MicroTRAK/P18 Development & Training Kit

Lab Book

Date: 20 December 2011

Document Revision: 1.02

BiPOM Electronics

16301 Blue Ridge Road, Missouri City, Texas 77489
Telephone: (713) 283-9970 Fax: (281) 416-2806
E-mail: info@bipom.com
Web: www.bipom.com

WARRANTY:

BiPOM Electronics warrants MicroTRAK/P18 for a period of 1 year. If the Kit becomes defective during this period, BiPOM Electronics will at its option, replace or repair the Kit. This warranty is void if the product is subjected to physical abuse or operated outside stated electrical limits. BiPOM Electronics will not be responsible for damage to any external devices connected to the Kit. BiPOM Electronics disclaims all warranties express or implied warranties of merchantability and fitness for a particular purpose. In no event shall BiPOM Electronics be liable for any indirect, special, incidental or consequential damages in connection with or arising from the use of this product. BiPOM's liability is limited to the purchase price of this product.

TABLE OF CONTENTS

Introduction

The purpose of the MicroTRAK/P18 is to familiarize the student with developing practical applications using the PIC18F series of microcontrollers. PIC18F is based on the Microchip® PIC® microcontroller architecture, a very popular microcontroller system with applications ranging from industrial, medical, home automation to automotive.

The MicroTRAK/P18 consists of the following components:

- MicroTRAK Carrier Board
- MINI-MAX/P18 Microcontroller Board
- TB-1 Training Board
- PROTO-1 Prototyping Board
- I/O Module
- LCD242 LCD with backlight
- KP1-4X4 Keypad
- Carrier board with extra voltage regulator
- MPLAB with Assembler, Debugger and Simulator (free download from Microchip web site)
- Micro-IDE Integrated Development Environment with support for MPLAB from Microchip
- Example projects
- Serial downloader
- Cables
- Adapter
- Training Manuals
- Labbook

The following external items are required for each training kit station:

- Personal Computer (PC) running Windows 98/NT/2000/XP/Vista or Windows 7 (32-bit or 64-bit)
  Minimum 256MB memory and 1 GB of available hard disk space.

- One available RS232 Serial port (COM1 through COM8). If your PC does not have a serial port, you can use our CBL-USB-COM-1 product to convert USB to RS232.

Figure 1 shows all the components connected together:

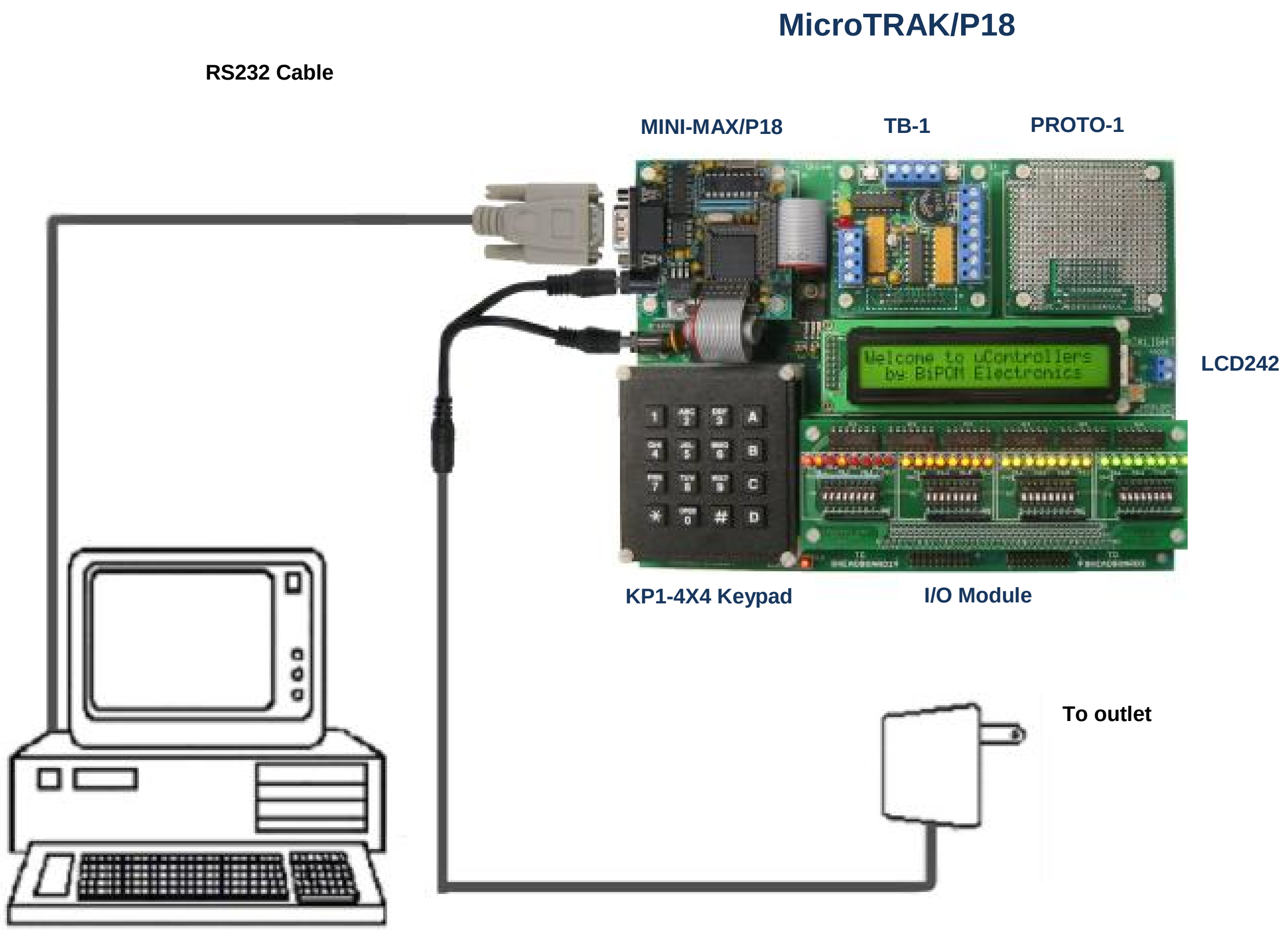


Figure 1

Getting Started

Log on to Windows before using MicroTRAK/P18. Enter the user name and password that your instructor has given to you before the Lab. Make sure to log out when you are done with the PC at the end of the Lab. Do not share your username or password with anybody.

The programs are written as C and Assembly Language Program files (with the extension .C or .asm but they are plain text files). These C and Assembly Program files are created using MPLAB program from Microchip.

You can edit and save programs and download to the Training Board using MPLAB. Creating programs and running them on the Training Board consists of the following steps:

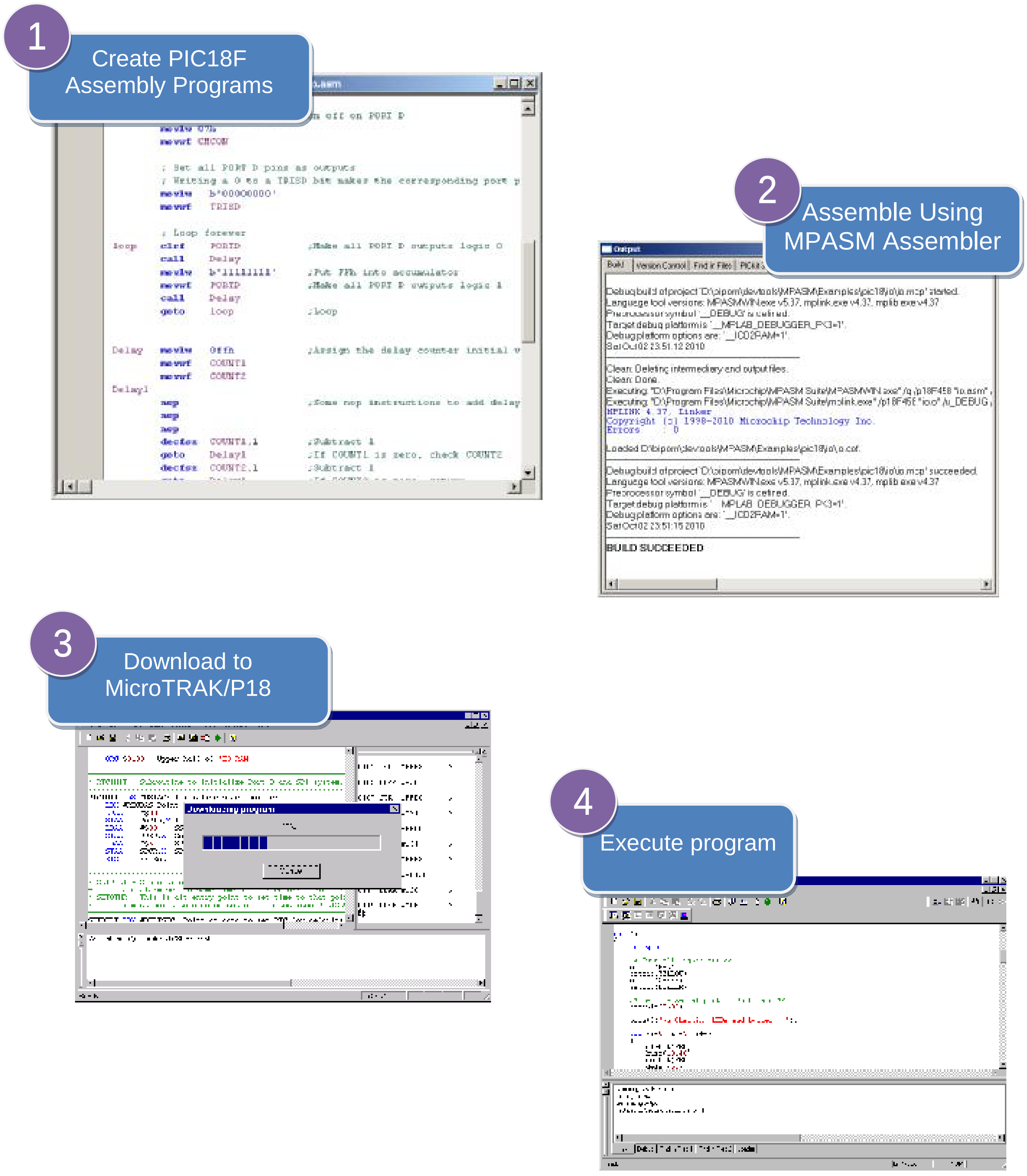
1. Edit an existing or create a new program using MPLAB Program Editor.

2. Compile the program using MPLAB Build Toolkit (C Compiler or Assembler).

3. Download the program to the Training Board using Micro-IDE Serial Loader.

4. Run and debug the program on the Training Board using Micro-IDE Terminal Window.

Figure 2 illustrates these steps.


1
Create PIC18F Assembly Programs
2
Assemble Using MPASM Assembler
Output
Build
Version Control
Find in Files
3
Download to MicroTRAK/P18
Downloading program
4
Execute program

```
                                n off on PORT D
        movlw 07h
        movwf CMCON

        ; Set all PORT D pins as outputs
        ; Writing a 0 to a TRISD bit makes the corresponding port p
        movlw   b'00000000'
        movwf   TRISD

        ; Loop forever
loop    clrf    PORTD           ;Make all PORT D outputs logic 0
        call    Delay
        movlw   b'11111111'     ;Put FFh into accumulator
        movwf   PORTD           ;Make all PORT D outputs logic 1
        call    Delay
        goto    loop            ;loop


Delay   movlw   0ffh            ;Assign the delay counter initial v
        movwf   COUNT1
        movwf   COUNT2
Delay1
        nop                     ;Some nop instructions to add delay
        nop
        nop
        decfsz  COUNT1,1        ;Subtract 1
        goto    Delay1          ;If COUNT1 is zero, check COUNT2
        decfsz  COUNT2,1        ;Subtract 1
```

```
Debug build of project 'D:\bipom\devtools\MPASM\Examples\pic18\io\io.mcp' started.
Language tool versions: MPASMWIN.exe v5.37, mplink.exe v4.37, mplib.exe v4.37
Preprocessor symbol '__DEBUG' is defined.
Target debug platform is '__MPLAB_DEBUGGER_PK3=1'.
Debug platform options are: '__ICD2RAM=1'.
Sat Oct 02 23:51:12 2010
----------------------------------------------------------------------
Clean: Deleting intermediary and output files.
Clean: Done.
Executing: "D:\Program Files\Microchip\MPASM Suite\MPASMWIN.exe" /q /p18F458 "io.asm" /
Executing: "D:\Program Files\Microchip\MPASM Suite\mplink.exe" /p18F458 "io.o" /u_DEBUG /
MPLINK 4.37, Linker
Copyright (c) 1998-2010 Microchip Technology Inc.
Errors    : 0

Loaded D:\bipom\devtools\MPASM\Examples\pic18\io\io.cof.
----------------------------------------------------------------------
Debug build of project 'D:\bipom\devtools\MPASM\Examples\pic18\io\io.mcp' succeeded.
Language tool versions: MPASMWIN.exe v5.37, mplink.exe v4.37, mplib.exe v4.37
Preprocessor symbol '__DEBUG' is defined.
Target debug platform is '__MPLAB_DEBUGGER_PK3=1'.
Debug platform options are: '__ICD2RAM=1'.
Sat Oct 02 23:51:15 2010
----------------------------------------------------------------------
BUILD SUCCEEDED
```

Figure 2

LAB1 – Introduction to MicroTRAK/P18

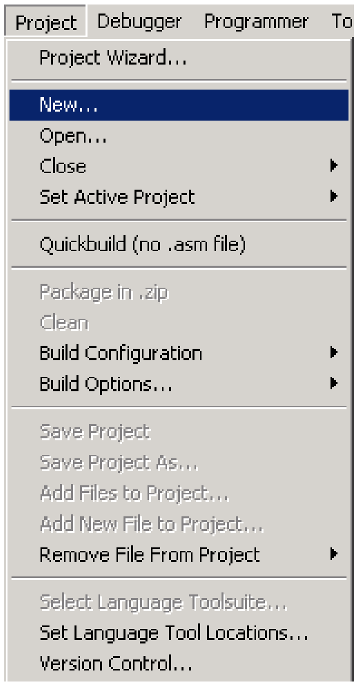
Overview

The purpose of this lab is to familiarize you with MicroTRAK/P18 and the program development environment. In this lab, you will create a simple program in C language, compile the program to form a hex file, download the program to the MINI-MAX/P18 Microcontroller board and execute the program.

