MPLAB® XC8 入门指南

对于刚刚开始使用 MPLAB® XC8 C 编译器的编程人员，尤其是不熟悉嵌入式编程或 Microchip 器件的编程人员，本文档提供了一个起点。

以下标题链接到本指南中的各节：

虽然 MPLAB XC8 C 编译器的目标器件包括数百种 8 位 PIC® 器件，但本指南使用的是 PIC18F87J11 单片机 （MCU）与 PICDEM™ PIC18 Explorer 开发板。但是，本文档中介绍的信息可与 XC8 C 编译器配合使用，为几乎任何 8 位 MCU 和硬件创建和编译等效代码。

本指南介绍的是在 MPLAB X 集成开发环境 （Integrated Development Environment，IDE）中使用编译器；但您也可以从命令行中使用它。如果有开发板，可以将代码下载到您所用器件上并在器件上运行代码。此外，您还可以在 MPLAB X IDE 中使用软件模拟器来确认您的代码的操作。

为了说明如何开始使用 MPLAB XC8 C 编译器，下文将指导您完成创建一个可以编译并运行的项目的过程。该项目的功能是使一个与端口引脚连接的 LED 发生闪烁。要实现该功能，需要执行以下概括列出的操作。在您阅览本指南的页面时，将会展开并更详细地介绍这些操作。

- 在源文件中包含 <xc.h>。
- 使用 `config` pragma 伪指令设置器件配置位。
- 禁止任何使用端口所用引脚的外设。
- 初始化端口的数据方向寄存器，并将值写入端口锁存器。
- 使用延时来确保可以看到状态变化。

本指南假定您在开始之前已安装并激活（如适用）MPLAB X IDE 和 MPLAB XC8 C 编译器。此外，您也可以使用编译器的评估版，或以免费模式运行的编译器。

关于安装或激活编译器的帮助，请参见 *Installing and Licensing MPLAB XC C Compilers*（DS50002059）。该文档可以从 Microchip Technology 网站 www.microchip.com 下载。

在 MPLAB X IDE 中创建项目

本节介绍如何在 MPLAB X IDE 中使用 MPLAB XC8 C 编译器创建项目。

以下步骤说明了其过程：

步骤 1 设置项目类型。

步骤 2 选择目标器件。

步骤 3 选择器件连接器。

步骤 4 选择运行项目代码的工具。

步骤 5 仅适用于一些调试器工具选择。

步骤 6 选择编译源代码的工具。

步骤 7 指定项目名称和路径。

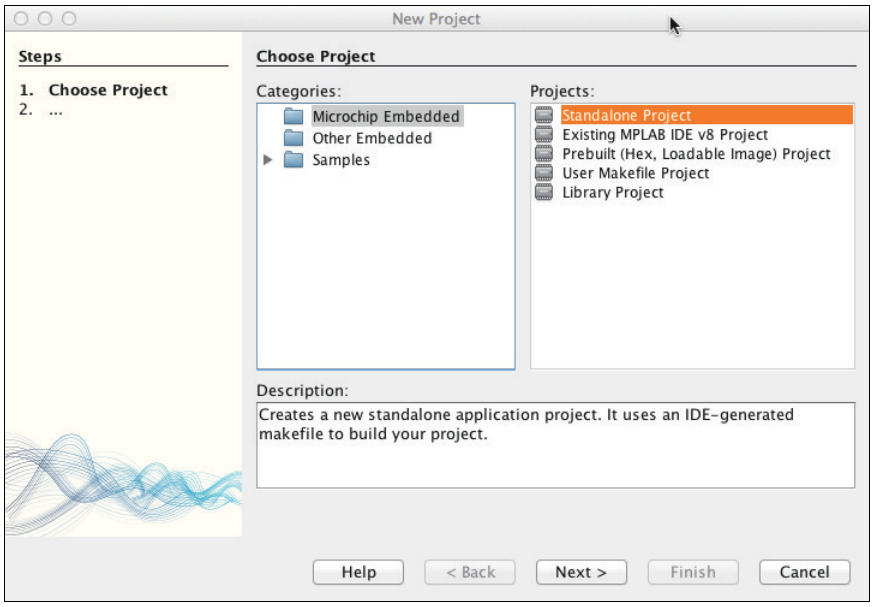
步骤 8 完成项目的创建。

如果使用的不是 MPLAB X IDE，或者您已经熟悉创建项目的过程，请跳至下一节 **“ 基础代码 ”**。关于 MPLAB X IDE 的完整信息，可在线阅读 《MPLAB® X IDE 用户指南》（DS50002027C_CN）。

步骤 1 设置项目类型。

在 MPLAB X IDE 中，选择 *File（文件）>New Project...（新建项目 ...）*。在打开的窗口中（如图 1-1 所示），选择 “Microchip Embedded”（Microchip 嵌入式）类别，并从 Projects （项目）域中选择 “Standalone Project” （独立项目）。

图 1-1： NEW PROJECT 窗口

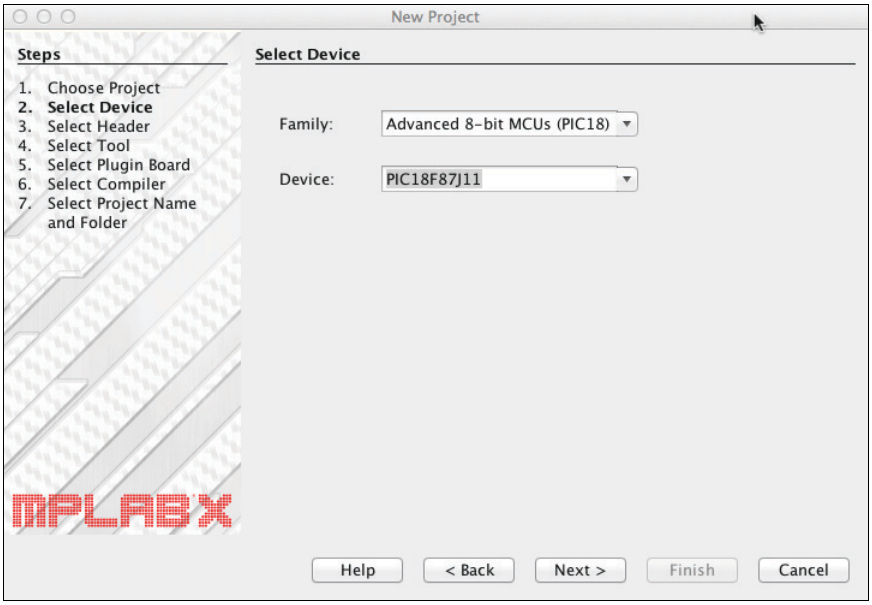


步骤 2 选择目标器件。

该选择必须与您所用硬件上的器件完全匹配。（如果在没有硬件的情况下使用软件模拟器，则可以选择任意器件。）

为了让选择器件变得更简单，器件按系列进行组织。MPLAB XC8 可以针对 8 位单片机系列中的任何器件进行编译。在图 1-2 中，已经从 PIC18 系列中选择了 PIC18F87J11。

