



---

---

## MPLAB Analysis Tool Suite User's Guide

---

---

---

### Notice to Development Tools Customers

---



**Important:**

All documentation becomes dated, and Development Tools manuals are no exception. Our tools and documentation are constantly evolving to meet customer needs, so some actual dialogs and/or tool descriptions may differ from those in this document. Please refer to our website ([www.microchip.com/](http://www.microchip.com/)) to obtain the latest version of the PDF document.

Documents are identified with a DS number located on the bottom of each page. The DS format is DS<DocumentNumber><Version>, where <DocumentNumber> is an 8-digit number and <Version> is an uppercase letter.

**For the most up-to-date information**, find help for your tool at [onlinedocs.microchip.com/](http://onlinedocs.microchip.com/).

---



---

---

# Table of Contents

---

Notice to Development Tools Customers.....	1
1. Introduction.....	3
1.1. Licenses.....	3
2. MPLAB Code Coverage.....	4
2.1. MPLAB Code Coverage Overview.....	4
2.2. Using MPLAB Code Coverage in a Safety Environment .....	6
2.3. MPLAB Code Coverage Details.....	7
2.4. Get the Software.....	7
2.5. Enable/Disable Code Coverage.....	7
2.6. View Code Coverage Output.....	8
2.7. Understand Code Coverage Output.....	10
2.8. Create a Code Coverage HTML Report .....	19
2.9. Command Line Support for Code Coverage in MDB.....	21
3. MISRA Check.....	23
3.1. MISRA Check Overview.....	23
3.2. Using MISRA Check in a Safety Environment .....	23
3.3. Perform MISRA Check.....	24
3.4. MISRA Check Options Tab.....	25
3.5. Command Line Support for MISRA Check.....	26
4. Revision History.....	28
4.1. Revision A (December 2021).....	28
The Microchip Website.....	29
Product Change Notification Service.....	29
Customer Support.....	29
Product Identification System.....	30
Microchip Devices Code Protection Feature.....	30
Legal Notice.....	30
Trademarks.....	31
Quality Management System.....	32
Worldwide Sales and Service.....	33

## 1. Introduction

MPLAB Analysis Tool Suite is a collection of analysis tools that integrates with MPLAB X IDE and supports all Microchip MCU, MPU, CEC and DSC devices. It features the MPLAB® Code Coverage and the MISRA® (Motor Industry Software Reliability Association) Check in the IDE.

The MPLAB Code Coverage feature provides visibility as to what portions of your code are being executed. The MISRA Check in the IDE provides guidelines to ensure safe, secure, portable and reliable C code in embedded systems.

### 1.1 Licenses

A license for the MPLAB Analysis Tool Suite can be purchased through the Microchip Direct web site. Licenses for the MPLAB Analysis Tool Suite are as follows:

- MPLAB Analysis Tool Workstation License  
Part Number: SW006027-2 [www.microchip.com/en-us/development-tool/SW006027-2](http://www.microchip.com/en-us/development-tool/SW006027-2)
- MPLAB Analysis Tool High Priority Access (HPA) Workstation License  
Part Number: SW006027-2H [www.microchip.com/en-us/development-tool/SW006027-2H](http://www.microchip.com/en-us/development-tool/SW006027-2H)
- MPLAB Analysis Tool Network Server License  
Part Number: SW006027-2N [www.microchip.com/en-us/development-tool/SW006027-2N](http://www.microchip.com/en-us/development-tool/SW006027-2N)
- MPLAB Analysis Tool HPA Network Server License  
Part Number: SW006027-2NH [www.microchip.com/en-us/development-tool/SW006027-2NH](http://www.microchip.com/en-us/development-tool/SW006027-2NH)

Licenses can be purchased on Microchip Direct.

## 2. MPLAB Code Coverage

### 2.1 MPLAB Code Coverage Overview

The MPLAB® Code Coverage feature in the MPLAB Analysis Tool Suite provides visibility as to what portions of your code are being executed. Run your test cases to completion for a visual display of coverage. See [AoU-09-COV].

The MPLAB Code Coverage feature requires the following tools:

- An MPLAB XC C compiler, either Free or PRO, that supports code coverage output, starting with MPLAB XC8 v2.35, MPLAB XC16 v2.0, and MPLAB XC32 v4.0. See [AoU-02-COV].
- The MPLAB Analysis Tool Suite license, which includes Code Coverage, (see [1.1. Licenses](#)) that provides visibility as to what portions of your code are being executed. See [AoU-05-COV].
- MPLAB X IDE v6.0 or later, which supports the display of code coverage data from an MPLAB XC C compiler with the MPLAB Analysis Tool Suite license. See [AoU-04-COV].

**Note:** This manual is written for MPLAB X IDE v6.0 or later.

**Note:** Use MPLAB X IDE for the best code coverage experience.

Code coverage is displayed in the MPLAB X IDE as:

- Editor text highlighted by colors representing coverage: green = executed, yellow = partially executed, and red = not executed.
- Program memory highlighted by colors representing coverage.
- A Code Coverage tab with a report displaying color percentages of code covered. This information may be written to an HTML Report for later viewing.

Figure 2-1. Code Coverage - Editor Window and Code Coverage Tab

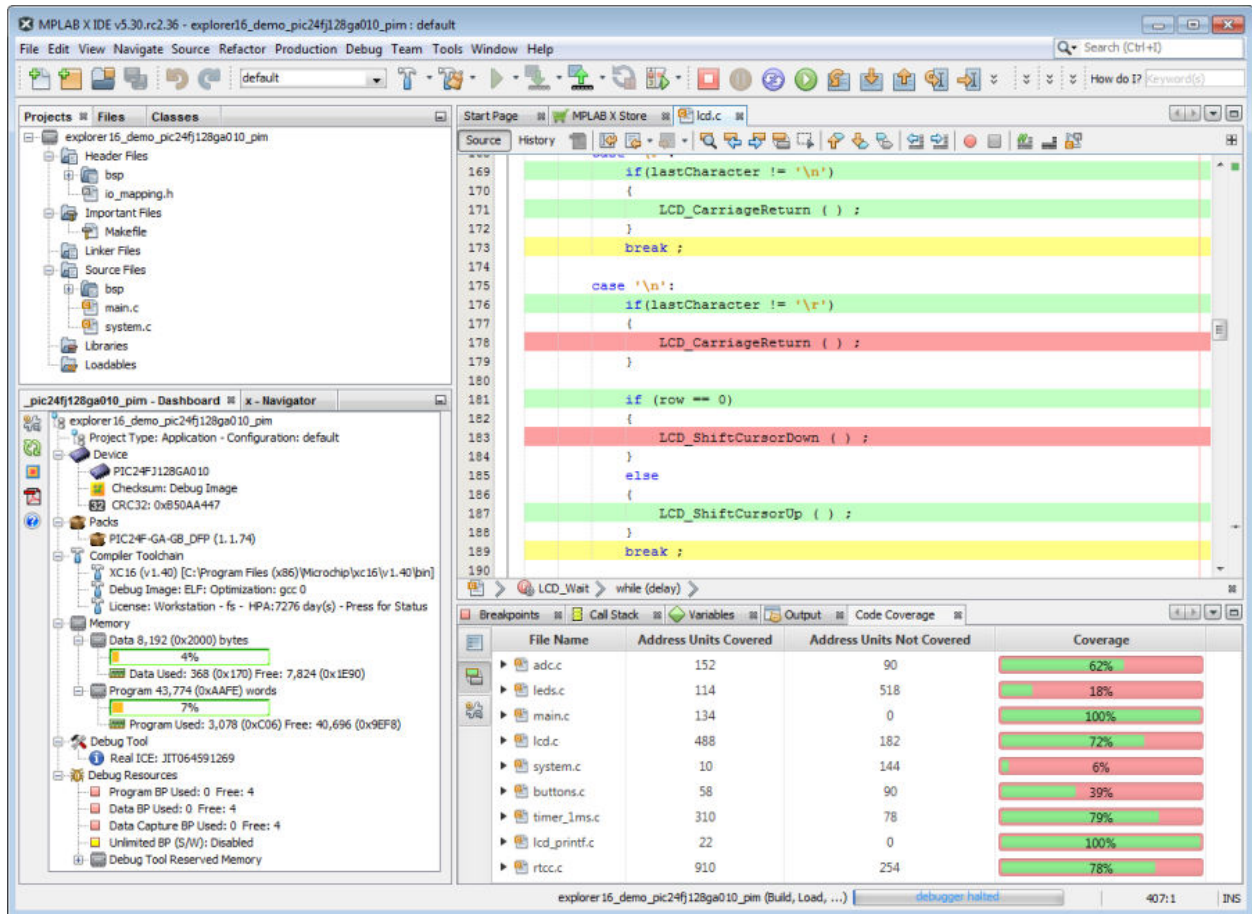
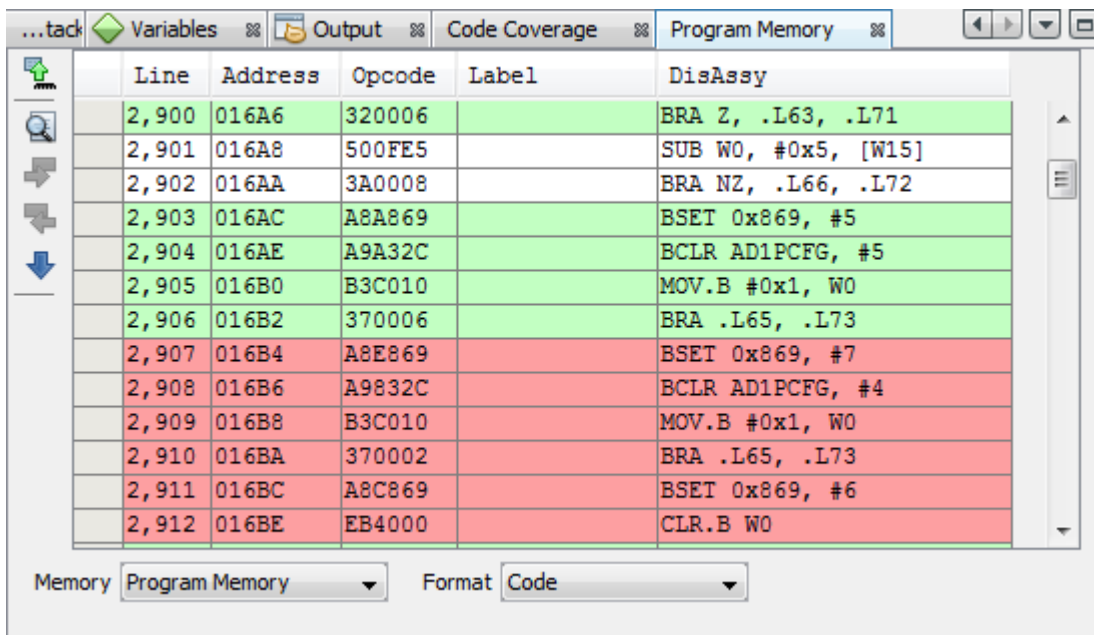


