



MPLAB® XC8 嵌入式工程师用户指南——AVR® MCU

简介

本文档提供了 5 个适用于 8 位 AVR 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）。有关 CCI 的更多信息，请参见“*MPLAB® XC8 C Compiler User's Guide for AVR® MCU*”（DS50002750）。

读者需要掌握一些单片机和 C 编程语言的相关知识。

目录

简介.....	1
1. 点亮或熄灭 LED.....	4
1.1. 配置位.....	4
1.2. 包含头文件.....	5
1.3. 用于 LED 的端口的访问.....	5
2. 使用延迟函数使 LED 闪烁.....	7
2.1. While()循环和位翻转功能.....	7
2.2. 延迟函数.....	7
3. 通过按下按钮和中断来翻转 LED.....	8
3.1. 用于按钮的端口的访问.....	8
3.2. 引脚电平变化中断.....	9
4. 如果电位器值低于 ADC 值, 则点亮 LED.....	10
4.1. MCC 系统资源配置.....	10
4.2. MCC ADC 资源配置.....	11
4.3. MCC GPIO 引脚资源配置.....	14
4.4. MCC 引脚资源配置.....	15
4.5. MCC 中断管理器资源配置.....	15
4.6. MCC 代码生成.....	16
4.7. 修改后的 main.c 代码.....	17
5. EE 数据读写操作后 LED 闪烁.....	20
5.1. MCC 系统资源配置.....	20
5.2. MCC 存储器资源配置.....	21
5.3. MCC GPIO 引脚资源配置.....	22
5.4. MCC 引脚资源配置.....	23
5.5. MCC 代码生成.....	24
5.6. 修改后的 main.c 代码.....	25
6. 在 MPLAB X IDE 中运行代码.....	27
6.1. 创建项目:	27
6.2. 选择通用 C 接口 (CCI)	27
6.3. 调试示例.....	27
7. 获取硬件和软件.....	28
8. 其他信息.....	29
Microchip 网站.....	30
产品变更通知服务.....	30
客户支持.....	30
Microchip 器件代码保护功能.....	30

1. 点亮或熄灭 LED

本示例将点亮 ATmega4809 Curiosity Nano 开发板上的用户 LED。有关更多信息，请参见 [7. 获取硬件和软件](#) 一节。

```
/*
 * File:    main.c
 * Author:  Microchip Technology Inc.
 *
 * Created on July 28, 2020 9:55 AM
 */

// ATmega4809 Configuration Bit Settings

// 'C' source line config statements

#include <xc.h>

FUSES = {
    .WDTCFG = 0x00, // WDTCFG {PERIOD=OFF, WINDOW=OFF}
    .BODCFG = 0x00, // BODCFG {SLEEP=DIS, ACTIVE=DIS, SAMPFREQ=1KHZ, LVL=BODLEVEL0}
    .OSCCFG = 0x02, // OSCCFG {FREQSEL=20MHZ, OSCLOCK=CLEAR}
    .SYSCFG0 = 0xC0, // SYSCFG0 {EESAVE=CLEAR, RSTPINCFG=GPIO, CRCSRC=NOCRC}
    .SYSCFG1 = 0x07, // SYSCFG1 {SUT=64MS}
    .APPEND = 0x00, // APPEND
    .BOOTEND = 0x00, // BOOTEND
};

LOCKBITS = 0xC5; // {LB=NOLOCK}

int main(void) {

    PORTF.DIRSET = PIN5_bm; // set PF5 to be output

    PORTF.OUTCLR = PIN5_bm; // clear PF5 - LED on

    //PORTF.OUTSET = PIN5_bm; // set PF5 - LED off

    while (1) {
    }

    return(0);
}
```

1.1 配置位

Microchip 器件具有配置位（或熔丝），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：

- 振荡器配置——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：

```
FUSES = { ...
    .OSCCFG = 0x02, // OSCCFG {FREQSEL=20MHZ, OSCLOCK=CLEAR}
    ... }
```

- 看门狗定时器配置——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：

```
FUSES = {
    .WDTCFG = 0x00, // WDTCFG {PERIOD=OFF, WINDOW=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：
LOCKBITS = 0xC5; // {LB=NOLOCK}

注：您也可以使用 `#pragma config` 设置配置位。有关详细信息，请参见“*MPLAB® XC8 C Compiler User's Guide for AVR® MCU*”（DS50002750）。

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 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：

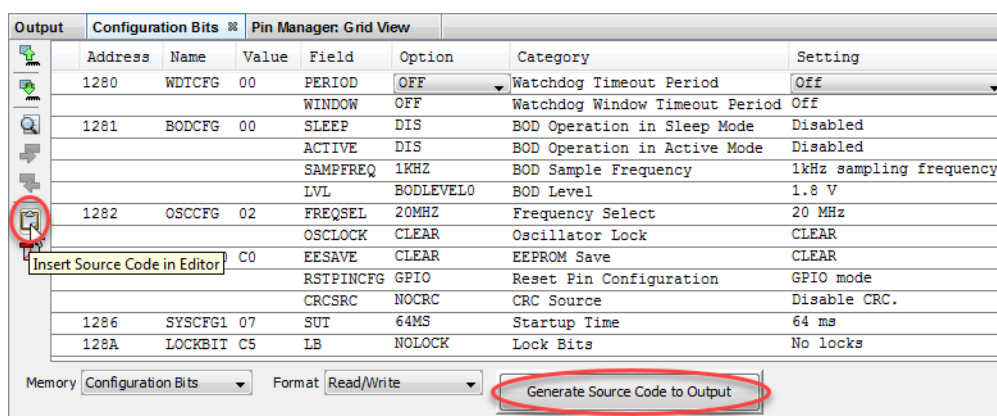
MPLAB XC8 Installation Directory/docs/avr_chipinfo.html

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。

对于 AVR 器件，单击  以读取配置存储器，即 FUSES 和 LOCKBITS 值。

图 1-1. 配置位窗口



Address	Name	Value	Field	Option	Category	Setting
1280	WDTCFG	00	PERIOD	OFF	Watchdog Timeout Period	Off
			WINDOW	OFF	Watchdog Window Timeout Period	Off
1281	BODCFG	00	SLEEP	DIS	BOD Operation in Sleep Mode	Disabled
			ACTIVE	DIS	BOD Operation in Active Mode	Disabled
			SAMPFREQ	1KHZ	BOD Sample Frequency	1kHz sampling frequency
			LVL	BODLEVEL0	BOD Level	1.8 V
1282	OSCCFG	02	FREQSEL	20MHZ	Frequency Select	20 MHz
			OSCCLOCK	CLEAR	Oscillator Lock	CLEAR
			EESAVE	CLEAR	EEPROM Save	CLEAR
			RSTPINCFG	GPIO	Reset Pin Configuration	GPIO mode
			CRCSRC	NOCRC	CRC Source	Disable CRC.
1286	SYSCFG1	07	SUT	64MS	Startup Time	64 ms
128A	LOCKBIT	C5	LB	NOLOCK	Lock Bits	No locks

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。或者单击 **Generate Source Code to Output**（生成源代码并输出）按钮，将其复制并粘贴至代码中。

关于此窗口的更多信息，请参见 MPLAB X IDE 文档。

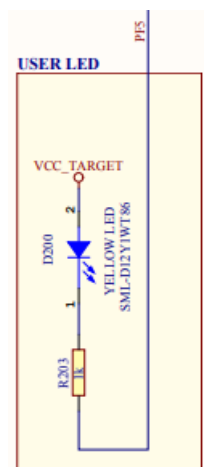
1.2 包含头文件

`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 `avr/io.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。

可在 MPLAB XC8 安装目录的 `avr/avr/include/avr` 或 `dfp/xc8/avr/include/avr` 子目录下，或从通过 `-mdfp` 选项指定的路径，找到本头文件和其他头文件。

1.3 用于 LED 的端口的访问

每个 I/O 引脚 `Pxn` 都可以由 `PORTx` 中的寄存器控制。每个引脚组 `x` 都有自己的一组 `PORT` 寄存器。在本示例中，`PORTF` 的 `PF5` 用于点亮或熄灭用户 LED。查看开发板原理图可知，`PF5` 必须为低电平才能使 LED 点亮，必须为高电平才能使 LED 熄灭。在工具包窗口中找到原理图链接。



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。

要将 PF5 仅用作输出引脚，向 `PORTF.DIRSET` 寄存器的 bit 5 写入“1”。为进行此操作而不影响其他位的值，为每个寄存器位提供了掩码。在本示例中，`PIN5_bm = 00001000`。