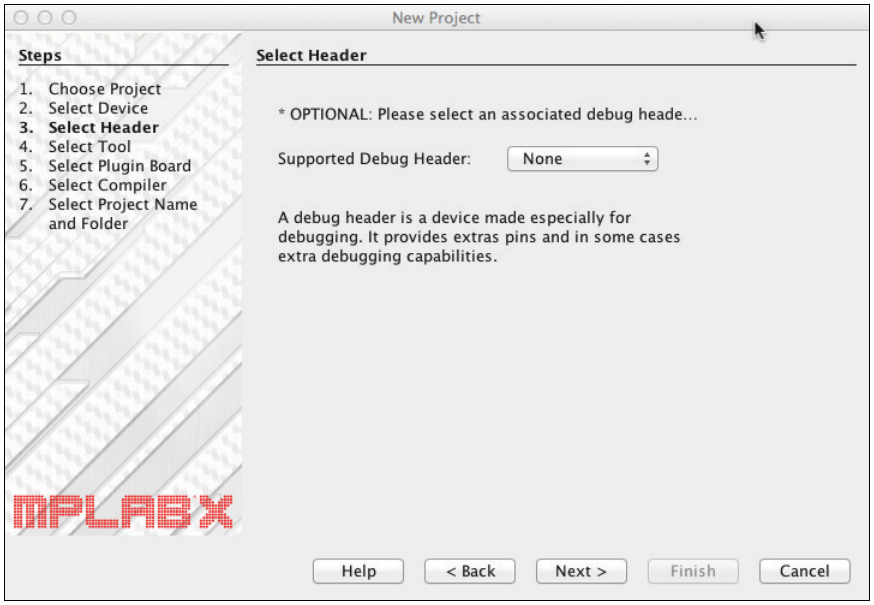
图 1-2：　　器件选择对话框



步骤 3 选择器件连接器。

本指南中不需要使用调试功能，所以该选择可以为 None （无），如图 1-3 所示。

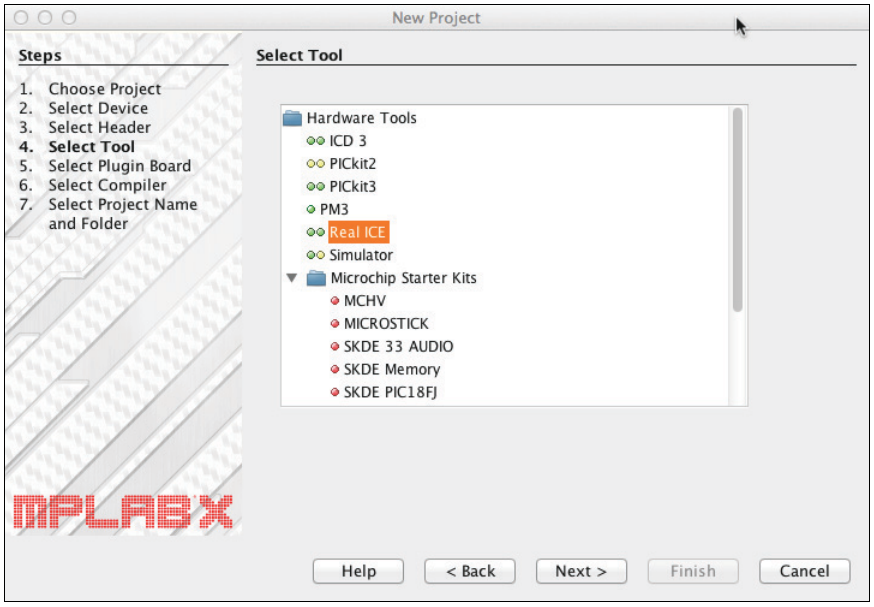
图 1-3： 连接器选择对话框



步骤 4 选择运行项目代码的工具。

如果您具有调试器，并希望对硬件使用它，则从列表中选择该调试器；否则，请选择 Simulator（软件模拟器）。图 1-4 显示了选择 MPLAB REAL ICE™ 作为编程器 / 调试器来运行所生成的代码。

图 1-4： 工具选择对话框



步骤 5 仅适用于一些调试器工具选择。

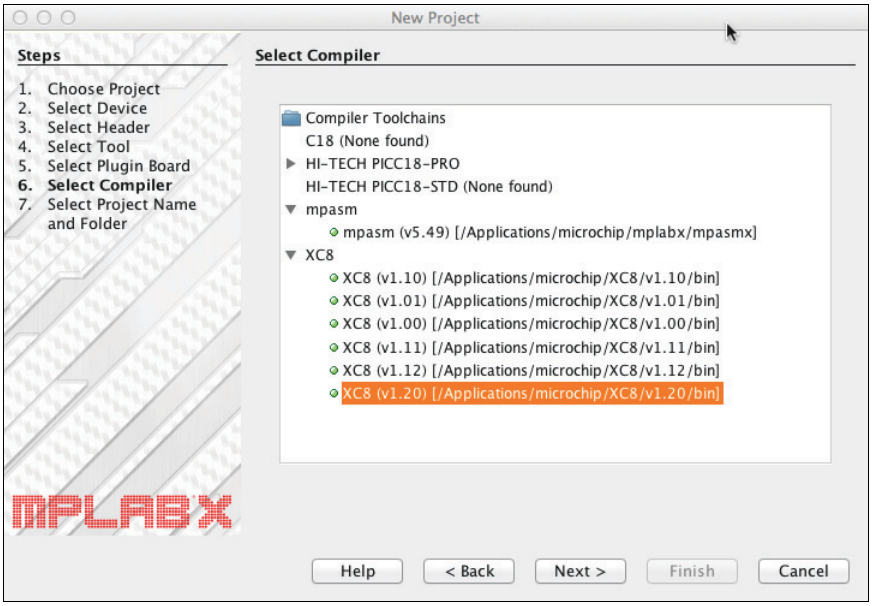
除非您必须使用特定的接插板，否则请选择 None （如果出现图 1-5 所示的对话框）。

图 1-5： 接插板选择对话框

步骤 6 选择编译源代码的工具。

如图 1-6 中的 Select Compiler（选择编译器）窗口所示，XC8 展示小部件下可能会列出 MPLAB XC8 编译器的几个版本。请选择最新的版本。您可以在开发过程中更改该选择。

图 1-6： 编译器选择对话框

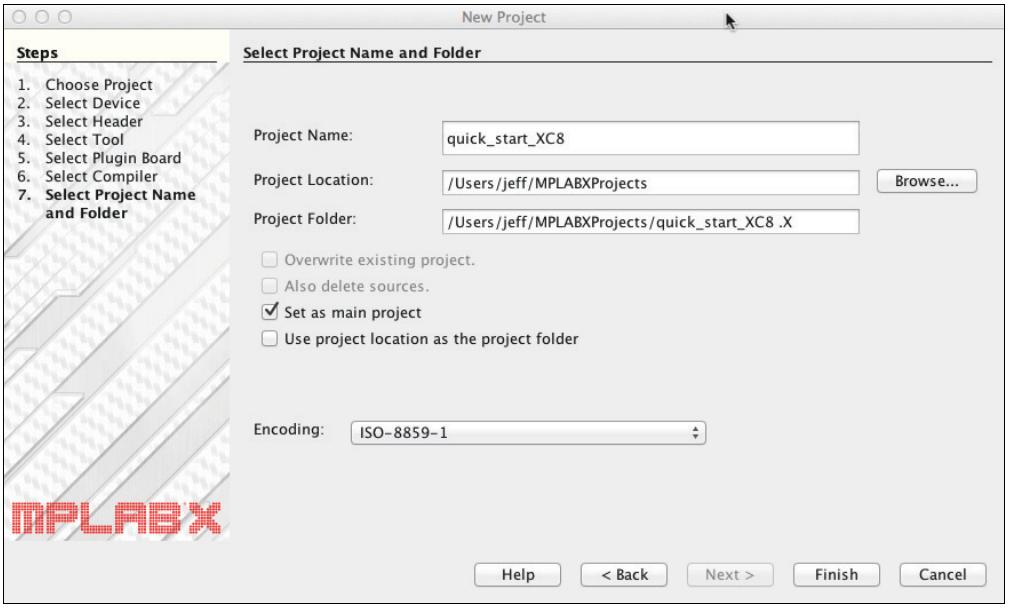


步骤 7 指定项目名称和路径。

在 Project Name （项目名称）域中输入项目的名称。如果默认项目路径不合适，则单击 **Browse...**（浏览 ...）。在此例中，为了说明目的已经选择了名称 `quick_start_XC8`，如图 1-7 所示。

