
面向嵌入式工程师的MPLAB® XC16用户指南

简介

本文档提供了5个适用于16位器件和MPLAB® XC16 C编译器的代码示例。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”。

```
#include <xc.h> ← 请参见第1.1节

// PIC24FJ128GA010 Configuration Bit Settings

// For more on Configuration Bits, ← 请参见第1.2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

#define LEDS_ON_OFF 0x55 ← 请参见第1.3节

int main(void) {

    // 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 = LEDS_ON_OFF; // write to port latch

    return 0;
}
```

1.1 头文件<xc.h>

本头文件允许源文件中的代码访问编译器或器件特定的功能。可在MPLAB XC16安装目录的support子目录下找到本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。

1.2 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。

1.2.1 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 <http://www.microchip.com> 上搜索相应的数据手册。

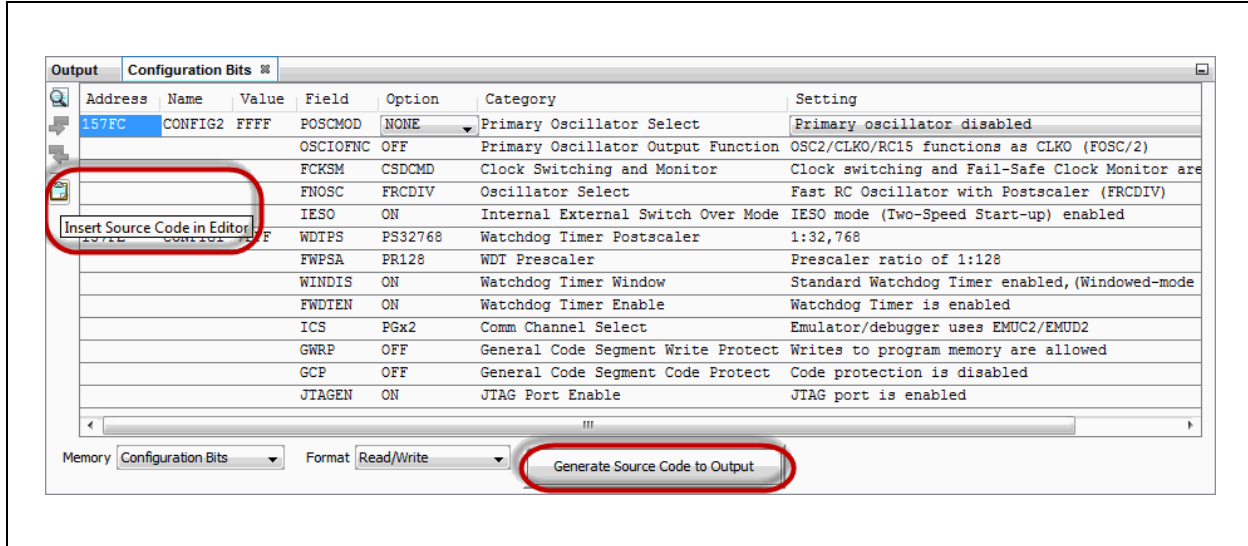
有关每款器件可用配置位的更多信息，请参见 MPLAB XC16 安装路径下的如下文件：

MPLAB XC16 Installation Directory/docs/config_index.html

1.2.2 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： 配置位窗口



选择完设置后，单击您想要放置pragma伪指令的位置处的代码，并单击**Insert Source Code in Editor**（在编辑器中插入源代码）图标。

或者，您也可单击**Generate Source Code to Output**（生成要输出的源代码），然后将pragma伪指令从输出窗口复制到您的代码中。

1.3 为LED值定义宏

如下一节所述，要写入LED的值已赋值给描述性宏（LEDS_ON_OFF），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。惟一的问题在于这些引脚默认设置为模拟I/O，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 `_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。

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

// 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

#define LEDS_ON_OFF 0x55
#define LEDS_OFF_ON 0xAA
#define IC_DELAY 1500000

int main(void) {

    // 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

        LATA = LEDS_ON_OFF; // write to port latch

        // delay value change ← 请参见第2.3节
        __delay32(IC_DELAY); // delay in instruction cycles

        LATA = LEDS_OFF_ON; // write to port latch

        __delay32(IC_DELAY); // 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，需在循环的第一部分为PORTA赋值宏LEDS_ON_OFF，然后在循环的第二部分为PORTA赋值互补宏LEDS_OFF_ON。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。此时，将返回-1。