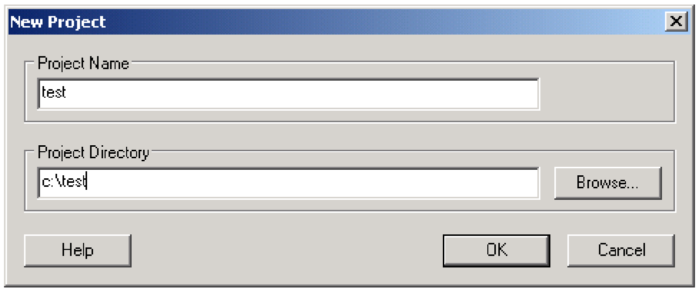
The knowledge developed in this lab will be very useful in subsequent labs when working with the PIC18F458 single-chip flash microcontroller to develop programs.

Instructions

To create your own project, start MPLAB. Select Project->New:



Specify project name and location:

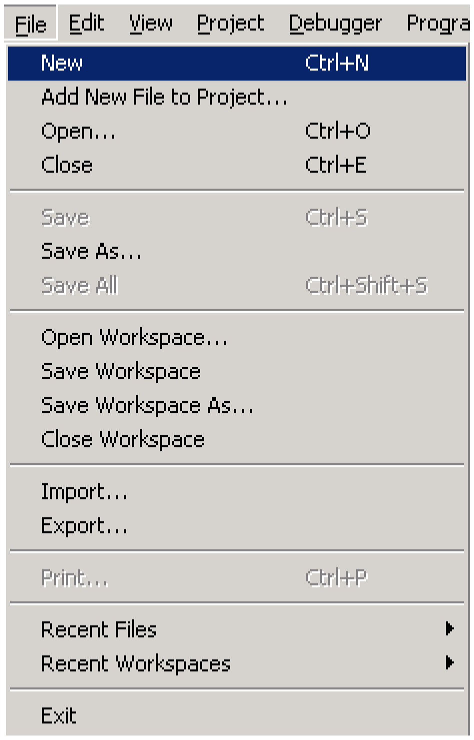

New Project

Project Name: test

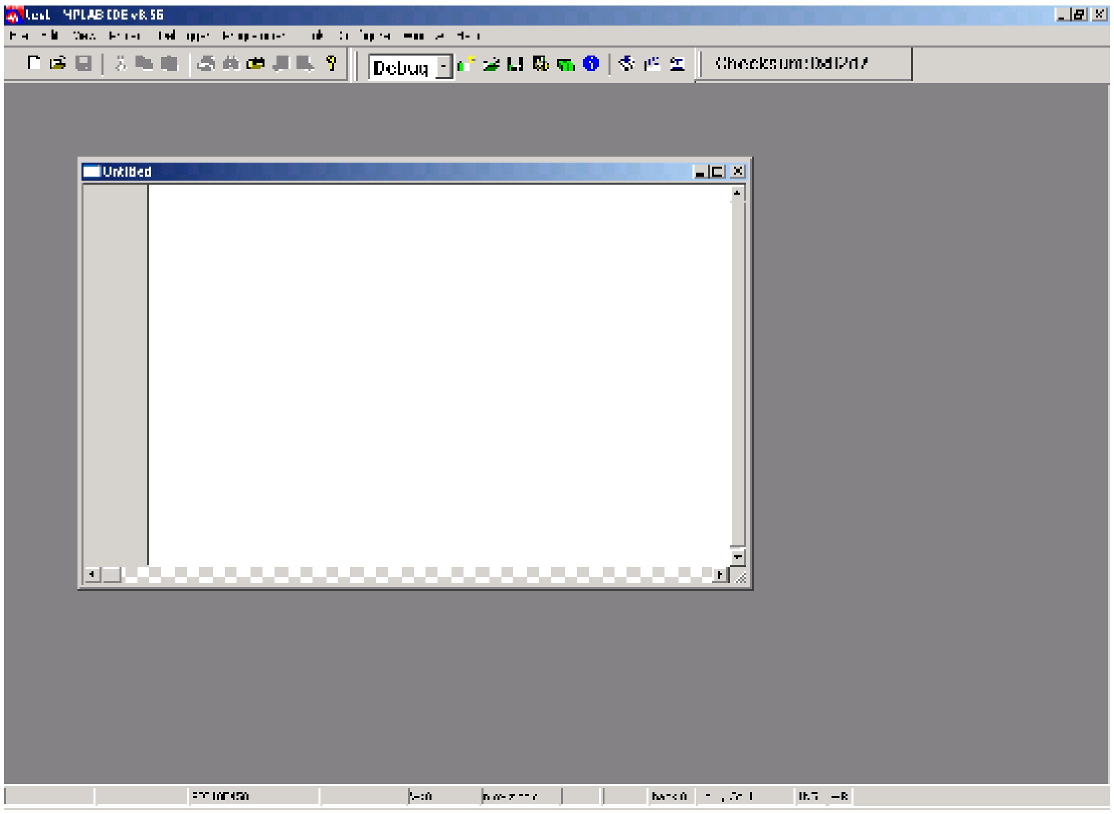
Project Directory: c:\test

Click OK.

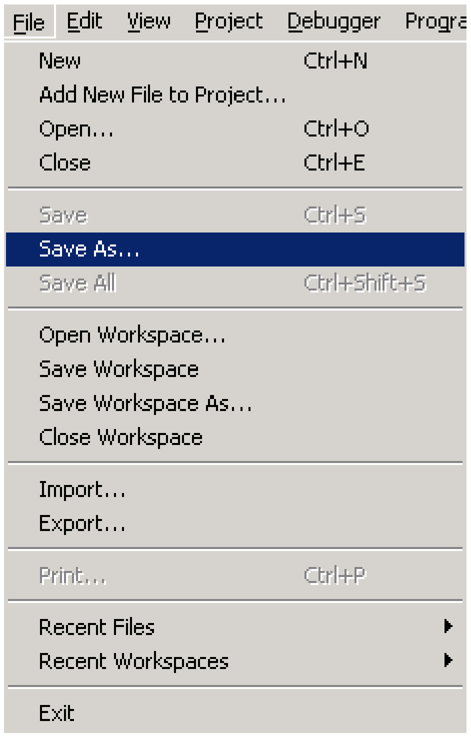
Create a new file by selecting File->New:



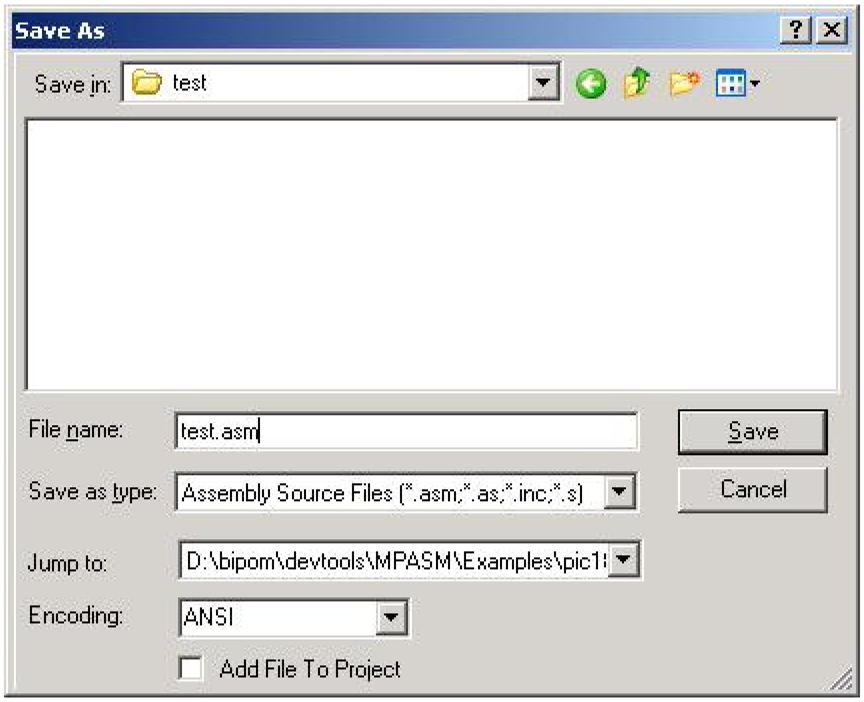
MPLAB will open a new window and show the empty file:



Select File->Save As:



Save the file as ***test.asm*** to c:\test folder:



Save in: test

File name: test.asm

Save as type: Assembly Source Files (*.asm;*.as;*.inc;*.s)

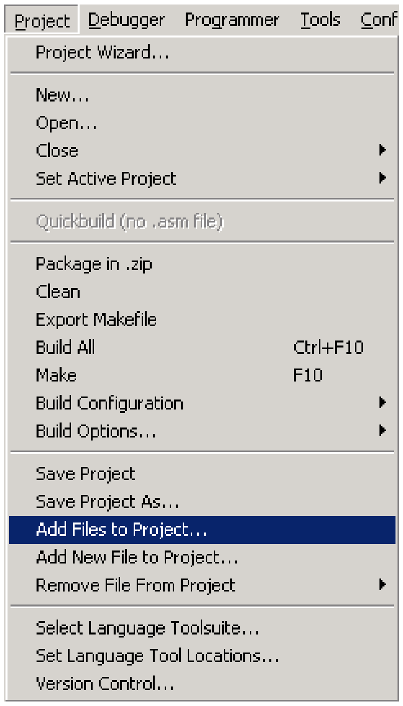
Jump to: D:\bipom\devtools\MPASM\Examples\pic1

Encoding: ANSI

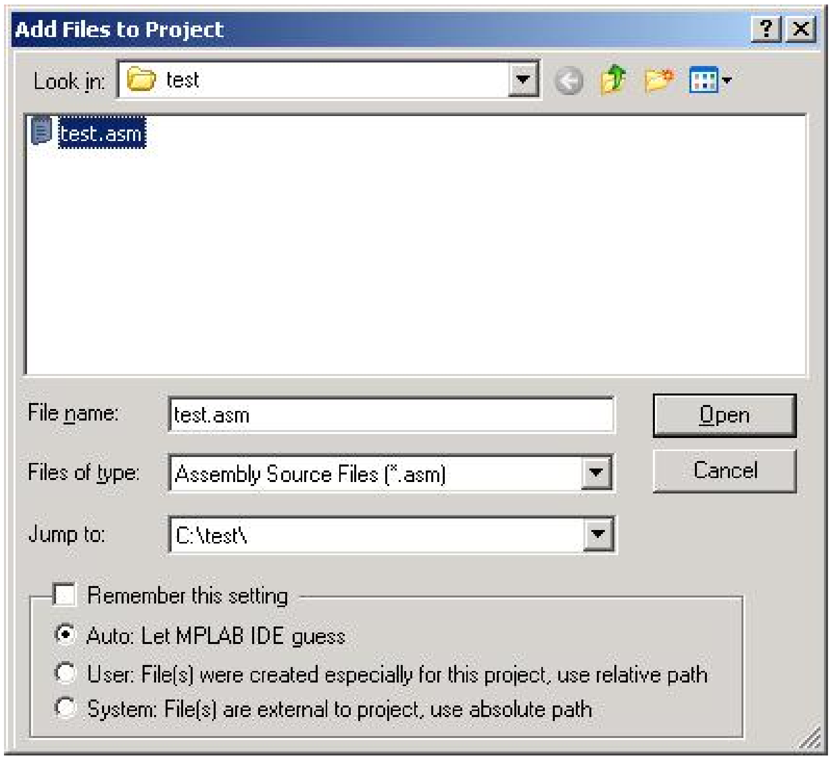
Add File To Project

Click Save.

Now add this new file ***test.asm*** to the test project. Select Project->Add Files to Project:



Select ***test.asm*** from ***c:\test*** to add to the project:



Look in: test

test.asm

File name: test.asm

Files of type: Assembly Source Files (*.asm)

Jump to: C:\test\

☐ Remember this setting

● Auto: Let MPLAB IDE guess

○ User: File(s) were created especially for this project, use relative path

○ System: File(s) are external to project, use absolute path

Click Open.

In the test.asm source file window, type the following to configure the program for PIC18F458 microcontroller:

```
list      p=18f458

#include <p18F458.inc>

CONFIG OSC = HS , BOR = ON , BORV = 45 , WDT = OFF, STVR = ON
```

Note: When typing, put a tab character before each line in the assembler. Otherwise, you may get warnings from the assembler.

list command specifies the microcontroller type as PIC18F458

#include defines the include file to include in this project. In this case, we use the include file p18F458.inc that contains the PIC18F458 specific register names and definitions.