要在 IDE 中将当前项目区分为主项目 （存在多个项目时），请单击 “Set as main project” （设置为主项目）。

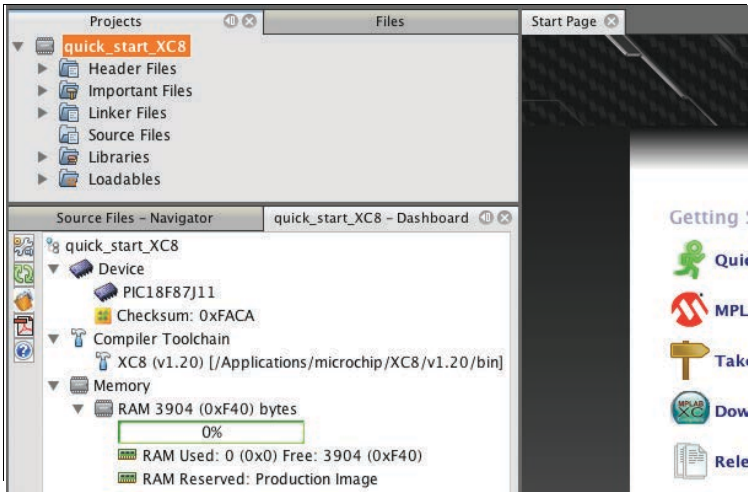
图 1-7： 项目名称和路径对话框



步骤 8 完成项目的创建。

单击 **Finish**（完成），项目就会被创建。 Projects 窗口[1] 中会出现一个代表该项目的图标，如图1-8所示。Projects窗口显示在图中的左上角。在Projects窗口下面，Dashboard（仪表板）会提供更详细的项目信息。

图 1-8： PROJECTS 窗口



1. 如果该窗格在默认情况下不可见，则可能需要选择 *Windows （窗口）>Projects*。

基础代码

此处介绍的代码实际上是一个小程序，可以用作您的所有 MPLAB XC8 项目的基础。虽然该代码解读起来很平凡，但它是完全有效的，并可根据需要编译并执行。

以下步骤 （链接到后面的步骤说明）说明了代码创建过程：

步骤 1 创建一个新的源文件。

步骤 2 为源文件输入适合的名称。

步骤 3 向新文件中添加骨架代码。

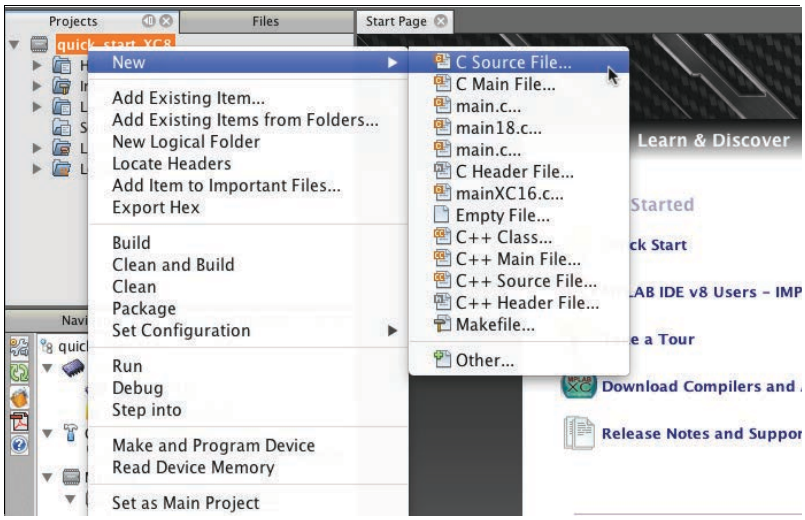
步骤 4 保存您的工作。

步骤 1 创建一个新的源文件。

使用 MPLAB X IDE，可以通过几种方法来创建源文件。以下方法是最基本的，并且它会历经源代码创建过程的所有方面。

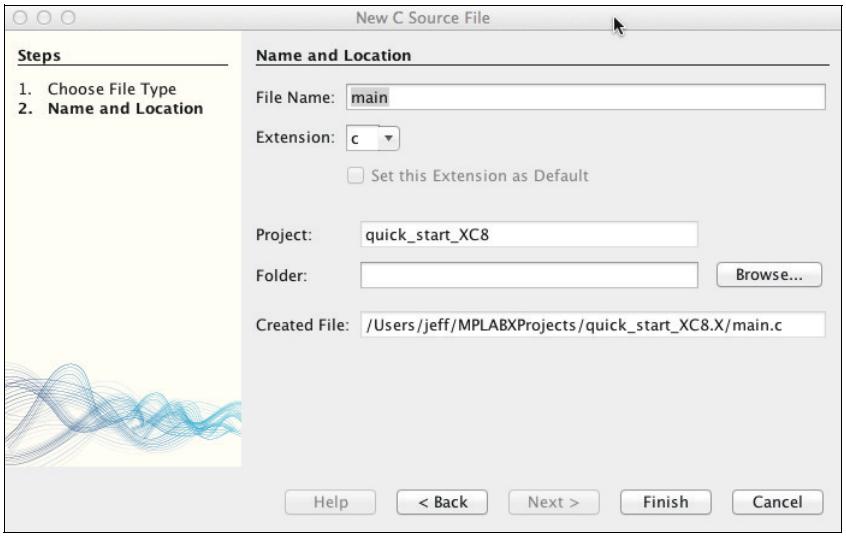
如图 1-9 中的橙色部分所示，右键单击代表您所创建新项目 `quick_start_XC8` 的项目图标。在弹出命令和目标列表中选择 *New （新建）>C Source File... （C 源文件...）*。

图 1-9： 新建文件弹出菜单



这会打开 New C Source File （新建 C 源文件）窗口，如图 1-10 所示。

图 1-10： NEW C SOURCE FILE 窗口



步骤 2 为源文件输入适合的名称。

确保项目名称正确。

如图 1-10 所示，这些设置将创建一个名为 `main.c` 的文件。单击 **Finish** 之后，Projects（项目）窗口中会出现一个代表该文件的图标。此外，还会在文本编辑器中打开该文件。此时，该文件是空的。

步骤 3 向新文件中添加骨架代码。

将以下文本复制或输入到新的源文件 `main.c` 中。

```
#include <xc.h>

int main(void)
{
    return 0;
}
```

该初始代码可用作使用 MPLAB XC8.C 创建的每个项目的起点。

每个 C 程序都必须具有一个且只有一个名为 `main()` 的函数；但是，该函数的确切原型会因编译器而异。对于所有 MPLAB XC 编译器，可以使用上面所示的原型。由于 `main()` 返回一个 `int`，所以必须具有指定了返回值的 `return` 语句。值 `0` 表示 `main()` 成功返回。

包含头文件 `<xc.h>` 将使该源文件中的代码可以访问特定于编译器或特定于器件的功能。由于这种访问非常普遍，所以您几乎需要在所有源文件中包含它。

步骤 4 保存您的工作。

通过选择 *File>Save（保存）*来确保保存您的工作。

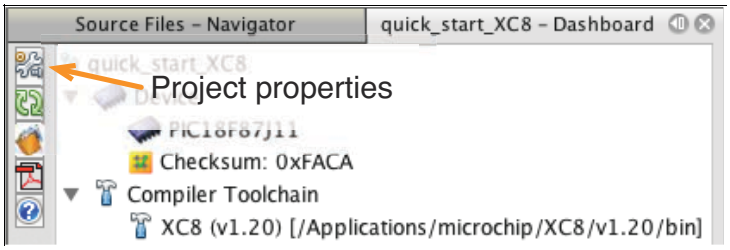
如果不使用 MPLAB X IDE，您可以使用任意编辑器在一个文件中输入以上程序，前提是将它保存为纯文本，并且文件使用 `.c` 扩展名。

编译

正如前面提到的，新的程序是一个有效的 C 程序。这意味着它可以进行编译。本节将说明如何编译代码。

在编译您的源代码时， MPLAB X IDE 知道执行哪个编译器，但可以使用选项来改变编译器的工作方式。对于大多数项目，默认选项都是可以接受的。如果您确实需要调整编译器选项，可以通过 Project Properties （项目属性）对话框来进行。使用项目仪表板中左侧最上端的按钮打开该对话框，如图 1-11 所示。在该对话框中，您还可以更改其他项目属性，例如与项目关联的器件或编译器。

