

# MICROCHIP PICKit 2 ®

## DESCRIPTION

The Microchip PICKit 2 device programmer and in-circuit debugger consists of software which runs on a PC and hardware. Programmer-only software comes with the PICKit 2. MPLAB includes both programming and in-circuit debugging software.



PICKit 2 shown with AC164110 RJ-11 to ICSP adapter including cable (attached)

With the PICKit 2, you can step through assembly source code on-screen while observing what the hardware is (or is not) doing. You can select registers to watch (view) by their labels (names) as you step through the program. You can set a breakpoint (place to stop program execution) and run the program at normal speed and exercise the hardware up to that breakpoint. Then you can look at the contents of the registers of your choosing.

The unit has a USB connector for serial communication with a host PC and a 6-pin female header for communication with a flash PIC microcontroller. The PIC microcontroller resides on a so-called "target" board which is the user's (your) board.

A Microchip PN AC164110 RJ-11 to in-circuit serial programming adapter facilitates connecting the PICKit 2 to a target board equipped with an RJ-11 phone jack. The PN AC164110 includes a 5 inch long 6-conductor phone cable.

The PICKit 2 can operate in two operating modes:

- Programmer mode
- Debugger mode

In the programmer mode, only your code is programmed into the device for stand alone (without the PICKit 2 connected) operation.

In both the programmer and debugger operating modes, the user code is programmed into the PIC microcontroller. In the debugger mode, the PIC debugger code used by the PICKit 2 is also programmed into program memory locations reserved for the purpose.

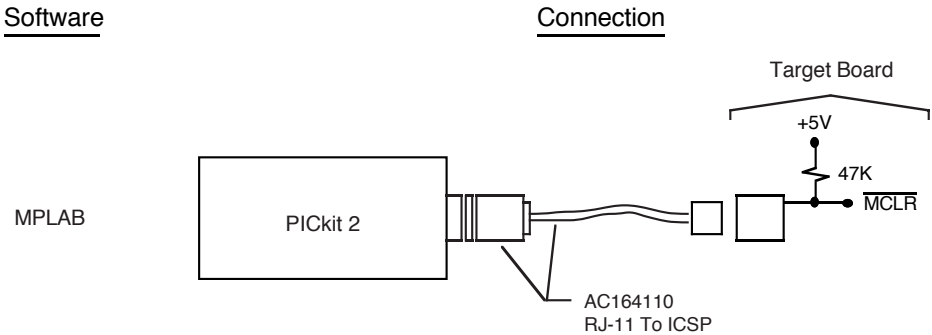
The PICKit 2 can operate in two hardware configurations:

- Device programmer for programming flash devices. Microchip offers a universal programming module (UPM would you believe!)(PN AC162049) which connects to the 6-conductor modular cable. It has a 40-pin zero insertion force (ZIF) socket. The PICKit 2/UPM combination allows users to program devices in a ZIF socket rather than on a target (user) board. As an alternative, you may easily build your own programmer boards for the devices you use. See the information which is also provided at the PICKit 2 link on the [sq-1.com](http://sq-1.com) web site entitled "Programming A PIC16F84A Using The PICKit 2 And A Simple Programming Adapter".
- Debugger/programmer for working with flash devices with in-circuit debugging capability. The PIC microcontroller resides on the user's application board called the "target" board. The PIC microcontroller may be programmed for testing and debugging (debugger mode) or programmed for stand alone operation (programmer mode).

For debugging low pin count devices, interface boards are required:

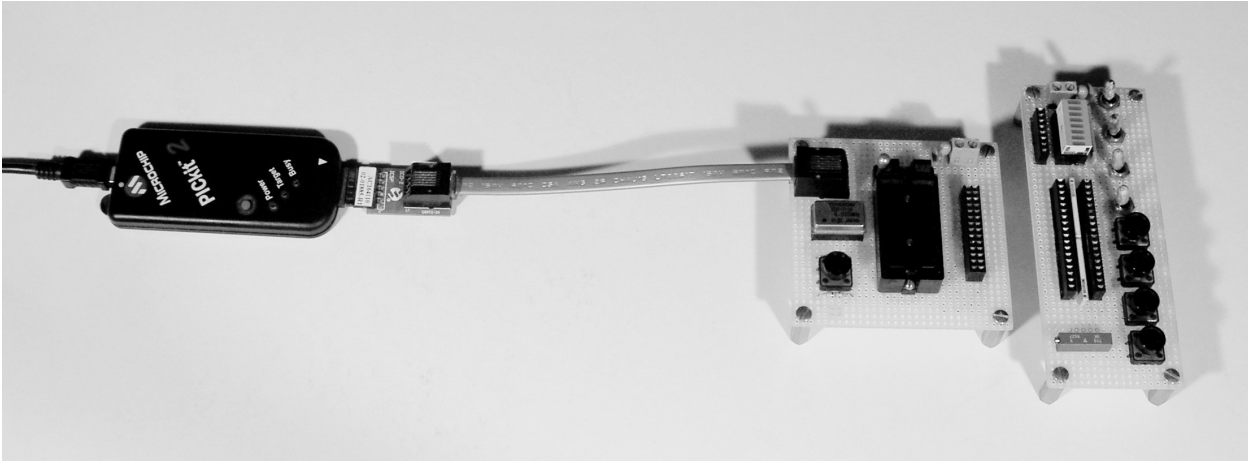
- 8-pin devices            8-pin header interface board (AC162050) required
- 14-pin devices        14-pin header interface board (AC162052) required

We will discuss only the debugger/programmer configuration using 18-pin or larger 16F devices. MPLAB is required for use along with a USB connection to host PC.



## USER BOARD = TARGET BOARD

The 87s board described earlier is set up to be used with the PICKit 2. You can easily set up your own board in a similar way by providing the modular phone jack and using the 47 K pull-up resistor on MCLR (reset).



PICKit 2, 87s Board, and 87s Companion Board

## USING THE MICROCHIP PICKIT 2 - PIC F882 AS EXAMPLE

### General Considerations

When Microchip's PICKit 2 is employed, the following F882 resources are utilized by the PICKit 2:

- The  $\overline{\text{MCLR}}/\text{Vpp}$  pin is used for programming.
- Port B bits 6 and 7 are used for PICKit 2 serial communication with the F882.
- Lose 1 stack level.
- File registers used by the PICKit 2:  
0x70, 0xF0, 0x170, 0x1B4 - 0x1BF, 0x1F0
- Program memory locations used by PICKit 2:  
0x0680 - 0x07FF

For other devices, use MPLAB to get information:

- Help>Topics.
- Under "Debuggers" in the Help Topics dialog, select "MPLAB ICD 2" (ICD 2 is correct). Click "OK".
- Under the "Contents" tab, select "MPLAB ICD 2 Overview".
- Select "Resources used by ICD 2".
- Under "Program/Data Memory Used", see "Program Memory Used" and "File Registers Used" to determine what program memory and file register resources are used by the PICKit 2 for the specific device you will be using.

For porting F84 applications to the F882 where the F882 will be programmed by the PICKit 2, the following must be done:

- Move start of general purpose file registers from 0x0C to 0x20.
- Configuration bits:
  - LVP disabled, RB3 is digital I/O.
  - DEBUG enabled (PICKit 2 mode enabled).
- Cannot use port B bits 6 and 7 (used by PICKit 2).
- Lose 1 stack level when PICKit 2 is used.
- File registers and program memory locations listed above must not be used by your program.

