Application Development with MPLAB® IDE

Installation

For some Windows® Operating Systems, administrative access is required to install software on a PC. MPLAB IDE must be installed on the PC hard drive – it cannot be run on a server.

CD-ROM Method:

Place the CD into the drive. Follow the on-screen menu to install MPLAB IDE. If no on-screen menu appears, use Explorer to find and execute the CD menu by double-clicking on the executable file MP*xxx*.exe (where *xxx* represents the version) in the root directory of the CD.

Download Method:

From the Microchip web site (www.microchip.com), double-click on the downloaded executable file to begin installation.

Note: For Windows NT® systems, Microsoft recommends reinstalling the service pack after ANY software or device driver is installed.

Minimum Configuration

- Compatible Intel Pentium® class system
- Supported Windows operating system (see list below)
- 32 MB memory (128 MB recommended)
- 85 MB of hard disk space
- Internet Explorer 5.0 or greater for installation and on-line Help

Considerations for Installation

Operating systems:

- Windows® 98 SE
- Windows ME
- Windows NT® 4.0 SP6a Workstations (NOT Servers)
- Windows 2000 SP2
- Windows XP Home and Professional

Tools associated with MPLAB IDE may not support the same operating systems as MPLAB IDE. See individual tool README files for more information.

Invoking MPLAB IDE with a Command Line

MPLAB IDE can be invoked through the command line interface as follows:

```
mplab [<file>] [/<option>]
```

Open the workspace `<file>` in MPLAB IDE. Any projects contained in the workspace will be opened also. Display of the splash screen can be suppressed with the option `nosplash`:

```
mplab myproj.mcw /nosplash
```

Opens MPLAB IDE and the workspace named `myproj`. The MPLAB splash screen is not displayed.

Checklist for Getting Started with MPLAB IDE

- When using hardware emulators, in-circuit debuggers or programmers, make certain the correct drivers are installed and the correct power up sequence is followed (see *Help* Getting Started topic for the individual components.
- Make sure the proper device is selected in *Configure>Select Device*.
- Make sure that the Language toolsuite is active and points to the correct executables for the toolsuite (*Project>Select Language Toolsuite*).
- Use the Project Wizard to create a project.
- Set default search paths and directories for language tool components using the Project Wizard or *Project>Set Language Tool Locations*.
- Use Template files or a previous code file as the starting point for new code.
- Double click on errors in the Output window to correct source code errors.
- Make sure that configuration bits are set correctly for debugging (Configure>Configuration Bits). For debugging, WDToff should usually be selected. These items can be set in the source code with the `__config` directive.
- If problems are encountered, check the on-line help "Limitations" for the processor and debugger being used (in the Troubleshooting section of the on-line help for each tool), also check the README files.

Help Resources

If problems are experienced with MPLAB IDE operation, refer to the Troubleshooting section of the MPLAB IDE Help and the *MPLAB IDE User's Guide*. Visit the Microchip web site – www.microchip.com for:

- On-line support.
- Downloads of the latest development tools, data sheets, application notes, user's guides, articles and sample programs.
- Web Conference, Design Tips and Device Errata.
- Microchip Change Notification System – automatically sends change notification bulletins for silicon and development tools to subscribers.
- Development Systems Information Line and Technical Support:
  1-800-755-2345 for U.S. and most of Canada
  1-480-792-7302 for the rest of the world

Shortcut Keys

Menu Item	Toolbar Icon	Shortcut Keys	Function
Projects		Ctrl + F10	Build All
		F10	Make
Debug		F9	Run Program
		F5	Halt Program
			Animate
		F7	Step Into
		F8	Step Over
		F6	Processor Reset
		F2	Open Breakpoints Dialog
Window Control		Ctrt + F4	Close Open Window
		Ctrl + F6	Go To Next Open Window
		F1	Help For Window

Serial Communications

Tools supported: MPLAB ICD 2, PICSTART® Plus and PRO MATE® II.