CONFIG command specifies the special hardware configuration values for PIC18F458. In this case:

OSC = HS (this means, we selected high speed crystal as the clock source)

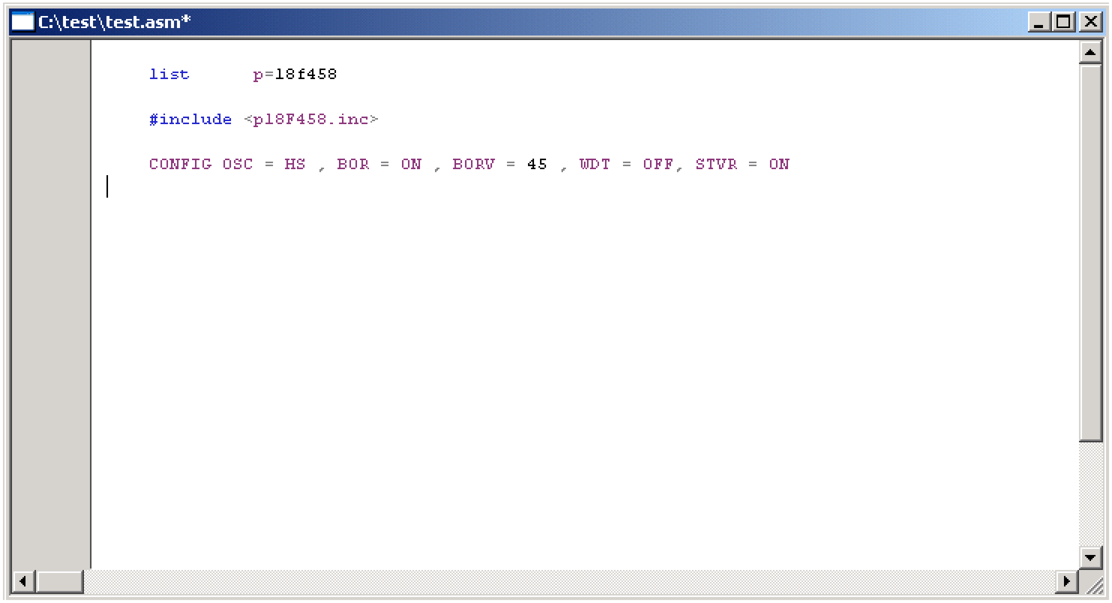
BOR = ON (this means that Brownout detect feature is enabled)

BORV = 45 (this means that the Brownout value is set at 4.5 Volts, supply voltages below 4.5 Volts will results in microcontroller reset)

WDT = OFF (Watchdog timer is disabled)

STVR = ON (A stack overflow will reset the PIC18F458)

The resulting source file window will look like this:



```
	list	p=18f458

	#include <p18F458.inc>

	CONFIG OSC = HS , BOR = ON , BORV = 45 , WDT = OFF, STVR = ON
```

Now type the following small assembly language program:

```
clrf      TRISB
bsf       PORTB, 6

movlw   b'00000000'
movwf   TRISC

movlw   b'11111110'
movwf   PORTC
```

b is the symbol for binary notation. For example, **b'11111110'** is the equivalent of **254** decimal or **FE** hex**.**

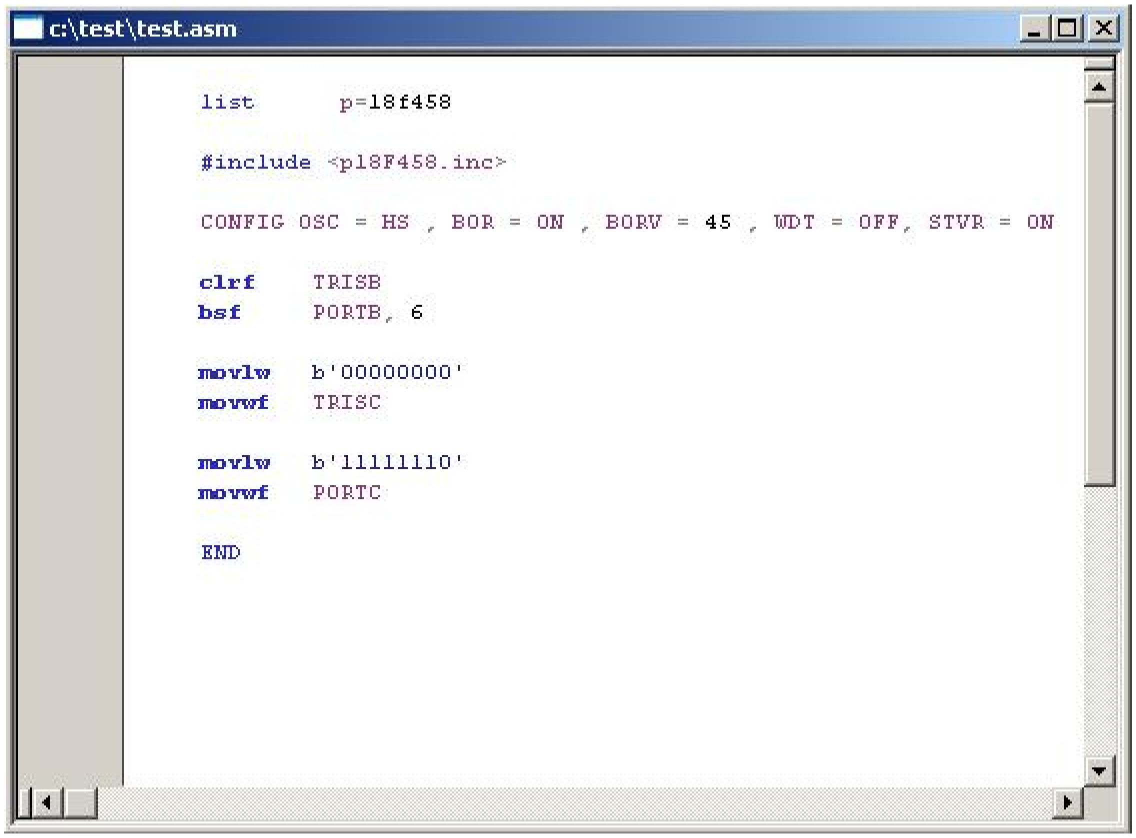
TRISB is the data direction register for I/O PORT B. **PORTB** is the data register for I/O PORT B.

TRISC is the data direction register for I/O PORT C. **PORTC** is the data register for I/O PORT C.

This simple program first sets port pin PORTB,6 to prevent reset on the MINI-MAX/P18 board (Otherwise, the secondary processor will reset the main processor within 2 seconds).

The program then sets first bit of PORT C (also referred to as RC0) as an output and then clears this bit.

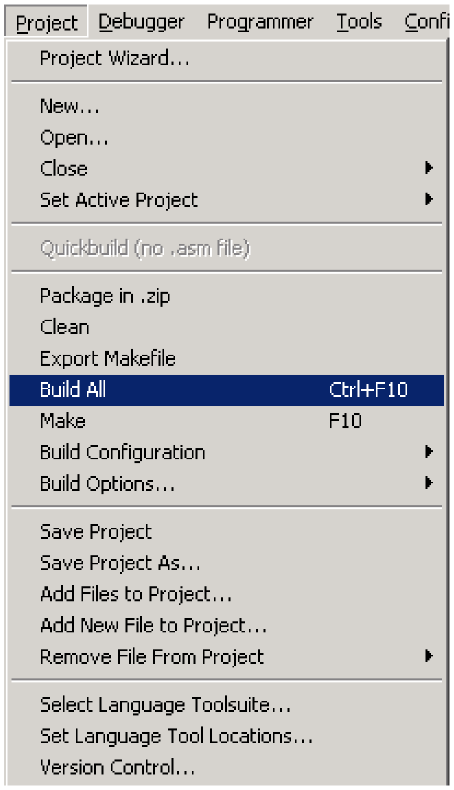
Finally, add the keyword **END** at the end of your program to tell the assembler where the program ends. The resulting source file will look like this:



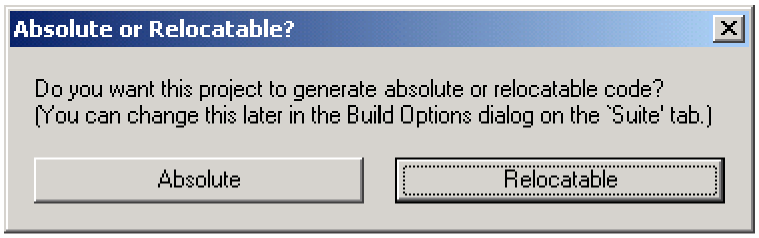
```
list      p=18f458

#include <p18F458.inc>

CONFIG OSC = HS , BOR = ON , BORV = 45 , WDT = OFF, STVR = ON

clrf     TRISB
bsf      PORTB, 6

movlw    b'00000000'
movwf    TRISC

movlw    b'11111110'
movwf    PORTC

END
```

Save the file by selecting File->Save.

The next step is to build the project. Select Project->Build All:



On the first build, MPLAB will prompt with the following question:

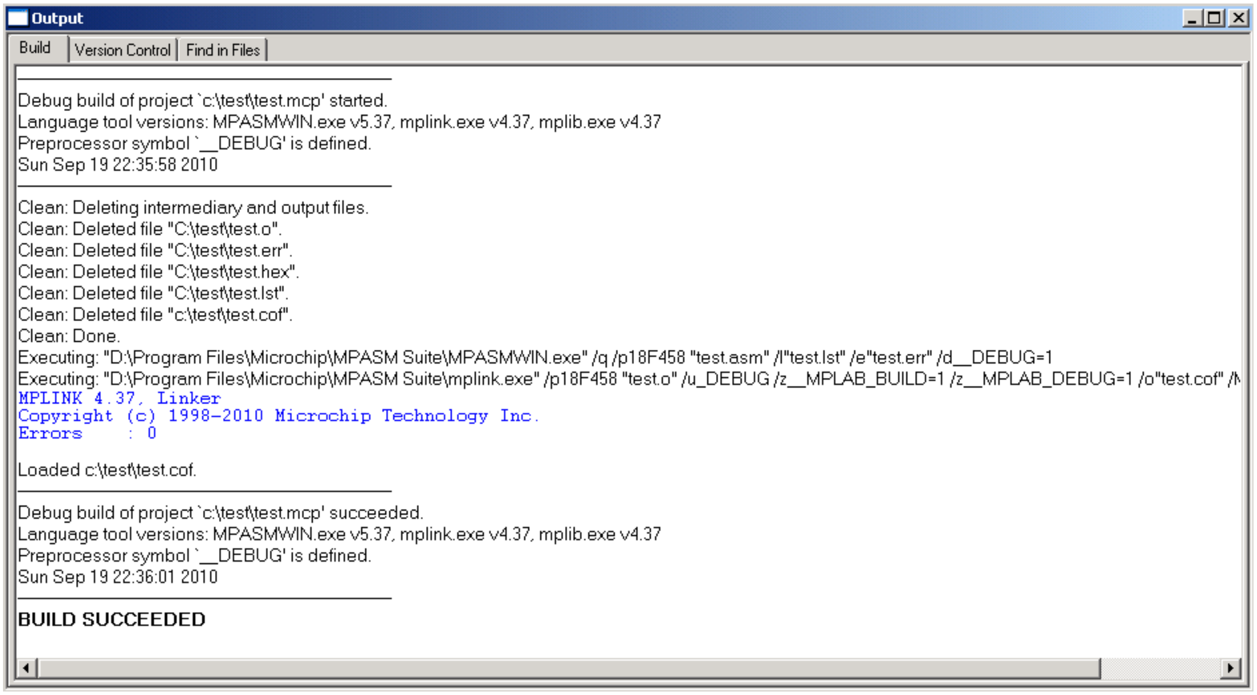


Absolute or Relocatable?

Do you want this project to generate absolute or relocatable code?
(You can change this later in the Build Options dialog on the `Suite' tab.)

Select Absolute.

MPLAB will open the Build Window and build the project successfully if the source file was typed correctly:



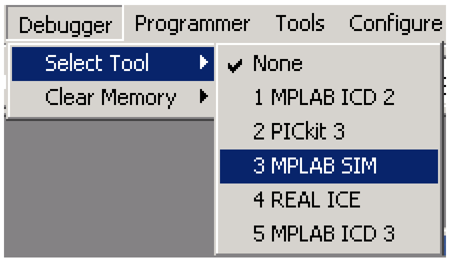
```
----------------------------------------------------------------------
Debug build of project `c:\test\test.mcp' started.
Language tool versions: MPASMWIN.exe v5.37, mplink.exe v4.37, mplib.exe v4.37
Preprocessor symbol `__DEBUG' is defined.
Sun Sep 19 22:35:58 2010
----------------------------------------------------------------------
Clean: Deleting intermediary and output files.
Clean: Deleted file "C:\test\test.o".
Clean: Deleted file "C:\test\test.err".
Clean: Deleted file "C:\test\test.hex".
Clean: Deleted file "C:\test\test.lst".
Clean: Deleted file "c:\test\test.cof".
Clean: Done.
Executing: "D:\Program Files\Microchip\MPASM Suite\MPASMWIN.exe" /q /p18F458 "test.asm" /l"test.lst" /e"test.err" /d__DEBUG=1
Executing: "D:\Program Files\Microchip\MPASM Suite\mplink.exe" /p18F458 "test.o" /u_DEBUG /z__MPLAB_BUILD=1 /z__MPLAB_DEBUG=1 /o"test.cof" /
MPLINK 4.37, Linker
Copyright (c) 1998-2010 Microchip Technology Inc.
Errors    : 0

Loaded c:\test\test.cof.
----------------------------------------------------------------------
Debug build of project `c:\test\test.mcp' succeeded.
Language tool versions: MPASMWIN.exe v5.37, mplink.exe v4.37, mplib.exe v4.37
Preprocessor symbol `__DEBUG' is defined.
Sun Sep 19 22:36:01 2010
----------------------------------------------------------------------
BUILD SUCCEEDED
```

The successful build generates several files, including ***test.hex*** which is the output file that will be downloaded to the MINI-MAX/P18 board.