Figure 2-2. Code Coverage - Program Memory Window



## 2.2 Using MPLAB Code Coverage in a Safety Environment



**Attention:** To use the MPLAB Analysis Tool Suite in a Safety Environment, you must use a functional safety version of the Microchip compiler. The newer function safety versions of the compilers that support the MPLAB Analysis Tool Suite may not be available yet. Check the [www.microchip.com/en-us/solutions/functional-safety/mplab-development-ecosystem-for-functional-safety](http://www.microchip.com/en-us/solutions/functional-safety/mplab-development-ecosystem-for-functional-safety) for the appropriate functional safety compiler licenses:

- MPLAB XC8 Functional Safety License (with MPLAB Analysis Tool Suite support)
- MPLAB XC16 Functional Safety License (with MPLAB Analysis Tool Suite support)
- MPLAB XC32 Functional Safety License (with MPLAB Analysis Tool Suite support)

MPLAB Code Coverage has been designed according to the Microchip Development Tools standard process. Microchip asserts certain Assumptions of Use (AoUs) to meet technical and functional safety requirements at the system level. The AoUs are described below. References to some of these AoUs can be found in other sections in this manual, where they are addressed as part of a specific topic being discussed.

It is the responsibility of the System Integrator to address all the AoUs listed in this manual and to ensure that the AoUs listed have been observed by every component or item, contributed to the application being analyzed by MPLAB Code Coverage.

The System Integrator has two options:

- Make sure each assumption is fulfilled
- Disregard the assumption

In both cases, the System Integrator shall provide evidence of the fulfillment and/or a detailed explanation of why disregarding the assumption will not violate a safety requirement, or how the assumption has been sufficiently addressed in a different manner.

As indicated by the classification documents provided by Microchip along with the functional safety version of the compiler, the MPLAB Code Coverage tool is classified TCL 1. This classification is based on the assumptions of use and use cases provided in the FMEA.

**Table 2-1. Degree of Obligation**

Shall	Binding
Should	Recommendation

Assumption of Use	Description
[AoU-01-COV]	The tool user shall provide a corroborating data source or methodology.
[AoU-02-COV]	The MPLAB Code Coverage tool shall only be used in conjunction with supported Functional Safety versions of the MPLAB XC compilers.
[AoU-03-COV]	Code Coverage shall be used in conjunction with a bona fide Functional Safety process.
[AoU-04-COV]	Code Coverage shall be used with a suitable version of the MPLAB X IDE.
[AoU-05-COV]	Users of Code Coverage shall possess a valid license.
[AoU-06-COV]	Code Coverage shall only be used during development and debugging, and shall not be used in the application while in operation by the end user.
[AoU-07-COV]	Code Coverage shall not be used outside of a safe and controlled testing environment.
[AoU-08-COV]	Code Coverage shall only be used on supported Microchip microcontrollers and DSCs.
[AoU-09-COV]	Tests shall be run to completion and the application in a quiescent state before examining and analyzing code coverage data.

## 2.3 MPLAB Code Coverage Details

Use the MPLAB Code Coverage license with MPLAB XC C compiler and MPLAB X IDE versions that support code coverage to view code coverage output. See [AoU-02-COV], [AoU-04-COV] and [AoU-05-COV].

MPLAB Code Coverage works with anything built from C source code. C source code which is not colored is non-existent in the executable image. It is not to be counted as covered or not covered. For example, the common "while(TRUE)" statement will never generate executable instructions.

MPLAB Code Coverage is supported on any debug tool supported by MPLAB X IDE.

### Compiler Operation

Code coverage is supported by MPLAB XC C compiler instrumentation. The compiler adds a minimal amount of code in program memory to update flags in RAM to indicate coverage. To remove this code from the application when development ends, disable code coverage per 2.5. [Enable/Disable Code Coverage](#) and run the application. This ensures the code is removed from the product that is operating in the field. See [AoU-04-COV].

**Note:** Code coverage is ELF only.

### MPLAB X IDE Operation

The IDE will display highlighted covered code and percentages of covered code for project files. You may select which files to include for coverage by setting up file properties. See 2.7.2.3. [Code Coverage by Project File](#).

A report of the coverage may be generated also. See 2.8. [Create a Code Coverage HTML Report](#).

## 2.4 Get the Software

To use code coverage, you will need to acquire the following tools.

### MPLAB X IDE

MPLAB X IDE support for viewing code coverage output began with version v5.25, with additional features added in v5.30. However, in MPLAB X IDE v6.0, code coverage is bundled with MISRA Check in the MPLAB Analysis Tool Suite license. See [AoU-04-COV, AoU-05-COV].

The IDE may be downloaded for free at [www.microchip.com/mplab/mplab-x-ide](http://www.microchip.com/mplab/mplab-x-ide).

### MPLAB XC C Compilers

The MPLAB Analysis Tool Suite license may be used with any MPLAB XC C compiler (Free and PRO) that support code coverage. See [AoU-02-COV].

Support begins with the following versions:

- MPLAB XC8 v2.35
- MPLAB XC16 v2.0
- MPLAB XC32 v4.0

MPLAB XC C compilers may be downloaded at [www.microchip.com/mplab/compilers](http://www.microchip.com/mplab/compilers).

### MPLAB Analysis Tool Suite License

An MPLAB Analysis Tool Suite license may be purchased and activated like other compiler licenses. For more information, see "*Installing and Licensing MPLAB<sup>®</sup> XC C Compilers*" (DS50002059) for details. See [AoU-05-COV]. For license part numbers see 1.1. [Licenses](#).

A license will work for all MPLAB XC C compilers on that computer. This license may be used with Free and PRO compilers.

## 2.5 Enable/Disable Code Coverage

By default, MPLAB Code Coverage is disabled. To enable code coverage, complete the following steps:

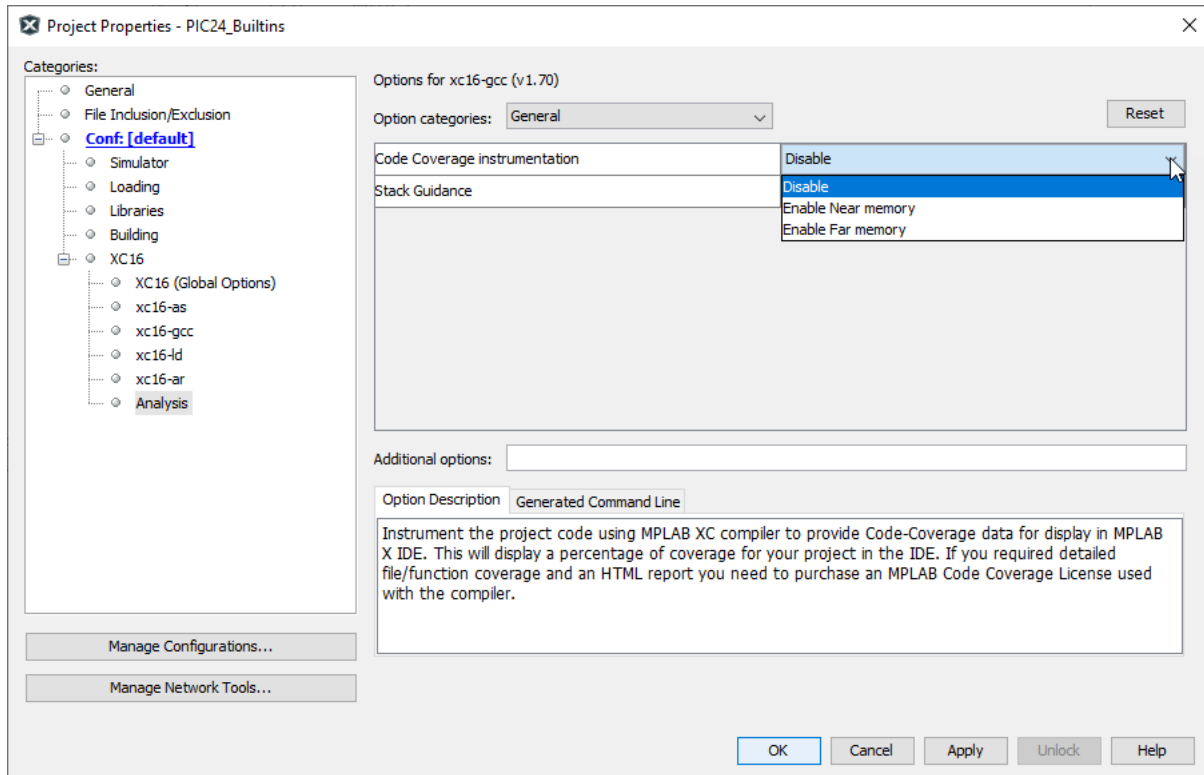
To open the Project Properties dialog, select *Tools>Analysis>Code Coverage* or you can right click on the name of your project in the Projects window and select "Properties."

