AVR® DA Training Manual

Low-Power Modes Using Curiosity Nano

Prerequisites

- **Hardware Prerequisites**
  - AVR128DA48 Curiosity Nano Board (DM164151) with a small hardware modification described in this document
  - Power Debugger (ATPOWERDEBUGGER)
  - Two Micro-USB cables: one for the Power Debugger and one for the Curiosity Nano Board
  - Two small wires with holes on both ends for pins
- **Software Prerequisites**
  - MPLAB® X 5.40
  - MCC plug-in for MPLAB X version 3.95
  - AVR library for MCC version 2.3.0
  - Atmel Studio newest version

Introduction

This document shows the differences in power consumption between a pure software approach and the use of CIPs for accomplishing different tasks.

This hands-on training covers the following topics:

- Assignment 1: Event System Compared to Interrupts
- Assignment 2: Software Accumulation Compared to Hardware Accumulation for Analog-to-Digital Converter (ADC)

Basic theory related to each assignment is introduced at the beginning of each assignment, the functionality is then verified through the Atmel Studio Data Visualizer.

Icon Key Identifiers

The following icons are used in this document to identify different assignment sections and to reduce complexity.

Info: Delivers contextual information about a specific topic.

Tip: Highlights useful tips and techniques.

To do: Highlights objectives to be completed.

Result: Highlights the expected result of an assignment step.

⚠WARNING Indicates important information.

Execute: Highlights actions to be executed out of the target when necessary.

Table of Contents

1. Assignment 1: Event System Compared to Interrupts

In this assignment, the power consumption of two different approaches will be compared. The first approach involves waking up the microcontroller from Sleep through a timer overflow interrupt that starts the ADC conversion to read the temperature sensor. When the conversion is done, the Result Ready Interrupt is triggered, and the result of the conversion is read.

In the second approach, the Event System (EVSYS) module will be used. This module allows the user to connect different peripherals through events. The previous approach will be modified such that the timer overflow will trigger an event that starts the ADC conversion that reads the temperature sensor. The end of the conversion will wake the microcontroller from Sleep to read the result.

1.1 Hardware Description

The Curiosity board is based on an AVR128DA48 microcontroller, a Nano embedded debugger, an LED, and a button. The Nano debugger allows the user to program the board and run debugging features without the need for external hardware. It also provides UART-USB bridge capabilities.

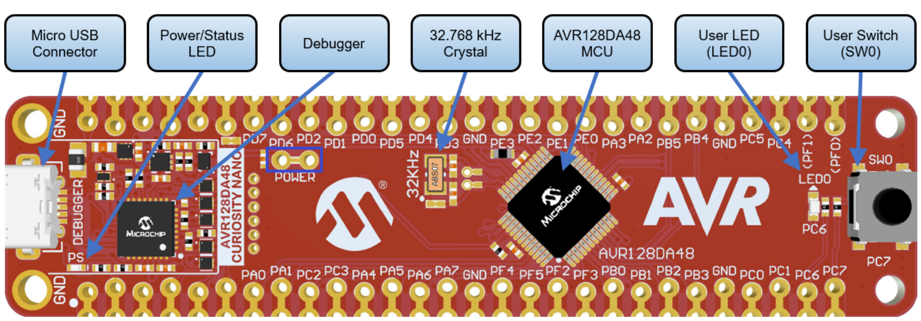
The Power Debugger is a programmer that also offers power outputs with configurable voltages and can be used to measure the current consumption of the devices connected to it.

1.2 Modifying the Curiosity Board to Obtain Accurate Power Measurements

To do: Modify the Curiosity Nano board by cutting a trace and soldering two pins.

If the power is observed form the micro-USB port, the Power debugger will consume some power and make the readings inaccurate.