2.3 __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。__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。

```
#include <xc.h>

// 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

// Interrupt function ←—— 请参见第3.1节

void __attribute__((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 `isr()`

可使用 `interrupt` 属性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：

<http://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

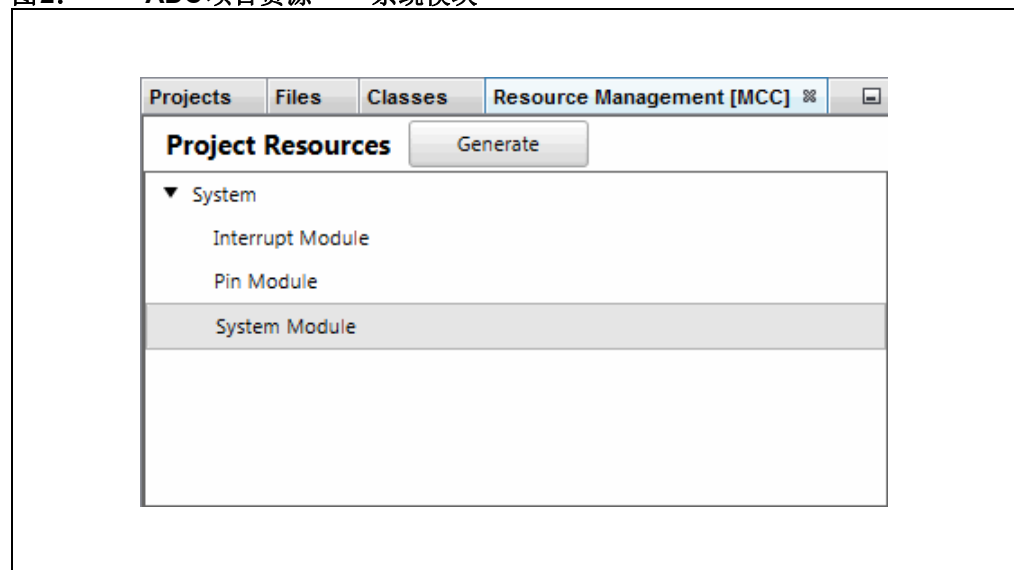


图3: ADC项目系统模块配置

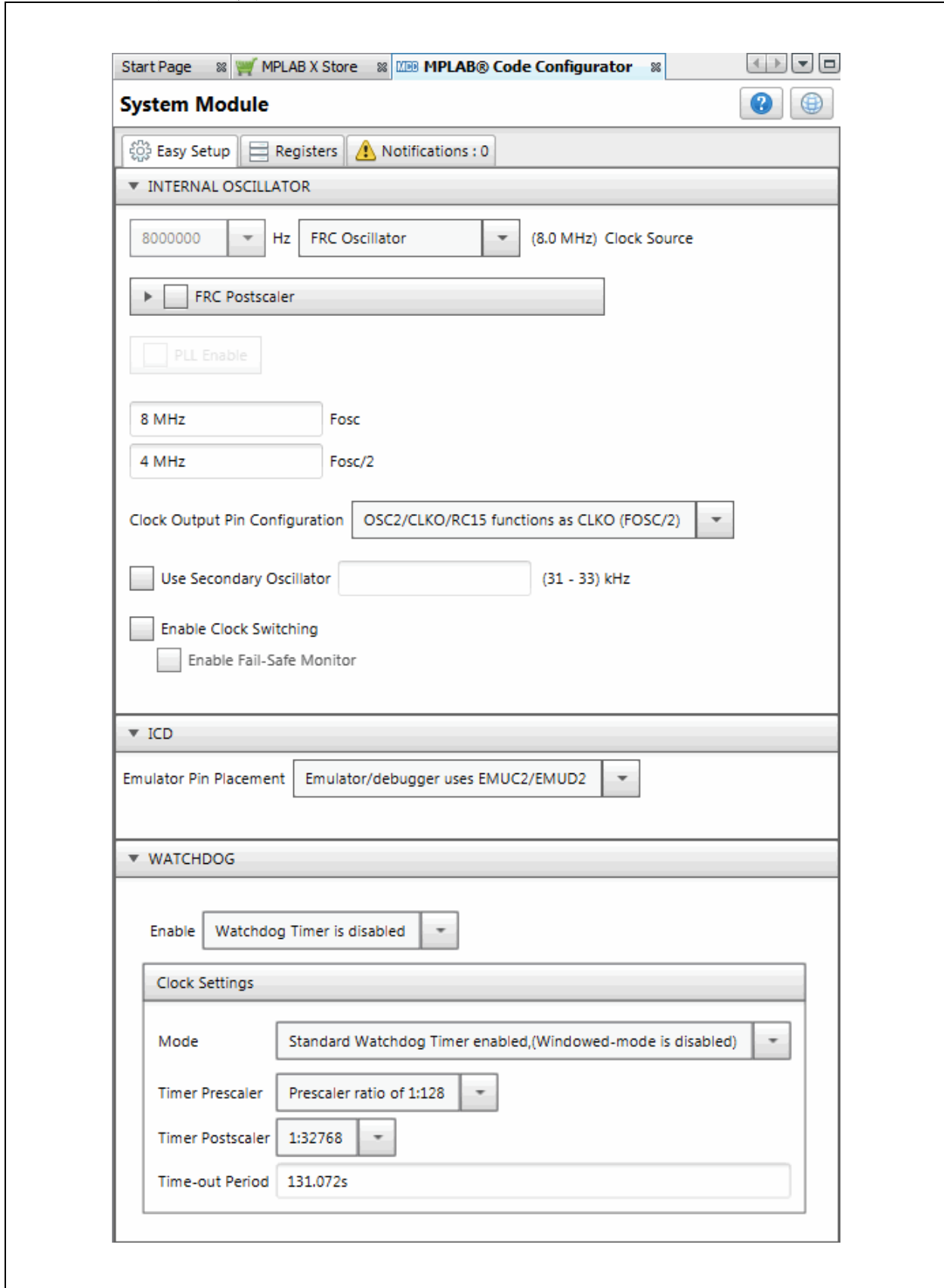


图4: ADC项目资源——ADC模块

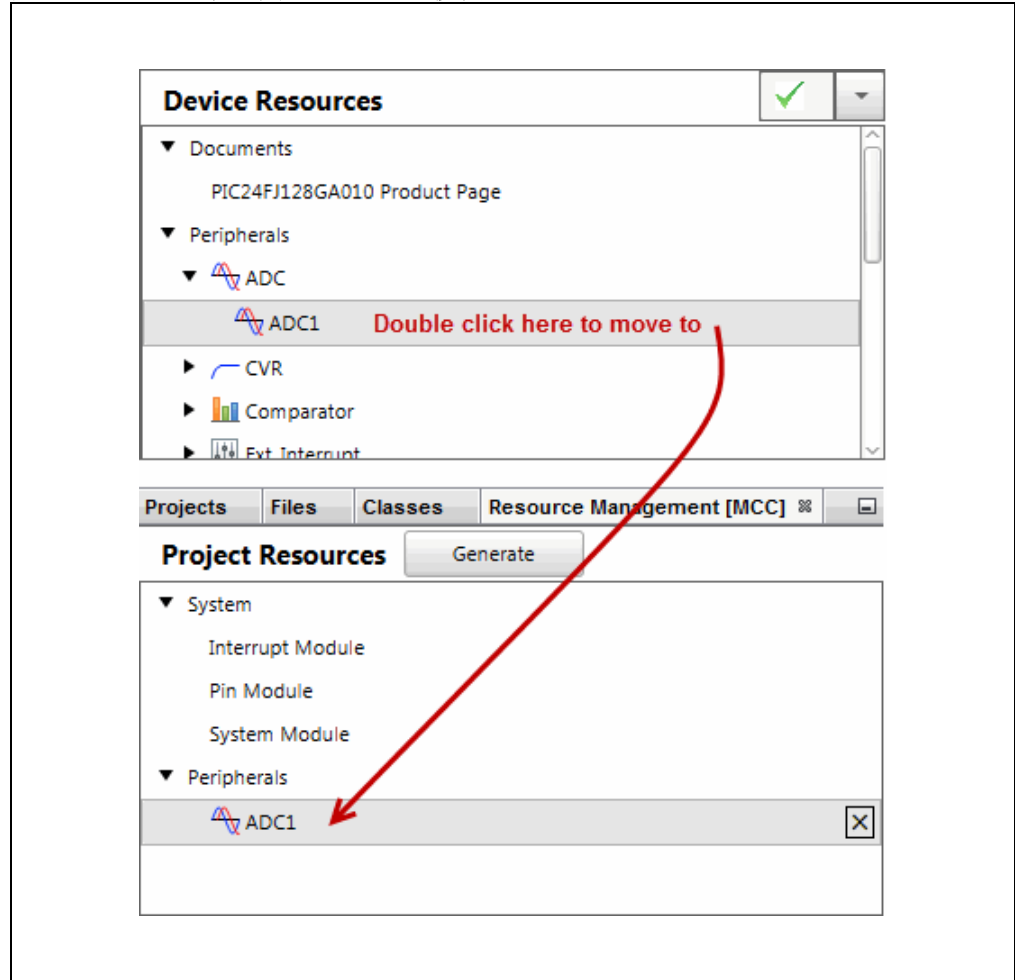


图5: ADC项目ADC1配置

The screenshot shows the MPLAB Code Configurator interface for configuring the ADC1 module. Under 'Hardware Settings', 'Enable ADC' and 'Enable Auto Sampling' are checked. The 'ADC Clock' section shows 'Conversion Clock Source' set to FOSC/2, 'Conversion Clock' set to 1 TCY, 'Acquisition Time' set to 15 TAD, and 'TAD' set to 2.5E-7s. Other settings include 'Differential Sampling' (AVSS), 'Conversion Trigger' (Internal