图 1-11： PROJECT PROPERTIES 按钮



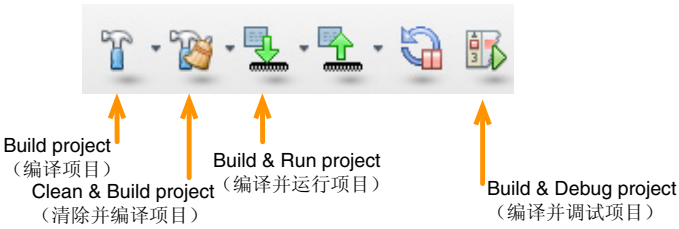
有几种方法可以执行编译器。工具栏上有一些按钮可用于快速访问不同的编译操作，但您也可以通过 ***Run*** *（运行）*和 ***Debug*** *（调试）*菜单访问它们。一些操作只会编译代码；其他一些操作则会编译并*执行*代码。编译和运行步骤都可以设为在发布或调试模式下进行。

调试模式操作会在器件上使能调试执行程序。这将允许访问断点等调试功能。为了让调试执行程序可以工作，它必须使用一些正常情况下可供您的代码使用的器件存储器。调试编译会确保为调试执行程序保留该存储器。

发布模式操作不允许使用任何调试功能，但器件的所有存储器都可供您的项目使用。您将使用这种编译模式来生成适合您要发布的产品的生产映像。

图 1-12 显示了用于编译代码的最常用工具栏按钮。

图 1-12：　　编译按钮



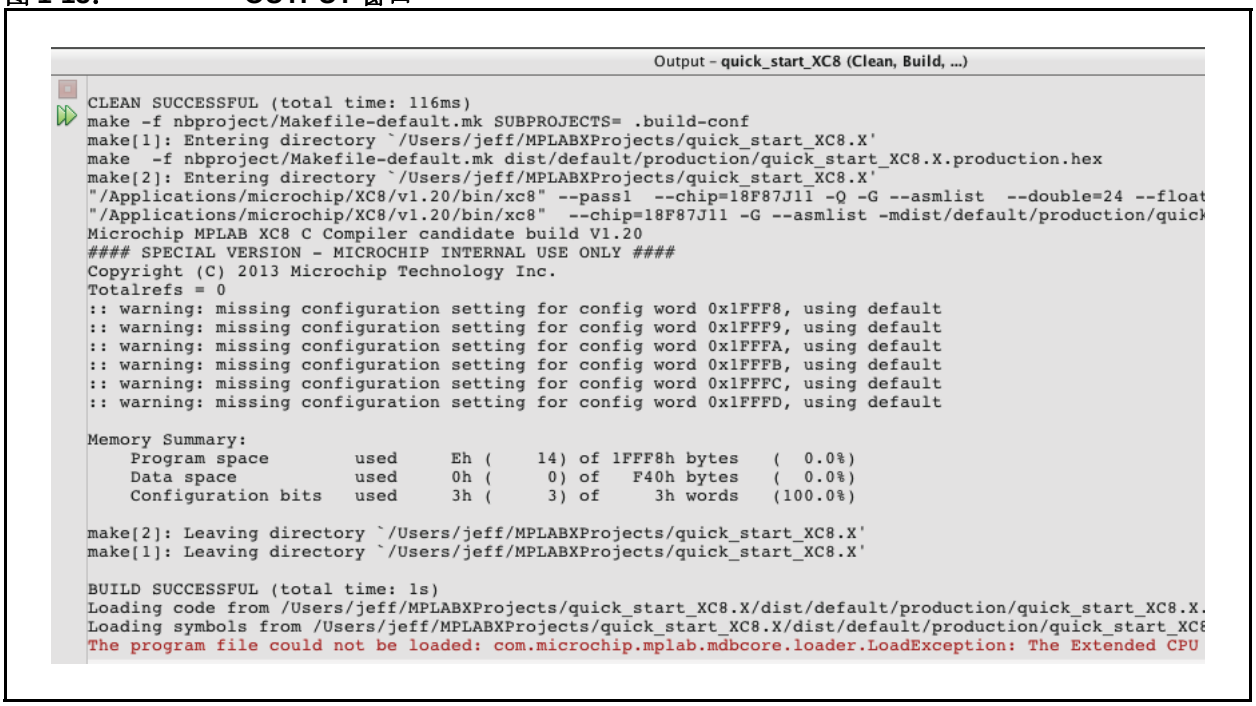
从左到右，所示的按钮会执行以下功能：

- 编译 （发布）自上次编译以来发生修改的任何项目源文件，然后进行链接
- 编译 （发布）*所有*项目源文件，然后进行链接
- 编译 （发布）自上次编译以来发生修改的任何项目源文件，进行链接，然后下载并运行代码
- 编译 （调试）自上次编译以来发生修改的任何项目源文件，进行链接，然后下载并在使能调试执行程序的情况下运行代码

对于本演示，请单击 Build （编译）或 Clean and Build （清除并编译）。

此时会调用与您的项目关联的编译器，并编译您的项目中的每个源文件（当前只有一个），然后链接到一个二进制映像。您可以在 Output（输出）窗口中看到编译过程的记录（如果该窗口在工作区中不可见，则会打开该窗口）。其内容类似于图 1-13 所示。

图 1-13： OUTPUT 窗口



请注意，其中存在几条有关缺少配置设置的警告[1]，但它们并没有使编译过程停止，窗口下半部分的 BUILD SUCCESSFUL（编译成功）消息指示代码已成功编译。 Output 窗口底部的红色错误指示所编译的代码与目标器件不匹配。下一节将说明如何配置器件，这可以解决这些警告和错误。

如果从终端进行编译，请使用以下命令行。

```
xc8 --chip=18f87j11 main.c
```

根据情况调整器件和源文件名称。如果编译器不处于搜索路径中，则应使用包含应用程序名称 `xc8` 的完整路径。

1. 如果针对非 PIC18F87J11 器件进行编译，看到的警告可能会更多或更少。

指定器件配置位

虽然这个新程序是有效的 C 程序，但它不太可能会在器件上正确运行。为了确保正确工作，所有 Microchip 8 位 PIC 器件都必须进行配置。一些配置设置会影响器件的基本操作，例如指令时钟的配置设置。如果该设置不正确，时钟可能不会运行。

上一节的 Output 窗口中显示的警告可以指示器件配置的潜在问题，但即使未看到编译器发出的任何警告，您也必须指定这些配置设置，这一点很重要。

配置设置通过器件中的一些特殊位指定。MPLAB XC8 C 编译器使用了一些 pragma 伪指令，使您可以在代码中指定配置位设置。基于这些 pragma 伪指令得到的值会被合并到您的项目的编译二进制映像中，并下载到器件中。

配置设置的数量和类型会因器件而异。要了解每个设置控制的方面，请参见您所用MCU的数据手册。在此例中，器件为 PIC18F87J11，其数据手册为 《PIC18F87J11 系列数据手册》（DS39778D_CN），可从 www.microchip.com 下载。

完成配置您的器件所需的 pragma 伪指令的最简单方法是使用 Configuration Bits（配置位）窗口，它是 MPLAB X IDE 的一项功能。以下步骤介绍了如何使用该窗口来获取完成 pragma 伪指令所需的信息。

