MPLAB® XC8 User's Guide for Embedded Engineers - AVR MCUs

Introduction

This document presents five code examples for 8-bit AVR MCU devices and the MPLAB® XC8 C compiler using the Common Code Interface (CCI). For more on CCI, see the "*MPLAB® XC8 C Compiler User's Guide for AVR MCU*" (DS50002750).

Some knowledge of microcontrollers and the C programming language is necessary.

Table of Contents

1. Turn LED On or Off

This example will light the User LED on the ATmega4809 Curiosity Nano board. For more information, see section 7. Get Hardware and Software.

```
/*
 * 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 Configuration Bits

Microchip devices have configuration bits, or fuses, that enable and/or set up device features.

Note: If you do not set Configuration bits correctly, your device will not operate at all, or at least not as expected.

Which Configuration Bits to Set

In particular, be aware of the followings settings:

- Oscillator configuration - This must match your hardware's oscillator circuitry. If this is not correct, the device clock may not run. Typically, development boards use high-speed crystal oscillators. From the example code:

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

- Watchdog timer configuration- It is recommended that you disable this timer until it is required. This prevents unexpected Resets. From the example code:

```
FUSES = {
.WDTCFG = 0x00, // WDTCFG {PERIOD=OFF, WINDOW=OFF}
... }
```

- Code protection - Turn off code protection until it is required. This ensures that device memory is fully accessible. From the example code:

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

Note: You may also set configuration bits with `#pragma config`. For details, see the "*MPLAB® XC8 C Compiler User's Guide for AVR® MCU*" (DS50002750).

See your device data sheet for the name and function of corresponding configuration bits. Use the part number to search www.microchip.com for the appropriate data sheet.

For more information about configuration bits that are available for each device, see the following file in the location where MPLAB XC8 was installed:

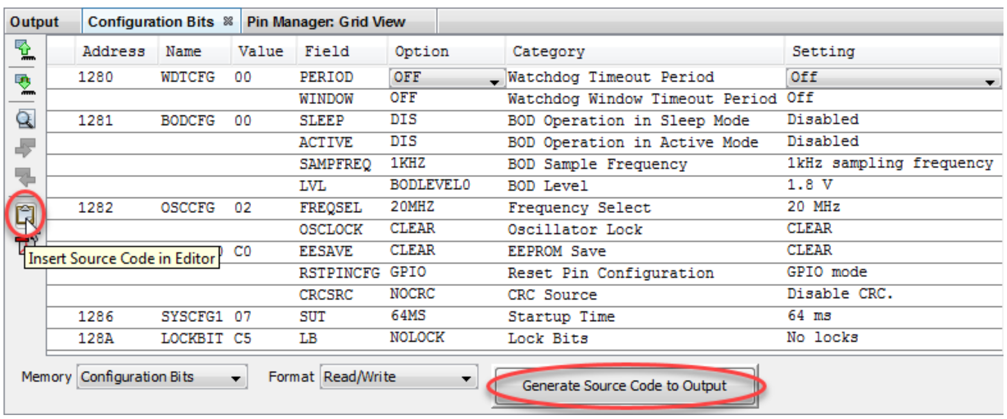
`MPLAB XC8 Installation Directory/docs/avr_chipinfo.html`

How to Set Configuration Bits

In MPLAB X IDE, you can use the Configuration Bits window to view and set these bits. Select *Window>Target Memory Views>Configuration Bits* to open this window.

For AVR devices, click  to read configuration memory, i.e., FUSES and LOCKBITS values.

Figure 1-1. Configuration Bits Window



Once the settings are selected, click in code where you want this information placed and then click the **Insert Source Code in Editor** icon, as shown in the example code. Alternatively click the **Generate Source Code to Output** button to copy and paste into code.

See MPLAB X IDE documentation for more information on this window.

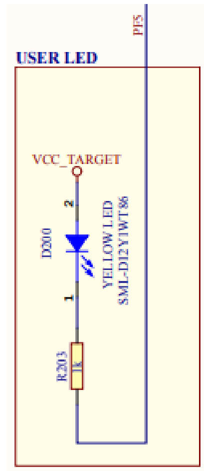
1.2 Included Header File

The `xc.h` header file allows code in the source file to access compiler-specific or device-specific features. Based on your selected device, the compiler sets macros that allow `avr/io.h` to vector to the correct device-specific header file. Do not include a device-specific header in your code or your code will not be portable.

This and other header files can be found in the MPLAB XC8 installation directory in the `avr/avr/include/avr` or `dfp/xc8/avr/include/avr` subdirectories, or from paths specified via the `-mdfp` option.

1.3 Port Access for LED

Each I/O pin Pxn can be controlled by the registers in PORTx. Each pin group x has its own set of PORT registers. For this example, PF5 of PORTF is used to turn the User LED on or off. Examining the board schematics shows that PF5 must be low for the LED to be lit and high for the LED to be off. Find the schematics link in the Kit Window.



Digital I/O device pins may be multiplexed with peripheral I/O pins. To ensure that you are using digital I/O only, disable the other peripheral(s). Do this by using the predefined C variables that represent the peripheral registers and bits. These variables are listed in the device-specific header file in the compiler include directory. To determine which peripherals share which pins, refer to your device data sheet.

To use PF5 as an output only pin, write bit 5 in the PORTF.DIRSET register to '1'. To do this without disturbing the value of the other bits, masks are provided for each register bit. In this case, PIN5_bm = 00001000.

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

Writing bit 5 in PORTF.OUTCLR to '1' will clear that bit (set to ‘0’). This will turn the LED on.

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

Alternately, writing bit 5 in PORTF.OUTSET to '1' will set that bit (set to ‘1’). To turn the LED off, comment out the above instruction and uncomment the following instruction.

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

To view information on PORT registers, highlight a register name in the Editor, right click, and select *Navigage>Online Datasheet*.

2. Flash LED Using Delay Function

This example is a modification of the previous code. Instead of just turning on or off the User LED, this code will make the LED flash.

```
/*
 * 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 The while() Loop and Toggle Function

Writing a bit in PORTF.OUTTGL to '1' will toggle that bit. For PF5:

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

To continually toggle the pin, and flash the LED, the `while(1)` loop is used.

2.2 The Delay Function

Because the speed of execution will, in most cases, cause the LED to flash faster than the eye can see, execution needs to be slowed. `_delay_ms()` is a built-in function of the compiler. To use this function, the header `util/delay.h` must be included. Also, the speed of the processor must be specified:

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

For more details on the delay built-in, see the "*MPLAB XC8 C Compiler User's Guide for AVR MCU*" (DS50002750).

