面向嵌入式工程师的MPLAB® XC16用户指南

简介

本文档介绍了5个适用于16位器件和MPLAB® XC16 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® XC16 C编译器用户指南》（DS50002071E_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. 使用延时函数使LED闪烁
3. 使用中断作为延时在LED上递增计数
4. 使用ADC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

1. 点亮或熄灭LED

本示例将使用PIC24FJ128GA010接插模块（Plug-In Module，PIM）交替点亮Explorer 16/32开发板上的LED。更多信息，请参见第B节“获取软件和硬件”。

```
// PIC24FJ128GA010 Configuration Bit Settings

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

// CONFIG2
#pragma config POSCMOD = XT    // XT Oscillator mode selected
#pragma config OSCIOFNC = ON   // OSC2/CLKO/RC15 as port I/O (RC15)
#pragma config FCKSM = CSDCMD // Clock Switching and Monitor disabled
#pragma config FNOSC = PRI     // Primary Oscillator (XT, HS, EC)
#pragma config IESO = ON       // Int Ext Switch Over Mode enabled

// CONFIG1
#pragma config WDTPS = PS32768 // Watchdog Timer Postscaler (1:32,768)
#pragma config FWPSA = PR128   // WDT Prescaler (1:128)
#pragma config WINDIS = ON     // Watchdog Timer Window Mode disabled
#pragma config FWDTEN = OFF    // Watchdog Timer disabled
#pragma config ICS = PGx2      // Emulator/debugger uses EMUC2/EMUD2
#pragma config GWRP = OFF      // Writes to program memory allowed
#pragma config GCP = OFF       // Code protection is disabled
#pragma config JTAGEN = OFF    // JTAG port is 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

int main(void) {

    unsigned char portValue = 0x55;    <-- 请参见第1.3节

    // Port A access    <-- 请参见第1.4节

    AD1PCFG = 0xFFFF;    // set to digital I/O (not analog)
    TRISA = 0x0000;      // set all port bits to be output
    LATA = portValue;    // write to port latch

    return 0;
}
```

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。

1.1.1 要设置的配置位

请特别注意以下几项：

- **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 FNOSC = PRI
#pragma config POSCMOD = XT
```

- **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 FWDTEN = 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 GCP = OFF
```

使用其他16位器件（而非本示例中使用的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。

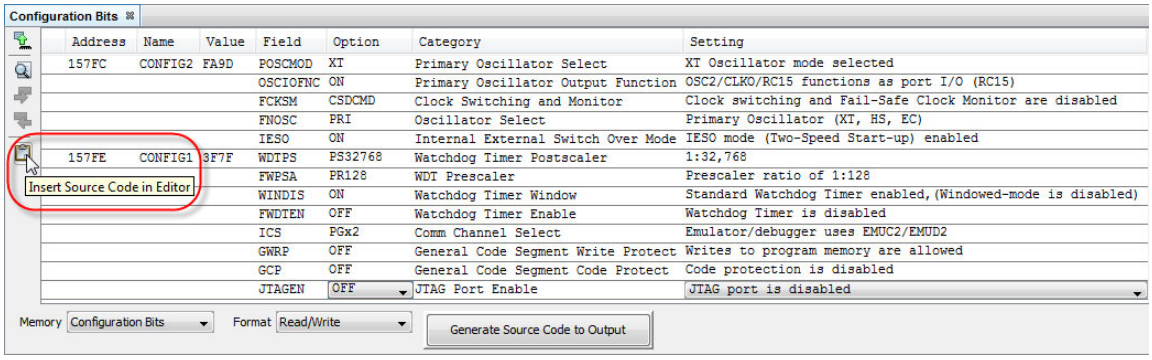
有关每款器件可用配置位的更多信息，请参见MPLAB XC16安装路径下的如下文件：

MPLAB XC16 Installation Directory/docs/config_index.html

1.1.2 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>PIC Memory Views>Configuration Bits*（窗口>PIC存储器视图>配置位）打开该窗口。

图1： Configuration Bits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 头文件<xc.h>

此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。可在MPLAB XC16安装目录的support子目录下找到此头文件和其他头文件。

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。

1.3 用于LED值的变量

如下一节所述，要写入LED的值已赋值给一个变量（portValue），即LED D3、D5、D7和D9将点亮，LED D4、D6、D8和D10将熄灭。有关演示板原理图位置的信息，请参见第B节“获取软件和硬件”。

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A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：

```
AD1PCFG = 0xFFFF;  // 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。示例中使用LATA。宏LEDS_ON_OFF赋值给锁存器：

```
LATA = LEDS_ON_OFF;   // write to port latch
```

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TRISD和TRISB。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。对于本示例：

```
TRISA = 0x0000;   // set all port bits to be output
```

2. 使用延时函数使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。

```
// PIC24FJ128GA010 Configuration Bit Settings

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

// CONFIG2
#pragma config POSCMOD = XT   // XT Oscillator mode selected
#pragma config OSCIOFNC = ON  // OSC2/CLKO/RC15 as port I/O (RC15)
#pragma config FCKSM = CSDCMD // Clock Switching and Monitor disabled
#pragma config FNOSC = PRI    // Primary Oscillator (XT, HS, EC)
#pragma config IESO = ON      // Int Ext Switch Over Mode enabled

// CONFIG1
#pragma config WDTPS = PS32768 // Watchdog Timer Postscaler (1:32,768)
#pragma config FWPSA = PR128   // WDT Prescaler (1:128)
#pragma config WINDIS = ON     // Watchdog Timer Window Mode disabled
#pragma config FWDTEN = OFF    // Watchdog Timer disabled
#pragma config ICS = PGx2      // Emulator/debugger uses EMUC2/EMUD2
#pragma config GWRP = OFF      // Writes to program memory allowed
#pragma config GCP = OFF       // Code protection is disabled
#pragma config JTAGEN = OFF    // JTAG port is disabled

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

#include <xc.h>
#include <libpic30.h>     <-- 请参见第2.1节

int main(void) {

    unsigned char portValue;

    // Port A access
    AD1PCFG = 0xFFFF; // set to digital I/O (not analog)
    TRISA = 0x0000;   // set all port bits to be output

    while(1) {    <-- 请参见第2.2节

        portValue = 0x55;
        LATA = portValue; // write to port latch

        // delay value change    <-- 请参见第2.3节

        __delay32(1500000);    // delay in instruction cycles

        portValue = 0xAA;
        LATA = portValue; // write to port latch

        __delay32(1500000);    // delay in instruction cycles

    }
    return -1;
}
```

2.1 库头文件

在本例中，使用libpic30编译器库中的__delay32()函数。要访问该库，必须包含libpic30.h。

2.2 while()循环和变量值

要使端口A上的LED发生变化，首先为变量portValue赋值0x55（LED 0, 2, 4, 6点亮），然后为该变量赋互补值0xAA （LED 1,3,5,7 点亮）。使用了while(1)执行循环。

如果主函数返回，则意味着存在错误，因为while循环通常不应结束。因此，将返回-1指示错误。

2.3 __delay32()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。__delay32()是编译器可以使用的库函数。

有关延时函数的更多详细信息，请参见参考手册《16位语言工具函数库》（DS51456D_CN）。

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函数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。

```
// PIC24FJ128GA010 Configuration Bit Settings

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

// CONFIG2
#pragma config POSCMOD = XT   // XT Oscillator mode selected
#pragma config OSCIOFNC = ON  // OSC2/CLKO/RC15 as port I/O (RC15)
#pragma config FCKSM = CSDCMD // Clock Switching and Monitor disabled
#pragma config FNOSC = PRI    // Primary Oscillator (XT, HS, EC)
#pragma config IESO = ON      // Int Ext Switch Over Mode enabled

// CONFIG1
#pragma config WDTPS = PS32768 // Watchdog Timer Postscaler (1:32,768)
#pragma config FWPSA = PR128   // WDT Prescaler (1:128)
#pragma config WINDIS = ON     // Watchdog Timer Window Mode disabled
#pragma config FWDTEN = OFF    // Watchdog Timer disabled
#pragma config ICS = PGx2      // Emulator/debugger uses EMUC2/EMUD2
#pragma config GWRP = OFF      // Writes to program memory allowed
#pragma config GCP = OFF       // Code protection is disabled
#pragma config JTAGEN = OFF    // JTAG port is disabled

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

#include <xc.h>

// Interrupt function (CCI)     ◄—— 请参见第3.1节

void __interrupt(no_auto_psv) _T1Interrupt(void){
    // static variable for permanent storage duration
    static unsigned char portValue = 0;
    // write to port latch
    LATA = portValue++;
    // clear this interrupt condition
    _T1IF = 0;
}

int main(void) {

    // Port A access
    AD1PCFG = 0xFFFF; // set to digital I/O (not analog)
    TRISA = 0x0000;   // set all port bits to be output