步骤 1 打开 Configuration Bits 窗口。

步骤 2 查看每个设置。

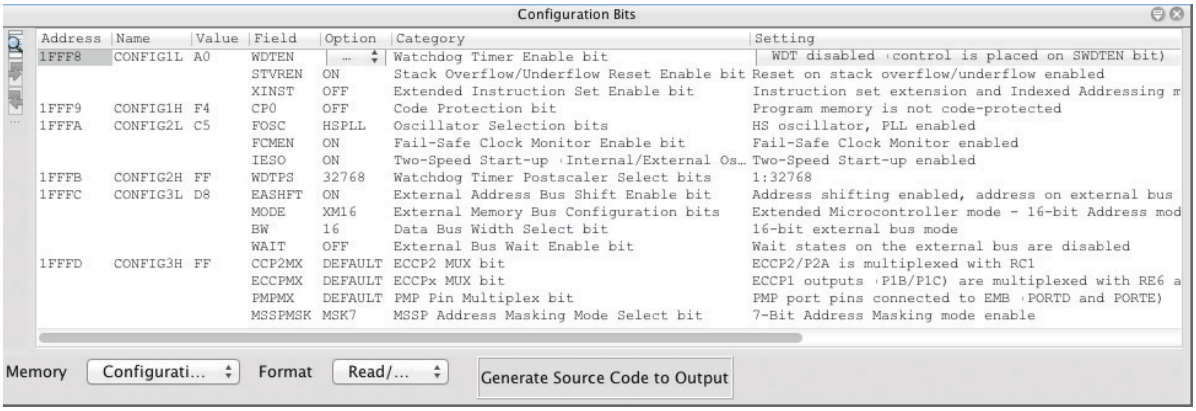
步骤 3 生成可以实现您所选设置的 pragma 伪指令。

步骤 4 将代码从该窗口中复制到您选择的源文件中。

步骤 1 打开 Configuration Bits 窗口。

从菜单中选择 *Window >PIC Memory Views （PIC 存储器视图）>Configuration Bits*。该窗口类似于图 1-14 所示的窗口，它会列出与配置位的位置和值相关的信息。

图 1-14： CONFIGURATION BITS 窗口



Name （名称）和 Field （字段）列可以帮助您查找器件数据手册中的等效设置。

Category （类别）列描述设置控制的方面。

Setting （设置）列显示该设置的当前状态。

步骤 2 查看每个设置。

特别注意以下设置，如果指定不正确，这些设置几乎肯定会导致运行时失败：

振荡器部分

它必须与您所用硬件的振荡器电路匹配。如果该设置不正确，器件时钟可能不会运行。如果您使用软件模拟器作为调试工具，则可以忽略该设置。通常情况下，开发板使用高速晶体振荡器。

看门狗定时器

建议您禁止该定时器，直到需要它时才使能。这可以防止意外复位。

代码保护

关闭代码保护，直到需要它时才开启。这可以确保器件是完全可访问的。

扩展指令集

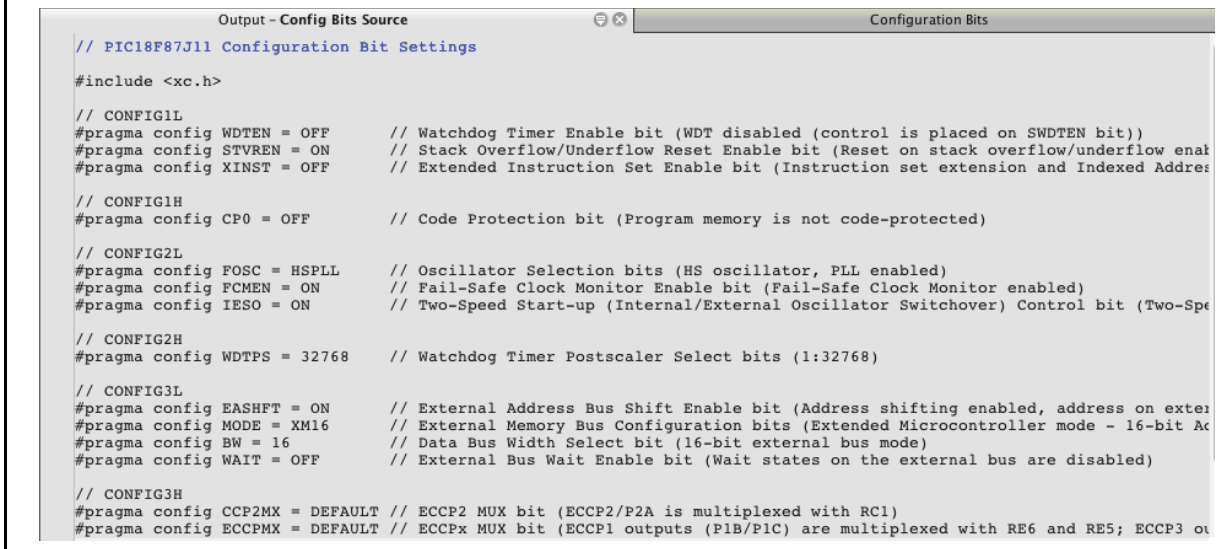
*必须*禁止该 PIC18 设置。 MPLAB XC8 C 编译器不支持该指令集。

通过单击 Setting 列中的相关行，然后从下拉列表中选择相应设置来更改设置。

步骤 3 生成可以实现您所选设置的 pragma 伪指令。

单击 **Generate Source Code to Output**（生成要输出的源代码）按钮。生成的代码将显示在 Config Bits Source （配置位源代码）窗口中，如图 1-15 所示。

图 1-15： CONFIG BITS SOURCE 窗口



步骤 4 将代码从该窗口中复制到您选择的源文件中。

它不是可执行代码，应放置在函数定义之外。代码已被复制到 `main.c` 文件 （为清楚起见，省略了注释），如下所示[1]。

```
#include <xc.h>

// CONFIG1
#pragma config WDTEN = OFF
#pragma config STVREN = ON
#pragma config XINST = OFF
#pragma config CP0 = OFF

// CONFIG2
#pragma config FOSC = HSPLL
#pragma config FCMEN = ON
#pragma config IESO = ON
#pragma config WDTPS = 32768

// CONFIG3
#pragma config EASHFT = ON
#pragma config MODE = XM16
#pragma config BW = 16
#pragma config WAIT = OFF
#pragma config CCP2MX = DEFAULT
#pragma config ECCPMX = DEFAULT
#pragma config PMPMX = DEFAULT
#pragma config MSSPMSK = MSK7

int main(void)
{
    return 0;
}
```

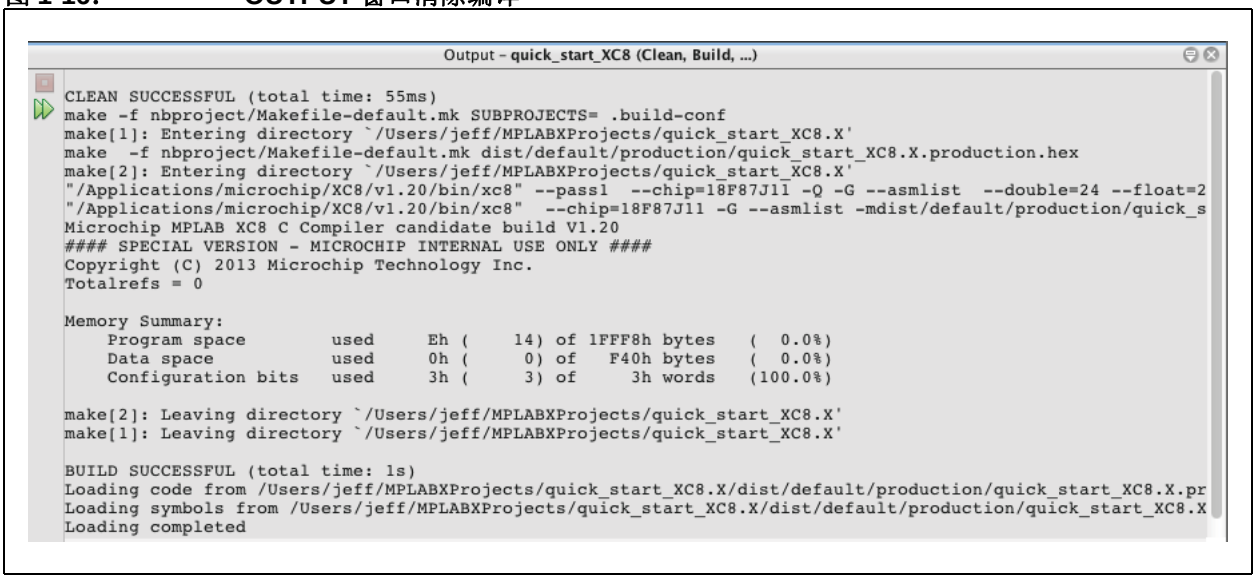
1. 该代码是特定于 PIC18F87J11 器件和 PICDEM PIC18 Explorer 开发板的。您必须使用特定于您所用器件和硬件的配置设置。