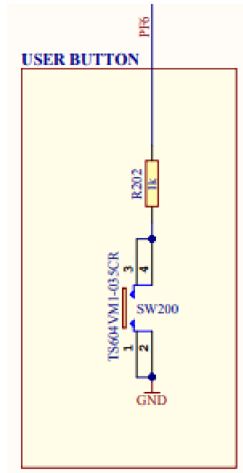
3. Toggle LED Using Button Press and Interrupts

This example is a modification of the previous code. This time the User LED will be turned on or off by clicking the User button. When the button is clicked, interrupts will be used to toggle the LED state.

```
/*
 * 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 Port Access for Button

For this example, PF6 of PORTF is used to sense if a button press has occurred. Examining the board schematics shows that the internal port pull-up will need to be enabled so that the PF6 will go from ‘1’ to ‘0’ when the button is pressed.



While you could use DIRCLR to set PF6 as an input pin, the reset value of PORTx is 0x00 which sets all pins to inputs.

PINCTRL enables the pull-up and configures the input/sense on PF6. The sense configuration determines how a port interrupt can be triggered.

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

3.2 Interrupt on Pin Change

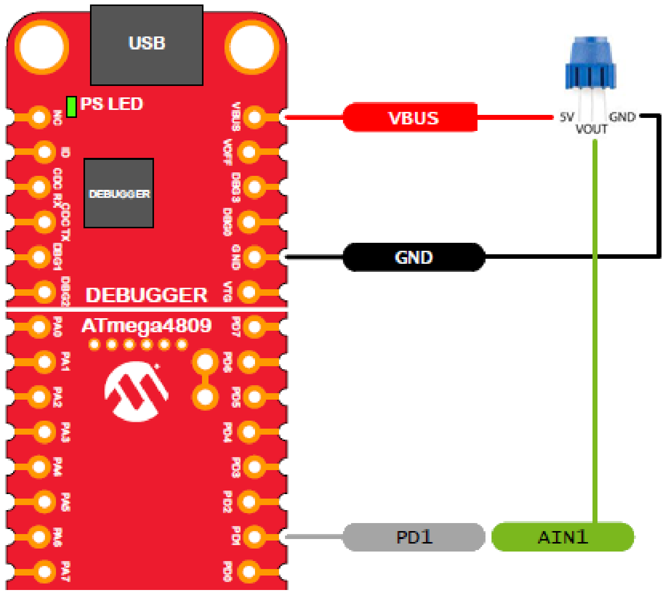
In `main()` code, `ei()` enables global interrupts.

`__interrupt(`*`vector`*`)` specifies `btnInt()` as the interrupt function. Port interrupt vectors are of the form `PORTx_Port_vect_num` (see `iom4809.h`.) To ensure only the PF6 change triggers an interrupt, the interrupt flag is checked. Then the LED is toggled, and the interrupt flag is cleared. Writing a '1' to a flag's bit location will clear the flag.

4. Light LED if Potentiometer Value Below ADC Value

