MPLAB® XC8嵌入式工程师用户指南——PIC® MCU

简介

本文档提供了5个适用于8位PIC MCU 器件和MPLAB XC8 C编译器的代码示例，这些代码示例使用通用C接口（Common C Interface，CCI）。关于CCI的更多信息，请参见《MPLAB® XC8 C编译器用户指南》（DS50002053D_CN）。

读者需要掌握一些单片机和C编程语言的相关知识。

1. 点亮或熄灭LED
2. 使用 `_delay()` 函数使LED闪烁
3. 使用中断作为延时在LED上递增计数
4. 使用A/D在LED上显示电位器值
5. 在LED上显示EEPROM数据值

A 在MPLAB X IDE中运行代码
B 获取软件和硬件

© 2016-2020 Microchip Technology Inc.

1. 点亮或熄灭LED

本示例将在带PIC16F1719单片机（MCU）的Explorer 8板上交替点亮各LED。更多信息，请参见**第B节“获取软件和硬件”**。

```
// PIC16F1719 Configuration Bit Settings

// For more on Configuration Bits,   <-- 请参见第1.1节
// consult your device data sheet

// CONFIG1
#pragma config FOSC = ECH      // External Clock, 4-20 MHz
#pragma config WDTE = OFF      // Watchdog Timer (WDT) disabled
#pragma config PWRTE = OFF     // Power-up Timer disabled
#pragma config MCLRE = ON      // MCLR/VPP pin function is MCLR
#pragma config CP = OFF        // Flash Memory Code Protection off
#pragma config BOREN = ON      // Brown-out Reset enabled
#pragma config CLKOUTEN = OFF // Clock Out disabled.
#pragma config IESO = ON       // Internal/External Switchover on
#pragma config FCMEN = ON      // Fail-Safe Clock Monitor enabled

// CONFIG2
#pragma config WRT = OFF       // Flash Memory Self-Write Protect off
#pragma config PPS1WAY = ON    // PPS one-way control enabled
#pragma config ZCDDIS = ON     // Zero-cross detect disabled
#pragma config PLLEN = OFF     // Phase Lock Loop disable
#pragma config STVREN = ON     // Stack Over/Underflow Reset enabled
#pragma config BORV = LO       // Brown-out Reset low trip point
#pragma config LPBOR = OFF     // Low-Power Brown Out Reset disabled
#pragma config LVP = OFF       // Low-Voltage Programming disabled

// #pragma config statements should precede project file includes.
// Use project enums instead of #define for ON and OFF.

#include <xc.h>   <-- 请参见第1.2节
#include <stdint.h>

void main(void) {

    uint8_t portValue = 0x05;   <-- 请参见第1.3节

    // Port B access   <-- 请参见第1.4节

    ANSELB = 0x0; // set to digital I/O (not analog)
    TRISB = 0x0;  // set all port bits to be output
    LATB = portValue; // write to port latch - RB[0:3] = LED[4:7]

    // Port D access
    ANSELD = 0x0; // set to digital I/O (not analog)
    TRISD = 0x0;  // set all port bits to be output
    LATD = portValue; // write to port latch - RD[0:3] = LED[0:3]

}
```


© 2016-2020 Microchip Technology Inc.

1.1 配置位

Microchip 器件具有配置寄存器，其中的位可用于使能和/或设置器件功能。

> **注：** 如果未正确设置配置位，器件将无法运行，或至少不按预期运行。

要设置的配置位

请特别注意以下设置：

- **振荡器选择**——该项必须与硬件的振荡器电路匹配。如果设置有误，*器件时钟可能无法运行*。通常情况下，开发板使用高速晶振。示例中的相关代码如下：
  ```
  #pragma config FOSC = ECH
  ```
- **看门狗定时器**——建议在必要时禁止该定时器。这样可防止*意外复位*。示例中的相关代码如下：
  ```
  #pragma config WDTE = OFF
  ```
- **代码保护**——在必要时关闭代码保护。这样可确保*器件存储器可完全访问*。示例中的相关代码如下：
  ```
  #pragma config CP = OFF
  ```

使用其他8位器件（而非本示例中使用的PIC16F1719 MCU）时，可能需要设置不同的配置位。请参见具体器件数据手册了解相应配置位的名称和功能。可使用部件编号在 https://www.microchip.com 上搜索相应的数据手册。

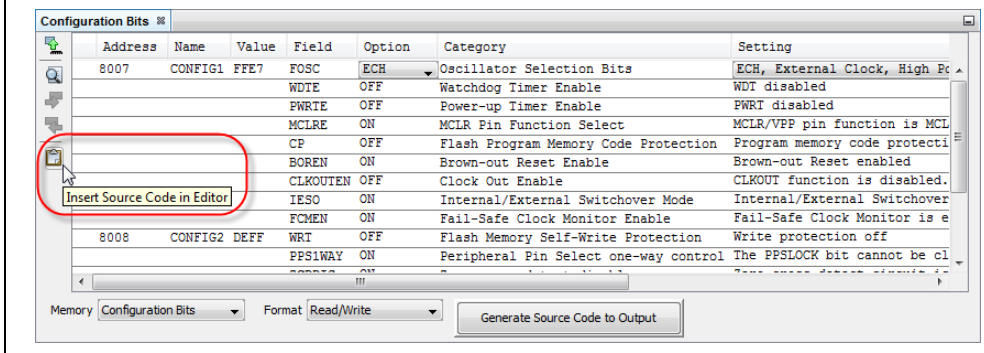
有关每款器件可用配置位的更多信息，请参见MPLAB XC8安装路径下的如下文件：

MPLAB XC8 Installation Directory/docs/chips

设置配置位的方法

在MPLAB X IDE中，可使用Configuration Bits（配置位）窗口查看和设置这些位。请选择*Window>Target Memory Views>Configuration Bits*（窗口>目标存储器视图>配置位）打开该窗口。

图1： 配置位窗口



选择好设置后，在需要添加此信息的代码处单击，然后单击**Insert Source Code in Editor**（在编辑器中插入源代码）图标，如示例代码中所示。

关于此窗口的更多信息，请参见MPLAB X IDE文档。

1.2 包含头文件

<xc.h>头文件允许源文件中的代码访问编译器特定或器件特定的功能。编译器将根据您选择的器件设置相应的宏，以使xc.h指向正确的器件特定头文件。请勿将器件特定的头文件包含在您的代码中，否则将导致代码不可移植。

stdint.h头文件定义固定长度的整型。例如，uint8_t是无符号8位整型。

可在MPLAB XC8安装目录的pic/include子目录下找到这些头文件和其他头文件。

1.3 用于LED值的变量

如下一节所述，要写入LED的值已赋给一个变量（portValue），即LED D1、D3、D5和D7将点亮，LED D2、D4、D6和D8将熄灭。有关电路板原理图（**第B.4节“获取并设置Explorer 8板”**）的信息，请参见*Explorer 8 Development Board User's Guide*（DS40001812）。

1.4 端口访问

器件数字I/O引脚可与外设I/O引脚复用。为确保当前仅使用数字I/O，需禁止其他外设。为此，可使用代表外设寄存器及其位的预定义C变量。这些变量列于编译器include目录下的器件特定头文件中。关于哪些外设共用哪些引脚的信息，请参见具体器件的数据手册。

对于本节中的示例，端口B和端口D引脚与默认禁止的外设复用。端口引脚默认设置为模拟，因此需要将它们设置为数字I/O。对于端口B：

```
ANSELB = 0x0; // set to digital I/O (not analog)
```

器件引脚连接至器件的数字I/O端口（PORT）或锁存器（LAT）寄存器。本示例中使用LATD和LATB。宏LEDS_ON_OFF赋值给两个锁存器。对于端口D：

```
LATD = portValue; // write to port latch - RD[0:3] = LED[0:3]
```

此外，还有一个寄存器用于指定引脚的方向是输入还是输出，它称为TRIS寄存器。本节的示例中使用TRISB和TRISD。将位设置为0可将引脚设为输出，将位设置为1可将引脚设为输入。对于端口B：

```
TRISB = 0x0; // set all port bits to be output
```


© 2016-2020 Microchip Technology Inc.

2. 使用_delay()函数使LED闪烁

本示例在上一示例代码的基础上进行修改。本代码不仅点亮LED，还将使LED交替闪烁。

```
// PIC16F1719 Configuration Bit Settings

// For more on Configuration Bits, consult your device data sheet

// CONFIG1
#pragma config FOSC = ECH      // External Clock, 4-20 MHz
#pragma config WDTE = OFF      // Watchdog Timer (WDT) disabled
#pragma config PWRTE = OFF     // Power-up Timer disabled
#pragma config MCLRE = ON      // MCLR/VPP pin function is MCLR
#pragma config CP = OFF        // Flash Memory Code Protection off
#pragma config BOREN = ON      // Brown-out Reset enabled
#pragma config CLKOUTEN = OFF // Clock Out disabled.
#pragma config IESO = ON       // Internal/External Switchover on
#pragma config FCMEN = ON      // Fail-Safe Clock Monitor enabled

// CONFIG2
#pragma config WRT = OFF       // Flash Memory Self-Write Protect off
#pragma config PPS1WAY = ON    // PPS one-way control enabled
#pragma config ZCDDIS = ON     // Zero-cross detect disabled
#pragma config PLLEN = OFF     // Phase Lock Loop disable
#pragma config STVREN = ON     // Stack Over/Underflow Reset enabled
#pragma config BORV = LO       // Brown-out Reset low trip point
#pragma config LPBOR = OFF     // Low-Power Brown Out Reset disabled
#pragma config LVP = OFF       // Low-Voltage Programming disabled

// #pragma config statements should precede project file includes.
// Use project enums instead of #define for ON and OFF.

#include <xc.h>
#include <stdint.h>

void main(void) {

    uint8_t portValue;

    // Port B access
    ANSELB = 0x0; // set to digital I/O (not analog)
    TRISB = 0x0;  // set all port bits to be output

    // Port D access
    ANSELD = 0x0; // set to digital I/O (not analog)
    TRISD = 0x0;  // set all port bits to be output

    while(1) {   ◄—— 请参见第2.1节

        portValue = 0x05;
        LATB = portValue; // RB[0:3] = LED[4:7]
        LATD = portValue; // RD[0:3] = LED[0:3]

        // delay value change   ◄—— 请参见第2.2节

        _delay(25000); // delay in instruction cycles
```

```
        portValue = 0x0A;
        LATB = portValue; // RB[0:3] = LED[4:7]
        LATD = portValue; // RD[0:3] = LED[0:3]
        _delay(25000); // delay in instruction cycles

    }
}
```