请注意，源代码中的配置 pragma 伪指令和 Configuration Bits 窗口之间不存在链接关系。如果需要调整配置设置，您必须在源代码中手动编辑 pragma 伪指令。或者，在 Configuration Bits 窗口中更改配置，重新生成源代码，然后用新生成的 pragma 伪指令替换现有的 pragma 伪指令。

在 MPLAB X IDE 之外还有一个位置，您可以找到与支持的 Microchip 器件相关的设置和值。编译器的下载文件中包含了一个 HTML 指南。在您的编译器安装路径的 DOCS 目录中，打开文件 `pic_chipinfo.html` 或 `pic18_chipinfo.html`。单击您所使用目标器件的链接，页面中会显示对应于 `config` pragma 伪指令的设置和值。

将配置 pragma 伪指令包含在源代码中，现在将能够像上一节一样进行编译，但不会产生任何警告或错误。图 1-16 显示了成功的编译操作。

图 1-16： OUTPUT 窗口清除编译



访问器件寄存器

上一节中编译的代码仍然没有任何运行时功能。现在可以设置器件来执行一项任务。本节说明如何访问器件的特殊功能寄存器 （SFR），并点亮一个与端口连接的 LED。

以下代码会设置端口 D 的数据方向，然后向该端口的锁存器写入一个值[1]。

```
#include <xc.h>

// your configuration bit settings go here
// configuration code (indicated earlier) omitted for brevity

int main(void)
{
    // code to access your port replaces the following
    TRISD = 0x0;    // set all port D bits to be output
    LATD = 0x55;    // write a value to the port D latch

    return 0;
}
```

在该代码中，使用特殊的标识符来表示 SFR。这些标识符与通过包含 <xc.h> 而定义的变量相关联。它们可以像任何其他 C 变量一样使用，但它们各自会被赋予一个地址，这些地址会将它们映射到它们代表的寄存器。请注意，写入这些变量就会写入寄存器，因而可能更改器件的状态。仅仅读取这些变量的操作有时可能会影响器件。

上面提及的标识符与它们所代表的寄存器的名称（由器件数据手册指定）相同。但是，实际情况可能并不总是如此，特别是对于代表 SFR 内的位的标识符。

要了解在访问器件上的 SFR 时要使用哪些名称，请按照以下步骤来确定用于任何器件的寄存器名称。

步骤 1 创建一个包含 <xc.h> 的源文件。

步骤 2 验证器件是否合适，编译代码，并检查错误。

步骤 3 查看编译器在步骤 2 中产生的预处理文件。

1. 一些器件没有与端口关联的锁存寄存器，您需要直接写入端口，例如 PORTD 寄存器。具体请查询器件数据手册。

步骤 1 创建一个包含 <xc.h> 的源文件。

该文件的内容无关紧要。本指南中开始时使用的骨架代码（即空 `main()` 函数）是理想的选择。您可以使用上一节中的文件。

步骤 2 验证器件是否合适，编译代码，并检查错误。

步骤 3 查看编译器在步骤 2 中产生的预处理文件。

编译后通常会留下该文件。如果源文件名为 `main.c`，则预处理文件名为 `main.pre`。如果在命令行上进行编译，则文件会留在与源文件相同的目录中。如果使用 MPLAB X IDE，则单击 Files（文件）窗口，如图 1-17 左上角所示。在该窗口中的项目文件夹下查看，位于 `build/default/production` 文件夹中[1]。

图 1-17：　　显示了寄存器名称的预处理文件

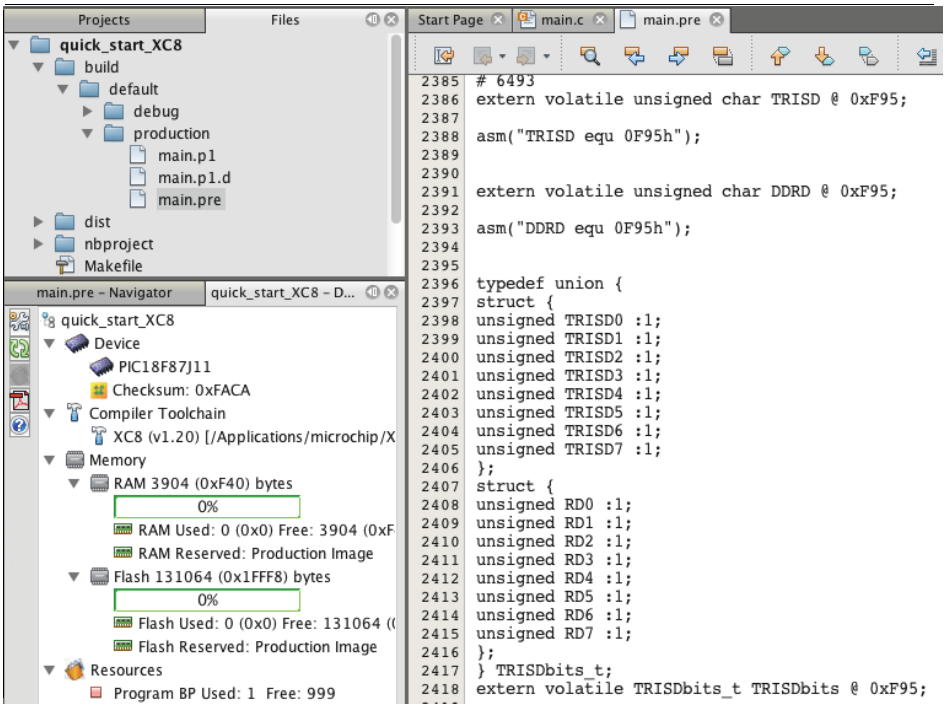


图 1-17 还显示了 PIC18F87J11 的该预处理文件的内容。它显示在右侧，在编辑器中打开。它包含了代表 SFR 的所有变量的 C 定义。

在图 1-17 中，可以看到变量 TRISD 被定义为 `unsigned char` 类型，被放置在地址 0xF95 处。如果查看 PIC18F87J11 的数据手册，将可以确认这确实是 TRISD 寄存器的地址。此外，可以注意到该变量具有一个别名 DDRD。您还可以看到在该寄存器内定义的位，举例来说，您可以使用结构位域 `TRISDbits.TRISD7` 或其别名 `TRISDbits.RD7` 来访问端口方向寄存器的 MSb。如果使用其他方式找不到寄存器，可以使用寄存器的地址在文件中查找相应的名称。该预处理文件是一个中间文件，对它所做的任何更改都会在下次编译时丢失，这一点很重要。

1. `production` 目录用来存放发布（生产）编译的中间文件； `debug` 目录存放调试编译的相同文件，请参见 **“编译”**。

在本节开头给出的源代码会向端口 D 锁存器写入一个值。如果您运行这段代码并在软件模拟器或仿真器中观察该锁存器的内容，会看到该寄存器中存储的值为 0x55。但是，这并不意味着端口锁存器中的值会出现在任何器件引脚上 —— 实际上，对于先前指定的配置位和 PIC18F87J11 器件，该端口*不会*被映射到引脚上。也即，如果您测量对应于端口 D 的引脚的电压，或将 LED 与这些引脚连接，您可能不会看到您期望的电压或点亮的 LED。为了确保端口 D 与器件引脚连接，还需要执行其他一些操作。

禁止共用端口引脚的外设

本节介绍为了确保将端口锁存器中存储的值送至器件引脚上，必须向示例项目中添加的额外代码。未包含该代码是许多简单的入门程序无法按预期工作的常见原因。

8 位 Microchip PIC 器件具有大量的片上外设，但引脚的数量有限。许多外设的 IO 线可能共用相同的引脚。但是，在任意给定时刻只有一个外设可以使用某个引脚。

