MPLAB X IDE 用户指南

MPLAB® X IDE 用户指南

客户须知

重要：
本文档如同所有其他文档一样具有时效性。Microchip 将不断改进工具和文档以满足客户的需求，因此实际使用中有些对话框和/或工具说明可能与本文档所述之内容有所不同。请访问我们的网站（www.microchip.com）获取最新文档。

文档均标记有“DS”编号。该编号出现在每页底部的页码之前。DS 编号的命名约定为“DSXXXXXXXXA_CN”，其中“XXXXXXXX”为文档编号，“A”为文档版本。

欲了解开发工具的最新信息，请参见 MPLAB® IDE 在线帮助。从 Help（帮助）菜单选择 Help Content（帮助内容），打开现有在线帮助文件列表。

目录

1. 什么是 MPLAB X IDE？

MPLAB® X IDE 是一款软件程序，用于为 Microchip 单片机和数字信号控制器开发应用程序。

该开发工具称为集成开发环境或 IDE，因为它提供了单一的集成“环境”来开发用于嵌入式单片机的代码。MPLAB X IDE 包含功能强大的工具，可帮助您探索、配置、开发、调试和验证嵌入式设计。MPLAB X IDE 与 MPLAB 开发生态系统（软件和工具）无缝协作，其中许多软件和工具都是完全免费的。

1.1 嵌入式系统概述

嵌入式系统通常为一种利用小型单片机（如 Microchip PIC®或 AVR®单片机（MCU））功能的设计。这些单片机将微处理器（类似于个人计算机中的 CPU）与某些称为外设的附加电路相结合，加上同一芯片上的其他电路就构成了一个需要极少其他外部器件的小型控制模块。此单个器件可以被嵌入到其他电子和机械设备中，以实现低成本的数字控制。

1.1.1 嵌入式控制器和个人计算机之间的区别

嵌入式控制器和个人计算机之间的主要区别在于：嵌入式控制器专用于某项特定任务，或一组特定任务。而个人计算机则设计为能够运行多种不同类型的程序，并能够连接到多种不同的外部设备。嵌入式控制器只有一个程序，因此成本低廉，只要能够保证处理专项任务所需的计算能力和硬件即可。

而个人计算机的核心具有比较昂贵的通用中央处理单元（Central Processing Unit，CPU），它还包含了很多其他外部设备（内存、磁盘驱动器、视频控制器和网络接口电路等）。嵌入式系统具有低成本的智能型单片机（MCU），在同一芯片上还有很多外设电路，而外部器件则相对很少。

通常，嵌入式系统属于不可见部件，或者是其他产品（如充电电钻机、冰箱或车库开门器）的子模块。这些产品中的控制器只执行整个设备的一小部分功能。控制器给这些设备中的关键子系统增添了低成本的智能。

举例来说，烟雾探测器就是一种嵌入式系统。它的功能是检测传感器输出的信号，如果信号指示有烟雾存在，则发出警报。可以使烟雾探测器中的小程序无限循环，不停地对烟雾传感器输出的信号进行采样；也可以使烟雾探测器中的小程序处于低功耗的“休眠”模式，由传感器的输出信号将它唤醒，唤醒后烟雾探测器中的小程序就会发出警报。该程序可能还具有一些其他功能，如用户测试功能和电池欠压报警。

虽然配备传感器和音频输出设备的个人计算机通过编程也可以实现上述功能，但这并不是一种节约成本的解决方案（而且它也不可能依靠 9 伏的电池以无人照管方式运行多年）。嵌入式设计使用廉价的单片机，为我们日常生活环境中的方方面面提供智能化处理，如烟雾探测器、相机、手机、家用电器、汽车、智能卡以及安防系统。

1.1.2 单片机的组件

单片机（MCU）具有片上程序存储器（图 1-1 或图 1-2），用于存储固件或编码指令以运行程序（图 1-3 或图 1-4）。程序计数器（Program Counter，PC）用于对程序存储器（包括复位和中断地址）进行寻址。硬件堆栈用于代码中的调用和返回指令，所以它与程序存储器配合工作，但不是后者的一部分。器件数据手册中介绍了程序存储器操作、向量和堆栈的详细信息。

图 1-1. PIC® MCU 数据手册——程序存储器和堆栈

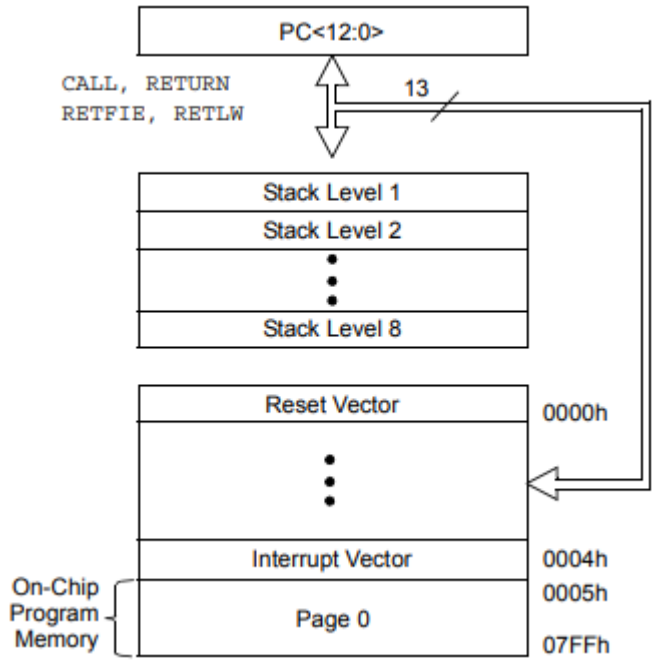


图 1-2. AVR® MCU 数据手册——程序存储器

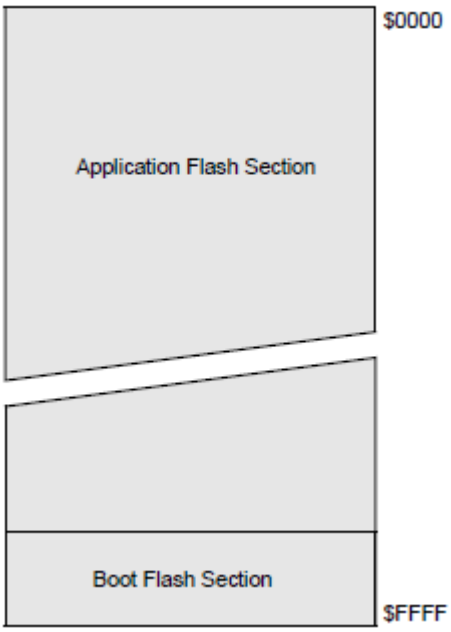


图 1-3. PIC® MCU 数据手册——指令集（摘录）

RRF	Rotate Right f through Carry
Syntax:	[*label*] RRF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	See description below
Status Affected:	C
Description:	The contents of register 'f' are rotated one bit to the right through the Carry flag. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is placed back in register 'f'.

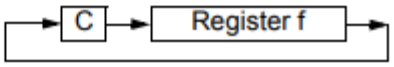


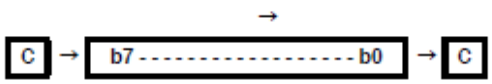
图 1-4. AVR® MCU 指令集（摘录）

ROR – Rotate Right through Carry

Description:

Shifts all bits in Rd one place to the right. The C Flag is shifted into bit 7 of Rd. Bit 0 is shifted into the C Flag. This operation, combined with ASR, effectively divides multi-byte signed values by two. Combined with LSR it effectively divides multi-byte unsigned values by two. The Carry Flag can be used to round the result.

Operation:



	Syntax:	Operands:	Program Counter:
(i)	ROR Rd	0 ≤ d ≤ 31	PC ← PC + 1

16-bit Opcode:

1001	010d	dddd	0111

Status Register (SREG) and Boolean Formula:

I	T	H	S	V	N	Z	C
–	–	–	⇔	⇔	⇔	⇔	⇔

S: N ⊕ V, For signed tests.

V: N ⊕ C (For N and C after the shift)

N: R7
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if the result is $00; cleared otherwise.

C: Rd0
Set if, before the shift, the LSB of Rd was set; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

```
            lsr   r19          ; Divide r19:r18 by two
            ror   r18          ; r19:r18 is an unsigned two-byte integer
            brcc  zeroenc1     ; Branch if carry cleared
            asr   r17          ; Divide r17:r16 by two
            ror   r16          ; r17:r16 is a signed two-byte integer
            brcc  zeroenc2     ; Branch if carry cleared
            ...
zeroenc1:   nop                ; Branch destination (do nothing)
            ...

zeroenc1:   nop                ; Branch destination (do nothing)
```

Words: 1 (2 bytes)
Cycles: 1

单片机还具有数据或“文件寄存器”存储器。该存储器包含特殊功能寄存器（Special Function Register，SFR）和通用寄存器（General Purpose Register，GPR）。SFR 是供 CPU 和外设功能用于控制所需器件操作的寄存器。GPR 用于存储程序进行计算或临时存储所需的变量。一些单片机具有额外的数据 EEPROM 存储器。与程序存储器一样，器件数据手册中介绍了数据存储器使用和操作的详细信息。

表 1-1. PIC® MCU 数据手册——文件寄存器示例

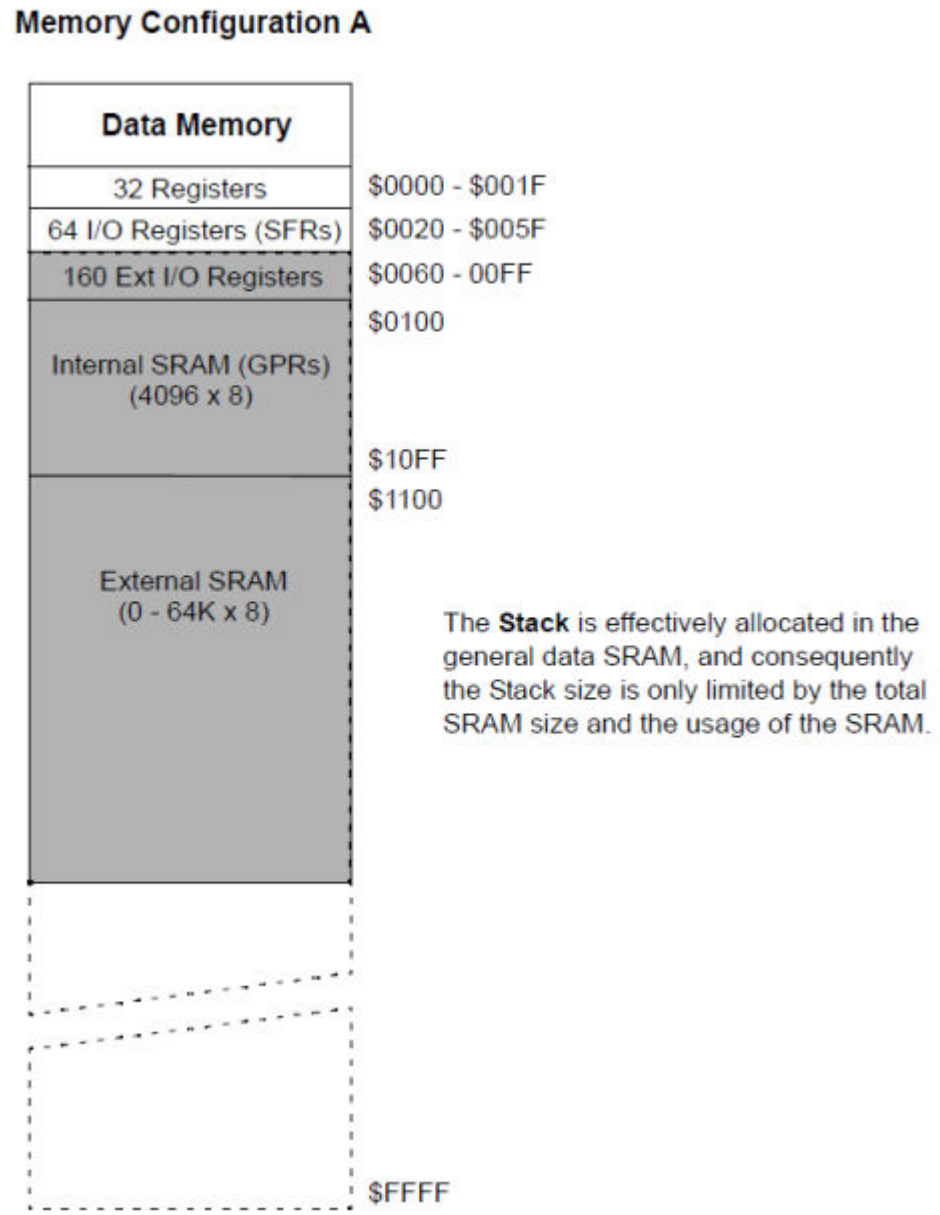
存储区 0	文件地址	存储区 1	文件地址	存储区 2	文件地址	存储区 3	文件地址
间接地址(1)	00h	间接地址(1)	80h	间接地址(1)	100h	间接地址(1)	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h	PORTA	105h	TRISA	185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
PORTC	07h	TRISC	87h	PORTC	107h	TRISC	187h
	08h		88h		108h		188h
	09h		89h		109h		189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDAT	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2(1)	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh		18Eh
TMR1H	0Fh	OSCCON	8Fh	EEADRH	10Fh		18Fh
T1CON	10h	OSCTUNE	90h		110h		190h
TMR2	11h		91h		111h		191h
T2CON	12h	PR2	92h		112h		192h
SSPBUF	13h	SSPADD(2)	93h		113h		193h
SSPCON	14h	SSPSTAT	94h		114h		194h
CCPR1L	15h	WPUA	95h	WPUB	115h		195h
CCPR1H	16h	IOCA	96h	IOCB	116h		196h
CCP1CON	17h	WDTCON	97h		117h		197h
RCSTA	18h	TXSTA	98h	VRCON	118h		198h
TXREG	19h	SPBRG	99h	CM1CON0	119h		199h
RCREG	1Ah	SPBRGH	9Ah	CM2CON0	11Ah		19Ah
	1Bh	BAUDCTL	9Bh	CM2CON1	11Bh		19Bh
PWM1CON	1Ch		9Ch		11Ch		19Ch
ECCPAS	1Dh		9Dh		11Dh	PSTRCON	19Dh
ADRESH	1Eh	ADRESL	9Eh	ANSEL	11Eh	SRCON	19Eh
ADCON0	1Fh	ADCON1	9Fh	ANSELH	11Fh		19Fh

.....（续）

存储区 0	文件地址	存储区 1	文件地址	存储区 2	文件地址	存储区 3	文件地址
通用寄存器 96 字节	20h	通用寄存器 80 字节	A0h	通用寄存器 80 字节	120h		1A0h
			EFh		16Fh		
		访问 70h-7Fh	F0h	访问 70h-7Fh	170h	访问 70h-7Fh	1F0h
	7Fh		FFh		17Fh		1FFh

空单元格：未实现数据存储单元，读为 0。
注 1：不是实际存在的寄存器。
注 2：在某些情况下地址 93h 也可以访问 SSP 掩码（SSPMSK）寄存器。

图 1-5. AVR® MCU 数据手册——SRAM 数据存储器和堆栈



除了存储器之外，单片机在同一芯片上还具有一些外设电路。一些外设称为输入/ 输出（Input/Output，I/O）端口。I/O 端口是单片机上的引脚，可以用作输出，并驱动为高电平或低电平以发送信号、闪烁指示灯或驱动扬声器——发送任何能够通过线路传输的信息。这些引脚通常为双向引脚，也可以配置为输入，以使程序能够对外部开关或传感器做出响应，或是与某些外部器件通信。

图 1-6. PIC® MCU 数据手册——框图（摘录）

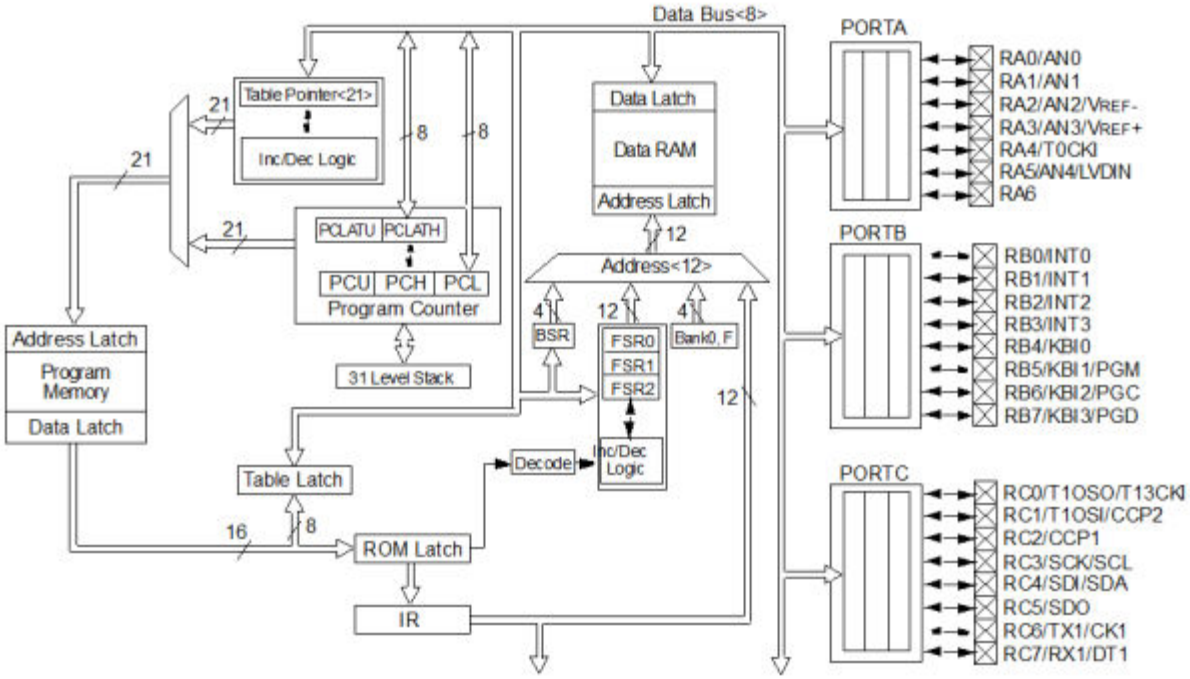
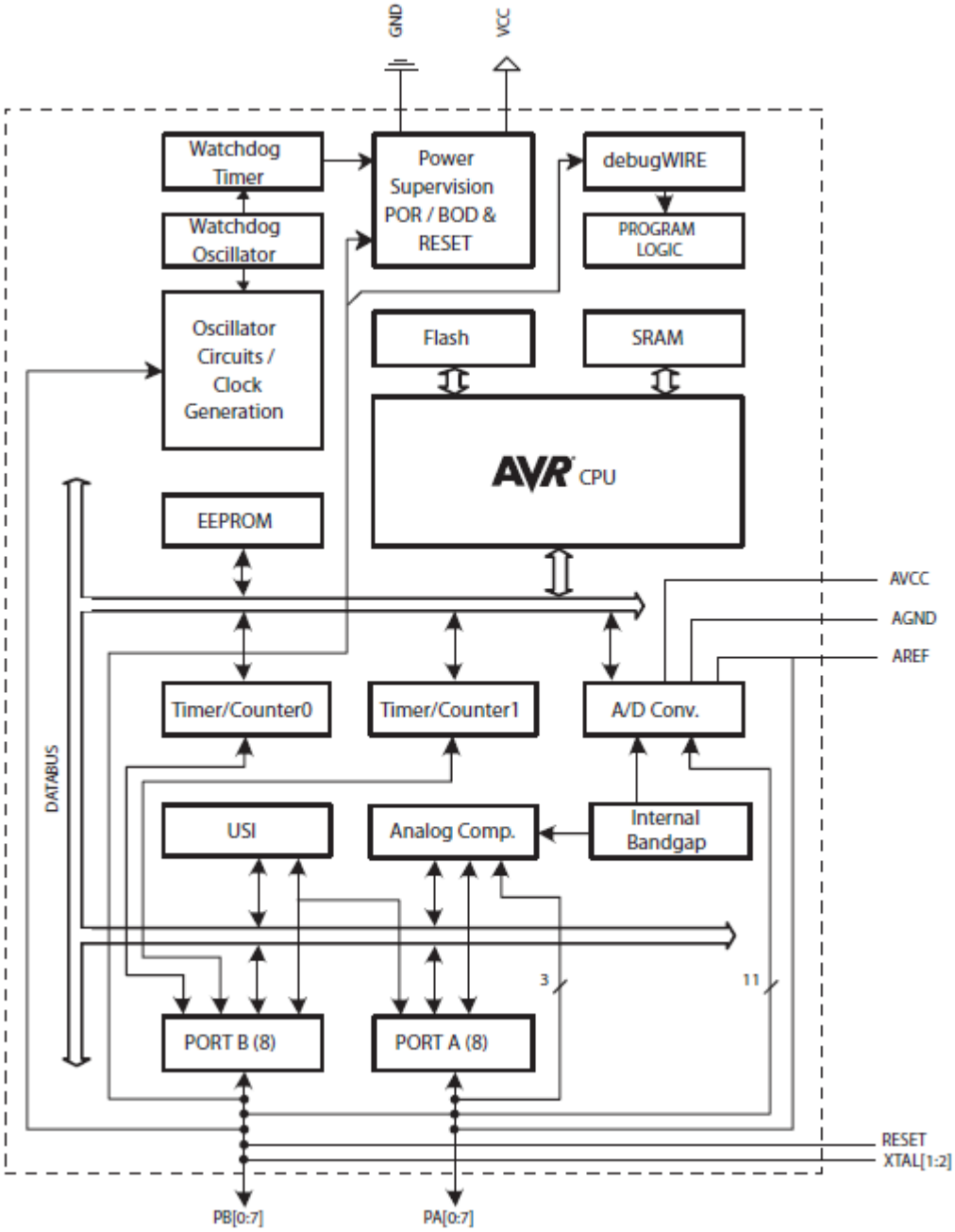


图 1-7. AVR® MCU 数据手册——框图（摘录）



为设计这种系统，选择应用所需的外设。

下面列出了常用的外设：

- 模数转换器（Analog-to-Digital Converter，ADC）允许单片机连接到传感器并接收变化的电压。
- 串行通信外设可以通过几根线缆以串行方式与另一个单片机、局域网或互联网通信。
- PIC MCU 上称为“定时器”的外设可以精确地测量信号事件，并生成和捕捉通信信号以及输出精确的波形，甚至可以在因电源故障或硬件故障导致单片机“挂起”或跑飞时自动将其复位。
- 其他外设可以检测外部电源何时降到了危险值，以便让单片机能够及时存储关键信息，从而在完全掉电之前安全关闭。

应该使用哪种 PIC MCU 在很大程度上取决于应用运行程序所需的外设和存储器大小。其他因素可能包括单片机的功耗及其“外形因素”，即安装到目标设计中的物理封装尺寸和特性。

图 1-8. PIC® MCU 器件封装示例

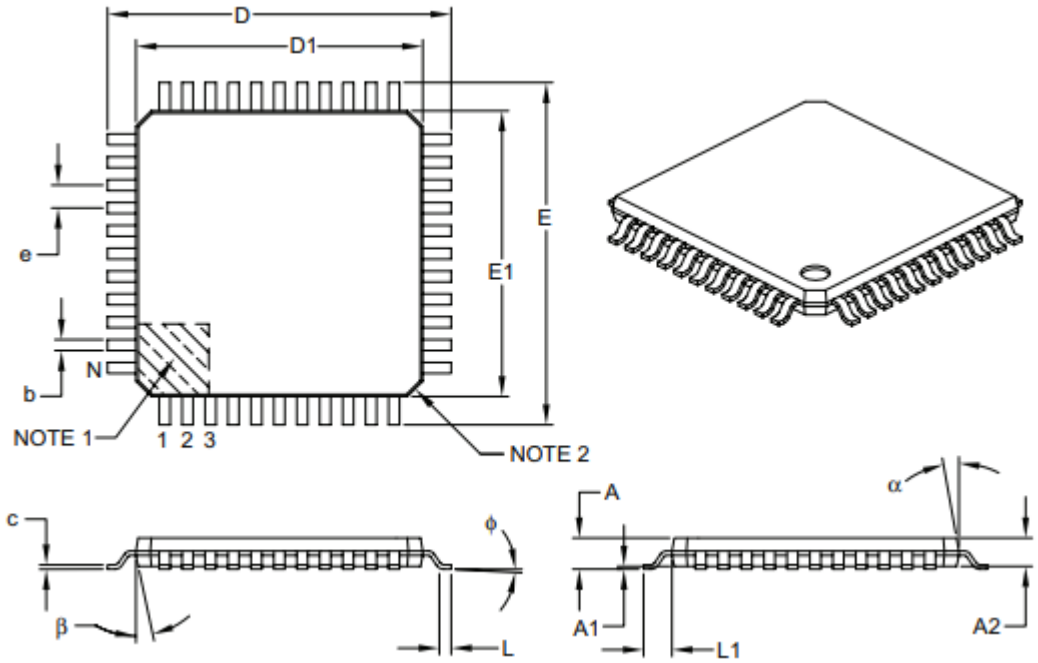
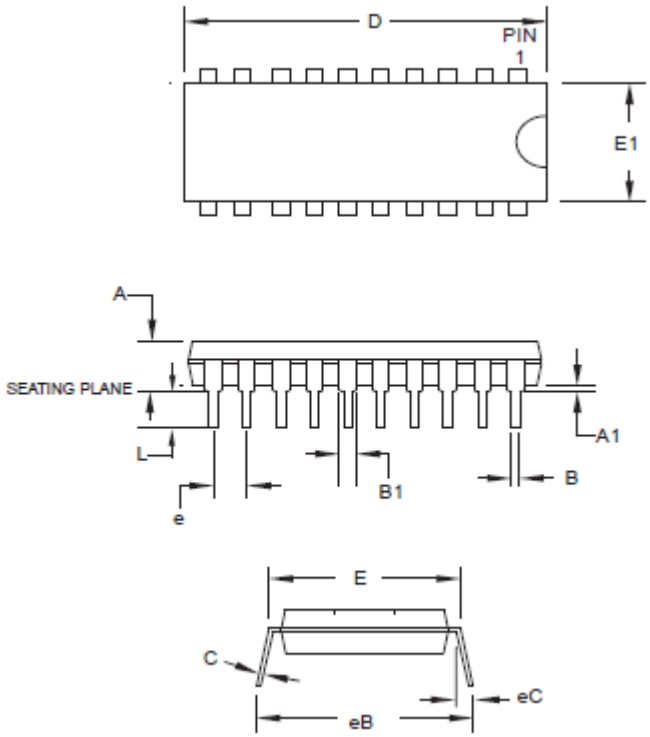


图 1-9. AVR® MCU 器件封装示例



当上电且振荡器开始产生时钟信号时，单片机会变为工作状态。根据不同的单片机型号，可能会有几种内部和外部振荡器工作模式。

图 1-10. PIC® MCU 数据手册——时序图（摘录）

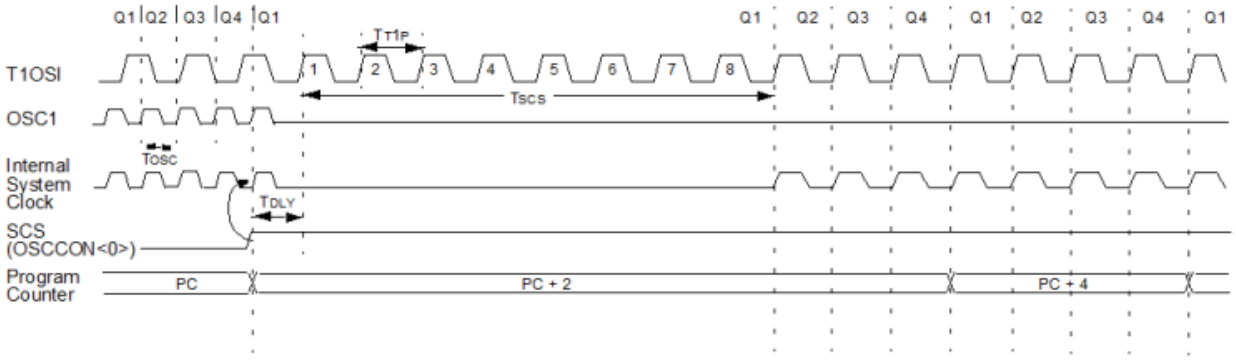
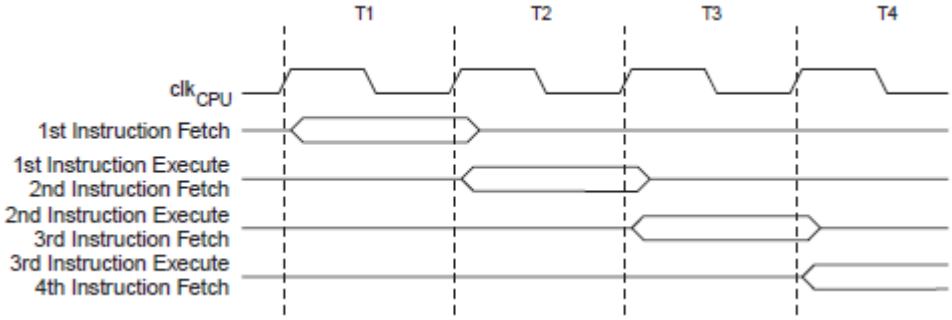


图 1-11. AVR® MCU 数据手册——时序（摘录）



1.1.3 使用 MPLAB X IDE 实现嵌入式系统设计

嵌入式控制器开发系统是在计算机上运行的程序系统，它有助于编写、编辑和调试程序代码（嵌入式系统应用的灵魂），并将其烧写到单片机中。MPLAB X IDE 就是这样一种系统；它包含设计和部署嵌入式系统应用所需的全部组件。

开发嵌入式控制器应用的典型任务有：

1. 创建高阶设计。根据所需的功能和性能，决定最适合应用的 MCU，然后设计相关的硬件电路。在决定由哪些外设和引脚控制硬件之后，编写固件（控制嵌入式应用中的硬件的软件）。可以使用语言工具编写和编辑代码，这些语言工具有汇编器（可以直接将汇编代码转换为机器码）或编译器（允许使用更通用的语言创建程序）。汇编器和编译器允许使用函数标号来标识代码子程序，变量名可与其用途相关联，并采用有助于在可维护的结构中组织代码的程序结构，从而使代码易于理解。
2. 使用汇编器和/或编译器以及链接器编译、汇编和链接软件以将您的代码转换为“0 和 1 序列”——可被 MCU 识别的机器码。机器码最终将变为固件（编程到单片机中的代码）。
3. 测试代码。通常，复杂的程序不一定会按照预期运行，要得到正确的结果，还需要除去设计中的“缺陷”（bug）。您可以通过调试器观察与所编写的带有符号和函数名的程序源代码相对应的机器码中“0 和 1 序列”的执行。在调试过程中，您可以测试代码，观察变量在程序执行过程中各个点的值、进行“what if”检查、更改变量值和单步调试程序。
4. 将代码“烧写”到单片机中，验证其在最终的应用中是否能正确执行。

当然，其中的每个步骤都可能非常复杂。重要的是必须关注设计中的细节，并依靠 MPLAB X IDE 及其组件来完成每个步骤，这样就不会不断地将时间浪费在学习上。

虽然可使用 MPLAB X IDE 对电路和代码进行建模，以便做出关键的设计决定，步骤 1 仍需由设计人员完成。

MPLAB X IDE 真正起帮助作用的是在步骤 2 至步骤 4。它的程序编辑器有助于使用选定的语言工具编写正确的代码。编辑器可以识别汇编器和编译器的编程语法结构，从而自动将源代码以不同颜色区分，这有助于确保代码在语法上的正确性。项目管理器有助于组织应用程序中使用的各种文件：源文件、处理器描述头文件以及库文件。编译了代码之后，您还可以控制编译器以何种程度优化代码大小或执行速度，以及将在器件中的什么位置存储各个变量及程序数据。您也可以指定“存储器模型”以使您的应用能最佳地利用单片机的存储器。如果在编译应用程序时语言工具报

错，则会显示出错的行，双击它即可转到对应的源文件，以便立即编辑。编辑后，可以尝试重新编译您的应用程序。对于复杂的代码，由于要编写和测试很多子代码段，因此通常会经过许多次这样的编写-编译-修正过程。MPLAB X IDE 会以最快的速度执行这一过程，从而使您能够尽快转入下一个步骤。

代码编译没有错误之后，还需要对其进行测试。MPLAB X IDE 具有称为“调试器”的组件和适用于 Microchip 所有 MCU 的免费软件模拟器，可帮助测试代码。即使当硬件还没有完成时，您也可以使用软件模拟器开始测试代码，软件模拟器就是模拟单片机执行的软件程序。软件模拟器可以接收模拟输入（激励信号），以便模拟固件对外部信号的响应。软件模拟器可以测量代码执行时间、单步调试代码以观察变量和外设，并跟踪代码以生成详细的程序运行记录。

一旦硬件进入样机阶段，就可以使用诸如在线仿真器或在线调试器的硬件调试器了。这些调试工具使用在许多带有闪存程序存储器的器件中内置的特殊电路，在实际的应用中实时运行代码。它们可以“查看”目标单片机中的程序和数据存储器，并且可以停止和开始执行程序，使您可以直接使用安装到应用中的单片机测试代码。

应用程序正确运行之后，就可以使用某一 Microchip 器件编程器或开发编程器来对单片机编程了。这些编程器可以校验最终代码是否正确编程到器件中。

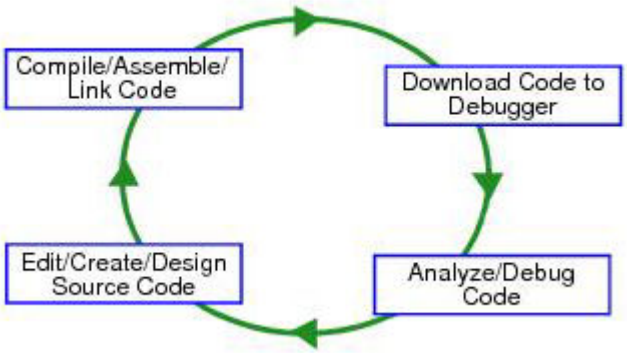
MPLAB X IDE 支持大多数 PIC MCU、所有 dsPIC® DSC 以及越来越多的 AVR 和 SAM MCU。

1.2 开发周期

编写应用程序的过程通常称为开发周期——因为第一次就可以完成从设计到实现的所有步骤而不出任何差错是很少的。通常，编写代码之后要进行测试和修改才能生成正确执行的应用程序。

集成开发环境让嵌入式设计工程师能够顺利完成这种开发周期，而不会因为要在各种工具之间切换而分心。使用 MPLAB X IDE，所有功能都集成在一起，工程师就可以专心完成应用，而不会因为要切换不同的工具和工作模式而中断开发。

图 1-12. 设计周期



MPLAB X IDE 就是一种“包装器”（wrapper），它协调同一图形用户界面的所有工具——这一过程通常是自动完成的。例如，一旦代码编写完成，就可以将其转换为可执行指令，并下载到单片机中观察它的运行。这一过程需要多种工具：编写代码的编辑器、管理文件及设置的项目管理器、将源代码转换为机器码的编译器或汇编器，以及某种连接到目标单片机的硬件或用来模拟单片机运行的软件。

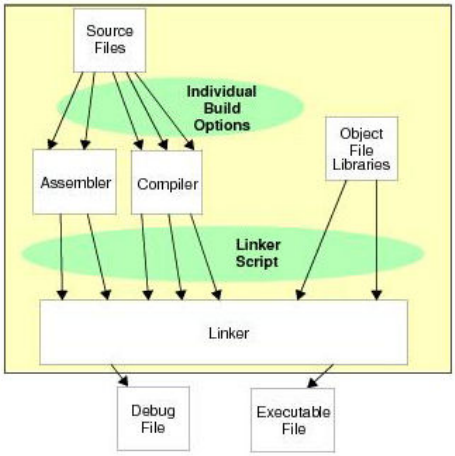
1.3 项目管理器

项目管理器管理要编辑的文件和其他相关文件，以便将这些文件送至语言工具进行汇编或编译，并最终送至链接器。

链接器的任务是将来自汇编器、编译器和库的目标代码片段存放到嵌入式控制器的恰当存储区，并确保各个模块之间可以相互作用（或“链接在一起”）。

从汇编、编译到链接的整个操作过程被称为项目“build”（编译）。根据需要，对于每个文件，对为语言工具所指定属性的调用可以不同，并且编译过程将所有的语言工具操作集成到一起。

图 1-13. MPLAB® X IDE 项目管理器



源文件是遵从汇编器或编译器规则编写的文本文件。汇编器和编译器将源文件转换为中间机器码模块和占位符，以作为函数和数据存储的参考。

链接器解析这些占位符，并将所有模块合并为一个可执行的机器码文件。链接器还会生成一个调试文件，允许 MPLAB X IDE 将正在执行的机器码与源文件相关联。

文本编辑器用于编写代码。它可以识别文本中的语法结构，并采用彩色编码来区分各种元素，如指令助记符、C 语言结构和注释。编辑器支持编写源代码常用的操作。编写好代码之后，编辑器可以配合其他工具，显示调试器中代码的执行。可以在编辑器中设置断点（停止或“暂停”代码执行），而且将鼠标指针悬停在变量名上还可以查看变量的值。可以将变量的名称从源代码文本窗口拖放到 Watches（观察）窗口中，然后可以在其中观察它们在每个断点之后或在代码执行过程中的变化值。

1.4 语言工具

语言工具就是诸如交叉汇编器和交叉编译器的程序。大多数人都比较熟悉在计算机上运行的语言工具，比如 Visual Basic 或 C 编译器。

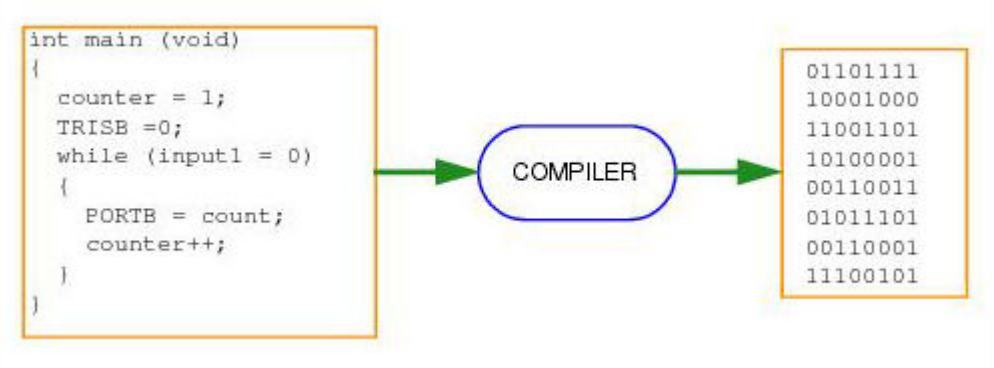
当使用嵌入式系统的语言工具时，就会使用“交叉汇编器”或“交叉编译器”。这些工具与常见编译器的区别在于，它们在计算机上运行，但生成的代码却在其他微处理器（或单片机）上运行。

语言工具还会生成调试文件，MPLAB X IDE 使用这个文件将机器指令和存储单元与源代码相关联。这种集成让 MPLAB X IDE 编辑器能够设置断点，允许在 Watches 窗口中查看变量的内容，并允许您单步调试源代码，观察应用程序的执行。

嵌入式系统语言工具与在计算机上运行和执行的编译器还有一点不同，那就是对存储空间非常敏感。生成的代码量越小越好，因为这样可以尽可能地减少目标对存储器的占用，从而降低成本。这就意味着需要特定于机器的知识来优化和增强代码的技术。

对于计算机，复杂程度适中的程序通常就会大到数兆字节。而简单的嵌入式系统程序则可以小至几千字节，甚至更小。中等规模的嵌入式系统可能需要 32K 或 64K 的代码，以实现相对复杂的功能。某些嵌入式系统会使用数兆字节的空间，以存储大型表、用户文本消息或数据日志。

图 1-14. 编译器将源代码转换为机器指令



1.5 目标调试

在集成开发环境（Integrated Development Environment，IDE）中，代码的执行是在调试器中测试的。调试器可以是软件程序，用来模拟单片机的操作以便进行测试；也可以是硬件工具，用来在应用中执行程序从而对其进行分析。

IDE 和调试

在项目设计周期的末期，调试通常会变得紧迫。随着交付期的临近，让应用程序按最初设计运行是部署产品之前的最后一步，这通常是产品交付延期的最大因素。这就是集成开发环境最重要之处。调试和修改代码、重新编译、下载和测试——所有都需要时间。在一个环境中使用所有工具可以减少该“周期”的时间。能否在最后这几个步骤中找出关键缺陷对嵌入式系统设计人员是一种考验。使用正确的工具可以节省时间。使用 MPLAB X IDE 可以选择多种工具，不过这些工具的界面都是类似的，因此从软件模拟器到低成本的在线调试器、再到强大的在线仿真器的学习过程也会比较简单。

软件调试器

MPLAB X IDE 中内置了软件模拟器，因此不需要任何其他硬件就可以对程序进行测试。软件模拟器是一种软件调试器，软件模拟器的调试功能与硬件调试器的功能几乎完全相同，学习起来非常容易。由于软件模拟器使用计算机中的 CPU 来模拟单片机的操作，因此它通常比实际单片机运行得慢一些。

硬件调试器

在 MPLAB X IDE 中，您可以使用两种类型的硬件：编程器和硬件调试器。编程器直接将机器码从 PC 烧写到目标单片机的内部存储器中。然后就可以将单片机插入到应用中了，并希望单片机中的程序能够按设计运行。

但是，代码通常都不会完全按照预期运行，工程师需要检查代码及其在应用中的运行状况，以决定如何修改原始源代码，使之能够按预期运行。这个过程称为调试。如前所述，软件模拟器可以用于测试代码的运行，但是一旦向单片机烧写了固件，软件模拟器之外的很多因素就发挥作用了。仅使用编程器，虽然可以更改代码，并将其重新烧写到单片机，然后插入目标板进行重新测试，但如果代码比较复杂，这个过程就非常费时费力，而且很难弄清楚硬件中究竟发生了什么问题。

在这种情况下，硬件调试器就非常有用了。硬件调试器可以是在线仿真器或在线调试器，使用具有特殊内置调试功能的单片机。硬件调试器和软件模拟器一样，允许工程师在代码执行到各个点时检查变量，在硬件与其专用电路交互的同时单步调试程序。

1.6 器件编程

应用程序经过调试并在开发环境中运行后，还需要对其进行测试。可以使用在线仿真器、在线调试器、开发编程器或器件编程器对器件进行编程。MPLAB X IDE 可以设置为编程器功能，从而为器件烧写程序。目标应用程序现在可视为接近完成的状态。样机开发编程器可以快速制作和评估样机。某些应用程序可以在器件焊接到目标 PCB 上之后进行编程。使用在线串行编程（In-Circuit Serial Programming™，ICSP™），可以在生产过程中将固件编程到应用中，从而能够在嵌入式应用开发的末期将更新的版本编程到嵌入式应用中。支持在线调试的器件甚至可以在生产完成后重新插入到在线调试器中，以进行质量检测和下一代固件的开发。

生产编程可以使用生产编程器和 MPLAB IPE（随 MPLAB X IDE 一起安装）实现。

1.7 MPLAB X IDE 的组件

MPLAB X IDE 包括：

- 功能全面的程序文本编辑器，它还可以作为调试器的窗口使用。
- 项目管理器（以 Projects（项目）窗口的形式显示），可提供 IDE 和语言工具之间的集成和通信。
- 一些汇编器/链接器套件，用于为项目所使用器件开发固件。
- 提供断点、单步、Watches 窗口和现代调试器所有功能的调试器引擎。调试器引擎与调试工具（包括软件和硬件）配合工作。
- 适用于所有 PIC 和 dsPIC 器件以及许多 AVR 和 SAM 器件的软件模拟器。该软件模拟器实际上由几个特定于器件的软件模拟器可执行文件组成。MPLAB X IDE 将基于项目所使用器件来决定使用哪一个软件模拟器可执行文件。

可以获取或购买一些可选的组件，与 MPLAB X IDE 配合工作：

编译器语言工具

Microchip 的 MPLAB XC C 编译器可以为 MCU 提供高度集成的优化代码。MPLAB X IDE 项目管理器可以调用这些编译器以及社区提供的编译器和第三方编译器，以编译自动载入到目标调试器中的代码，进行即时测试和校验。

编程环境/框架

MPLAB 代码配置器（MPLAB Code Configurator，MCC）是一款免费的图形编程环境，可生成简单易懂的 C 语言代码，将其无缝插入您的项目中。选择 *Tools>Plugins>Available Plugins*（工具>插件>可用插件）即可将 MCC 插件安装到 MPLAB X IDE 中。

MPLAB® Harmony 是系统服务、设备驱动程序和以可移植外设库为基础的其他库的框架，能够提供灵活、可移植且一致的软件“构件”，可供您开发自己的嵌入式 PIC32 应用程序。MPLAB Harmony 配置器（MPLAB Harmony Configurator，MHC）是一款适合 MPLAB Harmony 的省时硬件配置实用程序，在 *Tools>Plugins>Available Plugins* 下以插件形式提供。

编程器

MPLAB ICD 4 在线调试器等生产编程器能够在生产环境下将代码编程到目标器件中。编程器可以与 MPLAB X IDE 或 MPLAB IPE 配合使用，用于控制代码和数据编程，以及控制对配置位的编程来设定目标单片机或数字信号控制器的各种工作模式。

在线调试器和仿真器

在线调试器和仿真器可用于调试目标器件上的应用程序代码。通过使用部分片上资源，这些工具可以将代码下载到插入到应用中的目标单片机中，并设置断点、单步调试以及监视寄存器和变量。仿真器包含了额外的调试功能，例如跟踪。

插件工具

提供了几个可增加 MPLAB X IDE 的功能的插件。例如，MPLAB Data Visualizer 提供了一种机制以图形方式实时显示应用程序输出。

1.8 MPLAB X IDE 帮助

MPLAB X IDE 是基于 NetBeans 平台构建的。因此，许多 NetBeans 功能现在也成为 MPLAB X IDE 功能。关于 NetBeans 帮助主题的更多信息，请参见：

https://docs.oracle.com/cd/E50453_01/doc.80/e50452/toc.htm

帮助内容

要完全了解 MPLAB X IDE 行为，请参见所有帮助文件。要启动帮助，请选择 *Help>Help Contents*（帮助>帮助内容）。这会将所有帮助文件合并为一个，所以可能需要多一点的时间才会打开。

工具帮助内容

此外，您还可以在 *Help>Tool Help Contents*（帮助>工具帮助内容）下查看选定工具的独立帮助文件。启动独立帮助文件的速度会较快，并且提供的主题搜索范围会较小。

开发人员帮助

Microchip Developer Help（开发人员帮助）可供您查找关于高级功能的提示：

http://microchipdeveloper.com/

在 Developer Help 中的“How do I?”（如何实现？）旁的文本框中输入文本以查找主题。

图 1-15. Microchip Developer Help

1.9 其他 MPLAB X IDE 文档

除了帮助之外，Start Page（起始页）上还给出了其他文档、视频、论坛和 Wiki 的链接。

图 1-16. MPLAB X IDE 起始页

1.10 Microchip 网站

Microchip Technology 通过我公司网站提供开发工具的在线支持：

http://www.microchip.com/development-tools/

1.11 Microchip 商店

自 MPLAB X IDE v3.xx 起，可通过 IDE 访问 Microchip 商店购买开发工具软件和硬件产品。

	商店图标

- MPLAB X Store（MPLAB X 商店）选项卡——该选项卡位于 Start（开始）选项卡旁，用于提供产品的链接。
- Help>MPLAB X Store（帮助>MPLAB X 商店）——Help 菜单下用于打开 MPLAB X Store 选项卡的链接。
- MPLAB X Store 工具栏图标——MPLAB X Store 工具栏上用于打开 MPLAB X Store 选项卡的图标。

1.12 编程中心

使用 Microchip Programming Center（Microchip 编程中心），可向您的器件添加代码或订购预编程的免费样片。您可以单击相应图标转至 Microchip 直销网站，以订购此项增值服务。

	编程中心图标

1.13 MPLAB X IDE 更新

MPLAB X IDE 是一个不断改进的程序，具有成千上万的用户。Microchip Technology 在不断地设计具有新功能的新型单片机。许多新的 MPLAB X IDE 功能都源自客户的需求和内部使用。不断开发的新设计和发布的新型单片机促使 MPLAB X IDE 将不断改进。

MPLAB X IDE 计划大约每几个月更新一次版本，以便增加新的器件支持和新的功能。

对于在 MPLAB X IDE 新版本发布时正处于开发中途的项目，“最佳做法”是不要更新为新版本，除非有令人信服的理由需要这样做，例如，新版本包含针对阻碍当前工作的缺陷进行的缺陷修复。新项目开始时是更新为新版本的最佳时机。

MPLAB X IDE 软件的每一次新的发布都实现了新的功能，因此印刷文档的内容肯定会比在线帮助滞后。在线帮助是解决有关 MPLAB X IDE 中任何问题的最好参考。

要获得关于 MPLAB X IDE 及其组件的更新通知，请在 myMICROCHIP Personalized Notification Service（myMICROCHIP 个性化通知服务）的 Development Tools（开发工具）部分进行订阅，网址如下：

http://www.microchip.com/pcn

1.14 在 IDE 之外工作

MPLAB X IDE 旨在帮助您编写、调试和发布用于嵌入式系统的应用程序。但是，您所在的公司可能存在一些要求，使您需要在 IDE 之外进行代码开发。

要了解如何在 MPLAB X IDE 之外编译代码，请参见：

http://microchipdeveloper.com/mplabx:work-outside

2. 使用前须知

当您准备使用 MPLAB X IDE 之前，请执行以下操作：

- 查看自述文件/发行说明中的安装要求
- 安装 MPLAB X IDE（和 JRE）
- 安装 USB 驱动程序（对于硬件工具）
- 连接目标
- 安装语言工具（汇编器和/或编译器）
- 启动 IDE
- 在 Microchip 商店购买相关产品

此外，还需学习如何启动 IDE 的多个实例、IDE 的多个版本，以及如何使用启动参数。

2.1 查看安装要求

“Readme for MPLAB X IDE”中指出了安装要求以及其他安装问题。桌面的 *Learn & Discover>Getting Started>Users Guide & Release Notes*（了解和发现>入门>用户指南和发行说明）下提供了该 HTML 文件的链接。

2.2 安装 JRE 和 MPLAB X IDE

当您安装 MPLAB X IDE（基于 NetBean 平台）时，还会一同安装 Java 运行环境（Java Runtime Environment，JRE）8。该 JRE 仅与 MPLAB X IDE 配合使用，而非在计算机上注册为实现一般用途。

对于 Windows® XP，将需要安装 JRE 7。MPLAB X IDE 安装程序将在您的系统中搜索 JRE 7，如果**未**找到，则将推荐下载网站。

Mac OS X 10.8（Mountain Lion）是第一个支持 JRE8 的版本。因此，MPLAB X IDE 需要使用 Mac OS X 10.8 或以上版本。

2.3 安装 USB 设备驱动程序（对于硬件工具）

MPLAB X IDE 安装程序包含一个预安装程序，能够将所支持硬件工具的 USB 设备驱动程序安装到您的系统中。在 MPLAB X IDE 安装完成后，当硬件工具插入计算机时，操作系统（Operating System，OS）应将驱动程序和工具相关联（即，完成驱动程序安装）。不过对于某些 OS 和工具而言，要正确相关联，可能需要额外的步骤。

macOS®或 Linux® OS 的 USB 驱动程序安装

如果您计划在 Mac 或 Linux 计算机上安装 MPLAB X IDE，则无需额外的步骤。

Windows® OS 的 USB 驱动程序安装

如果您计划在使用 Windows 操作系统的个人计算机上安装 MPLAB X IDE，请遵循以下说明。

注： MPLAB IDE v8 的 USB 硬件工具驱动程序与 MPLAB X IDE 的不同。

下表分别列出了受影响和不受影响的工具。

说明适用于以下工具	您不需要为以下工具执行任何操作
MPLAB REAL ICE™在线仿真器	PICkit™ 2 或 PICkit 3 在线调试器
MPLAB ICD 3、MPLAB ICD 4 或 MPLAB PICkit 4 在线调试器	其他 MPLAB 入门工具包
MPLAB PM3 器件编程器	
PIC32 入门工具包	

按照以下章节的说明来确定您的安装方法。

2.3.1 安装 MPLAB X IDE 之前

请注意，如果您的 WinUSB 驱动程序的 Windows OS 版本低于 IDE 预安装程序的版本，则系统驱动程序将被替换。如果希望保留您的 WinUSB 驱动程序版本，请在安装 MPLAB X IDE 之前重命名这些文件。

WinUSB 驱动程序文件位于以下位置：

32 位操作系统

`C:\Windows\system32\WinUSB.dll`（**32 位**）

`C:\Windows\system32\drivers\WinUSB.sys`（**64 位**）

64 位操作系统

`C:\Windows\SysWOW64\WinUSB.dll`（**32 位**）

`C:\Windows\system32\WinUSB.dll`（**64 位**）

`C:\Windows\system32\drivers\WinUSB.sys`（**64 位**）

2.3.2 如果系统上未安装 MPLAB IDE v8.xx

不需要执行任何操作；在安装 MPLAB X IDE 时，将预装 USB 驱动程序。在您将工具插入计算机 USB 端口时，将显示“New Hardware Found”（发现新硬件）通知。然后，安装将自动继续，或者您必须按照向导执行操作并选择“Automatically select driver”（自动选择驱动程序）。但是，如果该过程未成功安装驱动程序（将预装驱动程序与您的工具相关联），则需要手动安装/关联驱动程序。关于说明和驱动程序位置，请参见 2.3.3.4 如果需要手动切换驱动程序。

2.3.3 如果系统上已安装 MPLAB IDE v8.xx

如果计算机上已安装 MPLAB IDE v8.xx 或以下版本，则可能需要运行 MPLAB Device Driver Switcher，从 MPLAB IDE v8 驱动程序切换到 MPLAB X IDE 驱动程序。

要确定哪些驱动程序与您的工具相关联，应按以下步骤操作：

1. 将所需工具插入个人计算机上的 USB 连接器。
2. 打开计算机的“设备管理器”窗口。
   2.1. 对于 MPLAB IDE v8.xx，驱动程序应位于“Custom USB Device>Microchip Custom USB Device”（定制 USB 设备>Microchip 定制 USB 设备）下。
   2.2. 对于 MPLAB X IDE，驱动程序应位于“Microchip Tools>Microchip WinUSB Device”（Microchip 工具>Microchip WinUSB 设备）下。
3. 如果具有 MPLAB IDE v8.xx 驱动程序，则将需要使用 Switcher 切换到 MPLAB X IDE 驱动程序。

要切换到 MPLAB X IDE 预装驱动程序，应按以下步骤操作：

1. 如果具有 Windows XP 64，请手动切换
2. 如果具有 Windows 7 或 Windows 8，请使用管理员模式
3. 切换 USB 设备驱动程序（对于硬件工具）

如果 Switcher 未成功切换驱动程序，请参见：

1. 如果需要手动切换驱动程序
2. 工具通信问题

2.3.3.1 如果具有 Windows XP 64，请手动切换

注： 不再支持 Microsoft Windows XP Professional SP3。但 MPLAB X IDE 可继续在该操作系统上运行。

如果您的计算机上当前使用的是 Windows XP 64 位操作系统，则将无法使用 MPLAB Device Driver Switcher 从 MPLAB IDE v8 驱动程序切换到 MPLAB X IDE 驱动程序；必须手动切换驱动程序。请参见 2.3.3.4 如果需要手动切换驱动程序。

2.3.3.2 如果具有 Windows 7 或 Windows 8，请使用管理员模式

要运行 MPLAB Device Driver Switcher，必须处于管理员模式。

要以管理员身份运行 Switcher 实用程序，应右键单击可执行文件并选择“以管理员身份运行”。下面给出了可执行文件的位置：

- 32 位操作系统：`C:\Program Files\Microchip\MPLABX\vx.xx\Switcher\32Bit\MPDDSwitch.exe`
- 64 位操作系统：`C:\Program Files (x86)\Microchip\MPLABX\vx.xx\Switcher\64Bit\MPDDSwitch64.exe`

建议首先使用 Switcher GUI 实用程序来切换驱动程序。如果该操作有问题，可以使用命令行应用程序切换驱动程序。

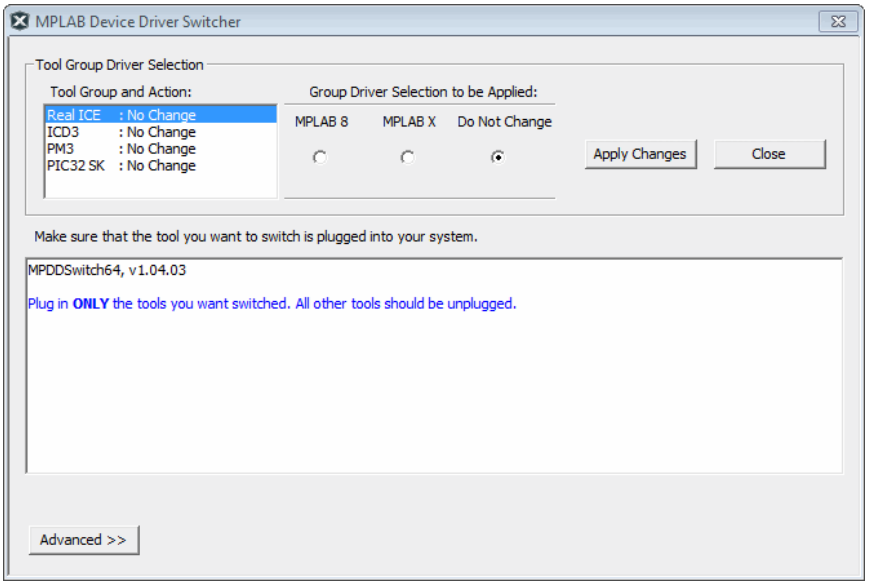
要以管理员身份运行命令行应用程序（`mchpdds32.exe` 或 `mchpdds64.exe`），应先以管理员身份打开命令提示符：*开始*>*所有程序*>*附件*>*命令提示符*，右键单击并选择“以管理员身份运行”。该操作将打开“管理员：命令提示符”。之后，可遵循 `ReadMe32.txt` 或 `ReadMe64.txt` 文件中提供的说明来完成驱动程序切换。

2.3.3.3 切换 USB 设备驱动程序（对于硬件工具）

MPLAB Device Driver Switcher 是一款 GUI 应用程序，用于在 MPLAB IDE v8 设备驱动程序和 MPLAB X IDE 设备驱动程序之间进行切换。单击桌面图标可启动 Switcher 实用程序。

图 2-1. MPLAB Driver Switcher 图标

图 2-2. Switcher 实用程序



Switcher 操作

要切换 USB 设备驱动程序：

1. 在“Tool Group and Action”（工具组和操作）下单击来选择要为之切换驱动程序的已连接工具。
   注： 如果在 Switcher 运行时工具未连接，将不会为相应工具安装或切换驱动程序。
2. 单击“MPLAB 8”或“MPLAB X”的单选按钮。
3. 单击 **Apply Changes**（应用更改）按钮。大文本窗口中将显示切换进度。这可能需要一些时间。
4. 当程序/批处理完成时，在“设备管理器”窗口中查看驱动程序的名称。对于 MPLAB X IDE，驱动程序应位于“Microchip Tools>Microchip WinUSB Device”下。对于 MPLAB IDE v8.xx，驱动程序应位于“Custom USB Device>Microchip Custom USB Device”下。

Switcher 疑难解答

1. 如果 GUI 无法切换驱动程序，则通过单击 **Advanced**（高级）按钮检查驱动程序文件的路径。再次运行 Switcher。
2. 如果 GUI 仍然无法切换驱动程序，则需要手动切换驱动程序。关于说明和驱动程序位置，请参见 2.3.3.4 如果需要手动切换驱动程序。

Switcher 可执行文件位置

可以在 MPLAB X IDE 安装文件夹的 Switcher 子文件夹中找到 Switcher 可执行文件，默认情况下保存在以下位置：

- 32 位操作系统：`C:\Program Files\Microchip\MPLABX\vx.xx\Switcher`
- 64 位操作系统：`C:\Program Files (x86)\Microchip\MPLABX\vx.xx\Switcher`

其中，`vx.xx` 表示 MPLAB X IDE 版本。

2.3.3.4 如果需要手动切换驱动程序

如果需要手动切换或关联 USB 设备驱动程序：

1. 打开“设备管理器”（在“控制面板”下）。在“Microchip Tools”下查找您的工具，或者，如果无法在其中找到它，则在“其他设备”下查找“未知设备”。
2. 右键单击工具名称或“未知设备”，然后选择“更新驱动程序软件”。
3. 在“更新驱动程序软件”对话框中，选择“浏览我的计算机以查找驱动程序软件”。
   注： 不要选择“自动搜索更新的驱动程序软件”。这会将 Windows 默认驱动程序与您的工具相关联，进而可能导致无法正常工作。如果无意中选择了该项，则后退或退出，并重复这些步骤来切换到正确的驱动程序。
4. 查找对应于系统的正确设备驱动程序。
   对于 MPLAB X IDE，设备驱动程序的默认位置为：

   ```
   C:\Program Files\Microchip\MPLABX\vx.xx\Switcher\32Bit\
   winusb\x86\MCHPWinUSBDevice.inf
   ```

   或

   ```
   C:\Program Files (x86)\Microchip\MPLABX\vx.xx\Switcher\64Bit\
   winusb\amd64\MCHPWinUSBDevice.inf
   ```

   其中，`vx.xx` 表示 MPLAB X IDE 的版本号。
5. 如果弹出“Windows 安全”对话框，请选择“安装此驱动程序软件”继续切换到所选驱动程序。

2.3.3.5 工具通信问题

1. 如果使用的是扩展坞或集线器，并在插入工具之后发生问题，则可能需要将该工具直接插入计算机上的 USB 端口。
2. 如果需要手动切换到设备驱动程序，则需要指向 `32Bit` 或 `64Bit` 文件夹中的 `INF` 文件。有关详细信息，请参见 2.3.3.4 如果需要手动切换驱动程序。

2.4 连接目标（对于硬件工具）

对于在线调试器和仿真器，请参见以下资料以确定如何将硬件工具与目标连接：

- 开发工具设计建议
- 调试头规范（如果使用调试头）
- 工具文档

对于专用编程器，请参见工具文档来了解连接信息。

如果使用 Microchip 演示板、评估工具包或参考设计作为目标板，请参见随附文档来了解设置信息。

2.5 安装语言工具

安装 MPLAB X IDE 时，也会安装以下语言工具：MPASM 工具链。

此外，以下 C 编译器工具链（编译器、汇编器和链接器等）也可用于 MPLAB X IDE。要选择编译器工具链，请考虑好希望使用的器件，然后选择支持该器件的工具链。

- MPLAB XC C 编译器
- AVR 和 Arm® C 编译器
- AVRASM2——随 Atmel Studio 一起安装

MPLAB XC 编译器提供免费版或专业版（功能全面，进行代码优化）编译器许可证。要安装这些编译器并获取许可证，请查看文档《安装 MPLAB® XC C 编译器并获取许可证》（DS50002059J_CN）。MPLAB X IDE 包含了为安装的编译器获取许可证以及检入和检出网络许可证的选项。请参见 *Tools>Licenses*（工具>许可证）。

AVR 和 Arm GNU C 编译器（GNU C Compiler，GCC）始终免费。按照安装程序画面在计算机上进行安装。

注： 确保将这些编译器安装在 MPLAB X IDE 能搜索到的位置，例如，将 MPLAB XC C 编译器安装在 `C:\Program Files\Microchip` 下。否则，必须在 MPLAB X IDE 中指定查找位置（见 4.7 设置语言工具位置）。

2.6 启动 IDE 并查看桌面

双击 MPLAB X IDE 图标来启动程序。

图 2-3. MPLAB X IDE 图标

MPLAB X IDE 是基于 NetBeans 平台构建的。如果您熟悉 NetBeans IDE，则 MPLAB X IDE 桌面看起来会很熟悉。但是，下列选项卡是 MPLAB X IDE 特有的。

- Start Page
- MPLAB X Store

在 Start Page 中，存在 3 个含有链接的选项卡。如果不希望在启动时显示 Start Page，请取消选中任何选项卡下的“Show on Startup”（在启动时显示）。

图 2-4. MPLAB X IDE 桌面

2.7 在 MPLAB X Store 上选购工具

单击 MPLAB X Store 选项卡可从 Microchip 商店购买工具。该选项卡用于展示与 MPLAB X IDE 有关的一些最新在售产品。还可以浏览类别以查找其他产品。

可直接从 Help 菜单或工具栏图标访问此选项卡。

图 2-5. MPLAB X Store

更多功能即将上线。计划将支持从 Start Page 的 My MPLAB IDE（我的 MPLAB IDE）登录 Microchip 直销网站。之后，您将能够：

- 查看软件相关警报（例如 HPA 到期）
- 访问您的 mySoftware 帐户，可在其中注册和下载许可证以及续订 HPA

2.8 启动 IDE 的多个实例

在将硬件工具（MPLAB PICkit 4 等）与 MPLAB X IDE 的实例配合使用之前，需要进行一些设置。完成任何硬件工具设置之后，可以从 IDE 实例自己的目录中调用 IDE 实例。

2.8.1 设置硬件工具使之与多个实例配合工作

默认情况下，最多可以使用 5 个 IDE 实例。如果希望使用更多实例，则需要手动修改“mchpdefport”文件。

“mchpdefport”文件的使用和格式

“mchpdefport”文件为 IDE 和底层 USB 库（DLL、so 或 dylib 文件）提供工具热插拔使用所需的信息。该文件的格式如下：

```
localhost
30000
30002
30004
30006
30008
```

第一行指示运行 IDE 的主机名。

其他行代表底层库与上层 IDE 进行通信所使用的端口或套接字编号。每个 IDE 实例将分配到一个不同的端口或套接字。IDE 实例之间的所有通信都是对用户隐藏的。

最多使用 5 个 MPLAB X IDE 实例时，不需要更改该文件。如果需要 5 个以上的实例，可以通过编辑该文件来添加更多的端口或套接字编号。

“mchpdefport”文件的位置

在 IDE 默认安装中，“mchpdefport”文件位于以下位置，具体取决于操作系统：

操作系统	位置

Windows（64 位）	`C:\Windows\system32`
	和
	`C:\Windows\SysWOW64`
	注： 两处的“mchpdefport”文件都必须修改。
	注： 如果在 64 位系统上使用 32 位编辑器来编辑 `C:\Windows\system32` 中的“mchpdefport”，则实际上修改的是 `SysWOW64` 中的“mchpdefport”文件。必须使用 64 位编辑器才能编辑 `system32` 中的文件。

Windows（32 位）	`C:\Windows\system32`
Linux	`/etc/.mplab_ide`
Mac（OS X）	`/etc/.mplab_ide`

2.8.2 调用 IDE 的实例

MPLAB X IDE 要求每个实例都具有自己的用户目录。因此，在一个实例中设置的首选项或添加的插件将不会反映到另一个实例中。

要调用多个实例，需要使用 `--userdir` 选项指定一个目录来启动 IDE。

Window 操作系统

创建带有 `--userdir` 选项的快捷方式。例如，在 Windows 7 操作系统上：

1. 右键单击桌面，然后选择 ***New>Shortcut***（新建>快捷方式）。
2. 浏览至已安装的（默认）MPLAB X IDE 可执行文件：`C:\Program Files (x86)\Microchip\MPLABX \vx.xx\mplab_platform\bin\mplab_ide.exe`，其中 `vx.xx` 代表 MPLAB X IDE 版本。

3. 在该行的末尾，输入用户目录的位置。可以将目录置于此类 MPLAB X IDE 信息的默认位置下：`--userdir "C:\Users\MyFiles\AppData\Roaming\.mplab_ide\dev\anydir"`，也可将其置于任何所需位置：`--userdir anydir`
4. 单击 OK（确定）。

Linux 操作系统

不带任何参数启动已安装版本（在桌面图标上单击）将使用`$(HOME)/.mplab_ide` 用户目录运行。要更改用户目录，请运行`$InstallationDir/mplab_platform/bin/mplab_ide` shell 脚本，向其传递参数`--userid anydir`。例如，要运行 MPLAB X IDE 的两个不同实例：

```
$ /opt/microchip/mplabx/vx.xx/mplab_platform/bin/mplab_ide --userdir ~/.anydir1 &
$ /opt/microchip/mplabx/vx.xx/mplab_platform/bin/mplab_ide --userdir ~/.anydir2 &
```

其中，`vx.xx` 表示 MPLAB X IDE 版本。

此外，也可以创建嵌入用户 ID 的桌面图标。

Mac 操作系统

打开一个 Shell 窗口，并输入以下命令行在备用用户目录中执行 MPLAB X IDE 安装。可以将目录置于此类 MPLAB X IDE 信息的默认位置下：

```
$/bin/sh /Applications/microchip/mplabx/vx.xx/mplab_ide.app/Contents/
Resources/mplab_platform/bin/mplab_ide --userdir "${HOME}/Library/
Application Support/mplab_ide/dev/anydir"
```

也可将其置于任何所需位置：

```
$/bin/sh /Applications/microchip/mplabx/vx.xx/mplab_ide.app/Contents/
Resources/mplab_platform/bin/mplab_ide --userdir anydir
```

2.9 启动 IDE 的多个版本

在 MPLAB X IDE v3.00 发布之前，可通过安装到不同的目录在 PC 上安装不同版本的 IDE。自 MPLAB X IDE v3.00 起，默认会执行该操作，例如：

```
C:\Program Files (x86)\Microchip\MPLABX\v3.00
```

可同时启动和运行不同的版本。每个版本都有自己的用户目录。

可以在同一台 PC 上启动一个版本的 MPLAB X IDE 和一个版本的 MPLAB IDE（v8 或更早版本），但请注意“**安装 USB 设备驱动程序（对于硬件工具）**”中提及的 USB 硬件驱动程序限制。

2.10 使用启动参数启动

许多命令行选项来自 NetBeans 平台。请参见：

http://wiki.netbeans.org/FaqStartupParameters

此外，可通过使用`--help` 选项找到 MPLAB®X IDE 选项的列表。例如，在 64 位 Windows® 7 系统中：

```
>C:\Program Files (x86)\Microchip\MPLABX\vx.xx\mplab_platform\bin\mplab_ide --help
```

3. 教程

本教程提供关于处理 MPLAB X IDE 项目的指导示例。

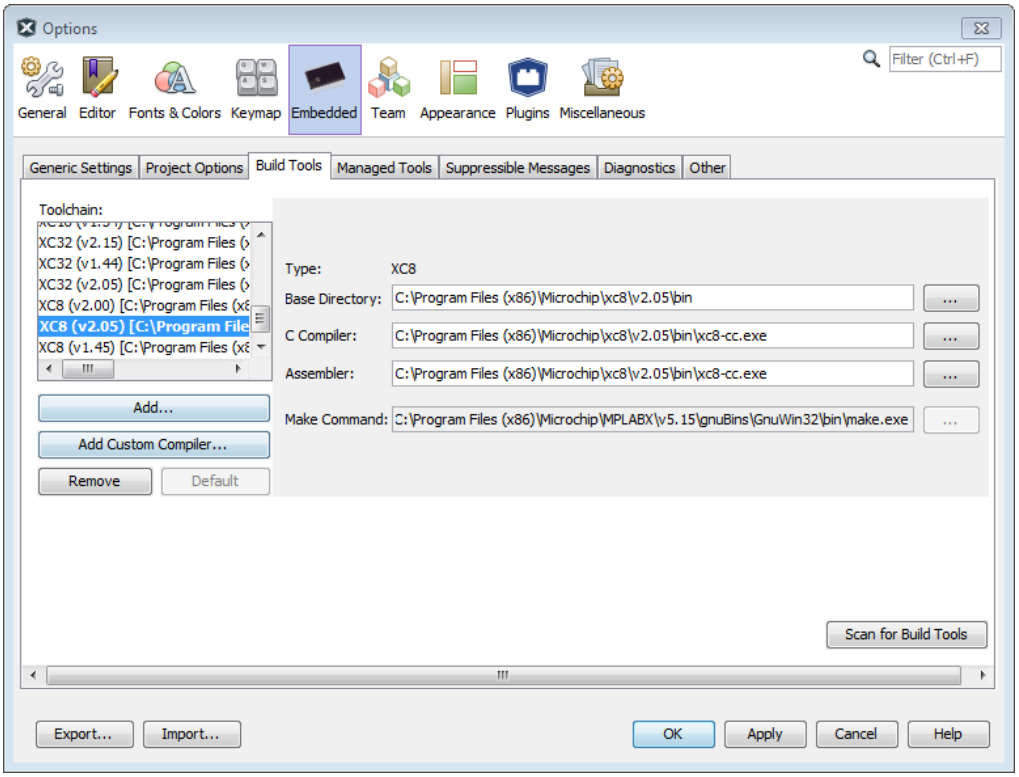
示例项目将在四个 LED 上显示电位器值。说明了在 MPLAB X IDE 中下载并打开项目后，如何设置/更改项目属性以及运行项目。此外，还将演示调试功能，例如设置断点、单步执行代码、查看变量值以及查看/更改寄存器和存储器值。

3.1 安装和设置软件

下载并安装下列软件。关于安装的详细信息，请参见章节 2. 使用前须知。

- 安装 MPLAB X IDE。从以下网址下载免费的 IDE：http://www.microchip.com/mplabx。
- 安装 MPLAB XC8 C 编译器。从以下网址下载免费的 MPLAB XC 编译器：http://www.microchip.com/xc。

启动 MPLAB X IDE。确保 IDE 窗口 *Tools>Options>Embedded>Build Tools*（工具>选项>已安装工具>编译工具）中显示了编译器。如果未显示，请单击 **Add**（添加）按钮手动查找编译器的安装位置。

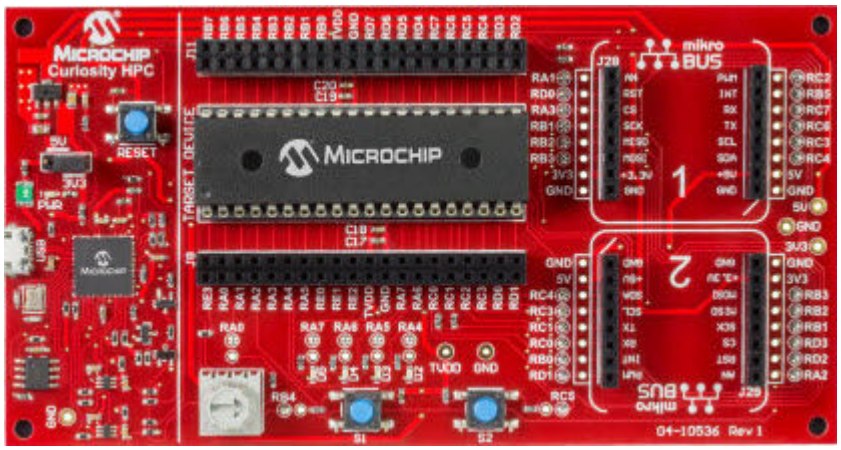


3.2 连接硬件

将使用装有 PIC16F18875 的 Curiosity 高引脚数（High Pin Count，HPC）开发板（DM164136）。更多信息，请访问以下产品页面：

- DM164136
- PIC16F18875

使用支持的 USB 电缆，将电缆一端的 mini USB 连接器插入评估板，然后将另一端的 USB 连接器插入计算机。将自动安装驱动程序。



3.3 下载示例代码

转至 MPLAB Xpress 代码示例。

然后完成以下选择：

- 在 **Tags**（标签）下，选中“#Getting Started”（#入门）复选框。
- 在 **Board**（电路板）下，选中“Curiosity HPC Board”（Curiosity HPC 电路板）复选框。
- 在 **Title**（标题）下，单击“CuriosityHPC-Basics”。

MPLAB Xpress Code Examples

Title	Author	Like	Watch	Import	Tags	Board	Device
Search		From	From	From	#Getting Started	Curiosity Board	Search
		To	To	To	ADC	Curiosity HPC Board	
					ADCC	Curiosity Nano	
CuriosityHPC-Basics		0	0	282	#Getting Started, ADCC,...	Curiosity HPC B...	PIC16F18875

“CuriosityHPC-Basics”页面上将给出电路板和器件相关信息的链接。单击 **Download**（下载）即可下载示例项目。解压后可获取内容。

提示： 下载到 `Users>UserName>MPLABXProjects` 目录。

CuriosityHPC-Basics

Download 85 Open 298

3.4 在 MPLAB X IDE 中打开示例项目

在 MPLAB X IDE 中，单击“Open Project”（打开项目）图标。滚动列表，选择“CuriosityHPC-Basics”项目，然后单击 **Open Project**。

查看 Projects 和 Files 窗口

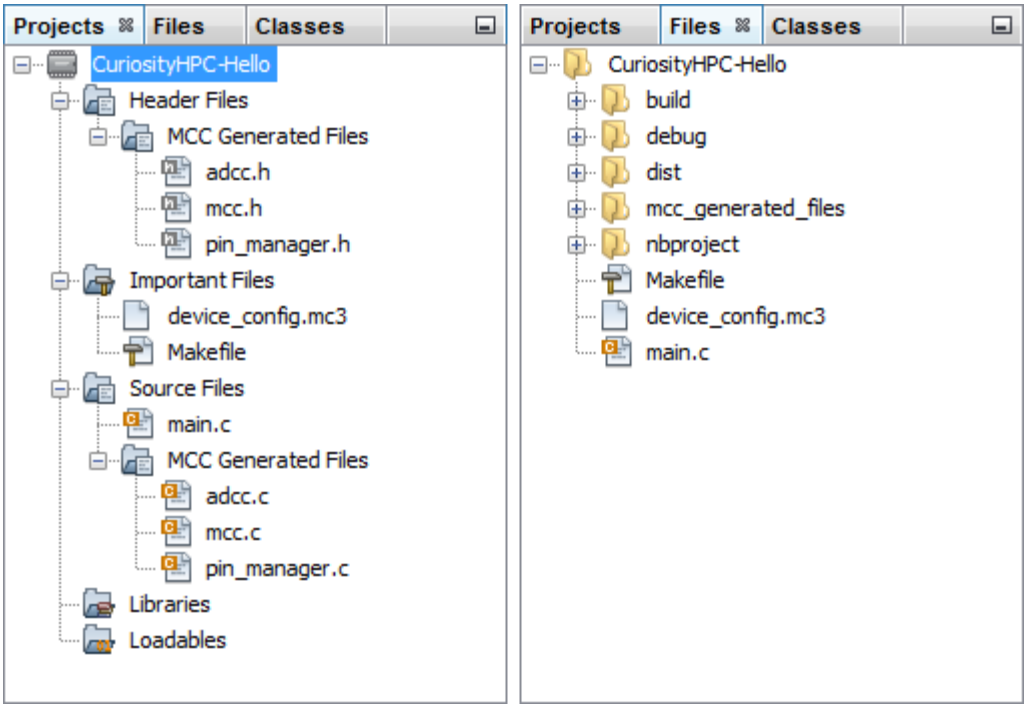
Projects 窗口将显示文件按类别进行分组的项目树。关于这些类别的更多信息，请参见 8.1 Projects 窗口视图。

单击 Files（文件）窗口可查看项目文件的文件管理器视图。请参见 8.2 Files 窗口视图。

如果在上述任一窗口中双击文件名，则会在编辑器窗口中打开相关文件。要关闭该选项卡，请单击文件名旁的“x”。

在 Projects 窗口中右键单击某个项目名称可查看弹出（上下文）菜单。该操作同样适用于项目的子文件夹和窗口背景。

图 3-1. Projects 和 Files 窗口

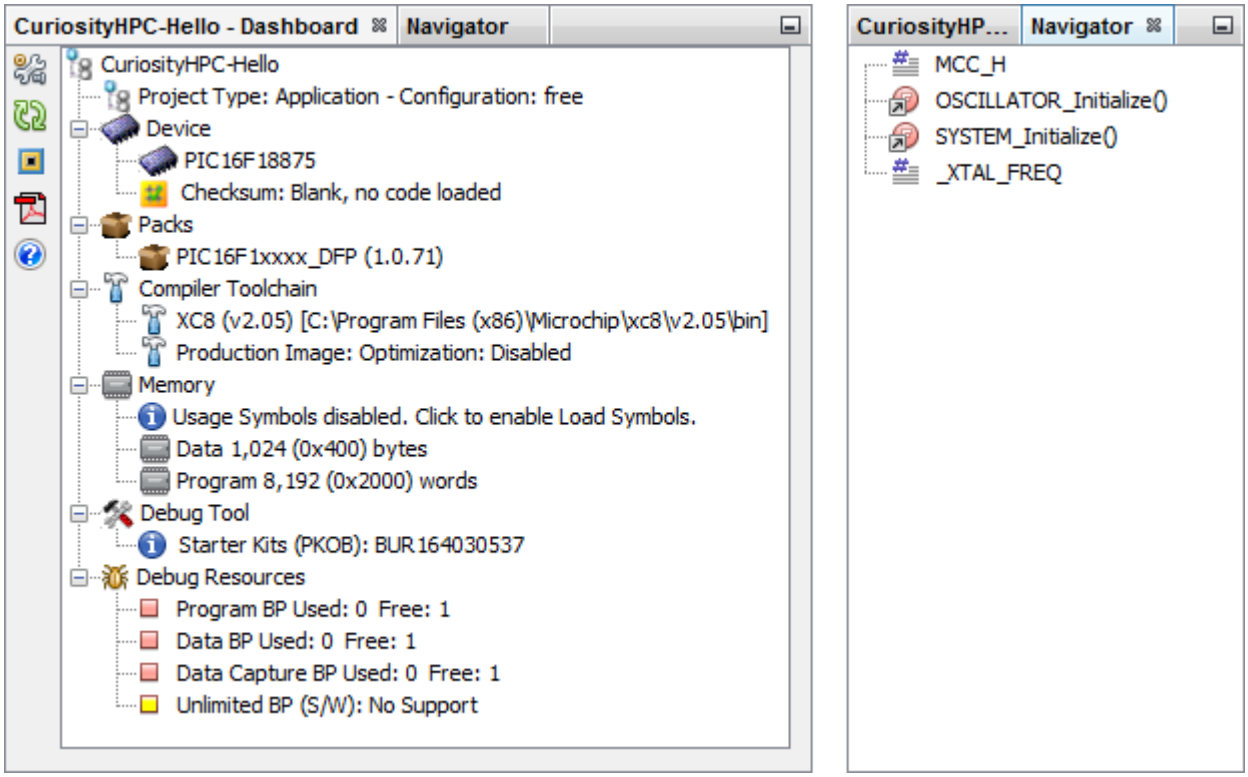


查看 Dashboard（仪表板）和 Navigator（导航器）窗口

Dashboard 窗口显示有关项目的详细信息。更多信息，请参见 12.6 Dashboard 窗口。

Navigator 窗口提供当前选定文件的紧凑视图，方便在文件的不同部分之间进行导航。

图 3-2. 项目打开时的 Dashboard 窗口



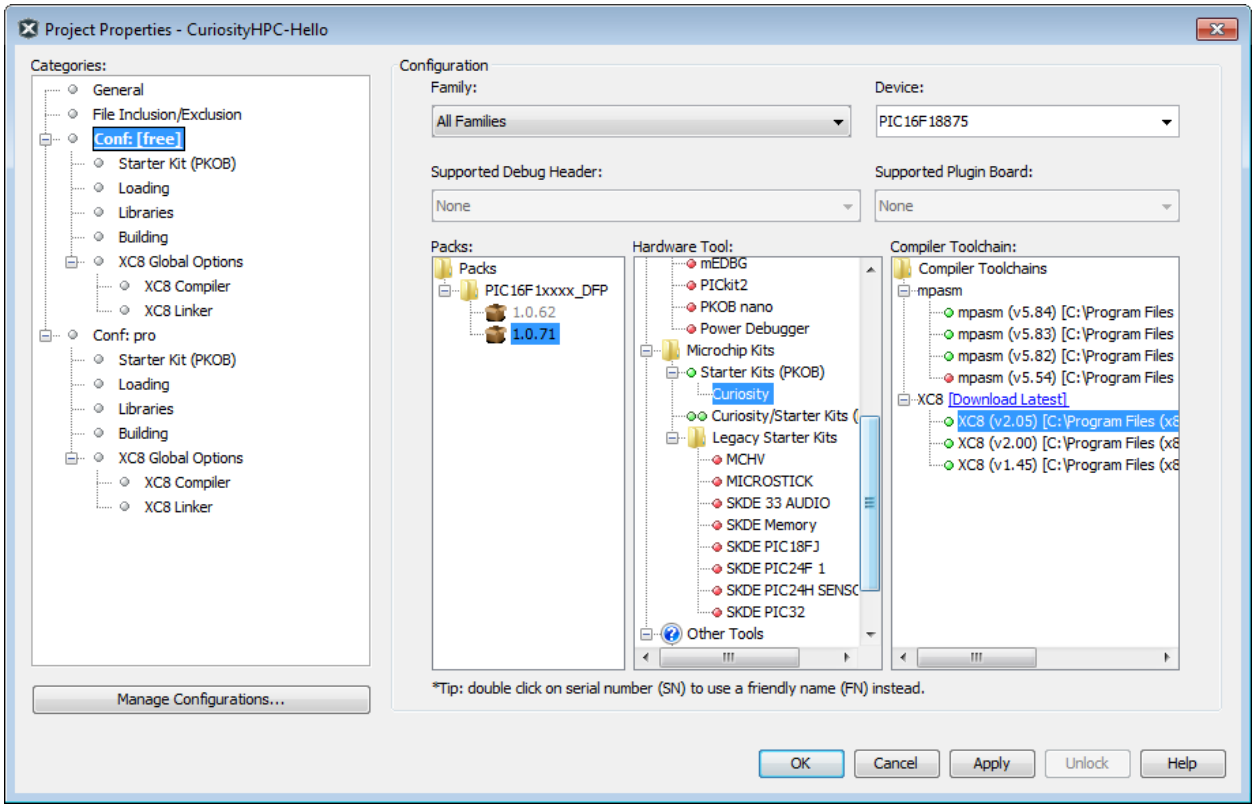
3.5 设置项目属性

在 Projects 窗口中，右键单击项目名称以显示下拉菜单。选择“Properties”（属性）。

查看 Project Properties（项目属性）窗口。单击 **OK** 保存所有更改。

Categories（类别）	项目被设置为使用“免费”版编译器。如果您拥有已获得许可证的版本，则可以选择“pro.”（专业版）。
Device（器件）	确认其与电路板上的器件匹配。
Packs（包）	图中所示的包版本适用于 MPLAB X IDE v5.20。
Hardware（硬件）	应选择 Curiosity 作为带有 PKOB（内置调试器）的入门工具包。
Compiler Toolchain（编译器工具链）	教程项目是使用 MPLAB XC8 v1.37 创建的。可以切换为最新编译器版本（如图所示），也可以从以下位置下载原始项目版本： 下载归档软件

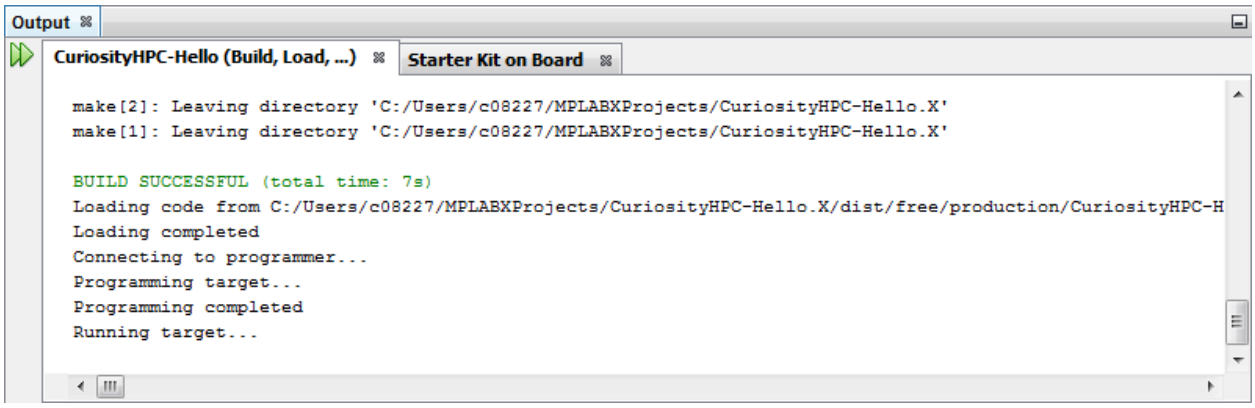
图 3-3. Project Properties 窗口



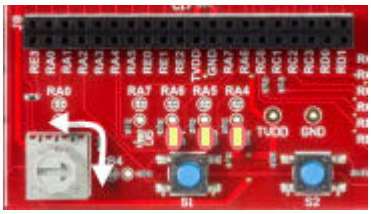
3.6 运行代码

单击 Run Project（运行项目）图标  运行程序。Output（输出）窗口中将会显示运行进度（见下图）。或者，使用“Hold in Reset”（保持复位）按钮  使器件在复位和运行状态之间切换。

图 3-4. Output 窗口——代码运行



旋转电路板上的电位器，以查看表示电位器值的 LED 是否点亮。



3.7 调试代码

本教程中使用的代码已经过测试并成功运行。但是，在您开发应用程序时，可能需要调试自己的代码。

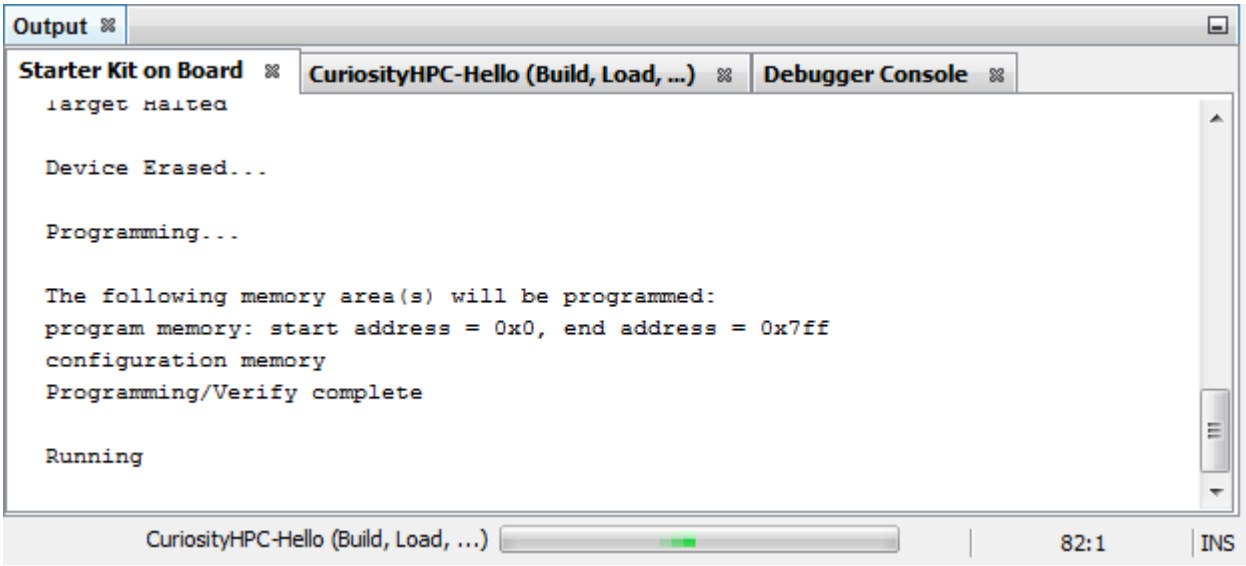
单击“Debug Project”（调试项目）图标  可开始调试会话。Output 窗口中将会显示调试进度（见下图）。

暂停应用程序代码		Pause（暂停）
再次运行代码		Continue（继续）
结束代码执行		Finish Debugger Session（完成调试器会话）

关于调试 C/C++代码项目的更多信息，另请参见 NetBeans 帮助：

调试 C/C++项目教程

图 3-5. Ouptut 窗口——在调试模式下运行的代码



3.8 设置断点

在调试代码时，通过在代码中特定位置暂停执行来检查变量值的功能会很有用。要执行该操作，请使用断点。

关于断点的更多信息，另请参见 NetBeans 帮助：

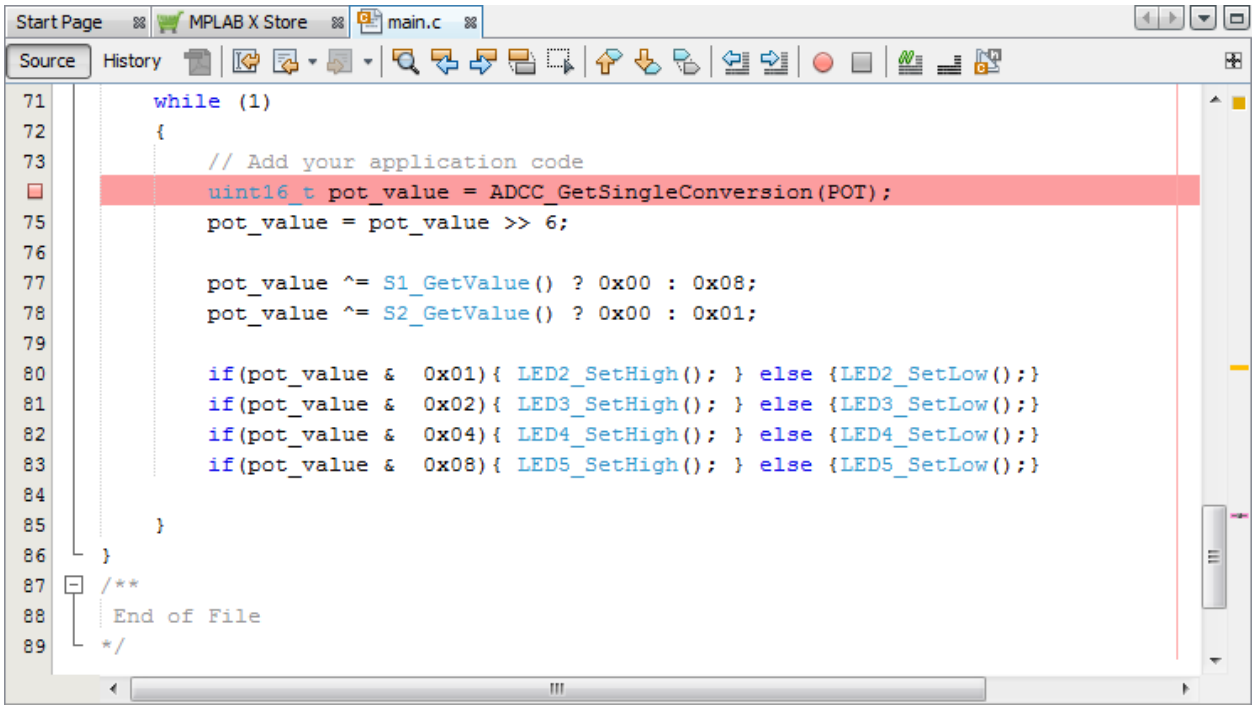
Debugging C/C++ Projects Tutorial>Creating Breakpoints

3.8.1 设置断点

如果代码正在运行，请通过单击“Finish Debugger Session”图标  结束执行。

在 Projects 窗口的“Source Files”（源文件）下，双击文件 `main.c` 在编辑器窗口中将其打开。在下图中，通过单击代码行的左边缘在代码行上设置断点。

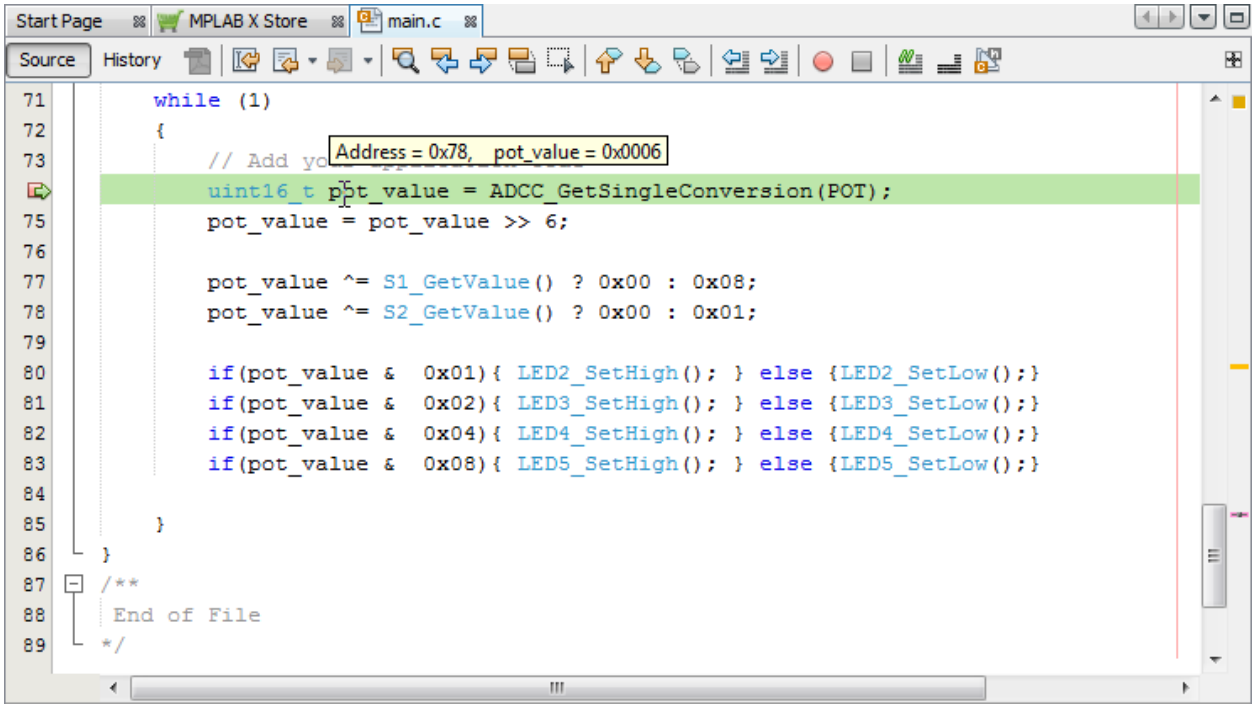
图 3-6. 代码中设置的断点



3.8.2 在断点处暂停

单击“Debug Project”图标  再次执行程序。程序将在断点处暂停。将鼠标悬停在 `pot_value` 变量上以查看其当前值。

图 3-7. 在断点处暂停程序执行



3.8.3 疑难解答

如果在编辑器的代码行旁出现灯泡警告图标，可能需要设置 include 文件的路径。请参见：

设置 Include 目录路径

注： 对于 MPLAB XC8 v2.xx，AVR 和 PIC MCU 均受支持，因此编译器的安装目录结构已更改。

3.9 单步执行代码

单步功能用于在断点处暂停之后单步执行代码。单步功能可以检查变量值的变化或确定程序流是否正确。

Step Over（单步跳过）——执行程序的一行源代码。如果该行是一个函数调用，则执行整个函数，然后停止。

Step Into——执行程序的一行源代码。如果该行是一个函数调用，则程序执行到该函数的第一条语句，然后停止。

Step Out（单步跳出）——执行程序的一行源代码。如果该行是一个函数调用，则执行函数，并将控制返回给调用方。

Run to Cursor（运行至光标位置）——运行当前项目，直到文件中的光标位置处停止程序执行。

图 3-8. 单步执行期间的代码

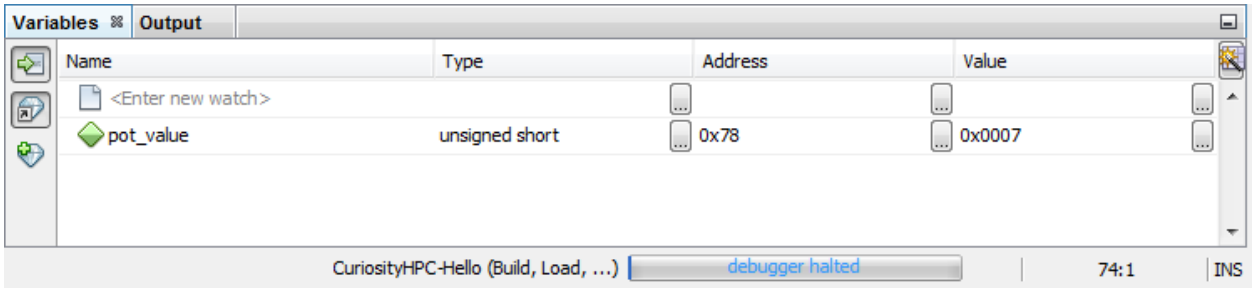
```
71      while (1)
72      {
73          // Add your application code
            uint16_t pot_value = ADCC_GetSingleConversion(POT);
75          pot_value = pot_value >> 6;
76
            pot_value ^= S1_GetValue() ? 0x00 : 0x08;
78          pot_value ^= S2_GetValue() ? 0x00 : 0x01;
79
80          if(pot_value &  0x01){ LED2_SetHigh(); } else {LED2_SetLow();}
81          if(pot_value &  0x02){ LED3_SetHigh(); } else {LED3_SetLow();}
82          if(pot_value &  0x04){ LED4_SetHigh(); } else {LED4_SetLow();}
83          if(pot_value &  0x08){ LED5_SetHigh(); } else {LED5_SetLow();}
84
85      }
86  }
87  /**
88   End of File
89  */
```

3.10 查看变量值

可以在 Variables（变量）窗口中查看变量值的变化。要打开该窗口，请选择 *Window>Debugging>Variables*（窗口>调试>变量）。关于该窗口的更多信息，请参见 4.17 观察局部变量值变化。

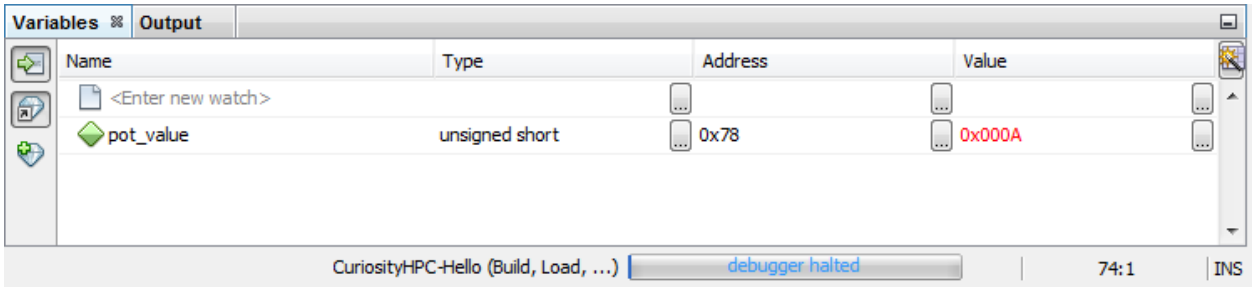
单击“Debug Project”图标  。代码将执行，然后在断点处停止。Variables 窗口中将显示 `pot_value` 信息。

图 3-9. Variables 窗口——在第一个断点暂停



旋转电路板上的电位器，使值发生变化。单击“Continue”图标  。代码将再次执行，然后在断点处停止。查看 Variables 窗口，可发现值已发生变化（红色文本）。

图 3-10. Variables 窗口——在第二个断点暂停



3.11 观察符号值变化

在 Watches 窗口中观察全局符号值或 SFR 值的变化。在程序执行期间确定这些值是否为预期值可以帮助您调试代码。

3.11.1 编辑代码

单击“Finish Debugger Session”图标  。

现在 pot_value 为局部符号。要使其成为全局符号，请在 main() 之前对其进行声明。

```
#include "mcc_generated_files/mcc.h"
uint16_t pot_value;
/*                         Main application
 */
void main(void)
```

然后从第一次使用该符号的行中删除 uint16_t。

```
    while (1)
    {
        // Add your application code
        pot_value = ADCC_GetSingleConversion(POT);
```

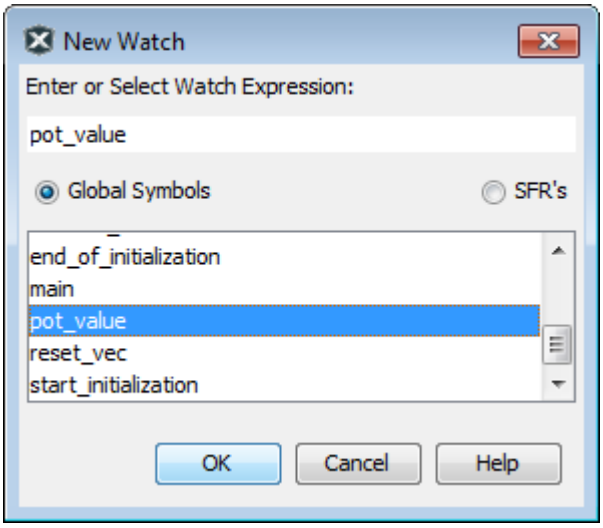
在该行上放置一个断点，然后再次单击“Debug Project”图标  。程序应在该行暂停。

3.11.2 创建新的观察

要创建新的观察：

1. 选择 *Debug>New Watch*（调试>新建观察）。此时将打开 New Watch 对话框。
2. 输入观察表达式（在此例中为 pot_value），然后单击 **OK**。现在，Watches 窗口将显示在桌面上，其中会列出符号。

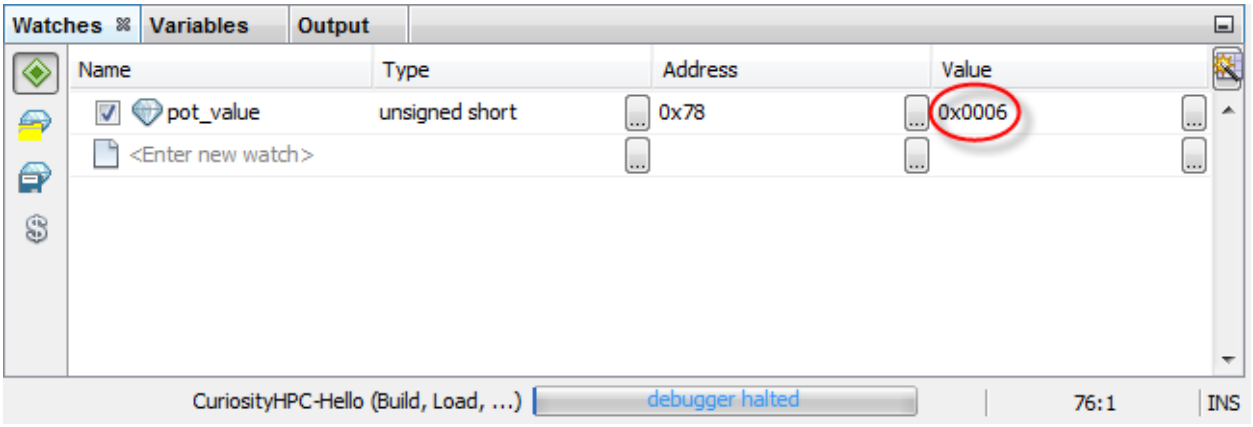
图 3-11. New Watch 符号



3.11.3 在 Watches 窗口中查看值变化

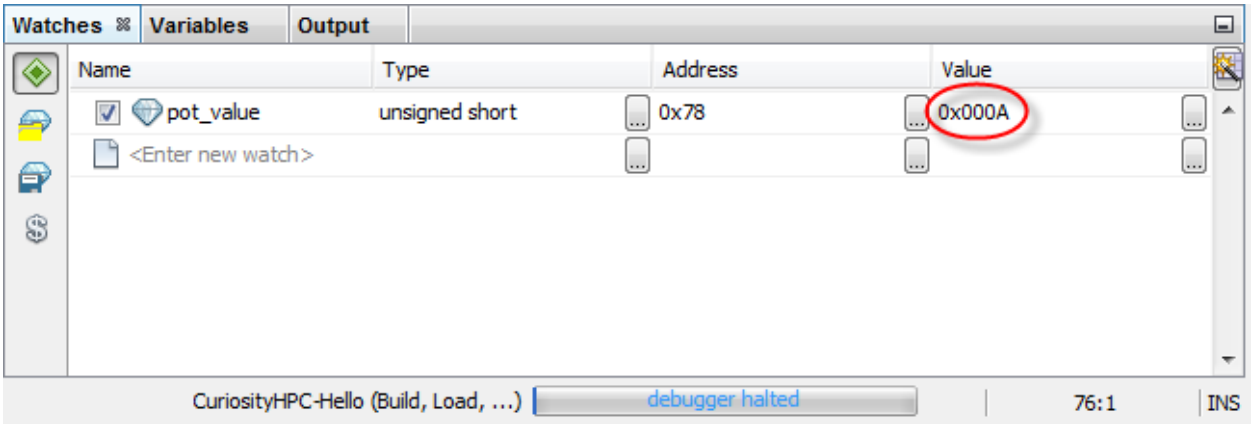
单击“Debug Project”图标  。代码将执行，然后在断点处停止。Watches 窗口中将显示 pot_value 信息。

图 3-12. Watches 窗口——在第一个断点暂停



旋转电路板上的电位器，使值发生变化。单击“Continue”图标  。代码将再次执行，然后在断点处停止。查看 Watches 窗口，可发现值已发生变化。

图 3-13. Watches 窗口——在第二个断点暂停



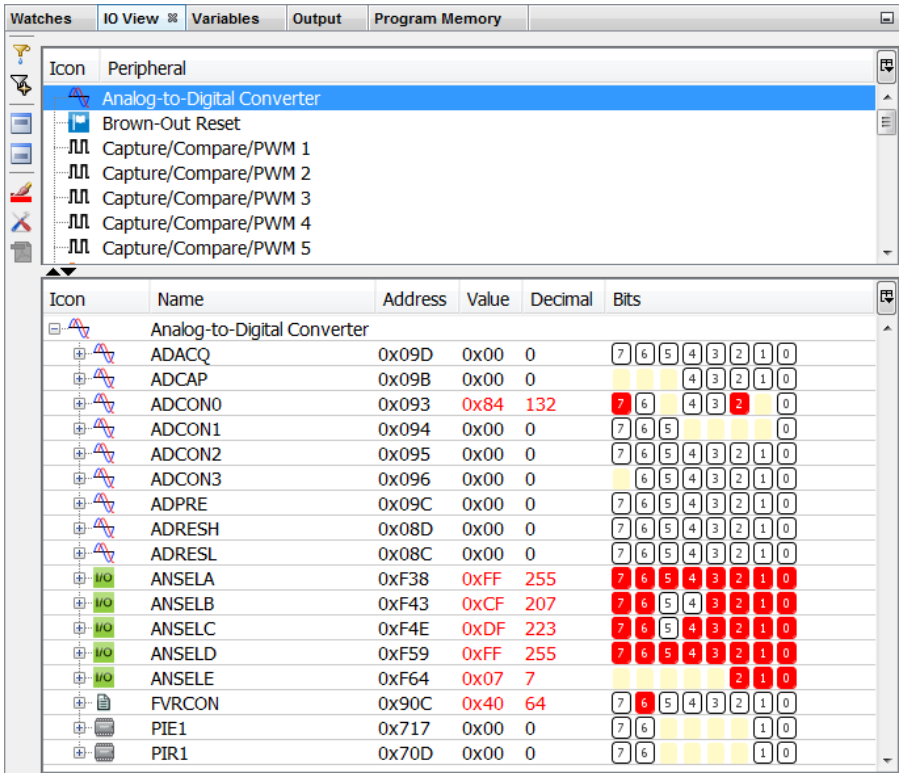
关于该窗口的更多信息，请参见 12.19 Watches 窗口。

3.12 查看 I/O 寄存器

I/O View（I/O 视图）窗口提供了活动项目关联器件的 I/O 存储器映射的图形视图。调试时，该调试工具将显示实际的寄存器内容，从而可以验证外设配置。它也可以用于修改寄存器的内容，而无需重新编译。

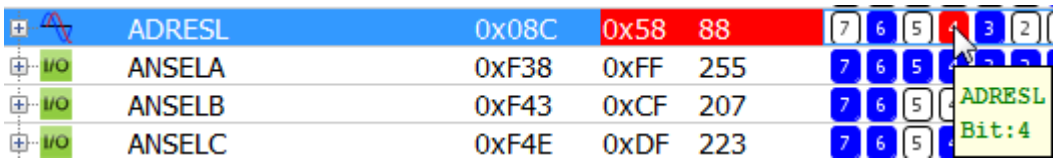
通过选择 *Window>Debugging>IO View*（窗口>调试>IO 视图）来打开 I/O View 窗口。在窗口上部的 Peripheral（外设）部分，单击“Analog-to-Digital Converter”（模数转换器）。与该外设相关的寄存器将显示在下面的 Register（寄存器）部分。

图 3-14. IO View 窗口



由于 ADC 用于将电位器信号转换为用于 LED 显示的数字信号，因此如果旋转电位器，某些值将发生变化。旋转电位器，然后单击“Continue”图标  。

也可以通过单击寄存器值直接对其进行更改。



关于该窗口的更多信息，请参见以下各节：

- 5.20 查看项目的寄存器（I/O View）
- 12.8 IO View 窗口

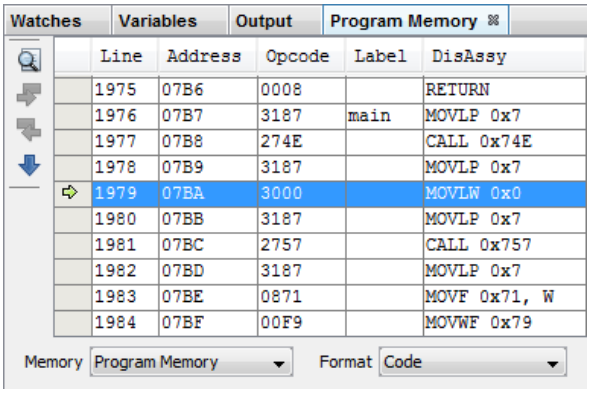
3.13 查看器件存储器（包括配置位）

MPLAB X IDE 具有灵活的、抽象化的存储器窗口，它们提供了不同类型器件存储器的可定制视图。选择 *Window>Target Memory Views*（窗口>目标存储器视图）可查看特定于器件的存储器窗口的列表。

例如，要查看闪存：

1. 选择 *Window>Target Memory Views>Program Memory*（窗口>目标存储器视图>程序存储器）。
2. 此时将打开 Program Memory 窗口，显示上次暂停位置。

图 3-15. 在断点暂停时的存储器窗口内容



要设置存储器窗口选项，右键单击存储器窗口以弹出一个菜单，其中包含各种选项，例如显示选项、填充存储器、表导入/导出以及输出到文件。菜单的内容取决于窗口。请参见 12. MPLAB X IDE 窗口和对话框。