2.1 while()循环和变量值

要使端口B和端口D上的LED发生变化，首先为变量portValue赋值0x05（LED 0, 2, 4, 6点亮），然后为该变量赋互补值0x0A（LED 1,3,5,7 点亮）。使用了while(1)执行循环。

2.2 _delay()函数

由于执行速度在大多数情况下都会导致LED的闪烁速度超出人眼的识别能力，因此需要降低执行速度。_delay()是编译器的内置函数。

有关内置延时函数的更多详细信息，请参见《MPLAB® XC8 C编译器用户指南》（DS50002053D_CN）。

© 2016-2020 Microchip Technology Inc.

3. 使用中断作为延时在LED上递增计数

本示例在上一示例代码的基础上进行修改。尽管上一示例中的延时循环对于降低循环执行速度很有用，但是会在程序中引入停滞时间。为避免这一问题，可使用定时器中断。

```
// PIC16F1719 Configuration Bit Settings

// For more on Configuration Bits, consult your device data sheet

// CONFIG1
#pragma config FOSC = ECH      // External Clock, 4-20 MHz
#pragma config WDTE = OFF      // Watchdog Timer (WDT) disabled
#pragma config PWRTE = OFF     // Power-up Timer disabled
#pragma config MCLRE = ON      // MCLR/VPP pin function is MCLR
#pragma config CP = OFF        // Flash Memory Code Protection off
#pragma config BOREN = ON      // Brown-out Reset enabled
#pragma config CLKOUTEN = OFF // Clock Out disabled.
#pragma config IESO = ON       // Internal/External Switchover on
#pragma config FCMEN = ON      // Fail-Safe Clock Monitor enabled

// CONFIG2
#pragma config WRT = OFF       // Flash Memory Self-Write Protect off
#pragma config PPS1WAY = ON    // PPS one-way control enabled
#pragma config ZCDDIS = ON     // Zero-cross detect disabled
#pragma config PLLEN = OFF     // Phase Lock Loop disable
#pragma config STVREN = ON     // Stack Over/Underflow Reset enabled
#pragma config BORV = LO       // Brown-out Reset low trip point
#pragma config LPBOR = OFF     // Low-Power Brown Out Reset disabled
#pragma config LVP = OFF       // Low-Voltage Programming disabled

// #pragma config statements should precede project file includes.
// Use project enums instead of #define for ON and OFF.

#include <xc.h>
#include <stdint.h>

// Interrupt function  <-- 请参见第3.1节

void __interrupt() tcInt(void){
   // only process Timer0-triggered interrupts
   if(INTCONbits.TMR0IE && INTCONbits.TMR0IF) {
            // static variable for permanent storage duration
            static uint8_t portValue;
            // write to port latches
            LATB = (portValue++ >> 4);  // RB[0:3] = LED[4:7]
            LATD = portValue++;  // RD[0:3] = LED[0:3]
            // clear this interrupt condition
            INTCONbits.TMR0IF = 0;
   }
}

void main(void){

    // Port B access
    ANSELB = 0x0; // set to digital I/O (not analog)
    TRISB = 0x0;  // set all port bits to be output

    // Port D access
    ANSELD = 0x0; // set to digital I/O (not analog)
    TRISD = 0x0;  // set all port bits to be output
```

```
    // Timer0 setup   <-- 请参见第3.2节

    OPTION_REG = 0xD7; // timer 0 internal clock, prescaler 1:256
    INTCONbits.TMR0IE = 1; // enable interrupts for timer 0
    ei(); // enable all interrupts

    while(1);
}
```

3.1 中断函数isr()

可使用__interrupt()说明符使函数成为中断函数。由于tcInt()中断函数可能必须处理多个中断源，因此添加了相关代码以确保计数器portValue仅在Timer0产生中断时递增。

3.2 Timer0设置

由于变量值更改在中断服务程序中执行，因此还需要向主程序中添加相关代码以使能和设置定时器、允许定时器中断以及更改锁存器赋值。

要允许所有中断，请使用xc.h中定义的ei()。

© 2016-2020 Microchip Technology Inc.

4. 使用A/D在LED上显示电位器值

本示例与上一示例使用相同的器件以及相同的端口B和端口D LED。但在本示例中，将使用演示板上电位器的值通过端口A提供A/D输入，然后进行转换并显示在LED上。

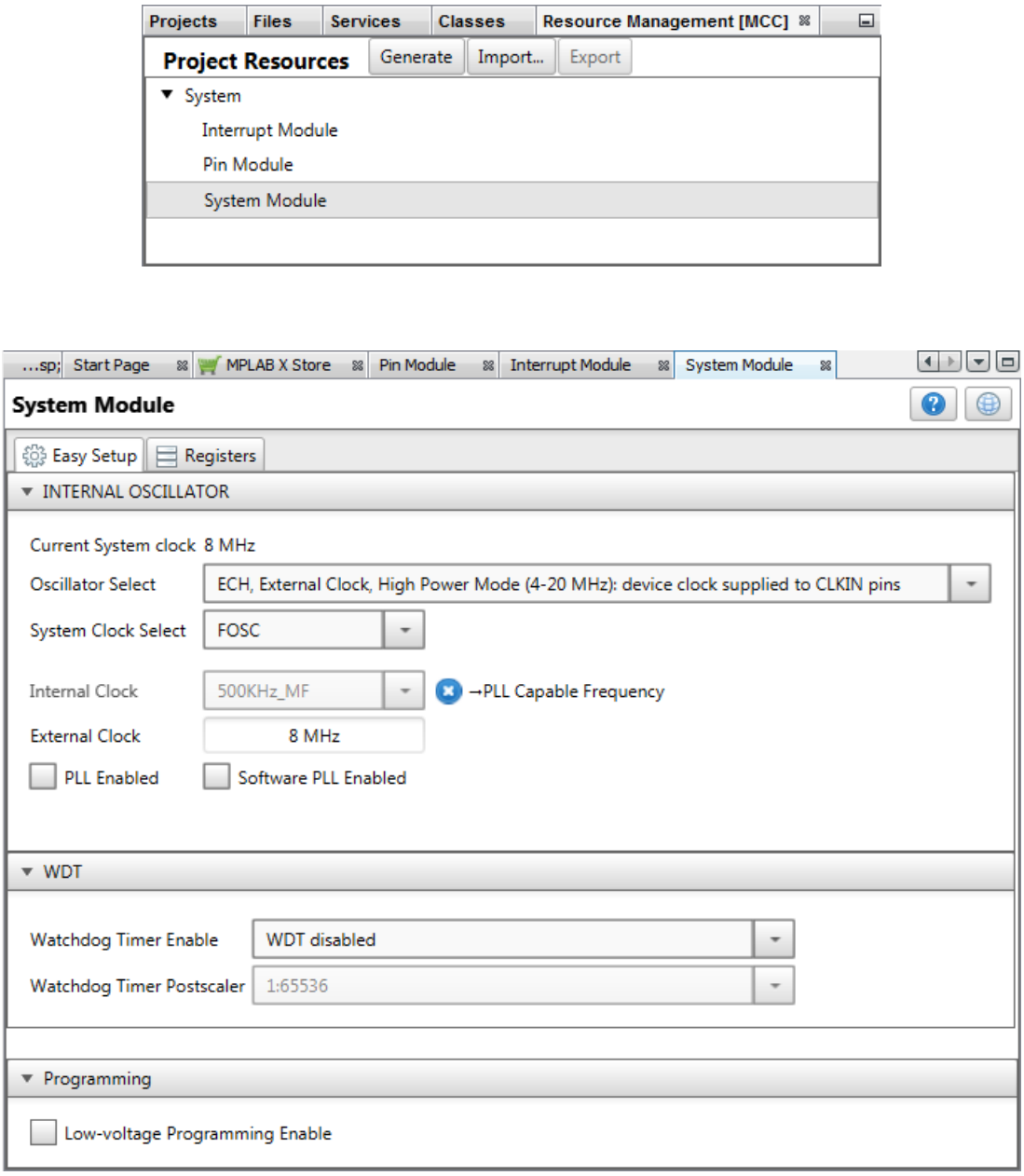
代码无需手动编写，而是使用MPLAB代码配置器（MPLAB Code Configurator，MCC）生成。MCC是一款插件，可在MPLAB X IDE菜单*Tools>Plugins*（工具>插件）的**Available Plugins**（可用插件）选项卡下安装。有关如何安装插件的更多信息，请参见MPLAB X IDE帮助。