## First F882 Program For Use With The PICkit 2

For our first PICkit 2 project, we will modify pict1.asm from **Easy Microcontrol'n** which simply writes a bit pattern to the eight port B pins which are connected to 8 LEDs.

```
=====PICT1.ASM=====10/14/97==
    list    p=16F84a
    __config h'3ff1'
    radix   hex

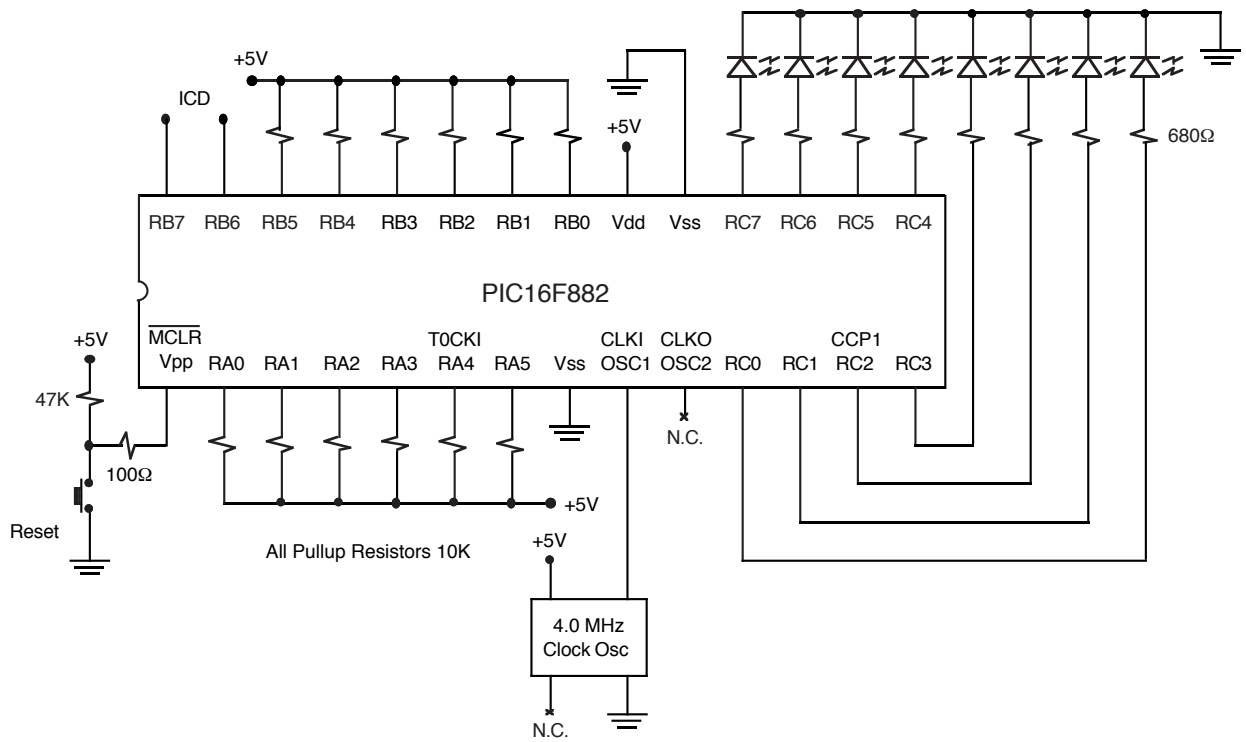
;-----
;    cpu equates (memory map)
portb    equ    0x06
;-----

    org    0x000

;
start    movlw  0x00    ;load W with 0x00
         tris   portb  ;copy W tristate, port B
;
         movlw  0x0f    ;load W with 0x0F
         movwf  portb  ;load port B with contents
;
         goto   circle ;done
;
         end

;-----
;at device program time, select:
;    code protection off
;    watchdog timer disabled (default is enabled)
;    standard crystal XT (using 4 MHz osc for test)
;    power-up timer on
=====
```

Not all of the 8 bits of port B are available for your use as RB7 and RB6 are used by the PICkit 2. We will use port C instead.



With the F882 , the A/D converters are turned off on reset by default.

Type the following example in MPLAB (or cut and paste it).

```
=====F882ICDxt.ASM=====3/24/11==
    list    p=16f882
    __config    h'2007',    h'23e1'    ;config reg 1
    __config    h'2008',    h'3fff'    ;config reg 2
    radix    hex
;-----
;    cpu equates (memory map)
status    equ    0x03
portc    equ    0x07
trisc    equ    0x87
;    bit equates
rp0    equ    5
;-----
start    org    0x000
        bsf    status,rp0    ;switch to bank 1
        movlw  b'00000000'    ;port C outputs
        movwf  trisc
        bcf    status,rp0    ;switch back to bank 0
        movlw  0x0f            ;load w with bit pattern
        movwf  portc          ;load port C with contents of W
circle   goto   circle        ;done
;
        end
;-----
;at device programming time, select:
;    code protection off
;    write protection off
;    watchdog timer disabled)
;    external clock osc XT, 4 MHz
;    RA6 clock out, n.c.
;    mclr enabled
;    power-up timer enabled
;    brown-out reset enabled @ 4.0v
;    low voltage programming disabled
;    debug disabled
;    fail safe clock monitor disabled
;    int/ext switchover disabled
=====
```

Put it in the same directory/folder as MPLAB x.xx. Call it F882icd.asm. Create a new project called F882icd.mcp.

Assemble (Project>Make Project) F882icd.asm to create the .hex file F882icd.hex.