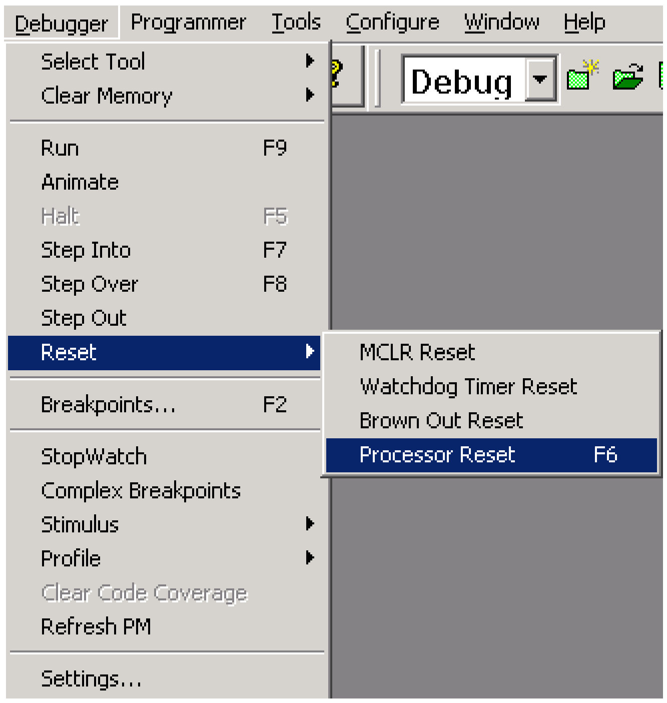
Simulation

You can simulate the microcontroller and single-step through your program without actually having the MINI-MAX/P18 connected to your PC. MPLAB has a built-in PIC® microcontroller simulator

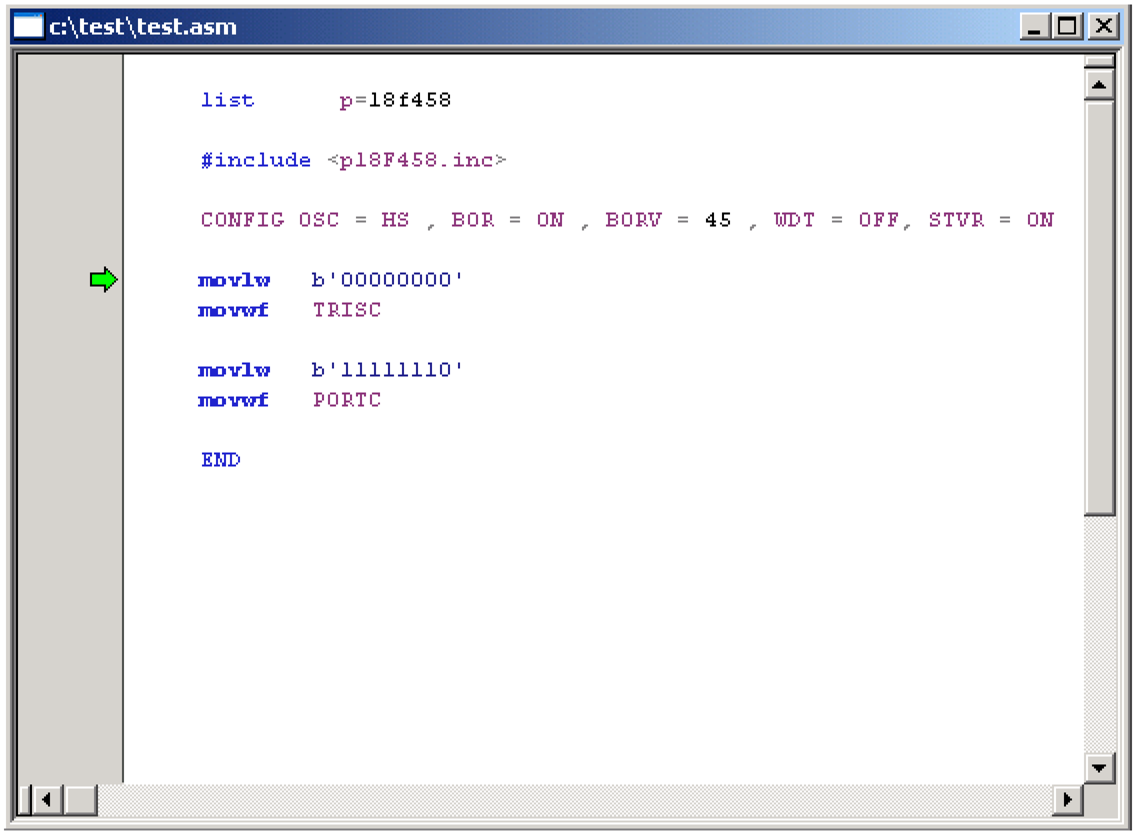
To start the simulation, first specify the simulator/debugger to use by selecting Debugger->Select Tool->MPLAB SIM. MPLAB SIM is the simulator that is built-in to MPLAB.



We will start the simulation by resetting the 18F458 microcontroller (in simulation mode). Select Debugger->Reset->Processor Reset:



This will reset the simulated microcontroller and place the current program position indicator (Program Counter) on the first executable line of code:



```
    list      p=18f458

    #include <p18F458.inc>

    CONFIG OSC = HS , BOR = ON , BORV = 45 , WDT = OFF, STVR = ON

    movlw   b'00000000'
    movwf   TRISC

    movlw   b'11111110'
    movwf   PORTC

    END
```

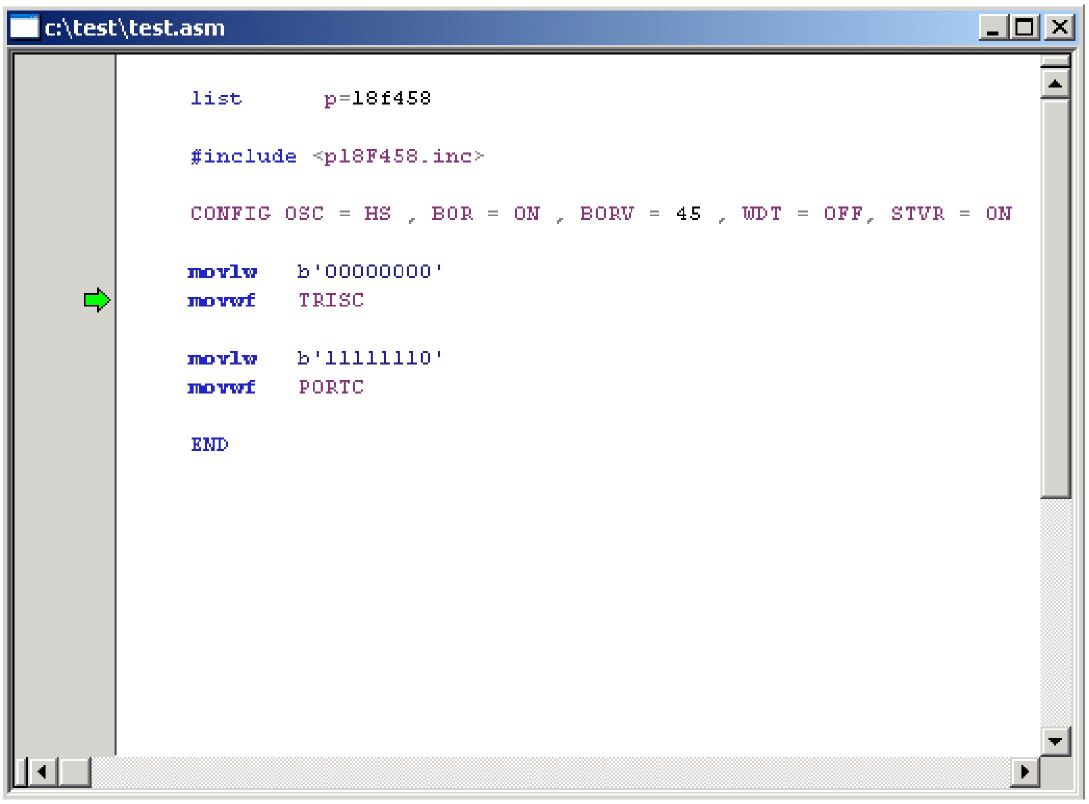
The green arrow shows the current instruction that is about to be executed.

Selecting the Debugger menu, you will notice that there are keyboard shortcuts for Simulator operations:

F7 for Step Into
F8 for Step Over
F9 for Run
F2 for Breakpoints

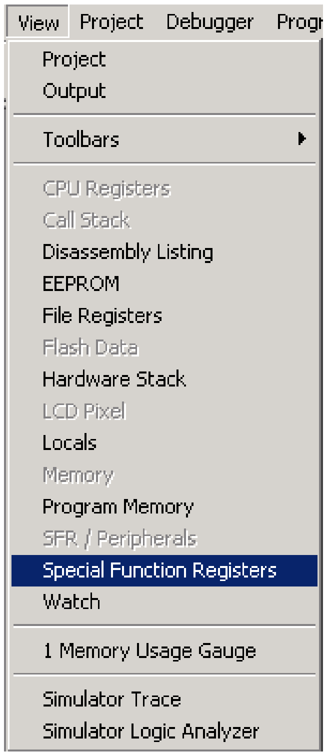
These keyboard shortcuts are faster and easier to use than menu operations and are recommended.

Press F7 and the green arrow will move to the next executable instruction:

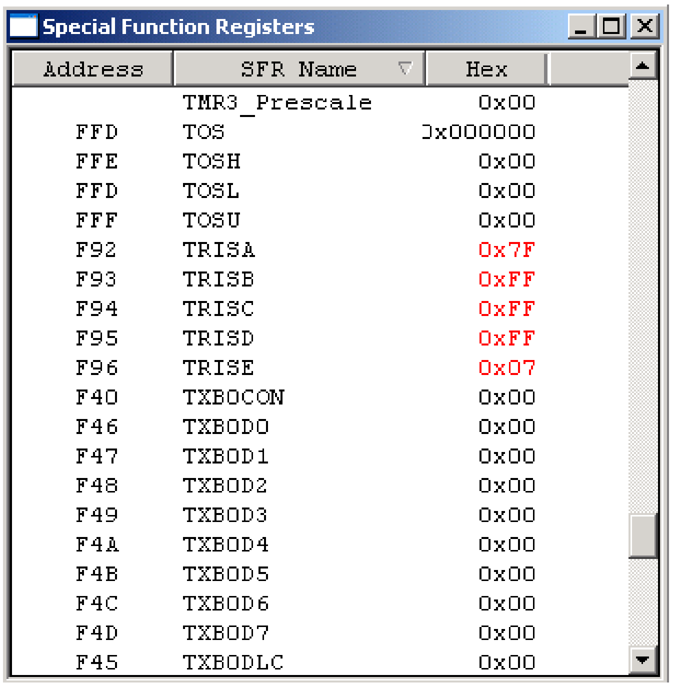


```
    list      p=18f458

    #include <p18F458.inc>

    CONFIG OSC = HS , BOR = ON , BORV = 45 , WDT = OFF, STVR = ON

    movlw   b'00000000'
    movwf   TRISC

    movlw   b'11111110'
    movwf   PORTC

    END
```

To view the simulated PIC18F458 registers, select View->Special Function Registers:



Special Function Registers window will appear:


Special Function Registers

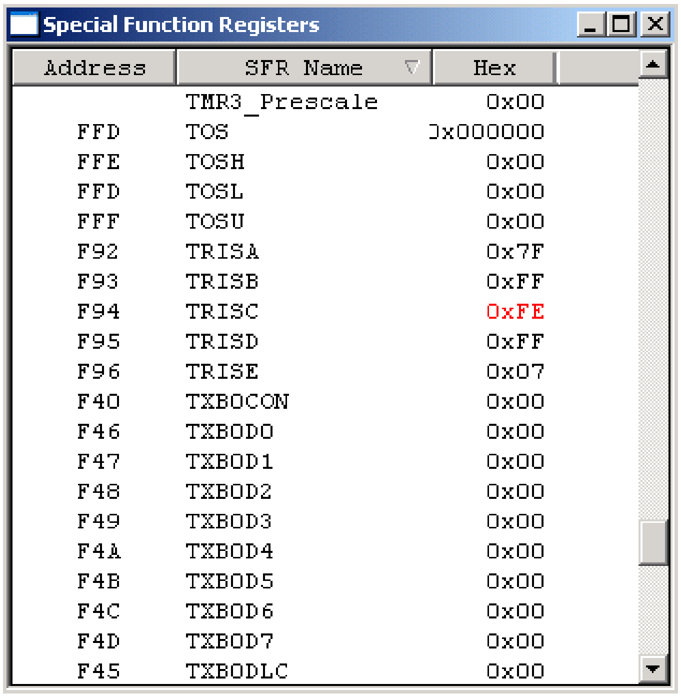
Address	SFR Name	Hex
	TMR3_Prescale	0x00
FFD	TOS	0x000000
FFE	TOSH	0x00
FFD	TOSL	0x00
FFF	TOSU	0x00
F92	TRISA	0x7F
F93	TRISB	0xFF
F94	TRISC	0xFF
F95	TRISD	0xFF
F96	TRISE	0x07
F40	TXB0CON	0x00
F46	TXB0D0	0x00
F47	TXB0D1	0x00
F48	TXB0D2	0x00
F49	TXB0D3	0x00
F4A	TXB0D4	0x00
F4B	TXB0D5	0x00
F4C	TXB0D6	0x00
F4D	TXB0D7	0x00
F45	TXB0DLC	0x00

Click on **SFR Name** column to sort by SFR name. Scroll to TRISC. The value of TRISC is now FF hex. This is the default value upon reset.