有关MCC的信息（包括《MPLAB®代码配置器用户指南》（DS40001725B_CN）），请访问MPLAB代码配置器网页：

https://www.microchip.com/mplab/mplab-code-configurator

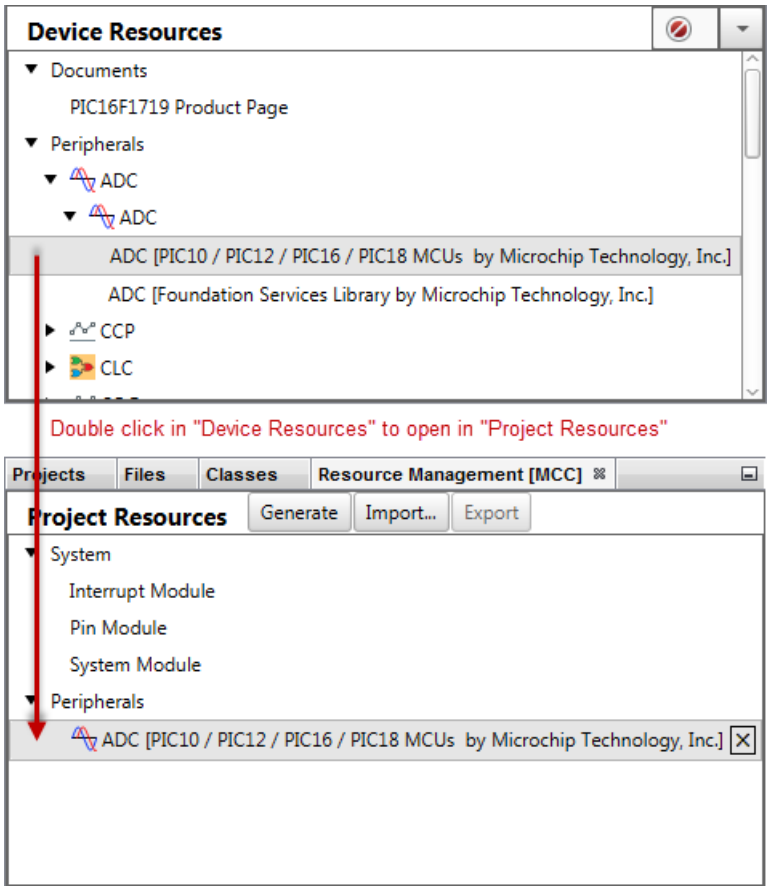
本示例中的MCC GUI设置如以下各图所示。

图2： ADC项目系统资源配置



© 2016-2020 Microchip Technology Inc.

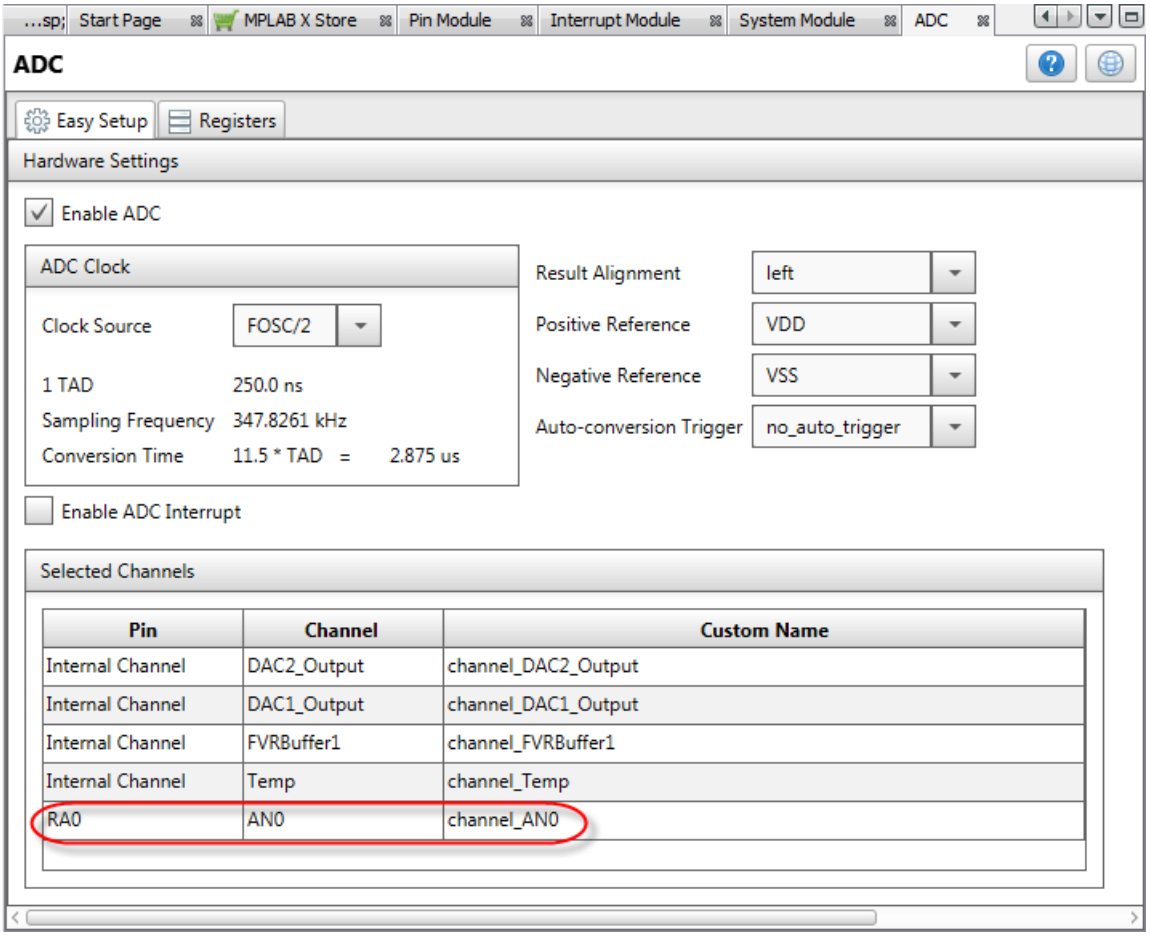
图3： ADC项目ADC资源选择



PIC10/PIC12/PIC16/PIC18 MCU选择使您可以通过Easy Setup（轻松设置）视图直接进行外设的细节设置。

Foundation Services Library（基础服务库）本质上较为抽象，通常提供外设的基本功能，并可简化配置过程。

图4：　　ADC项目ADC资源配置



RA0至AN0映射将在图5中的选择完成后显示。

© 2016-2020 Microchip Technology Inc.

图5：　　ADC项目ADC引脚资源网格

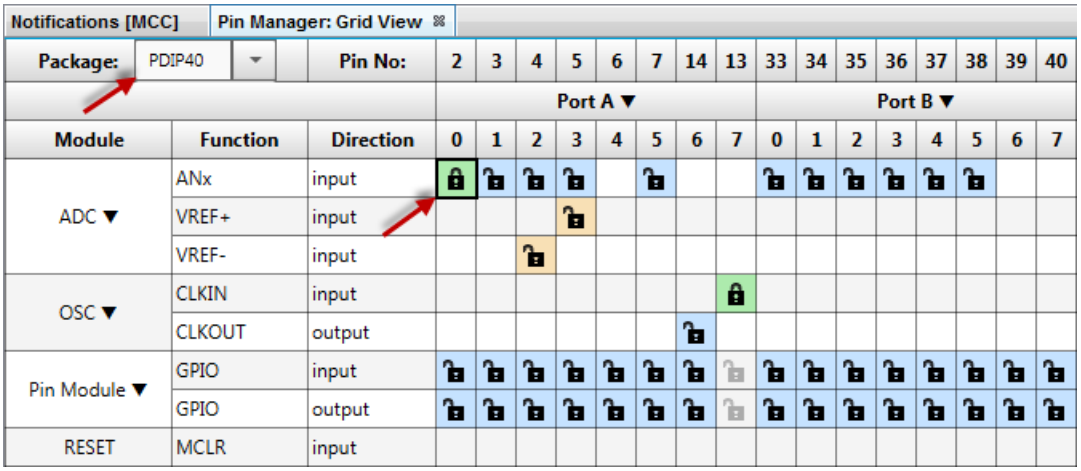
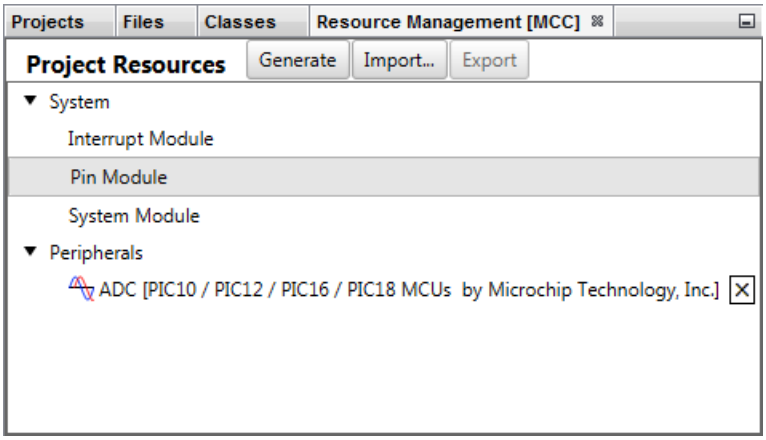
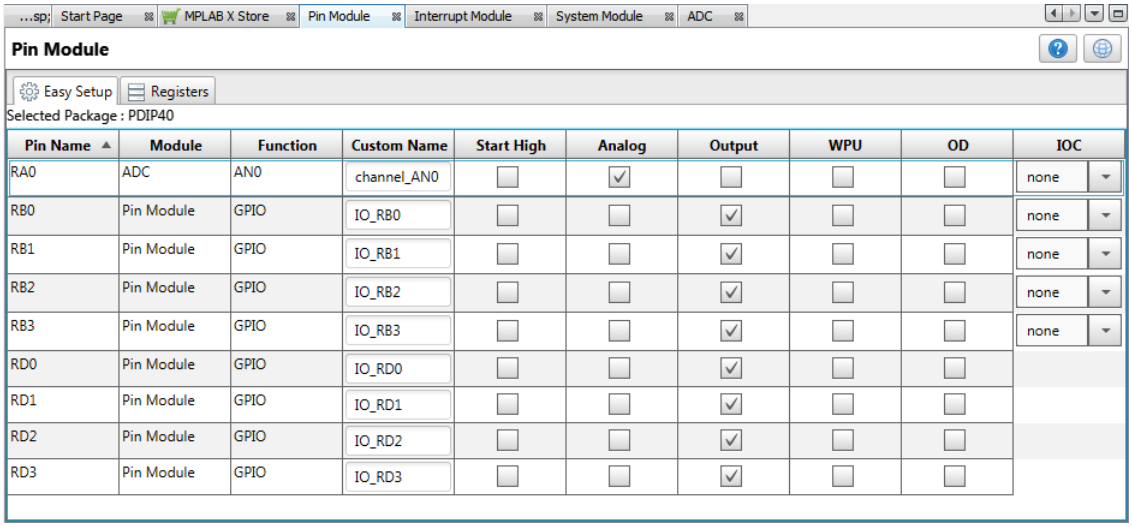