- Do not use the COM port or port interrupt with another device or damage to that device could result.
- Do not use third-party communications drivers
- Disable FIFO's and change Flow Control to Hardware. Reboot the PC before trying to communicate with the tool.
- Make sure the COM port selected in SW (COM1, COM2, etc.) matches the physical connection in HW.
- Make sure the COM port baud rate selected in SW matches the physical ability of the port HW. If a baud rate of 57600 does not work, try switching to 19200.
- MPLAB ICD 2: If using target clock and/or power, make sure they are present.

USB Communications

Tools supported: MPLAB ICD 2, PICkit® 1, MPLAB ICE 4000

- Install MPLAB IDE before plugging in any USB device.
- If using target clock and/or power, make sure they are present.
- Windows NT does not support USB.

CAUTION

Use the MPLAB ICE supplied driver (MPLAB IDE subdirectory Drivers*nn*, where *nn* represents the version of Windows on the PC.) MPLAB ICD 2 will not work without the supplied driver and the correct driver cannot be installed without deleting the incorrect USB driver from the Hardware Control Panel of Windows. PICkit 1 uses the standard Windows USB driver.

Wizards, Walk-Throughs and Tutorials

Listed below are several items to explore when getting started with MPLAB IDE. Review the following features to become familiar with the program:

- MPLAB IDE Walk-Through (MPLAB IDE *Help*, Quick Start)
- Project Wizard (*Project>Project Wizard*)
- MPLAB SIM Tutorial (MPLAB SIM Help)
- MPLAB SIM30 Tutorial (MPLAB SIM30 Help)
- MPLAB ICD 2 Setup Wizard (*Debugger>MPLAB ICD 2 Setup Wizard*)
- MPLAB ICE Complex Trigger Walk-Through (MPLAB ICE Help)

Messages and Warnings (MPASM® User's Guide-page 51)

MPASM's error messages and warnings can be controlled by two methods:

1) The MPLAB "*Project>Build Options* Categories: Output" dialog, or
2) The MPASM ERRORLEVEL directive in the source code. When using mid-range PICmicro® MCUs, often the bank messages may swamp out other error messages, so the following can be used:

```
ERRORLEVEL -302
```

These are common values for controlling ERRORLEVEL:

```
    0 - all msgs & warnings
    1 - warnings & errors
    2 - errors only
-306 - no page msgs
-302 - no bank msgs
-202 - no arg range msg
```

An example of using multiple values for ERRORLEVEL:

```
ERRORLEVEL -302,-306,-202
```

Conditional Assembly (MPASM User's Guide-page 56)

Conditional assembly is used to selectively control which sections of the source code are passed on to the assembler. Only the TRUE part of the expression is assembled, and the false portion is ignored. This allows variables to control how the same source code might be compiled for debugging, testing or for final device programming. `IF` and `ELSE` cannot be used during run time, i.e., they cannot be used like the C language constructs to determine actions that happen when the program executes – `IF/ELSE/ENDIF/IFDEF/IFNDEF/ENDIF` are only recognized when the code is assembled and do not exist as conditionals in the final object code.

```
variable temp=1
  IF temp=0
     movlw 0x0A
  ELSE
     movlw 0x1E
  ENDIF

  #define test
  IFDEF test
     movlw 0x01
  IFNDEF test
     movlw 0x02
  ENDIF
```

Macros (MPASM User's Guide-page 83)

Macros allow "shorthand" code fragments. Rather than type out similar source code, a Macro can be defined that will generate the source code and fill in various arguments with variable names that are used at specific points in the code. Like conditional assembly constructs IF/ELSE Macros do not have any run-time behavior, and are evaluated and expanded before the code is assembled. Unlike the `while` in the C language, the Macro `while` is used to expand code to multiple lines as determined by its parameters *before* the code is assembled.

```
multiply macro arg1, dest_hi,
    local i = 0
    movlw arg1
    movwf mulplr
    while i < 8
      addwf dest_hi
        ; Place code to be repeated i times here
      i += 1
    endw
  endm
```

Banking (MPASM User's Guide-page 40)

Data accesses can be portable and bank independent by using the `banksel` directive. In the PIC18XXX architecture, SFRs and variables in the access area do not need the `banksel` directive. `banksel` will set all bank bits regardless of the currently selected bank and is only needed when accessing a variable that is not in the currently selected bank.

```
    banksel   temp1
    movf      temp1
```