Note the following:

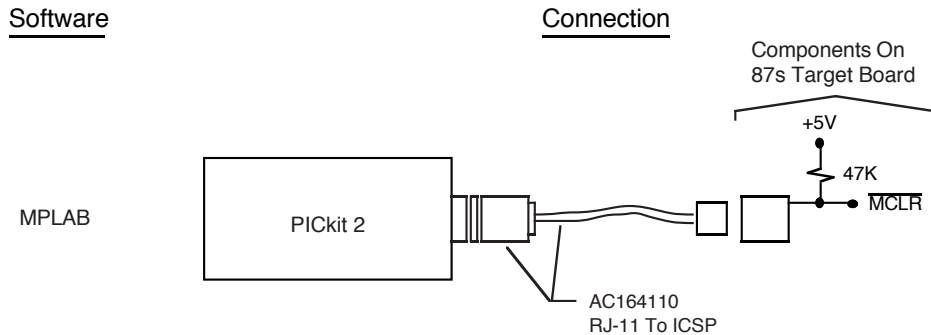
- Two `__CONFIG` assembler directives are included.
- When using the PICkit 2 (as opposed to a device programmer), brown-out reset is disabled and debug mode must be enabled.

Since bits 7 and 6 of port B are used by the PICkit 2 for communication with the host PC, we are left with 6 of the 8 port B lines for our applications. When writing your application code, it is alright to use instructions which write to all 8 bits of port B and TRISB. The PICkit 2 takes care of the house keeping so that PICkit 2 communications on bits 7 and 6 are not affected by these instructions.

## MPLAB OPERATIONS

### Setting Up The PICKit 2

Connect the PICKit 2 to your PC and power as shown.



Connect the PICKit 2 to your 87s board using a short 6-conductor modular phone cable. Connect your 87s board to a suitable +5 volt DC power supply.

When powering-up the PICKit 2 and target board the first time or for use with a new project, use the following procedure:

1. At start-up, NO power should be applied to the 87s board.
2. Power-up the host PC.
3. Connect the PICKit 2 to the host PC. The green "Power" LED in the PICKit 2 should be on.
4. Start MPLAB. At this point, it is assumed that you have a .hex file suitable to be programmed into the F882 on your 87s board.
5. Configure>Select Device.  
Select PIC16F882.
6. Select PICKit 2 as the debugger to be used.  
Debugger>Select Tool>PICKit 2.
7. The Output Window will indicate status.
8. Select Debugger>Settings.
9. The PICKit 2 Settings dialog box will open. Click on "Connect On Startup".  
Click OK.
10. With the F882 in the ZIF socket, power-up your 87s board.
11. Open the output window.  
View>Output.
12. Select Debugger>Connect.
13. Observe the activity in the output window. On completion, the last text line should read "PICKit 2 Ready".
14. You should now be able to debug and erase the F882 on your 87s board.

Minimize the output window.

Reverse the procedure to power-down (87s off, PICKit 2 off, PC off).

Open your project (F882icd.mcp in this case).

To check status, use Window>Output.

To get back to the assembly source file window, use Window>MPLABx.xx\F882icd.asm.

Two \_\_config directives included in the source code make the following selections:

- Oscillator type XT.
- MCLR enabled.
- Watchdog timer off/disable.
- Power-up timer on/enabled.
- Brown out detect off/disabled (must).
- LVP disabled (must).
- Debug enabled.
- Code protection off.
- Write protection off.

Click on Debugger>Program to program the F882. The PICkit 2 will put the debug code in the F882, program your code into the F882, and put the debug code used by the PICkit 2 in the F882.

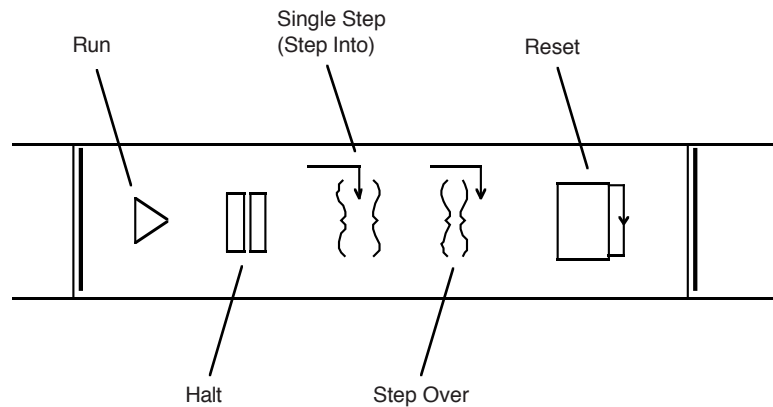
Stuff happens. Window>Output to check programming status. When programming is complete, four pairs of "Programming/Verifying messages" will appear in the MPLAB output window followed by "Debug mode entered" and "PICkit 2 Ready".