图6： ADC项目引脚资源配置





在图7中选择引脚RB0:3和RD0:3 后，它们将出现在上面的窗口中。

前面在图5中选择了RA0。

一旦引脚出现在窗口中，即可为每个引脚选择引脚配置。

© 2016-2020 Microchip Technology Inc.

图7： ADC项目GPIO引脚资源——网格

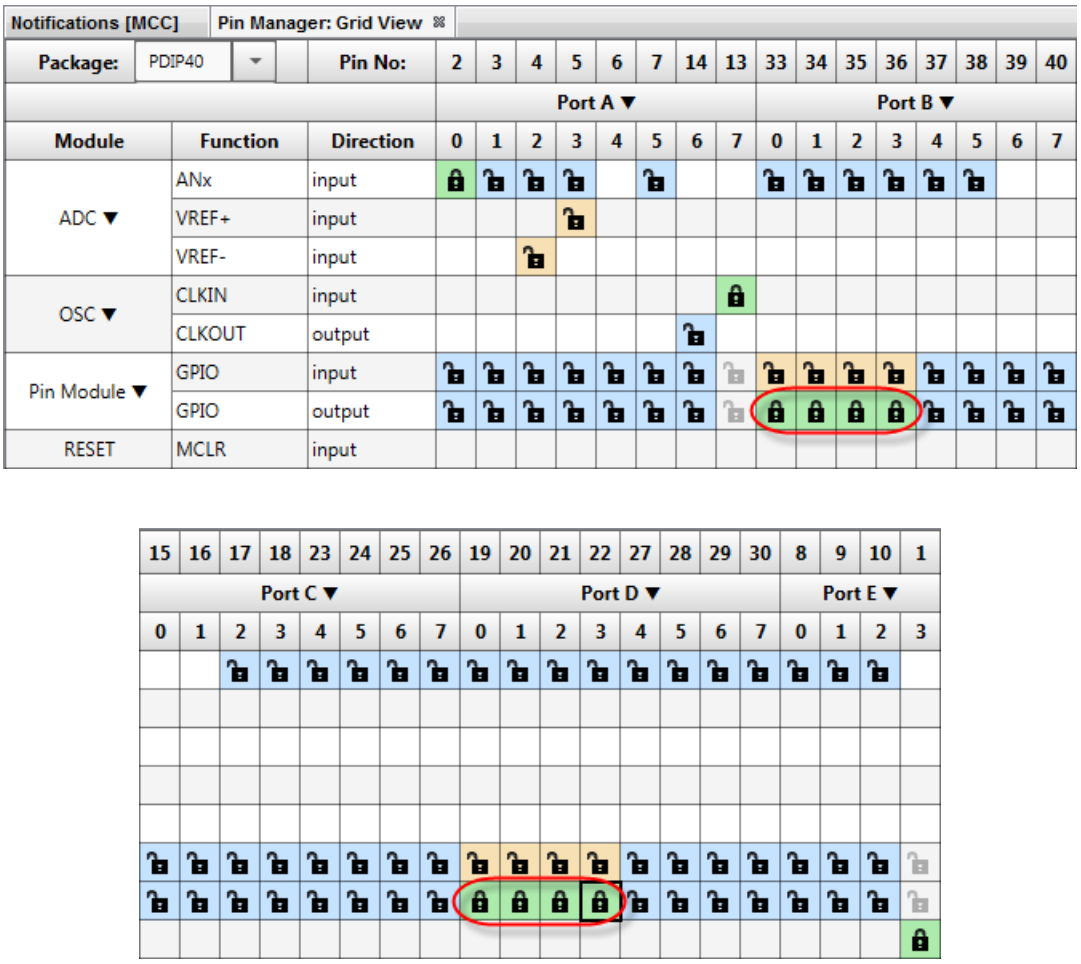
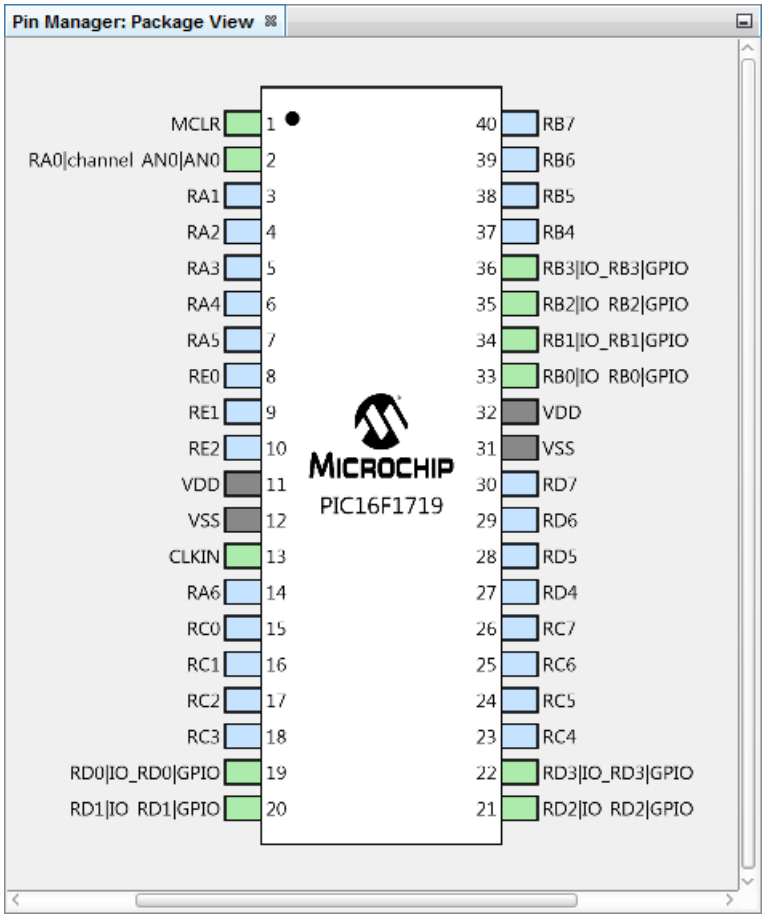


图8：　　ADC项目GPIO引脚资源——封装

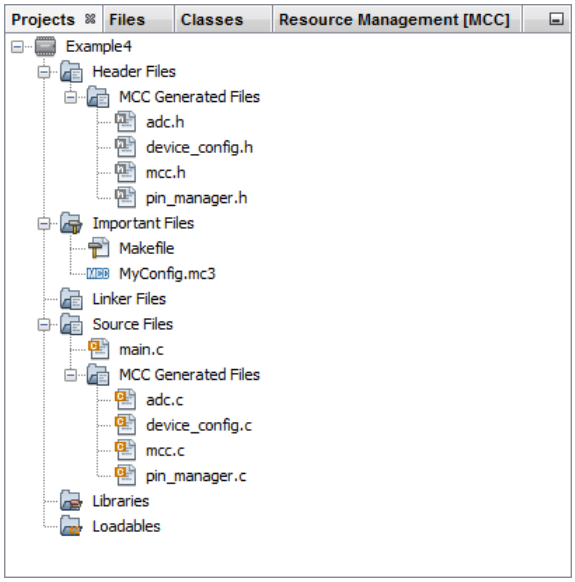


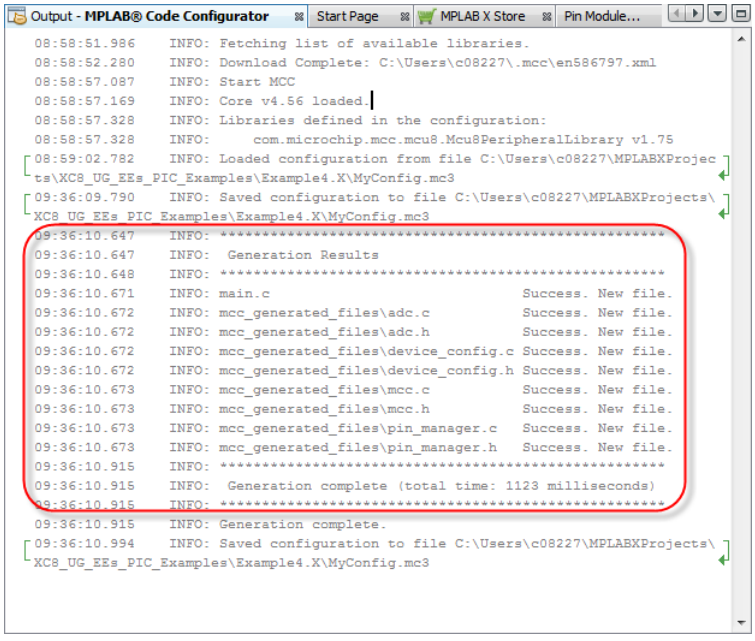
© 2016-2020 Microchip Technology Inc.