Press F7 to step one more instruction. This will cause the instruction

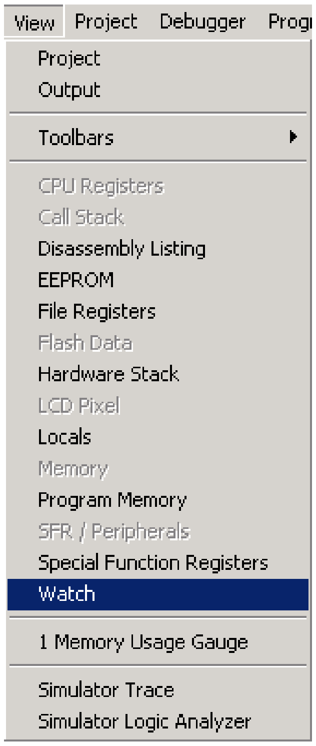
```
movwf   TRISC
```

to execute and this instruction will assign the value of W (Accumulator) to TRISC. Since W was assigned a value of b'11111110' (FE hex), TRISC will now have this value. The new value of TRISC is updated and shown in the Special Function Registers window:

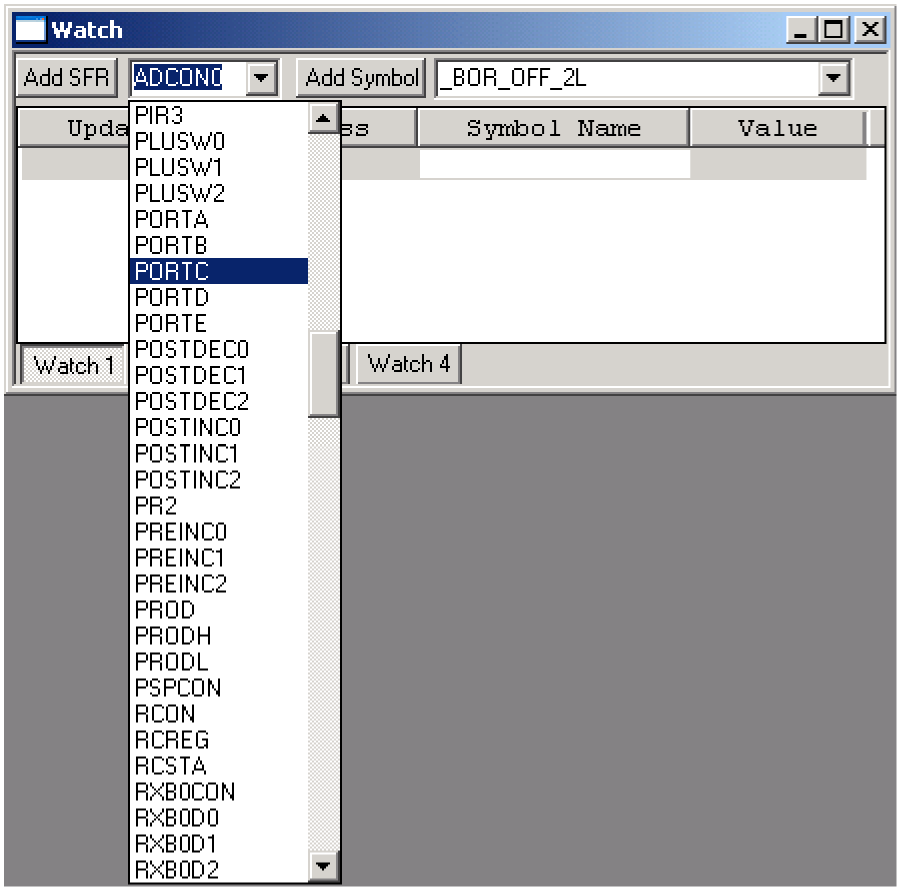


Address	SFR Name	Hex
	TMR3_Prescale	0x00
FFD	TOS	Ɔx000000
FFE	TOSH	0x00
FFD	TOSL	0x00
FFF	TOSU	0x00
F92	TRISA	0x7F
F93	TRISB	0xFF
F94	TRISC	0xFE
F95	TRISD	0xFF
F96	TRISE	0x07
F40	TXB0CON	0x00
F46	TXB0D0	0x00
F47	TXB0D1	0x00
F48	TXB0D2	0x00
F49	TXB0D3	0x00
F4A	TXB0D4	0x00
F4B	TXB0D5	0x00
F4C	TXB0D6	0x00
F4D	TXB0D7	0x00
F45	TXB0DLC	0x00

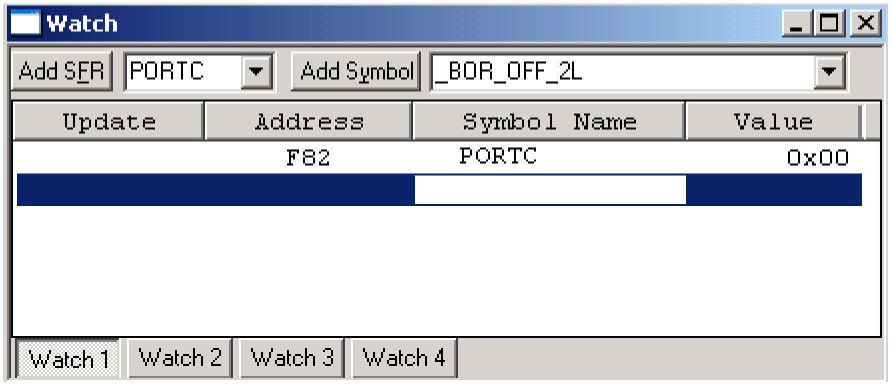
Another convenient method of watching variables and special function registers is to use the Watch window. Select View->Watch:



This will open Watch window. Select the Special Function Register to be watched from the left pull down list. In this case, we are using PORTC in our program so we can watch the value of PORTC:


Watch
Add SFR
ADCON0
Add Symbol
_BOR_OFF_2L
Upda
ss
Symbol Name
Value
Watch 1
Watch 4
PIR3
PLUSW0
PLUSW1
PLUSW2
PORTA
PORTB
PORTC
PORTD
PORTE
POSTDEC0
POSTDEC1
POSTDEC2
POSTINC0
POSTINC1
POSTINC2
PR2
PREINC0
PREINC1
PREINC2
PROD
PRODH
PRODL
PSPCON
RCON
RCREG
RCSTA
RXB0CON
RXB0D0
RXB0D1
RXB0D2

After selecting PORTC, click the Add SFR button. PORTC will be added to the list of variables to be watched and its current value will be displayed:

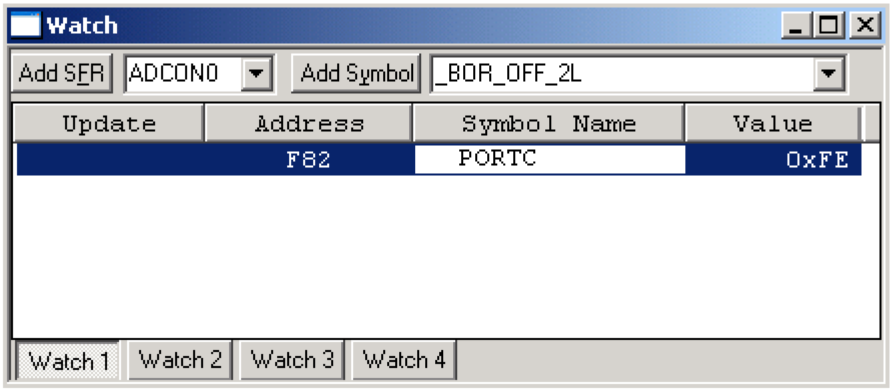


Update	Address	Symbol Name	Value
	F82	PORTC	0x00

Press F7 twice to single step over the lines:

movlw   b'11111110'
movwf   PORTC

The value of PORTC will change from to FE hex:

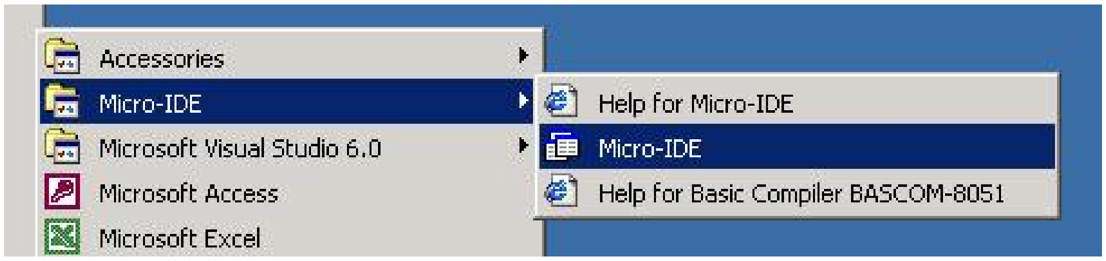


Update	Address	Symbol Name	Value
	F82	PORTC	0xFE

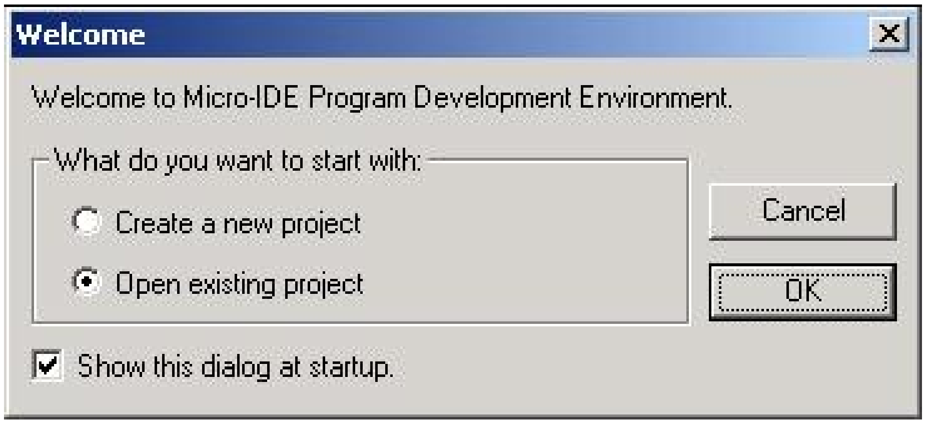
Downloading Programs

To download projects, Micro-IDE is used. We have created project files for both Micro-IDE and MPLAB for all the PIC® examples. The code can be developed and simulated in MPLAB and downloaded to the actual hardware using Micro-IDE. Micro-IDE also has a terminal emulator window that allows monitoring the RS232 serial output from the MINI-MAX/P18 board.

Start Micro-IDE by selecting Start, Programs and Micro-IDE. Select the Micro-IDE option under Micro-IDE folder. This will start Micro-IDE.



When Micro-IDE is started, the Project selection window appears:



Welcome to Micro-IDE Program Development Environment.

What do you want to start with:

Create a new project

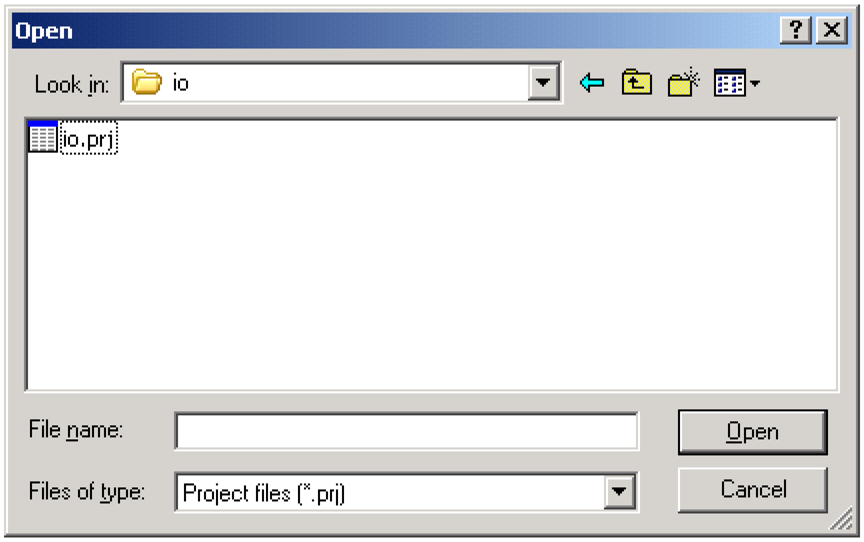
[✓ Open existing project]

[✓] Show this dialog at startup.

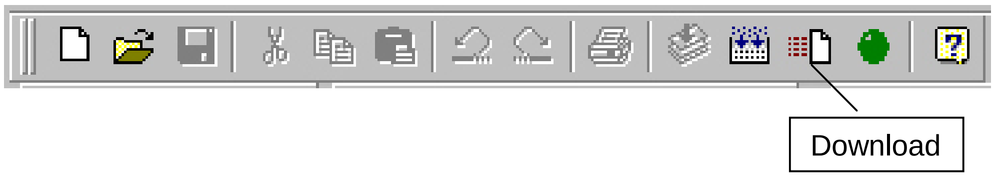
Click Cancel for this first time because you will first configure communications before opening a project.