1. Click on "Analysis" under Categories.
2. Under the Option categories (General) select your choice for "Code Coverage Instrumentation." Selections differ for each MPLAB XC compiler (see the table below).

**Table 2-2. Enable Code Coverage Options by Compiler**

MPLAB XC C Compiler	Enable Options	Description
XC8	Disable	Disable code coverage.
	Enable	Enable code coverage.
XC16	Disable	Disable code coverage.
	Enable Near memory	Use Near RAM space for code coverage instrumentation ( <b>recommended</b> ). If there is a link error, then select Far.
	Enable Far memory	Use Far RAM space for code coverage instrumentation.
XC32	Disable	Disable code coverage.
	Enable	Enable code coverage.

**Figure 2-3. Code Coverage Options - MPLAB XC16 Example**



## 2.6 View Code Coverage Output

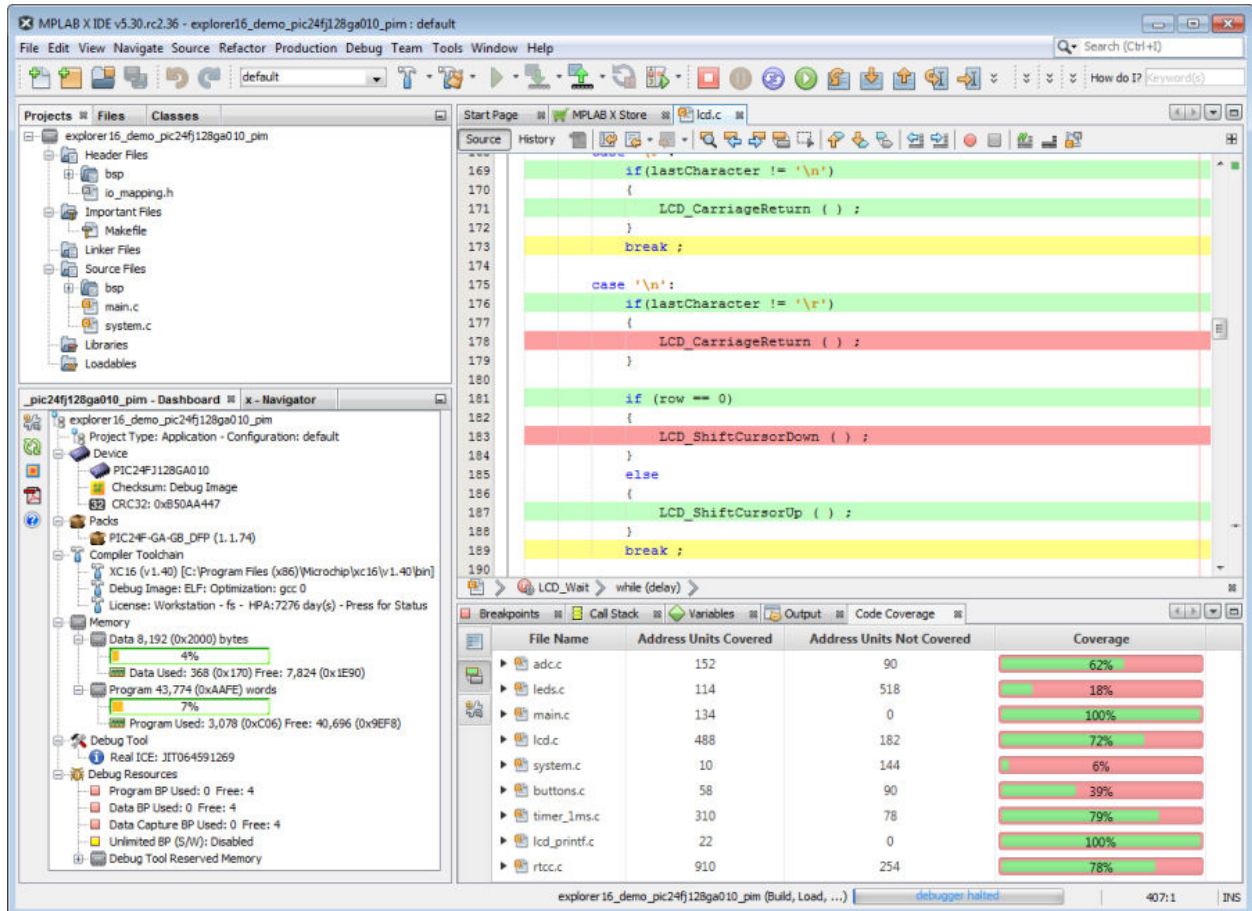
After you have enabled code coverage, as described in [2.5. Enable/Disable Code Coverage](#), debug your code and execute all test cases to completion. Then halt execution automatically or manually and step as needed till all code execution is complete.



To view code coverage:

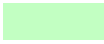


- Right click on project name and select “Show Code Coverage Summary” or select *Window>Debugging>Code Coverage*.
- To see highlight in Editor window you may need to click in it. Covered code will be highlighted in the window.
- To see highlight in Program (or Execution) Memory window, open it under *Windows>Target Memory Views*. Covered instructions will be highlighted in the window.
- A summary report will be shown in the Code Coverage tab.

**Figure 2-4. Code Coverage in MPLAB X IDE**



## 2.6.1 Highlight Colors

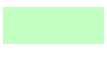

The meaning of highlight colors in the Editor window and Program or Execution Memory window are described in the table below.

Highlight Color	Highlight Name	Meaning
	Green	Covered and Executed
	Yellow	Covered and Partially Executed (Editor window only)
	Red	Covered but not Executed
	No Color	No coverage information generated <sup>1</sup>

.....continued		
Highlight Color	Highlight Name	Meaning
<p><b>Note 1:</b> Reasons why no coverage information would be generated include:</p> <ul style="list-style-type: none"> <li>• Some C constructions may not generate code for a C source line</li> <li>• Optimizations may cause code not to be generated for a C source line</li> </ul> <p>There may be other circumstances that C source lines do not produce executable code. See the examples in <a href="#">2.7. Understand Code Coverage Output</a>.</p>		

### 2.6.2 Coverage Colors




On the Code Coverage tab, the colors on the bar chart under “Coverage” have the following meanings:

Coverage Color	Coverage Name	Meaning
	Green	Total code covered and executed by test suite(s).
	Red	Total code covered by test suite(s).

For more on this tab, see [2.7.1.2. Code Coverage Window](#).

### 2.6.3 Code Coverage Tab Buttons

Click on buttons in the gutter of this tab for the following functions.

	Generate HTML Report. This report will show the same data presented in the Code Coverage window.
	Toggle (enable/disable) color highlighting in editor window.
	Open Project Properties to enable/disable code coverage.

## 2.7 Understand Code Coverage Output

The following sections detail what the different IDE displays are showing you about code coverage.

## 2.7.1 Demo Code Example

Figure 2-5. Editor Window - Demo Code Example

```

169     if(lastCharacter != '\n')
170     {
171         LCD_CarriageReturn ( ) ;
172     }
173     break ;
174
175     case '\n':
176     if(lastCharacter != '\r')
177     {
178         LCD_CarriageReturn ( ) ;
179     }
180
181     if (row == 0)
182     {
183         LCD_ShiftCursorDown ( ) ;
184     }
185     else
186     {
187         LCD_ShiftCursorUp ( ) ;
188     }
189     break ;
190

```

The Explorer 16/32 Board Demo Code for a PIC24FJ128GA010 PIM was used and comes from:  
[www.microchip.com/DM240001-2](http://www.microchip.com/DM240001-2)

The Explorer 16/32 Board Demo Code for a PIC24FJ128GA010 PIM was used and comes from:

[www.microchip.com/DM240001-2](http://www.microchip.com/DM240001-2)

### 2.7.1.1 Demo Code and Coverage Colors

For the `case '\n'` statement shown:

- Both `if` statement conditions are executed and so shown as green (covered and executed).
- Both `if` statements have evaluated as false and so the following functions are shown as red (covered but not executed).
- The `else` of the second `if` statement applies and so the following function is executed and shown as green.
- The `break` of the `case` statement is shown as yellow or partially covered (covered but partially executed) which seems unexpected as both `if` statements have executed. To understand what is going on with partial coverage it is useful to view the Program Memory window.