This ADC Window Comparator example will demonstrate how to initialize the ADC, set the conversion window comparator low threshold, enable the conversion Window mode, enable the Free Running mode, start the conversion, and then wait until the conversion is done to turn on the LED if the ADC result is below the set threshold or turn off the LED it the result is above the threshold. A potentiometer was used as the analog source.

Figure 4-1. ATmega4809 Curiosity Nano ADC Connections



For more on this and other ADC examples, see TB3209: "*Getting Started with ADC*" (DS90003209).

Instead of generating code by hand, the MPLAB Code Configurator (MCC) is used. The MCC is a plug-in available for installation under the MPLAB X IDE menu *Tools>Plugins*, **Available Plugins** tab. See MPLAB X IDE Help for more on how to install plugins.

For information on the MCC, including the "*MPLAB® Code Configurator 3.xx User's Guide*" (DS40001829), go to the MPLAB Code Configurator web page at:

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

For this example, the MCC UI was set up as shown in the following sections.

4.1 MCC System Resource Configuration

Figure 4-2. Project Resources - System Module

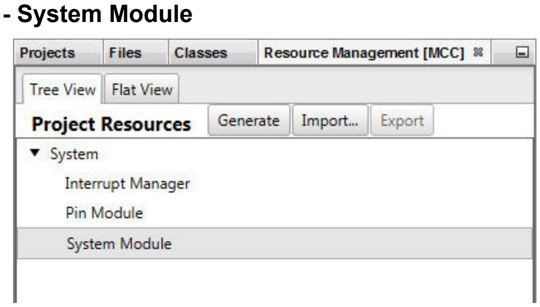
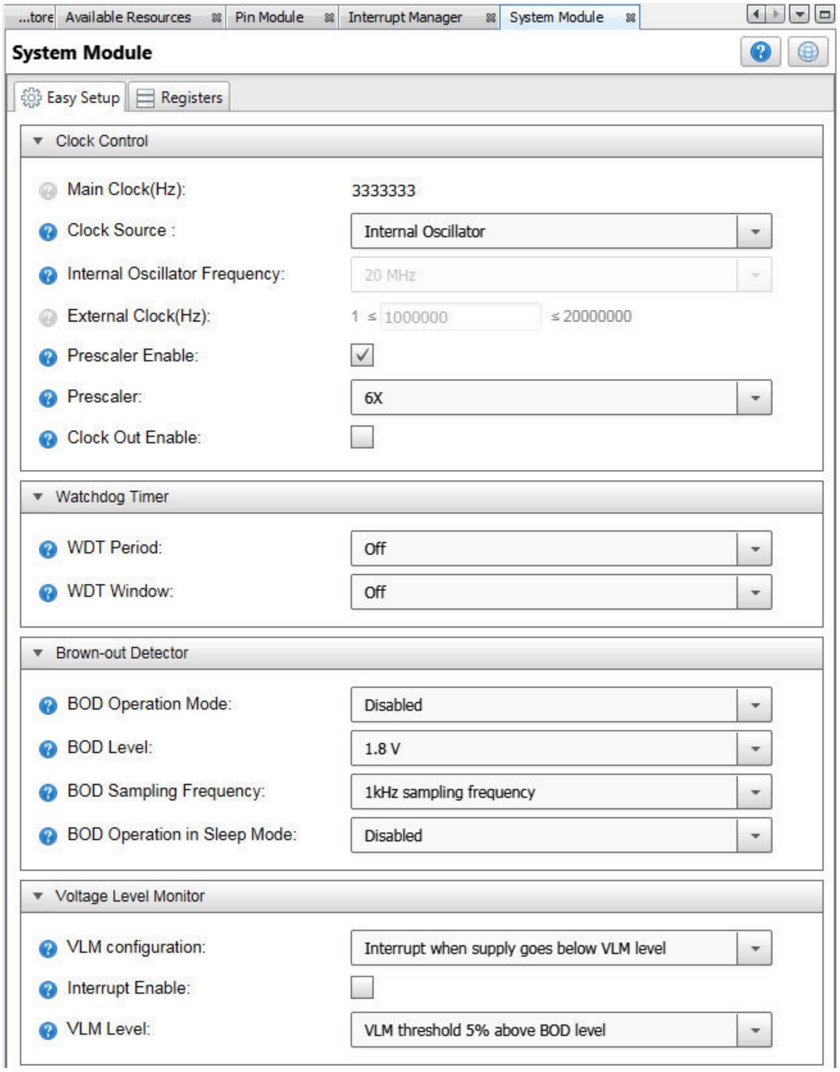


Figure 4-3. System Module Configuration



4.2 MCC ADC Resource Configuration

Although only the “Easy Setup” tab for ADC Resource Configuration is shown, you should also review the “Registers” tab for setups not shown (MUXPOS) and for alternate data entry.

Figure 4-4. ADC Resource Selection

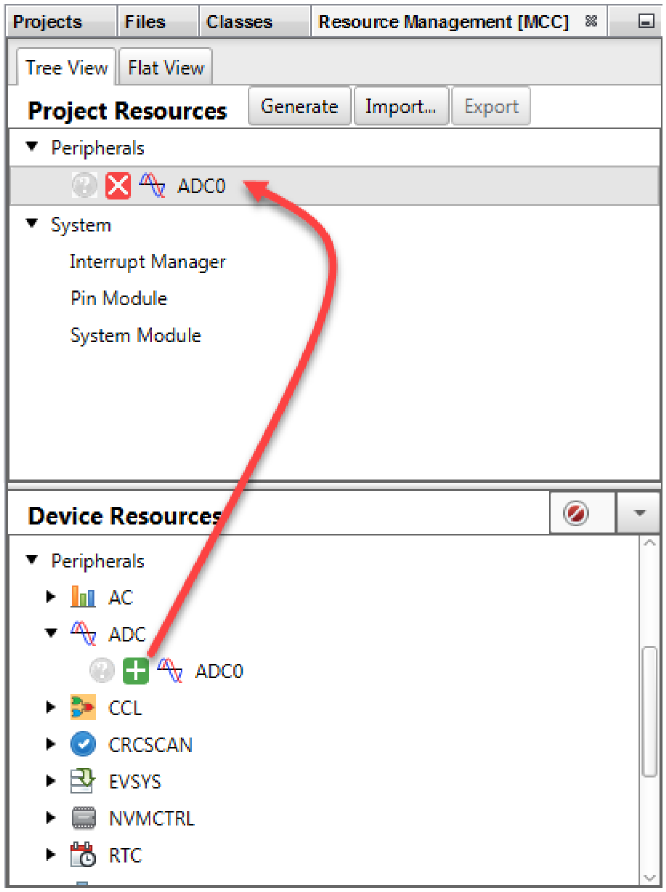
