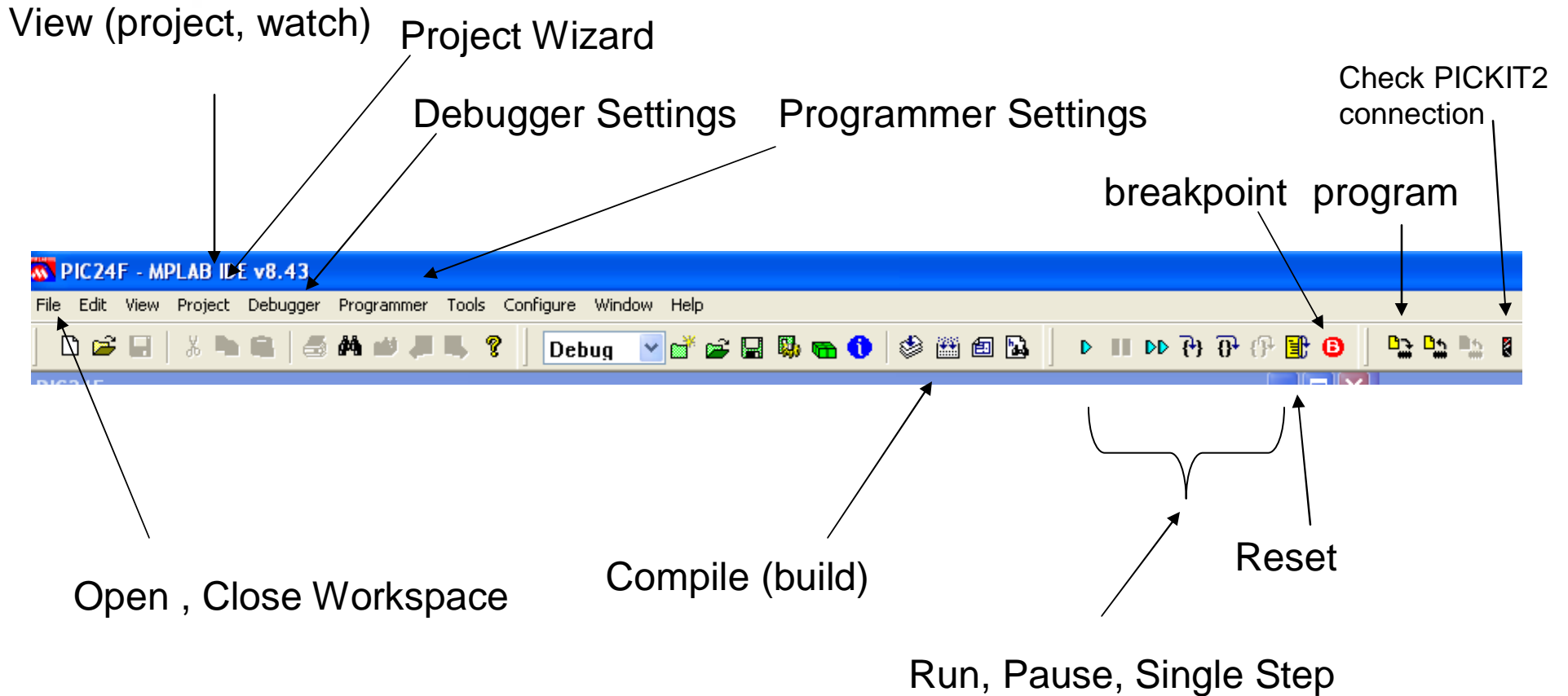


- Plug in the PICKIT2
- Select DEBUG here
- Select PICKIT2 under Debugger menu
- Output should appear as follows