Note that each time your code is changed, it is necessary to reprogram the chip as described above. A dialog box will pop up to remind you.

Select Window>C:\xxxxx\F882icd.asm and go back to the assembly source code window.

## Toolbar

To debug programs using the PICkit 2, we will use four buttons on the debug toolbar at the top of the MPLAB window. You can discover the rest by "hacking" as the need or curiosity arises!



### To Run A Program In Real Time Via The Toolbar

Click on the run button on the toolbar. 

Hex 0x0F should now be displayed at the port C LEDs.

### To Halt The F882 Via The Toolbar

Click on the halt button on the toolbar. 

The last line of code will have a green arrow pointing to it.

### To Reset The F882 Via The Toolbar

Click on the reset processor button on the toolbar.



The line of code after "start" will have a green arrow pointing to it.

## Watch Window

To create a watch window:

View>Watch.

A Watch window called "Watch 1" appears. "Watch 1" is indicated by the highlighted button in the lower left hand corner of the screen.

Special function registers (SFR) may be selected for viewing.

Scroll up and down the SFR menu (to the right of the Add SFR button) and select the SFR registers you would like to watch as your program executes.

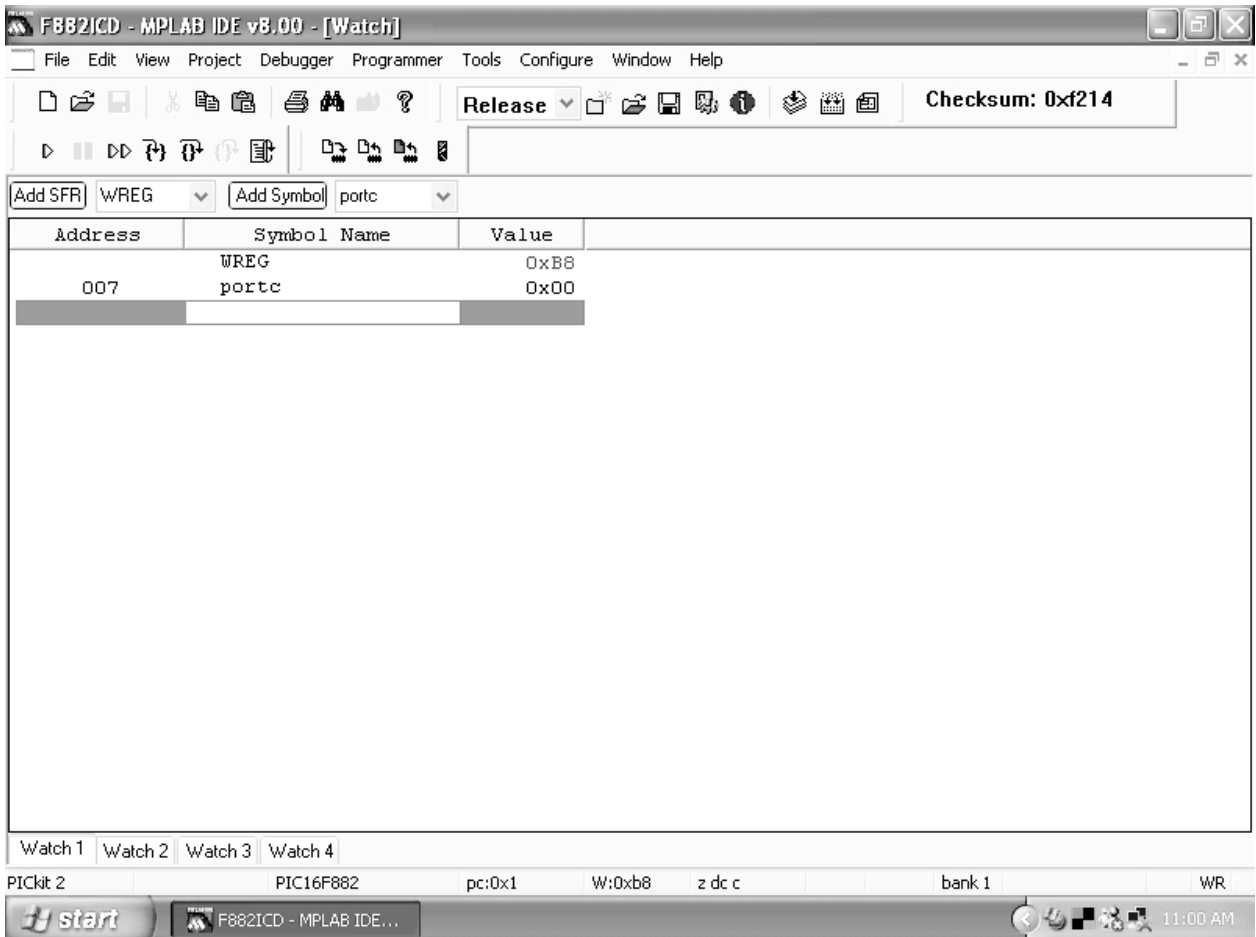
In this case, select W by highlighting "WREG". Click on the Add SFR button.