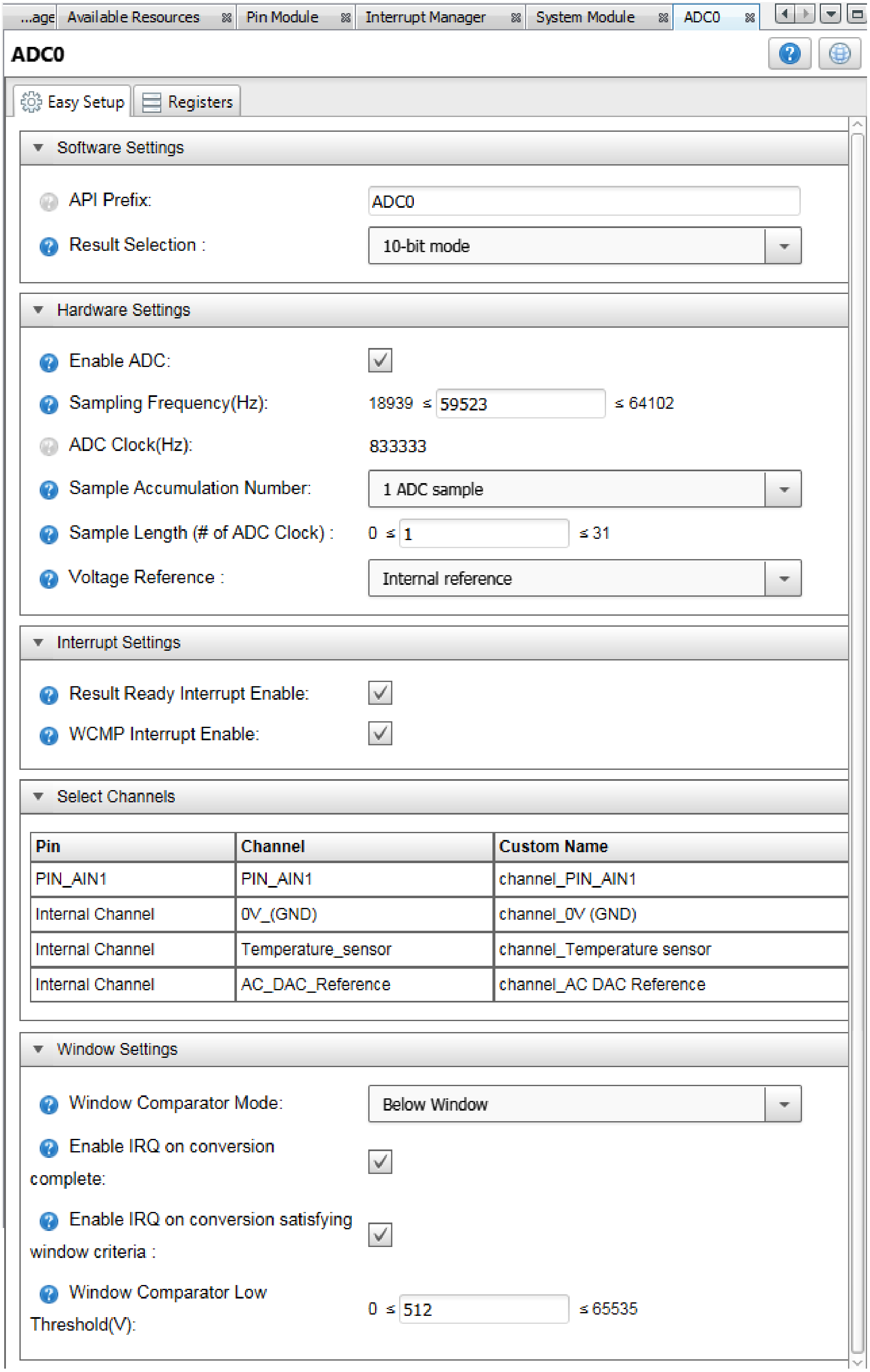


Figure 4-5. ADC Resource Configuration



4.3 MCC GPIO Pin Resource Configuration

Figure 4-6. GPIO Pin Resource - Grid

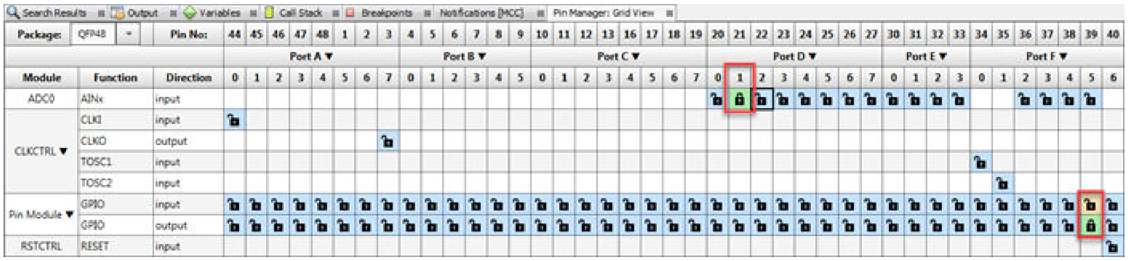
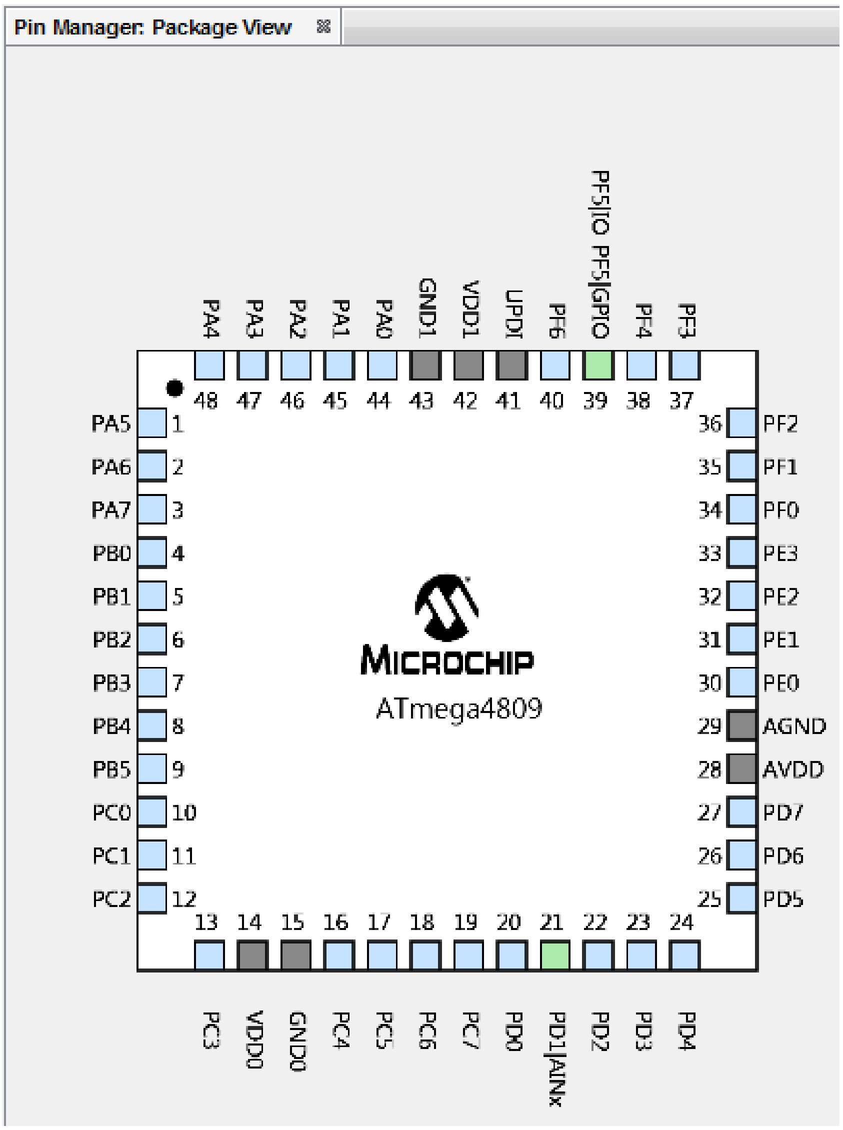


Figure 4-7. GPIO Pin Resource - Package



4.4 MCC Pin Resource Configuration

Figure 4-8. Project Resources - Pin Module

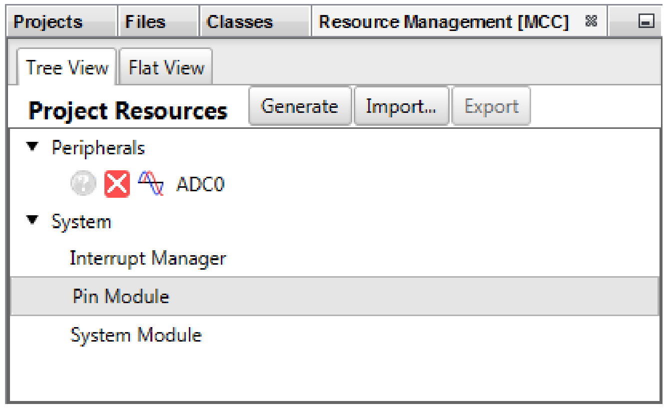
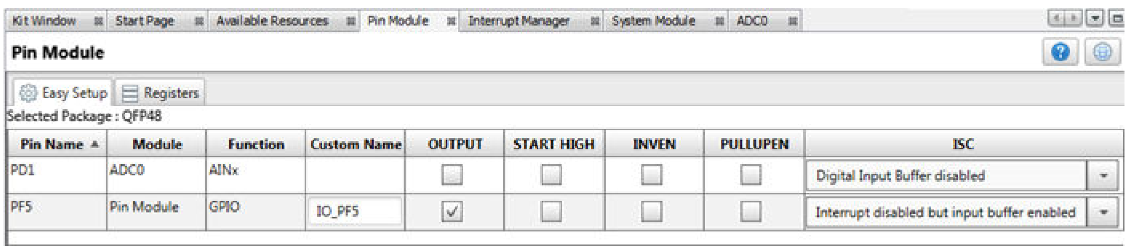


Figure 4-9. Pin Module Configuration



4.5 MCC Interrupt Manager Resource Configuration

Figure 4-10. Project Resources - Interrupt Manager

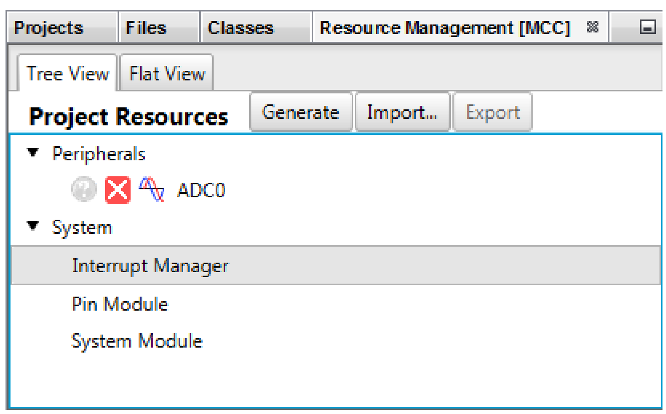
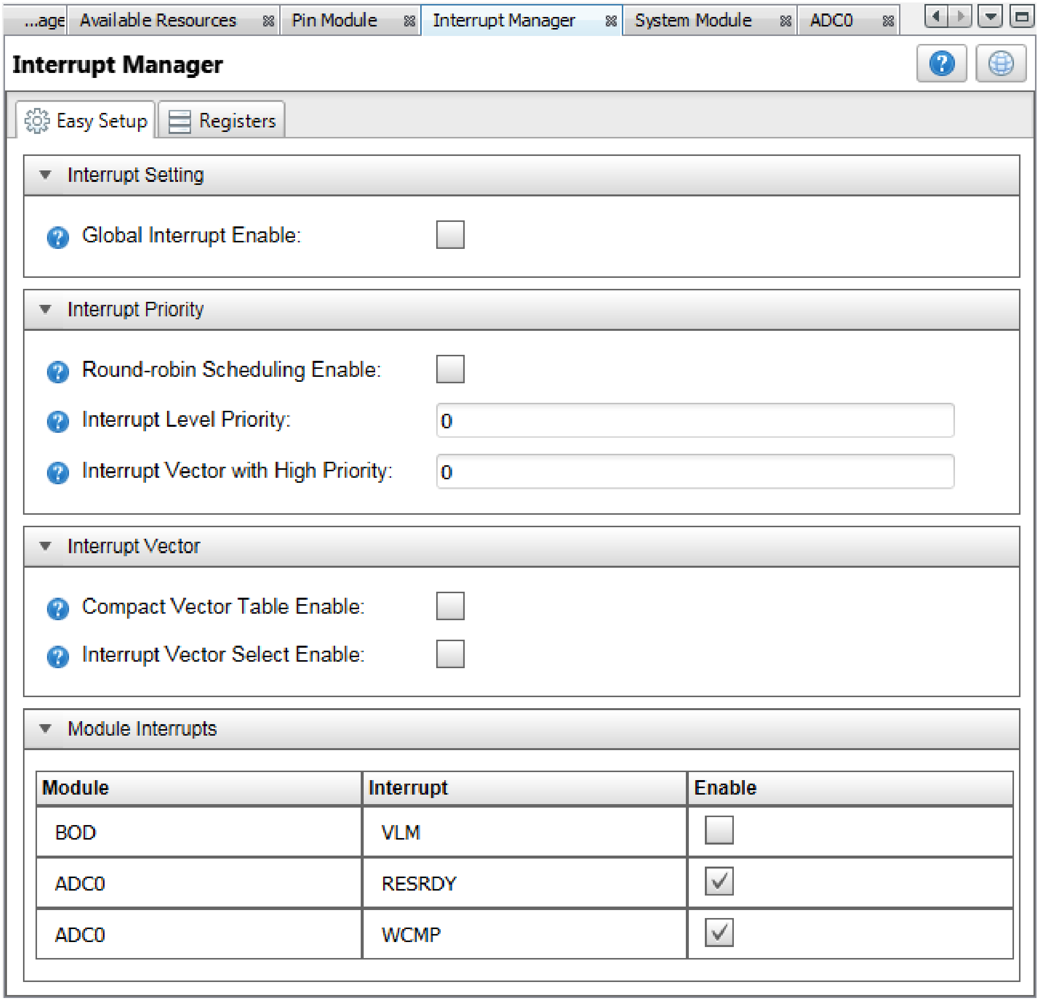


Figure 4-11. Interrupt Manager Configuration



4.6 MCC Code Generation

When the code is configured (as shown in the previous figures), click the **Generate** button on the "Project Resources" window. Code generated by the MCC is modular. Therefore main, system, and peripheral code are all in individual files. Also, each peripheral has its own header file.

Editing of `main.c` is always required to add functionality to your program. Review the generated files to find any functions or macros you may need in your code.

Figure 4-12. Code Generated in Project Tree

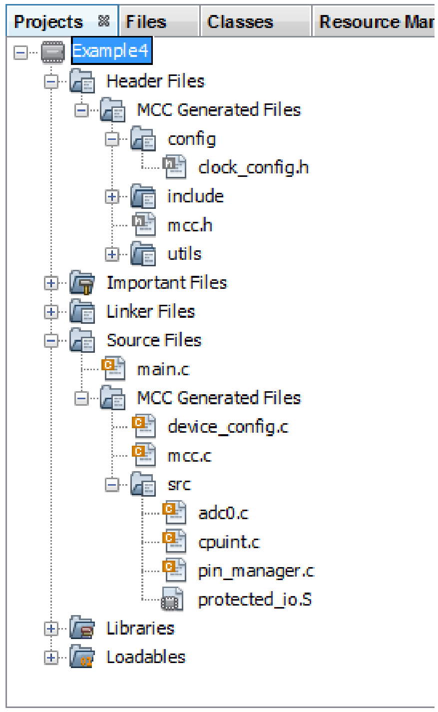
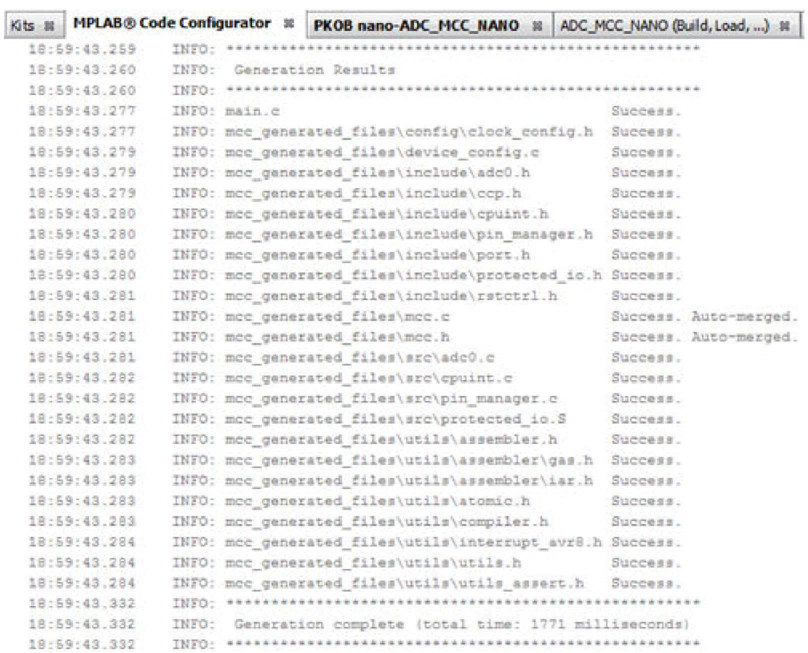


Figure 4-13. Code Generation Progress in Output Window



4.7 main.c Modified Code

The `main.c` template file has been edited as shown below. Some comments have been removed.

Note: `<xc.h>` is automatically included by "`mcc_generated_files/mcc.h`".