数字IO端口的处理方式与任何其他外设相同——除非将某个端口指定为使用某个引脚，否则不会将它与外界连接。在*许多*情况下，默认情况下端口不与引脚连接。

要了解哪些外设共用一个引脚，请参见器件数据手册。引脚图可提供快速参考，但许多 PIC 器件数据手册都包含一个充分说明每个引脚用法的章节。

使用以下步骤来确定哪些外设可能需要初始化，从而使端口与端口引脚连接。

步骤 1 在器件数据手册中找到引脚 I/O 说明或类似的表。

步骤 2 着重于表中列出的第一个备用外设。

步骤 3 确定禁止外设需要写入 SFR 的值。

步骤 4 对表中列出的所有其他外设从步骤 1 开始重复该过程。

步骤 1 在器件数据手册中找到引脚 I/O 说明或类似的表。

图 1-18 显示了从 PIC18F87J11 器件 （80 引脚封装）对应的表中摘取的内容。

图 1-18： PIC18F87J11 引脚表的摘取内容

Pin Name	Pin Number 80-TQFP	Pin Type	Buffer Type	Description
				PORTD is a bidirectional I/O port.
RD0/AD0/PMD0	72			
RD0		I/O	ST	Digital I/O.
AD0		I/O	TTL	External Memory Address/Data 0.
PMD0(6)		I/O	TTL	Parallel Master Port data.
RD1/AD1/PMD1	69			
RD1		I/O	ST	Digital I/O.
AD1		I/O	TTL	External Memory Address/Data 1.
PMD1(6)		I/O	TTL	Parallel Master Port data.
RD2/AD2/PMD2	68			
RD2		I/O	ST	Digital I/O.
AD2		I/O	TTL	External Memory Address/Data 2.
PMD2(6)		I/O	TTL	Parallel Master Port data.

对于本指南中使用的 PICDEM PIC18 Explorer 开发板， LED 与端口 D 连接。（如果需要，请查看用户指南来了解开发板连接。）端口 D 使用的引脚标记为 RD0、RD1、RD2 等。从图中可以看到，这些引脚标记还包含了其他复用信息，例如 RD0/AD0/PMD0。该表表明端口 D、外部存储器总线和并行主端口全都共用相同的引脚。端口 D 使用的其他引脚 （图 1-18 中未显示）也与 SPI 外设共用。

步骤 2 着重于表中列出的第一个备用外设。

在器件数据手册中查找与该外设相关的章节，并查找控制该外设的 SFR。

步骤 3 确定禁止外设需要写入 SFR 的值。

例如，与外部存储器总线相关的章节指示 MEMCON SFR 中的 EBDIS 位控制外部存储器模块。如图 1-19 所示的器件数据手册摘取内容所示，EBDIS 在 POR（上电复位）时的值为 0，这意味着总线处于工作状态。必须通过将 EBDIS 位置 1 来禁止该模块。

图 1-19：　　外部总线寄存器说明摘取内容

REGISTER 8-1: MEMCON: EXTERNAL MEMORY BUS CONTROL REGISTER

R/W-0	U-0	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0
EBDIS	—	WAIT1	WAIT0	—	—	WM1	WM0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **EBDIS**: External Bus Disable bit
1 = External bus is enabled when the microcontroller accesses external memory; otherwise, all external bus drivers are mapped as I/O ports
0 = External bus is always enabled, I/O ports are disabled

对并行主端口执行类似的过程之后，可以看到该端口由 PMPEN 位控制。它的 POR 值为 0，这意味着该外设在 POR 后已被禁止。因此，对于该外设，程序中无需额外的代码。[1]

步骤 4 对表中列出的所有其他外设从步骤 1 开始重复该过程。

通过重复这些步骤，现在可以如下扩展 PIC18F87J11 的测试程序[2]。

```
#include <xc.h>

// your configuration bit settings go here
// configuration code (indicated earlier) omitted for brevity

int main(void)
{
    // intialization code for your device replaces the following
    WDTCONbits.ADSHR = 1;   // enable alternate access to MEMCON
    MEMCONbits.EBDIS = 1;   // turn off external memory bus

    // code to access your port replaces the following
    TRISD = 0x0;    // set all port D bits to be output
    LATD = 0x55;    // write a value to the port latch

    return 0;
}
```

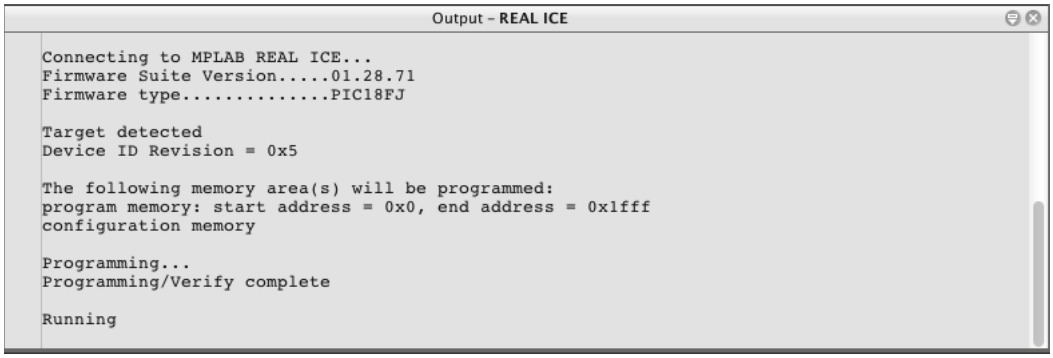
1. 显式禁止该外设并没有什么坏处。如果您的程序执行软复位，则该位的值可能不再为 0。
2. 为访问该器件上的 MEMCON 寄存器， ADSHR 位必须置 1。

下载并运行代码

上一节结尾处列出的源代码适用于 PIC18F87J11 器件。请确认您针对所用器件编写的代码按预期工作。在本节中，您可以编译、下载并运行二进制映像。

单击 Build and Run project （编译并运行项目）按钮，如图 1-12 所示。或者，如果您希望研究硬件调试器的功能，可以单击 Build and Debug project （编译并调试项目）。这两个按钮都会确保您的代码编译的二进制映像是最新的，并且会下载并执行您的代码。如果单击 Build and Debug project 按钮，它还会在您的器件中启动调试执行程序。您可以在 Output 窗口中查看代码的编译情况，与图 1-20 类似。

图 1-20： 代码的执行



在您的代码运行时，您可以看到 LED 按您在代码中指定的方式点亮，或能够测量分配给您所用端口的引脚上的电压。为端口赋予值 0x55 时，将会每秒点亮与端口的引脚连接的 LED。

但是，如果您使用的是软件模拟器，则应停止代码的执行，并确认该端口中包含的值。并且，这只能证明您正确写入了该端口，而不能证明端口将与实际器件上的引脚连接。

如果希望研究单步、断点或其他调试功能，请参见 MPLAB X IDE 文档。

实现主循环

按实际情况来说，我们在上一节中运行的简单测试程序会执行一些语句，然后终止。在执行到达 `main()` 末尾之后，由编译器添加的代码会跳回到复位向量处。然后，该器件会再次执行运行时启动代码和 `main()` 函数。并不需要这些软复位。在本节中，介绍了编译器的其他功能。您将了解如何更改程序，使它向 LED 写入不同的值，以及如何调整 `main()`，使它永远不会终止。

在以下代码中，我们通过添加无限循环来防止 `main()` 终止。在该循环中，我们将计数器的值（`portValue`）赋予端口锁存器并递增该计数器，使端口值随时间变化。此外还添加了一个延时子程序，以便可以看到 LED 的各个状态。

```
#include <xc.h>

// your configuration bit settings go here
// configuration code (indicated earlier) omitted for brevity

unsigned char portValue;

int main(void)
{
    // intialization code for your device replaces the following
    WDTCONbits.ADSHR = 1;   // enable alternate access to MEMCON
    MEMCONbits.EBDIS = 1;   // turn off external memory bus

    // code to access your port replaces the following
    TRISD = 0x0;    // set all port D bits to be output

    while(1) {
        LATD = portValue++;
        _delay(40000);
    }

    return 0;   // we should never reach this
}
```