Figure 2-6. Program Memory - Demo Code Example

The screenshot shows the MPLAB IDE interface. The top pane displays C code with line numbers 185 to 199. The code includes an `else` block with `LCD_ShiftCursorUp ( ) ;` and a `break ;` statement, followed by a `case '\\b':` block with `LCD_ShiftCursorLeft ( ) ;`, `LCD_PutChar ( ' ' ) ;`, `LCD_ShiftCursorLeft ( ) ;`, and `break ;`, and a `case '\\f':` block with `LCD_ClearScreen ( ) ;` and `break ;`. The code is color-coded: the `else` block is pink, the `case '\\b':` block is red, and the `case '\\f':` block is green. The bottom pane shows the Program Memory window with a breadcrumb trail: `LCD_PutChar > switch (inputCharacter) > case '\\n' > if (row == 0) > else >`. The window has tabs for Code Coverage, Variables, Call Stack, Breakpoints, Output, and Program Memory. The Program Memory tab is active, showing a table of instructions.

Line	Address	Opcode	Label	DisAssy
2,557	013F8	070051		RCALL_LCD_CarriageReturn, .LFE3, .LFB4, .L78, .L84
2,558	013FA	BFC870		MOV.B row, WREG
2,559	013FC	E00400		CPO.B W0
2,560	013FE	3A0003		BRA NZ, .L37, .L61
2,561	01400	A8E859		BSET 0x859, #7
2,562	01402	0700C7		RCALL_LCD_ShiftCursorDown, .LFE7, .LFB8, .L136, .L142
2,563	01404	370033		BRA .L35, .L72
2,564	01406	A8085A		BSET 0x85A, #0
2,565	01408	0700B7		RCALL_LCD_ShiftCursorUp, .LFE6, .LFB7, .L125, .L131
2,566	0140A	370030		BRA .L35, .L72
2,567	0140C	A8C858		BSET __cc_bits_lcd_c_4c21a10b, #6
2,568	0140E	07005D		RCALL_LCD_ShiftCursorLeft, .LFE4, .LFB5, .L89, .L99
2,569	01410	B3C200		MOV.B #0x20, W0
2,570	01412	07FFD8		RCALL_LCD_PutChar
2,571	01414	07005A		RCALL_LCD_ShiftCursorLeft, .LFE4, .LFB5, .L89, .L99
2,572	01416	37002A		BRA .L35, .L72
2,573	01418	A84859		BSET 0x859, #2
2,574	0141A	07002E		RCALL_LCD_ClearScreen

To see the view, shown in the image above:

- Halt program execution.
- Open the Program Memory window: *Window>Target Memory Views>Program Memory*.
- Place a breakpoint at the second `if` statement of `case '\\n'` in the Editor window.
- Debug until the program pauses at the breakpoint.
- Step Into once.

In the Program Memory window, you can see the instructions for the `else` clause and `break` statement. The compiler has combined the jump back from the function call and the jump out of the `case` loop into one jump, which associates the `break` statement with the loop. Therefore it appears yellow since the `if` statement condition for true was never executed.

Viewing Program Memory can help you understanding why a line is partially covered (yellow). In general partially-covered lines can be minimized by writing tests to remove red lines (covered by not executed). Then the remaining yellow lines can be examined in assembly in the Program Memory window.

## 2.7.1.2 Code Coverage Window

To open the Code Coverage window, select *Windows>Debugging>Code Coverage*. This window shows you how successful your test code was at covering your application code.

When program execution is halted, the current coverage percentages for each file in the application will be shown. Click on the arrow to see a breakdown of coverage for functions in the file.

Coverage is expressed in Address Units, where an Address Unit represent the atomic unit of memory addressable by the execution portion of the project device architecture.

The Coverage percentage (in green) represents  $x/(x+y)$ , where  $x$  = address units covered and  $y$  = address units not covered.

**Figure 2-7. Code Coverage Window**

File Name	Address Units Covered	Address Units Not Covered	Coverage
adc.c	184	94	66%
ADC_Read10bit	86	12	87%
ADC_ChannelEnable	28	12	70%
ADC_ReadPercentage	0	66	0%
ADC_SetConfiguration	70	4	94%
leds.c	162	558	22%
main.c	134	0	100%
lcd.c	522	188	73%
system.c	10	144	6%
buttons.c	74	90	45%
timer_1ms.c	322	84	79%
lcd_printf.c	22	0	100%
rtcc.c	922	282	76%

## 2.7.2 Simple Code Example

Figure 2-8. Editor Window - Simple Code Example

```

1  /*
2  * File:   main.c
3  */
4
5  #include "xc.h"
6
7  volatile int i, j = 0, n = 5, m;
8  volatile int a = 1, b = 0, c, d = 10, e = 20;
9
10 int main(void) {
11
12     // simple if statement
13     if(a < b ) j = 1;
14
15     // for statement example
16     for (i = 0; i < n; i++) {
17         m = j++;
18         if (m > i){
19             break;
20         }
21     }
22
23     // if-else example
24     if (a || b)
25         c = calcAdd(d,e);
26     else
27         c = calcSub(d,e);
28
29     // while example
30     //while(1);
31
32     return 0;
33 }
34

```

Simple C code constructs are used to demonstrate code coverage.

### 2.7.2.1 Simple Code and Coverage Colors

#### No Color - No Coverage Data

Several C constructs do not generate executable code, so have no highlight. Examples shown above include:

- Preprocessor declarations - #include statements
- Variable declarations and initializations
- Comments or commented out code

#### Green - Covered and Executed

Full coverage for a line occurs when the code on that line is executed completely.

The `main()` function executes and returns completely, so the beginning, return and ending lines show full coverage.

The lines of the `for` loop are all executed to completion except for the `break` statement, because the `if` condition is never met. In loops and conditional statements, there are one or more conditions or branches which must all be tested for full coverage.

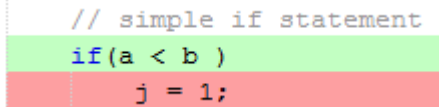
Functions that are called and complete on a line, such as `calcAdd()`, are fully covered.

### Yellow - Partial Coverage

When code on a line is covered but only partially executed, this is considered partial coverage.

For the simple `if` statement, the evaluation of `a<b` is executed but the assignment `j=1` is not. Therefore the line is partially covered. If the statements were on different lines, you will see:

```
// simple if statement
if(a < b )
    j = 1;
```



The `if-else` statement uses binary logic to determine the branch condition. Because of the variable values chosen, the statement is only partially covered because when `a` has a nonzero value, `b` will not be evaluated.

### Red - Covered but not Executed

As discussed above, in loops and conditional statements there are one or more conditions or branches which must all be tested for full coverage.

The `if` statement in the `for` loop is never true, so the following `break` statement is not executed.

The condition of the `if-else` statement is evaluated so the `else` branch is never selected and the following line with `calcSub()` is not executed.

## 2.7.2.2 Limitations of Coverage in the Editor

The Editor window displays lines of C code. C is a high-level language where one line of C code can represent one or more device instructions, depending on the device. Therefore, looking at the actual device instructions in the Program (or Execution Memory) window may be necessary to determine the reason(s) for the coverage shown.

An example is the `while(1)` loop commented out in the code. If it is uncommented and the program is run again and then paused, the following text may display in the Output window:

```
No source code lines were found at current PC 0x330. Open program memory view to see
instruction code disassembly.
```

To open the Program Memory (8- or 16-bit devices) window, select *Windows>Target Memory Views>Program Memory*. To open the Execution Memory (32-bit devices) window, select *Windows>Target Memory Views>Execution Memory*.

Figure 2-9. Program Memory - Simple Code Example

```

29 // while example
30 while(1);
31
32 return 0;
33 }
34

```

Line	Address	Opcode	Label	DisAssy
400	0031E	070009		RCALL calcAdd
401	00320	884040		MOV W0, c
402	00322	370005		BRA .L9, .L26
403	00324	A8A814		BSET __cc_bits_main_c_4d7c040b, #5
404	00326	804081		MOV 0x810, W1
405	00328	804070		MOV 0x80E, W0
406	0032A	07000B		RCALL calcSub
407	0032C	884040		MOV W0, c
408	0032E	A8E814		BSET __cc_bits_main_c_4d7c040b, #7
409	00330	37FFFE		BRA .L9, .L26
410	00332	FA0004	calcAdd	LNK #0x4
411	00334	A80818		BSET __cc_bits_math_c_4dae163e, #0
412	00336	780F00		MOV W0, [W14]
413	00338	980711		MOV W1, [W14+2]

At address 0x330 is a branch instruction; therefore the compiler has generated this representation of while(1). This will not be seen as source code and so the Output text is generated.

### 2.7.2.3 Code Coverage by Project File

For the Simple Code Example, the Code Coverage window percentages are not difficult to see given the coverage in the Editor window. For more complex applications (such as the Demo Code), the breakdown of coverage by file and function provides useful information for improving tests and coverage.