counter ends sampling and starts conversion), 'Output Format' (Absolute decimal result, unsigned, right-justified), 'Positive Voltage Ref' (AVDD), and 'Negative Voltage Ref' (AVSS). The 'Selected Channels' table is as follows:

Channel	Custom Name	Scan Enable
AN5	channel_AN5	<input checked="" type="checkbox"/>

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

Output		Pin Manager: Grid [MCC]																															
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													<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>										
	VREF+	input									<input checked="" type="checkbox"/>																						
	VREF-	input									<input checked="" type="checkbox"/>																						

图7: ADC项目资源——引脚模块

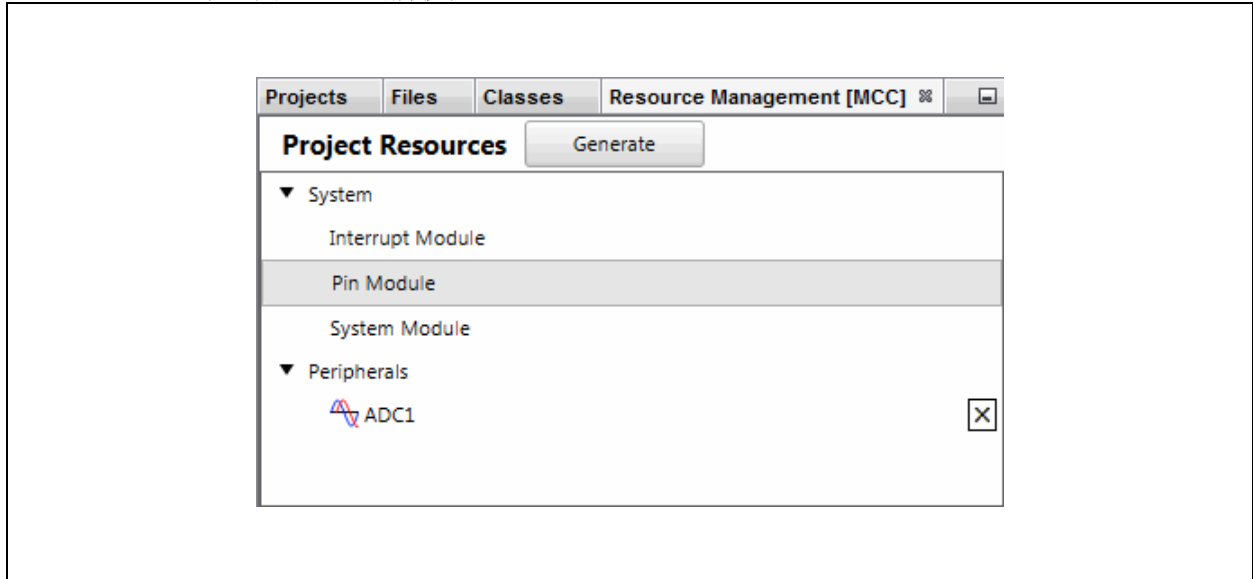


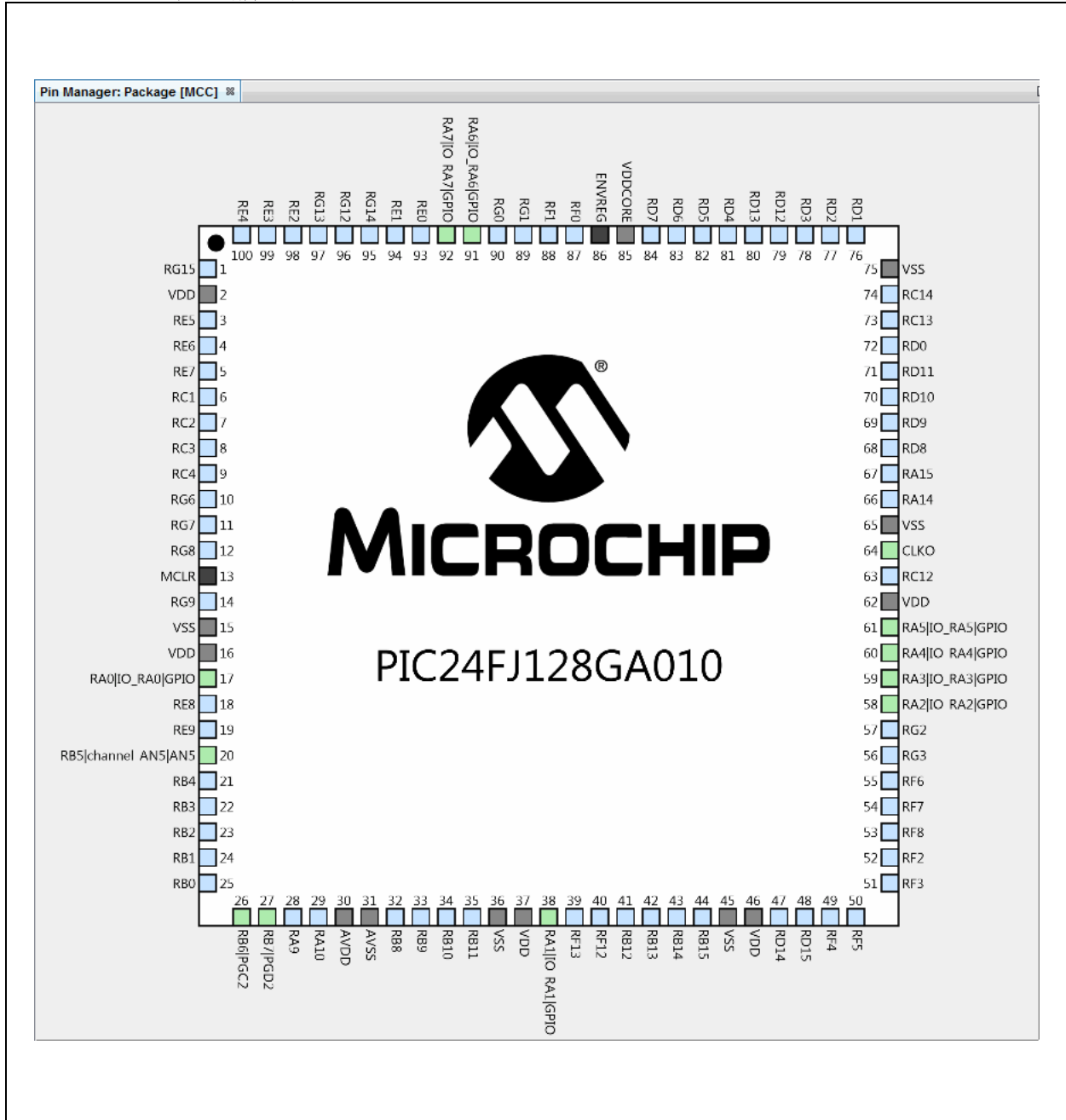
图8: ADC项目 I/O 引脚配置

The screenshot shows the 'Pin Module' configuration window for package TQFP100. The table below lists the configured pins and their settings.

Pin Name	Module	Function	Custom Name	Start High	Analog	Output	WPU	WPD	OD	IOC