Paging (MPASM User's Guide-page 40)

PICmicro MCUs other than the PIC18XXX family can have more than one program memory page. This convenient, portable directive can switch execution from one page to another. Code ported from these PICmicro devices to the PIC18XXX devices will ignore the `pagesel` directive. `pagesel` will set all page bits regardless of the currently selected page and is only needed when issuing a CALL or GOTO instruction whose destination is not in the currently executing page.

```
    pagesel   boot_routine
    goto      boot_routine
```

Radix (MPASM User's Guide-page 91)

When using MPASM, numbers can be interpreted in a variety of number bases. The default for the entire source file can be set using the `radix` directive:

```
    radix dec
```

Within the source, code values can be entered in number bases other than the default using the following constructs:

```
    D'123'        .123      ;  decimal
    H'1AF'        0x1F      ;  hexadecimal
    O'777'                  ;  octal
    B'00111001'             ;  binary
    0b00111001              ;  binary
    'A'           'C'       ;  7-bit ASCII
    dt'This is a string'    ;  ASCII string
```

Template Files

Template files for all Microchip microcontrollers are located in the MPLAB IDE installation directory:

`C:\Program Files\MPLAB IDE\MCHIP_Tools\Template\Code`
`C:\Program Files\MPLAB IDE\MCHIP_Tools\Template\Object`

The Object folder contains a template for files using MPASM and MPLINK®. The Code folder is for files using MPASM without MPLINK. These files are like the examples below, but are more complete with comments and examples. They can be used to start a project. A Code template file will build with MPASM without any modifications. An Object template file needs to have a linker script added to the project in order to build. The following sample code segments are similar to the template files, but simplifed.

Sample Code for Project with MPLINK

When using assembly language projects with MPLINK, the format is slightly different than when using MPASM without the linker. Instead of using `ORG` statements, sections defined in the linker script determine where the code is located. In the following code snippet, `UDATA` sets up the area for the variable `temp_count`. `EEDATA` is defined by setting the code section to 0xF00000 for the PIC18XXX devices (0x2100 for all other PICmicro MCUs) and using the `de` directive. Sections are set up for the actual program using the `CODE` directive and placing this code in the various program memory areas of the target device.

```
                list    p=16f877a
                include <p16f877a.inc>
                __CONFIG CP_OFF & ... & _LVP_ON
                CODE    0x2100 ; EEDATA
                de      1,2,3,5,8,13,21
                UDATA   0x020 ; RAM
temp_count      RES     4 ; Reserve 4 bytes of RAM for
                            32-bit variable TEMPCOUNT
RESET_VECTOR    CODE    0x0000
                goto    Start
INT_VECTOR      CODE    0x004
                   ; ... interrupt code here
                RETFIE
MAIN            CODE
Start           clrf    temp_count+3
                   ; ... main application code here
              nop
              END
```

Sample Code for Project without MPLINK

Code that is written for MPASM without MPLINK is often legacy code. Most new code should be written using the linker, since MPLAB's debugging facilities can use local variables and C language constructs. When MPASM is used without MPLINK, the path name to source files must be less than 62 characters. For PIC18XXX devices, a HIGH priority interrupt can be used to respond to an unnested interrupt. The advantage is that the HIGH priority interrupt uses a three byte stack to automatically save and restore `STATUS`, `BSR` and `WREG`.

```
 LIST        P=18F452
  #include   <P18F452.INC>
  __CONFIG   CONFIG1H,  _OSCS_OFF_1H & _HS_OSC_1H
  __CONFIG   CONFIG7H,  _EBTRB_OFF_7H
  QUEUESIZE  EQU 0x10
  CBLOCK     0x080    ; RAM
     STATUS_TEMP, WREG_TEMP,BSR_TEMP
     queue:  QUEUE_SIZE
  ENDC

  ORG        0xf00000  ; EEDATA
  DE         "Test Data",0,1,2,3,4,5

  ORG        0x0000
  goto       Main
  ORG        0x0008
  bra        HighInt
 LowInt
  ORG        0x0018
  movff      STATUS,STATUS_TEMP
  movff      WREG,WREG_TEMP
  movff      BSR,BSR_TEMP
  …
  movff      BSR_TEMP,BSR
  movff      WREG_TEMP,WREG
  movff      STATUS_TEMP,STATUS
  retfie

 HighInt:
  movlw      0xFF
     ;…
  retfie     FAST
 Main:
  clrf       REG_TEMP
  END
```