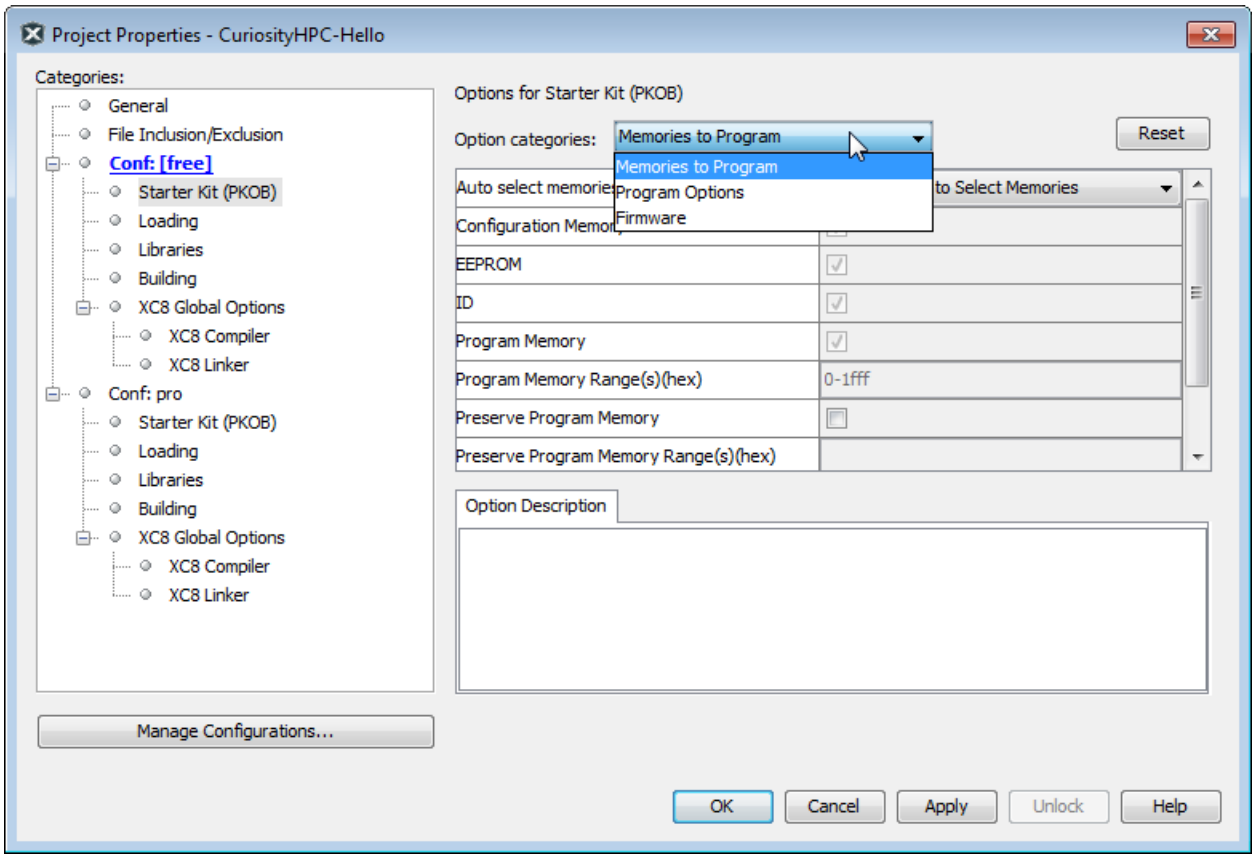
要刷新闪存窗口：

1. 暂停程序（Finish Debugger Session）。
2. 单击“Read Device Memory”（读取器件存储器）图标  。

3.14 对器件编程

调试完代码之后，可以将它编程到目标器件上。

首先，在 Project Properties 窗口中检查编程选项。对于本教程，无需更改任何选项。



最后，通过单击“Make and Program”（Make 并编程）图标  对器件进行编程。

注： 不是所有编程功能都包含在 MPLAB X IDE 中。关于更多编程支持，请参见 MPLAB X IDE 安装随附的 MPLAB IPE。

4. 基本任务

以下步骤说明如何在 MPLAB® X IDE 中处理项目。

表 4-1. 处理基本任务

1 准备工作	1. 开始之前，先安装 MPLAB X IDE，设置所有硬件工具（安装 USB 驱动程序并正确地与目标连接），并安装用于选定器件的语言工具。 2. 然后，启动 MPLAB X IDE 开始本教程。
2 创建并编译项目	1. 通过使用 New Project（新建项目）向导创建新项目。然后，查看桌面窗格中的变化。 2. 打开 Project Properties 窗口，以查看或更改项目属性、设置或更改调试器/编程器工具选项以及设置或更改语言工具选项。 3. 在 Tools Options（工具选项）对话框中设置语言工具位置和设置其他工具选项。 4. 创建新文件以将其添加到项目中或向项目中添加现有文件。在 File（文件）窗口中输入或编辑应用程序代码。 5. 研究编辑器用法的其他功能。 6. 添加、设置库和目标文件。 7. 设置文件和文件夹属性以从编译中保留或排除各个文件或整个文件夹。 8. 为项目设置编译属性。在此，可以选择项目配置类型、编译前和编译后步骤，其中使用校验和作为用户 ID，并在编译时装入替代的十六进制文件。 9. 编译项目。
3 执行代码	1. 运行代码以执行代码。 2. 调试代码以在调试会话中执行代码。
4 调试代码	1. 使用断点控制程序执行。在行内或通过 Breakpoint（断点）窗口设置断点。 2. 在程序执行时单步执行代码。 3. 在 Watches 和 Variables 窗口中观察符号值变化。 4. 查看或更改器件存储器。存储器类型取决于选定的器件。此外，请参见在 Configuration Bits 窗口中设置配置值。
5 对器件编程	1. 使用简单的工具栏按钮对器件编程。

4.1 创建新项目

*项目*包含一组源文件以及有关如何编译和链接源文件与运行所得到的程序的信息。IDE 将项目信息存储在包含 makefile 和元数据文件的项目文件夹中。尽管建议将源文件目录置于项目文件夹下以实现项目可移植性，但并非必须这样做。

在 IDE 中，即使程序包含在单个源文件中，也始终在项目内工作。每个项目必须具有一个 makefile，以便 IDE 可以编译该项目。项目的 makefile 可由 IDE 生成，也可使用之前在 IDE 之外创建的 makefile。

使用由 IDE 生成的 makefile 的项目被称为**管理型项目**，具有多种类型。使用 New Project 向导在 MPLAB X IDE 中创建项目时，可以从可用项目类型中进行选择。

使用在 IDE 之外创建的 makefile 的项目被称为**用户 Makefile 项目**。关于创建此类 makefile 的更多信息，请参见 Creating Makefiles Outside of MPLAB® X IDE。

通过 IDE 的 **12.15 Projects 窗口**处理项目。

可通过更改配置类型将独立（应用程序）项目更改为库项目。有关详细信息，请参见 4.10.1 更改项目配置类型。

4.1.1 MPLAB X IDE v5 新项目格式

v5.00 及更高版本中的新项目格式支持包含各版本器件信息的包。包位于以下位置：

```
<MPLAB X IDE 安装目录>\v5.xx\packs
```

管理项目

在 v5.00 或更高版本中创建或更新到 v5.00 或更高版本的项目不向后兼容 v4.xx。

打开在 v5.00 之前版本中创建的项目时，将显示一条消息，说明系统会将该项目升级到当前项目版本。

- 如果选择 **Yes**（是），该项目将升级到当前版本，并且无法再使用早期版本的 IDE 打开。
- 如果选择 **No**（否），则无法在 v5.00 或更高版本中保存对项目的任何修改。必须在 v4.xx 中进行更改。

如果需要将升级到 v5.xx 的项目恢复到 v4.xx，请在 MPLAB X IDE 中安装位于 *Tools>Plugins>Available Plugins>Save As v4.xx Project*（工具>插件>可用插件>另存为 v4.xx 项目）下的插件。安装该插件后，可以通过选择 *Tools>Embedded>Save as MPLAB X v4.xx Project*（工具>已安装工具>另存为 MPLAB X v4.xx 项目）在旧版本中保存项目。

关于器件包

过去（v4.20 和更早版本），每个 MPLAB X IDE 版本中均内置了受支持的器件（*PIC）文件。这些 MPLAB X IDE 版本无法更新支持的器件，必须等到下一个版本才能更新器件支持。

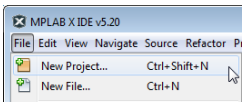
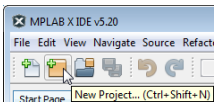
现在（v5.00 和最新版本），器件文件被分组到各版本器件系列包中。虽然每个 MPLAB X IDE 版本都随附器件包，但是可通过升级器件包版本以包含新器件、新器件功能支持或器件缺陷修正支持。

关于器件包的更多信息，请参见：

5.1 使用器件包

4.1.2 启动 New Project 向导

可以通过以下多种方式来启动 New Project 向导。

	*File>New Project*（文件>新建项目）（或 Ctrl+Shift+N）
	New Project 图标（在文件工具栏上）
	**Start Page**，**Learn & Discover** 或 **My MPLAB X IDE**（我的 MPLAB X IDE）选项卡，“Projects”>“Create New”（新建）

4.1.3 步骤 1：选择项目

选择一个项目类别。在大多数情况下，将从“Microchip Embedded”（Microchip 嵌入式）中选择项目类型：

以下选项可用于选择项目类别：

- Standalone Project（独立项目）——创建新的 C 和/或汇编代码项目。这类项目显示在该部分。
- Existing MPLAB IDE v8 Project（现有 MPLAB IDE v8 项目）——将现有 MPLAB IDE v8 项目转换为 MPLAB X IDE 项目。有关详细信息，请参见 5.3 导入现有 MPLAB IDE v8 项目。
- Prebuilt (Hex, Loadable Image) Project（预编译（十六进制可装入映像）项目）——将现有项目映像装入 MPLAB X IDE 中。有关详细信息，请参见 5.4 预编译项目。
- User Makefile Project（用户 Makefile 项目）——创建一个使用外部 makefile 的项目。有关详细信息，请参见 6.6 创建用户 Makefile 项目。

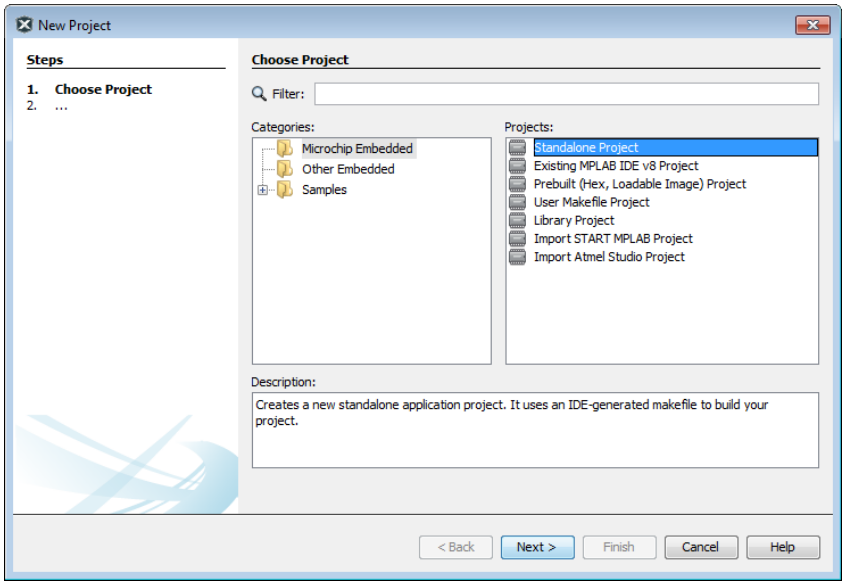
- Library Project（库项目）——新建一个将编译为库（而非可执行十六进制文件）的 C 和/或汇编代码项目。有关详细信息，请参见 5.7 库项目。
- Import START MPLAB Project（导入 START MPLAB 项目）和 Import Atmel Studio Project（导入 Atmel Studio 项目）——导入这些 Atmel 项目。有关详细信息，请参见 5.8 导入 Atmel Studio 7 或 Atmel START 项目。

还可使用其他选项：

- 5.9 其他嵌入式项目 ——来自其他供应商的项目。
- 5.10 示例项目 ——包含不同器件系列的现成项目，以及不同器件系列的项目模板。

完成选择之后，单击 **Next>**（下一步>）移至下一个对话框。

图 4-1. 项目向导——选择项目



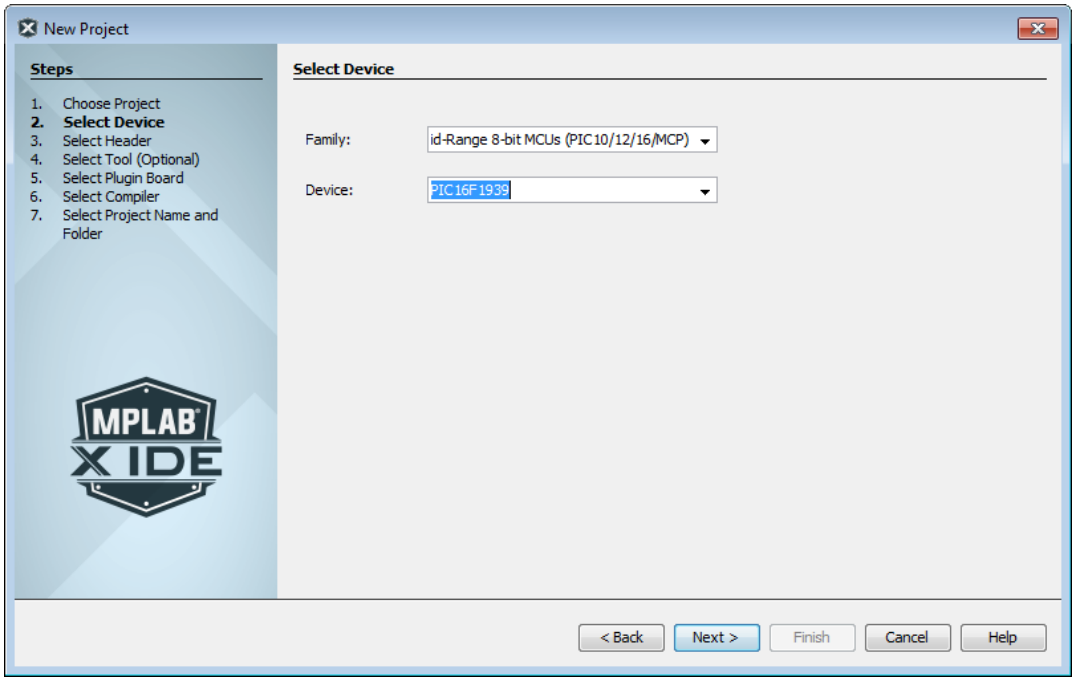
4.1.4 步骤 2：选择器件

从“Device”下拉列表中选择将在应用中使用的器件。要缩小选择列表，请先选择“Family”（系列）。

对于 LF 器件，注意需要将物理器件设置为使用低于“F”型器件电压值的电压供电，通常为 3.3V（而非 5.0V）。如果电压较高会损坏器件，MPLAB X IDE 将显示 Warning（警告）对话框发出提示。

单击 **Next>**。

图 4-2. 项目向导——选择器件



4.1.5 步骤 3：选择调试头

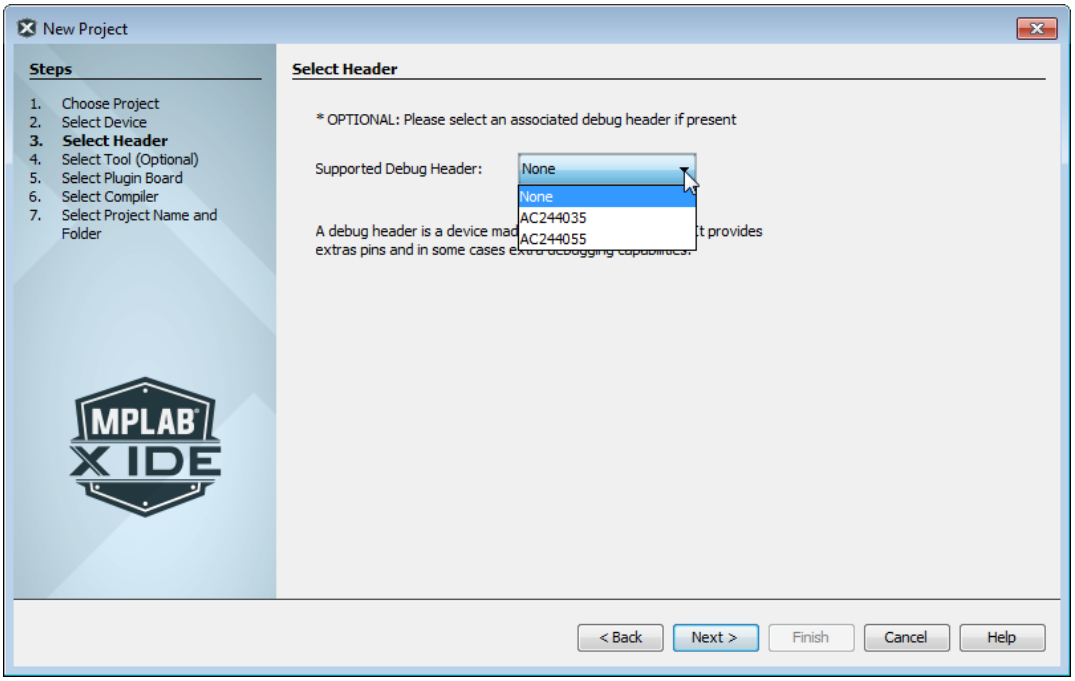
为项目器件选择调试头。如果调试头不可用，MPLAB X IDE 将不会显示此步骤。如果调试头可用，请参见下面的“开发人员帮助”主题获取更多信息：

- 处理器扩展包和调试头
- 仿真扩展包和仿真头

完成后，单击 **Next>**。

注： 稍后可使用 Project Properties 窗口选择调试头（如果可用）。

图 4-3. 项目向导——选择调试头



4.1.6 步骤 4：选择工具

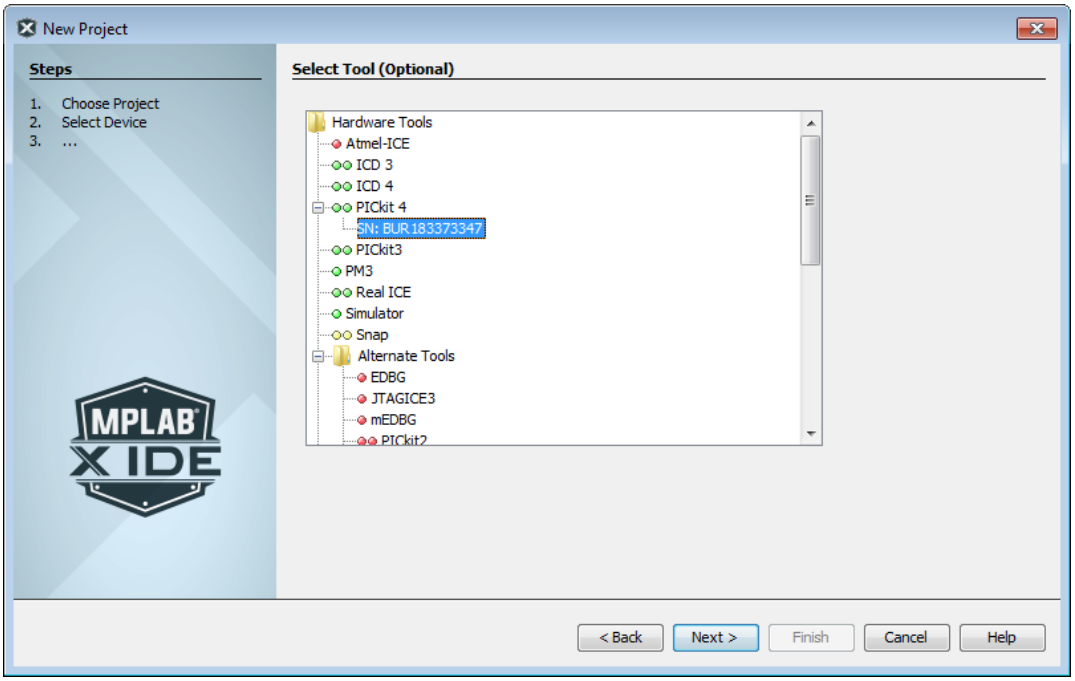
选择调试工具。

关于工具名称旁边的彩色圆圈的信息，请参见 4.1.6.1 项目工具支持。

对于硬件工具，可以注意到在已连接到计算机的所有工具下面都会指定一个序列号（Serial Number，SN）。这使您可以从几个已连接的硬件工具中进行选择。

选择您的工具，然后单击 **Next>**。

图 4-4. 项目向导——选择工具

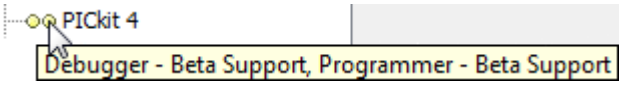


4.1.6.1 项目工具支持

项目器件的项目工具支持通过工具名称前的彩色圆形（指示灯）指示。

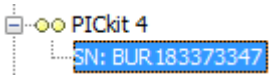
指示灯	颜色	支持
●	绿色	完全（已实现并已经过完全测试）
●	黄色	Beta（已实现但未经过完全测试）
●	红色	无

如果看不到颜色，可将鼠标移至指示灯上来弹出关于支持的文本。



对于可用作调试器或编程器的硬件工具，工具名称旁会有两个指示灯，其中的第一个（最左侧）指示灯表示调试器支持，第二个指示灯表示编程器支持。

同样对于硬件工具，可以注意到在已连接到计算机的所有工具下面都会指定一个序列号（SN）。这使您可以从几个已连接的硬件工具中进行选择。



如果使用的是第三方硬件工具，但未在此处列出，请确保已正确安装了该工具。更多信息，请参见 6.8 与第三方硬件工具配合使用。

4.1.7 步骤 5：选择接插板

选择 MPLAB REAL ICE 在线仿真器接插板。如果未将此仿真器选作调试工具，MPLAB X IDE 将不会显示此步骤。

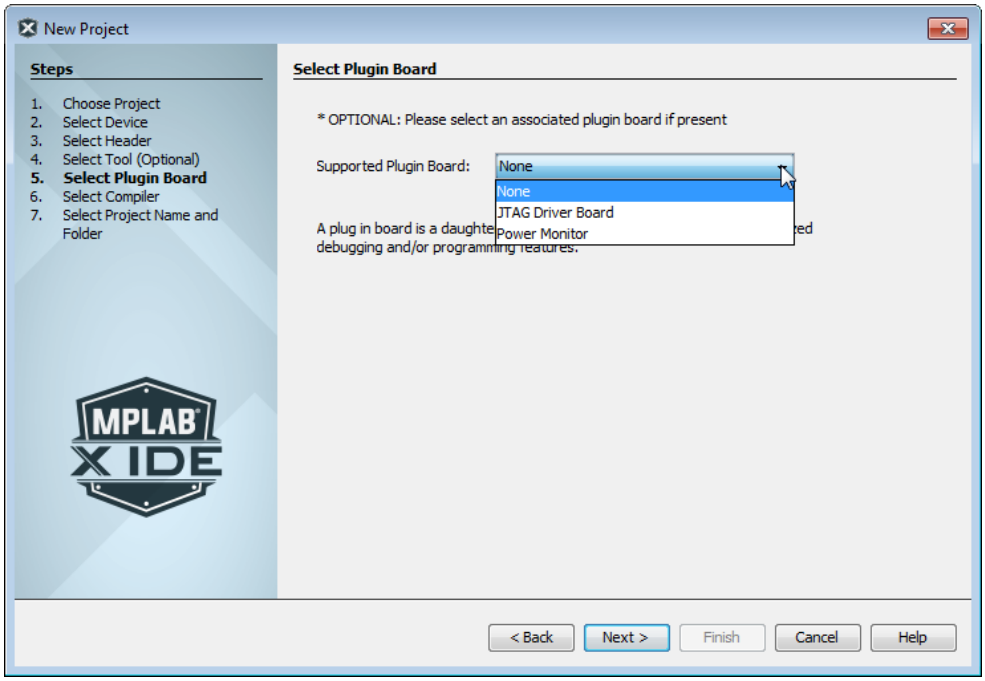
接插板是插入到仿真器的驱动板插槽的电路板，用于提供或添加调试功能。

表 4-2. 仿真器接插板

支持的接插板	电路板说明
无	标准通信驱动板
无	高速通信驱动板
JTAG 驱动板	JTAG 适配器板
电源监视器板	电源监视器板（也插入逻辑探针连接器）

选择您的工具，然后单击 **Next>**。

图 4-5. 项目向导——选择接插板



4.1.8 步骤 6：选择编译器

选择语言工具，C 编译器或汇编器。可选语言工具显示为以下形式：

工具缩写（工具版本）[工具可执行路径]

关于工具名称旁边的彩色圆圈（指示灯）的信息，请参见 4.1.6.1 项目工具支持。

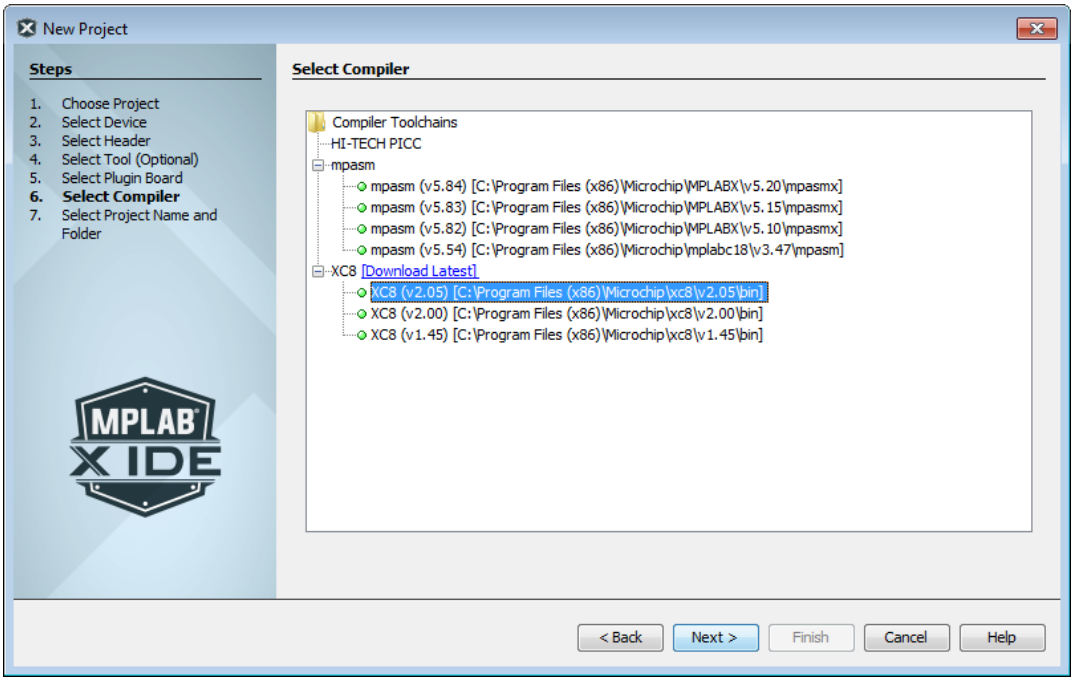
如果未列出语言工具或者未显示任何支持，应确保已安装语言工具。请参见 2.5 安装语言工具。

注： 对于 AVR GNU 和 Arm GNU 工具链，请确保将编译器安装在 MPLAB X IDE 可搜索到的位置。例如，对于 Windows OS，安装在 `C:\Program Files (x86)\Microchip` 下。否则，您将必须输入路径。请参见 4.7 设置语言工具位置。

如果 Build Tools（编译工具）下列出了您的工具，但未显示相关支持，则您的项目器件可能不受该工具或工具版本支持。请考虑选择或安装支持该器件的其他语言工具。或者，您也可以使用“Download Latest”（下载最新版本）下载并安装最新版本的 MPLAB XC 编译器，该编译器可能支持您的器件。

选择您的工具，然后单击 **Next>**。

图 4-6. 项目向导——选择编译器（或其他语言工具）



4.1.8.1 支持的语言工具

下表列出了最新和传统 MPLAB X IDE 所支持的语言工具。最新语言工具支持新器件。传统语言工具则不支持。

表 4-3. Microchip 语言工具——最新

工具链	全名	器件支持
XC8	适用于 PIC MCU 的 MPLAB XC8 C 编译器	8 位 PIC® MCU
	适用于 AVR MCU 的 MPLAB XC8 C 编译器	8 位 AVR® MCU
XC16	MPLAB XC16 C 编译器	16 位 PIC MCU 和 dsPIC® DSC
XC32	适用于 PIC32M MCU 的 MPLAB XC32 C/C++编译器	32 位 PIC32M MCU
	适用于 PIC32C/SAM MCU 的 MPLAB XC32 C/C++编译器	32 位 PIC32C 和 SAM MCU
Arm® GNU	Arm GNU C 编译器	32 位 Arm MCU
AVR GNU	AVR GNU C 编译器	8 位和 32 位 AVR MCU
MPASM	MPASM 汇编器、MPLINK 目标链接器和实用程序	8 位 PIC MCU

表 4-4. Microchip 语言工具——传统

工具链	全名
8 位 PIC MCU 语言工具	
C18*	适用于 PIC18 MCU 的 MPLAB C 编译器
HI-TECH PICC	适用于 PIC10/12/16 MCU 的 HI-TECH C 编译器
HI-TECH PICC18	适用于 PIC18 MCU 的 HI-TECH C 编译器
16 位 PIC MCU 和 dsPIC DSC 语言工具	
ASM30**	适用于 PIC24 MCU 和 dsPIC DSC 的 MPLAB 汇编器、目标链接器和实用程序

.....（续）

工具链	全名
C30	适用于 PIC24 MCU 和 dsPIC DSC 的 MPLAB C 编译器
C24	适用于 PIC24 MCU 的 MPLAB C 编译器（C30 的子集）
dsPIC	适用于 dsPIC DSC 的 MPLAB C 编译器（C30 的子集）
HI-TECH DSPICC	适用于 PIC24 MCU 和 dsPIC DSC 的 HI-TECH C 编译器
32 位 PIC MCU 语言工具	
C32	适用于 PIC32 MCU 的 MPLAB C 编译器
HI-TECH PICC32	适用于 PIC32 MCU 的 HI-TECH C 编译器
*大多数编译器都随附汇编器、链接器和实用程序。但 MPLAB C18 由 MPASM 工具链提供支持。 **自 MPLAB X IDE v1.30 起，不再包含。请使用任一 16 位编译器随附的汇编器。	

关于每种语言工具的更多信息，请参见语言工具文档。

对于第三方语言工具链（CCS 等），请参见 Start page 上“Release Notes and Support Documentation”（发行说明和支持文档）下的“Readme for Third Party Tools.htm”文件。

4.1.8.2 下载最新 MPLAB XC 编译器

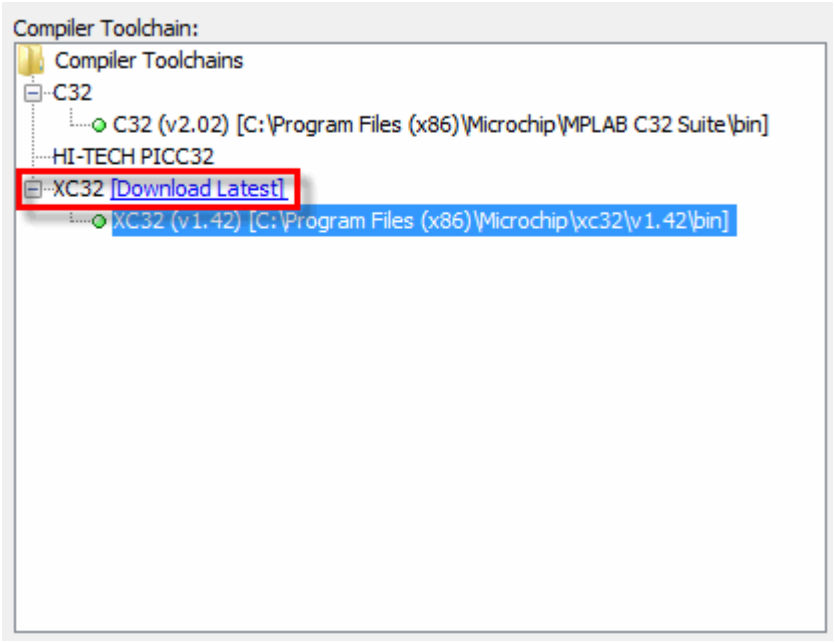
单击“Download Latest”链接，下载最新版本的 MPLAB XC C 编译器。另外，对于某些操作系统，该操作还将安装编译器并更新编译器选择树。

“Download Latest”链接位于 New Project 向导的步骤 6.“Select Compiler”（选择编译器）的 Compiler Toolchains>XCn（编译器工具链>XCn）下，其中 n = 8、16 或 32。

注： 如果没有互联网连接，则不会显示该链接。

项目创建完成后，该链接将出现在 Project Properties 窗口中。

图 4-7. 下载最新版本



单击“Download Latest”链接后：

1. 将显示一个消息对话框，告知您将要进行的操作。您现在可以停止下载和安装。

2. 如果您已经安装了最新的编译器，则会出现一个消息对话框，询问您是否要继续。
3. 系统将要求您选择编译器安装程序的目标位置。
4. 下载时，将出现一个进度窗口，指示剩余的下载量。
5. Windows 和 Mac OS 系统：下载安装程序后，将立即开始安装编译器。安装编译器后，MPLAB X IDE 会尝试将编译器添加到选择树中。如果无法添加，系统会通知您需要手动将编译器添加到 MPLAB X IDE（4.7 设置语言工具位置）。
6. Linux OS 系统：必须手动安装下载的编译器。添加后，MPLAB X IDE 应能够识别编译器。如果无法识别，将需要手动将编译器添加到 MPLAB X IDE（4.7 设置语言工具位置）。

4.1.9 步骤 7：选择项目名称和文件夹

选择项目名称、位置和其他项目功能。

输入项目名称。按照约定，项目名称末尾将带有.X。请参见“Project Folder”（项目文件夹）文本框。

浏览至文件夹位置。如果需要，可以创建一个新的项目文件夹。默认情况下，项目将放置在：

- Windows 7 及以上版本——C:\Users\UserName\MPLABXProjects
- Linux——/home/UserName/MPLABXProjects
- Mac——/Users/UserName/MPLABXProjects

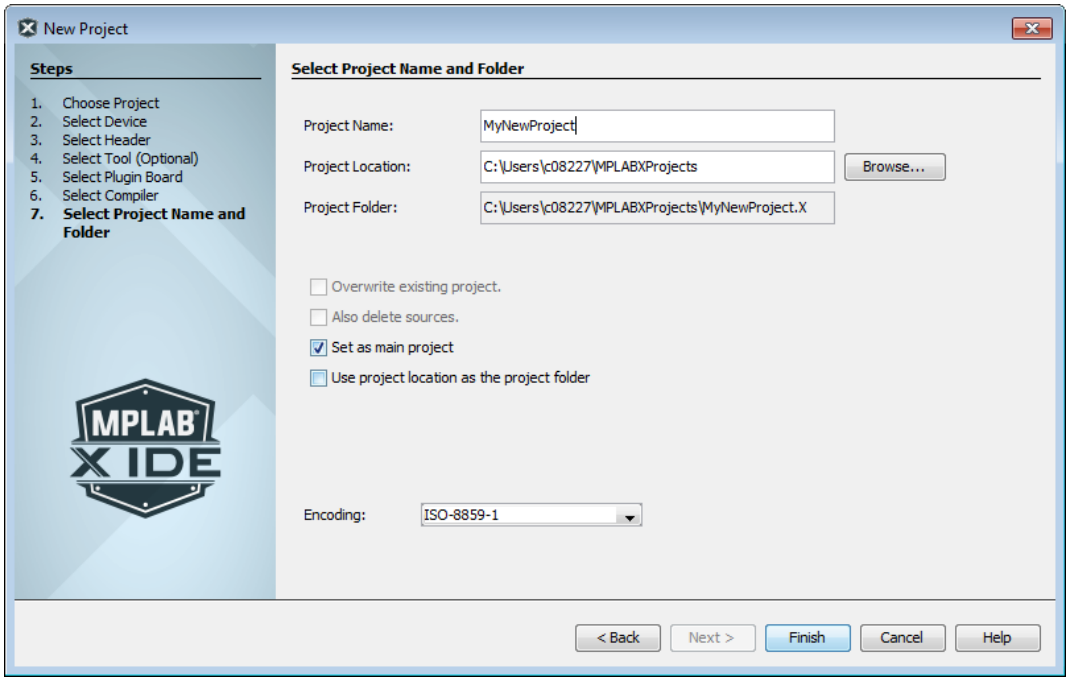
表 4-5. 其他功能

Set as main project（设置为主项目）	将该项目指定为主项目
Use project location as the project folder（使用项目位置作为项目文件夹）	在与项目位置相同的文件夹中创建项目。 如果要导入 MPLAB IDE v8 项目并希望 MPLAB X IDE 项目位于同一文件夹中（例如，当您希望编译相同时），这将十分有用。 MPLAB X IDE 使用文件夹存储项目信息，而 MPLAB IDE v8 使用.mcp 文件。因此，只能使一个 MPLAB X IDE 项目与一个 MPLAB IDE v8 项目（.mcp 文件）共享文件夹。
Encoding（编码）	选择项目的编码。默认为 ISO-8859-1 (Latin 1)字符集。 该选择将指定代码语法着色标记，它可以在 *Tools>Options*（对于 macOS 为 *MPLAB X IDE>Preferences*（MPLAB X IDE>首选项）），**Fonts and Colors**（字体和颜色）按钮，**Syntax**（语法）选项卡下进行编辑。

结束后，单击 **Finish**（完成）完成新项目的创建。

注： 可通过更改配置类型将独立（应用程序）项目更改为库项目。有关详细信息，请参见 4.10.1 更改项目配置类型。

图 4-8. 项目向导——选择项目名称和文件夹



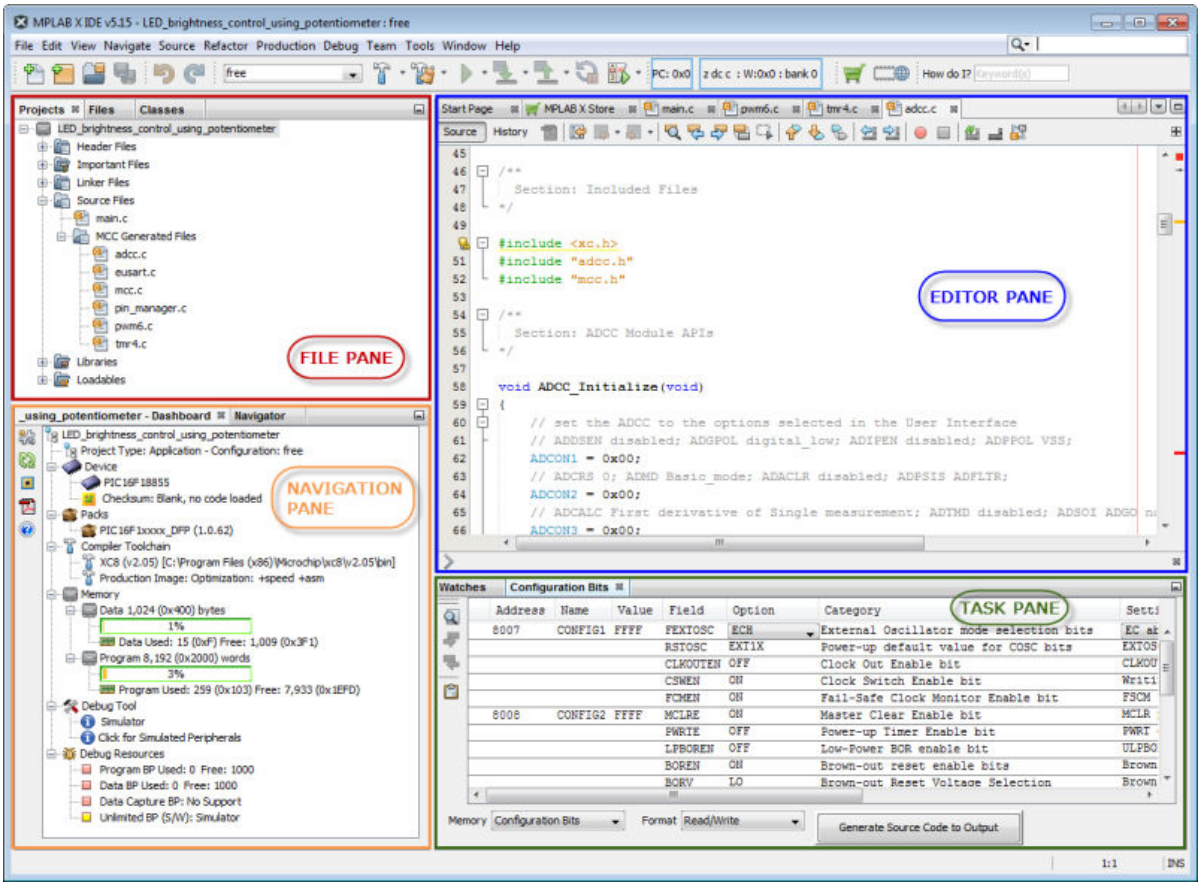
4.2 查看桌面上的变化

MPLAB X IDE 桌面分为四个**窗格**。创建项目之后，将会在这些窗格中打开几个**窗口**。

桌面窗格

- **文件窗格**——显示与文件相关的信息。
  - Projects 窗口将显示文件按类别进行分组的项目树。
  - Files 窗口将根据计算机上的文件夹组织来显示项目文件。
  - Classes（类）窗口将显示代码中的所有类及其函数、变量和常量。双击某个项可查看其声明。
- **导航窗格**——显示关于文件窗格中的项的信息。
  - Dashboard 窗口将显示关于选定项目的信息。
  - Navigator 窗口将显示关于选定项目文件中的符号和变量的信息。
- **编辑器窗格**——用于查看和编辑项目文件。Start Page 和 MPLAB X Store 也在此处显示。
- **任务窗格**——显示编译、调试或运行应用程序产生的任务输出。

图 4-9. MPLAB X IDE 桌面——窗格



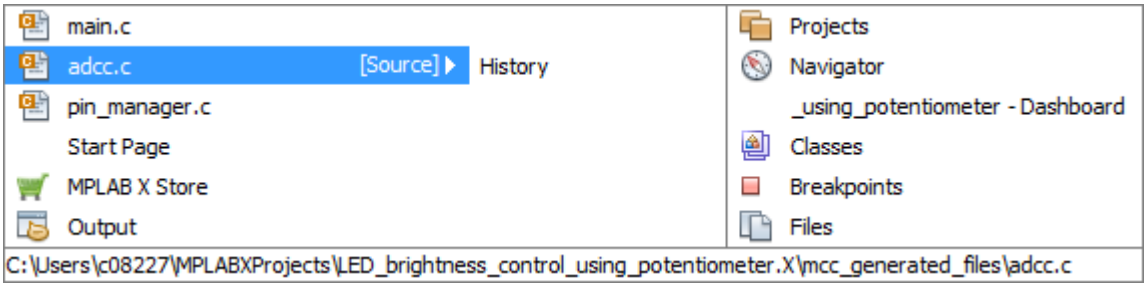
桌面窗口

尽管窗格用于描述 IDE 界面的各个部分，但窗口可以从一个窗格移至另一个窗格，所以窗格并非始终代表窗口内容。因此，大多数文档中仅讨论窗口。

要在桌面上的窗口之间移动：

- 请单击一个窗口，使其成为焦点窗口。
- 当处于一个窗口中时，可以将焦点切换到任何其他打开的窗口，方法是按住<Ctrl>，然后按下<Tab>。从弹出列表中选择窗口。

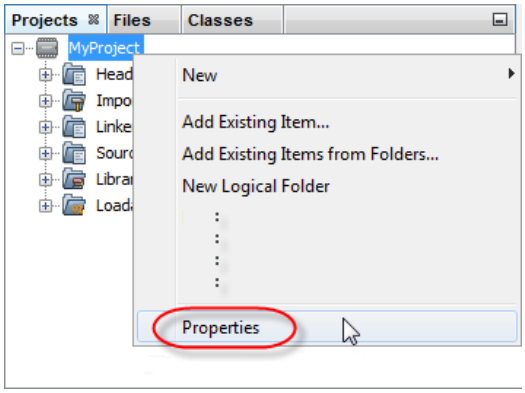
图 4-10. 切换窗口焦点



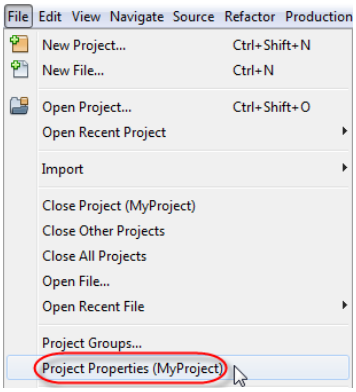
4.3 打开 Project Properties

创建项目后，可以在 Project Properties 窗口中查看或更改项目属性。可通过执行以下操作之一来访问该窗口。

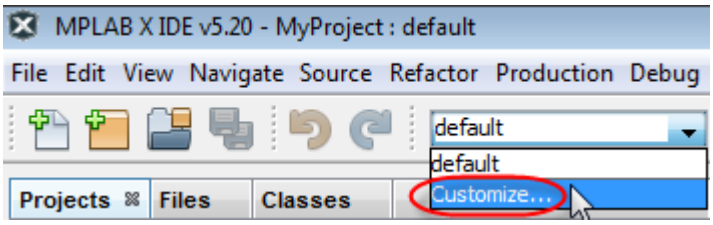
1.右键单击 Projects 窗口中的项目名称并选择“Properties”。



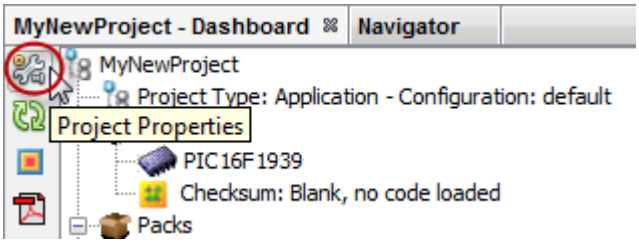
2.单击 Projects 窗口中的项目名称，然后选择 *File>Project Properties*（文件>项目属性）。



3.单击 Projects 窗口中的项目名称，然后单击“Set Project Configuration”（设置项目配置）下拉菜单，选择“Customize”（定制）。



4.单击 Dashboard 窗口上的“Project Properties”图标。



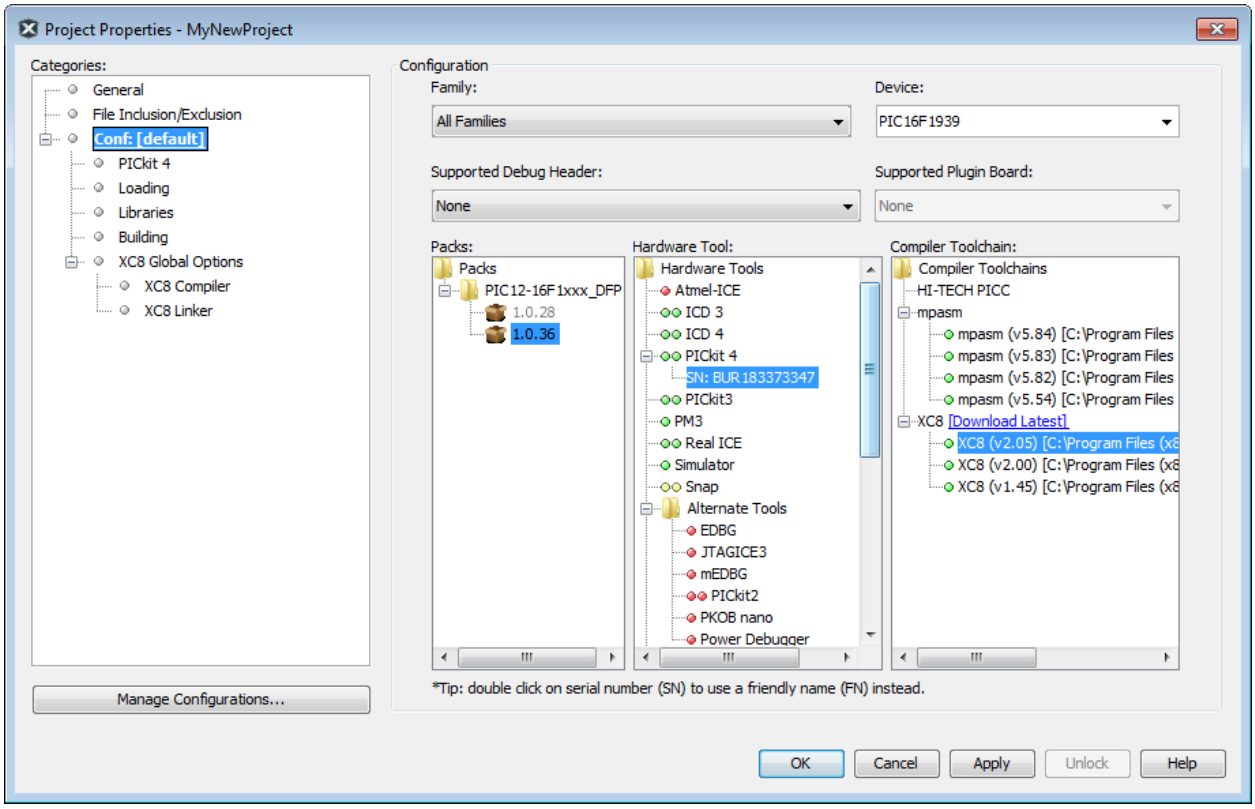
4.4 查看或更改项目属性

单击“Conf: [default]”类别可显示一般项目配置，例如项目器件、相关的调试/编程器工具和语言工具。

Categories	创建项目配置时将其设置为“default”（默认）。可通过单击 **Manage Configurations**（管理配置）按钮创建不同的配置。请参见 6.4 处理多个配置。

Device	应为模拟的器件或电路板上的器件。 更换器件时需谨慎，尤其是换成其他系列时，因为包、硬件工具和编译器都需要更新。
Packs	此为所选器件的器件系列包（Device Family Pack，DFP）版本。最新版本的 MPLAB X IDE 将包含最新 DFP 版本。但是，可以改为其他版本（如可用）。请参见 5.1 使用器件包。
Hardware	应选择一个受支持的硬件调试工具（或软件模拟器）。请参见 4.1.6.1 项目工具支持。
Compiler Toolchain	应选择一个受支持的语言工具（通常为编译器）。请参见 4.1.6.1 项目工具支持。 要获取最新版本的 MPLAB XC C 编译器，请单击“Download Latest”链接。请参见 4.1.8.2 下载最新 MPLAB XC 编译器。

图 4-11. Project Properties 窗口——默认配置

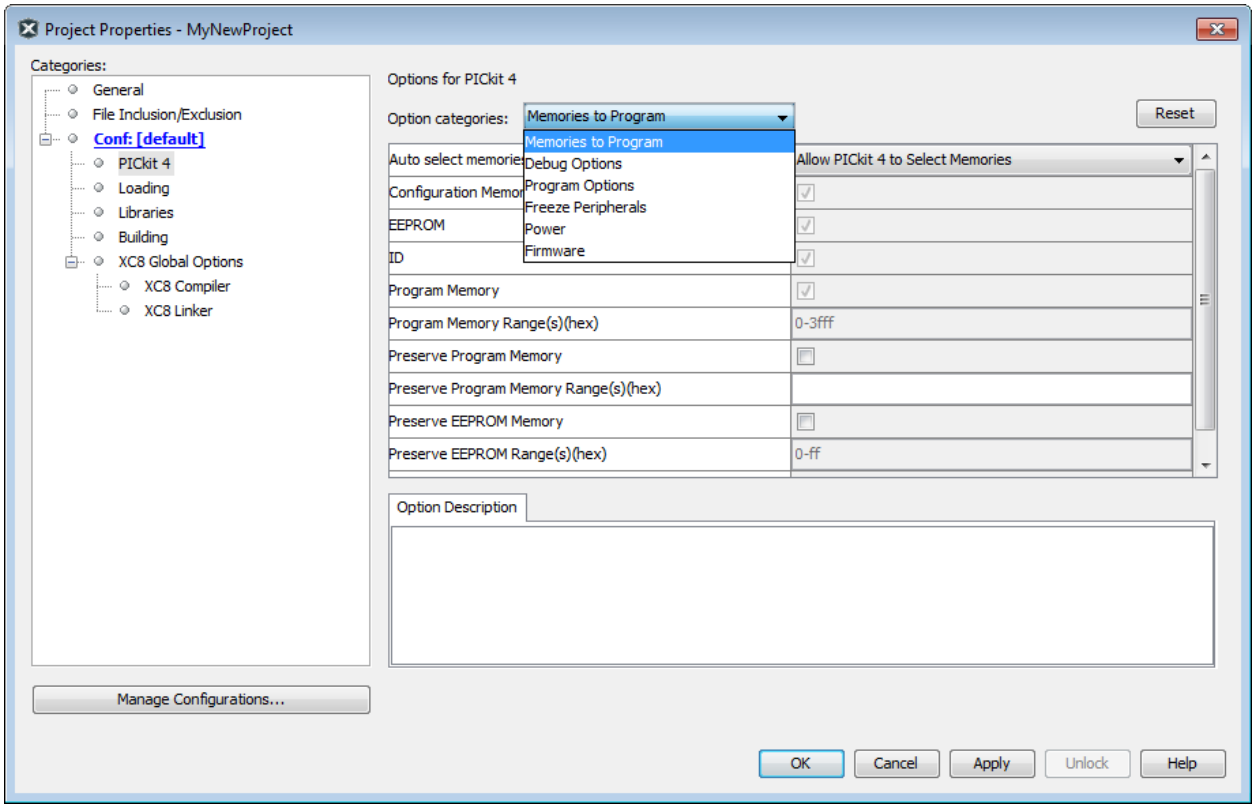


4.5 设置或更改调试器/编程器工具选项

在 Project Properties 窗口中设置工具选项。

单击硬件工具或软件模拟器（在 Conf:[default]下）以查看相关的设置选项。关于这些选项含义的更多信息，请参见工具文档。

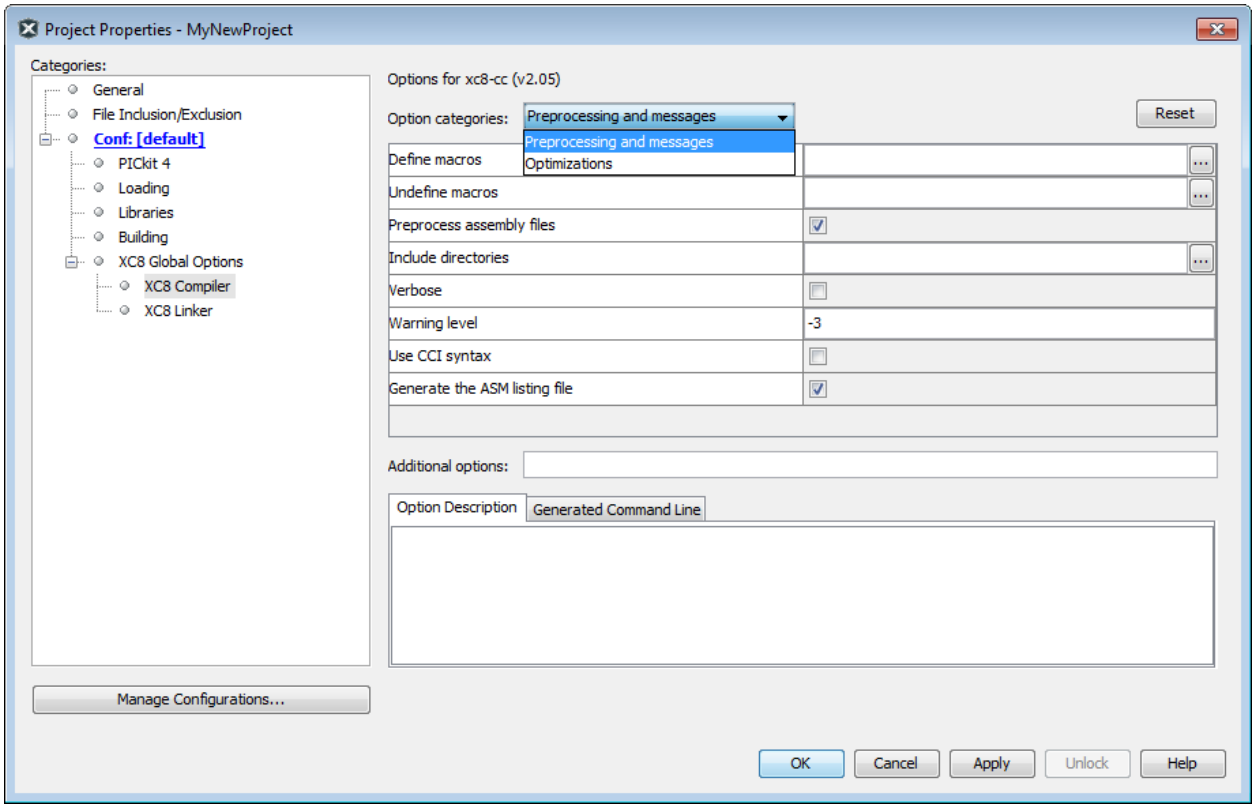
图 4-12. 工具设置页面示例



4.6 设置或更改语言工具选项

可在 Project Properties 窗口中单击语言工具查看相关设置选项。关于这些选项含义的更多信息，请参见语言工具文档。

图 4-13. 语言工具设置页面



IDE 选项与命令行选项

输入 IDE 的选项的格式可能与输入命令行的选项的格式不同。单击某个选项的名称可在“Option Description”（选项说明）选项卡中查看有关该选项的更多信息。此外，在输入或选择选项数据之后，单击“Generated Command Line”（生成的命令行）选项卡以查看其内容，并确保由选项生成的命令行代码符合您的预期。

全局选项

设置适用于语言工具套件中所有语言工具的全局选项。还可使用“User Comments”（用户注释）选项卡输入有关为何选择某些选项的信息。

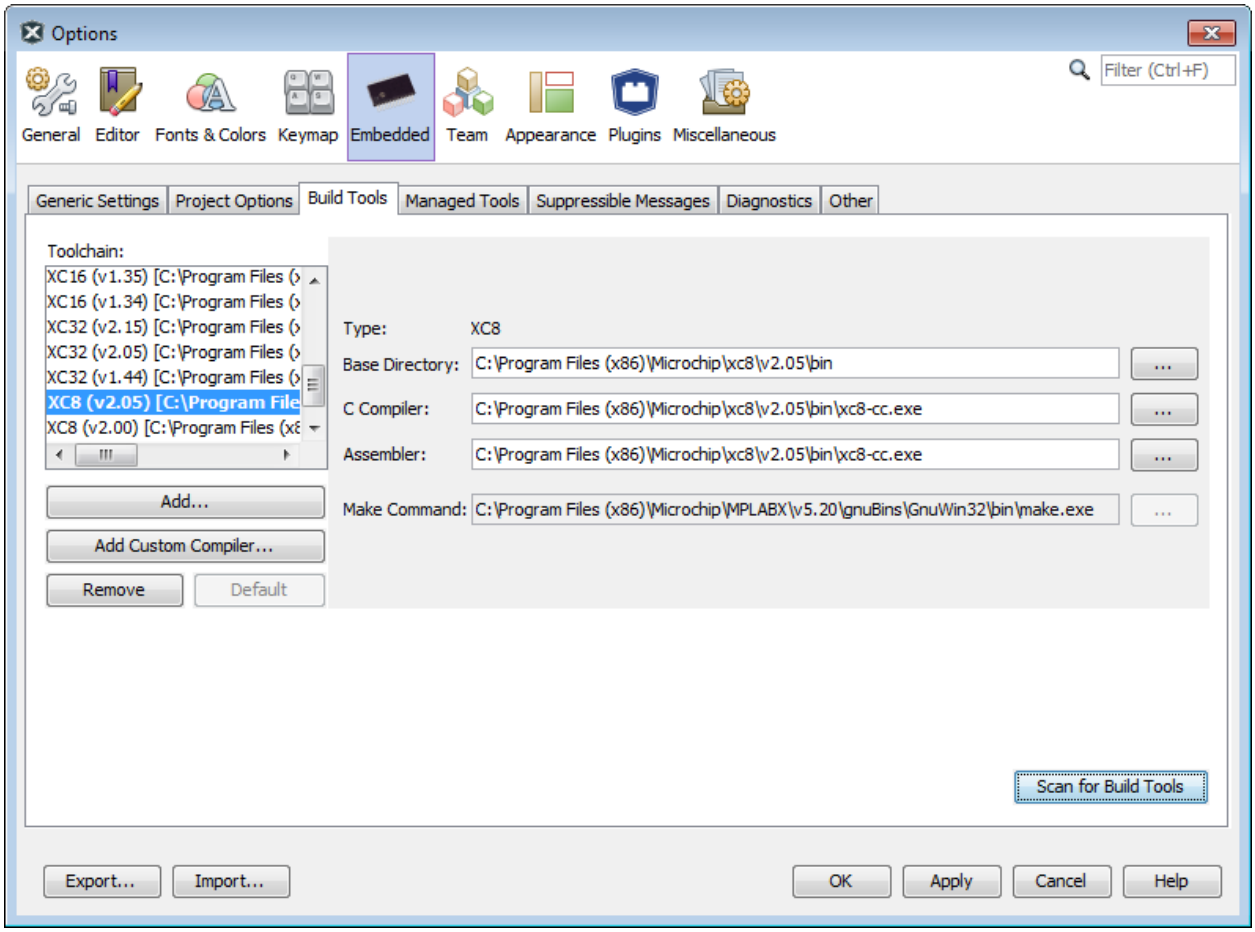
4.7 设置语言工具位置

要了解哪些语言工具可用于 MPLAB X IDE，以及要查看或更改其路径，应按照以下说明操作：

- 对于 macOS——通过以下方式访问编译工具：从主菜单栏选择 *MPLAB X IDE>Preferences>Embedded>Build Tools*（MPLAB X IDE>首选项>已安装工具>编译工具）。
- 对于其他操作系统——通过以下方式访问编译工具：*Tools>Options>Embedded>Build Tools*。

该窗口应自动填充所有已安装的工具链。另请参见 12.16.3 Build Options 选项卡。

图 4-14. 语言工具（编译器）位置



4.7.1 添加或更改工具链

如果您的工具未在“Toolchain”（工具链）下列出，则可尝试以下功能：

- Scan for Build Tools（扫描编译工具）——扫描环境路径并列出计算机上已安装的语言工具。
- Add——通过输入包含工具可执行文件的目录（即根目录）的路径，将工具手动添加到列表中。通常，这是工具安装目录中的 bin 子目录。

如果有多个编译器版本，请从列表中选择一个。

要更改工具的路径，可输入新路径或通过浏览设置。

4.7.2 删除工具链

要删除现有工具链，应执行以下操作：

1. 单击以从列表中选择工具链。
2. 单击 Remove（删除）按钮。

该操作不会卸载计算机上的编译器；仅仅会删除该编译器与 MPLAB X IDE 的关联。要重新添加工具链，请参见第 4.7.1 节“添加或更改工具链”。

如果要从计算机上卸载编译器，请按照上述说明首先在 MPLAB X IDE 中删除对编译器的引用。然后卸载编译器。如果先卸载编译器再启动 MPLAB X IDE，则工具链路径将显示为红色。要删除对该已卸载编译器的引用，请按照上述步骤进行操作。

4.7.3 关于工具链路径

MPLAB X IDE 会在默认安装路径和 PATH 环境变量下搜索工具链。MPLAB XC16 在 Windows 64 位操作系统下的示例默认路径：

```
C:\Program Files (x86)\Microchip\xc16\vx.xx\bin
```

在安装编译器时，可以选择安装位置。既可以：

- 安装在默认位置

也可以

- 安装到其他位置，然后将该位置添加到 PATH 中

如果要安装在不同于默认位置的其他位置，应将该位置添加到 PATH 中。

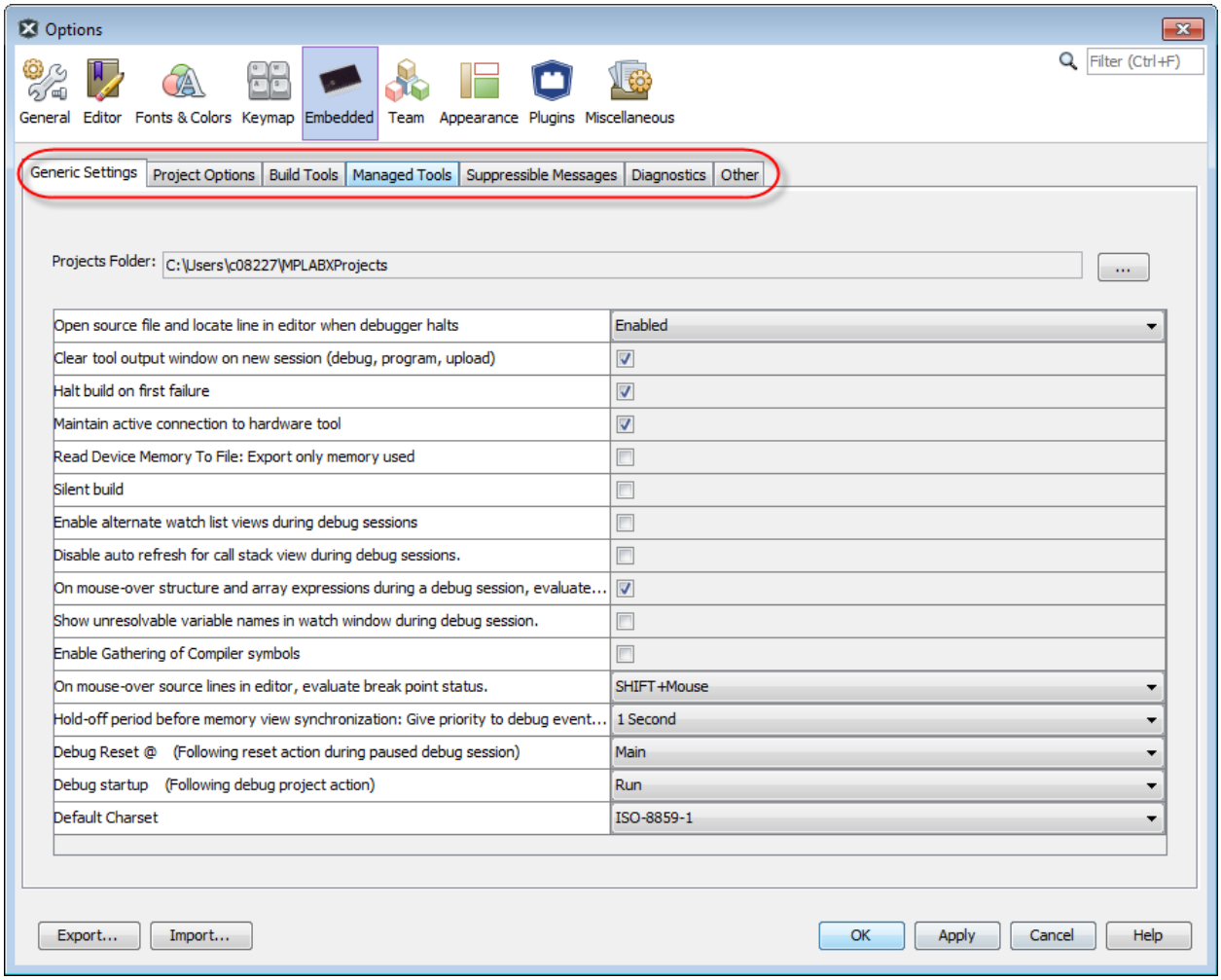
如果不选择让安装程序将新位置添加到 PATH 中，可以在 Tools>Options，Embedded 按钮，Build Tools 选项卡中为 MPLAB X IDE 指明编译器位置。

最后，可以手动将新位置添加到 PATH 变量中。

如果 MPLAB X IDE 已知该工具链，但是找不到路径，则该路径将显示为红色文本。

4.8 设置其他工具选项

除了编译路径之外，还可以设置其他 Embedded 选项。在 Options 窗口 Embedded 类别中的选项卡中进行选择。请参见 12.16 Tools>Options 窗口，Embedded。



4.9 将文件添加到项目

创建新项目后，需要向其添加文件以创建应用程序。您可以新建一个空文件来写入代码，也可以导入或引用现有文件。使用编辑器时，可利用语言工具特定的颜色标识和其他功能来创建或更新任何文件内容。

文件链接顺序

文件添加到项目的顺序将决定其链接顺序。如果已从 MPLAB IDE v8 导入了一个项目，则顺序将由 .mcp 文件决定。添加到项目的新文件将最后链接。如果删除文件然后重新添加，则该文件将移至链接列表的末尾。

库和目标文件链接

可以引用链接器要使用的库和目标文件。

从编译中排除文件

可以设置文件或文件夹级别选项，以从编译中排除文件。

4.9.1 启动 New File 向导

要新建一个空文件来添加代码到项目，可使用 New File（新建文件）向导。可以通过以下多种方式来启动向导。

	*File>New File*（文件>新建文件）（或 Ctrl+N）

	New File 图标（在文件工具栏上）
	在 Projects/Files（项目/文件）窗口中右键单击项目文件夹（例如“Source Files”），然后选择 *New>Other*（新建>其他）。

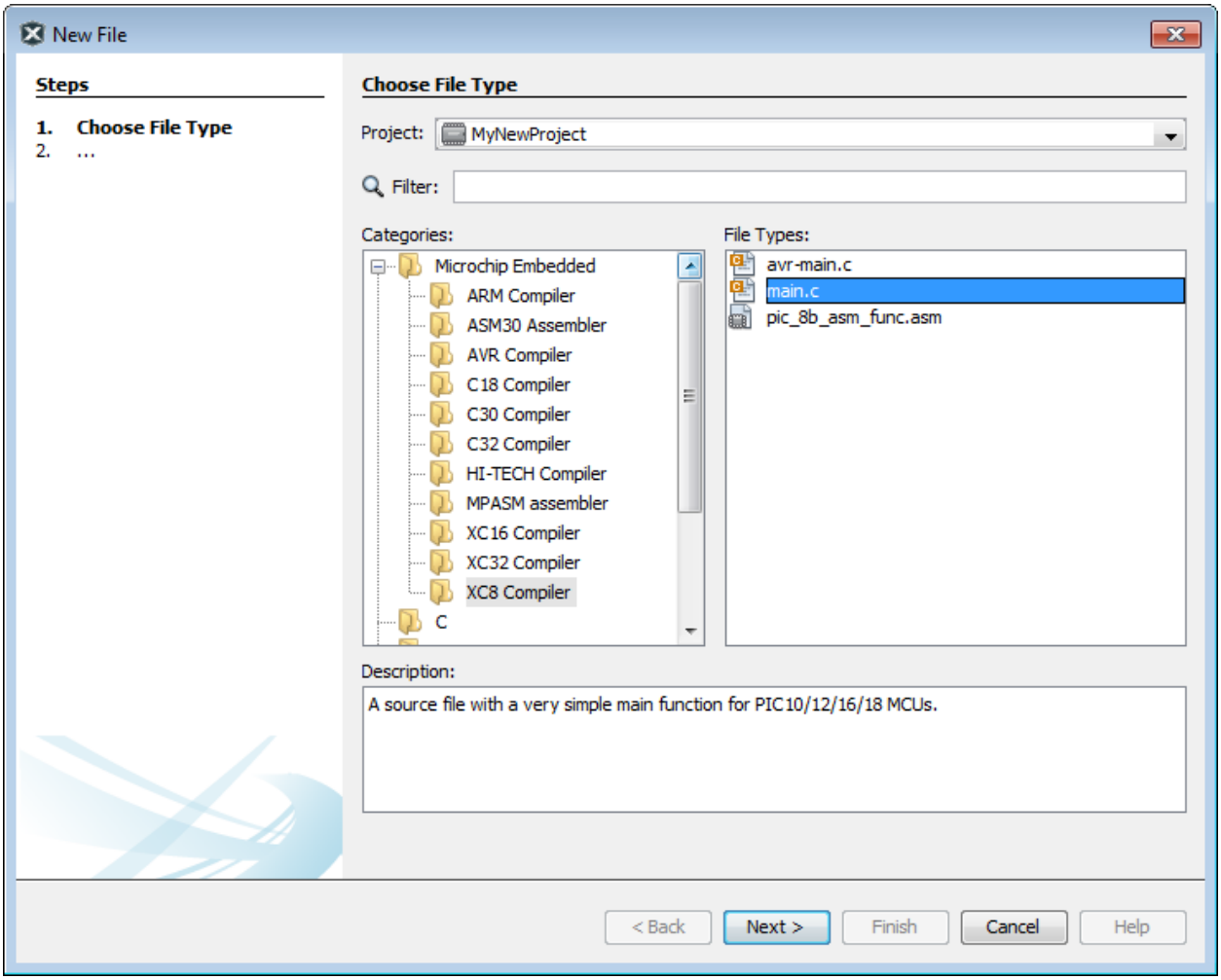
4.9.2 创建一个新文件

按照 New File 向导中的步骤创建一个新文件。

步骤 1.选择文件类型

通过展开“Microchip Embedded”来选择文件类别，以找到适当的选项。然后选择一种文件类型。单击 **Next>**。

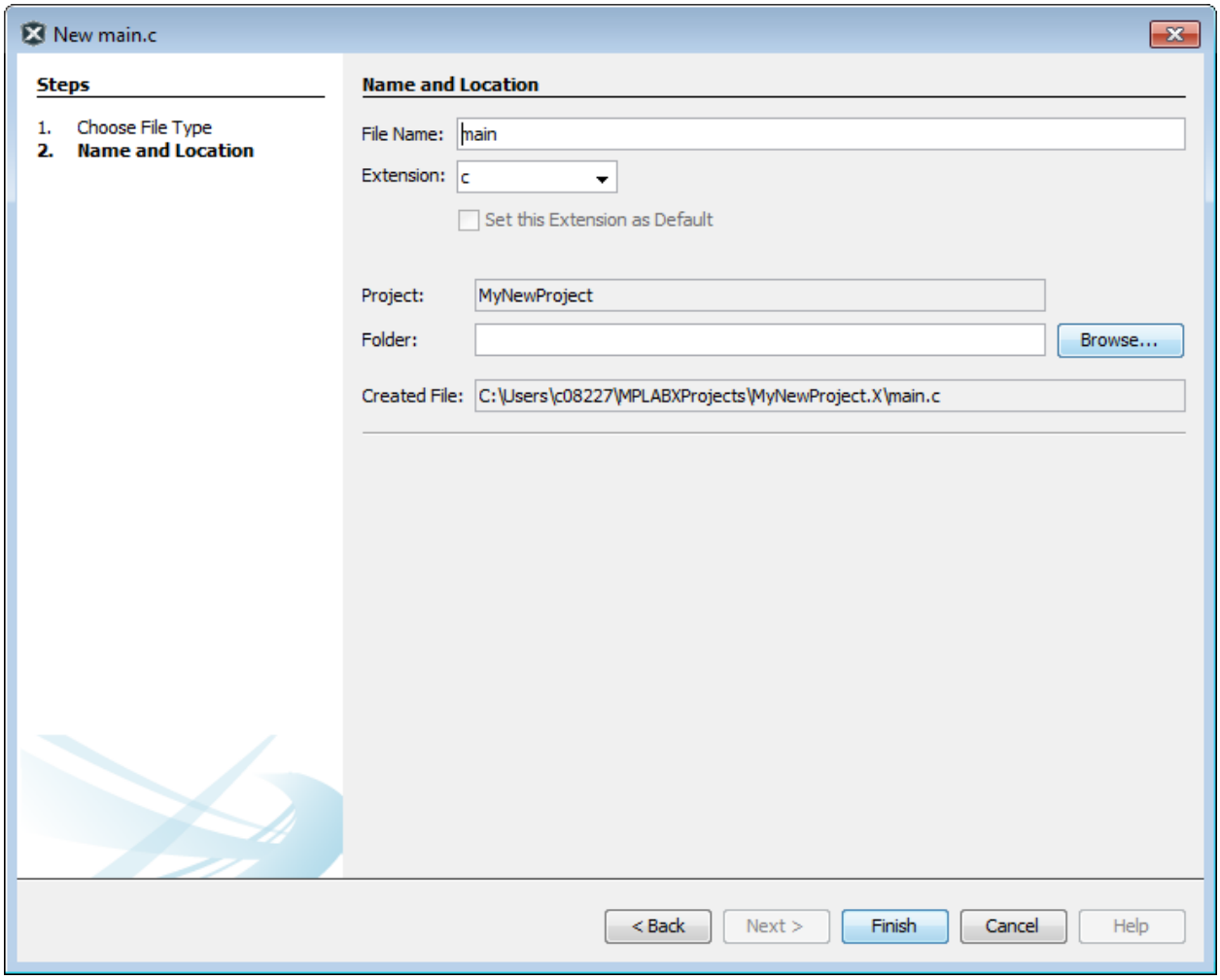
图 4-15. 文件向导——选择类别和类型



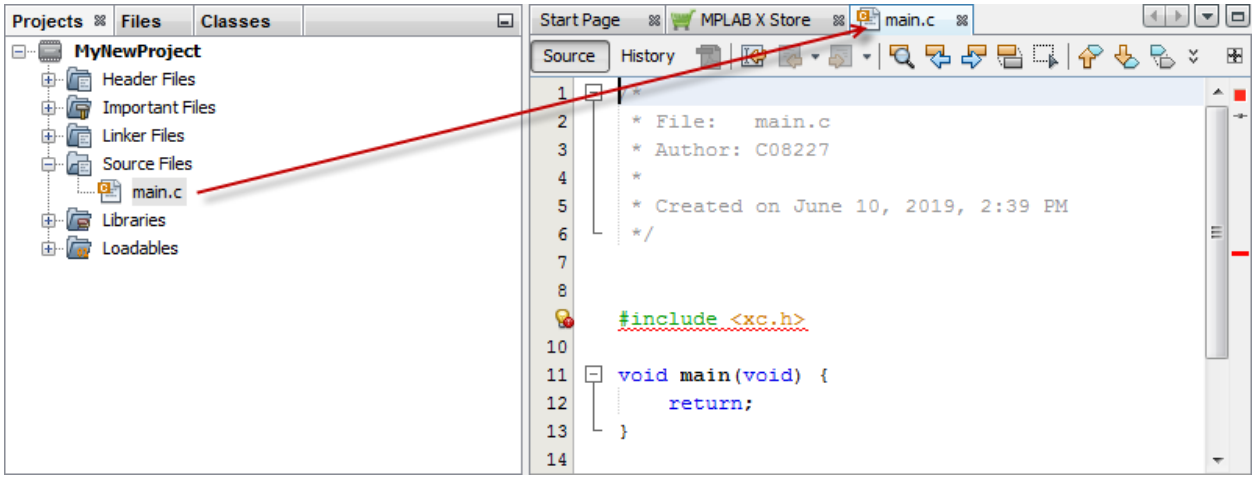
步骤 2.名称和位置

为文件命名并将其置于项目文件夹中。单击 **Finish**。

图 4-16. 文件向导——选择关联项目



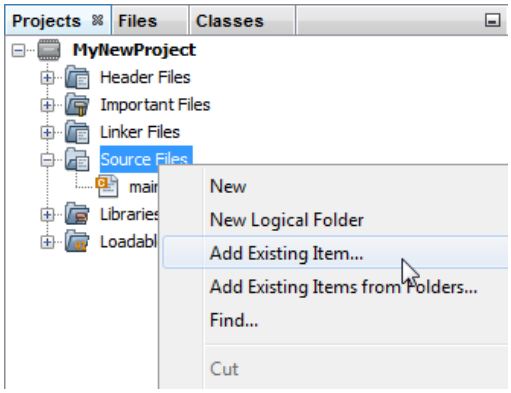
将显示包含文件名称的编辑器选项卡。在该选项卡下输入文件内容。文件中的文本将根据文件类型进行语法着色标记。



4.9.3 将现有文件添加到项目中

可以通过执行以下其中一项操作，将现有文件添加到项目中：

- 右键单击 Project/File（项目/文件）窗口中的项目并选择“Add Existing Item”（添加现有项）。添加后，可以根据需要将文件拖放到文件夹中。
- 右键单击 Project/File 窗口中的一个文件夹（例如，“Source Files”），然后选择“Add Existing Item”。



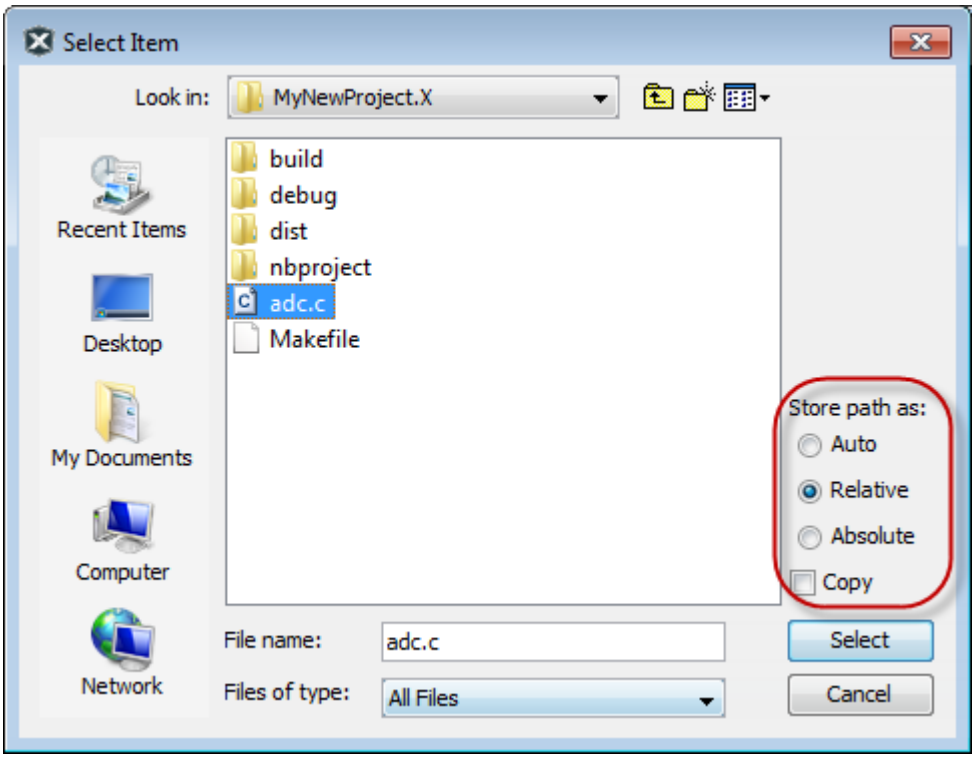
4.9.3.1 添加项目文件夹中的文件

添加位于主项目文件夹下的文件时，可以选择是否采用下列方式添加：

- 自动——让 MPLAB X IDE 决定如何将文件放在最合适的位置
- 相对——指定相对于项目的文件位置（最易移植的项目）
- 绝对——通过绝对路径指定文件位置
- 复制——将指定文件复制到项目文件夹

选择路径模式后，MPLAB X IDE 将记住该设置。要进行更改，请参见 12.16.2 Project Options 选项卡下的“File Path Mode”（文件路径模式）。

文件将出现在 Projects 窗口中。还将显示带有文件名称的编辑器选项卡。



4.9.3.2 添加项目文件夹外的文件

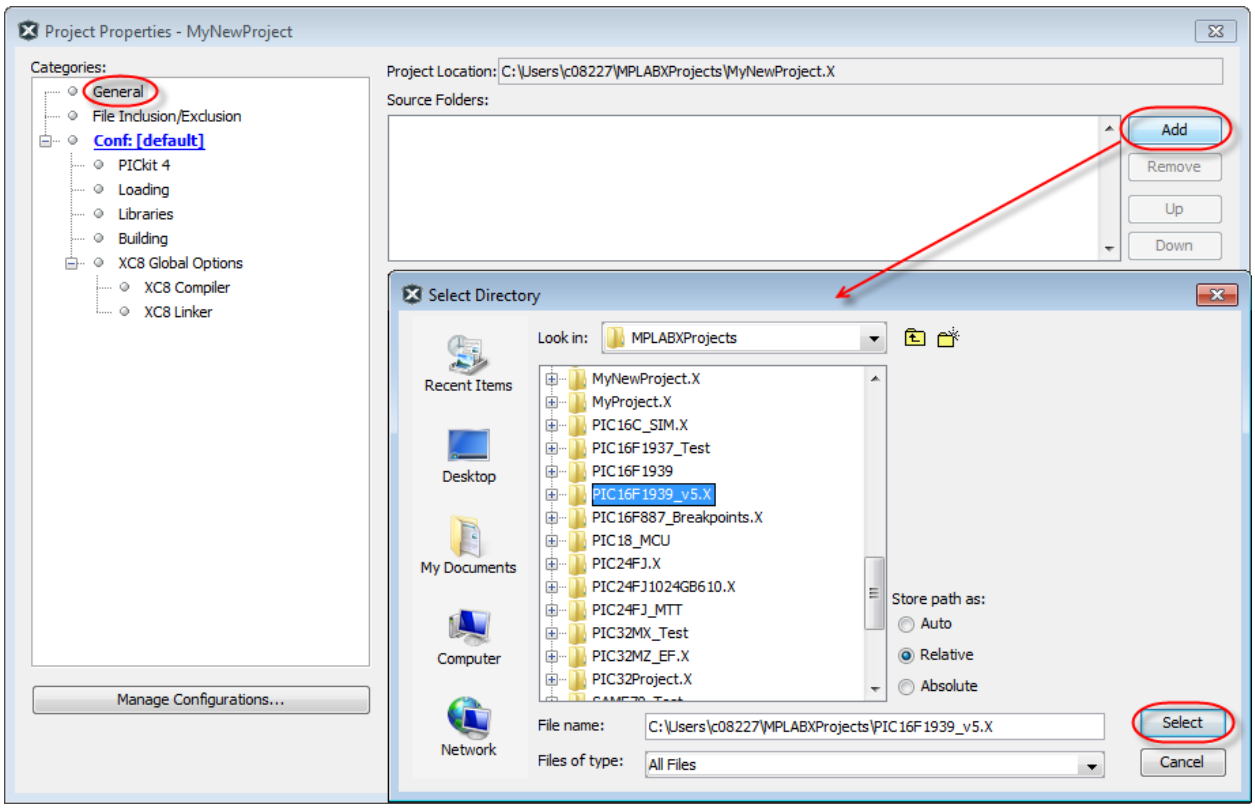
添加不在项目文件夹中的源文件时，采用“Relative”（相对）方式添加该文件（存储路径）。该操作将创建一个外部文件夹（_ext），这样项目就可以在编译时找到文件。

要利用//TODO 操作项和文件上下文菜单等导航功能，必须将文件所在位置告知项目。为此：

1. 在 Projects 窗口中，右键单击项目名称，并选择“Properties”。
2. 在“Categories”下，单击“General”（常规）。
3. 单击“Source Folders”（源文件夹）旁的 Add。
4. 浏览到已添加到项目中的外部文件的路径。单击 **Select**（选择）按钮。
5. 单击 **Apply**（应用）或 **OK**。然后重新编译项目。

导入 MPLAB IDE v8 项目时，源文件不在项目文件夹中。可使用 *File>Import>MPLAB IDE v8 Project*（文件>导入>MPLAB IDE v8 项目）自动导入文件。

图 4-17. 指定外部文件路径



4.9.4 编辑文件中的代码

要创建新代码或更改现有代码，可使用 NetBeans 编辑器。关于编辑器的更多信息，请参见 NetBeans 帮助主题：

https://netbeans.org/features/java/editor.html

另请参见有关该编辑器的 MPLAB X IDE 相关信息：

7. 编辑器

4.9.5 添加和设置库与目标文件

可以在以下位置引用链接器要使用的库文件：

- Projects 窗口中的 Libraries（库）文件夹
- Project Properties 窗口

可以在语言工具库管理器中完成其他库文件设置。

4.9.5.1 Libraries 文件夹

在 Projects 窗口中，右键单击 Libraries 文件夹可看到以下选项：

- Add Library Project（添加库项目）
  make 一个项目，将其生成的库作为项目所需输出，即建立依赖关系。更多相关信息，请参见 NetBeans 帮助主题：

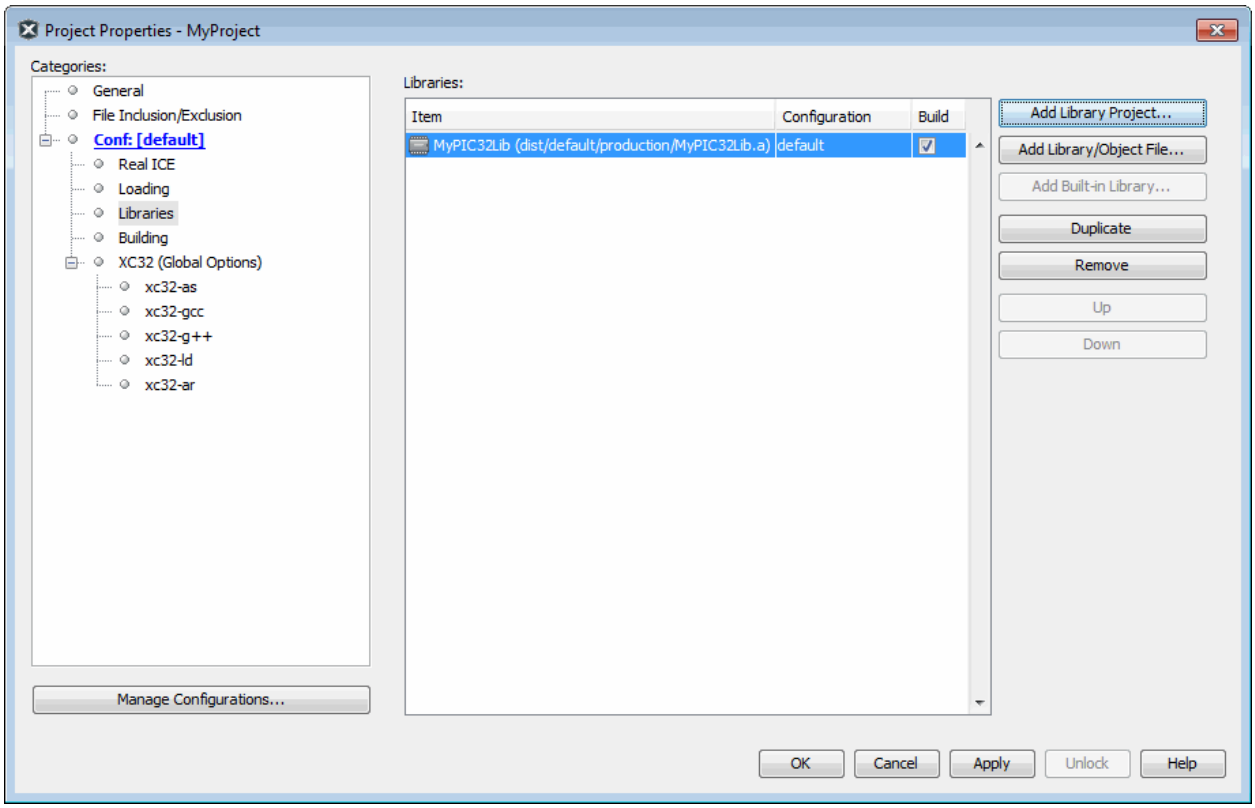
  https://netbeans.org/kb/73/java/project-setup.html#projects-dependencies。
- Add Library/Object File（添加库/目标文件）
  将现有的库文件或预编译（目标）文件添加到项目中。
- Properties
  打开项目的 Properties 窗口可选择附加选项。

4.9.5.2 Project Properties 窗口：Libraries 类别

打开 Project Properties 窗口，然后单击“Libraries”类别。添加到 Projects 窗口中 Libraries 文件夹中的所有文件都将在此处显示。可以使用右侧的按钮添加其他文件并管理这些文件。

库列表中文件的顺序决定链接顺序，这在项目中包含的库相互引用时很重要。引用二级库中对象的库应放在链接顺序列表中的较高位置。链接顺序出错可能会导致链接期间出现“未定义引用”错误。关于链接顺序的详细信息，请参见语言工具文档。

图 4-18. 管理库/目标文件和设置链接顺序

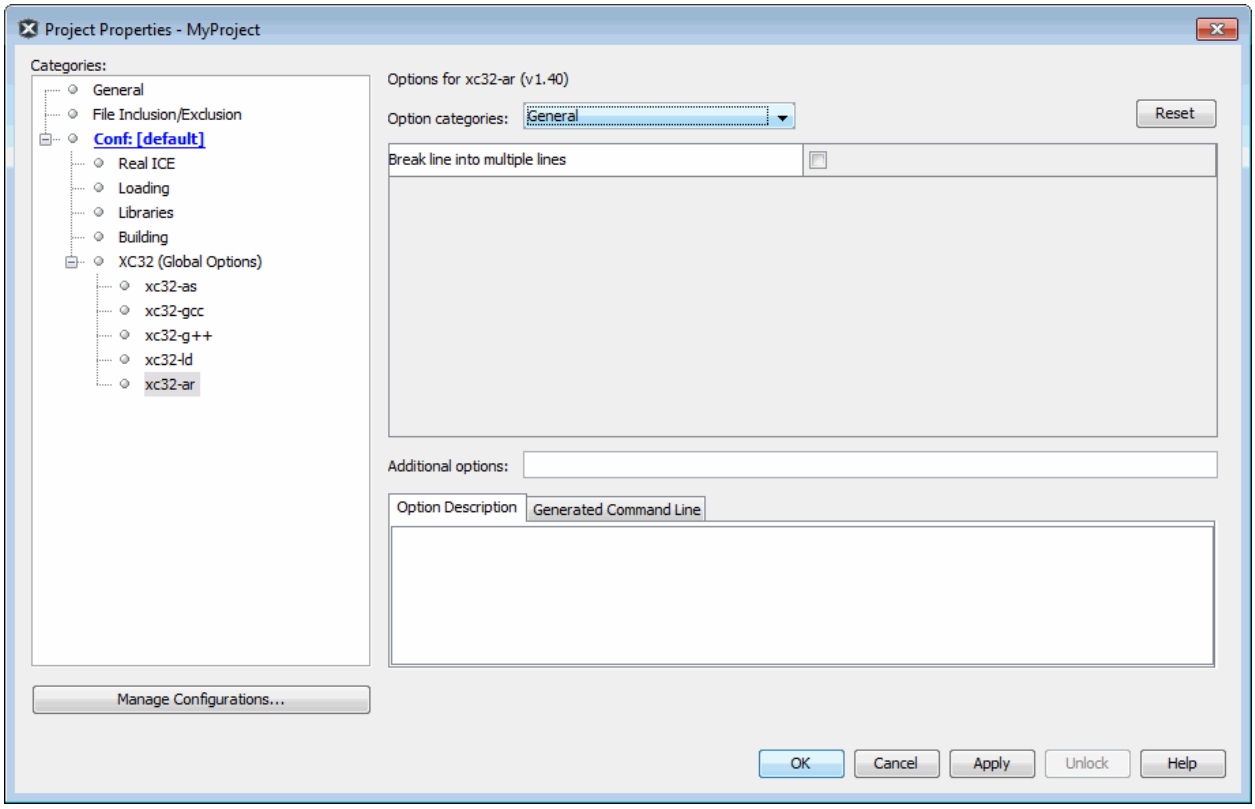


4.9.5.3 Project Properties 窗口：Librarian 类别

打开 Project Properties 窗口，然后单击语言工具类别下的库管理器/归档器。请查阅语言工具文档，确定库管理器的可执行文件名称。

在左窗格中，单击链接器的名称。然后，在右窗格中选择选项“Libraries”。从该列表中选择库选项。

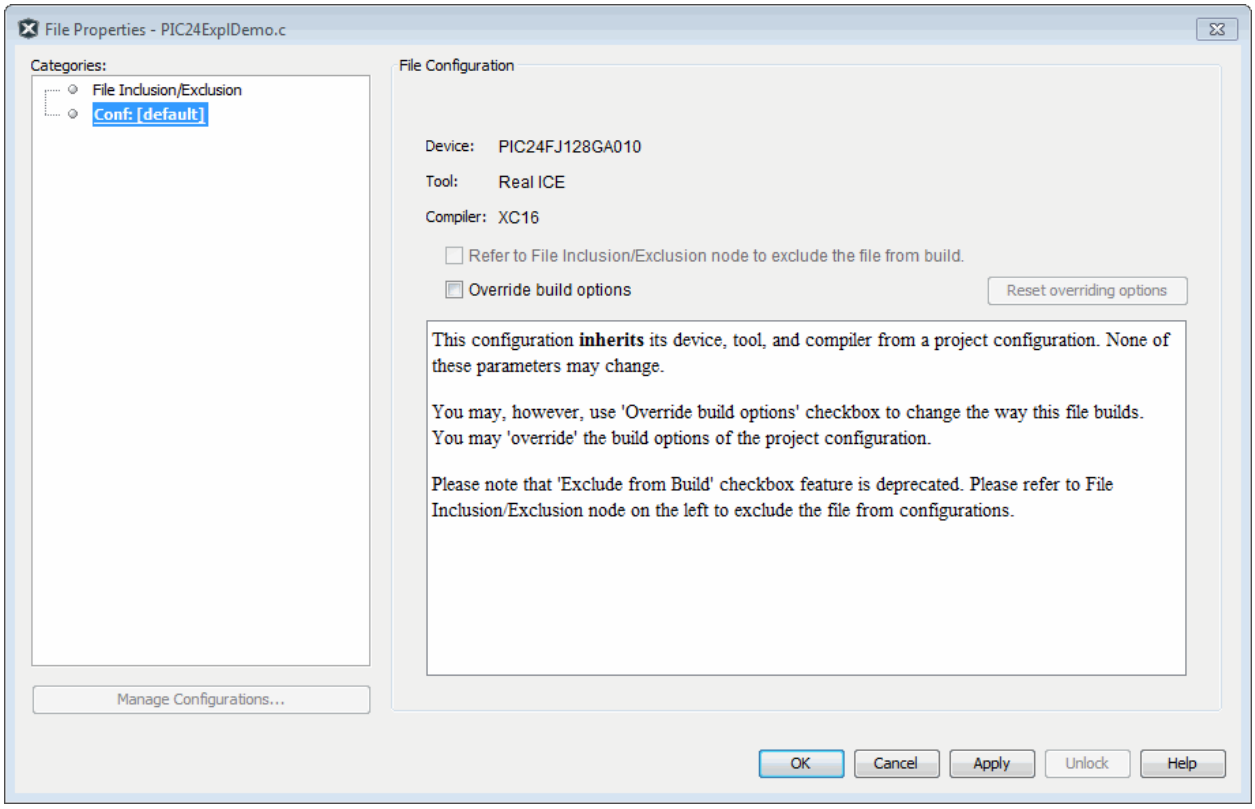
图 4-19. 在库管理器中设置库选项



4.9.6 设置文件和文件夹属性

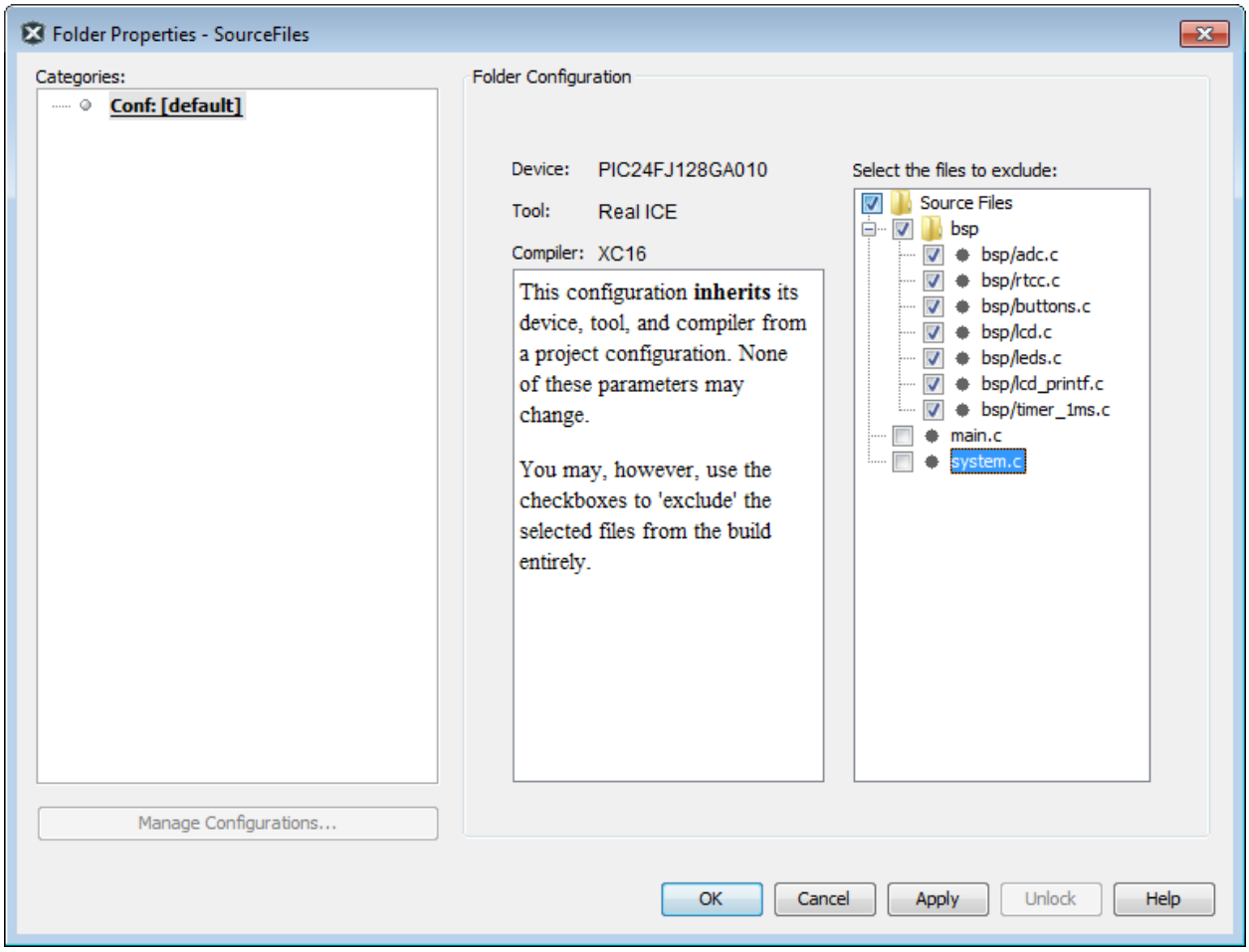
可使用 File Properties（文件属性）对话框以不同于项目中其他文件的方式编译项目文件。在 Projects 窗口中右键单击文件名称并选择“Properties”，可访问该对话框。可以通过选择从项目编译中排除文件，也可以使用此处选择的选项来覆盖项目编译选项。要覆盖编译选项，应选中复选框，然后单击“Apply”以查看“Categories”下显示的选定选项。

图 4-20. 文件属性



通过在 Projects 窗口中右键单击文件夹名称，然后选择“Properties”，可以从编译过程中排除整个（虚拟）文件夹。选择从编译中排除文件夹或各个文件。选择其他文件夹的上层文件夹时，将选择该文件夹中的全部内容。您可以根据需要取消选择文件或其他文件夹。

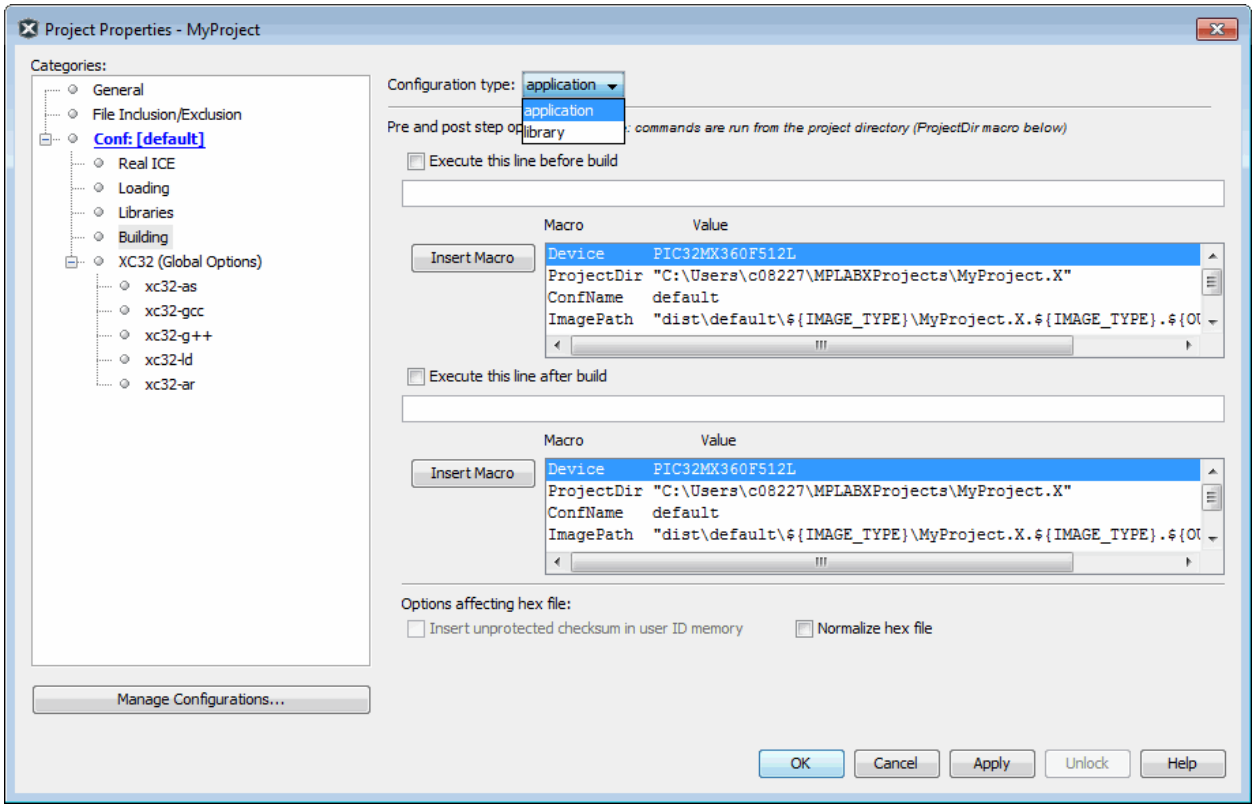
图 4-21. 文件夹属性



4.10 设置编译属性

在编译项目之前，可以在 Project Properties 窗口的 Building（编译）类别中设置其他编译属性。

图 4-22. 编译属性



4.10.1 更改项目配置类型

自 MPLAB X IDE v2.15 起，在 New Project 向导中创建独立项目或库项目时，所创建项目的配置类型分别为“application”（应用程序）或“library”（库）。这意味着可以根据需要切换现有项目的配置类型，以下部分中指定了注意事项。

预编译项目和用户 Makefile 项目不受该功能的影响（即，这些项目一旦创建，便不能更改为其他类型）。

从独立项目切换为库项目

将项目配置类型从“application”切换为“library”时，需自行删除 `main()`。此外，一旦配置类型为“library”，将不会编译/链接“application”项目中的任何可加载项。关于可加载项的更多信息，请参见 5.5 可装入项目、文件和符号。

如果工具链（即 XC8）不支持库，则组合框将被禁止，您无法将配置切换为“library”。

从库项目切换为独立项目

将项目配置类型从“library”切换为“application”时，需自行添加 `main()`。

4.10.2 Execute this line before/after the build

输入要在编译过程最初或最后执行的命令。这些命令会被插入 `nbproject/Makefile-$CONF.mk` 文件，使您可以定制编译过程。如果需要在要调用的脚本或程序中引用一些与项目相关的项（如映像名称），请使用所提供的宏。

您可以自己输入宏，也可以单击 **Insert Macro**（插入宏）将宏名称复制到编辑框中的当前位置。命令将在 make 过程中运行，而当前目录设置为 MPLAB X IDE 项目目录。项目目录定义为包含 `nbproject` 文件夹的项目。

表 4-6. 宏

名称*	功能
Device	当前选定项目配置针对的器件

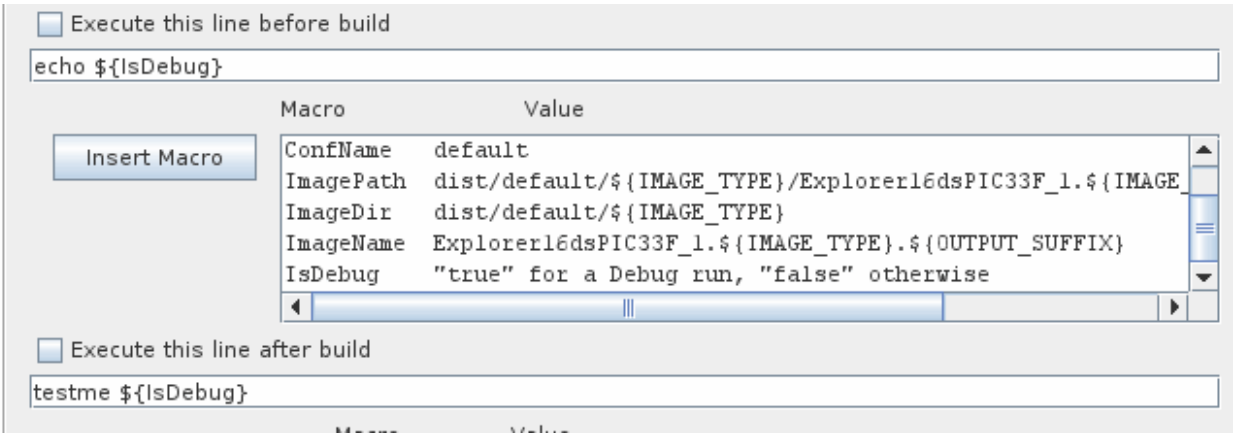
.....（续）

名称*	功能
ProjectDir	项目文件在 PC 上的位置
ConfName	当前选定项目配置的名称
ImagePath	编译映像的路径
ImageDir	包含编译映像的目录
ImageName	编译映像的名称
IsDebug	对于调试模式为“true”；否则为“false”
*可能会有其他宏可用，具体取决于所用的编译器。	

示例：仅在调试期间执行某个过程

在下面所示的窗口中，单击“IsDebug”宏来选择它，然后单击 **Insert Macro** 将其插入脚本。

图 4-23. 编译属性——编译前/后



检查运行的脚本中的`$1`（Linux 或 Mac 操作系统）或`%1`（Windows 操作系统）的值。

Bash 代码示例

```
#!/bin/bash
echo is $1
if [ "$1" == "true" ]; then
echo We are in debug
else
echo We are in production
fi
```

批处理代码示例

```
@echo off
if "%1" == "true" goto debug
:production
@echo We are in production
goto end
:debug
@echo We are in debug
:end
```

4.10.3 Insert unprotected checksum in user ID memory（在用户 ID 存储区中插入未受保护的校验和）

如果器件支持该功能（未呈灰显状态），则单击复选框时将使用编译后生成的校验和数字作为用户 ID。

对于 PIC32 系列器件，校验和取决于用户 ID。因此，PIC32 器件不支持该功能。

4.10.4 Normalize hex file

当行地址按递增顺序放置，且数据尽可能缓冲为一种大小（16 字节）时，即表示十六进制文件进行了规范化处理。

例如，以下行（记录）来自文件 `myfile.hex`，该文件是 MPLAB XC16 C 编译器/链接器的输出：

```
:0801f800361e0000361e000057
:020000040000fa
:10020800361e0000361e0000361e0000361e000096
```

当该文件进行规范化之后，文件数据的形式如下：

```
:10022800361E0000361E0000361E0000361E000076
:10023800361E0000361E0000361E0000361E000066
:10024800361E0000361E0000361E0000361E000056
```

其格式为：

```
:bbaaaarrdd...ddcc
```

其中：

:	起始记录
bb	字节计数
aaaa	地址
rr	记录类型
dd	数据
cc	校验和

在 MPLAB X IDE 中，使用应用程序 Hexmate 来对 Intel 十六进制文件进行规范化处理。选中“Normalize hex file”（规范化十六进制文件）选项时，在编译后会调用以下命令：

```
hexmate myfile.hex -omyfile.hex
```

实际上，Hexmate 会将整个十六进制文件解包，并将数据放置在由十六进制文件指定的地址处。然后，它会将产生的存储器映像重新打包为一个新的十六进制文件。在产生的文件中，所有数据都按升序放置，并且是连续的。如果地址中存在空隙，则输出文件中也会存在空隙（不会填充未使用的地址）。

关于使用 Hexmate 的更多信息，请参见所安装的 MPLAB XC8 C 编译器的 `docs` 文件夹中的《MPLAB® XC8 C 编译器用户指南》（DS50002053D_CN）。尽管 Hexmate 在 MPLAB XC8 文档中介绍，但它可以与任何编译器一起使用。

规范化的十六进制文件对于通过串行连接烧写代码（如自举程序）非常有用，因为这最大程度减小了记录字节的差异。

4.11 编译项目

对于 MPLAB X IDE，不需要先编译项目再运行或调试。编译是运行和调试过程的一部分。但是，对于初始开发或重要更改，可能需要确保在尝试运行或调试之前先进行项目编译。

要编译项目：

- 在 Projects 窗口中，右键单击项目名称，并选择“Build”（编译）。您还可以选择“Clean and Build”（清除并编译）在编译之前删除中间文件。

- 单击“Build Project”（编译项目）或“Clean and Build Project”（清除并编译项目）工具栏图标。

Output 窗口中将会显示编译进度。

可用的编译功能如下：

表 4-7. 工具栏按钮上的编译选项

按钮图标	功能	说明
	编译项目	make 项目中的所有文件。
	针对调试而编译	make 项目中的所有文件，并在编译映像中添加调试执行程序。
	使用专业版比较进行编译	如果在免费模式下使用 MPLAB XC C 编译器，则也可以在专业版模式下进行编译，存在差异时还能查看差异的比较输出结果。请参见 5.25 比较 MPLAB XC 编译器免费版与专业版许可证。
	清除并编译	删除先前的编译文件，并 make 项目中的所有文件。
	针对调试进行清除和编译	删除先前的编译文件，并 make 项目中的所有文件。向编译映像添加调试执行程序。
	清除并使用专业版比较进行编译	删除先前的编译文件，然后使用专业版比较进行编译。

要在 Output 窗口中查看错误：

1. 在 Output 窗口中单击右键并选择“Filter”（筛选器）。
2. 在 Filter 对话框中，选中“Match Case”（区分大小写），然后输入“: error”，从而在 Output 窗口中仅显示导致停止编译的错误。
3. 使用<Ctrl>+<G>可开启和关闭筛选器。

关于错误的讨论，请参见语言工具文档。

要查看校验和信息：

打开 Dashboard 窗口（见 5.19 查看仪表板显示）可查看编译后的校验和。

4.12 运行代码

成功编译代码之后，可以运行应用程序。

运行如何工作

单击“Run Project”工具栏图标  时，将发生以下事件：

- 如果 make 过程确定需要进行 build，则进行 build。
- 对于在线调试器/仿真器和编程器，将自动使用映像（不带调试执行程序）对目标器件进行编程，然后将释放器件使之运行，即，将不会使能任何断点或其他调试功能。

  注： 要在编程后将器件保持在复位状态，请单击“Hold in Reset”图标  而不是“Run Project”。
- 对于软件模拟器，应用程序将只是执行，没有调试功能。