    // Timer1 setup     ◄—— 请参见第3.2节

    T1CON = 0x8010; // timer 1 on, prescaler 1:8, internal clock
    _T1IE = 1; // enable interrupts for timer 1
    _T1IP = 0x001; // set interrupt priority (lowest)

    while(1);

    return -1;
}
```

3.1 中断函数

可使用通用C接口（CCI）的__interrupt()说明符使函数成为中断函数。此外，还应指定程序空间可见性（Program Space Visibility，PSV）。本示例比较简单，不使用任何PSV。有关PSV的更多信息，请参见《MPLAB® XC16 C编译器用户指南》（DS50002071E_CN）。

使用特定于Timer1的主中断向量_T1Interrupt()。编译器安装目录的docs文件夹下提供了每个器件的中断向量表。

在中断函数中，计数器portValue会在Timer1产生中断时递增。

3.2 Timer1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。

4. 使用ADC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端口A LED。但在本示例中，将使用演示板上电位器（滑块）的值通过端口B提供模数转换器（Analog-to-Digital Converter，ADC）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的安装信息以及《MPLAB®代码配置器用户指南》（DS40001725B_CN），请访问以下URL中的MPLAB代码配置器网页：

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

本示例中的MCC设置如以下各图所示。

图2：　ADC项目资源——系统模块