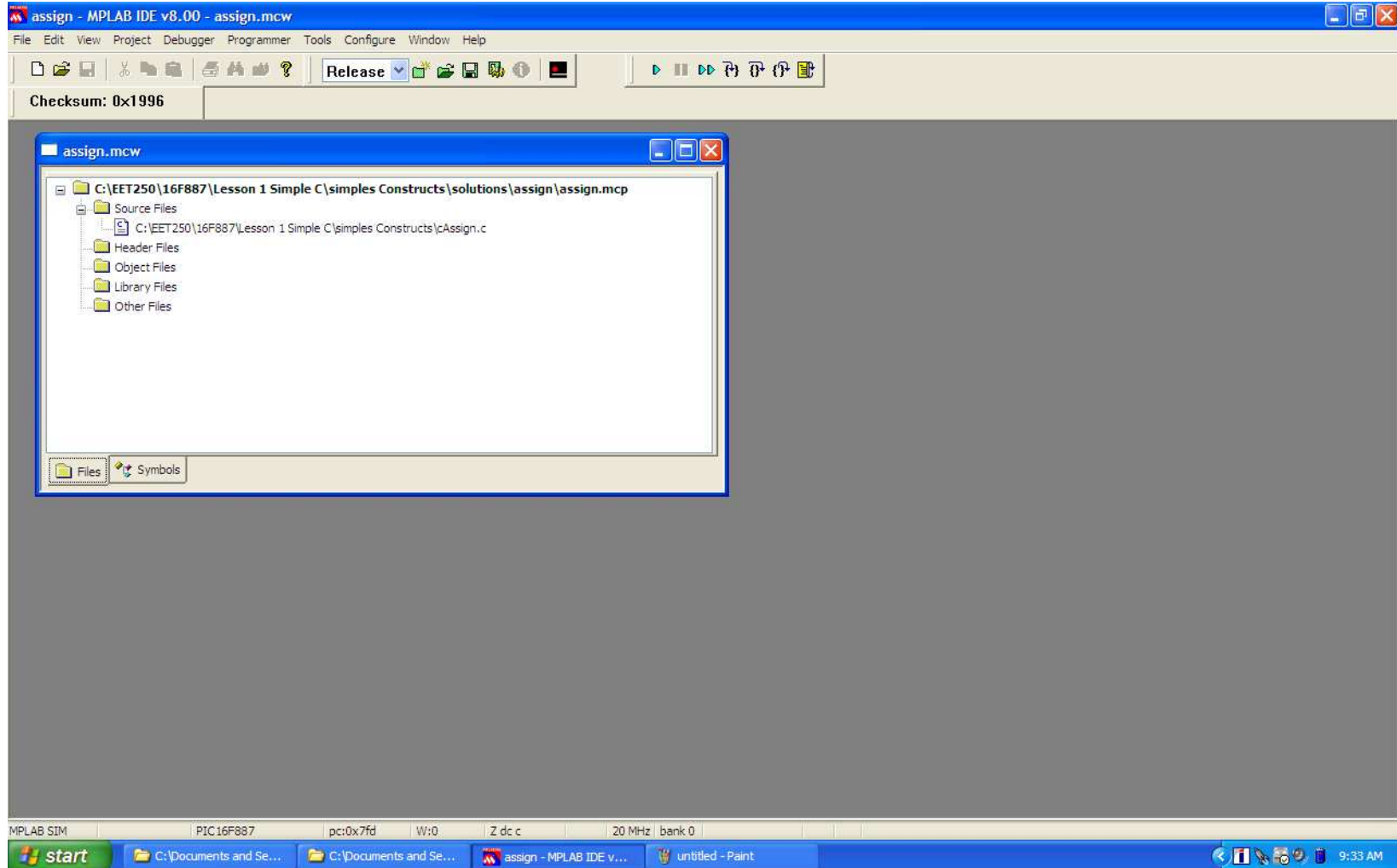


MPLAB IDE Tool Bar-DEBUG





View Project





View Source

The screenshot shows the MPLAB IDE v8.00 interface. The main window displays the source code for the file `cAssign.c`. The code includes a preprocessor directive for `<pic.h>`, a multi-line comment describing the program's purpose, and a `main()` function. The `main()` function contains several assignment statements and a `while` loop.

```
1 #include <pic.h>
2 /* cAssign.c - A brief Look at Assignment Statements
3
4 This Program Demonstrates how Assignment Statements work
5 in C.
6
7 myke predko
8 04.09.28
9
10 */
11
12 int i; // Uninitialized Variable Declaration
13 int j = 23; // Variable Declared with Initial
14 // value assignment (Initialization)
15
16 main()
17 {
18     i = 47; // The variable "i" assigned (or loaded
19 // with) the constant value 47
20     i = j; // The variable "i" assigned contents of
21 // of "j".
22     i = j = 1; // "i" and "j" assigned the same value.
23
24     while(1 == 1);
25
26 } // End cAssign
27
```

The status bar at the bottom of the IDE shows the following information: MPLAB SIM, PIC16F887, pc:0x7fd, W:0, Z dc c, 20 MHz, bank 0, Ln 1, Col 1, INS, WR.