Output 窗口中将会显示运行进度。

运行应用程序代码

1. 在 Projects 窗口中，选择项目或将其设为主项目（右键单击项目并选择“Set as Main Project”）。
2. 通过以下任一方式运行程序：

- 单击“Run Program”（运行程序）图标  。
- 单击“Make and Program Device”（Make 并编程器件）图标  。

Output 窗口中将会显示运行进度。

4.12.1 运行注意事项

运行程序时，需要注意 MPLAB X IDE 在运行时（Run 或 Debug）连接至硬件工具。因此，在 MPLAB X IDE 中进行的设置只会在运行时被传递给该工具。只有运行时再次启动时，才会更新在暂停期间更改的设置。

要总是连接到硬件工具（与 MPLAB IDE v8 中一样），请参见 *Tools>Options*（对于 macOS 为 *MPLAB X IDE>Preferences*），**Embedded** 按钮，**Generic Settings**（通用设置）选项卡，“Maintain active connection to hardware tool”（保持硬件工具的连接处于活动状态）复选框。

另请参见 4.20 对器件编程。

4.13 调试代码

如果代码未成功运行，则应调试它。

调试如何工作

单击“Debug Project”工具栏图标  时，将发生以下事件：

- 如果 make 过程确定需要进行 build，则进行 build。
- 对于在线调试器/仿真器，将自动使用映像（包括调试执行程序）对目标器件或调试头进行编程并将启动调试会话。
- 对于软件模拟器，将会启动调试会话。

Output 窗口中将会显示调试进度。

调试应用程序代码

要调试应用程序代码：

1. 在 Projects 窗口中，选择项目或将其设为主项目（右键单击项目并选择“Set as Main Project”）。
2. 单击“Debug Project”图标  或“Step Into”图标开始调试。

要暂停应用程序代码：

单击“Pause”图标  暂停程序执行。

要再次运行代码：

单击“Continue”图标  再次启动程序执行。

要结束代码的执行：

单击“Finish Debugger Session”图标  结束程序执行。

要启动调试器：

如果代码是针对调试而编译的，并且只是希望启动调试工具，则可以通过选择“Debug Project”图标旁边的向下箭头并选择“Launch Debugger”（启动调试器）来实现。

表 4-8. 调试图标

图标	功能	相关菜单项
	调试项目	Debug>Debug Project
	单步进入	Debug>Step Into
	暂停	Debug>Pause
	继续	Debug>Continue
	完成调试会话	Debug>Finish Debugger Session

4.13.1 生成的调试宏

MPLAB X IDE 会生成一些调试宏，供 Microchip 语言工具使用。下表列出了传递给 Microchip 编译器和汇编器的宏。

表 4-9. Microchip 工具调试宏

宏名称	关联工具	功能
__DEBUG	所有	指定这是调试编译
__MPLAB_REAL_ICE__ __MPLAB_ICD3__ __MPLAB_PK3__ __MPLAB_PICKIT2__	XC8	指定使用的硬件调试工具 格式为__MPLAB_xxx__，其中的 xxx 代表硬件工具说明符。
__MPLAB_DEBUGGER_REAL_ICE __MPLAB_DEBUGGER_ICD3 __MPLAB_DEBUGGER_PK3 __MPLAB_DEBUGGER_PICKIT2	XC16、XC32 和 MPASM	指定使用的硬件调试工具 格式为__MPLAB_DEBUGGER_xxx，其中的 xxx 代表硬件工具说明符。
__MPLAB_DEBUGGER_PIC32MXSK	XC32	指定使用的入门工具包

您可以在自己的代码中使用这些宏。例如：

```
#ifdef __DEBUG
fprintf(stderr,"This is a debugging message\n");
#endif
```

4.13.2 调试注意事项

调试代码时，请考虑以下问题：

- 只有处于调试会话（调试模式）下时，才能激活许多调试功能——例如，在观察或存储器窗口中查看变量的值。
- MPLAB X IDE 操作会始终连接到硬件工具。要仅在运行时连接，请参见 *Tools>Options*（对于 macOS 为 *MPLAB X IDE>Preferences*），**Embedded** 按钮，**Generic Settings** 选项卡，然后取消选中“Maintain active connection to hardware tool”。
  取消选中后，MPLAB X IDE 中进行的设置只会在运行时被传递给该工具。只有运行时再次启动时，才会更新在暂停期间更改的设置。
- 对于某些应用，可能需要分解调试操作，独立执行调试步骤。为此，请使用 *Debug>Discrete Debugger Operation*（调试>分解调试器操作）下的步骤。

- 调试时禁止编译器优化，或使用“调试”优化（如果您使用的编译器具有此功能）。在 Project Properties 窗口中的“Categories”下，选择语言工具可执行文件。然后从“Options categories”（选项类别）中选择“Optimizations”（优化）并选择最低级别，即“0”（如果可用）。

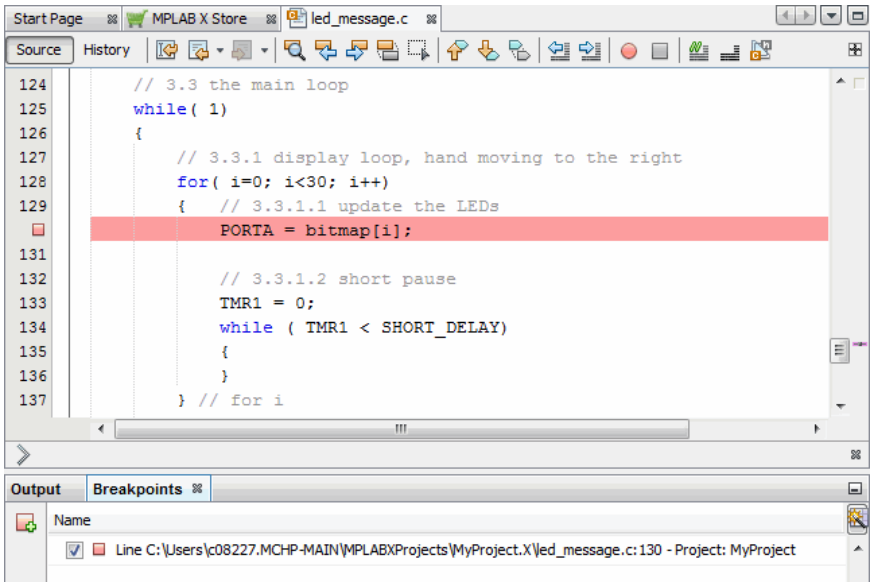
4.14 使用断点控制程序执行

在调试代码时，通过在代码中特定位置暂停执行来检查变量值的功能会很有用。要执行该操作，请使用断点。

关于断点的更多信息，请参见 NetBeans 帮助主题：

https://netbeans.org/kb/docs/cnd/debugging.html#breakpoints

图 4-24. 断点和 Breakpoints 窗口



相关信息

12.2 Breakpoints 窗口

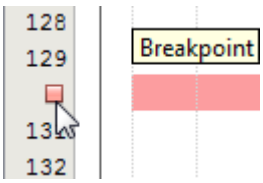
12.3 New Breakpoint 对话框

4.14.1 设置或清除简单行断点

要在某个代码行上设置或清除断点，请执行以下操作之一：

- 在源代码编辑器中单击代码行的左边缘（见图）
- 单击代码行，然后按下 Ctrl+F8
- 选择 *Debug>Toggle Line Breakpoint*（调试>翻转行断点）

图 4-25. 单击以使能/禁止断点



4.14.2 使用 Breakpoint 对话框设置断点

要打开 New Breakpoint（新建断点）对话框，请完成以下步骤：

1. 单击 Breakpoints（断点）窗口左上角的“Create New Breakpoint”（新建断点）图标  。

2. 选择 *Debug>New Breakpoint*（调试>新建断点）。选择断点类型并设置选项。

表 4-10. 断点类型

选择	说明
Line（行）	在编辑器窗口中的某个代码行中断。
Data（数据）	在读/写数据（RAM）存储器中的某个地址时中断。其中包括 SFR 和符号存储单元。
Address（地址）	在执行程序（闪存）存储器中的某个地址时中断。
Event（事件）	在发生特定事件（例如，复位、休眠/唤醒和看门狗定时器超时等）时中断。
Function（函数）	在进入指定函数时中断。

关于每种断点类型可用的选项，请参见 12.3 New Breakpoint 对话框。

4.14.3 在 Breakpoints 窗口中设置或清除断点

要在 Breakpoints 窗口中查看和翻转断点，请完成以下步骤：

1. 通过选中/取消选中复选框来翻转断点。
2. 选择 *Window>Debugging>Breakpoints*（窗口>调试>断点）。

4.14.4 设置断点序列（依赖于器件）

断点序列是断点的列表，将一直执行直到执行最后一个断点之后才会暂停执行。当存在到特定指令的多个执行路径，并且只希望执行一个特定路径时，顺序断点会很有用。

要**创建断点序列**，请完成以下步骤：

1. 右键单击某个现有断点或按 Shift 并单击选择一组现有断点，然后右键单击该组。
2. 从弹出菜单中，转到“Complex Breakpoint”（复杂断点）并选择“Add a New Sequence”（添加新序列）。
3. 在对话框中输入序列的名称，然后单击 OK。
4. 断点会出现在新序列下。
5. 要向序列中添加更多现有断点，请右键单击断点并选择 *Complex Breakpoint>Add to Name*（复杂断点>添加至 Name），其中的 *Name* 是序列的名称。
6. 要向序列中添加新断点，请右键单击序列并选择“New Breakpoint”。

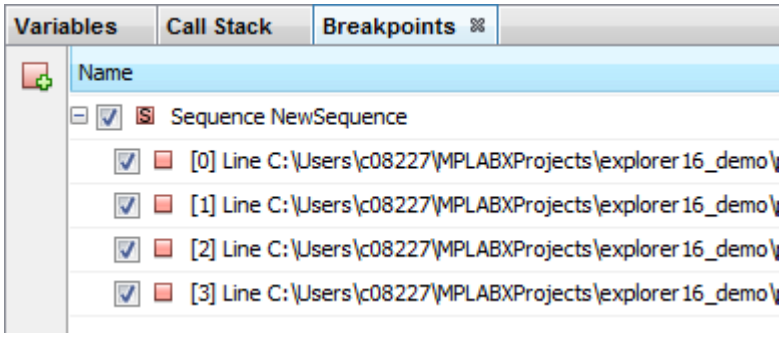
要**选择序列顺序**，请完成以下步骤：

1. 展开序列来查看所有项。
2. 右键单击某项，然后选择 *Complex Breakpoints>Move Up/Complex Breakpoints>Move Down*（复杂断点>上移/复杂断点>下移）。断点的执行顺序为从下到上；序列中的最后一个断点最先执行。

要**删除序列**或断点，请执行以下任一操作：

- 右键单击相应的项并选择“Disable”（禁止），可临时删除该项。
- 右键单击相应的项并选择“Delete”（删除），可永久删除该项。

图 4-26. 断点序列示例



4.14.5 设置断点元组（依赖于器件）

对于 MPLAB X IDE，一个元组代表断点的逻辑与列表。如果某个变量会在多个位置发生修改，而您只需在变量于一个特定位置发生修改时中断代码执行，逻辑与断点会很有用。

只能对两个断点进行逻辑与，并且它们必须包括一个程序存储器断点和一个数据存储器断点。断点 1 和断点 2 必须同时发生，程序才会暂停。

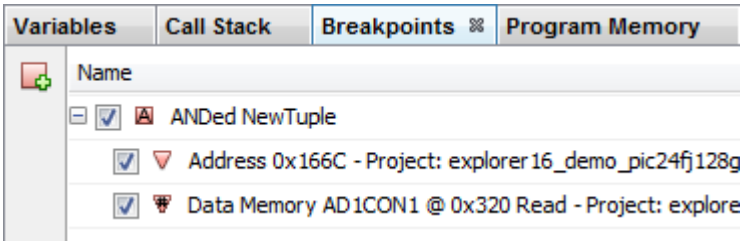
要**创建断点元组**：

1. 单击 Breakpoints 窗口左上角的“Create New Breakpoint”图标  来打开 New Breakpoint 对话框。
2. 创建地址断点。单击 **OK** 将其添加到 Breakpoints 窗口中。
3. 重复步骤 1 和步骤 2 来创建一个数据断点。
4. 右键单击一个断点，然后选择 *Complex Breakpoint>Add to New Tuple*（复杂断点>添加至新元组）。
5. 在对话框中输入元组名称并单击 **OK**。
6. 断点会出现在新元组下。
7. 在另一个断点上单击右键，并选择 *Complex Breakpoint>Move to Name*（复杂断点>移至 Name），其的中 *Name* 是元组的名称。

要**删除元组**或断点，请执行以下操作之一：

- 右键单击相应的项并选择“Disable”，可临时删除该项。
- 右键单击相应的项并选择“Delete”，可永久删除该项。

图 4-27. 断点元组示例



4.14.6 查看断点地址

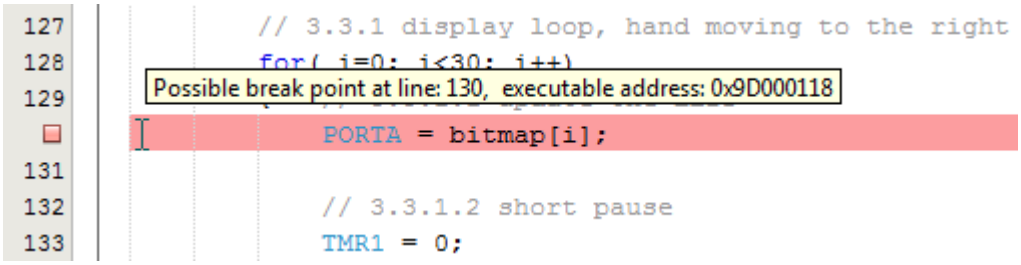
要在调试会话之前和期间查看断点的地址，请完成以下步骤：

1. 在 *Tools>Options*，**Embedded** 按钮，**Generic Settings** 选项卡下，为“On mouseover source lines in editor, evaluate break point status”（鼠标悬停在编辑器中的源代码行上时，对断点状态求值）选择具体鼠标按键操作。
2. 设置一个行断点。
3. 单击 Debug Project 图标以调试代码。
4. 将光标放在断点图标和源代码开头之间，然后执行步骤 1 所选的鼠标按键操作以查看鼠标悬停时的信息。

单个断点地址

在许多情况下，您会看到单个地址。打开程序/执行存储器窗口（*Window>Target Memory Views*）可查看存储器地址和该地址的说明。

图 4-28. 查看断点地址

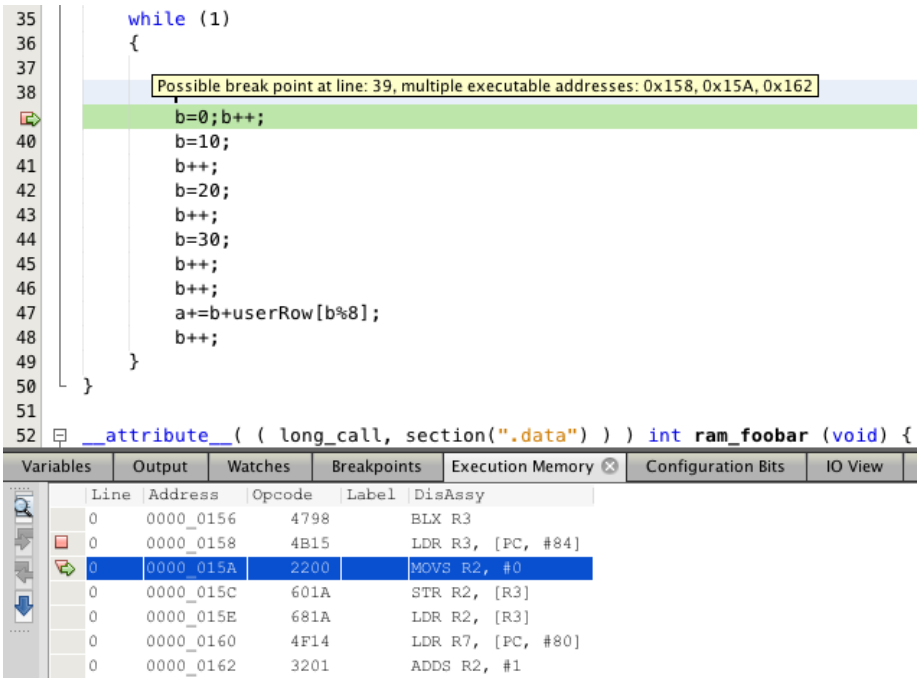


多个断点地址

在某些情况下，您将看到多个可能的断点地址。原因可能是提高了编译器优化程度（-O）。在 MPLAB X IDE v5.20 之前的版本中，仅显示第一个地址。而现在可显示所有地址。

打开程序/执行存储器窗口（*Window>Target Memory Views*）可查看每个存储器地址和该地址的说明。如果断点不起作用，则可以在存储器窗口中选择其他地址。当断点命中时，它仍将在代码中以相同的行号显示。

图 4-29. 查看多个断点地址



4.14.7 硬件断点使用

如果必须在某个代码行暂停，需要：

- 从软件断点切换到硬件断点
- 设置硬件断点

请注意，调试将导致在同一代码行暂停。再次调试可继续执行程序，或者执行单步来执行断点所在行的代码。

4.14.8 双分区器件和断点

由于调试的性质，一旦调试器在活动分区中暂停，便必须在活动分区中设置双分区器件断点。因此，从分区 1 开始，无法在分区 2 中设置断点。您需要运行到切换分区为止，切换到分区 2 后设置位于分区 2 中的断点。

4.14.9 断点应用

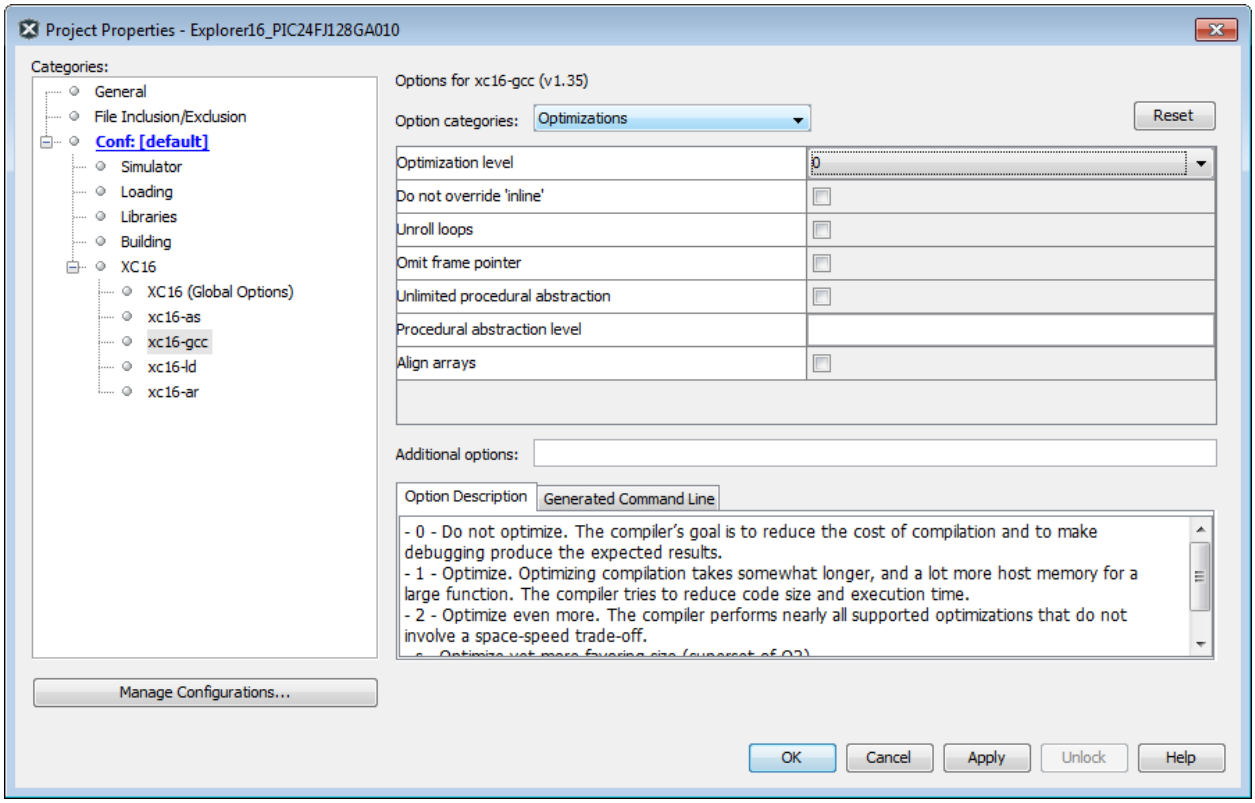
要确定两个断点之间的时序，请使用跑表（见 5.15 使用跑表）。

要确定断点资源，请打开 Dashboard 窗口（见 5.19 查看仪表板显示），以查看可用和已用断点数，以及是否支持软件断点。

4.14.10 编译器优化和断点

编译器优化可能改变程序的执行方式。这会影响断点的存储器地址映射，从而产生多种地址可能性。调试代码时，建议使用最低级别的优化以避免出现断点问题。

图 4-30. 最低优化级别示例



4.14.11 断点资源使用

硬件调试器（例如，在线调试工具）支持有限数量的断点。可用断点数量取决于选定的器件。要查看可用断点的数量和跟踪已用断点的数量，请参见 Dashboard 窗口（5.19 查看仪表板显示）。

以下 MPLAB X IDE 功能使用断点来完成其功能。

	Step Over
	Step Out
	Run to Cursor
	Reset to Main（复位至主程序）

如果在没有断点可用时尝试使用这些功能之一，将会显示一个对话框，指示所有资源都已被使用。

4.14.12 断点感知

使用工具提示和图标，可以在调试会话内外显示断点信息：

- 当退出调试会话且未发生编译时，如果尝试设置断点，鼠标工具提示将显示蓝色撕裂图标，表明没有可用的调试信息。
- 当退出调试会话并进行了编译时，如果优化了一行，则可能支持断点，也可能不支持断点。
- 当处于调试会话中时，工具提示将显示是否可以设置断点。

如果不需要用于显示工具提示的选项，可以将其禁止（*Tools>Options*，**Embedded**，“Generic Settings”）。

4.14.13 撕裂断点图标

在某些情况下，断点图标会显示为撕裂的形式（）：

- 当退出调试会话且未发生编译时，如果尝试设置断点，鼠标工具提示将显示撕裂图标，表明没有可用的调试信息。
- 当处于调试会话中时，由于路径、文件或文件夹命名限制，可能无法找到源文件（请参见路径、文件和文件夹命名限制）。

4.15 单步执行代码

使用 Debug 菜单和调试工具栏上的单步功能之一，可从代码起始处或断点暂停之后单步执行代码。检查变量值的变化（见下一节）或确定程序流是否正确。

有几种方法可以单步执行代码：

	Step Over——执行程序的一行源代码。如果该行是一个函数调用，则执行整个函数，然后停止。
	Step Into——执行程序的一行源代码。如果该行是一个函数调用，则程序执行到该函数的第一条语句，然后停止。
	Step Out——执行程序的一行源代码。如果该行是一个函数调用，则执行函数，并将控制返回给调用方。
	Run to Cursor——运行当前项目，直到文件中的光标位置，并停止程序执行。

除了编辑器窗口之外，还可以在 Disassembly（反汇编）窗口（5.16 查看 Disassembly 窗口）和 Program Memory 窗口中单步执行代码。

4.16 观察符号值变化

在 Watches 窗口中观察选择更改的符号的值。在程序执行期间了解这些值是否为预期值可以帮助您调试代码。

可以添加到 Watches 窗口中的符号包括：

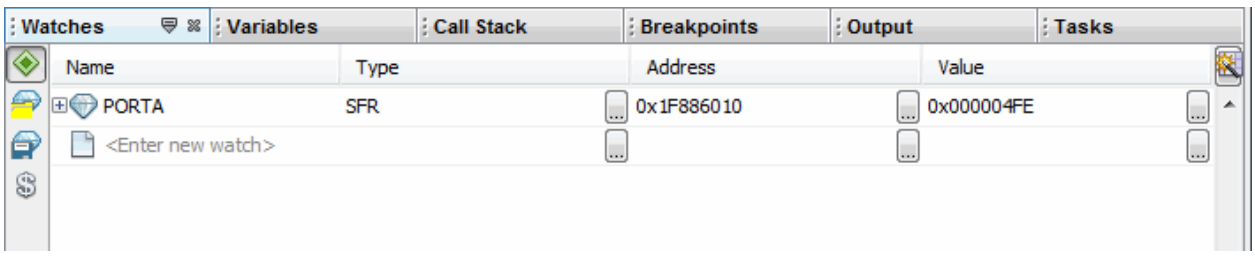
- 全局符号——在编译后可见
- SFR——特殊功能寄存器（依赖于器件）
- 绝对地址

一般情况下，只有在从调试中暂停时，才能看到更新后的值。但是，一些工具支持运行时更新（您可以在程序运行时看到值的变化）。要确定您的工具是否支持该功能，请参见工具文档。

对于除 PIC32 MCU 之外的所有器件，在运行时观察中使用的符号必须调整长度，使之与器件存储器匹配。也即，使用 8 位器件时需要 8 位符号。

对于 C 语言枚举类型，可以在窗口中输入枚举标号（文本）或整数值。对于标号，请注意它们是区分大小写的。

图 4-31. Watches 窗口——程序暂停



要查看 Watches 窗口，请执行以下操作之一：

- 选择 *Window>Debugging>Watches*（窗口>调试>观察）打开窗口。

- 如果 Output 窗口已打开，则在窗口中单击 Watches 选项卡。

要直接创建新的观察：

可以通过以下操作之一，向 Watches 窗口直接添加符号：

- 双击名称列，并输入一个全局符号、SFR 或绝对地址（0x300）。
- 在编辑器窗口中右键单击某个全局符号或 SFR，并选择“New Watch”。
- 在编辑器窗口选择全局符号或 SFR，并将其拖放到 Watches 窗口中。

要使用 New Watch 对话框创建新的观察：

可以通过以下操作之一，向 Watches 窗口添加符号或 SFR：

- 在 Watches 窗口中单击右键并选择“New Watch”或选择 *Debug>New Watch*。单击选择按钮可查看全局符号或 SFR。在列表中单击某个名称，然后单击 **OK**。
- 在编辑器窗口中选择符号或 SFR 名称，然后从右键菜单中选择“New Watch”。该名称将填充到窗口中。单击 **OK**。

要创建新的运行时观察：

向 Watches 窗口中添加运行时观察之前，需要先设置时钟：

1. 右键单击项目名称，并选择“Properties”。
2. 单击调试工具名称（例如，PICkit 4），并选择选项类别“Clock”（时钟）。
3. 设置运行时指令速度。

要添加全局符号或 SFR 作为运行时观察，请按照“要使用 New Watch 对话框创建新的观察：”下的说明操作，只是此时选择“New Runtime Watch”（新建运行时观察）而不是“New Watch”。

对于除 PIC32 MCU 之外的所有器件，在运行时观察中使用的符号必须调整长度，使之与器件存储器匹配。也即，使用 8 位器件时需要 8 位符号。

要查看符号变化：

1. 调试，然后暂停程序。
2. 单击 Watches 选项卡，使窗口处于活动状态。
3. 对于观察符号，继续调试会话并暂停可查看值的变化。对于运行时观察符号，继续调试并在程序执行时观察值的变化。

只有处于调试会话中，才能看到符号（全局符号、SFR、数组和寄存器位域等）的值。

注： 这种方式无法查看宏的值。在调试期间右键单击编辑器中的宏，使用 Macro Expansion（宏扩展）窗口。

要更改观察符号的基数：

右键单击该符号所在的行，并选择“Display Value As”（值的显示方式）。

要执行其他任务：

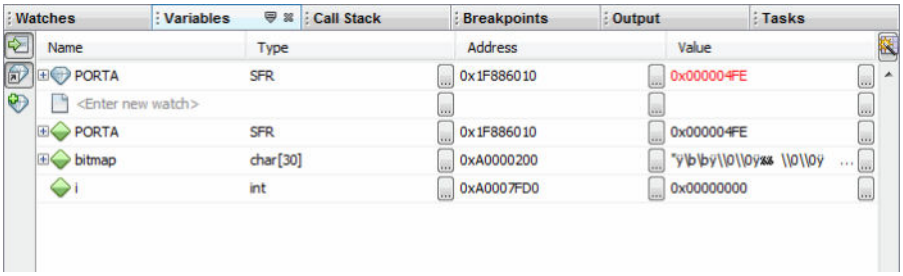
关于观察的更多信息，请参见 12.19 Watches 窗口。

4.17 观察局部变量值变化

在 Variables 窗口中观察局部变量值变化。在程序执行期间确定这些值是否为预期值可以帮助您调试代码。

一般情况下，只有在从调试中暂停时，才能看到更新后的值。但是，一些工具支持运行时更新。要确定您的工具是否支持该功能，请参见工具文档。

图 4-32. Variables 窗口——程序暂停



要查看 Variables 窗口，请执行以下操作之一：

- 选择 *Window>Debugging>Variables* 打开窗口。
- 如果 Output 窗口已打开，则在窗口中单击 Variables 选项卡。

要查看变量变化：

1. 调试，然后暂停程序。
2. 单击“Variables”选项卡，查看窗口和局部变量值。

要更改变量的基数：

右键单击该变量所在的行，并选择“Display Value As”。

4.18 查看或更改器件存储器

MPLAB X IDE 具有灵活的、抽象化的存储器窗口，它们可在调试期间提供不同类型器件存储器的定制视图。只有在从调试中暂停时，才能在该窗口中看到更新后的值。

查看器件存储器

1. 在窗格中单击某个窗口，使窗格处于活动状态。存储器窗口将在该窗格中打开。
2. 从 *Window>Target Memory Views* 中选择一个存储器视图。下表说明了可用的选项。并非所有存储器类型在所有器件上都提供。

表 4-11. 存储器视图——8 位和 16 位器件

类型	说明
Program Memory	器件上的所有程序存储器（ROM）
File Registers（文件寄存器）	器件上的所有文件寄存器（RAM）存储器
SFR	所有特殊功能寄存器（SFR）
Peripherals	与外设有关的所有 SFR
Configuration Bits（配置位）	所有配置寄存器
EE Data Memory（EE 数据存储器）	器件上的所有 EE 数据存储器
User OTP Memory（用户 OTP 存储器）	用户 OTP 存储器
User ID Memory（用户 ID 存储器）	用户 ID 存储器

表 4-12. 存储器视图——32 位器件

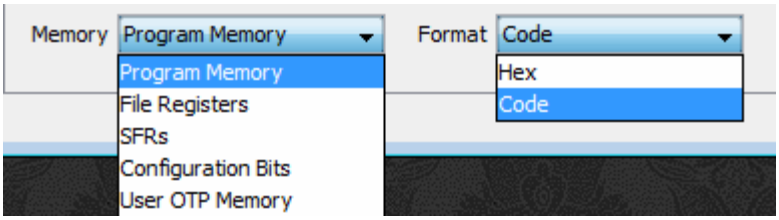
类型	说明
Execution Memory（执行存储器）	器件上的所有闪存
Data Memory（数据存储器）	器件上的所有 RAM

.....（续）

类型	说明
外设	与外设有关的所有特殊功能寄存器（SFR）
Configuration Bits	所有配置寄存器
CPU Memory（CPU 存储器）	所有 CPU 存储器
User ID Memory	用户 ID 存储器

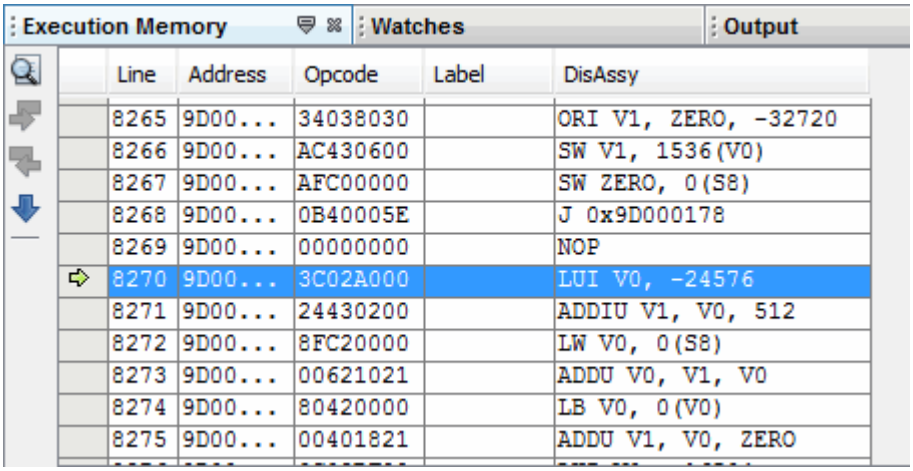
3. 存储器窗口打开时，可以通过在 Memory 下拉框中选择不同存储器来更改视图。也可以在 Format（格式）下拉框中更改当前存储器显示格式。选择哪种格式取决于所选存储器和器件的类型。

图 4-33. 存储器和存储器格式选择



4. 在调试并暂停之后，窗口中将填入所选的存储器。
5. 通过单击该窗口选项卡上的“x”关闭该窗口。

图 4-34. 存储器窗口内容



更改器件存储器

要更改存储器值，您必须调试代码。在代码运行时，您无法更改存储器。

注： 数据将仅在调试过程中发生更改。应用程序代码不会发生更改。

使用以下信息来更改存储器值：

- 在存储器窗口中，通过在相应列中单击，并选择或输入新数据来更改值。对于一些窗口，文本将使用红色来说明发生了更改。
- 大多数存储器窗口的上下文（右键单击）菜单上都具有 Fill memory（填充存储器）功能。
- 对于程序存储器，必须重新编译才能看到更改。使用 *Debug>Discrete Debugger Operation* 可使用更改后的数据对目标编程并启动调试器。

通过上下文菜单设置存储器窗口选项

在存储器窗口中单击右键将弹出一个上下文菜单，其中具有各种选项，例如显示选项、填充存储器、表导入/导出和输出到文件。菜单的内容取决于窗口。请参见 12. MPLAB X IDE 窗口和对话框。

刷新选定的存储器窗口

对于显示程序存储器、EEPROM、用户 ID 或配置位存储器的存储器窗口，可以通过执行以下操作刷新视图：

1. 如果正在进行调试，则除非工具和器件支持调试时读取，否则需要暂停程序（见下文）。
2. 单击名为“Read Device Memory”的图标  。

调试时读取

对于大多数器件，必须先暂停程序（Finish Debugger Session），之后才能读取器件存储器。对于某些器件，可以在调试模式下进行读取（调试时读取），此时您将知道它是可用的，因为“Read Device Memory”图标将在调试时不灰显。

当前，仅 MPLAB REAL ICE 在线仿真器和 MPLAB ICD 3 支持调试时读取。

调试时读取将以目标振荡器的速度进行，所以如果目标器件正以极低速度运行，则读取可能需要很长时间。您可以强制进行快速 ICSP 读取，方法是先完成调试会话再执行读取，因为当不处于调试会话中时，将总是进行 ICSP 读取。

4.19 在 Configuration Bits 窗口中设置配置值

从 *Window>Target Memory Views* 中选择“Configuration Bits”，可打开 Configuration Bits 窗口。关于使用存储器窗口的详细信息，请参见 4.18 查看或更改器件存储器。

Configuration Bits 窗口可以帮助您开发自己的配置位设置。不过，您必须在用于生产的代码中设置配置位。

4.19.1 使用 Configuration Bits 窗口

您可以在调试会话过程中在 Configuration Bits 窗口中临时更改配置位（见 4.18 查看或更改器件存储器）。完成所需的设置之后，您可以通过几种方式来保留这些设置（如以下各节所述）。

如果未处于调试会话中，则无法编辑配置位。此外，如果进行重新编译，Configuration Bits 窗口中的所有更改都会丢失。

关于不同器件的配置位设置的总结，请参见 14. 配置设置汇总。

注： 对于 AVR 或 SAM 器件，该窗口中的红色文本表示需要执行操作。将鼠标悬停在上方可查看弹出的说明文本。

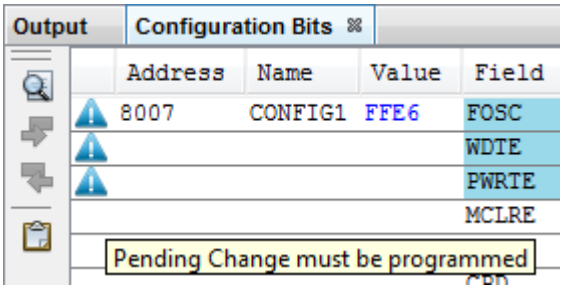
编辑配置设置

始终可在该窗口中编辑值，而不像其他存储器窗口那样必须处于调试模式。但是，如果进行重新编译，Configuration Bits 窗口中的所有更改都会丢失。

初次进入 Configuration Bits 视图时以及执行 Reset to Defaults（复位为默认值）菜单操作时，可编辑字段为默认值。当配置位用源代码表示时，所有可编辑字段的值都将初始化为源代码中赋的值。这是在编译操作之后以及成功加载之后发生的。

在配置窗口中更改默认值时，系统将提醒您这是一项临时更改，只有将其编程到目标器件后才能永久保留。

图 4-35. 临时更改



可以按以下方式更改值：

- **选择框值**：在“Option”或“Setting”（设置）列中提供选择框值。只需单击所需字段的列值，然后从列表中进行选择即可更改。

- **手动输入值：**可在“Option”或“Setting”列中手动输入值。“Option”文本将指出存在用户可选范围，例如“User range: 0x0 - 0x3F”（用户范围：0x0-0x3F）。“Setting”文本将指出需要输入值，例如，“Enter Hexadecimal value”（输入十六进制值）。

  单击所需字段的“Option”或“Setting”文本后，相应的“Value”（值）列将变为可编辑状态。输入的值将基于绝对（原始）值进行验证，因此无需担心将应用已编辑字段值的实际位置。如果尝试用“0”填充输入值以接近实际掩码位置，则该操作将被滤除，并尽可能修正为绝对值。

Generate Source Code to Output

单击 Configuration Bits 窗口底部的“Generate Source Code to Output”（生成源代码到输出）（下面第一个图）。该操作将在 Output 窗口中生成配置位设置，随后可以将该设置复制到代码中（下面第二个图）。

图 4-36. Configuration Bits 窗口——Generate Source Code to Output

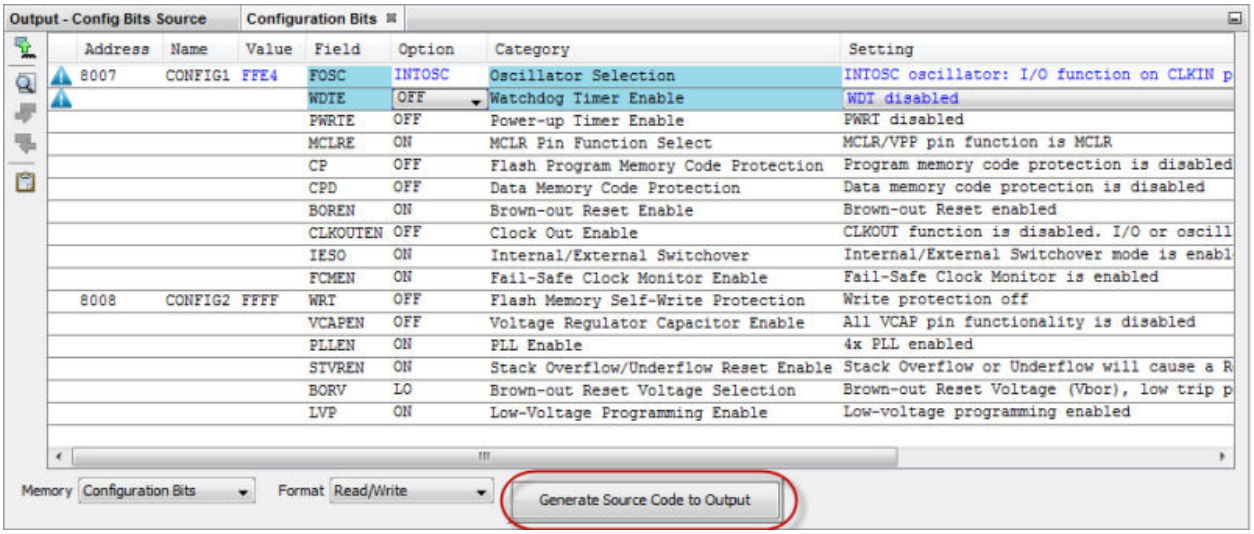
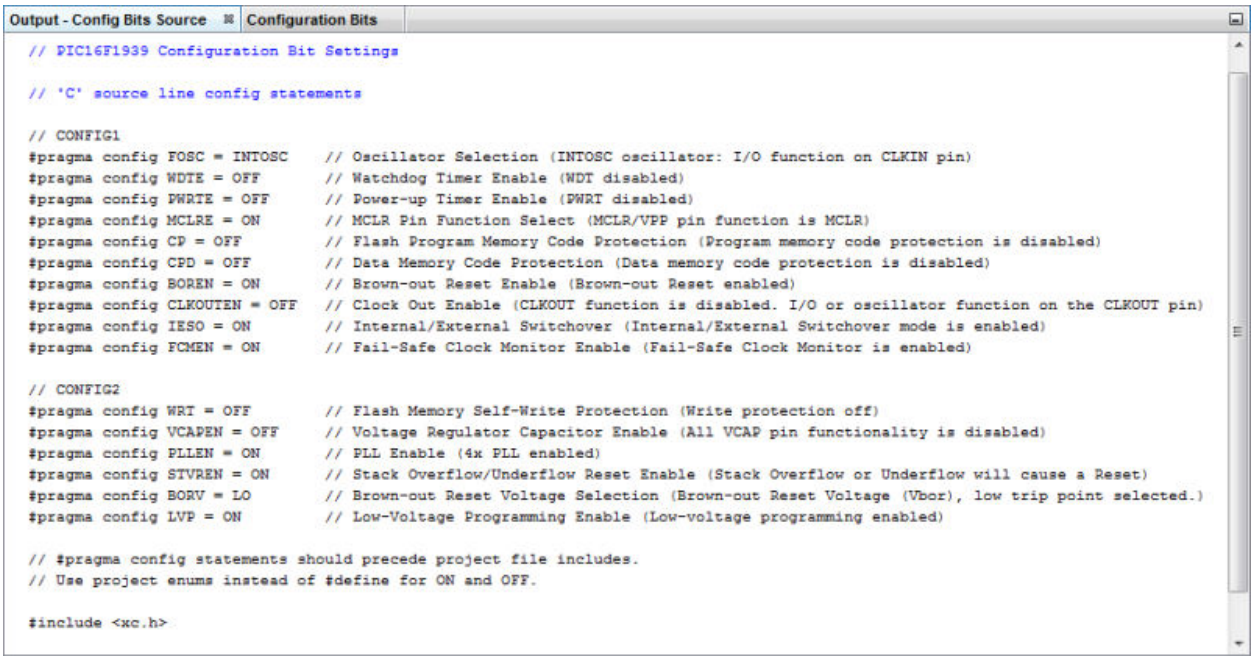


图 4-37. Output 窗口——生成的源代码



Insert Source Code in Editor

可以从 Configuration Bits 窗口上下文菜单或侧栏“剪贴板”图标（下面第一个图）中选择“Insert Source Code In Editor”（在编辑器中插入源代码）。在编辑器中打开或创建一个主项目文件，将光标置于想要应用配置位设置的位置，然后单击图标（下面第二个图）。

图 4-38. Configuration Bits 窗口——Insert Source Code in Editor

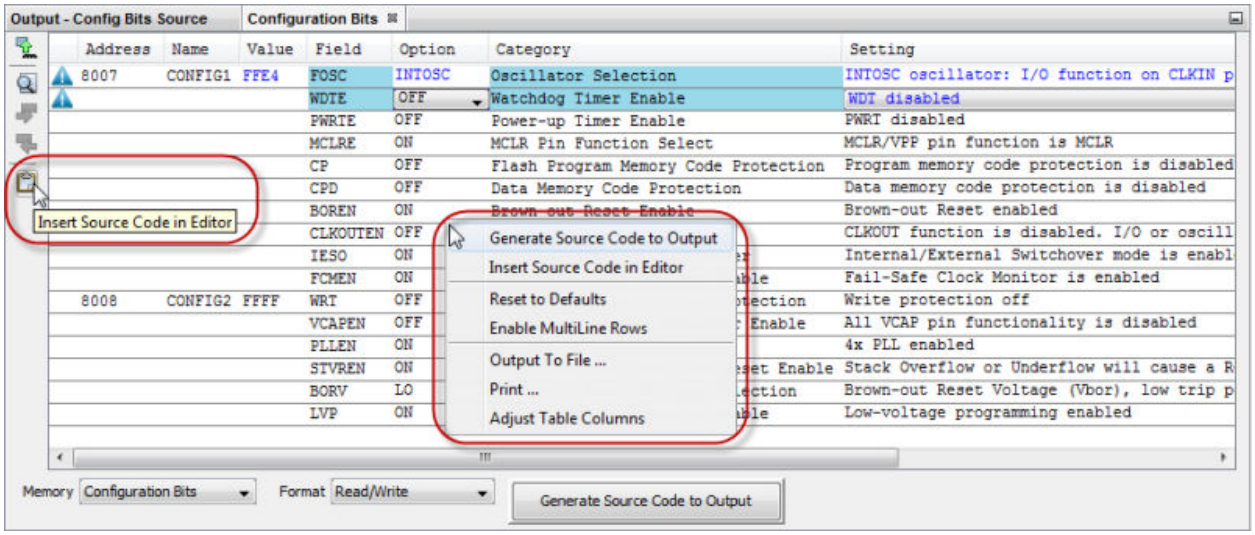
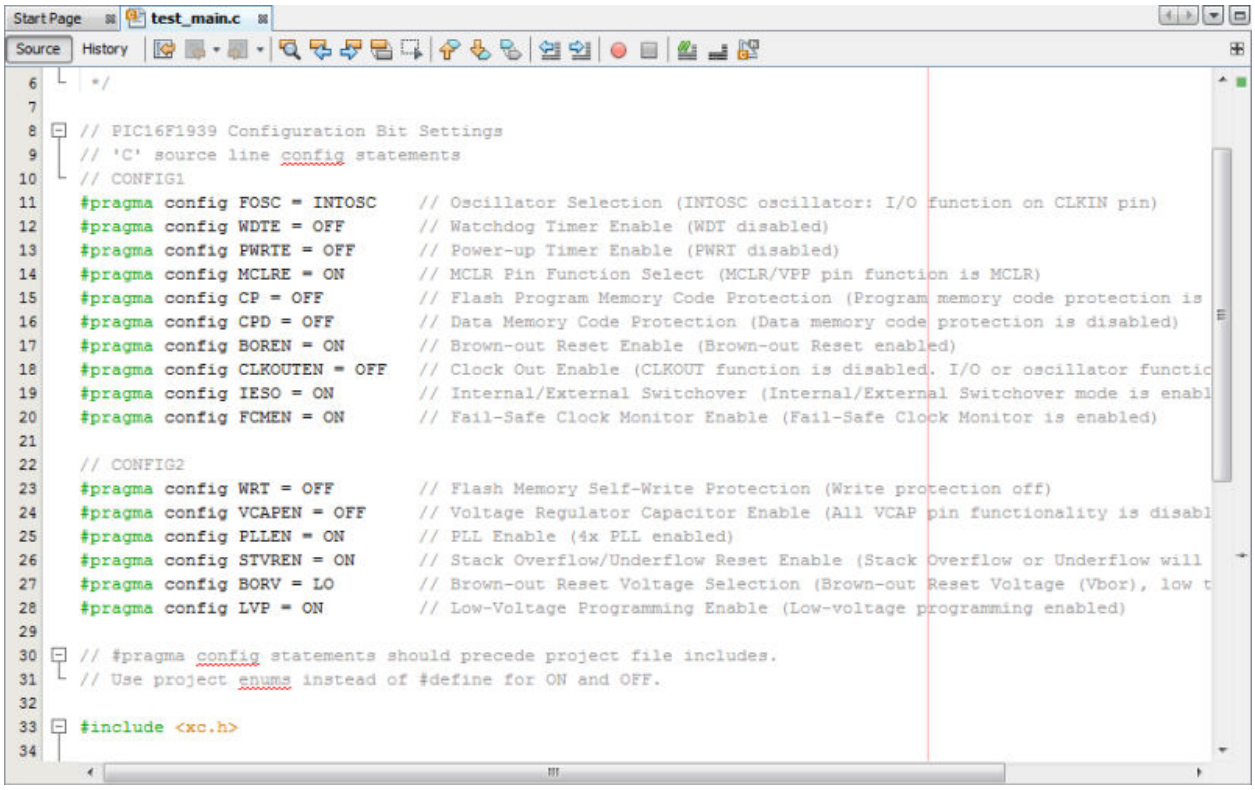


图 4-39. 编辑器窗口——生成的源代码



Program Device for Debugging

单击 IDE 工具栏图标“Program device for debugging”（针对调试对器件编程），可将 Configuration Bits 窗口的当前值编程到器件中。请参见 4.20 对器件编程。

4.19.2 在汇编项目中设置配置位

对于 MPLAB XC 汇编项目，仍可使用 Configuration Bits 窗口。（这不适用于 MPASM 或 MPLAB ASM30 项目。）向项目中添加一个简单的 C 文件以包含生成的`#pragma config` 宏。之后，便无需手动输入汇编字 `config`。

汇编字 `config` 和`#pragma config` 之间没有区别；它们都会写入一个程序字，并且得到的程序大小相同。不过，`#pragma config` 语法更为灵活，可以改进验证、格式化和描述性注释。

4.20 对器件编程

调试完代码之后，可以将它编程到目标器件上。

设置项目编程属性

在 Project Properties 窗口中设置编程选项：

1. 右键单击 Projects 窗口中的项目名称并选择“Properties”。
2. 在“Categories”下，单击将用于编程代码的硬件工具，例如 PM3。
3. 复查“Memories to Program”（要编程的存储器）选项类别下的设置。如果希望使用 Preserve Memory（保护存储器）选项，请确保代码未被代码保护。代码通过以下方式保护：编程器读取它需要保存的部分、执行器件批量擦除、重新编程器件，然后使用先前保存的内容重写保护的区域。
4. 复查“Program Options”（编程选项）类别下的设置。
5. 根据您使用的硬件工具，可能还会有其他编程选项类别。复查每个类别，确保设置对于您的项目是正确的。
6. 将 RTDM 与 DMCI 和电机控制应用程序等一起使用时，您需要选中“Loading”（装入）类别下的“Load symbols when programming or building for production (slows process)”（为生产进行编程或编译时装入符号（减慢过程））复选框。

关于编程选项的更多信息，请参见硬件工具文档。

按照需要设置编程选项后，您可以继续对器件编程。

执行编程

要使用调试后的代码对目标器件编程，请单击工具栏按钮 Make and Program Device Project（Make 并编程器件项目）。

表 4-9 列出了其他与编程相关的功能。第一个功能通过单击按钮来激活。对于其他功能，请单击按钮图标旁边的向下箭头。

表 4-13. 工具栏按钮上的编程功能

按钮图标	功能	详细信息
	Make and Program Device	对项目进行编译（如需要）并对器件进行编程。 程序将在编程完成后立即开始执行。
	Program Device for Debugging	将使用调试映像对器件进行编程。 程序将在编程完成后立即开始执行。
	Program Device for Production（针对生产对器件编程）	将使用生产映像对器件进行编程。 程序将在编程完成后立即开始执行。
	Programmer to Go PICkit 3（使用 PICkit 3 的脱机编程）	使用 PICkit 3 的脱机编程（Programmer to Go）功能。
	Read Device Memory	将目标存储器中的内容传输到 MPLAB X IDE。
	Read Device Memory to File（将器件存储器读取至文件）	将目标存储器的内容传输到指定文件中。
	Read EE/Flash Data Memory to a File（将 EE/闪存数据存储器读取至文件）	将目标数据存储器的内容传输到指定文件中。

.....（续）

按钮图标	功能	详细信息
	Hold In Reset	使器件在复位和运行之间切换。

使用 MPLAB 集成生产环境编程

不是所有编程功能都包含在 MPLAB X IDE 中。关于更多编程支持，请参见 MPLAB X IDE 安装随附的 MPLAB IPE。

	MPLAB IPE 的桌面图标

5. 附加任务

以下步骤说明如何在 MPLAB X IDE 中执行更多的任务。

表 5-1. 执行附加任务

1 处理项目	1. 了解如何在项目中使用器件包。 2. 选择如何打开现有 MPLAB X IDE 项目。 3. 将现有 MPLAB IDE v8 项目导入 MPLAB X IDE。 4. 使用预编译项目导入向导打开预编译的映像（Hex、ELF 或 COF）。 5. 使用可装入项目、文件和符号合并或替换项目十六进制文件。一种常见的应用是使用可装入对象来合并自举程序和应用程序代码，如可装入项目和文件：自举程序中所述。 6. 创建库项目以将其输出编译为库。 7. 使用导入 Atmel Studio 7 或 Atmel START 项目向导来打开现有 Studio 或 Start 项目。 8. 通过其他嵌入式项目或示例项目创建项目。 9. 使用其他文件类型，而不只是 Microchip 编译器支持的文件类型。此外，修改或创建代码模板来更改在项目中使用的默认文件模板。 10. 在项目中切换硬件或语言工具。 11. 为现有项目修改项目文件夹和编码。
2 调试代码	1. 使用跑表来确定断点之间的时间。 2. 查看 Disassembly 窗口来查看反汇编的代码。 3. 要浏览函数调用，请使用查看调用堆栈或查看调用图。 4. 使用查看仪表板显示来查看项目信息，例如断点资源、校验和以及存储器使用情况。 5. 在设计和调试期间查看项目的寄存器（I/O View）。
3 管理代码	1. 通过使用重构和性能分析工具改进代码*。 2. 通过内置文件历史记录使用本地历史记录控制源代码。或者使用版本控制系统控制源代码。 3. 通过使用团队服务器和问题跟踪系统在代码开发和错误跟踪方面进行协作*。 4. 比较 MPLAB XC 编译器免费版与专业版许可证以确定哪些优化最适合您的应用。
4 添加功能	1. 添加插件工具来协助代码开发。
*要查看该功能，请参见 **Start Page**，**My MPLAB X IDE** 选项卡，“Extend MPLAB”（扩展 MPLAB）部分，“Selecting Simple or Full-Featured Menus”（选择简单或全功能菜单）主题。	

5.1 使用器件包

MPLAB X IDE 需要关于所支持器件的特定信息，以便模拟或仿真这些器件。该信息位于器件（.PIC）文件中，并包含器件电源要求、编程方法和架构等相关数据。

自 MPLAB X IDE v5.00 起，器件文件被分组到各版本器件系列包（DFP）中。虽然每个 MPLAB X IDE 版本都随附器件包，但是可通过升级器件包版本以包含新器件、新器件功能支持或器件缺陷修正支持。

例如，要在 Windows OS 计算机上查找包含 PIC16F19197 器件文件的器件系列包（DFP）：

```
C:\Program Files (x86)\Microchip\MPLABX\v5.00\packs\Microchip\PIC16F1xxxx_DFP
\1.0.38\edc\PIC16F19197.PIC
```

其中：

- 安装位置：`C:\Program Files (x86)\Microchip\MPLABX`

- 版本：\v5.00
- 包含包的文件夹：\packs
- 器件系列包：\Microchip\PIC16F1xxxx_DFP
- 包版本：\1.0.38
- 包含器件文件的文件夹：\edc
- 器件文件：\PIC16F19197.PIC

更多信息，请参见“开发人员帮助”：

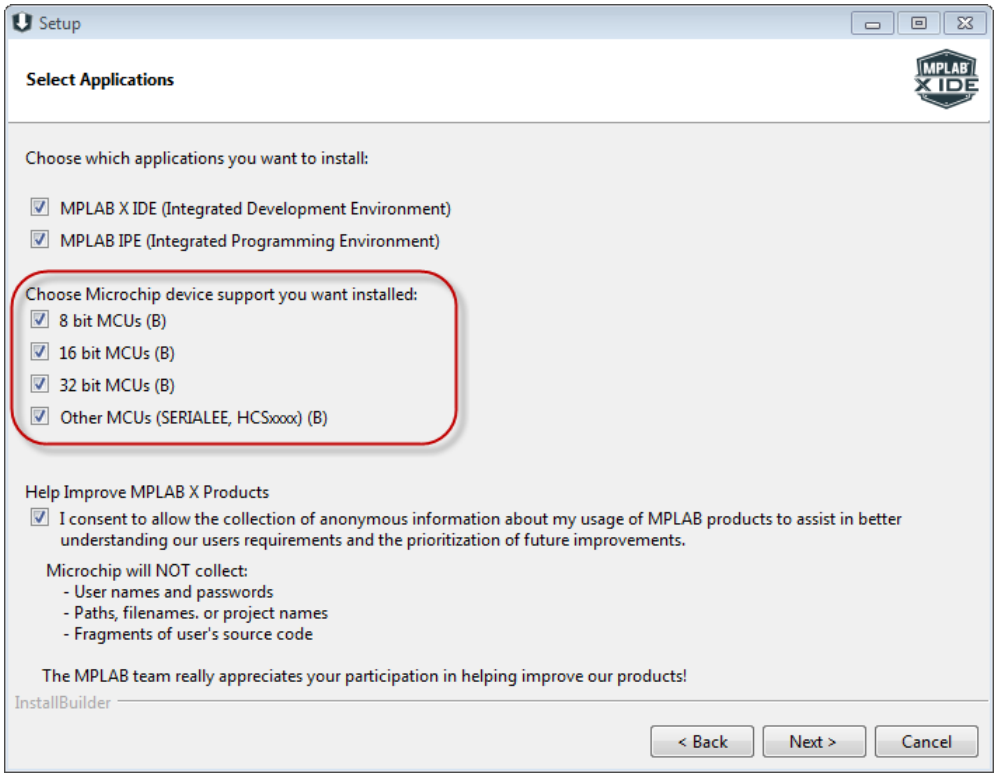
项目中的器件包

器件包常见问题解答

5.1.1 在 MPLAB X IDE 安装期间安装包

在安装 MPLAB X IDE（自 MPLAB X IDE v5.20 起）时，现在可以“Choose Microchip device support you want installed”（选择要安装的 Microchip 器件支持）。该操作将安装与所选器件架构相关的包。仅安装所需的包可节省磁盘空间并加快安装速度。

这也称为选择性安装程序。



关于安装 MPLAB X IDE 的信息，请参见 MPLAB X IDE 自述文件。

5.1.2 在 MPLAB X IDE 安装之后安装包

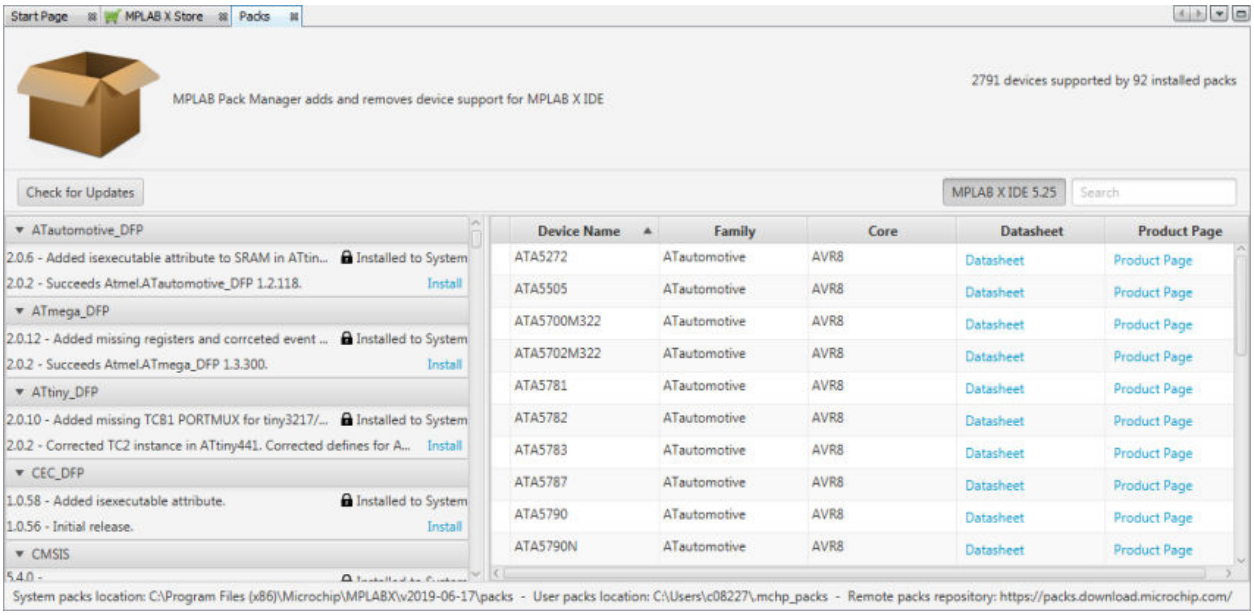
选择 *Tools>Packs*（工具>包），以打开包含可通过 MPLAB 包管理器安装的各版本包的列表。

首次打开时，将筛选列表以查找已知与当前 IDE 版本兼容的包。单击 IDE 版本按钮可禁止该功能，以便查看所有包版本。有关示例，请参见下文。

MPLAB X IDE 5.25	筛选包列表

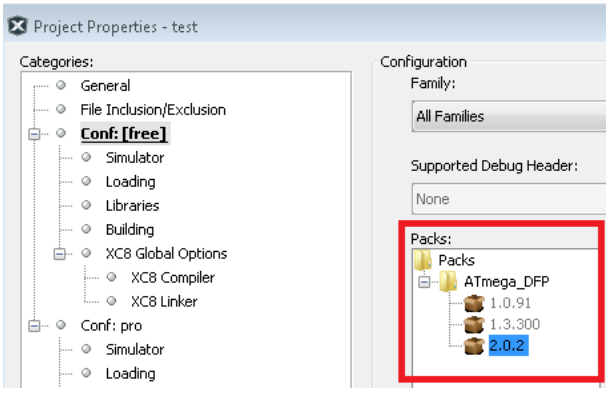
MPLAB X IDE 5.25	显示所有包版本

图 5-1. MPLAB 包管理器



5.1.3 切换已安装的包

对于在 MPLAB X IDE 中打开的项目，在 Projects 窗口中右键单击项目名称，然后选择“Properties”以打开 Project Properties 窗口。查看 Packs 列表以查看项目器件的可用包。



呈灰显的包不包含在 MPLAB X IDE 的相应版本中。要安装包，请参见 5.1.2 在 MPLAB X IDE 安装之后安装包。关闭筛选器可查看所有版本，以便安装所需的版本。

注： 如果在 Project Properties 中将器件切换为未安装器件包的器件，则**不会**收到提示。您必须通过 MPLAB 包管理器手动安装器件包。

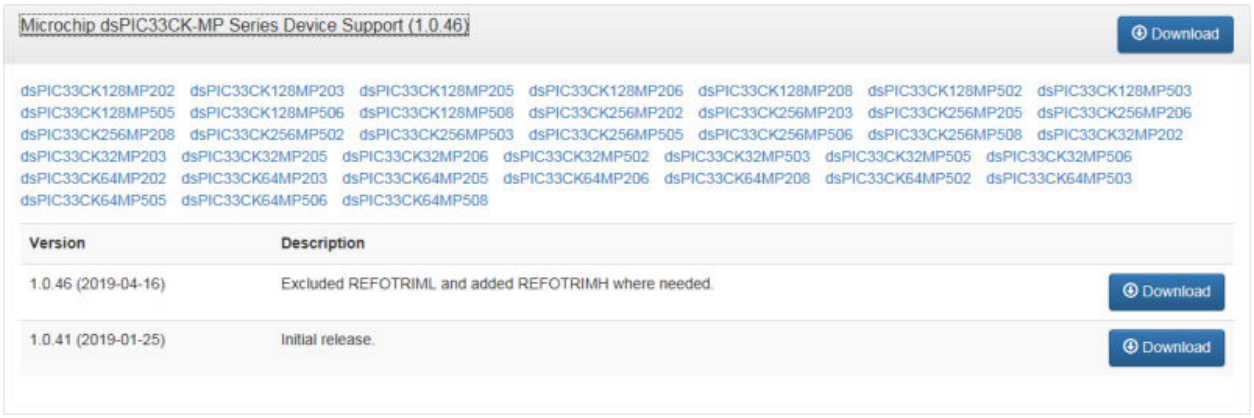
5.1.4 网络上的包

要从网络上查找和下载包版本，请访问：

https://packs.download.microchip.com/

单击包名称可查看包信息，例如支持的器件和包版本历史记录。

图 5-2. Microchip 包资源库



5.2 打开现有 MPLAB X IDE 项目

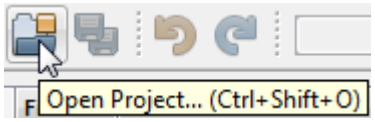
对于现有项目，无论是您创建的项目还是下载的示例项目，可通过多种方法来打开。

通过 IDE 菜单打开

选择 *File>Open Project*（文件>打开项目）或 *File>Open Recent Project*（文件>打开最近项目）。

通过图标打开

单击 **Open Project** 图标。



通过拖放项目文件夹打开

在文件管理器窗口中选择一个项目文件夹。将该文件夹拖放到**编辑器窗格**（见 4.2 查看桌面上的变化）。项目将在 Projects 窗口中打开。

图 5-3. 拖放文件夹

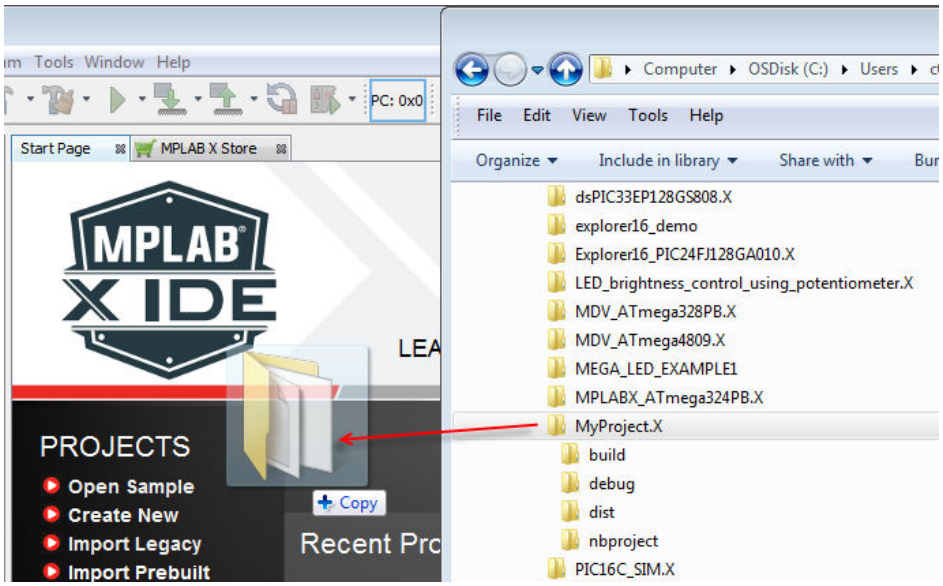
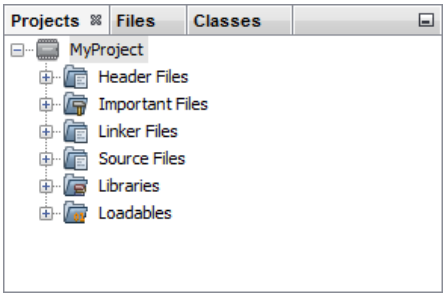


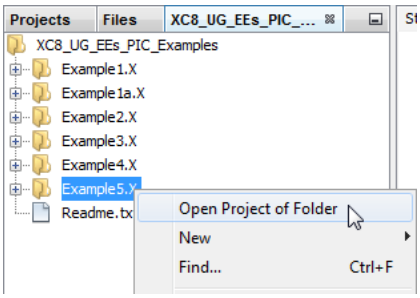
图 5-4. 打开的项目



通过拖放包含项目的文件夹打开

要打开包含项目的文件夹，请在文件管理器窗口中选择该文件夹。将该文件夹拖放到**编辑器窗格**（见上一步）。将在**文件窗格**中打开一个在选项卡上显示文件夹名称的窗口。文件夹和相关项目将显示在该窗口中。要打开该文件夹中的项目，请右键单击项目名称，然后选择“Open Project of Folder”（打开文件夹的项目）。项目将在 Projects 窗口中打开。

图 5-5. 打开的包含项目的文件夹



5.3 导入现有 MPLAB IDE v8 项目

使用 New Project 向导将 MPLAB IDE v8 项目导入 MPLAB X IDE 项目时，需要注意以下事项：

不会转移 MPLAB IDE v8 工作区设置

在 MPLAB IDE v8 的工作区中保存的设置（例如工具设置）将不会转移到新的 MPLAB X IDE 项目中。关于工作区中所存储内容的信息，请参见 MPLAB IDE v8 帮助（*MPLAB IDE Reference>Operational Reference>Saved Information*（MPLAB IDE 参考>工作参考>保存的信息））。

项目设置（例如编译器、链接器和汇编器选项）会转移到新的 MPLAB X IDE 项目中。

MPLAB IDE v8 项目的文件可能发生更改

使用“用于 PIC24 MCU 和 dsPIC DSC 的 MPLAB C 编译器”（即 MPLAB C30）以及 COFF 调试文件格式的 MPLAB IDE v8 项目可能会有所更改。

- 除非 MPLAB X IDE 项目使用 COFF 库（在这种情况下，项目格式将继续使用 COFF），否则该项目将转换为 ELF/DWARF 调试文件格式。
- MPLAB IDE v8 的文件扩展名不区分大小写，例如 `.c` 和 `.C` 相同。但是，MPLAB X IDE 区分大小写，`.c` 与 C 代码文件相关联，而 `.C` 则与 C++代码文件相关联。因此，如果导入的 MPLAB IDE v8 项目的 C 代码被指定为 `.C`，MPLAB X IDE 会将 `.C` 文件重命名为 `.c` 以避免错误的编译器行为。

MPLAB IDE v8 项目的编译可能发生更改

导入到 MPLAB X IDE 中的 MPLAB IDE v8 项目的编译可能与原始项目有所不同。

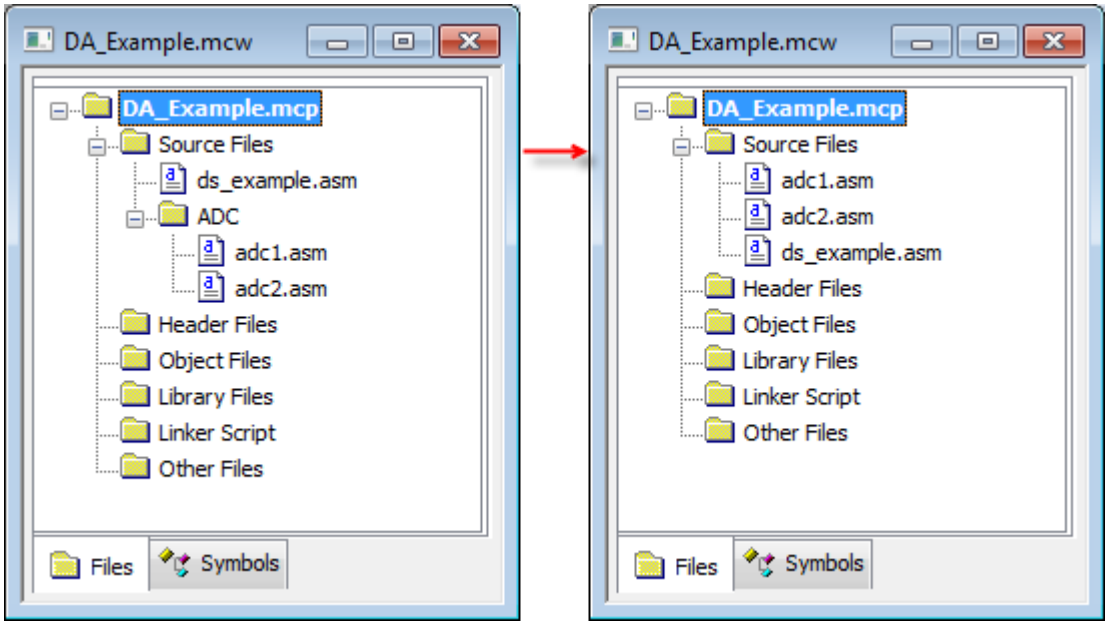
- 请勿在项目代码中使用 `__FILE__`、`assert()`、`__TIME__` 或 `__DATE__`，否则编译（和校验和）将有所不同。

- 使用逻辑子文件夹的 MPLAB IDE v8 项目在导入 MPLAB X IDE 时将不会具有相同的校验和。要使用相同的校验和进行编译，请将逻辑子文件夹中的文件移至主逻辑文件夹（即 Source Files 和 Header Files（头文件）等）中，然后导入 MPLAB X IDE。

要删除 Source Files 下的逻辑子文件夹：

- 将文件拖动到 Source Files 下。
- 选择相应文件夹并单击<Delete>（<删除>）。
- 也可以右键单击该文件夹，然后选择“Remove from Project”（从项目中删除），将文件夹从项目中删除。但是该文件夹在 PC 上仍存在，不会被删除。

图 5-6. 移动文件并删除子文件夹



5.3.1 打开导入 MPLAB IDE v8 项目向导

可通过下述两种方法打开该向导。

Import Legacy Project 选项

要打开 Import Legacy Project（导入旧项目）向导，请执行以下操作之一：

- 在 **Start Page** 上，单击 **Learn & Discover** 或 **My MPLAB X IDE** 选项卡中“Projects”部分的“Import Legacy”（导入旧项目）链接。
- 选择 *File>Import>MPLAB IDE v8 Project*。

New Project 选项

要打开 New Project 向导，请执行以下操作之一：

- 在 **Start Page** 上，单击 **Learn & Discover** 或 **My MPLAB X IDE** 选项卡中“Projects”部分的“Create New”链接。
- 选择 *File>New Project*（或 Ctrl+Shift+N）。
- 单击工具栏上的“New Project”图标。 

向导将启动，指导您完成新项目设置。

步骤 1.选择项目：选择“Microchip Embedded”类别，然后选择项目类型“Existing MPLAB IDE v8 Project”。单击 **Next**。

5.3.2 使用导入 MPLAB IDE v8 项目向导

请按照以下步骤导入 MPLAB IDE v8 项目。步骤根据“Import Legacy Project”选项编号（见 5.3.1 打开导入 MPLAB IDE v8 项目向导）。

单击 **Next** 转至下一步。

步骤 1.查找 MPLAB IDE v8 项目*.mcp 文件。输入或浏览到旧项目。

步骤 2.选择器件。从“Device”下拉列表中选择将在应用中使用的器件。要缩小选择列表，请先选择 Family。

步骤 3.选择调试头。如果存在可用于选定器件的调试头，则会出现该步骤。要确定调试是否需要调试头，或器件是否具有片内调试电路，请参见以下文档之一。然后选择是否使用调试头。

- 处理器扩展包（PEP）和调试头规范
- 仿真扩展包（EEP）和仿真头用户指南

步骤 4.选择工具。从列表中选择将用于开发应用的开发工具。要确定针对您器件的工具支持，请参见 4.1.6.1 项目工具支持。

步骤 5.选择接插板。选择一个接插板（如果需要使用的话）。

步骤 6.选择编译器。从列表中选择将用于开发应用程序的语言工具（编译器）。要确定针对您器件的工具支持，请参见 4.1.6.1 项目工具支持。

步骤 7.选择项目名称和文件夹。建议不要更改默认名称和位置，以保持两个项目的可维护性。请参见下图中的示例。

文件位置：

新项目不会将源文件复制到其文件夹中，而是引用 v8 文件夹中的文件位置。

要创建一个独立的 MPLAB X IDE 项目，需要创建一个新项目，并将 MPLAB IDE v8 源文件复制到其中。

主项目：

选中复选框可使该项目在导入后成为主项目。

项目位置：

MPLAB X IDE 项目位置不在 MPLAB IDE v8 项目文件夹中，因此应将其取消选中。

文件格式：

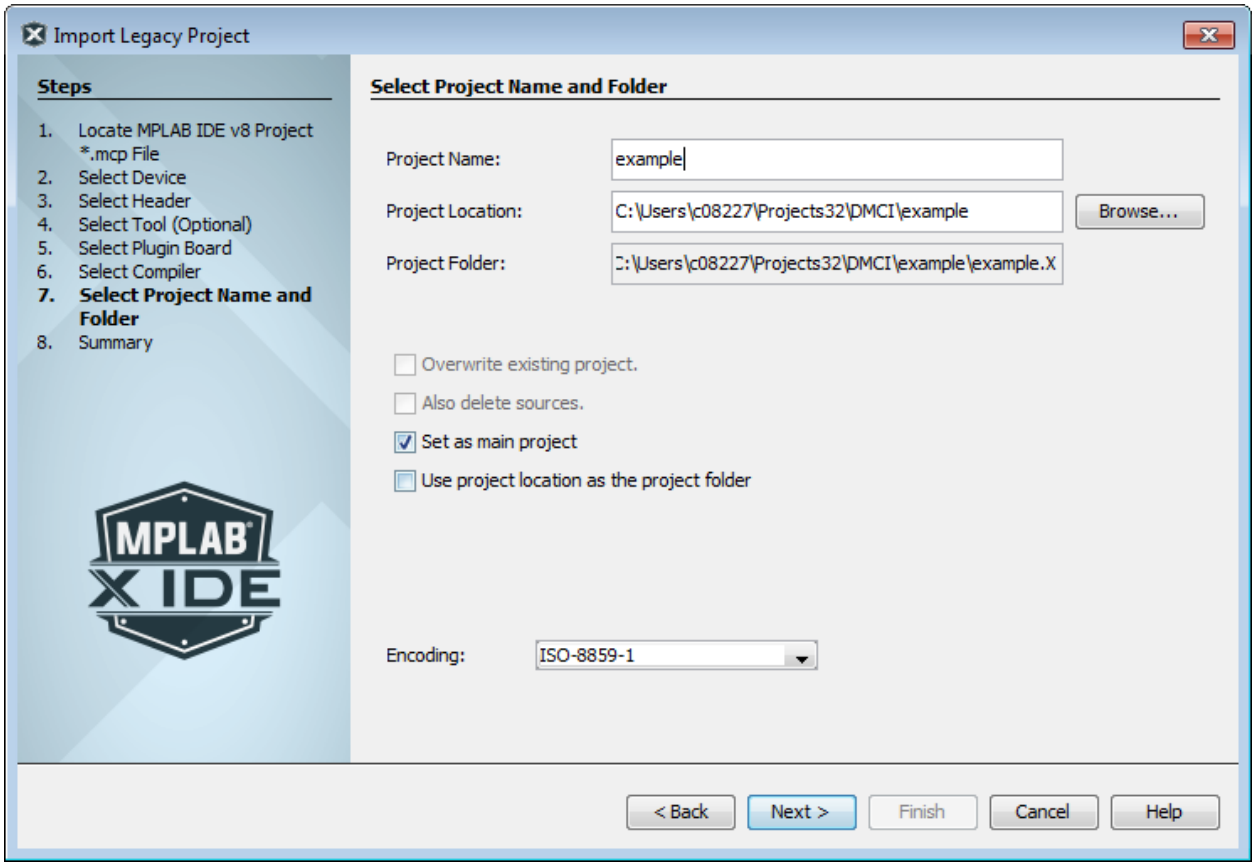
“ISO-8859-1”是从 MPLAB IDE v8 导入项目时使用的默认字符编码。

选择的编码方式应与导入的项目中使用的编码方式相匹配。例如，如果 MPLAB IDE v8 格式为“950 (ANSI/OEM - Traditional Chinese Big5)”（950（ANSI/OEM——繁体中文 Big5）），则从下拉列表中选择“Big5”。

步骤 8.摘要。在单击 **Finish** 之前查看摘要。如果有任何内容不正确，请使用 **Back**（后退）按钮后退并更改它。

旧项目将在 Projects 窗口中打开。

图 5-7. 步骤 7 对话框示例



5.4 预编译项目

通过使用 Import Image File（导入映像文件）向导，基于预编译的可装入映像（Hex、ELF 或 COF 文件）创建项目。

注： 要将预编译映像编程到器件中，需要单击 **Make and Program Device** 图标，虽然它将仅对器件进行编程（无 make 操作）。

有两种方法可以打开该向导：Start Page 选项和 New Project 选项。

Import Prebuilt 项目选项

要打开“Import Image File”向导，请执行以下操作之一：

- 在 **Start Page** 上，单击 **Learn & Discover** 或 **My MPLAB X IDE** 选项卡中“Projects”部分的“Import Prebuilt”（导入预编译项目）链接。
- 选择 *File>Import>Hex/ELF (Prebuilt) File*（文件>导入>十六进制/ELF（预编译）文件）。

New Project 选项

关于打开 New Project 向导的方法，请参见 4.1.2 启动 New Project 向导。

向导将启动，指导您完成新项目设置。

- **步骤 1.选择项目**：选择“Microchip Embedded”类别，然后选择项目类型“Prebuilt (Hex, Loadable Image) Project”。单击 **Next**。
- 将打开“Import Image File”向导。

Import Image File 向导

请按照以下步骤导入映像文件。步骤根据“Import Prebuilt 项目”选项编号。

单击 **Next** 转至下一步。

步骤 1.导入映像文件。选择映像文件的名称和位置。您可以浏览至一个位置。

步骤 2.选择器件。从“Device”下拉列表中选择将在应用中使用的器件。要缩小选择列表，请先选择“Family”。

步骤 3.选择调试头。如果存在可用于选定器件的调试头，则会出现该步骤。要确定调试是否需要调试头，或器件是否具有片内调试电路，请参见以下文档之一。然后选择是否使用调试头。

- 处理器扩展包（PEP）和调试头规范
- 仿真扩展包（EEP）和仿真头用户指南

步骤 4.选择工具。从列表中选择将用于开发应用的开发工具。要确定针对您器件的工具支持，请参见 4.1.6.1 项目工具支持。

步骤 5.选择项目名称和文件夹。选择新项目的名称和位置。您可以浏览至一个位置。完成后单击 **Finish**。

新项目将在 Projects 窗口中打开。

关于将项目导出为十六进制文件的信息，请参见 12.15 Projects 窗口中的“项目菜单”。

5.5 可装入项目、文件和符号

使用可装入项目和文件来合并项目、合并十六进制文件、合并项目与十六进制文件或替换项目十六进制文件。Hexmate 应用程序用于将项目或装入的十六进制文件合并为一个文件。关于使用该应用程序的更多信息，请参见所安装的 MPLAB XC8 C 编译器的 docs 文件夹中的《MPLAB® XC8 C 编译器用户指南》（DS50002053D_CN）。另请参见 9.5 错误中的“HEXMATE 冲突报告地址错误”部分。

使用可装入符号在生产编译或编程期间装入调试符号，例如将 RTDM 与 DMCI 和电机控制应用程序等一起使用时。

可装入项目或文件对于创建合并的自举程序和应用程序代码很有用。请参见 5.6 可装入项目和文件：自举程序。

下面列出了当前项目和可装入对象的组合。

表 5-2. 可装入对象组合

当前项目	可装入对象	注意事项
独立项目 现有 MPLAB IDE v8 项目 库项目	独立	无
	十六进制文件	无
	ELF 或 COF 文件	可以调试，但不能编译。 Output 窗口中将会显示错误。
预编译（十六进制）	十六进制文件	不会自动检查是否有重叠的存储器区域。 可以调试，但不能编译。 Build 按钮将被禁止。
	ELF 或 COF 文件	
预编译（ELF 或 COF）	十六进制文件	
	ELF 或 COF 文件	

下面列出了这些选项。

表 5-3. 可装入对象选项

Add Loadable Project(s)（添加可装入项目）	将一个或多个现有项目装入当前项目中。 在编译当前项目时，将编译所有项目并将十六进制文件合并为一个。所有调试文件也将进行合并（ELF 或 COF）。

Add Loadable File(s)（添加可装入文件）	将一个或多个现有十六进制文件装入当前项目中。 在编译当前项目时，十六进制文件将与其他十六进制文件合并为一个文件。 注：您将无法再调试包含十六进制文件的项目。请使用可装入项目进行调试。
Add Alternate File（添加备用文件）	装入要使用的备用十六进制文件。 该选项用于提供一个编译后步骤，您可以在该步骤中将项目十六进制文件复制或移动到另一个位置，使用诸如 HEXMATE 之类的工具来将您的文件和另一个十六进制文件合并，然后将文件装回到 IDE 中。

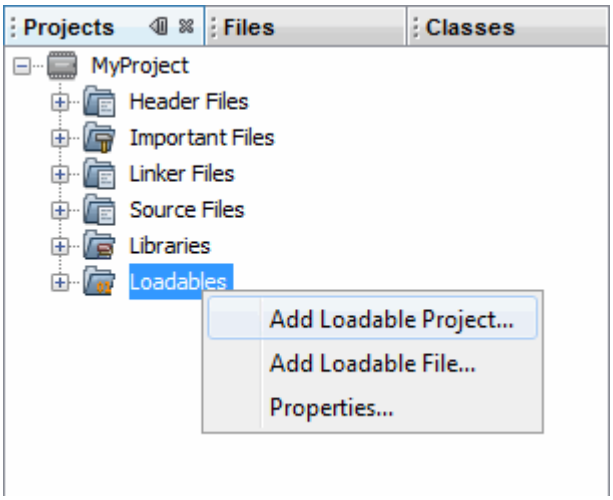
5.5.1 Projects 窗口——可装入对象设置

右键单击 Projects 窗口中的“Loadables”（可装入对象）文件夹（见下图）并选择一个选项：

- Add Loadable Project——选择将现有项目添加到当前项目中。重复该步骤来添加更多项目。
- Add Loadable Files——选择将现有十六进制文件添加到当前项目中。重复该步骤来添加更多十六进制文件。
- Properties——打开 Project Properties 窗口来进行装入。请参见 5.5.2 Project Properties 窗口——装入设置。

编译当前项目以编译所有项目，并将十六进制文件合并为一个。所有调试文件也将进行合并。

图 5-8. PROJECTS 窗口——LOADABLES 文件夹

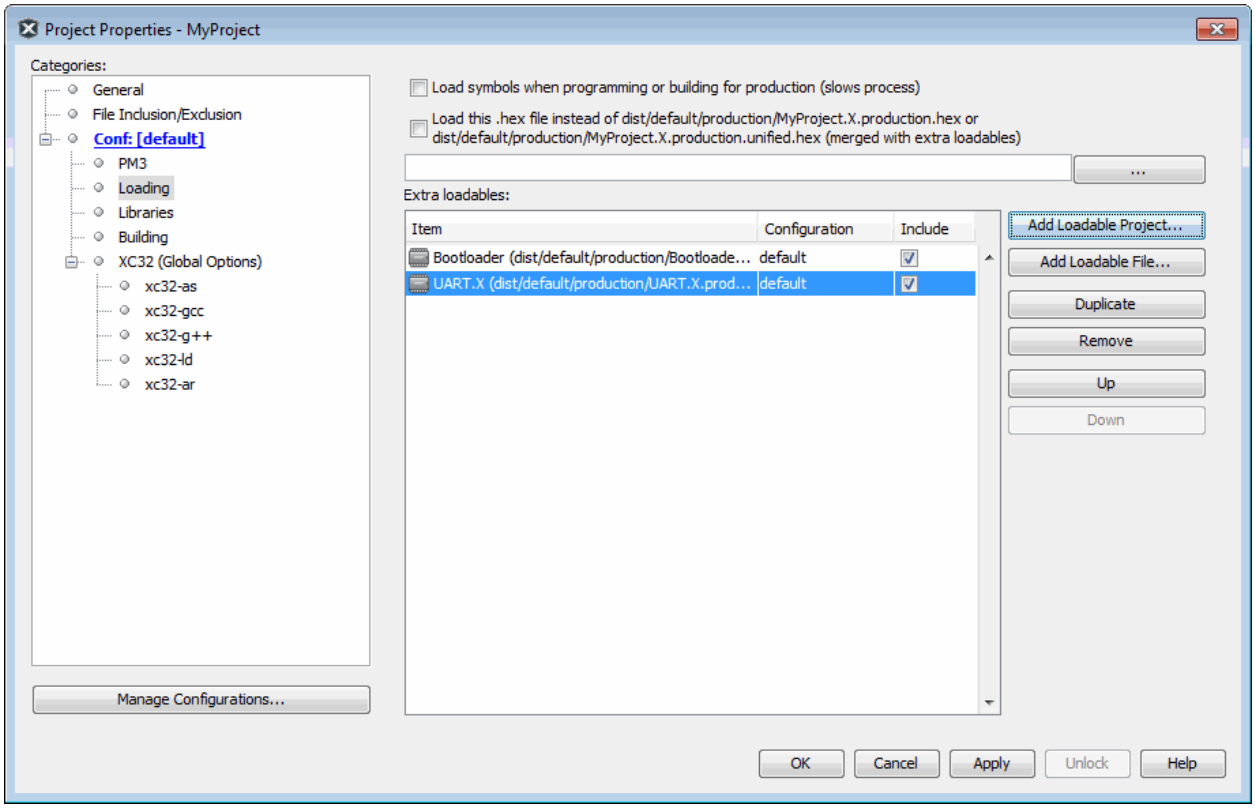


5.5.2 Project Properties 窗口——装入设置

打开 Project Properties 窗口（*File>Project Properties*），然后单击“Loading”。

- 将当前项目与其他项目合并
- 将当前项目十六进制文件与其他十六进制文件合并
- 装入备用十六进制文件
- 在编程/编译期间装入调试符号

图 5-9. Project Properties——装入

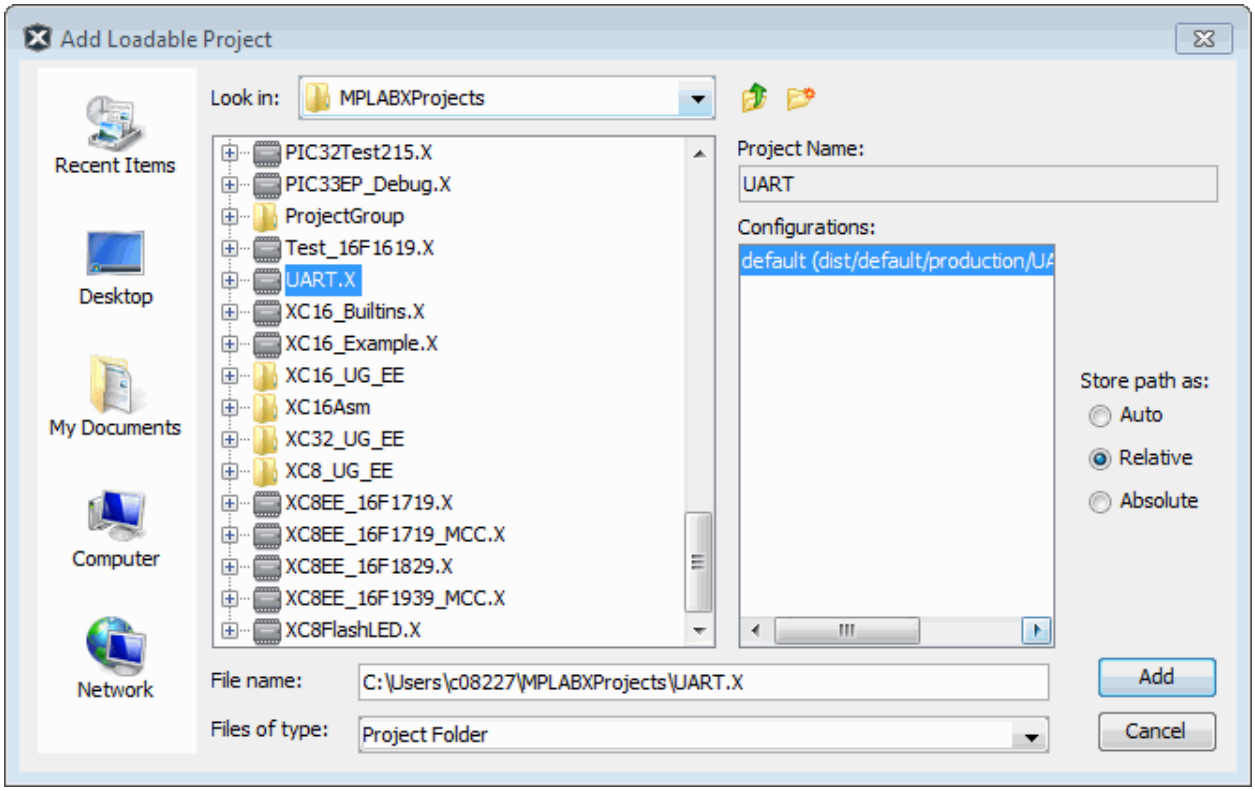


将当前项目与其他项目合并

1. 单击 **Add Loadable Project**。将出现 Add Loadable Project 对话框。
2. 浏览至一个项目，并选择该项目。
3. 在“Configurations”（配置）下，选择要用于该项目的编译配置。如果尚未向该项目添加其他配置，则将只会看到“default”。
4. 在“Store path as”（路径存储方式）下，选择如何存储有关可装入项目路径的信息。可选择以下选项：
   - 4.1. Auto——允许 MPLAB IDE 根据可装入项目的位置确定可装入项目路径是相对路径还是绝对路径。如果主项目目录包含可装入项目，则引用为相对路径。否则，引用为绝对路径。
   - 4.2. Relative——通过相对（相对于主项目）路径引用可装入项目。这是添加路径的首选方法。
   - 4.3. Absolute——通过绝对路径引用可装入项目。在某些情况下，该模式很有用，但是当将项目从一个系统移至另一个系统时，通常需要修改这些路径。
5. 单击 **Add** 关闭该对话框。
6. 在 Project Properties 窗口中，如果要在编译当前项目时编译该项目，请确保选中“Include”（包含）复选框。
7. 如果添加了多个项目，则此处显示的顺序将决定将十六进制文件添加到当前项目的十六进制文件的顺序。使用 **Up**（向上）和 **Down**（向下）可更改顺序。
8. 单击 **Apply** 或 **OK** 接受更改。

下次编译当前项目时，也会编译此处列出的项目（如果选中了“Include”），它们的十六进制文件将与当前项目的十六进制文件合并，以创建单个输出十六进制文件。所有调试文件也将进行合并。

图 5-10. 添加可装入项目



将当前项目十六进制文件与其他十六进制文件合并

1. 单击 **Add Loadable File**。将出现 Add Loadable File 对话框。
2. 浏览至一个十六进制文件，选择该文件。
3. 在“Store path as”下，选择如何存储有关可装入文件路径的信息。可选择以下选项：
   - 3.1. Auto——允许 MPLAB IDE 根据可装入文件的位置确定可装入文件路径是相对路径还是绝对路径。如果主项目目录包含可装入文件，则引用为相对路径。否则，引用为绝对路径。
   - 3.2. Relative——通过相对（相对于主项目）路径引用可装入文件。这是添加路径的首选方法。
   - 3.3. Absolute——通过绝对路径引用可装入文件。在某些情况下，该模式很有用，但是当将项目从一个系统移至另一个系统时，通常需要更正这些路径。
4. 单击 **Add** 关闭该对话框。
5. 在 Project Properties 窗口中，如果添加了多个文件，则此处显示的顺序将决定将十六进制文件添加到当前项目十六进制文件的顺序。使用 **Up** 和 **Down** 可更改顺序。
6. 单击 **Apply** 或 **OK** 接受更改。

下次编译当前项目时，此处列出的十六进制文件将与当前项目的十六进制文件（`CurrentProject.X.Production.hex`）合并，以创建单个输出十六进制文件。即在同一文件夹中创建单个文件（`CurrentProject.X.unified.hex`）作为当前项目十六进制文件。

装入备用十六进制文件

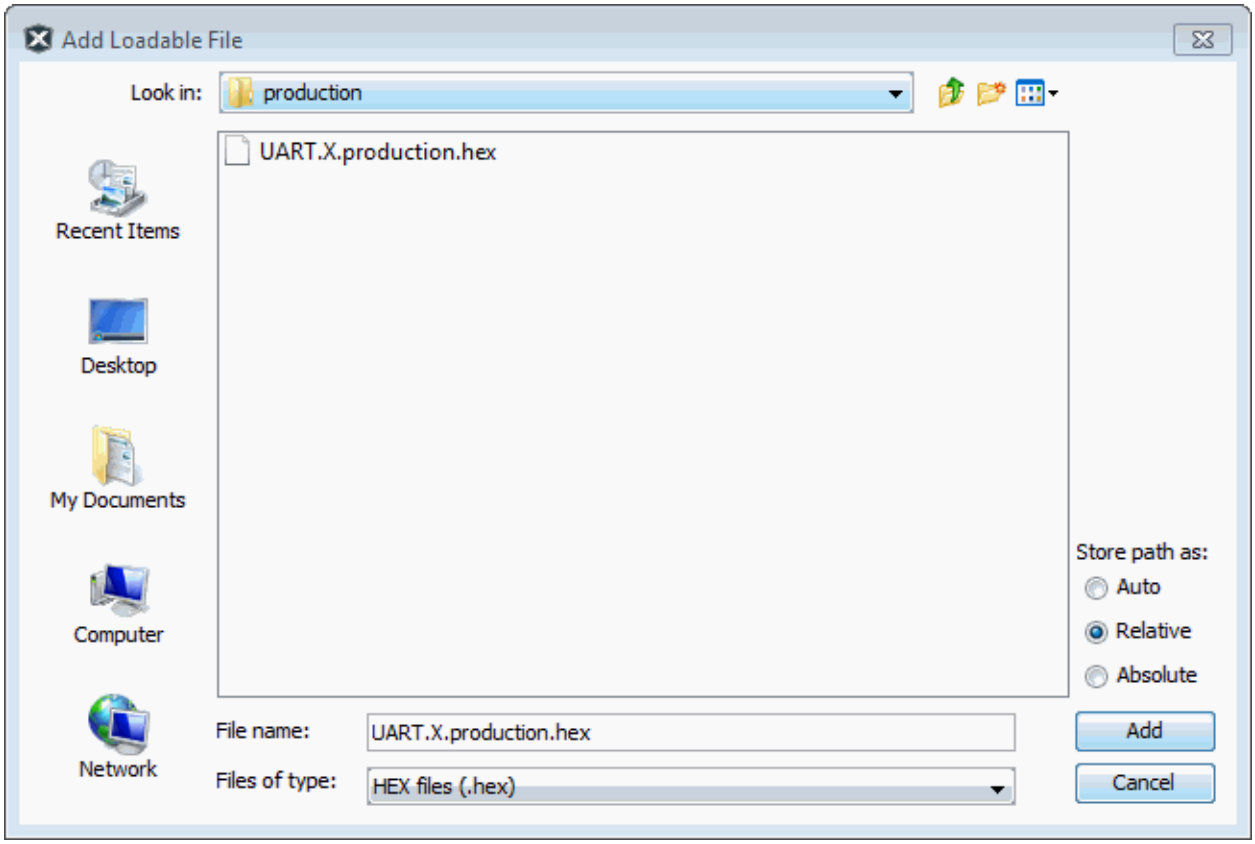
1. 单击以选中“Load this .hex file instead of ...”（装入该.hex 文件而非...）。
2. 单击省略号（...）按钮以打开 Add Loadable File 对话框。关于使用该对话框添加十六进制文件的详细信息，请参见上一主题。

下次编译当前项目时，备用十六进制文件将在编译完成时装入。

在编程/编译期间装入调试符号

“Load symbols when programming or building for production (slows process)”复选框允许您在编程时装入调试符号，例如将 RTDM 与 DMCI 和电机控制应用程序一起使用时。在其他情况下，应取消选中该复选框以加快编程和编译速度（默认）。

图 5-11. 添加可装入文件



5.5.3 使用可装入对象的首选方法

使用可装入对象的建议方法为：

1. 选择设置了器件配置位的项目并将其设为主项目（单击右键并选择“Set as Main Project”）的项目。可装入对象不应具有器件配置设置，因为这会与主项目发生冲突。
2. 将其他项目作为可装入对象添加到该主项目中。如果正装入的项目具有多个项目配置，请确保在装入时指定配置。

将包含可装入对象的项目指定为主项目时，可以确保在按下 Build 按钮时，所有可装入对象中的更改在编译操作中生效。

5.6 可装入项目和文件：自举程序

要将自举程序与应用程序代码合并：

1. 为应用程序创建一个项目，并为自举程序创建一个项目。
2. 将自举程序项目或十六进制文件装入应用程序项目中。关于如何执行该操作的信息，请参见 5.5.1 Projects 窗口——可装入对象设置或 5.5.2 Project Properties 窗口——装入设置。

下次编译应用程序项目时，所产生的十六进制文件将为一个合并的自举程序/应用程序十六进制文件。所有调试文件也将进行合并。

关于编译错误，请参见 5.5.3 使用可装入对象的首选方法或以下几节。

用于 PIC18 MCU 的 MPLAB C 编译器（MPLAB C18）

该编译器提供了应用程序启动代码（c018x.o），它从复位向量（地址 0）处开始，用于初始化软件堆栈，以及可选地初始化 idata 段，然后跳转到 main()。如果该启动代码保留在应用程序中，则总是会与自举程序代码复位发生冲突，您会收到一个关于数据冲突的链接器错误消息。

解决方法是对启动代码进行编辑，使之从地址 0 之外的位置开始。

MPLAB XC8——PIC18 MCU 示例

以下 Microchip 网络研讨会详细介绍了如何使用 MPLAB XC8 和 MPLAB X IDE 合并用于 PIC18 MCU 的自举程序与应用程序代码：Linking PIC18 Bootloaders and Applications

MPLAB XC16 自举程序示例

关于自举程序示例，请参见《MPLAB® XC16 C 编译器用户指南》（DS50002071E_CN）或帮助文件。

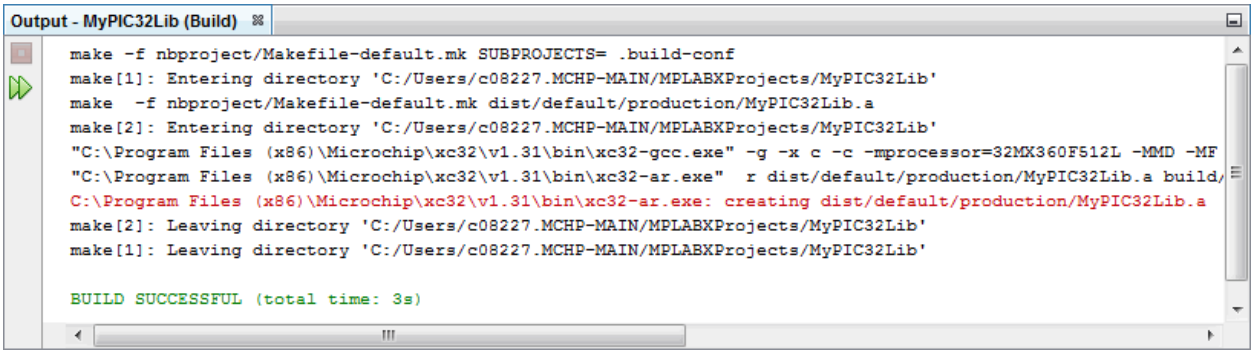
MPLAB XC32 自举程序示例

请参见 Microchip 网站上的《PIC32 自举程序》应用笔记（DS01388B_CN）。

5.7 库项目

创建一个将生成库文件（*.lib 或*.a，具体取决于使用的编译器）而非十六进制文件的项目（见红色文本）。该项目本身无法运行，因为它没有 main()函数。生成的库将用于另一个具有自己的 main()函数的项目。

图 5-12. 库项目示例



关于打开 New Project 向导的方法，请参见 4.1.2 启动 New Project 向导。

向导将启动，指导您完成新项目设置。单击 **Next** 转至下一步。

步骤 1.选择项目。选择“Microchip Embedded”类别，然后选择项目类型“Library Project”。

步骤 2.选择器件。从“Device”下拉列表中选择将在应用中使用的器件。要缩小选择列表，请先选择“Family”。

步骤 3.选择调试头。如果存在可用于选定器件的调试头，则会出现该步骤。要确定调试是否需要调试头，或器件是否具有片内调试电路，请参见以下文档之一：

- 处理器扩展包和调试头
- 仿真扩展包和仿真头

步骤 4.选择工具。从列表中选择将用于开发应用的开发工具。要确定针对您器件的工具支持，请参见 4.1.6.1 项目工具支持。

步骤 5.选择编译器。从列表中选择将用于开发应用程序的语言工具（编译器）。要确定针对您器件的工具支持，请参见 4.1.6.1 项目工具支持。

步骤 6.选择项目名称和文件夹。选择新项目的名称和位置。您可以浏览至一个位置。完成后单击 **Finish**。

新项目将在 Projects 窗口中打开。

可以根据需要将库项目更改为独立（应用程序）项目。有关详细信息，请参见 4.10.1 更改项目配置类型。

5.8 导入 Atmel Studio 7 或 Atmel START 项目

使用 New Project 向导或导入向导为 MPLAB X IDE 导入 Atmel Studio 7 项目或 Atmel START 导出文件。

可通过下述两种方法打开该向导。

导入项目选项

要打开导入 Atmel Studio 或 START 项目向导，请选择以下方式：

- `File>Import>Atmel Studio Project`（文件>导入>Atmel Studio 项目）
- `File>Import>START MPLAB Project`（文件>导入>START MPLAB 项目）

New Project 选项

要打开 New Project 向导，请执行以下操作之一：

- 在 Start Page 上，单击 **Learn & Discover** 或 **My MPLAB X IDE** 选项卡中“Projects”部分的“Create New”链接。
- 选择 *File>New Project*（或 Ctrl+Shift+N）。
- 单击工具栏上的“New Project”图标。

向导将启动，指导您完成新项目设置。

步骤 1.选择项目：选择“Microchip Embedded”类别，然后选择项目类型“Import Atmel Studio Project”或“Import Atmel START MPLAB Project”（导入 Atmel START MPLAB 项目）。单击 **Next**。

5.8.1 导入 Atmel 项目——查找项目文件

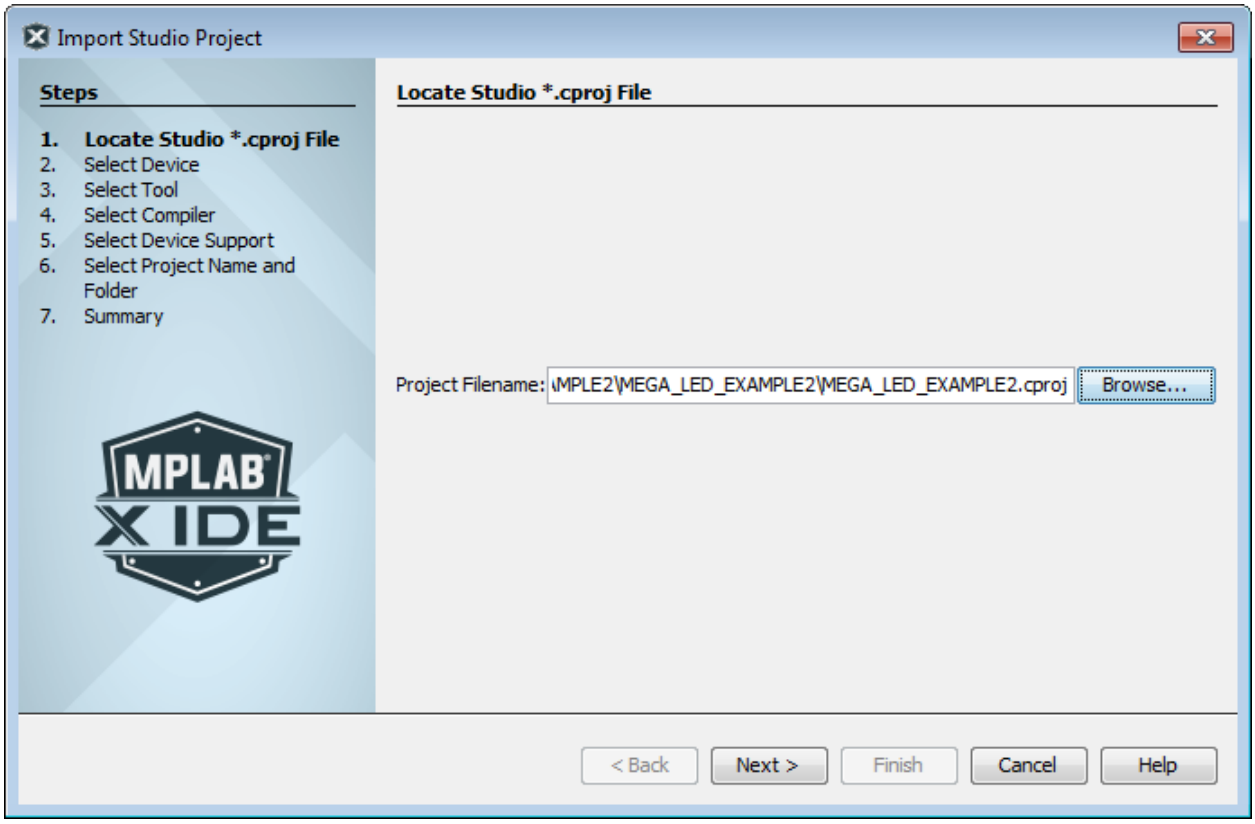
步骤 1 或 2.查找 Atmel Studio 或 START 项目文件

输入或浏览到所需的文件。例如，在 Windows 7 操作系统上，项目文件可能位于以下位置：

`C:\Users\Admin\Documents\Atmel Studio\7.0\`

对于 Atmel Studio，查找`*.cproj` 或`*.cppproj` 项目文件。对于 Atmel START，查找`*.atzip` 项目文件。

图 5-13. 导入 Atmel Studio 项目

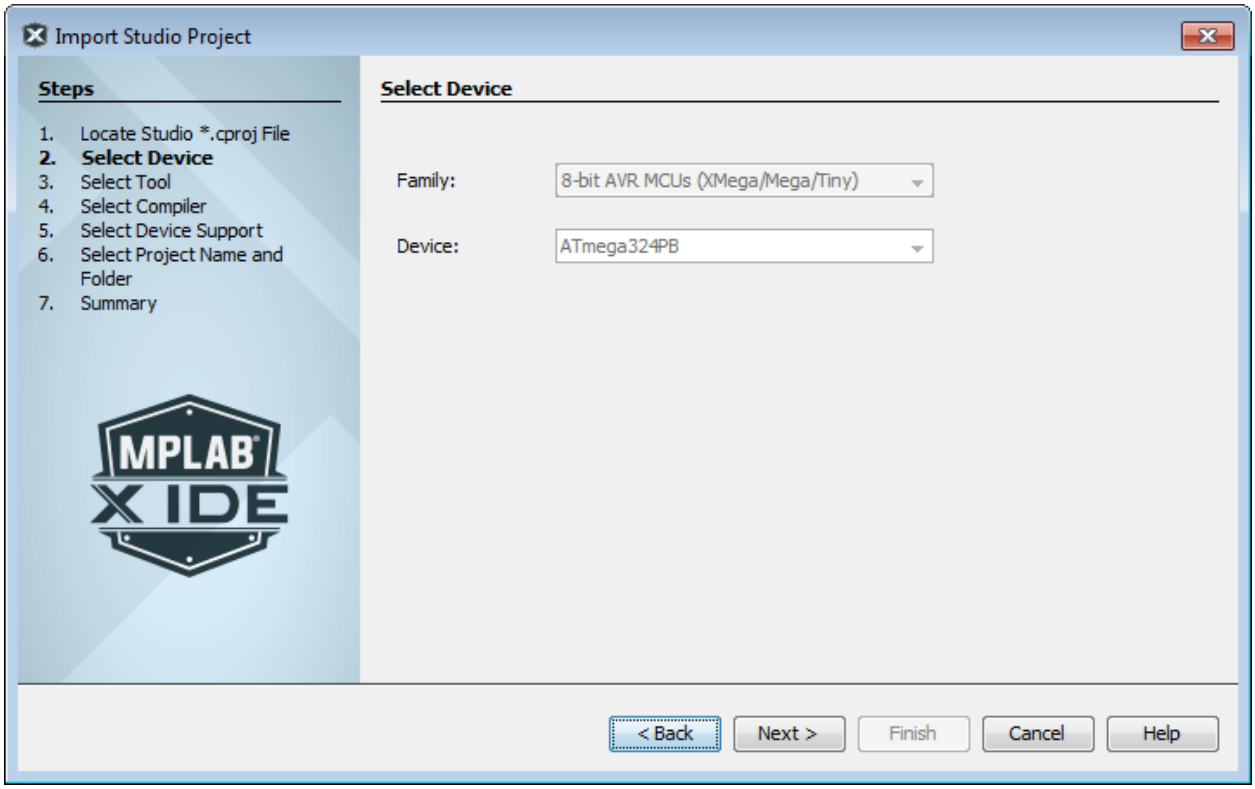


5.8.2 导入 Atmel 项目——选择器件

步骤 2 或 3.选择器件

该窗口将填充所导入项目中的器件信息。

图 5-14. 导入 Atmel 项目——选择器件



5.8.3 导入 Atmel 项目——选择工具

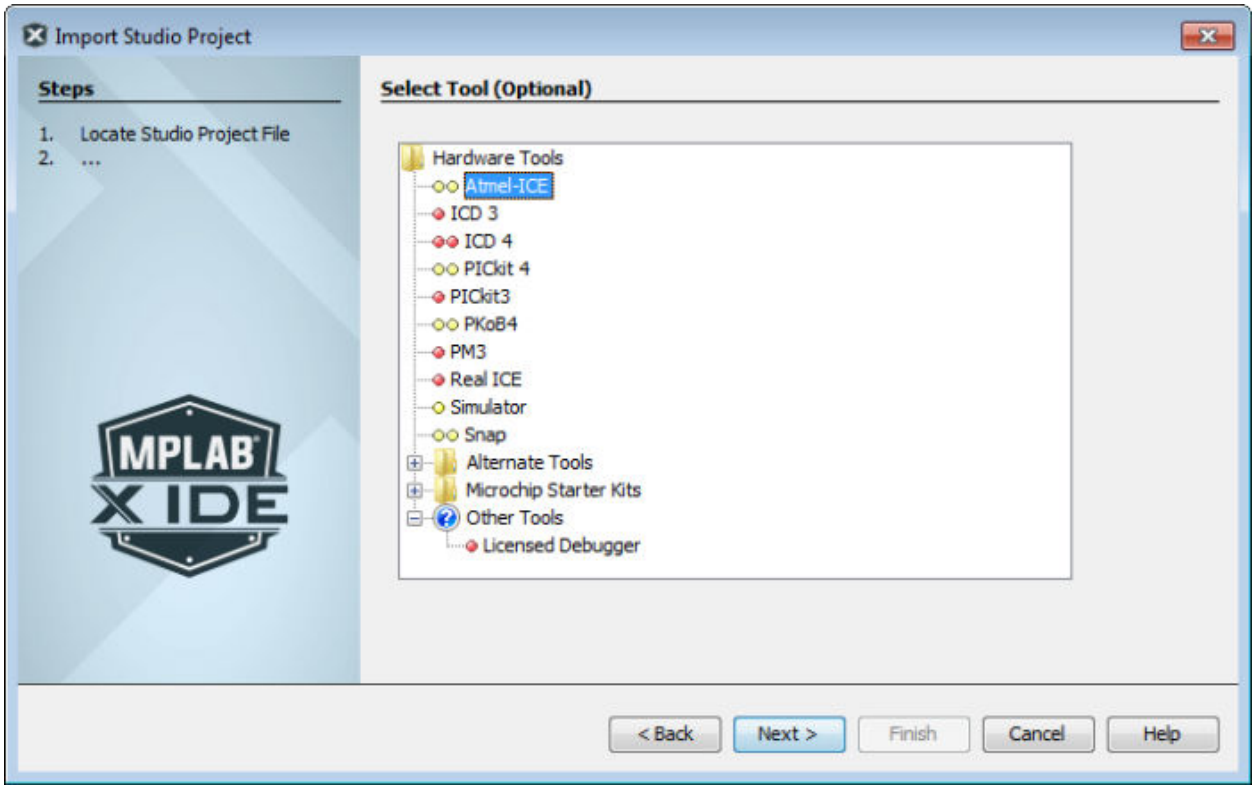
步骤 3 或 4.选择工具

该窗口将填充 Atmel 项目中的调试工具（如果适用于 MPLAB X IDE）。您可以从列表中选择其他工具，但是您可能希望将原始项目工具保留到导入完成，以确保项目正常工作。之后，您可以根据需要为项目选择其他工具。