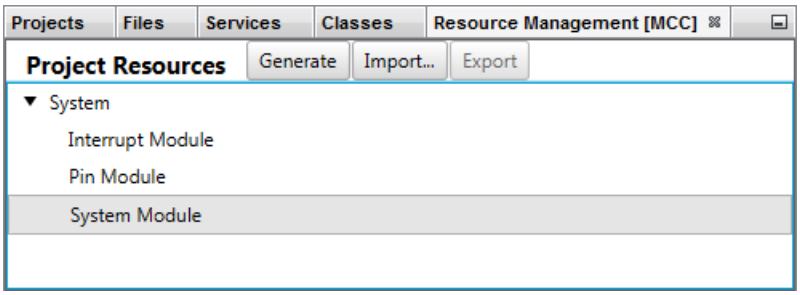


图3： ADC项目系统模块配置

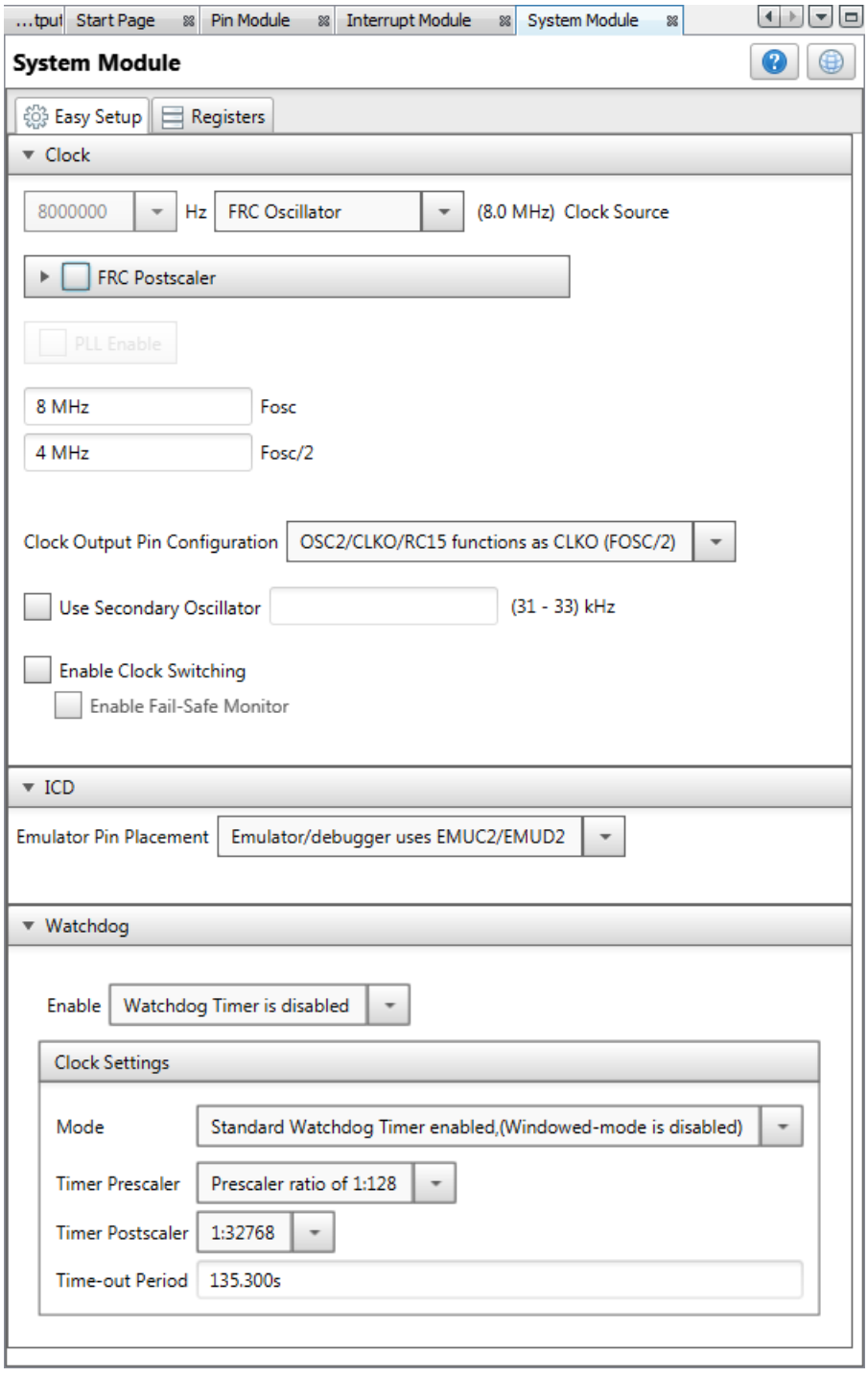
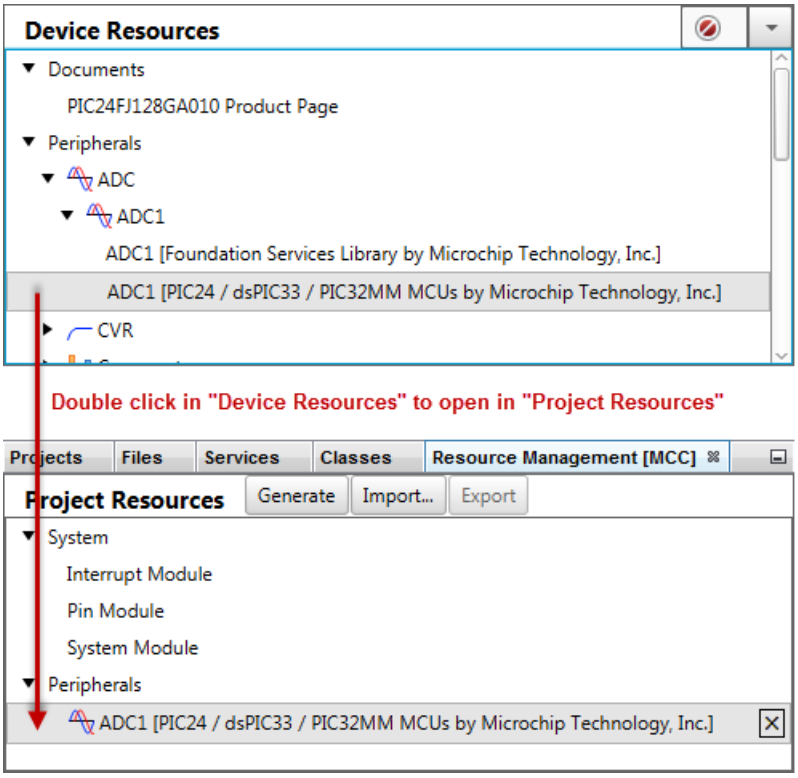


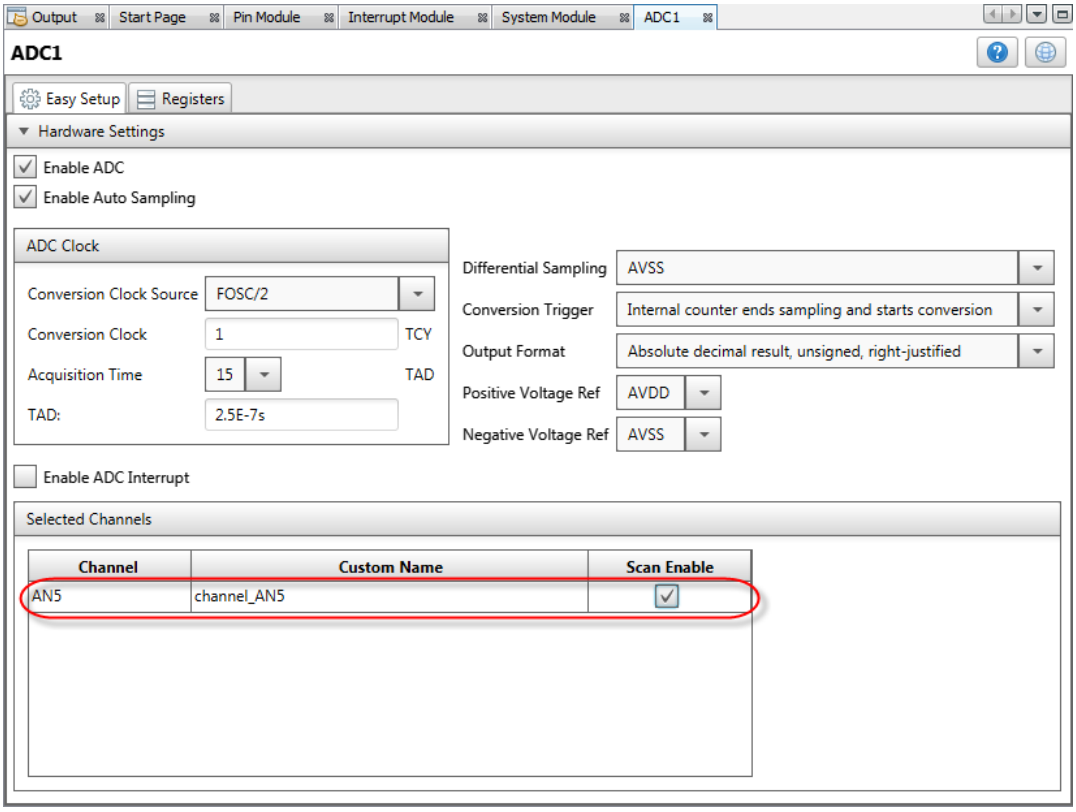
图4：　　ADC项目ADC1选择



PIC24/dsPIC33/PIC32MM 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5： ADC项目ADC1配置



RB5至AN5映射将在图6中的选择完成后显示。

图6： ADC项目ADC1引脚资源

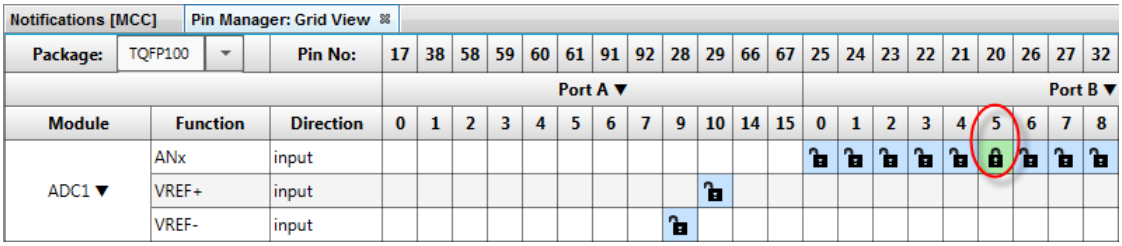


图7：　　ADC项目资源——引脚模块

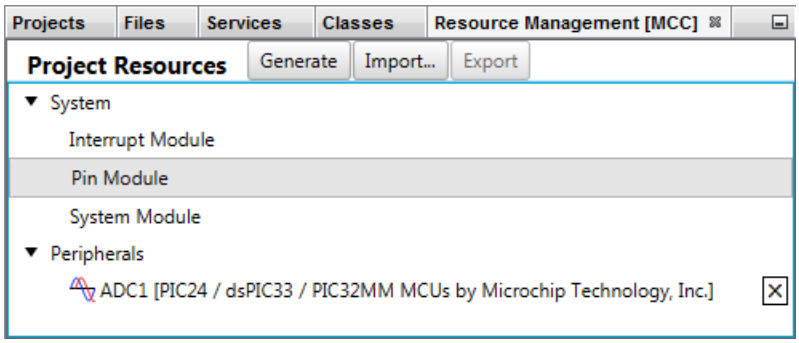
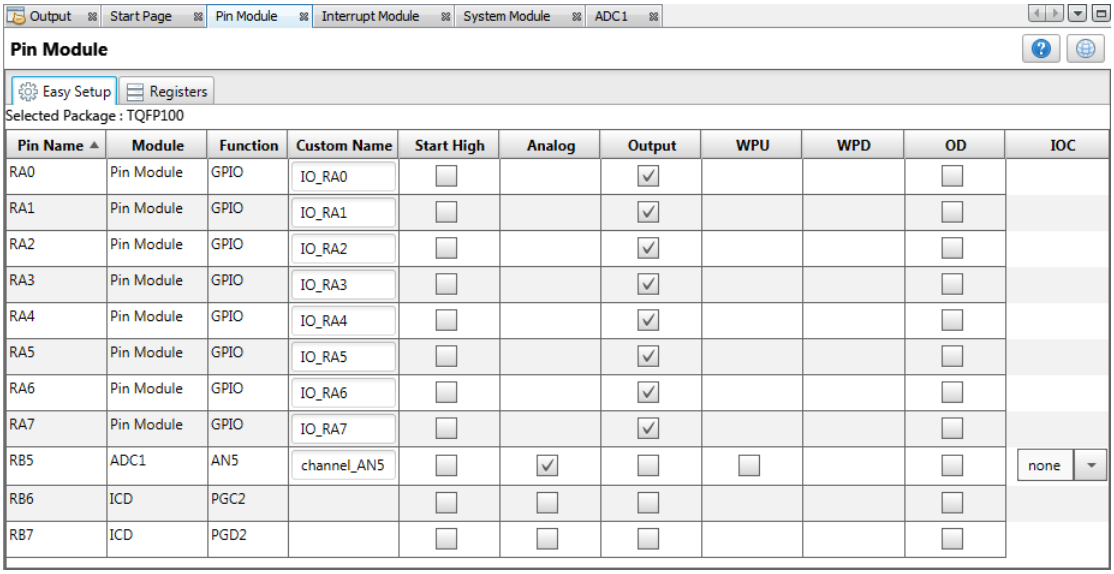


图8：　　ADC项目I/O引脚配置



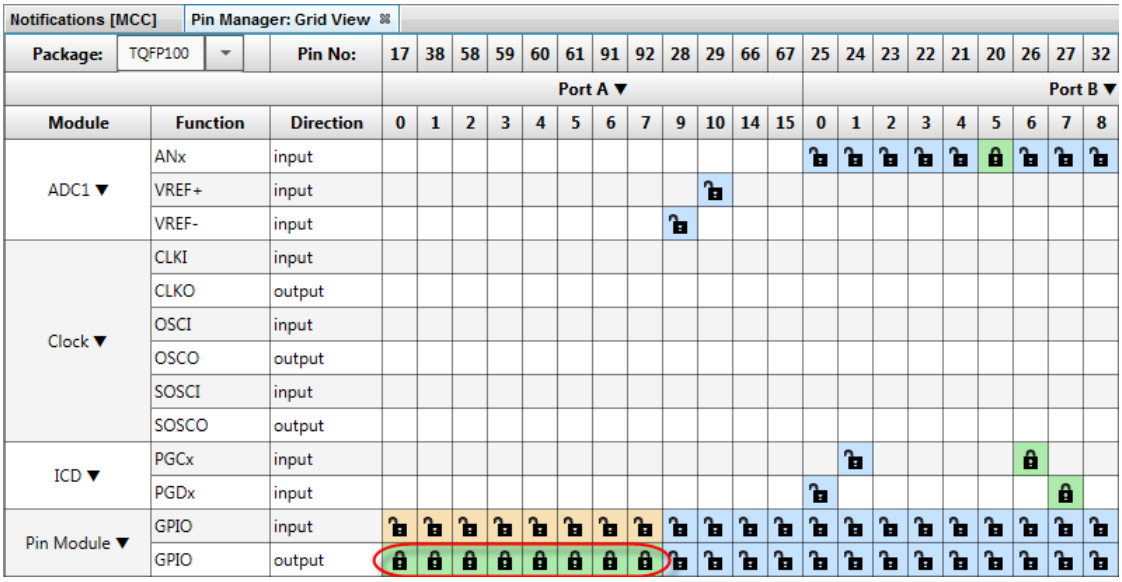
在图9中选择引脚RA0:7后，它们将出现在上面的窗口中。

前面在图6中选择了RB5。

预先选择RB6和RB7用于调试通信。

一旦引脚出现在窗口中，即可为每个引脚查看或选择引脚配置。

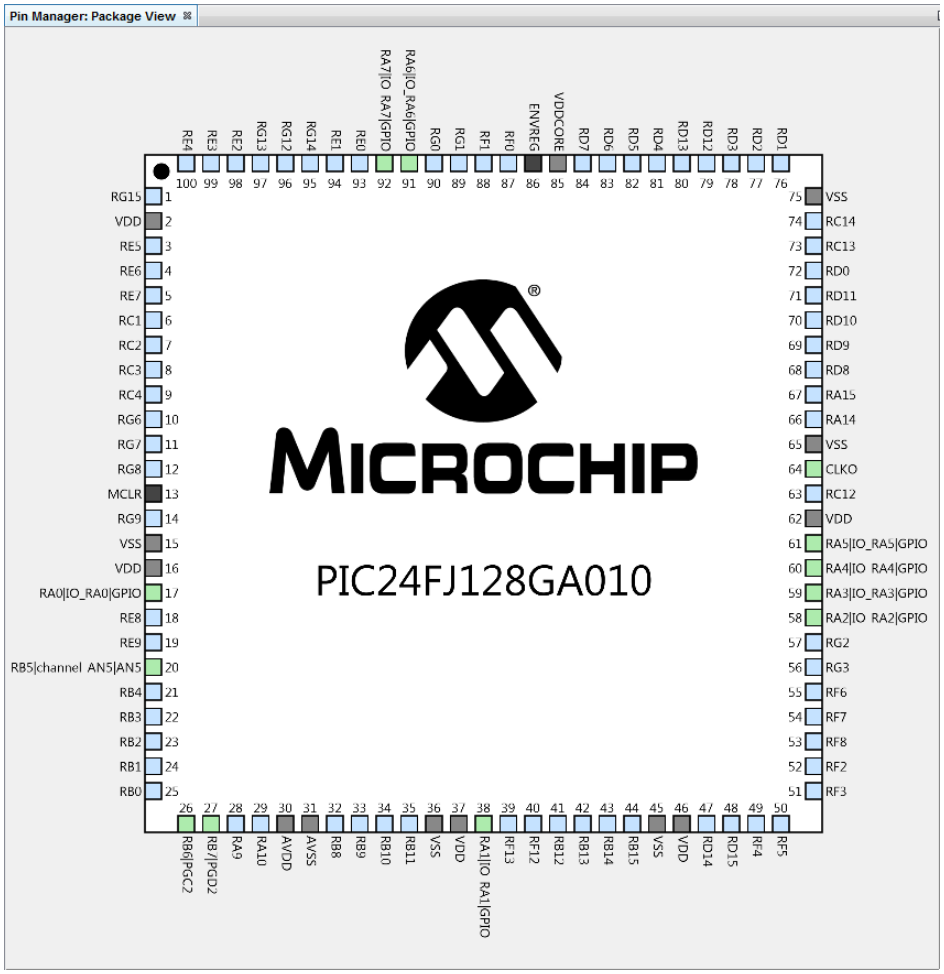
图9： ADC项目I/O引脚资源



Notifications [MCC] | Pin Manager: Grid View

Package: TQFP100	Pin No:	17	38	58	59	60	61	91	92	28	29	66	67	25	24	23	22	21	20	26	27	32
		Port A ▼												Port B ▼								
Module	**Function**	**Direction**	0	1	2	3	4	5	6	7	9	10	14	15	0	1	2	3	4	5	6	7	8