RA0	Pin Module	GPIO	IO_RA0	<input type="checkbox"/>		<input checked="" type="checkbox"/>			<input type="checkbox"/>	
RA1	Pin Module	GPIO	IO_RA1	<input type="checkbox"/>		<input checked="" type="checkbox"/>			<input type="checkbox"/>	
RA2	Pin Module	GPIO	IO_RA2	<input type="checkbox"/>		<input checked="" type="checkbox"/>			<input type="checkbox"/>	
RA3	Pin Module	GPIO	IO_RA3	<input type="checkbox"/>		<input checked="" type="checkbox"/>			<input type="checkbox"/>	
RA4	Pin Module	GPIO	IO_RA4	<input type="checkbox"/>		<input checked="" type="checkbox"/>			<input type="checkbox"/>	
RA5	Pin Module	GPIO	IO_RA5	<input type="checkbox"/>		<input checked="" type="checkbox"/>			<input type="checkbox"/>	
RA6	Pin Module	GPIO	IO_RA6	<input type="checkbox"/>		<input checked="" type="checkbox"/>			<input type="checkbox"/>	
RA7	Pin Module	GPIO	IO_RA7	<input type="checkbox"/>		<input checked="" type="checkbox"/>			<input type="checkbox"/>	
RB5	ADC1	AN5	channel_AN	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>	none