Data In Program Space (MPASM User's Guide-page 44)

Data can be placed in program memory space using these directives:

```
    DA    "abcdef"            (14-bit packed)
    DATA  12, "testing", 'N'  (12,14,16-bit)
    DB    't', 0x0f,'\n'      (8-bit)
    DT    "hello"             (RETLW 8-bit)
    DW    "diag", 0x12EB      (12,14,16-bit)
    FILL  0x1234, 0x10        (12,14,16-bit)
```

MPLAB SIM Stimulus

Using MPLAB SIM as the selected debugger (*Debugger>Select Tool*), simulated electronic signals can be applied to pins and registers. There are two types of stimulus for pins: Synchronous or Asynchronous. Synchronous stimuli are synchronized with the instruction cycles of the device being simulated, and Asynchronous stimuli are applied by the user in real time as the simulator executes.

MPLAB SIM File Stimulus (.fsti, .ssti, .rsti)

This is synchronized stimulus on I/O pins or file registers, set up from files. A File Stimulus file (.fsti) is composed of one or more Synchronous Stimulus files. A Synchronous Stimulus file (.ssti) contains information on triggers used for applying stimulus to either a pin or register. Register stimulus is specified in a Register Stimulus file (.rsti). Create these files using the *Debugger>Stimulus*, File Stimulus tab. Once a File Stimulus file has been created/edited, the stimulus is applied as the program is run under MPLAB IDE. When the designated trigger is activated, the designated event occurs.

MPLAB SIM Pin Stimulus (.psti)

This is Synchronous or Asynchronous stimulus on I/O pins. A Pin Stimulus file (.psti) contains data on whether pins are set high or low at program counter addresses. Addresses in the list will be executed sequentially. Create this file using the *Debugger>Stimulus*, Pin Stimulus tab. Once a Pin Stimulus file has been created/edited, the stimulus may be applied as follows:

- Asynchronous stimulus (Type = Asynch): Press the Fire button. The designated event occurs on the designated pin.
- Synchronous stimulus (Type = Synch): Check the enable field. A message is sent to MPLAB SIM to update the stimuli.

MPLAB IDE Saved Information

Information concerning the setup of MPLAB IDE is saved as follows:

Workspaces

A workspace contains the following information:

- Selected device, debug tool and/or programmer.
- Debug tool/programmer settings information.
- *Configure>Settings*, Program Loading tab information.
- Configuration bits settings.
- Open IDE windows and their location.
- Other IDE system settings.

This information is saved in the .mcw file.

Projects

A project contains the following information:

- The group of files needed to build an application.
- File associations to various build tools.
- Build options.

This information is saved in the .mcp file. Multiple projects can be set up in the workspace by using *Configure>Settings* Project tab and deselecting "Use one-to-one project/workspace model."

Registry

The registry file of the Windows OS saves the following information:

- Language tool names and installation locations.
- Most items on the *Configure>Settings*, Workspace tab.
- All items on the *Configure>Settings*, Projects tab.

INI Files

The initialization (.ini) file saves the following information:

- Editor settings in the mpeditor.ini file.