ADC1 ▼	ANx	input																					
	VREF+	input																					
	VREF-	input																					
Clock ▼	CLKI	input																					
	CLKO	output																					
	OSCI	input																					
	OSCO	output																					
	SOSCI	input																					
	SOSCO	output																					
ICD ▼	PGCx	input																					
	PGDx	input																					
Pin Module ▼	GPIO	input																					
	GPIO	output																					

图10： ADC项目引脚封装



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（图7）。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。

可通过生成陷阱文件来捕捉潜在的错误。尽管本应用程序中不会使用中断，但会生成中断管理器文件供将来使用。

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11：　通过MCC生成的代码的ADC项目树

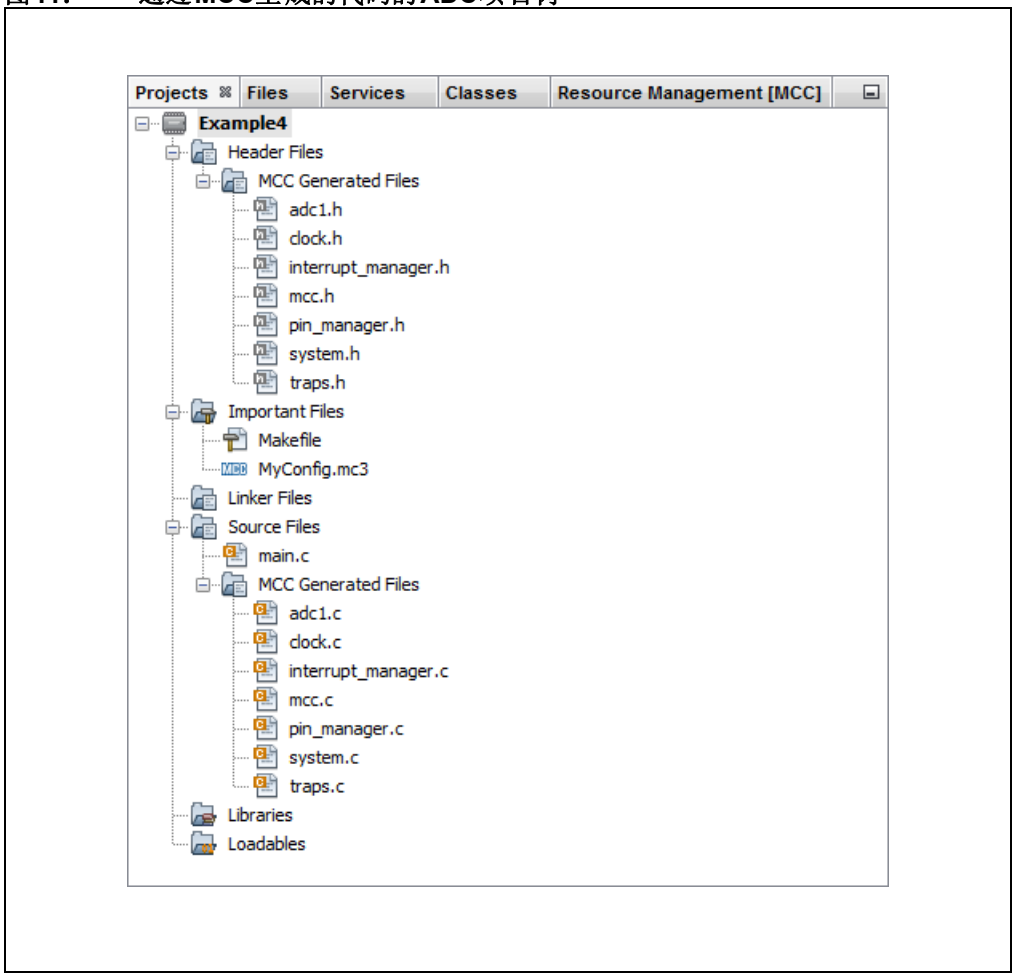
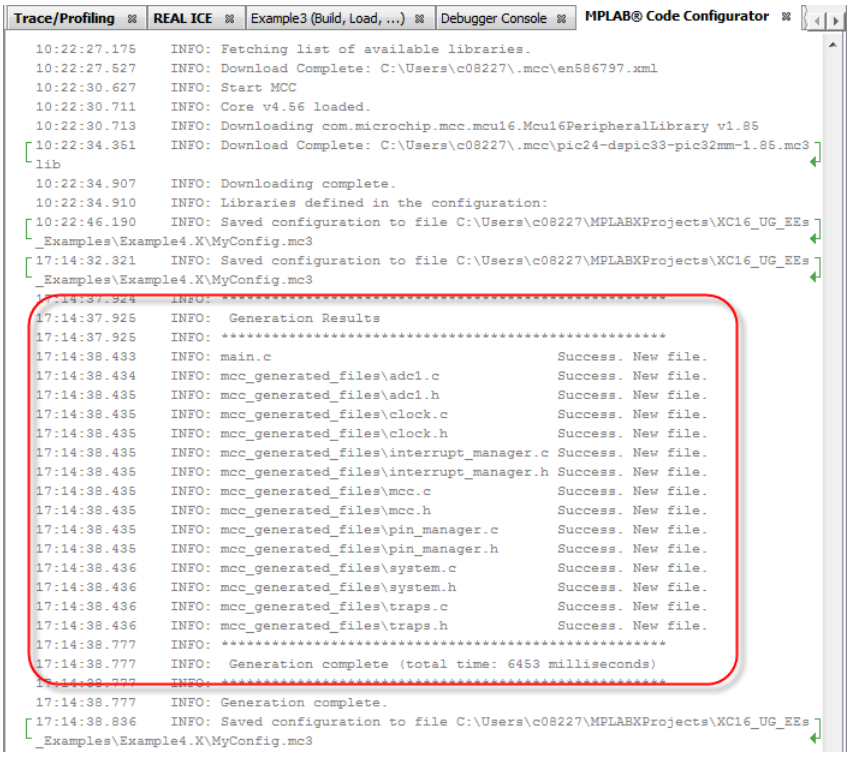


图12：　　通过MCC生成的ADC代码



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()中的代码以绿色显示。