General purpose registers you are using may also be selected for viewing.

Scroll up and down the Add Symbol menu (to the right of the Add Symbol button) and select the general purpose registers you would like to watch as your program executes.

In this case, select port C by highlighting "portc". Click on the Add Symbol button.

The menus are made up from two sources. One is a standard list of special function registers for the device you are using contained in MPLAB. This list is narrowed down to the ones you are actually using in your application. The other is the general purpose registers identified in the equates in your source code.

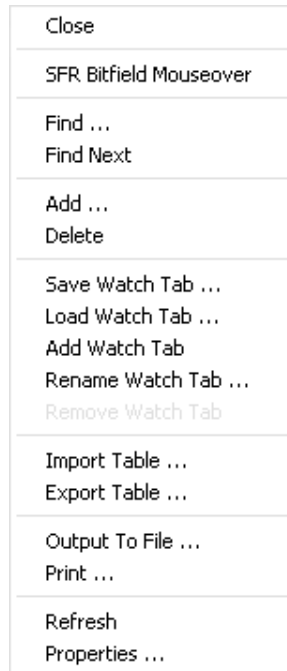


The watch window Watch 1 is now on-screen almost ready for use. I find it helpful to have the contents of a port displayed in binary. Let's customize (edit) Watch 1.

Select "portc" by clicking on it/highlighting it.

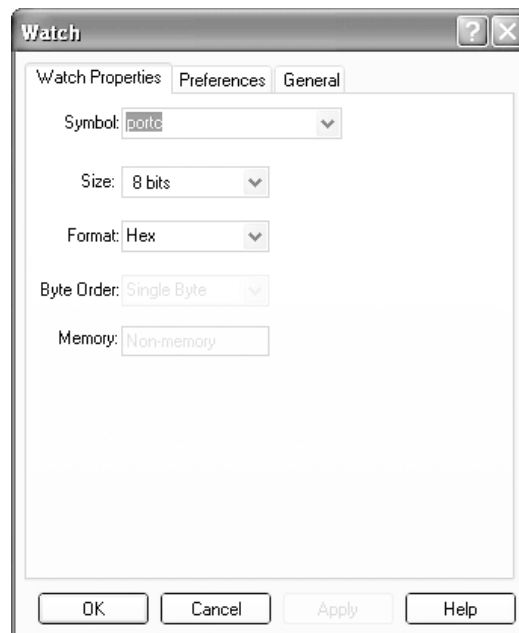
Right click on the value for Port C in Watch 1.

A drop-down list appears.



Click on Properties.

The Watch dialog box appears. The symbol "portc" is already selected under the Watch Properties tab.