Variable Storage (MPASM User's Guide-page 65)

When using MPLINK, the `res` directive should be used to create variable storage space in RAM. When using multiple source files, variables are local by default. To use variables created in another source file, the variable must be declared `extern`. In the source file where they are created, the variable must be declared `global`. In source files where variables are defined use:

```
var1      res 1
var2      res 2
global    var1, var2
```

Place this declaration in source files that need to use these global variables:

```
extern    var1, var2
```

Watchdog Timer (WDT) Considerations

The Watchdog Timer can interfere with debugging, and some tools may need to have the WDT turned off while developing code. Here are some things to watch out for:

- Make sure that configuration bits are set correctly for debugging (*Configure>Configuration Bits*). For debugging, WDToff should usually be selected. These items can be set in the source code with the `__config` directive. If the code is rebuilt while debugging, the `__config` directives will overwrite manual changes to the configuration bits settings.
- If the code is resetting unexpectedly with the simulator or ICE, make sure the WDT configuration bit has not inadvertently been left in its default ON state.
- MPLAB ICD 2 will NOT work with the WDT enabled. It must be off while debugging. When programming, make sure the config bit is set correctly for the application.
- When finished debugging, if MPLAB ICD 2 and SLEEP are being used and the WDT is expected to wake up the application, make sure that WDTon is set when the part is being programmed.

Writing to EEPROM Data Memory

Check the data sheet for individual devices to ensure that the device has EEPROM data memory and the algorithm is the same. Most PIC18XXX devices use this sequence:

```
     movlw data_ee_addr  ; eedata address
     movwf EEADR
     movlw data_ee_data  ; data to be written
     movwf EEDATA
     bcf   EECON1, EEPGD ; point to eedata
     bsf   EECON1, WREN
     bcf   INTCON, GIE   ; disable interrupts

     movlw 0x55          ; start write sequence
     movwf EECON2
     movlw 0xAA
     movwf EECON2

     bsf   EECON1,WR     ; enable interrupts
     bsf   INTCON, GIE
     sleep               ; wait for write complete
     bcf   EECON1,WREN   ; disable eedata writes
```

Reading EEPROM Data Memory

This code sequence can be used to read data in the PIC18XXX device EEPROM data area:

```
     movlw data_ee_addr  ; eedata address
     movwf EEADR
     bcf   EECON1,EEPGD  ; point to eedata
     bsf   EECON1,RD     ; eedata read
     movf  EEDATA,W      ; move data to W reg
```

Crossing Page Boundary Detection

When using PIC16XXX parts, care must be taken if data tables in program memory cross 256 word page boundaries. The following code demonstrates a technique to detect this so the starting address of the table can be readjusted to avoid a page boundary.

```
     ORG    0x10       ; Page 0
     MOVF   offset,W   ; w reg = offset
     CALL   Table
     . . .
     ORG    0x20       ; Page 0
Table
     ADDWF  PCL,F      ; Compute offset
     DT     "ABCD"     ; RETLW Expansion
TableEnd               ; Page 0

  IF((Table&&0xFF00) != (TableEnd-1&&0xFF00))
     ERROR "Table crosses page boundary"
  ENDIF
```

Linker Sections (MPASM User's Guide-page 73)

Linker sections are defined in the linker script (*.LKR) for the project. These files describe the memory sections for the target device and allow the application to have control over where data and code is placed. In the application, section names are then used to switch between allocating variables and positioning code in program memory. Addresses can be specific or left for the linker to resolve.

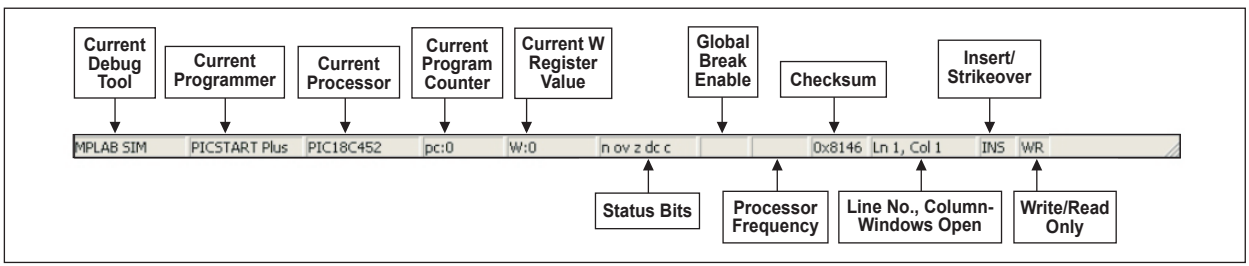
```
 main         code        0x0100
 data         code_pack   0x1200
 my_strings   idata       0x0280
 my_ram       udata
 my_access    udata_acs   0x40
 my_ov        udata_ovr
my_shared         udata_shr         0xE0
```