```
/*
    (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 Associated Variables

The `channel` variable is needed for the `ADC0_StartConversion()` function.

```
adc_0_channel_t channel = ADC_MUXPOS_AIN1_gc;
```

In `adc0.h`, `adc_0_channel_t` is defined as type `ADC_MUXPOS_t`.

```
typedef ADC_MUXPOS_t adc_0_channel_t;
```

In the device-specific `io.h` file (in this case `iom4809.h`), `ADC_MUXPOS_t` is declared (`ADC_MUXPOS_AIN2_gc` through `ADC_MUXPOS_AIN14_gc` removed for brevity).

```
/* 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;
```

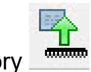
4.7.2 ADC Window Comparison

The functions used to execute ADC operation and comparison may be found in the `adc0.c` file.

- `ADC0_Enable()` - Enable the ADC module.
- `ADC0_StartConversion(channel)` - Start the ADC conversion.
- `ADC0_IsConversionDone()` - In the `while()` loop, check for when a conversion is complete.

- `ADC0_GetWindowResult()` - If conversion result is below threshold the value is true (turn on LED); if not, the value is false (turn off LED).

5. Flash LED after EEData Write and Read

This example demonstrates how to write to and read from EEPROM Data (EE Data) memory. After writing and reading complete successfully, the LED is flashed. To view EEPROM memory before and after writing, open *Window>Target Memory Views>EEPROM Memory* and then Read Device Memory .

Again, MPLAB Code Configurator (MCC) is used to generate most of the code. To find out how to install and get the user's guide for MCC, see:

4. Light LED if Potentiometer Value Below ADC Value

For this example, the MCC GUI was set up as shown in the following sections.

5.1 MCC System Resource Configuration

Figure 5-1. Project Resources - System Module

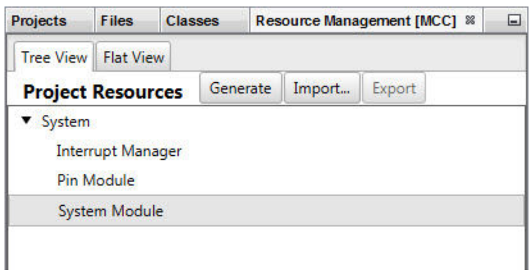
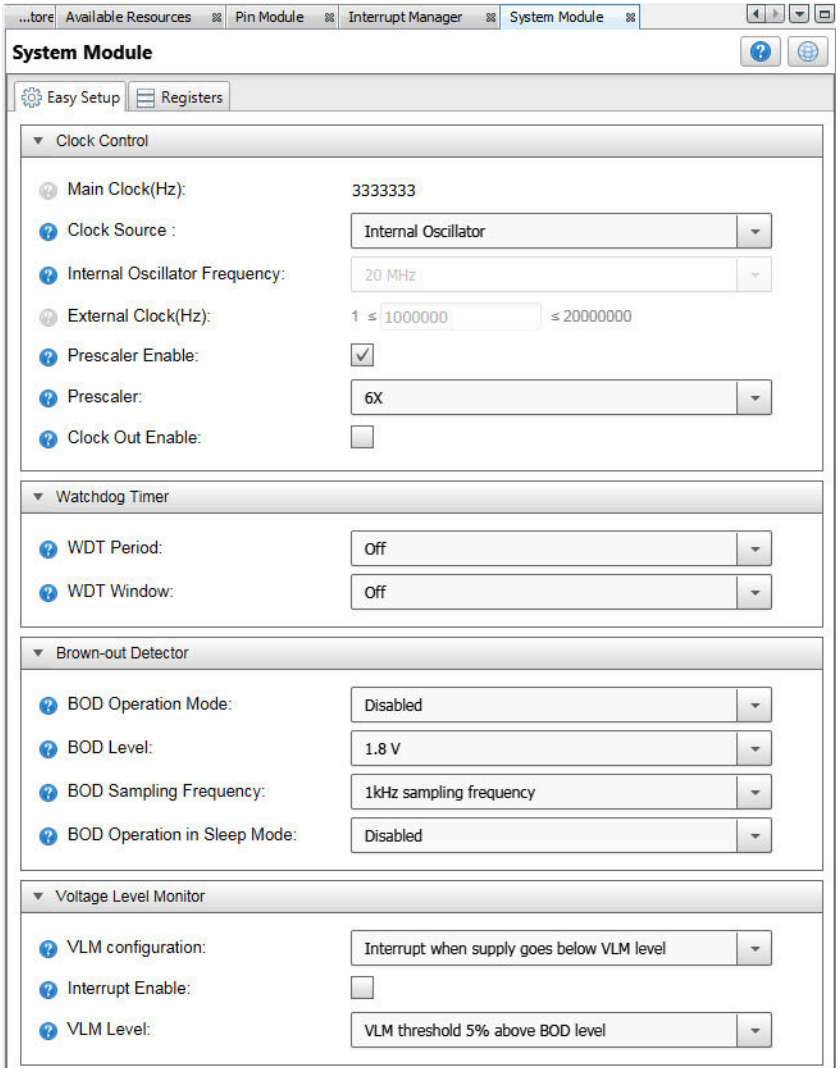