```
PORTF.DIRSET = PIN5_bm; // set PF5 to be output
```

向 `PORTF.OUTCLR` 中的 bit 5 写入“1”，会使该端口位清零（置为“0”）。这将点亮 LED。

```
PORTF.OUTCLR = PIN5_bm; // clear PF5 - LED on
```

或者，向 `PORTF.OUTSET` 中的 bit 5 写入“1”会将该端口位置 1（置为“1”）。若要熄灭 LED，需注释掉前一条指令，然后取消注释掉后一条指令。

```
PORTF.OUTSET = PIN5_bm; // set PF5 - LED off
```

要查看有关 PORT 寄存器的信息，在编辑器中高亮选中一个寄存器名称，右键单击，然后选择 [Navigate>Online Datasheet](#)（导航>在线数据手册）。

2. 使用延迟函数使 LED 闪烁

本示例在上一示例代码的基础上进行修改。此代码不仅点亮或熄灭用户 LED，还会使 LED 闪烁。

```
/*
 * File:    main.c
 * Author: Microchip Technology Inc.
 *
 * Created on July 28, 2020 10:34 AM
 */

// ATmega4809 Configuration Bit Settings

// 'C' source line config statements

// After any reset, CLR_PER = CLK_MAIN/Prescaler = 20MHz / 6 = 3.3MHz
#define F_CPU (3300000UL)

#include <xc.h>
#include <util/delay.h>

FUSES = {
    .WDTCFG = 0x00, // WDTCFG {PERIOD=OFF, WINDOW=OFF}
    .BODCFG = 0x00, // BODCFG {SLEEP=DIS, ACTIVE=DIS, SAMPFREQ=1KHZ, LVL=BODLEVEL0}
    .OSCCFG = 0x02, // OSCCFG {FREQSEL=20MHZ, OSCLOCK=CLEAR}
    .SYSCFG0 = 0xC0, // SYSCFG0 {EESAVE=CLEAR, RSTPINCFG=GPIO, CRCSRC=NOCRC}
    .SYSCFG1 = 0x07, // SYSCFG1 {SUT=64MS}
    .APPEND = 0x00, // APPEND
    .BOOTEND = 0x00, // BOOTEND
};

LOCKBITS = 0xC5; // {LB=NOLOCK}

int main(void) {

    PORTF.DIRSET = PIN5_bm; // set PF5 to be output

    while (1) {
        PORTF.OUTTGL = PIN5_bm; // toggle PF5
        _delay_ms(500);
    }

    return(0);
}
```

2.1 While()循环和位翻转功能

向 PORTF.OUTTGL 中的一个位写入“1”将使该位翻转状态。对于 PF5:

```
PORTF.OUTTGL = PIN5_bm; // toggle PF5
```

若要保持翻转引脚并使 LED 闪烁，则需使用 while(1) 循环。

2.2 延迟函数

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_ms() 是编译器的内置函数。要使用此函数，必须包含头文件 util/delay.h。另外，必须指定处理器的速度：

```
#define F_CPU (3300000UL)
```

有关内置延迟函数的更多详细信息，请参见“*MPLAB® XC8 C Compiler User's Guide for AVR® MCU*” (DS50002750)。

3. 通过按下按钮和中断来翻转 LED

本示例在上一示例代码的基础上进行修改。这次，将通过单击用户按钮点亮或熄灭用户 LED。单击按钮时，将使用中断来翻转 LED 状态。

```

/*
 * File:    main.c
 * Author: Microchip Technology Inc.
 *
 * Created on August 3, 2020 10:12 AM
 */

// ATmega4809 Configuration Bit Settings

// 'C' source line config statements

#include <xc.h>

FUSES = {
    .WDTCFG = 0x00, // WDTCFG {PERIOD=OFF, WINDOW=OFF}
    .BODCFG = 0x00, // BODCFG {SLEEP=DIS, ACTIVE=DIS, SAMPFREQ=1KHZ, LVL=BODLEVEL0}
    .OSCCFG = 0x02, // OSCCFG {FREQSEL=20MHZ, OSCLOCK=CLEAR}
    .SYSCFG0 = 0xC0, // SYSCFG0 {EESAVE=CLEAR, RSTPINCFG=GPIO, CRCSRC=NOCRC}
    .SYSCFG1 = 0x07, // SYSCFG1 {SUT=64MS}
    .APPEND = 0x00, // APPEND
    .BOOTEND = 0x00, // BOOTEND
};

LOCKBITS = 0xC5; // {LB=NOLOCK}

// Interrupt function
void __interrupt(PORTF_PORT_vect_num) btnInt(void)
{
    if(PORTF.INTFLAGS == PIN6_bm) // check PF6 interrupt
    {
        PORTF.OUTTGL = PIN5_bm; // toggle LED

        PORTF.INTFLAGS = PIN6_bm; // clear interrupt
    }
}

int main(void)
{
    //LED init
    PORTF.DIRSET = PIN5_bm; // set PF5 to be output
    PORTF.OUTSET = PIN5_bm; // set PF5 - LED off

    //BUTTON init
    //Reset value of all PORTF pins is '0', which is input
    PORTF.PIN6CTRL = PORT_PULLUPEN_bm | PORT_ISC_FALLING_gc; //enable pullups on PF6, IRQ on falling edge

    ei(); //enable global interrupts

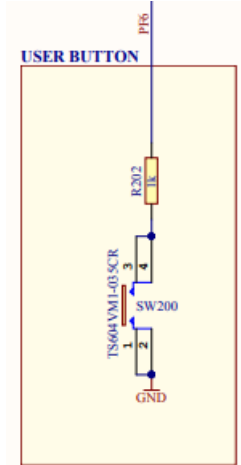
    while (1) {
        //wait for button press
    }

    return 0;
}

```

3.1 用于按钮的端口的访问

在本示例中，PORTF 的 PF6 用于检测是否按下了按钮。查看开发板原理图可知，需要使能端口的内部上拉，这样按下按钮时 PF6 将从“1”变为“0”。



虽然您可以使用 `DIRCLR` 将 PF6 设置为输入引脚，但 `PORTx` 的复位值为 `0x00`，这会将所有引脚设置为输入。

通过 `PINCTRL` 使能上拉并将 PF6 配置为用于输入/检测。中断检测的配置决定如何触发端口中断。