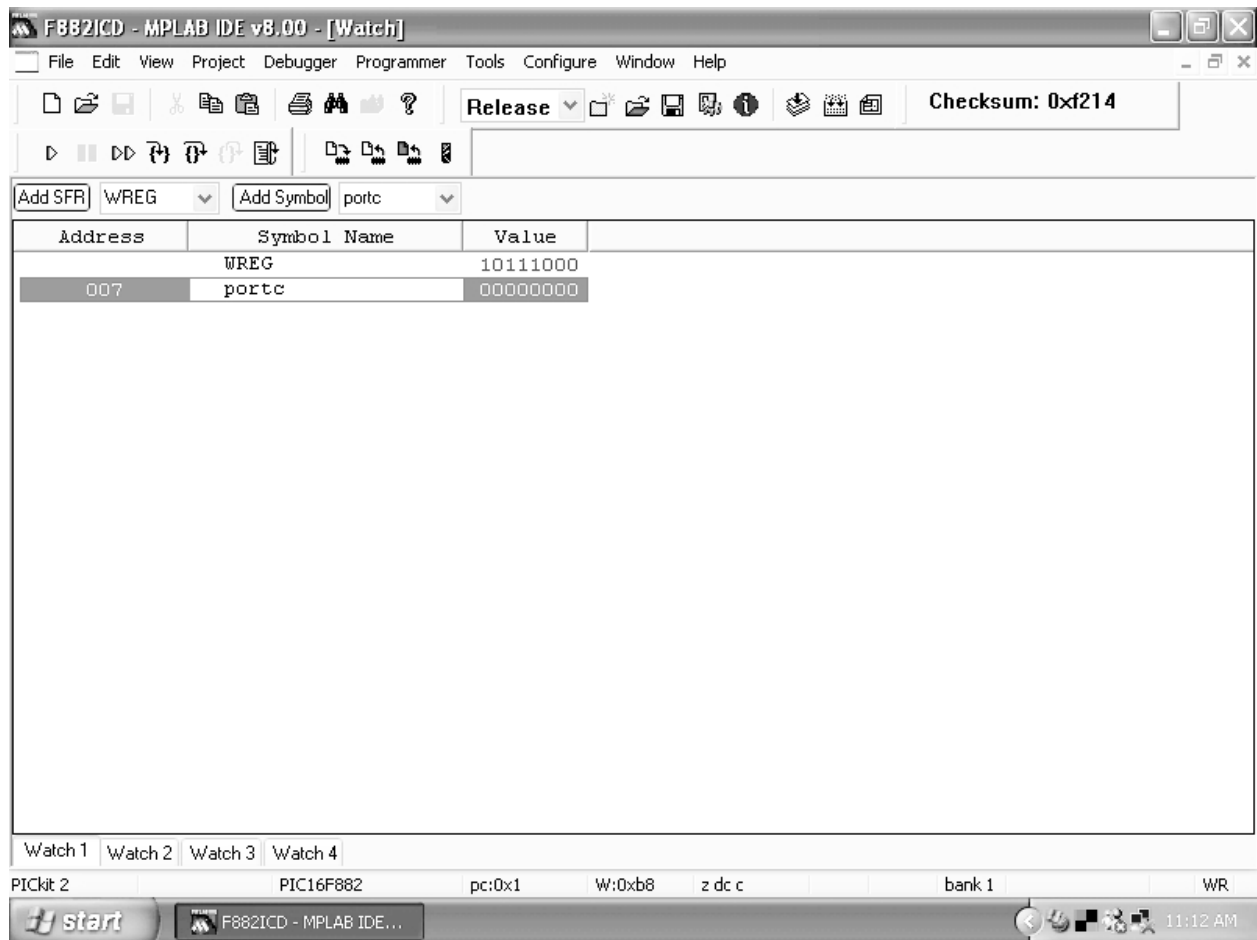


Under Format, select Binary.

Click on Apply.

Close the Watch dialog box.

The watch window now looks like this:



To delete a register from a watch window, highlight the register's symbol name in the watch window. Press the right mouse button. A drop-down list will appear. Click on Delete.

Notice that clicking on Window (menu bar) shows that "Watch" is checked.

If you close this watch window and then want to open it later, use View>Watch.

If the project is saved, the watch window will be saved as part of the project.

Watch windows may be saved as a file. Click the right mouse button in the window to open a menu. Select Save Watch to save the the currently selected watch window to a file.

To delete a watch window, the file must be deleted using the usual Windows file management procedure.

## Single Stepping

Reset the processor via the toolbar icon. Notice that the first line after "start" in your program is highlighted. The processor is stopped at the point where the highlighted line is the one about to be executed. Click on the single step (step into) button on the toolbar.



The next instruction is executed. Notice that some time is required for communication between the PC and the F882 to fetch the contents of the selected F882 registers for display. You can single step your way through F882icd.asm to the instruction labeled "circle". Observe the changes of the contents of the W register and port C in the watch window. Use Window>Cascade to observe the .asm and Watch windows simultaneously.

As you single stepped through the code the first time, the LEDs were all off until `movwf portc` was executed. Now step through the second time. Four of the LEDs will come on after `movwf trisc` is executed (sooner than the first trip through). The reason is that the contents of the port C register is 0x00 after reset and remains so until some other data is written to it. Writing to the TRISC register to make port C outputs results in the contents of the port C register being output immediately. On the first trip through the code, the contents is 0x00. On the second trip through the code, the contents is 0x0F.