Figure 5-2. System Module Configuration



5.2 MCC Memory Resource Configuration

To add EE Data to Project Resources:

1. Under Device Resources, find and expand NVMCTRL (Non-Volatile Memory Control).
2. Click on the green plus sign to add under Project Resources.
3. Click on NVMCTRL to view resource configuration settings. For this example, no changes will be made.

Figure 5-3. NVMCTRL (EE Data) Resource Selection

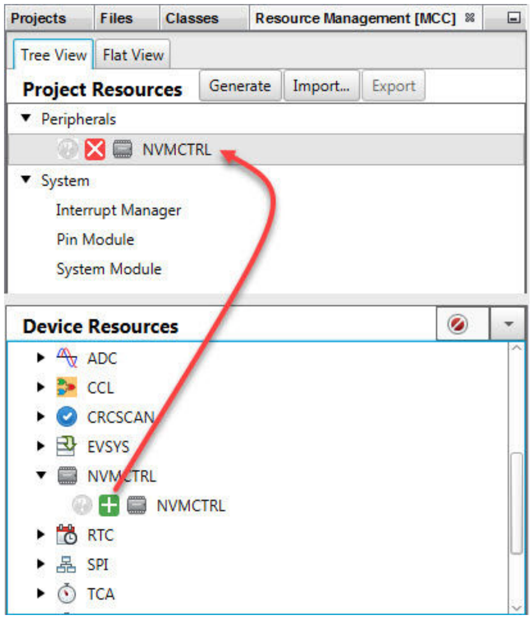
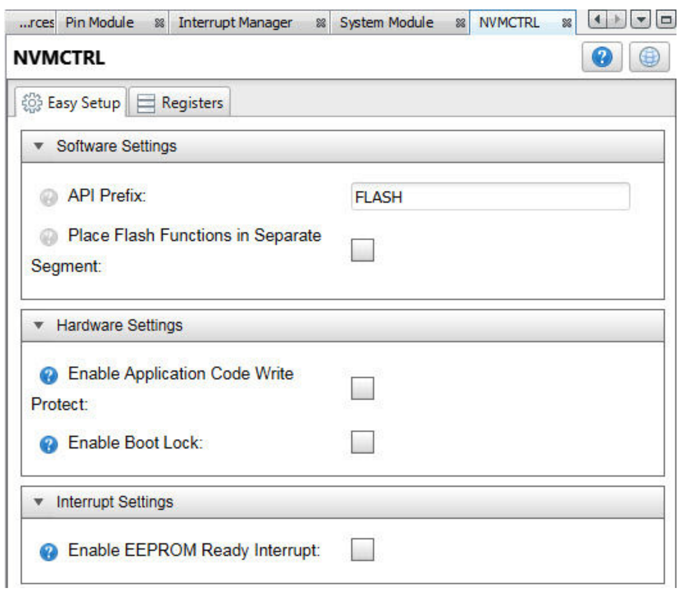


Figure 5-4. NVMCTRL (EE Data) Resource Configuration



5.3 MCC GPIO Pin Resource Configuration

In order to flash the LED, Port B pin 5 must be set as an output.

Figure 5-5. GPIO Pin Resource - Grid

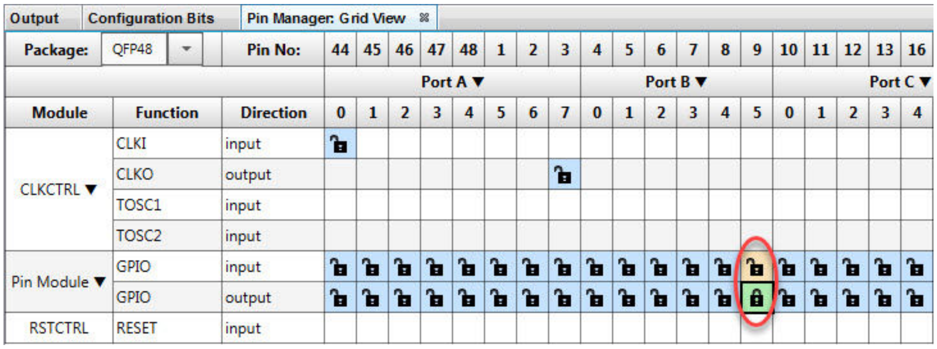
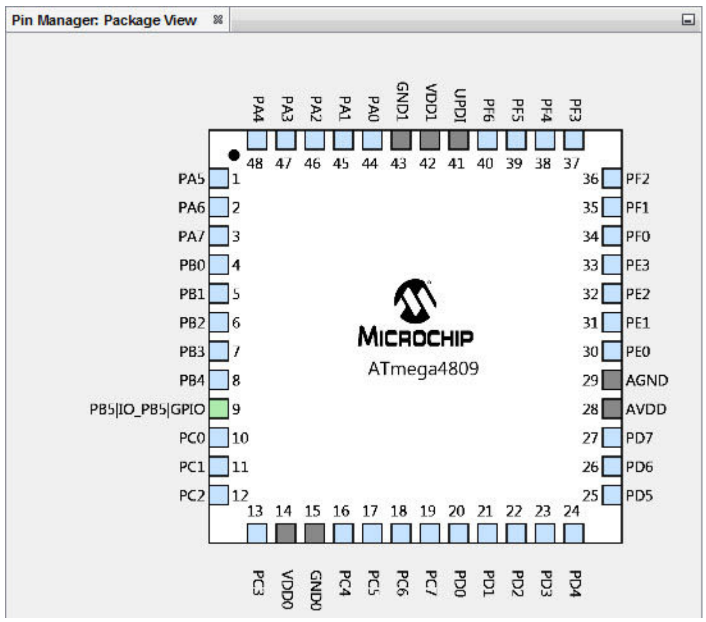


Figure 5-6. GPIO Pin Resource - Package



5.4 MCC Pin Resource Configuration

Under Project Resources, click on “Pin Module” to view Pin Module configuration settings.

PB5 appears in the Pin Module window because it was selected in the Pin Manager: Grid View window. No changes to the pin configuration will be made for this example.