编译并运行该代码。如果您使用的是硬件，则需要确保与端口连接的 LED 从 0 到 0xFF 递增二进制值。

请注意，端口本身不会递增。在这种表达式中使用端口寄存器可能触发读 - 修改 - 写问题。总是使用一个变量来存放您希望端口采用的值。根据需要修改该变量，然后将该变量的值赋予端口或端口锁存器。

此处使用的延时子程序（请注意前导下划线字符）实际上是编译器的一个内置函数，但您可以在《MPLAB® XC8 C 编译器用户指南》（DS50002053D_CN）的附录中找到关于该函数和编译器库函数的帮助。如果没有延时，LED 的闪烁速度会太快，看起来很暗淡。您可能需要调整延时的长度，以适应您所用器件的时钟频率。

使用中断

在本节中，会将上一节中给出的代码转换为使用中断。这可以完全在 C 语言中完成。对于 8 位器件，编译器会生成切换现场的代码，并自动链接到中断向量处。

以下是与我们在上一节中看到的代码功能等同的代码。它使用 Timer0 来产生中断，而不是使用延时。与中断关联的代码会递增计数器变量。 `main()` 中的 `while()` 循环会将计数器值写入端口 （LED），与先前一样。

```
#include <xc.h>

// your configuration bit settings go here
// configuration code (indicated earlier) omitted for brevity

unsigned char portValue;    // our counter variable

void interrupt myIsr(void)
{
    // only process timer-triggered interrupts
    if(INTCONbits.TMR0IE && INTCONbits.TMR0IF) {
        portValue++;
        INTCONbits.TMR0IF = 0;  // clear this interrupt condition
    }
}

int main(void)
{
    WDTCONbits.ADSHR = 1;   // enable alternate access to MEMCON
    MEMCONbits.EBDIS = 1;   // turn off external memory bus

    TRISD = 0x0;

    T0CON = 0b10001000;     // enable the timer as 16 bit...
                            // internal clock, no prescaler
    INTCONbits.TMR0IE = 1;  // enable interrupts for timer 0
    ei();                   // enable all interrupts

    while(1) {
        LATD = portValue;
    }

    return 0;
}
```

编译并运行该代码之后，可以看到 LED 与上一节一样发生翻转。要调整 LED 变化速率，举例来说，可以通过使能定时器的预分频器来降低其时钟频率。

请注意，使用了 `interrupt` 说明符来将函数 `myIsr()` 转变为中断函数。由于这一个中断函数可能需要处理多个中断源，所以添加了代码来确保只有在相应定时器产生中断时才递增计数器。在中断函数内放置尽可能少的代码是一种好做法。

在 `main()` 中，可以注意到使用了二进制常量（使用 `0b` 前缀），以便可以轻松看到 T0CON 中的位。请记住，如果您使用的是其他器件，则需要查看器件的数据手册，确定寄存器名称和这些寄存器内为使定时器正确工作必须正确设置的位。

编译器宏 `ei()` 用于允许中断，但如果您愿意，可以显式地将 INTCON 寄存器中的 GIE 位置 `1`。

结论

使用本指南中提出的基本概念和方法，您将能够为 8 位 PIC 器件编写相当高级的程序。您将能够正确配置器件，确定所有 SFR 的名称和您的器件使用的 SFR 位。此外，您可以让器件外设触发中断，让您的代码对这些事件进行响应。

熟悉该编译器实现的 C89 ANSI 标准 C 语言是非常重要的。《MPLAB® XC8 C 编译器用户指南》（DS50002053D_CN）提供了关于编译器操作和非标准语法的更详细信息。该文档可以在编译器安装目录的 DOCS 目录中找到。此外，也可以通过单击项目仪表板中的 Compiler Help（编译器帮助）按钮（仪表板的垂直按钮行底端的蓝色 “?” 按钮，如图 1-11 所示）来访问它。

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 愿与那些注重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。

ISBN：978-1-63277-705-8

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
═ ISO/TS 16949 ═

Microchip 位于美国亚利桑那州 Chandler 和 Tempe 与位于俄勒冈州 Gresham 的全球总部、设计和晶圆生产厂及位于美国加利福尼亚州和印度的设计中心均通过了 ISO/TS-16949:2009 认证。 Microchip 的 PIC® MCU 与 dsPIC® DSC、KEELOQ® 跳码器件、串行 EEPROM、单片机外设、非易失性存储器和模拟产品严格遵守公司的质量体系流程。此外， Microchip 在开发系统的设计和生产方面的质量体系也已通过了 ISO 9001:2000 认证。

全球销售及服务网点

美洲

公司总部 Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 1-480-792-7200
Fax: 1-480-792-7277
技术支持：
http://www.microchip.com/support
网址：www.microchip.com

亚特兰大 Atlanta
Duluth, GA
Tel: 1-678-957-9614
Fax: 1-678-957-1455

奥斯汀 Austin, TX
Tel: 1-512-257-3370

波士顿 Boston
Westborough, MA
Tel: 1-774-760-0087
Fax: 1-774-760-0088

芝加哥 Chicago
Itasca, IL
Tel: 1-630-285-0071
Fax: 1-630-285-0075

克里夫兰 Cleveland
Independence, OH
Tel: 1-216-447-0464
Fax: 1-216-447-0643

达拉斯 Dallas
Addison, TX
Tel: 1-972-818-7423
Fax: 1-972-818-2924

底特律 Detroit
Novi, MI
Tel: 1-248-848-4000

休斯敦 Houston, TX
Tel: 1-281-894-5983

印第安纳波利斯 Indianapolis
Noblesville, IN
Tel: 1-317-773-8323
Fax: 1-317-773-5453

洛杉矶 Los Angeles
Mission Viejo, CA
Tel: 1-949-462-9523
Fax: 1-949-462-9608

纽约 New York, NY
Tel: 1-631-435-6000

圣何塞 San Jose, CA
Tel: 1-408-735-9110

加拿大多伦多 Toronto
Tel: 1-905-673-0699
Fax: 1-905-673-6509

亚太地区

亚太总部 Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2943-5100
Fax: 852-2401-3431

中国 - 北京
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

中国 - 成都
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

中国 - 重庆
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

中国 - 东莞
Tel: 86-769-8702-9880

中国 - 杭州
Tel: 86-571-8792-8115
Fax: 86-571-8792-8116

中国 - 香港特别行政区
Tel: 852-2943-5100
Fax: 852-2401-3431

中国 - 南京
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

中国 - 青岛
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

中国 - 上海
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

中国 - 沈阳
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

中国 - 深圳
Tel: 86-755-8864-2200
Fax: 86-755-8203-1760

中国 - 武汉
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

中国 - 西安
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

中国 - 厦门
Tel: 86-592-238-8138
Fax: 86-592-238-8130

中国 - 珠海
Tel: 86-756-321-0040
Fax: 86-756-321-0049

亚太地区

台湾地区 - 高雄
Tel: 886-7-213-7828

台湾地区 - 台北
Tel: 886-2-2508-8600
Fax: 886-2-2508-0102

台湾地区 - 新竹
Tel: 886-3-5778-366
Fax: 886-3-5770-955

澳大利亚 Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

印度 India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

印度 India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

印度 India - Pune
Tel: 91-20-3019-1500

日本 Japan - Osaka
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310

日本 Japan - Tokyo
Tel: 81-3-6880-3770
Fax: 81-3-6880-3771

韩国 Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

韩国 Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 或
82-2-558-5934

马来西亚 Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

马来西亚 Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

菲律宾 Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

新加坡 Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

泰国 Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

欧洲

奥地利 Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

丹麦 Denmark-Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

法国 France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

德国 Germany - Dusseldorf
Tel: 49-2129-3766400

德国 Germany - Karlsruhe
Tel: 49-721-625370

德国 Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

意大利 Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

意大利 Italy - Venice
Tel: 39-049-7625286

荷兰 Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

波兰 Poland - Warsaw
Tel: 48-22-3325737

西班牙 Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

瑞典 Sweden - Stockholm
Tel: 46-8-5090-4654

英国 UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820

07/14/15