```
/**
  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"

unsigned int value = 0;

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

    while (1) {

        // Wait for conversion      <-- 请参见第4.2节
        // and then get result
        while(!ADC1_IsConversionComplete());
        value = ADC1_ConversionResultGet();

        // Shift for MSb
        value = value >> 2;

        // Write to Port Latch/LEDs      <-- 请参见第4.3节
        LATA = value;

    }
    return -1;
}
/**
 End of File
 */
```

4.2 ADC转换及结果

MCC通过设置AD1CON1寄存器中的各位来使能ADC、使用自动采样采集以及使用内部计数器结束采样并启动转换。因此，main()代码只需要等待转换结束并获取结果。

使用adc1.c模块中的如下函数：

```
bool ADC1_IsConversionComplete(void)
uint16_t ADC1_ConversionResultGet(void)
```

有关设置其他ADC功能的信息，请参见《PIC24F系列参考手册》的“第17章 10位模数转换器（ADC）”（DS39705A_CN）。

由于只有8个LED可用，而ADC转换结果为10位，因此变量value中的转换结果将发生移位以显示最高8位。分辨率将有所降低。

4.3 写入端口锁存器和LED

ADC转换结果value将显示在端口A的LED上。

5. 在LED上显示EEPROM数据值

本示例使用Microchip的PIC24F32KA304 PIM与Explorer 16/32板搭配，演示如何读写EEPROM数据（EEData）。读取的值显示在从三个端口访问的LED上。

使用MPLAB代码配置器（MCC）来生成部分代码。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ADC在LED上显示电位器值”。

本示例中使用MCC GUI来设置系统（振荡器速度和配置位等）以及端口A、B和C的通用I/O（General Purpose I/O，GPIO）（图13）。但是，MCC中目前没有用于16位器件的EEData器件资源。

使用EEData模块所需的代码在器件数据手册和《dsPIC33/PIC24系列参考手册》的“第5章 数据EEPROM”中提供，二者均位于以下器件网页中：

https://www.microchip.com/PIC24F32KA304

图13： EEDATA项目资源——系统模块

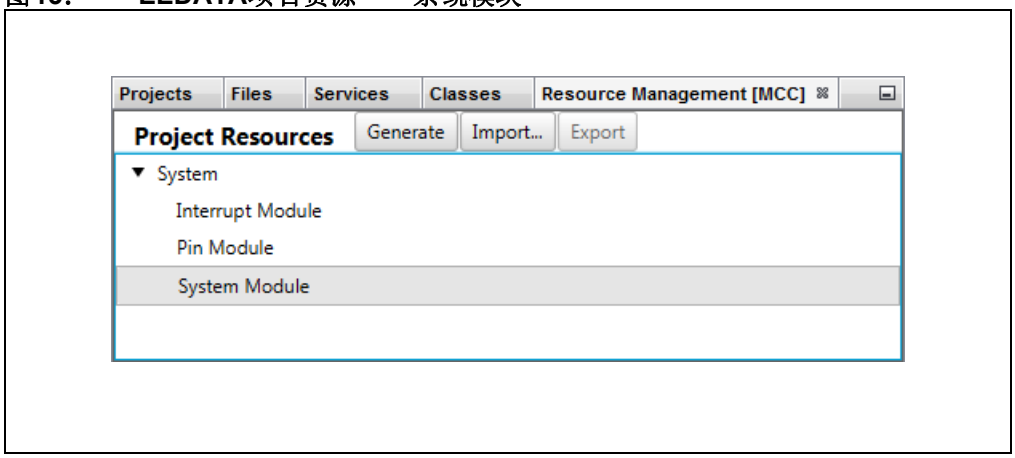


图14： EEDATA项目系统模块配置

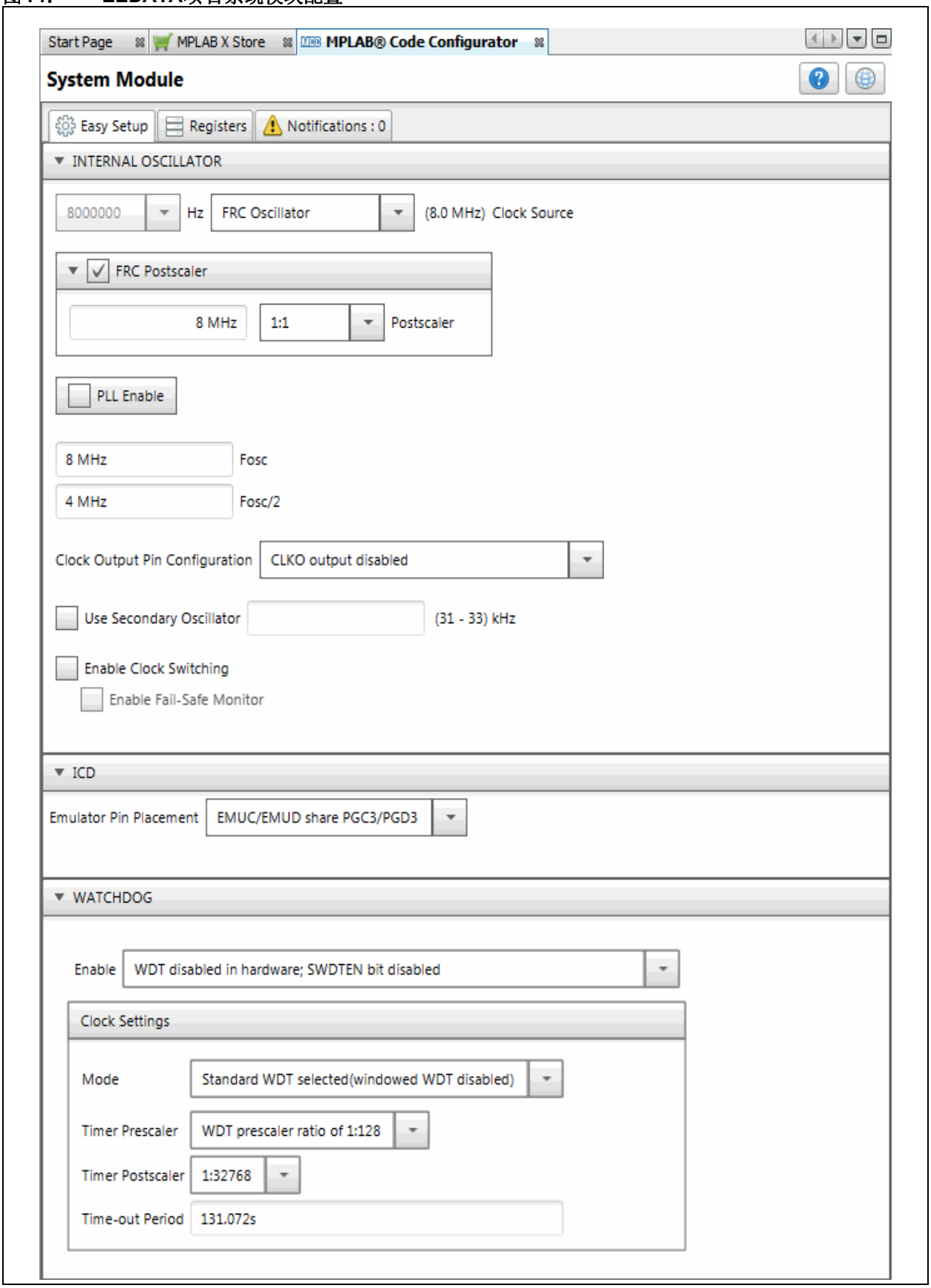


图15： EEDATA项目资源——引脚模块