Status Bar

Title	Description	Possible Entries, Notes
Current Debug Tool	Set by *Debugger>Select Tool*	MPLAB SIM, MPLAB ICD 2 or MPLAB ICE
Current Programmer	Set by *Programmer>Select Programmer*	PRO MATE II, PICSTART Plus, MPLAB ICD 2, PICkit 1
Current Processor	Set by *Configure>Select Device*	PIC16F877A, PIC16F54, PIC17C756, PIC18C452, PIC18C8620, PIC18F4320, etc.
Current Program Counter	Set by running or single stepping target device.	Usually 0 at reset, otherwise, the current program counter address. Double click on this entry to bring up the Change Program Counter dialog.
Current W Register Value	Set by code running in target debugger.	Can be any value from 0 to 0xFF.
Status Bits	Upper Case = Set (1) Lower Case = Reset (0)	n OV c dc Z (example) Set as condition flags when code is running. ov = overflow, z = zero, dc = decimal carry, c = carry
Global Break Enable	Set by *Debugger>Settings* Break Options EN = All breakpoints are enabled. DIS = All current breakpoints are disabled.	EN/DIS. Allows turning all current breakpoints on/off. Current breakpoints are retained, but can be disabled/enabled with this switch.
Processor Frequency	Set by *Debugger>Settings* Clock or measured automatically by ICE.	20 MHz (example) Shows simulated or emulated target device core frequency (not crystal frequency).
Checksum	Set by code in program memory and configuration bits.	0xF125 (example) Shows current 16-bit checksum over entire program memory, EEdata (when available) and configuration bits area.
Line No., Column-Windows Open	Displays current line number and column in file.	Ln 23, Col 14 (example) Only displayed when an editor window is open and has focus.
Insert/ Strikeover	Toggles typing mode. INS = Insert characters OVR = Type over characters	INS/OVR
Write/Read Only	Displays file write/read status. WR = Editable file RO = Read only file	WR/RO. Only WR files can be edited.

The status bar appears at the bottom of the MPLAB IDE desktop and provides up-to-date information on the status of the MPLAB IDE session. When an application is running, it displays "Running" and a progress bar. When an application is not running, the information provided includes the listed information.

MPLAB IDE Status Bar



Tips & Tricks: Speed

- When using MPLAB ICE or MPLAB ICD 2, windows that are open will be refreshed at each breakpoint. To speed things up when single-stepping, close all unnecessary windows. Breakpoints will be faster if variables of interest are added to a Watch window rather than viewed in the File Register or SFR Window.
- When using the emulator or MPLAB ICD 2 with devices that have large program memories, select *Configure>Settings* Program Loading tab and uncheck "Clear Memory after successfully building a project". Speed will be increased because potentially large sections of memory won't be erased if build errors are encountered.
- When viewing the file registers right click and de-select "Full Memory Update".
  - only the displayed registers will be updated, even when scrolling.
  - changed registers will not show in red.

Tips & Tricks: Editor and Files

- Add header files which are included in the source files to the project so they will also be included in the "Find in Project" search.
- When using "Find in Files" items found can be double clicked to open the source file at that line.
- Advanced editor features like "Comment Block" and "Match Braces" are available on the right mouse button menu.
- A current file can be added to the projects as long as it has been saved at least once.
- If only one MPASM source file is implemented and the linker is not used, the total directory path and file names cannot exceed 62 characters. Use multiple source files or enable the linker to remove this restriction.

Tips & Tricks: Shortcuts

- Highlight variable names in the source code and drag them to the Watch window.
- Placing the cursor over a variable name or special function register in the source file to show the current value of that variable or special function register.
- Click on the PC in the status bar to bring up the "Change Program Counter" dialog.
- In many windows, data can be changed by selecting the field and typing in new values. File registers, special function registers and program memory instructions can be changed by selecting the current value or instruction and typing in a new one.