```
PORTF.PIN6CTRL = PORT_PULLUPEN_bm | PORT_ISC_FALLING_gc;
```

3.2 引脚电平变化中断

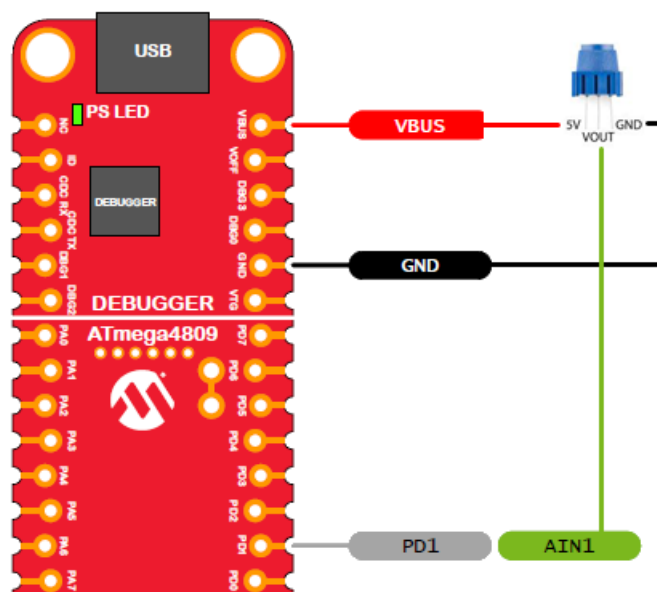
在 `main()` 代码中，`ei()` 允许全局中断。

`__interrupt(vector)` 指定 `btnInt()` 为中断函数。端口中断向量的格式为 `PORTx_Port_vect_num`（见 `iom4809.h`）。为确保仅 PF6 电平变化触发中断，检查该中断标志。然后翻转 LED 状态，清零该中断标志。向该中断标志所在位写入“1”将使该标志清零。

4. 如果电位器值低于 ADC 值，则点亮 LED

这个 ADC 窗口比较器示例将演示如何初始化 ADC，设置转换窗口比较器的低阈值，使能转换窗口模式，使能自由运行模式，开始转换，然后等待转换完成后，如果 ADC 结果低于设定的阈值，则点亮 LED，如果该结果高于阈值，则熄灭 LED。使用一个电位器作为模拟源。

图 4-1. ATmega4809 Curiosity Nano 开发板上的 ADC 连接



有关本示例和其他 ADC 示例的更多信息，请参见 TB3209: “Getting Started with ADC” (DS90003209)。

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 v3.xx 用户指南》(DS40001829B_CN))，请访问 MPLAB 代码配置器网页：

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

本示例中的 MCC UI 设置如以下各小节所示。

4.1 MCC 系统资源配置

图 4-2. 项目资源——系统模块

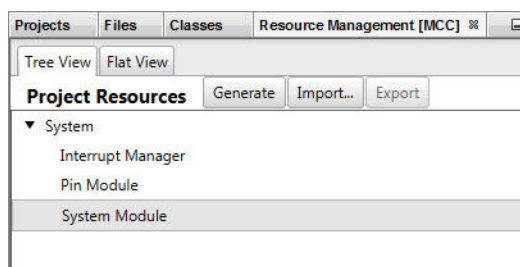
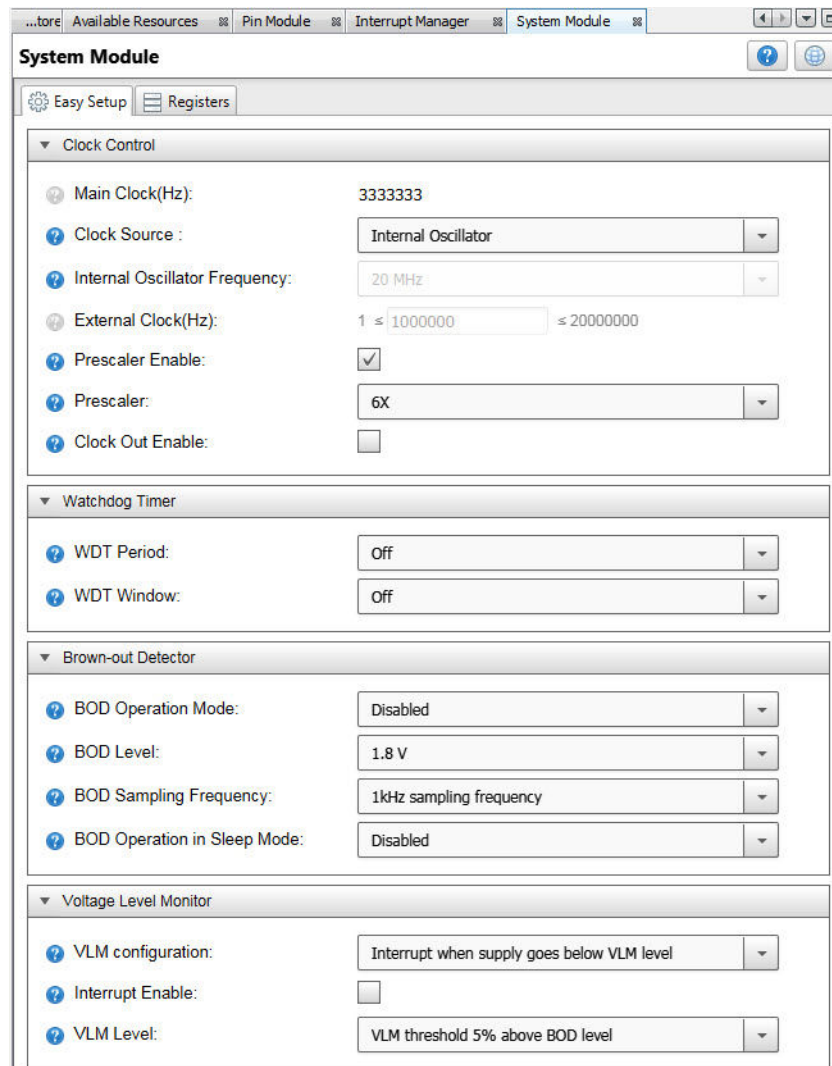


图 4-3. 系统模块配置



4.2 MCC ADC 资源配置

虽然此处只显示了 ADC 资源配置的“Easy Setup”（轻松设置）选项卡，您还应当查看“Registers”（寄存器）选项卡，以了解未显示的设置（MUXPOS）和备用数据输入。

图 4-4. ADC 资源选择

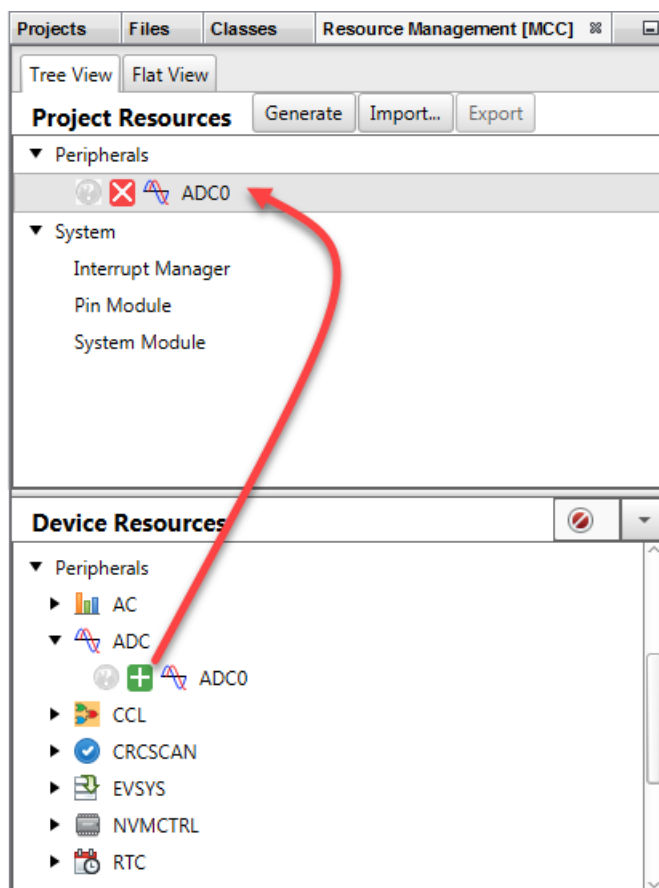


图 4-5. ADC 资源配置

...age Available Resources Pin Module Interrupt Manager System Module ADC0

ADC0

Easy Setup Registers

Software Settings

API Prefix: ADC0

Result Selection: 10-bit mode

Hardware Settings

Enable ADC: ☒

Sampling Frequency(Hz): 18939 ≤ 59523 ≤ 64102

ADC Clock(Hz): 833333

Sample Accumulation Number: 1 ADC sample

Sample Length (# of ADC Clock): 0 ≤ 1 ≤ 31

Voltage Reference: Internal reference

Interrupt Settings

Result Ready Interrupt Enable: ☒

WCMP Interrupt Enable: ☒

Select Channels

Pin	Channel	Custom Name
PIN_AIN1	PIN_AIN1	channel_PIN_AIN1
Internal Channel	0V_(GND)	channel_0V (GND)
Internal Channel	Temperature_sensor	channel_Temperature sensor
Internal Channel	AC_DAC_Reference	channel_AC DAC Reference

Window Settings

Window Comparator Mode: Below Window

Enable IRQ on conversion complete: ☒

Enable IRQ on conversion satisfying window criteria: ☒

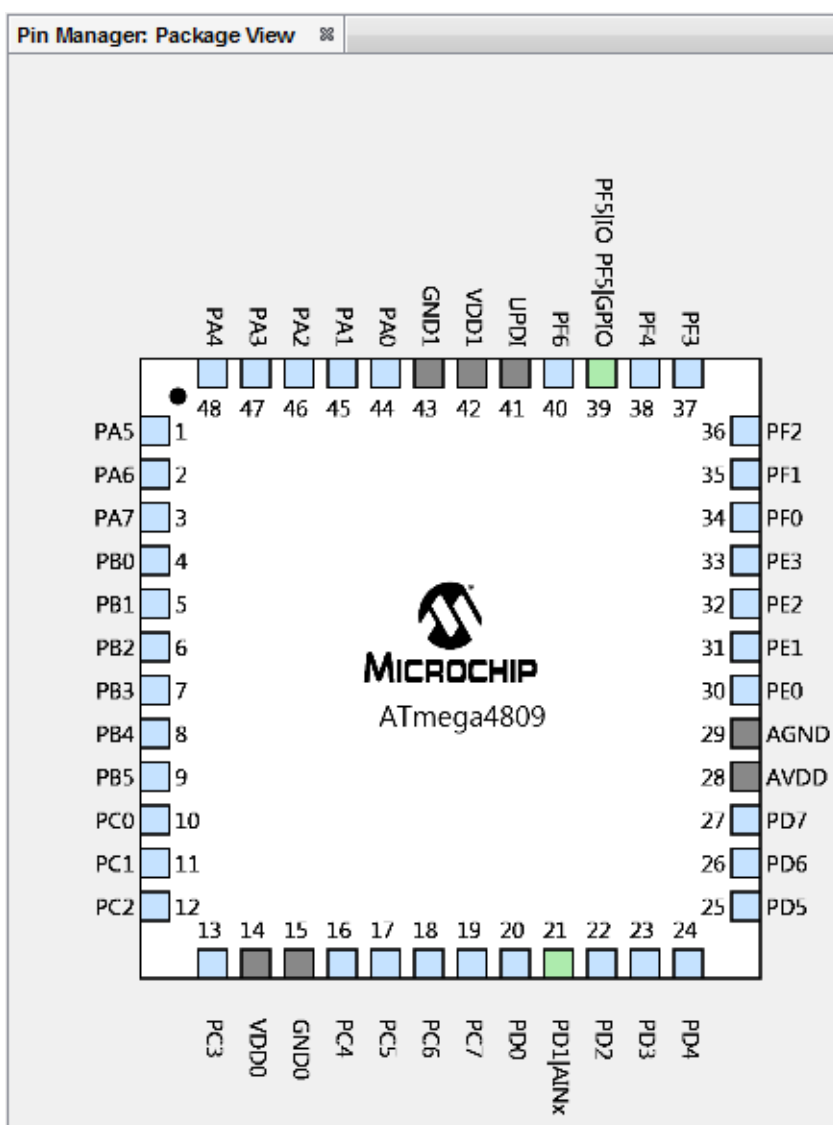
Window Comparator Low Threshold(V): 0 ≤ 512 ≤ 65535

4.3 MCC GPIO 引脚资源配置

图 4-6. GPIO 引脚资源——网格

Search Results				Output		Variables		Call Stack		Breakpoints		Notifications [MCC]		Pin Manager: Grid View																																	
Package:	QFP48		Pin No:		44	45	46	47	48	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	30	31	32	33	34	35	36	37	38	39	40
				Port A ▼						Port B ▼						Port C ▼						Port D ▼						Port E ▼						Port F ▼													
Module	Function	Direction	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	0	1	2	3	4	5	6		
ADC0	AINx	input																																													
CLKCTRL ▼	CLK1	input																																													
	CLK0	output																																													
	TOSC1	input																																													
	TOSC2	input																																													
Pin Module ▼	GPIO	input																																													
	GPIO	output																																													
RSTCTRL	RESET	input																																													

图 4-7. GPIO 引脚资源——封装



4.4 **MCC 引脚资源配置**

图 4-8. 项目资源——引脚模块

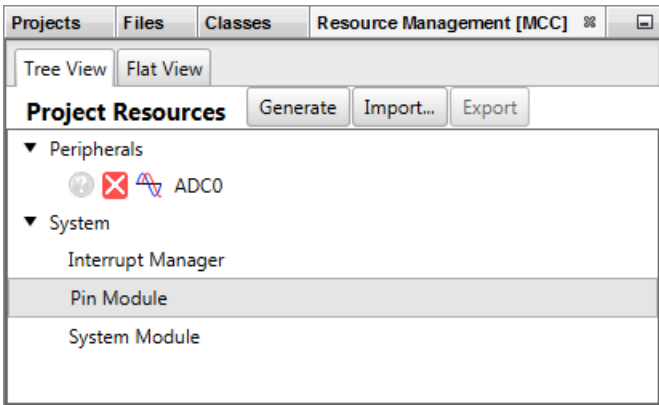
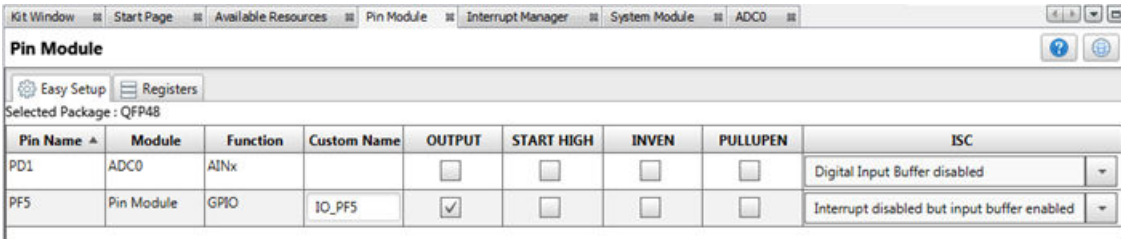


图 4-9. 引脚模块配置



4.5 **MCC 中断管理器资源配置**

图 4-10. 项目资源——中断管理器

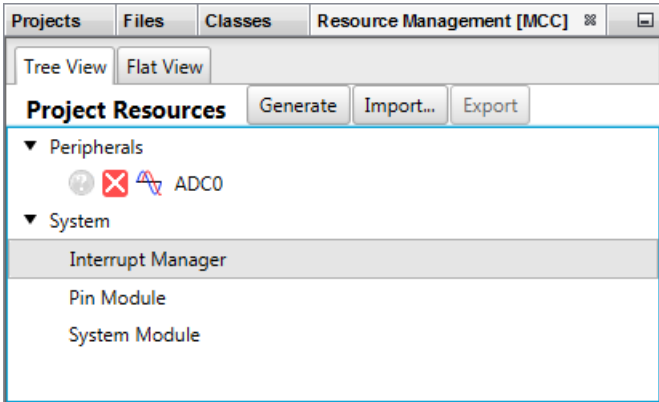


图 4-11. 中断管理器配置

...age Available Resources Pin Module Interrupt Manager System Module ADC0

Interrupt Manager

Easy Setup Registers

▼ Interrupt Setting

Global Interrupt Enable: ☐

▼ Interrupt Priority

Round-robin Scheduling Enable: ☐

Interrupt Level Priority: 0

Interrupt Vector with High Priority: 0

▼ Interrupt Vector

Compact Vector Table Enable: ☐

Interrupt Vector Select Enable: ☐

▼ Module Interrupts

Module	Interrupt	Enable
BOD	VLM	<input type="checkbox"/>
ADC0	RESRDY	<input checked="" type="checkbox"/>
ADC0	WCMP	<input checked="" type="checkbox"/>

4.6 MCC 代码生成

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 4-12. 在项目树中生成的代码

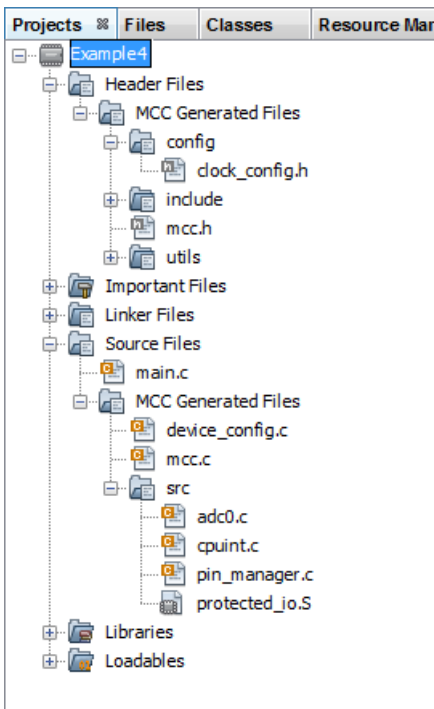


图 4-13. Output（输出）窗口中的代码生成进度