按照上述各图配置完代码后，请单击“Project Resources”（项目资源）窗口上的**Generate**（生成）按钮。通过MCC生成的代码是模块化的。因此，主程序代码、系统代码和外设代码均位于单独的文件中。此外，每个外设都有自己的头文件。

向程序中添加功能时始终需要编辑`main.c`。请查看生成的文件以找到您的代码中可能需要的任何函数或宏。

图9：　　通过MCC生成的ADC代码





4.1 修改后的main.c代码

main.c模板文件已经过编辑，如下所示。部分注释已删除（< >内）。新添加到main()中的代码以红色显示。

```
/**
  Generated Main Source File

<See generated main.c file for file information.>
 */

/*
  (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use
  this software and any derivatives exclusively with Microchip
  products.

<See generated main.c file for additional copyright information.>
 */

#include "mcc_generated_files/mcc.h"

/*
                         Main application
 */
void main(void) {
    // initialize the device
    SYSTEM_Initialize();

    // <No interrupts used - see generated main.c file for code.>

    while (1) {

        // Select A/D channel <-- 请参见第4.2节

        ADC_SelectChannel(channel_AN0);

        // Start A/D conversion
        ADC_StartConversion();

        // Wait for ADC to complete <-- 请参见第4.3节

        while(!ADC_IsConversionDone());

        // Write to Port Latches <-- 请参见第4.4节

        LATD = ADRESH; // RD[0:3] = LED[0:3]
        LATB = (ADRESH >> 4); // RB[0:3] = LED[4:7]

    }
}
/**
 End of File
 */
```


© 2016-2020 Microchip Technology Inc.

4.2 选择A/D通道和启动转换

使用adc.c模块中的如下函数：

```
void ADC_SelectChannel(adc_channel_t channel)
```

变量channel属于adc.h中定义的typedef adc_channel_t。对于本示例，电位器输入位于RA0上，因此选择channel_AN0。

使用如下函数启动A/D转换：

```
void ADC_StartConversion()
```

4.3 等待ADC完成

使用adc.c模块中的如下函数：

```
bool ADC_IsConversionDone()
```

该函数会返回ADCON0bits.GO_nDONE位（在器件头文件中定义）的取反值。但是，main while循环中需要该位的实际值，因此需要将返回值再次取反。

4.4 写入端口锁存器

由于只有8个LED，因此仅显示ADRESH的值。低4位通过LATD显示在LED 0至LED 3上，高4位会先进行移位，以便能够通过LATB显示在LED 4至LED 7上。

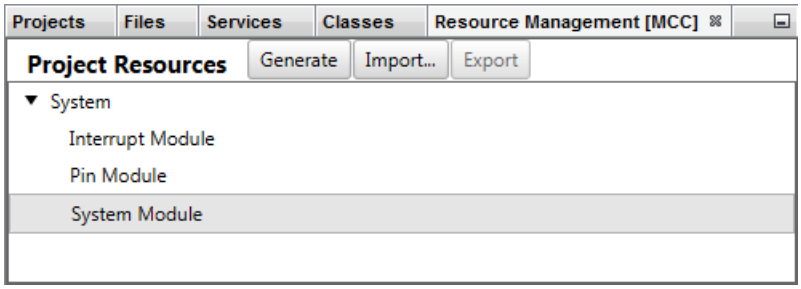
5. 在LED上显示EEPROM数据值

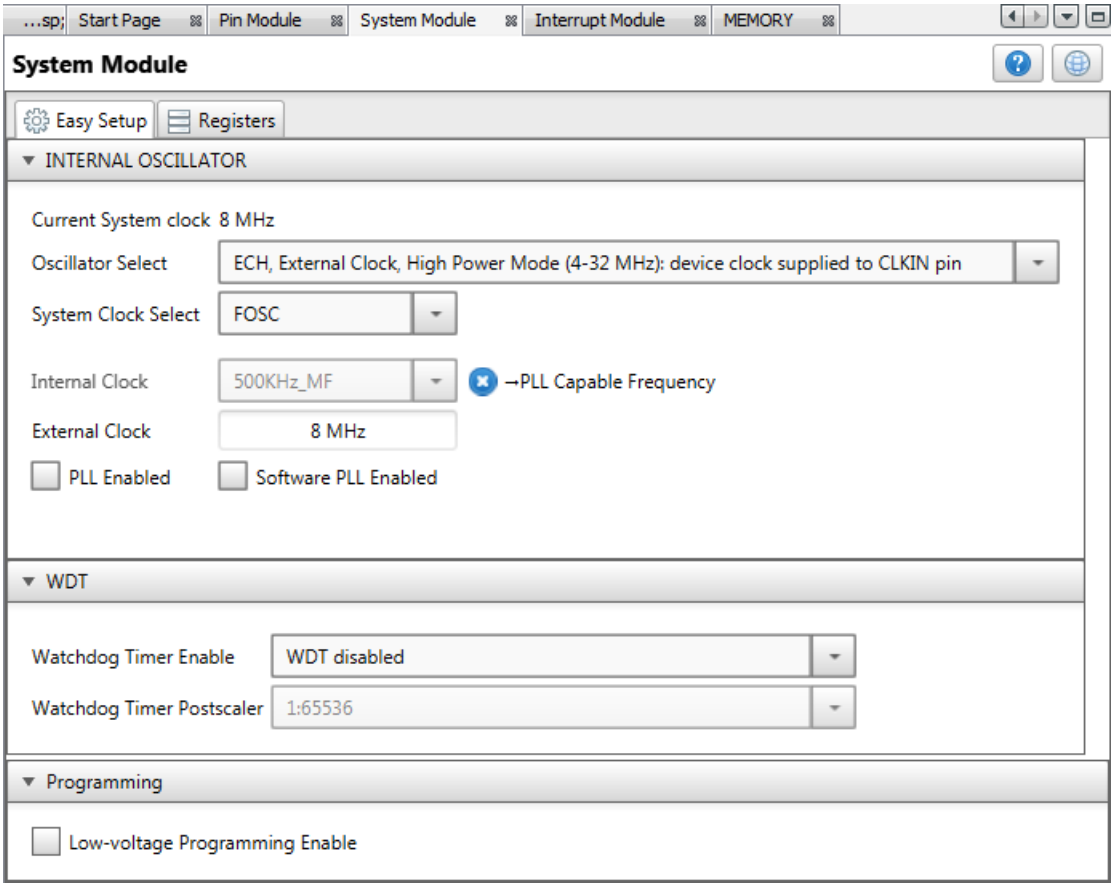
本示例使用另一款Microchip器件PIC16F1939 MCU来演示如何读写EEPROM数据（EEData）。读取的值显示在端口D和端口B LED上。

再次使用MPLAB代码配置器（MCC）来生成大部分代码。欲了解如何安装MCC和获取MCC用户指南的信息，请参见：
第4节“使用A/D在LED上显示电位器值”。

本示例中的MCC GUI设置如以下各图所示。

图10： EEDATA项目系统资源配置





© 2016-2020 Microchip Technology Inc.