RB6	ICD	PGC2		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			<input type="checkbox"/>	
RB7	ICD	PGD2		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			<input type="checkbox"/>	

在图9中选中引脚RA0:7后，这些引脚将出现在上面的窗口中。
 图6中之前选中的是RB5。
 RB6和RB7预先选择用于调试通信。
 在窗口中显示后，可为每个引脚查看或选择引脚配置。

图 10: ADC 项目引脚封装

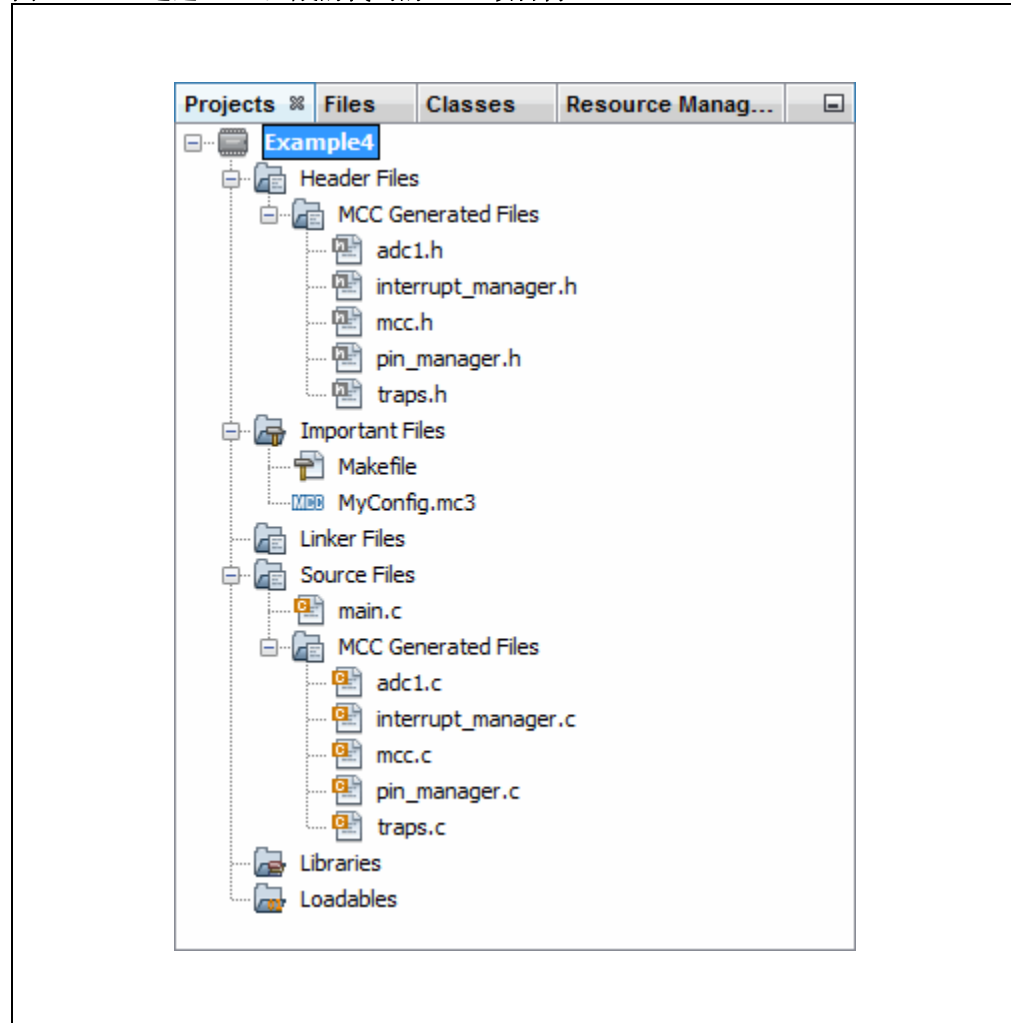


按照上述各图配置完代码后，请单击“Project Resources”（项目资源）窗口上的 **Generate**（生成）按钮（图7）。通过MCC生成的代码是模块化的。因此，主程序代码、系统代码和外设代码均位于单独的文件中。此外，每个外设都有自己的头文件。

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



4.1 main.c修改后的代码

main.c模板文件已经过编辑，如下所示。部分注释用< >括起，表示已删除。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"

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

本示例使用另一款Microchip 器件PIC24F32KA304 MCU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12）。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：

<http://www.microchip.com/PIC24F32KA304>

图12: EEDATA项目资源——系统模块

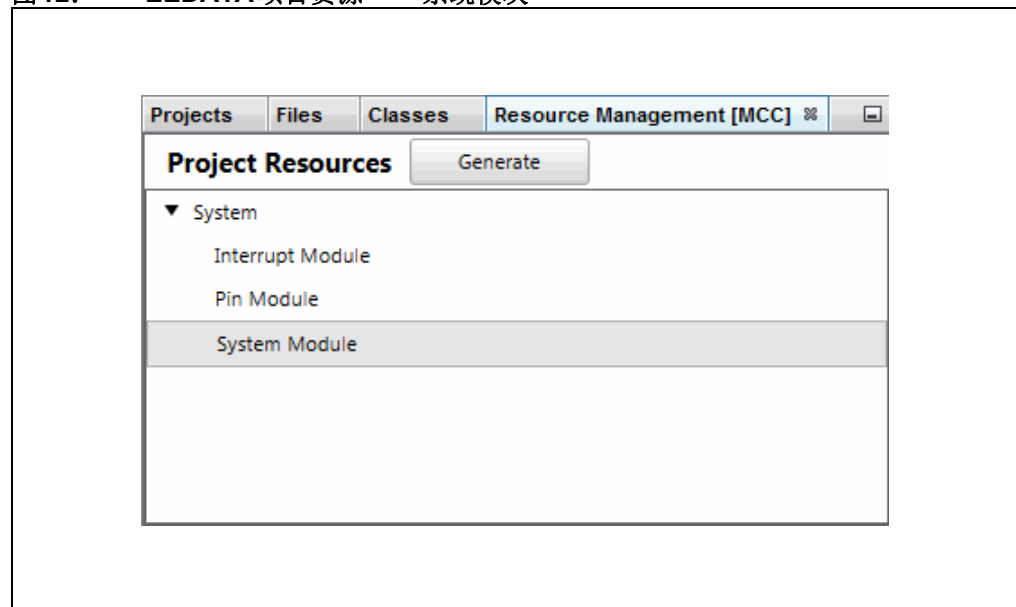


图 13: EEDATA 项目系统模块配置

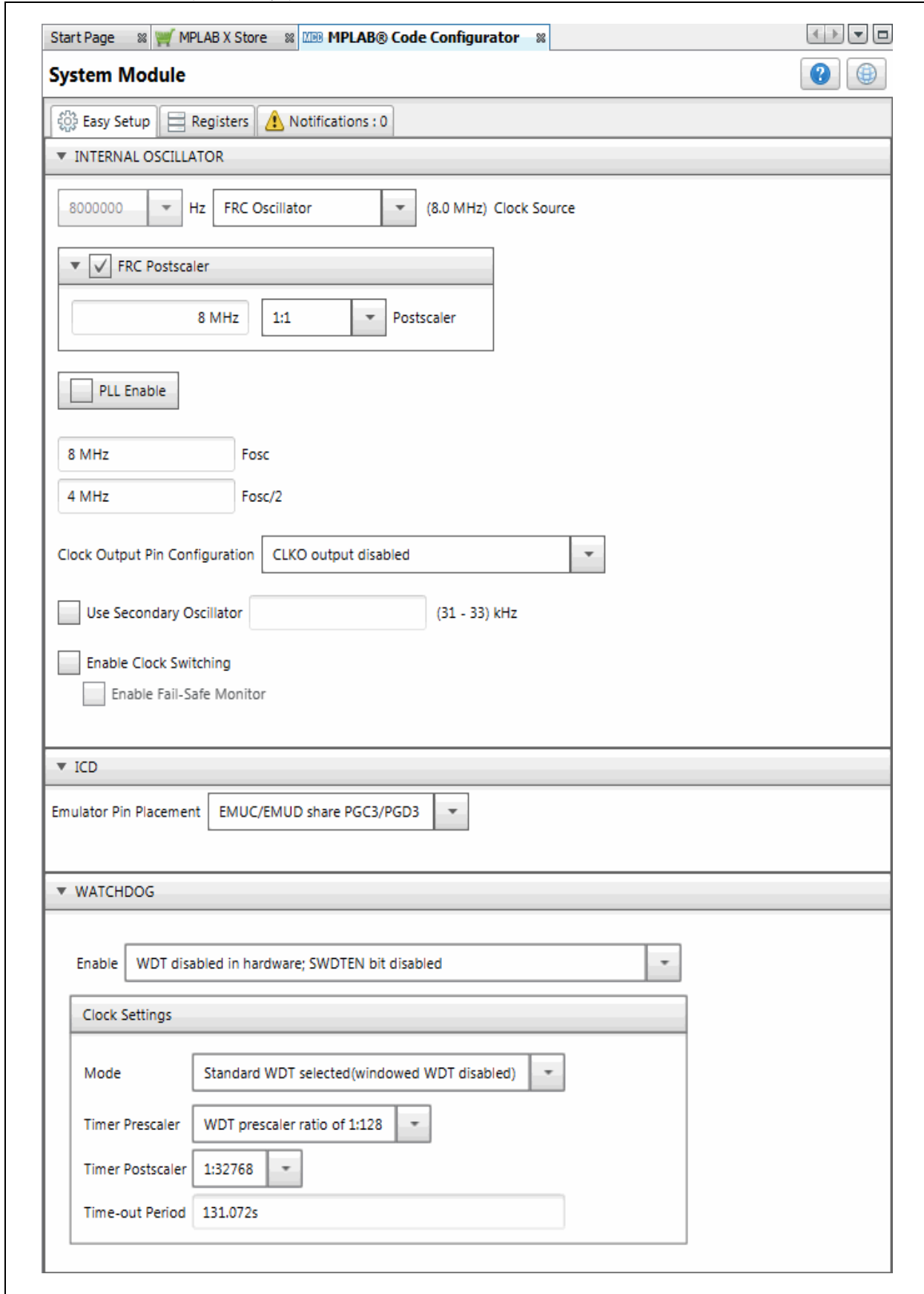


图 14: EEDATA 项目资源——引脚模块