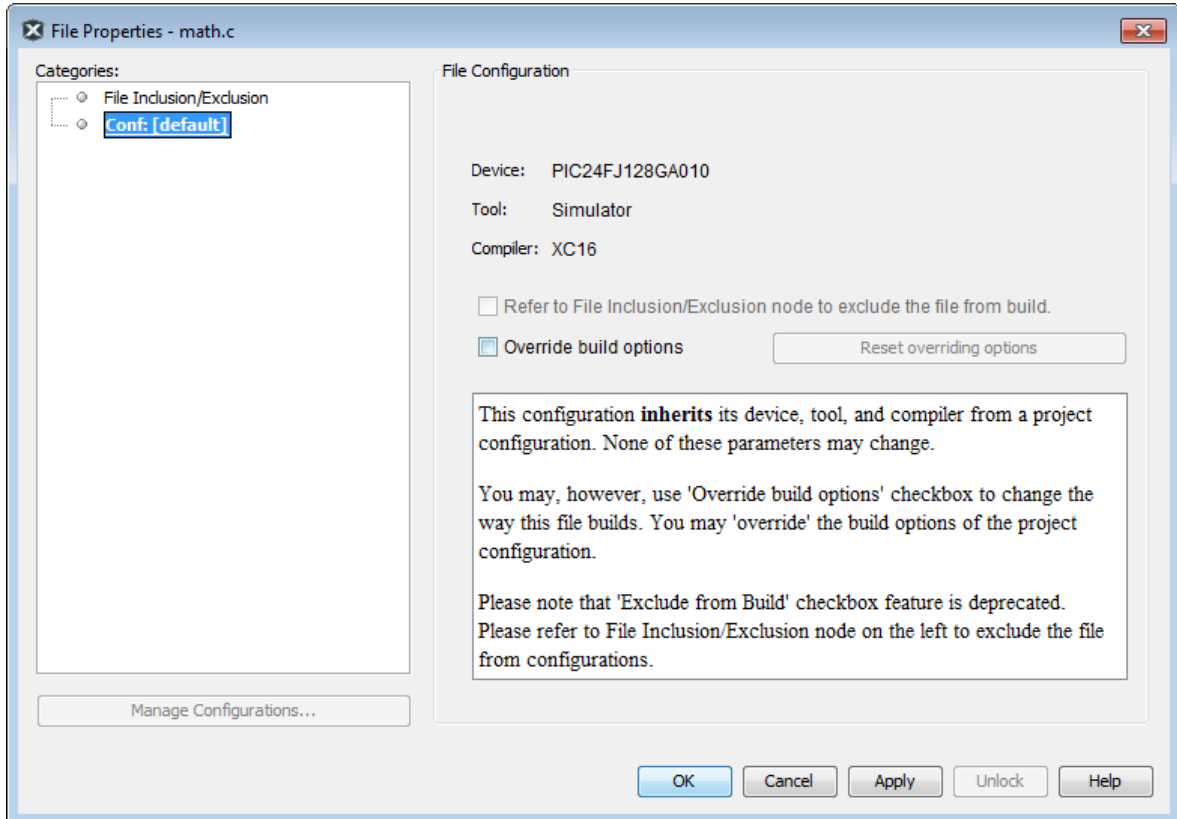
Figure 2-10. Code Coverage Window - Simple Code Example

File Name	Address Units Covered	Address Units Not Covered	Coverage
math.c	16	16	50%
calcAdd	16	0	100%
calcSub	0	16	0%
main.c	72	14	83%
main	72	14	83%

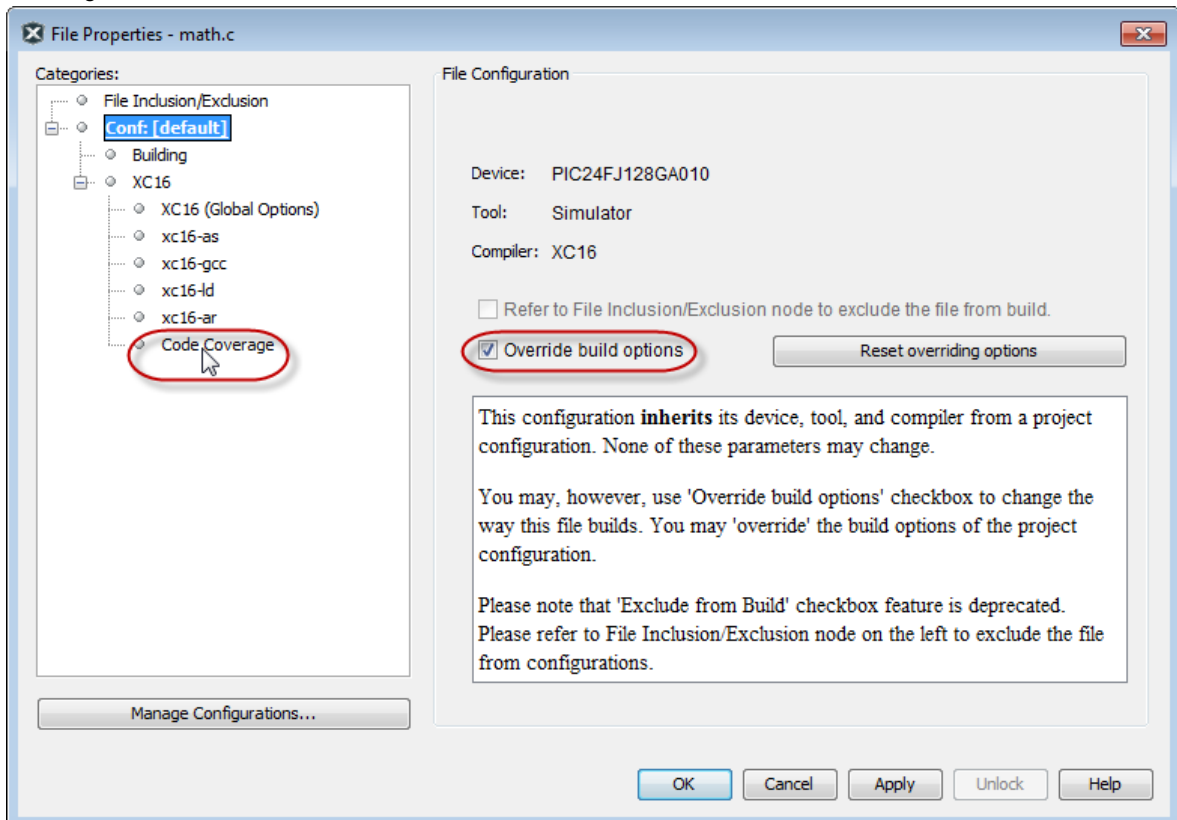
To provide targeted code coverage for specific file(s) in the project, you can change the file build properties.

1. Right click on a project file and select “Properties” from the drop-down menu. The “File Properties” dialog will open.

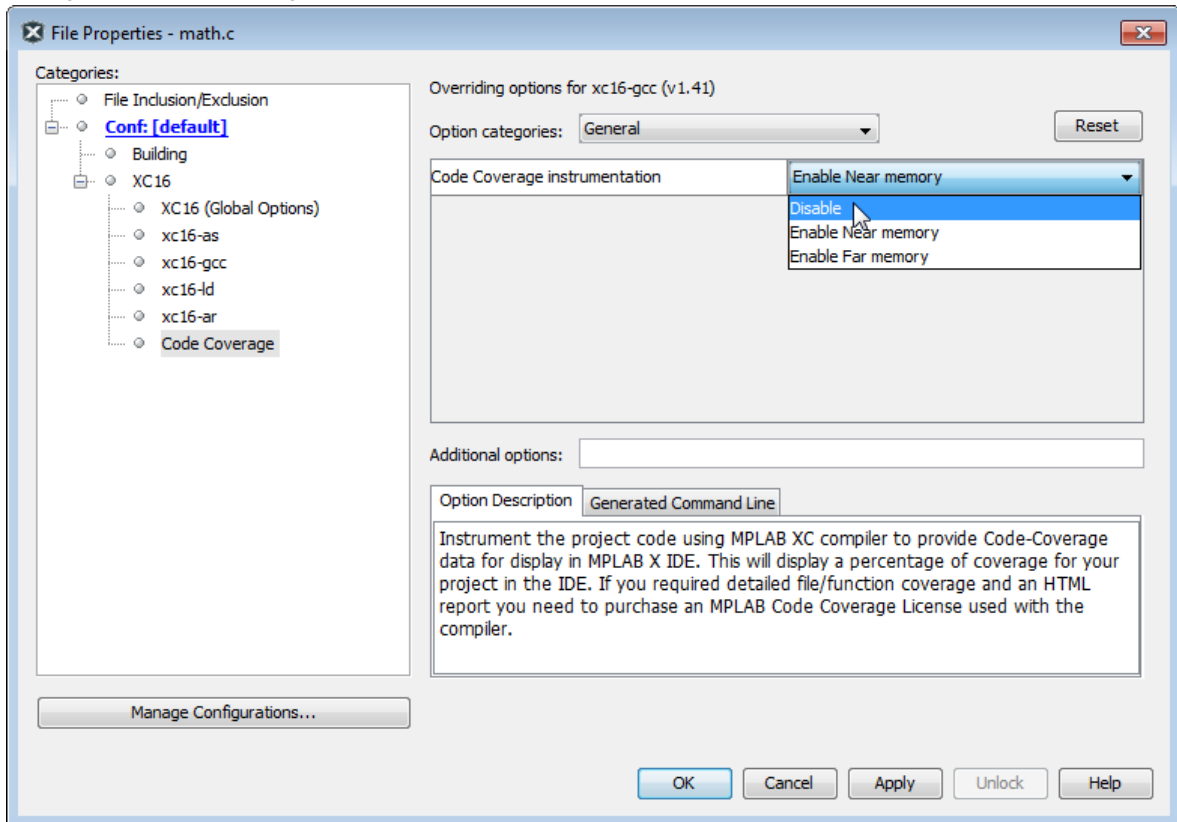




- In this dialog, check "Override build options." Now other build option selections will appear. Click on "Code Coverage."