图11： EEDATA项目存储器资源配置

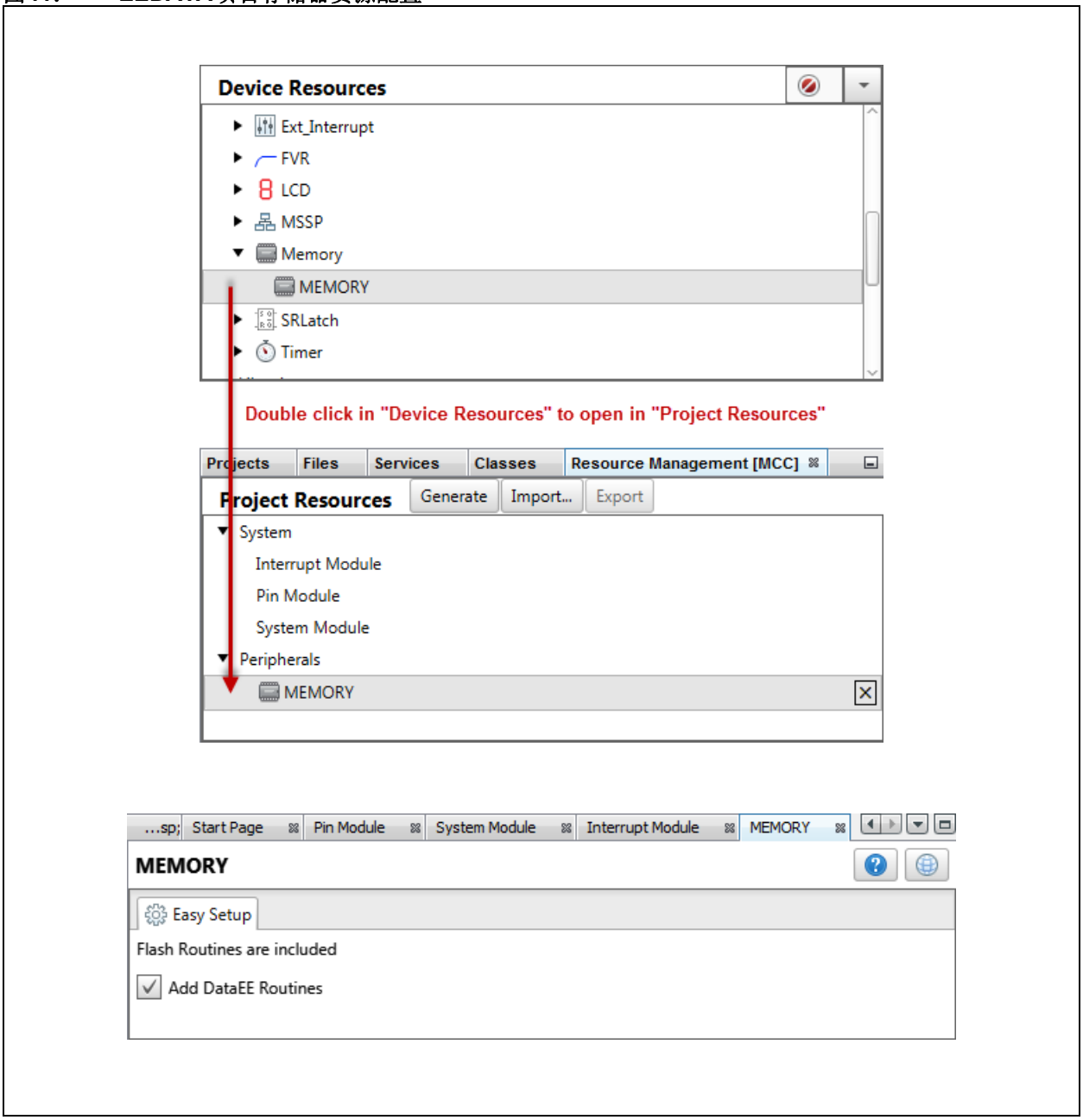
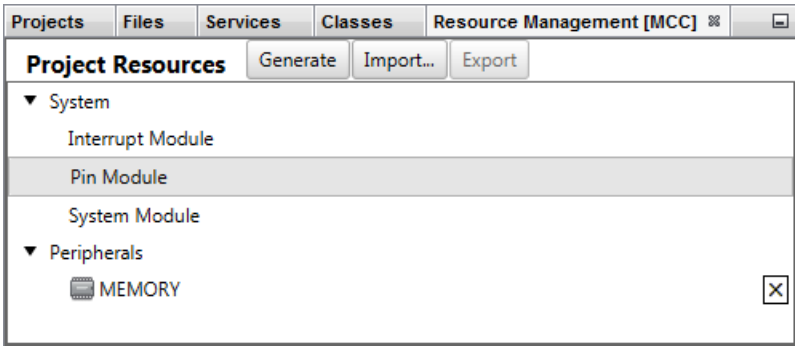
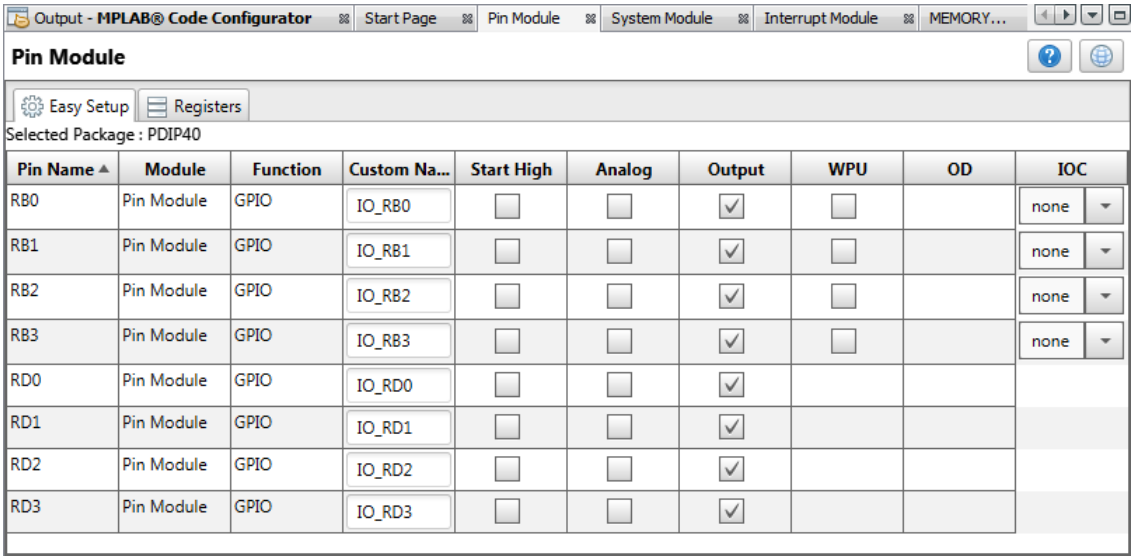


图12： EEDATA项目引脚资源配置





Pin Name	Module	Function	Custom Na...	Start High	Analog	Output	WPU	OD	IOC
RB0	Pin Module	GPIO	IO_RB0	☐	☐	☑	☐		none
RB1	Pin Module	GPIO	IO_RB1	☐	☐	☑	☐		none
RB2	Pin Module	GPIO	IO_RB2	☐	☐	☑	☐		none
RB3	Pin Module	GPIO	IO_RB3	☐	☐	☑	☐		none
RD0	Pin Module	GPIO	IO_RD0	☐	☐	☑			
RD1	Pin Module	GPIO	IO_RD1	☐	☐	☑			
RD2	Pin Module	GPIO	IO_RD2	☐	☐	☑			
RD3	Pin Module	GPIO	IO_RD3	☐	☐	☑			

在图13中选择引脚RB0:3和RD0:3 后，它们将出现在上面的窗口中。
一旦引脚出现在窗口中，即可为每个引脚选择引脚配置。

© 2016-2020 Microchip Technology Inc.

图13： EEDATA项目GPIO引脚资源——网格

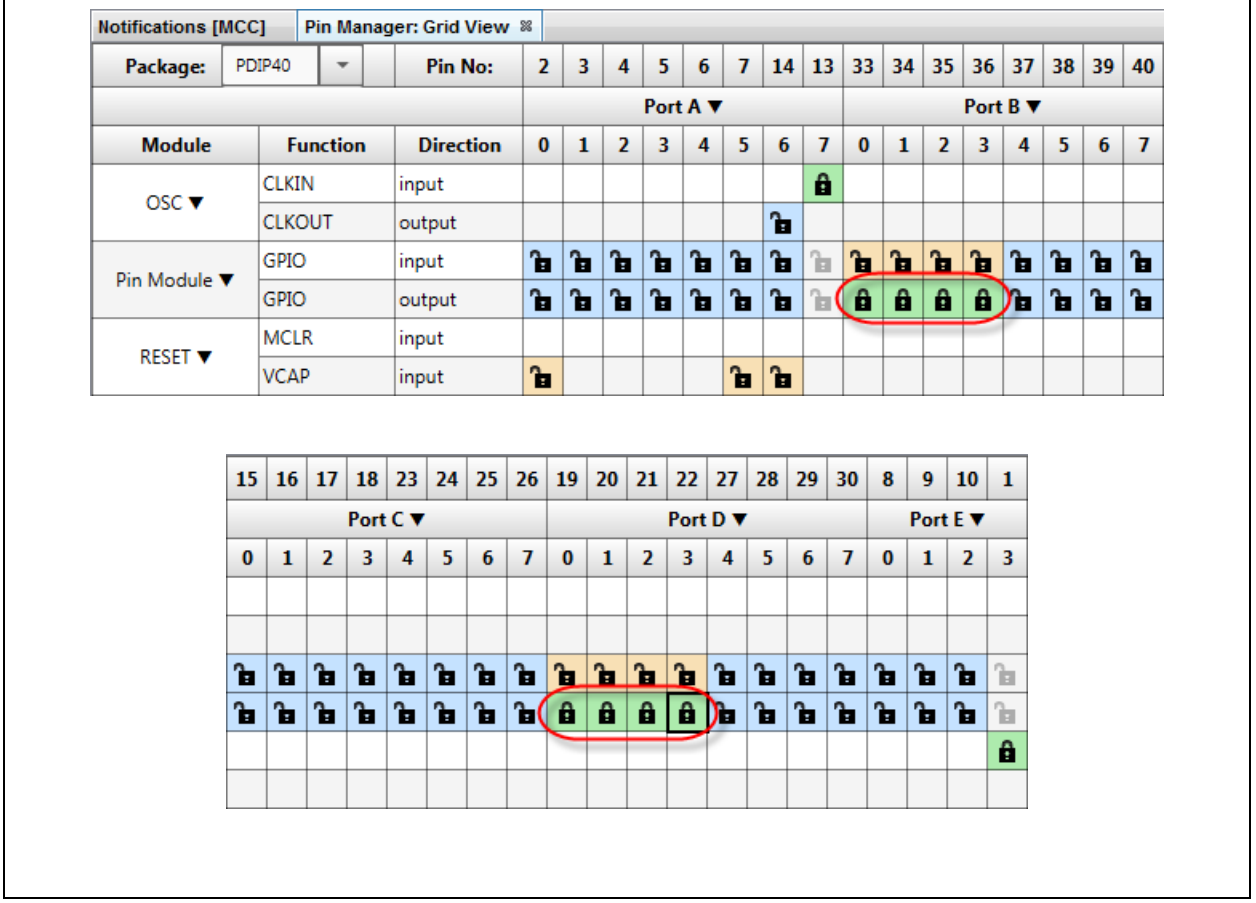
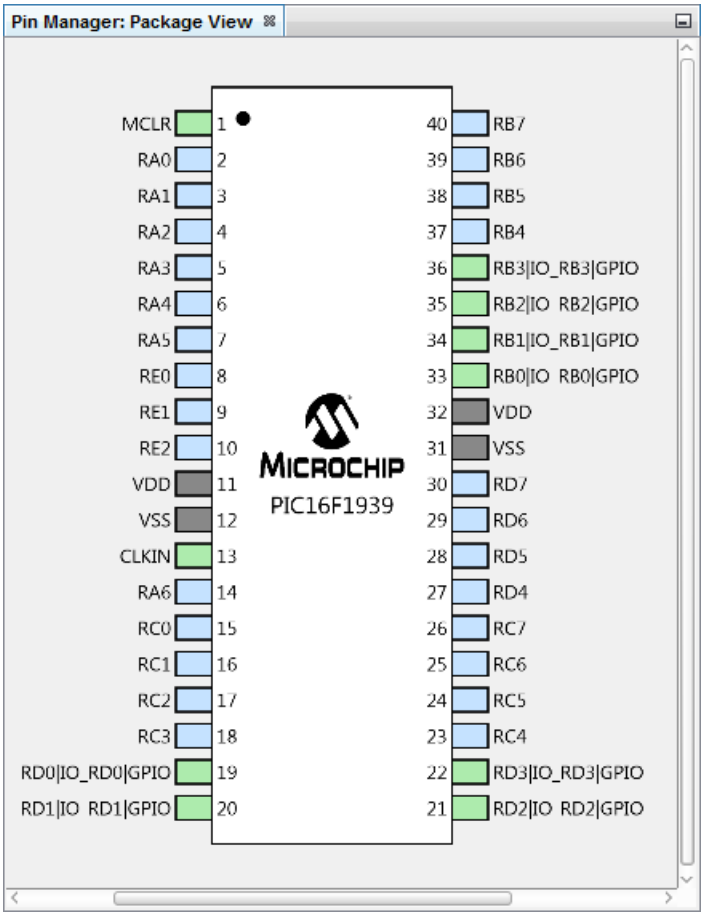