Figure 5-7. Project Resources - Pin Module

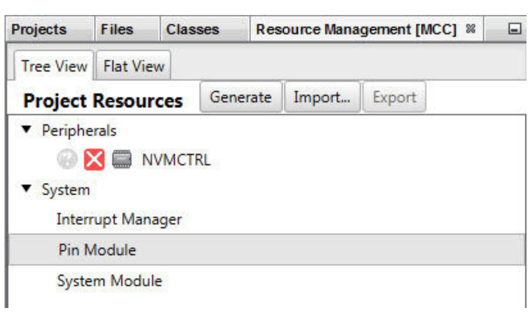
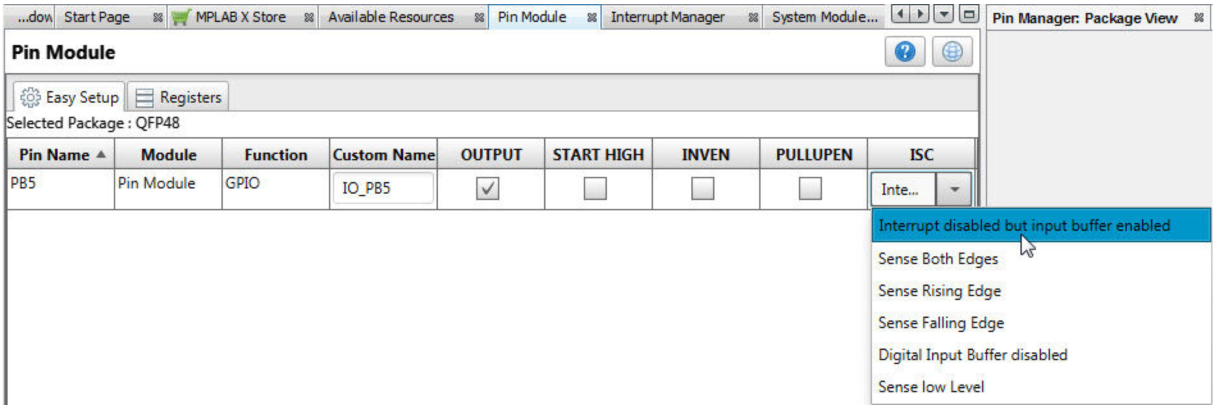


Figure 5-8. Pin Module Configuration



5.5 MCC Code Generation

When the code is configured (as shown in the previous figures), click the **Generate** button on the “Project Resources” window. Code generated by the MCC is modular. Therefore main, system, and peripheral code are all in individual files. Also, each peripheral has its own header file.

Editing of `main.c` is always required to add functionality to your program. Review the generated files to find any functions or macros you may need in your code.

Figure 5-9. Code Generated in Project Tree

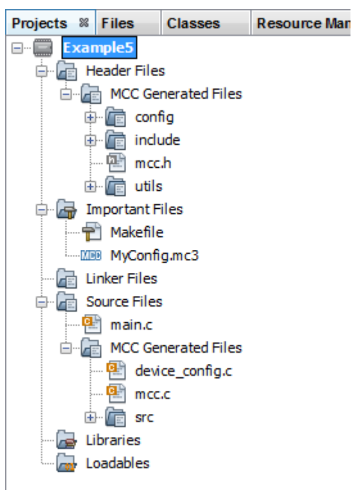
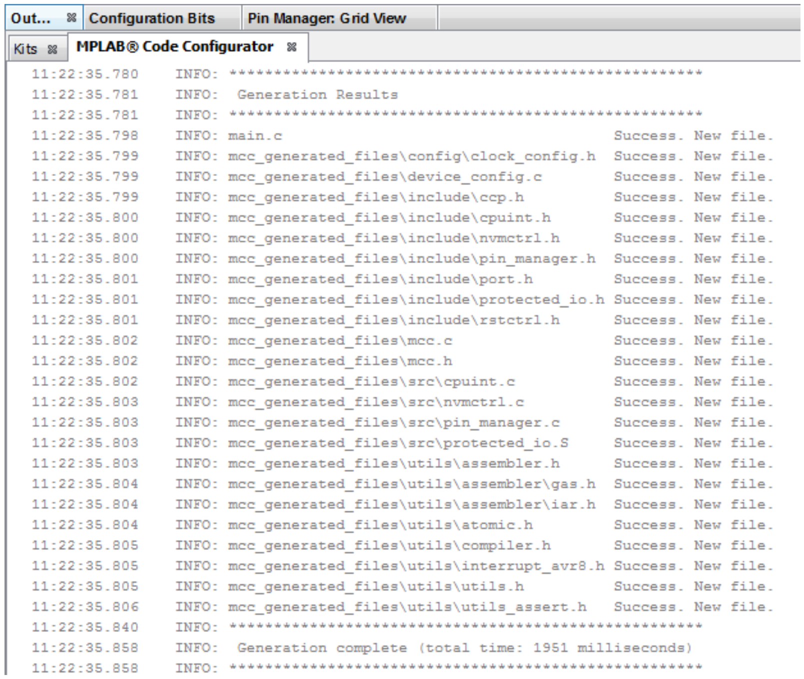