工具名称旁边将显示所选器件的工具支持级别。

关于该显示的详细信息，请参见“Basic Tasks>Create New Project”（基本任务>新建项目）。

图 5-15. 导入 Atmel 项目——选择工具



5.8.4 导入 Atmel 项目——选择编译器

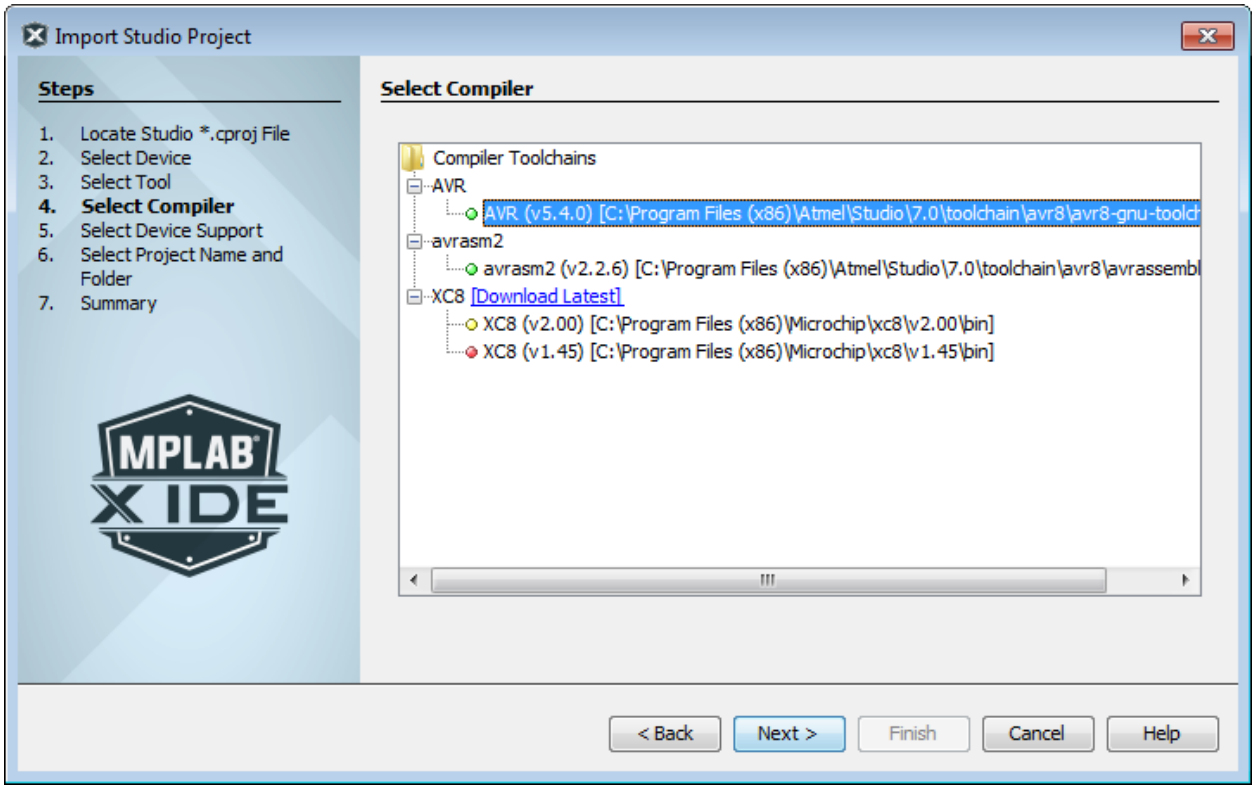
步骤 4 或 5.选择编译器

该窗口将填充 Atmel 项目中的编译器（如果适用于 MPLAB X IDE）。您可以从列表中选择其他编译器，但是您可能希望将原始项目编译器保留到导入完成，以确保项目正常工作。之后，您可以根据需要为项目选择其他编译器。

工具名称旁边将显示所选器件的工具支持级别。

如果未列出兼容的编译器，请单击 **Cancel**（取消）并按照“Basic Tasks>Create New Project”下的说明进行操作。

图 5-16. 导入 Atmel 项目——选择编译器



5.8.5 导入 Atmel 项目——选择器件支持

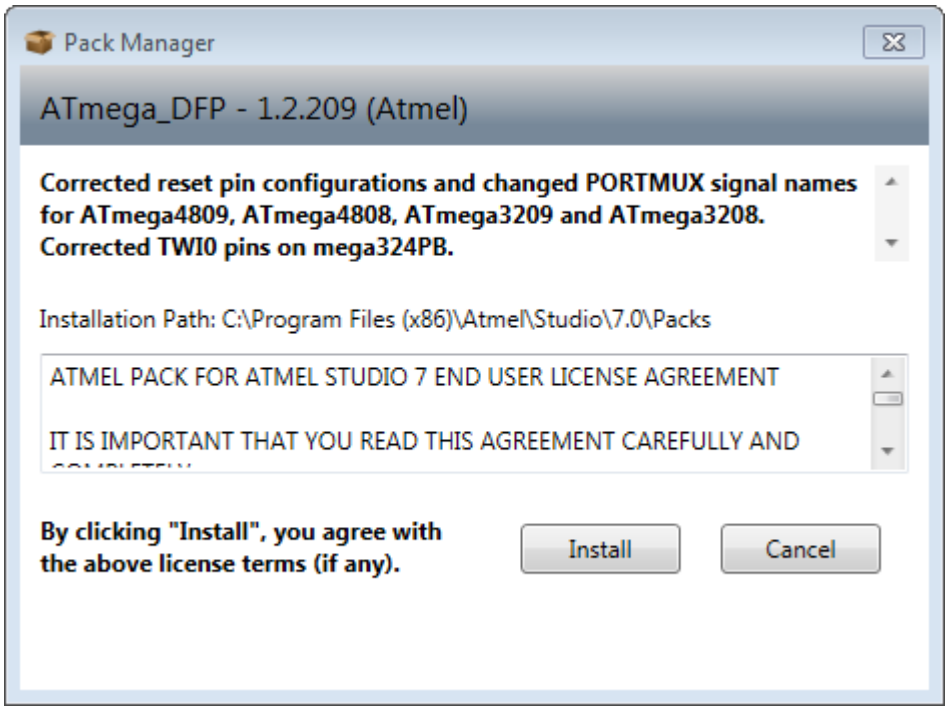
步骤 5 或 6.选择器件支持

自 MPLAB X IDE v5.00 起，器件受各版本器件包的支持。如果 MPLAB X IDE 不支持器件包版本：

1. 如果选中“Override Default Device Support”（覆盖默认器件支持），可以单击帮助图标（问号）按照说明下载包支持。下载包后，将安装在 Atmel Studio 中（见下图）。

图 5-17. 导入 Atmel 项目——选择器件支持

图 5-18. 器件包安装



2. 如果选中“Override Default Device Support”，可以在 Atmel Studio 包目录中查找正确的版本。例如：

   `C:\Program Files (x86)\Atmel\Studio\7.0\packs\atmel\ATmega_DFP\1.2.209`

3. 如果取消选中“Override Default Device Support”，可以在项目中使用 MPLAB X IDE 器件包版本（最新版本）。例如，MPLAB X IDE 项目包位于以下位置：

```
C:\Program Files (x86)\Microchip\MPLABX\v5.xx\packs\Atmel\ATmega_DFP
```

5.8.6 导入 Atmel 项目——选择项目名称和文件夹

步骤 6 或 7.选择项目名称和文件夹

建议不要更改默认名称和位置，以保持两个项目的可维护性。

项目文件位置：

对于 Atmel Start，源文件将被复制到 MPLAB X IDE 项目中。

对于 Atmel Studio，新项目不会将源文件复制到其文件夹中，而是引用 Atmel 项目文件夹中的文件位置。

要创建一个独立的 MPLAB X IDE 项目，需要创建一个新项目，并将源文件复制到其中。

主项目：

选中“Set as main project”可使该项目在导入后成为主项目。

项目位置：

对于 Atmel Studio，建议不要选中“Use project location as the project folder”，而应使 MPLAB X IDE 项目与 Atmel 项目位于同一位置。对于 Windows 操作系统，如果 Atmel 项目路径太长（见下文警告），请先更改该项目的路径，然后再导入到 MPLAB X IDE。

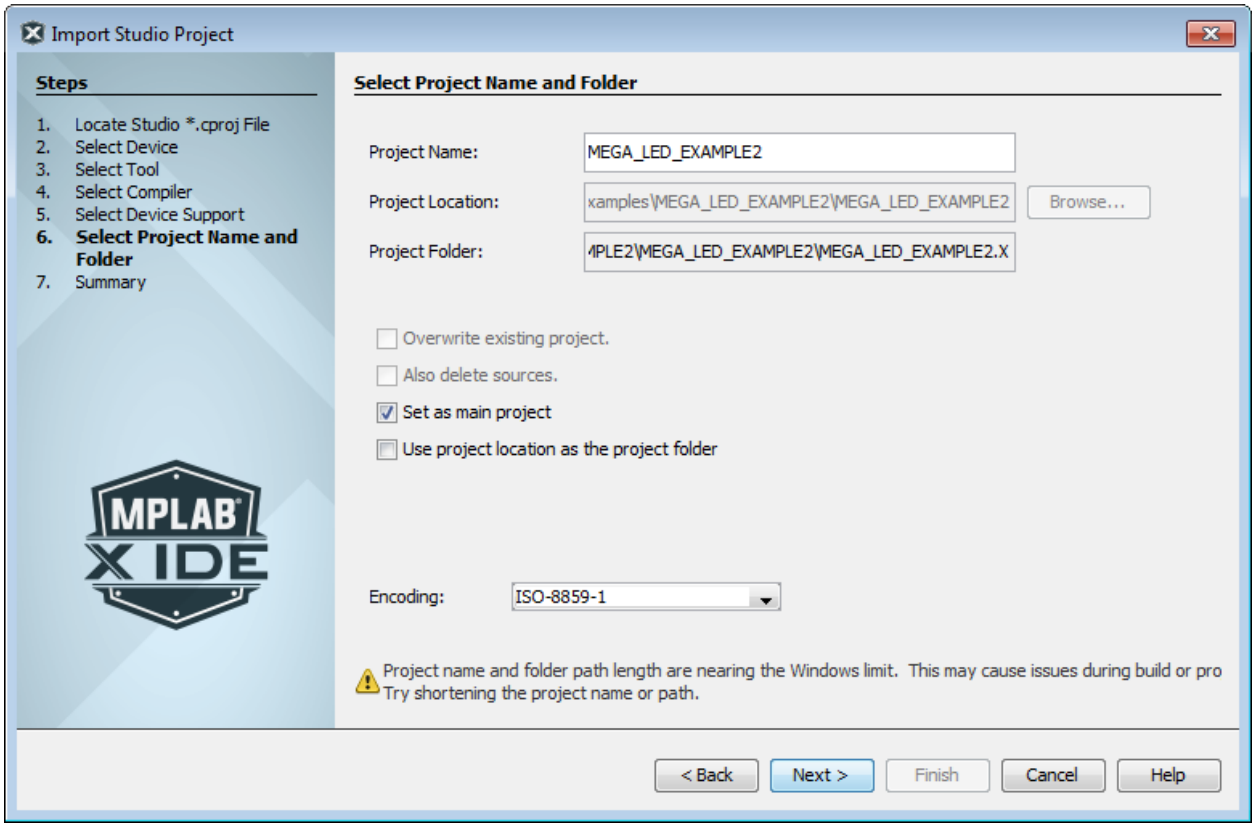
文件格式设置/编码：

选择的编码方式应与导入的项目中使用的编码方式相匹配。例如，如果格式为“950 (ANSI/OEM - Traditional Chinese Big5)”，则从下拉列表中选择“Big5”。

路径长度（Windows 操作系统）：

如果项目路径太长，则可能需要将其缩短以避免编译期间出现问题。

图 5-19. 导入 Atmel 项目——选择项目名称和文件夹



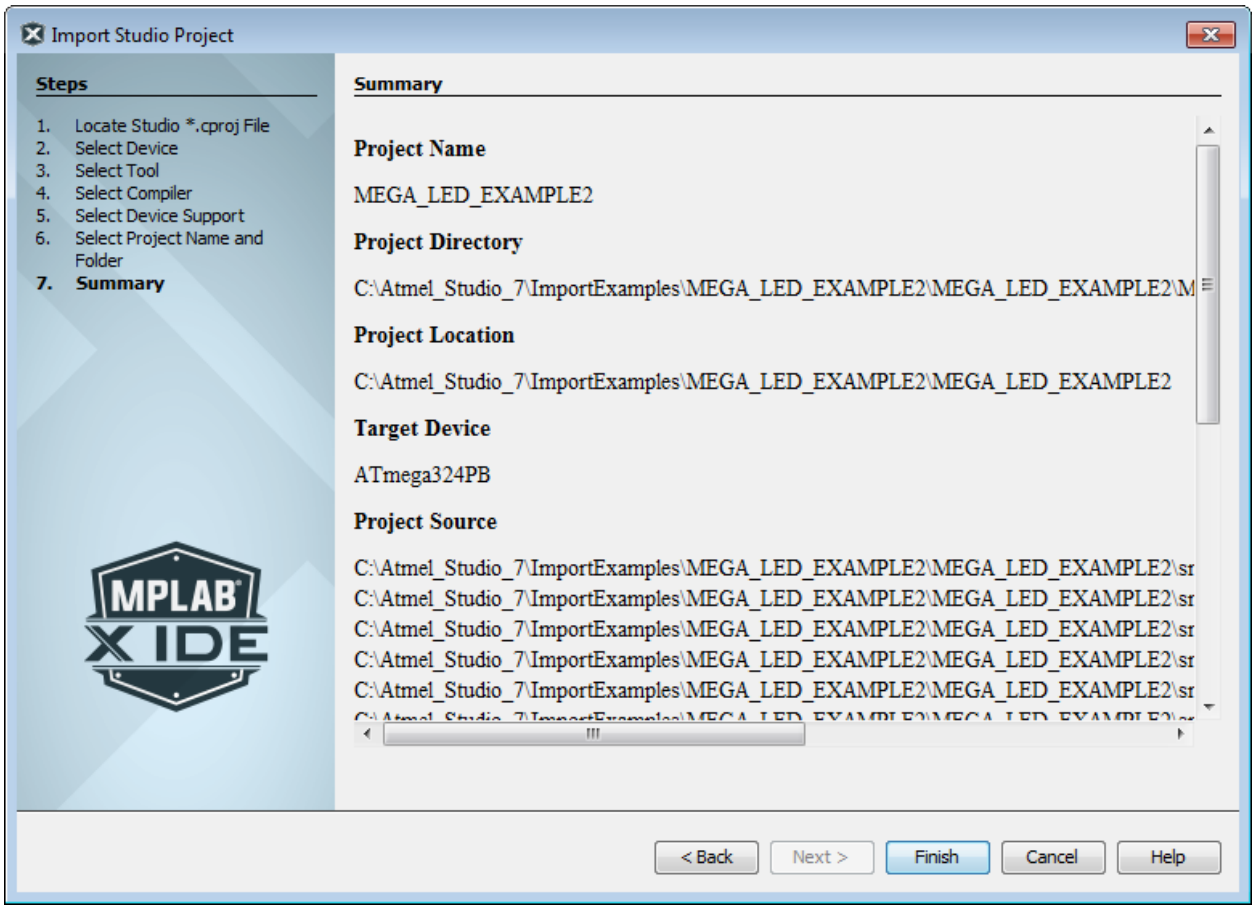
5.8.7 导入 Atmel 项目——摘要

步骤 7 或 8.摘要

在单击 **Finish** 之前查看摘要。如果有任何内容不正确，请使用 **Back** 按钮后退并更改它。

导入项目时，请注意 Output 窗口中的任何警告或其他消息。可能需要进行相应的更改才能编译新项目。

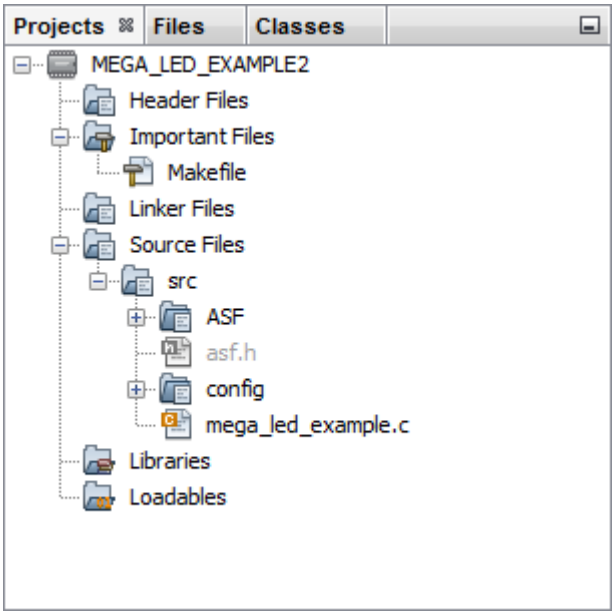
图 5-20. 导入 Atmel 项目——摘要



新项目将在 Projects 窗口中打开。

注： 原始 Atmel 项目中的源文件将进行链接，而非复制。因此，如果修改 Atmel 项目或 MPLAB X IDE 项目中的源文件，则这两个项目中都会收到通知。

图 5-21. 导入 Atmel 项目——Projects 窗口



5.9 其他嵌入式项目

MPLAB X IDE 可以基于选定的其他嵌入式项目创建项目。

1. 选择 *File>New Project*。
2. 单击“Categories”下的“Other Embedded”（其他嵌入式），并从可用嵌入式项目的列表中进行选择。
3. 继续创建一个 MPLAB X IDE 项目。

该功能会将现有文件导入 MPLAB X IDE 项目。尚不支持转换其他嵌入式项目设置或代码。

关于如何使用 MPLAB X IDE 的信息，请参见：

- 教程
- 基本任务

关于可用编译器的信息，请参见：

https://www.microchip.com/mplab/compilers

5.10 示例项目

创建示例项目可以帮助您了解关于 Microchip 器件、工具和 MPLAB X IDE 的信息。

1. 选择 *File>New Project*。
2. 在“Categories”下单击“Samples>Microchip Embedded”（示例>Microchip 嵌入式），并从可用嵌入式项目（控制演示板指示灯闪烁的项目）或模板项目的列表中进行选择。更多信息，请参见“说明”。

演示板的部件编号如下：

- Explorer 16 演示板：DM240001
- PICDEM 2 Plus：DM163022-1

5.11 使用其他文件类型

选择 *File>New File* 时，会为您列出许多文件类型。先前已介绍了 Microchip 编译器文件的使用。不过，您还可以根据项目语言工具或项目需要选择其他文件类型，从而创建一个特定文件。

表 5-4. 文件类型

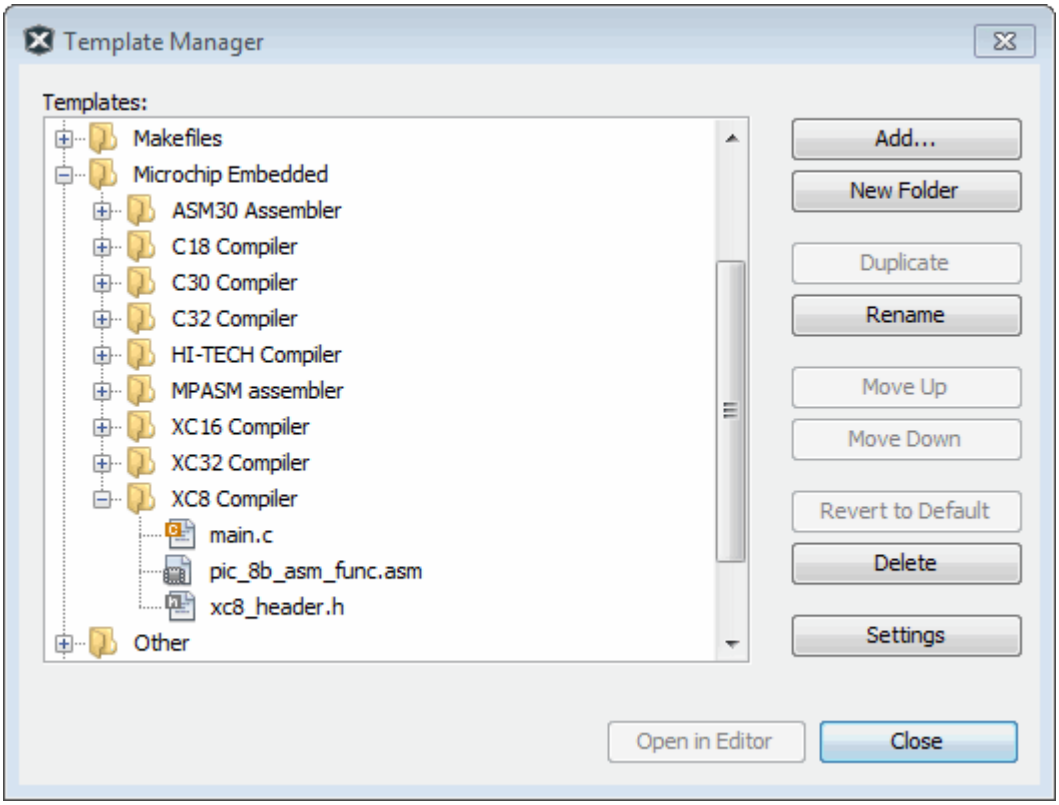
类别	文件类型
Microchip Embedded	用于 MPLAB X IDE 中支持的语言工具的文件 选择您的编译器文件夹可查看可用的文件类型。
C	通用 C 文件
C++	通用 C++文件
Assembler（汇编器）	通用汇编文件
Shell Scripts（Shell 脚本）	Shell 脚本文件：Bash、C 和 Korn 等
Makefiles	Makefile 文件
XML	XML 文件
Other（其他）	其他类型的文件，例如 HTML 和 JavaScript 等。 如果此处未列出您所需的文件类型，请选择“Empty File”（空文件）。在下一个窗口中，使用所需扩展名来命名文件。

5.12 修改或创建代码模板

创建要添加到项目中的文件时（4.9.2 创建一个新文件），将会对新文件使用模板。要更改该模板，请选择 *Tools>Templates*（工具>模板），然后选择“Open in Editor”（在编辑器中打开）来编辑模板。您也可以使用该对话框中的“Add”或“Duplicate”（复制）来创建新模板。

可以使用“New Folder”（新建文件夹）来创建用以存放模板的新文件夹。请注意，MPLAB X IDE 会筛选掉除 Microchip Embedded、Shell Scripts、Makefiles 和 Other 之外的所有文件夹，因此文件或文件夹应创建在这些文件夹下。

图 5-22. 模板管理器



您可以通过选择 *Tools>Options*（对于 macOS 为 *MPLAB X IDE>Preferences*），**Editor** 按钮，**Code Templates**（代码模板）选项卡来设置模板选项（见下图）。

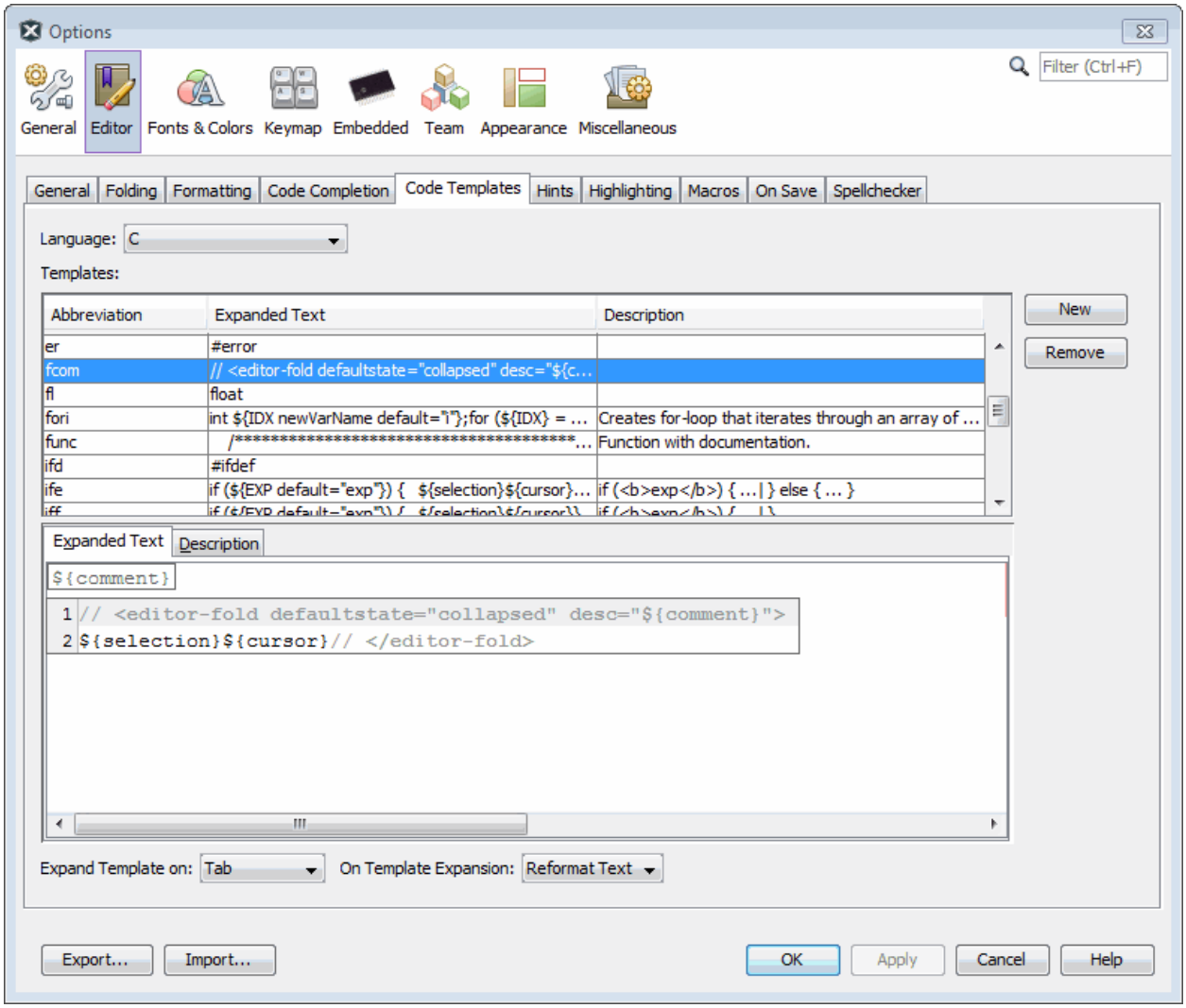
对于 C 代码，需要注意“fcom”选项。在编辑器窗口中输入“fcom”，然后按“Tab”键可在源代码中插入以下文本：

```
// <editor-fold defaultstate="collapsed" desc="${comment}">
${selection}${cursor}// </editor-fold>
```

该选项使您可以隐藏/查看代码段。

在 **Code Template** 选项卡中，选中“fcom”将在“Expanded Text”（展开的文本）部分显示“${comment}”。将鼠标悬停在该文本上可查看全文。

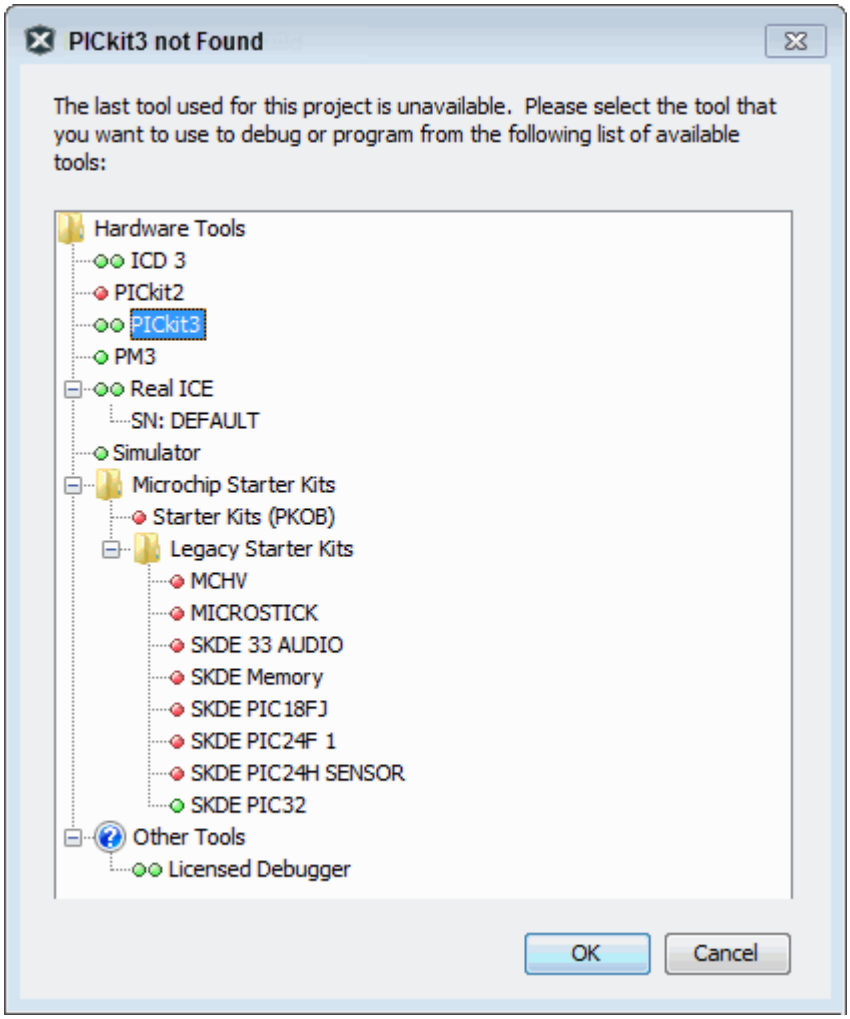
图 5-23. 模板选项



5.13 切换硬件或语言工具

当尝试运行或调试一个项目而该项目选择的调试工具与当前连接的工具不同时，会显示一个对话框，允许您为该项目选择其他硬件工具。

图 5-24. 切换硬件工具对话框



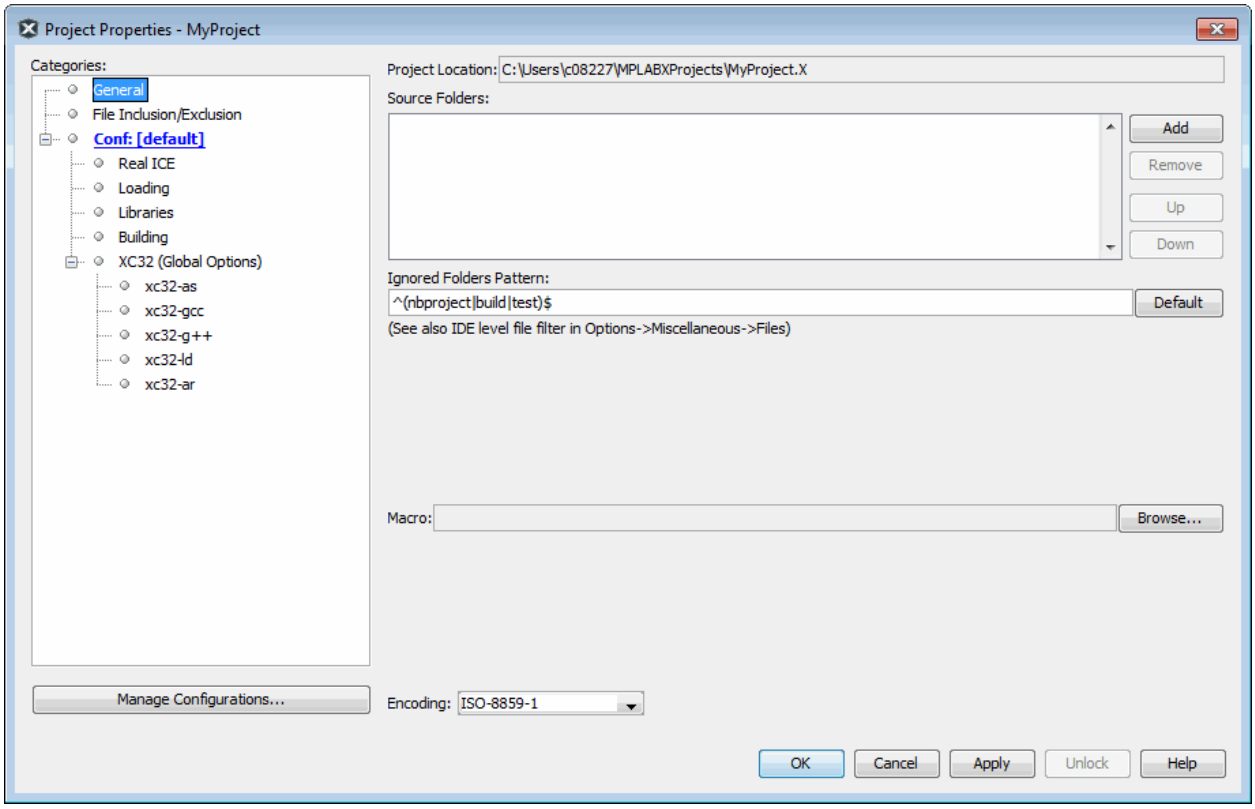
您还可以插入两个或更多硬件工具，并在 Project Properties 窗口（见 4.3 打开 Project Properties）中切换它们。

要在编译器工具链（语言工具）的不同版本之间切换，请使用 Project Properties 窗口。

5.14 修改项目文件夹和编码

当您创建项目时，需要指定项目文件夹和编码。创建项目之后，您可以使用 Project Properties 窗口中的“General”类别添加或忽略项目文件夹和更改项目编码。

图 5-25. 项目属性——常规



Project Location（项目位置）

查看当前项目位置。要更改项目位置，请参见 8.9 移动、复制或重命名项目。

Source Folders

添加供 MPLAB X IDE 查找项目文件时搜索的文件夹。

注： 添加项目文件夹之外的文件可能会使项目的可移植性降低。

Ignored Folders Pattern（忽略文件夹模式）

根据指定的正则表达式模式忽略项目文件夹中的文件夹。

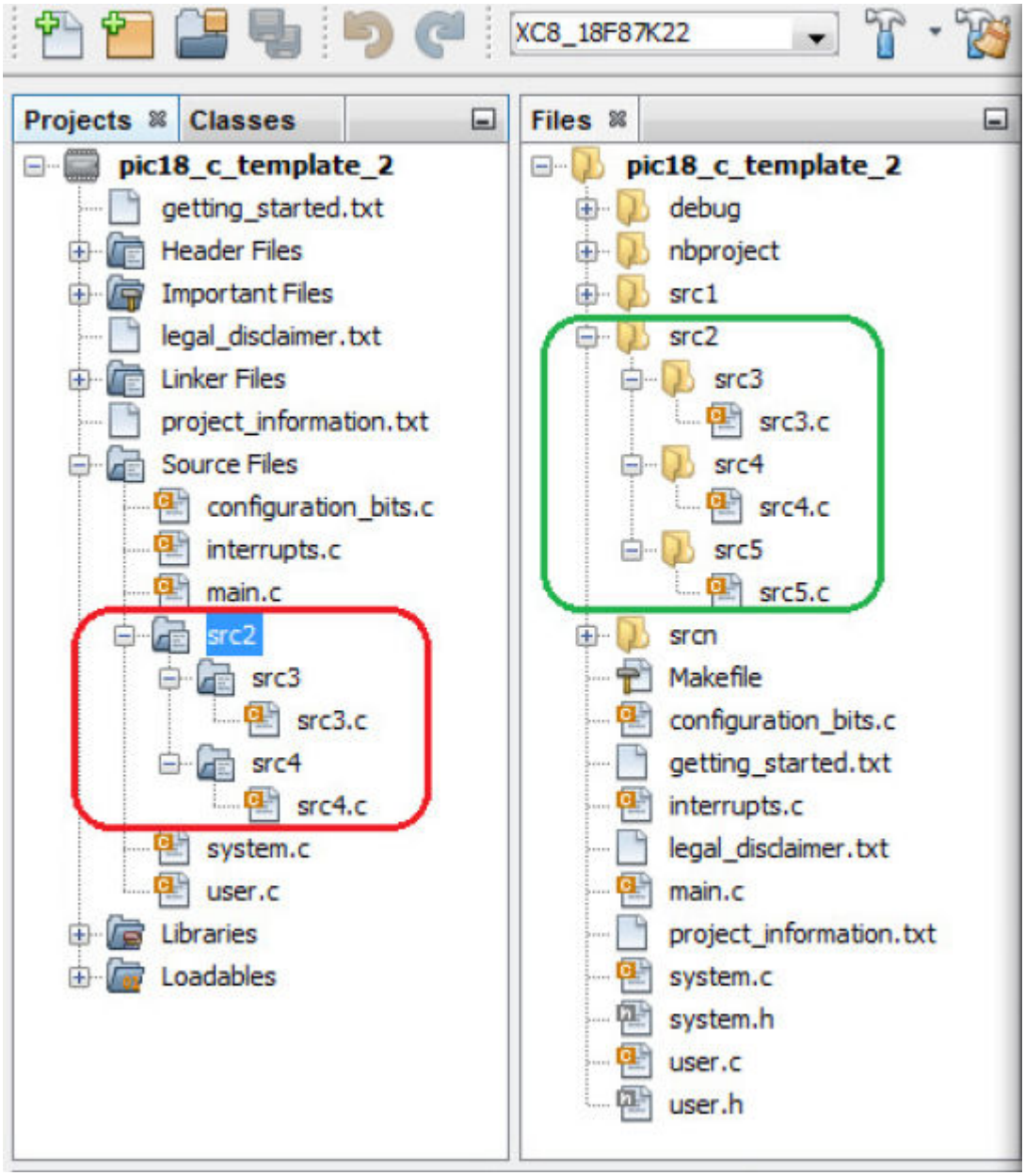
示例： 排除文件夹 `src5`

要从项目中排除 `src5` 文件夹及其内容，请在“Ignored Folders Pattern”文本框中输入以下正则表达式：

```
^(src5|nbproject|build|test)$
```

查看 Projects 窗口会发现 `src5` 文件夹不存在。查看 Files 窗口会发现该文件夹仍存在，但实际上已被从项目中排除。

图 5-26. 从项目中排除文件夹



项目编译完成后，请确保没有提及 Output 窗口的 Build 选项卡中包含的 `src5` 文件夹或 `scr5.c` 文件。

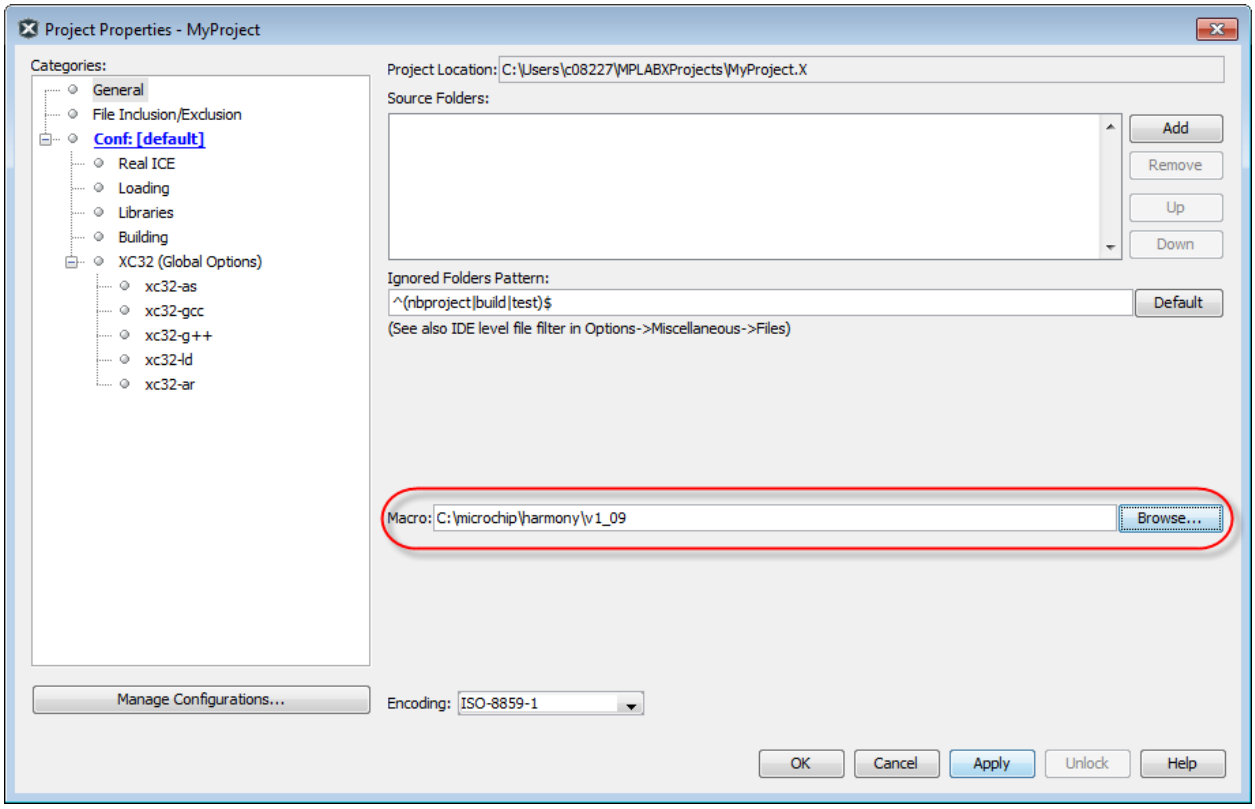
Macro（宏）

在“Add Existing Items”（添加现有项）选择对话框中输入要使用的源文件的路径，以链接到指定的源文件。

示例：MPLAB Harmony 文件

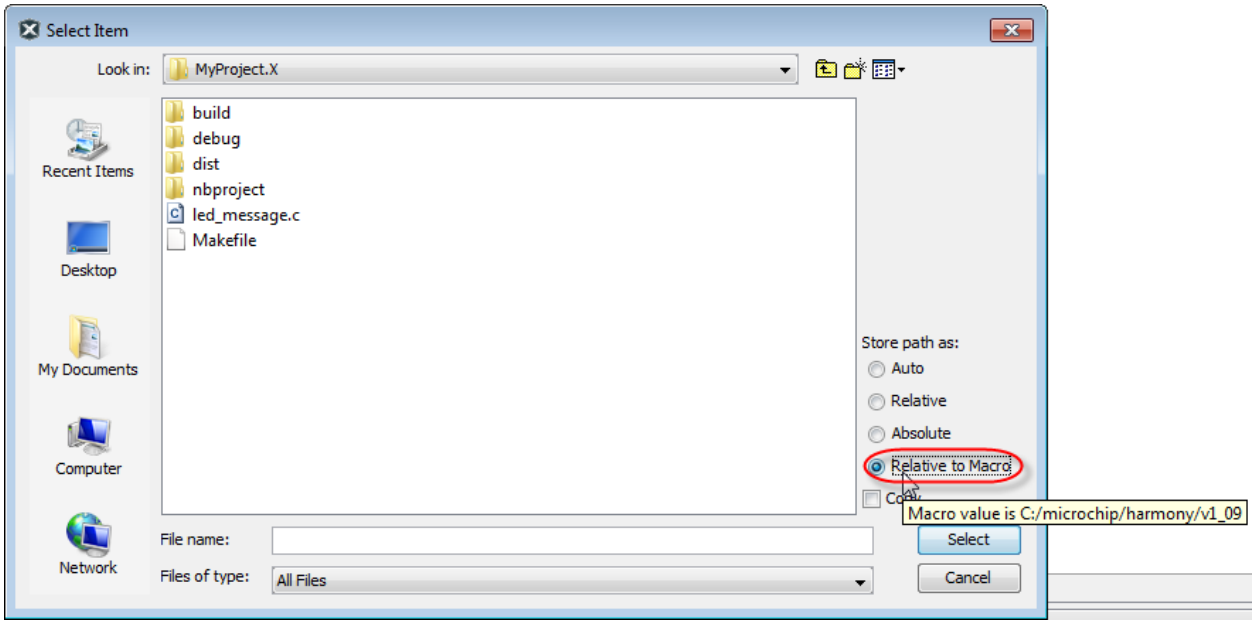
选择 MPLAB Harmony 源文件的路径。

图 5-27. 输入要用于宏的路径



转到 Projects 窗口，右键单击“Source Files”文件夹，然后为“Add Existing Items”选择其中一个选项。在对话框中，可选择从 Macro 中选择文件。将鼠标悬停在“Relative to Macro”（相对于宏）上可以查看宏值。

图 5-28. 选择从 Macro 中选择文件



Encoding

更改项目编码。该选择将指定代码语法着色标记，它可以在 *Tools>Options*（对于 macOS 为 *MPLAB X IDE>Preferences*），**Fonts and Colors** 按钮，**Syntax** 选项卡下进行编辑。

5.15 使用跑表

使用跑表来确定两个断点之间的时间。

要使用跑表：

1. 在要启动跑表的位置添加一个断点。
2. 在要停止跑表的位置添加另一个断点。
3. 选择 *Window>Debugging>Stopwatch（窗口>调试>跑表）*。单击窗口左侧的 **Properties** 图标，并选择启动和停止断点。
4. 再次调试程序，以获取跑表计时结果。

图 5-29. 跑表设置

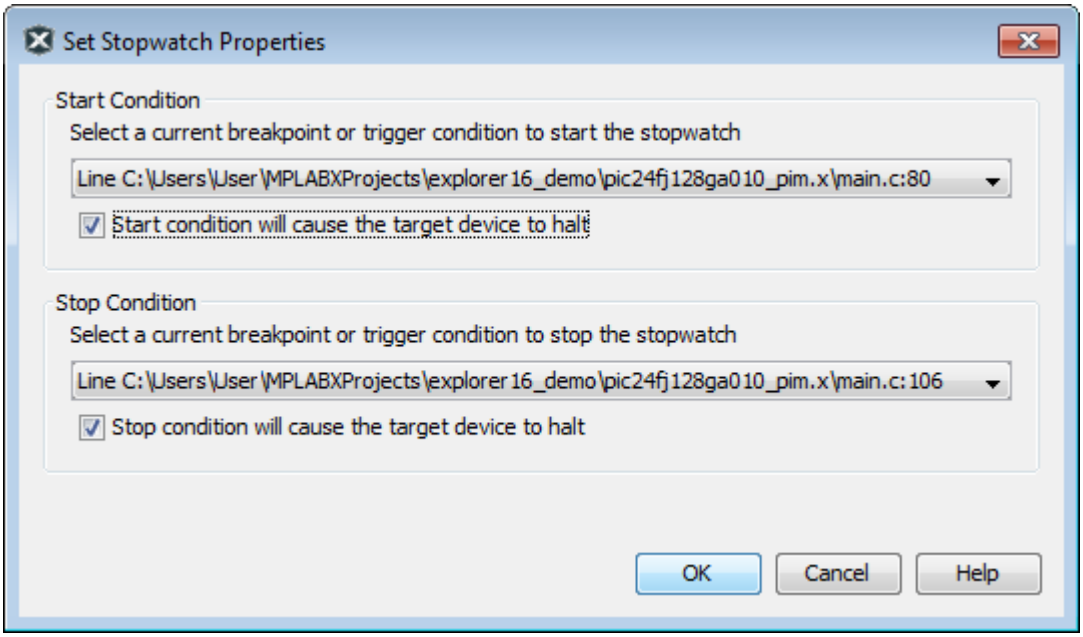
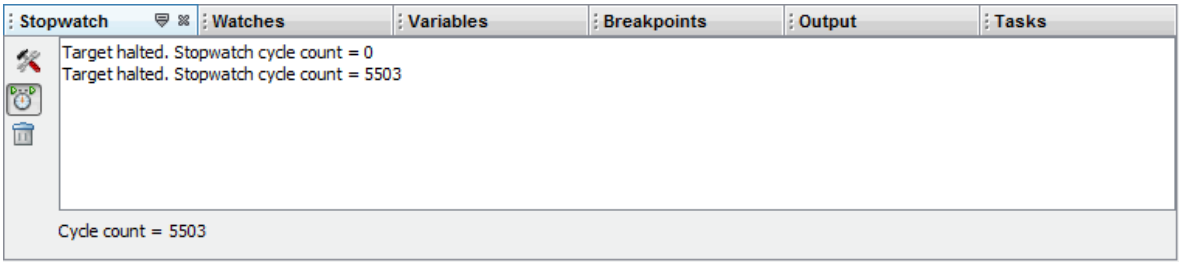


图 5-30. 具有内容的 Stopwatch 窗口



跑表在窗口左侧具有以下图标：

表 5-5. 跑表图标

图标	图标文本	说明
	Properties	设置跑表属性。选择一个当前断点或触发信号来启动跑表，另一个断点或触发信号用于停止跑表。
	Reset Stopwatch on Run（在运行时复位跑表）	在开始运行时将跑表时间复位为零。
	Clear History（清除历史记录）	清除跑表窗口。

.....（续）

图标	图标文本	说明
	Clear Stopwatch（清除跑表）	（仅对于软件模拟器）在复位器件之后复位跑表。

5.16 查看 Disassembly 窗口

在该窗口中查看反汇编的代码。选择 *Window>Debugging>Output>Disassembly Listing File*（窗口>调试>输出>反汇编列表文件）打开窗口。必须在调试会话（Debug>Debug Project）中暂停才能查看窗口内容。

在链接器生成的列表文件中也可以找到该信息。通过选择 *File>Open File*（文件>打开文件）并浏览查找 `ProjectName.lst` 文件打开该文件。

查看整个反汇编文件的一种快速方法是在 Disassembly 窗口中单击右键并选择“Disassembly Listing File”。

注： Disassembly 窗口将反汇编每一条指令，但没有与指令相关联的存储区历史记录。因此，在窗口中显示的 SFR 名称将为对应于存储区 0 的名称。

图 5-31. Disassembly 窗口（鼠标悬停显示变量值）

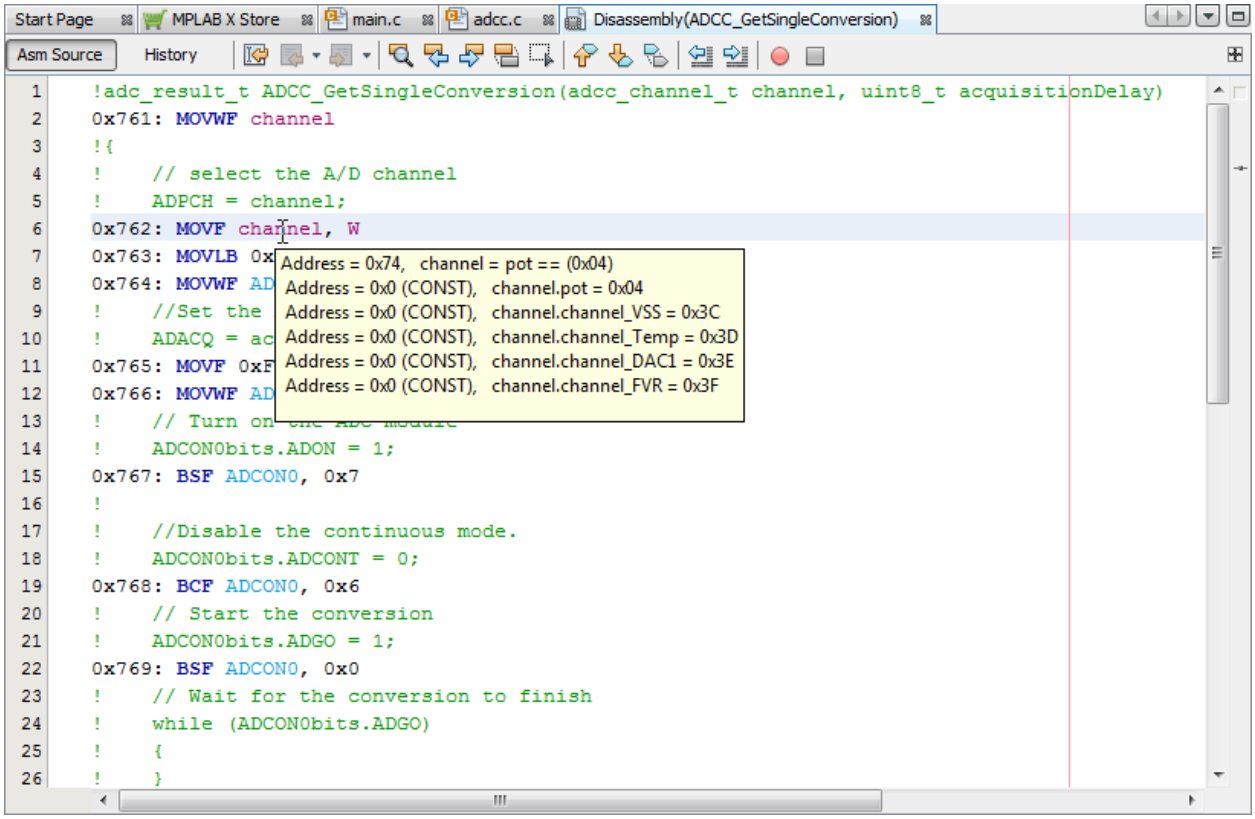


表 5-6. Disassembly 窗口上下文菜单

菜单项	说明
Step Instruction（单步执行指令）、Step Over 和 Run to Cursor	请参见 4.15 单步执行代码。
Cut（剪切）、Copy 和 Paste（粘贴）	在窗口中剪切、复制或粘贴所选文本。
Select in Projects（在项目中选择）	打开 Projects 窗口，然后选择包含所选内容的文档。

.....（续）	
菜单项	说明
Disassembly Listing File	查看反汇编列表文件的内容。

5.17 查看调用堆栈

对于 16 位和 32 位器件，可以通过软件 Call Stack（调用堆栈）窗口查看正在执行的 C 代码中的 CALL 和 GOTO 指令。该窗口不适用于汇编代码（使用调用堆栈时，建议关闭代码优化）。

Call Stack 窗口将显示函数及其参数，按照它们在正执行的程序中的调用顺序列出。

要查看调用堆栈：

1. 调试，然后暂停程序。
2. 选择 *Window>Debugging>Call Stack*（窗口>调试>调用堆栈）。此时将打开 Call Stack 窗口。另请参见 12.4 Call Stack 窗口。

关于调用堆栈的更多信息，请观看 NetBeans 视频：

https://www.youtube.com/watch?v=iS8OOUPmTUc

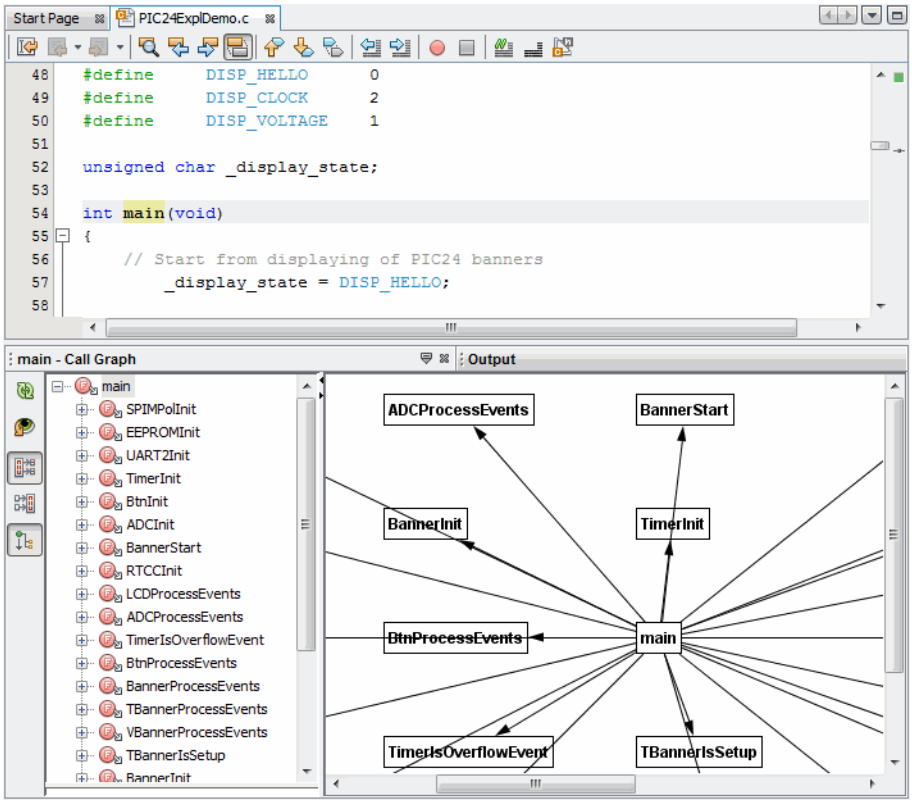
5.18 查看调用图

Call Graph（调用图）窗口会显示从选定函数调用的函数或调用该函数的函数的树视图。

要查看调用图：

- 右键单击一个函数并从下拉菜单中选择“Show Call Graph”（显示调用图）。

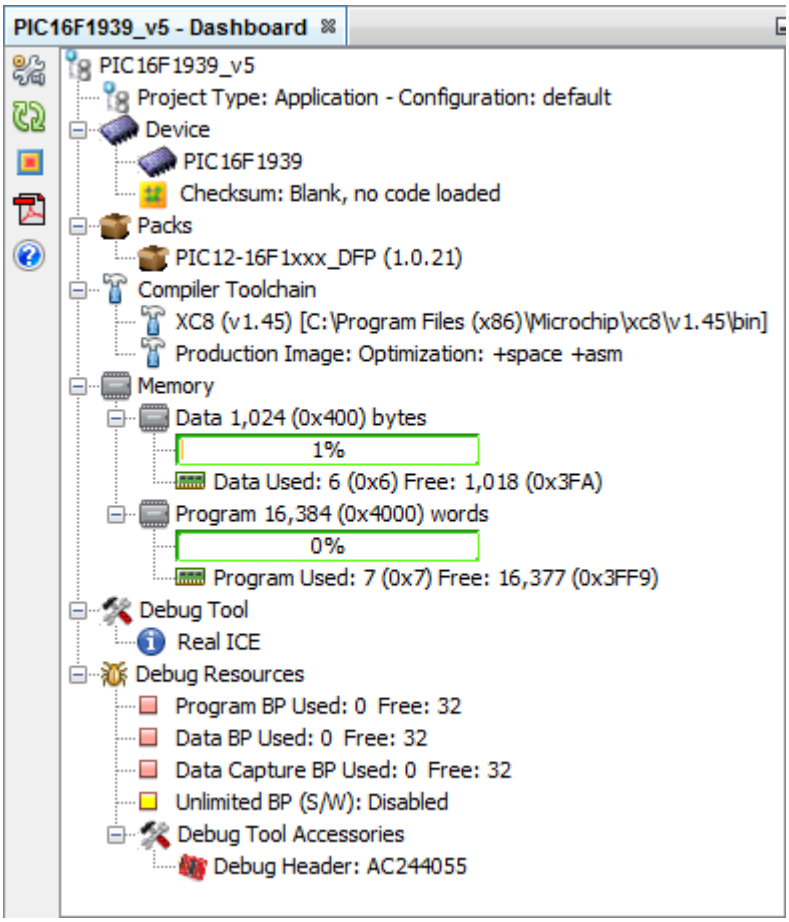
图 5-32. 主函数的调用图



5.19 查看仪表板显示

选择 *Window>Dashboard*（窗口>仪表板）可显示项目信息。

图 5-33. 仪表板显示



显示的项目将为：

- 如果未选择主项目，则显示 Projects 窗口中的活动项目。在 Projects 窗口单击某个项目，可使其成为活动项目。
- 主项目。在 Projects 窗口右键单击某个项目，然后选择“Set as Main Project”。
- 将不显示任何其他项目（活动或非活动）。

5.19.1 仪表板组

Dashboard 窗口项目信息分为几个特定的部分。

表 5-7. 仪表板组

组	定义和内容
Project Name（项目名称）	• **活动项目/主项目**的名称 • **项目类型**（应用程序或库）以及**项目配置**（默认或用户自定义）
Device	• 项目器件 • 在运行或调试期间生成的任何**状态标志** • 装入器件存储器中的内容的**校验和**。 必须编译才能看到它。请参见 6.14 校验和。

.....（续）

组	定义和内容
Packs	包含各版本器件信息（.PIC 文件和其他文件，具体取决于器件）的包。
Compiler Toolchain	• **项目工具链名称**（例如 XC8），工具**版本号**（在( )中）和可执行文件**路径**（在[ ]中）。 • **映像类型**（生产或调试）和工具链**优化**级别，对应于许可证类型。 关于优化的更多信息，请参见语言工具文档。 • 关于**编译器许可证***的信息： – 许可证类型：工作站或网络 – 许可证模式：FREE（免费版）、STD（标准版）或 PRO（专业版） – 许可证席位：可用席位总数 – 可用许可证：可用许可证数 – HPA——距离 HPA 结束的天数 – 到期——距离许可证到期的天数 – 按下以查看状态——在 Output 窗口中查看状态 * 必须至少使用 MPLAB XC8 v1.34、MPLAB XC16 v1.25 或 MPLAB XC23 v1.40 才能予以显示。 关于编译器许可证的更多信息，请参见： https://www.microchip.com/mplab/compilers
Memory	• 禁止使用情况符号。单击消息可使能装入符号。在 Project Properties 窗口的 Loading 类别中，选中“Load symbols when programming or building for production (slows process)”。 • 器件上的**存储器类型**（数据或程序）以及**可用存储空间**。 • **Memory Used**（已用存储器）可指示剩余的存储空间。 编译器会输出存储器摘要，详细列出程序空间、配置位，ID 单元和 EEPROM（如果器件上存在）的使用情况。 这些存储空间的总和（以字为单位）应与仪表板上显示的 Program Used（已用程序存储空间）一致。如果器件的存储空间较小，则应检查映射文件。
Debug Tool（调试工具）	**硬件调试工具** • 调试工具**名称**和序列号 • 调试工具**连接**。默认情况下，工具连接始终处于活动状态。要更改该状态，请转至 *Tools>Options*（对于 macOS 为 *MPLAB X IDE>Preferences*），**Embedded** 按钮，**Generic Settings** 选项卡，“Maintain active connection to hardware tool”。 • 单击 **Refresh Debug Tool Status**（刷新调试工具状态）按钮可查看硬件调试工具固件版本和当前电压。 **软件模拟器** • 调试工具**名称**，即软件模拟器。 • 单击 **Click for Simulated Peripherals**（单击查看模拟的外设）可查看所支持器件外设的列表。
Debug Resources（调试资源）	**硬件断点** 当前**正在使用**的断点数量和**可用于**项目器件的断点数量。 **软件断点** 项目器件上是否**支持**软件断点。 **调试工具附件** 该项目中使用了哪些调试工具附件（例如调试头）。

5.19.2 仪表板图标

通过 Dashboard 窗口左侧的图标可访问相应功能。

表 5-8. 侧边栏图标

图标	功能
	**Project Properties** 显示 Project Properties 窗口。
	**Refresh Debug Tool Status** 单击它可查看硬件调试工具详细信息。
	**Toggle Software Breakpoint - Enabled/Disabled（翻转软件断点——使能/禁止）** 单击它可使能或禁止软件断点。 图标中会显示该功能的状态： • 红色中心：禁止 • 绿色中心：使能 • 黑色中心：不支持——超出器件或工具可用的断点数量时。
	**Open Device Data sheets（打开器件数据手册）** 从 Microchip 网站获取器件数据手册： https://www.microchip.com/ 单击可打开已保存的本地数据手册或打开浏览器以转至 Microchip 网站来搜索数据手册。
	**Compiler Help（编译器帮助）** 单击可打开编译器安装目录的 docs 文件夹中文档的主索引（如可用）。

5.20 查看项目的寄存器（I/O View）

使用 I/O View 窗口（*Window>Debugging>IO View*）可查看当前项目的目标器件的寄存器概览。它可作为设计过程中的快速参考，能够在项目处于调试模式时显示寄存器值。

I/O View 是一个拆分窗格视图，顶部窗格中包含外设，底部窗格中包含所选外设的寄存器。

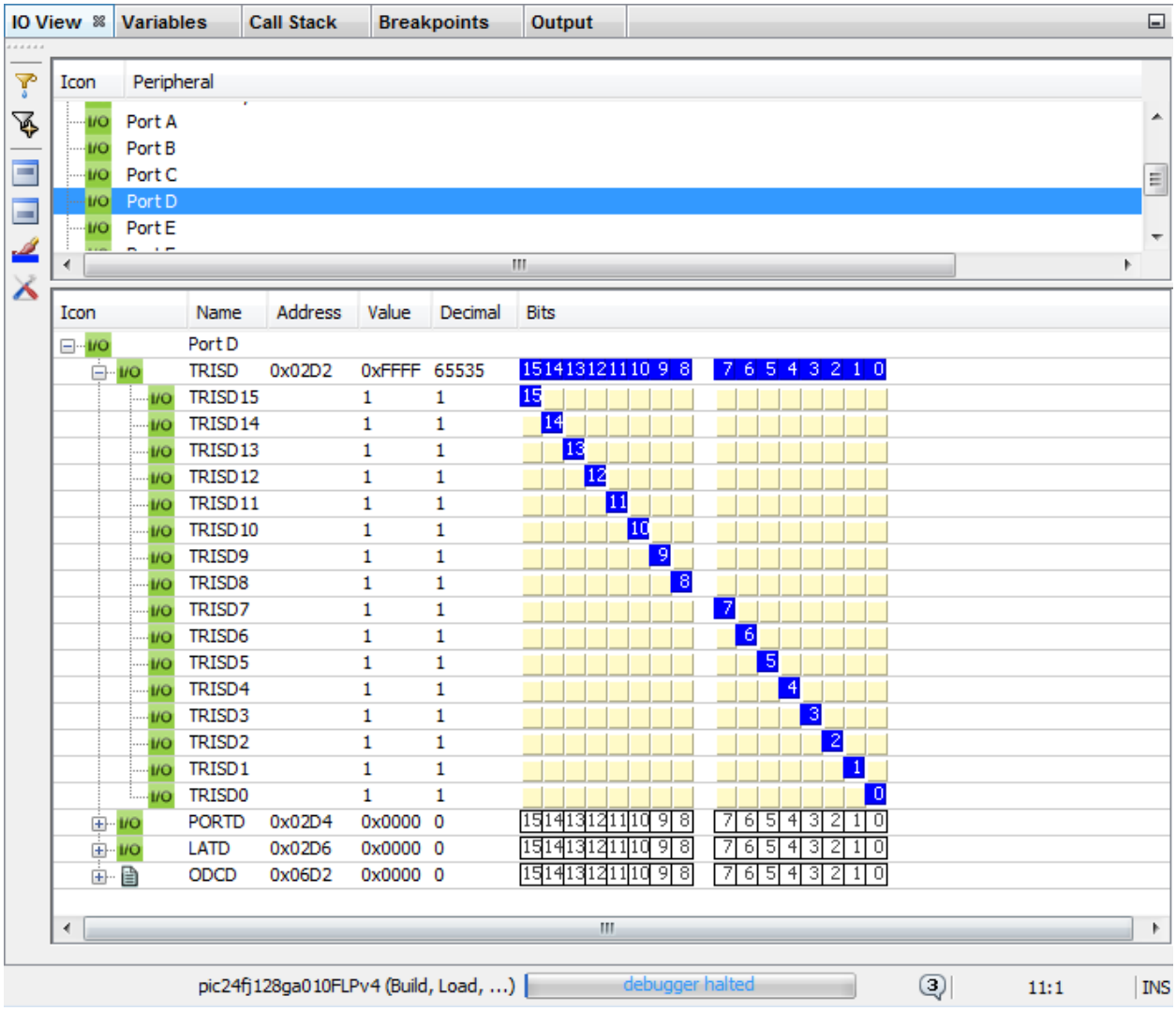
- 顶部窗格显示寄存器并在组合柜中提供其指定编辑值，例如 Config 组合框。
- 底部窗格显示寄存器字，带有用于编辑的定制位控制和已更改的彩色位。

关于该窗口的更多信息，请参见 12.8 IO View 窗口。

在调试暂停期间编辑

调试模式暂停后，您可以在窗口中编辑值。

图 5-34. 调试暂停时的 I/O View



要更改某个位，只需单击该位即可翻转值。寄存器中的每一位都显示在单独的列中。置 1 的位单元格将呈深色（默认），而清零的位将没有颜色（默认为白色）。

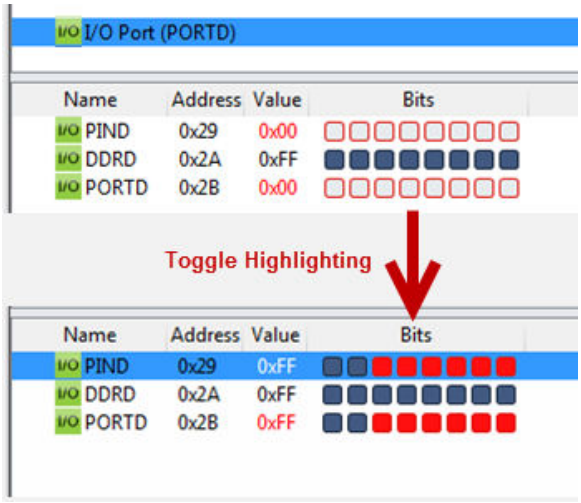
通过单击值字段并写入新值可以更改任何值。

有些值和位是只读的，因此无法修改，而有些位是只写的。更多信息，请参见各个器件的文档。位或值设置完成后，将立即从器件中读回，以确保 I/O View 仅显示器件中的实际值。如果设置了新值，但 I/O 视图未按预期更新，则该寄存器可能是只写的，或者根本无法访问。

更改颜色

如果某个寄存器在上一次显示之后发生更改，则显示画面中将以红色值和位进行指示。如果某个位在上一次显示之后已置 1，则将显示红色实心方块。如果已清零，则将仅显示红色边框。可以在工具栏中开启或关闭该功能。

图 5-35. I/O View——翻转已修改位的高亮显示



5.21 改进代码

通过使用代码重构和/或性能分析来改进代码。

重构代码是一种使代码变得简单而无需改变其功能的方法。当前可以对 C 代码执行以下操作：

- 查找整个文件中的函数使用实例
- 重命名整个文件中的函数和参数

更多信息，请参见 7.6 C 代码重构。

代码性能分析是在程序运行时对 CPU Usage（CPU 使用情况）、Memory Usage（存储器使用情况）和 Thread Usage（线程使用情况）工具进行检查。每当运行 C 项目时，都会自动运行性能分析工具。

如果需要使用 NetBeans IDE 菜单中的功能 （但此类功能在 MPLAB X IDE 中不常用或未显示），请转到 *Tools>Plugins>Installed>MPLAB Hidden Menu Filter*（工具>插件>已安装>MPLAB 隐藏菜单筛选器），然后选择此项并单击“Deactivate”（禁止）。

5.22 使用本地历史记录控制源代码

MPLAB X IDE 具有一个内置的本地文件历史记录功能，这是 NetBeans 平台的一项优势。该功能为本地项目和文件提供内置的版本控制支持，类似于传统的版本控制系统。可用工具包括本地 DIFF 和文件恢复。

要查看文件的本地历史记录：

- 单击编辑器窗口中的 **History**（历史记录）按钮
- 右键单击 Projects 或 Files 窗口中的文件，然后选择 *History>Show History*（历史记录>显示历史记录）

之前对文件所做的任何更改都将在此处列出。

图 5-36. 编辑器窗口中的本地历史记录

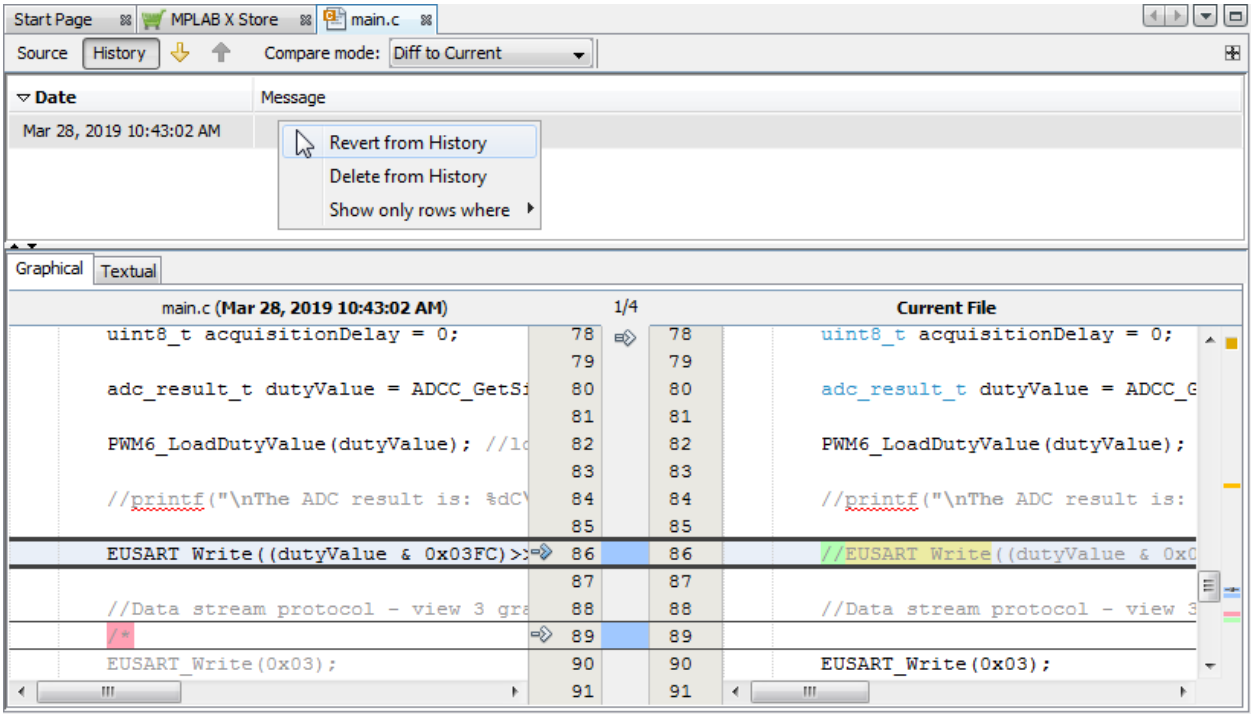


表 5-9. 本地历史记录上下文菜单

菜单项	说明
Revert from History（从历史记录还原）	还原到先前的版本。Revert to（还原到）对话框将打开，其中包含文档的所有先前版本。选择一个版本，并单击 **OK** 即可还原到该版本。
Delete from History（从历史记录中删除）	从历史记录中删除更改内容。 要还原从历史记录中删除的项，右键单击 Projects 或 Files 窗口中的文件，然后选择 *History>Revert Deleted*（历史记录>还原删除的内容）。
Show only rows where（仅显示满足条件的行）	在 Date（日期）列下单击右键，可选择日期筛选方式（Date ==（日期 ==）或 Date < >（日期< >））。 在 Message（消息）列下单击右键，可选择消息筛选方式（Message ==（消息 ==）或 Message < >（消息< >））。

5.23 使用版本控制系统控制源代码

开发包含许多文件的复杂应用程序时，尤其是作为团队的一部分时，控制源代码更改至关重要。MPLAB X IDE 提供了一种内置的源文件版本控制方法，并支持外部版本（源代码或版本）控制程序。

MPLAB X IDE 支持使用多种版本控制系统（Revision Control System，RCS）。经过设置后，可通过上下文菜单访问版本控制操作。

当前支持的源代码/版本控制系统包括 Git、Subversion 和 Mercurial。可通过插件支持其他系统（例如 CVS）。

将版本控制系统用于 MPLAB X IDE 项目时：

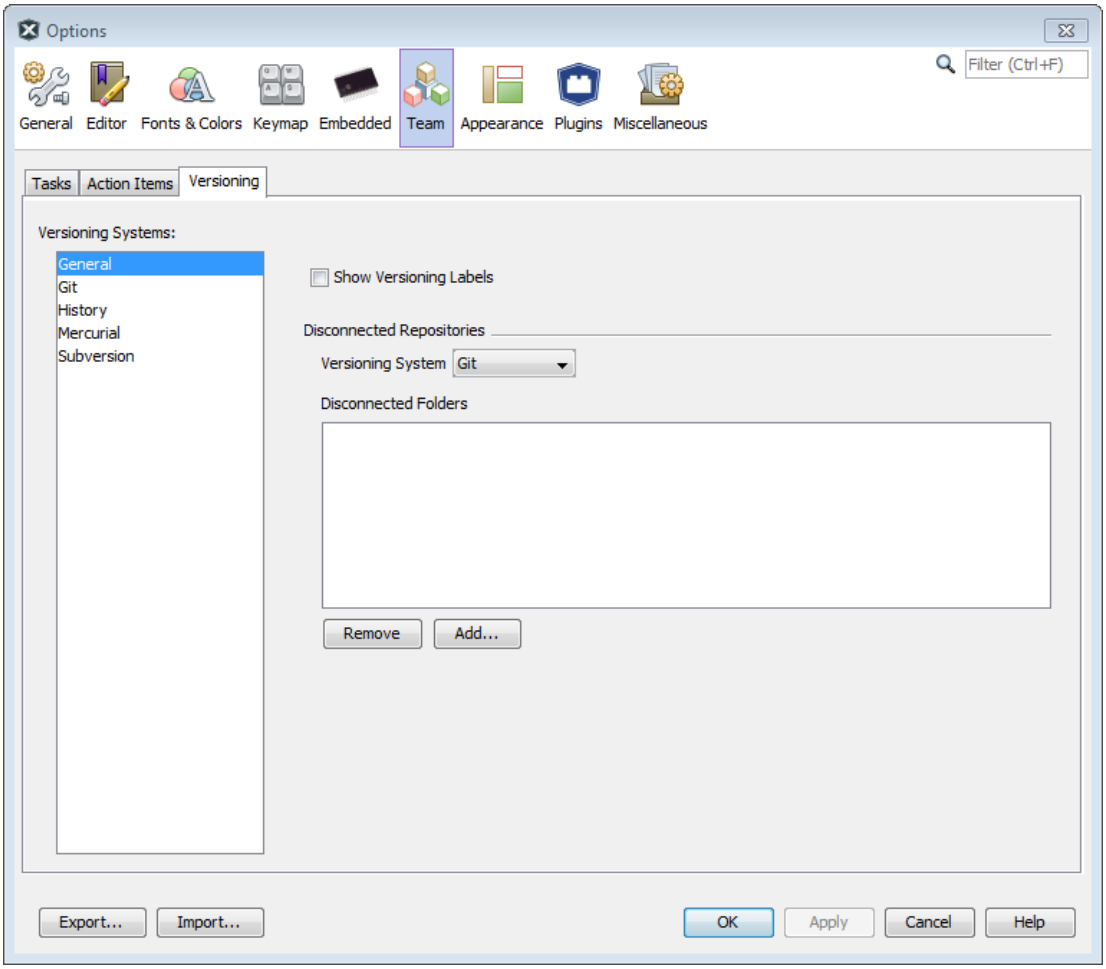
1. 尝试将所有项目文件保留在项目中。项目外的文件也将需要包含在资源库中，以便项目可以找到它们。
2. 在 MPLAB X IDE 中，仅提交项目中的文件。这一点至关重要，因为包含不需要的文件可能会阻止项目在其他机器上进行编译。请参见“保存项目文件”。
3. 使用 Merge Conflicts Resolver（合并冲突解决程序）窗口解决签入时的所有冲突。请参见“解决版本控制文件中的冲突”。

4. 根据“编译包含只读文件的项目”所述检查只读文件。

要设置源代码/版本控制：

1. Team（团队）菜单——从子菜单中选择一个版本控制程序并设置该版本控制程序（见下图）。
2. *Tools>Options*（对于 macOS 为 *MPLAB X IDE>Preferences*），Team，Versioning（版本控制）——设置版本控制选项。
3. *Window>Versioning*（窗口>版本控制）——打开版本控制窗口。

图 5-37. 版本控制选项



使用 Git

可通过以下链接获取关于 Git 的更多信息：

- Git 网站： http://git-scm.com
- 维基百科：Git： https://en.wikipedia.org/wiki/Git
- Git 教程： https://git-scm.com/docs/gittutorial
- 在 NetBeans IDE 中使用 Git 支持： https://netbeans.org/kb/docs/ide/git.html

使用 Subversion

可通过以下链接获取关于 Subversion 的更多信息：

- Apache™ Subversion®网站： http://subversion.apache.org/
- 维基百科：Apache™ Subversion®： http://en.wikipedia.org/wiki/Apache_Subversion
- Subversion®的版本控制： http://svnbook.red-bean.com/
- 在 NetBeans IDE 中使用 Subversion 支持： https://netbeans.org/kb/docs/ide/subversion.html

使用 Mercurial

可通过以下链接获取关于 Mercurial 的更多信息：

- Mercurial 网站： http://mercurial.selenic.com/
- 维基百科：Mercurial： http://en.wikipedia.org/wiki/Mercurial
- Hginit：Mercurial 教程： http://hginit.com/
- 在 NetBeans IDE 中使用 Mercurial 支持： https://netbeans.org/kb/docs/ide/mercurial.html

使用 CVS

要在 MPALB X IDE 中使用 CVS，必须安装 CVS 插件。请参见 5.26 添加插件工具。

- CVS——并发版本系统： http://www.nongnu.org/cvs/
- 维基百科：并发版本系统： http://en.wikipedia.org/wiki/Concurrent_Versions_System
- Source Forge：什么是 CVS？： http://sourceforge.net/apps/trac/sourceforge/wiki/CVS
- 在 NetBeans IDE 中使用 CVS 支持： https://netbeans.org/kb/docs/ide/cvs.html

使用其他版本控制系统

其他版本控制系统可作为插件添加。请参见 5.26 添加插件工具。

5.23.1 保存项目文件

有些项目文件需要保存到资源库中。

下表列出了需要或不需要提交到版本控制资源库的项目文件。

表 5-10. 保存到资源库中的项目文件

目录或文件	提交？
项目目录	
`Makefile`	✓
源文件	✓
build 目录	✗
dist 目录	✗
nbproject 目录	
`configurations.xml`	✓
`project.properties`	✓
`project.xml`	✓
`Makefile-*`	✗
`Package-*`	✗
私有目录	✗

.....（续）

目录或文件	提交？
绿色对勾：生成项目映像所必需的 红 X：这些目录/文件会被重新生成，因此不需要保存	

关于项目结构的更多信息，请参见“Files 窗口视图”。

5.23.2 解决版本控制文件中的冲突

您尝试签入的文件有可能已在版本系统资源库中被他人修改，因此难免会发生冲突。

为了理解如何解决这些问题，请参见下面的示例。

示例：`configurations.xml` 中的冲突

要解决文件 `configurations.xml` 中的冲突：

1. 将在以下位置提示冲突：
   1.1. 警告对话框
   1.2. 输出窗口
2. 单击 Projects 窗口中的 **Files** 选项卡。展开 `nbproject` 文件夹。查找名为 `configurations.xml` 的文件。该文件应以红色字体显示，表示发生了冲突。
3. 右键单击 configuration.xml，然后选择 *RCS>Resolve Conflicts*（RCS>解决冲突），其中 *RCS* 为版本控制系统。此时将打开 Merge Conflicts Resolver 窗口。
4. 在 Merge Conflicts Resolver 窗口中可修复冲突。
5. 修复错误后，关闭项目。如果右键单击该项目的上下文菜单不起作用，可使用 *File>Close Project*（文件>关闭项目）关闭该项目。
6. 重新打开项目。冲突应已得到解决。

5.23.3 编译包含只读文件的项目

如果要编译的项目中的文件已被签入到资源库中并且某些文件可能为只读状态：

1. 将最小数量的文件提交给源控制。5.23.1 保存项目文件中对此进行了说明。
2. 使文件 `nbproject/configurations.xml` 和 `nbproject/project.xml` 变为可写状态。

5.24 在代码开发和错误跟踪方面进行协作

使用 MPLAB X IDE 内支持的团队服务器与您所在的小组进行代码开发协作。使用“Team”菜单登录到您的帐户、创建或打开项目、共享项目、获取资源、发送聊天消息或显示联系人列表。使用错误跟踪系统在跟踪问题方面进行协作

支持的菜单项包括：

- *Window>Services*（窗口>服务）——Services 窗口是运行时资源的主要入口点。其中显示重要运行时资源（通过 IDE 注册的服务器、数据库和 Web 服务）的逻辑视图。在 *Help>Help Contents* 下搜索“Services Window”（Services 窗口）可获得更多详细信息。
- *Team>Find Task*（团队>查找任务）——在“Find Tasks”窗口中，选择一个项目任务，选择条件，然后单击 **Search**（搜索）。
- *Team>Report Task*（团队>报告任务）——在“Report a New Task”（报告新任务）窗口中，选择新项目任务，指定问题详细信息，然后单击 **Submit**（提交）。

关于团队项目和问题跟踪的更多信息，请参见 NetBeans 帮助主题：

https://netbeans.org/kb/docs/ide/team-servers.html

要了解关于这些工具的更多信息，请参见：

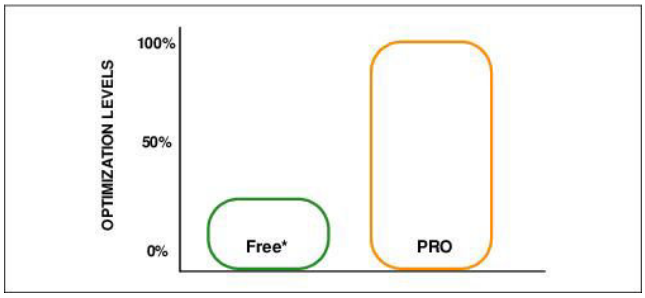
- Git—— http://git-scm.com
- Mercurial—— http://mercurial.selenic.com/

- Subversion—— http://subversion.apache.org/

5.25 比较 MPLAB XC 编译器免费版与专业版许可证

从 Microchip 网站下载 MPLAB XC C 编译器时，该编译器将以专业版许可证试用 60 天，之后恢复为免费版许可证。如果想继续使用大多数优化功能，可购买专业版许可证。

图 5-38. 每种许可证的优化级别

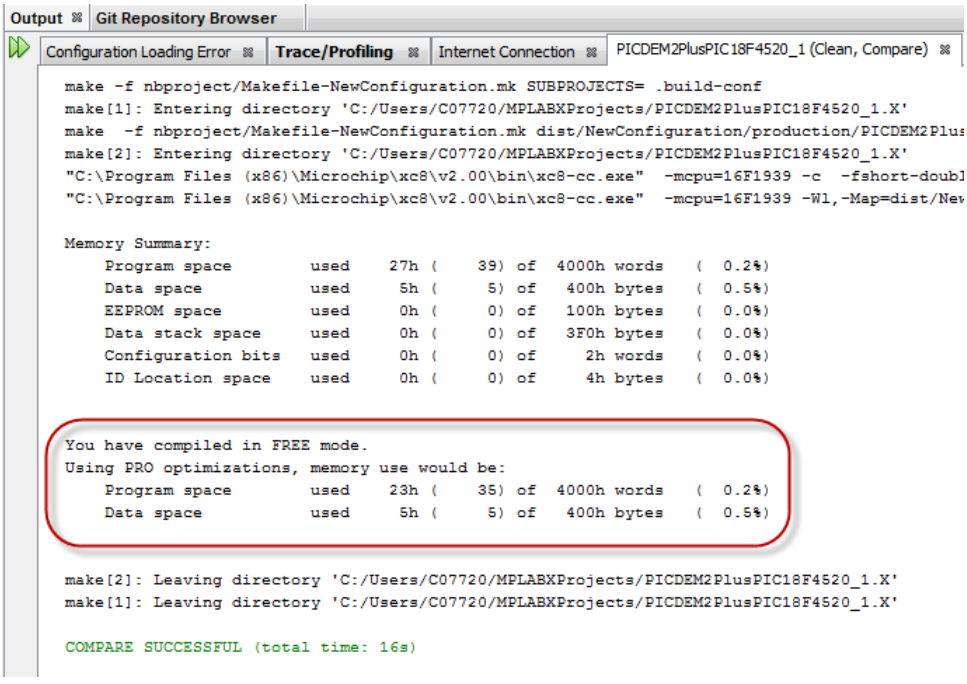


要确定是否需要购买专业版许可证，可以在 MPLAB X IDE 中对您的应用程序运行比较工具。在运行工具栏上，可以选择“Build with PRO comparison”（使用专业版比较进行编译）或“Clean and Build with PRO comparison”（清除并使用专业版比较进行编译）。下面给出了示例输出。

注： 如果您的代码过大而无法在免费版模式下编译，则将无法进行比较。但对于 MPLAB XC8 编译器，可以使用--MAXIPIC 选项，然后重试。

无论选择哪种优化，这些选项都可以正常编译项目，并且还可以创建无效的专业版编译，然后比较两个结果。您仍将获得原始编译作为输出。所有专业版编译部分均将被删除。

图 5-39. 专业版比较



5.26 添加插件工具

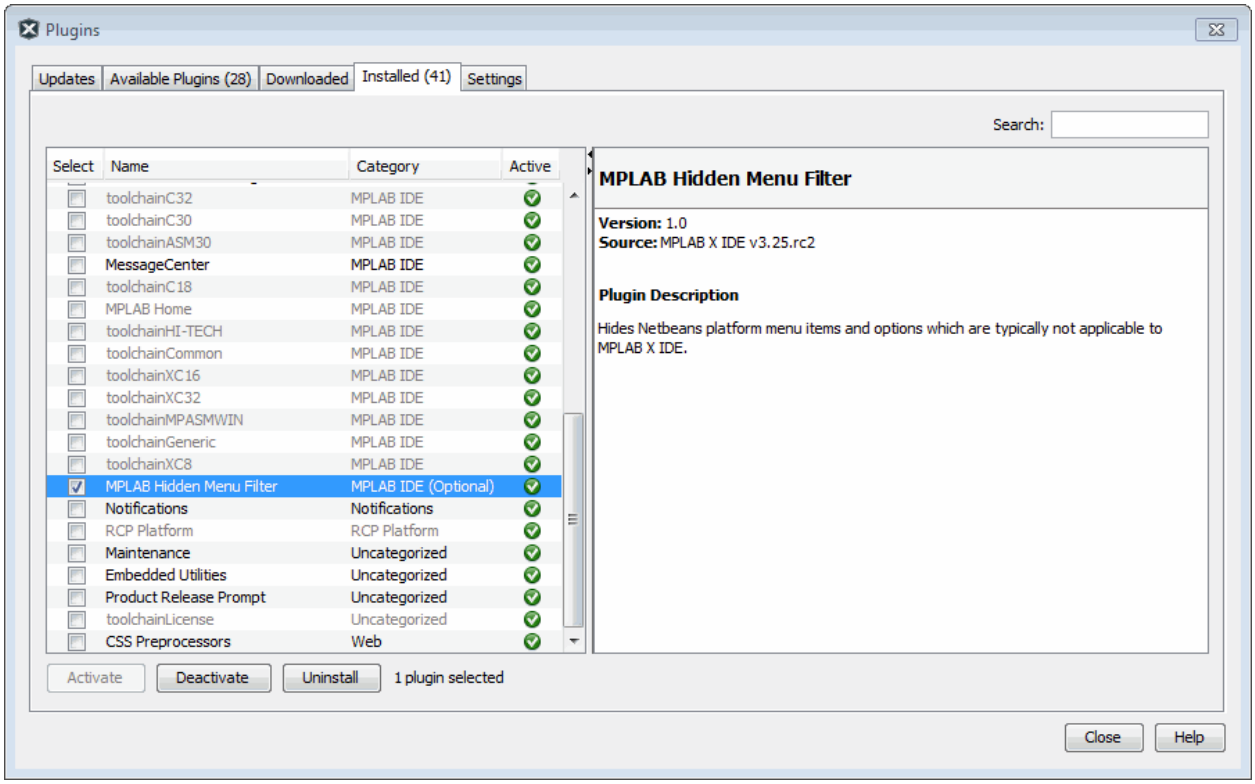
MPLAB X IDE 插件工具（如 DMCI）可从 IDE 中的插件管理器或 Embedded Code Source 网站获得。

查看已安装的插件

MPLAB X IDE 预装了许多插件。对于未呈灰显的插件，除非被归类为可选项才可以禁止或卸载，否则不建议这样做。

要查看已安装的插件，请选择 *Tools>Plugins*，然后单击 **Installed** 选项卡。

图 5-40. 已安装的 Microchip 插件工具



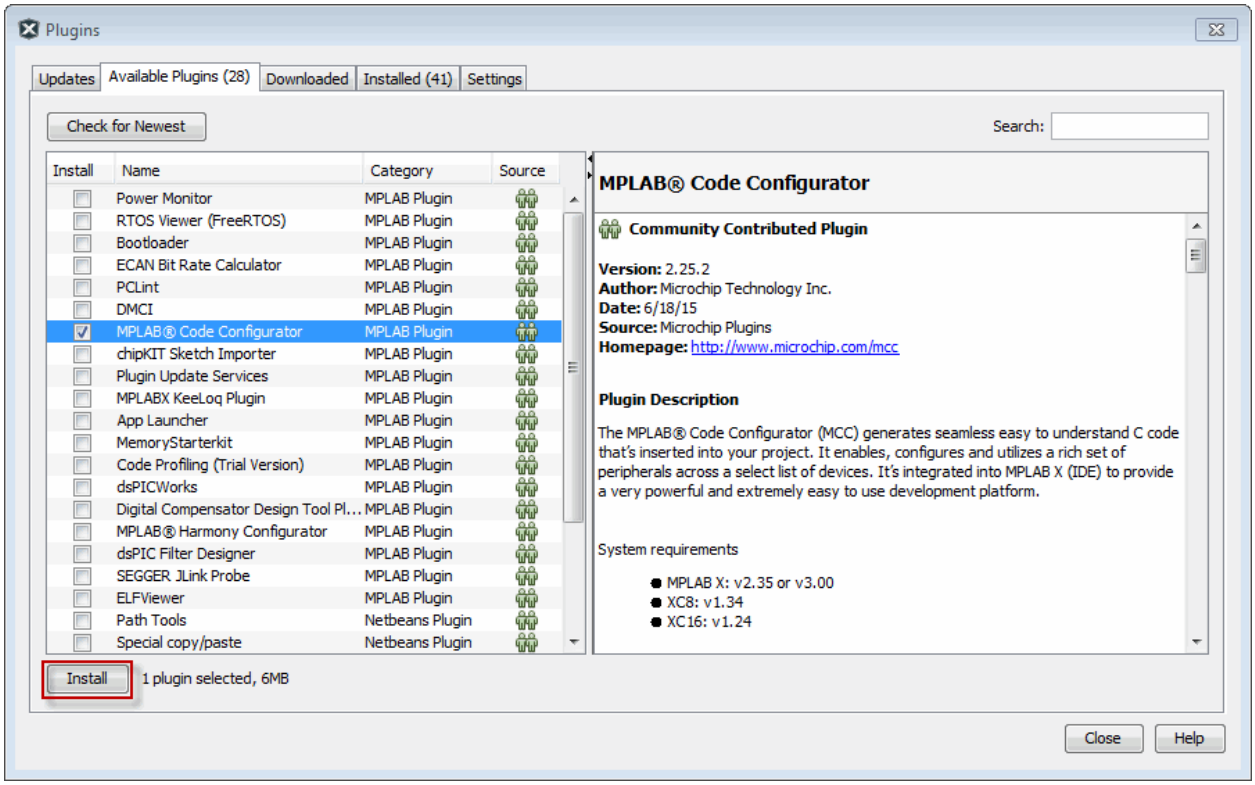
通过插件管理器添加插件

要查看和安装可用插件：

1. 在 MPLAB X IDE 中，选择 *Tools>Plugins*，然后单击 **Available Plugins** 选项卡。
2. 通过选中对应的复选框来选择插件，然后单击 Install（安装）（见下图）。
3. 按照屏幕说明下载并安装插件。
   注： 一些插件的功能可能依赖于其他插件中的模块。在这种情况下，插件管理器向您发出警告。
4. 完成安装后，该插件将列在 Installed 选项卡下，在该选项卡中可将其禁止、重新激活或卸载。
5. 在 Tools>Embedded 下查找您的工具。如果未看到它，则可能需要关闭并重新打开 MPLAB X IDE。

单击 **Help** 按钮可阅读关于安装插件的更多信息。

图 5-41. 可用的 Microchip 插件工具

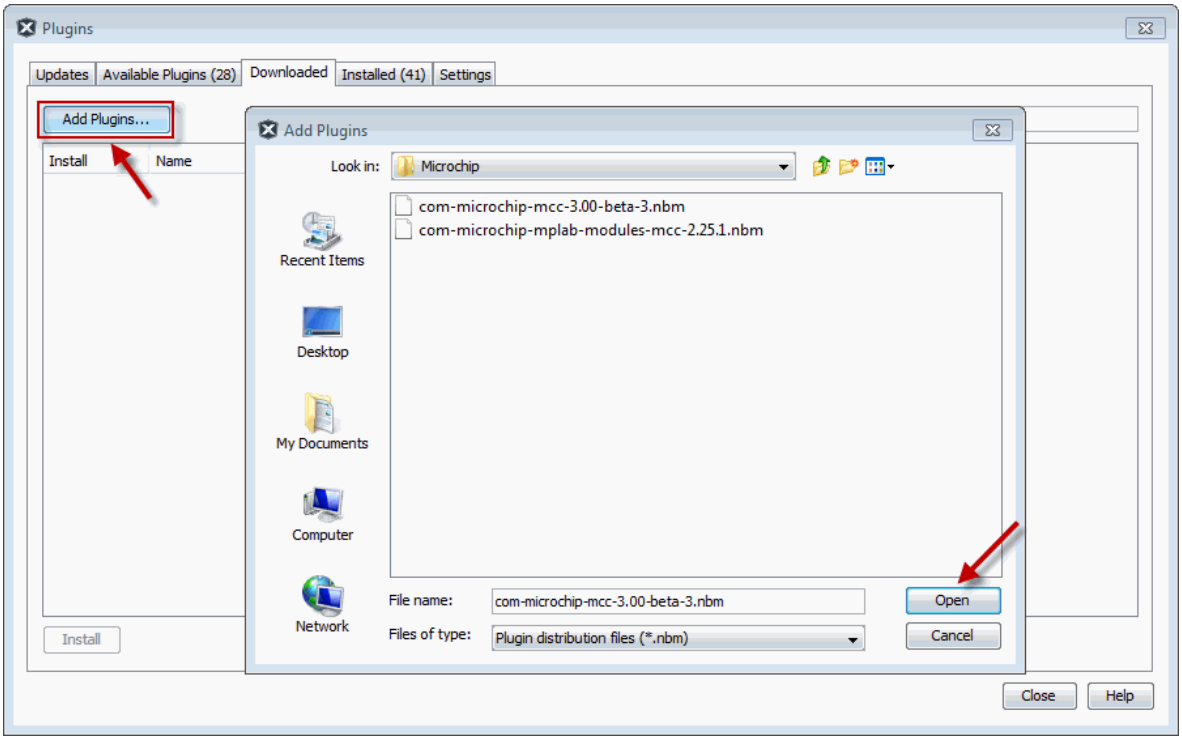


通过 Embedded Code Source 添加插件

要查看、下载和安装可用插件：

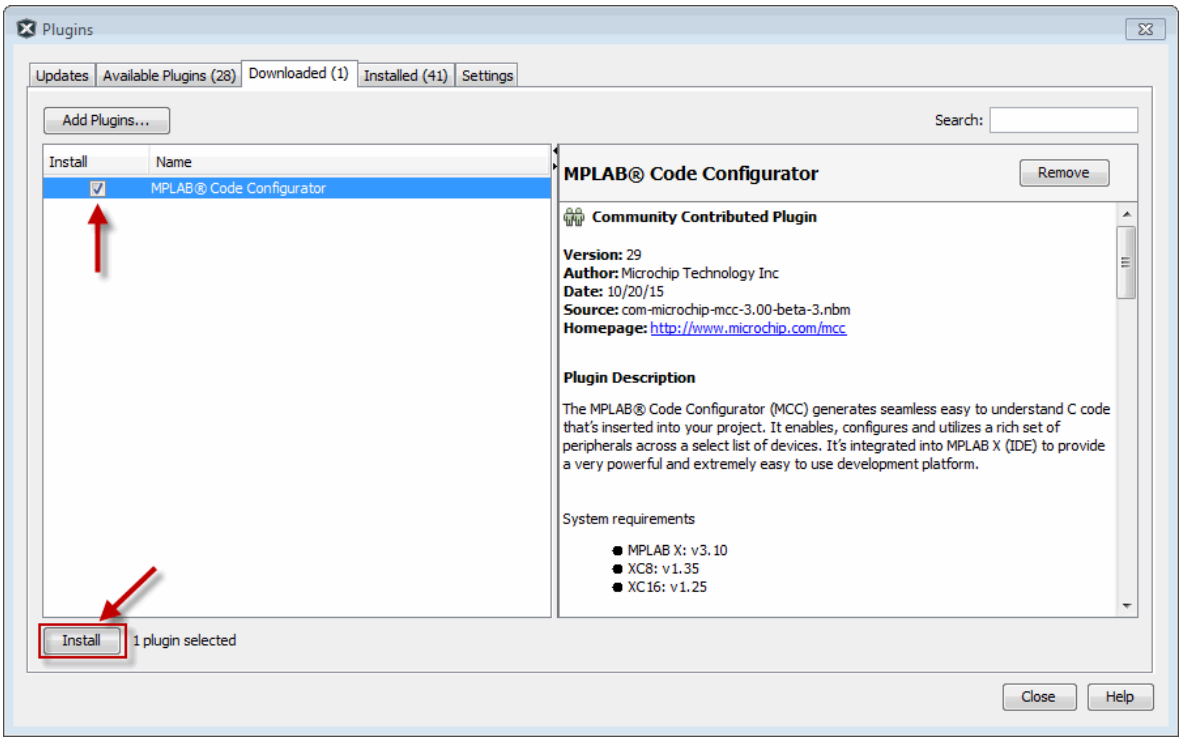
1. 访问 Embedded Code Source 网站：
   http://www.embeddedcodesource.com/codedeveloper/microchip-technology
2. 选择一个插件，然后按照网站上的说明将其下载到计算机上的文件夹中。
3. 从下载的 ZIP 文件中提取 `.nbm` 文件。
4. 在 MPLAB X IDE 中，选择 *Tools>Plugins*，然后单击 **Downloaded**（已下载）选项卡。
5. 单击 **Add Plugins**（添加插件）。查找、选择并打开 `.nbm` 文件。

图 5-42. 已下载的 Microchip 插件工具



6. 单击 Install 安装插件。确保已选中插件名称旁的复选框。

图 5-43. 安装已下载的 Microchip 插件工具



7. 按照屏幕说明下载并安装插件。
   注： 一些插件可能依赖于其他插件中的模块，以便实现其功能。在这种情况下，插件管理器向您发出警告。
8. 完成安装后，该插件将列在 Installed 选项卡下，在该选项卡中可将其禁止、重新激活或卸载。

9. 在 *Tools>Embedded* 下查找您的工具。如果未看到它，则可能需要关闭并重新打开 MPLAB X IDE。

升级插件

要确定您的插件是否有更新：

1. 在 MPLAB X IDE 中，选择 *Tools>Plugins*，然后单击 Updates（更新）选项卡。
2. 单击 **Check for Updates**（检查更新）。
3. 如果出现更新，请确保已选中插件名称旁的复选框，然后单击 **Update**（更新）。
4. 按照屏幕说明更新插件。

安装 MPLAB X IDE 的新版本将不会更新已安装的插件。插件是针对某个版本的界面进行测试的。不是所有插件都可以在不同版本之间移植，所以不能在新版本中继续使用它们。对于 NetBeans 也是如此。因此，在安装新版本的 MPLAB X IDE 时，可能需要重新安装插件。

配置更新中心

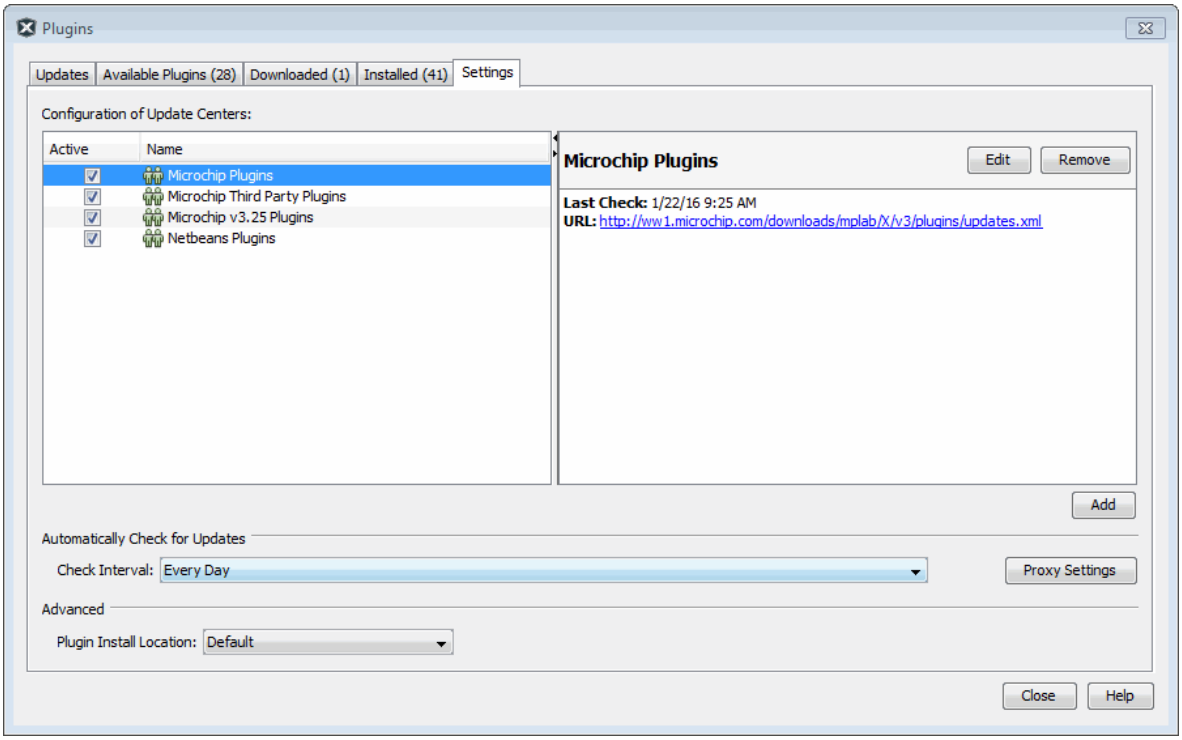
要在 Plugins 窗口中查看可用插件，必须配置一个或多个更新中心。MPLAB X IDE 安装时已配置了两个 Microchip 更新中心：

- lhttp://ww1.microchip.com/downloads/mplab/X/plugins/updates.xml
- http://ww1.microchip.com/downloads/mplab/X/thirdpartyplugins/updates.xml

要在插件管理器中配置另一个更新中心：

1. 选择 *Tools>Plugins* 并单击 Settings（设置）选项卡。
2. 单击 Add 按钮，打开 Update Center Customizer（更新中心定制器）对话框。
3. 输入更新中心的名称。
4. 输入更新中心的 URL。
5. 单击 **OK**。

图 5-44. 配置 Microchip 更新中心



配置已安装的插件

在 MPLAB X IDE 中安装插件后，可能需要对其进行配置。插件配置选项位于 *Tools>Options* 的 Plugins 下。请参见“Tools>Options>Plugins 窗口”。

插件代码位置

插件代码与 MPLAB X IDE 用户配置数据存储在一起。请参见 8.7 查看用户配置数据。

6. 高级任务和概念

以下主题将讨论如何执行更高级的任务，并详细介绍一些有用的概念。

表 6-1. 了解高级任务和概念

1 加快速度	1. 在 IDE 运行速度过慢时为 MPLAB X IDE 加速。 2. 使用并行 make 在 MPLAB X IDE 中加快编译速度。
2 处理项目	1. 在开发复杂应用程序时，处理多个项目。 2. 处理多个配置，允许为同一项目代码搭配选用不同的项目属性。 3. 创建双核项目以与 dsPIC33CH 双核器件配合使用。 4. 创建用户 Makefile 项目以在 MPLAB X IDE 中使用外部 Makefile。 5. 将链接资源用于源文件文件夹以更改项目源码库或库版本（MPLAB Harmony）。 6. 在 MPLAB X IDE 项目中使用第三方工具。 7. 使用代码覆盖率测试已执行代码的百分比。 8. 记录数据以捕捉执行和调试问题。 9. 打包 MPLAB X IDE 项目以压缩项目中的文件。 10. 与目标板的硬件工具连接有时可以确定调试功能。请参见硬件工具连接和调试。
3 概念	1. 什么是校验和。 2. 不同种类的配置。

6.1 加快 MPLAB X IDE 速度

如果 MPLAB X IDE 的运行速度太慢，可以考虑下述建议。

增加计算机的堆大小

您可以在文件 `mplab_ide.conf` 中修改为 MPLAB X IDE 分配的存储空间大小。建议您在开始编辑该文件之前先进行备份。如果更改了该文件的内容，所做的更改将在下一次运行 MPLAB X IDE 时生效。

- ***Windows OS 64 位***——`C:\Program Files (x86)\Microchip\MPLABX\vx.xx\mplab_platform\etc`
- ***Windows OS 32 位***——`C:\Program Files\Microchip\MPLABX\vx.xx\mplab_platform\etc`
- ***Linux OS***——`/opt/microchip/mplabx/vx.xx/mplab_platform/etc`
- ***macOS***——`/Applications/microchip/mplabx/vx.xx/Contents/Resources/mplab_platform/etc`

其中，`vx.xx` 表示 MPLAB X IDE 版本。

以下行包含了默认值：

```
default_options="-J-Dnb.FileChooser.useShellFolders=false -J-Dcrownking.stream.verbosity=very-quiet -J-Xms256m -J-Xmx512m -J-XX:PermSize=128m -J-XX:MaxPermSize=384m -J-XX:+UseConcMarkSweepGC -J-XX:+CMSClassUnloadingEnabled"
```

粗体区域为：

`-Xms256m` 指示 JVM 初始时至少为堆分配 256 MB。

`-Xmx512m` 指示 JVM 最多为堆分配 512 MB，而不超出。

`-XX:PermSize=128m` 指示 JVM 为跟踪不进入堆中的其他数据所需的空间分配 128 MB。

`-XX:MaxPermSize=384m` 指示 JVM 为不进入堆中的其他数据最多分配 384 MB。

除非收到以下错误，否则应该不需要修改 PermSize 或 MaxPermSize： `java.lang.OutOfMemoryError: PermGen space`。

一般情况下，最重要的方面为`-Xmx512m`；它会限制 MPLAB X IDE 将使用的堆的最大空间大小。看起来好像使用较大的堆可能会有所帮助。但增大用于 MPLAB X IDE 的存储空间意味着用于其他应用程序和系统功能的存储空间会减少。

您可以通过使能存储器监视器来监视 IDE 使用了多少存储空间。右键单击工具栏区域中的空白区域并选择存储器，或选择 *View>Toolbars>Memory*（视图>工具栏>存储器）。

除非您在 `mplab_ide.conf` 中更改值，否则上限值不会超出 512 MB。建议以 128 MB 为增量更改`-Xmx512m` 设置。如果具有大量的存储空间，则增大量可以超出 128 MB，但是需要确保为系统的其余部分留下足够的存储空间。

更改调试窗口的使用

如果使用调试工具导致 MPLAB X IDE 速度下降，尝试进行以下操作：

- 仅显示需要查看的窗口。当使用调试工具时，MPLAB X IDE 调试器会检查每个打开的窗口。
- 在单步调试过程中，减少对堆栈窗口的更新。在 *Tools>Options*（在 macOS 中为 *MPLAB X IDE>Preferences*）窗口，**Embedded** 按钮，**Generic Settings** 选项卡中，选中“Disable auto refresh for call stack view during debug sessions”（在调试会话期间禁止调用堆栈视图自动刷新）和“On mouse over structure and array expressions, evaluate integral members only”（鼠标悬停在结构体和数组表达式上时，仅计算整数成员的值）。

6.2 加快编译速度

根据您的计算机的配置，您可能可以使用并行 make（见 12.16.2 Project Options 选项卡）来加快项目编译速度。并不是所有语言工具都支持并行 make。

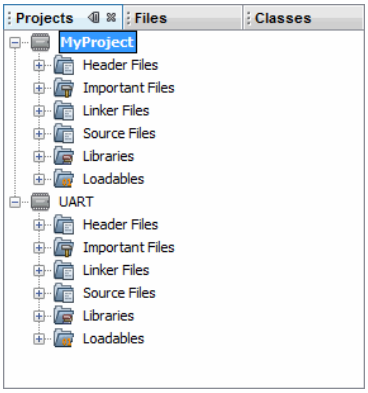
另一种选项是考虑您的操作系统（OS）。一些操作系统的文件访问速度更快。MPLAB X IDE 支持 Windows、Linux 和 Mac 操作系统。研究一下哪一种操作系统可能适合您。

6.3 处理多个项目

MPLAB X IDE 允许您处理多个项目。

可以在 MPLAB X IDE 中打开多个项目并在 Projects 窗口中查看。关于该窗口的更多信息，请参见 12.15 Projects 窗口。

图 6-1. Projects 窗口中的多个项目



打开多个项目

可以直接在 MPLAB X IDE 中创建（*File>New Project*）或打开（*File>Open Project*）多个项目。

也可以使用--open 伪指令通过快捷方式打开项目，例如：

```
"C:\Program Files (x86)\Microchip\MPLABX\v3.15\mplab_platform\bin\
mplab_ide.exe" --open "C:\MPLAB_X_Projects\MyProject1"
```

之后可以创建多个快捷方式，从多个不同位置打开多个项目。

定义项目类型

每个项目都可以设置为活动项目，或在开发的焦点项目，如下所述。但是，任何时候都应该只有一个主项目。

主项目

通过右键单击项目名称并选择“Set as Main Project”，可以选择一个项目作为主项目。然后，主项目名称将以粗体显示。在这种情况下，所有工具栏功能都将在一个主项目上执行。