```
MPLAB® Code Configurator PKOB nano-ADC_MCC_NANO ADC_MCC_NANO (Build, Load, ...)
18:59:43.259 INFO: *****
18:59:43.260 INFO: Generation Results
18:59:43.260 INFO: *****
18:59:43.277 INFO: main.c Success.
18:59:43.277 INFO: mcc_generated_files\config\clock_config.h Success.
18:59:43.279 INFO: mcc_generated_files\device_config.c Success.
18:59:43.279 INFO: mcc_generated_files\include\adc0.h Success.
18:59:43.279 INFO: mcc_generated_files\include\ccp.h Success.
18:59:43.280 INFO: mcc_generated_files\include\cpuint.h Success.
18:59:43.280 INFO: mcc_generated_files\include\pin_manager.h Success.
18:59:43.280 INFO: mcc_generated_files\include\port.h Success.
18:59:43.280 INFO: mcc_generated_files\include\protected_io.h Success.
18:59:43.281 INFO: mcc_generated_files\include\rstctrl.h Success.
18:59:43.281 INFO: mcc_generated_files\mcc.c Success. Auto-merged.
18:59:43.281 INFO: mcc_generated_files\mcc.h Success. Auto-merged.
18:59:43.281 INFO: mcc_generated_files\src\adc0.c Success.
18:59:43.282 INFO: mcc_generated_files\src\cpuint.c Success.
18:59:43.282 INFO: mcc_generated_files\src\pin_manager.c Success.
18:59:43.282 INFO: mcc_generated_files\src\protected_io.S Success.
18:59:43.282 INFO: mcc_generated_files\utils\assembler.h Success.
18:59:43.283 INFO: mcc_generated_files\utils\assembler\gas.h Success.
18:59:43.283 INFO: mcc_generated_files\utils\assembler\iar.h Success.
18:59:43.283 INFO: mcc_generated_files\utils\atomic.h Success.
18:59:43.283 INFO: mcc_generated_files\utils\compiler.h Success.
18:59:43.284 INFO: mcc_generated_files\utils\interrupt_avr8.h Success.
18:59:43.284 INFO: mcc_generated_files\utils\utils.h Success.
18:59:43.284 INFO: mcc_generated_files\utils\utils_assert.h Success.
18:59:43.332 INFO: *****
18:59:43.332 INFO: Generation complete (total time: 1771 milliseconds)
18:59:43.332 INFO: *****
```

4.7 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。

注：<xc.h>通过“mcc_generated_files/mcc.h”自动包含。