Figure 5-10. Code Generation in the Output Window



5.6 main.c Modified Code

The `main.c` template file has been edited as shown below. Some comments have been removed.

```
/*
    (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 Data Associated Variables

Variables used to store data from an EE Data read or write must match the types specified in the read/write function prototype, referenced from `mcc.h`, and found in `nvmctrl.h`:

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

From `stdint.h` (also referenced), `uint8_t` is the same as `unsigned char`.

5.6.2 Write to EE Data

In this example, data is written to EE Data and then read back.

The function to write one byte of data to EE Data, `FLASH_WriteEepromByte()`, may be found in `nvmctrl.c`. Within this function is a loop that waits until any previous writes have finished before starting the next write. So there is no need to check if a write is complete, i.e., using `FLASH_IsEepromReady()`.

5.6.3 Read from EE Data

After EE Data is written, memory values are read into a RAM array. The function to read one byte of data to EE Data, `FLASH_ReadEepromByte()`, may be found in `nvmctrl.c`.

Once the values are read, a while loop flashes the LED to indicate successful program completion.

6. Run Code in MPLAB X IDE

Follow the instructions below to execute example code in MPLAB X IDE.

6.1 Create a Project

1. Launch MPLAB X IDE.
2. From the IDE, launch the New Project Wizard .

Follow the screens to create a new project:

1. **Choose Project**: Select “Microchip Embedded” and then select “Standalone Project.”
2. **Select Device** and Tool: Select the example device. Select your hardware debug tool, SNxxxxxx. If you do not see a serial number (SN) under your debug tool name, ensure that your debug tool is correctly installed. See your debug tool documentation for details.
3. Select Header: None.
4. Select Plugin Board: None.
5. **Select Compiler**: Select XC8 (latest version number) [bin location]. If you do not see a compiler under XC8, ensure the compiler is correctly installed and that MPLAB X IDE is aware of it (*Tools>Options>Embedded>Build Tools*). See MPLAB XC8 and MPLAB X IDE documentation for details.
6. **Select Project Name and Folder**: Name the project.

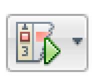
6.2 Select the Common Compiler Interface (CCI)

After your project is created, right click on the project name in the Projects window and select Properties. In the dialog box, click on the “XC8 Compiler” category, select the “Preprocessing and messages” option category, and check “Use CCI syntax.” Click the **OK** button.

6.3 Debug the Examples

Do one of the following, based on the example you are using:

1. For examples 1, 2, and 3, create a file to hold the example code:
   - 1.1. Right click on the “Source Files” folder in the Projects window. Select *New>main.c*. The “New main.c” dialog opens.
   - 1.2. Under “File name,” enter a name (e.g., example*n*), where *n* is the example number.
   - 1.3. Click **Finish**. The file opens in an editor window.
   - 1.4. Delete the template code in the file. Then cut and paste the example code from this user’s guide into the empty editor window and select *File>Save*.
2. For examples 4 and 5, follow the instructions in each section to generate code using MCC and then edit the `main.c` file with the code shown.

Finally, select Debug Project  to build, download to a device, and execute the code. View the demo board LEDs for output. Click Finish Debug Session  to end execution.

7. Get Hardware and Software

For the MPLAB XC8 projects in this document, the ATmega4809 Curiosity Nano development board is powered from and communicates with the PC using a USB connection. MPLAB X IDE was used for development.

Get MPLAB X IDE and MPLAB XC8 C Compiler

MPLAB X IDE v5.45 and later can be found at:

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

The MPLAB XC8 C compiler v2.31 and later can be found at:

www.microchip.com/mplab/compilers

Get the MPLAB Code Configurator (MCC)

In MPLAB X IDE, go to *Tools>Plugins>Available Plugins* and install "MPLAB Code Configurator".

More on MCC can be found at:

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

Get AVR® MCUs

The AVR MCU used in the examples are available at:

www.microchip.com/ATmega4809

Get the ATmega4809 Curiosity Nano

The ATmega4809 Curiosity Nano board is available at:

www.microchip.com/DevelopmentTools/ProductDetails/DM320115

About the Potentiometer in Example 4

SparkFun Trimmer 10KΩ 0.5W PC Pin Top

8. Additional Information

Some videos with further information on using AVR devices in MPLAB X IDE.

Import Studio 7 Project into MPLAB X IDE

Create a New Project/Project Dashboard

Context Datasheet Help & AVR® Interrupts

The Microchip Website

Microchip provides online support via our website at www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: www.microchip.com/support

Product Identification System

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.

PART NO.	[X][(1)]	-	X	/XX	XXX
Device	Tape and Reel Option		Temperature Range	Package	Pattern

Device:	PIC16F18313, PIC16LF18313, PIC16F18323, PIC16LF18323	
Tape and Reel Option:	Blank	= Standard packaging (tube or tray)
	T	= Tape and Reel[(1)]
Temperature Range:	I	= -40°C to +85°C (Industrial)
	E	= -40°C to +125°C (Extended)
Package:[(2)]	JQ	= UQFN
	P	= PDIP
	ST	= TSSOP
	SL	= SOIC-14
	SN	= SOIC-8
	RF	= UDFN
Pattern:	QTP, SQTP, Code or Special Requirements (blank otherwise)	

Examples:

- PIC16LF18313- I/P Industrial temperature, PDIP package
- PIC16F18313- E/SS Extended temperature, SSOP package

Notes:

1. Tape and Reel identifier only appears in the catalog part number description. This identifier is used for ordering purposes and is not printed on the device package. Check with your Microchip Sales Office for package availability with the Tape and Reel option.
2. Small form-factor packaging options may be available. Please check www.microchip.com/packaging for small-form factor package availability, or contact your local Sales Office.

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods being used in attempts to breach the code protection features of the Microchip devices. We believe that these methods require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Attempts to breach these code protection features, most likely, cannot be accomplished without violating Microchip's intellectual property rights.
- Microchip is willing to work with any customer who is concerned about the integrity of its code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is "unbreakable." Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Legal Notice

Information contained in this publication is provided for the sole purpose of designing with and using Microchip products. Information regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications.

THIS INFORMATION IS PROVIDED BY MICROCHIP “AS IS”. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION. Use of Microchip devices in life support and/or safety applications is entirely at the buyer’s risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Trademarks

The Microchip name and logo, the Microchip logo, Adaptec, AnyRate, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PackeTime, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

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 logo, Quiet-Wire, SmartFusion, SyncWorld, Temux, TimeCesium, TimeHub, TimePictra, TimeProvider, WinPath, and ZL are registered trademarks of Microchip Technology Incorporated in the U.S.A.

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 logo, 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, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

The Adaptec logo, Frequency on Demand, Silicon Storage Technology, and Symmcom are registered trademarks of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

ISBN: 978-1-5224-7498-2

AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: www.microchip.com/support Web Address: www.microchip.com **Atlanta** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **Austin, TX** Tel: 512-257-3370 **Boston** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **Chicago** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **Dallas** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **Detroit** Novi, MI Tel: 248-848-4000 **Houston, TX** Tel: 281-894-5983 **Indianapolis** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **Los Angeles** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **Raleigh, NC** Tel: 919-844-7510 **New York, NY** Tel: 631-435-6000 **San Jose, CA** Tel: 408-735-9110 Tel: 408-436-4270 **Canada - Toronto** Tel: 905-695-1980 Fax: 905-695-2078	**Australia - Sydney** Tel: 61-2-9868-6733 **China - Beijing** Tel: 86-10-8569-7000 **China - Chengdu** Tel: 86-28-8665-5511 **China - Chongqing** Tel: 86-23-8980-9588 **China - Dongguan** Tel: 86-769-8702-9880 **China - Guangzhou** Tel: 86-20-8755-8029 **China - Hangzhou** Tel: 86-571-8792-8115 **China - Hong Kong SAR** Tel: 852-2943-5100 **China - Nanjing** Tel: 86-25-8473-2460 **China - Qingdao** Tel: 86-532-8502-7355 **China - Shanghai** Tel: 86-21-3326-8000 **China - Shenyang** Tel: 86-24-2334-2829 **China - Shenzhen** Tel: 86-755-8864-2200 **China - Suzhou** Tel: 86-186-6233-1526 **China - Wuhan** Tel: 86-27-5980-5300 **China - Xian** Tel: 86-29-8833-7252 **China - Xiamen** Tel: 86-592-2388138 **China - Zhuhai** Tel: 86-756-3210040	**India - Bangalore** Tel: 91-80-3090-4444 **India - New Delhi** Tel: 91-11-4160-8631 **India - Pune** Tel: 91-20-4121-0141 **Japan - Osaka** Tel: 81-6-6152-7160 **Japan - Tokyo** Tel: 81-3-6880- 3770 **Korea - Daegu** Tel: 82-53-744-4301 **Korea - Seoul** Tel: 82-2-554-7200 **Malaysia - Kuala Lumpur** Tel: 60-3-7651-7906 **Malaysia - Penang** Tel: 60-4-227-8870 **Philippines - Manila** Tel: 63-2-634-9065 **Singapore** Tel: 65-6334-8870 **Taiwan - Hsin Chu** Tel: 886-3-577-8366 **Taiwan - Kaohsiung** Tel: 886-7-213-7830 **Taiwan - Taipei** Tel: 886-2-2508-8600 **Thailand - Bangkok** Tel: 66-2-694-1351 **Vietnam - Ho Chi Minh** Tel: 84-28-5448-2100	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **Denmark - Copenhagen** Tel: 45-4485-5910 Fax: 45-4485-2829 **Finland - Espoo** Tel: 358-9-4520-820 **France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **Germany - Garching** Tel: 49-8931-9700 **Germany - Haan** Tel: 49-2129-3766400 **Germany - Heilbronn** Tel: 49-7131-72400 **Germany - Karlsruhe** Tel: 49-721-625370 **Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **Germany - Rosenheim** Tel: 49-8031-354-560 **Israel - Ra'anana** Tel: 972-9-744-7705 **Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781 **Italy - Padova** Tel: 39-049-7625286 **Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340 **Norway - Trondheim** Tel: 47-72884388 **Poland - Warsaw** Tel: 48-22-3325737 **Romania - Bucharest** Tel: 40-21-407-87-50 **Spain - Madrid** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **Sweden - Gothenberg** Tel: 46-31-704-60-40 **Sweden - Stockholm** Tel: 46-8-5090-4654 **UK - Wokingham** Tel: 44-118-921-5800 Fax: 44-118-921-5820