活动项目

还可以在不设置主项目的情况下工作。没有主项目时，活动项目就是焦点项目。

通过在 Projects 窗口中单击项目，可以将项目设为活动项目。在这种情况下，IDE 将通过上下文确定您正在处理的项目，即，如果将编辑器焦点设置在某个文件中，则拥有该文件的项目将成为活动项目。活动项目接收所有操作，例如，单击 Debug Project 按钮时。

开发项目代码

要为每个项目开发和调试代码：

1. 开始调试项目，方法是在 Projects 窗口中单击项目，然后选择 *Debug>Debug Project*。
2. 如果正在同时运行多个调试会话，打开 Sessions（会话）窗口（*Window>Debugging>Sessions*（窗口>调试>会话）），您可以在当前正在运行的任何调试会话之间切换。

从一个调试会话切换到另一个时，Watches 窗口和变量以及存储器会切换为显示正在调试的当前选定项目。状态位也会跟随被调试项目而变化。Dashboard 将跟随最后选定的项目变化，无论它是被调试项目还是项目窗口中的项目。这是由设计决定的。

如果在 *Tools>Options*（对于 macOS 为 *MPLAB X IDE>Preferences*），**Embedded** 按钮，**Generic Settings** 选项卡下选中了“Maintain active connection to hardware tool”，则当您切换会话/项目时，**将**执行重新连接。这是由设计决定的。

对多个项目分组

处理多个项目的另一种方式是进行分组。选择 *File>Project Group*（文件>项目组），并选择或创建一个项目组（使用 Create New Group（新建组）对话框）。

6.4 处理多个配置

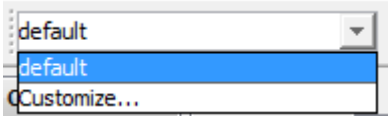
MPLAB X IDE 允许为同一项目使用多个编译配置。对于可以在多个平台上进行编译的代码（例如 Microchip 应用程序库演示项目），这会很有用。

创建新项目时，会创建“default”配置。要创建其他配置，请按照下述步骤进行操作：

6.4.1 Manage Configurations 对话框

要创建自己的配置，请先执行以下任一操作：

- 调出工具栏上的下拉菜单，然后选择“Customize”。Project Properties 窗口将打开。

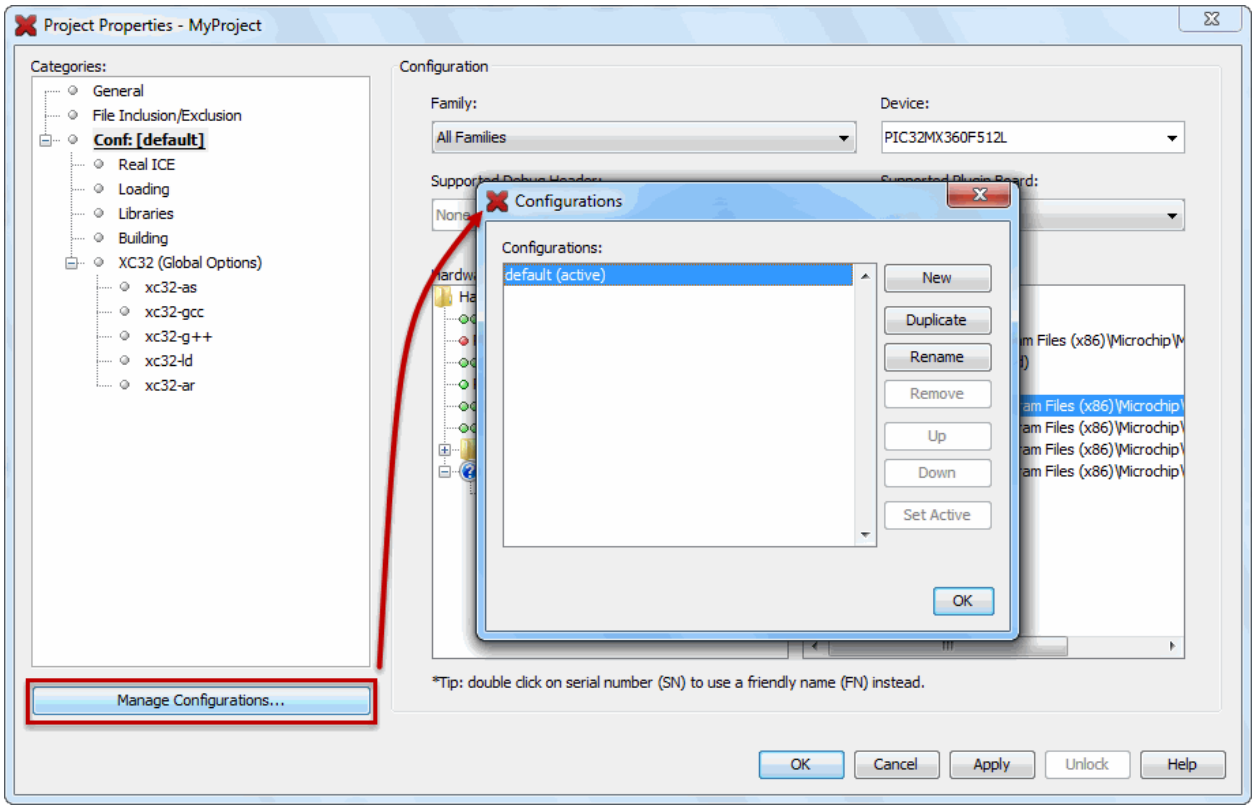


- 右键单击项目名称并选择“Properties”，以打开 Project Properties 窗口。

在 Project Properties 窗口中，单击 **Manage Configurations** 以打开 Configurations 对话框。可以重命名现有配置，也可以添加或从现有配置复制新的配置。名称中的空格都将替换为下划线。

为项目创建多个配置时，可以从 **Manage Configurations** 或下拉菜单中选择活动配置。

图 6-2. 项目属性——Configurations 对话框

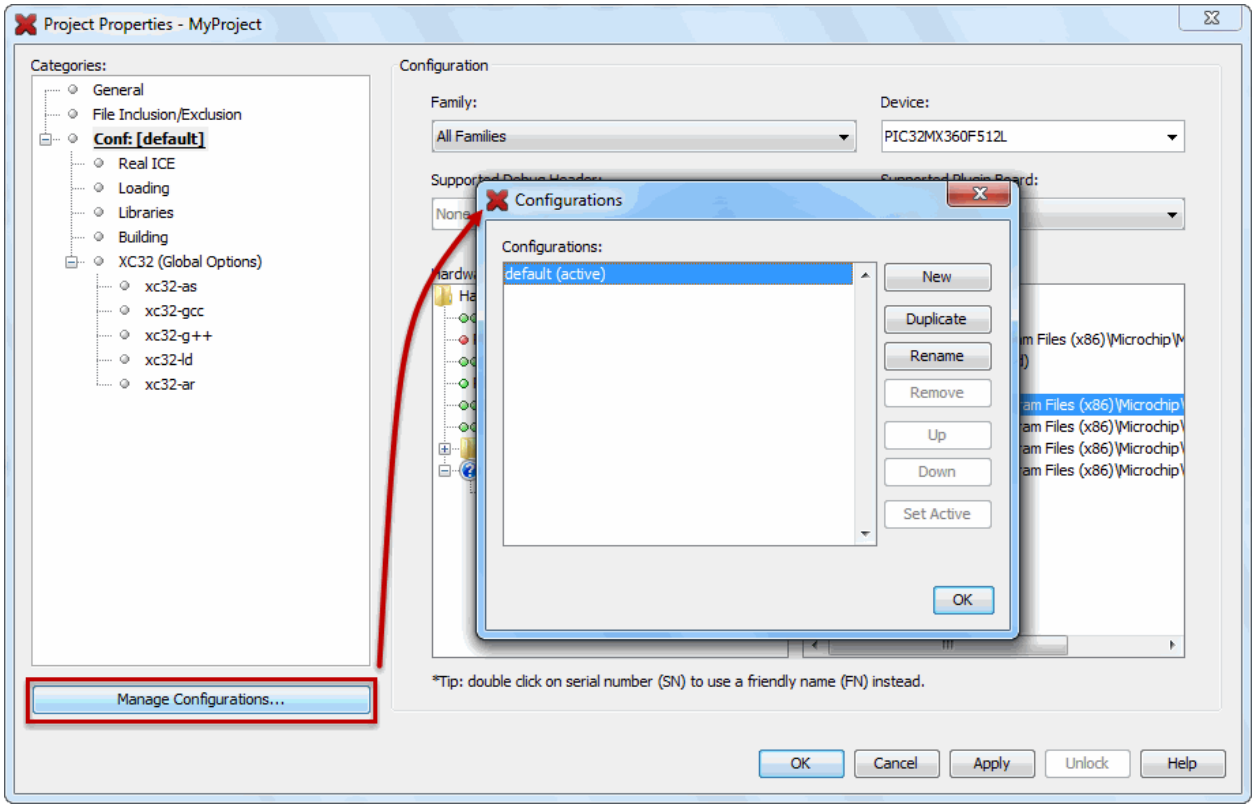


6.4.2 添加新配置

要向项目中添加新配置：

1. 单击 Project Properties 窗口中的 **Manage Configurations**。
2. 在 Configurations 对话框中，选择项目配置，然后单击 **New**。
3. 在 New Configuration Name（新配置名称）对话框中输入名称，例如“My Test”（见下图）。单击 **OK** 关闭该对话框。在“Configurations”下，该名称将更改为“My_Test”，因为不允许使用空格。
4. 单击 **OK** 返回到 Project Properties 窗口。现在应该可以看到调试配置（Conf: My_Test）。

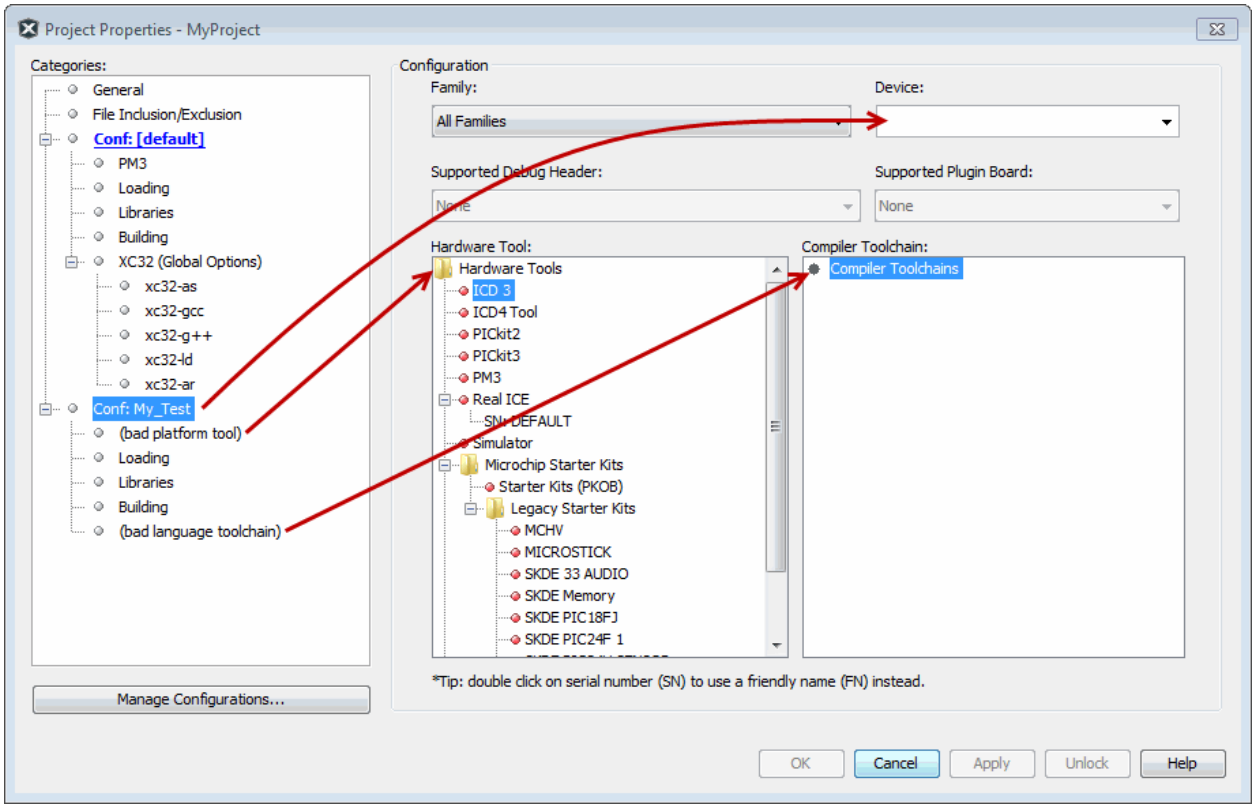
图 6-3. 创建新配置



添加新配置后，必须在 Project Properties 窗口中分配若干项（见下图）。

1. Device——必须先选择此项以查看硬件工具和编译器支持。
2. Hardware Tool（硬件工具）
3. Compiler Toolchain

图 6-4. 新配置



6.4.3 添加副本配置

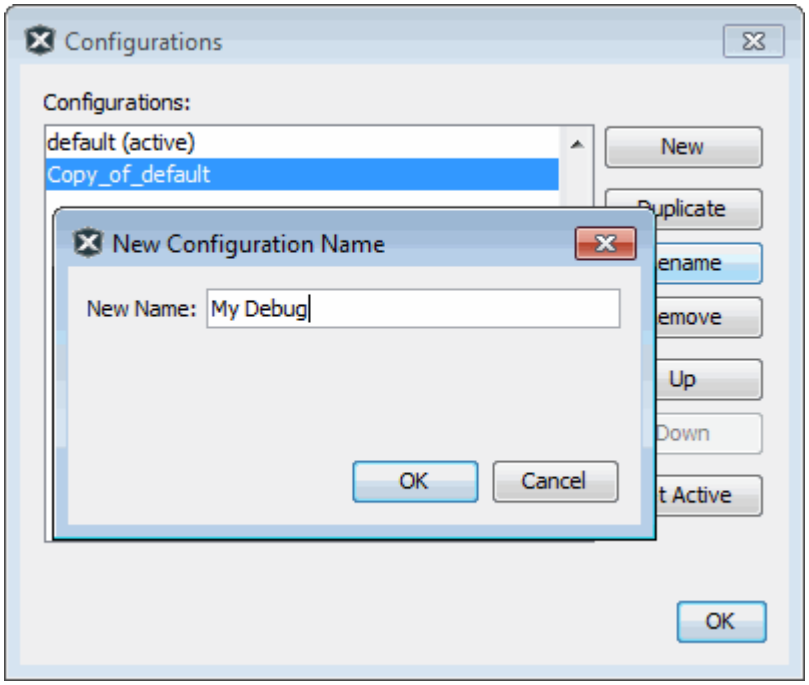
您可以添加一个现有配置的副本来作为新配置，然后对其进行编辑。

添加副本配置的其中一个原因是可以创建自己的调试配置。虽然 MPLAB X IDE 提供了用于 Microchip 工具的调试宏（见 4.13 调试代码中的“生成的调试宏”），但您可能希望使用自己的调试宏或希望使用第三方工具设置相同的调试功能。

要设置**调试配置**：

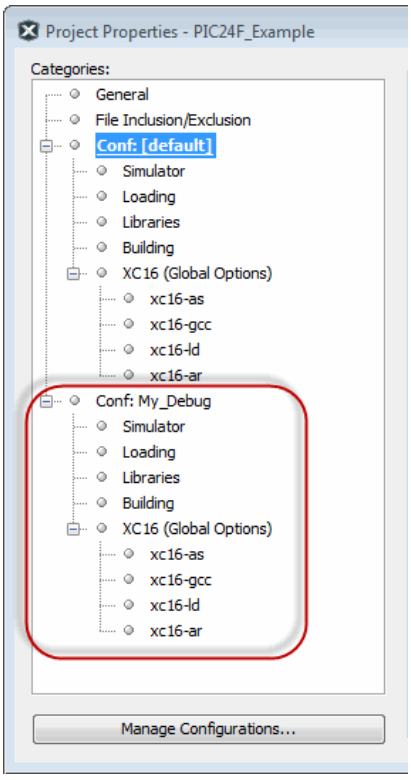
1. 单击 Project Properties 窗口中的 **Manage Configurations**。
2. 在 Configurations 对话框中，选择项目配置，然后单击 **Duplicate**。
3. 单击 **Rename**（重命名）并在 New Configuration Name 对话框中输入名称，例如“My Debug”（见下图）。单击 **OK** 关闭该对话框。在“Configurations”下，该名称将更改为“My_Debug”，因为不允许使用空格。

图 6-5. 创建调试配置



4. 单击 **OK** 返回到 Project Properties 窗口。现在应该可以看到调试配置（Conf: My_Debug）（见下图）。

图 6-6. 调试配置



要在代码中使用宏来切换到其他配置（例如调试），请参见 6.4.6 配置宏。

6.4.4 管理配置中的文件

可通过将项目文件单独或成组分配给已创建的配置来管理项目文件。

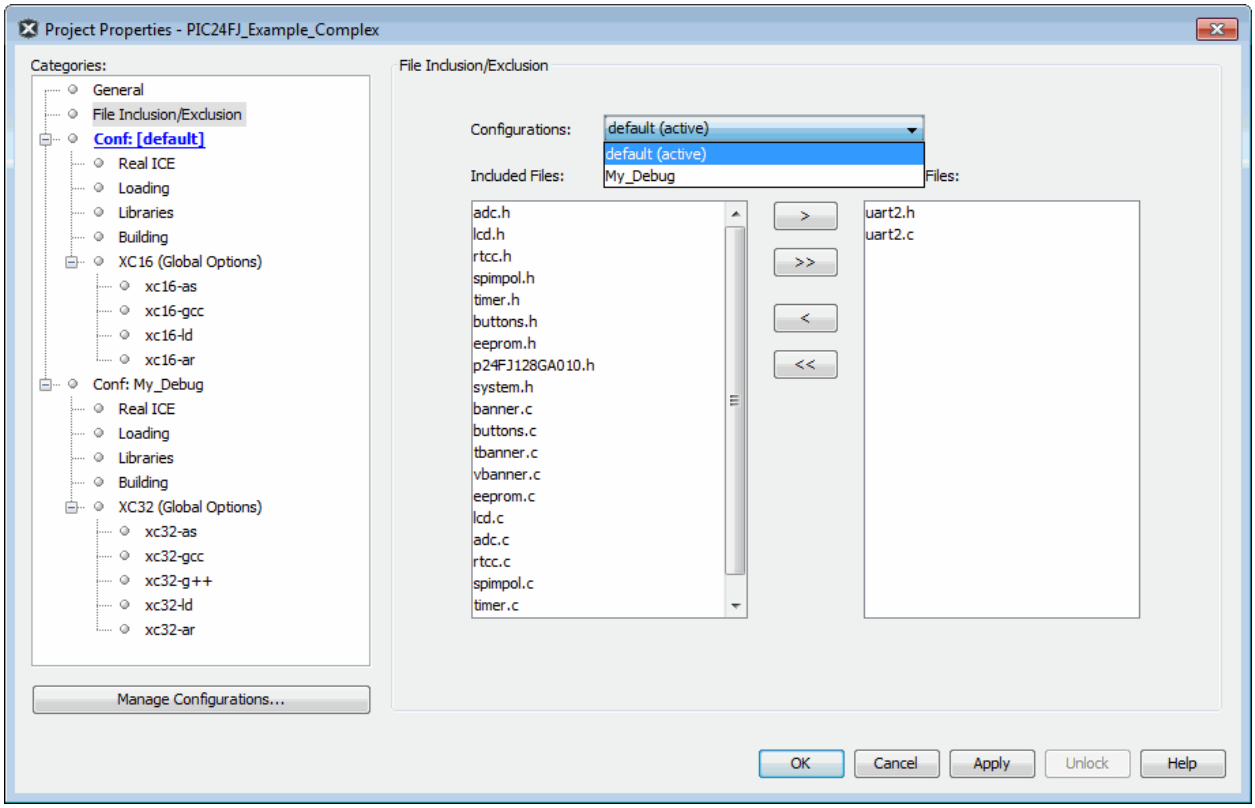
Projects 窗口

在工具栏的“Set Project Configuration”下拉框中选择所需配置。然后选择一个或多个文件（按住 Shift 并单击或按住 Ctrl 并单击），然后右键单击其中一个文件以弹出上下文菜单。选择“Exclude file(s) from current configuration”（从当前配置中排除文件）。要再次包含文件，重复相应步骤直到出现上下文菜单“Include file(s) from current configuration”（包含当前配置中的文件）。

Project Properties 窗口

要将所需配置设为应用于许多文件，请在 Projects 窗口中右键单击项目名称，以打开 Project Properties 窗口。选择“File Inclusion/Exclusion”（文件包含/排除）类别。从 Configurations 下拉框中选择一个配置，然后指定要为该配置包含或排除的文件。

图 6-7. 项目属性配置



6.4.5 配置名称

配置名称中不允许使用某些特殊字符，目的在于能够将源代码中的配置名称用作定义宏（“配置宏”）。

连字符（-）和与号（&）等特殊字符将自动替换为下划线（_）。

6.4.6 配置宏

添加新配置或副本配置时，既可以创建自己的配置相关宏，也可以将自动生成的宏用于代码中的#ifdef 或其他语句。这些宏属于 MPLAB XC C 编译器预处理器宏。

用户定义的宏

要定义自己的预处理器宏：

1. 在 Project Properties 窗口中，单击项目配置以将其激活。在该配置下，单击工具链中的编译器。
2. 在“Preprocessing and messages”（预处理和消息）选项类别下，找到用于定义宏的选项，然后单击关联的文本框。
3. 在弹出对话框中，输入宏名称，然后单击 **OK**。现在，该宏与活动配置相关联。

可以将宏命名为与配置相同的名称（例如，6.4.3 添加副本配置中的“My_Debug”），也可以使用其他任何名称（例如“DebugConfig”）。

宏名称（和字符串，如下所述）区分大小写。

您现在可以在条件文本中使用预处理器宏：

```
// If My_Debug configuration is active
#ifdef DebugConfig
fprintf(stderr,"This is a debugging message\n");
#endif
```

此外，还可为宏分配与配置名称匹配的字符串值，以便在常规代码中使用：

```
#ifdef DebugConfig
#define CFG_NAME “My_Debug”;
// other defines
#endif
#ifdef DefaultConfig
#define CFG_NAME “My_Default”;
// other defines
#endif
:
int main(int argc, char** argv)
{
printf(CFG_NAME);
return{EXIT_SUCCESS};
}
```

CFG_NAME 宏将始终保持活动配置的名称。

自动生成的宏

创建新配置或副本配置（例如 My_Test）时，还会创建一个相关宏，例如 XPRJ_My_Test。该宏将具有与配置名称（“My_Test”）匹配的字符串值。

可以将该宏用于#ifdef 语句（如上一节所述），或者也可以在编译时使用宏的值，如下所示：

```
int main(int argc, char** argv)
{
printf(CFG_NAME);
return{EXIT_SUCCESS};
}
```

CFG_NAME 宏将自动保持活动配置的名称。

6.5 创建双核项目

双核器件（例如 dsPIC33CH DSC）搭载两个独立运行的内核（主内核与从内核），这两个内核在应用程序开发期间可以单独进行编程和调试。两个处理器（主内核和从内核）子系统都有自己的中断控制器、时钟发生器、ICD、端口逻辑、I/O 多路开关和 PPS。该系列器件相当于在单个芯片上拥有两个完整的 dsPIC® DSC。

双核器件具有多种工作模式，分别对应于不同的项目创建和使用方式：

仅主内核编程和调试

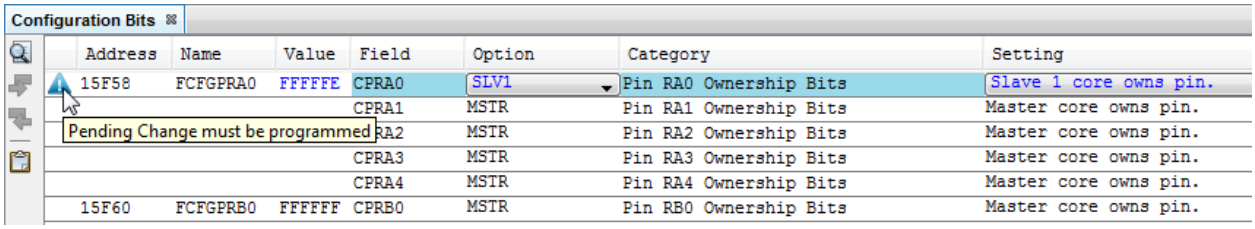
默认情况下，双核器件处于“仅主内核”模式；主从接口（Master Slave Interface，MSI）的 MSI1 主内核控制寄存器（MSI1CON）中的从内核使能位（SLVEN）置为 0。

使用双核器件该工作模式的项目与使用任何 Microchip 器件的项目并无不同。有关详细信息，请参见“创建新项目”。

仅从内核编程

为使从内核项目独立工作，需要为从内核分配端口。因此，需要一个主内核项目（例如 MasterStub），以便使用配置位设置（例如将哪些端口分配给从内核）对主内核进行编程。有关详细信息，请参见“在 Configuration Bits 窗口中设置配置位”。

图 6-8. 主内核配置位



对于仅从内核编程模式，需要两个项目：

1. 按照“创建新项目”所述创建一个 MasterStub 项目，并确保使能从内核（SLVEN =“1”）。如前文所述设置配置位。
2. 按照“创建新项目”所述创建一个 SlaveOnly 项目，但使用器件的“S1”版本，例如 dsPIC33CH128MP508S1。

必须先编程 MasterStub，然后再编程 SlaveOnly。

仅从内核调试

要调试上一节中的 SlaveOnly 项目，应在配置设置中使能后台调试（S1DEBUG）：

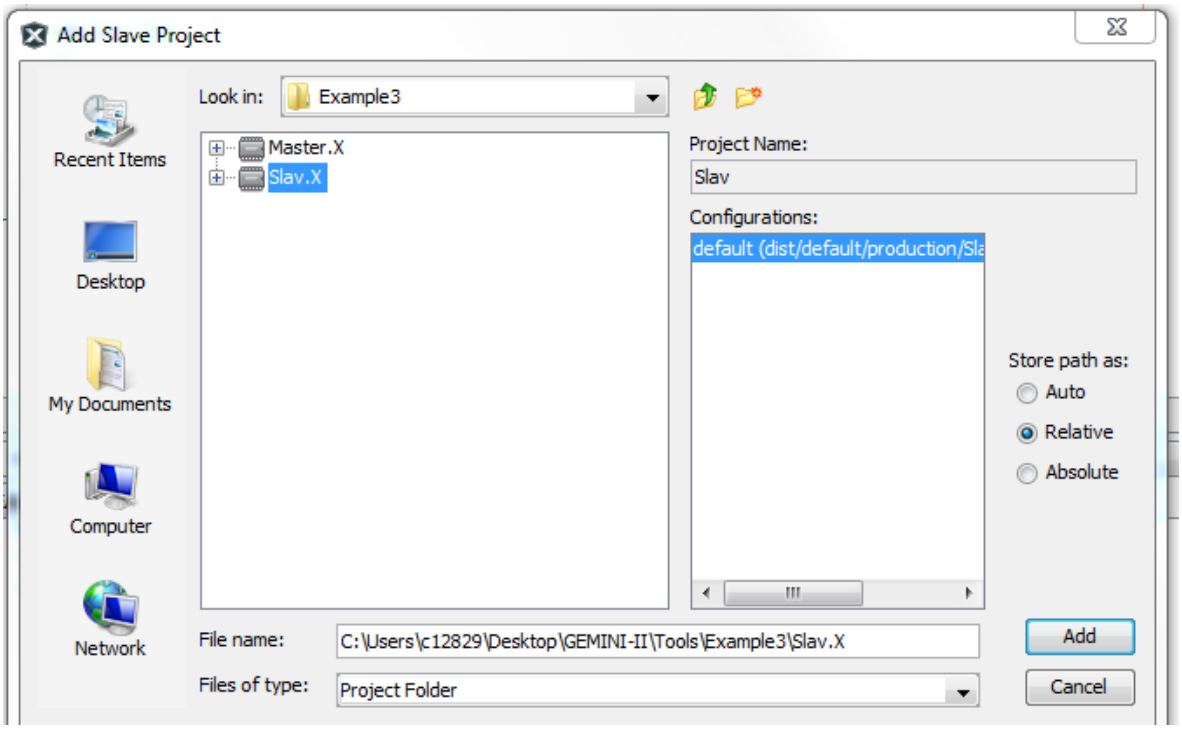
```
// FS1ICD
#pragma config S1ICS = PGD2 // ICD Communication Channel Select bits
                            // (Communicate on PGEC2 and PGED2)
#pragma config S1DEBUG = ON // Background Debug (Enabled)
 :
```

主内核与从内核同时编程和调试

在该模式下，主内核项目将链接到从内核项目。即，从内核代码位于主内核中，并且主内核必须将代码传送到从内核。

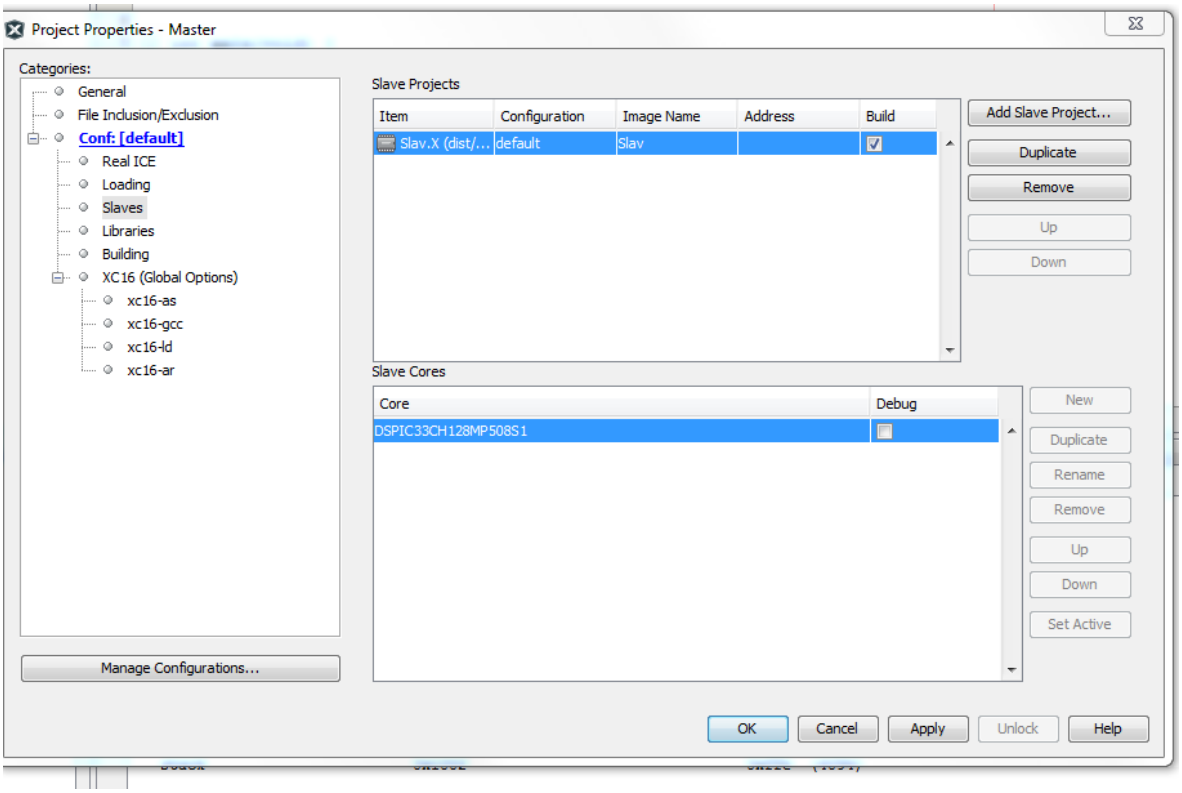
1. 按照“仅从内核编程”所述创建主内核项目和从内核项目。
2. 在 Projects 窗口中，主内核项目将具有一个名为“Slaves”的文件夹。右键单击该文件夹，然后单击“Add Slave Project”（添加从内核项目）。然后在该对话框中选择从内核项目，并单击 **Add**。

图 6-9. 添加从内核项目



3. 右键单击主内核项目，然后选择“Properties”以打开 Project Properties 窗口。单击“Slaves”（从内核）类别，并确保选中“Build”复选框。单击 **Apply**。

图 6-10. 包含从内核项目的主内核项目



4. 在主内核代码中添加：

```
#include "Slav.h"
//The function to transfer the code from slave to master and start the slave is below.
 _program_slave(1,0,Slav);
 _start_slave();
```

5. 通过编译和编程主内核，对主内核项目和从内核项目同时进行编译，并对主内核和从内核同时进行编程。

6.6 创建用户 Makefile 项目

创建一个使用外部 makefile 的项目。如果您具有一个在 MPLAB X IDE 之外编译的项目，但现在希望使用 MPLAB X IDE 进行调试，则这会很有用。

要创建 makefile 项目，请使用 New Project 向导。

6.6.1 新建项目——用户 Makefile

关于打开 New Project 向导的方法，请参见 4.1.2 启动 New Project 向导。

向导将启动，指导您完成新项目设置。

步骤 1.选择项目：选择“Microchip Embedded”类别，然后选择项目类型“User Makefile Project”。

步骤 2.选择器件：从“Device”下拉列表中选择项目中使用的器件。要缩小选择列表，请先选择 Family。

步骤 3.选择调试头：如果存在可用于选定器件的调试头，则会出现该步骤。要确定调试是否需要调试头，或器件是否具有片内调试电路，请参见以下文档之一。然后选择是否使用调试头。

- 处理器扩展包（PEP）和调试头规范
- 仿真扩展包（EEP）和仿真头用户指南

步骤 4.选择工具：从列表中选择将用于调试应用的开发工具。工具名称旁边将显示所选器件的工具支持级别。绿色表示完全支持，黄色表示 beta 支持，红色表示尚不支持。

步骤 5.选择接插板：如果存在可用于选定器件的接插板，则会出现该步骤。调出“Supported Plugin Board”（支持的接插板）下方的下拉列表可查看可用接插板。

步骤 6.选择项目名和文件夹：选择新项目的名称和位置。见表 1。

步骤 7.创建用户 Makefile 项目：输入数据以设置您的 makefile 项目（见表 2）。关于该窗口中常规数据输入的说明如下：

- 建议在路径中使用正斜杠（/）。不过您可以改用反斜杠（\），但需要进行转义（\\）。
- 建议使用不带空格的路径名和文件名。但如果具有空格，则当您在该窗口中输入路径和文件名时，请使用转义空格（空格之前加上反斜杠）。已知 GNU Make 会对于空格产生一些奇怪的问题，因为空格是其分隔符。

步骤 8.完成：新项目将在 Projects 窗口中打开。

在创建项目之后，在 Project Properties 窗口的“Makefile”类别中更改 Makefile 项目设置。

关于将项目导出为十六进制文件的信息，请参见 12.15 Projects 窗口中的“项目菜单”。

表 6-2. 表 1：选择项目名称和文件夹选项

项	说明
Project Name	为项目指定一个名称。
Project Location	输入或浏览到项目的位置。
Project Folder	根据 Project Name 和 Project Location 查看文件夹位置。 关于项目文件夹与工作文件夹关系的信息，请参见 6.6.2 用户 Makefile 项目文件夹和工作文件夹。
Set as main project	将创建的项目设置为主项目。默认已选中。

.....（续）

项	说明
Overwrite existing project（覆盖现有项目）	如果项目名称已被使用，则该复选框将处于激活状态。
Also delete sources（同时删除源代码）	如果项目已存在并包含源代码，则该复选框将处于激活状态。
Use project location as the project folder	在与项目位置相同的文件夹中创建项目。 如果要导入 MPLAB IDE v8 项目并希望 MPLAB X IDE 项目位于同一文件夹中（例如，当您希望编译相同时），这将十分有用。 MPLAB X IDE 使用文件夹来存储项目信息，而 MPLAB IDE v8 使用.mcp 文件。因此，只能使一个 MPLAB X IDE 项目与一个 MPLAB IDE v8 项目（.mcp 文件）共享文件夹。
Encoding	默认设置为 ISO-8859-1。通常无需更改此设置。

表 6-3. 表 2：创建用户 Makefile 项目选项

项	说明
Working Directory（工作目录）	指定外部项目的位置。这是执行编译、调试编译和清除命令的默认位置。 关于项目文件夹与工作文件夹关系的信息，请参见 6.6.2 用户 Makefile 项目文件夹和工作文件夹。 建议通过浏览来选择目录，因为这会为您处理格式问题*。
Build command（编译命令）	当在 MPLAB X IDE（图标或菜单）中请求 Run 命令时，将根据该命令执行编译。 编译路径*：对于 Windows，使用引号。对于 Linux 或 Mac OS，必须对所有空格进行转义。
Debug build command（调试编译命令）	当在 MPLAB X IDE（图标或菜单）中请求 Debug 命令时，将根据该命令执行调试编译。 调试编译路径*：对于 Windows，使用引号。对于 Linux 或 Mac OS，必须对所有空格进行转义。 另请参见： http://microchipdeveloper.com/mplabx:work-outside-compile-for-debug
Clean command（清除命令）	当在 MPLAB X IDE（图标或菜单）中请求 Clean 命令时，将根据该命令执行清除。 清除路径*：对于 Windows，使用引号。对于 Linux 或 Mac OS，必须对所有空格进行转义。
Image name（映像名称）	输入生产映像文件（十六进制）的路径和名称。 映像路径*：必须对所有空格进行转义。
Debug image name（调试映像名称）	输入调试映像文件（ELF 或 COF）的路径和名称。 调试映像路径*：必须对所有空格进行转义。
User Include Paths（用户包含路径）	输入或浏览到用户包含文件。在该对话框中管理包含路径（图 1）。
User Macros（用户宏）	添加或解析并添加宏。 编辑：输入宏或浏览到包含一个宏的文件。在该对话框中管理宏（图 2）。 解析：自动解析宏（仅适用于 MPLAB XC 编译器）。按照该对话框中的说明进行操作。完成后，选择 Replace any current macro(s) listed（替换列出的任何当前宏）或 Append to any current list of macros（附加到任何当前宏列表）。关于如何使用该对话框的示例，请参见 6.6.3 Makefile 项目示例。

图 6-11. 选择包含路径对话框

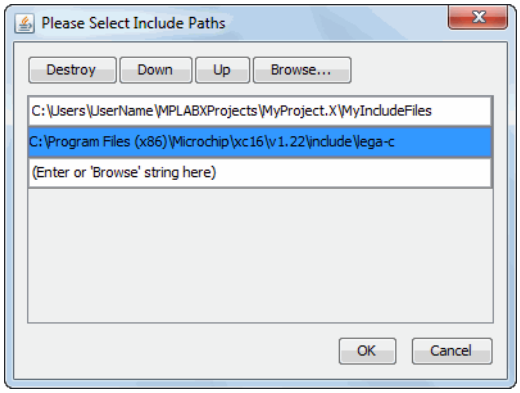
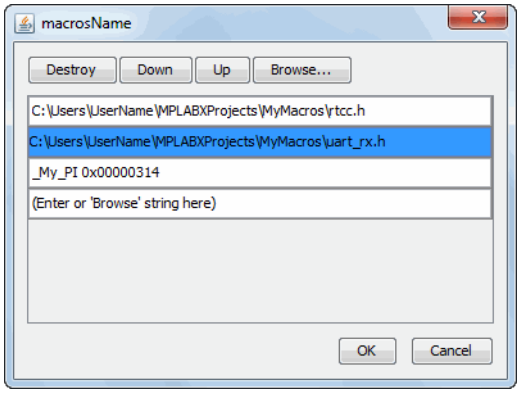


图 6-12. macrosName 对话框



6.6.2 用户 Makefile 项目文件夹和工作文件夹

项目文件夹包含自己的 `Makefile` 和 `nbproject/Makefiles`。MPLAB X IDE 将运行项目文件夹中的 makefile，然后切换到工作文件夹（目录）并运行“编译命令”或“调试编译命令”。实际上，将运行 GNU Make，然后运行被指定为编译命令的任何内容。

如果使用绝对路径定义位置，则项目文件夹可以位于任何位置（与工作文件夹相关）。但是，这会降低项目的可移植性。应使工作文件夹与项目文件夹相关（例如，位于项目文件夹的下一级），以提高项目的可移植性。

6.6.3 Makefile 项目示例

该示例说明了如何在 MPLAB X IDE 中创建用户 makefile 项目，该项目将针对在 IDE 外部创建的用户代码运行批处理文件或 make。用户代码包含一个文件 `t.c`。下面列出了该文件中包含的简单代码。

示例： `t.c` 代码

```
#include <xc.h>
int main(void) {
while(1) {
}
return 0;
}
```

对于生产，代码链接到 `t_production.hex`；对于调试，代码链接到 `t_debug.elf`。

对于该 `t.c` 示例，假设批处理文件 `makeme.bat` 和用户 makefile `Makefile` 位于（Windows 操作系统）以下目录中：

```
C:\Users\MCHP\MPLABXProjects\makestuff\myOwnCodeWithItsOwnMakefile
```

用您的 *`UserName`* 替换 `MCHP`，以在您的 PC 上重新创建示例。

- 用户 Makefile 项目——基于批处理文件创建代码
- 用户 Makefile 项目——基于用户 Makefile 创建代码
- 用户 Makefile 项目——完成
- 用户 Makefile 项目——解析宏

6.6.3.1 用户 Makefile 项目——基于批处理文件创建代码

要编译代码，简单的批处理文件 `makeme.bat` 将以 `production`、`debug` 或 `clean` 为参数。如果传递 `production`，则编译 `t_production.hex`。如果传递 `debug`，则编译 `t_debug.elf`。如果传递 `clean`，则擦除 `.elf` 和 `.hex` 文件。

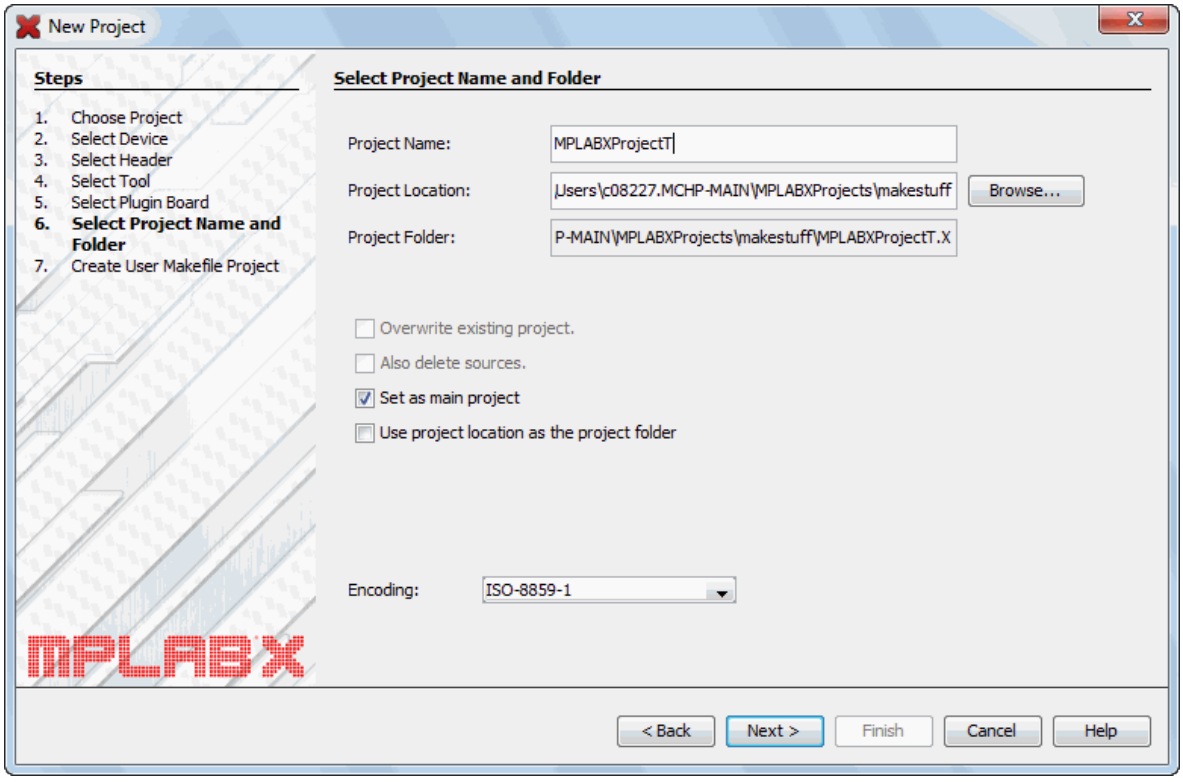
示例： `makeme.bat` 代码

```
::assumes xc32-gcc and xc32-bin2hex are on the path
rem @echo off
set COMPILER=xc32-gcc
set BIN2HEX=xc32-bin2hex
set PROCESSOR=32MX360F512L
if "%1" == "production" goto production
if "%1" == "clean" goto clean
:debug
%COMPILER% -mprocessor=%PROCESSOR% -g -D_DEBUG -o t_debug.elf t.c
goto end
:production
%COMPILER%  -mprocessor=%PROCESSOR% -o t_production.elf t.c
%BIN2HEX% t_production.elf
goto end
:clean
del /F *.elf
del /F *.hex
goto end
:end
```

要在 MPLAB X IDE 中创建新的用户 makefile 项目，请按照 6.6.1 新建项目——用户 Makefile 中的步骤进行操作：

1. **选择项目：**依次选择“Microchip Embedded”和“User Makefile Project”。
2. **选择器件：**选择 PIC32MX360F512L 器件。
3. **选择调试头：**对于该器件，无可用调试头。
4. **选择工具：**选择硬件工具（通过 SN）或软件模拟器。本示例中使用的是软件模拟器。
5. **选择接插板：**本示例中未使用。
6. **选择项目名和文件夹：**对于本示例，项目文件夹与 `myOwnCodeWithItsOwnMakefile` 同级。因此，项目位置将为：`C:\Users\MCHP\MPLABXProjects\makestuff`。项目名称为 MPLABXProjectT。

图 6-13. 用户 Makefile 项目——选择项目名称和文件夹



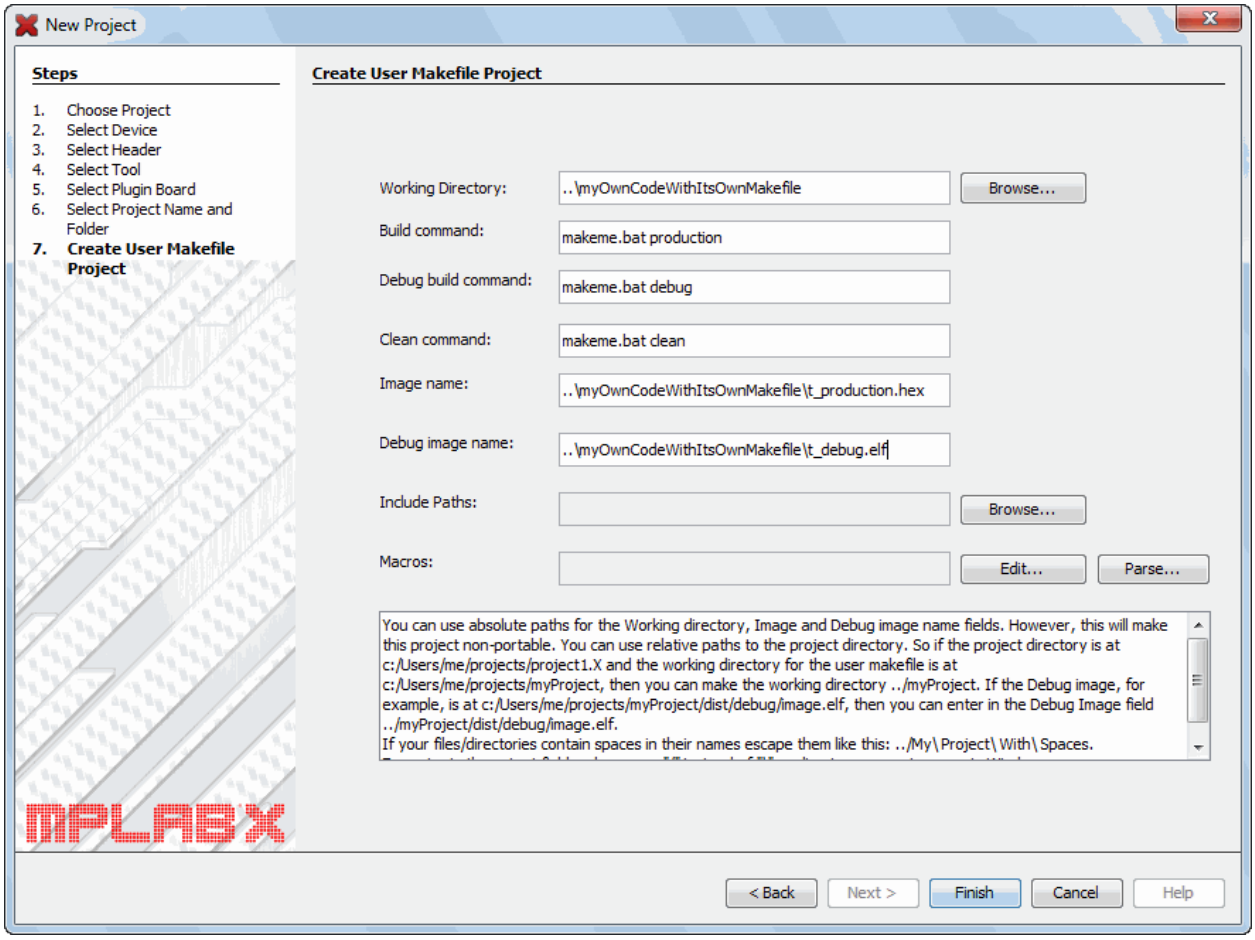
7. **创建用户 Makefile 项目：**输入信息来设置您的 makefile 项目。
Working Directory 是运行批处理文件编译、调试编译和清除命令的位置。该目录可设置为绝对路径：

   `C:\Users\MCHP\MPLABXProjects\makestuff\myOwnCodeWithItsOwnMakefile`。

   但是，这样做会使 MPLAB X IDE 项目的可移植性降低。因此，将以相对于 MPLAB X IDE 项目目录的相对路径输入 Working Directory、Image name 和 Debug image name。

   单击 **Finish**，创建项目。

图 6-14. 用户 Makefile 项目——基于批处理文件创建



6.6.3.2 用户 Makefile 项目——基于用户 Makefile 创建代码

要编译代码，简单的用户 makefile `Makefile` 将以 `production`、`debug` 或 `clean` 为参数。如果传递 `production`，则编译 `t_production.hex`。如果传递 `debug`，则编译 `t_debug.elf`。如果传递 `clean`，则擦除 `.elf` 和 `.hex` 文件。

示例：用户 Makefile 代码

```
# Assumes xc32gcc and xc32bin2hex are on the path
COMPILER=xc32gcc
BIN2HEX=xc32bin2hex
PROCESSOR=32MX795F512L
production: t_production.hex
debug: t_debug.elf
t_debug.elf: t.c
$(COMPILER) mprocessor=$(PROCESSOR) -g -D_DEBUG -o t_debug.elf
t.c
t_production.hex: t_production.elf
$(BIN2HEX) t_production.elf
t_production.elf: t.c
$(COMPILER) mprocessor=$(PROCESSOR) o t_production.elf t.c
clean:
del /F *.hex
del /F *.elf
```

要在 MPLAB X IDE 中创建新的用户 makefile 项目，请按照“用户 Makefile 项目——基于批处理文件创建代码”中的步骤 1-6 进行操作。 对于步骤 7，按照下图进行设置。

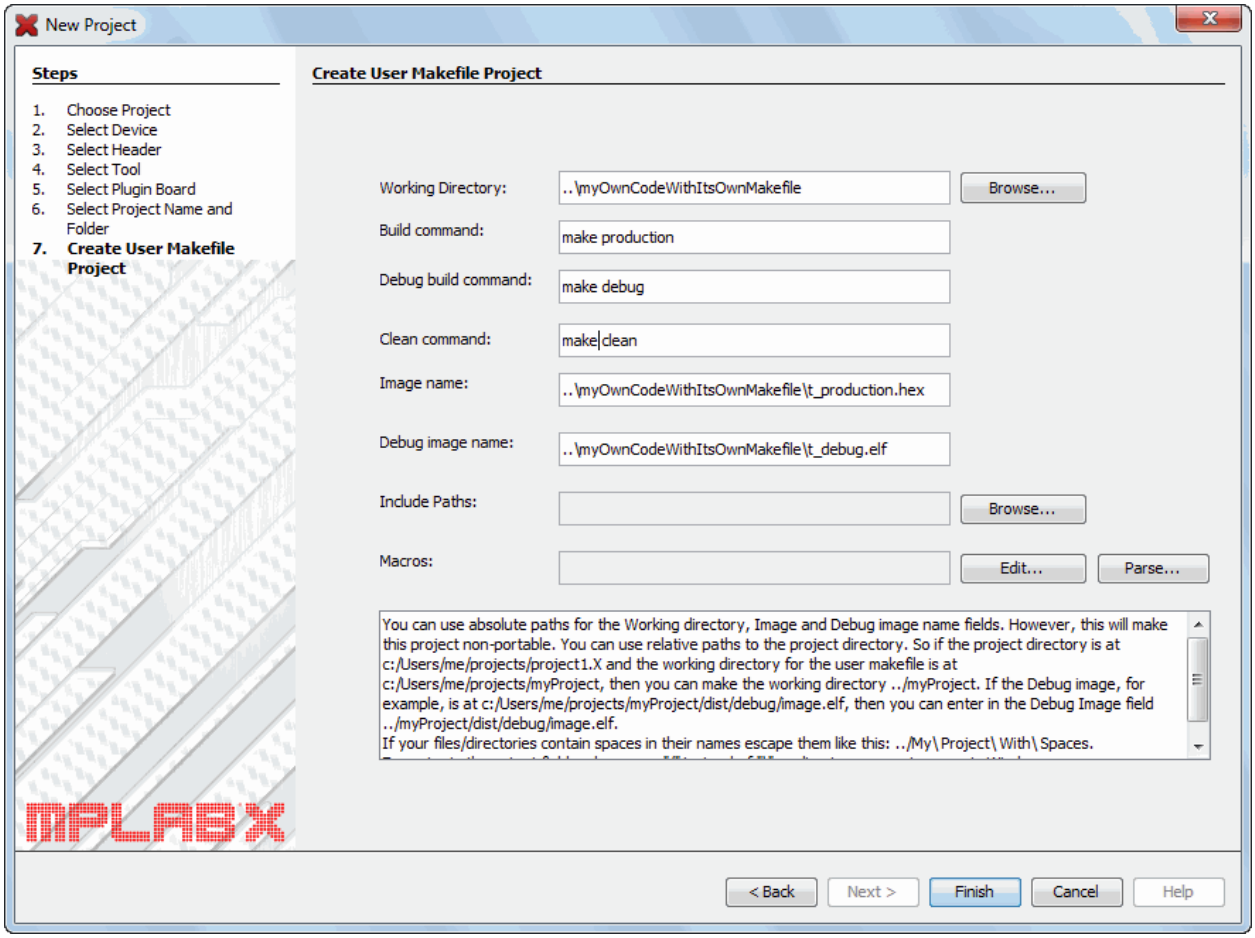
Working Directory 是运行用户 Makefile 编译、调试编译和清除命令的位置。因此，将以相对于 MPLAB X IDE 项目目录的相对路径输入 Working Directory、Image name 和 Debug image name。

MPLAB X IDE 使用 GNU Make 进行编译、调试编译和清除。根据“用户 Makefile 项目文件夹和工作文件夹”，MPLAB X IDE 将在项目文件夹中运行 Makefile，这会将目录更改为工作文件夹。make 将在该位置查找 makefile 并查找用户定义的 Makefile，该文件将定义编译、调试编译和清除的方式。

注： 另一种定义命令（例如 Build 命令）的方法是使用`-f` 选项来指定用户 makefile，例如：`make -f Makefile production`。

单击 **Finish**，创建项目。

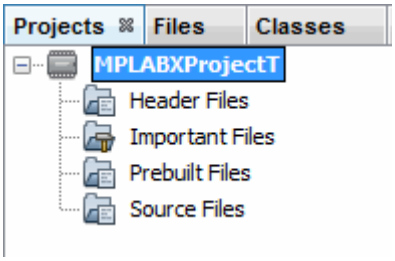
图 6-15. 用户 Makefile 项目——基于 Makefile 创建



6.6.3.3 用户 Makefile 项目——完成

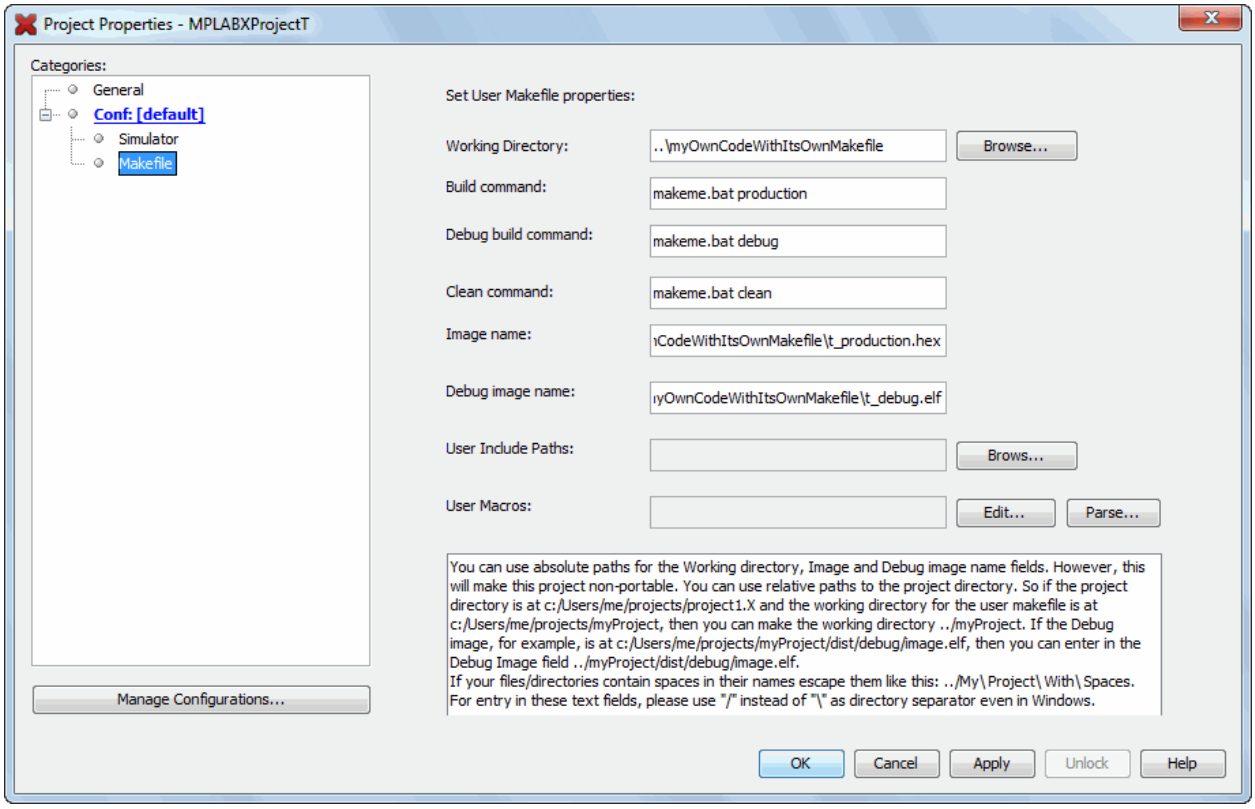
创建项目后，Projects 窗口中应显示以下内容。右键单击项目名可以编译项目。

图 6-16. 用户 Makefile 项目——项目树



如果要更改设置，请右键单击项目名称，并选择“Properties”。

图 6-17. 用户 Makefile 项目——项目属性



对于本示例，编译（t_production.hex）和调试编译（t_debug.elf）的结果位于 myOwnCodeWithItsOwnMakefile 工作目录下。由于 MPLAB X IDE 中仅显示 MPLABXProjectT 项目目录，因此编译后 IDE Projects 窗口中不会显示任何更改。

6.6.3.4 用户 Makefile 项目——解析宏

您可以解析 MPLAB XC C 编译器的宏并将其添加到 makefile 项目中。为此，需要在命令行上使用相关参数调用编译器，然后将整个输出复制到“Parse macros”（解析宏）对话框的第一个面板中。

要在命令行上运行编译器，需要一个 C 文件。对于本示例，复制下面的代码并将其粘贴到名为 xcmain.c 的文件：

```
#include <xc.h>
void main(void) {
return;
}
```

确保您的计算机已知 MPLAB XC 编译器的路径。 然后在命令行上执行以下操作：

```
xc32-gcc -dM -E xcmain.c > macros32.txt
```

使用的编译器是 MPLAB XC32 C 编译器，而不是 MPLAB XC32++编译器，因此 xc32-gcc 是可执行文件。选项-dM 通知预处理器在预处理结束时仅输出有效的宏定义列表。 选项-E 在预处理阶段之后停止执行。重定向（>）字符将输出发送到文件 macros.txt 中。

对于其他 MPLAB XC C 编译器，可以使用：

```
xc16-gcc -dM -E xcmain.c > macros16.txt
xc8 --pre -v --chip=8BitMCU xcmain.c > macros8.txt
```

其中，8BitMCU 应替换为您的 8 位器件。对于 macros8.txt 文件，还需要进行额外的编辑来将宏与文件中的其他文本隔离。

要打开“Parse macros”对话框：

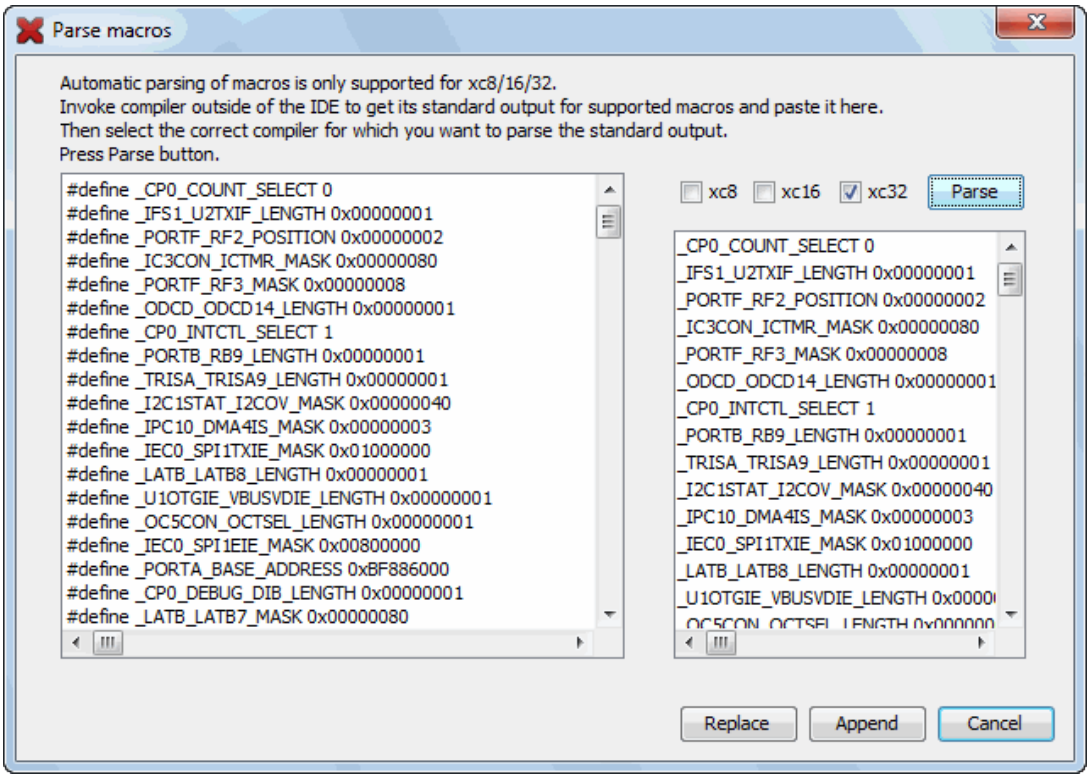
1. 右键单击项目名称，并选择“Properties”。
2. 在“Categories”下，选择“Makefile”。
3. 在“Set User Makefile properties”（设置用户 Makefile 属性）下，转至“User Macros”，然后单击 **Parse**（解析）按钮。

要解析宏：

1. 复制 `macros.txt` 的内容，并将其粘贴到对话框左侧面板。
2. 选中 MPLAB XC C 编译器对应的复选框，在本例中为“xc32”。
3. 单击 **Parse** 按钮。

单击 **Replace**（替换）按钮可替换 User Macros 文本框的当前内容，单击 **Append**（附加）按钮可附加到文本框中的现有文本，单击 **Cancel** 按钮可使现有文本保持不变。

图 6-18. Parse macros 对话框



6.7 使用源文件文件夹的链接资源

宏可用于指向源码库（或库版本），并可更改为移至另一个源码库。这在 MPLAB Harmony 应用中非常有用。

在 MPLAB X IDE 中打开项目，选择 *File>Project Properties*。单击“General”类别，然后单击“Macro”旁边的“Browse”（浏览）。找到并打开包含源码库的文件夹。单击 **OK**。在项目中，以“Relative to Macro”的形式打开该源码库中的文件。

要更改为另一个源码库，请浏览至“Macro”下包含新源码库的文件夹，然后选择该文件夹。现在，旧源码库中与新源码库匹配的所有文件都将引用新源码库。

6.8 与第三方硬件工具配合使用

与第三方硬件工具配合使用时，应满足以下基本安装要求：

- 为硬件获取并安装任何相关的 USB 驱动程序。USB 驱动程序通常可从第三方网站获取，随附安装说明。

- 为该工具安装 MPLAB X IDE 插件。有关详细信息，请参见 5.26 添加插件工具。

安装完成后，您将能够在 Project Properties 窗口（见 4.4 查看或更改项目属性）中查看和选择硬件工具。

6.9 使用代码覆盖率功能

通过代码覆盖率可以了解代码的哪些部分被执行了。要查看代码覆盖率输出，必须为 MPLAB X IDE 搭配使用支持代码覆盖率的 MPLAB XC C 编译器。支持代码覆盖率的 MPLAB X IDE 和编译器的最低版本如下：

- MPLAB X IDE v5.25
- MPLAB XC8 v2.10
- MPLAB XC16 v1.40
- MPLAB XC32 v2.30

在 MPLAB X IDE 中，可以在 Project Properties 窗口的编译器类别下使能代码覆盖率。使能后，调试测试代码，待其遍历执行后暂停。选择 *Window>Debugging>Code Coverage*（窗口>调试>代码覆盖率）可在 **Code Coverage** 选项卡窗口中查看已执行代码占总代码的百分比。

欲了解关于覆盖率的更多详细信息，可以购买 MPLAB Code Coverage 附加许可证（SW006026-COV）。这将激活编译器的全部代码覆盖率功能，其中包括：

- 用彩色高亮显示编辑器文本来表示代码覆盖率
- 写入 Code Coverage 选项卡窗口的详细报告

更多信息，请参见：

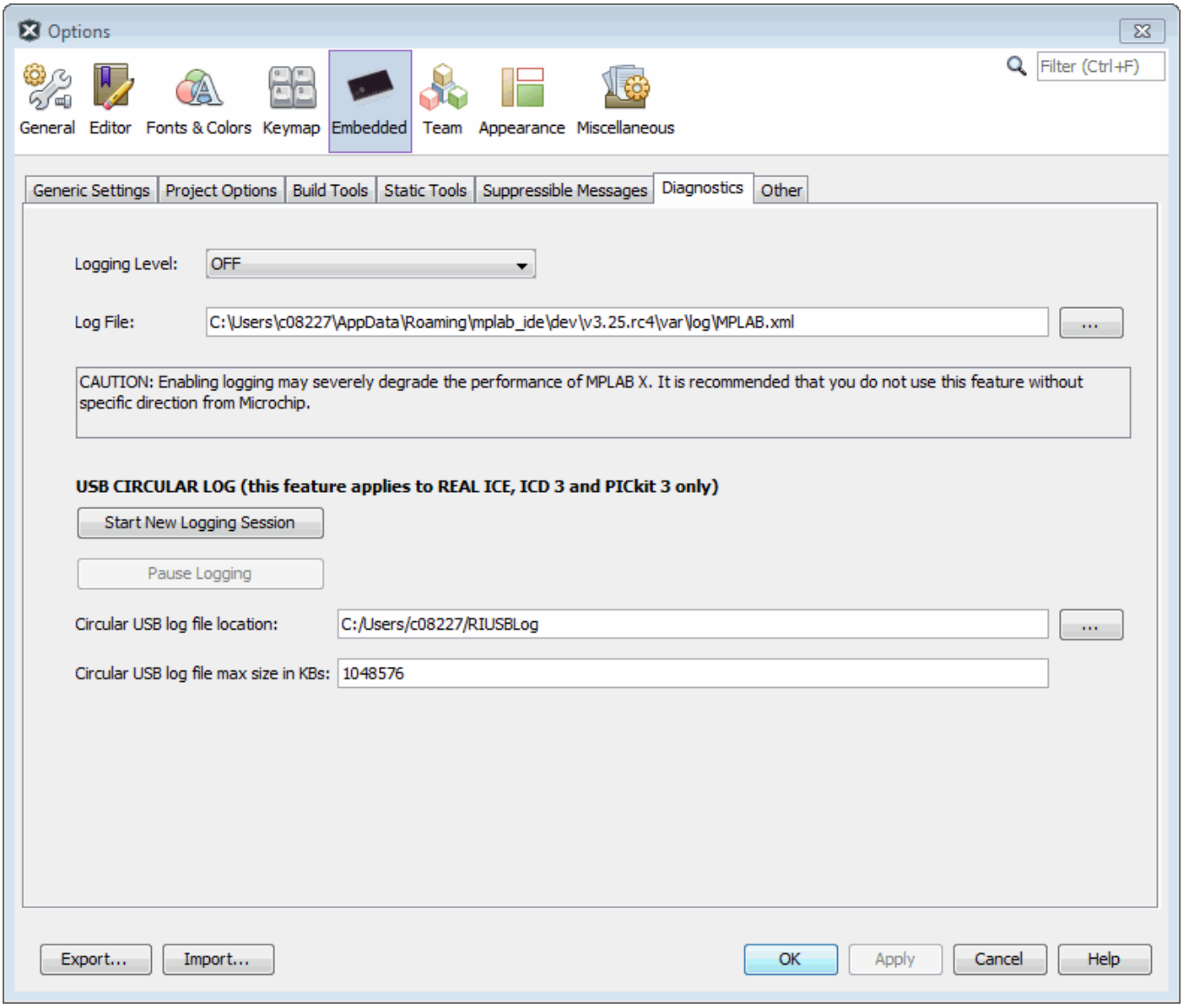
http://www.microchip.com/mplab/codecoverage

6.10 记录数据

日志文件可用于捕捉程序执行和调试中出现的问题。

当您遇到 MPLAB X IDE 错误或问题，需要与技术支持联系时，系统会要求您将数据捕捉到两个日志文件中：MPLAB X IDE 日志文件和 NetBeans 平台日志文件。此外，还将要求您提供其他支持信息。

图 6-19. 数据记录



6.10.1 MPLAB® X IDE 日志文件

MPLAB X IDE 日志文件基于 Java Logger 类来生成数据。

要设置日志文件:

1. 选择 *Tools>Options*（对于 macOS 为 *MPLAB X IDE>Preferences*），**Embedded** 按钮，**Diagnostics**（诊断）选项卡。
2. 从下拉框中选择一个日志记录级别。
   注： 日志记录级别越高，收集的数据就越多，但应用程序的运行速度就越慢。
3. 选择日志文件的位置（路径）。

要记录数据日志:

1. 使用日志记录级别“Finest”设置日志文件。
2. 记下日志文件的名称和位置。
3. 重复会导致错误或问题的步骤。
4. 找到日志文件并将其连同其他请求的信息一起发送给技术支持。

6.10.2 NetBeans 平台日志文件

NetBeans 日志文件会生成关于执行 NetBeans 平台的信息。

***要记录数据日志*:**

1. 通过选择 *View>IDE log*（视图>IDE 日志）打开日志文件 Output 窗口。
2. 在窗口中单击右键并选择“Clear”（清除）。
3. 重复会导致错误或问题的步骤。
4. 在窗口中单击右键并选择“Save As”（另存为）将文本保存到一个文件中。
5. 将该文件连同其他请求的信息一起发送给技术支持。

6.10.3 其他支持信息

尽管日志十分有用，但技术支持仍然需要您提供一些其他信息来帮助他们解决您的问题。这些信息包括：

- 对应于日志文件的执行步骤。
- 关于项目工具的详细信息，例如使用了哪些调试和语言工具以及选择了这些工具的哪些版本。
- 其他 MPLAB X IDE 项目数据（必要时）。

6.10.4 日志记录注意事项

日志记录时：

- 日志文件应尽可能简短且集中。也就是说，与其长时间运行代码并进行记录直到发生错误，不如尝试更为集中地捕捉出错前的运行情况。
- 对于硬件工具，循环日志对于确定 USB 通信问题十分有用。

6.11 打包 MPLAB X IDE 项目

右键单击项目，然后选择“Package”（打包），将项目打包为 zip 文件（`.zip` 文件）。虽然 MPLAB X IDE 具有打包为 zip 文件的功能，但它无法解压缩它们。所以需要一个单独的程序来解压缩该项目。

为了打包项目，该功能会检查项目文件，以确定要包含的项目文件的位置。将仅包含在项目文件夹中包含并且使用相对而不是绝对路径的文件。

项目打包文件中包含源代码文件、`makefile` 和 `nbproject` 目录。

除非已将头文件添加到项目中，否则不包括这些文件。因此，如果需要任何用户生成的头文件，则可能无法编译解压缩的项目。

6.12 硬件工具连接和调试

硬件调试工具与目标之间的连接可以确定除器件相关调试功能外，还有哪些调试功能可用。

硬件工具与目标之间的连接——PIC MCU

表 6-4. 硬件工具与目标之间的连接——PIC MCU

连接	硬件支持(1)	调试支持(2,3,4)	跟踪支持(1)	支持速度
标准（ICSP）通信	**RI**、**ICD4** 和 **ICD3**：模块化电缆（6 引脚 RJ-11）；**Snap**、**PK4** 和 **PK3**：带状电缆（6 引脚 SIL）	基本和高级	**RI**：本机跟踪	15 MIPS 或更低 PIC32：取决于器件

.....（续）

连接	硬件支持(1)	调试支持(2,3,4)	跟踪支持(1)	支持速度
高速（LVDS）通信	**RI**：高性能工具包（AC244002）	基本和高级	**RI**：本机跟踪， **RI**：SPI 跟踪	大于 15 MIPS PIC32：取决于器件
	RI：高性能工具包 + 隔离器（AC244005）	基本和高级	无	大于 15 MIPS PIC32：取决于器件
JTAG	**RI**：JTAG 适配器（AC244007）， **SJL**	基本	无	PIC32：取决于器件
逻辑端口	**RI**：逻辑端口探针（4 引脚）	N/A	**RI**：I/O 端口跟踪	取决于器件
	RI：跟踪接口工具包（AC244006）	N/A	**RI**：指令跟踪（PIC32 MCU，一些仿真头）	取决于器件
标准（ICSP）通信 + 逻辑端口	**RI**：电源监视器（AC244008）	基本	无	15 MIPS 或更低 PIC32：取决于器件

1. 关于工具的缩写，请参见上表。
2. 支持取决于器件，即某些调试功能并非在所有器件上都可用。
3. *基本调试支持*：运行、暂停、复位、单步进入、单步跳过和软件/硬件断点。
4. *高级调试支持*：数据捕捉和运行时观察等。

表 6-5. 硬件工具与目标之间的连接——AVR MCU

连接	硬件支持(1,2)	器件支持(3)	调试支持(4,5,6)	跟踪支持	支持速度
JTAG	AI 和 PK4	所有 AVR MCU	基本	否	32 kHz 至 7.5 MHz
aWire	AI	AVR 32 位 MCU	基本	否	7.5 kbps 至 7 Mbps
PDI 双线	AI 和 PK4	AVR XMEGA® MCU	基本	否	32 kHz 至 7.5 MHz
debugWIRE	AI 和 PK4	AVR 8 位 MCU	基本	否	4 kbps 至 0.5 Mbps
UPDI	AI 和 PK4	AVR 8 位 MCU	基本	否	最高 750 kbps
SPI	AI 和 PK4	AVR 8 位 MCU	仅限编程	N/A	8 kHz 至 5 MHz
TPI	AI 和 PK4	tinyAVR 8 位 MCU	仅限编程	N/A	取决于器件

1. 关于工具的缩写，请参见上表。
2. ICD4 和 PK4 需要 AC102015 调试器适配器板。
3. 器件支持即将提供，暂时可能不可用。
4. 支持取决于器件，即某些调试功能并非在所有器件上都可用。
5. *基本调试支持*：运行、暂停和软件/硬件断点。
6. *仅限编程*：该连接无需调试。

表 6-6. 硬件工具与目标之间的连接——SAM MCU

连接	硬件支持(1,2)	器件支持(3)	调试支持(4,5,6)	跟踪支持	支持速度
JTAG	ICD4	SAM MCU	基本	否	32 kHz 至 7.5 MHz

.....（续）

连接	硬件支持(1,2)	器件支持(3)	调试支持(4,5,6)	跟踪支持	支持速度
SWD	ICD4	SAM MCU	基本和高级	ITM——3 Mbps	32 kHz 至 10 MHz

1. 关于工具的缩写，请参见上表。
2. ICD4 和 PK4 需要 AC102015 调试器适配器板。
3. 器件支持即将提供，暂时可能不可用。
4. 支持取决于器件，即某些调试功能并非在所有器件上都可用。
5. 基本调试支持：运行、暂停和软件/硬件断点。
6. 高级调试支持：相较于基本调试支持的附加功能；取决于器件。

表 6-7. 工具缩写

缩写	工具名称
Snap	MPLAB Snap 在线调试器/生产编程器
PK4	MPLAB PICkit 4 在线调试器/生产编程器
PK3	PICkit 3 在线调试器/开发编程器
ICD4	MPLAB ICD 4 在线调试器/生产编程器
ICD3	MPLAB ICD 3 在线调试器/生产编程器
RI	MPLAB REAL ICE 在线仿真器/生产编程器
AI	Atmel-ICE 在线调试器/编程器
SJL	SEGGER J-Link 调试探头（第三方）

关于取决于器件的调试功能的更多信息，请参见：

`<MPLAB X IDE 安装目录>docs/FeatureSupport/HWToolDebugFeatures.html`

6.13 使用 IDE 脚本

MPLAB X IDE 具备脚本功能，可通过编程来控制 IDE 行为。可为 Microchip MPU 项目开发脚本，也可为 MCU 项目开发脚本。

关于脚本

脚本采用 Jython 语言编写。脚本可以访问 API，从而允许您：

- 控制调试会话状态（运行和暂停等）。
- 在调试时访问器件的存储器。
- 基于函数回调设置断点，这些函数确定是否应暂停执行。这样便可实现条件断点。
- 将操作置于 Projects 窗口中，单击该窗口将在一个脚本中运行多种方法。
- 提供用于调试事件的钩子函数：发生给定事件时将调用一个函数。

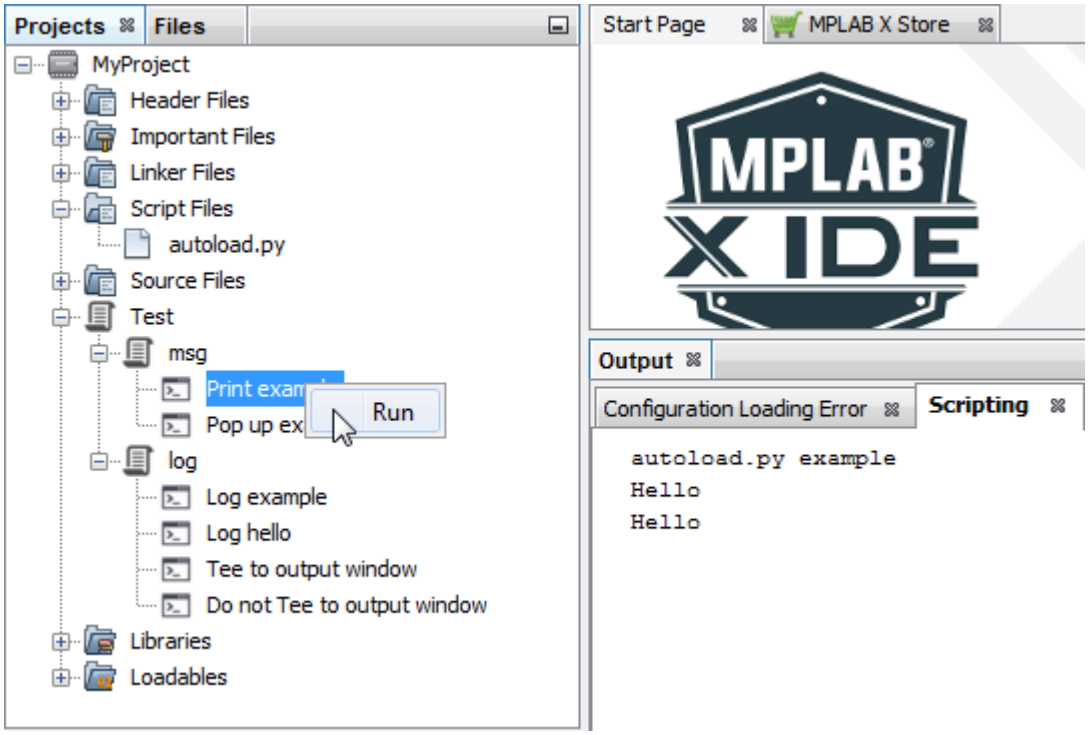
使用脚本

要访问脚本功能，请按以下步骤操作。

1. 关闭将使用脚本的项目。
2. 创建名为 `autoload.py` 的文件。用于说明 API 的自说明 `autoload.py` 位于`<MPLAB X IDE 安装文件夹>\docs\example_code` 下。
3. 将文件复制到您的项目目录中。根据需要在此处编辑文件。

4. 重新打开项目，项目树中将出现一个“Script Files”（脚本文件）文件夹。文件 `autoload.py` 将位于该文件夹下。
5. 根据需要运行“Test”（测试）下的脚本。

图 6-20. 项目中未经编辑的 autoload.py



6.14 校验和

校验和是从 MCU 或 DSC 的应用程序代码和其他设置得到的十六进制数。关于如何生成校验和的详细信息，请参见器件编程规范。

编译项目时将创建校验和。可以在 Dashboard 窗口中查看生成的校验和： 5.19 查看仪表板显示

Checksum: 0x339C

校验和用于检测编译间的错误。例如，在一台计算机上编译了一个项目并生成了一个校验和。如果将该项目复制到另一台计算机并在其中进行编译，则校验和相同表示已成功复制该项目，而校验和不同则表示复制失败。

如果将校验和用于错误检测，则在开发项目代码时应格外小心。不要使用编译器宏 `__TIME__` 或 `__DATE__`。此外，如果文件路径会发生更改，请勿使用宏 `__FILE__` 或 `assert()`。

对于某些器件，校验和可用作用户 ID：4.10.3 Insert unprotected checksum in user ID memory（在用户 ID 存储区中插入未受保护的校验和）。

6.15 配置

许多不同的上下文中都使用了术语“配置”。下表列出了相关术语及其定义。

表 6-8. 配置上下文

术语	定义
配置位 配置熔丝 配置字 配置字节 配置寄存器	所有这些术语均指器件中物理存储器的一部分，用于配置器件。关于可用配置设置，请参见器件的数据手册。
Configuration Bits 窗口	MPLAB X IDE 窗口，显示可用于项目器件的配置设置。关于该窗口的更多信息，请参见 4.19 在 Configuration Bits 窗口中设置配置值。
项目配置	项目的编译设置。创建项目时，配置为“[default]”。可以更改此名称，并且可以为项目创建多个配置。 更多信息，请参见 6.4 处理多个配置。 此数据存储在文件 `configurations.xml` 中。
项目配置类型	针对特定项目类型的设置。这仅适用于配置为“application”的独立项目和配置为“library”的库项目。有关详细信息，请参见 4.10.1 更改项目配置类型。
用户配置数据	用户创建的 MPLAB X IDE 配置设置。请参见 8.7 查看用户配置数据。

7. 编辑器

MPLAB X IDE 是基于 Netbeans 平台而构建的，提供了一个内置编辑器来创建新代码或更改现有代码。

关于 NetBeans 源代码编辑器的信息，请参见：

https://docs.oracle.com/cd/E50453_01/doc.80/e50452/working_nbeans.htm#A1151635

NetBeans 帮助主题下提供了关于编辑器的 C 编译器信息：

https://netbeans.org/kb/docs/cnd/navigating-editing.html

7.1 编辑器用法

要开始使用编辑器，可以使用 *File>New File* 创建文件或使用 *File>Open File* 打开现有文件。以下几节介绍了附加的控件和功能。

桌面控件

以下桌面项与编辑器关联：

- File 菜单——用于在编辑器窗口中打开文件（见 10.1.1 File 菜单）。
- Edit（编辑）菜单——用于访问编辑命令（见 10.1.2 Edit 菜单）。
- 编辑器工具栏——也可用于访问编辑命令——位于每个文件的编辑器窗口顶部（见 10.2.10 编辑器工具栏）。
- 窗口右键（上下文）菜单——用于访问更多的命令。

显示控件

下表列出了基本显示控件。7.2 编辑器选项下给出了其他格式控件。

操作	说明
最大化源代码编辑器。	执行以下操作之一： • 在源代码编辑器中双击文件的选项卡。 • 确保源代码编辑器窗口为焦点，然后按 Shift-Esc。 • 选择 *Window>Configure Window>Maximize*（窗口>配置窗口>最大化）。
将最大化的源代码编辑器还原为其之前的大小。	执行以下操作之一： • 在源代码编辑器中双击文件的选项卡。 • 按<Shift> + <Esc>。 • 选择 *Window>Configure Window>Restore*（窗口>配置窗口>还原）。
显示行号。	选择 *View>Show Line Numbers*（视图>显示行号）。
同时查看两个文件。	1. 打开两个或更多文件。 2. 单击其中一个文件的选项卡，然后将其拖到要放置文件的窗口的一侧。当出现红色预览框以指示窗口即将放置的位置时，释放鼠标按钮将窗口放下。 该窗口支持水平或垂直拆分，具体取决于将选项卡拖动到的位置。
拆分单个文件的视图。	1. 在源代码编辑器中右键单击文档的选项卡，然后选择“Clone Document”（克隆文档）。 2. 单击已克隆文档的选项卡，然后将其拖到要放置副本的窗口部分。
在打开的文件之间切换。	1. 按住<Ctrl>。 2. 按<Tab>。此时将出现一个选择框。 3. 选择要切换到的文件。

.....（续）

操作	说明
自动格式化代码。	右键单击源代码编辑器，然后选择“Format”。 如果选择了任何文本，则仅格式化该文本。如果未选择任何文本，则将格式化整个文件。

基本编辑器功能

下表总结了编辑器的一些更常用的功能。

编辑器功能	参考
支持 Unicode。	默认情况下，在 IDE 中新创建的项目将使用 ISO-8859-1 字符编码。要更改该设置： 在 Projects 窗口中右键单击项目名称并选择 Properties。 在左侧栏的“Categories”下，选择“General”。 在页面底部，查找“Encoding”并进行更改。
代码基于语法进行着色。	*Tools>Options*（对于 macOS 为 *MPLAB X IDE>Preferences*），**Fonts and Colors** 按钮，**Syntax** 选项卡。 基于项目创建过程中设置的编码。
在输入代码时标记错误。	请参见目录以获取关于 Netbeans Help（Netbeans 帮助）主题的信息： https://netbeans.org/kb/docs/java/editor-codereference.html#coloring
着色标记可用于快速访问多个符号和错误等。	*Tools>Options*（对于 macOS 为 *MPLAB X IDE>Preferences*），**Fonts and Colors** 按钮，**Annotations**（标注）选项卡
智能代码补全可提供建议和提供提示。	*Tools>Options*（对于 macOS 为 *MPLAB X IDE>Preferences*），**Editor** 按钮，**Code Completion**（代码补全）选项卡
在调试模式下将鼠标悬停在符号上可查看值。	调试时暂停，将鼠标悬停在符号上可查看值。 **注：** 这种方式无法查看宏的值。在调试期间右键单击（对于 macOS，则采用 Cmd+Alt+Click）编辑器中的宏，可以使用 Macro Expansion 窗口。
可以折叠和展开汇编及 C 代码。	请参见 7.5 代码折叠。
右键单击函数（`delay(x)`）可查找使用实例。这可以限制在函数内进行查找（例如，局部 `i` 变量）。	请参见 Netbeans Help 主题： http://wiki.netbeans.org/Find_Usages_in_Compiled_Dependencies
右键单击函数（`delay(x)`）可显示调用图 调用图一侧具有一些按钮，可用于切换顺序等。	请参见“查看调用图”。
在源代码中使用关键字（示例：`//TODO`）创建注释，将会自动对操作项进行扫描并添加到 Action Items（操作项）窗口中。	请参见 Netbeans Help 主题： https://netbeans.org/features/ide/index.html 获取 Options 窗口，Team: Action Items 的相关信息。
即使没有版本控制系统，也可以使用文件历史记录查看最近的更改并进行还原。	5.22 使用本地历史记录控制源代码
导航通过一些功能项进行简化，例如“Go to file”（转至文件）、“Go to type”（转至类型）、“Go to symbol”（转至符号）、“Go to header”（转至头文件）和“Go to declaration”（转至声明）。	请参见“Navigate 菜单”。

.....（续）	
编辑器功能	参考
用于改善代码结构的重构选项（重命名函数和变量以及查找所有函数等）。	请参见 7.6 C 代码重构。

编辑器疑难解答

关于在编辑器中高亮显示的代码的错误，请参见编译器文档。关于错误高亮显示的信息，请参见 NetBeans 帮助主题：

https://netbeans.org/kb/docs/java/editor-codereference.html#coloring

9.5 错误下给出了具体的错误。

7.2 编辑器选项

要设置编辑器选项：

1. 选择 *Tools>Options*（对于 macOS 为 *MPLAB X IDE>Preferences*），打开 Options 对话框。
2. 单击 **Editor** 按钮。然后单击选项卡以设置编辑器功能。

下面列出了每个选项卡及其选项。

表 7-1. General 选项卡

项	说明
Code Folding（代码折叠）	选中时可使能代码折叠，并选择要折叠的代码类型。关于 Code Folding 的更多信息，请参见 7.5 代码折叠。 另请参见 NetBeans 帮助主题： https://netbeans.org/kb/docs/java/editor-codereference.html
Camel Case Behavior（驼峰大小写行为）	选中时可使能驼峰大小写导航。

表 7-2. Formatting（格式设置）选项卡

项	说明
Language（语言）	选择将应用格式设置的编程语言。
Category（类别）	选择一个类别，当前只有“Tabs and Indents”（制表符和缩进）。
Expand Tabs to Spaces（将制表符扩展为空格）	选中时可将制表符转换为空格。
Number of Spaces per Indent（每个缩进的空格数）	输入每个缩进的等效空格数。
Tab Size（制表符大小）	输入制表符的大小。
Right Margin（右边距）	输入右边距的大小。

表 7-3. Code Completion 选项卡

项	说明
Language	选择将应用代码补全的编程语言。
Completion options（补全选项）	选中时可使能代码补全选项。关于代码补全的更多信息，请参见 NetBeans 帮助主题：https://netbeans.org/kb/docs/java/editor-codereference.html

表 7-4. Code Templates 选项卡

项	说明
Language	选择将应用模板的编程语言。
Templates	输入上述指定语言的模板信息。 ***Abbreviation***（缩写）：输入要在编辑器中键入的缩写。 ***Expanded Text***：在输入缩写和“Expand template on”（模板展开条件）字符之后，在编辑器中将缩写展开为该文本。 ***Description***（说明）：添加模板项的可选说明。 关于代码模板的更多信息，请参见 NetBeans 帮助主题： https://netbeans.org/kb/docs/java/editor-codereference.html
Expand template on	选择要在编辑器中输入什么字符以将缩写文本展开为展开文本。

表 7-5. Hints（提示）选项卡

项	说明
Language	选择将应用提示的编程语言。
Hint（提示）窗口	在 Hint 窗口中，选择一个事件并指定在发生该事件时希望接收到的“提示”：Error（错误）、Warning 或 Warning on current line（在当前行上的警告）。 关于使用提示的更多信息，请参见 NetBeans 帮助主题： https://netbeans.org/kb/docs/java/editor-codereference.html

表 7-6. Macros（宏）选项卡

项	说明
Macros	添加、删除和定义编辑器宏。 ***要创建新的宏***，请单击 **New**（新建）按钮，输入新宏的名称，然后从列表中选择新宏并在 Macro Code（宏代码）编辑器中输入代码（包含在引号中）。 ***要设置宏的键盘快捷键***，可以从列表中选择一个宏，单击 **Set Shortcut**（设置快捷键）按钮并在对话框中输入快捷键。使用该快捷键可运行您的宏。 ***要删除宏***，可以从列表中选择一个宏并单击 **Remove** 按钮。 ***要修改宏代码***，可以从列表中选择一个宏，并在 Macro Code 编辑器中编辑代码。
Macro Code	单击上面的宏名称，然后输入宏代码（包含在引号中）。 关于宏关键字的列表，请参见： http://wiki.netbeans.org/FaqEditorMacros **注：** 通常，通过录制宏来添加新宏会比在 Macro Code 编辑器中手动添加更简便。请使用 *Edit>Start/Stop Macro Recording*（编辑>开始/停止宏录制）。

7.3 代码中的超链接

代码中的超链接可帮助您导航到与链接内容有关的信息。

C 代码中的超链接

超链接导航功能使您可以从函数、变量或常量的调用跳至其声明。

要使用超链接，请执行以下操作之一：

- 将鼠标悬停在某个函数、变量或常量上，同时按下<Ctrl>键（Windows 和 Linux）或<Command>键（Mac）。此时会显示一个超链接，以及包含有关元素信息的工具提示。单击该超链接，编辑器会跳至声明。按下<Alt> + <向左箭头>可跳回到调用。
- 将鼠标悬停在某个标识符上，并按下<Ctrl> + <B>或<Command> + <B>，编辑器会跳至声明。按下<Alt> + <向左箭头>可跳回到调用。

按下<Alt> + <向左箭头>和<Alt> + <向右箭头>可向后和向前遍历光标位置的历史记录。

汇编代码中的超链接

要导航到汇编源文件中包含的头文件，可以按下<Ctrl>键（Windows 和 Linux）或<Command>键（Mac），同时将鼠标光标放置在由`#include`语句引用的文件名上。然后单击鼠标选择按钮，在编辑器中在包含文件自己的文件选项卡中打开包含文件。

7.4 编辑器红色感叹号

源代码编辑器会对源代码中的语法和语义错误（例如，缺少分号和标识符未解析）进行高亮显示。错误会以带红色下划线的形式显示，并且错误所在行旁边的编辑器窗口的左边缘均会显示红色感叹号“字形”  。将鼠标悬停在此字形上时，会显示一条错误消息。

注： 编辑器中的错误并非都是真正的错误，也可能是无法识别的符号。应尝试编译代码，以确定是否仍然可以进行编译。更多详细信息，请参见“由无法识别的符号引起的错误”部分。

错误条件和附加标记

执行以下任一操作时可能会出错：

- 输入新代码
- 编辑现有代码
- 向项目中添加现有文件
- 将文件移至项目中的其他位置
- 更改项目或其源文件之一的属性

如果文件中存在任何错误，则编辑器窗口右侧的错误条中会显示红色标记。错误条代表整个文件，而不仅仅是当前显示的行。双击错误条中的标记，即可跳至该标记所代表的行。

由无法识别的符号引起的错误

编译器的预解析器或解析器无法识别的符号可能被标记为错误，但实际上可能不是错误（代码将会编译）。因此，如果您怀疑是这类错误，可尝试继续编译项目。

关于示例，请参见 9.5 错误下列出的以下内容：

- 对于 8 位代码，#asm 和#endasm 会导致编辑器窗口中出现红色感叹号。
- 对于 16 位代码，MPLAB XC16 packed 属性语句会导致编辑器窗口中出现红色感叹号。

7.5 代码折叠

代码折叠使您可以隐藏代码的某些部分，从而可以专注于其他部分。可折叠（隐藏）或展开（显示）的 C 或汇编代码部分取决于您选择的选项。

默认情况下会使能代码折叠。对于大多数 C 或汇编代码，可折叠的代码部分通过代码左侧的“-”和“+”图标来指示。

使能并设置代码折叠

使用以下步骤来使能并设置代码折叠：

1. 选择 *Tools>Options*（对于 macOS 为 *MPLAB X IDE>Preferences*），打开 Options 对话框。
2. 单击 **Editor** 按钮。
3. 单击 **General** 选项卡，并选中“Use code folding”（使用代码折叠）。

4. 选中其他选项来指定要折叠的部分。如果发现其中未列出要折叠的代码的类型，可以进行定制折叠。

展开/折叠代码折叠块

在编辑器中，单击某个方法旁边的“-”图标，它会发生折叠。单击已折叠方法旁边的“+”图标，它会发生展开。折叠的代码通过省略号框来表示。将光标停留在省略号框上时，IDE 会显示折叠的方法。

图 7-1. 展开的代码

```
54   int main(void)
55   {
56       // <editor-fold defaultstate="collapsed" desc="initialization">
57       _display_state = DISP_HELLO; // Start from displaying of PIC24 banners
58       AD1PCFG = 0xffff; // Setup PortA IOs as digital
59       SPIMPolInit(); // Setup SPI to communicate to EEPROM
60       EEPROMInit(); // Setup EEPROM IOs
61       UART2Init(); // Setup the UART
62       TimerInit(); // Setup the timer
63       mLCDInit(); // Setup the LCD
64       BtnInit(); // Setup debounce processing
65       ADCInit(); // Setup the ADC
66       BannerStart(); // Setup the banner processing
67       RTCCInit(); // Setup the RTCC
68       // </editor-fold>
69           while (1) {
70                   LCDProcessEvents();
71                   ADCProcessEvents();
```

图 7-2. 折叠的代码

```
54   int main(void)
55   {
56       // <editor-fold defaultstate="collapsed" desc="initialization">
57       _display_state = DISP_HELLO; // Start from displaying of PIC24 banners
58       AD1PCFG = 0xffff; // Setup PortA IOs as digital
59       SPIMPolInit(); // Setup SPI to communicate to EEPROM
60       EEPROMInit(); // Setup EEPROM IOs
61       UART2Init(); // Setup the UART
62       TimerInit(); // Setup the timer
63       mLCDInit(); // Setup the LCD
64       BtnInit(); // Setup debounce processing
65       ADCInit(); // Setup the ADC
66       BannerStart(); // Setup the banner processing
67       RTCCInit(); // Setup the RTCC
68       // </editor-fold>
69           while (1) {
70                   LCDProcessEvents();
71                   ADCProcessEvents();
```

代码折叠——行内汇编和 MPLAB C18 C

在 MPLAB C18 中，代码折叠不适用于使用 _asm 和 _endasm 伪指令的嵌入汇编代码。变通方法是将代码块放在代码/文件末尾附近。

定制代码折叠——C

要对 C 代码进行定制折叠，请执行以下操作之一：

- 在代码前后输入以下注释：

```
// <editor-fold defaultstate="collapsed" desc="user-description">
C code block to fold
// </editor-fold>
```

或者

- 输入 `fcom`，然后按<Tab>键来自动输入上面的注释文本。

输入该注释之后，就可以为代码定制它：

`defaultstate`	输入 `collapsed` 或 `expanded`，
`desc`	输入代码的说明。在折叠之后，将仍然可以看到该说明，以供参考。

上面的“展开/折叠代码折叠块”部分给出了一个定制代码折叠的示例。

更多信息，请参见 NetBeans 帮助主题：

https://ui.netbeans.org/docs/ui/code_folding/cf_uispec.html

定制代码折叠——汇编

要对汇编代码进行定制折叠，请执行以下操作：

- 对于 MPASM 汇编器或 MPLAB XC16 汇编器代码，在代码前后输入以下注释：

```
; <editor-fold defaultstate="collapsed" desc="user-description">
要折叠的汇编代码块
; </editor-fold>
```

- 对于 MPLAB XC32 汇编器代码，在代码前后输入以下注释：

```
// <editor-fold defaultstate="collapsed" desc="user-description">
要折叠的汇编代码块
// </editor-fold>
```

或者

- 对于 MPASM 汇编器或 MPLAB XC16 汇编器代码，输入 `fcom`；然后按<Tab>键来自动输入以上相关注释文本。
- 对于 MPLAB XC32 汇编器代码，输入 `fcom//`，然后按<Tab>键来自动输入以上相关注释文本。

汇编代码将按照与 C 代码相同的方式折叠。关于代码折叠的示例，请参见上面的 C 代码示例。

展开/折叠定制代码折叠块

要展开/折叠代码块，请转到 View 菜单，或在编辑器窗口中单击右键，并选择如下选项之一：

1. *Code Folds>Collapse Fold*（代码折叠>折叠）可隐藏代码块。
2. *Code Folds>Expand Fold*（代码折叠>展开折叠）可显示代码块。
3. *Code Folds>Collapse All*（代码折叠>全部折叠）可隐藏所有折叠代码块。
4. *Code Folds>Expand All*（代码折叠>全部展开）可显示所有折叠代码块。

上面的“展开/折叠代码折叠块”部分给出了一个定制代码折叠的示例。

7.6 C 代码重构

重构指的是使用小转换来重构代码，而不改变任何程序行为。正如重构表达式以使之更易于理解或修改，重构代码也是为了使它更易于阅读、更易于理解，以及可更快速地更新。并且，正如重构后的表达式必须产生相同的结果，重构后的程序也必须在功能上等效于原始的源代码。

重构代码的一些常见动机包括：

- 使代码更易于更改或更易于添加新功能

- 降低复杂性，以便更好地理解
- 删除不必要的重复
- 使代码可以用于其他需要或更一般的需求
- 提高代码性能

IDE 的重构功能可简化代码重构，其方式为评估您希望进行的更改、显示应用程序会受到影响的部分，并对您的代码进行所有必要的更改。例如，如果您使用 Rename 操作来更改类名，IDE 将在您的代码中查找该名称的所有使用实例，并可为您更改该名称的每一处实例。

Refactor（重构）菜单

使用 IDE 的重构操作时，您可以更改代码的结构，并使代码的其余部分更新为反映您所做的更改。

重构操作可通过 Refactor 菜单使用（见 10.1.6 Refactor 菜单）。

撤消/重做重构更改

对于使用 Refactor 菜单中的命令所做的任何更改，可使用撤消/重做工具栏菜单上的按钮来撤消（或重做）。

	**Undo**（撤消）——IDE 将取消受重构影响的所有文件中的所有更改。
	**Redo**（重做）——IDE 将对所有受影响的文件重复进行重构更改。

如果在执行重构之后，任何受影响的文件发生了修改，重构 Undo/Redo 操作将不可用。

查找函数使用实例

您可以使用 Find Usages（查找使用实例）命令，确定在项目源代码中使用的每一个函数实例。

要查找项目中使用了函数的位置：

1. 在 Navigator 窗口或源代码编辑器窗口中，右键单击函数名称，并选择 Find Usages——<Alt>+<F7>（对于 macOS 为<Ctrl>+<F7>）。
2. Find Usages 命令会显示调用该函数的代码行。在 Find Usages 对话框中，单击 Find。Usages（使用实例）窗口会显示文件名和在该文件中找到的每个使用实例所在的代码行。

要跳转至函数的某个特定实例，请执行以下操作之一：

- 在 Usages 窗口中，使用左窗格中的向上和向下箭头按钮，从函数的一个使用实例转至下一个使用实例。
- 双击某个代码行，打开该文件并将光标定位到该代码行上。

其他 IDE 查找机制

可用于搜索项目中使用了特定文本的所有位置的其他 IDE 工具包括：

- ***Finding and Replacing Text***（查找并替换文本）。在编辑器中打开的源文件中搜索使用了特定文本的所有位置。选择 *Edit>Find*（编辑>查找）可打开 Find 对话框，选择 *Edit>Replace*（编辑>替换）可打开 Replace 对话框。这些命令会查找所有匹配字符串，无论字符串是否为 C 代码元素。
- ***Find in Projects***（在项目中查找）。与 Find 命令一样，Find in Projects 命令也会搜索匹配字符串，不论字符串是否为函数名称。选择 *Edit>Find in Projects*（编辑>在项目中查找）可打开“Find in Projects”对话框，然后输入要查找的文本字符串。

要在源文件中查找声明函数的位置，可以在 Navigator 窗口中双击函数。如果该函数是在其他源文件中声明的，则右键单击该函数，并从上下文菜单中选择 *Navigate>Go To Declaration*（导航>转至声明）。

重命名函数或参数

要重命名函数或参数，并更新整个项目中对它的引用：

1. 在源代码编辑器中，右键单击该函数或参数，并从上下文菜单中选择 *Refactor>Rename*（重构>重命名）。
2. 在 Rename 对话框中，输入新的名称。
3. （可选）单击 **Preview**（预览）。在源代码编辑器底部的 Refactoring（重构）窗口中，查看将更改的代码行，并清除不希望更改的任何代码的复选框。

4. 单击 **Do Refactoring**（执行重构）应用选定的更改。

要进行快速的原地重命名，请将光标放在要重命名的项上，并按下 Ctrl-R 组合键。然后输入新的名称。要完成重命名，请按 **Escape**（ESC 键）。

移动、复制和安全删除 C 代码

这些功能是特定于 C++代码的。请参见 10.1.6 Refactor 菜单。

7.7 C/C++代码错误伪指令

在代码中使用`#error` 伪指令时，最好将语句括在括号中，如下所示：

```
(#error This line shown only when there is an error.)
```

否则，语法解析器将像 C 编译器一样处理`#error` 伪指令，并立即停止。因此，紧跟在该伪指令之后的文本将在编辑器中呈灰显来予以表示。

8. 项目文件和文件夹

MPLAB X IDE 具有多个用于项目文件和文件夹的窗口（通常位于文件窗格中），这些窗口分别称为 Projects、Files、Favorites（收藏夹）和 Classes。

建议查看本章中与项目文件和文件夹相关的其他主题：

- 路径、文件和文件夹名称限制
- 路径、文件和文件夹项目建议
- 查看用户配置数据
- 导入 MPLAB IDE v8 项目——相对路径
- 移动、复制或重命名项目
- 删除项目

8.1 Projects 窗口视图

Projects 窗口是项目的主要入口点。该窗口显示重要项目内容的逻辑视图。关于该窗口的更多信息，请参见 12.15 Projects 窗口。

您至少需要将源文件添加到您的项目中。MPLAB X IDE 会为您查找许多默认文件（头文件和链接描述文件）。您可以添加库文件和预编译的目标文件，以及已编辑的头文件和链接描述文件。不包含在编译中的文件可以放在“Important Files”（重要文件）下。

图 8-1. Projects 窗口——简单 C 代码项目

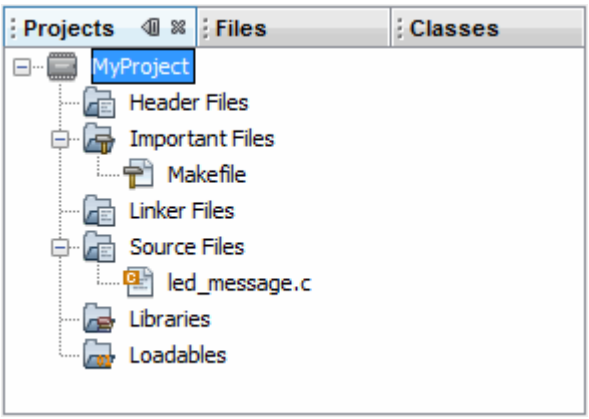


表 8-1. Projects 窗口定义

虚拟文件夹	包含的文件
Header Files	头文件（.h 或.inc）
Important Files	重要的文件，例如 makefile。其他文档可以放在此处，例如数据手册 PDF。
Linker Files（链接器文件）	链接器文件（.ld、.gld 或.lkr）
Source Files	源文件（.c、.cpp 或.asm）
Libraries	库文件（.a 或.lib）
Loadables	预编译的目标文件（.o）

另请参见 NetBeans 帮助主题。选择 *Help>Help Contents*，然后在 Search 选项卡中输入“Projects Window C or C++”（C 或 C++项目窗口）。在列表中找到第一个出现的“Projects Window”匹配项。

8.2 Files 窗口视图

Files 窗口允许用户在基于目录的视图中查看所有项目内容，甚至包括 Projects 窗口中未显示的文件和文件夹。

图 8-2. Files 窗口——简单 C 代码项目

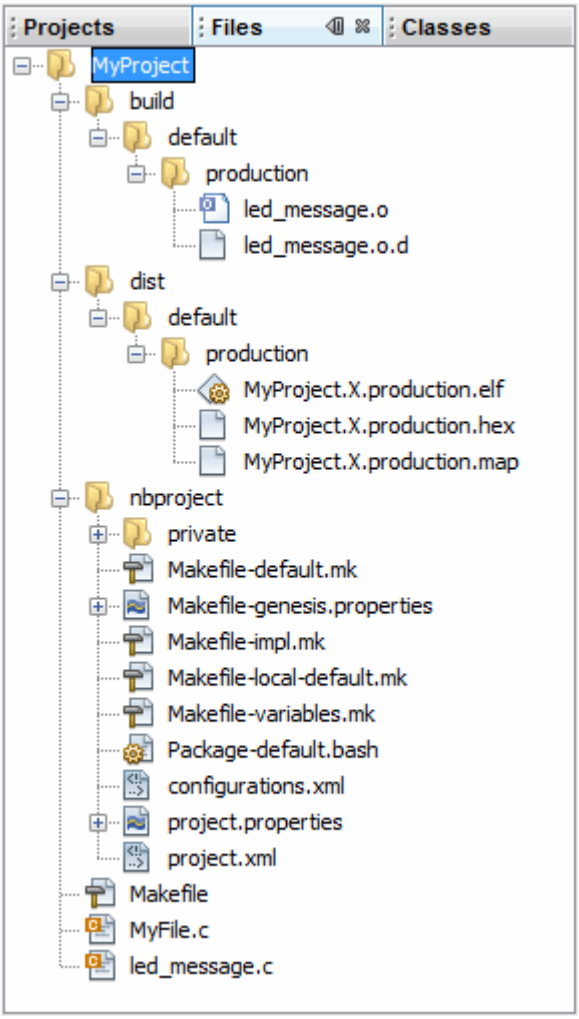


表 8-2. Files 窗口定义

文件夹	说明
MyProject	项目文件夹，其中包含 Makefile 文件和 C 代码或汇编应用程序文件。 Makefile 是项目的主 makefile。该文件在项目创建时生成，此后将不会触及它（不会重新生成它）。如果您熟悉 GNU Make，可以对该文件进行更改。但 MPLAB X IDE 提供了添加编译前步骤和编译后步骤的方法（Project Properties），可以使用它们而不修改 Makefile 本身。
build(1)	中间文件的文件夹。文件将根据项目配置、使用情况和位置而包含在子文件夹中。 编译文件包括： • 运行文件（.o） • 依赖性文件（.o.d） • HI-TECH®中间文件（.p1）

.....（续）

文件夹	说明
dist(1)	输出文件的文件夹。文件将根据项目配置、使用情况和位置而包含在子文件夹中。 分发文件包括： • 可执行文件（.hex） • ELF 或 COF 目标文件（.elf 或.cof） • 库文件（.a 或.lib）
nbproject	makefile 和元数据文件夹。包含以下文件： • 私有文件夹包含项目的用户和计算机特定设置 • 项目 makefile • 特定于配置的 makefile • project.xml，IDE 生成的元数据文件 • configurations.xml，元数据文件
default, MyConfig(2)	项目配置文件夹。如果开发人员未创建任何配置，所有代码都将位于 default 中。
production, debug(2)	生产和调试版本文件夹。
_ext(2)	外部项目文件的文件夹。如果引用了项目文件夹之外的某个文件，则它将列在此处。

1. 不需要将这些文件夹检入源代码控制。编译过程将创建它们。另请参见 5.23 使用版本控制系统控制源代码。
2. 上图中未列出这些项。

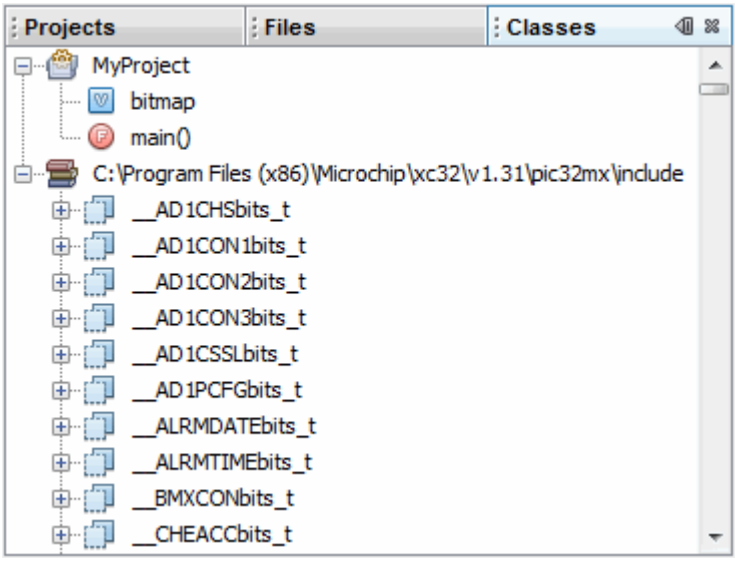
另请参见 NetBeans 帮助主题。选择 *Help>Help Contents*，然后在 Search 选项卡中输入“Files Window C and C++”（文件窗口 C 和 C++）。在列表中找到第一个出现的“文件窗口”匹配项。

8.3 Classes 窗口视图

对于 C 或 C++代码，可以在 Classes 窗口（*Window>Classes*（窗口>类））中查看所有类（函数）以及每个类的成员和字段。

另请参见 NetBeans 帮助主题。选择 *Help>Help Contents*，然后在 Search 选项卡中输入“Using the Classes Window”（使用 Classes 窗口）。

图 8-3. Classes 窗口——简单 C 代码项目



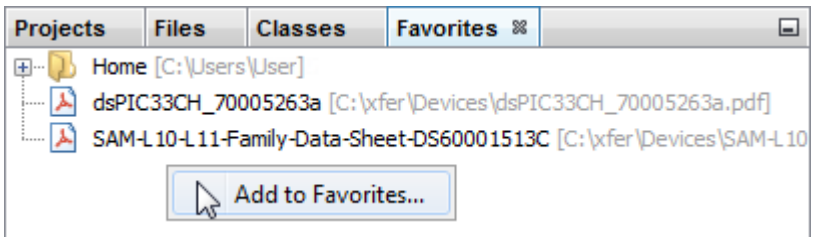
8.4 Favorites 窗口视图

通过 Favorites 窗口（*Window>Favorites*），您可以访问计算机或网络上的任何文件或文件夹，无论它是否处于一个项目中。

首次打开 Favorites 窗口时，它仅包含计算机的 home 目录。

- 要添加文件或文件夹，可以在 Favorites 窗口中单击右键并选择“Add to Favorites”（添加到收藏夹）。
- 要添加文件，可以在 Projects 窗口中右键单击文件名，并选择 *Tools>Add to Favorites*（工具>添加到收藏夹）。

图 8-4. 含示例的 Favorites 窗口



另请参见 NetBeans 帮助主题。选择 *Help>Help Contents*，在 Search 选项卡中输入“Favorites Window”（Favorites 窗口）。

8.5 路径、文件和文件夹名称限制

路径、文件和文件夹名称只能使用以下 ASCII 字符：

- A 至 Z
- a 至 z
- 0 至 9
- 下划线：_
- 短划线：-
- 句点：.