```
/*
(c) 2018 Microchip Technology Inc. and its subsidiaries.
```

```

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

#include "mcc_generated_files/mcc.h"
adc_0_channel_t channel = ADC_MUXPOS_AIN1_gc;

/*
    Main application
*/
int main(void)
{
    /* Initializes MCU, drivers and middleware */
    SYSTEM_Initialize();

    //Enable ADC and start conversion
    ADC0_Enable();
    ADC0_StartConversion(channel);

    while (1){
        if (ADC0_IsConversionDone())
        {
            if(ADC0_GetWindowResult())
            {
                PORTF.OUTCLR = PIN5_bm; // clear PF5 - LED on

            }
            else
            {
                PORTF.OUTSET = PIN5_bm; // set PF5 - LED off
            }
        }
    }
}
/**
    End of File
*/

```

4.7.1 ADC 相关变量

Channel 为 ADC0_StartConversion() 函数所需的变量。

```
adc_0_channel_t channel = ADC_MUXPOS_AIN1_gc;
```

在 adc0.h 中，adc_0_channel_t 定义为 ADC_MUXPOS_t 类型。

```
typedef ADC_MUXPOS_t adc_0_channel_t;
```

在器件特定的 io.h 文件（在本示例中为 iom4809.h）中，声明 ADC_MUXPOS_t（为简洁起见，略去了 ADC_MUXPOS_AIN2_gc 至 ADC_MUXPOS_AIN14_gc）。