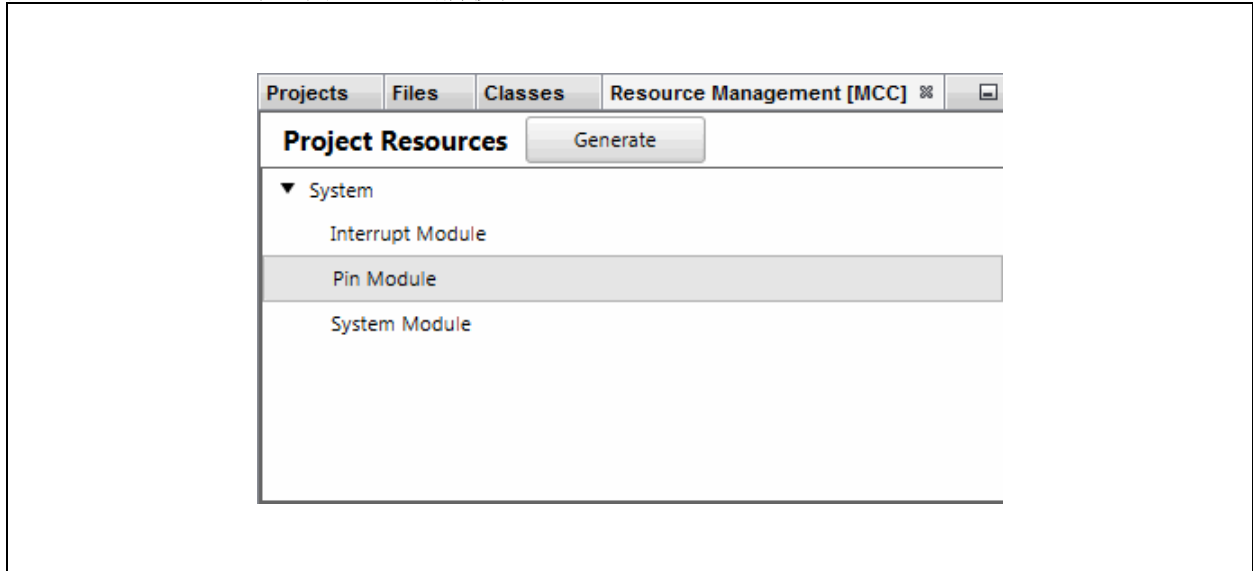


图 15: EEDATA 项目 I/O 引脚配置

The screenshot shows the 'Pin Module' configuration window in the MPLAB Code Configurator. The window title is 'Pin Module' and it shows 'Selected Package: TQFP44'. Below the title bar is a table with the following columns: Pin Name, Module, Function, Custom Name, Start High, Analog, Output, WPU, WPD, OD, and IOC. The table contains 12 rows of pin configurations.

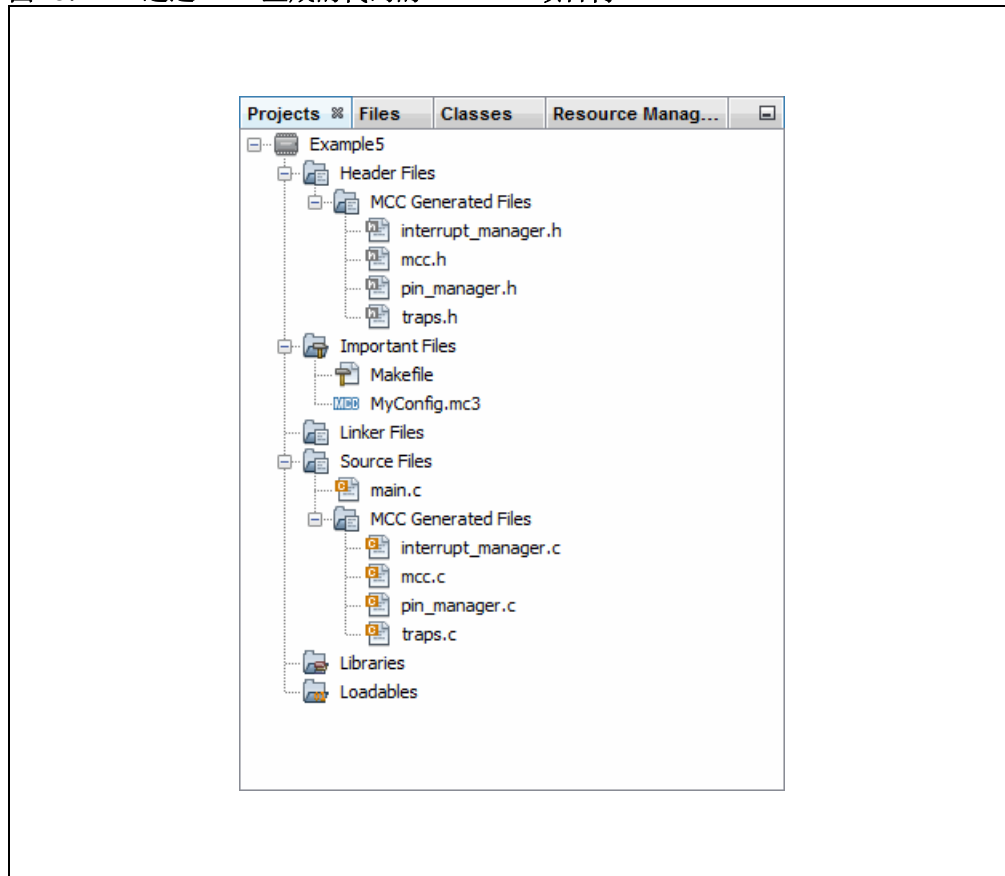
Pin Name	Module	Function	Custom Name	Start High	Analog	Output	WPU	WPD	OD	IOC
RA9	Pin Module	GPIO	LED0	<input type="checkbox"/>		<input checked="" type="checkbox"/>			<input type="checkbox"/>	none
RA10	Pin Module	GPIO	LED2	<input type="checkbox"/>		<input checked="" type="checkbox"/>			<input type="checkbox"/>	none
RA11	Pin Module	GPIO	LED1	<input type="checkbox"/>		<input checked="" type="checkbox"/>			<input type="checkbox"/>	none
RB2	Pin Module	GPIO	LED6	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	none
RB3	Pin Module	GPIO	LED7	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	none