Open the ***io.prj*** project that is under:

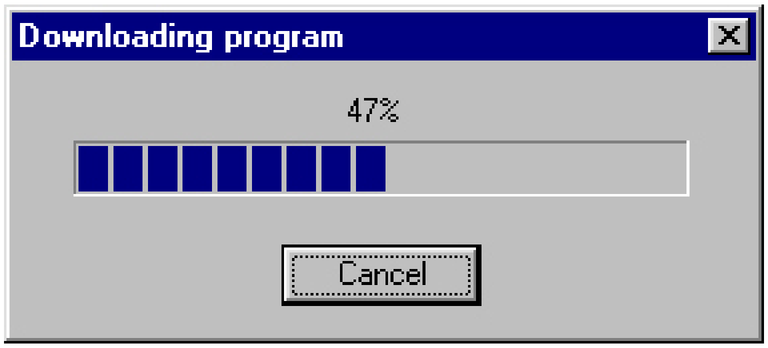
\bipom\devtools\MPASM\Examples\pic18



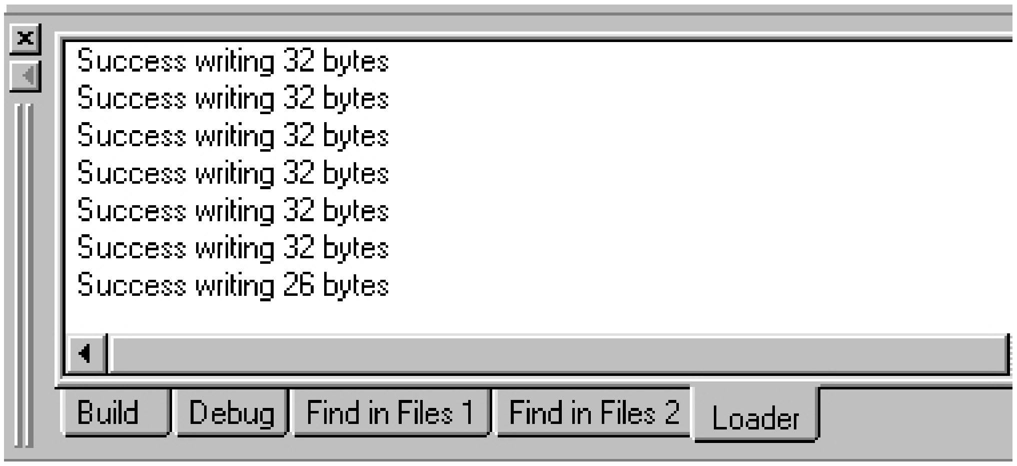
Download the file to the board by selecting Download under Build menu:



If the board is powered and connected properly to the PC, a progress dialog will appear:



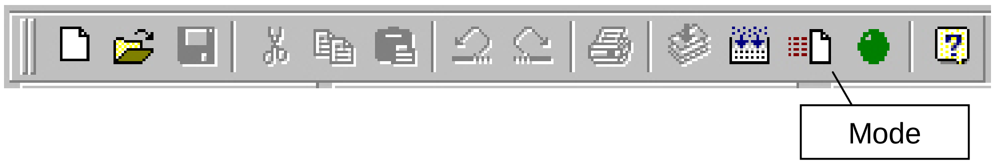
The progress dialog will disappear following a successful download. Details of the download are shown on the Output Window:



```
Success writing 32 bytes
Success writing 32 bytes
Success writing 32 bytes
Success writing 32 bytes
Success writing 32 bytes
Success writing 32 bytes
Success writing 26 bytes
```

When the download is finished, the progress indicator disappears. This means that the board has received the program successfully.

After the program has been successful downloaded, it can be started using the Mode button on the main Toolbar:



Mode button puts the board into **Run** or **Program** mode. In Run mode, the microcontroller is executing the program in its memory. In Program mode, the microcontroller is in Reset state so no programs are running. In Program mode, microcontroller's flash memory can be changed and a new program can be downloaded.

The Mode button is Red in Program mode and Green in Run mode. Following a download, the Mode button will be Red. Click the Mode button to change the mode to Run mode. The program **io.hex** that you just downloaded starts executing.

The program **io.hex** will blink the LED's that are connected to PORT D on MicroTRAK's I/O Module.

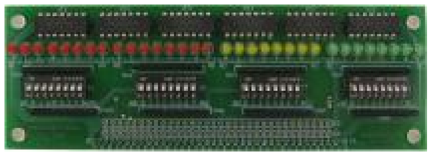
Congratulations !!! You have built and executed your first program on MicroTRAK.

Click the Mode button once again so it turns Red. The board is in Program mode now and it will stop running and the LED's will not blink any more.

LAB2 – Input/Output

Overview

This lab will familiarize you with the I/O Module included in MicroTRAK/P18 for monitoring and control all I/O ports of the microcontroller.



Information

I/O Module allows access to all input/output (I/O) ports of the PIC18F458 microcontroller on the MicroTRAK development platform. I/O Module has 32 switches to control the PIC18F458 microcontroller inputs and 32 LED's to indicate the port statuses as logic LOW or logic HIGH.

Signal	LED	Switch		Signal	LED	Switch
RE2	HL1	S1-1		RB7	HL17	S2-1
RE0	HL3	S1-2		RC2	HL19	S2-2
RE1	HL5	S1-3		RA0	HL21	S2-3
No Connect	HL7	S1-4		RA1	HL23	S2-4
RB0	HL9	S1-5		RC5	HL25	S2-5
RB1	HL13	S1-6		RA2	HL27	S2-6
RB2	HL15	S1-7		RC3	HL29	S2-7
RB3	HL17	S1-8		RC4	HL31	S2-8

Signal	LED	Switch		Signal	LED	Switch
RD0	HL2	S3-1		RC7	HL18	S4-1
RD1	HL4	S3-2		RC6	HL20	S4-2
RD2	HL6	S3-3		RB6	HL22	S4-3
RD3	HL8	S3-4		RB3 or RB5	HL24	S4-4
RD4	HL10	S3-5		RC1	HL26	S4-5
RD5	HL12	S3-6		RA4	HL28	S4-6
RD6	HL14	S3-7		RC0	HL30	S4-7
RD7	HL16	S3-8		RA5	HL32	S4-8

Exercise

Write a program that turns on all LED's connected to the corresponding ports (RA,RB,RC,RD) and reads status of the corresponding switches.

NOTE: There are a few "special" port pins you can change status (by switches, or by software), but this can disturb normal operation of the setup. You should exclude these pins in your program.

Special ports: PB6, RC6 and RC7

For example: If you set RC6 or RC7 to logic low, this will break serial communications with the PC since these pins are used as RS232 Transmit and Receive pins on PIC18F458.

PB6 is monitored by the secondary microcontroller (PIC16F648). If PB6 is set low, PIC16F648 will reset the main processor 18F458.

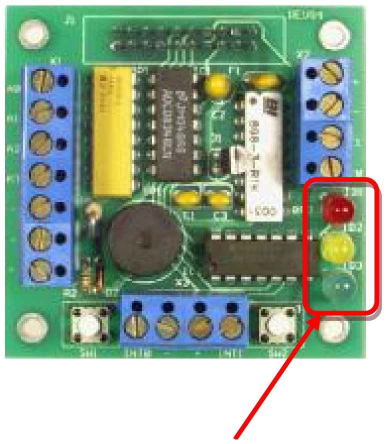
LAB3 – Traffic Lights

Overview

The purpose of this lab is to control the outputs of the micro-controller in a given sequence. Green, yellow and red Light Emitting Diodes (LED's) on the TB-1 board are connected to micro-controller outputs.

First, write a program to turn on only one LED and then turn off the same LED, Then, improve the program make the LED blink.

The next part of the lab is to read the input switches. As switches are mechanical objects, some de-bounce time (dead time) will also be placed in the program. You will control an LED with one switch: As long as switch is active, the corresponding LED is On and when switch is inactive, the corresponding LED is Off. Then, the LED will be made to blink as long as the switch is On. Finally, you will write a little traffic light controller where the green, red, yellow LED's on the TB-1 board simulate the traffic lights and the switches simulate the car presence sensors at a crossroad.



Information

Most of the microcontroller pins and the power supply pins are available on the 20-pin connector (J4) for interfacing to external circuitry, prototyping boards and peripheral boards, including the TB-1 Training Board. Table 2 shows the J4 pin-out of the microcontroller pins.

MINI-MAX/P18 Expansion (J4)

Signal	Pin	Pin	Signal
RC7	20	19	RC6
RB6	18	17	RB3 or RB5
RC1	16	15	RA4
RC0	14	13	RA5
RB7	12	11	RC2
RA0	10	9	RA1
RC5	8	7	RA2
RC3	6	5	RC4
VCC	4	3	GND
VCC	2	1	GND

Table 2

All of RC ports and some of RA and RB ports of PIC18F458 are accessible via the J3 connector. TB-1, which is connected to the microcontroller board as explained in Lab1, will be used as it already has input switches and output LED's connected. Table 3 describes the pin-out on the J3 connector and their designation as respective to input (switches) or output (LED's).

Microcontroller Pin	Connector on J3	Input or Output
RB6	18	Input, Active Low, Switch 1
RB3 or RB5	17	Input, Active Low, Switch 2
RA1	9	Output, Active High, Red LED
RC2	11	Output, Active High, Yellow LED
RB7	12	Output, Active High, Green LED

Table 3

Active Low means that the microcontroller must be programmed such that it will interpret the logic low as an active switch. Similarly, Active High means that to activate an LED, the corresponding microcontroller port pin must be high logic. The port direction registers (TRISA, TRISB, TRISC) define the state (Input or Output) of the microcontroller I/O lines.

<u>Exercise</u>

Initially turn the individual LED's On and Off without interaction with the switches. Write a program to:

- Turn Red LED On only for a brief time interval and than Off
- Turn Yellow LED On only for a brief time interval and than Off
- Turn Green LED On only for a brief time interval and than Off
- Blink Red LED only
- Blink all three LED's

You now have the capability of controlling more than one output line. Next step is to read an input line. When the Switch is not activated, the corresponding port is high and the LED should be turned On. When the Switch is activated, blink an LED:

- Turn Red Led On if Switch 1 is inactive.
- Turn Yellow Led On if Switch 2 is inactive.
- Blink Red Led as long as Switch 1 is active and Red Led is On if Switch 1 is inactive.
- Blink Yellow Led as long as Switch 2 is active and Yellow Led is On if Switch 2 is inactive.

Write a program to generate the results shown in Table 4 for a traffic light controller:

Switch 1	Switch 2	Red LED	Yellow LED	Green LED
Open	Open	OFF	OFF	ON
Open	Closed	ON	ON	OFF
Closed	Open	OFF	OFF	ON
Closed	Closed	OFF	OFF	OFF

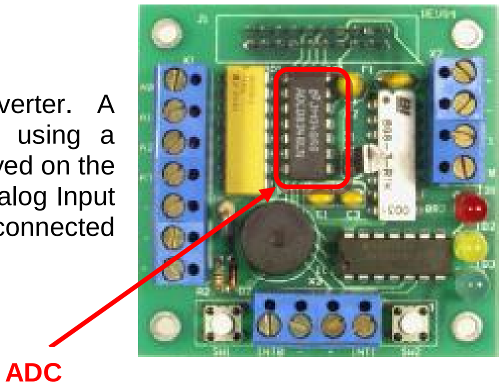
Table 4

Build the program and download to MicroTRAK/P18. Open and close Switch 1 and Switch 2 to change the LED's to make sure that your program is handling every case correctly.

LAB4 – Analog To Digital Conversion on TB-1

Overview

This lab is used to familiarize you with the operation of an A/D converter. A varying voltage is applied to the input of the A/D converter by using a potentiometer and its output is read into the micro-controller and displayed on the terminal screen. The A/D and the potentiometer are connected the Analog Input terminal blocks of TB-1. LM35 type temperature sensor will then be connected to the A/D and temperature values will be displayed on the board.



Information

Voltages that continuously vary as a function of time are defined as analog voltages and can have any value within certain range, e.g. 0V to 5V, or –5V to + 5V, or 0V to +12V etc. Digital voltages only have two values i.e. a 1 or a 0. An Analog to Digital Converter (ADC) converts the value of the analog voltage into digital format or code that is then processed by the microcontroller.

On the TB-1 Training Board, there is an 8-bit ADC (ADC0834 from National Semiconductor) as shown in Figure 5.

The pin-out for the 14-pin ADC0834 is shown in Table 5.

Pin#	Mnemonic	Description
1	V+	
2	CS	Active low
3	CH0	Analog Input Channel 0, 0V to 2.5V DC
4	CH1	Analog Input Channel 1, 0V to 2.5V DC
5	CH2	Analog Input Channel 2, 0V to 2.5V DC
6	CH3	Analog Input Channel 3, 0V to 2.5V DC
7	DGND	Digital Ground
8	AGND	Analog Ground
9	Vref	Reference voltage for the A/D, 2.5V DC
10	DO	Output Signal generated by the A/D
11	SARS	Output, when high A/D in progress, when low data output
12	CLK	Input, Clock signal to the A/D
13	DI	Input, Data to the A/D
14	VCC	Input, Power to the A/D

Table 5

Table 6 shows the connections of ADC0834 to J3 connector:

J3 Pin#	Microcontroller Pin#	A/D Mnemonic	A/D Pin#
14	RA5	CS	2
5	RC5	DO	10
N/C	N/C	SARS	11
6	RC3	CLK	12
13	RC5	DI	13

Table 6

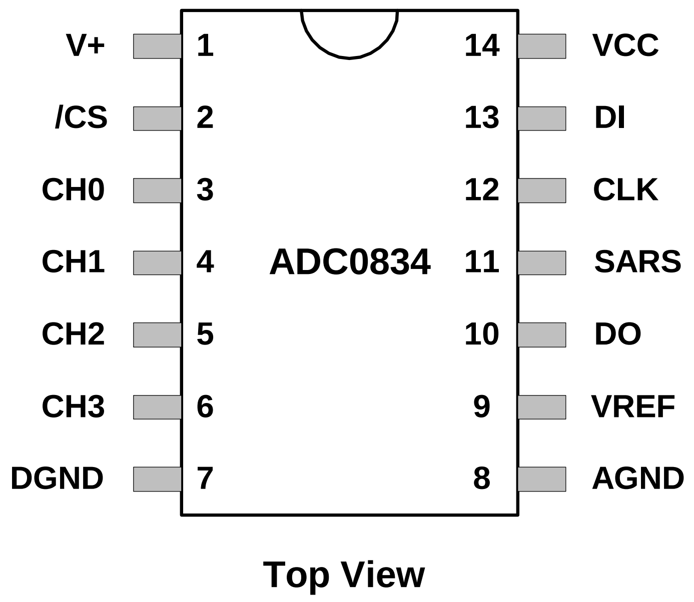


Figure 5

Single-Ended Multiplexer Mode

Multiplexer Address			Channel Number			
SGL/ DIF	ODD/ SIGN	SELECT 1	0	1	2	3
1	0	0	+			
1	0	1			+	
1	1	0		+		
1	1	1				+