If you want to play with this, add the following line of code and move the "start" label.

```
org    0x000
start  clrf  portc      ;LEDs off
      bsf   status,rp0 ;switch to bank 1
      etc.
```

Test for results.

To quit single-stepping, click on the reset button in the toolbar.

## Break Point

### *Break On Address Match*

One breakpoint may be set. The processor breaks after executing the instruction stored at that address.

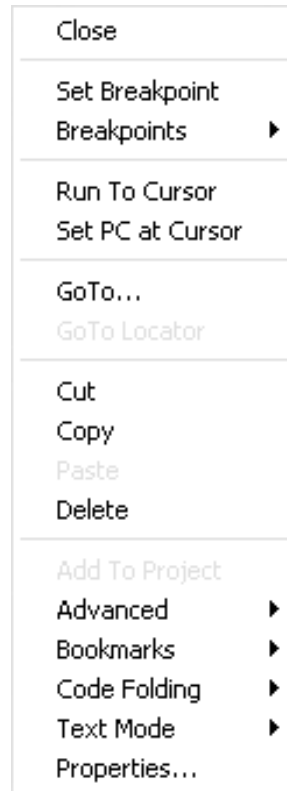
To set a breakpoint on address match:

Halt the F882 and reset it using the buttons on the toolbar.

Highlight last line of code you want executed and place the I-beam (I) cursor on it.

Click the right mouse button.

A dialog box will appear.



Select Set Break Point.

The line of code selected will now have a red "B" next to it in the left margin.

Click the toolbar run button to run to the breakpoint.

The F882 will stop at the address you selected and the next line of code will have a green arrow next to it.

### *Clear Breakpoint*

Debugger>Breakpoints.

A dialog box appears.

Select Breakpoint.

Click Remove.

Click OK.

After code execution stops at a breakpoint, you may single step forward from there.

## Powering Down

The power-down sequence is the reverse of the power-up sequence

- Turn off power to the target board.
- Power-down the PICkit 2.
- Turn off power to the host computer.

## Starting Up - Opening An Existing Project

When powering-up the PICkit 2 and target board, use the following procedure:

1. At start-up, NO power should be applied to the 87s board.
2. Power-up the PC.
3. Power the PICkit 2 . The green "Power" LED in the PICkit 2 should be on.
4. With the PIC microcontroller in the ZIF socket, power-up your 87s board.
5. Start MPLAB and open your project.
6. The output window will be open for a short time. If all is well, "PICkit 2 Ready" will be displayed just before the output window closes (on it's own).

The PICkit 2 selections will be the ones you made the last time the project was saved.

You are now ready to go to work.

## Operating 87s Board Stand Alone After Debug'n

During the debugging process, background debugging is enabled via one of the configuration bits. This causes code to be written into the program memory space used by the debugger which tells the F882 to look for the debugger on reset. When operating the microcontroller stand alone, the debugger will not be found and your program will not be executed. To get around this, you will need to:

- Debugger>Select Tool>None
- Programmer> Select Programmer>PICkit 2 (programmer mode).
- Configure>Configuration Bits. Background debug enabled is shown \*.
- Click on "Enabled" and change the selection to "Disabled" \*.
- Reprogram the F882. Programmer>Program.
- Open (select) the Output window to check programming status. Look for ... Programming Succeeded".

\* Sometimes this change is made for you when changing tools from debugger to programmer. Sometimes it is not (!).

Since port lines RB7 and RB6 will not be used by the PICkit 2 when the 87s board is operated in stand alone mode, you may want to connect them to pullup resistors.

## Reconnecting The PICkit 2 After 87s Board Stand Alone Operation

To reconnect the PICkit 2 to do more debugging or test the hot new feature you now want to add to your project, you will need to reverse the procedure outlined above:

- Programmer>Select Programmer>None
- Debugger>Select Tool>PICkit 2 (debugger mode).
- Configure>Configuration Bits. Background debug disabled is shown \*.
- Click on "Disabled" and change the selection to "Enabled" \*.
- Proceed with debugging.

\* Sometimes this change is made for you when changing tools from programmer to debugger. Sometimes it is not (!).

Since port lines RB7 and RB6 will be used by the PICkit 2, remember to remove the pullup resistors (if any).

## CONCLUSION

You will find that, when using MPLAB, the sequence or selection of dialog boxes that come up and the selections made in them are somewhat dependent on what you did last, perhaps going back to a project you did last month. Also, Microchip likes to make software changes. Be prepared to hack a little!