RB5	ICD	PGED3		<input type="checkbox"/>		<input type="checkbox"/>			<input type="checkbox"/>	none
RB6	ICD	PGEC3		<input type="checkbox"/>		<input type="checkbox"/>			<input type="checkbox"/>	none
RB12	Pin Module	GPIO	LED5	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	none
RC8	Pin Module	GPIO	LED3	<input type="checkbox"/>		<input checked="" type="checkbox"/>			<input type="checkbox"/>	none
RC9	Pin Module	GPIO	LED4	<input type="checkbox"/>		<input checked="" type="checkbox"/>			<input type="checkbox"/>	none

在图 16 中选中引脚 RA9:11、RB2:3、RB12 和 RC8:9 后，这些引脚将出现在上面的窗口中。RB6 和 RB7 预先选择用于调试通信。在窗口中显示后，可为每个引脚查看或选择引脚配置。

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18: 通过MCC生成的代码的EEDATA项目树



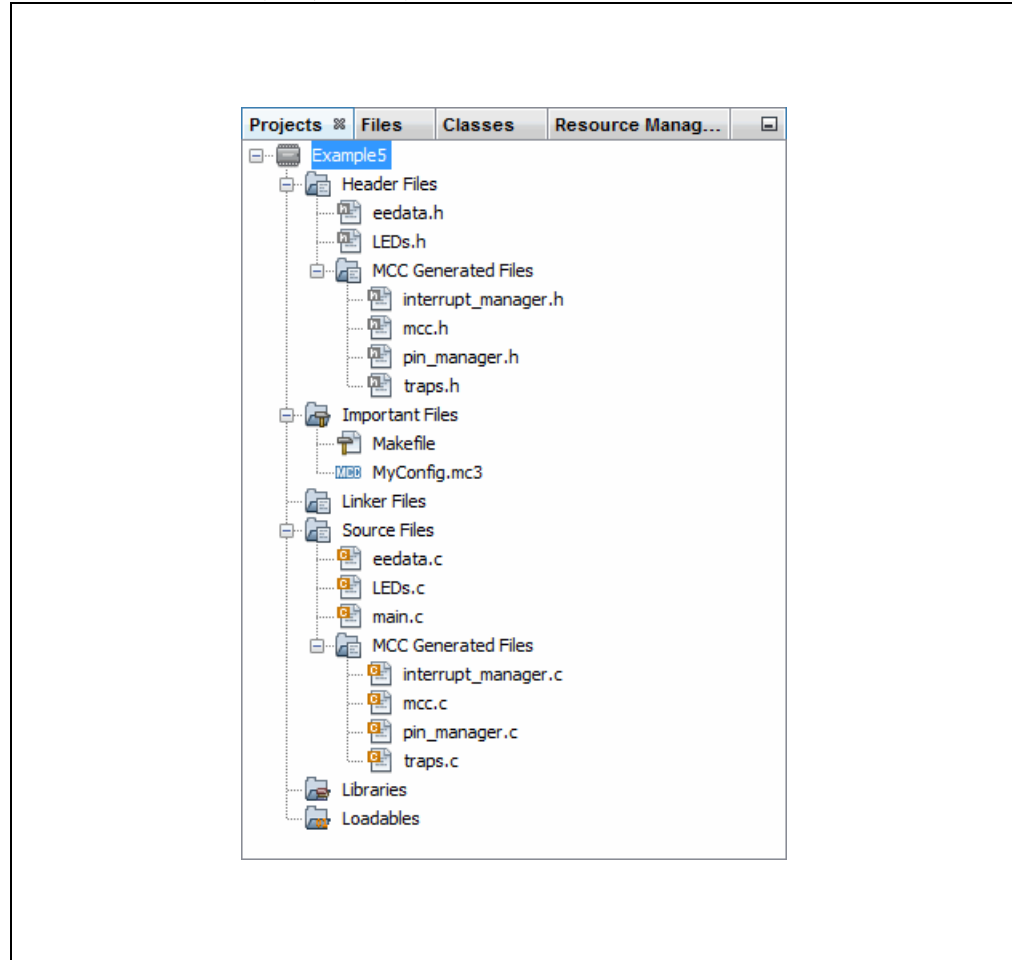
GPIO生成的文件默认为模拟输入，因此必须在`pin_manager.c`中将其更改为数字输入（第5.1节）。

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19所示。

图19: 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 pin_manager.c修改后的代码

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用< >括起，表示已删除。已更改的代码以红色显示。