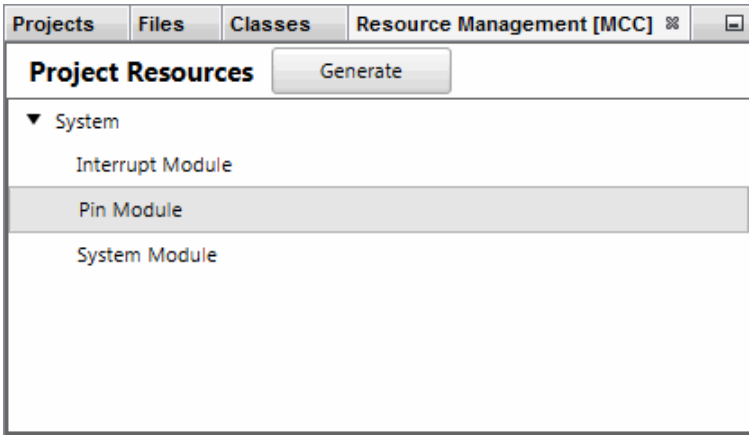
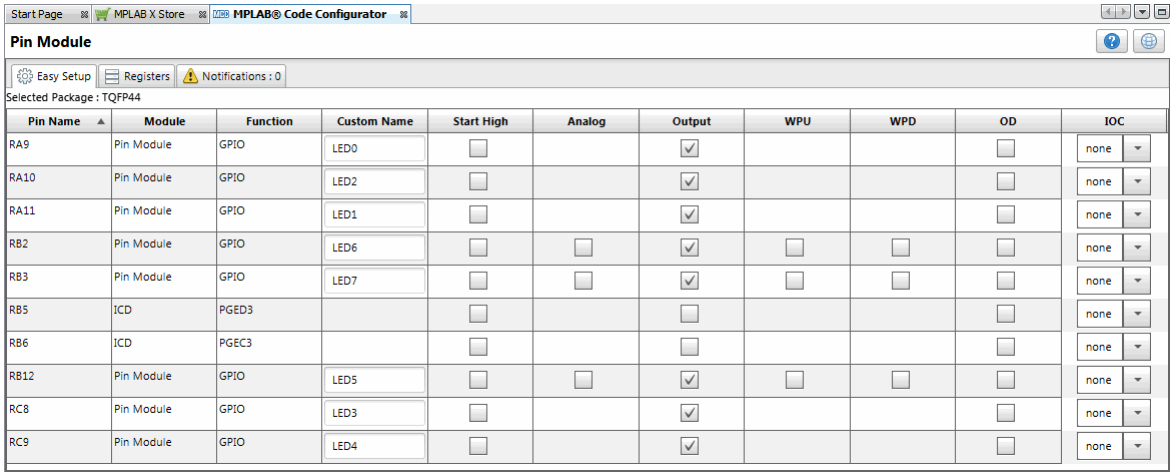


图16： EEDATA项目I/O引脚配置



Pin Name	Module	Function	Custom Name	Start High	Analog	Output	WPU	WPD	OD	IOC
RA9	Pin Module	GPIO	LED0	☐		☑			☐	none
RA10	Pin Module	GPIO	LED2	☐		☑			☐	none
RA11	Pin Module	GPIO	LED1	☐		☑			☐	none
RB2	Pin Module	GPIO	LED6	☐	☐	☑	☐	☐	☐	none
RB3	Pin Module	GPIO	LED7	☐	☐	☑	☐	☐	☐	none
RB5	ICD	PGED3		☐		☐			☐	none
RB6	ICD	PGEC3		☐		☐			☐	none
RB12	Pin Module	GPIO	LED5	☐	☐	☑	☐	☐	☐	none
RC8	Pin Module	GPIO	LED3	☐		☑			☐	none
RC9	Pin Module	GPIO	LED4	☐		☑			☐	none

在图17中选择引脚RA9:11、RB2:3、RB12和RC8:9后，它们将出现在上面的窗口中。
预先选择RB6和RB7用于调试通信。
一旦引脚出现在窗口中，即可为每个引脚查看或选择引脚配置。

图17： EEDATA项目I/O引脚资源

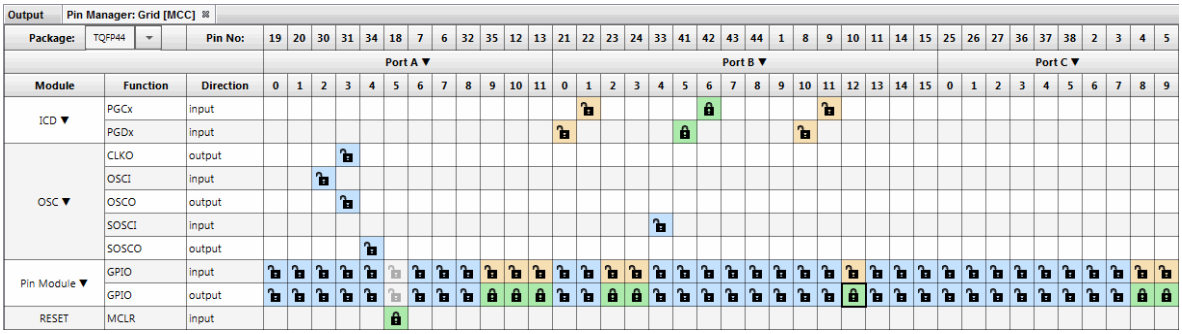
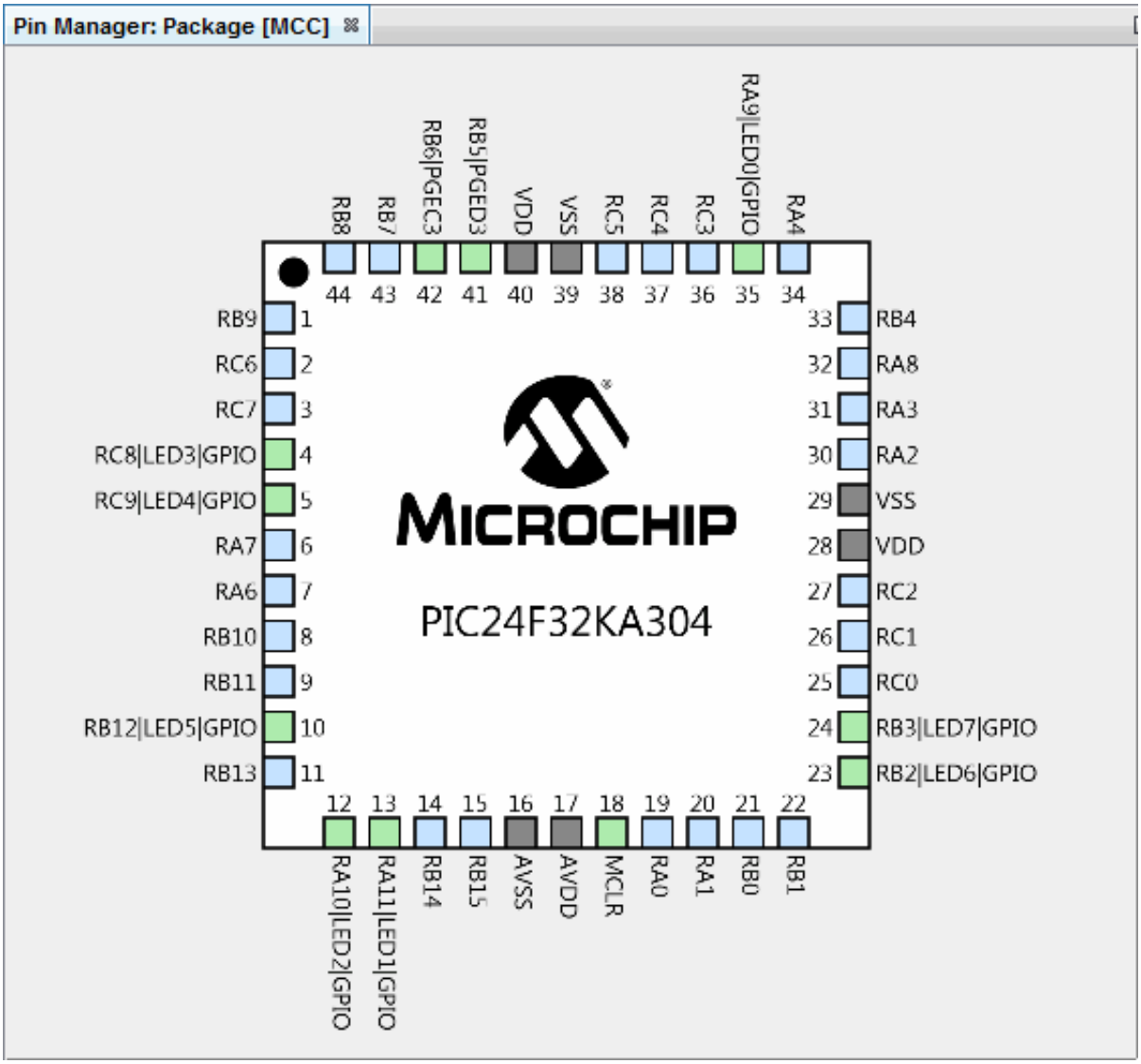


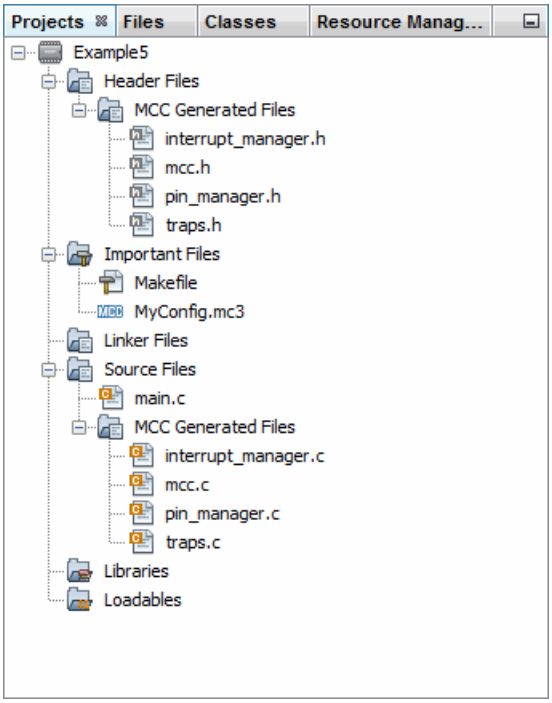
图18： EEDATA项目引脚封装



按照上述各图配置完代码后，请单击“Project Resources”窗口上的**Generate**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。

可通过生成陷阱文件来捕捉潜在的错误。尽管本应用程序中不会使用中断，但会生成中断管理器文件供将来使用。

图19： 通过MCC生成的代码的EEDATA项目树



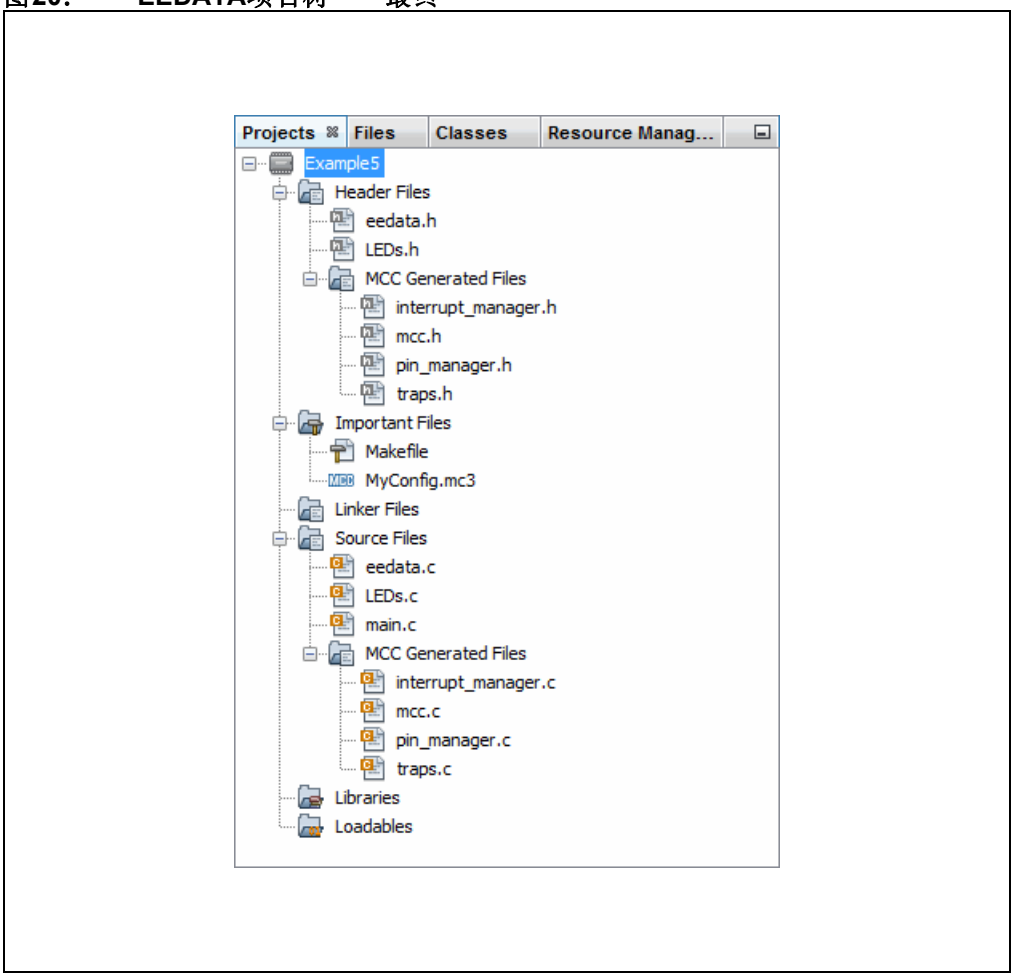
GPIO生成的文件默认为模拟输入，因此必须在pin_manager.c中将其更改为数字输入（第5.1节）。

另外，由于LED不只连接到一个端口，而是分布在三个端口上，因此需要使用额外的类型定义和代码来将正确的LED值赋值给端口引脚。头文件LEDs.h（第5.2节）和C文件LEDs.c（第5.3节）已添加到项目中。

如前文所述，MCC中目前没有用于16位器件的EEData器件资源，因此需要手动添加代码。头文件eedata.h（第5.4节）和C文件eedata.c（第5.5节）已添加到项目中。

最终的项目树将如图20所示。

图20： EEDATA项目树——最终



向程序中添加功能时始终需要编辑main.c（第5.6节）。请查看生成的文件和附加文件以找到您的代码中可能需要的任何函数或宏。

5.1 修改后的pin_manager.c代码

main.c模板文件已经过编辑，如下所示。部分注释和生成的内容已删除（< >内）。已更改的代码以绿色显示。