```

/* Analog Channel Selection Bits select */
typedef enum ADC_MUXPOS_enum
{
    ADC_MUXPOS_AIN0_gc = (0x00<<0), /* ADC input pin 0 */
    ADC_MUXPOS_AIN1_gc = (0x01<<0), /* ADC input pin 1 */
    :
    ADC_MUXPOS_AIN15_gc = (0x0F<<0), /* ADC input pin 15 */
    ADC_MUXPOS_DACREF_gc = (0x1C<<0), /* AC DAC Reference */
    ADC_MUXPOS_TEMPSENSE_gc = (0x1E<<0), /* Temperature sensor */
    ADC_MUXPOS_GND_gc = (0x1F<<0), /* 0V (GND) */
} ADC_MUXPOS_t;

```

4.7.2 ADC 窗口比较

用于执行 ADC 操作和比较的函数可在 adc0.c 文件中找到。

- ADC0_Enable()——使能 ADC 模块。
- ADC0_StartConversion(channel)——开始 ADC 转换。

如果电位器值低于 **ADC** 值，则点亮 **LED**

- `ADC0_IsConversionDone()` ——在 `while()` 循环中，检查转换何时完成。
- `ADC0_GetWindowResult()` ——如果转换结果低于阈值，则该值为真（点亮 **LED**）；否则，该值为假（熄灭 **LED**）。

5. EE 数据读写操作后 LED 闪烁

本示例将演示如何对 EEPROM 数据（EE 数据）存储器进行读写操作。读写操作成功完成后，LED 闪烁。要在写入前后查看 EEPROM 存储器，打开 **Window>Target Memory Views>EEPROM Memory**（窗口>目标存储器视图

>EEPROM 存储器），然后单击“Read Device Memory”（读取器件存储器）。

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：

4. 如果电位器值低于 ADC 值，则点亮 LED

本示例中的 MCC GUI 设置如以下各小节所示。

5.1 MCC 系统资源配置

图 5-1. 项目资源——系统模块

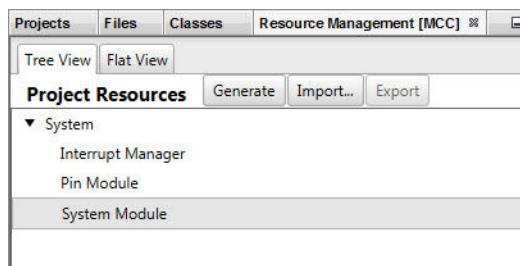
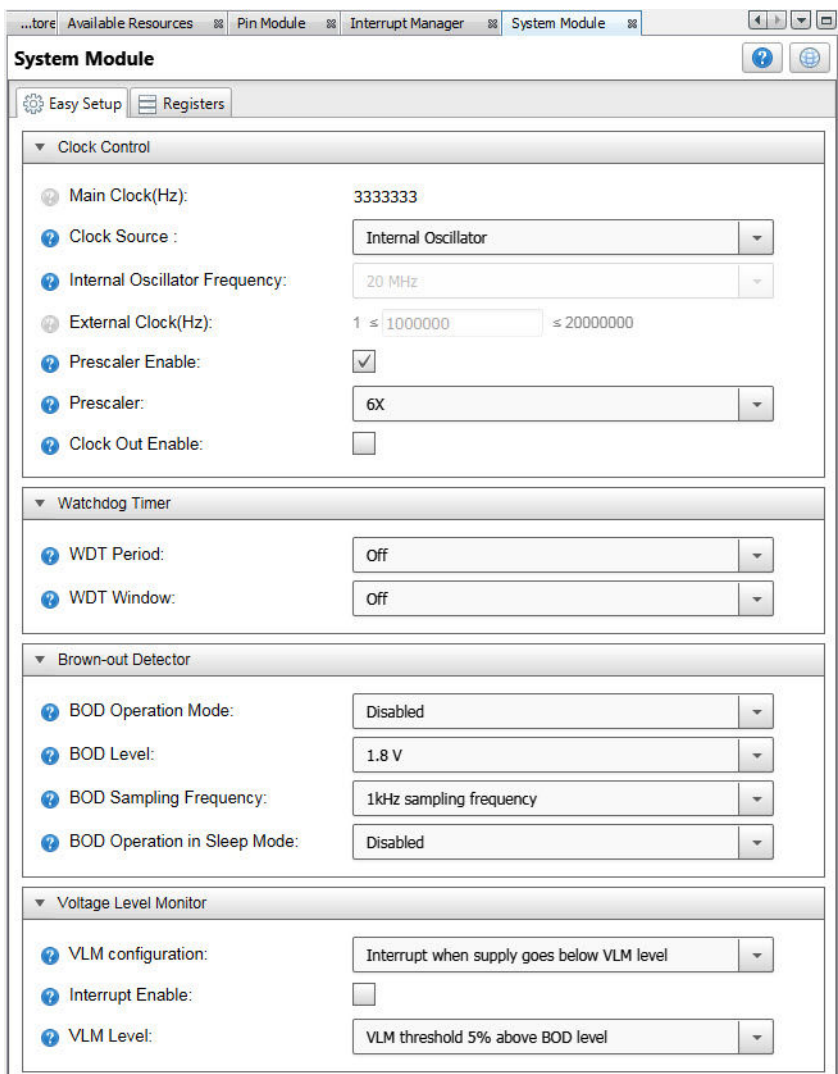


图 5-2. 系统模块配置



5.2 MCC 存储器资源配置

要将 EE 数据添加到项目资源：

1. 在 **Device Resources**（器件资源）下，找到并展开 **NVMCTRL**（非易失性存储器控制）。
2. 单击绿色加号，将其添加到 **Project Resources** 下。
3. 单击 **NVMCTRL** 查看资源配置设置。对于本示例，将不做任何更改。

图 5-3. NVMCTRL (EE 数据) 资源选择

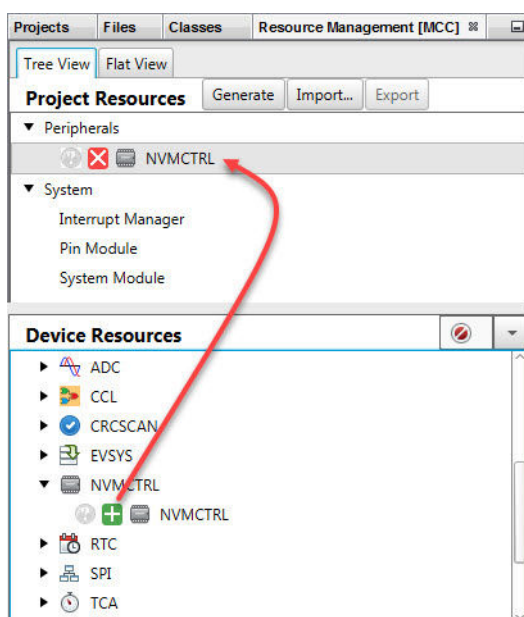
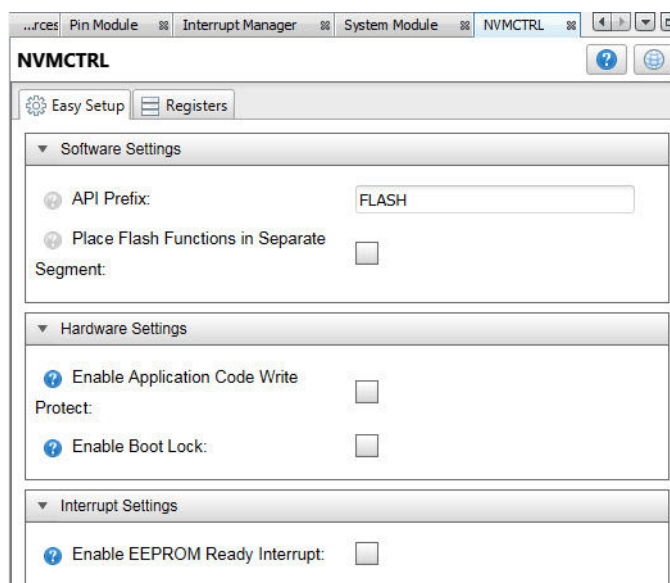


图 5-4. NVMCTRL (EE 数据) 资源配置



5.3 MCC GPIO 引脚资源配置

为了使 LED 闪烁，端口 B 引脚 5 必须设置为输出。

图 5-5. GPIO 引脚资源——网格











































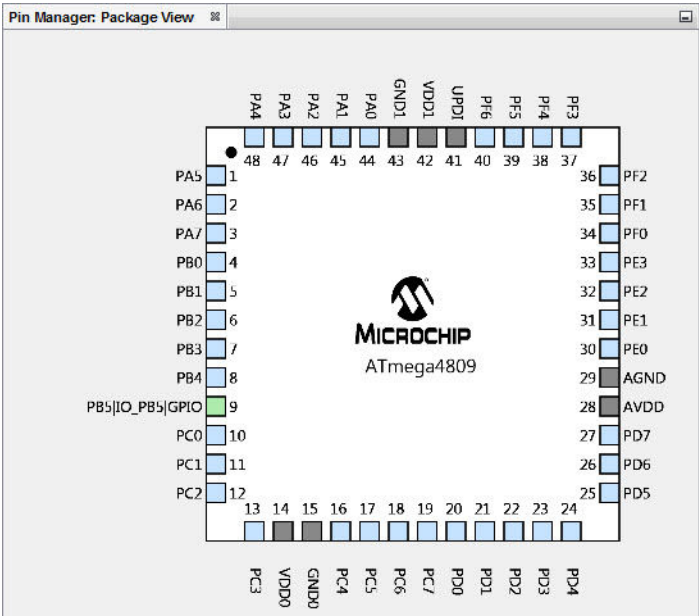
Output	Configuration Bits		Pin Manager: Grid View 																			
Package:	QFP48		Pin No:	44	45	46	47	48	1	2	3	4	5	6	7	8	9	10	11	12	13	16
			Port A ▼							Port B ▼							Port C ▼					
Module	Function	Direction	0	1	2	3	4	5	6	7	0	1	2	3	4	5	0	1	2	3	4	
CLKCTRL ▼	CLKI	input																				
	CLKO	output																				
	TOSC1	input																				
	TOSC2	input																				
Pin Module ▼	GPIO	input																				
	GPIO	output																				
RSTCTRL	RESET	input																				

图 5-6. GPIO 引脚资源——封装



5.4 MCC 引脚资源配置

在“Project Resources”下，单击“Pin Module”（引脚模块）查看引脚模块配置设置。

Pin Module 窗口中显示 PB5，这是因为已在 Pin Manager: Grid View（引脚管理器：网格视图）窗口中选择了该引脚。对于本示例，无需对引脚配置进行任何更改。

图 5-7. 项目资源——引脚模块

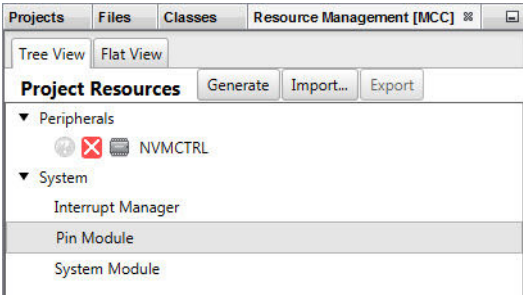
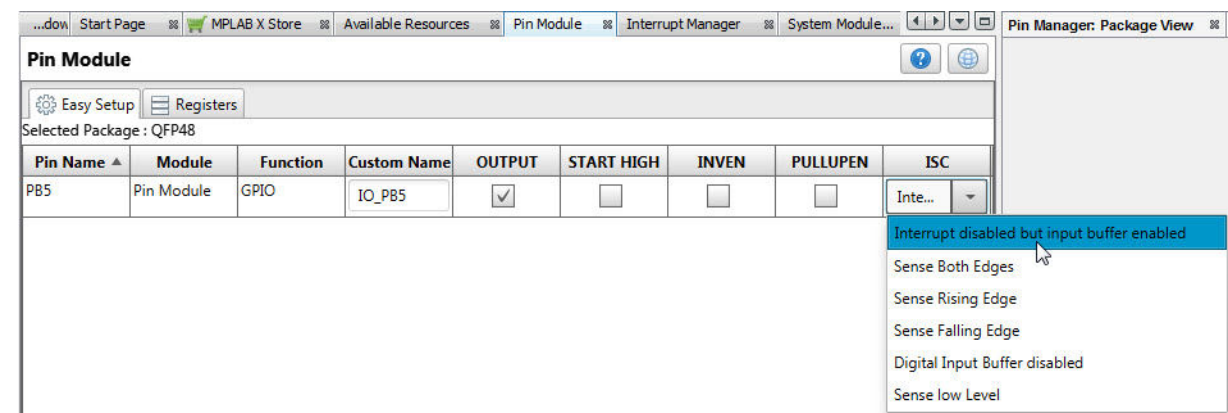


图 5-8. 引脚模块配置



5.5 MCC 代码生成

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。

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 5-9. 在项目树中生成的代码

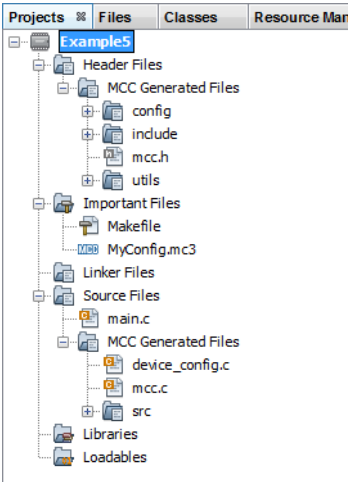


图 5-10. Output 窗口中的代码生成

Out...	Configuration Bits	Pin Manager: Grid View
Kits	MPLAB® Code Configurator	
11:22:35.780	INFO: *****	
11:22:35.781	INFO: Generation Results	
11:22:35.781	INFO: *****	
11:22:35.798	INFO: main.c	Success. New file.
11:22:35.799	INFO: mcc_generated_files\config\clock_config.h	Success. New file.
11:22:35.799	INFO: mcc_generated_files\device_config.c	Success. New file.
11:22:35.799	INFO: mcc_generated_files\include\ccp.h	Success. New file.
11:22:35.800	INFO: mcc_generated_files\include\cpuint.h	Success. New file.
11:22:35.800	INFO: mcc_generated_files\include\nvmctrl.h	Success. New file.
11:22:35.800	INFO: mcc_generated_files\include\pin_manager.h	Success. New file.
11:22:35.801	INFO: mcc_generated_files\include\port.h	Success. New file.
11:22:35.801	INFO: mcc_generated_files\include\protected_io.h	Success. New file.
11:22:35.801	INFO: mcc_generated_files\include\rstctrl.h	Success. New file.
11:22:35.802	INFO: mcc_generated_files\mcc.c	Success. New file.
11:22:35.802	INFO: mcc_generated_files\mcc.h	Success. New file.
11:22:35.802	INFO: mcc_generated_files\src\cpuint.c	Success. New file.
11:22:35.803	INFO: mcc_generated_files\src\nvmctrl.c	Success. New file.
11:22:35.803	INFO: mcc_generated_files\src\pin_manager.c	Success. New file.
11:22:35.803	INFO: mcc_generated_files\src\protected_io.S	Success. New file.
11:22:35.803	INFO: mcc_generated_files\utils\assembler.h	Success. New file.
11:22:35.804	INFO: mcc_generated_files\utils\assembler\gas.h	Success. New file.
11:22:35.804	INFO: mcc_generated_files\utils\assembler\iar.h	Success. New file.
11:22:35.804	INFO: mcc_generated_files\utils\atomic.h	Success. New file.
11:22:35.805	INFO: mcc_generated_files\utils\compiler.h	Success. New file.
11:22:35.805	INFO: mcc_generated_files\utils\interrupt_avr8.h	Success. New file.
11:22:35.805	INFO: mcc_generated_files\utils\utils.h	Success. New file.
11:22:35.806	INFO: mcc_generated_files\utils\utils_assert.h	Success. New file.
11:22:35.840	INFO: *****	
11:22:35.858	INFO: Generation complete (total time: 1951 milliseconds)	
11:22:35.858	INFO: *****	

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。