Differential Multiplexer Mode

Multiplexer Address			Channel Number			
SGL/ DIF	ODD/ SIGN	SELECT 1	0	1	2	3
0	0	0	+	-		
0	0	1			+	-
0	1	0	-	+		
0	1	1			-	+

ADC0834 Timing

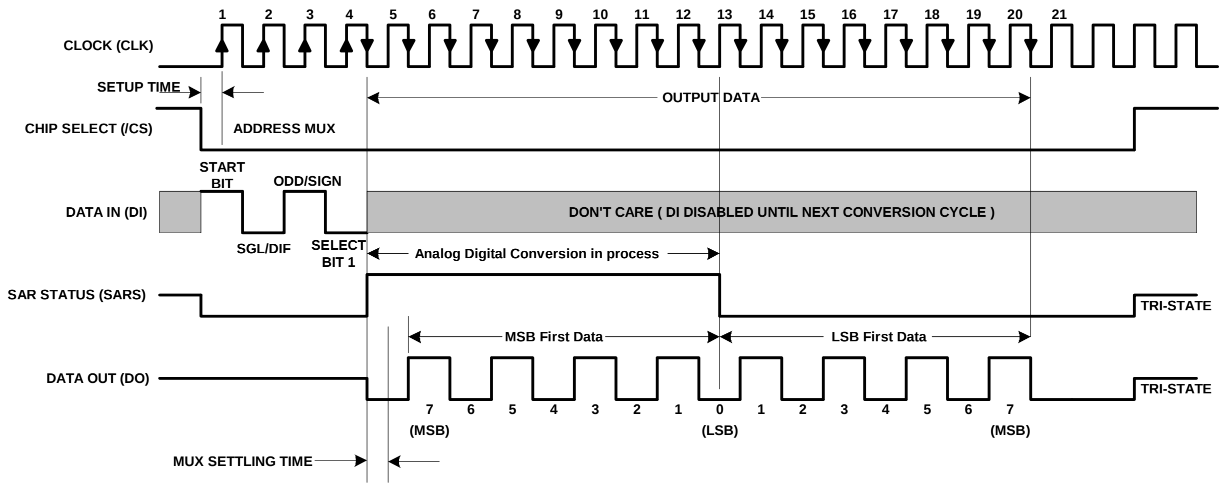


Figure 6

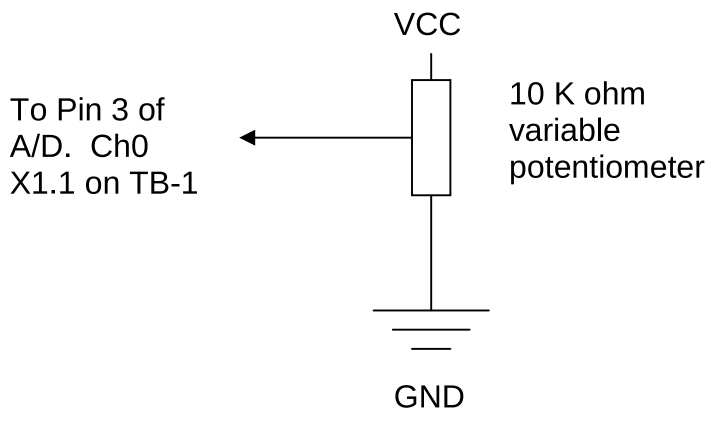


Figure 7

Pin 3 of the A/D receives the Analog input voltage signal. A DC Voltmeter when placed on Pin 3 should read between VCC and GND as the potentiometer is varied from one extreme to the other as shown in Figure 7. The LM35 temperature sensor will also be connected to this input later.

The A/D converter consists of 4-input multiplexed analog channels, which may be software configured as 4 single-ended channels or 2 differential channels, or a new pseudo differential option. The input format is assigned during MUX addressing sequence prior to start of conversion and this selects the analog input and their mode (single or differential).

Steps for A/D Conversion

- Initially the DI and CS inputs must be high.
- Pull the CS (chip select) line low and hold low for the entire duration of the conversion. The converter is waiting for the Start bit and the MUX assignment.
- A clock is generated by the microcontroller and sent to the ADC0834 clock input.
- The start bit is a logic "1" on the DI input line, after which the MUX assignment word on the DI line is presented. The status of the DI line is clocked on each rising edge of the clock line.
- The SARS status output line from the ADC0834 goes high, indicating that a conversion is now in progress and the DI line is disabled.
- The data out (DO) line comes out of Tri-state and provides a leading zero for one clock period.
- After 8 clock periods, the conversion is complete. The SARS line returns low half a clock cycle later.
- The DO line now generates the binary equivalent of the analog value as 8 output bits with Least Significant Bit (LSB) first. The DO line than goes low and remains low until Chip Select (CS) is made high. This clears all the internal registers and another conversion may now be started by making the CS go low again and repeating the process.

Exercise

Write a program that uses the A/D in single ended mode with the potentiometer connected to Channel 0 or Pin3 on the A/D through the terminal block on the TB-1 board. Vary the analog input voltage and display the value on the screen. Then, connect the potentiometer to Pin4 also and display its value. Connect the temperature sensor to Pin3 and potentiometer to Pin4 and display both the values simultaneously on the terminal screen.

LAB5 – Analog To Digital Conversion on MINI-MAX/P18

Overview

The purpose of this lab is to familiarize you with the 10-bit Analog Digital Converter (ADC) on the PIC18F458 on the MINI-MAX/P18 board. The potentiometer (that we used in Lab4) is connected to Analog Port Terminal on MINI-MAX/P18. Measurement results for all the channels will be displayed on the terminal screen.

Information

PIC18F458 has an A/D control register address - 0x17 and data register address-0x18.

A/D Control register description

Bit	Description
7	ADFM: A/D Result Format Select bit 1 = Right justified, 6 Most Significant bits of ADRESH are read as ‘0’ 0 = Left justified, 6 Least Significant bits of ADRESL are read as ‘0’
6	ADCS2: A/D Clock Divide by 2 Select bit 1 = A/D Clock source is divided by 2 when system clock is used 0 = Disabled
5-4	ADCS1:ADCS0: A/D Conversion Clock Select bits If ADSC2 = 0: 00 = FOSC/2 01 = FOSC/8 10 = FOSC/32 11 = FRC (clock derived from the internal A/D module RC oscillator) If ADSC2 = 1: 00 = FOSC/4 01 = FOSC/16 10 = FOSC/64 11 = FRC (clock derived from the internal A/D module RC oscillator)

A/D Control register description-continue

Bit	Description
3-0	PCFG<3:0>: A/D Port Configuration Control bits PCFG AN4 AN3 AN2 AN1 AN0 VREF+ VREF- C/R 0000 A A A A A AVDD AVSS 5/0 0001 A V+ A A A AN3 AVSS 4/1 0010 A A A A A AVDD AVSS 5/0

	0011	A	V+	A	A	A	AN3	AVSS	4/1
	0100	D	A	D	A	A	AVDD	AVSS	3/0
	0101	D	V+	D	A	A	AN3	AVSS	2/1
	011x	D	D	D	D	D	AVDD	AVSS	0/0
	1000	A	V+	V-	A	A	AN3	AN2	3/2
	1001	A	A	A	A	A	AVDD	AVSS	5/0
	1010	A	V+	A	A	A	AN3	AVSS	4/1
	1011	A	V+	V-	A	A	AN3	AN2	3/2
	1100	A	V+	V-	A	A	AN3	AN2	3/2
	1101	D	V+	V-	A	A	AN3	AN2	2/2
	1110	D	D	D	D	A	AVDD	AVSS	1/0
	1111	D	V+	V-	D	A	AN3	AN2	1/2

Legend:
A = Analog input
D = Digital I/O
V+ = VREF+
V- = VREF-
C/R = Number of Analog input channels/Number of A/D Voltage references

Exercise

Write a program that uses the built-in 10-bit ADC on PIC18F458, reads all channels and displays their values on the terminal screen. Connect the potentiometer consistently to the each of analog inputs, vary the analog input voltage and observe results.

LAB6 – Timers and Interrupts

Overview

This lab is designed to familiarize the student with timers and interrupts. External interrupts on switch closures are generated and background timers are made to run for timing certain external events. The switch and the LED's for this experiment are already present on the TB-1 board. Results will be displayed on the terminal screen.

Information

MicroTRAK/P18 uses the PIC18F458 microcontroller with interrupt capabilities.

An interrupt is executed when the microcontroller stops its normal code and branches to a predefined section of the memory to execute specific instructions. After this, the microcontroller goes back to its normal operation from where it had left before. Six interrupt vectors are present on the microcontroller:

- Two external interrupts INT0 and INT1
- Three timer interrupts Timer0, Timer1, Timer2
- One serial interrupt

Each of these interrupt sources can be individually enabled or disabled by setting or clearing a bit in the Special Function Register IE, which also contains global disable bit EA that disables or enables all the interrupts at once.

INT0 and INT1:

First exercise is to generate an external interrupt INT0 which is tied to port P3.2 (SW1 on TB-1). A change in the logic level of P3.2 will generate an interrupt in the program. Table 7 defines the registers and the bits that are used for controlling external interrupt 0 (INT0). The same holds true for INT1 except instead of IT0, IE0 and EX0 consider IT1, IE1 and EX1 and port number is P3.3 (SW2 on the TB-1).

INT0

Register	BIT	Description
TCON	IT0	Set than low level triggered, clear than edge triggered
TCON	IE0	Set by microcontroller when interrupt occurs and than automatically cleared when interrupt is processed
IE	EA	Set to enable global interrupt
IE	EX0	Set to enable external interrupt

Table 7

Every time SW1 is pressed on TB-1 (so that port pin P3.2 on the microcontroller is logic low) , the microcontroller will generate an interrupt and an action will occur, e.g. incrementing a counter and displaying its value on the terminal screen.

Timer0

Register TMOD is the timer counter mode control register and the TCON register turns the timer On/Off and indicates an overflow by setting a flag bit.

Register	**Bit**	**Description**
TCON	TR0	Set indicates Timer0 is On, clear indicates Timer0 is Off
IE	EA	Set indicates Global Interrupt is enabled
TCON	TF0	Set by the microcontroller when Timer0 overflows, cleared by the microcontroller when vectored to the interrupt service routine

The TMOD register enables the user to select different modes of operation for Timer 0 and Timer 1. Write a program to run a background timer of 30 milliseconds. Every time the 30-millisecond period is over, the program will jump to an interrupt routine and toggle the state of an I/O (P1.3 Red LED) line so the Red LED will blink every 30 milliseconds.

To further expand the program, a counter should also be incremented in the interrupt routine. When the counter exceeds a certain value (e.g. 1000) another I/O line (P1.1 Yellow LED) should toggle and the counter should reset back to zero (1000 * 30 milliseconds = 30,000 milliseconds = 30 seconds). Second LED will blink On and Off every 30 seconds which can be easily monitored using a wristwatch or PC's clock.

The same can be done with Timer1 where by TR1 and TF1 are used.

IE Register

Bit #	**Mnemonic**	**Description**
IE.7 (MSB)	EA	Global Interrupt Enable
IE.6	------	Not Implemented
IE.5	ET2	Timer2 interrupt enable bit.
IE.4	ES	Serial Port interrupt enable
IE.3	ET1	Timer1 interrupt enable bit
IE.2	EX1	External Interrupt 1 enable bit
IE.1	ET0	Timer0 interrupt enable bit
IE.0 (LSB)	EX0	External Interrupt 0 enable bit

Enable Bit = 1 enables the interrupt
Enable Bit = 0 disables the interrupt

TCON Register

Bit #	Mnemonic	Description
TCON.7 (MSB)	TF1	Timer1 Overflow flag
TCON.6	TR1	Set indicates Timer1 is On and Clear indicates Timer1 is Off
TCON.5	TF0	Timer0 Overflow flag
TCON.4	TR0	Set indicates Timer0 is On and Clear indicates Timer0 is Off
TCON.3	IE1	Set by microcontroller when Timer1 interrupt occurs and cleared by microcontroller when interrupt is processed
TCON.2	IT1	Set than low level triggered, clear than edge triggered
TCON.1	IE0	Set by microcontroller when Timer0 interrupt occurs and cleared by microcontroller when interrupt is processed
TCON.0 (LSB)	IT0	Set than low level triggered, clear than edge triggered

LAB7 - 4x4 Keypad

Overview

Student connects a keypad to the keypad connector on the MicroTRAK Carrier Board and programs the keypad.

Information

The MicroTRAK Carrier Board has a 10-pin keypad connector, which is connected to Port 2 of the microcontroller. The pin-out of the connector is shown below in Table 8.

Microcontroller Port Pin	**Keypad Connector Pin#**
P2.0	1
P2.1	2
P2.2	3
P2.3	4
P2.4	5
P2.5	6
P2.6	7
P2.7	8
Gnd	9
Vcc	10

Table 8

Many keypads are wired as a matrix of rows and columns. The internal connections of a 4-row by 4-column keypad are shown in Figure 8.