3. Change the “Code Coverage instrumentation” option to “Disable” and then click **OK**.



4. The file will now show in the project as bolded. Run and halt the project and examine the Code Coverage window. The file will no longer be listed here. Open the file in the Editor is see that no code coverage highlighting is available.

The screenshot displays the MPLAB X IDE v5.30 interface for a project named 'SimpleXC16Example'. The main window shows the source code for 'math.c' with the following content:

```

1  /*
2  * File:   math.c
3  */
4
5  /*
6  * Prototypes
7  */
8  int calcAdd(int a,int b);
9  int calcSub(int a,int b);
10
11 /*
12 * Functions
13 */
14 int calcAdd(int a,int b){
15     return a+b;
16 }
17
18 int calcSub(int a,int b){
19     return a-b;
20 }


```

Below the code editor, the 'Code Coverage' tab is active, displaying a table with the following data:

File Name	Address Units Covered	Address Units Not Covered	Coverage
main.c	72	14	83%

The status bar at the bottom indicates 'SimpleXC16Example (Build, Load, ...)' and 'debugger halted'.

## 2.8 Create a Code Coverage HTML Report

Code coverage information may be saved into a file by clicking on the **Generate HTML Report** button  on the **Code Coverage** tab.

For information on the meaning of Coverage colors, see section 2.6.1. [Highlight Colors](#).

Figure 2-11. Source Files List and Coverage

Source files
<a href="#">All</a>
<a href="#">adc.c</a>
<a href="#">leds.c</a>
<a href="#">main.c</a>
<a href="#">lcd.c</a>
<a href="#">system.c</a>
<a href="#">buttons.c</a>
<a href="#">timer_1ms.c</a>
<a href="#">lcd_printf.c</a>
<a href="#">rtcc.c</a>

All Source files
<a href="#">adc.c</a> (62%)
<a href="#">leds.c</a> (18%)
<a href="#">main.c</a> (100%)
<a href="#">lcd.c</a> (72%)
<a href="#">system.c</a> (6%)
<a href="#">buttons.c</a> (39%)
<a href="#">timer_1ms.c</a> (79%)
<a href="#">lcd_printf.c</a> (100%)
<a href="#">rtcc.c</a> (78%)

Figure 2-12. Source Files List and Coverage Details

Coverage Report - All Source files			
Package	Address units covered	Address units not covered	Coverage
Source files	2198	1356	61%
<a href="#">adc.c</a>	152	90	
<a href="#">buttons.c</a>	58	90	
<a href="#">lcd.c</a>	488	182	
<a href="#">lcd_printf.c</a>	22	0	
<a href="#">leds.c</a>	114	518	
<a href="#">main.c</a>	134	0	
<a href="#">rtcc.c</a>	910	254	
<a href="#">system.c</a>	10	144	
<a href="#">timer_1ms.c</a>	310	78	
Functions in these source file(s)		Coverage	
<a href="#">ADC_ChannelEnable</a>	66%		
<a href="#">ADC_Read10bit</a>	82%		
<a href="#">ADC_ReadPercentage</a>	0%		
<a href="#">ADC_SetConfiguration</a>	94%		

## 2.9 Command Line Support for Code Coverage in MDB

MPLAB Code Coverage support has been added for command line tools using MDB. To generate a code coverage report in MDB:

- Generate an ELF file by enabling the code coverage property. See 2.5. [Enable/Disable Code Coverage](#).
- Select report type (`html`, `gcov`, `all`) – see table below.
- Provide `html` report path (Default: project location)– see table below.
- Set `replacehtmlreport` to true if the report already exists– see table below.
- An HTML Report will be generated if you invoke the below example using MDB.

### Example

```
Device PIC16F886
set xccodecoverage.reporttype html
set xccodecoverage.htmlreportpath d:\report
set xccodecoverage.replacehtmlreport true
Hwtool SIM
Program "d:\testCoverage.elf"
Break main.c:41
Run
Wait 2000
Quit
```

Refer to the Microchip Debugger (MDB) User's Guide (DS50002102) for more information on command options. The online help is located at [onlinedocs.microchip.com/](http://onlinedocs.microchip.com/) where you can search for "Microchip Debugger." A PDF of the MDB User's Guide can be found on the MPLAB X IDE web page [www.microchip.com/en-us/development-tools-tools-and-software/mplab-x-ide](http://www.microchip.com/en-us/development-tools-tools-and-software/mplab-x-ide), under the **Documentation** tab.

---

## 3. MISRA Check

### 3.1 MISRA Check Overview

The MPLAB®MISRA® Check feature in the MPLAB Analysis Tool Suite is a set of C coding standards developed by the Motor Industry Software Reliability Association (MISRA). MISRA guidelines help to ensure safe, secure, portable and reliable C code in embedded systems.

By executing a 'MISRA Check' from the MPLAB X IDE Tools menu, static code analysis is done against a set of MISRA rules. In MPLAB X IDE, MISRA C:2012 rules are used.

The MISRA Check feature is an analysis tool bundled in the MPLAB X IDE v6.0 or later. See [AoU-04-MISRA].

The MPLAB Analysis Tool Suite license includes MISRA Check (see 1.1. Licenses). See [AoU-08-MISRA].

### 3.2 Using MISRA Check in a Safety Environment



**Attention:** To use the MPLAB Analysis Tool Suite in a Safety Environment, you must use a functional safety version of the Microchip compiler. The newer function safety versions of the compilers that support the MPLAB Analysis Tool Suite may not be available yet. Check the [www.microchip.com/en-us/solutions/functional-safety/mplab-development-ecosystem-for-functional-safety](http://www.microchip.com/en-us/solutions/functional-safety/mplab-development-ecosystem-for-functional-safety) for the appropriate functional safety compiler licenses:

- MPLAB XC8 Functional Safety License (with MPLAB Analysis Tool Suite support)
- MPLAB XC16 Functional Safety License (with MPLAB Analysis Tool Suite support)
- MPLAB XC32 Functional Safety License (with MPLAB Analysis Tool Suite support)

MISRA Check has been designed according to the Microchip Development Tools standard process. Microchip asserts certain Assumptions of Use (AoUs) to meet technical and functional safety requirements at the system level. The AoUs are described below. References to some of these AoUs can be found in other sections in this manual, where they are addressed as part of a specific topic being discussed.

It is the responsibility of the System Integrator to address all the AoUs listed in this manual and to ensure that the AoUs listed have been observed by every component or item, contributed to the application being analyzed by MISRA Check.

The System Integrator has two options:

- Make sure each assumption is fulfilled
- Disregard the assumption

In both cases, the System Integrator shall provide evidence of the fulfillment and/or a detailed explanation of why disregarding the assumption will not violate a safety requirement, or how the assumption has been sufficiently addressed in a different manner.

As indicated by the classification documents provided by Microchip along with the functional safety version of the compiler, the MISRA Check (be very careful, this cannot be an approximation) is classified TCL 1. This classification is based on the assumptions of use and use cases provided in the classification documents.

**Note:** MISRA Check has been designed according to the Microchip Development Tools standard process. Microchip asserts certain Assumptions of Use (AoUs) to meet technical and functional safety requirements at the system level. The AoUs are described below. References to some of these AoUs can be found in other sections in this manual, where they are addressed as part of a specific topic being discussed. The most important Assumption of Use is that MISRA Check must be cross-checked against another tool of the same type (e.g. This MISRA Check must be cross-checked with another, independent MISRA Check) or another method to increase confidence of the results presented.

Table 3-1. Degree of Obligation

Shall	Binding
Should	Recommendation


Assumption of Use	Description
[AoU-01-MISRA]	MISRA Check shall be cross-checked against another tool of the same type, or another method to increase confidence of the results presented.
[AoU-02-MISRA]	MISRA Check shall only be used in conjunction with supported Functional Safety versions of the MPLAB XC compilers.
[AoU-03-MISRA]	MISRA Check shall be used in conjunction with a bona fide Functional Safety process.
[AoU-04-MISRA]	MISRA Check shall be used with a suitable version of the MPLAB X IDE v6.0 or higher on supported platforms as specified in MPLAB X IDE documentation
[AoU-05-MISRA]	Users of MISRA Check shall possess a valid license.
[AoU-06-MISRA]	MISRA Check shall only be used on supported Microchip microcontrollers and DSCs.
[AoU-07-MISRA]	The MISRA Check shall be properly configured for the rules and files being checked to ensure the output reports are based on the interface used.
[AoU-08-MISRA]	MISRA Check shall only use the Cpp Check tool shipped with the MPLAB X IDE v6.0 or later.

### 3.3 Perform MISRA Check

Executing static code analysis can help you identifying potential issues or risky constructs in the source code.

#### MISRA Check Execution

Execute static code analysis as specified below.