```

/*
    (c) 2018 Microchip Technology Inc. and its subsidiaries.

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

#include "mcc_generated_files/mcc.h"
#include <util/delay.h>

#define LED_ON_OFF_DELAY 500
#define NUM_EE_VALUES 8
#define EE_ADR_START 8

eeprom_adr_t ee_address;
nvmctrl_status_t status;
volatile uint8_t RAMArray[NUM_EE_VALUES];

/*
    Main application
*/
int main(void)
{
    /* Initializes MCU, drivers and middleware */
    SYSTEM_Initialize();

    /* Declare loop variable */
    uint8_t i;

    if (!FLASH_Initialize()) {

        ee_address = EE_ADR_START;

        // Write EEPROM Data
        for(i=0; i<NUM_EE_VALUES; i++){
            status = FLASH_WriteEepromByte(ee_address, i);
            ee_address++;
        }
    }
}

```

```

    ee_address = EE_ADR_START;

    // Read EEPROM Data
    for (i=0; i<NUM_EE_VALUES; i++){
        RAMArray[i] = FLASH_ReadEepromByte(ee_address);
        ee_address++;
    }

}

while (1){
    PORTF.OUTTGL = PIN5_bm; // toggle PB5
    _delay_ms(LED_ON_OFF_DELAY);
}
}
/**
    End of File
*/

```

5.6.1 EE 数据相关变量

用于存储 EE 数据存储器读/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 `nvmctrl.h` 中找到：