The Windows taskbar at the bottom shows the Start button and several open applications: C:\Documents and Se..., assign - MPLAB IDE v..., untitled - Paint, and Microsoft PowerPoint ... The system clock shows 9:36 AM.



Build

assign - MPLAB IDE v8.00

File Edit View Project Debugger Programmer Tools Configure Window Help

Checksum: 0x1996

assign.mcw

- C:\EET250\16F887\Lesson 1 Simple C\simples Constructs\solutions\assign\assign.mcp
 - Source Files
 - C:\EET250\16F887\Lesson 1 Simple C\simples Constructs\cAssign.c
 - Header Files
 - Object
 - Libran
 - Other

C:\EET250\16F887\Lesson 1 Simple C\simples Constructs\cAssign.c

```
1 #include <pic.h>
2 /* cAssign.c - a brief Techon Systems
3
4 This Pr
5 in C.
6
7 myke pr
8 04.09.2
9
10
11
12
13 int i;
14 int j =
15
16 #main()
17 {
18
19     i =
20
21     i =
22
23     i =
24
25     whi
26
27 //
```

Output

Build Version Control Find in Files MPLAB SIM

Build C:\EET250\16F887\Lesson 1 Simple C\simples Constructs\solutions\assign\assign for device 16F887
Using driver C:\Program Files\HI-TECH Software\PICC\LITE\9.60\bin\picl.exe

Executing: "C:\Program Files\HI-TECH Software\PICC\LITE\9.60\bin\picl.exe" -C "C:\EET250\16F887\Lesson 1 Simple C\simples Co
Executing: "C:\Program Files\HI-TECH Software\PICC\LITE\9.60\bin\picl.exe" -oassign.cof -massign.map cAssign.obj -chip=16F887

Memory Summary:

Space	used	hex	dec	of	total	percentage
Program space	used	3Dh	(61)	of	800h words	(3.0%)
Data space	used	7h	(7)	of	80h bytes	(4.0%)
EEPROM space	used	0h	(0)	of	100h bytes	(0.0%)
Configuration bits	used	0h	(0)	of	2h words	(0.0%)
ID Location space	used	0h	(0)	of	4h bytes	(0.0%)

Loaded C:\EET250\16F887\Lesson 1 Simple C\simples Constructs\solutions\assign\assign.cof.

MPLAB SIM PIC16F887 pc:0x7fd W:0 Z dc c 20 MHz bank 0 WR

start C:\Documents and Se... assign - MPLAB IDE v... untitled - Paint Microsoft PowerPoint ... 9:37 AM



Open and Set Watch

The screenshot shows the MPLAB IDE v8.00 interface. The 'View' menu is open, and the 'Watch' option is selected. The 'Watch' window is displayed, showing a table with columns for Address, Symbol Name, and Value. The table contains two entries: one for address 020 with symbol name 'i' and value 0x0001, and another for address 022 with symbol name 'j' and value 0x0001. The source code in the background shows the following C code:

```
12  +/  
13  
14  int i; // Uninitialized Variable Declaration  
15  int j = 23; // Variable Declared with Initial  
16 // value assignment (Initialization)  
17  
18  |main()  
19  |  
20  | {  
21  |     i = 47; // The variable "i" assigned (or loaded  
22  | // with) the constant value 47  
23  |     i = j; // The variable "i" assigned contents of  
24  | // of "j".  
25  |     i = j = 1; // "i" and "j" assigned the same value..  
26  
27  |     while(1 == 1);
```



Set Breakpoint

The screenshot displays the MPLAB IDE v8.00 interface. The main window shows a C program named `cAssign.c` with a breakpoint (red 'B' icon) set at line 20. The code includes a `main()` function with several assignment statements. A Watch window is open, showing the current values of variables `i` and `j` at memory addresses 020 and 022, both containing the value `0x0001`.

```
1 #include <pic.h>
2 /* cAssign.c - A brief Look at Assignment Statements
3
4 This Program Demonstrates how Assignment Statements work
5 in C.
6
7
8 myke predko
9 04.09.28
10
11
12
13
14 int i; // Uninitialized Variable Declaration
15 int j = 23; // Variable Declared with Initial
16 // value assignment (Initialization)
17
18 main()
19 {
20     i = 47; // The variable "i" assigned (or loaded
21 // with) the constant value 47
22     i = j; // The variable "i" assigned contents of
23 // of "j".
24     i = j = 1; // "i" and "j" assigned the same value.
25
26     while(1 == 1);
27
```