- **MISRA Check Main Project:** Click on the toolbar icon  or select from the Source menu.
- **MISRA Check Project:** In the Projects window, right click on a project name and select.
- **MISRA Check File:** In the Projects window, right click on a project file and select.
- **Run MISRA Check each time project is rebuilt:** Enable as part of build by checking *Tools>Options>Embedded>MISRA Check>Run* each time project is rebuilt.

#### Static Code Analysis Output

The results of static code analysis can be output in one of the following ways.

- **Output window:** Under the “MISRA Check” tab.
- **HTML report file:** Enable in *Tools>Options>Embedded>MISRA Check*.
- **CSV report file:** Enable in *Tools>Options>Embedded>MISRA Check*.

Rule violations are identified with “MISRA rule violated” (rule number and description) and location in file where violation was located.

#### MISRA Rule Settings

MISRA rules settings are located in *Tools>Options>Embedded>MISRA Check*.

#### MISRA Check Execution from the Command Line

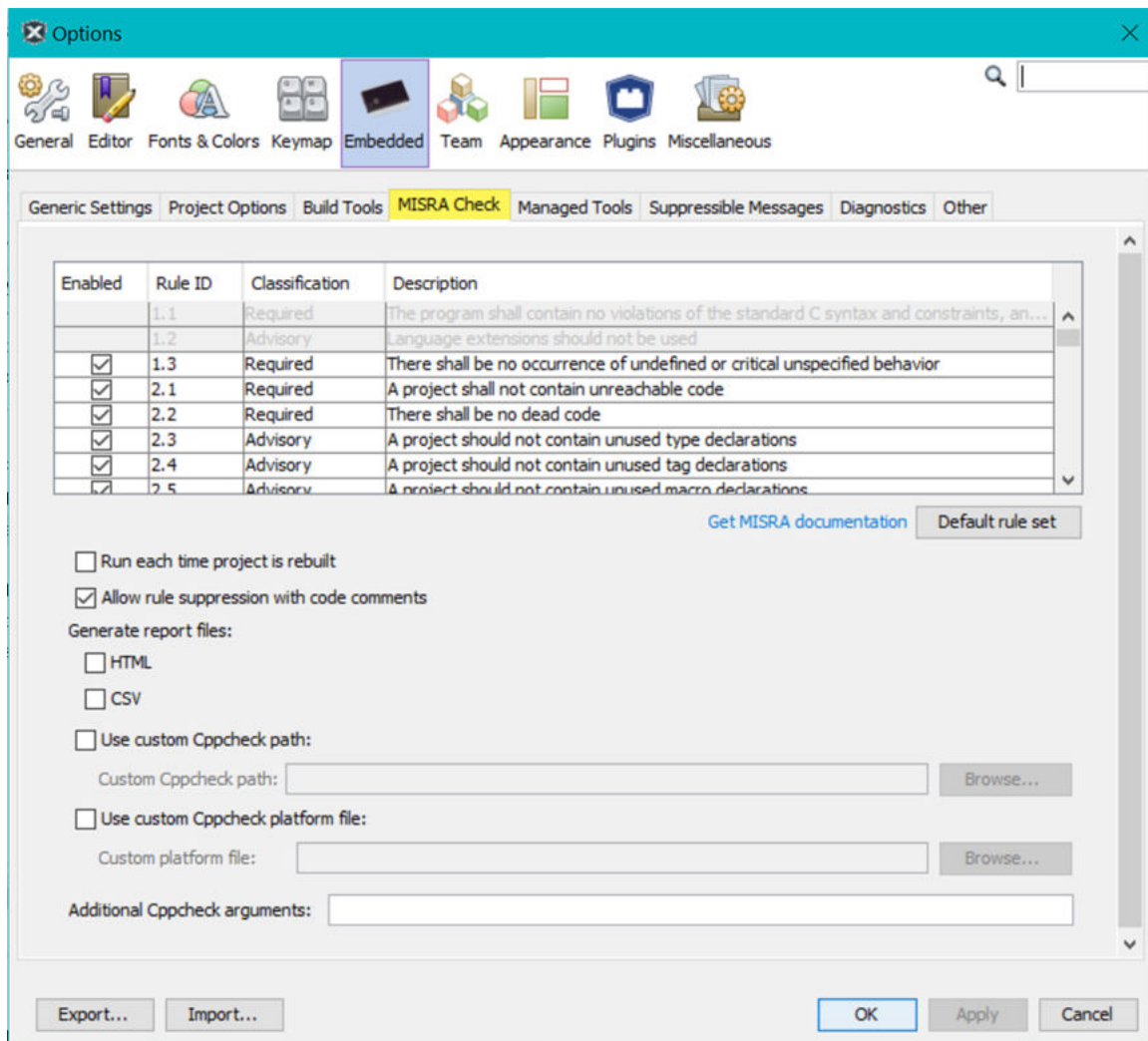
`misracli.bat`, for Windows, or `misracli.sh` for Linux/Mac is the command line application for MISRA rule checking. See [3.5. Command Line Support for MISRA Check](#).



### 3.4 MISRA Check Options Tab

The MISRA® Check tab below supports checking your application code against these standards.

To open the MISRA Check tab, select *Tools>Analysis>MISRA Check>MISRA Options*.



All MISRA C:2012 rules are listed in the table in the above figure. Grayed out rules are to be checked manually. Applied rules can be enabled and disabled.

**Table 3-2. MISRA Check Options**

Option	Description
Run each time project is rebuilt	Check to execute static code analysis as part of the build.
Allow rule suppression with code comments	Suppress rules that have been commented in the code.

.....continued	
Option	Description
Generate report files	<p>Check to generate MISRA Check reports in HTML or CSV format. Reports are timestamped. HTML or CSV reports are written to &lt;project folder&gt;\report. Also a “report” folder is added to the project tree from which the files may be viewed. The HTML report lists:</p> <ul style="list-style-type: none"> <li>• When the check was executed</li> <li>• Rule violations – issues found</li> <li>• What was checked</li> <li>• And which rules was applied</li> </ul> <p>The CVS report lists:</p> <ul style="list-style-type: none"> <li>• Rule violations – issues found</li> </ul>
Use custom Cppcheck path	Provide a link to a custom Cppcheck to override the MPLAB X IDE bundled Cppcheck. Use this to extend/modify/update the applied rule checking.
Additional Cppcheck arguments:	Provide more Cppcheck arguments.

### 3.5 Command Line Support for MISRA Check

MISRA Check support is available as a command line tool. See [AoU-07-MISRA].

#### Installation

- Install the latest MPLAB X IDE.
- Open a native command prompt (Windows, Linux and Mac) and navigate to the MPLAB X IDE installation's bin folder. For example, on Windows: `C:\Program Files\Microchip\MPLABX\v6.0\mplab_platform\bin`.
- The Misra Check command line utility is available as `misracli.bat` (Windows) or `misracli.sh` (Linux/Mac).

#### Examples

The following examples use the Windows version (for Linux/Mac use `misracli.sh` instead of `misracli.bat`).

#### Normal use:

The MISRA Check command line interface will enable all rules by default, but you can disable specific ones using the `-disable` option.

To get a full list of all the available MISRA Check command line interface options run:

```
C:\Program
  Files\Microchip\MPLABX\[vX.XX]\mplab_platform\bin\misracli.bat
  -help
```

- `misracli.bat myFile.c` runs MISRA Check on a single file and report errors in the output.
- `misracli.bat myDirectory` runs MISRA Check recursively on all `.c, .cpp, *.cxx, .cc, .h, .hpp, .hxx` and `*.hh` files under a given directory.
- `misracli.bat` returns 0 if no errors are found and a non-zero value if errors are detected.

#### Special options

- `misracli.bat --help` lists all options with a description.
- `misracli.bat --version` displays a version number.
- Adding the option `--quiet` only displays warnings and errors (example: missing source files or incorrect use of `--cppcheck param`).
- Adding the option `--silent` disables all output (but the return value should be correct and it can be used together with reporting).

- Using the option `--workdir=myDirectory` runs the analysis in another folder than the current working directory.
- Using the option `--cppcheck="C:\Program Files\myCustomCppCheckInstallation\cppcheck.exe"` uses the users installed `cppcheck.exe` (available from [cppcheck.sourceforge.io/](http://cppcheck.sourceforge.io/)).

### Reporting:

- Adding the option `--html=myReport.html` generates an html report.
- Adding the option `--csv=myReport.csv` generates a csv report.

### Rule handling:

- Adding the option `--disable="1.2 1.3"` checks the files but disable checking of rule 1.2 and 1.3.
- Adding the option `--nosuppression` disallows inlined code comments which otherwise would suppress rules.

### 4. Revision History

#### 4.1 Revision A (December 2021)

Initial release of this document.

---

---

## The Microchip Website

---

Microchip provides online support via our website at [www.microchip.com/](http://www.microchip.com/). This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

---

## Product Change Notification Service

---

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to [www.microchip.com/pcn](http://www.microchip.com/pcn) and follow the registration instructions.

---

## Customer Support

---

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: [www.microchip.com/support](http://www.microchip.com/support)

---

---

## Product Identification System

---

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.