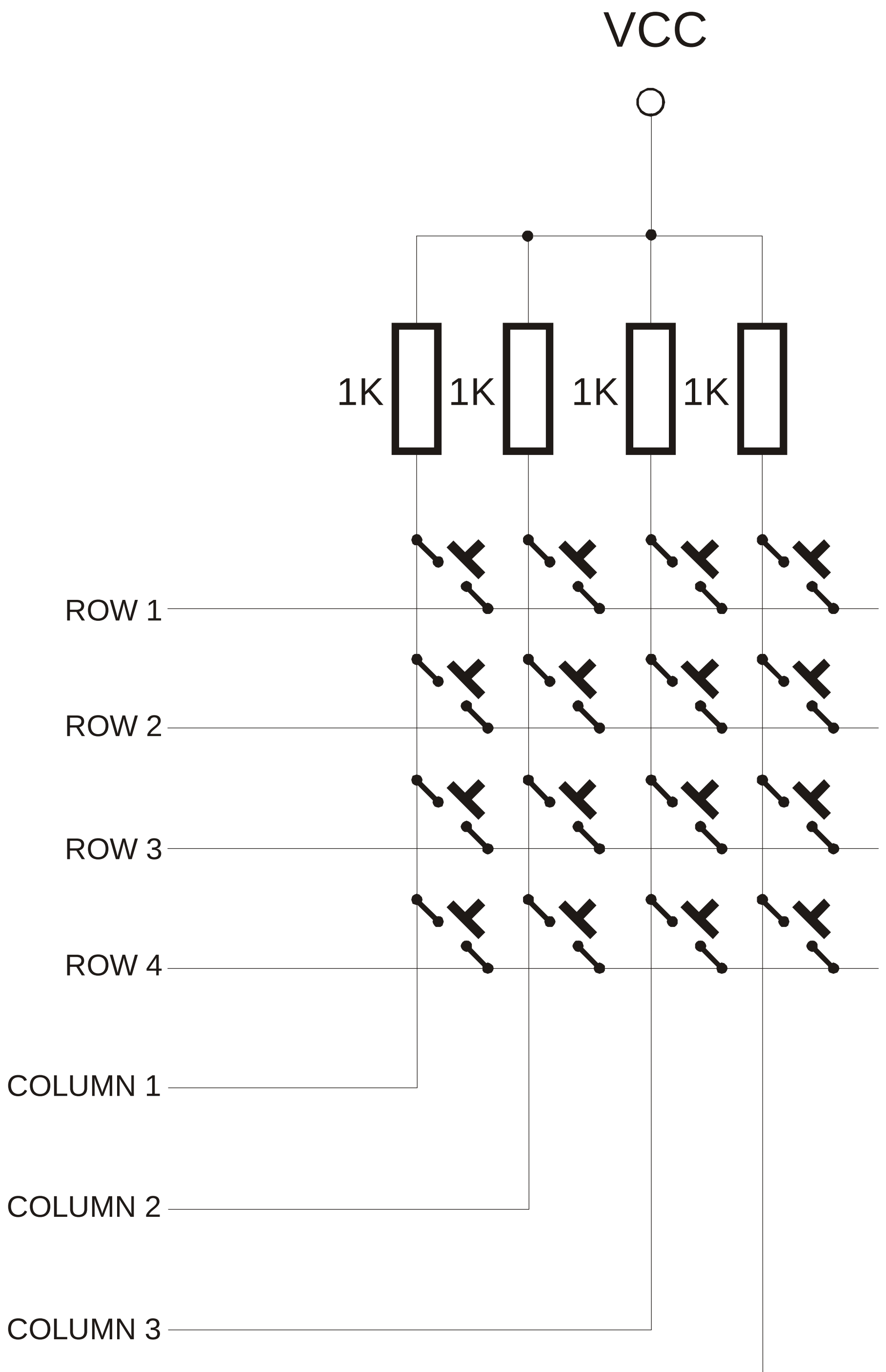

VCC
1K
1K
1K
1K
ROW 1
ROW 2
ROW 3
ROW 4
COLUMN 1
COLUMN 2
COLUMN 3
COLUMN 4

Figure 8

Matrix connection saves on the number of connections and microcontroller port lines. For example, a 4-row by 4-column keypad would require 17 wires (16 + ground) if each key was individually connected to microcontroller ports. Using the matrix approach and scanning the keypad under software control reduces the number of wires and port pins to 8 (4 rows + 4 columns).

When a key is pressed, the row for that key will be physically connected to the column for that key. Therefore, the port input for the column will be at the same logic level as the port output for the row.
Since the columns (inputs) are normally at the HIGH logic level due to pull-up resistors, the only way to make a column LOW will be to press a key and make the row for that key LOW. By periodically strobing each row LOW one row at a time, and reading the column input levels during each strobe, one can determine which key is pressed.

This is illustrated by Table 9 for the 4 by 4 keypad. In the Row Mask, Row 1 is assigned to the Most Significant Bit and Row 4 is assigned to the Least Significant Bit. Similarly, in the Column Mask, Column 1 is assigned to the Most Significant Bit and Column 4 is assigned to the Least Significant Bit.

Action	**Row Mask**	**Column Mask**
No keys were pressed	XXXX	1111
Row 1 Column 1 key pressed	0111	0111
Row 1 Column 2 key pressed	0111	1011
Row 1 Column 3 key pressed	0111	1101
Row 1 Column 4 key pressed	0111	1110
Row 2 Column 1 key pressed	1011	0111
Row 2 Column 2 key pressed	1011	1011
Row 2 Column 3 key pressed	1011	1101
Row 2 Column 4 key pressed	1011	1110
Row 3 Column 1 key pressed	1101	0111
Row 3 Column 2 key pressed	1101	1011
Row 3 Column 3 key pressed	1101	1101
Row 3 Column 4 key pressed	1101	1110
Row 4 Column 1 key pressed	1110	0111
Row 4 Column 2 key pressed	1110	1011
Row 4 Column 3 key pressed	1110	1101
Row 4 Column 4 key pressed	1110	1110

Table 9

Port2 is connected to the Keypad connector with the configuration shown in Table 10.

Port2.0	Row1	Output
Port2.1	Row2	Output
Port2.2	Row3	Output
Port2.3	Row4	Output
Port2.4	Column 1	Input
Port2.5	Column 2	Input
Port2.6	Column 3	Input
Port2.7	Column 4	Input

Table 10

In the port direction register, the port pins connected to rows are defined as outputs and the port pins connected to columns are defined as inputs. Each key on the keypad is assigned a given value by the programmer before hand.

The Keypad algorithm can be based on the following rules.

Algorithm

- A Column is generally high (output).
- One row at a time is made to go low (input) and than the columns are read.
- If one or more columns are low than the switches of the corresponding columns are active and their respective values should be displayed on the terminal.

Exercise

Determine the pin-out and the matrix layout of your keypad using the ohmmeter function of your multi-meter. Write a program that displays – on the terminal screen- the key being pressed on the keypad.

LAB8 – Liquid Crystal Display (LCD)

Overview

This Lab familiarizes the student with industry-standard alphanumeric LCD's by connecting the LCD to the LCD connector of MicroTRAK/P18 and writing a program in C to display various characters using 4-bit mode.

Information

Dot matrix LCD displays are readily available from many companies such as Sharp, Hitachi and OPTREX. The LCD's generally come in display formats of 16 X 1, 16 X 2, 24 X 2 and 40 X 4 (column X row). These typically have 8 data lines DB0 – DB7 (Data 0 through Data 7), VCC (Power), GND (Ground), RS (Register Select), R/W (Read/Write), E (Enable).

This exercise will use a 24 X 2 display, to be connected to the LCD connector. The three control lines are explained below.

RS Register Select Control
1 = LCD in data mode
0 = LCD in command mode

E Data / Control state
Rising Edge = Latches control state
Falling Edge = Latches data

R/W Read / Write control
1 = LCD to write data
0 = LCD to read data

In the 4-bit mode, data is transferred either on the lower or upper nibble of the port, this saves in I/O lines but the program occupies more space as two commands are required to display a character.

Table 11 shows the connection between the display and the LCD connector in 4-bit mode with low nibble.

LCD Connector Pin#	**Microcontroller Pin**	**LCD Display Pin**	**LCD Display Pin#**
1	GND	GND	1
2	VCC	VCC	2
3	Vee (PIC)	VEE	3
4	RE2	RS	4
5	RE0	R/W	5
6	RE1	E	6
7	not connected	not connected	7
8	not connected	not connected	8
9	not connected	not connected	9
10	not connected	not connected	10
11	RB0	DB4	11
12	RB1	DB5	12
13	RB2	DB6	13
14	RB3	DB7	14

Table 11

Exercise

Using the LCD datasheet, write a program that displays “Hello World” on the first row of the LCD. Then, display the same message on the second row of the LCD.

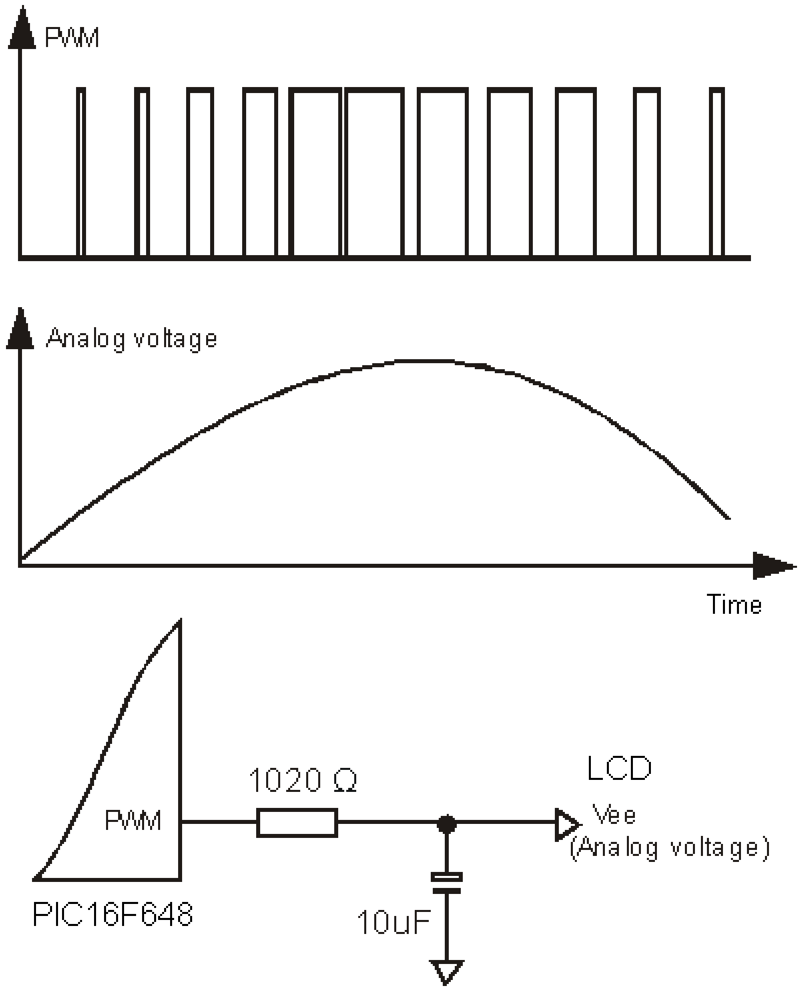
LAB9 - How to adjust LCD contrast

Overview

Student adjusts LCD contrast, using DAC embedded in secondary microcontroller PIC16F648 on MINI-MAX/P18 board.

Information

The contrast of the LCD module is adjusted by means of the Vee Voltage (pin 3). Port pin RB3 of secondary PIC controller PIC16F648 on the MINI-MAX/P18 is connected to Vee pin of the LCD through a low-pass (RC) filter. This enables software contrast adjustment. PIC16F648 communicates with the main microcontroller PIC18F458 using I2C 2-wire communications and acts as a slave peripheral device. 16F648 factory firmware has a built-in Pulsed Width Modulation (PWM) feature that can be controlled from the main microcontroller. PWM output produces a digital waveform with programmable duty cycle and is converted to an analog voltage using a low-pass filter is used (see figure below):



PIC16F648 factory firmware has two PWM modes: 4-bit and 10-bit PWM.

For setting the 4-bit PWM value, you should send a numeric value from 0 to 15:

4 bits PWM							
0	0	0	0	PWM3	PWM2	PWM1	PWM0

For setting the 10-bit PWM value, you should first send the highest 4 bits of 10-bit PWM value and then you should send lowest 6 bits + 0x40 (bit 6 set) as second byte:

	10 bits PWM							
first byte	0	0	0	0	PWM9	PWM8	PWM7	PWM6
second byte	0	1	PWM5	PWM4	PWM3	PWM2	PWM1	PWM0

Exercise

Write a program that uses 4-bit PWM for contrast adjustment and displays on the first row of the LCD the current contrast.

Then, improve the program to use 10-bit PWM.

LAB10 - Buzzer

Overview

Student uses Pulse Width Modulation (PWM) techniques to vary sounds from the buzzer on the TB-1 to generate different notes using software. Student then generates a little musical piece using the notes that he/she programmed.

Information

A buzzer or simple speaker will generate music when a series of square waves is applied to its positive input with the negative grounded. The frequency of the square wave should be less than 12KHz to be audible. Varying the frequency of the square wave will generate different musical tones.

The buzzer on the TB-1 is connected to port pin RA0 on the micro controller. This pin needs to be programmed as an output pin and square waves of varying frequencies and duty cycle need to be generated. Since the Pulse Width Modulation feature is on port pin RC2, we will connect RA0 and RC2 together using a small jumper wire.

Exercise

- Connect microcontroller port pins RA0 and RC2 together as shown. You can use a small jumper wire or an E-Z-Hook wire from BiPOM (Part number 9110-4) for this purpose.

- Program RC2 on the microcontroller as an output pin
- Decide a period or frequency of square wave between 1KHz to 15KHz.
- Generate a signal of 50% Duty cycle, where
  Duty Cycle = On Time / (On Time + Off Time) of square wave.
- Activate buzzer for approx 5 to 10 seconds
- Change frequency of buzzer and note the different sound
- Change only duty cycle and note the different intensity.