Address	Symbol Name	Value
020	i	0x0001
022	j	0x0001

Watch 3 | Watch 4

MPLAB SIM PIC16F887 pc:0 W:0 Z dc c 20 MHz bank 0 Ln 20, Col 1 INS WR

start C:\Documents and Se... assign - MPLAB IDE v... untitled - Paint Microsoft PowerPoint ... 9:43 AM



Execute and stop on break

The screenshot shows the MPLAB IDE v8.00 interface. The main window displays the source code for 'Assign.c'. The code includes a preprocessor directive, a comment, and a main function. The main function contains several assignment statements and a while loop. The Watch window is open, showing the values of variables 'i' and 'j' at memory addresses 020 and 022 respectively.

```
#include <pic.h>
/* cAssign.c - A brief Look at Assignment Statements
3
4 This Program Demonstrates how Assignment Statements work
5 in C.
6
7
8 myke predko
9 04.09.28
10
11
12
13
14 int i; // Uninitialized Variable Declaration
15 int j = 23; // Variable Declared with Initial
16 // value assignment (Initialization)
17
18 main()
19 {
20 | i = 47; // The variable "i" assigned (or loaded
21 // with) the constant value 47
22 | i = j; // The variable "i" assigned contents of
23 // of "j".
24 | i = j = 1; // "i" and "j" assigned the same value..
25
26 while(1 == 1);
27
```

Address	Symbol Name	Value
020	i	0x0000
022	j	0x0017

The status bar at the bottom shows the target device as PIC16F887, the program counter at 0x7ed, and the current instruction at 20 MHz, bank 0, line 20, column 1. The taskbar shows the Start button and several open applications: C:\Documents and Se..., assign - MPLAB IDE v..., untitled - Paint, and Microsoft PowerPoint ...



Single Step and Watch

assign - MPLAB IDE v8.00

File Edit View Project Debugger Programmer Tools Configure Window Help

Checksum: 0x1996

assign.mcw

- C:\EET250\16F887\Lesson 1 Simple C\simples Constructs\solutions\assign\assign.mcp
 - Source Files
 - C:\EET250\16F887\Lesson 1 Simple C\simples Constructs\Assign.c
 - Header Files
 - Object Files
- C:\EET250\16F887\Lesson 1 Simple C\simples Constructs\Assign.c

```
1 #include <pic.h>
2 /* cAssign.c - A brief Look at Assignment Statements
3
4 This Program Demonstrates how Assignment Statements work
5 in C.
6
7
8 myke predko
9 04.09.28
10
11
12
13
14 int i; // Uninitialized Variable Declaration
15 int j = 23; // Variable Declared with Initial
16 // value assignment (Initialization)
17
18 main()
19 {
20     i = 47; // The variable "i" assigned (or loaded
21 // with) the constant value 47
22     i = j; // The variable "i" assigned contents of
23 // of "j".
24     i = j = 1; // "i" and "j" assigned the same value.
25
26     while(1 == 1);
27
```

Watch

Address	Symbol Name	Value
020	i	0x002F
022	j	0x0017

MPLAB SIM PIC16F887 pc:0x7F2 W:0x2F Z dc c 20 MHz bank 0 Ln 22, Col 1 INS WR

start C:\Documents and Se... assign - MPLAB IDE v... untitle - Paint Microsoft PowerPoint...

9:46 AM



Formatting Watch Variables

The screenshot shows the MPLAB IDE interface. The main window displays the Watch window with a table of variables:

Address	Symbol Name	Value
020	i	23
022	j	0x0017

The Watch Properties dialog box is open, showing the following settings:

- Symbol: i
- Size: 16 bits
- Format: Decima
- Signed
- Byte Order: HighLow
- Memory: File Register

The status bar at the bottom indicates the simulation is running on a PIC16F887 at 20 MHz, with the PC register at 0x7f6.