Windows 操作系统

Windows 操作系统的最大路径长度为 260 个字符。关于 Microsoft 给出的解释，可访问：

https://msdn.microsoft.com/en-us/library/windows/desktop/aa365247(v=vs.85).aspx

尽管 Windows 会尝试阻止您创建太长的路径，但是您可以通过某些应用程序、剪切和粘贴等方式创建长路径。如果项目源文件使用这类路径，则该项目不会正确编译。

此外，Windows 操作系统的命令行限制为 8191 个字符。有关详细信息，请参见：

“命令行过长（Windows 操作系统）”一节

跨平台操作系统

如果计划在不同的平台（Windows、Mac 或 Linux 操作系统）上使用 MPLAB X IDE，请注意以下问题：

- 在相对路径中使用斜杠“/”。反斜杠“\”仅在 Windows 操作系统平台上有效。例如：`#include headers/myheader.h`。
- Linux 操作系统区分大小写，因此 `genericTypeDefs.h` 与 `GenericTypeDefs.h` 不同。

8.6 路径、文件和文件夹项目建议

将现有文件或文件夹添加到项目中时，请考虑 8.5 路径、文件和文件夹名称限制中讨论的名称限制。具体来说，就是注意路径不要过长。尽管 Windows 操作系统的限制已知（路径长度为 260 个字符，命令行为 8191 个字符），但在其他系统上，路径过长可能会引发问题。如果您在编译时遇到问题，请考虑缩短路径长度或者将文件或文件夹移至项目文件夹中。

为了确保项目的可移植性达到最佳状态，请在导入文件或文件夹时使用相对路径。

相关信息

4.9.3 将现有文件添加到项目中

6.11 打包 MPLAB X IDE 项目

8.5 路径、文件和文件夹名称限制

8.7 查看用户配置数据

MPLAB X IDE 用户目录（userdir）存储用户配置数据（例如窗口布局、编辑器设置、菜单和工具栏定制项）和各种模块设置（例如已知编译器的列表）。

当您安装插件（*Tools>Plugins*）时，插件信息（代码）会被存储在 userdir 中，而不是 MPLAB X 安装目录中。但是，可以在 *Tools>Plugins>Settings>Advanced*（工具>插件>设置>高级）下，通过从“Plugin Install Location”（插件安装位置）下拉菜单中选择“Force install into shared directories”（强制安装到共享目录）来更改此位置。

要确定用户目录所在的位置，可以选择 *Help>About*（帮助>关于），并查看 **Userdir** 旁的路径。该信息由 MPLAB X IDE 创建和管理。

8.8 导入 MPLAB IDE v8 项目——相对路径

对于位于以下位置的 MPLAB IDE v8 项目：

```
C:\MyProjects\mplab8project
```

导入 MPLAB X IDE 中后，将为：

```
C:\MyProjects\mplab8project\mplab8project.X
```

默认情况下，MPLAB X IDE 导入的项目将放置在 MPLAB IDE v8 项目文件夹下，以保持这两个项目的可维护性。但是，您可以将 MPLAB X IDE 项目文件夹放置在任意位置。此外，项目的名称将设置为 `mplab8project.X`，但如果愿意的话，您可以重命名它。存在一项约定，即 MPLAB X IDE 项目的名称以 `.X` 结尾，但这只是一项约定。

新项目不会将源文件复制到其文件夹中，而是引用 v8 文件夹中的文件位置。要创建一个独立的 MPLAB X IDE 项目，需要创建一个新项目，并将 MPLAB IDE v8 源文件复制到其中。

有关详细信息，请参见 5.3 导入现有 MPLAB IDE v8 项目。

8.9 移动、复制或重命名项目

在创建项目之后，您可能发现需要进行更改。使用上下文（右键单击）项目菜单上的命令，可以在 MPLAB X IDE 内移动、复制或重命名您的项目。此外，还有一个选项用于删除项目。有关详细信息，请参见 12.15 Projects 窗口下的“项目菜单”。

您也可以使用外部工具；项目文件结构并不要求您使用 MPLAB X IDE。

8.10 删除项目

要在 MPLAB X IDE 中删除项目：

1. 在 Projects 窗口中，右键单击项目名称，并选择“Delete”。
2. 在“Delete Project”（删除项目）对话框中，您可以删除：
   2.1. 计算机中的项目文件——某些源文件将保留。
   2.2. 项目文件及其所有源文件（通过复选框）。

项目将被删除，因而 MPLAB X IDE 中将不能再看到它。

注： 在项目关闭时，MPLAB X IDE 并不会释放分配给项目的所有资源。MPLAB X IDE 基于 Java 运行，所以不会立即释放资源。在 Java 下，最好让 JRE 决定何时在软件代码内调用垃圾回收（Garbage Collection，GC）。但是，您可以通过使能存储器工具栏并单击它来强制执行 GC。

9. 疑难解答

本节旨在帮助您对在使用 MPLAB X IDE 时遇到的任何问题、错误或疑问进行疑难解答。如果这些信息未能帮助到您，请参见“支持”来了解联系 Microchip Technology 的方法。

9.1 USB 驱动程序安装问题

要安装正确的 USB 驱动程序，请参见 2.3 安装 USB 设备驱动程序（对于硬件工具）。

要解决出现的错误，请参见 2.3.3.5 工具通信问题。

9.2 操作系统问题

不同的操作系统在路径、文件和文件夹命名方面存在不同的问题。更多信息，请参见 8.5 路径、文件和文件夹名称限制。

9.3 NetBeans 平台问题

NetBeans 平台在用于 MPLAB X IDE 的平台版本方面可能存在问题。关于更多帮助，请访问以下 NetBeans 网站：

http://www.netbeans.org

http://netbeans.org/community/releases/index.html

以下问题在 MPLAB X IDE 中产生的影响已知。

- 前景颜色选为“Inherited”会导致类别文本在编辑器中不可见
- 没有可见的调试指示
- 无法解析标识符
- 在快速单步执行代码时取消选中 Watches 窗口中的 SFR
- 9.3.5 导航子菜单时菜单有时会消失

9.3.1 前景颜色选为“Inherited”会导致类别文本在编辑器中不可见

当 Field（字段）的前景颜色为“Inherited”（继承）时，弹出完成窗口（工具提示）中的文本将不可见。弹出完成窗口将使未选中的条目在白色背景上显示为白色文本。每次向下滚动列表时，会有一个条目在蓝色背景上显示为白色文本。其他情况下，该功能似乎都可以正常工作。

该问题被记录为有关字体和颜色的 Netbeans Bugzilla 问题：

https://netbeans.org/bugzilla/show_bug.cgi?id=249766

Options->Fonts & Colors（选项->字体和颜色）。为任何[Syntax]类别选择“Inherited”选项作为前景颜色都会导致文本在编辑器中不可见。查看首选项 xml 文件“org-netbeans-modules-editor-settings-CustomFontsColors-tokenColorings.xml”时，可明确一点：没有链接到受影响类别（例如“Field”或“Identifier”（标识符））的有效前景颜色属性。

变通方法：

1. 选择 *Tools>Options*，**Fonts & Colors** 按钮，**Syntax** 选项卡。
2. 对于“Language”，选择“C”。
3. 在“Category”下，单击“Field”。
4. 在左侧，可看到“Foreground”（前景）被选择为“Inherited”。从列表中选择一种颜色。
5. 单击 **OK**。

9.3.2 没有可见的调试指示

尽管看似在进行调试，但没有 Pause 箭头或行高亮显示，并且断点似乎已损坏（装订线中的撕裂图标）。这可能是由于文件路径命名不正确所导致。请参见 8.5 路径、文件和文件夹名称限制。

9.3.3 无法解析标识符

对于嵌套在无名称结构体中的结构体，实时解析器显示未解析的标识符。这是由于 NetBeans 本机 CND 不支持匿名字段声明所导致。变通方法是为字段指定一个标识符。缺点是必须在代码中任何使用该字段的位置指定其名称。

例如：

```
struct dados {
    unsigned char signature1;
    unsigned char signature2;
};
typedef union {
    unsigned char data [2];
    struct dados;          // unresolved ident.
} EEPROM_DATA;

void main(void) {
    EEPROM_DATA edata;
    edata.data [0] = 0xAA;     // this is ok
    edata.signature2 = 0xDD;   // unresolved ident.
}
```

另请参见：

https://netbeans.org/bugzilla/show_bug.cgi?id=256329

9.3.4 在快速单步执行代码时取消选中 Watches 窗口中的 SFR

当您单步执行代码时，光标会在源代码中更新，以便从 Watches 窗口中删除焦点。随后，Netbeans 会重新绘制整个 Watches 窗口，这将删除所有高亮显示的内容。

9.3.5 导航子菜单时菜单有时会消失

对于 Windows 操作系统，在选定 MPLAB X IDE 菜单后选择某个菜单项的子菜单时，该菜单有时会消失。可能需要重新启动 MPLAB X IDE 才能修复该问题。

9.4 MPLAB X IDE 问题

下面列出了 MPLAB X IDE 的问题。如果这里未列出您所遇到的问题，请参见与您项目的调试和编译器工具相关的文档。

- 导入 MPLAB IDE v8 项目——设置
- 导入 MPLAB IDE v8 项目——修改的链接描述文件
- MPLAB C18 C 编译器和结构
- 项目为空/项目文件呈灰显状态
- 编辑器中#error 语句后的代码为灰色
- Internet 连接不够稳定，无法下载最新的工具链

9.4.1 导入 MPLAB IDE v8 项目——设置

在 MPLAB IDE v8 的工作区中保存的设置（例如工具设置）将不会转移到新的 MPLAB X IDE 项目中。关于工作区中所存储内容的信息，请参见 MPLAB IDE v8 帮助（*MPLAB IDE Reference>Operational Reference>Saved Information*）。

9.4.2 导入 MPLAB IDE v8 项目——修改的链接描述文件

如果您修改了 MPLAB IDE v8 项目链接描述文件，从而使它包含某个目标文件，则在导入到 MPLAB X IDE 时，由于 MPLAB IDE v8 和 MPLAB X IDE 的编译路径不同，链接器将无法找到该文件。

所以，可能会看到类似以下错误：

```
<install path>ld.exe: cannot find file.o
```

因为在 MPLAB IDE v8 中，与编译有关的所有文件都处于一个目录中，而在 MPLAB X IDE 中编译文件则处于不同子目录中。

您可以编辑您的链接描述文件，通过使用通配符来使之适用于 MPLAB X IDE。例如，将：

```
/* Global-namespace object initialization - MPLAB v8*/
.init   :
{
KEEP (crti.o(.init))
:
} >kseg0_program_mem
```

更改为：

```
/* Global-namespace object initialization - MPLAB X*/
.init   :
{
KEEP (*crti.o(.init))
:
} >kseg0_program_mem
```

或者，您也可以使用 address 属性，使您可以在 C 代码中设置函数的地址。

```
int __attribute__((address(0x9D001000))) myfunction (void) {}
```

这使您可以将函数放置在某个地址处，而无需修改默认的链接描述文件。

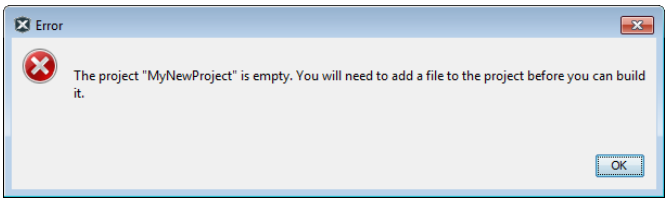
9.4.3 MPLAB C18 C 编译器和结构体

RAM 存储器中的结构体的指针值将无法在 Watches 窗口中正确显示。这是因为编译器未存储完整的调试信息。变通方法是改用 MPLAB XC8 C 编译器。

9.4.4 项目为空/项目文件呈灰显状态

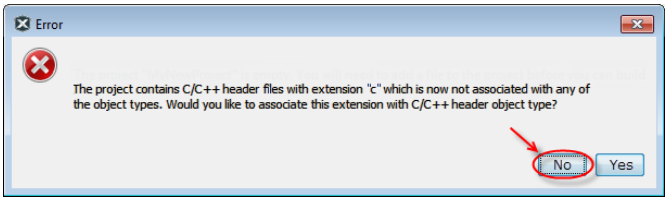
如果您装入了损坏的项目并错误地回答了一个对话框问题，则 Projects 窗口可能会呈灰显状态，并且 MPLAB X IDE 会显示由于没有源文件而无法编译该项目。

图 9-1. 项目为空错误



将损坏的项目装入 MPLAB X IDE 时，会发生这种问题。将出现一个对话框，询问 C 源文件是否应与头文件关联。此时您可以通过单击“No”（默认）来更正该问题。

图 9-2. 新的头文件扩展错误



但是，如果您单击“Yes”，则源文件将被视为头文件，项目为空。要修复项目，可以使用以下任一方法：

- 退出 MPLAB X IDE。
- 转至 MPLAB X IDE 用户目录（在 *Help>About* 对话框中指定），然后删除您的 MPLAB X IDE 版本的目录。例如，从以下位置删除文件夹 v3.05：`C:\Users\UserName\AppData\Roaming\mplab_ide\dev\v3.05`
- 再次启动 MPLAB X IDE。

或者

- 转至 *Tools>Options*，Embedded 按钮，**Other** 选项卡。
- 从头文件关联中删除“c”。
- 将“c”添加到源文件关联中。
- 单击 **OK**。

9.4.5 编辑器中#error 语句后的代码为灰色

请参见 7.7 C/C++代码错误伪指令。

9.4.6 Internet 连接不够稳定，无法下载最新的工具链

由于确定您的 Internet 连接不稳定，因此以下链接中涵盖的功能不可用。

4.1.8.2 下载最新 MPLAB XC 编译器

9.5 错误

在 IDE 中，错误可能具有多种形式，通常通过窗口中的图标或 Output 窗口中的消息显示。将鼠标悬停在图标上会弹出可能对问题进行说明的文本。对于文本消息，请参见联机帮助来搜索错误。

下面列出了 IDE 或编辑器的一些常见错误和解决方案：

9.5.1 命令行太长（Windows 操作系统）

当运行项目的 makefile 时，会将所执行的每个命令传递给本地的本机 shell。对于 Windows 操作系统，存在 8191 个字符的限制。因此，对于需要链接数百个 C 文件的项目，可能会超出该限制并显示该错误。

Linux OS 和 macOS 通常可采用非常长的命令行。如果希望知道您的系统能接受的命令行长度，可在命令行中输入 `getconf ARG_MAX`。

响应文件变通方法

如果使用的是 MPLAB XC8 或 MPLAB XC32 C 编译器 v1.01 及更高版本，或者 MPLAB XC16 C 编译器 v1.22 及更高版本，可以在调用链接器时使用响应文件来避开该问题。

- 对于 MPLAB XC8，在 Project Properties*窗口中，在“Categories”中单击“XC8 linker”（XC8 链接器）。在“Option categories”（选项类别）下选择“Additional Options”（其他选项）。选中“Use response file to link”（使用响应文件进行链接）旁的复选框。
- 对于 MPLAB XC16，在 Project Properties*窗口中，在“Categories”中单击“xc16-ld”。在“Option categories”下选择“General”。选中“Use response file to link”旁的复选框。
- 对于 MPLAB XC32，在 Project Properties*窗口中，在“Categories”中单击“xc32-ld”。在“Option categories”下选择“General”。选中“Use response file to link”旁的复选框。

*要打开 Project Properties 窗口，请参见 4.3 打开 Project Properties。

库变通方法

对于所有其他编译器，可以创建一个 MPLAB X IDE 库项目，并将一些源文件从主项目移至库项目。然后将库项目添加到主项目中。

要创建库项目，请选择 *File>New Project*，单击“Microchip Embedded”类别，然后选择“Library project”作为项目。

要将库添加到主项目中，请打开 Project Properties 窗口（*File>Project Properties*），在“Categories”下单击“Libraries”，然后单击 **Add Library Project** 按钮。

归档复选框变通方法

对于 MPLAB XC16 和 MPLAB XC32，现提供了一个复选框，当您将多组长文件归档到库中而可能导致链接行超出字符限制时，可以使用该复选框。可使用“Break into multiple lines”（拆分为多行）选项告知编译器将归档行拆分为更小的行，以避免违反该限制。

该选项位于以下位置：

Project Properties 窗口，“xc16-ar”或“xc32-ar”类别，“General”选项类别，“Break into multiple lines”复选框。

9.5.2 无法为 MinGW 堆保留空间

MPLAB X IDE 在其 make 过程中会使用 MinGW。在 Windows 操作系统中，可能存在虚拟存储器分配问题。

Windows 7 或 8

要更改 Virtual Memory（虚拟存储器）设置，请单击*“开始”>“控制面板”*。然后单击“系统”，并选择*“安全”>“系统”>“高级系统设置”或“系统保护”*。在**高级**选项卡上，单击**性能设置**按钮。在该对话框中单击**高级**选项卡，然后单击**更改**按钮。按如下所述输入定制大小值：

- 初始大小（MB）= 当前已分配（显示在底部）
- 最大大小（MB）= 建议（显示在底部）

单击**设置**，单击 **OK**，并重新启动您的个人计算机。

Windows 10

请参见：

https://www.geeksinphoenix.com/blog/post/2016/05/10/how-to-manage-windows-10-virtual-memory.aspx

9.5.3 目标路径太长/文件名或扩展名太长（Windows 操作系统）

Windows 操作系统的最大路径长度为 260 个字符。有关详细信息，请参见 8.5 路径、文件和文件夹名称限制。

9.5.4 项目 ProjectName 为空

请参见 9.4.4 项目为空/项目文件呈灰显状态。

9.5.5 对于 8 位代码，＃asm 和#endasm 会导致编辑器窗口中出现红色感叹号

`#asm` 和`#endasm` 语句是用于在 MPLAB XC8 C 程序中声明嵌入汇编代码的有效结构。但是在 IDE 编辑器中，C 预解析器不会将它们视为有效伪指令，并且将在这些语句旁边显示一个红色感叹号。不过，该代码仍可工作。

如果要消除编辑器中的红色感叹号，可以使用 `asm()` 语句。对于所有 MPLAB XC C 编译器，这都是使用嵌入汇编的首选方法。

示例：

```
// Inline Code that causes error
    #asm
        movlw 0x20;
        nop;
        nop;
    #endasm
// Workaround for inline assembly - that will not cause error
// (Multi line asm code)
    asm("\
    movlw 0x20;\
```

```
    nop;\
    nop;");
// Workaround for inline assembly - that will not cause Error

// (Single line asm code)
    asm("movlw 0x20; nop; nop");
```

9.5.6 对于 16 位代码，MPLAB XC16 packed 属性语句会导致编辑器窗口中出现红色感叹号

以下语句将被编译器解析器标记为错误，但会进行编译：

```
unsigned long long Bits8 : 8 __attribute__ ((packed));
```

解析器限定符规则支持任何顺序，但期望所有限定符均在标识符之前指定。因此，使用以下语句之一不会产生错误：

```
unsigned long long attribute ((packed)) Bits8 : 8;
```

或者

```
attribute ((packed)) unsigned long long Bits8 : 8;
```

9.5.7 Hexmate 冲突报告地址错误

当 Hexmate 生成冲突报告时，十六进制文件中的冲突的地址有时与存储器窗口中显示的地址（数据在存储器中的位置）不同。例如：

表 9-1. 十六进制地址与存储器地址

器件系列	十六进制文件中的地址	存储器中的地址
基础	0x100	0x80*
中档	0x100	0x80*
PIC18 MCU	0x100	0x100
16 位器件	0x100	0x80*
32 位器件	0x100	0x100
*对于某些器件，冲突报告中显示的地址是存储器中数据的地址的两倍大。		

9.5.8 编程错误

如果在器件被代码保护的情况下尝试对其进行编程，则将收到关于地址区域的编程失败错误消息。

“Erase Device Memory Main Project”（擦除器件存储器主项目）  功能可以帮助确定这是否为编程问题。该功能以可选图标形式提供，您可以将其添加到工具栏（*View>Toolbars>Customize*（视图>工具栏>定制））。也可以使用 MPLAB IPE。

1. 确保代码中的代码保护已关闭。
2. 擦除器件存储器，该操作将擦除配置位设置。
3. 重新编程器件

如果编程成功，则代码保护便是问题所在。

9.5.9 不支持脱机编程

配置（config）字过多的器件当前无法支持脱机编程（Programmer to Go，PTG）。PTG 希望一次发送所有配置字，但对于具有大量配置字的器件而言，MPLAB X IDE 一次发送一个配置字。

9.5.10 鼠标悬停将在编辑器中显示“Not Recognized”

使用鼠标悬停在编辑器中显示值时，请确保：

1. 处于调试模式（*Debug>Debug Project*）。除非正在调试，否则无法看到符号值。
2. 当前未将鼠标悬停在宏上。这种方式无法查看宏的值。右键单击宏并选择 *Navigate>View Macro Expansion*（导航>查看宏扩展），使用 Macro Expansion 窗口。

9.5.11 -10000 以下的错误代码

-10000 以下的错误代码表示 Microsoft 错误代码。要确定 Microsoft 错误代码，请先使数字为正（-10121 变为 10121），然后减去 10000（10121-10000 = 121）。Microsoft 错误代码请参见：

https://msdn.microsoft.com/en-us/library/windows/desktop/ms681382(v=vs.85).aspx

9.6 论坛

如果未在此处看到您的问题，请访问我们的论坛：

http://www.microchip.com/forums/f238.aspx

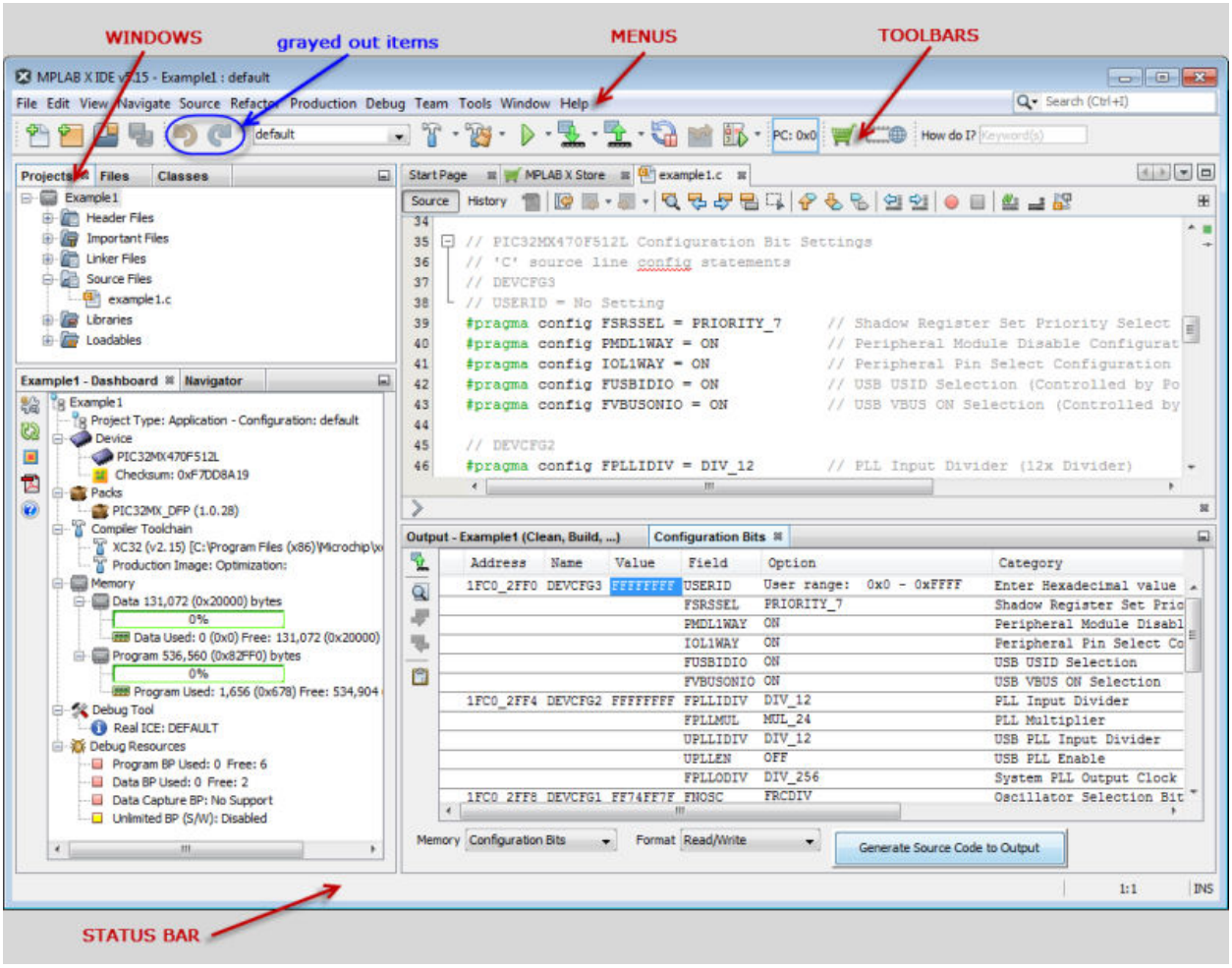
此处可以找到关于问题的讨论和最新发布的解决方法。

10. 桌面参考

MPLAB® X IDE 桌面是一个可调整大小的窗口，它独立于其他桌面项而工作。桌面包含菜单、工具栏、状态栏和选项卡式窗口。本章将介绍菜单、工具栏和状态栏。窗口和对话框在其各自的章节中介绍。

注： 当 NetBeans 帮助主题谈及某个工作区时，它指的是桌面，而不是 MPLAB IDE v8（及其更早版本）的工作区。

图 10-1. MPLAB® X IDE 桌面



10.1 菜单

许多 MPLAB® X IDE 功能可通过位于桌面顶部的菜单栏以菜单项形式进行访问。跟随省略号（...）的菜单项将打开一个对话框。关于对话框的更多信息，请参见 12. MPLAB X IDE 窗口和对话框。

菜单项旁会列出菜单项的快捷键。

示例： “New File”的快捷键为 Control-N（CTRL+N）。*Help>Keyboard Shortcuts Card*（帮助>键盘快捷方式卡）下提供了关于快捷键的更多信息。

菜单项可能由于各种原因而灰显。请参见 10.4 灰显或缺失的项和按钮。

通过在窗口中单击右键可显示更多的上下文菜单。关于这些菜单的更多信息，请参见 12.15 Projects 窗口。

下面列出了可用的菜单。

注： 要查看所有可用的菜单和菜单项，请参见 Start Page，My MPLAB IDE>Extend MPLAB>Selecting Simple or Full-Featured Menus（我的 MPLAB IDE>扩展 MPLAB>选择简单或全功能菜单）。不是所有这些菜单项都适用于嵌入式开发。

10.1.1 File 菜单

本节列出了 File 菜单中的菜单项。

关于其中一些菜单项的键盘快捷方式，请参见 *Help>Keyboard Shortcuts Card*。

表 10-1. File 菜单选项

命令	操作
New Project	使用 New Project 向导创建一个新项目。
New File	使用 New File 向导创建一个新文件。
Open Project	打开现有项目。
Open Recent Project	显示最近打开的所有项目的列表，以供选择。
Import	导入以下对象之一： • Hex/ELF...（预编译）文件——使用其他工具编译。 • MPLAB IDE v8 项目——启动 Import Legacy Project 向导。 • 其他嵌入式——从其他嵌入式平台导入。 • 从 ZIP——从压缩项目导入，即，解压缩并导入。
Close Project	关闭当前项目。
Close Other Projects（关闭其他项目）	关闭除主项目外的所有其他项目。
Close All Projects（关闭所有项目）	关闭所有已打开的项目。
Open File	打开现有文件。
Open Recent File（打开最近的文件）	显示最近打开的所有文件的列表，以供选择。
Project Group	将当前项目与某个组相关联。
Project Properties	打开 Project Properties 窗口。
Save（保存）	保存当前文件。
Save As	使用新的路径和/或名称保存当前文件。
Save All（全部保存）	保存所有已打开的文件。如果选择了“Compile on Save”（保存时编译）功能，则这也将编译项目文件。
Page Setup（页设置）	设置页以进行打印。
Print（打印）	显示当前文件的打印预览，并允许进行打印。
Print to HTML（打印为 HTML）	将当前文件打印为 HTML 格式的新文件。
Exit（退出）	退出 MPLAB® X IDE。

10.1.2 Edit 菜单

本节列出了 Edit 菜单中的菜单项。

关于其中一些菜单项的键盘快捷方式，请参见 *Help>Keyboard Shortcuts Card*。

表 10-2. Edit 菜单选项

命令	操作
Undo	回退（一次一个）除 Save 之外的一系列编辑器操作。
Redo	回退（一次一个）一系列 Undo 命令。
Cut	删除当前选定内容并将其放置到剪贴板上。
Copy	将当前选定内容复制到剪贴板上。
Paste	将剪贴板的内容粘贴到插入点。
Paste Formatted（粘贴格式化内容）	将格式化的剪贴板内容粘贴到插入点。
Paste from History（从历史记录粘贴）	从复制历史记录粘贴。
Delete	删除当前选定内容。
Select All（全选）	选择当前文档或窗口中的所有内容。
Select Identifier（选择标识符）	选择距离光标最近的词。
Find Selection（查找选定内容）	查找当前选定内容的实例。
Find Next（查找下一处）	查找已找到文本的下一个实例。
Find Previous（查找上一处）	查找已找到文本的上一个实例。
Find	查找文本字符串。
Replace	查找文本的字符串，并使用指定字符串替换它。
Find Usages	查找选定代码的使用实例和子类型。
Find in Projects	在项目内查找指定的文本、对象名称和对象类型。
Replace in Projects（在项目中替换）	在项目内替换文本、对象名称和对象类型。
Start Macro Recording（开始宏录制）	开始录制按键操作。
Stop Macro Recording	停止录制按键操作。 要管理宏，请选择 *Tools>Options*，**Editor** 按钮，**Macros** 选项卡。

10.1.3 View 菜单

本节列出了 View 菜单中的菜单项。

关于其中一些菜单项的键盘快捷方式，请参见 *Help>Keyboard Shortcuts Card*。

表 10-3. View 菜单选项

命令	操作
Editors（编辑器）	选择一个可用编辑器来查看您的文件。 • History——显示该文件的历史编辑记录。 • Source（源代码）——显示该文件中的源代码。
Split（拆分）	将当前编辑器窗口及其内容在垂直或水平方向上拆分为两个窗口。
Code Folds>Collapse Fold	如果插入点处于可折叠文本部分，则将这些代码行折叠为一行。
Code Folds>Expand Fold	如果源代码编辑器中的当前选定行代表几个已折叠的行，则展开折叠内容来显示所有行。
Code Folds>Collapse All	在源代码编辑器中折叠所有可折叠文本部分。

.....（续）

命令	操作
Code Folds>Expand All	在源代码编辑器中展开所有可折叠文本部分。
Code Folds>Collapse Fold Tree（代码折叠>折叠折叠树）	将所有折叠部分折叠到一个折叠嵌套组中。
Code Folds>Expand Fold Tree（代码折叠>展开折叠树）	展开一个折叠嵌套组中的所有折叠部分。
Web Browser（Web 浏览器）	打开默认的 Web 浏览器并转至 NetBeans 主页。
IDE Log	在 Output 窗口的选项卡中打开日志文件。
Toolbars>File（工具栏>文件）等	显示相关的工具栏。
Toolbars>Small Toolbar Icons（工具栏>工具栏小图标）	使用工具栏上的小图标。
Toolbars>Reset Toolbars（工具栏>复位工具栏）	复位为默认的工具栏设置。
Toolbars>Customize	定制现有工具栏上的项，并允许创建一个新的项。
Show Editor Toolbar（显示编辑器工具栏）	在 File 选项卡上显示编辑器工具栏。
Show Line Numbers	在左边缘显示行号。
Show Non-printable Characters（显示不可打印的字符）	即使字符不可打印（非 ASCII），也进行显示。
Show Breadcrumbs（显示痕迹导航）	在编辑器窗口的底部显示痕迹导航或项目文件中文件的路径。
Show Indent Guide Lines（显示缩进参考线）	显示表示缩进的浅色线条。
Show Diff Sidebar（显示 Diff 侧边栏）	显示 DIFF 侧边栏。
Show Versioning Labels（显示版本控制标签）	显示版本标签。
Synchronize Editor with Views（将编辑器与视图同步）	将编辑器与打开的视图同步。
Show Only Editor（仅显示编辑器）	在整个 IDE 窗口中显示编辑器窗口。
Full Screen（全屏）	将窗口展开到屏幕的完整长度和宽度。

10.1.4 Navigate 菜单

本节列出了 Navigate 菜单中的菜单项。

关于其中一些菜单项的键盘快捷方式，请参见 *Help>Keyboard Shortcuts Card*。

表 10-4. Navigate 菜单选项

命令	操作
Go to File	查找并打开特定文件。
Go to Type	查找并打开特定的类或接口。
Go to Symbol	查找指定的符号名称。
Go to Previous Window（转至上一个窗口）	将焦点转至上一个选定的窗口。

.....（续）

命令	操作
Go to Source（转至源文件）	显示包含选定类的定义的源文件。
Go to Declaration	跳转至光标下的项的声明。
Go to Super Implementation（转至超类实现）	跳转至光标下的项的超类实现。
Last Edit Location（上一个编辑位置）	将编辑器滚动到上一次发生编辑的位置。
Back	后退。
Forward（前进）	前进。
Go to Line（转至行）	跳转至指定的行。
Toggle Bookmark（翻转书签）	在代码行上设置书签。
Bookmark History Popup Next（书签历史弹出下一个）	转至 Bookmark（书签）窗口中的下一个书签（*Window>IDE Tools>Bookmarks*（窗口>IDE 工具>书签））。
Bookmark History Popup Previous（书签历史弹出上一个）	转至 Bookmark 窗口中的上一个书签（*Window>IDE Tools>Bookmarks*）。
Next Error（下一个错误）	将源代码编辑器滚动到包含下一个编译错误的行。
Previous Error（上一个错误）	将源代码编辑器滚动到包含上一个编译错误的行。
Select in Projects	打开 Projects 窗口，并选择其中的当前文档。
Select in Files（在文件中选择）	打开 Files 窗口，并选择其中的当前文档。
Select in Classes（在类中选择）	打开 Classes 窗口，并选择其中的当前文档。
Select in Favorites（在收藏夹中选择）	打开 Favorites 窗口，并选择其中的当前文档。

10.1.5 Source 菜单

本节列出了 Source 菜单中的菜单项。

关于其中一些菜单项的键盘快捷方式，请参见 *Help>Keyboard Shortcuts Card*。

表 10-5. Source 菜单选项

命令	操作
Format	格式化选定代码，如果未选择任何内容，则格式化整个文件。
Remove Trailing Spaces（删除尾随空格）	删除行尾空格。
Shift Left（左移）	将选定的一行或多行向左移动一个制表符距离。
Shift Right（右移）	将选定的一行或多行向右移动一个制表符距离。
Move Up	将选定的一行或多行上移一行。
Move Down	将选定的一行或多行下移一行。
Move Code Element Up（上移代码元素）	将选定的代码元素上移一行。
Move Code Element Down（下移代码元素）	将选定的代码元素下移一行。
Duplicate Up（向上复制）	将选定的一行或多行向上复制一行。

.....（续）

命令	操作
Duplicate Down（向下复制）	将选定的一行或多行向下复制一行。
Toggle Comment（翻转注释）	翻转当前行或选定行的注释。
Complete Code（补全代码）	显示代码补全框。
Insert Code（插入代码）	弹出可用于生成构造函数、getter 和 setter 等常见结构的上下文菜单。
Fix Code（修复代码）	显示编辑器提示。当有提示时，IDE 会在显示灯泡时通知您。
Show Method Parameters（显示方法参数）	选择下一个参数。只有选定（高亮显示）一个参数时，该快捷方式才会起作用。
Show Documentation（显示文档）	显示光标下的项的文档。
Insert Next Matching Word（插入下一个匹配的词）	当您输入词的起始字符时，生成下一个已在代码中其他位置使用的词。
Insert Previous Matching Word（插入上一个匹配的词）	当您输入词的起始字符时，生成上一个已在代码中其他位置使用的词。
Inspect（检查）	对选定的文件、包或项目运行指定检查。
Scan for External Changes（扫描外部更改）	扫描文件是否在 MPLAB® X IDE 之外进行了更改。

10.1.6 Refactor 菜单

本节列出了 Refactor 菜单中的菜单项。可看到的项取决于要重构的对象（变量和函数等）的类型。更多信息，请参见 7.6 C 代码重构。

关于其中一些菜单项的键盘快捷方式，请参见 *Help>Keyboard Shortcuts Card*。

表 10-6. Refactor 菜单选项

命令	操作
Rename	可用于将变量或函数的名称更改为更有意义的形式。此外，它还会更新项目中的所有源代码，使之按新名称引用元素。
Move（移动）	将某个类移至另一个包或另一个类中。此外，项目中的所有源代码也会更新为在新的位置引用该类。
Copy	将某个类复制到相同或不同的包中。
Safely Delete（安全删除）	检查对某个代码元素的引用，然后如果没有任何其他代码引用它，则自动删除该元素。
Change Function Parameter（更改函数参数）	更改选定函数的参数数量和名称。
Encapsulate Fields（封装字段）	**仅限 C++：**单击 C++源文件可查看该选项。 为字段自动生成 getter 方法和 setter 方法，并可选择更新所有引用该字段代码，以使用 getter 和 setter 方法访问该字段。 另请参见： http://wiki.netbeans.org/Refactoring

10.1.7 Production 菜单

本节列出了 Production（生产）菜单（之前为 Run 菜单）中的菜单项。

注： Run 操作已从该菜单中删除，但仍在工具栏上以图标形式提供：  。

关于其中一些菜单项的键盘快捷方式，请参见 *Help>Keyboard Shortcuts Card*。

表 10-7. Production 菜单选项

命令	操作
Build Project	编译项目中的所有文件。
Clean and Build Project	删除（清除）先前生成的项目文件，然后重新编译项目中的文件。
Batch Build Project（批量编译项目）	编译项目的多个配置（仅嵌入式项目可用）。
Set Project Configuration	选择项目配置——Default。
Set Main Project（设置主项目）	通过从已打开项目的列表中选择来设置主项目。
Configuration Bits	打开 Configuration Bits 窗口。
Check File（检查文件）	根据标准来检查文件（XML 相关）。
Validate File（验证文件）	根据标准来验证文件（XML 相关）。
Repeat Build/Run（重复编译/运行）	暂停之后再次运行。
Stop Build/Run（停止编译/运行）	结束运行。

10.1.8 Debug 菜单

以下是 Debug 菜单中的菜单项。

表 10-8. Debug 菜单选项

命令	操作
Debug Project	调试主项目或选定项目。
Discrete Debugger Operation	执行以下调试操作，一次执行一个步骤（分离式）： • Build for Debugging（带调试执行程序进行编译）。 • Program Device for Debugging（使用调试编译进行编程）。 • Launch Debugger。 • Live Connect Debug（实时连接调试）。 这对于在调试期间和使用入门工具包期间更改存储器窗口设置很有用。
Finish Debugger Session	结束调试会话。
Pause	暂停调试——使用“Continue”可继续。
Continue	“Pause”之后继续调试，直到达到下一个断点或程序末尾。
Step Over	执行程序的一个源代码行。如果该行是一个函数调用，则执行整个函数，然后停止。
Step Into	执行程序的一个源代码行。如果该行是一个函数调用，则程序执行到该函数的第一条语句，然后停止。
Step Out	执行程序的一个源代码行。结束当前函数的执行，在紧跟该函数调用的下一个源代码行停止。
Step Instruction	执行一条机器指令。 如果该指令是一个函数调用，则执行函数，并将控制返回给调用方。
Run to Cursor	运行当前项目，直到文件中的光标位置处停止程序执行。

.....（续）

命令	操作
Reset	将器件复位。
Set PC at Cursor（将 PC 设置在光标位置）	将程序计数器（PC）值设置为光标的行地址。
Focus Cursor at PC（将光标焦点设置在 PC 处）	将光标移动到当前 PC 地址处，并使该地址在窗口中居中。
Stack>Make Callee Current（堆栈>将被调用方设为当前调用）	将被调用的方法设为当前调用。仅当在 Call Stack 窗口中选定某个调用时可用。
Stack>Make Caller Current（堆栈>将调用方设为当前调用）	将产生调用的方法设为当前调用。仅当在 Call Stack 窗口中选定某个调用时可用。
Stack>Pop Topmost Call（堆栈>弹出顶部的调用）	弹出堆栈顶部的调用。
Stack>Pop To Current Stack Frame（堆栈>弹出并压入当前堆栈帧）	弹出当前帧并压入堆栈顶部。
Stack>Pop Last Debugger Call（堆栈>弹出最后一个调试器调用）	弹出调试器工具的最后一个调用并压入堆栈顶部。
Toggle Line Breakpoint	在程序中的光标位置处添加行断点或删除断点。
New Breakpoint	在指定行、异常或方法处设置新的断点。
New Watch	添加指定符号来进行观察。
New Run Time Watch（新建运行时观察）	添加指定符号来观察将在程序运行/执行时改变值的符号。
Disconnect from Debug Tool（与调试工具断开）	断开 MPLAB® X IDE 和调试工具之间的通信。 要重新连接，请选择 Run 或 Debug。
Run Debugger/Programmer Self Test（运行调试器/编程器自检）	执行调试工具自检。 对于支持自检的工具，请按照工具文档来设置硬件，然后运行该测试来确认操作是否正确。
Hardware Tool Emergency Boot Firmware Recovery（硬件工具紧急引导固件恢复）	运行该实用程序将硬件工具引导固件恢复为出厂状态。有关详细信息，请参见 MPLAB ICD 4 或 PICkit 4 帮助。

10.1.9 Team 菜单

本节列出了 Team 菜单中的菜单项。

关于其中一些菜单项的键盘快捷方式，请参见 *Help>Keyboard Shortcuts Card*。

表 10-9. Team 菜单选项

命令	操作
Shelve Changes（搁置更改）	暂时将工作目录中一些尚未提交的更改作为补丁文件搁置。
Git, Mercurial, Subversion	显示特定于每个版本管理系统的子菜单。关于子菜单选项的更多信息，请参见产品文档。
History	显示文件的历史记录或将文件还原为历史版本。
Find Task	在版本控制系统中查找问题。

.....（续）	
命令	操作
Report Task	向版本控制系统报告问题。
Create Build Job（创建编译作业）	使用版本控制系统创建编译。

10.1.10 Tools 菜单

本节列出了 Tools 菜单中的菜单项。

关于其中一些菜单项的键盘快捷方式，请参见 *Help>Keyboard Shortcuts Card*。

表 10-10. Tools 菜单选项

命令	操作
Embedded	如果已添加插件则可见——从子菜单中选择插件。
Licenses	管理编译器许可证。有关详细信息，请参见以下内容的 Documentation（文档）选项卡：https://www.microchip.com/mplab/compilers • Roam Network Licenses（检入/检出网络许可证）。 • Activate Workstation License（激活工作站许可证）。 • License Status（许可证状态）。 • Change Licensing Type（更改许可类型）。
Packs	打开 MPLAB 包管理器，为您的项目器件选择相应版本的器件包。
Templates	打开 Template Manager（模板管理器）。
DTDs and XML Schemas（DTD 和 XML 模式）	打开 DTD 和 XML Schemas Manager。
Plugins	打开 Plugins Manager。有关详细信息，请参见 NetBeans 帮助主题：http://wiki.netbeans.org/InstallingAPlugin
Plugins Download（插件下载）	打开选择对话框，您可以在其中： • 访问“Embedded Code Source”网站——下载一个插件，稍后可使用插件管理器的“Downloaded”选项卡进行安装。 • 转到 MPLAB X 插件管理器——直接从插件管理器中的“Available Plugins”列表安装插件。
Options	打开 Options 对话框。 对于 macOS：请使用 MPLAB X IDE>Preferences。 请参见 12.16 Tools>Options 窗口，Embedded。

10.1.11 Window 菜单

本节列出了 Window 菜单中的菜单项。

关于其中一些菜单项的键盘快捷方式，请参见 *Help>Keyboard Shortcuts Card*。

表 10-11. Window 菜单选项

命令	操作
Projects	打开 Projects 窗口。 请参见 8.1 Projects 窗口视图。

.....（续）

命令	操作
Files	打开 Files 窗口。 请参见 8.2 Files 窗口视图。
Classes	打开 Classes 窗口。 请参见 8.3 Classes 窗口视图。
Favorites	打开 Favorites 窗口。 请参见 8.4 Favorites 窗口视图。
Services	打开 Services 窗口。 关于该窗口的更多信息，请参见 NetBeans 帮助。
Dashboard	打开 Dashboard 窗口。 请参见 5.19 查看仪表板显示。
Navigator	打开 Navigator 窗口。 关于该窗口的更多信息，请参见 NetBeans 帮助。
Action Items	打开 Action Items 窗口。 请参见 12.1 Action Items 窗口。
Tasks	打开 Task List（任务列表）窗口。
Output	打开或将 Output 窗口移至最前。 请参见 12.13 Output 窗口。
Editor	打开一个空的编辑器窗口。 请参见 7.1 编辑器用法。
Debugging>Output	打开调试输出窗口。
Debugging>Variables	打开 Local Variables（局部变量）调试器窗口。 请参见 4.17 观察局部变量值变化。
Debugging>Watches	打开 Watches 调试器窗口。 请参见 4.16 观察符号值变化。
Debugging>Call Stack	打开 Call Stack 调试器窗口。 请参见 5.17 查看调用堆栈。
Debugging>Breakpoints	打开 Breakpoints 窗口。 请参见 4.14 使用断点控制程序执行。
Debugging>Sessions	打开 Sessions 窗口。
Debugging>Sources（调试>源代码）	打开 Sources 窗口。
Debugging>Disassembly（调试>反汇编）	打开 Disassembly 窗口。 请参见 12.7 Disassembly 窗口。
Debugging>PIC AppIO（调试>PIC AppIO）	打开 Application In/Out（应用程序输入/输出）窗口。 适用于 MPLAB® REAL ICE™在线仿真器。
Debugging>Trace（调试>跟踪）	打开 Trace 窗口。 适用于软件模拟器或 MPLAB REAL ICE 在线仿真器。
Debugging>Stopwatch	打开 Stopwatch 窗口。 请参见 5.15 使用跑表。

.....（续）

命令	操作
Debugging>PC Profiling（调试>PC 性能分析）	打开 PC Profiling 窗口。 适用于 MPLAB® REAL ICE™在线仿真器。
Debugging>Triggers（调试>触发）	打开 Trigger In（触发输入）窗口。 适用于 MPLAB® REAL ICE™在线仿真器。
Debugging>Debugger Console（调试>调试器控制台）	打开控制台窗口，以便在命令行上输入命令。
Web>Web Browser（Web>Web 浏览器）	打开默认的 Web 浏览器。
Web>CSS	打开 CSS Styles（CSS 样式）窗口。您可以在其中编辑 CSS 文件中的 HTML 元素和选择器的规则声明。
IDE Tools>Bookmarks	打开 Bookmarks 窗口。显示已打开项目的文件中的书签列表。
IDE Tools>Notifications（IDE 工具>通知）	打开 Notifications 窗口。显示当前 IDE 会话中发生的 IDE 错误和警告的列表。
IDE Tools>Terminal（IDE 工具>终端）	打开本地或远程主机的 Terminal 窗口。
IDE Tools>Processes（IDE 工具>进程）	打开 Processes 窗口。可以将调试工具连接到正在运行的进程。
PIC Memory Views>Memory（PIC 存储器视图>存储器）	打开指定的存储器窗口。显示的存储器取决于项目器件。请参见： 12.9 存储器窗口——8 位和 16 位器件 12.10 存储器窗口——32 位器件
Simulator>Stimulus（软件模拟器>激励）	打开 Simulator Stimulus（软件模拟器激励）窗口。 有关详细信息，请参见 MPLAB X 软件模拟器文档。
Simulator>Analyzer（软件模拟器>分析器）	打开 Simulator Analyzer（软件模拟器分析器）窗口。 有关详细信息，请参见 MPLAB X 软件模拟器文档。
Simulator>IOPin（软件模拟器>IO 引脚）	打开 Simulator IO Pin（软件模拟器 IO 引脚）窗口。 有关详细信息，请参见 MPLAB X 软件模拟器文档。
Simulator>Register Trace（软件模拟器>寄存器跟踪）	打开 Simulator Register Trace（软件模拟器寄存器跟踪）窗口。 有关详细信息，请参见 MPLAB X 软件模拟器文档。
Configure Window> Maximize	将活动窗口最大化为完整的 IDE 窗口大小。
Configure Window>Float（配置窗口>浮动）	打开带有一个选项卡的浮动 IDE 窗口。
Configure Window> Float Group（配置窗口>浮动组）	打开带有多个选项卡的浮动 IDE 窗口组。
Configure Window>Minimize（配置窗口>最小化）	将活动窗口最小化为标准大小。
Configure Window> Minimize Group（配置窗口>最小化组）	将窗口组最小化为标准大小。
Configure Window>Dock（配置窗口>停靠）	将浮动选项卡还原到主窗口。
Configure Window> Dock Group（配置窗口>停靠组）	将浮动选项卡组还原到主窗口。
Configure Window> Clone Document（配置窗口>克隆文档）	使用新选项卡打开同一文档。

.....（续）

命令	操作
Configure Window>Split Window（配置窗口>拆分窗口）	垂直或水平拆分活动文档。
Configure Window>New Document Tab Group（配置窗口>新建文档选项卡组）	将编辑器窗口拆分为两组选项卡，并将选定的选项卡归为新组。
Configure Window>Collapse Document Tab Group（配置窗口>折叠文档选项卡组）	将单独的选项卡组合并为一组。
Reset Window（复位窗口）	将窗口复位为其默认设置。
Close Window（关闭窗口）	关闭当前窗口中的当前选项卡。如果窗口没有任何选项卡，则关闭整个窗口。
Close All Documents（关闭所有文档）	关闭源代码编辑器中所有已打开的文档。
Close Other Documents（关闭其他文档）	关闭除活动文档之外的所有其他已打开文档。
Document Groups（文档组）	创建已命名文档组。
Documents（文档）	打开 Documents 对话框，在其中您可以保存并关闭已打开文档的组。

10.1.12 Help 菜单

本节列出了 Help 菜单中的菜单项。

关于其中一些菜单项的键盘快捷方式，请参见 *Help>Keyboard Shortcuts Card*。

表 10-12. Help 菜单选项

命令	操作
Help Contents	显示 JavaHelp 查看器，其中会列出所有已安装的帮助文件。
Tool Help Contents	显示单个 JavaHelp 查看器，其中会列出单个工具帮助文件。
Release Notes（版本说明）	打开适用于 MPLAB 所支持工具的版本说明的链接列表。其他 MPLAB X IDE 支持文档链接也将一并列出。
Online Docs and Support（在线文档和支持）	打开 NetBeans 支持网页。
Keyboard Shortcuts Card	显示键盘快捷方式文档。
MPLAB X Store	打开 MPLAB X Store 选项卡。
Start Page	打开或将 Start Page 选项卡移至所有其他已打开选项卡之前。
About	显示关于 MPLAB® X IDE 的窗口。

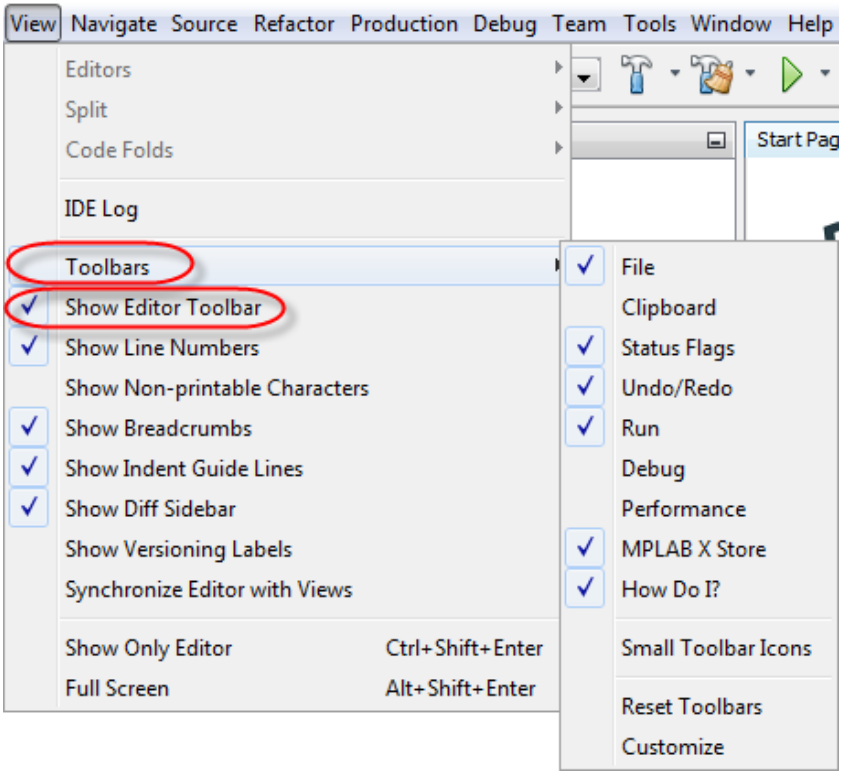
10.2 工具栏

MPLAB X IDE 会根据您正在使用的功能或工具来显示不同的工具栏。这些工具栏上的图标提供了日常任务的快捷方式。

以下内容在使用工具栏时十分有用：

- 工具栏功能
- 定制工具栏
- 灰显或缺失的项和按钮

图 10-2. View > Toolbars



10.2.1 文件工具栏

文件工具栏当前包含以下功能的按钮图标。File 菜单上也提供了这些功能。

图标	图标文本	功能
	New File	使用 New File 向导创建一个新文件。
	New Project	使用 New Project 向导创建一个新项目。
	Open Project	打开现有项目。
	Save All Files（保存所有文件）	保存所有已打开的文件。

相关信息

10.2.11 工具栏功能
10.2.12 定制工具栏

10.2.2 剪贴板工具栏

剪贴板工具栏当前包含以下功能的按钮图标。Edit 菜单上也提供了这些功能。

图标	图标文本	功能
	Cut	删除当前选定内容并将其放置到剪贴板上。

.....（续）

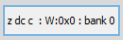
图标	图标文本	功能
	Copy	将当前选定内容复制到剪贴板上。
	Paste	将剪贴板的内容粘贴到插入点。

相关信息

10.2.11 工具栏功能
10.2.12 定制工具栏

10.2.3 状态标志工具栏

状态标志工具栏包含以下功能：

图片	图片文本	说明
	Program Counter（程序计数器）	程序计数器（PC）当前值。
	Status Flags	器件状态标志。关于状态位和/或寄存器的信息，请参见器件数据手册。

10.2.4 取消/重做工具栏

取消/重做工具栏当前包含以下功能的按钮图标。Edit 菜单上也提供了这些功能。

图标	图标名称	功能
	Undo	回退（一次一个）除 Save 之外的一系列编辑器操作。
	Redo	回退（一次一个）一系列 Undo 命令。

10.2.5 运行工具栏

运行工具栏当前包含以下功能的按钮图标。Run、Debug 和项目上下文菜单上也提供这些功能。

图标	图标文本	功能
	Set Project Configuration	选择项目配置。选择“default”或“Customize”。有关详细信息，请参见 Project Properties 窗口。
	Build Project	编译所有项目文件。
	>Build for Debugging	为调试而编译所有项目文件，包括调试执行程序。Discrete Debugger Operation 的第一步。
	Clean and Build Project	删除来自先前编译的文件，然后编译所有项目文件。
	>Clean and Build for Debugging	删除来自先前编译的文件，然后编译所有项目文件以进行调试，包括调试执行程序。Discrete Debugger Operation 的第一步。
	Run Project	编译，对目标编程，并运行选定项目。

.....（续）		
图标	图标文本	功能
	Make and Program Device Project	编译，对目标编程，并使选定项目保持复位。
	>Program Device for Debugging	使用调试编译对目标编程。Discrete Debugger Operation 的第二步。
	>Program Device for Production	使用生产编译对目标编程。
	>Erase Device Memory（>擦除器件存储器）	擦除器件程序存储器。
	>Programmer To Go PICkit3（>PICkit3 脱机编程）	将 PICkit 3 作为项目调试工具时，使用脱机编程（PTG）对目标编程。
	Read Device Memory	读取目标器件存储器并装入 MPLAB X IDE。
	>Discrete Multi-Partition Read Operation（>分离式多分区读操作）	对于多分区（双分区）器件，读取指定闪存分区。
	>Read Device Memory to a File	读取目标器件存储器并保存至文件。
	>Read EE/Flash Data Memory to a File	读取目标器件数据存储器并保存至文件。
	Hold in Reset/ Release from Reset（从复位释放）	可以使选定项目保持复位或从复位释放（交替进行）。
	Debug Project	编译，对目标编程，并开始调试选定项目。
	>Launch Debugger	启动调试器并运行目标代码。Discrete Debugger Operation 的第三步。
	>Live Connect Debug	对于 MEC 器件，请连接至运行的目标并检查存储器内容。允许对生产器件进行一些调试。
> = 可通过向下箭头图标实现		

相关信息

10.2.11 工具栏功能
10.2.12 定制工具栏

10.2.6 调试工具栏

调试工具栏当前包含以下功能的按钮图标。Debug 菜单上也提供了这些功能。

图标	图标文本	功能
	Finish Debugger Session	结束调试会话。
	Pause	暂停调试。使用“Continue”可继续。
	Reset	运行当前项目，直到文件中的光标位置，并停止程序执行。
	Continue	继续调试，直到达到下一个断点或程序末尾。

.....（续）

图标	图标文本	功能
	Step Over	执行程序的一个源代码行。如果该行是一个函数调用，则执行整个函数，然后停止。
	Step Into	执行程序的一个源代码行。如果该行是一个函数调用，则程序执行到该函数的第一条语句，然后停止。
	Step Out	执行程序的一个源代码行。如果该行是一个函数调用，则执行函数，并将控制返回给调用方。
	Run to Cursor	运行当前项目，直到文件中的光标位置，并停止程序执行。
	Set PC at Cursor	将程序计数器（PC）值设置为光标的行地址。
	Focus Cursor at PC	将光标移动到当前 PC 地址处，并使该地址在窗口中居中。

相关信息

10.2.11 工具栏功能

10.2.2 剪贴板工具栏

10.2.7 性能工具栏

性能工具栏包含：

项	项名称	功能
	Memory Usage Display（存储器使用情况显示）	单击可强制进行垃圾回收，以此释放存储空间。
	Profile the Application（对应用程序进行性能分析）	单击可开始对应用程序进行性能分析。再次单击可停止性能分析并保存至文件（`.npss`）。 关于 PC 采样和性能分析的更多信息，请参见 MPLAB® REAL ICE™在线仿真器文档。

10.2.8 MPLAB X Store 工具栏

MPLAB X Store 工具栏当前包含以下功能的按钮图标。Help 菜单上也提供了 MPLAB X Store 功能。

图标	图标文本	功能
	MPLAB X Store	单击可打开 IDE 中的 **MPLAB X Store** 选项卡。在此可以了解有关产品、服务和如何订购器件的信息。
	Programming Center	单击可在浏览器选项卡中显示 Microchip 编程中心的订购信息。Microchip 能够以一种快速、安全、可靠且经济高效的方式为您编程器件。

10.2.9 How Do I 工具栏

该功能需要 Internet 连接。

在 Developer Help 中搜索信息。在文本框中输入您的问题。另请参见：

https://microchipdeveloper.com/

10.1.12 Help 菜单

10.2.10 编辑器工具栏

编辑器工具栏当前包含以下功能的按钮图标。Edit 菜单和 Source 菜单上也提供了这些功能。该工具栏显示在包含当前文件源代码的选项卡的顶部。

图 10-3. 编辑器工具栏

图标	图标文本	功能
Source	Source	查看源代码。
History	History	查看源代码编辑的历史记录。
	Online Data Sheet（在线数据手册）	打开项目器件数据手册可在源代码中主动查找选定外设和 SFR（高亮显示）。 当前该功能仅适用于 AVR 器件。
	Last Edited（上次编辑）	移至上次进行编辑的行。
	Back	后退。
	Forward	前进。
	Find Selection	查找选定文本的第一个匹配项。
	Find Previous Occurrence（查找上一个匹配项）	查找选定文本的上一个匹配项。
	Find Next Occurrence（查找下一个匹配项）	查找选定文本的下一个匹配项。
	Toggle Highlight Search（翻转高亮显示搜索）	开启/关闭为搜索选定的文本。
	Toggle Rectangular Selection（翻转矩形选定内容）	开启/关闭为搜索选定的文本行。
	Previous Bookmark（上一个书签）	向后浏览书签。
	Next Bookmark（下一个书签）	向前浏览书签。
	Toggle Bookmark	在代码行上设置书签。
	Shift Left	将选定的一行或多行向左移动一个制表符距离。
	Shift Right	将选定的一行或多行向右移动一个制表符距离。
	Start Macro Recording	开始录制按键操作。
	Stop Macro Recording	停止录制按键操作。

.....（续）		
图标	图标文本	功能
	Comment（注释）	通过添加“//”使选定行变为注释。
	Uncomment（取消注释）	通过删除“//”使选定注释变为代码行。
	Go to Header/Source（转至头文件/源文件）	在头文件和相关源代码之间移动。

相关信息

10.2.11 工具栏功能
10.2.12 定制工具栏

10.2.11 工具栏功能

工具栏具有以下功能：

- 将鼠标**悬停**在图标上可弹出图标功能。
- **单击工具栏并拖动**到工具栏区域中的另一个位置。
- 在工具栏区域中**单击右键**，可显示/隐藏工具栏或更改一些工具栏的内容。
- 选择 **View>Toolbars** 可显示/隐藏工具栏、更改一些工具栏的内容或创建定制工具栏。

10.2.12 定制工具栏

您可以使用 Customize Toolbars（定制工具栏）窗口定制 MPLAB X IDE 工具栏。选择 *View>Toolbars>Customize* 打开窗口。

可用图标包括 Clean Only（仅清除）、Run 和 Set PC to Cursor（将 PC 设置为光标位置）等。

向工具栏中添加功能

- 将一个图标从 Customize Toolbars 窗口拖动到工具栏上。

从工具栏中删除功能

- 将一个图标从工具栏拖动到 Customize Toolbars 窗口上。

添加您自己的工具栏

- 单击“New Toolbar”（新建工具栏），并命名新的工具栏。

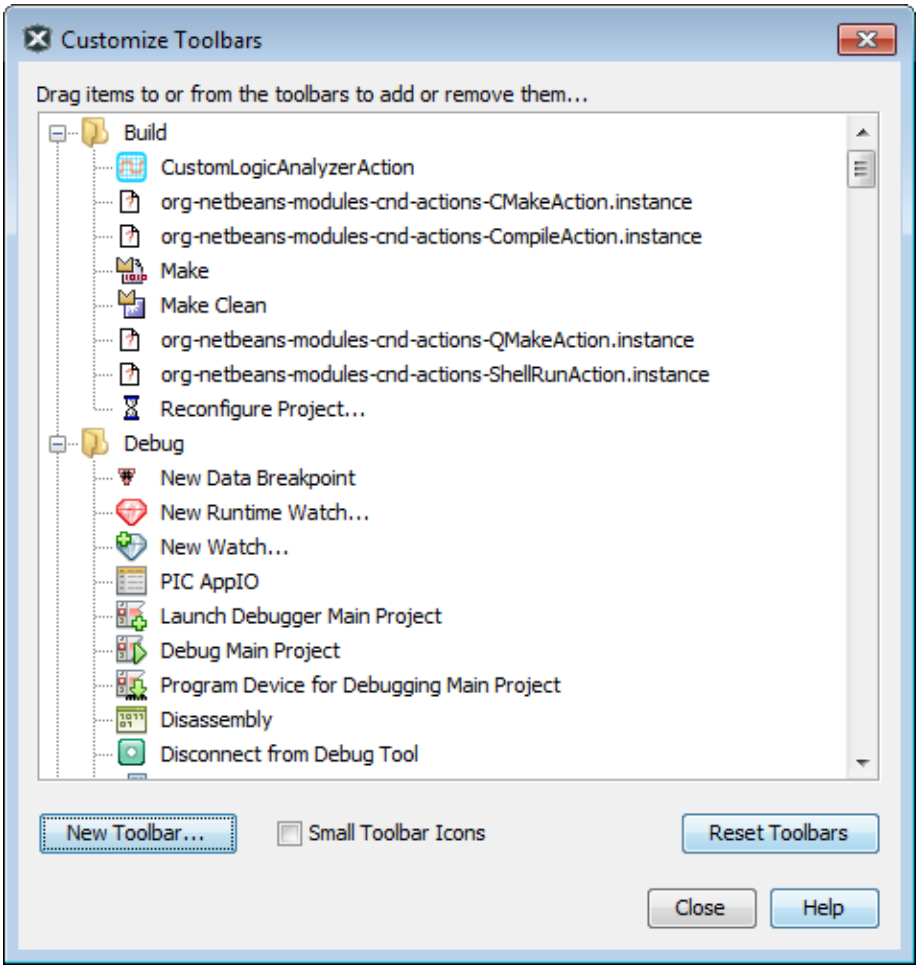
更改工具栏图标的大小

- 选中复选框“Small Toolbar Icons”可使图标变小。
- 取消选中可使图标变大。

还原为默认工具栏

- 单击“Reset Toolbars”。

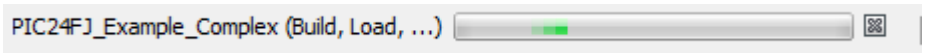
图 10-4. Customize Toolbars 窗口



10.3 状态栏

状态栏提供关于 MPLAB IDE 会话状态的最新信息。还提供编辑器信息。

图 10-5. 调试期间的状态栏

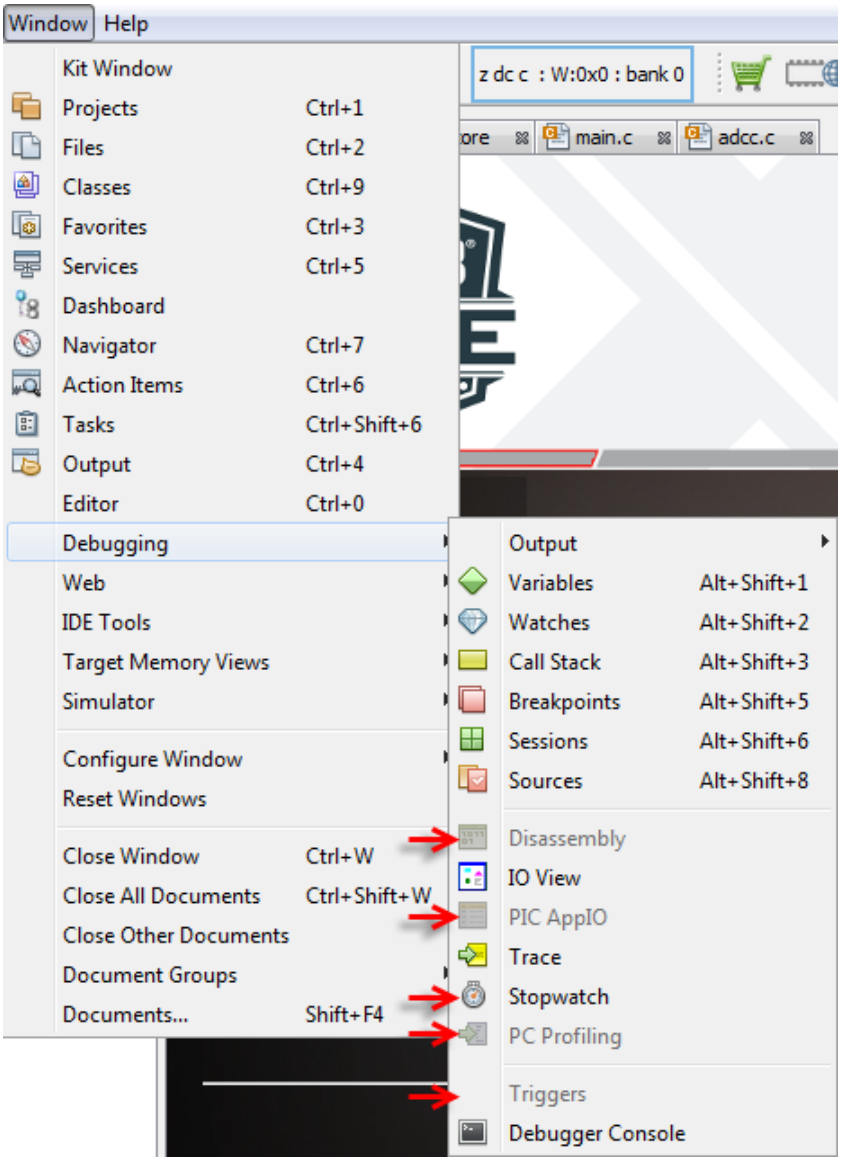


10.4 灰显或缺失的项和按钮

菜单项、工具栏按钮或状态栏项会因为几个原因而灰显（不可用）或缺失：

- 项/按钮与选定器件不具有的器件功能相关联，例如，PIC16F877A 不支持外部存储器。
- 项/按钮与选定工具不具有的工具功能相关联，例如“Step Out”对于 MPLAB ICD 3 不可用。
- 项/按钮与项目相关，但尚未选择任何项目，例如，项目编译功能不可用（No Active Project（无活动项目））。
- 项/按钮对于选定器件或工具不受支持。
- 项/按钮正在执行其功能，所以无法再次选择，例如，当程序正在运行时，“Run Project”将灰显。
- 项/按钮与另一个项互斥，例如，当程序正在运行时，“Pause”可用，而“Continue”则灰显；当程序暂停时，“Continue”可用，而“Pause”则灰显。

图 10-6. 灰显的菜单项


Window
Help
Kit Window
Projects Ctrl+1
Files Ctrl+2
Classes Ctrl+9
Favorites Ctrl+3
Services Ctrl+5
Dashboard
Navigator Ctrl+7
Action Items Ctrl+6
Tasks Ctrl+Shift+6
Output Ctrl+4
Editor Ctrl+0
Debugging
Web
IDE Tools
Target Memory Views
Simulator
Configure Window
Reset Windows
Close Window Ctrl+W
Close All Documents Ctrl+Shift+W
Close Other Documents
Document Groups
Documents... Shift+F4
z dc c : W:0x0 : bank 0
main.c
adcc.c
Output
Variables Alt+Shift+1
Watches Alt+Shift+2
Call Stack Alt+Shift+3
Breakpoints Alt+Shift+5
Sessions Alt+Shift+6
Sources Alt+Shift+8
Disassembly
IO View
PIC AppIO
Trace
Stopwatch
PC Profiling
Triggers
Debugger Console

11. MPLAB X IDE 窗口行为

MPLAB X IDE 的每个窗口都有一项特定功能，用于帮助开发人员创建和调试应用程序代码。不过，如果开发人员想要充分利用每个窗口，则需要了解部分甚至全部窗口中的功能。

11.1 MPLAB X IDE 窗口管理

关于管理 IDE 窗口的信息，请参见 NetBeans 帮助主题：

Managing IDE Windows

下面提供了更多窗口信息。

11.1.1 窗口数据更新

打开的窗口在程序暂停时发生更新（闪存存储器窗口除外，这些窗口必须进行读取）。程序暂停包括运行和单步执行后暂停。暂停时的更新会产生以下影响：

- 速度——更新需要一定的时间。要减少更新时间，请关闭所有未用窗口。
- 数据覆盖——暂停时会读取在已打开窗口中显示的文件寄存器的值。关于读取时的寄存器操作，请参见器件数据手册。

11.1.2 窗口数据更改

MPLAB X IDE 窗口数据可以按以下所述进行编辑。如果无法编辑数据，则说明该信息不可供您更改。

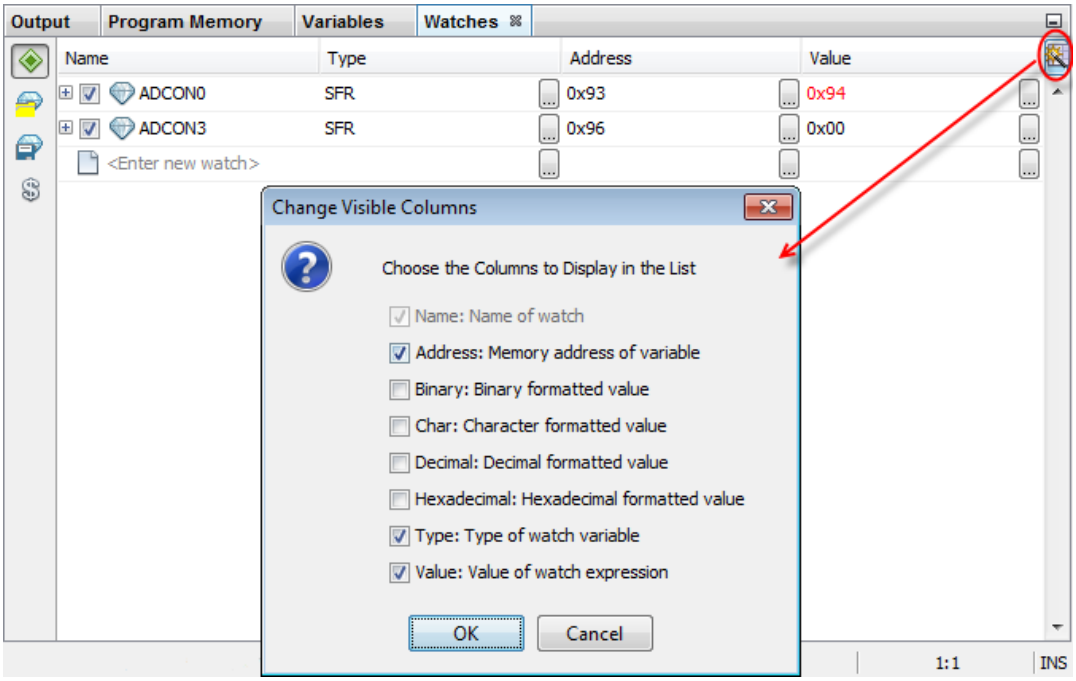
- 数据可以“就地”编辑。通过双击选择某个项，然后输入新值，或单击某个项旁边的省略号（...）并在弹出窗口中输入新值。
- 在只有某些选项时，可以从下拉列表中选择数据。

11.1.3 窗口列显示

对于某些窗口，可以更改列显示。单击窗口右上角的图标。

	更改列显示。
	更改可见列。

图 11-1. 更改可见列示例



11.1.4 窗口焦点

要确保一个窗口具有焦点，不仅要单击窗口框架，还要单击按钮、表格单元格或下拉组合框。

11.2 分区数据存储器和窗口中显示的值

该功能仅适用于使用分区数据存储器（RAM）的 8 位 PIC MCU 器件。

在 MPLAB X IDE v5.20 及以下版本中，寄存器地址所显示的值中始终包含存储区偏移量。不过，这并非始终正确，只有当指令位于程序计数器（PC）时，才能精确计算实际值。

在 MPLAB X IDE v5.25 及以上版本中，只有当 PC 与寄存器地址位于同一行时，寄存器地址才包含存储区偏移量。PC 不位于该行时，值默认为代码中显示的值（无存储区偏移量或存储区 0）。

例 11-1. v5.20 到 v5.25 的行为变化

以下图中的示例代码片段为例。下表列出了 Program Memory 窗口（DisAssy 列）和 Disassembly 窗口中的反汇编代码的显示行为。

图 11-2. 示例代码片段

```
14      asm("MOVLB 0x3A"); // change to bank 58
15      asm("CLRF 0x74"); // clear common RAM register
```

表 11-1. v5.20 到 v5.25 的行为变化

MPLAB X IDE	第 15 行的反汇编——未在 PC 处	第 15 行的反汇编——在 PC 处
v5.20 及以下版本	CLRF 0x1D74	CLRF 0x1D74
v5.25 及以上版本	CLRF 0x74	CLRF 0x1D74

12. MPLAB X IDE 窗口和对话框

MPLAB X IDE 桌面具有四个部分（即窗格），其中包含选项卡式窗口。窗口仅在选择某种功能之后可见。窗口可能具有关联的对话框，以供输入特定信息。

例如，首次打开 MPLAB X IDE 时，只会打开 **Start Page** 和 **MPLAB X Store** 选项卡式窗口。打开某个项目之后，将打开基本窗口——即左上窗格中的 Projects 和 Files 窗口，左下窗格中的 Dashboard 和 Navigator 窗口，以及右下窗格中的 Output 窗口。**Start Page** 和 **MPLAB X Store** 的选项卡式窗口将移至右上窗格。

MPLAB X IDE 窗口是基本 NetBeans 平台窗口和 MPLAB X IDE 特定窗口的组合。从菜单项中选择时，将打开对话框。与窗口一样，MPLAB X IDE 对话框是基本 NetBeans 平台对话框和 MPLAB X IDE 特定对话框的组合。关于 NetBeans 窗口和对话框的信息，请参见 13.1 NetBeans 特定窗口和窗口菜单。

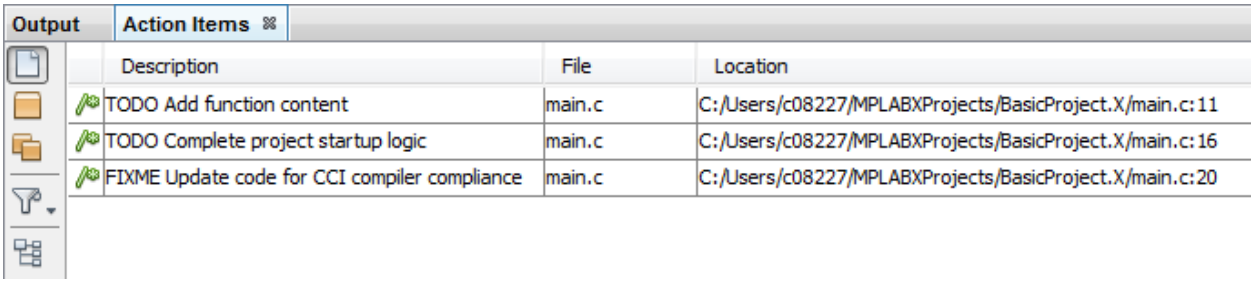
12.1 Action Items 窗口

通过 *Window>Action Items* 打开 Action Items 窗口。

Action Items 窗口将统一列出您的各个文件和项目中需要解决的问题，其中包括编译器警告和错误。可通过在代码中添加注释指定操作项。例如，下图中使用了以下代码行：

```
//TODO Add function content
        :
//TODO Complete project startup logic
        :
//FIXME Update code for CCI compiler compliance
```

图 12-1. Action Items 窗口示例



可以使用筛选器来确定列表中显示的条目，单击列标题可对列表进行排序。可使用 Action Items 窗口图标来修改和处理显示的项。

更多信息，请参见： http://microchipdeveloper.com/mplabx:tasks-list

12.1.1 Action Items 窗口显示

这种显示格式会在以下各列中显示数据：

- Description——所需操作的说明。
- File——文件操作的名称。
- Location（位置）——文件位置。

Action Items 窗口图标显示在窗口的左侧。

表 12-1. Action Items 窗口图标

图标	图标文本	说明
	Show action items for currently edited file only（仅显示当前已编辑文件的操作项）	选定后，IDE 将扫描编辑器中当前选定的文件并列出在该文件中找到的操作项。在编辑器中打开一个新文件时，将会对该文件进行扫描并且仅显示该文件中的操作项。

.....（续）		
图标	图标文本	说明
	Show action items for the selected project（显示选定项目的操作项）	选定后，IDE 将扫描当前选定项目中的所有文件并在项目窗口中显示所有操作项。
	Show action items for all opened projects（显示所有已打开项目的操作项）	选定后，IDE 将扫描所有已打开项目中的所有文件并在窗口中列出所有操作项。
	Click here to select a filter（单击此处可选择一个筛选器）	单击 Filter 按钮可修改操作项的筛选器或创建一个新的筛选器。可以单击该按钮上的下拉列表，选择一个筛选器应用于操作项列表。
	Group action items by category（按类别对操作项进行分组）	选定后，将按类别对操作项列表进行分组。

12.1.2 Action Items 窗口菜单

选中窗口中的某一行并单击右键可弹出该菜单。

表 12-2. Action Items 窗口上下文菜单

项	说明*
Show（显示）	显示错误/警告在代码中的位置。
Scope（范围）	显示项的范围——当前文件、当前项目和所有项目。
Filter	显示项的筛选方式。也可更改筛选方式。
Refresh（刷新）	刷新窗口内容。
Sort By（排序方式）	指定窗口内容的排列方式。

12.2 Breakpoints 窗口

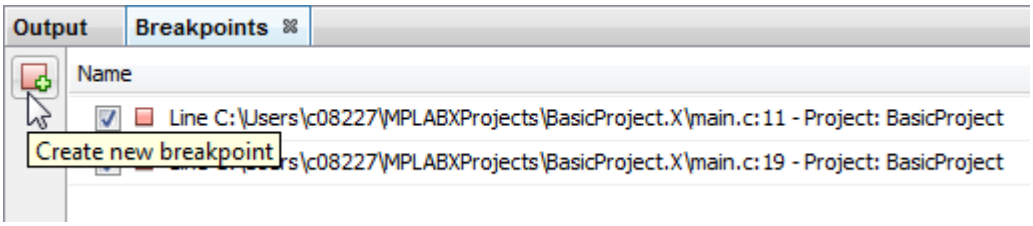
通过选择 *Window>Debugging>Breakpoints* 打开窗口。

Breakpoints 窗口用于管理代码中的断点。关于使用 Breakpoints 窗口的更多信息，请参见 4.14 使用断点控制程序执行。

New Breakpoint 对话框

单击 Breakpoint 窗口中的 **Create New Breakpoint** 按钮可打开 4.14.2 使用 Breakpoint 对话框设置断点并创建新断点。将列出项目器件所允许的断点类型。

图 12-2. New Breakpoint 图标



Breakpoint 窗口菜单

右键单击窗口的空白区域可弹出以下菜单。

表 12-3. Breakpoints 窗口菜单——未选中断点行

项	说明
New Breakpoint	打开 New Breakpoint 对话框。
Enable All（全部使能） Disable All（全部禁止）	使能或禁止全部断点。
Delete All（全部删除）	删除全部断点。

右键单击窗口中的现有断点行可弹出以下菜单。

表 12-4. Breakpoints 窗口菜单——选中断点行

项	说明
Go to Source	显示断点在代码中的位置。
Complex Breakpoint	对于复杂断点，请将该断点添加到新序列或新元组。还需指定在序列或元组中的位置。
Disable Enable（使能）	禁止或重新使能该断点的操作。
New Breakpoint	打开 New Breakpoint 对话框。
Enable All Disable All	使能或禁止全部断点。
Delete Delete All	删除该断点或全部断点。
Customize	定制断点行为。

12.3 New Breakpoint 对话框

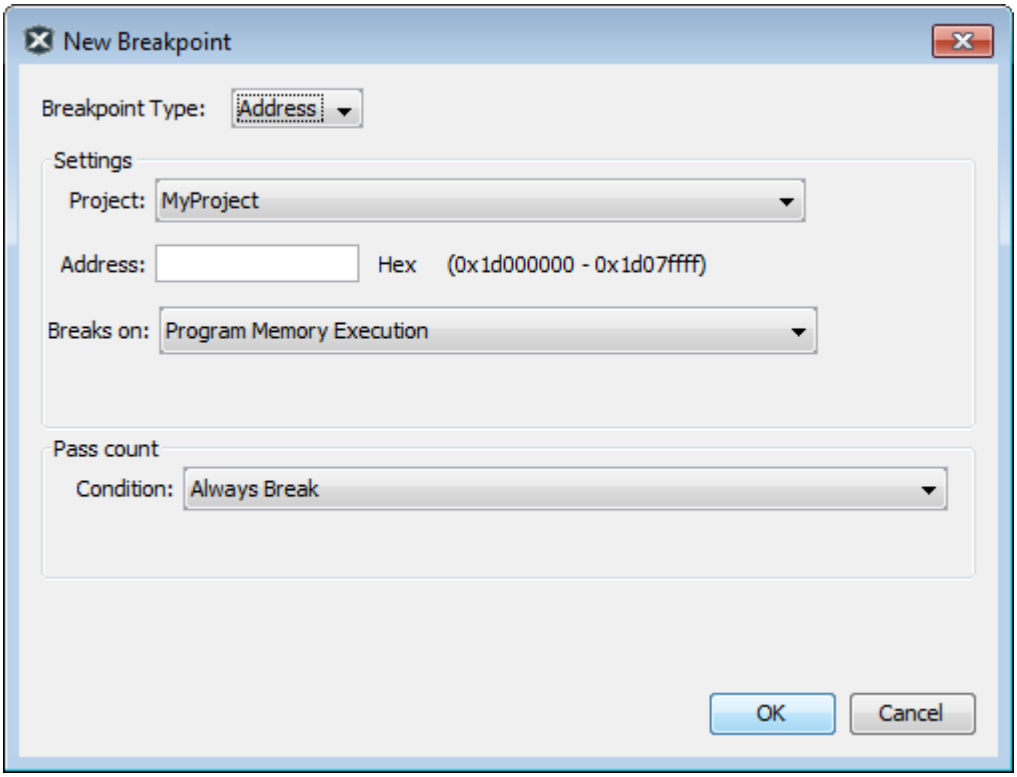
从 Breakpoints 窗口打开该对话框（*Window>Debugging>Breakpoints*）。单击“New Breakpoint”图标  或从上下文菜单中选择“New Breakpoint”。

对应于断点的选项由所选断点的类型和所选器件决定（并非所有器件具有所有选项）。

断点的类型包括：

- 12.3.1 行断点
- 12.3.2 数据断点
- 12.3.3 地址断点
- 12.3.4 事件断点
- 12.3.5 函数断点
- 12.3.6 次数计数操作

图 12-3. New Breakpoint 对话框——Address



12.3.1 行断点

以下选项可用于在代码行上指定断点。

表 12-5. Breakpoint Type（断点类型）：Line——Settings（设置）

项	说明
Settings	通过用鼠标左键单击文件行旁边的编辑器装订线可创建新的行断点。或者，从编辑器上下文菜单中选择“Toggle Line Breakpoint”。

12.3.2 数据断点

以下选项可用于数据存储器断点。

表 12-6. Breakpoint Type：Data——Settings

项	说明
Project（项目）	从下拉列表中选择一个打开的项目 它是代码中包含了该断点的项目。
Symbols（符号）	输入全局符号或 SFR 的名称，或通过单击 **Symbols** 浏览选择一个。根据“Breaks on”（中断条件）访问该符号会触发代码执行发生暂停。
Enable Range Address（使能范围地址）	选中时可设置范围断点。 取消选中时可设置单个断点。
Address Address (Start)（地址（起始））	根据“Enable Range Address”的选择，该项可能为“Address”或“Address (Start)”。 输入数据存储器中的十六进制地址 根据“Breaks on”访问该地址会触发代码执行发生暂停。

.....（续）	
项	**说明**
Address (End)（地址（结束））	如果选中了“Enable Range Address”，则会使能该输入框。 输入数据存储器中的十六进制地址。
Breaks on	**Read（读）、Write（写）和 Read or Write（读或写）：**当上述符号或地址被读取、写入或者读取或写入时，中断代码执行。 **Read Specific Value（读特定值）、Write Specific Value（写特定值）和 Read or Write Specific Value（读或写特定值）：**当上述符号或地址被读取并具有下面指定的值、被写入下面指定的值或者根据下面指定的值发生读取或写入时，中断代码执行。 **注：** 对于 dsPIC DSC，具有对应于 X 和 Y 总线的读和写选项。
Value	对“Breaks on”选择 Read Specific Value、Write Specific Value 或 Read or Write Specific Value 时，在此处输入一个十六进制值。
Value Comparison（值比较）	*仅对于 PIC16F1xxx 器件——* 与指定的“Value”进行比较： **= Value：**等于值 **!= Value：**不等于值 **> Value：**大于值 **< Value：**小于值
Data Value Mask（数据值掩码）	*仅对于 PIC16F1xxx 器件——* 在与“Value”比较时使用掩码 输入一个介于范围 0x00 到 0xhh 之间的值，其中： 0x00：不比较任何位 0xhh：比较所有位

表 12-7. Breakpoint Type：Data——Pass Count（次数计数）

项	说明
Condition（条件）	确定何时发生“Breaks on”下指定的中断。 **Always Break（总是中断）：**总是在满足“Breaks on”条件时发生中断。 **Break occurs Count Instructions after Event（事件之后的 Count 条指令后发生中断）：**发生事件（“Breaks on”条件）后，在实际中断之前先执行 Count 条指令。 **Event must occur Count times（事件必须发生 Count 次）：**只有发生 Count 次事件（“Breaks on”条件）之后，才会实际发生中断。
Count（计数）	根据指定的条件，输入事件后的指令计数数量或事件数量。请参见 12.3.4 事件断点。
Trigger Options（触发选项）	*仅对于 PIC16F1xxx 器件——* 选择何时触发： • 到达断点时不触发输出 • 到达断点时触发输出

.....（续）	
项	**说明**
Interrupt Context（中断上下文）	*仅对于 PIC16F1xxx 器件——* 地址/数据断点的中断上下文限定符。选择如下： • 总是中断（在 ISR 和主代码中均中断） • 仅在主干（非中断）上下文中发生中断——仅在主代码中发生中断 • 仅在中断上下文中发生中断——仅在 ISR 代码中发生中断

12.3.3 地址断点

以下选项可用于程序/执行存储器断点。

表 12-8. Breakpoint Type：Address——Settings

项	说明
Project	从下拉列表中选择一个打开的项目。 它是代码中包含了该断点的项目。
Enable Range Address	选中时可设置范围断点。 取消选中时可设置单个断点。
Address Address (Start)	根据“Enable Range Address”的选择，该项可能为“Address”或“Address (Start)”。 输入数据存储器中的十六进制地址。 根据“Breaks on”访问该地址会触发代码执行发生暂停。
Address (End)	如果选中了“Enable Range Address”，则会使能该输入框。 输入数据存储器中的十六进制地址。
Breaks on	**Program Memory Execution**（程序存储器执行）：到达上面指定的地址时，中断代码执行。 **TBLRD Program Memory**（表读程序存储器）：在对以上指定的地址进行表读时，中断代码执行。 **TBLWT Program Memory**（表写程序存储器）：在对以上指定的地址进行表写时，中断代码执行。

表 12-9. Breakpoint Type：Data——Pass Count

项	说明
Condition	确定何时发生“Breaks on”下指定的中断。 **Always Break**：总是在满足“Breaks on”条件时发生中断。 **Break occurs Count Instructions after Event**：发生事件（“Breaks on”条件）后，在实际中断之前先执行 Count 条指令。 **Event must occur Count times**：只有发生 Count 次事件（“Breaks on”条件）之后，才会实际发生中断。
Count	根据指定的条件，输入事件后的指令计数数量或事件数量。请参见 12.3.4 事件断点。
Trigger Options	*仅对于 PIC16F1xxx 器件——* 选择何时触发： • 到达断点时不触发输出 • 到达断点时触发输出

.....（续）	
项	**说明**
Interrupt Context	*仅对于 PIC16F1xxx 器件——* 地址/数据断点的中断上下文限定符。选择如下： • 总是中断（在 ISR 和主代码中均中断） • 仅在主干（非中断）上下文中发生中断——仅在主代码中发生中断 • 仅在中断上下文中发生中断——仅在 ISR 代码中发生中断
*另请参见“对于某些器件（PIC16F1xxx MCU），提供了增强型事件断点操作”部分。	

12.3.4 事件断点

以下选项可用于事件断点。

表 12-10. Breakpoint Type：Event

项*	说明
Project	从下拉列表中选择一个打开的项目。 它是代码中包含了该断点的项目。
Break on clock mode switch（发生时钟模式切换时中断）	当时钟模式切换时发生中断。
Break on Reset instruction（发生复位指令时中断）	发生器件复位指令时中断。
Break on Sleep（休眠时中断）	进入休眠时中断。
Break on stack over/underflow（堆栈上溢/下溢时中断）	在堆栈上溢或下溢时中断。
Break on wake up（唤醒时中断）	器件从休眠中唤醒时中断。
Break when watchdog timer has expired（看门狗定时器计满时中断）	在看门狗定时器周期结束时中断。
Break on execution out of bounds（执行超出范围时中断）	在程序试图移出正常程序存储器/空间时中断。
Break on MCLR Reset（MCLR 复位时中断）	发生主复位（MCLR）时中断。
Break on trigger in signal（触发输入信号时中断）	检测到触发输入脉冲时中断。
*从列表中选择将导致执行代码暂停（中断）的事件。某些事件可能对于您的器件不可用。	

对于某些器件（PIC16F1xxx MCU），提供了增强型事件断点操作。

操作	说明
Break（中断）	根据指定的选项中断（暂停）执行。
Trigger out（触发输出）	根据指定的选项发出触发输出脉冲。
Break and trigger out（中断和触发输出）	根据指定的选项中断（暂停）执行并发出触发输出脉冲。

12.3.5 函数断点

以下选项可用于函数断点。

表 12-11. Breakpoint Type：Function——Settings

项	说明
Symbol/Hex Address（符号/十六进制地址）	输入或选择函数的符号表示或地址。

表 12-12. Breakpoint Type：Function——Conditions（条件）

项	说明
Condition	选中即可使能。然后输入中断条件。
Break when hit count exceeds（超出命中计数时中断）	选中即可使能。然后输入一个表示将进入函数的次数的整数，超过该次数后将发生中断。

表 12-13. Breakpoint Type：Function——Actions（操作）

项	说明
Suspend（暂停）	中断时，暂停： **No thread (continue)（无线程（继续））**：不暂停任何程序 **Breakpoint thread**（断点线程）：仅暂停函数线程 **All threads**（所有线程）：暂停所有线程（而非主程序）
Thread ID（线程 ID）	设置断点线程的 ID。
Print Text（打印文本）	中断时，输入要在 Output 窗口中显示的文本。

12.3.6 次数计数操作

使用次数计数使您可以延迟中断，直到达到指定的计数。

Break occurs Count Instructions after Event

计数是在断点之后、停止之前执行的指令数。

例如，

- 0 表示立即停止执行
- 1 表示在一条额外指令之后停止执行
- 10 表示在十条额外指令之后停止执行

Event must occur Count times

计数是在停止之前执行事件的次数。

例如，

- 0 表示立即停止执行
- 1 表示执行事件一次，然后在下一次（事件第二次发生时）停止
- 10 表示执行事件十次，然后在下一次（事件第十一次发生时）停止

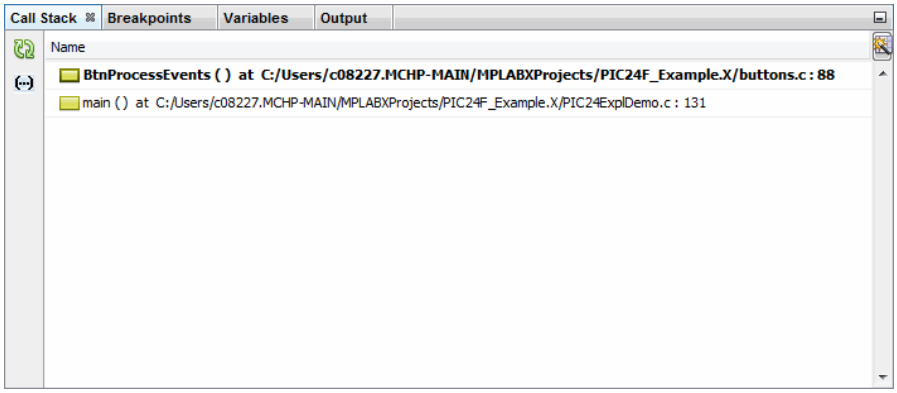
12.4 Call Stack 窗口

选择 *Window>Debugging>Call Stack* 打开 Call Stack 窗口。必须处于调试模式才能查看内容。

Call Stack 窗口将显示 C 函数及其参数，按照它们在正执行的程序中的调用顺序列出。未优化的 C 代码性能最佳。

Call Stack 适用于 16 位和 32 位器件。该窗口在调试暂停或停止时更新。

图 12-4. Call Stack 窗口



暂停或停止时窗口中提供以下图标。

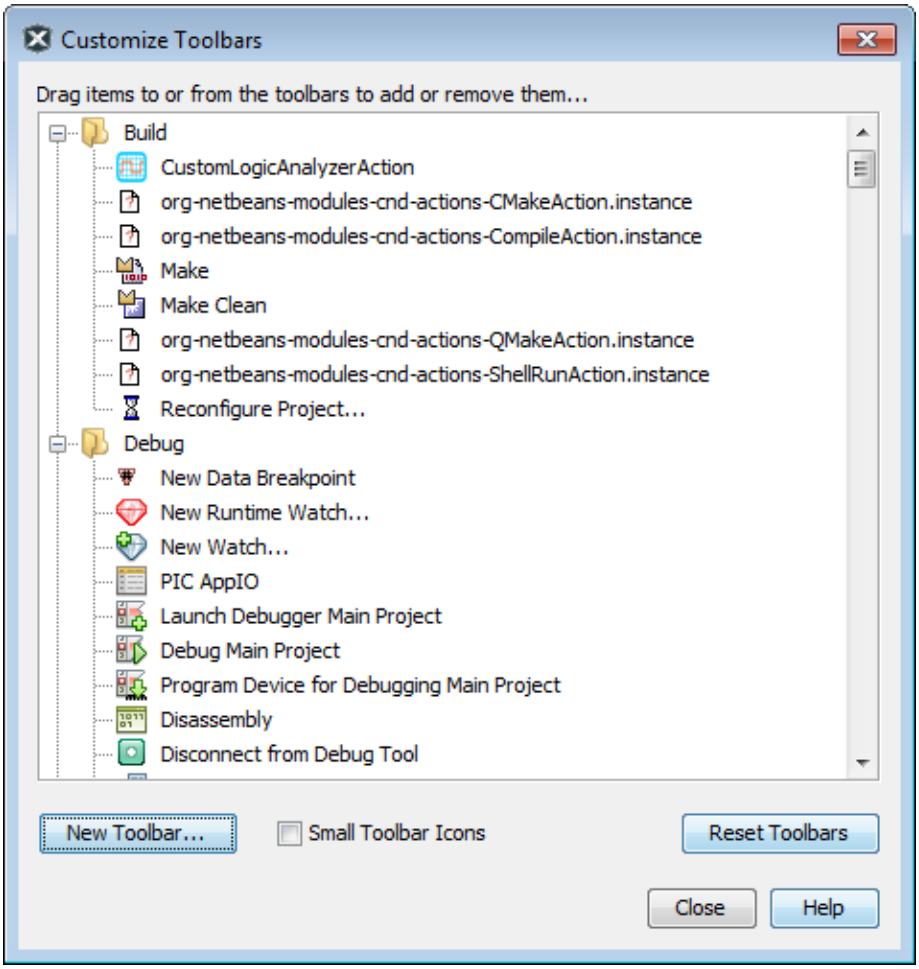
表 12-14. Call Stack 图标

图标	说明
	自动或手动刷新窗口内容。取决于 *Tools>Options>Embedded>Generic Settings* 下的“Disable auto refresh for call stack view during debug sessions”的值——未选中 = 假（默认），选中 = 真。请参见 12.16.1 Generic Settings 选项卡。 假 = 自动刷新（绿色图标）：暂停或停止时自动更新窗口内容。如果窗口关闭或未处于焦点，则选中窗口并单击该按钮即可显示更新内容，无需再次运行并暂停/停止。 真 = 手动刷新（橙色图标）：暂停或停止时不自动更新窗口内容。必须单击该按钮才能在暂停/停止时更新窗口内容。
	选中可禁止/使能函数参数变量的求值。
	更改可见列。对于该窗口，显示或隐藏调用堆栈帧的位置。

12.5 Customize Toolbars 窗口

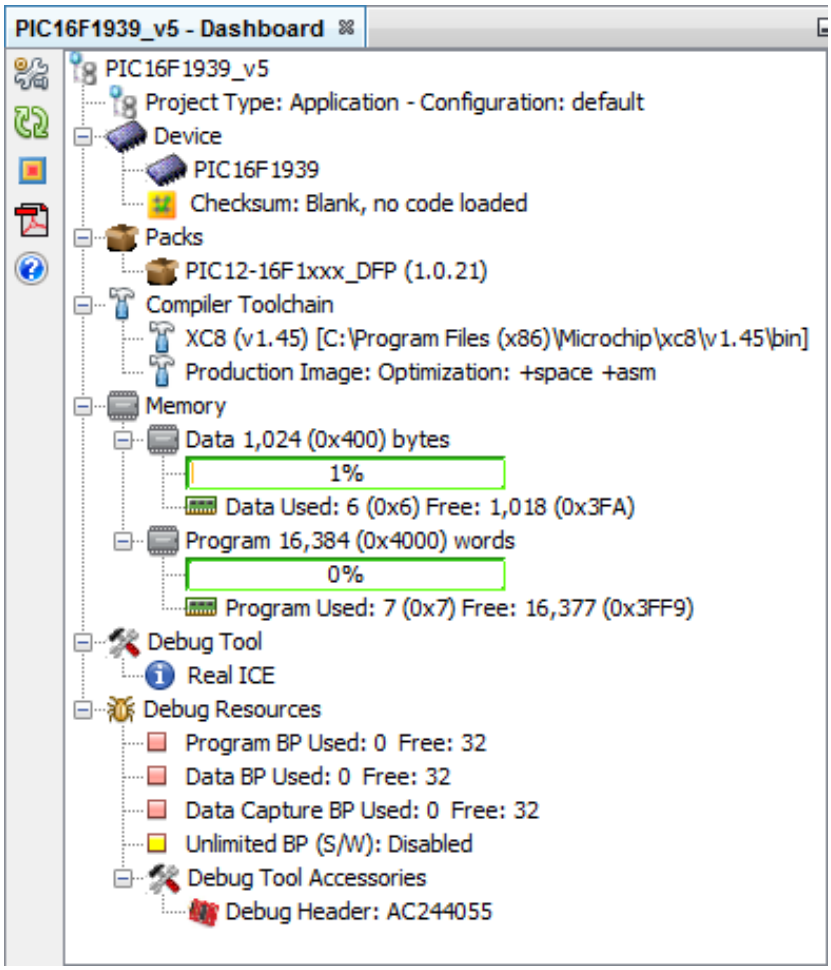
您可以使用 Customize Toolbars 窗口定制 MPLAB X IDE 工具栏。选择 *View>Toolbars>Customize* 打开窗口。

关于定制说明，请参见 10.2.12 定制工具栏。



12.6 Dashboard 窗口

Dashboard 窗口包含常规项目信息，例如校验和、编译器版本、存储器使用情况和断点资源。更多信息，请参见：5.19 查看仪表板显示。



12.7 Disassembly 窗口

在 Disassembly 窗口中查看反汇编的 C 代码。选择 *Window>Debugging>Disassembly（窗口>调试>反汇编）*打开该窗口。必须处于调试模式。

调试时，“Step Over”功能在源代码上下文中使用。反汇编调试使用“Step In”（单步进入）功能实现。如果反汇编调试期间不需要“Step In”（例如在需要单步跳过而非单步进入 CALL 指令的情况下），则可以使用“Run to Cursor”功能来实现。将光标置于需要运行至的 PC 所在代码行上，然后单击“Run to Cursor”/F4 按钮。

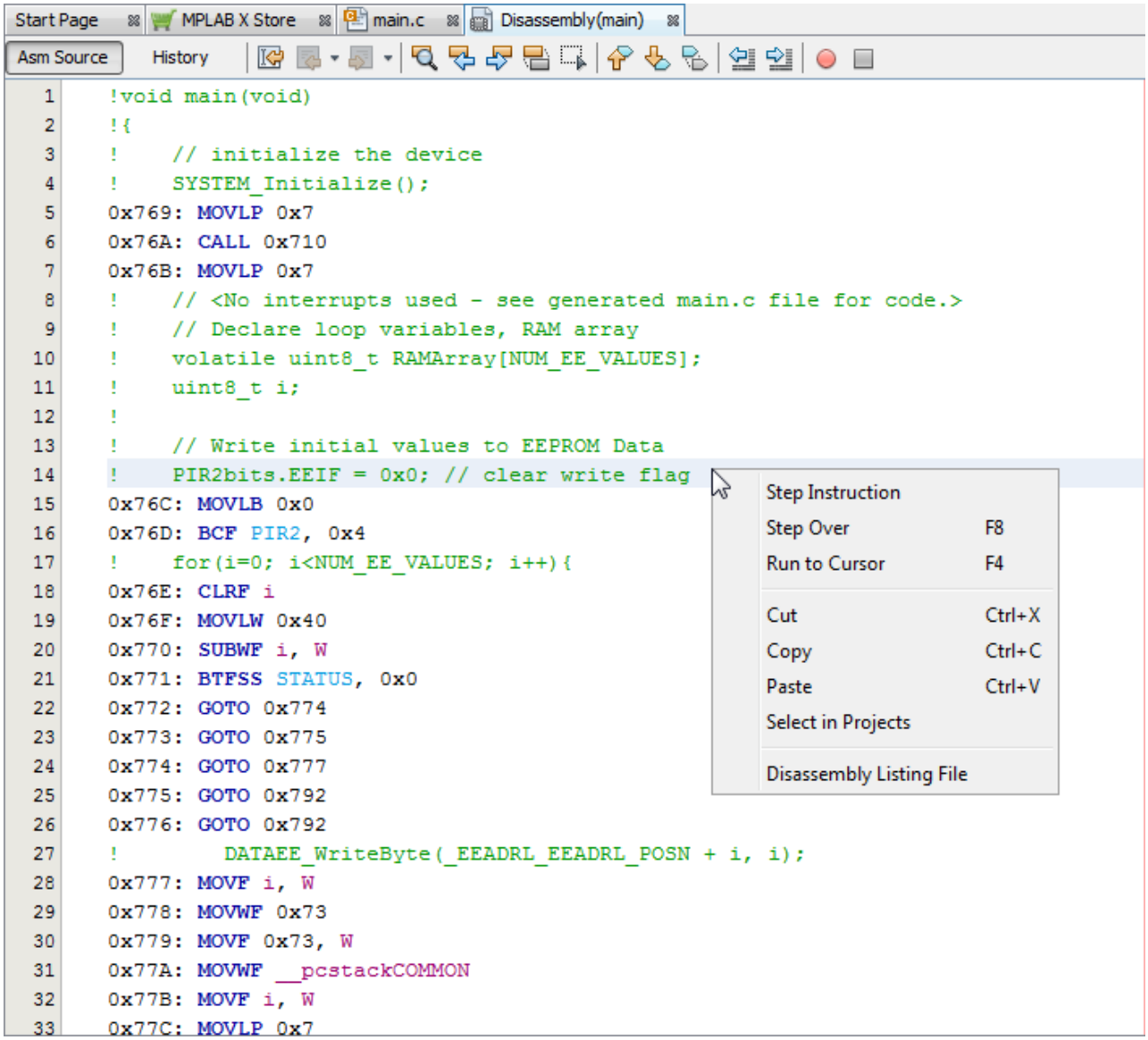
在窗口中单击右键可查看上下文菜单。

查看整个反汇编文件的一种快速方法是选择“Disassembly Listing File”。

如果项目器件使用 RAM 分区，另请参见：

11.2 分区数据存储器和窗口中显示的值

图 12-5. Disassembly 窗口



12.8 IO View 窗口

使用 I/O View 窗口（*Window>Debugging>IO View*）可查看当前项目的目标器件的寄存器概览。关于操作的概述，请参见：

5.20 查看项目的寄存器（I/O View）

IO View 窗口的各个部分

该窗口的默认视图是垂直拆分的，顶部是外设组，底部是寄存器。每个外设通常都已定义设置和值枚举，具体信息可通过在外设视图（顶部）中展开寄存器来显示。寄存器视图（底部）将显示属于所选外设组的全部寄存器。如果未选择任何外设，则视图为空。此外，也可通过展开每个寄存器来显示属于该寄存器的预定义值分组。

图 12-6. 具有内容的 I/O View 窗口

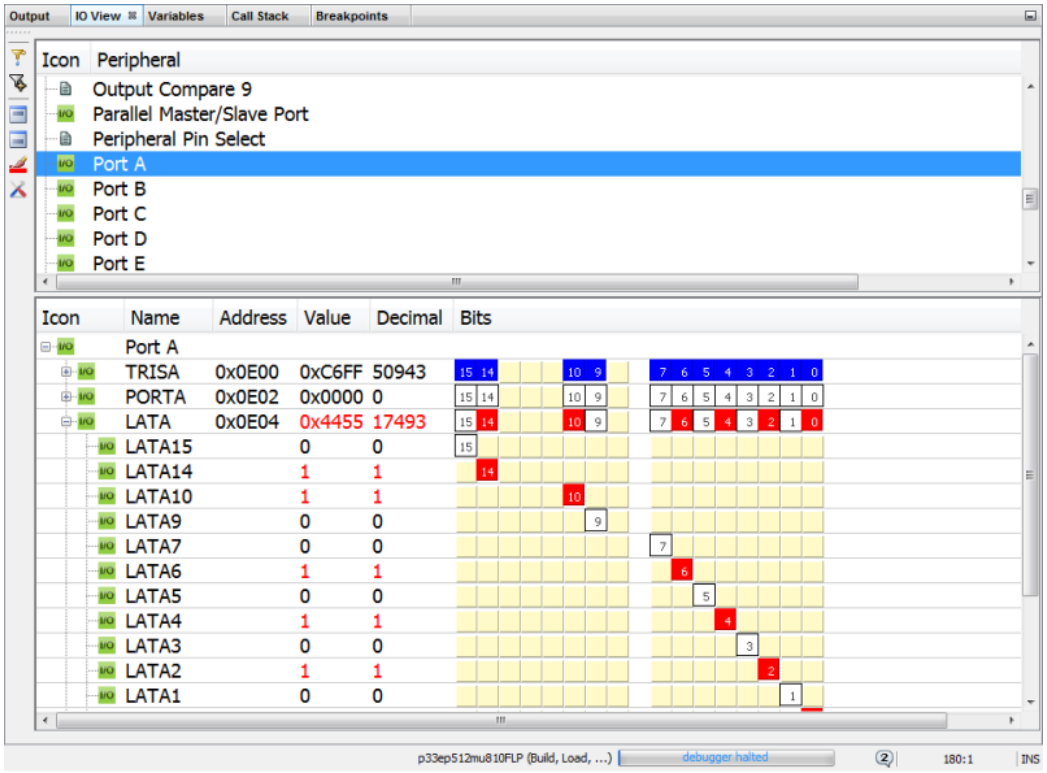


表 12-15. IO View 图标

图标	功能
	Filter Groups（筛选组）。对视图中显示的外设组进行筛选。 单击该图标可打开 Select 对话框。可通过按住 Shift 并单击（范围）或按住 Ctrl 并单击（单个）的方式选择外设。也可以指定一个寄存器文本筛选器文件。
	Filter Select All Groups（筛选所有组）。
	Toggle Peripheral Pane（翻转外设窗格）。 单击可显示/隐藏外设（顶部）窗格。
	Toggle Register Pane（翻转寄存器窗格）。 单击可显示/隐藏寄存器（底部）窗格。
	Showing Standard Bits（显示标准位）。翻转后可高亮显示已修改位。 对于已修改位，切换后可显示标准位（不高亮显示）。 单击蓝线图标可更改为红线图标。
	Highlighting Modified Bits（高亮显示已修改位）。翻转后可显示标准位。 已修改位在画面中高亮显示。 单击红线图标可更改为蓝线图标。

.....（续）

图标	功能
	翻转已修改位的高亮显示。 如果上一个图标选为“Highlighting Modified Bits”，则单击该图标可翻转已修改位的高亮显示类型。 单击可打开 IO View Settings（IO 视图设置）对话框。

表 12-16. I/O View 上下文菜单

项	功能
Expand All	在窗格/各部分中展开所有折叠的内容。
Expand Row（展开行）	在选定行中展开折叠的内容。
Collapse All	在窗格/各部分中折叠所有展开的内容。
Adjust Table Columns（调整表列）	仅限寄存器窗格。调整表列的宽度。

12.9 存储器窗口——8 位和 16 位器件

存储器窗口（*Window>Target Memory Views*）会显示许多器件存储器类型，例如 SFR 和配置位。使用“Memory”和“Format”下拉框可以定制您的窗口。

关于这些控件的更多信息，请参见 4.18 查看或更改器件存储器。

关于相关对话框的详细信息，请参见 12.11 与存储器窗口关联的对话框。

图 12-7. Window>Target Memory Views——PIC16F1939

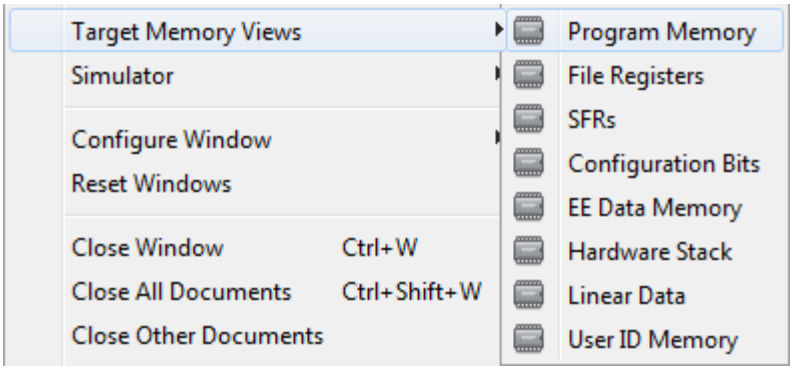
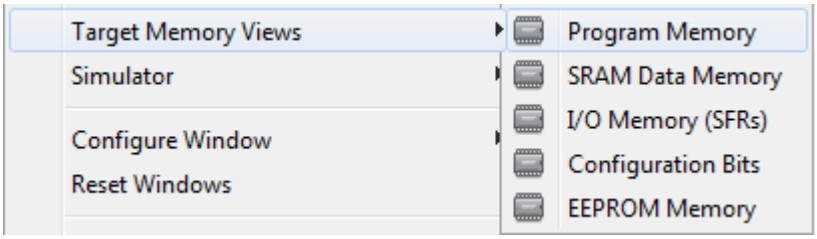


图 12-8. Window>Target Memory Views——ATtiny817



12.9.1 Program Memory 窗口

Program Memory 窗口会显示处于当前所选处理器的程序存储器范围内的存储单元。如果所选器件支持并使能了外部程序存储器，则它也会显示在 Program Memory 窗口中。