**PART NO.**    [X]<sup>(1)</sup> - X    /XX    XXX  
Device    Tape and Reel    Temperature    Package    Pattern  
          Option            Range

Device:	PIC16F18313, PIC16LF18313, PIC16F18323, PIC16LF18323	
Tape and Reel Option:	Blank	= Standard packaging (tube or tray)
	T	= Tape and Reel <sup>(1)</sup>
Temperature Range:	I	= -40°C to +85°C (Industrial)
	E	= -40°C to +125°C (Extended)
Package: <sup>(2)</sup>	JQ	= UQFN
	P	= PDIP
	ST	= TSSOP
	SL	= SOIC-14
	SN	= SOIC-8
	RF	= UDFN
Pattern:	QTP, SQTP, Code or Special Requirements (blank otherwise)	

Examples:

- PIC16LF18313- I/P Industrial temperature, PDIP package
- PIC16F18313- E/SS Extended temperature, SSOP package

### Notes:

1. Tape and Reel identifier only appears in the catalog part number description. This identifier is used for ordering purposes and is not printed on the device package. Check with your Microchip Sales Office for package availability with the Tape and Reel option.
2. Small form-factor packaging options may be available. Please check [www.microchip.com/packaging](http://www.microchip.com/packaging) for small-form factor package availability, or contact your local Sales Office.

---

## Microchip Devices Code Protection Feature

---

Note the following details of the code protection feature on Microchip products:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip product is strictly prohibited and may violate the Digital Millennium Copyright Act.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is “unbreakable”. Code protection is constantly evolving. Microchip is committed to continuously improving the code protection features of our products.

---

## Legal Notice

---

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these

---

terms. Information regarding device applications is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. Contact your local Microchip sales office for additional support or, obtain additional support at [www.microchip.com/en-us/support/design-help/client-support-services](http://www.microchip.com/en-us/support/design-help/client-support-services).

THIS INFORMATION IS PROVIDED BY MICROCHIP "AS IS". MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, OR CONSEQUENTIAL LOSS, DAMAGE, COST, OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION.

Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

## Trademarks

---

The Microchip name and logo, the Microchip logo, Adaptec, AnyRate, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, CryptoMemory, CryptoRF, dsPIC, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

AgileSwitch, APT, ClockWorks, The Embedded Control Solutions Company, EtherSynch, Flashtec, Hyper Speed Control, HyperLight Load, IntelliMOS, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet- Wire, SmartFusion, SyncWorld, Temux, TimeCesium, TimeHub, TimePictra, TimeProvider, TrueTime, WinPath, and ZL are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, Augmented Switching, BlueSky, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, Espresso T1S, EtherGREEN, GridTime, IdealBridge, In-Circuit Serial Programming, ICSP, INICnet, Intelligent Paralleling, Inter-Chip Connectivity, JitterBlocker, Knob-on-Display, maxCrypto, maxView, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, NVM Express, NVMe, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, RTAX, RTG4, SAM-ICE, Serial Quad I/O, simpleMAP, SimpliPHY, SmartBuffer, SmartHLS, SMART-I.S., storClad, SQL, SuperSwitcher, SuperSwitcher II, Switchtec, SynchroPHY, Total Endurance, TSHARC, USBCheck, VariSense, VectorBlox, VeriPHY, ViewSpan, WiperLock, XpressConnect, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

The Adaptec logo, Frequency on Demand, Silicon Storage Technology, Symmcom, and Trusted Time are registered trademarks of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

© 2021, Microchip Technology Incorporated and its subsidiaries. All Rights Reserved.

ISBN: 978-1-5224-9492-8

---

---

AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro,  $\mu$ Vision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

## Quality Management System

---

For information regarding Microchip's Quality Management Systems, please visit [www.microchip.com/quality](http://www.microchip.com/quality).



## Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
<p><b>Corporate Office</b> 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Tel: 480-792-7277 Technical Support: <a href="http://www.microchip.com/support">www.microchip.com/support</a> Web Address: <a href="http://www.microchip.com">www.microchip.com</a></p> <p><b>Atlanta</b> Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455</p> <p><b>Austin, TX</b> Tel: 512-257-3370</p> <p><b>Boston</b> Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088</p> <p><b>Chicago</b> Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075</p> <p><b>Dallas</b> Addison, TX Tel: 972-818-7423 Fax: 972-818-2924</p> <p><b>Detroit</b> Novi, MI Tel: 248-848-4000</p> <p><b>Houston, TX</b> Tel: 281-894-5983</p> <p><b>Indianapolis</b> Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380</p> <p><b>Los Angeles</b> Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800</p> <p><b>Raleigh, NC</b> Tel: 919-844-7510</p> <p><b>New York, NY</b> Tel: 631-435-6000</p> <p><b>San Jose, CA</b> Tel: 408-735-9110 Tel: 408-436-4270</p> <p><b>Canada - Toronto</b> Tel: 905-695-1980 Fax: 905-695-2078</p>	<p><b>Australia - Sydney</b> Tel: 61-2-9868-6733</p> <p><b>China - Beijing</b> Tel: 86-10-8569-7000</p> <p><b>China - Chengdu</b> Tel: 86-28-8665-5511</p> <p><b>China - Chongqing</b> Tel: 86-23-8980-9588</p> <p><b>China - Dongguan</b> Tel: 86-769-8702-9880</p> <p><b>China - Guangzhou</b> Tel: 86-20-8755-8029</p> <p><b>China - Hangzhou</b> Tel: 86-571-8792-8115</p> <p><b>China - Hong Kong SAR</b> Tel: 852-2943-5100</p> <p><b>China - Nanjing</b> Tel: 86-25-8473-2460</p> <p><b>China - Qingdao</b> Tel: 86-532-8502-7355</p> <p><b>China - Shanghai</b> Tel: 86-21-3326-8000</p> <p><b>China - Shenyang</b> Tel: 86-24-2334-2829</p> <p><b>China - Shenzhen</b> Tel: 86-755-8864-2200</p> <p><b>China - Suzhou</b> Tel: 86-186-6233-1526</p> <p><b>China - Wuhan</b> Tel: 86-27-5980-5300</p> <p><b>China - Xian</b> Tel: 86-29-8833-7252</p> <p><b>China - Xiamen</b> Tel: 86-592-2388138</p> <p><b>China - Zhuhai</b> Tel: 86-756-3210040</p>	<p><b>India - Bangalore</b> Tel: 91-80-3090-4444</p> <p><b>India - New Delhi</b> Tel: 91-11-4160-8631</p> <p><b>India - Pune</b> Tel: 91-20-4121-0141</p> <p><b>Japan - Osaka</b> Tel: 81-6-6152-7160</p> <p><b>Japan - Tokyo</b> Tel: 81-3-6880-3770</p> <p><b>Korea - Daegu</b> Tel: 82-53-744-4301</p> <p><b>Korea - Seoul</b> Tel: 82-2-554-7200</p> <p><b>Malaysia - Kuala Lumpur</b> Tel: 60-3-7651-7906</p> <p><b>Malaysia - Penang</b> Tel: 60-4-227-8870</p> <p><b>Philippines - Manila</b> Tel: 63-2-634-9065</p> <p><b>Singapore</b> Tel: 65-6334-8870</p> <p><b>Taiwan - Hsin Chu</b> Tel: 886-3-577-8366</p> <p><b>Taiwan - Kaohsiung</b> Tel: 886-7-213-7830</p> <p><b>Taiwan - Taipei</b> Tel: 886-2-2508-8600</p> <p><b>Thailand - Bangkok</b> Tel: 66-2-694-1351</p> <p><b>Vietnam - Ho Chi Minh</b> Tel: 84-28-5448-2100</p>	<p><b>Austria - Wels</b> Tel: 43-7242-2244-39 Fax: 43-7242-2244-393</p> <p><b>Denmark - Copenhagen</b> Tel: 45-4485-5910 Fax: 45-4485-2829</p> <p><b>Finland - Espoo</b> Tel: 358-9-4520-820</p> <p><b>France - Paris</b> Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79</p> <p><b>Germany - Garching</b> Tel: 49-8931-9700</p> <p><b>Germany - Haan</b> Tel: 49-2129-3766400</p> <p><b>Germany - Heilbronn</b> Tel: 49-7131-72400</p> <p><b>Germany - Karlsruhe</b> Tel: 49-721-625370</p> <p><b>Germany - Munich</b> Tel: 49-89-627-144-0 Fax: 49-89-627-144-44</p> <p><b>Germany - Rosenheim</b> Tel: 49-8031-354-560</p> <p><b>Israel - Ra'anana</b> Tel: 972-9-744-7705</p> <p><b>Italy - Milan</b> Tel: 39-0331-742611 Fax: 39-0331-466781</p> <p><b>Italy - Padova</b> Tel: 39-049-7625286</p> <p><b>Netherlands - Drunen</b> Tel: 31-416-690399 Fax: 31-416-690340</p> <p><b>Norway - Trondheim</b> Tel: 47-72884388</p> <p><b>Poland - Warsaw</b> Tel: 48-22-3325737</p> <p><b>Romania - Bucharest</b> Tel: 40-21-407-87-50</p> <p><b>Spain - Madrid</b> Tel: 34-91-708-08-90 Fax: 34-91-708-08-91</p> <p><b>Sweden - Gothenberg</b> Tel: 46-31-704-60-40</p> <p><b>Sweden - Stockholm</b> Tel: 46-8-5090-4654</p> <p><b>UK - Wokingham</b> Tel: 44-118-921-5800 Fax: 44-118-921-5820</p>