```
/**
  System Interrupts Generated Driver File

<See generated pin_manager.c for file information.>

Copyright (c) 2013 - 2015 released Microchip Technology Inc. All
rights reserved.

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

/**
    Section: Includes
 */
#include <xc.h>
#include "pin_manager.h"

/**
    void PIN_MANAGER_Initialize(void)
 */
void PIN_MANAGER_Initialize(void) {

<See generated pin_manager.c for port setup information.>

/**************************************************************************
 * Setting the Analog/Digital Configuration SFR(s)
 *************************************************************************/
    ANSA = 0x0;
    ANSB = 0x0;
    ANSC = 0x0;

}
```

5.2 LEDs.h代码

部分注释已删除（< >内）。

```
/*-----------------------------------------------------------------
 * PICF32KA304 LEDs header
 *
 * (c) Copyright 1999-2015 Microchip Technology, All rights reserved
 *
<See generated header files for additional copyright information.>
 */

/*******************************************************************
 * Union of structures to hold value for display on LEDs
 * LAT_LEDx - bit fields of value
 * w - entire value
 *******************************************************************/
typedef union {
    struct {
      unsigned LAT_LED0:1;
      unsigned LAT_LED1:1;
      unsigned LAT_LED2:1;
      unsigned LAT_LED3:1;
      unsigned LAT_LED4:1;
      unsigned LAT_LED5:1;
      unsigned LAT_LED6:1;
      unsigned LAT_LED7:1;
    };
    struct {
      unsigned w:16;
    };
} LAT_LEDSBITS;
extern volatile LAT_LEDSBITS LAT_LEDSbits;

/* LAT_LEDSBITS */
#define _LED0 LAT_LEDSbits.LAT_LED0
#define _LED1 LAT_LEDSbits.LAT_LED1
#define _LED2 LAT_LEDSbits.LAT_LED2
#define _LED3 LAT_LEDSbits.LAT_LED3
#define _LED4 LAT_LEDSbits.LAT_LED4
#define _LED5 LAT_LEDSbits.LAT_LED5
#define _LED6 LAT_LEDSbits.LAT_LED6
#define _LED7 LAT_LEDSbits.LAT_LED7
#define _LEDS LAT_LEDSbits.w

/*******************************************************************
 * Function: DisplayValueOnLEDs
 * Precondition: None.
 * Overview: Display input value on Explorer 16 LEDs
 * Input: Value to display
 * Output: None.
 *******************************************************************/
void DisplayValueOnLEDs(unsigned int value);
/**
 End of File
 */
```

5.3 LEDs.c代码

部分注释已删除（< >内）。

```
/**
  Display on LEDs Source File

<See LEDs.c for file description information.>

 */
/*
Copyright (c) 2013 - 2015 released Microchip Technology Inc. All
rights reserved.