图14： EEDATA项目GPIO引脚资源——封装

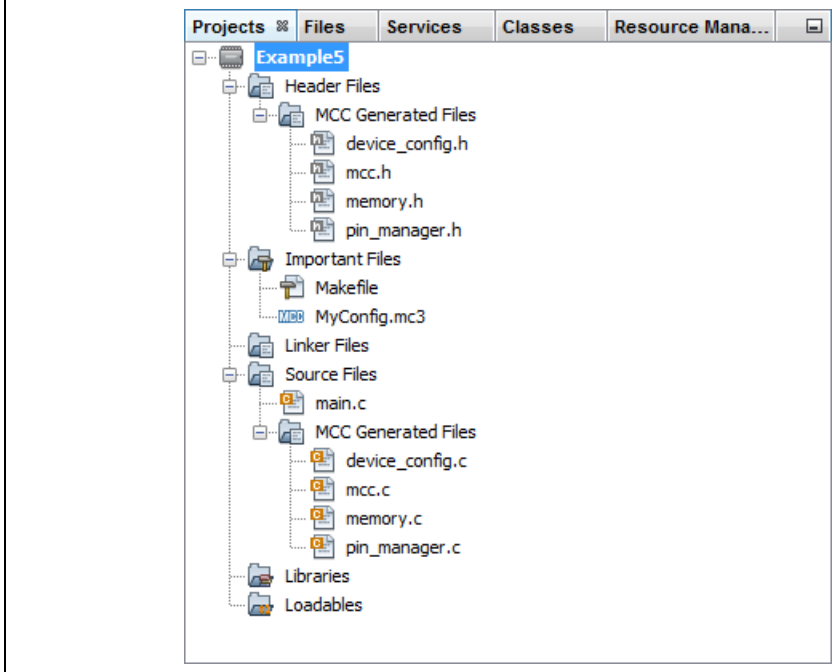


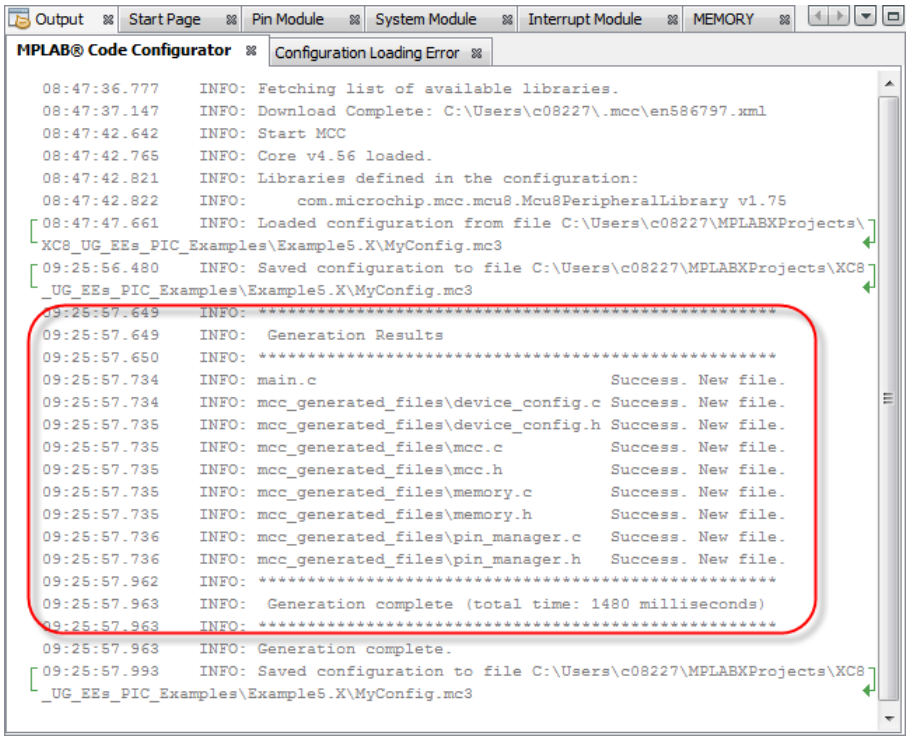
© 2016-2020 Microchip Technology Inc.

按照上述各图配置完代码后，请单击“Project Resources”（项目资源）窗口上的**Generate**（生成）按钮。通过MCC生成的代码是模块化的。因此，主程序代码、系统代码和外设代码均位于单独的文件中。此外，每个外设都有自己的头文件。

向程序中添加功能时始终需要编辑`main.c`。请查看生成的文件以找到您的代码中可能需要的任何函数或宏。

图15： 通过MCC生成的代码的EEDATA项目树





5.1 修改后的main.c代码

main.c模板文件已经过编辑，如下所示。部分注释已删除（< >内）。新添加的代码以红色显示。

```
/**
  Generated Main Source File

<See generated main.c file for file information.>
 */

/*
  (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use
  this software and any derivatives exclusively with Microchip
  products.

<See generated main.c file for additional copyright information.>
 */

#include "mcc_generated_files/mcc.h"

#define NUM_EE_VALUES 64
#define INSTR_CYCLE_DELAY 25000

/*
                         Main application
 */
void main(void) {
    // initialize the device
    SYSTEM_Initialize();

    // <No interrupts used - see generated main.c file for code.>

    // Declare RAM array, loop variable   <-- 请参见第5.2节

    volatile uint8_t RAMArray[NUM_EE_VALUES];
    uint8_t i;

    // Write initial values to EEPROM Data   <-- 请参见第5.3节
    PIR2bits.EEIF = 0x0; // clear write flag

    for(i=0; i<NUM_EE_VALUES; i++){
        DATAEE_WriteByte(_EEADRL_EEADRL_POSN + i, i);
        while(!PIR2bits.EEIF);  // check for write finished
        PIR2bits.EEIF = 0x0;
    }

    while(1){
        // Read from EEPROM and display   <-- 请参见第5.4节
        for(i=0; i<NUM_EE_VALUES; i++){
            RAMArray[i] = DATAEE_ReadByte(_EEADRL_EEADRL_POSN + i);
            LATD = RAMArray[i]; // RD[0:3] = LED[0:3]
            LATB = (RAMArray[i] >> 4); // RB[0:3] = LED[4:7]
            _delay(INSTR_CYCLE_DELAY); // delay value change
        }
```


© 2016-2020 Microchip Technology Inc.

```
        // Write to EEPROM in reverse order
        for(i=0; i<NUM_EE_VALUES; i++){
            DATAEE_WriteByte(_EEADRL_EEADRL_POSN +
              (NUM_EE_VALUES - 1) - i, RAMArray[i]);
            while(!PIR2bits.EEIF);  // check for write finished
            PIR2bits.EEIF = 0x0;
        }

    };

}
/**
 End of File
 */
```

5.2 EEData相关变量

用于存储EEData读/写数据的变量必须与读/写函数原型中指定的类型匹配，函数原型由mcc.h引用且可在memory.h中找到：

```
void DATAEE_WriteByte(uint8_t bAdd, uint8_t bData);
uint8_t DATAEE_ReadByte(uint8_t bAdd);
```

从stdint.h引用时，uint8_t与unsigned char相同。

5.3 写入EEData

本示例中两次写入EEData：第一次用于初始化EEData存储器中的值，第二次用于更改数据以实现动态显示。

写入EEData需要多个周期，因此使用写完成标志来确定写操作何时完成（PIR2bits.EEIF）。该标志最初为清零状态，并且会在每次写操作完成时再次清零。（该标志必须用软件清零。）

5.4 读取EEData

写入EEData后，存储器值会被读入RAM数组，然后显示在端口D和端口B LED上。该写循环中使用RAM数组中的值更改EEData存储器中的值。

由于执行速度在大多数情况下都会导致LED的闪烁速度超出人眼的识别能力，因此需要再次使用_delay()函数（同例2）来降低执行速度。

© 2016-2020 Microchip Technology Inc.