Tips & Tricks: Warnings and Errors

- Double click on an error message in the output window to open the source file at the line with an error.
- MPLAB ICD 2 warnings can be individually suppressed in the *Debugger>Settings* Warnings tab.
- If problems are encountered, read *Debugger>Settings* Limitations to ensure that the function desired is not something that has known restrictions.
- Double click on MPLAB ICD 2 warnings or error messages in the output window to bring up more information on that particular warning or error.

Tips & Tricks: Other Cool Things

- Special Function Registers can be sorted by address, name or value by double clicking on the column heading.
- Drag and drop entire columns in the SFR window and Watch window to re-order data. Right click on column headings to hide/show columns.
- When working with PIC16C9XX parts, select *View>LCD Pixel* to bring up an LCD pixel display that can be used for simulating the LCD.
- When using multiple projects in a workspace, go to the *Configure>Settings* Program Loading tab to uncheck "Clear program memory upon loading a program," especially when working on programs that need to be built and loaded separately into memory. If only one project is being loaded into memory at a time, it is recommended to clear memory (leave box checked).
- Use MPLINK to build projects rather than just MPASM. This provides better debugging, and has no restrictions on such things as path length.

Tips & Tricks: Editor Right Click Menu

In the editor window, click on the right mouse button to bring up a host of shortcuts:

Set Breakpoint
Breakpoints...
  Disable All
  Enable All
  Remove All Breakpoints
Run to Cursor
Set PC at Cursor

Undo Ctrl+Z
Redo Ctrl+Y

Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Delete Del

Select All Ctrl+A
Bookmarks...
  Toggle Bookmark
  Next Bookmark
  Previous Bookmark
  Clear All Bookmarks

Find Ctrl+F
Find Next F3
Replace Ctrl+H
Go To Ctrl+G

Add to Project
Advanced
  Text Mode...
    <select File type for context highlighting>
    Match Brace
    Comment Block
    Uncomment Block
Properties...
  Editor Preferences
    Line numbers on/off
    Auto indent on/off
    Word wrap on/off
    Mouseover variable highlighting
  Editor Font/colors
  Editor Tabs

Shortcut Keys for Programming

Menu Item	Toolbar Icon	Function
Programmer		Blank Check
		Read
		Program
		Verify
		Erase FLASH Device

Microchip Components Available for MPLAB IDE

Simulators	HW Debuggers	Programmers	Language Tools
MPLAB® SIM*	MPLAB ICE 2000	PICSTART® Plus	MPLAB C17
MPLAB SIM30*	MPLAB ICE 4000	PRO MATE® II	MPLAB C18
	MPLAB ICD 2		MPLAB C30
		PICkit™ 1	MPASM®*
		MPLAB PM3†	MPLINK®*

*Included free with MPLAB IDE software.
†Estimated availability: Q4 2003

Americas
Atlanta (770) 640-0034
Boston (978) 692-3848
Chicago (630) 285-0071
Dallas (972) 818-7423
Detroit (248) 538-2250
Kokomo (765) 864-8360
Los Angeles (949) 263-1888
Phoenix (480) 792-7966
San Jose (408) 436-7950
Toronto (905) 673-0699

Asia/Pacific
Australia 61-2-9868-6733
China-Beijing 86-10-85282100
China-Chengdu 86-28-86766200
China-Fuzhou 86-591-7503506
China-Hong Kong SAR 852-2401-1200
China-Qingdao 86-532-5027355
China-Shanghai 86-21-6275-5700
China-Shenzhen 86-755-82901380
China-Shunde 86-765-8395507
India 91-80-2290061
Japan 81-45-471- 6166
Korea 82-2-554-7200
Singapore 65-6334-8870
Taiwan 886-2-2717-7175
Taiwan-Kaohsiung 886-7-536-4818

Europe
Austria 43-7242-2244-399
Denmark 45-4420-9895
France 33-1-69-53-63-20
Germany 49-89-627-144-0
Italy 39-0331-742611
Netherlands 31-416-690399
United Kingdom 44-118-921-5869

As of 7/28/03

MICROCHIP
InControl

Microchip Technology Inc. • 2355 West Chandler Blvd.
Chandler, AZ 85224-6199 • 480-792-7200
www.microchip.com

DS51410A