<See generated header files for additional copyright information.>
 */

#include "mcc_generated_files/mcc.h"
#include "LEDs.h"

volatile LAT_LEDSBITS LAT_LEDSbits;

/*********************************************************************
 * Function: DisplayValueOnLEDs
 * Precondition: None.
 * Overview: Display input value on Explorer 16 LEDs
 * Input: Value to display
 * Output: None.
 ********************************************************************/
void DisplayValueOnLEDs(unsigned int value) {

    _LEDS = value;

    _LATA9  = _LED0;
    _LATA10 = _LED1;
    _LATA11 = _LED2;
    _LATC8  = _LED3;
    _LATC9  = _LED4;
    _LATB12 = _LED5;
    _LATB2  = _LED6;
    _LATB3  = _LED7;

}

/**
 End of File
 */
```

5.4 eedata.h代码

部分注释已删除（< >内）。

```
/*-------------------------------------------------------------------
 * PICF32KA304 Data EEPROM header
 *
 * (c) Copyright 1999-2015 Microchip Technology, All rights reserved
 *
<See generated header files for additional copyright information.>
 */

/*********************************************************************
 * Function: EEData_WTL
 * Precondition: None.
 * Overview: Write one word of EEData
 * Input: Action to take: Erase or Write, Data to write
 * Output: None.
 ********************************************************************/
void EEData_WTL(unsigned int action, unsigned int data);

/*********************************************************************
 * Function: EEData_Erase
 * Precondition: None.
 * Overview: Set up erase of one word of EEData
 * Input: None.
 * Output: None.
 ********************************************************************/
void EEData_Erase(void);

/*********************************************************************
 * Function: EEData_Write
 * Precondition: None.
 * Overview: Set up write of one word of EEData
 * Input: Data to write
 * Output: None.
 ********************************************************************/
void EEData_Write(unsigned int data);

/*********************************************************************
 * Function: EEData_Read
 * Precondition: None.
 * Overview: Read one word of EEData
 * Input: None.
 * Output: Value read from EEData
 ********************************************************************/
unsigned int EEData_Read(void);

/**
 End of File
 */
```

5.5 eedata.c代码

部分注释已删除（< >内）。

```
/**
  Data EEPROM Write and Read

<See eedata.c for file description information.>

 */
/*
Copyright (c) 2013 - 2015 released Microchip Technology Inc. All
rights reserved.

<See generated header files for additional copyright information.>
 */

#include <xc.h>
#include "eedata.h"

#define ERASE_EEWORD 0x4058
#define WRITE_EEWORD 0x4004

int __attribute__ ((space(eedata))) eeData = 0x0;
unsigned int offset = 0x0;

/******************************************************************
 * Function: EEData_WTL
 * Precondition: None.
 * Overview: Write one word of EEData
 * Input: Action to take: Erase or Write, Data to write
 * Output: None.
 ******************************************************************/
void EEData_WTL(unsigned int action, unsigned int data) {

    // Set up NVMCON to write one word of data EEPROM
    NVMCON = action;

    // Set up a pointer to the EEPROM location to be written
    TBLPAG = __builtin_tblpage(&eeData);
    offset = __builtin_tbloffset(&eeData);
    __builtin_tblwtl(offset, data);

    // Issue Unlock Sequence & Start Write Cycle
    __builtin_write_NVM();

    // Wait for completion
    while(NVMCONbits.WR);
}

/******************************************************************
 * Function: EEData_Erase
 * Precondition: None.
 * Overview: Set up erase of one word of EEData
 * Input: None.
 * Output: None.
 ******************************************************************/
void EEData_Erase(void) {

    EEData_WTL(ERASE_EEWORD, 0);
}
```

```
/*****************************************************************
 * Function: EEData_Write
 * Precondition: None.
 * Overview: Set up write of one word of EEData
 * Input: Data to write
 * Output: None.
 *****************************************************************/
void EEData_Write(unsigned int data) {

    EEData_WTL(WRITE_EEWORD, data);
}

/*****************************************************************
 * Function: EEData_Read
 * Precondition: None.
 * Overview: Read one word of EEData
 * Input: None.
 * Output: Value read from EEData
 *****************************************************************/
unsigned int EEData_Read(void) {

    // Set up a pointer to the EEPROM location to be read
    TBLPAG = __builtin_tblpage(&eeData);
    offset = __builtin_tbloffset(&eeData);

    // Read the EEPROM data
    return __builtin_tblrdl(offset);
}

/**
 End of File
 */
```

5.6 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）。已添加的代码以绿色显示。

```
/**
  Generated Main Source File

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

#include "mcc_generated_files/mcc.h"
#include "eedata.h"
#include "LEDs.h"
#include "libpic30.h"

#define IC_DELAY 1000000

unsigned int data_write = 0x0;
unsigned int data_read = 0x0;

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

    while (1) {

        data_write++;

        // Erase one word of data EEPROM
        EEData_Erase();

        // Write one word of data EEPROM
        EEData_Write(data_write);

        // Read one word of data EEPROM
        data_read = EEData_Read();

        // Display result on LEDs
        DisplayValueOnLEDs(data_read);

        // Delay change on LEDs so visible
        __delay32(IC_DELAY); // delay in instruction cycles

    }

    return -1;
}
/**
 End of File
 */
```

请参见第5.7节

请参见第5.8节

请参见第5.9节

5.7 擦除并写入EEData

要在EEData中写入单个字，必须遵循以下序列：

1. 擦除一个数据EEPROM字。
2. 将数据字写入数据EEPROM锁存器。
3. 将数据字编程到EEPROM。

用于擦除EEData并写入一个字的代码位于eedata.c中（第5.5节）。

对于PIC24F32KA304器件，在擦除或写入EEData之前，需要将一个密钥序列写入NVMKEY（在NVMCON中）。

使用内置函数来简化编码：

- `unsigned int __builtin_tblpage(const void *p);`
- `unsigned int __builtin_tbloffset(const void *p);`
- `void __builtin_tblwtl(unsigned int offset, unsigned int data);`
- `void __builtin_write_NVM(void);`

有关这些函数的详细信息，请参见《MPLAB® XC16 C编译器用户指南》（DS50002071E_CN）的附录G“内置函数”。

5.8 读取EEData

对于本示例，写入EEData后，将读取EEData的字。

用于读取EEData的一个字的代码位于eedata.c中（第5.5节）。

使用内置函数来简化编码：

- `unsigned int __builtin_tblpage(const void *p);`
- `unsigned int __builtin_tbloffset(const void *p);`
- `unsigned int __builtin_tblrdl(unsigned int offset);`

有关这些函数的详细信息，请参见《MPLAB® XC16 C编译器用户指南》（DS50002071E_CN）的附录G“内置函数”。

5.9 在LED上显示数据以及延时

在演示板LED上显示数据对于该器件而言更为复杂，因为三个端口均与LED连接。因此，需使用联合和结构数据类型，以便可以赋值整个数据值（LAT_LEDSbits.w），之后可以访问各个位，以便将其赋值给正确的端口引脚进行显示（例如，LATAbits.LATA9 = LAT_LEDSbits.LAT_LED0）。

用于创建联合和结构的代码位于LEDs.h中（第5.2节）。

用于将LED值赋值给端口引脚的代码位于LEDs.c中（第5.3节）。

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（同第2节）来降低执行速度。

A. 在MPLAB X IDE中运行代码

A.1 创建项目：

首先，创建一个项目：

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. 按照屏幕指示创建一个新项目：
   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）**：通过序列号（Serial Number，SN）SNxxxxxx选择硬件调试工具。如果调试工具名称下未显示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XC16（*最新版本号*）[`bin`文件夹位置]。如果XC16下未显示编译器，请确保编译器已正确安装且MPLAB X IDE可找到可执行文件。选择*Tools>Options*（工具>选项），单击**Build Tools**（编译工具）选项卡上的**Embedded**（已安装工具）按钮，查找您的编译器。有关详细信息，请参见MPLAB XC16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“xc16-gcc”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）复选框。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`文件和其他文件。

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1-1：　　工具栏图标

调试运行　　暂停

B. 获取软件和硬件

对于本文档中的MPLAB XC16项目，装有PIC24F PIM的Explorer 16/32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 XC16 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 XC16 C编译器v1.35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接插模块（PIM）

Microchip Technology网站上的以下位置提供了示例中使用的PIC MCU PIM：

PIC24FJ128GA010：https://www.microchip.com/MA240011

PIC24F32KA304：https://www.microchip.com/MA240022

B.4 获取并设置Explorer 16/32板

Explorer 16/32开发板、原理图和文档可从以下网址找到：
https://www.microchip.com/dm240001-2

下表列出了跳线和开关设置。

表1-1： 项目的跳线/开关选择

跳线/开关	选择	跳线/开关	选择
JP2	闭合	J37	断开
J19	断开	J38	断开
J22	断开	J39	默认
J23	默认	J41	断开
J25	闭合	J42	断开
J26	闭合	J43	默认
J27	断开	J44	默认
J28	断开	J45	默认
J29	断开	J50	闭合
J33	断开		

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

注：

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：978-1-5224-5729-9

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