对于 MPLAB X 软件模拟器，当程序存储器值发生更改或处理器暂停时，Program Memory 窗口中的数据会发生更新。

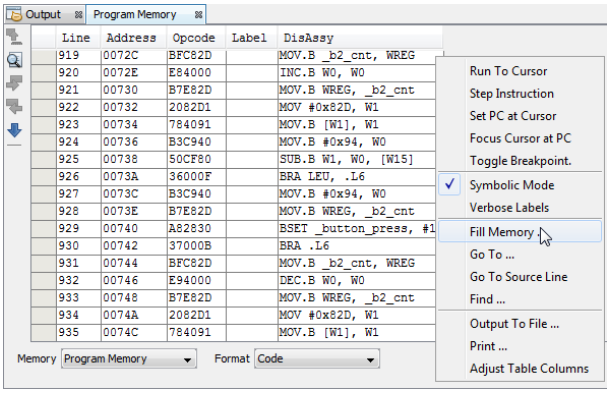
对于任何 Microchip 硬件调试工具（例如，MPLAB ICD 4 在线调试器），当某个程序存储器值发生更改或处理器暂停时，Program Memory 窗口中的数据**不会**发生更新；您必须对器件存储器执行读操作。

使用硬件工具进行调试时，某些寄存器的每半字节数据可能会显示“R”，以表示保留的资源。

如果项目器件使用 RAM 分区，另请参见：

11.2 分区数据存储器和窗口中显示的值

图 12-9. 具有内容的 Program Memory 窗口



12.9.1.1 Program Memory 窗口显示

您可以通过从窗口底部的 Format 下拉框中进行选择来指定存储器在窗口中的显示方式。

使用硬件工具进行调试时，某些寄存器的每半字节数据可能会显示“R”，以表示保留的资源。

Code（代码）Format

Code 格式会显示反汇编的十六进制代码和符号。窗口将具有以下列：

- Debug Info（调试信息）——可用于调试的信息。一个指针会显示程序计数器的当前位置。
- Line——参考行号。
- Address——操作码十六进制地址。
- Opcode（操作码）——十六进制操作码，以 2 或 3 字节块的形式显示
  对于大多数 PIC MCU，这些块代表字。对于 PIC18CXXX 器件，这些块代表 2 字节。对于 dsPIC DSC 器件，这些块代表 3 字节。
- Label (Symbolic Only)（标号（仅符号））——以符号格式显示的操作码标号。
- Disassembly——操作码助记符的反汇编版本。

Hex（十六进制）格式

该格式将以十六进制代码的形式显示程序存储器信息。窗口将具有以下列：

- Address——下一列中的操作码的十六进制地址。
- Opcode Blocks（操作码块）——十六进制操作码，以 2 或 3 字节块的形式显示
  对于大多数 PIC MCU，这些块代表字。对于 PIC18CXXX 器件，这些块代表 2 字节。对于 dsPIC DSC 器件，这些块代表 3 字节。
  高亮显示的操作码块代表程序计数器的当前位置。
- ASCII——相应操作码行的 ASCII 表示形式。

PSV Mixed（PSV 混合）（仅适用于 dsPIC DSC/PIC24 器件）

该格式会以操作码和 PSV 区域（CORCON 寄存器中的 PSV 位置 1）的形式显示程序存储器。窗口将具有以下列：

- Debug Info——可用于调试的信息。一个指针会显示程序计数器的当前位置。
- Line——参考行号。
- Address——操作码十六进制地址。
- PSV Address（PSV 地址）——操作码的数据空间十六进制地址。

- Data——格式化为数据形式的操作码。
- Opcode——十六进制操作码，以 3 字节块的形式显示。
- Label（标号）——以符号格式显示的操作码标号。
- Disassembly——操作码助记符的反汇编版本。

更多信息，请参见《dsPIC30F 系列参考手册》（DS70046E_CN）。

PSV Data（PSV 数据）（仅适用于 dsPIC DSC/PIC24 器件）

当程序空间在数据空间中可见时（CORCON 寄存器中的 PSV 位置 1），该格式会以文件寄存器的形式显示程序存储器。窗口将具有以下列：

- Address——数据的程序空间十六进制地址。
- PSV Address——数据的数据空间十六进制地址。
- Data Blocks（数据块）——十六进制数据，以 3 字节块的形式显示。高亮显示的数据块代表程序计数器的当前位置。
- ASCII——相应数据行的 ASCII 表示形式。

更多信息，请参见《dsPIC30F 系列参考手册》（DS70046E_CN）。

12.9.1.2 Program Memory 窗口图标

图标位于窗口左侧。

表 12-17. Program Memory 窗口图标

图标	图标文本	功能
	Refresh by Read Device Memory（通过读取器件存储器刷新）	与调试工具栏“Read Device Memory”图标的功能相同——将器件存储器的内容上传到 MPLAB X IDE。
	Find	指定要在窗口中查找的字符串。可选择全字匹配或区分大小写。
	Find Next	查找 Find 的下一个字符串实例。
	Find Previous	查找 Find 的上一个字符串实例。
	Go To（转至）	转至指定的行号/地址。

12.9.1.3 Program Memory 窗口菜单

在设置为 CODE 或 PSV MIXED 格式的存储器窗口数据区域中单击右键可弹出该菜单。并非所有项对于所有器件均可见。

表 12-18. Program Memory 窗口菜单——Code/PSV Mixed

项	说明
Run to Cursor	将程序运行至当前光标位置。
Step Instruction	执行单步指令。
Set PC at Cursor	将程序计数器（PC）设置在光标位置。
Focus Cursor at PC	将光标移动到当前 PC 地址处，并使该地址在窗口中居中。
Toggle Breakpoint（翻转断点）	翻转（开启/关闭）现有断点。
Symbolic Mode（符号模式）	显示反汇编的十六进制代码和符号。

.....（续）

项	说明
Verbose Labels（详细标签）	**仅限 PSV Mixed 格式** 显示内部编译器标签。
Fill Memory	使用 Data 中的值从 Start Address（起始地址）到 End Address（结束地址）填充存储器。 在 Fill Memory 对话框中指定其他选项。
Go To	转至在 Go To 对话框中指定的地址/函数。
Go To Source Line（转至源代码行）	转至编辑器中的相应源代码行。
Find	查找在 Find 对话框中指定的文本。
Output To File（输出到文件）	使用 Output To File 对话框将显示的窗口内容写入文本文件。
Print	使用 Print 对话框打印该窗口的内容。 **注：** 对于具有大容量存储器的器件，打印的页数可能会很多。这种情况下建议将窗口内容打印到一个文件中（Print 对话框，**General** 选项卡，“Print to File”（打印到文件）复选框），然后从该文件中选择需要打印哪些页。
Adjust Table Columns	自动调整列。

在设置为 HEX 或 PSV DATA 格式的存储器窗口数据区域中单击右键可弹出该菜单。并非所有项对于所有器件均可见。

表 12-19. Program Memory 窗口菜单——Hex/PSV Data

项	说明*
Hex Display Width（十六进制显示宽度）	**仅限 Hex 格式** 设置十六进制显示宽度（选项取决于所选器件）。
Fill Memory	使用 Data 中的值从 Start Address 到 End Address 填充存储器。 在 Fill Memory 对话框中指定其他选项。
Go To	转至在 Go To 对话框中指定的地址/函数。
Find	查找在 Find 对话框中指定的文本。
Output To File	使用 Output To File 对话框将显示的窗口内容写入文本文件。
Import Table（导入表）	使用 Import Table 对话框将表格数据从文件导入存储器窗口。
Export Table（导出表）	使用 Export Table 对话框将表格数据从存储器窗口导出到文件中。
Print	使用 Print 对话框打印该窗口的内容。 **注：** 对于具有大容量存储器的器件，打印的页数可能会很多。这种情况下建议将窗口内容打印到一个文件中（Print 对话框，General 选项卡，“Print to File”复选框），然后从该文件中选择需要打印哪些页。
Adjust Table Columns	自动调整列。

12.9.2 File Registers 窗口

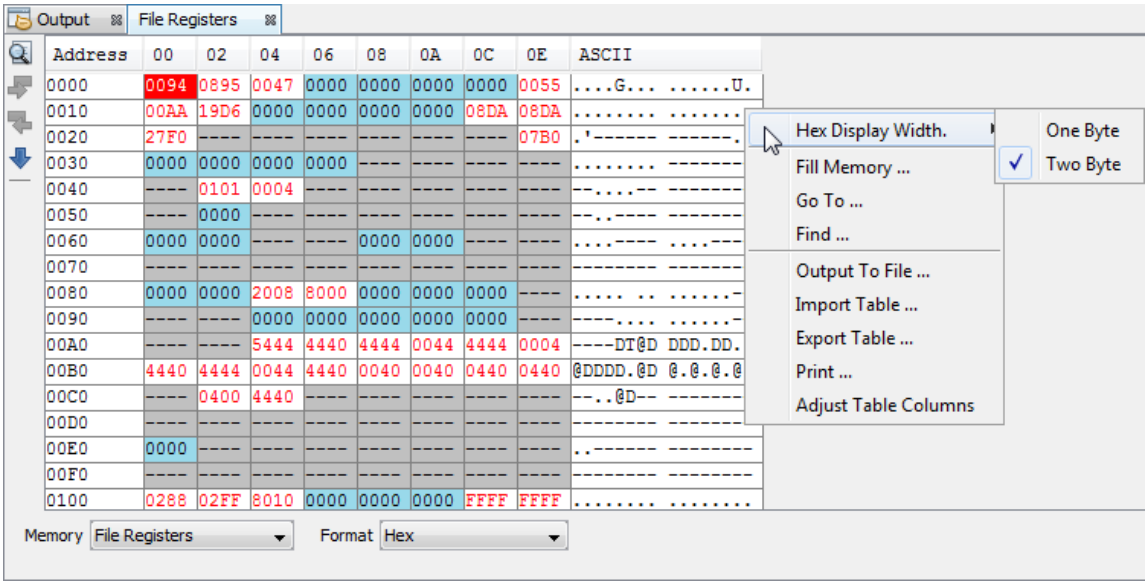
File Registers 窗口会显示选定器件的所有文件寄存器。当文件寄存器值发生更改，或处理器被查询时，File Registers 窗口中的数据会发生更新（更新后的值显示为红色）。

某些数据所在的单元格是彩色的：

- 蓝色单元格代表 SFR。
- 绿色单元格代表项目变量。
- 灰色单元格代表保留的存储空间。

注： 为了加快使用某些硬件工具进行调试的速度，请关闭该窗口。改为使用 SFR 或 Watches 窗口。

图 12-10. 具有内容的 File Register 窗口



12.9.2.1 File Registers 窗口显示

您可以通过从窗口底部的 Format 下拉框中进行选择来指定存储器在窗口中的显示方式。

使用硬件工具进行调试时，某些寄存器的每半字节数据可能会显示“R”，以表示保留的资源。

Hex

该格式将以十六进制数据的形式显示文件寄存器信息。窗口将具有以下列：

- Address——下一列中的数据的十六进制地址。
- Data Blocks——十六进制数据，以 1 字节或 2 字节块的形式显示。
- ASCII——相应数据行的 ASCII 表示形式。

Symbol

该格式将以符号方式显示每个文件寄存器，并以十六进制、十进制、二进制和字符格式显示相应的数据。窗口将具有以下列：

- Address——数据十六进制地址。
- Symbol Name（符号名称）——数据的符号名称。
- Radix Information（基数信息）——Hex、Decimal（十进制）、Binary（二进制）和 Char（字符）这四列中会显示基数信息。十六进制以 1 或 2 字节块的形式显示。

Dual Port（双端口）（仅适用于 dsPIC33FJ DSC/PIC24HJ MCU 器件）

该格式将以十六进制数据的形式显示文件寄存器信息。窗口将具有以下列：

- Address——数据十六进制地址。
- DMA Address（DMA 地址）——相对于芯片 DMA 地址的偏移量。
- Data Blocks——十六进制数据，以 1 字节或 2 字节块的形式显示。
- ASCII——相应数据行的 ASCII 表示形式。

关于 dsPIC33F DSC 和 PIC24H MCU 器件的信息，请参见 Microchip 网站上的器件数据手册以及 dsPIC33F 和 PIC24H 参考手册章节。

XY Data（XY 数据）（仅适用于 dsPIC DSC 器件）

该格式将以十六进制数据的形式显示文件寄存器信息。窗口将具有以下列：

- Address——数据的 X 总线十六进制地址。
- Y Bus（Y 总线）——数据的 Y 总线十六进制地址（如支持）。
- Data Blocks——十六进制数据，以 2 字节块的形式显示。
- ASCII——相应数据行的 ASCII 表示形式。

关于 dsPIC DSC 器件的更多信息，请参见《dsPIC30F 系列参考手册》（DS70046E_CN）。

12.9.2.2 File Registers 窗口图标

图标位于窗口左侧。

表 12-20. File Registers 图标

图标	图标文本	功能
	Find	指定要在窗口中查找的字符串。可选择全字匹配或区分大小写。
	Find Next	查找 Find 的下一个字符串实例。
	Find Previous	查找 Find 的上一个字符串实例。
	Go To	转至指定的行号/地址。

12.9.2.3 File Registers 窗口菜单

在存储器窗口数据区域中单击右键可弹出该菜单。并非所有项对于所有器件均可见。

表 12-21. File Registers 窗口上下文菜单

项	说明
Hex Display Width	**仅限 Hex 格式** 设置十六进制显示宽度（选项取决于所选器件）。
Fill Memory	使用 Data 中的值从 Start Address 到 End Address 填充存储器。在 Fill Memory 对话框中指定其他选项。
Go To	转至在 Go To 对话框中指定的地址/函数。
Find	查找在 Find 对话框中指定的文本。
Output To File	使用 Output To File 对话框将显示的窗口内容写入文本文件。
Import Table	**仅限 Hex、Dual Port 或 XY Data 格式** 使用 Import Table 对话框将表格数据从文件导入存储器窗口。
Export Table	**仅限 Hex、Dual Port 或 XY Data 格式** 使用 Export Table 对话框将表格数据从存储器窗口导出到文件中。
Print	使用 Print 对话框打印该窗口的内容。 **注：** 对于具有大容量存储器的器件，打印的页数可能会很多。这种情况下建议将窗口内容打印到一个文件中（Print 对话框，**General** 选项卡，“Print to File”复选框），然后从该文件中选择需要打印哪些页。

.....（续）

项	说明
Adjust Table Columns	自动调整列。

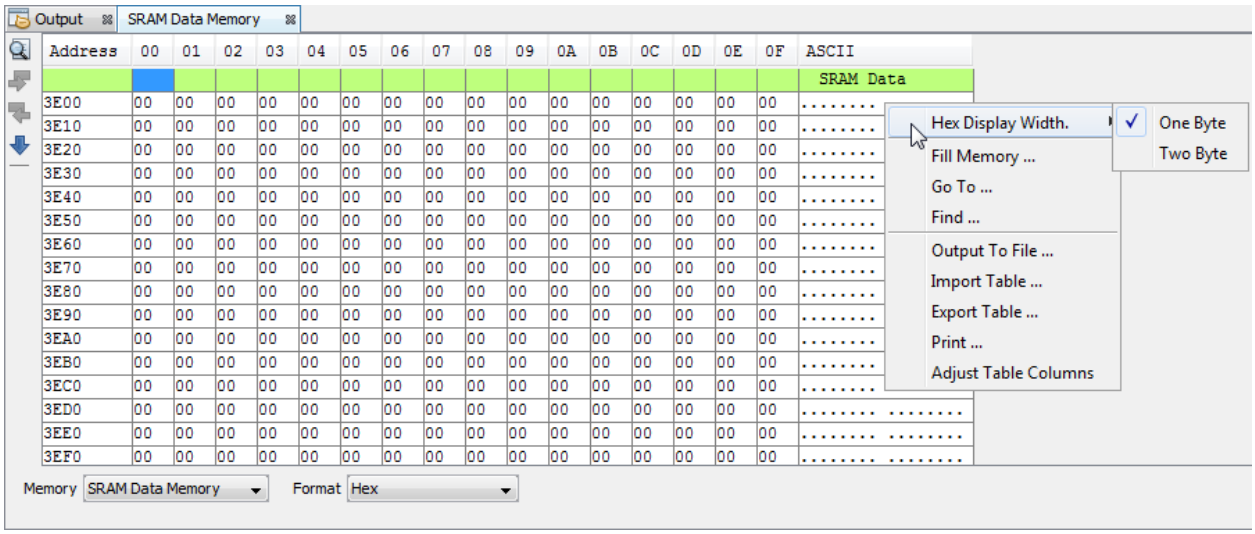
12.9.3 SRAM Data Memory 窗口

SRAM Data Memory（SRAM 数据存储器）窗口显示选定 AT 器件的全部 SRAM。当寄存器值发生更改，或处理器被查询时，该窗口中的数据会发生更新（更新后的值显示为红色）。

使用硬件工具进行调试时，某些寄存器的每半字节数据可能会显示“R”，以表示保留的资源。

注： 为了加快使用某些硬件工具进行调试的速度，请关闭该窗口。改用 I/O Memory (SFRs)（I/O 存储器（SFRs））窗口或 Watches 窗口。

图 12-11. 具有内容的 SRAM Data Memory 窗口



12.9.3.1 SRAM Data Memory 窗口显示

您可以通过从窗口底部的 Format 下拉框中进行选择来指定存储器在窗口中的显示方式。

Hex

该格式将以十六进制数据的形式显示文件寄存器信息。窗口将具有以下列：

- Address——下一列中的数据的十六进制地址。
- Data Blocks——十六进制数据，以 1 字节或 2 字节块的形式显示。
- ASCII——相应数据行的 ASCII 表示形式。

Symbol

该格式将以符号方式显示每个文件寄存器，并以十六进制、十进制、二进制和字符格式显示相应的数据。窗口将具有以下列：

- Address——数据十六进制地址。
- Symbol Name——数据的符号名称。
- Radix Information——Hex、Decimal、Binary 和 Char。

这四列中会显示基数信息。十六进制以 1 或 2 字节块的形式显示。

12.9.3.2 SRAM Data Memory 窗口图标

图标位于窗口左侧。

表 12-22. SRAM Data Memory 窗口图标

图标	图标文本	功能
	Find	指定要在窗口中查找的字符串。可选择全字匹配或区分大小写。
	Find Next	查找 Find 的下一个字符串实例。
	Find Previous	查找 Find 的上一个字符串实例。
	Go To	转至指定的行号/地址。

12.9.3.3 SRAM Data Memory 窗口菜单

在存储器窗口数据区域中单击右键可弹出该菜单。并非所有项对于所有器件均可见。

表 12-23. SRAM Data Memory 窗口上下文菜单

项	说明
Hex Display Width	**仅限 Hex 格式** 设置十六进制显示宽度（选项取决于所选器件）。
Fill Memory	使用 Data 中的值从 Start Address 到 End Address 填充存储器。在 Fill Memory 对话框中指定其他选项。
Go To	转至在 Go To 对话框中指定的地址/函数。
Find	查找在 Find 对话框中指定的文本。
Output To File	使用 Output To File 对话框将显示的窗口内容写入文本文件。
Import Table	**仅限 Hex 格式** 使用 Import Table 对话框将表格数据从文件导入存储器窗口。
Export Table	**仅限 Hex 格式** 使用 Export Table 对话框将表格数据从存储器窗口导出到文件中。
Print	使用 Print 对话框打印该窗口的内容。 **注：** 对于具有大容量存储器的器件，打印的页数可能会很多。这种情况下建议将窗口内容打印到一个文件中（Print 对话框，**General** 选项卡，“Print to File”复选框），然后从该文件中选择需要打印哪些页。
Adjust Table Columns	自动调整列。

12.9.4 SFRs 窗口

特殊功能寄存器（SFRs）窗口会显示所选处理器的 SFR 的内容。对于查看 SFR，该窗口提供的格式会比常规文件寄存器窗口更有用，因为其中会包含每个 SFR 名称，并提供了几种数字格式。要仅查看几个 SFR，您可能会希望使用 Watches 窗口，该窗口可以帮助解决使用硬件调试工具时的速度问题（即，窗口更新速率更快）。

每次发生中断时，特殊功能寄存器的内容都会发生更新。

使用硬件工具进行调试时，某些寄存器的每半字节数据可能会显示“R”，以表示保留的资源。

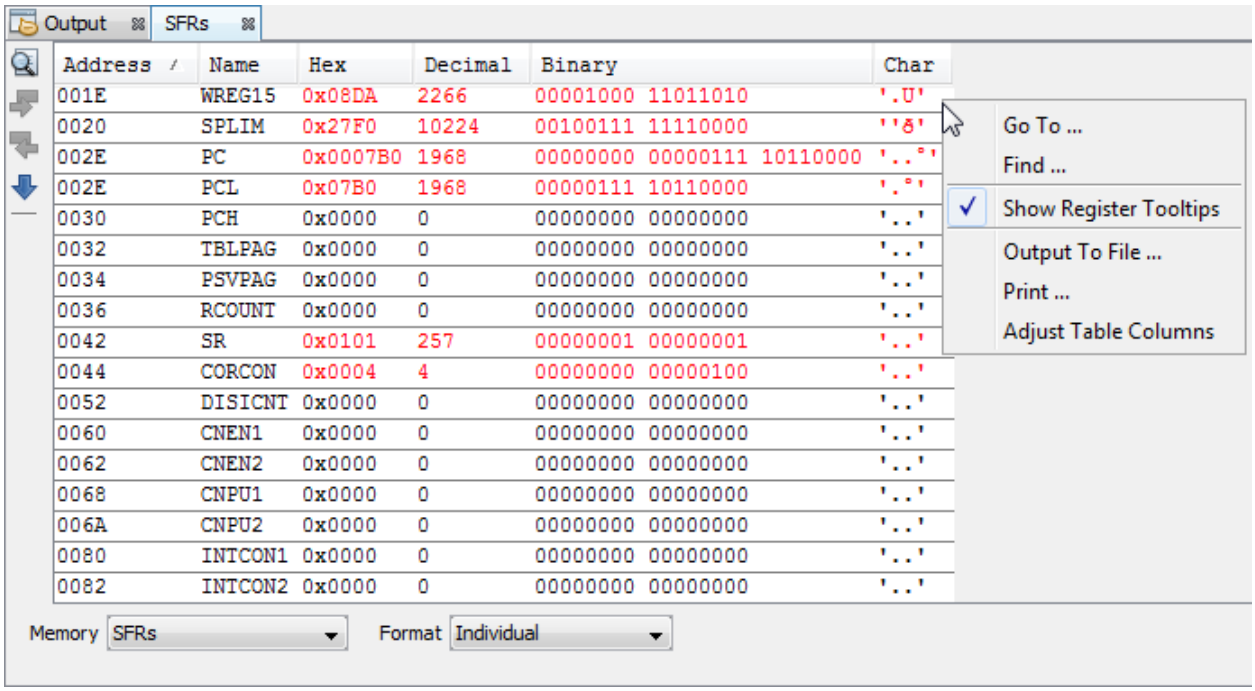
可见的寄存器

如果某个数据存储器寄存器在器件上未物理实现，则它可能不会出现在 SFR 列表中。某些工具（如软件模拟器）可能会让您看到实际器件上不存在的寄存器，如预分频器。

单步执行

如果选择了“Freeze Peripherals On Halt”（暂停时冻结外设），则在单步执行时 SFR 或 Watches 窗口中的 I/O 端口位不会发生更新。引脚会被修改，但获取新值的读请求会由于冻结而被阻止，无法进行更新，直到下一个单步或运行命令为止。

图 12-12. 具有内容的 SFRs 窗口



12.9.4.1 SFRs 窗口显示

您可以通过从窗口底部的 Format 下拉框中进行选择来指定存储器在窗口中的显示方式，具体包括以下两种选项：

- Individual（独立）——同时查看所有 SFR 寄存器。
- Peripherals——查看功能组中的 SFR 寄存器。

Individual

在这种显示方式中，SFR 按地址列出。

数据显示在以下各列中。

- Address——SFR 十六进制地址。
- Name（名称）——SFR 的符号名称。
- Radix Information——Hex、Decimal、Binary 和 Char
  这四列中会显示基数信息。十六进制以 1 字节块的形式显示。

Peripheral

在这种显示方式中，SFR 按其相关的器件外设分组。

数据显示在以下各列中：

- Address——SFR 十六进制地址。
- Name——外设的名称或 SFR 的符号名称。
- Radix Information——Hex、Decimal、Binary 和 Char
  这四列中会显示基数信息。十六进制以 1 字节块的形式显示。

单击 **Filter Peripheral**（筛选外设）按钮可仅查看选定外设的 SFR。

12.9.4.2 SFRs 窗口图标

图标位于窗口左侧。

表 12-24. SFRs 窗口图标

图标		功能
	Find	指定要在窗口中查找的字符串。可选择全字匹配或区分大小写。
	Find Next	查找 Find 的下一个字符串实例。
	Find Previous	查找 Find 的上一个字符串实例。
	Go To	转至指定的行号/地址。
	Filter Peripherals（筛选外设）	**仅限 Peripheral 格式** 对视图中显示的外设进行筛选。 单击该图标可打开 Select 对话框。可通过按住 Shift 并单击（范围）或按住 Ctrl 并单击（单个）的方式选择外设。
	Filter Select All Peripherals（筛选所有外设）	**仅限 Peripheral 格式** 筛选所有外设。

12.9.4.3 SFRs 窗口菜单

在存储器窗口数据区域中单击右键可弹出该菜单。

表 12-25. SFRs 窗口上下文菜单

项	说明*
Find	查找在 Find 对话框中指定的文本。
Output To File	使用 Output To File 对话框将显示的窗口内容写入文本文件。
Print	使用 Print 对话框打印该窗口的内容。 **注：** 对于具有大容量存储器的器件，打印的页数可能会很多。这种情况下建议将窗口内容打印到一个文件中（Print 对话框，**General** 选项卡，“Print to File”复选框），然后从该文件中选择需要打印哪些页。
Adjust Table Columns	自动调整列。

12.9.5 I/O Memory (SFRs)窗口

I/O Memory 窗口显示所选 AT 处理器的特殊功能寄存器（SFR）的内容。对于查看 SFR，该窗口提供的格式会比常规 SRAM Data Memory 窗口更有用，因为其中会包含每个 SFR 名称，并提供了几种数字格式。要仅查看几个 SFR，您可能会希望使用 Watches 窗口，该窗口可以帮助解决使用硬件调试工具时的速度问题（即，窗口更新速率更快）。

每次发生中断时，特殊功能寄存器的内容都会发生更新。

使用硬件工具进行调试时，某些寄存器的每半字节数据可能会显示“R”，以表示保留的资源。

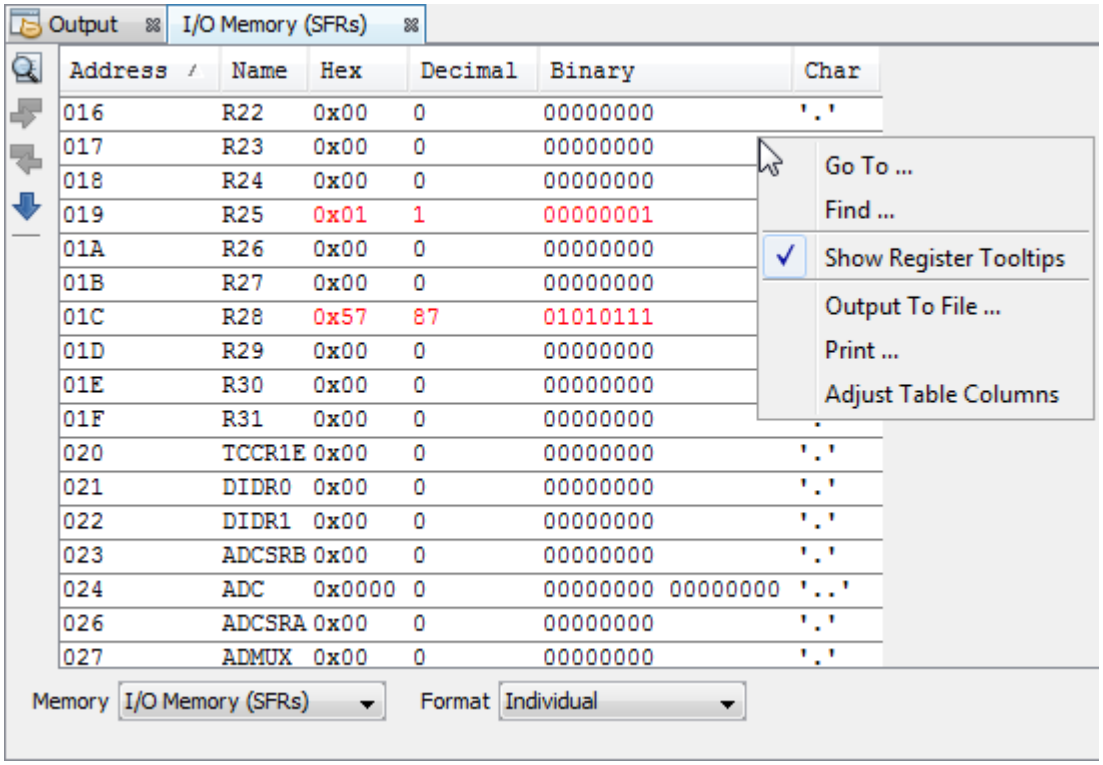
可见的寄存器

如果某个数据存储器寄存器在器件上未物理实现，则它可能不会出现在 SFR 列表中。某些工具（如软件模拟器）可能会让您看到实际器件上不存在的寄存器。

单步执行

如果选择了“Freeze Peripherals On Halt”，则在单步执行时 SFR 或 Watches 窗口中的 I/O 端口位不会发生更新。引脚会被修改，但获取新值的读请求会由于冻结而被阻止，无法进行更新，直到下一个单步或运行命令为止。

图 12-13. 具有内容的 I/O Memory 窗口



12.9.5.1 I/O Memory 窗口显示

您可以通过从窗口底部的 Format 下拉框中进行选择来指定存储器在窗口中的显示方式。

Individual

在这种显示方式中，SFR 按地址列出。

数据显示在以下各列中。

- Address——SFR 十六进制地址。
- Name——SFR 的符号名称。
- Radix Information——Hex、Decimal、Binary 和 Char。
  这四列中会显示基数信息。十六进制以 1 字节块的形式显示。

Peripheral

在这种显示方式中，SFR 按其相关的器件外设分组。

数据显示在以下各列中：

- Address——SFR 十六进制地址。
- Name——外设的名称或 SFR 的符号名称。
- Radix Information——Hex、Decimal、Binary 和 Char。
  这四列中会显示基数信息。十六进制以 1 字节块的形式显示。

单击 **Filter Peripheral** 图标可仅查看选定外设的 SFR。

CPU Registers（CPU 寄存器）

在这种显示方式中，CPU 寄存器（PC 和工作寄存器 Rn）按地址列出。

数据显示在以下各列中：

- Address——SFR 十六进制地址。
- Name——外设的名称或 SFR 的符号名称。

- Radix Information——Hex、Decimal、Binary 和 Char。
  这四列中会显示基数信息。十六进制以 1 字节块的形式显示。

12.9.5.2 I/O Memory 窗口图标

图标位于窗口左侧。

表 12-26. I/O Memory 窗口图标

图标		功能
	Find	指定要在窗口中查找的字符串。可选择全字匹配或区分大小写。
	Find Next	查找 Find 的下一个字符串实例。
	Find Previous	查找 Find 的上一个字符串实例。
	Go To	转至指定的行号/地址。
	Filter Peripherals	**仅限 Peripheral 格式** 对视图中显示的外设进行筛选。 单击该图标可打开 Select 对话框。可通过按住 Shift 并单击（范围）或按住 Ctrl 并单击（单个）的方式选择外设。
	Filter Select All Peripherals	**仅限 Peripheral 格式** 筛选所有外设。

12.9.5.3 I/O Memory 窗口菜单

在存储器窗口数据区域中单击右键可弹出该菜单。

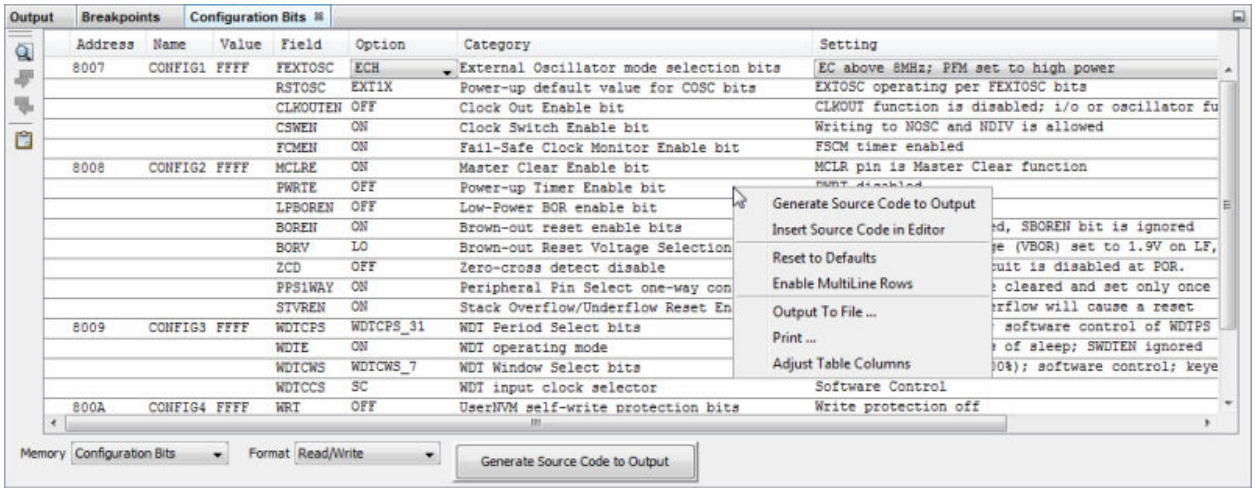
表 12-27. I/O Memory 窗口上下文菜单

项	说明
Go To	转至在 Go To 对话框中指定的地址/函数。
Find	查找在 Find 对话框中指定的文本。
Show Register Tooltips（显示寄存器工具提示）	显示或隐藏寄存器的工具提示。将鼠标悬停在寄存器名称上可弹出相关信息。
Output To File	使用 Output To File 对话框将显示的窗口内容写入文本文件。
Print	使用 Print 对话框打印该窗口的内容。 **注：** 对于具有大容量存储器的器件，打印的页数可能会很多。这种情况下建议将窗口内容打印到一个文件中（Print 对话框，**General** 选项卡，“Print to File”复选框），然后从该文件中选择需要打印哪些页。
Adjust Table Columns	自动调整列。

12.9.6 Configuration Bits 窗口

4.19 在 Configuration Bits 窗口中设置配置值介绍了关于使用 Configuration Bits 窗口的详细信息。

图 12-14. 具有内容的 Configuration Bits 窗口



12.9.6.1 Configuration Bits 窗口显示

您可以通过从窗口底部的 Format 下拉框中进行选择来指定存储器在窗口中的显示方式。可用的格式取决于您的器件。

当您对窗口中的配置位设置满意时，可以单击 **Generate Source Code to Output** 按钮，将特定于器件的配置代码写入 Output 窗口。然后，您可以在编辑器窗口中将该代码剪切并粘贴到您的应用程序中。

表 12-28. Configuration Bits 窗口显示

列标题	定义
Address	配置字/字节的地址。
Name	配置寄存器的名称。
Value	配置字/字节的当前值。
Field*	对于代码中设置的配置位，宏的字段部分。 例如，WDTE 是宏_WDTE_OFF 的字段部分。
Option*	对于代码中设置的配置位，宏的选项部分。 例如，OFF 是宏_WDTE_OFF 的选项部分。
Category	相应配置字/字节中的配置位的名称。
Setting	配置位的当前设置。使用下拉列表可更改设置。配置字/字节的值将相应地发生更改。
*并非所有器件都支持。	

12.9.6.2 Configuration Bits 窗口图标

图标位于窗口左侧。

表 12-29. Configuration Bits 窗口图标

图标	图标文本	功能
	Refresh by Read Device Memory	与调试工具栏“Read Device Memory”图标的功能相同——将器件存储器的内容上传到 MPLAB X IDE。
	Find	指定要在窗口中查找的字符串。可选择全字匹配或区分大小写。
	Find Next	查找 Find 的下一个字符串实例。

.....（续）

图标	图标文本	功能
	Find Previous	查找 Find 的上一个字符串实例。
	Paste	将生成的源代码粘贴到编辑器中的光标处。 如果编辑器未处于焦点，则 Output 窗口将显示警告，生成的源代码将置于该窗口中。

12.9.6.3 Configuration Bits 窗口菜单

在存储器窗口数据区域中单击右键可弹出该菜单。并非所有项对于所有 MCU 均可见。

表 12-30. Configuration Bits 窗口菜单

项	说明
Generate Source Code to Output	根据需要在窗口中设置配置位。然后选择该选项以在 Output 窗口中生成代码，您可以将其剪切并粘贴到程序中以设置配置位。 与窗口上的按钮功能相同。
Insert Source Code to Editor	根据需要在窗口中设置配置位。然后选择该选项以在编辑器窗口中生成代码，您可以将其剪切并粘贴到程序中以设置配置位。
Reset to Defaults	将所有配置位的值复位为其 MPLAB X IDE 默认值。
Enable Multiline Rows	如果列的长度短于内容的长度（例如，Category），则允许换行显示内容。否则，内容将被截断，并用省略号（...）表示。
Output To File	使用 Output To File 对话框将显示的窗口内容写入文本文件。
Print	使用 Print 对话框打印该窗口的内容。 **注：** 对于具有大容量存储器的器件，打印的页数可能会很多。这种情况下建议将窗口内容打印到一个文件中（Print 对话框，**General** 选项卡，“Print to File”复选框），然后从该文件中选择需要打印哪些页。
Adjust Table Columns	自动调整列。

12.9.7 EE Data Memory 窗口

对于任何具有 EEPROM 数据存储器的单片机器件（如 PIC16F1829），EEPROM 窗口会显示 EEPROM 数据。其中会显示所选器件的数据/操作码十六进制信息。

对于 MPLAB X 软件模拟器，当 EEPROM 寄存器值发生更改或处理器暂停时，EEPROM 窗口中的数据会发生更新。

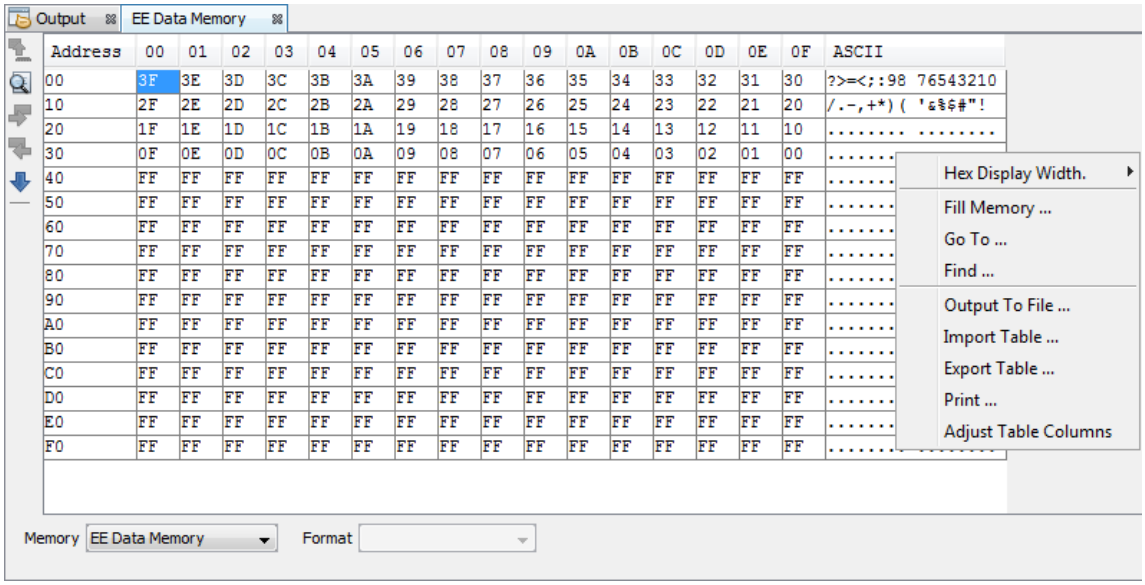
对于任何 Microchip 硬件调试工具（例如，MPLAB PICkit 4 在线调试器），当某个 EEPROM 寄存器值发生更改或处理器暂停时，EEPROM 窗口中的数据**不会**发生更新；您必须对器件存储器执行调试时读取操作（器件/调试头支持该功能时），或者必须先退出调试会话再读取器件数据。

要与编程器配合使用，需要指定 EEPROM 数据存储器的起始地址。下表列出了一些通用值，但请查看您选择的器件的编程规范，以确定正确的地址。

表 12-31. 编程器——数据 EEPROM 起始地址

器件	通用起始地址
中档 MCU	0x2100
增强型中档 MCU	0x1E000
PIC18F MCU	0xF00000
PIC24 MCU 和 dsPIC DSC	0x7FFE00

图 12-15. 具有内容的 EE Data Memory 窗口



12.9.7.1 EE Data Memory 窗口显示

这种显示格式会在以下各列中显示数据：

- Address——下一列中的数据的十六进制地址。
- Data Blocks——十六进制数据，以 1、2 或 4 字节块的形式显示，可从菜单中选择。
- ASCII——相应数据行的 ASCII 表示形式。

12.9.7.2 EE Data Memory 窗口图标

图标位于窗口左侧。

表 12-32. EE Data Memory 窗口图标

图标	图标文本	功能
	Refresh by Read Device Memory	与调试工具栏“Read Device Memory”图标的功能相同——将器件存储器的内容上传到 MPLAB X IDE。
	Find	指定要在窗口中查找的字符串。可选择全字匹配或区分大小写。
	Find Next	查找 Find 的下一个字符串实例。
	Find Previous	查找 Find 的上一个字符串实例。
	Go To	转至指定的行号/地址。

12.9.7.3 EE Data Memory 窗口菜单

在存储器窗口数据区域中单击右键可弹出该菜单，如下所示。

表 12-33. EE Data Memory 窗口上下文菜单

项	说明*
Hex Display Width	**仅限 Hex 格式** 设置十六进制显示宽度。 （选项取决于所选器件）。

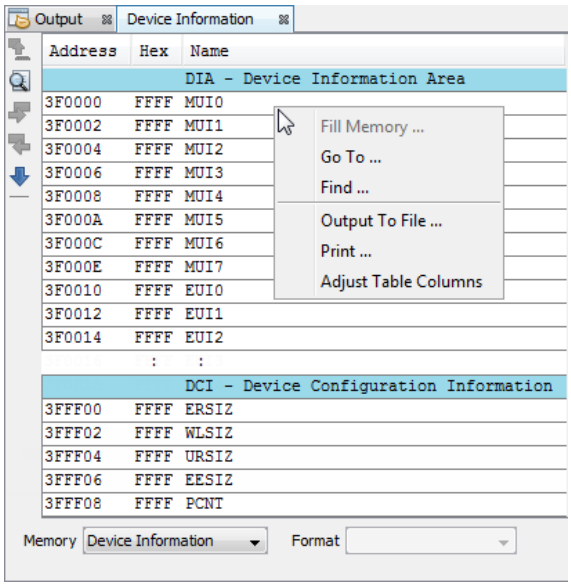
.....（续）

项	说明*
Fill Memory	使用 Data 中的值从 Start Address 到 End Address 填充存储器。 在 Fill Memory 对话框中指定其他选项。
Go To	转至在 Go To 对话框中指定的地址/函数。
Find	查找在 Find 对话框中指定的文本。
Output To File	使用 Output To File 对话框将显示的窗口内容写入文本文件。
Import Table	使用 Import Table 对话框将表格数据从文件导入存储器窗口。
Export Table	使用 Export Table 对话框将表格数据从存储器窗口导出到文件中。
Print	使用 Print 对话框打印该窗口的内容。 **注：** 对于具有大容量存储器的器件，打印的页数可能会很多。这种情况下建议将窗口内容打印到一个文件中（Print 对话框，General 选项卡，“Print to File”复选框），然后从该文件中选择需要打印哪些页。
Adjust Table Columns	自动调整列。

12.9.8 Device Information 窗口

Device Information（器件信息）窗口显示支持该存储区的器件（当前为 PIC18F24K42 和众多 PIC16Fxxxxx 器件）的存储器的器件信息区域内容。DIA 包含内部温度指示器模块的校准数据，存储 Microchip 惟一标识符字和以 mV 为单位的“固定参考电压”电压读数。有关详细信息，请参见器件数据手册。

图 12-16. Device Information 窗口



12.9.8.1 Device Information 窗口显示

这种显示格式会在以下各列中显示数据：

- Address——下一列中的数据的十六进制地址。
- Hex——十六进制数据，以 1、2 或 4 字节块的形式显示，可从菜单中选择。
- Name——相应数据行的说明。

12.9.8.2 Device Information 窗口图标

图标位于窗口左侧。

表 12-34. Device Information 窗口图标

图标	图标文本	功能
	Refresh by Read Device Memory	与调试工具栏“Read Device Memory”图标的功能相同——将器件存储器的内容上传到 MPLAB X IDE。
	Find	指定要在窗口中查找的字符串。可选择全字匹配或区分大小写。
	Find Next	查找 Find 的下一个字符串实例。
	Find Previous	查找 Find 的上一个字符串实例。
	Go To	转至指定的行号/地址。

12.9.8.3 Device Information 窗口菜单

在存储器窗口数据区域中单击右键可弹出该菜单。并非所有项对于所有器件均可见。

表 12-35. Device Information 窗口上下文菜单

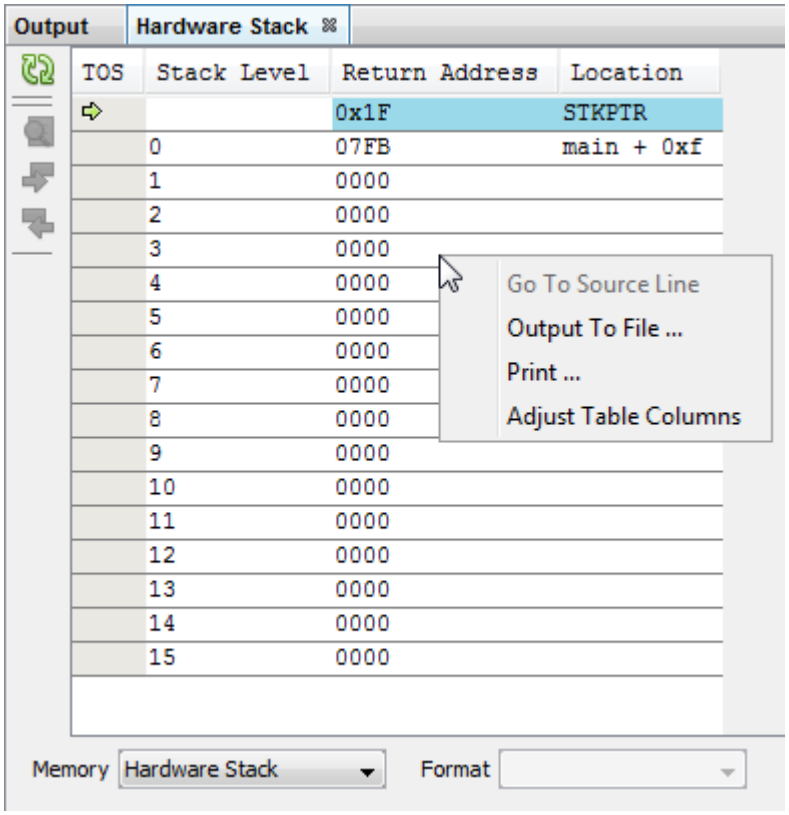
项	说明
Fill Memory	使用 Data 中的值从 Start Address 到 End Address 填充存储器。 在 Fill Memory 对话框中指定其他选项。
Go To	转至在 Go To 对话框中指定的地址/函数。
Find	查找在 Find 对话框中指定的文本。
Output To File	使用 Output To File 对话框将显示的窗口内容写入文本文件。
Print	使用 Print 对话框打印该窗口的内容。 **注：** 对于具有大容量存储器的器件，打印的页数可能会很多。这种情况下建议将窗口内容打印到一个文件中（Print 对话框，**General** 选项卡，“Print to File”复选框），然后从该文件中选择需要打印哪些页。
Adjust Table Columns	自动调整列。

12.9.9 Hardware Stack 窗口

暂停时查看硬件堆栈的内容。

要打开 Hardware Stack（硬件堆栈）窗口，请选择 *Window>PIC Memory Views>Hardware Stack*（窗口>PIC 存储器视图>硬件堆栈）。第一堆栈级别由“0”表示。

图 12-17. 具有内容的 Hardware Stack 窗口



12.9.9.1 Hardware Stack 窗口显示

堆栈深度取决于器件。这种显示格式会在以下各列中显示数据：

- TOS——栈顶当前位置。
- Stack Level（堆栈级别）——项在堆栈中的位置。
- Return Address（返回地址）——从堆栈返回的十六进制地址。
- Location——返回地址代码的反汇编。

使用硬件工具进行调试时，某些寄存器的每半字节数据可能会显示“R”，以表示保留的资源。

12.9.9.2 Hardware Stack 窗口图标

暂停或停止时窗口中提供以下图标。

表 12-36. Hardware Stack 窗口图标

图标	图标文本	说明
	Auto Refresh（自动刷新） Manual Refresh（手动刷新）	自动或手动刷新窗口内容。 取决于 *Tools>Options>Embedded>Generic Settings* 下的“Disable auto refresh for call stack view during debug sessions”的值——未选中 = 假（默认），选中 = 真。请参见 12.16.1 Generic Settings 选项卡。 假 = 自动刷新（绿色图标）：暂停或停止时自动更新窗口内容。如果窗口关闭或未处于焦点，则选中窗口并单击该按钮即可显示更新内容，无需再次运行并暂停/停止。 真 = 手动刷新（橙色图标）：暂停或停止时不自动更新窗口内容。必须单击该按钮才能在暂停/停止时更新窗口内容。
	Find	指定要在窗口中查找的字符串。可选择全字匹配或区分大小写。

.....（续）

图标	图标文本	说明
	Find Next	查找 Find 的下一个字符串实例。
	Find Previous	查找 Find 的上一个字符串实例。

12.9.9.3 Hardware Stack 窗口菜单

在存储器窗口数据区域中单击右键可弹出该菜单。

表 12-37. Hardware Stack 窗口上下文菜单

项	说明*
Go To Source Line	转至编辑器中的相应源代码行。
Output To File	使用 Output To File 对话框将显示的窗口内容写入文本文件。
Print	使用 Print 对话框打印该窗口的内容。 **注：** 对于具有大容量存储器的器件，打印的页数可能会很多。这种情况下建议将窗口内容打印到一个文件中（Print 对话框，**General** 选项卡，“Print to File”复选框），然后从该文件中选择需要打印哪些页。
Adjust Table Columns	自动调整列。

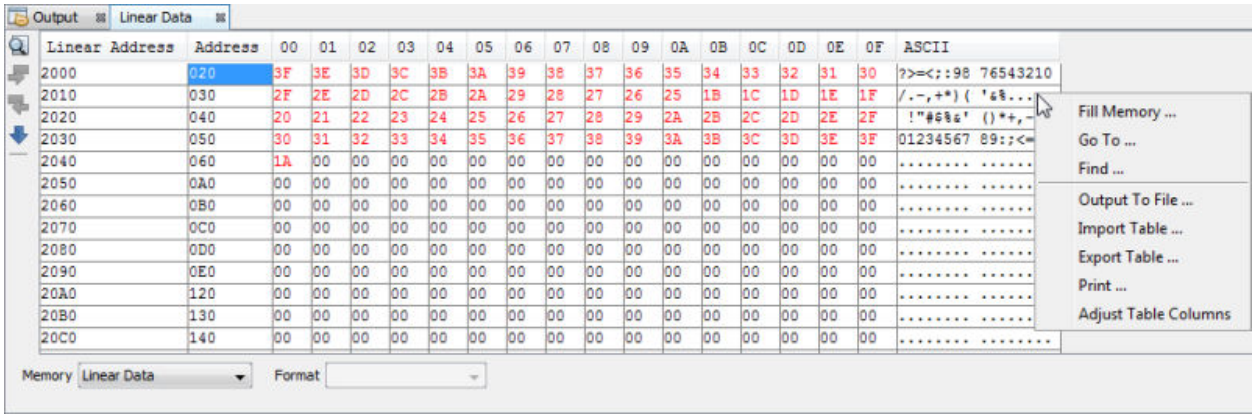
12.9.10 Linear Data 窗口

Linear Data（线性数据）窗口显示具有此类存储器的任何单片机器件的闪存数据。其中会显示所选器件的数据/操作码十六进制信息。

对于 MPLAB X 软件模拟器，当线性数据寄存器值发生更改或处理器暂停时，Linear Data 窗口中的数据会发生更新。

对于任何 Microchip 硬件调试工具（例如，MPLAB ICD 4 在线调试器），当某个线性数据寄存器值发生更改或处理器暂停时，Linear Data 窗口中的数据**不会**发生更新；您必须对器件存储器执行调试时读取操作（器件/调试头支持该功能时），或者必须先退出调试会话再读取器件数据。

图 12-18. 具有内容的 Linear Data 窗口



12.9.10.1 Linear Data 窗口显示

这种显示格式会在以下各列中显示数据：

- Address——下一列中的数据的十六进制地址。
- Data Blocks——十六进制数据，以 1、2 或 4 字节块的形式显示，可从菜单中选择。

- ASCII——相应数据行的 ASCII 表示形式。

12.9.10.2 Linear Data 窗口图标

图标位于窗口左侧。

表 12-38. Linear Data 窗口图标

图标	图标文本	功能
	Find	指定要在窗口中查找的字符串。可选择全字匹配或区分大小写。
	Find Next	查找 Find 的下一个字符串实例。
	Find Previous	查找 Find 的上一个字符串实例。
	Go To	转至指定的行号/地址。

12.9.10.3 Linear Data 窗口菜单

在存储器窗口数据区域中单击右键可弹出该菜单，如下表所示。

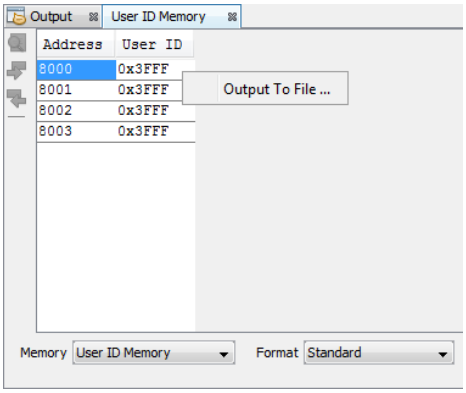
表 12-39. Linear Data 窗口上下文菜单

项	说明*
Fill Memory	使用 Data 中的值从 Start Address 到 End Address 填充存储器。 在 Fill Memory 对话框中指定其他选项。
Go To	转至在 Go To 对话框中指定的地址/函数。
Find	查找在 Find 对话框中指定的文本。
Output To File	使用 Output To File 对话框将显示的窗口内容写入文本文件。
Import Table	使用 Import Table 对话框将表格数据从文件导入存储器窗口。
Export Table	使用 Export Table 对话框将表格数据从存储器窗口导出到文件中。
Print	使用 Print 对话框打印该窗口的内容。 **注：** 对于具有大容量存储器的器件，打印的页数可能会很多。这种情况下建议将窗口内容打印到一个文件中（Print 对话框，**General** 选项卡，“Print to File”复选框），然后从该文件中选择需要打印哪些页。
Adjust Table Columns	自动调整列。

12.9.11 User ID Memory 窗口

一些器件具有可用于存储校验和或其他代码标识（ID）数字的存储单元。在编程/校验期间，这些存储单元是可读写的。根据不同的器件，它们可能还可以在正常执行期间通过 TBLRD 和 TBLWT 指令访问。

图 12-19. 具有内容的 User ID Memory 窗口

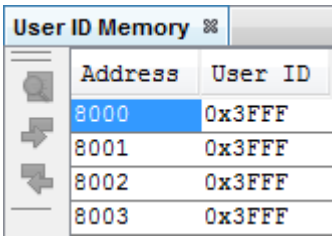


12.9.11.1 User ID Memory 窗口显示

要确定可在此处输入的值，请参见器件编程规范。对于大多数器件，这会设置器件 ID 字的低半字节；高半字节设置为“0”。高半字节只能通过编程方式写入，例如通过使用表写操作。

您可以通过从窗口底部的 Format 下拉框中进行选择来指定存储器在窗口中的显示方式。

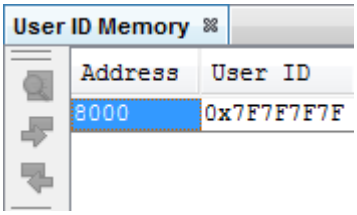
图 12-20. 标准显示



在标准显示中，数据显示在以下列中。

- Address——用户 ID 十六进制地址。
- User ID——用户 ID 存储器的内容（以十六进制表示）。

图 12-21. 传统显示



在传统显示中，数据显示在以下列中。

- Address——用户 ID 十六进制起始地址。
- User ID——用户 ID 存储器的内容（以十六进制表示）。

12.9.11.2 User ID Memory 窗口图标

图标位于窗口左侧。

表 12-40. User ID Memory 窗口图标

图标	图标文本	功能
	Find	指定要在窗口中查找的字符串。可选择全字匹配或区分大小写。
	Find Next	查找 Find 的下一个字符串实例。

.....（续）		
图标	图标文本	功能
	Find Previous	查找 Find 的上一个字符串实例。

12.9.11.3 User ID Memory 窗口菜单

在存储器窗口数据区域中单击右键可弹出该菜单，如下表所示。

表 12-41. User ID Memory 窗口上下文菜单

项	说明*
Output To File	使用 Output To File 对话框将显示的窗口内容写入文本文件。

12.10 存储器窗口——32 位器件

存储器窗口（*Window>Target Memory Views*）会显示许多器件存储器类型，例如 SFR 和配置位。使用“Memory”和“Format”下拉框可以定制您的窗口。

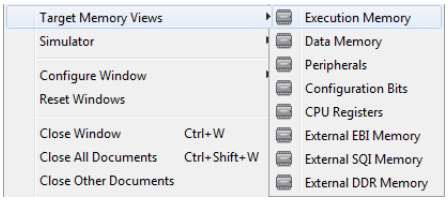
关于这些控件的更多信息，请参见 4.18 查看或更改器件存储器。

关于相关对话框的详细信息，请参见 12.11 与存储器窗口关联的对话框。

注： 对于 8 位、16 位和 32 位器件，Configuration Bits 窗口和 User ID 窗口相似。惟一的区别是窗口中的值反映了器件可用的程序存储器。

图 12-22. Window>Target Memory Views——PIC32MX795F512L

图 12-23. Window>Target Memory Views——ATSAME70Q21B



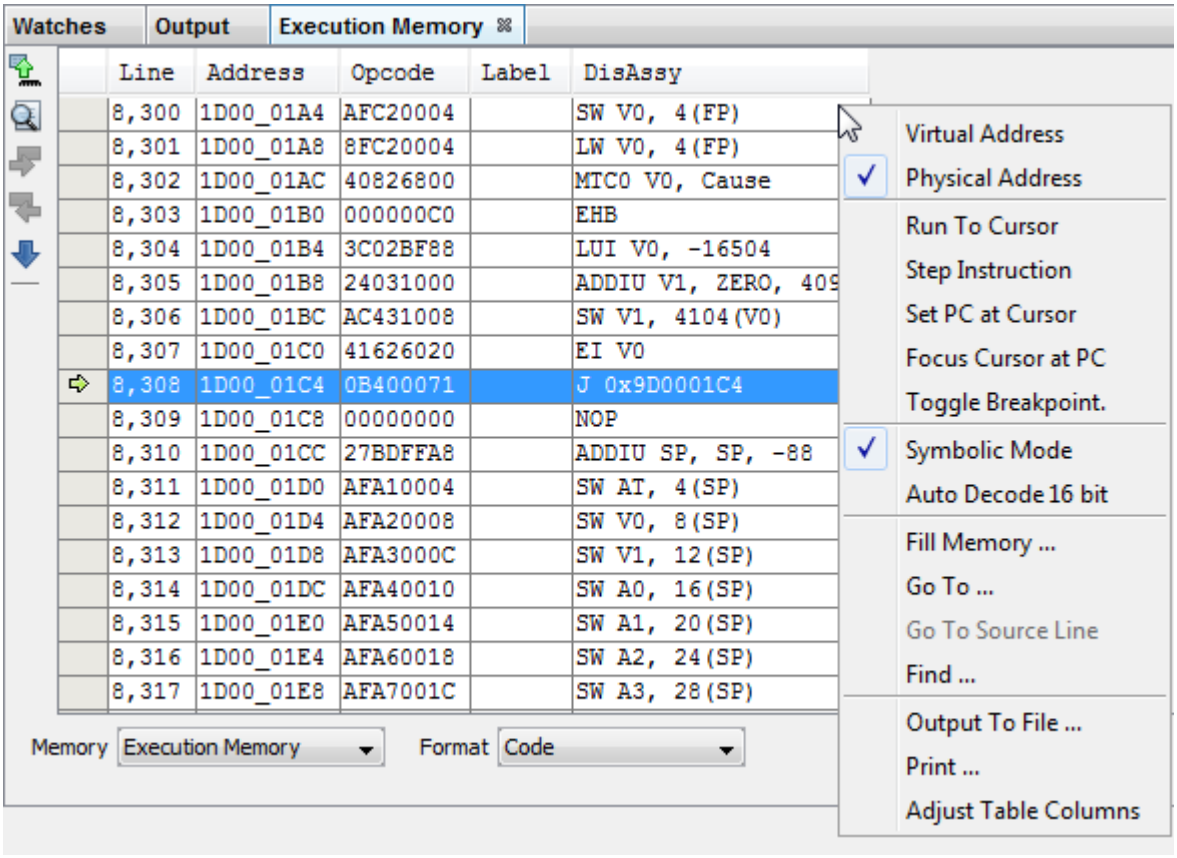
12.10.1 Execution Memory 窗口

Execution Memory 窗口会显示处于当前所选 32 位器件的程序和/或数据存储器范围内的存储单元。

对于 MPLAB X 软件模拟器，当执行存储器值发生更改或处理器暂停时，Execution Memory 窗口中的数据会发生更新。

对于任何 Microchip 硬件调试工具（例如，MPLAB ICD 4 在线仿真器），当某个执行存储器值发生更改或处理器暂停时，Execution Memory 窗口中的数据**不会**发生更新；您必须对器件存储器执行读操作。

图 12-24. 具有内容的 Execution Memory 窗口



12.10.1.1 Execution Memory 窗口显示

您可以通过从窗口底部的 Format 下拉框中进行选择来指定存储器在窗口中的显示方式。

使用硬件工具进行调试时，某些寄存器的每半字节数据可能会显示“R”，以表示保留的资源。

Code 格式

该格式将以十六进制代码的形式显示执行存储器信息。窗口将具有以下列：

- Line——对应于存储器地址的参考行号。
- Address——操作码的物理十六进制地址。
- Opcode——十六进制操作码，以 4 字节块的形式显示。高亮显示的操作码代表程序计数器的当前位置。
- Label——以符号格式显示的操作码标号。
- Disassembly——操作码助记符的反汇编版本。

Data 格式

该窗口将具有以下列：

- Address——下一列中的数据的十六进制地址。
- Data Blocks——十六进制数据，以 4 字节块的形式显示。
- ASCII——相应数据行的 ASCII 表示形式。

12.10.1.2 Execution Memory 窗口图标

图标位于窗口左侧。

表 12-42. Execution Memory 窗口图标

图标	图标文本	功能
	Refresh by Read Device Memory	与调试工具栏“Read Device Memory”图标的功能相同——将器件存储器的内容上传到 MPLAB X IDE。
	Find	指定要在窗口中查找的字符串。可选择全字匹配或区分大小写。
	Find Next	查找 Find 的下一个字符串实例。
	Find Previous	查找 Find 的上一个字符串实例。
	Go To	转至指定的行号/地址。

12.10.1.3 Execution Memory 窗口菜单

在设置为 CODE 格式的存储器窗口数据区域中单击右键可弹出该菜单。并非所有项对于所有 32 位 MCU 均可见。

表 12-43. Execution Memory 窗口上下文菜单——Code 格式

项	说明
Virtual Address (KSEG[0:1])（虚拟地址（KSEG[0:1]））	选择要显示的虚拟地址。有关详细信息，请参见器件数据手册。
Physical Address（物理地址）	选择要显示的物理地址。
Run to Cursor	将程序运行至当前光标位置。
Step Instruction	执行单步指令。
Set PC at Cursor	将程序计数器（PC）设置在光标位置。
Focus Cursor at PC	将光标移动到当前 PC 地址处，并使该地址在窗口中居中。
Toggle Breakpoint	翻转（开启/关闭）现有断点。
Symbolic Mode	显示反汇编的十六进制代码和符号。
Auto Decode 16 bit（自动解码 16 位）	使用 MIP16/microMIPS 指令解码。有关详细信息，请参见器件数据手册。
Fill Memory	使用 Data 中的值从 Start Address 到 End Address 填充存储器。 在 Fill Memory 对话框中指定其他选项。
Go To	转至在 Go To 对话框中指定的地址/函数。
Go To Source Line	转至编辑器中的相应源代码行。
Find	查找在 Find 对话框中指定的文本。
Output To File	使用 Output To File 对话框将显示的窗口内容写入文本文件。
Print	使用 Print 对话框打印该窗口的内容。 **注：** 对于具有大容量存储器的器件，打印的页数可能会很多。这种情况下建议将窗口内容打印到一个文件中（Print 对话框，**General** 选项卡，“Print to File”复选框），然后从该文件中选择需要打印哪些页。
Adjust Table Columns	自动调整列。

在设置为 DATA 格式的存储器窗口数据区域中单击右键可弹出该菜单。并非所有项对于所有 32 位 MCU 均可见。

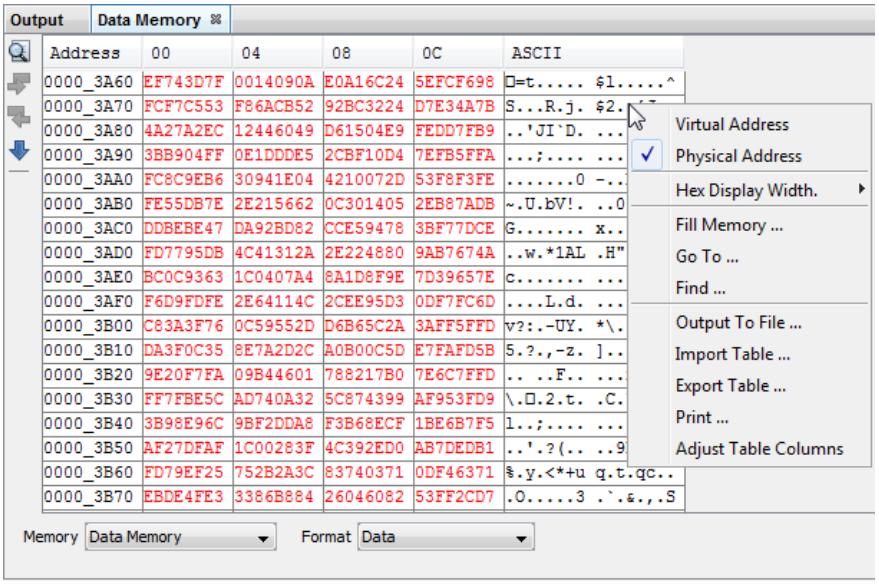
表 12-44. Execution Memory 窗口上下文菜单——Data 格式

项	说明
Virtual Address（KSEG[0:1]）	选择要显示的虚拟地址。有关详细信息，请参见器件数据手册。
Physical Address	选择要显示的物理地址。
Hex Width Display（十六进制宽度显示）	设置十六进制显示宽度。 （选项取决于所选器件）。 示例： 一个字节，例如 00 01 02 ...0E 0F 两个字节，例如 00 02 04 ...0C 0E 四个字节，例如 00 04 08 0C
Fill Memory	使用 Data 中的值从 Start Address 到 End Address 填充存储器。 在 Fill Memory 对话框中指定其他选项。
Go To	转至在 Go To 对话框中指定的地址/函数。
Find	查找在 Find 对话框中指定的文本。
Output To File	使用 Output To File 对话框将显示的窗口内容写入文本文件。
Import Table	使用 Import Table 对话框将表格数据从文件导入存储器窗口。
Export Table	使用 Export Table 对话框将表格数据从存储器窗口导出到文件中。
Print	使用 Print 对话框打印该窗口的内容。 **注：** 对于具有大容量存储器的器件，打印的页数可能会很多。这种情况下建议将窗口内容打印到一个文件中（Print 对话框，**General** 选项卡，“Print to File”复选框），然后从该文件中选择需要打印哪些页。
Adjust Table Columns	自动调整列。

12.10.2 Data Memory 窗口

Data Memory 窗口会显示处于当前所选 32 位器件的数据和/或程序存储器范围内的存储单元。

图 12-25. 具有内容的 Data Memory 窗口



12.10.2.1 Data Memory 窗口显示

您可以通过从窗口底部的 Format 下拉框中进行选择来指定存储器在窗口中的显示方式。

使用硬件工具进行调试时，某些寄存器的每半字节数据可能会显示“R”，以表示保留的资源。

Data 格式

该窗口将具有以下列：

- Address——下一列中的数据的十六进制地址。
- Data Blocks——十六进制数据，以 4 字节块的形式显示。
- ASCII——相应数据行的 ASCII 表示形式。

Code 格式

该窗口将具有以下列：

- Line——对应于存储器地址的参考行号。
- Address——操作码的物理十六进制地址。
- Opcode——十六进制操作码，以 4 字节块的形式显示
  高亮显示的操作码代表程序计数器的当前位置。
- Label——以符号格式显示的操作码标号。
- Disassembly——操作码助记符的反汇编版本。

12.10.2.2 Data Memory 窗口图标

图标位于窗口左侧。

表 12-45. Data Memory 窗口图标

图标	图标文本	功能
	Refresh by Read Device Memory	与调试工具栏“Read Device Memory”图标的功能相同——将器件存储器的内容上传到 MPLAB X IDE。
	Find	指定要在窗口中查找的字符串。可选择全字匹配或区分大小写。
	Find Next	查找 Find 的下一个字符串实例。

.....（续）

图标	图标文本	功能
	Find Previous	查找 Find 的上一个字符串实例。
	Go To	转至指定的行号/地址。

12.10.2.3 Data Memory 窗口菜单

在存储器窗口数据区域中单击右键可弹出该菜单。并非所有项对于所有 32 位 MCU 均可见。

表 12-46. Data Memory 窗口上下文菜单

项	说明
Virtual Address	选择要显示的虚拟地址。有关详细信息，请参见器件数据手册。
Physical Address	选择要显示的物理地址。
Symbolic Mode	**仅限 Code 格式** 显示反汇编的十六进制代码和符号。
Hex Width Display	**仅限 Data 格式** 设置十六进制显示宽度。 （选项取决于所选器件）。 示例： 一个字节，例如 00 01 02 ...0E 0F 两个字节，例如 00 02 04 ...0C 0E 四个字节，例如 00 04 08 0C
Fill Memory	使用 Data 中的值从 Start Address 到 End Address 填充存储器。 在 Fill Memory 对话框中指定其他选项。
Go To	转至在 Go To 对话框中指定的地址/函数。
Go To Source Line	**仅限 Code 格式** 转至编辑器中的相应源代码行。
Find	查找在 Find 对话框中指定的文本。
Output To File	使用 Output To File 对话框将显示的窗口内容写入文本文件。
Import Table	使用 Import Table 对话框将表格数据从文件导入存储器窗口。
Export Table	使用 Export Table 对话框将表格数据从存储器窗口导出到文件中。
Print	使用 Print 对话框打印该窗口的内容。 **注：** 对于具有大容量存储器的器件，打印的页数可能会很多。这种情况下建议将窗口内容打印到一个文件中（Print 对话框，**General** 选项卡，“Print to File”复选框），然后从该文件中选择需要打印哪些页。
Adjust Table Columns	自动调整列。

12.10.3 Peripherals 窗口

Peripherals 窗口会显示与器件外设相关的 SFR 的内容。要仅查看几个 SFR，您可能会希望使用 Watches 窗口，该窗口可以帮助解决使用硬件调试工具时的速度问题（即，窗口更新速率更快）。

每次发生中断时，SFR 的内容都会发生更新。

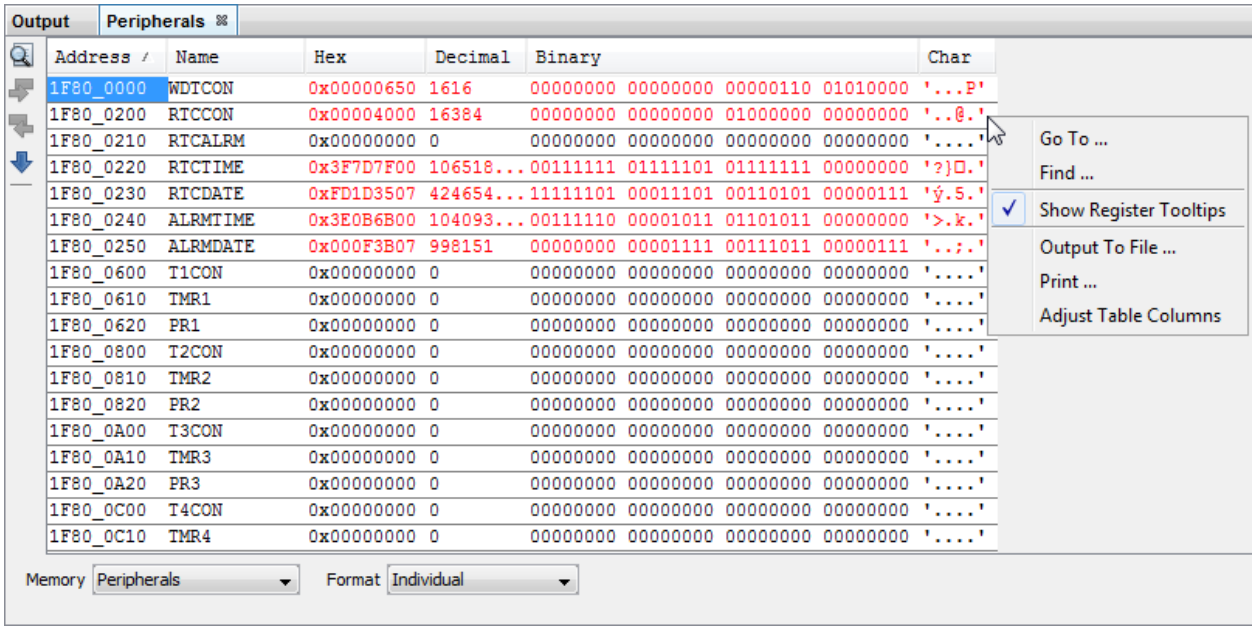
可见的寄存器

如果某个数据存储器寄存器在器件上未物理实现，则它可能不会出现在 SFR 列表中。某些工具（如软件模拟器）可能会让您看到实际器件上不存在的寄存器，如预分频器。

单步执行

如果选择了“Freeze Peripherals On Halt”，则在单步执行时 SFR 或 Watches 窗口中的 I/O 端口位不会发生更新。引脚会被修改，但获取新值的读请求会由于冻结而被阻止，无法进行更新，直到下一个单步或运行命令为止。

图 12-26. 具有内容的 Peripherals 窗口



12.10.3.1 Peripherals 窗口显示

您可以通过从窗口底部的 Format 下拉框中进行选择来指定存储器在窗口中的显示方式。

使用硬件工具进行调试时，某些寄存器的每半字节数据可能会显示“R”，以表示保留的资源。

- **Individual**——同时查看所有外设寄存器。
- **Groups**（组）——查看功能组中的外设寄存器。

两种格式的数据均显示在以下各列中。

- Address——SFR 物理十六进制地址。
- Virtual（虚拟）——由总线矩阵定义的 SFR 虚拟十六进制地址。
- Name——SFR 的符号名称。
- Radix Information——Hex、Decimal、Binary 和 Char。
  您可以通过右键单击列标题栏来添加要显示的基数信息。十六进制以 4 字节块的形式显示。

12.10.3.2 Peripherals 窗口图标

图标位于窗口左侧。

表 12-47. Data Memory 窗口图标

图标	图标文本	功能
	Find	指定要在窗口中查找的字符串。可选择全字匹配或区分大小写。
	Find Next	查找 Find 的下一个字符串实例。

.....（续）

图标	图标文本	功能
	Find Previous	查找 Find 的上一个字符串实例。
	Go To	转至指定的行号/地址。

12.10.3.3 Peripherals 窗口菜单

在存储器窗口数据区域中单击右键可弹出该菜单，如下表所示。

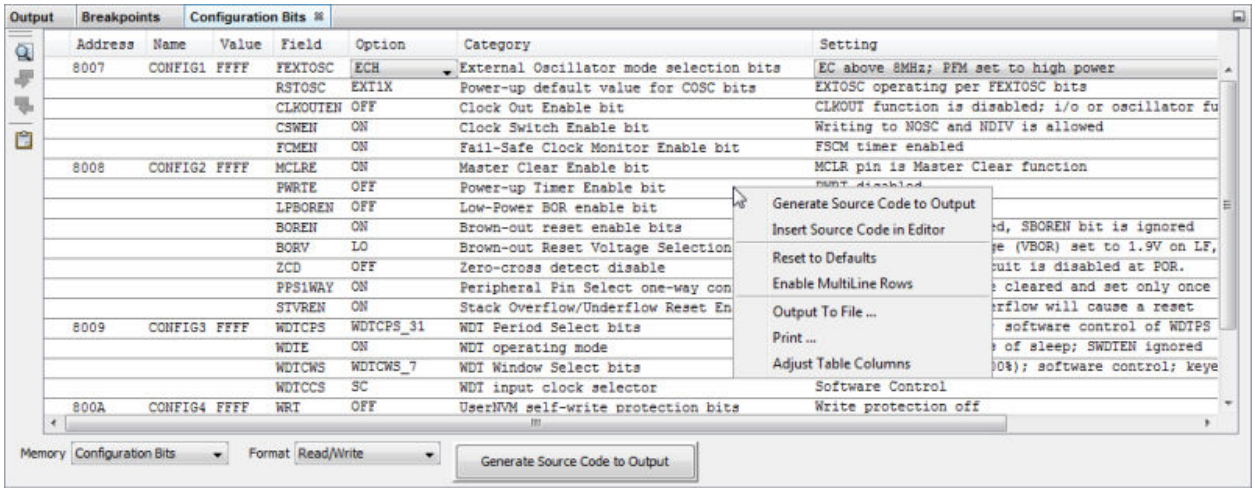
表 12-48. Peripherals 窗口上下文菜单

项	说明
Find	查找在 Find 对话框中指定的文本。
Output To File	使用 Output To File 对话框将显示的窗口内容写入文本文件。
Print	使用 Print 对话框打印该窗口的内容。 **注：** 对于具有大容量存储器的器件，打印的页数可能会很多。这种情况下建议将窗口内容打印到一个文件中（Print 对话框，**General** 选项卡，“Print to File”复选框），然后从该文件中选择需要打印哪些页。
Adjust Table Columns	自动调整列。

12.10.4 Configuration Bits 窗口

4.19 在 Configuration Bits 窗口中设置配置值介绍了关于使用 Configuration Bits 窗口的详细信息。

图 12-27. 具有内容的 Configuration Bits 窗口



12.10.4.1 Configuration Bits 窗口显示

您可以通过从窗口底部的 Format 下拉框中进行选择来指定存储器在窗口中的显示方式。可用的格式取决于您的器件。

当您对窗口中的配置位设置满意时，可以单击 **Generate Source Code to Output** 按钮，将特定于器件的配置代码写入 Output 窗口。然后，您可以在编辑器窗口中将该代码剪切并粘贴到您的应用程序中。

表 12-49. Configuration Bits 窗口显示

列标题	定义
Address	配置字/字节的地址。

.....（续）

列标题	定义
Name	配置寄存器的名称。
Value	配置字/字节的当前值。
Field*	对于代码中设置的配置位，宏的字段部分。 例如，WDTE 是宏_WDTE_OFF 的字段部分。
Option*	对于代码中设置的配置位，宏的选项部分。 例如，OFF 是宏_WDTE_OFF 的选项部分。
Category	相应配置字/字节中的配置位的名称。
Setting	配置位的当前设置。使用下拉列表可更改设置。配置字/字节的值将相应地发生更改。
*并非所有器件都支持。	

12.10.4.2 Configuration Bits 窗口图标

图标位于窗口左侧。

表 12-50. Configuration Bits 窗口图标

图标	图标文本	功能
	Refresh by Read Device Memory	与调试工具栏“Read Device Memory”图标的功能相同——将器件存储器的内容上传到 MPLAB X IDE。
	Find	指定要在窗口中查找的字符串。可选择全字匹配或区分大小写。
	Find Next	查找 Find 的下一个字符串实例。
	Find Previous	查找 Find 的上一个字符串实例。
	Paste	将生成的源代码粘贴到编辑器中的光标处。 如果编辑器未处于焦点，则 Output 窗口将显示警告，生成的源代码将置于该窗口中。

12.10.4.3 Configuration Bits 窗口菜单

在存储器窗口数据区域中单击右键可弹出该菜单。并非所有项对于所有 MCU 均可见。

表 12-51. Configuration Bits 窗口菜单

项	说明
Generate Source Code to Output	根据需要在窗口中设置配置位。然后选择该选项以在 Output 窗口中生成代码，您可以将其剪切并粘贴到程序中以设置配置位。 与窗口上的按钮功能相同。
Insert Source Code to Editor	根据需要在窗口中设置配置位。然后选择该选项以在编辑器窗口中生成代码，您可以将其剪切并粘贴到程序中以设置配置位。
Reset to Defaults	将所有配置位的值复位为其 MPLAB X IDE 默认值。
Enable Multiline Rows	如果列的长度短于内容的长度（例如，Category），则允许换行显示内容。否则，内容将被截断，并用省略号（...）表示。
Output To File	使用 Output To File 对话框将显示的窗口内容写入文本文件。

.....（续）

项	说明
Print	使用 Print 对话框打印该窗口的内容。 **注：** 对于具有大容量存储器的器件，打印的页数可能会很多。这种情况下建议将窗口内容打印到一个文件中（Print 对话框，**General** 选项卡，“Print to File”复选框），然后从该文件中选择需要打印哪些页。
Adjust Table Columns	自动调整列。

12.10.5 CPU Registers 窗口

CPU Registers 窗口会显示与器件 CPU 相关的 SFR 的内容。要仅查看几个 CPU SFR，您可能会希望使用 Watches 窗口，该窗口可以帮助解决使用硬件调试工具时的速度问题（即，窗口更新速率更快）。

每次发生中断时，CPU 寄存器的内容都会发生更新。

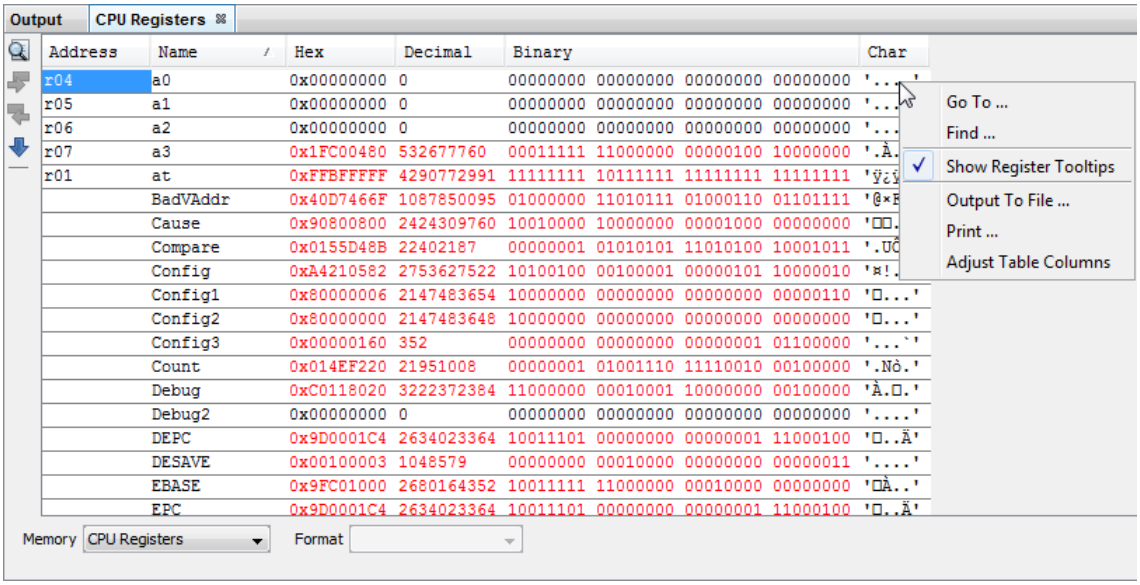
可见的寄存器

如果某个数据存储器寄存器在器件上未物理实现，则它可能不会出现在 SFR 列表中。某些工具（如软件模拟器）可能会让您看到实际器件上不存在的寄存器，如预分频器。

单步执行

如果选择了“Freeze Peripherals On Halt”，则在单步执行时 SFR 或 Watches 窗口中的 I/O 端口位不会发生更新。引脚会被修改，但获取新值的读请求会由于冻结而被阻止，无法进行更新，直到下一个单步或运行命令为止。

图 12-28. 具有内容的 CPU Registers 窗口



12.10.5.1 CPU Registers 窗口显示

您可以通过从窗口底部的 Format 下拉框中进行选择来指定存储器在窗口中的显示方式，具体包括以下两种选项：

- Blank（空白）——仅一个视图。
- Devices with FPU（带有 FPU 的器件）——对于带有浮点单元（Floating-Point Unit，FPU）的 PIC32 器件，格式选项包括：
  - All Registers（所有寄存器）
  - Only CPU Registers（仅 CPU 寄存器）
  - Only FPU Registers（仅 FPU 寄存器）

Blank、All Registers 或 Only CPU Registers 格式

数据显示在以下各列中。

- Address——SFR 物理十六进制地址。
- Name——SFR 的符号名称。
- Radix Information——Hex、Decimal、Binary 和 Char。
- Virtual——由总线矩阵定义的 SFR 虚拟十六进制地址。

Only FPU Registers 格式

数据显示在以下各列中。

- Address——SFR 物理十六进制地址。
- Name——SFR 的符号名称。
- Radix Information——Hex
- Floating Point（浮点）——Float（浮点）和 Double（双精度）

12.10.5.2 CPU Registers 窗口图标

图标位于窗口左侧。

表 12-52. Data Memory 窗口图标

图标	图标文本	功能
	Find	指定要在窗口中查找的字符串。可选择全字匹配或区分大小写。
	Find Next	查找 Find 的下一个字符串实例。
	Find Previous	查找 Find 的上一个字符串实例。
	Go To	转至指定的行号/地址。

12.10.5.3 CPU Registers 窗口菜单

在存储器窗口数据区域中单击右键可弹出该菜单，如下表所示。

表 12-53. CPU Registers 窗口上下文菜单

项	说明
Go To	转至在 Go To 对话框中指定的地址/函数。
Find	查找在 Find 对话框中指定的文本。
Show Register Tooltips	显示或隐藏寄存器的工具提示。将鼠标悬停在寄存器名称上可弹出相关信息。
Output To File	使用 Output To File 对话框将显示的窗口内容写入文本文件。
Print	使用 Print 对话框打印该窗口的内容。 **注：** 对于具有大容量存储器的器件，打印的页数可能会很多。这种情况下建议将窗口内容打印到一个文件中（Print 对话框，**General** 选项卡，“Print to File”复选框），然后从该文件中选择需要打印哪些页。
Adjust Table Columns	自动调整列。

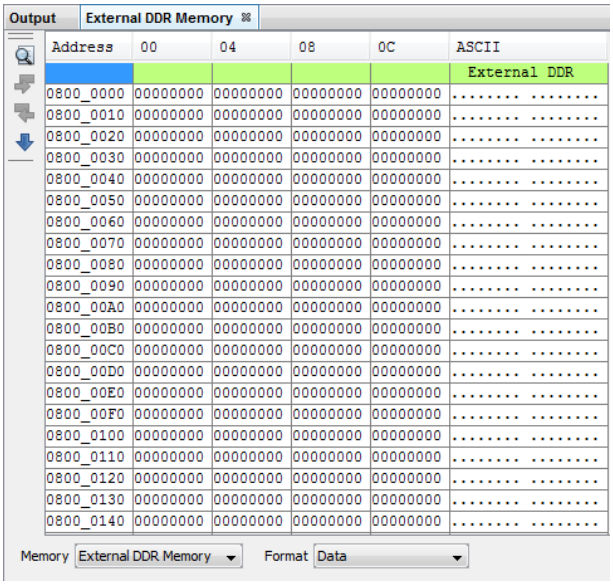
12.10.6 外部 EBI、SQI 或 DDR 存储器窗口

外部存储器窗口按总线类型显示外部存储器的内容：

- 外部总线接口（External Bus Interface，EBI）

- 串行四通道接口（Serial Quad Interface，SQI）
- 双倍数据速率（Double Data Rate，DDR）

图 12-29. External DDR Memory（外部 DDR 存储器）窗口



12.10.6.1 外部 EBI、SQI 或 DDR 存储器窗口显示

要在该窗口中查看外部存储器内容：

- 在代码中将函数/变量分配给外部存储器——有关详细信息，请参见《MPLAB® XC32 C/C++编译器用户指南》（DS50001686G_CN）。
- 装入符号——在 *File>Properties*（文件>属性），“Loading”类别中，选中“Load symbols when programming or building for production (slows process)”。
- 编译代码——编译项目。

您可以通过从窗口底部的 Format 下拉框中进行选择来指定存储器在窗口中的显示方式：

Code 格式

该窗口将具有以下列：

- Line——对应于存储器地址的参考行号。
- Address——操作码的物理十六进制地址。
- Opcode——十六进制操作码，以 4 字节块的形式显示
  高亮显示的操作码代表程序计数器的当前位置。
- Label——以符号格式显示的操作码标号。
- Disassembly——操作码助记符的反汇编版本。

Data 格式

该窗口将具有以下列：

- Address——下一列中的数据的十六进制地址。
- Data Blocks——十六进制数据，以 4 字节块的形式显示。
- ASCII——相应数据行的 ASCII 表示形式。

12.10.6.2 外部 EBI、SQI 或 DDR 存储器窗口图标

图标位于窗口左侧。

表 12-54. 外部存储器窗口图标

图标	图标文本	功能
	Refresh by Read Device Memory	与调试工具栏“Read Device Memory”图标的功能相同——将器件存储器的内容上传到 MPLAB X IDE。
	Find	指定要在窗口中查找的字符串。可选择全字匹配或区分大小写。
	Find Next	查找 Find 的下一个字符串实例。
	Find Previous	查找 Find 的上一个字符串实例。
	Go To	转至指定的行号/地址。

12.10.6.3 外部 EBI、SQI 或 DDR 存储器窗口菜单

在设置为 CODE 格式的存储器窗口数据区域中单击右键可弹出该菜单。并非所有项对于所有 32 位 MCU 均可见。

表 12-55. 外部存储器窗口上下文菜单——Code 格式

项	说明
External Address (KSEG[2:3])（外部地址（KSEG[2:3]））	选择要显示的地址。有关详细信息，请参见器件数据手册。
Physical Address	选择要显示的物理地址。
Run to Cursor	将程序运行至当前光标位置。
Step Instruction	执行单步指令。
Set PC at Cursor	将程序计数器（PC）设置在光标位置。
Focus Cursor at PC	将光标移动到当前 PC 地址处，并使该地址在窗口中居中。
Toggle Breakpoint	翻转（开启/关闭）现有断点。
Symbolic Mode	显示反汇编的十六进制代码和符号。
Auto Decode 16 bit	使用 MIP16/microMIPS 指令解码。有关详细信息，请参见器件数据手册。
Fill Memory	使用 Data 中的值从 Start Address 到 End Address 填充存储器。 在 Fill Memory 对话框中指定其他选项。
Go To	转至在 Go To 对话框中指定的地址/函数。
Go To Source Line	转至编辑器中的相应源代码行。
Find	查找在 Find 对话框中指定的文本。
Output To File	使用 Output To File 对话框将显示的窗口内容写入文本文件。
Print	使用 Print 对话框打印该窗口的内容。 **注：** 对于具有大容量存储器的器件，打印的页数可能会很多。这种情况下建议将窗口内容打印到一个文件中（Print 对话框，**General** 选项卡，“Print to File”复选框），然后从该文件中选择需要打印哪些页。
Adjust Table Columns	自动调整列。

在设置为 DATA 格式的存储器窗口数据区域中单击右键可弹出该菜单。并非所有项对于所有 32 位 MCU 均可见。

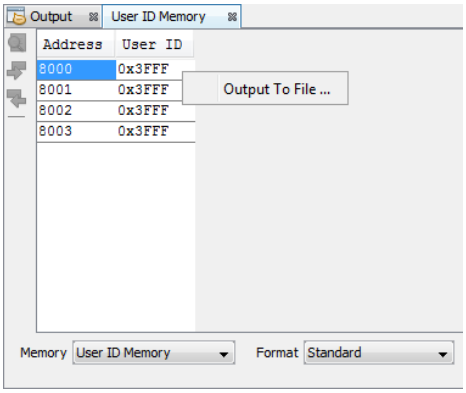
表 12-56. 外部存储器窗口上下文菜单——Data 格式

项	说明
External Address (KSEG[2:3])	选择要显示的地址。有关详细信息，请参见器件数据手册。
Physical Address	选择要显示的物理地址。
Hex Width Display	设置十六进制显示宽度 （选项取决于所选器件）。 示例： 一个字节，例如 00 01 02 ...0E 0F 两个字节，例如 00 02 04 ...0C 0E 四个字节，例如 00 04 08 0C
Fill Memory	使用 Data 中的值从 Start Address 到 End Address 填充存储器。 在 Fill Memory 对话框中指定其他选项。
Go To	转至在 Go To 对话框中指定的地址/函数。
Find	查找在 Find 对话框中指定的文本。
Output To File	使用 Output To File 对话框将显示的窗口内容写入文本文件。
Import Table	使用 Import Table 对话框将表格数据从文件导入存储器窗口。
Export Table	使用 Export Table 对话框将表格数据从存储器窗口导出到文件中。
Print	使用 Print 对话框打印该窗口的内容。 **注：** 对于具有大容量存储器的器件，打印的页数可能会很多。这种情况下建议将窗口内容打印到一个文件中（Print 对话框，**General** 选项卡，“Print to File”复选框），然后从该文件中选择需要打印哪些页。
Adjust Table Columns	自动调整列。

12.10.7 User ID Memory 窗口

一些器件具有可用于存储校验和或其他代码标识（ID）数字的存储单元。在编程/校验期间，这些存储单元是可读写的。根据不同的器件，它们可能还可以在正常执行期间通过 TBLRD 和 TBLWT 指令访问。

图 12-30. 具有内容的 User ID Memory 窗口

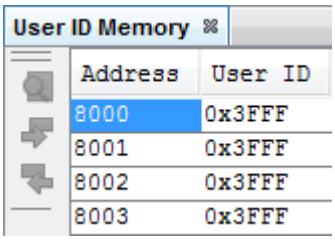


12.10.7.1 User ID Memory 窗口显示

要确定可在此处输入的值，请参见器件编程规范。对于大多数器件，这会设置器件 ID 字的低半字节；高半字节设置为“0”。高半字节只能通过编程方式写入，例如通过使用表写操作。

您可以通过从窗口底部的 Format 下拉框中进行选择来指定存储器在窗口中的显示方式。

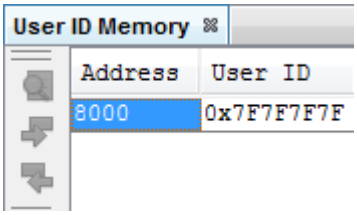
图 12-31. 标准显示



在标准显示中，数据显示在以下列中。

- Address——用户 ID 十六进制地址。
- User ID——用户 ID 存储器的内容（以十六进制表示）。

图 12-32. 传统显示



在传统显示中，数据显示在以下列中。

- Address——用户 ID 十六进制起始地址。
- User ID——用户 ID 存储器的内容（以十六进制表示）。

12.10.7.2 User ID Memory 窗口图标

图标位于窗口左侧。

表 12-57. User ID Memory 窗口图标

图标	图标文本	功能
	Find	指定要在窗口中查找的字符串。可选择全字匹配或区分大小写。
	Find Next	查找 Find 的下一个字符串实例。
	Find Previous	查找 Find 的上一个字符串实例。

12.10.7.3 User ID Memory 窗口菜单

在存储器窗口数据区域中单击右键可弹出该菜单，如下表所示。

表 12-58. User ID Memory 窗口上下文菜单

项	说明*
Output To File	使用 Output To File 对话框将显示的窗口内容写入文本文件。

12.11 与存储器窗口关联的对话框

存储器窗口具有关联的对话框，用于支持窗口数据的格式化、管理和输出。可从内容菜单访问这些对话框。

12.11.1 存储器窗口 Go To 对话框

转至目标存储器中的某个存储单元。可从某些目标存储器窗口上下文菜单访问该对话框。

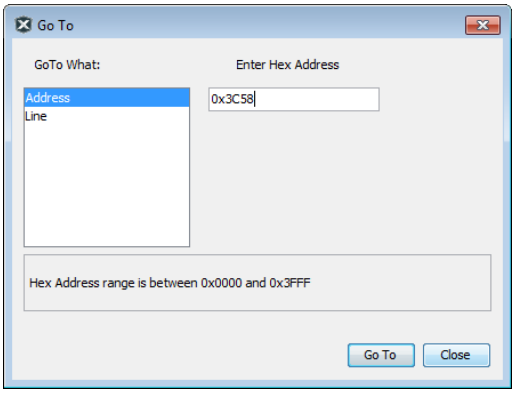


表 12-59. Go To 对话框

项	说明
Go To What（转至的内容）	输入想要转至的内容。单击该项可查看以下说明。 **Address**：输入一个十六进制地址。 **Symbol**：输入一个程序符号。 **Line**：输入一个行号。 **Label**：选择一个程序标号。 **Function**：选择一个程序函数。
Go to description（转至说明）	选择“Go to description”项可查看说明。

12.11.2 存储器窗口 Find 对话框

查找目标存储器中的某个存储单元。可从某些目标存储器窗口上下文菜单访问该对话框。

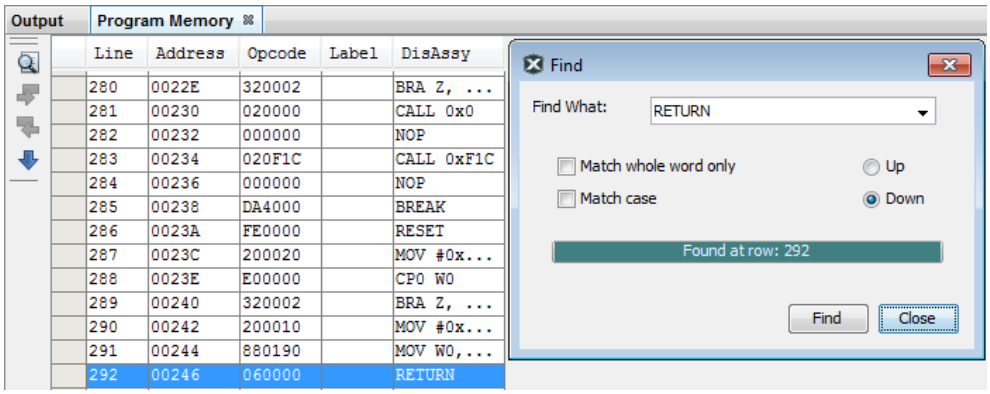


表 12-60. Find 对话框

项	说明
Find What（查找内容）	输入您要查找的文本。
Match（匹配）	可选择“Match whole word only”（仅全字匹配）和/或“Match case”。
Direction（方向）	可选择“Up”或“Down”搜索窗口。
Found At bar（找到位置栏）	找到相应项后，其位置将显示在该栏上。

12.11.3 存储器窗口 Output To File 对话框

将指定的存储器范围输出到文件。可从某些目标存储器窗口上下文菜单访问该对话框。

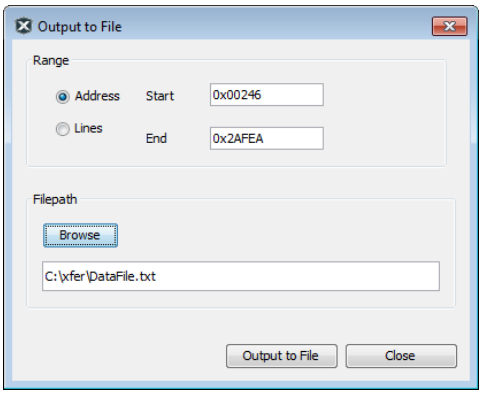


表 12-61. Output To File 对话框

项	说明
Range（范围）	输入要输出到文件的数据的范围。 **Address**：输入地址范围（起始地址至结束地址）。 **Lines**（行）：输入行范围——起始行至结束行。
Filepath（文件路径）	浏览到输出文件的位置，或在文本框中输入。

12.11.4 存储器窗口 Import Table/Export Table 对话框

以表格格式从文件目标存储器导入或导出到文件目标存储器。可从某些目标存储器窗口上下文菜单访问该对话框。

图 12-33. Export Table 对话框

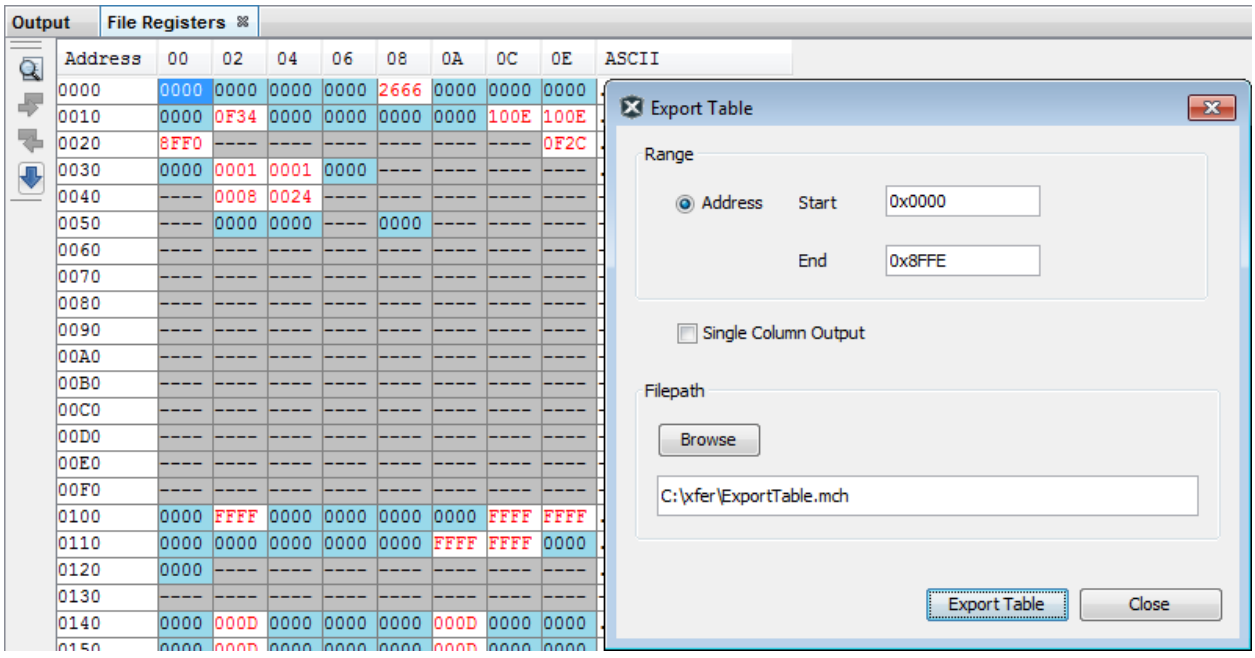


图 12-34. 导出表文件

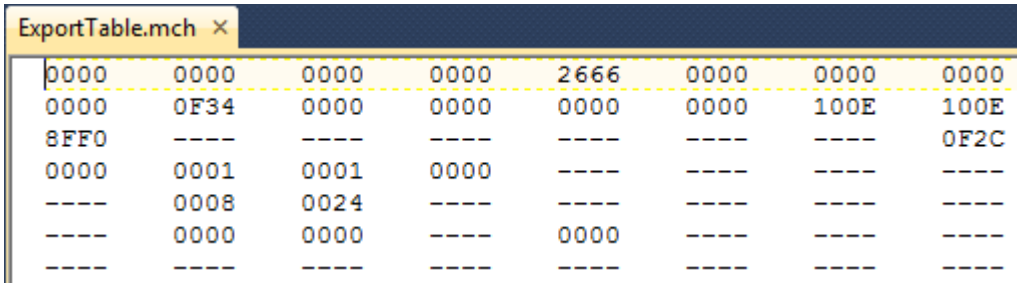


表 12-62. Import Table/Export Table 对话框

项	说明
Range	输入以表格格式导入或导出到文件的数据的范围。 **Address**：输入地址范围（起始地址至结束地址）。
Single Column Format（单列格式）	以单列格式**导出**表。
Filepath	浏览到输入或输出文件的位置，或在文本框中输入。

12.11.5 存储器窗口 Print 对话框

打印存储器窗口内容。可从某些目标存储器窗口上下文菜单访问该对话框。

可用的打印选项可能取决于所选的打印机。

图 12-35. Print 对话框

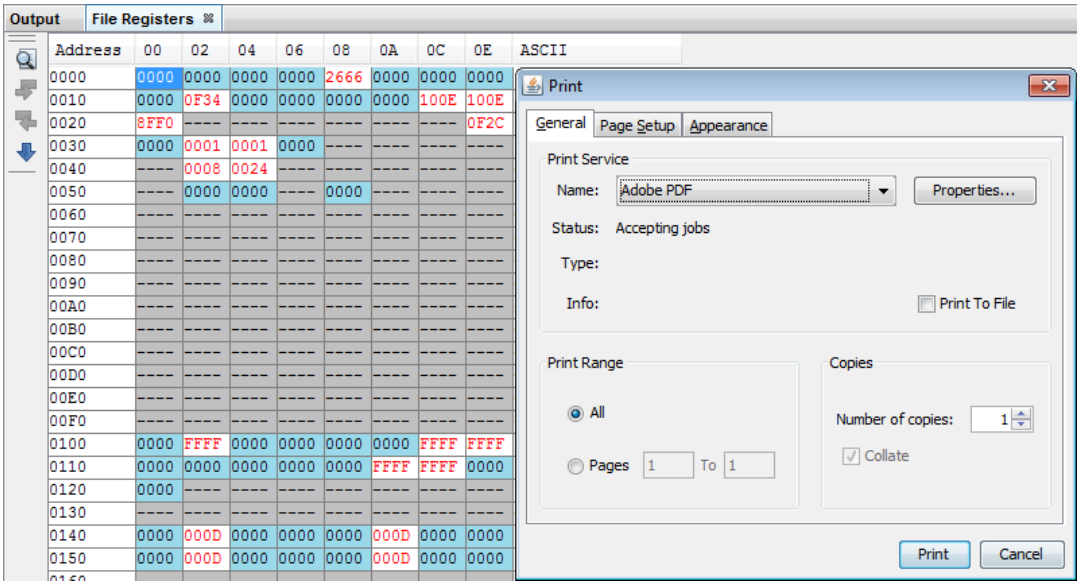


图 12-36. Print 对话框输出

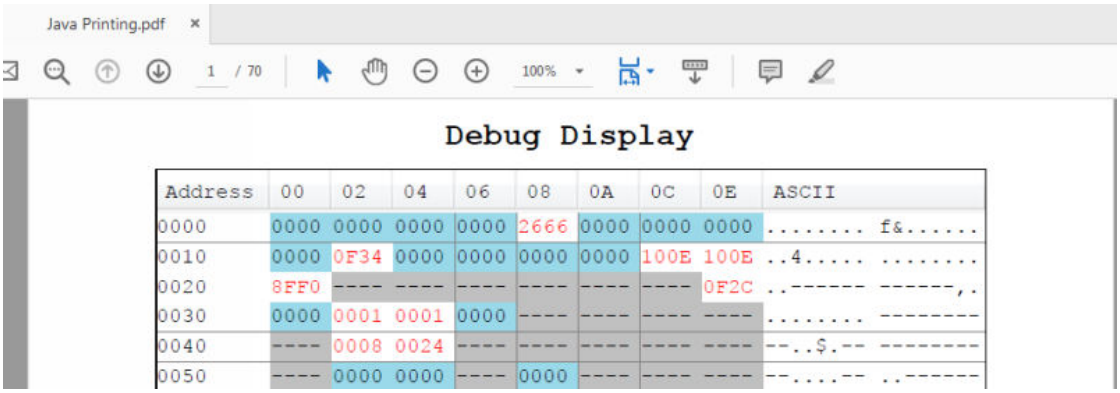


表 12-63. Print 对话框

项	说明
General 选项卡	设置常规打印选项。 **Print Service**（打印服务）：选择打印机/打印服务。单击 Properties 可设置打印服务。此外，也可选择打印到文件。 **Print Range**（打印范围）：要打印的页的范围。 **Copies**（副本）：要打印的副本的数量。
Page Setup	设置页选项。 **Media**（介质）：指定页大小和源。 **Orientation**（方向）：指定页方向。 **Margin**（边距）：指定页边距。
Appearance（外观）	设置打印输出的外观。 **Color appearance**（颜色外观）：指定彩色或单色。 **Quality**（质量）：指定草稿、普通或高质量。 **Sides**（打印面）：指定打印面和双面打印。 **Job Attributes**（作业属性）：打印标题，指定作业名称和用户。

12.12 Message Center

通过 *Window>Debugging>Output>Message Center（窗口>调试>输出>消息中心）*打开 Message Center 窗口。

Message Center 窗口收集所有输出窗口消息，然后按时间顺序将它们显示在一个视图中。可以通过选择不同的筛选器来显示不同的相关消息。

图 12-37. Message Center 窗口

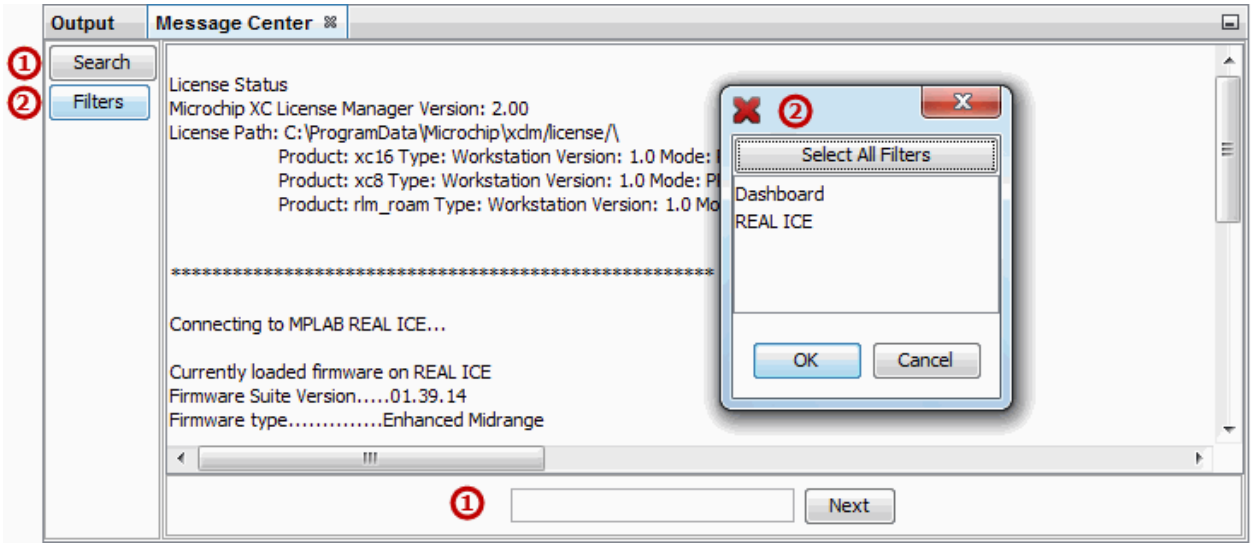


表 12-64. Message Center 按钮

编号	按钮	功能
1	Search	翻转搜索文本框。当搜索框可见时，可通过输入文本在 Message Center 窗口中搜索该文本字符串。
2	Filters（筛选器）	默认情况下，出现在 Output 窗口中的所有消息都会显示在 Message Center 窗口中。 单击该按钮可打开一个窗口，其中包含 Output 窗口选项卡式项的列表。通过从列表中选择一项或多项来筛选显示的消息。

12.13 Output 窗口

通过选择 *Window>Output*（窗口>输出）打开该窗口。

任务窗格包含多个窗口，其中一些是从 NetBeans 平台继承的，另一些是 MPLAB X IDE 特有的。Output 窗口包含 MPLAB X IDE 输出信息，该信息显示在窗口内的选项卡上。

表 12-65. Output 窗口选项卡项

项	说明
Debugger Console	显示主要的调试操作，例如“User program running”（用户程序运行）。
Tool-specific（工具特定信息）	显示工具固件版本、器件 ID 和操作状态。
Build, Load（编译，装入）	显示关于编译和程序装入的信息和状态。
Clean, Build, Load（清除，编译，装入）	显示关于清除、编译和程序装入的信息和状态。
Peripheral Output（外设输出）	显示使用软件模拟器作为调试工具时来自外设（例如 UART）的技术输出。

12.13.1 内容滚动

正在发生的编程操作将在 Output 窗口中显示和滚动。最后执行的操作将显示在窗口底部。

如果向上滚动查看内容并随后再次编程，光标不会从当前位置移开。这样一来，即使有更多内容填充到窗口下方，也能保持原先的位置。若要查看新内容，必须手动向下滚动。

12.13.2 上下文菜单

在 Output 窗口中单击右键将显示各种选项，如下所示。

表 12-66. Output 窗口上下文菜单

项	说明
Copy	将选定文本从 Output 窗口复制到剪贴板。
Paste	将选定文本从剪贴板复制到 Output 窗口。
Find	在 Output 窗口中查找选定的文本（或输入要查找的其他文本）。您可以使用正则表达式和匹配大小写。
Find Next	查找 Find 文本的下一个匹配项。
Find Previous	查找 Find 文本的上一个匹配项。
Filter	按文本或正则表达式筛选输出。
Wrap Text（文本折行）	对 Output 窗口中的文本进行折行。
Larger Font（较大的字体）	使字体变大。
Smaller Font（较小的字体）	使字体变小。
Choose Font（选择字体）	选择字体类型、样式和大小。
Save As	将选定文本保存到一个文件中。
Clear	清除 Output 窗口选项卡中的所有文本。
Close（关闭）	关闭 Output 窗口。