```
/**
 * 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

部分注释用< >括起，表示已删除。

```
/*-----  
 * 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

部分注释用< >括起，表示已删除。

```
/**
 * 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

部分注释用< >括起，表示已删除。

```
/*-----  
 * 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

部分注释用< >括起，表示已删除。

```
/**
 * 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_tblrddl(offset);
}

/**
End of File
*/

```

5.6 main.c修改后的代码

main.c模板文件已经过编辑，如下所示。部分注释用< >括起，表示已删除。已添加的代码以红色显示。

```
/**
 *
 * 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 ← 请参见第5.7节
        EEData_Erase();

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

        // Read one word of data EEPROM ← 请参见第5.8节
        data_read = EEData_Read();

        // Display result on LEDs ← 请参见第5.9节
        DisplayValueOnLEDs(data_read);

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

    }

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

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](#)）。

再次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_tblrdr(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

首先，创建一个项目：

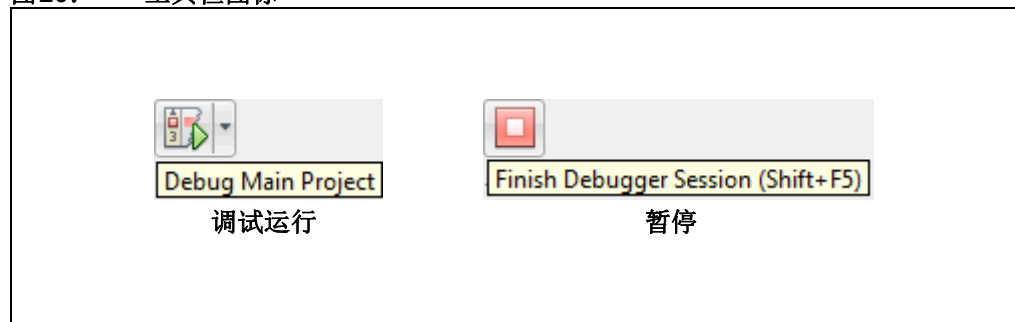
1. 启动MPLAB X IDE。
2. 从IDE中启动新建项目向导（**File>New Project**（文件>新建项目））。
3. 按照屏幕指示创建一个新项目：
 - a) **Choose Project（选择项目）**：选择“Microchip Embedded”（Microchip 嵌入式），然后选择“Standalone Project”（独立项目）。
 - b) **Select Device（选择器件）**：选择示例器件。
 - c) **Select Header（选择调试头）**：无。
 - d) **Select Tool（选择工具）**：通过序列号（Serial Number, 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。

现在，创建一个文件来保存示例代码（除非您使用了MCC）：

1. 右键单击**Projects**（项目）窗口中的项目名称。选择**New>Empty File**（新建>空白文件）。随即将显示**New Empty File**（新建空白文件）对话框。
2. 在“File name”（文件名）下输入名称。
3. 单击**Finish**（完成）。
4. 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，然后选择**File>Save**（文件>保存）。

编译代码并下载到器件中，然后选择**Debug Run**（调试运行）执行代码。演示板上的LED将交替点亮。单击**Halt**（暂停）结束执行。

图20: 工具栏图标



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 v3.45及以上版本：

<http://www.microchip.com/mplab/mplab-x-ide>

可从以下网址找到MPLAB XC16 C编译器v1.26及以上版本：

<http://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 v3.25及以上版本：

<http://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: <http://www.microchip.com/MA240011>

PIC24F32KA304: <http://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：

<http://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：

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

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。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，不得暗或以其他方式转让任何许可证。

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 and 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。

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

商标

Microchip 的名称和徽标组合、Microchip 徽标、AnyRate、AVR、AVR 徽标、AVR Freaks、BeaconThings、BitCloud、CryptoMemory、CryptoRF、dsPIC、FlashFlex、flexPWR、Heldo、JukeBlox、KEELOQ、KEELOQ 徽标、Kleer、LANCheck、LINK MD、maXStylus、maXTouch、MediaLB、megaAVR、MOST、MOST 徽标、MPLAB、OptoLyzer、PIC、picoPower、PICSTART、PIC32 徽标、Prochip Designer、QTouch、RightTouch、SAM-BA、SpyNIC、SST、SST 徽标、SuperFlash、tinyAVR、UNI/O 及 XMEGA 均为 Microchip Technology Inc. 在美国和其他国家或地区的注册商标。

ClockWorks、The Embedded Control Solutions Company、EtherSynch、Hyper Speed Control、HyperLight Load、IntelliMOS、mTouch、Precision Edge 和 Quiet-Wire 均为 Microchip Technology Inc. 在美国的注册商标。

Adjacent Key Suppression、AKS、Analog-for-the-Digital Age、Any Capacitor、AnyIn、AnyOut、BodyCom、chipKIT、chipKIT 徽标、CodeGuard、CryptoAuthentication、CryptoCompanion、CryptoController、dsPICDEM、dsPICDEM.net、Dynamic Average Matching、DAM、ECAN、EtherGREEN、In-Circuit Serial Programming、ICSP、Inter-Chip Connectivity、JitterBlocker、KleerNet、KleerNet 徽标、Mindi、MiWi、motorBench、MPASM、MPF、MPLAB Certified 徽标、MPLIB、MPLINK、MultiTRAK、NetDetach、Omniscient Code Generation、PICDEM、PICDEM.net、PICkit、PICtail、PureSilicon、QMatrix、RightTouch 徽标、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 Inc. 在美国和其他国家或地区的商标。

SQTP 为 Microchip Technology Inc. 在美国的服务标记。

Silicon Storage Technology 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。

© 2018, Microchip Technology Inc. 版权所有。

ISBN: 978-1-5224-2572-4



全球销售及服务中心

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: 45-4450-2828
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: 49-7131-67-3636

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: 47-7289-7561

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