A. 在MPLAB X IDE中运行代码

A.1 创建项目：

1. 启动MPLAB X IDE。
2. 从IDE中启动新建项目向导（*File>New Project*（文件>新建项目））。
3. 按照屏幕提示创建一个新项目：
   a) **Choose Project（选择项目）：**选择“Microchip Embedded”（Microchip嵌入式），然后选择“Standalone Project”（独立项目）。
   b) **Select Device（选择器件）：**选择示例器件。
   c) **Select Header（选择调试头）：**无。
   d) **Select Tool（选择工具）：**选择硬件调试工具，SNxxxxxx。如果调试工具名称下未显示序列号（Serial Number，SN），请确保调试工具已正确安装。有关详细信息，请参见调试工具文档。
   e) **Select Plugin Board（选择接插板）：**无。
   f) **Select Compiler（选择编译器）：**选择XC8（*最新版本号*）[`bin`文件夹位置]。如果XC8下未显示编译器，请确保编译器已正确安装且MPLAB X IDE已获知（*Tools>Options*（工具>选项），**Embedded**（已安装工具）按钮，**Build Tools**（编译工具）选项卡）。有关详细信息，请参见MPLAB XC8和MPLAB X IDE文档。
   g) **Select Project Name and Folder（选择项目名和文件夹）：**为项目命名。

A.2 选择通用 C 接口 （CCI）

创建完项目后，右键单击Projects（项目）窗口中的项目名称并选择Properties（属性）。在对话框中，单击“XC8 Compiler”（XC8编译器）类别，选择“Preprocessing and messages”（预处理和消息）选项类别，并选中“Use CCI syntax”（使用CCI语法）。单击**OK**（确定）按钮。

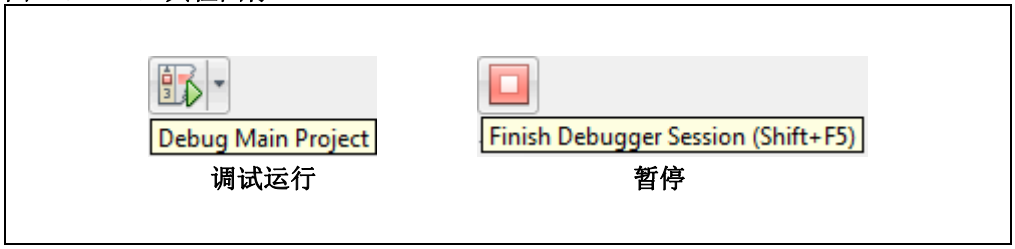
A.3 调试示例

根据您使用的示例执行以下操作之一：

1. 对于示例1、2和3，创建一个文件来保存示例代码：
   a) 右键单击Projects窗口中的“Source Files”文件夹。选择*New>main.c*（新建>main.c）。将打开“New main.c”（新建main.c）对话框。
   b) 在“File name”（文件名）下输入名称（如example*n*），*其中n*为示例编号。
   c) 单击**Finish**（完成）。将在编辑器窗口中打开文件。
   d) 删除文件中的模板代码。然后从本用户指南中剪切示例代码并粘贴到空白编辑器窗口中，并选择*File>Save*（文件>保存）。
2. 对于示例4和5，请按照每节中的说明使用MCC生成代码，然后用所显示的代码编辑`main.c`文件。

最后，选择Debug Run（调试运行）编译并下载代码到器件中，并执行代码。通过观察演示板上LED的状态查看输出。单击Halt（暂停）结束执行。

图16： 工具栏图标



调试运行　　暂停

© 2016-2020 Microchip Technology Inc.

B. 获取软件和硬件

对于本文档中的MPLAB XC8项目，装有PIC16F1719或PIC16F1939 MCU的Explorer 8板由9V外部电源供电，并使用标准（ICSP™）通信。使用MPLAB X IDE进行开发。

B.1 获取MPLAB X IDE和MPLAB XC8 C编译器

可从以下网址找到MPLAB X IDE v5.10及以上版本：
https://www.microchip.com/mplab/mplab-x-ide

可从以下网址找到MPLAB XC8 C编译器v2.00及以上版本：
https://www.microchip.com/mplab/compilers

B.2 获取MPLAB代码配置器（MCC）

可从以下网址找到MCC v3.66及以上版本：
https://www.microchip.com/mplab/mplab-code-configurator

B.3 获取PIC® MCU

可从以下网址找到本示例中使用的PIC MCU：
https://www.microchip.com/PIC16F1719
https://www.microchip.com/PIC16F1939

B.4 获取并设置Explorer 8板

可从以下网址找到Explorer 8开发工具包（DM160228）：
https://www.microchip.com/DM160228

下表列出了跳线设置。

表1-1： 项目的跳线选择

跳线	选择	说明
J2	BRD+5V	通过电源（而非USB）为开发板供电
J14	+5V	器件电源电压
J24	开路	使用+5V（而非3.3V）
J7	闭合	使能端口D <RD0:3>上的LED
J21	闭合	使能端口B <RB0:3>上的LED
J36	OSC1至RA7	OSC1 CLKIN（8 MHz外部振荡器）
J37	OSC2至RA6	OSC2 CLKOUT（8 MHz外部振荡器）
J51	PGD至RB7	ICSPDAT
J52	PGC至RB6	ISCPCLK

表1-2： 未使用的跳线选择

跳线	选择	说明
JP2	闭合	不使用LCD
J22、J23、J53和J54	开路	不使用LCD
J15和J16	开路	不使用Digilent Pmod™连接器
J43、J44、J45、J46和J47	开路	不使用mikroBUS
J41、J42、J48、J49和J50	开路	不使用mikroBUS
J4和J31	VCAP	不使用RA5和RA4

B.5 获取Microchip调试工具

可在如下开发工具网页中找到仿真器和调试器：
https://www.microchip.com/development-tools

© 2016-2020 Microchip Technology Inc.

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。

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，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。

AMBA、Arm、Arm7、Arm7TDMI、Arm9、Arm11、Artisan、big.LITTLE、Cordio、CoreLink、CoreSight、Cortex、DesignStart、DynamIQ、Jazelle、Keil、Mali、Mbed、Mbed Enabled、NEON、POP、RealView、SecurCore、Socrates、Thumb、TrustZone、ULINK、ULINK2、ULINK-ME、ULINK-PLUS、ULINKpro、µVision 和 Versatile 是 Arm Limited（或其子公司）在美国和 / 或其他国家 / 地区的商标或注册商标。

有关 Microchip 质量管理体系的更多信息，请访问 www.microchip.com/quality。

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。

© 2016-2020, Microchip Technology Incorporated 版权所有。

ISBN：978-1-5224-5681-0

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

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
Tel: 1-317-536-2380

洛杉矶 Los Angeles
Mission Viejo, CA
Tel: 1-949-462-9523
Fax: 1-949-462-9608
Tel: 1-951-273-7800

罗利 Raleigh, NC
Tel: 1-919-844-7510

纽约 New York, NY
Tel: 1-631-435-6000

圣何塞 San Jose, CA
Tel: 1-408-735-9110
Tel: 1-408-436-4270

加拿大多伦多 Toronto
Tel: 1-905-695-1980
Fax: 1-905-695-2078

亚太地区

中国 - 北京
Tel: 86-10-8569-7000

中国 - 成都
Tel: 86-28-8665-5511

中国 - 重庆
Tel: 86-23-8980-9588

中国 - 东莞
Tel: 86-769-8702-9880

中国 - 广州
Tel: 86-20-8755-8029

中国 - 杭州
Tel: 86-571-8792-8115

中国 - 南京
Tel: 86-25-8473-2460

中国 - 青岛
Tel: 86-532-8502-7355

中国 - 上海
Tel: 86-21-3326-8000

中国 - 沈阳
Tel: 86-24-2334-2829

中国 - 深圳
Tel: 86-755-8864-2200

中国 - 苏州
Tel: 86-186-6233-1526

中国 - 武汉
Tel: 86-27-5980-5300

中国 - 西安
Tel: 86-29-8833-7252

中国 - 厦门
Tel: 86-592-238-8138

中国 - 香港特别行政区
Tel: 852-2943-5100

中国 - 珠海
Tel: 86-756-321-0040

台湾地区 - 高雄
Tel: 886-7-213-7830

台湾地区 - 台北
Tel: 886-2-2508-8600

台湾地区 - 新竹
Tel: 886-3-577-8366

亚太地区

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

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

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

印度 India - Pune
Tel: 91-20-4121-0141

日本 Japan - Osaka
Tel: 81-6-6152-7160

日本 Japan - Tokyo
Tel: 81-3-6880-3770

韩国 Korea - Daegu
Tel: 82-53-744-4301

韩国 Korea - Seoul
Tel: 82-2-554-7200

马来西亚 Malaysia - Kuala Lumpur
Tel: 60-3-7651-7906

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

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

新加坡 Singapore
Tel: 65-6334-8870

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

越南 Vietnam - Ho Chi Minh
Tel: 84-28-5448-2100

欧洲

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

丹麦 Denmark - Copenhagen
Tel: 45-4485-5910
Fax: 45-4485-2829

芬兰 Finland - Espoo
Tel: 358-9-4520-820

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

德国 Germany - Garching
Tel: 49-8931-9700

德国 Germany - Haan
Tel: 49-2129-3766400

德国 Germany - Heilbronn
Tel: 49-7131-72400

德国 Germany - Karlsruhe
Tel: 49-721-625370

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

德国 Germany - Rosenheim
Tel: 49-8031-354-560

以色列 Israel - Ra'anana
Tel: 972-9-744-7705

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

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

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

挪威 Norway - Trondheim
Tel: 47-7288-4388

波兰 Poland - Warsaw
Tel: 48-22-3325737

罗马尼亚 Romania - Bucharest
Tel: 40-21-407-87-50

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

瑞典 Sweden - Gothenberg
Tel: 46-31-704-60-40

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

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

© 2016-2020 Microchip Technology Inc.