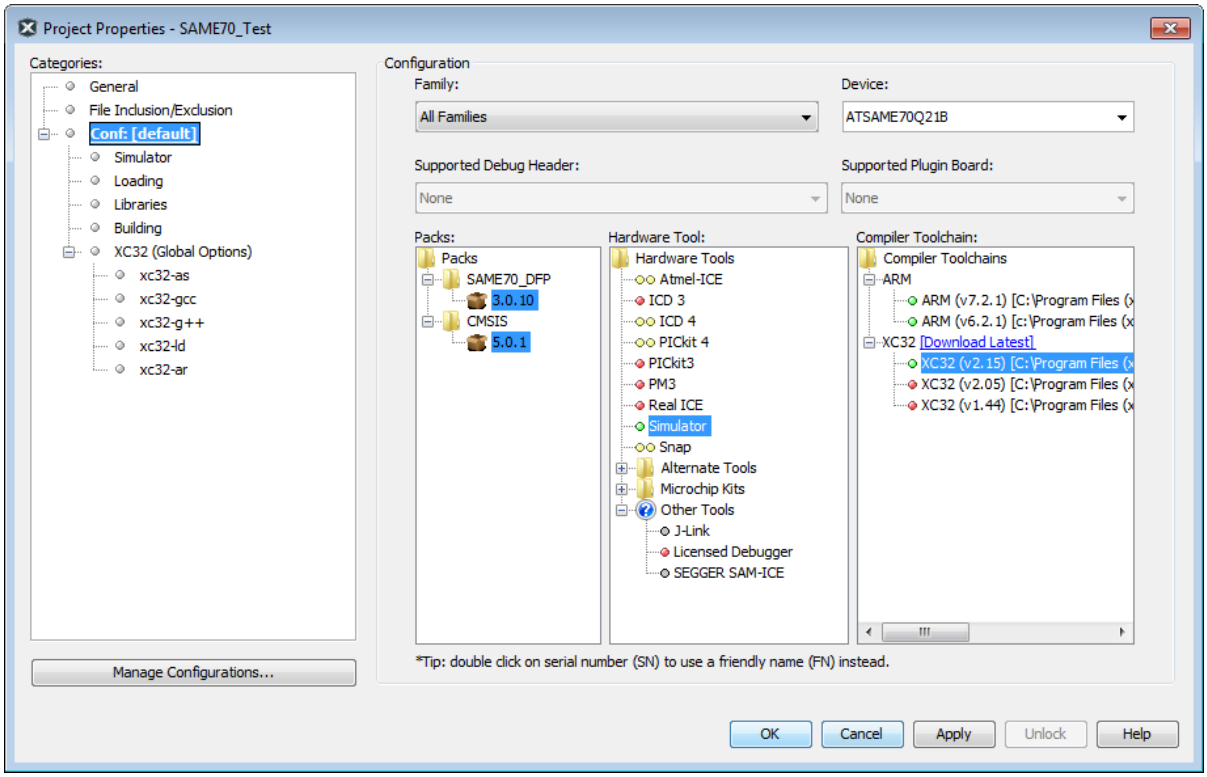
12.14 Project Properties 窗口

有几种方法可打开该窗口。请参见 4.3 打开 Project Properties。

该窗口用于查看或更改项目器件、工具和工具设置。更多信息，请参见：

- 4.4 查看或更改项目属性
- 4.5 设置或更改调试器/编程器工具选项
- 4.6 设置或更改语言工具选项
- 5.5 可装入项目、文件和符号
- 4.9.5 添加和设置库与目标文件
- 4.10 设置编译属性

图 12-38. Project Properties 窗口



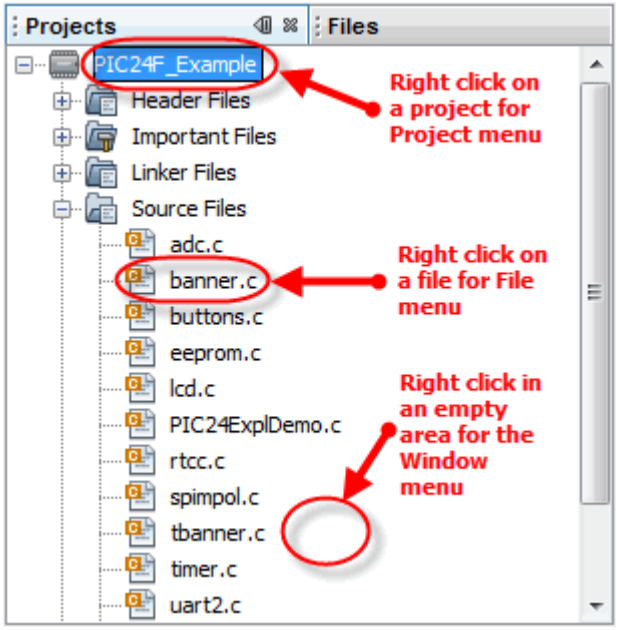
12.15 Projects 窗口

通过选择 *Window>Projects*（窗口>项目）打开该窗口。

Projects 窗口属于 NetBeans 平台窗口。但是，它已被定制为显示 MPLAB X IDE 项目的相关逻辑（虚拟）文件夹。此外，上下文（右键单击）菜单也已被定制为包含一些 MPLAB X IDE 特有的项。

- Projects 窗口——逻辑文件夹
- Projects 窗口——项目菜单
- Projects 窗口——File 菜单
- Projects 窗口——窗口菜单

图 12-39. Projects 窗口上下文菜单



Projects 窗口——逻辑文件夹

添加了适用于 MPLAB X 项目的额外逻辑文件夹。因此，显示的所有文件夹可以归类为如下文件类型之一：

- **Header Files**——MPLAB X IDE 在编译时不使用该类别。可以将它视为记录项目对于头文件的依赖性的方式，以及一种访问这些文件的简便方法。在 Projects 窗口中双击某个头文件，可以在编辑器中打开该文件。
- **Important Files**——不属于任何其他类别的所有文件会放入该文件夹中。
  您可以在 Projects 窗口中向该位置添加特定于项目的数据手册（PDF 文件）。然后，您可以双击 PDF 文件来打开数据手册（这要求已安装 PDF 阅读器）。
- **Linker Files**——您不需要向项目中添加链接描述文件，因为项目语言工具会查找对应于器件的相应通用链接描述文件。因此，该文件夹将为空，除非您自己创建了链接描述文件。
  该文件夹中应只有一个文件。如果具有多个链接描述文件，只有第一个会起作用——它是工具在链接步骤中使用的链接描述文件。
- **Source Files**——它们是被工具链接受作为其文件命令输入的惟一文件类型。
- **Libraries**——工具链应获取该文件夹中的所有文件以及目标文件，并将它们包含在最终链接步骤中。
- **Loadables**——要与项目十六进制文件进行合并或替换项目十六进制文件的项目和文件。

Projects 窗口——项目菜单

在 Projects 窗口中右键单击某个项目名称可弹出项目菜单。菜单项如下表所示。

表 12-67. 项目上下文菜单项

菜单项	说明
New	添加新项。对于 MPLAB® X IDE，提供了嵌入式文件类型。
Add Existing Item	向项目中添加现有文件。 文件的路径可以是自动（由 MPLAB X IDE 选择）、绝对或相对的。此外，您还可以选择将文件复制到项目文件夹中。
Add Existing Items from Folders（从文件夹中添加现有项）	添加指定文件夹中包含的所有文件。 您还可以通过指定模式来忽略某些文件夹。单击 **Default**（默认）可查看默认模式。另请参见 *Tools>Options*，**Miscellaneous**（其他）按钮，**Files** 选项卡，“Files ignored by the IDE”（被 IDE 忽略的文件）。

.....（续）

菜单项	说明
New Logical Folder（新建逻辑文件夹）	创建新的逻辑文件夹。 要查看实际文件夹，请参见 Files 窗口。
Locate Headers（查找头文件）	查找在 C 代码项目文件中调用的所有头文件（.h），并在列表窗口中显示它们，从而可以将它们添加到项目中 由于几种原因，您可能会希望向项目中添加头文件： • 可以从 Projects 窗口中的“Header Files”下打开这些文件，而不需要在您的计算机上搜寻它们。 • 通过“Save Project As”（将项目另存为）创建的项目将不包含头文件，除非将它们添加到项目中；因此，如果项目需要任何用户生成的头文件，则可能无法编译。 • 通过“Package”创建的压缩项目将不包含头文件，除非将它们添加到项目中；因此，如果解压后的项目需要任何用户生成的头文件，则可能无法编译。 该功能能够为所有 Microchip 工具链找到用户生成的头文件。
Add Item to Important Files（将项添加到重要文件中）	向项目中添加现有项。 文件的路径可以是自动（由 MPLAB X IDE 选择）、绝对或相对的。此外，您还可以选择将文件复制到项目文件夹中。 这对于向项目中添加软件模拟器文件、数据手册 PDF 和其他文件以供参考会非常有用。
Export Hex（导出十六进制）	将项目编译导出为十六进制文件。 在 MPLAB IDE v8 中，导出是对存储器对象进行提取。在 MPLAB X IDE 中，导出是编译的十六进制文件结果；因此，需要在代码中包含配置和 EEPROM 的所有必需的辅助存储器设置。
Build	根据在 Project Properties 窗口中选择的选项编译项目。 **注：** 当执行调试或运行时，将会自动编译项目。
Clean and Build	根据在 Project Properties 窗口中选择的选项清除并编译项目。 **注：** 当执行调试或运行时，将会自动编译项目。
Clean（清除）	通过删除先前编译的输出来清除项目。
Package	将当前项目打包为.zip 文件。
Set Configuration（设置配置）	打开 Project Properties 窗口，使您可以设置项目配置。
Run	执行项目代码。
Debug	在调试环境中执行项目代码。
Step In	单步执行在调试环境中运行的暂停代码。
Make and Program Device	对目标器件编程并保持在复位状态（不运行）。
Set as Main Project	将该项目设置为主项目。 在处理多个项目时非常有用。
Open Required Projects（打开所需的项目）	装入运行选定项目所需的所有其他项目。
Close	关闭所选的项目。
Rename	重命名项目。

.....（续）

菜单项	说明
Move	将项目移动到另一个位置。 随项目一起移动项目目录内的源文件。项目目录外的文件不会被移动。但是，项目将仍然保持对项目目录外的文件的引用——这是由设计决定的。
Copy	创建项目的副本。 如果源文件处于项目目录内，则源文件也会被复制到新位置。项目目录外的文件不会被复制。但是，项目将仍然保持对项目目录外的文件的引用——这是由设计决定的。
Delete	删除项目文件。 项目文件会被删除，但不删除项目目录下的内容。只有在选择“Also delete sources”时才会删除所有源文件。
Code Assistance（代码辅助）	选择在创建代码时的辅助功能：代码折叠和代码补全等。
Find	在该项目的文件中查找指定文本。
Share on Team Server（在团队服务器上共享）	在团队服务器上共享该项目。
Versioning	通过使用版本控制系统来控制该项目的版本。
Local History（本地历史记录）	查看或恢复为该项目的本地历史记录。
Properties	设置项目属性。 对于 MPLAB X IDE，Project Properties 窗口是特定于嵌入式开发的。

Projects 窗口——File 菜单

在 Projects 窗口中右键单击某个文件名称可弹出 File 菜单。下表列出了特定的菜单项。

表 12-68. 文件上下文菜单项

菜单项	说明
Open（打开）	在选项卡形式的编辑器窗口中打开该文件。
Cut	从项目中删除该文件，但将它的副本放到剪贴板上。
Copy	在剪贴板上放置文件的一个副本。
Paste	将文件的剪贴板副本粘贴到项目中。
Exclude file(s) from current configuration	选择一个或多个文件，将其从当前项目配置中排除。排除文件后，再次右键单击该文件将显示“Include file(s) from the current configuration”（在当前配置中包含文件）。 作用与打开 Project Properties 窗口，选择“File Inclusion/Exclusion”，包含/排除文件相同。
Compile File（编译文件）	仅编译当前文件。
Remove from Project	从项目中移除该文件。 该操作不会从 PC 中删除文件。要从项目和计算机中删除文件，请使用 Delete 键。
Rename	重命名文件。
Save as Template（另存为模板）	将当前文件保存为模板文件。

.....（续）

菜单项	说明
Local History	查看或恢复为该文件的本地历史记录。
Tools	文件工具包括： • **Apply Diff Patch**（应用 Diff 补丁）：应用由 Diff 创建的现有补丁。 • **Diff to**（比较差异）：显示该文件与此处指定的一个文件之间的差异。 • **Add to Favorites**：将该文件添加到 Favorites 窗口。
Properties	设置不同于项目属性的文件属性。 通过此菜单项可以选择从编译中排除该文件或通过选择不同配置来覆盖项目编译选项。

Projects 窗口——窗口菜单

在 Projects 窗口中的空白区域单击右键可弹出窗口菜单。下表列出了特定的菜单项。

表 12-69. 文件上下文菜单项

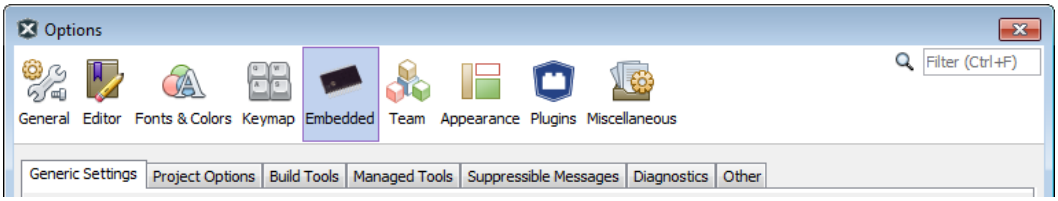
菜单项	说明
New Project	启动 New Project 向导。
New File	启动 New File 向导。
Open Project	打开现有项目。
Open Recent Project	打开最近项目列表中的现有项目。
Open Project Group（打开项目组）	打开项目组列表中的现有项目组。
Run Project	运行主项目。
Set Main Project	从已打开项目的列表中设置主项目。

12.16 Tools>Options 窗口，Embedded

选择 *Tools>Options*（对于 macOS 为 *MPLAB X IDE>Preferences*），然后单击 **Embedded** 按钮，可看到下列选项卡和选项。

Options 窗口属于 NetBeans 平台窗口。不过，它已针对 MPLAB X IDE 项目进行了定制。

图 12-40. Tools>Options>Embedded 选项卡



12.16.1 Generic Settings 选项卡

选择 *Tools>Options*（对于 macOS 为 *MPLAB X IDE>Preferences*），然后单击 **Embedded** 按钮，**Generic Settings** 选项卡，可看到下列选项。

表 12-70. Generic Settings 选项卡项

项	说明
Open source file and locate line in editor when debugger halts（在调试器暂停时打开源文件并定位到编辑器中的行）	暂停（例如由断点引起）时跳转到某个代码行以检查代码。 禁止时仅暂停执行。
Clear tool output window on new session (debug, program, upload)（在新会话（调试、编程和上传）时清除工具输出窗口）	在开始运行、调试或上传时清除 Output 窗口的内容。
Halt build on first failure（在第一次失败时暂停编译）	在进行编译时，在发生第一次失败时暂停该过程。 可以通过设置选定项目语言工具来确定产生哪些错误。转到 Project Properties 窗口中，并在“Categories”下选择语言工具。然后，在 Option Categories 中进行查找，找到一个允许您设置错误、警告和/或消息的类别。
Maintain active connection to hardware tool	如果选中，则将总是保持与硬件工具连接，而不只是在运行时（MPLAB® IDE v8 的行为）。 在选择该选项的情况下切换项目时（例如，在开发自举应用程序时），请确保工具和器件是相同的，以避免错误消息。 **注：** 该选项仅适用于当前项目和配置。如果切换项目或更改 Project Properties 中的项，将重新建立连接。
Read Device Memory to File: Export only memory used（将器件存储器读取至文件：仅导出已用存储空间）	对于“Read Device Memory to File”图标/功能，仅将器件已用存储空间的内容保存到文件中。
Silent build（静默编译）	进行编译，但不在 Output 窗口中产生消息。
Enable alternate watch list views during debug session（在调试会话期间使能备用观察列表视图）	在 Watches 窗口中显示 3 个观察视图方块。将观察视图与观察变量相关联。在单击观察视图方块时，将仅显示与该视图相关联的变量。所以，该功能的工作方式类似于一个筛选器。
Disable auto refresh for call stack view during debug sessions	使能后，暂停或停止时将自动更新内容。 如果窗口关闭或未处于焦点，则选中窗口并单击该按钮即可显示更新内容，无需再次运行并暂停/停止。 禁止后，暂停或停止时需手动更新内容。必须单击该按钮才能在暂停/停止时更新窗口内容。
On mouse over structure and array expressions, evaluate integral members only	选中后，当鼠标悬停在代码中的结构体或数组表达式上时，将仅显示整数成员。 取消选中后，当鼠标悬停在代码中的结构体或数组表达式上时，将显示所有成员。
Show unresolved variable names in Watches window during debug session（调试会话期间在 Watches 窗口中显示未解析的变量名称）	选中后，将在 Watches 窗口中显示未解析的变量。
Allow switch between pack versions（允许切换包版本）	选中后，可在 Project Properties 中显示的包之间切换。

.....（续）

项	说明
Enable gathering of compiler symbols（使能收集编译器符号）	在后台运行编译器，请求获取关于编译器内置宏和专用宏的信息。 默认情况下，该选项是禁止的。使能该选项会增加 IDE 收集符号信息所花费的时间。 **注：** 如果操作不依赖编译器的内部信息，则无需使能该功能。
On mouseover source lines in editor, evaluate break point status	选择触发求值的操作。 **Disable**：不进行求值。 **Mouse Only（仅鼠标）、Alt+Mouse（Alt+鼠标）、Shirt+Mouse（Shift+鼠标）、Shift+Alt+Mouse（Shift+Alt+鼠标）**：基于上述按键操作之一评估断点的值。
Hold-off period before memory view synchronization: Give priority to debug events (step over, step in, continue)（存储器视图同步前的拖延时间：优先考虑调试事件（单步跳过、单步进入和继续））	指定从器件停止到器件数据与存储器窗口同步之间的延时。 可以设置更长的延时，以便在更新之前可以发生更多的单步执行。这仅适用于通用 PIC 存储器视图，例如文件寄存器和数据存储器。
Debug Reset @ (following reset action during paused debug session)（调试复位@（在暂停调试会话期间的复位操作之后））	选择调试复位时的操作，即 *Debug>Reset*（调试>复位）： **Main**：复位时在 `main()` 处停止。 **Reset Vector**（复位向量）：复位时在复位向量处停止。 **注：** 这与器件的硬件复位无关。
Debug start-up (following debug project action)（调试启动（在调试项目操作之后））	指定 *Debug>Debug Project* 的代码执行： **Run**：立即开始执行。 **Main**：停止在 `main()` 处。 **Reset Vector**：停止在复位向量处。
Default Charset（默认字符集）	选择项目的默认字符集。

12.16.2 Project Options 选项卡

选择 *Tools>Options*（对于 macOS 为 *MPLAB X IDE>Preferences*），然后单击 **Embedded** 按钮，**Project Options**（项目选项）选项卡，可看到下列选项。

表 12-71. Project Options 选项卡项

项	说明
Make Options（Make 选项）	输入在编译项目时使用的 make 选项。 这些选项依赖于工具链。请参见语言工具文档。
File Path Mode	指定在项目中如何存储文件路径信息： **Auto**：项目文件夹内的文件的路径以相对方式存储；项目文件夹外的文件的路径以绝对方式存储。 **Always Relative**（总是相对）：所有路径都以相对于项目文件夹的方式存储。 **Always Absolute**（总是绝对）：所有路径都以绝对（完整路径）方式存储。
Save All Modified Files Before Running Make（运行 Make 之前保存所有修改过的文件）	如果选中，将在运行 make 之前保存 IDE 中所有未保存的文件。建议将该属性保留为选定，因为除非先将对 IDE 中文件的修改保存到磁盘中，否则 make 将无法识别到这些修改。

.....（续）

项	说明
Reuse Output Tabs from Finished Process（重用已完成进程的 Output 选项卡）	选中后，输出消息将显示在 Output 窗口中的相同选项卡中。取消选中后，将为每个新进程打开一个新的选项卡。
Use parallel make (`make -j 2n`)（使用并行 make（make -j 2n））	如果选中，make 将每次执行多个进程，其中的`-j`（或`--jobs`）是要并行运行的选项，2n 是进程数量，其中的 n 是计算机上可用处理器的数量。如果您的计算机不支持并行处理，则会禁止并行 make。 如果愿意，可以通过使用“Make Options”指定更多的进程。示例：`-j 10`。 **注：** 对于 MPLAB XC16 或 MPLAB C30，需要关闭过程抽象才能使用并行 make（*File>Project Properties*，编译器类别，“Optimizations”选项类别：取消选中“Unlimited procedural abstraction”（无限的过程抽象））。 **注：** MPASMX 无法在并行 make 下运行，无论是作为工具链还是作为 MPLAB C18 项目的一部分。因此，在使用 MPASMX 工具链的项目中或在使用包含至少一个`.asm`文件的 C18 工具链的项目中，会忽略并行 make 选项。
Force makefile regeneration when opening a project（打开项目时强制重新生成 makefile）	取消选中后，允许 MPLAB X IDE 确定在打开项目（例如，在另一台计算机上打开项目）后是否应重新生成 makefile。这是默认选项。 选中后，将在项目打开时强制重新生成 makefile。如果怀疑在您需要时不会重新生成 makefile，请使用该功能。 **注：** 重新生成可能需要一些时间。
Use "clean" target from Makefile（通过 Makefile “清除”目标）	取消选中后，将使用标准清除，即，删除所有编译产物。这种方法速度较快。 选中后，将使用 make 清除，即，执行清除时使用 make 文件。这样可以更好地控制清除的内容。

12.16.3 Build Options 选项卡

选择 *Tools>Options*（对于 macOS 为 *MPLAB X IDE>Preferences*），然后单击 **Embedded** 按钮，**Build Tools**（编译工具）选项卡，可看到下列选项。

确保已安装语言工具，否则它不会显示在“Toolchain”列表中。如果您确定语言工具已安装但不在列表中，请单击 Scan for Build Tools（扫描编译工具）以扫描编译器。如果仍未找到，请单击 Add，将语言工具添加到列表中。

MPLAB X IDE 中包含以下语言工具：

- MPASM 工具链——包含 MPASM 汇编器、MPLINK 链接器和实用程序。

其他工具可以从 Microchip 网站（www.microchip.com）或第三方获得。

表 12-72. Build Tools 选项卡项

项	说明
Toolchain	显示计算机上已安装的语言工具的列表。 为您的项目选择工具并确保（应用于）右侧的路径是正确的。 **Base directory**（根目录）：指向工具主文件夹的路径。 **C compiler**（C 编译器）：C 编译器（如可用）的路径。 **Assembler**：汇编器（如可用）的路径。 **Make command**（Make 命令）：由 MPLAB® X IDE 生成的 make 命令的名称。
Add	向列表中添加新的语言工具项。 此外，还可以考虑使用 **Scan for Build Tools**。

.....（续）

项	说明
Add Custom Compiler（添加定制编译器）	添加定制的编译器。 在这种情况下，您必须指定支持的 Microchip 器件。
Remove	从列表中删除语言工具项。 这不会从计算机中删除语言工具。
Default	单击某个工具，然后单击 **Default** 可将该工具设为选定器件的默认编译器/汇编器。
Scan for Build Tools	在计算机中的各个默认位置而**不是**整个系统中扫描已安装的编译器/汇编器。 如果安装在其他位置，请手动添加编译器。

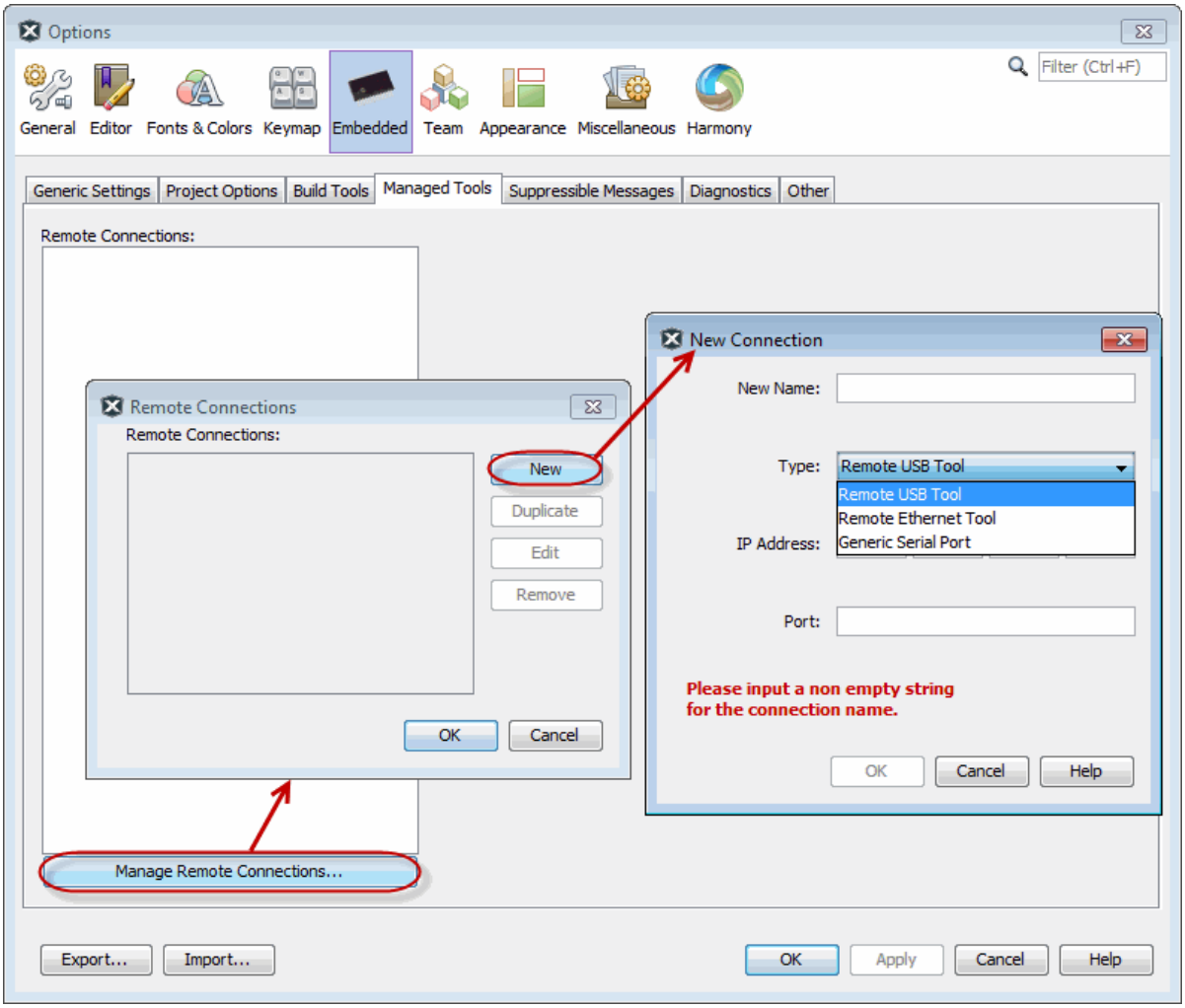
12.16.4 Managed Tools 选项卡

选择 *Tools>Options*（对于 macOS 为 *MPLAB X IDE>Preferences*），然后单击 **Embedded** 按钮，**Managed Tools**（管理的工具）选项卡，可看到下列选项。

选择要使用的远程连接。为以下项添加新的连接：

- Remote USB Tool（远程 USB 工具）（所需插件）
- Remote Ethernet Tool（远程以太网工具）
- Generic Serial Port（通用串行端口）

图 12-41. 新管理的工具

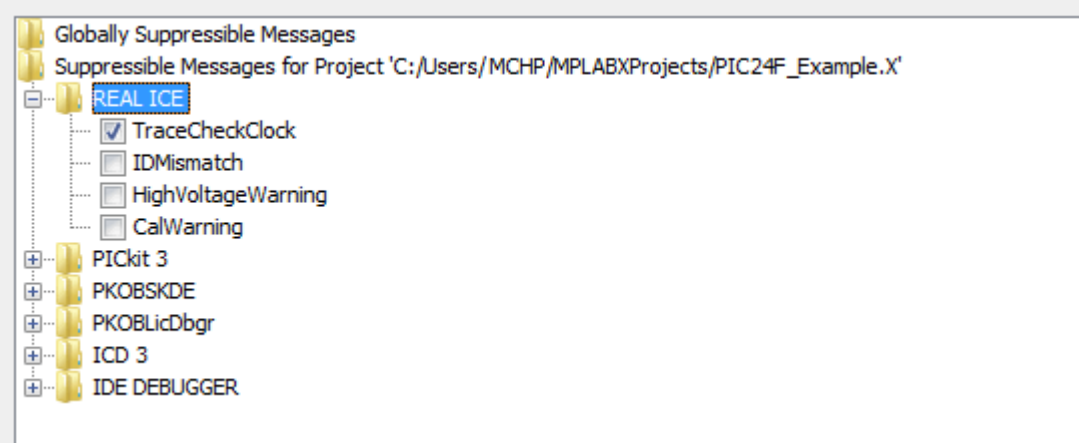


12.16.5 Suppressible Messages 选项卡

选择 *Tools>Options*（对于 macOS 为 *MPLAB X IDE>Preferences*），然后单击 **Embedded** 按钮，**Suppressible Messages**（可禁止的消息）选项卡，可看到下列选项。

选择要禁止的错误和警告消息。双击可用的文件夹，可以进一步选择可用项来进行禁止。

图 12-42. 可禁止的消息列表



12.16.6 Diagnostics 选项卡

选择 *Tools>Options*（对于 macOS 为 *MPLAB X IDE>Preferences*），然后单击 **Embedded** 按钮，**Diagnostics** 选项卡，可看到下列选项。

设置日志文件和其他诊断功能。

表 12-73. Diagnostics 选项卡项

项	说明
Logging Level（日志记录级别）	设置消息的日志记录级别： **OFF**（关闭）：无日志记录。 **SEVERE**（严重）：仅记录严重（错误）消息。 **WARNING**（警告）：仅记录警告消息。 **INFO**（信息）：仅记录信息性的消息。 **CONFIG**（配置）：仅记录配置信息。 **FINE**（略详细）：记录一些模块间通信。 **FINER**（较详细）：记录更多的模块间通信。 **FINEST**（最详细）：记录所有模块间通信。
Log File（日志文件）	日志文件的路径和名称。
USB Circular Log（USB 循环日志）	仅适用于 MPLAB REAL ICE 在线仿真器、MPLAB ICD 3 和 PICkit 3。
Start New Logging Session/Pause Logging（启动新的日志记录会话/暂停日志记录）	单击按钮可开始一个新的日志记录会话或暂停正在进行中的日志记录会话。
Circular USB log file location（循环 USB 日志文件位置）	指定日志文件的路径。
Circular USB log file max size in KBs（以 KB 表示的循环 USB 日志文件最大大小）	指定日志文件的大小。

12.16.7 Other 选项卡

选择 *Tools>Options*（对于 macOS 为 *MPLAB X IDE>Preferences*），然后单击 **Embedded** 按钮，**Other** 选项卡，可看到下列选项。

编辑 C/C++和汇编源文件以及头文件的可接受文件扩展名的列表。此外，设置每种类型的默认扩展名（以粗体显示）。

表 12-74. Other 选项卡项

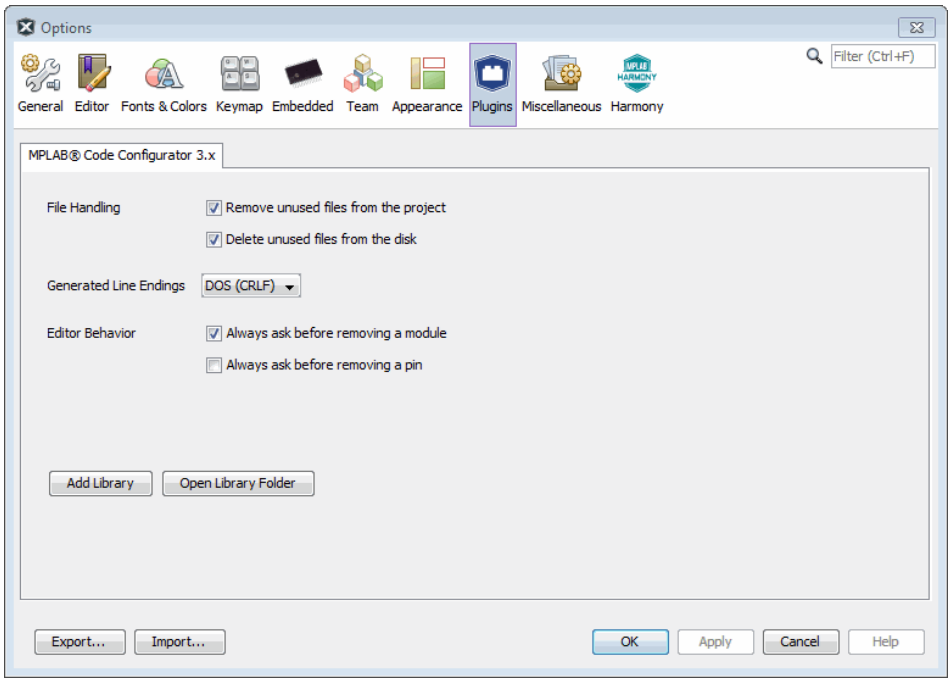
文件扩展名	默认扩展名
C/C++ Header（C/C++头文件）	H、SUNWCCh、**h**、hpp、hxx 和 tcc
C++	C、c++、cc、**cpp**、cxx 和 mm
C	**c**、i 和 m
Fortran	不适用。 MPLAB X IDE 不支持 Fortran 编程。
Assembler	AS、ASM、S、as、asm 和 **s**
Assembler Include（汇编器包含）	INC 和 **inc**

12.17 Tools>Options 窗口，Plugins

选择 *Tools>Options*（对于 macOS 为 *MPLAB X IDE>Preferences*）后单击 **Plugins** 按钮可打开该窗口。

Options 窗口属于 NetBeans 平台窗口。不过，它已针对 MPLAB X IDE 项目进行了定制。如果已安装插件（使用 *Tools>Plugins*），并且它们具有插件选项，可转到该窗口查看可用的插件选项。

图 12-43. Plugin 选项示例



12.18 Trace 窗口

通过选择 *Window>Debugging>Trace*（窗口>调试>跟踪）打开 Trace 窗口。必须处于调试模式才能查看跟踪数据。

跟踪使您可以在 Trace 窗口中记录代码的分步执行并检查该记录。跟踪当前可用于以下工具：

- 软件模拟器

- MPLAB REAL ICE 在线仿真器

在窗口中右键单击某个跟踪列可弹出上下文菜单（第一个表）。根据您正在使用的工具，可能会也可能不会看到所有菜单项。

第二个表定义了与跟踪关联的对话框。

表 12-75. Trace 窗口上下文菜单

菜单项	说明
Symbolic Mode	对于“Instruction”（指令）列，在显示立即数寄存器地址（例如 0x5）和符号寄存器宏（例如 PORTA）之间切换。
Go To	打开 Go To 对话框。 **Trigger**（触发）：转至触发行（0）。 **Top**（顶部）：转至顶部跟踪行。 **Bottom**（底部）：转至底部跟踪行。 **Trace Line**（跟踪行）：指定并转至跟踪行位置。
Go To Source Line	选择跟踪行，然后选择该选项可转至的相应源代码行。
Display Time（显示时间）	对于“Cycle”（周期）列（如果先前不可见，则将显示）： **As Hex Cycle Count**（十六进制周期计数）：以十六进制形式显示周期计数。 **As Decimal Cycle Count**（十进制周期计数）：以十进制形式显示周期计数。 **In Seconds Elapsed**（经过的秒数）：以经过的秒数的形式显示周期计数。 **In Engineering Format**（工程格式）：以相应的工程格式（10^3 的幂）显示周期计数。
Clear Trace File（清除跟踪文件）	清除跟踪显示中的数据。
Find	在跟踪显示中查找项。 打开 Find 对话框。
Output To File	将跟踪数据保存到一个文件中。 打开 Define Range（定义范围）对话框，它将打开 Save 对话框。
Print	打印数据。 打开 Print 对话框。
Adjust Table Columns	自动调整跟踪显示中的列，使之适应数据。
Reload View（重新装入视图）	在暂停时重新装入跟踪显示的原始数据视图。

表 12-76. 跟踪对话框

对话框	说明
Go To	指定要转至的跟踪行。
Find	在跟踪显示中查找行号或其他数据。
Output to File Range（输出到文件范围）	指定要输出到文件的行范围。 单击 **OK** 可转至 Save 对话框，从而将数据保存到一个文本文件中。
Save	将数据保存到一个文本文件中。
Print	在打印之前指定打印机、页面设置和外观。

12.19 Watches 窗口

要打开 Watches 窗口，请选择 *Window>Debugging>Watches*。必须处于调试模式才能观察符号值的变化。

选择全局符号和 SFR，并在程序暂停或运行时（如果支持）观察值的变化。

关于使用 Watches 窗口的信息，请参见 4.16 观察符号值变化。

- Watches 显示
- Watches 菜单
- Fractional Integer Properties 对话框

12.19.1 Watches 显示

下文各部分介绍了以下显示特性：图标、列和操作（按钮）。

图标

图标位于 Name 列中的名称对象的左侧：

表 12-77. Name 列图标

图标	说明
	观察对象
	程序存储器中的观察对象
	对象的静态字段
	对象的非静态字段

列

可通过以下方式更改窗口中显示的列：

- 右键单击标题，然后选中/取消选中要显示的标题。
- 单击窗口右上方的“Change Visible Columns”（更改可见列）图标  。在 Change Visible Columns 对话框中，选中/取消选中要显示的标题。

操作

操作位于窗口左侧的按钮上：

表 12-78. 操作按钮图标

按钮	操作
	翻转按钮： • 显示成员字段的详细（限定）名称。 • 显示成员字段的简要（相对）名称。
	从列表文件中导入观察。单击右键可查看选项。
	将所有观察导出到列表文件。
	设置 Value 字段的默认数字格式。

12.19.2 Watches 菜单

右键单击 Watches 窗口的空白区域（而非一行）可打开 Watches 窗口菜单，其中包含一些基本的功能。

表 12-79. Watches 窗口菜单——窗口

菜单项	说明
New Watch*	添加要观察的新符号。
New Runtime Watch*	**仅限 MPLAB REAL ICE 在线仿真器/软件模拟器。** 为选定符号添加新的运行时观察。
Export All Watches to List File（将所有观察导出到列表文件）	将关于要观察的符号的信息导出到一个文件（.xwatch）中。
Delete All	从 Watches 窗口中删除观察的所有符号。
*此外，还可以在代码中右键单击某个符号，将其添加到 New Watch 或 New Runtime Watch 中。	

右键单击某一行可打开 Watches 菜单，其中的附加功能取决于行中是否存在符号或使用的调试工具。

表 12-80. Watches 窗口菜单——行

菜单项	说明
New Watch	添加要观察的新符号。
New Runtime Watch	**仅限 MPLAB REAL ICE 在线仿真器。** 为选定符号添加新的运行时观察。
Run Time Update Interval（运行时更新间隔）	**仅限 MPLAB REAL ICE 在线仿真器。** 指定符号值的更新速率。 “No Delay”（无延时）将以您的计算机支持的速率尽可能快地进行更新。如果在运行时数据中发现错误，可以通过增加延时来降低更新速度。
Export Selected Watches to List File（将所选观察导出到列表文件）	将关于要观察的符号的信息导出到一个文件（.xwatch）中。 可使用 Shift 加单击（连续选择多个行）或 Ctrl 加单击（单独选择多个行）的方式来选择多个行。
Export Data（导出数据）	将观察数据导出到 CSV 文件或以字面字符串形式导出。
Delete	从观察列表中删除所选符号行。 还可以选择某一行并单击<Delete>键。
Delete All	从 Watches 窗口中删除观察的所有符号。
Display Value Column as（值列显示方式）	以所列格式之一在所选行中显示符号值。另请参见 12.19.3 Fractional/Integer Properties（小数/整数属性）对话框。
User Defined Size（用户定义的大小）	指定列表中程序存储器符号的大小。

12.19.3 Fractional/Integer Properties（小数/整数属性）对话框

此对话框旨在为评估 dsPIC 和 PIC32 器件所支持的全部定点运算模式提供更好的支持。通过从弹出菜单项 *Display Value Column as>Fractional Integer*（值列显示方式>小数/整数）查看该对话框来选择该功能，之后还可选择 *Display Value Column as>Fractional Properties*（值列显示方式>小数属性）。

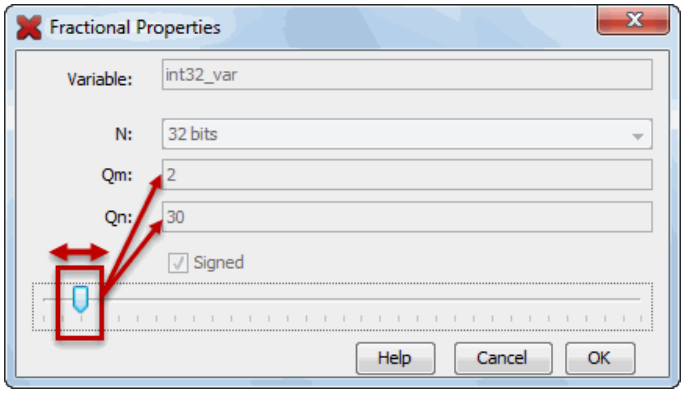
表 12-81. Fractional Properties 对话框项

对话框项	说明
Variable（变量）	变量的名称，供参考。
N*	可用于表示整数的总位数。

.....（续）

对话框项	说明
Qm	定点数据格式的整数部分。 “m”代表用于指定整数部分的位数。
Qn	定点数据格式的小数部分。 “n”代表用于指定小数部分的位数。
Signed（有符号）*	选中可使用一个位来表示有符号值。
Slider（滑动条）	单击并拖动滑动条可选择要进行定点小数值计算的定点位置（Qm 与 Qn）。
*如果变量类型未知，则这些项可编辑。	

图 12-44. Fractional Properties 对话框

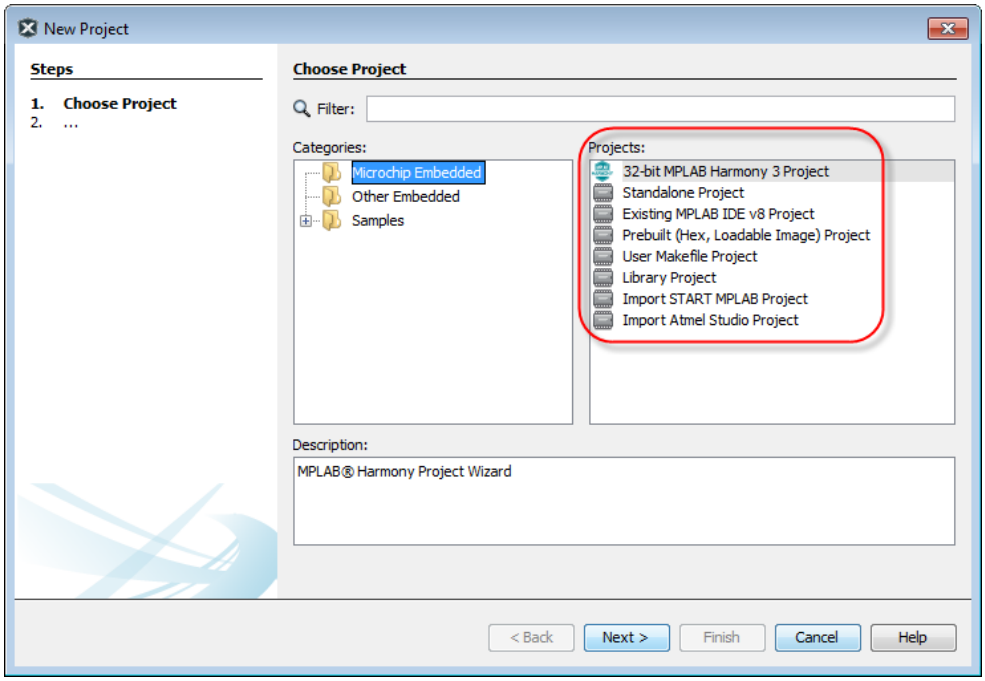


12.20 向导

向导是一系列连续的对话框窗口，可引导您完成整个过程。MPLAB X IDE 中的两个常用向导如下：

- New Project 向导（见创建新项目）。
- New File 向导（见创建新文件）。

图 12-45. New Project 类型



13. NetBeans 窗口和对话框

MPLAB X IDE 窗口和对话框是基本 NetBeans 窗口和对话框与 MPLAB X IDE 特定窗口和对话框的组合。NetBeans 项的帮助主题以白色背景显示，而 MPLAB X IDE 项的帮助主题以黄色背景显示。

本章将介绍 NetBeans 窗口和对话框。关于 MPLAB X IDE 特定窗口和对话框的信息，请参见 12. MPLAB X IDE 窗口和对话框。

13.1 NetBeans 特定窗口和窗口菜单

NetBeans 帮助或 web 文档中对 NetBeans 窗口进行了介绍（见“About This Help File”（关于本帮助文件）或“Preface”（前言））。要打开关于某个窗口的帮助，请单击 Window 选项卡选择它，然后按<F1>键。如果未找到任何帮助，可以选择 *Help>Help Contents* 并单击帮助文件的 **Search** 选项卡来搜索关于该窗口的信息。或者，请参见 NetBeans 帮助主题“Managing IDE Windows”（管理 IDE 窗口）的目录。

要打开大多数窗口，请参见 10.1.11 Window 菜单。

窗口可以进行停靠和取消停靠（右键单击 Window 选项卡），并具有特定于窗口的弹出或上下文菜单，菜单中带有多个项，例如：

- Fill（填充）
- Goto（转至）
- Edit Cells（编辑单元格）

在窗口中或窗口中某个项上单击右键将弹出上下文菜单。大多数内容菜单项也会位于桌面菜单栏的菜单上（见 10.1 菜单）。

通过选择 *Tools>Options*（对于 macOS 为 *MPLAB X IDE>Preferences*），**Miscellaneous** 按钮，**Appearance** 选项卡来设置窗口属性。

13.2 NetBeans 特定对话框

NetBeans 帮助中对 NetBeans 对话框进行了介绍。要打开关于某个对话框的帮助，请单击对话框上的 **Help** 按钮，或者如果没有 Help 按钮，则按<F1>键。如果未找到任何帮助，可以选择 *Help>Help Contents* 并单击帮助文件的 **Search** 选项卡来搜索关于该对话框的信息。

要打开大多数对话框，请参见 10.1 菜单。

14. 配置设置汇总

以下给出了说明如何在代码中为不同语言工具和相关器件设置配置位的示例。关于如何设置配置位的更多信息，请参见语言工具文档。对于一些语言工具，提供了配置设置文档，其中列出器件的所有配置设置。其他情况下，请参考您的器件头文件，查看其中的宏。

另一个选项是使用 Configuration Memory（配置存储器）窗口来设置位，然后单击“Generate Source Code to Output”。请参见 4.19 在 Configuration Bits 窗口中设置配置值。

用于 8 位和 16 位器件的 MPLAB 代码配置器（MCC）与用于 32 位器件的 MPLAB Harmony 这两款工具均可用于生成配置位以及项目代码。关于这些工具的更多信息，请访问 Microchip 网站。

最新工具链：AVR 和 Arm GCC 编译器、MPLAB XC C 编译器和 MPASM 汇编器。

传统工具链：HI-TECH PICC 和 PICC18 编译器、MPLAB C18 编译器、ASM30 汇编器以及 MPLAB C30 和 MPLAB C32 编译器。

14.1 AVR GCC 工具链

大多数配置位包含在器件熔丝字节中。要对熔丝位进行编程，请使用熔丝 API。要使用该 API，应包含<avr/io.h>头文件，该文件可提供设置 AVR 熔丝所需的一切内容。

要编程代码保护配置位或锁定位，请使用 Lockbit API。要使用该 API，应包含<avr/io.h>头文件。

下面给出了使用 FUSES 和 LOCKBITS 宏的示例。关于熔丝 API 和 Lockbit API 的更多信息，请参见：

https://www.microchip.com/webdoc/AVRLibcReferenceManual/group__avr__fuse.html

https://www.microchip.com/webdoc/AVRLibcReferenceManual/group__avr__lock.html

示例：ATtiny861 MCU

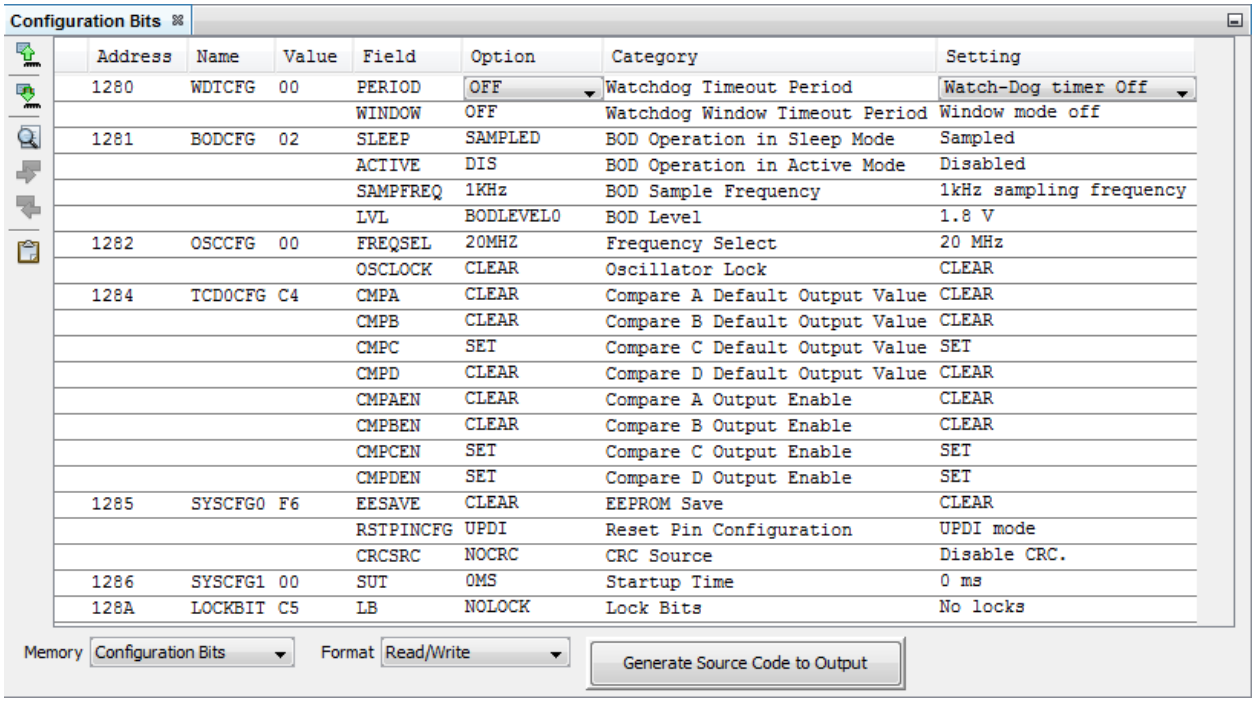
```
// ATtiny817 Configuration Bit Settings

// 'C' source line config statements

#include <avr/io.h>

FUSES = {
    0x00, // WDTCFG{PERIOD=OFF, WINDOW=OFF}
    0x02, // BODCFG{SLEEP=SAMPLED, ACTIVE=DIS, SAMPFREQ=1KHz, LVL=BODLEVEL0}
    0x00, // OSCCFG{OSCLOCK=CLEAR}
    0x00, // Reserved
    0xC4, // TCD0CFG{CMPA=CLEAR, CMPB=CLEAR, CMPC=SET, CMPD=CLEAR, CMPAEN=CLEAR,
CMPBEN=CLEAR, CMPCEN=SET, CMPDEN=SET}
    0xF6, // SYSCFG0{EESAVE=CLEAR, RSTPINCFG=UPDI, CRCSRC=NOCRC}
    0x00, // SYSCFG1{SUT=0MS}
    0x00, // APPEND
    0x00, // BOOTEND
};

LOCKBITS = {
    0xC5, // LOCKBIT{LB=NOLOCK}
};
```

以上示例是在 Configuration Bits 窗口（*Window>Target Memory Views>Configuration Bits*（窗口>目标存储器视图>配置位））中生成的。

首次打开 Configuration Bits 窗口时，数据将为红色，这意味着您需要读取器件存储器，请使用“Read Configuration Bits”（读取配置位）图标。使用“Program Configuration Bits”（编程配置位）图标对窗口中的数据进行更改，然后将更改写回器件。

要保留代码中的设置，应单击“Generate Source Code to Output”按钮，将该代码置于 Output 窗口中，以便将其复制粘贴到您的代码中，或者单击“Insert Source Code into Editor”图标，将该代码置于编辑器窗口中的光标处。

图 14-1. 示例：ATtiny817 MCU



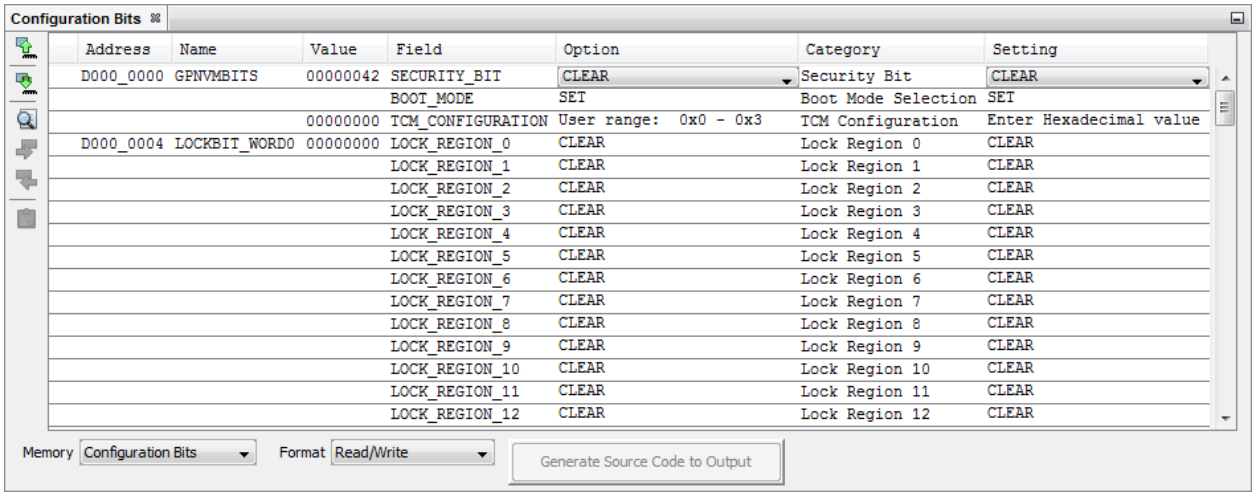
14.2 Arm GCC 工具链

大多数配置位包含在器件 GPNVM 位中。其中，GPNVM0 表示安全位。

要编程代码保护配置位，请使用锁定位。

要编程配置位，请使用 Configuration Bits 窗口（*Window>Target Memory Views>Configuration Bits*）。首次打开 Configuration Bits 窗口时，数据将为红色，这意味着您需要读取器件存储器，请使用“Read Configuration Bits”图标。使用“Program Configuration Bits”图标对窗口中的数据进行更改，然后将更改写回器件。

示例：ATSAME70Q21 MCU



14.3 XC 工具链

要创建尽可能便于移植的代码，请参见以下每个文档中的“公共编译器接口（CCI）”章节的 config 宏：

- 《MPLAB® XC8 C 编译器用户指南》（DS50002053D_CN）或相关帮助文件
- 《MPLAB® XC16 C 编译器用户指南》（DS50002071E_CN）或相关帮助文件
- 《MPLAB® XC32 C/C++编译器用户指南》（DS50001686G_CN）或相关帮助文件

要读取、更改和编程配置位，请参见 4.19 在 Configuration Bits 窗口中设置配置值。

14.4 MPASM 工具链

使用两种类型的汇编器伪指令来设置器件配置位：__config 和 config。不要在同一代码中混合使用__config 和 config。

14.4.1 __config

伪指令__config 用于 PIC10/12/16 MCU。它也可用于 PIC18 MCU（不包括 PIC18FXXJ 器件），但建议 PIC18 MCU 使用 config 伪指令。语法如下所示：

```
__config expr        ;For a single configuration word
```

或

```
__config addr, expr  ;For multiple configuration word
```

其中：

addr	配置字的地址。可以为立即数，但通常通过一个宏表示。 **注：** 宏必须按寄存器升序列出。
expr	代表指定配置位设置值的表达式。可以为立即数，但通常通过一个宏或多个宏的逻辑与表示。

宏在器件包含文件（*.inc）中指定，器件包含文件位于 Windows 操作系统（OS）默认目录中：

```
C:\Program Files\Microchip\MPLABX\mpasmx
```

伪指令的大小写**无关紧要**；__CONFIG 或__config 均可接受。宏的大小写应与头文件中匹配。

请参见《MPASM™汇编器、MPLINK™目标链接器、MPLIB™目标库管理器用户指南》（DS33014L_CN）中的“__config——设置处理器配置位”。

示例——PIC10/12/16 MCU

```
#include p16f877a.inc
;Set oscillator to HS, watchdog time off, low-voltage prog. off
__CONFIG _HS_OSC & _WDT_OFF & _LVP_OFF
```

示例——PIC18 MCU

```
#include p18f452.inc
;Oscillator switch enabled, RC oscillator with OSC2 as I/O pin.
__CONFIG    _CONFIG1, _OSCS_OFF_1 & _RCIO_OSC_1
;Watch Dog Timer enable, Watch Dog Timer PostScaler count - 1:128
__CONFIG    _CONFIG3, _WDT_ON_3 & _WDTPS_128_3
```

14.4.2 config

伪指令 config 用于 PIC18 MCU（包括 PIC18FXXJ 器件）。语法如下所示：

```
config setting=value [, setting=value]
```

其中：

setting	代表一个或多个配置位的宏。
value	代表指定配置位设置值的宏。多个设置可以在单个行上定义，使用逗号分隔。单个配置字节的设置也可以在多个行上定义。

宏在器件包含文件（*.inc）中指定，器件包含文件位于 Windows 操作系统默认目录中：

```
C:\Program Files\Microchip\MPLABX\mpasmx
```

伪指令的大小写无关紧要；__CONFIG 或__config 均可接受。 宏的大小写应与头文件中匹配。

请参见“*MPASM Assembler, MPLINK Object Linker, MPLIB Object Librarian User’s Guide*”（DS30003014）中的“config——设置处理器配置位（PIC18 MCU）”。

示例——PIC18 MCU

```
#include p18f452.inc
;Oscillator switch enabled, RC oscillator with OSC2 as I/O pin.
CONFIG     OSCS=ON, OSC=LP
;Watch Dog Timer enable, Watch Dog Timer PostScaler count - 1:128
CONFIG     WDT=ON, WDTPS=128
```

14.5 HI-TECH PICC 工具链

使用在 htc.h 头文件中指定的宏来设置 PIC10/12/16 MCU 器件的配置字：__CONFIG(x);

其中

x	代表指定配置位设置值的表达式。可以为立即数，但通常通过一个宏或多个宏的逻辑与表示。

宏在器件头文件（*.h）中指定，器件头文件位于 Windows 操作系统默认目录中：

```
C:\Program Files\HI-TECH Software\PICC\vx.xx\include
```

其中，vx.xx 是编译器的版本号。

对于具有多个配置字单元的器件，随后每次调用__CONFIG()都将按顺序修改下一个配置字。

宏的大小写应与相关头文件中匹配。对于 htc.c，__CONFIG()是正确的，但__config()不正确。

请参见“*HI-TECH C for PIC10/12/16 User’s Guide*”（DS51865）中的“Configuration Fuses”。

PICC 示例

```
#include <htc.h>
__CONFIG(WDTDIS & XT & UNPROTECT); // Program config. word 1
__CONFIG(FCMEN);                   // Program config. word 2
```

14.6 HI-TECH PICC-18 工具链

使用在 htc.h 头文件中指定的宏来设置 PIC18 MCU 器件的配置字：

```
__CONFIG(n,x);
```

其中

n	配置寄存器编号。
x	代表指定配置位设置值的表达式。可以为立即数，但通常通过一个宏或多个宏的逻辑与表示。

宏在器件头文件（*.h）中指定，器件头文件位于 Windows 操作系统默认目录中：

```
C:\Program Files\HI-TECH Software\PICC\vx.xx\include
```

其中，vx.xx 是编译器的版本号。

宏的大小写应与相关头文件中匹配。对于 htc.c，__CONFIG()是正确的，但__config()不正确。

请参见“*HI-TECH C Tools for the PIC18 MCU Family* 中的“Configuration Fuses”。

PICC-18 示例

```
#include <htc.h>
//Oscillator switch enabled, RC oscillator with OSC2 as I/O pin.
__CONFIG(1, LP);
//Watch Dog Timer enable, Watch Dog Timer PostScaler count - 1:128
__CONFIG(2, WDTEN & WDTPS128);
```

14.7 C18 工具链

#pragma config 伪指令指定要由应用程序使用的特定于器件的配置设置（即，配置位）：

```
#pragma config setting-list
```

其中

setting-list	一个或多个 setting-name = value-name 宏对的列表，使用逗号分隔。

宏在器件头文件（*.h）中指定，器件头文件位于 Windows 默认目录中：

```
C:\program files\microchip\mplabc18\vx.xx\.h
```

Pragma 伪指令的大小写**无关紧要**；#PRAGMA CONFIG 或#pragma config 均可接受。宏的大小写应与头文件中匹配。

请参见《MPLAB® C18 C 编译器用户指南》（DS51288J_CN）中的“Pragms”和“#pragma config”。

示例

```
#include <p18cxxx.h>
/*Oscillator switch enabled, RC oscillator with OSC2 as I/O pin.*/
#pragma config OSCS = ON, OSC = LP
/*Watch Dog Timer enable, Watch Dog Timer PostScaler count - 1:128*/
#pragma config WDT = ON, WDTPS = 128
```

14.8 ASM30 工具链

使用在器件包含文件中指定的宏来设置配置位：

```
config __reg, value
```

其中

__reg	配置寄存器名称宏。
value	代表指定配置位设置值的表达式。可以为立即数，但通常通过一个宏或多个宏的逻辑与表示。

宏在器件包含文件（*.inc）中指定，器件包含文件位于 Windows 操作系统默认目录中：

```
C:\Program Files\Microchip\MPLAB ASM30 Suite\Support\device\inc
```

其中，device 是选定 16 位器件的缩写，例如 PIC24H 或 dsPIC33F。

宏的大小写应与相关头文件中匹配。例如，config 是正确的，但 CONFIG 不正确。

请参见《MPLAB® ASM30、MPLAB® LINK30 和实用程序用户指南》（DS51317F_CN）中的“配置存储区中的输出段”。

示例

```
.include "p30fxxxx.inc"
;Clock switching off, Fail-safe clock monitoring off,
; Use External Clock
config __FOSC, CSW_FSCM_OFF & XT_PLL16
;Turn off Watchdog Timer
config __FWDT, WDT_OFF
```

14.9 C30 工具链

使用两种类型的编译器宏来设置器件配置位：一种类型用于 PIC24F MCU，一种类型用于 dsPIC30F 和 dsPIC33F/PIC24H 器件。

14.9.1 PIC24F 配置设置

使用在器件头文件中提供的宏来设置配置位：

```
_confign(value);
```

其中

n	配置寄存器编号。
value	代表指定配置位设置值的表达式。可以为立即数，但通常通过一个宏或多个宏的逻辑与表示。

宏在器件头文件（*.h）中指定，器件头文件位于 Windows 操作系统默认目录中：

```
C:\Program Files\Microchip\MPLAB C30\support\PIC24F\h
```

宏的大小写应与相关头文件中匹配。例如，_CONFIG1 是正确的，但_config1 不正确。

示例——PIC24F MCU

```
#include “p24Fxxxx.h”
//JTAG off, Code Protect off, Write Protect off, COE mode off, WDT off
_CONFIG1( JTAGEN_OFF & GCP_OFF & GWRP_OFF & COE_OFF & FWDTEN_OFF )
```

```
//Clock switching/monitor off, Oscillator (RC15) on,
// Oscillator in HS mode, Use primary oscillator (no PLL)
_CONFIG2( FCKSM_CSDCMD & OSCIOFNC_ON & POSCMOD_HS & FNOSC_PRI )
```

14.9.2 dsPIC30F/33F/PIC24H 配置设置

使用在器件头文件中提供的宏来设置配置位：

```
_reg(value);
```

其中

_reg	配置寄存器名称宏。
value	代表指定配置位设置值的表达式。可以为立即数，但通常通过一个宏或多个宏的逻辑与表示。

宏在器件头文件（*.h）中指定，器件头文件位于 Windows 操作系统默认目录中：

```
C:\Program Files\Microchip\MPLAB C30\support\device\h
```

其中，device 是选定 16 位器件的缩写，即 PIC24H、dsPIC30F 或 dsPIC33F。

宏的大小写应与相关头文件中匹配。例如，_FOSC 是正确的，但_fosc 不正确。

请参见《用于 PIC24 MCU 和 dsPIC® DSC 的 MPLAB® C 编译器用户指南》（DS51284H_CN）中的“配置位设置宏”。

示例——dsPIC30F DSC

```
#include “p30fxxxx.h”
//Clock switching and failsafe clock monitoring off,
// Oscillator in HS mode
_FOSC(CSW_FSCM_OFF & HS);
//Watchdog timer off
_FWDT(WDT_OFF);
//Brown-out off, Master clear on
_FBORPOR(PBOR_OFF & MCLR_EN);
```

示例——dsPIC33F/PIC24H 器件

```
#include “p33fxxxx.h”
// Use primary oscillator (no PLL)
_FOSCSEL(FNOSC_PRI);
//Oscillator in HS mode
_FOSC(POSCMD_HS);
//Watchdog timer off
_FWDT(FWDTEN_OFF);
//JTAG off
_FICD(JTAGEN_OFF);
```

14.10 C32 工具链

#pragma config 伪指令指定要由应用程序使用的特定于器件的配置设置（即，配置位）：

```
# pragma config setting-list
```

其中

setting-list:	一个或多个 setting-name = value-name 宏对的列表，使用逗号分隔。

宏在器件头文件（*.h）中指定，器件头文件位于 Windows 操作系统默认目录中：

```
C:\Program Files\Microchip\MPLAB C32\pic32mx\include\proc
```

Pragma 伪指令的大小写**无关紧要**：#PRAGMA CONFIG 或#pragma config 均可接受。宏的大小写应与头文件中匹配。

请参见“*MPLAB® C Compiler for PIC32 MCUs User’s Guide*”（DS50001686）中的“配置位访问”。

示例

```
#include “p32xxxx.h”
//Enables the Watchdog Timer,
// Sets the Watchdog Postscaler to 1:128
#pragma config FWDTEN = ON, WDTPS = PS128
//Selects the HS Oscillator for the Primary Oscillator
#pragma config POSCMOD = HS
```

15. 术语表

绝对段（Absolute Section）
具有链接器不能更改的固定（绝对）地址的 GCC 编译器段。

绝对变量/函数（Absolute Variable/Function）
使用 OCG 编译器的@ address 语法放置在绝对地址的变量或函数。

快速存取存储区（Access Memory）
仅限 PIC18——PIC18 器件中的一些特殊寄存器，对这些寄存器的访问与存储区选择寄存器（Bank Select Register，BSR）的设置无关。

访问入口点（Access Entry Point）
访问入口点提供了一种方法，可跨段将控制转移到某个可能未在链接时定义的函数。它们支持独立链接引导段和安全应用程序段。

地址（Address）
标识存储器中位置的值。

字母字符（Alphabetic Character）
字母字符指属于拉丁字母表（a,b,⋯,z,A,B, ⋯,Z）中字母的字符。

字母数字字符（Alphanumeric）
字母数字字符由字母字符和十进制数字（0,1, ⋯,9）组成。

逻辑与断点（ANDed Breakpoint）
为中断设置“逻辑与”条件，即只有断点 1 和断点 2 同时发生时，才会暂停程序。只有数据断点和程序存储器断点同时发生时，才会完成它。

匿名结构体（Anonymous Structure）
16 位 C 编译器——未命名的结构体。

PIC18 C 编译器——属于 C 联合体的成员的未命名结构体。匿名结构体成员可以像包含结构体的联合体的成员一样进行访问。例如，在以下代码中，hi 和 lo 是联合体 caster 中的匿名结构体的成员。

```
union castaway
int intval;
struct {
char lo; //accessible as caster.lo
char hi; //accessible as caster.hi
};
} caster;
```

ANSI
美国国家标准学会，是美国负责制订和批准标准的组织。

应用（Application）
可由 PIC®单片机控制的一组软硬件。

归档/归档器（Archive/Archiver）
归档/库是可重定位目标模块的集合。由将多个源文件编译/汇编为目标文件，然后使用归档器/库管理器将目标文件组合为一个归档/库文件生成。可将归档/库与目标模块和其他归档/库链接，生成可执行代码。

ASCII
美国信息交换标准码是使用 7 个二进制位来表示每个字符的字符集编码。它包括大写和小写字母、数字、符号以及控制字符。

汇编语言/汇编器（Assembly/Assembler）
汇编语言是以符号形式描述二进制机器码的编程语言。汇编器是将汇编语言源代码翻译成机器码的语言工具。

已分配段（Assigned Section）
在链接器命令文件中已分配到目标存储区的 GCC 编译器段。

异步（Asynchronously）
不同时发生的多个事件。通常用来指可能在处理器执行过程中的任意时刻发生的中断。

异步激励（Asynchronous Stimulus）
为模拟到所模拟器件的外部输入而生成的数据。

属性（Attribute）
C 语言程序中变量或函数的 GCC 特性，用于描述特定于机器的性质。

属性，段（Attribute, Section）
段的 GCC 特性，如“可执行”、“只读”或“数据”，它们可在汇编器.section 伪指令中指定为标志。

二进制（Binary）
使用数字 0 和 1，以 2 为基数的计数体制。最右边的位表示 1 的倍数，右边第二位表示 2 的倍数，右边第三位表示 2^2 = 4 的倍数，以此类推。

书签（Bookmarks）
使用书签可轻松地查找文件中的指定行。

在编辑器工具栏中选择 Toggle Bookmarks（翻转书签）可添加/删除书签。单击该工具栏上的其他图标可移动到下一个或上一个书签。

C/C++
C 语言是一种通用编程语言，具有简练的表达式、现代控制流和数据结构，以及丰富的操作符。C++是 C 语言的面向对象版本。

校准存储区（Calibration Memory）
用于保存 PIC 单片机片内 RC 振荡器或其他外设校准值的特殊功能寄存器或通用寄存器。

中央处理单元（Central Processing Unit）
器件的一部分，负责取出要执行的正确指令，对指令进行译码，然后执行指令。如果有必要，它和算术逻辑单元（Arithmetic Logic Unit，ALU）一起工作来完成指令的执行。它控制程序存储器的地址总线、数据存储器的地址总线和对堆栈的访问。

清除（Clean）
清除会删除活动项目的所有中间项目文件，例如目标文件、hex 文件和调试文件。编译项目时，将基于其他文件重新创建这些文件。

COFF
公共目标文件格式。这种格式的目标文件包含机器码、调试及其他信息。

命令行接口（Command Line Interface）
仅基于文本输入和输出，在程序和其用户之间进行通信的一种方式。

已编译堆栈（Compiled Stack）
编译器管理的存储区，在其中为变量静态分配空间。当软件堆栈或硬件堆栈机制无法在目标器件上高效实现时，将替换为已编译堆栈。

编译器（Compiler）
将用高级语言编写的源文件翻译成机器码的程序。

条件汇编（Conditional Assembly）
基于指定表达式在汇编时的值包含或忽略的汇编语言代码。

条件编译（Conditional Compilation）
只有当预处理器伪指令指定的某个常量表达式为真时才编译程序段的操作。

配置位（Configuration Bits）
可对其编程来设置 PIC MCU 或 dsPIC DSC 工作模式的专用位。配置位可或不可再编程。

控制伪指令（Control Directive）
汇编语言代码中根据汇编时指定表达式的值包含或忽略代码的伪指令。

CPU
请参见 *中央处理单元*。

交叉引用文件（Cross Reference File）
引用符号表的一个文件及引用符号的文件列表。如果定义了符号，列出的第一个文件是定义的位置。其他文件包含对符号的引用。

数据伪指令（Data Directive）
数据伪指令是指控制汇编器的程序或数据存储空间分配，并提供通过符号（即有意义的名称）引用数据项的方法的伪指令。

数据存储器（Data Memory）
在 Microchip MCU 和 DSC 器件中，数据存储器（RAM）由通用寄存器（GPR）和特殊功能寄存器（SFR）组成。某些器件还有 EEPROM 数据存储器。

数据监视与控制界面（Data Monitor and Control Interface，DMCI）
数据监视与控制界面（DMCI）是 MPLAB X IDE 中的一个工具。该界面可以对项目中的应用程序变量提供动态输入控制。应用程序生成的数据可以使用 4 个可动态分配图形窗口中的任意一个以图形方式进行查看。

调试/调试器（Debug/Debugger）
请参见 *ICE/ICD*。

调试信息（Debugging Information）
编译器和汇编器选项，在选中时，它们将提供不同程度的信息来用于调试应用程序代码。关于选择调试选项的详细信息，请参见编译器或汇编器文档。

弃用功能（Deprecated Feature）
由于历史原因仍然支持但最终将逐步淘汰且不再使用的功能。

器件编程器（Device Programmer）
用于对电可编程半导体器件（如单片机）进行编程的工具。

数字信号控制器（Digital Signal Controller）
数字信号控制器（DSC）是具有数字信号处理能力的单片机（如 Microchip 的 dsPIC DSC 器件）。

数字信号处理/数字信号处理器（Digital Signal Processing/Digital Signal Processor）
数字信号处理（DSP）指数字信号以及已转换为数字形式（经过采样的）的一般模拟信号（声音或图像）的计算机处理。数字信号处理器是设计为用于数字信号处理的微处理器。

伪指令（Directive）
源代码中控制语言工具操作的语句。

下载（Download）
下载是将数据从主机发送到其他设备（如仿真器、编程器或目标板）的过程。

DWARF
使用任意记录格式调试。DWARF 是用于 ELF 文件的调试信息格式。

EEPROM
电可擦除的可编程只读存储器。一种可电擦除的特殊 PROM。一次写或擦除一个字节数据。EEPROM 即使电源关闭时也能保留内容。

ELF
可执行链接格式。这种格式的目标文件包含机器码。调试和其他信息使用 DWARF 指定。ELF/DWARF 可提供优于 COFF 的优化代码调试。

仿真/仿真器（Emulation/Emulator）
请参见 *ICE/ICD*。

尾数法（Endianness）
多字节对象中的字节存储顺序。

环境（Environment）
MPLAB PM3——包含关于如何对器件编程的文件的文件夹。该文件夹可以转移到 SD/MMC 卡。

Epilogue
编译器生成代码的一部分，负责释放堆栈空间、恢复寄存器以及执行运行时模型中指定的任何其他特定于机器的要求。此代码在给定函数的任何用户代码之后、紧接在函数返回之前执行。

EPROM
可擦除的可编程只读存储器。通常通过紫外线照射来擦除的可编程只读存储器。

错误/错误文件（Error/Error File）
错误报告使程序不能继续处理的问题。而且，当问题比较明显时，错误还能标识出源文件名和行号。错误文件包含由语言工具生成的错误消息和诊断信息。

事件（Event）
对可能包含地址、数据、次数计数、外部输入、周期类型（取指和读/写）及时间戳的总线周期的描述。事件用于描述触发、断点和中断。

可执行代码（Executable Code）
可装入来执行的软件。

导出（Export）
以标准化的格式将数据从 MPLAB IDE/MPLAB X IDE 发送出。

表达式（Expression）
用算术或逻辑运算符分隔开的常量和/或符号的组合。

扩展单片机模式（Extended Microcontroller Mode）
在扩展单片机模式下，既可使用片内程序存储器，也可使用外部存储器。如果程序存储器地址大于 PIC18 器件的内部存储空间，执行自动切换到外部存储器。

扩展模式（Extended Mode）（PIC18 MCU）
在扩展模式下，编译器将使用扩展指令（即 ADDFSR、ADDULNK、CALLW、MOVSF、MOVSS、PUSHL、SUBFSR 和 SUBULNK）以及立即数变址寻址。

外部标号（External Label）
具有外部链接的标号。

外部链接（External Linkage）
如果可以在定义函数或变量的模块外部对函数或变量进行引用，则函数或变量具有外部链接。

外部符号（External Symbol）
具有外部链接的标识符符号。这可能是一个引用或一个定义。

外部符号解析（External Symbol Resolution）
链接器收集所有输入模块的外部符号定义来解析所有外部符号引用的过程。没有相应定义的任何外部符号引用都会导致报告链接器错误。

外部输入线（External Input Line）
用于根据外部信号设置事件的外部输入信号逻辑探针线（TRIGIN）。

外部 RAM（External RAM）
芯片外部的读/写存储器。

致命错误（Fatal Error）
引起编译立即停止的错误。不产生其他消息。

文件寄存器（File Register）
片内数据存储器，包括通用寄存器（GPR）和特殊功能寄存器（SFR）。

筛选器（Filter）
通过选择确定在跟踪显示或数据文件中包含/排除哪些数据。

修正（Fixup）
在由链接器重定位之后，使用绝对地址替换目标文件符号引用的过程。

闪存（Flash）
按块（而不是按字节）写或擦除数据的一种 EEPROM。

FNOP
强制空操作。强制 NOP 周期是双周期指令的第二个周期。由于 PIC 单片机的架构是流水线型，在执行当前指令的同时预取物理地址空间中的下一条指令。但是，如果当前指令改变了程序计数器，那么这条预取的指令就被忽略了，导致一个强制 NOP 周期。

帧指针（Frame Pointer）
引用堆栈中地址，并将基于堆栈的参数和基于堆栈的局部变量分隔开的指针。为访问当前函数的局部变量和其他值提供了方便。

独立（Free-Standing）
一种接受任何不使用复杂类型的严格符合程序的实现，而且在这种实现中，对库条款（ANSI 89 标准条款第 7 条）中规定的特性的使用，仅限于标准头文件<float.h>、<iso646.h>、<limits.h>、<stdarg.h>、<stdbool.h>、<stddef.h>和<stdint.h>的内容。

GPR
通用寄存器。器件数据存储器（RAM）的一部分，作为一般用途。

暂停（Halt）
停止程序执行。执行 Halt 与在断点处停止相同。

堆（Heap）
用于动态存储器分配的存储区，其中的存储器块按运行时确定的任意顺序进行分配和释放。

Hex 代码/Hex 文件（Hex Code/Hex File）
Hex 代码是以十六进制格式代码存储的可执行指令。Hex 代码包含在 hex 文件中。

十六进制（Hexadecimal）
使用数字 0-9 以及字母 A-F（或 a-f），以 16 为基数的计数体制。字母 A-F 表示值为 10-15（十进制）的十六进制数字。最右边的位表示 1 的倍数，右边第二位表示 16 的倍数，右边第三位表示 $16^2 = 256$ 的倍数，以此类推。

高级语言（High Level Language）
编写程序的语言，它比汇编语言更不依赖于具体的处理器。

ICE/ICD
*在线仿真器/在线调试器：*用于对目标器件进行调试和编程的硬件工具。仿真器具有比调试器更多的功能，例如跟踪。

*在线仿真/在线调试：*使用在线仿真器或调试器进行仿真或调试的行为。

*-ICE/-ICD：*带有在线仿真或调试电路的器件（MCU 或 DSC）。该器件总是安装在仿真/调试头板上，并用于通过在线仿真器或调试器进行调试。

ICSP
在线串行编程。使用串行通信和最少的器件引脚对 Microchip 嵌入式器件进行编程的方法。

IDE
集成开发环境，如 MPLAB IDE/MPLAB X IDE。

标识符（Identifier）
函数名或变量名。

IEEE
电气电子工程师协会。

导入（Import）
将数据从外部源（例如，hex 文件）传送到 MPLAB IDE/MPLAB X IDE 中。

已初始化数据（Initialized Data）
用初始值定义的数据。在 C 中，

```
int myVar=5;
```

定义了将存放到已初始化数据段中的一个变量。

指令集（Instruction Set）
特定处理器理解的机器语言指令的集合。

指令（Instruction）
告知中央处理单元执行特定操作，并可能包含操作中要使用的数据的位序列。

内部链接（Internal Linkage）
如果不能从定义函数或变量的模块外部访问它们，则这样的函数或变量具有内部链接。

国际标准化组织（International Organization for Standardization）
制订许多行业和技术（包括计算和通信）方面的标准的一个组织。也称为 ISO。

中断（Interrupt）
传递到 CPU 的信号，它使 CPU 暂停执行正在运行的应用程序，将控制权转交给中断服务程序（ISR），以处理事件。在完成 ISR 时，将恢复应用程序的正常执行。

中断处理程序（Interrupt Handler）
发生中断时处理特殊代码的程序。

中断服务请求（Interrupt Service Request，IQR）
使处理器暂停正常的指令执行并开始执行中断处理程序的事件。某些处理器有几种中断请求事件，允许具有不同优先级的中断。

中断服务程序（Interrupt Service Routine，ISR）
语言工具：处理中断的函数。

MPLAB IDE/MPLAB X IDE：当产生中断时进入的用户生成代码。代码在程序存储器中的位置通常取决于所产生中断的类型。

中断向量（Interrupt Vector）
中断服务程序或中断处理程序的地址。

左值（L-value）
引用可被检查和/或修改的对象的表达式。左值表达式用在赋值的左侧。

延时（Latency）
事件与其得到响应之间的延迟时间。

库/库管理器（Library/Librarian）
请参见*归档/归档器*。

链接器（Linker）
将目标文件和库组合起来生成可执行代码并解析一个模块对另外一个模块引用的语言工具。

链接描述文件（Linker Script File）
链接描述文件是链接器的命令文件。它们定义链接选项并描述目标平台上的可用存储器。

列表伪指令（Listing Directive）
列表伪指令是控制汇编器列表文件格式的伪指令。它们允许指定标题、分页及其他列表控制。

列表文件（Listing File）
列表文件是列出为每条 C 源语句生成的机器码以及源文件中遇到的汇编指令、汇编器伪指令或宏的 ASCII 文本文件。

小尾数法（Little Endian）
多字节数据的数据存储顺序机制，在这种机制中，低字节存储在较低的地址中。

局部标号（Local Label）
局部标号是用 LOCAL 伪指令在一个宏内部定义的标号。这些标号特定于宏实例化的一个给定实例。也就是说，声明为 local 的符号和标号在遇到 ENDM 宏后不再可访问。

逻辑探针（Logic Probe）
Microchip 的某些仿真器最多可连接 14 个逻辑探针。逻辑探针提供外部跟踪输入、触发输出信号、+5V 和公共接地端。

环回测试板（Loop-Back Test Board）
用于测试 MPLAB REAL ICE 在线仿真器的功能。

LVDS
低电压差分信号传输。一种通过铜线进行高速（每秒千兆位）数据传输的低噪声、低功耗、低幅值方法。

对于标准 I/O 信号传输，数据存储依赖于实际电压大小。电压值会受线路长度影响（线路越长，电阻就越高，这会使电压下降）。但对于 LVDS，存储数据仅通过正负电压值判断，而不是实际电压大小。因此，数据可以传输更长的线路距离，同时保持干净、一致的数据流。

来源：http://www.webopedia.com/TERM/L/LVDS.html

机器码（Machine Code）
处理器实际读和解释的计算机程序的表示。二进制机器码程序由一系列机器指令（可能还包含数据）组成。某个特定处理器的所有可用指令的集合称为它的“指令集”。

机器语言（Machine Language）
特定中央处理单元的指令集，不需翻译即可供处理器使用。

宏（Macro）
宏指令。以缩写形式表示指令序列的指令。

宏伪指令（Macro Directive）
控制宏定义体中执行和数据分配的伪指令。

Makefile
将用于 Make 项目的指令导出到一个文件中。可使用该文件通过 make 指令在 MPLAB IDE/MPLAB X IDE 外 Make 项目。

Make 项目（Make Project）
重新编译应用程序的命令，仅编译自上次编译完成后更改了的源文件。

MCU
单片机。microcontroller 的缩写形式。也写作 uC。

存储器模型（Memory Model）
对于 C 编译器，指应用程序可使用的存储区的表示。对于 PIC18 C 编译器，指一种描述，它指定指向程序存储器的指针的位数。

消息（Message）
显示出来的文本，警告在语言工具的操作中可能存在的问题。消息不会停止操作。

单片机（Microcontroller）
高度集成的芯片，它包括 CPU、RAM、程序存储器、I/O 端口和定时器。

单片机模式（Microcontroller Mode）
PIC18 单片机的一种程序存储器配置。在单片机模式下，仅允许内部执行。因此，在单片机模式下仅可使用片内程序存储器。

微处理器模式（Microprocessor Mode）
PIC18 单片机的一种程序存储器配置。在微处理器模式下，不使用片内程序存储器。整个程序存储器映射到外部。

助记符（Mnemonic）
可直接翻译成机器码的文本指令。也称为操作码。

模块（Module）
执行预处理器伪指令后，源文件的预处理输出。也称为翻译单元。

MPASM™汇编器（MPASM™ Assembler）
Microchip Technology 用于 PIC 单片机、KeeLoq®器件和 Microchip 存储器器件的可重定位宏汇编器。

用于器件的 MPLAB 语言工具（MPLAB Language Tool for Device）
用于指定器件的 Microchip 的 C 编译器、汇编器和链接器。根据应用中所使用的器件来选择语言工具类型。例如，如果为 PIC18 MCU 编写 C 代码，就选择用于 PIC18 MCU 的 MPLAB C 编译器。

MPLAB ICD
Microchip 的在线调试器，与 MPLAB IDE/MPLAB X IDE 配合使用。请参见 ICE/ICD。

MPLAB IDE/MPLAB X IDE
Microchip 的集成开发环境。MPLAB IDE/MPLAB X IDE 随附编辑器、项目管理器和软件模拟器。

MPLAB PM3
Microchip 的器件编程器。用于对 PIC18 单片机和 dsPIC®数字信号控制器进行编程。可与 MPLAB IDE/MPLAB X IDE 配合使用，也可单独使用。代替 PRO MATE II。

MPLAB REAL ICE™在线仿真器（MPLAB REAL ICE™ In-Circuit Emulator）
Microchip 的新一代在线仿真器，与 MPLAB IDE/MPLAB X IDE 配合使用。请参见 ICE/ICD。

MPLAB SIM
Microchip 的软件模拟器，与 MPLAB IDE/MPLAB X IDE 配合使用，支持 PIC MCU 和 dsPIC® DSC 器件。

用于器件的 MPLAB 入门工具包（MPLAB Starter Kit for Device）
Microchip 的入门工具包中包含着手研究指定器件所需的全部内容。查看有效工作的应用程序，然后调试和编程您自己的更改。

MPLIB™目标库管理器（MPLIB™ Object Librarian）
Microchip 的库管理器，与 MPLAB IDE/MPLAB X IDE 配合使用。MPLIB 库管理器是用于将由 MPASM 汇编器（mpasm 或 mpasmwin v2.0）或 MPLAB C18 C 编译器生成的 COFF 目标模块组合成库文件的目标库管理器。

MPLINK™目标链接器（MPLINK™ Object Linker）
MPLINK 链接器是 Microchip MPASM 汇编器和 Microchip C18 C 编译器的目标链接器。也可将 MPLINK 链接器与 Microchip MPLIB 库管理器配合使用。MPLINK 链接器设计为在 MPLAB IDE/MPLAB X IDE 中使用，尽管它也可独立于 MPLAB IDE/MPLAB X IDE 使用。

MRU
最近使用的。指可从 MPLAB IDE/MPLAB X IDE 主下拉菜单中选择的文件和窗口。

本机数据大小（Native Data Size）
对于本机跟踪，在 Watches 窗口中使用的变量大小必须等于选定器件的数据存储器的大小：对于 PIC18 器件为字节，对于 16 位器件为字。

嵌套深度（Nesting Depth）
宏可包含其他宏的最大深度。

节点（Node）
MPLAB IDE/MPLAB X IDE 项目组件。

非扩展模式（Non-Extended Mode）（PIC18 MCU）
在非扩展模式下，编译器不会使用扩展指令和立即数变址寻址。

非实时（Non Real Time）
指处理器执行到断点或单步执行指令，或 MPLAB IDE/MPLAB X IDE 运行在软件模拟器模式。

非易失性存储器（Non-Volatile Storage）
电源关闭时保存其内容的存储器件。

NOP
空操作。执行该指令时，除了程序计数器加 1 外没有任何其他影响。

目标代码/目标文件（Object Code/Object File）
目标代码是由汇编器或编译器生成的机器码。目标文件是包含机器码，也可能包含调试信息的文件。它可以直接执行；或为可重定位的，需要与其他目标文件（如库文件）链接来生成完全可执行的程序。

目标文件伪指令（Object File Directive）
仅当生成目标文件时使用的伪指令。

八进制（Octal）
使用数字 0-7，以 8 为基数的计数体制。最右边的位表示 1 的倍数，右边第二位表示 8 的倍数，右边第三位表示 8^2 = 64 的倍数，以此类推。

片外存储器（Off-Chip Memory）
片外存储器指 PIC18 器件的一种存储器选择，这种情况下存储器可位于目标板上，或所有程序存储器都由仿真器提供。从 Options>Development Mode（选项>开发模式）访问 Memory 选项卡可打开 Off-Chip Memory Selection（片外存储器选择）对话框。

操作码（Opcode）
操作码。请参见*助记符*。

运算符（Operator）
加号“+”和减号“-”之类的符号，它们在构成定义明确的表达式时使用。每个运算符都具有指定的优先级，用于确定求值的顺序。

OTP
可一次编程。非窗口封装的 EPROM 器件。由于 EPROM 需要紫外线照射来擦除其存储内容，因此只有窗口片是可擦除的。

次数计数器（Pass Counter）
每次一个事件（如执行特定地址处的一条指令）发生时都会递减 1 的计数器。当次数计数器的值为零时，事件满足。可将次数计数器分配给断点和跟踪逻辑，以及在 complex trigger（复杂触发）对话框中的任何顺序执行事件。

PC
个人计算机或程序计数器。

PC 主机（PC Host）
运行支持的 Windows 操作系统的任何 PC。

持久性数据（Persistent Data）
永不清除或初始化的数据。其目的是使应用程序可以在器件复位时保存数据。

虚拟字节（Phantom Byte）
dsPIC 架构中的未实现字节，在将 24 位指令字视为 32 位指令字时使用。虚拟字节出现在 dsPIC hex 文件中。

PIC MCU
PIC 单片机（MCU）指 Microchip 的所有 PIC 单片机系列。

PICkit 2 和 3（PICkit 2 and 3）
Microchip 的器件开发编程器，通过 Debug Express 实现调试功能。要了解每个工具支持哪些器件，请参见每个工具的自述文件。

插件（Plug-in）
MPLAB IDE/MPLAB X IDE 具有内置组件和接插模块，用于针对各种软件和硬件工具来配置系统。可在 Tools（工具）菜单下找到几个插件工具。

主机（Pod）
在线仿真器或调试器的机身。其他名称还有 *Puck*（如果外壳是圆的）和 *Probe*（不要与逻辑探头混淆）。

上电复位仿真（Power-on-Reset Emulation）
在应用开始上电时，将随机值写到数据 RAM 区中来模拟 RAM 中的未初始化值的软件随机过程。

Pragma 伪指令（Pragma）
对特定编译器有意义的伪指令。通常一条 pragma 伪指令用于向编译器传达实现定义的信息。

优先顺序（Precedence）
定义表达式中求值顺序的规则。

生产编程器（Production Programmer）
生产编程器是一种编程工具，其中设计了可对器件进行快速编程的资源。它具有在各种电压下进行编程的能力并完全符合编程规范。在生产环境中，应用电路需要在组装线上传送，时间是极其重要的，所以尽可能快地对器件编程至关重要。

配置文件（Profile）
对于 MPLAB SIM 软件模拟器，寄存器已执行激励的汇总列表。

程序计数器（Program Counter）
包含正在执行的指令的地址的存储单元。

程序计数器单元（Program Counter Unit）
16 位汇编器——程序存储器布局的概念化表示。对于每个指令字，程序计数器将递增 2。在可执行段中，2 个程序计数器单元相当于 3 个字节。在只读段中，2 个程序计数器单元相当于 2 个字节。

程序存储器（Program Memory）
*MPLAB IDE/MPLAB X IDE：*器件中存储指令的存储区。亦指仿真器或软件模拟器中包含下载的目标应用固件的存储器。

*16 位汇编器/编译器：*器件中存储指令的存储区。

项目（Project）
项目包含编译应用程序所需的文件（源代码和链接描述文件等），以及它们与各种编译工具和编译选项的关联。

Prologue
编译器生成代码的一部分，负责分配堆栈空间、保存寄存器，以及执行运行时模型中指定的任何其他特定于机器的要求。此代码在给定函数的任何用户代码之前执行。

样机系统（Prototype System）
指用户的目标应用或目标板的术语。

Psect
GCC 段的 OCG 等效形式，program section（程序段）的缩写。被链接器视为一个整体的代码块或数据块。

PWM 信号（PWM Signal）
脉宽调制信号。某些 PIC MCU 包含 PWM 外设。

限定符（Qualifier）
次数计数器使用的地址或地址范围，或用作复杂触发中另一个操作之前的事件。

基数（Radix）
数字基，十六进制或十进制，用于指定一个地址。

RAM
随机访问存储器（数据存储器）。可以以任意顺序访问这种存储器中的信息。

原始数据（Raw Data）
与一个段有关的代码或数据的二进制表示。

只读存储器（Read Only Memory）
存储器硬件，它允许快速访问其中永久存储的数据，但不允许添加或修改数据。

实时（Real Time）
当在线仿真器或调试器从暂停状态释放时，处理器以实时模式运行且与芯片的正常操作相同。在实时模式下，使能仿真器的实时跟踪缓冲区，并持续捕捉所有选择的周期，使能所有断点逻辑。在在线仿真器或调试器模式下，处理器实时执行，直到有效断点导致暂停，或者直到用户暂停执行。

在软件模拟器模式下，实时仅意味着单片机指令的执行速度与主机 CPU 可模拟的指令速度一样快。

递归调用（Recursive Call）
一个直接或间接调用自身的函数。

递归（Recursion）
已定义的函数或宏可调用自己的概念。当编写递归宏时要特别小心；当递归没有出口时容易陷入无限循环。

可重入函数（Re-entrant）
可以有多个同时运行的实例的函数。在下面两种情况下可能发生函数重入：直接或间接递归调用函数；或者在由函数转入的中断处理过程中又执行此函数。

精简（Relaxation）
将某一指令转换为功能相同但大小较小的指令的过程。这对于缩短代码长度非常有用。MPLAB XC16 当前知道如何将 CALL 指令精简为 RCALL 指令。当被调用的符号处于当前指令的+/- 32k 指令字范围内时，将会执行该操作。

可重定位（Relocatable）
一个目标文件，它的地址没有分配到存储器中的固定地址。

可重定位段（Relocatable Section）
16 位汇编器——地址不固定（绝对）的段。链接器通过一个称为重定位的过程来为可重定位段分配地址。

重定位（Relocation）
链接器执行的一个过程，在这个过程中，为可重定位段分配绝对地址，且可重定位段中的所有符号都更新为其新地址。

ROM
只读存储器（程序存储器）。不能修改的存储器。

Run
将仿真器从暂停状态释放，允许仿真器实时运行应用代码、实时改变 I/O 状态或实时响应 I/O 的命令。

运行时模型（Run-time Model）
描述目标架构资源的使用。

运行时观察（Run-time Watch）
Watches 窗口，其中的变量会在应用程序运行时更改。要确定如何设置运行时观察，请参见相应的工具文档。并不是所有工具都支持运行时观察。

场景（Scenario）
MPLAB SIM 软件模拟器用于激励控制的一种特定设置。

段（Section）
OCG psect 的 GCC 等效形式。被链接器视为一个整体的代码块或数据块。

段属性（Section Attribute）
赋予段的 GCC 特性（例如，access 段）。

顺序断点（Sequenced Breakpoint）
按顺序发生的断点。断点的执行顺序为从下到上；序列中的最后一个断点最先执行。

序列号快速批量编程（Serialized Quick Turn Programming）
序列化使您可以将序列号编程到器件编程器进行编程的每个单片机中。该编号可用作记录代码、密码或 ID 数字。

Shell
MPASM 汇编器 shell 是宏汇编器的提示输入接口。有两个 MPASM 汇编器 shell：一个针对 DOS 版本，一个针对 Windows 操作系统版本。

软件模拟器
模仿器件操作的软件程序。

单步执行（Single Step）
这一命令单步执行代码，一次执行一条指令。执行每条指令后，MPLAB IDE/MPLAB X IDE 更新寄存器窗口、观察变量及状态显示，使您可分析和调试指令执行。也可单步执行 C 编译器源代码，但不是每次执行一条指令，MPLAB IDE/MPLAB X IDE 将执行一行高级 C 语句生成的所有汇编指令。

Skew
不同时间出现在处理器总线上与指令执行有关的信息。例如，执行前一条指令的过程中取指时，被执行的操作码出现在总线上；当实际执行该操作码时，源数据地址及其值以及目标数据地址出现在总线上。当执行下一条指令时，目标数据值出现在总线上。跟踪缓冲区一次捕捉总线上的这些信息。因此，跟踪缓冲区的一条记录将包含三条指令的执行信息。执行一条指令时，从一条信息到另一条信息的捕捉周期数称为 skew。

Skid
当使用硬件断点来暂停处理器时，在处理器暂停前可能再执行一条或多条指令。断点后执行的指令条数称为 skid。

源代码（Source Code）
编程人员编写计算机程序的形式。采用某种正式的编程语言编写源代码，可翻译成机器码或被解释程序执行。

源文件（Source File）
包含源代码的 ASCII 文本文件。

特殊功能寄存器（Special Function Register，SFR）
数据存储器（RAM）的一部分，专用于控制 I/O 处理器功能、I/O 状态、定时器或其他模式及外设的寄存器。

SQTP
请参见*序列号快速批量编程*。

堆栈，硬件（Stack, Hardware）
PIC 单片机中调用函数时存储返回地址的存储区。

堆栈，软件（Stack, Software）
供应用程序用来存储返回地址、函数参数和局部变量的存储器。此存储器由程序中的指令在运行时动态分配。它支持可重入函数调用。

堆栈，已编译（Stack, Compiled）
编译器管理和分配的存储区，在其中为变量静态分配空间。当软件堆栈等机制无法在目标器件上高效实现时，将替换为已编译堆栈。它会阻止重入。

静态 RAM 或 SRAM（Static RAM or SRAM）
静态随机访问存储器。目标板上可读/写且不需要经常刷新的程序存储器。

状态栏（Status Bar）
状态栏位于 MPLAB IDE/MPLAB X IDE 窗口的底部，指示诸如光标位置、开发模式、器件和有效工具栏之类的当前信息。

单步进入（Step Into）
这一命令与 Single Step 相同。Step Into（与 Step Over 相对）在 CALL 指令后，单步执行子程序。

单步跳过（Step Over）
Step Over 允许调试代码时不单步执行子程序。当 step over 一条 `CALL` 指令时，下一个断点将设置在 `CALL` 指令后的下一条指令处。如果由于某种原因，子程序陷入无限循环或不正确返回，下一个断点将永远执行不到。除了对 `CALL` 指令的处理外，Step Over 命令与 Single Step 相同。

单步跳出（Step Out）
Step Out 使您可以跳出当前正在单步执行的子程序。该命令会执行该子程序中的剩余代码，然后在该子程序的返回地址处停止执行。

激励（Stimulus）
软件模拟器的输入（即为模拟对外部信号的响应而生成的数据）。通常数据采用文本文件中一系列动作的形式。激励可以是异步的，同步的（引脚），时钟激励和寄存器激励。

跑表（Stopwatch）
测量执行周期的计数器。

存储类别（Storage Class）
确定与指定对象相关联存储区的存在时间。

存储限定符（Storage Qualifier）
指明所声明对象的特殊属性（如 const）。

符号（Symbol）
符号是描述组成程序的各个部分的一种通用机制。这些部分包括函数名、变量名、段名、文件名和结构体/枚举/联合体标记名等。MPLAB IDE/MPLAB X IDE 中的符号主要指变量名、函数名和汇编标号。链接后符号的值就是其在存储器中的值。

符号，绝对（Symbol, Absolute）
表示一个立即数的值，例如通过汇编.equ 伪指令指定的定义。

系统窗口控件（System Window Control）
系统窗口控件位于窗口和某些对话框的左上角。单击该控件时通常会弹出包含“Minimize”（最小化）、“Maximize”（最大化）和“Close”项的菜单。

目标（Target）
指用户硬件。

目标应用程序（Target Application）
目标板上的软件。

目标板（Target Board）
构成目标应用的电路和可编程器件。

目标处理器（Target Processor）
目标应用板上的单片机。

模板（Template）
为以后插入自己的文件中使用而创建的文本行。MPLAB 编辑器将模板存储到模板文件中。

工具栏（Toolbar）
一行或一列图标，单击这些图标时将执行 MPLAB IDE/MPLAB X IDE 功能。

跟踪（Trace）
记录程序执行的仿真器或软件模拟器功能。仿真器将程序执行记录到其跟踪缓冲区内，并可上传到 MPLAB IDE/MPLAB X IDE 的跟踪窗口中。

跟踪存储区（Trace Memory）
跟踪存储区包含在仿真器内部。跟踪存储区有时称为跟踪缓冲区。

跟踪宏（Trace Macro）
一个通过仿真器数据来提供跟踪信息的宏。由于该宏属于软件跟踪，所以必须将它添加到代码中、必须重新编译或重新汇编代码，并且必须使用该代码对目标器件进行编程，之后跟踪才会工作。

触发输出（Trigger Output）
触发输出指可在任意地址或地址范围产生的仿真器输出信号，与跟踪和断点的设置无关。可设置任意个触发输出点。

三字母组合（Trigraph）
三字符序列，都以??开头，由 ISO C 定义用于替代单个字符。

未分配段（Unassigned Section）
在链接器命令文件中未分配到特定目标存储块的段。链接器必须找到用于分配未分配段的目标存储块。

未初始化数据（Uninitialized Data）
定义时未指定初始值的数据。在 C 中，

```
int myVar;
```

定义了将存放到未初始化数据段的一个变量。

上传（Upload）
上传功能将数据从一个工具（如仿真器或编程器）传送到主机计算机，或将数据从目标板传送到仿真器。

USB
通用串行总线。一种外部外设接口标准，用于通过电缆使用双向串行传输在计算机和外部外设之间进行通信。USB 1.0/1.1 支持 12 Mbps 的数据传输速率。USB 2.0（也称为高速 USB）支持最高 480 Mbps 的数据传输速率。

向量（Vector）
复位或中断发生时应用程序跳转到的存储地址。

Volatile
一个变量限定符，用于阻止编译器应用会影响存储器中变量访问方式的优化。

警告（Warning）
*MPLAB IDE/MPLAB X IDE：*提醒出现了可能导致器件、软件文件或设备物理损坏的通知。

*16 位汇编器/编译器：*警告报告可能存在问题的条件，但并不暂停处理。

观察变量（Watch Variable）
调试会话期间可在 Watches 窗口中监控的变量。

Watches 窗口（Watches Window）
Watches 窗口包含一系列观察变量，这些变量在每次执行到断点时更新。

看门狗定时器（Watchdog Timer，WDT）
PIC 单片机中在一段可选择长度的时间后复位处理器的定时器。使用配置位来使能、禁止和设置 WDT。

工作簿（Workbook）
MPLAB SIM 激励器用于生成 SCL 激励的一种设置。

Microchip 网站

Microchip 网站（http://www.microchip.com/）为客户提供在线支持。客户可通过该网站方便地获取文件和信息。我们的网站提供以下内容：

- **产品支持**——数据手册和勘误表、应用笔记和示例程序、设计资源、用户指南以及硬件支持文档、最新的软件版本以及归档软件
- **一般技术支持**——常见问题解答（FAQ）、技术支持请求、在线讨论组以及 Microchip 设计伙伴计划成员名单
- **Microchip 业务**——产品选型和订购指南、最新 Microchip 新闻稿、研讨会和活动安排表、Microchip 销售办事处、代理商以及工厂代表列表

产品变更通知服务

Microchip 的产品变更通知服务有助于客户了解 Microchip 产品的最新信息。注册客户可在他们感兴趣的某个产品系列或开发工具发生变更、更新、发布新版本或勘误表时，收到电子邮件通知。

欲注册，请访问 http://www.microchip.com/pcn，然后按照注册说明进行操作。

客户支持

Microchip 产品的用户可通过以下渠道获得帮助：

- 代理商或代表
- 当地销售办事处
- 应用工程师（ESE）
- 技术支持

客户应联系其代理商、代表或 ESE 寻求支持。当地销售办事处也可为客户提供帮助。本文档后附有销售办事处的联系方式。

也可通过 http://www.microchip.com/support 获得网上技术支持。

Microchip 器件代码保护功能

请注意以下有关 Microchip 器件代码保护功能的要点：

- Microchip 的产品均达到 Microchip 数据手册中所述的技术指标。
- Microchip 确信：在正常使用的情况下，Microchip 系列产品是当今市场上同类产品中最安全的产品之一。
- 目前，仍存在着恶意、甚至是非法破坏代码保护功能的行为。就我们所知，所有这些行为都不是以 Microchip 数据手册中规定的操作规范来使用 Microchip 产品的。这样做的人极可能侵犯了知识产权。
- Microchip 愿意与关心代码完整性的客户合作。
- Microchip 或任何其他半导体厂商均无法保证其代码的安全性。代码保护并不意味着我们保证产品是“牢不可破”的。

代码保护功能处于持续发展中。Microchip 承诺将不断改进产品的代码保护功能。任何试图破坏 Microchip 代码保护功能的行为均可视为违反了《数字器件千年版权法案（Digital Millennium Copyright Act）》。如果这种行为导致他人在未经授权的情况下，能访问您的软件或其他受版权保护的成果，您有权依据该法案提起诉讼，从而制止这种行为。

法律声明

提供本文档的中文版本仅为了便于理解。请勿忽视文档中包含的英文部分，因为其中提供了有关 Microchip 产品性能和使用情况的有用信息。Microchip Technology Inc.及其分公司和相关公司、各级主管与员工及事务代理机构对译文中可能存在的任何差错不承担任何责任。建议参考 Microchip Technology Inc.的英文原版文档。

本出版物中所述的器件应用信息及其他类似内容仅为您提供便利，它们可能由更新之信息所替代。确保应用符合技术规范，是您自身应负的责任。Microchip 对这些信息不作任何明示或暗示、书面或口头、法定或其他形式的声明或担

保，包括但不限于针对其使用情况、质量、性能、适销性或特定用途的适用性的声明或担保。Microchip 对因这些信息及使用这些信息而引起的后果不承担任何责任。如果将 Microchip 器件用于生命维持和/或生命安全应用，一切风险由买方自负。买方同意在由此引发任何一切伤害、索赔、诉讼或费用时，会维护和保障 Microchip 免于承担法律责任，并加以赔偿。除非另外声明，否则在 Microchip 知识产权保护下，不得暗中或以其他方式转让任何许可证。

商标

Microchip 的名称和徽标组合、Microchip 徽标、Adaptec、AnyRate、AVR、AVR 徽标、AVR Freaks、BesTime、BitCloud、chipKIT、chipKIT 徽标、CryptoMemory、CryptoRF、dsPIC、FlashFlex、flexPWR、HELDO、IGLOO、JukeBlox、KeeLoq、Kleer、LANCheck、LinkMD、maXStylus、maXTouch、MediaLB、megaAVR、Microsemi、Microsemi 徽标、MOST、MOST 徽标、MPLAB、OptoLyzer、PackeTime、PIC、picoPower、PICSTART、PIC32 徽标、PolarFire、Prochip Designer、QTouch、SAM-BA、SenGenuity、SpyNIC、SST、SST 徽标、SuperFlash、Symmetricom、SyncServer、Tachyon、TempTrackr、TimeSource、tinyAVR、UNI/O、Vectron 及 XMEGA 均为 Microchip Technology Incorporated 在美国和其他国家或地区的注册商标。

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 徽标、Quiet-Wire、SmartFusion、SyncWorld、Temux、TimeCesium、TimeHub、TimePictra、TimeProvider、Vite、WinPath 和 ZL 均为 Microchip Technology Incorporated 在美国的注册商标。

Adjacent Key Suppression、AKS、Analog-for-the-Digital Age、Any Capacitor、AnyIn、AnyOut、BlueSky、BodyCom、CodeGuard、CryptoAuthentication、CryptoAutomotive、CryptoCompanion、CryptoController、dsPICDEM、dsPICDEM.net、Dynamic Average Matching、DAM、ECAN、EtherGREEN、In-Circuit Serial Programming、ICSP、INICnet、Inter-Chip Connectivity、JitterBlocker、KleerNet、KleerNet 徽标、memBrain、Mindi、MiWi、MPASM、MPF、MPLAB Certified 徽标、MPLIB、MPLINK、MultiTRAK、NetDetach、Omniscient Code Generation、PICDEM、PICDEM.net、PICkit、PICtail、PowerSmart、PureSilicon、QMatrix、REAL ICE、Ripple Blocker、SAM-ICE、Serial Quad I/O、SMART-I.S.、SQI、SuperSwitcher、SuperSwitcher II、Total Endurance、TSHARC、USBCheck、VariSense、ViewSpan、WiperLock、Wireless DNA 和 ZENA 均为 Microchip Technology Incorporated 在美国和其他国家或地区的商标。

SQTP 为 Microchip Technology Incorporated 在美国的服务标记。

Adaptec 徽标、Frequency on Demand、Silicon Storage Technology 和 Symmcom 均为 Microchip Technology Inc.在除美国外的国家或地区的注册商标。

GestIC 为 Microchip Technology Inc.的子公司 Microchip Technology Germany II GmbH & Co. KG 在除美国外的国家或地区的注册商标。

在此提及的所有其他商标均为各持有公司所有。

© 2020，Microchip Technology Incorporated 版权所有。

ISBN：978-1-5224-5461-8

质量管理体系

有关 Microchip 的质量管理体系的信息，请访问 http://www.microchip.com/quality。

全球销售及服务网点

美洲	亚太地区	亚太地区	欧洲
公司总部 2355 West Chandler Blvd. Chandler, AZ 85224-6199 电话：480-792-7200 传真：480-792-7277 技术支持： http://www.microchip.com/support 网址： http://www.microchip.com **亚特兰大** 德卢斯，佐治亚州 电话：678-957-9614 传真：678-957-1455 **奥斯汀，德克萨斯州** 电话：512-257-3370 **波士顿** 韦斯特伯鲁，马萨诸塞州 电话：774-760-0087 传真：774-760-0088 **芝加哥** 艾塔斯卡，伊利诺伊州 电话：630-285-0071 传真：630-285-0075 **达拉斯** 阿迪森，德克萨斯州 电话：972-818-7423 传真：972-818-2924 **底特律** 诺维，密歇根州 电话：248-848-4000 **休斯顿，德克萨斯州** 电话：281-894-5983 **印第安纳波利斯** 诺布尔斯维尔，印第安纳州 电话：317-773-8323 传真：317-773-5453 电话：317-536-2380 **洛杉矶** 米慎维荷，加利福尼亚州 电话：949-462-9523 传真：949-462-9608 电话：951-273-7800 **罗利，北卡罗来纳州** 电话：919-844-7510 **纽约，纽约州** 电话：631-435-6000 **圣何塞，加利福尼亚州** 电话：408-735-9110 电话：408-436-4270 **加拿大 - 多伦多** 电话：905-695-1980 传真：905-695-2078	**澳大利亚 - 悉尼** 电话：61-2-9868-6733 **中国 - 北京** 电话：86-10-8569-7000 **中国 - 成都** 电话：86-28-8665-5511 **中国 - 重庆** 电话：86-23-8980-9588 **中国 - 东莞** 电话：86-769-8702-9880 **中国 - 广州** 电话：86-20-8755-8029 **中国 - 杭州** 电话：86-571-8792-8115 **中国 - 香港特别行政区** 电话：852-2943-5100 **中国 - 南京** 电话：86-25-8473-2460 **中国 - 青岛** 电话：86-532-8502-7355 **中国 - 上海** 电话：86-21-3326-8000 **中国 - 沈阳** 电话：86-24-2334-2829 **中国 - 深圳** 电话：86-755-8864-2200 **中国 - 苏州** 电话：86-186-6233-1526 **中国 - 武汉** 电话：86-27-5980-5300 **中国 - 西安** 电话：86-29-8833-7252 **中国 - 厦门** 电话：86-592-2388138 **中国 - 珠海** 电话：86-756-3210040	**印度 - 班加罗尔** 电话：91-80-3090-4444 **印度 - 新德里** 电话：91-11-4160-8631 **印度 - 浦那** 电话：91-20-4121-0141 **日本 - 大阪** 电话：81-6-6152-7160 **日本 - 东京** 电话：81-3-6880-3770 **韩国 - 大邱** 电话：82-53-744-4301 **韩国 - 首尔** 电话：82-2-554-7200 **马来西亚 - 吉隆坡** 电话：60-3-7651-7906 **马来西亚 - 槟榔屿** 电话：60-4-227-8870 **菲律宾 - 马尼拉** 电话：63-2-634-9065 **新加坡** 电话：65-6334-8870 **台湾地区 - 新竹** 电话：886-3-577-8366 **台湾地区 - 高雄** 电话：886-7-213-7830 **台湾地区 - 台北** 电话：886-2-2508-8600 **泰国 - 曼谷** 电话：66-2-694-1351 **越南 - 胡志明市** 电话：84-28-5448-2100	**奥地利 - 韦尔斯** 电话：43-7242-2244-39 传真：43-7242-2244-393 **丹麦 - 哥本哈根** 电话：45-4485-5910 传真：45-4485-2829 **芬兰 - 埃斯波** 电话：358-9-4520-820 **法国 - 巴黎** 电话：33-1-69-53-63-20 传真：33-1-69-30-90-79 **德国 - 加兴** 电话：49-8931-9700 **德国 - 哈恩** 电话：49-2129-3766400 **德国 - 海尔布隆** 电话：49-7131-72400 **德国 - 卡尔斯鲁厄** 电话：49-721-625370 **德国 - 慕尼黑** 电话：49-89-627-144-0 传真：49-89-627-144-44 **德国 - 罗森海姆** 电话：49-8031-354-560 **以色列 - 若那那市** 电话：972-9-744-7705 **意大利 - 米兰** 电话：39-0331-742611 传真：39-0331-466781 **意大利 - 帕多瓦** 电话：39-049-7625286 **荷兰 - 德卢内市** 电话：31-416-690399 传真：31-416-690340 **挪威 - 特隆赫姆** 电话：47-72884388 **波兰 - 华沙** 电话：48-22-3325737 **罗马尼亚 - 布加勒斯特** 电话：40-21-407-87-50 **西班牙 - 马德里** 电话：34-91-708-08-90 传真：34-91-708-08-91 **瑞典 - 哥德堡** 电话：46-31-704-60-40 **瑞典 - 斯德哥尔摩** 电话：46-8-5090-4654 **英国 - 沃金厄姆** 电话：44-118-921-5800 传真：44-118-921-5820