```

uint8_t FLASH_ReadEepromByte(eeprom_adr_t eeprom_adr);
nvmctrl_status_t FLASH_WriteEepromByte(eeprom_adr_t eeprom_adr, uint8_t data);

```

从 `stdint.h` 引用时，`uint8_t` 与 `unsigned char` 相同。

5.6.2 写入 EE 数据

在本示例中，数据会被写入 EE 数据，然后被读回。

将一个字节数据写入 EE 数据存储器的函数 `FLASH_WriteEepromByte()` 可在 `nvmctrl.c` 中找到。此函数中包括一个循环，该循环会等待之前所有写操作完成后，才开始下一次写操作。因此，无需检查写操作是否完成，即无需使用 `FLASH_IsEepromReady()`。

5.6.3 读取 EE 数据

写入 EE 数据后，存储器值会被读入 RAM 数组。将一个字节数据读取到 EE 数据存储器的函数 `FLASH_ReadEepromByte()` 可在 `nvmctrl.c` 中找到。


读取值后，`while` 循环使 LED 闪烁，以指示程序成功完成。

6. 在 MPLAB X IDE 中运行代码

请按照以下说明在 MPLAB X IDE 中执行示例代码。

6.1 创建项目：

1. 启动 MPLAB X IDE。

2. 从 IDE 中启动新建项目向导 。

按照屏幕提示创建一个新项目：

1. **Choose Project (选择项目)**：选择 “Microchip Embedded” (Microchip 嵌入式)，然后选择 “Standalone Project” (独立项目)。
2. **Select Device and Tool (选择器件和工具)**：选择示例器件。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。
3. **Select Header (选择调试头)**：无。
4. **Select Plugin Board (选择接插板)**：无。
5. **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>Embedded>Build Tools (工具>选项>已安装工具>编译工具)**)。有关详细信息，请参见 MPLAB XC8 和 MPLAB X IDE 文档。
6. **Select Project Name and Folder (选择项目名和文件夹)**：为项目命名。



6.2 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。

6.3 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：
 - 1.1. 右键单击 **Projects** 窗口中的 “Source Files” (源文件) 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。
 - 1.2. 在 “File name” (文件名) 下输入名称 (如 **examplen**)，其中 *n* 为示例编号。
 - 1.3. 单击 **Finish (完成)**。将在编辑器窗口中打开文件。
 - 1.4. 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 Project (调试项目)**  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 **Finish Debug Session (结束调试会话)**  结束执行。

7. 获取硬件和软件

对于本文档中的 MPLAB XC8 项目，ATmega4809 Curiosity Nano 开发板通过 USB 连接与 PC 通信、由 PC 供电。使用 MPLAB X IDE 进行开发。

获取 MPLAB X IDE 和 MPLAB XC8 C 编译器

可从以下网址找到 MPLAB X IDE v5.45 及以上版本：

www.microchip.com/mplab/mplab-x-ide

可从以下网址找到 MPLAB XC8 C 编译器 v2.31 及以上版本：

www.microchip.com/mplab/compilers

获取 MPLAB 代码配置器 (MCC)

在 MPLAB X IDE 中，转到 Tools>Plugins>Available Plugins (工具>插件>可用插件)，然后安装“MPLAB Code Configurator” (MPLAB 代码配置器)。

可从以下网址找到有关 MCC 的更多信息：

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

获取 AVR® MCU

可从以下网址找到本文档各示例中使用的 AVR MCU：

www.microchip.com/ATmega4809

获取 ATmega4809 Curiosity Nano 开发板

可从以下网址找到 ATmega4809 Curiosity Nano 开发板：

www.microchip.com/DevelopmentTools/ProductDetails/DM320115

关于示例 4 中的电位器

SparkFun Trimmer 10 kΩ 0.5W PC Pin Top

8. 其他信息

以下视频提供了有关在 MPLAB X IDE 中使用 AVR 器件的更多信息。

[Import Studio 7 Project into MPLAB X IDE](#)

[Create a New Project/Project Dashboard](#)

[Context Datasheet Help & AVR® Interrupts](#)

Microchip 网站

Microchip 网站 (www.microchip.com/) 为客户提供在线支持。客户可通过该网站方便地获取文件和信息。我们的网站提供以下内容:

- **产品支持**——数据手册和勘误表、应用笔记和示例程序、设计资源、用户指南以及硬件支持文档、最新的软件版本以及归档软件
- **一般技术支持**——常见问题解答 (FAQ)、技术支持请求、在线讨论组以及 Microchip 设计伙伴计划成员名单
- **Microchip 业务**——产品选型和订购指南、最新 Microchip 新闻稿、研讨会和活动安排表、Microchip 销售办事处、代理商以及工厂代表列表

产品变更通知服务

Microchip 的产品变更通知服务有助于客户了解 Microchip 产品的最新信息。注册客户可在他们感兴趣的某个产品系列或开发工具发生变更、更新、发布新版本或勘误表时, 收到电子邮件通知。

欲注册, 请访问 www.microchip.com/pcn, 然后按照注册说明进行操作。

客户支持

Microchip 产品的用户可通过以下渠道获得帮助:

- 代理商或代表
- 当地销售办事处
- 应用工程师 (ESE)
- 技术支持

客户应联系其代理商、代表或 ESE 寻求支持。当地销售办事处也可为客户提供帮助。本文档后附有销售办事处的联系方式。

也可通过 www.microchip.com/support 获得网上技术支持。

Microchip 器件代码保护功能

请注意以下有关 Microchip 器件代码保护功能的要点:

- Microchip 的产品均达到 Microchip 数据手册中所述的技术规范。
- Microchip 确信: 在正常使用的情况下, Microchip 系列产品非常安全。
- 目前, 仍存在着用恶意、甚至是非法的方法来试图破坏代码保护功能的行为。我们确信, 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。这种试图破坏代码保护功能的行为极可能侵犯 Microchip 的知识产权。
- 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。

法律声明

提供本文档的中文版本仅为了便于理解。请勿忽视文档中包含的英文部分, 因为其中提供了有关 Microchip 产品性能和使用情况的有用信息。Microchip Technology Inc. 及其分公司和相关公司、各级主管与员工及事务代理机构对译文中可能存在的任何差错不承担任何责任。建议参考 Microchip Technology Inc. 的英文原版文档。

本出版物中提供的信息仅仅是为方便您使用 **Microchip** 产品或使用这些产品来进行设计。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 “按原样”提供这些信息。**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，包括但不限于针对非侵权性、适销性和特定用途的适用性的暗示担保，或针对其使用情况、质量或性能的担保。

在任何情况下，对于因这些信息或使用这些信息而产生的任何间接的、特殊的、惩罚性的、偶然的或间接的损失、损害或任何类型的开销，**Microchip** 概不承担任何责任，即使 **Microchip** 已被告知可能发生损害或损害可以预见。在法律允许的最大范围内，对于因这些信息或使用这些信息而产生的所有索赔，**Microchip** 在任何情况下所承担的全部责任均不超出您为获得这些信息向 **Microchip** 直接支付的金额（如有）。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损害、索赔、诉讼或费用时，会维护和保障 **Microchip** 免于承担法律责任。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。

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**、**PackTime**、**PIC**、**picoPower**、**PICSTART**、**PIC32** 徽标、**PolarFire**、**Prochip Designer**、**QTouch**、**SAM-BA**、**SenGenuity**、**SpyNIC**、**SST**、**SST** 徽标、**SuperFlash**、**Symmetricon**、**SyncServer**、**Tachyon**、**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。

AgileSwitch、**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**、**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**、**Augmented Switching**、**BlueSky**、**BodyCom**、**CodeGuard**、**CryptoAuthentication**、**CryptoAutomotive**、**CryptoCompanion**、**CryptoController**、**dsPICDEM**、**dsPICDEM.net**、**Dynamic Average Matching**、**DAM**、**ECAN**、**Espresso T1S**、**EtherGREEN**、**IdealBridge**、**In-Circuit Serial Programming**、**ICSP**、**INICnet**、**Intelligent Paralleling**、**Inter-Chip Connectivity**、**JitterBlocker**、**maxCrypto**、**maxView**、**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**、**RTAX**、**RTG4**、**SAM-ICE**、**Serial Quad I/O**、**simpleMAP**、**SimpliPHY**、**SmartBuffer**、**SMART-I.S.**、**storClad**、**SQI**、**SuperSwitcher**、**SuperSwitcher II**、**Switchtec**、**SynchroPHY**、**Total Endurance**、**TSHARC**、**USBCheck**、**VariSense**、**VectorBlox**、**VeriPHY**、**ViewSpan**、**WiperLock**、**XpressConnect** 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。

© 2021, **Microchip Technology Incorporated** 版权所有。

ISBN: 978-1-5224-8126-3

质量管理体系

有关 **Microchip** 的质量管理体系的信息，请访问 www.microchip.com/quality。

全球销售及服务中心

美洲	亚太地区	亚太地区	欧洲
公司总部 2355 West Chandler Blvd. Chandler, AZ 85224-6199 电话: 480-792-7200 传真: 480-792-7277 技术支持: www.microchip.com/support 网址: www.microchip.com 亚特兰大 德卢斯, 佐治亚州 电话: 678-957-9614 传真: 678-957-1455 奥斯汀, 德克萨斯州 电话: 512-257-3370 波士顿 韦斯特伯鲁, 马萨诸塞州 电话: 774-760-0087 传真: 774-760-0088 芝加哥 艾塔斯卡, 伊利诺伊州 电话: 630-285-0071 传真: 630-285-0075 达拉斯 阿迪森, 德克萨斯州 电话: 972-818-7423 传真: 972-818-2924 底特律 诺维, 密歇根州 电话: 248-848-4000 休斯顿, 德克萨斯州 电话: 281-894-5983 印第安纳波利斯 诺布尔斯维尔, 印第安纳州 电话: 317-773-8323 传真: 317-773-5453 电话: 317-536-2380 洛杉矶 米慎维荷, 加利福尼亚州 电话: 949-462-9523 传真: 949-462-9608 电话: 951-273-7800 罗利, 北卡罗来纳州 电话: 919-844-7510 纽约, 纽约州 电话: 631-435-6000 圣何塞, 加利福尼亚州 电话: 408-735-9110 电话: 408-436-4270 加拿大 - 多伦多 电话: 905-695-1980 传真: 905-695-2078	澳大利亚 - 悉尼 电话: 61-2-9868-6733 中国 - 北京 电话: 86-10-8569-7000 中国 - 成都 电话: 86-28-8665-5511 中国 - 重庆 电话: 86-23-8980-9588 中国 - 东莞 电话: 86-769-8702-9880 中国 - 广州 电话: 86-20-8755-8029 中国 - 杭州 电话: 86-571-8792-8115 中国 - 香港特别行政区 电话: 852-2943-5100 中国 - 南京 电话: 86-25-8473-2460 中国 - 青岛 电话: 86-532-8502-7355 中国 - 上海 电话: 86-21-3326-8000 中国 - 沈阳 电话: 86-24-2334-2829 中国 - 深圳 电话: 86-755-8864-2200 中国 - 苏州 电话: 86-186-6233-1526 中国 - 武汉 电话: 86-27-5980-5300 中国 - 西安 电话: 86-29-8833-7252 中国 - 厦门 电话: 86-592-2388138 中国 - 珠海 电话: 86-756-3210040	印度 - 班加罗尔 电话: 91-80-3090-4444 印度 - 新德里 电话: 91-11-4160-8631 印度 - 浦那 电话: 91-20-4121-0141 日本 - 大阪 电话: 81-6-6152-7160 日本 - 东京 电话: 81-3-6880-3770 韩国 - 大邱 电话: 82-53-744-4301 韩国 - 首尔 电话: 82-2-554-7200 马来西亚 - 吉隆坡 电话: 60-3-7651-7906 马来西亚 - 槟榔屿 电话: 60-4-227-8870 菲律宾 - 马尼拉 电话: 63-2-634-9065 新加坡 电话: 65-6334-8870 台湾地区 - 新竹 电话: 886-3-577-8366 台湾地区 - 高雄 电话: 886-7-213-7830 台湾地区 - 台北 电话: 886-2-2508-8600 泰国 - 曼谷 电话: 66-2-694-1351 越南 - 胡志明市 电话: 84-28-5448-2100	奥地利 - 韦尔斯 电话: 43-7242-2244-39 传真: 43-7242-2244-393 丹麦 - 哥本哈根 电话: 45-4485-5910 传真: 45-4485-2829 芬兰 - 埃斯波 电话: 358-9-4520-820 法国 - 巴黎 电话: 33-1-69-53-63-20 传真: 33-1-69-30-90-79 德国 - 加兴 电话: 49-8931-9700 德国 - 哈恩 电话: 49-2129-3766400 德国 - 海布隆 电话: 49-7131-72400 德国 - 卡尔斯鲁厄 电话: 49-721-625370 德国 - 慕尼黑 电话: 49-89-627-144-0 传真: 49-89-627-144-44 德国 - 罗森海姆 电话: 49-8031-354-560 以色列 - 若那那市 电话: 972-9-744-7705 意大利 - 米兰 电话: 39-0331-742611 传真: 39-0331-466781 意大利 - 帕多瓦 电话: 39-049-7625286 荷兰 - 德卢内市 电话: 31-416-690399 传真: 31-416-690340 挪威 - 特隆赫姆 电话: 47-72884388 波兰 - 华沙 电话: 48-22-3325737 罗马尼亚 - 布加勒斯特 电话: 40-21-407-87-50 西班牙 - 马德里 电话: 34-91-708-08-90 传真: 34-91-708-08-91 瑞典 - 哥德堡 电话: 46-31-704-60-40 瑞典 - 斯德哥尔摩 电话: 46-8-5090-4654 英国 - 沃金厄姆 电话: 44-118-921-5800 传真: 44-118-921-5820