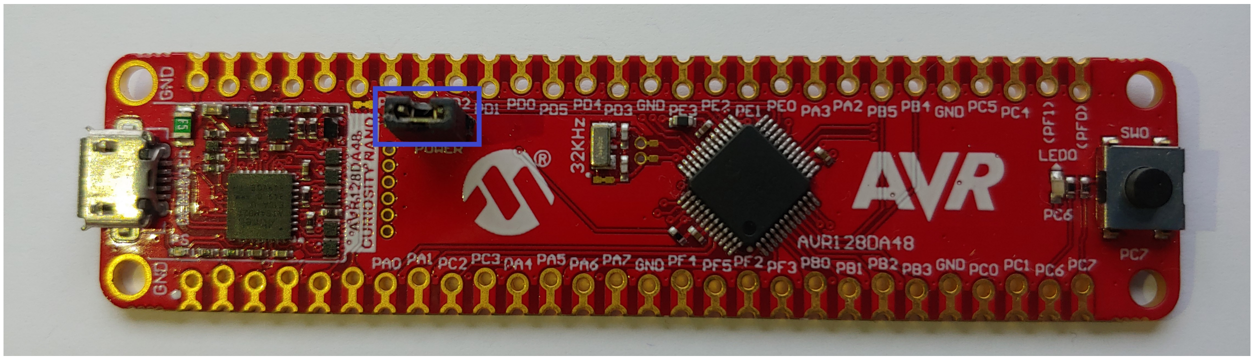
Figure 1-1. POWER Trace Location



There is a small trace labeled POWER on the Curiosity board. By using a sharp tool, cut the trace until the two holes are not in contact any more. Check this with a voltmeter.

After the trace has been cut, solder two pins in the holes. To program the board, the two pins have to be connected with a wire from now on.

Figure 1-2. Modified Curiosity Board



1.3 Explaining the Modules Used

The modules that will be used in this assignment are the Real-Time Clock (RTC), the ADC, the Voltage Reference (VREF) and the Event System (EVSYS).

The RTC will work as a timer that counts to one second.

The ADC is a module that transforms an analog voltage into a digital 12-bit number. It can be made to start a conversion from the EVSYS. It also raises an event and an interrupt when it finishes a conversion. The ADC will be used to read the internal temperature sensor.

The Voltage Reference is a peripheral that provides stabilized voltages to other peripherals. It also enables the temperature sensor if it is set to provide 2.048V reference to the ADC.

The Event System is a routing network enabling inter-peripheral communication without involving the CPU. The communication is done by sending events from one peripherals to another triggering actions. Events are latency-free and never lost, enabling better real-time and reliable applications.

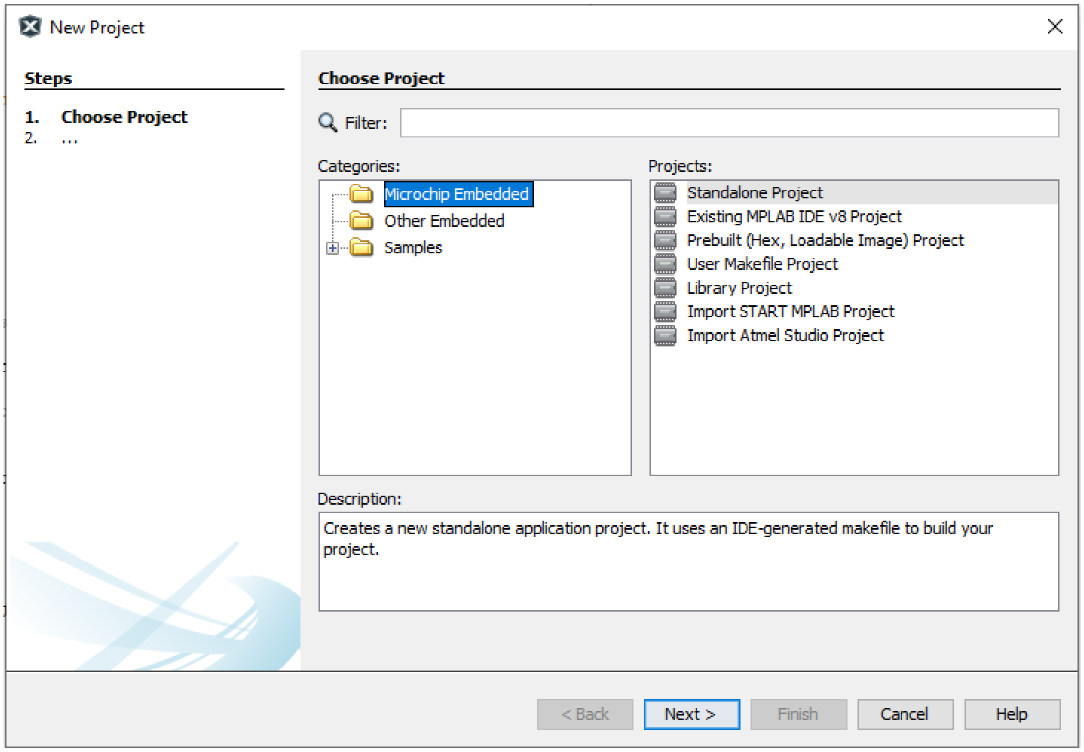
1.4 Creating the Software Implementation Program

To do: Configure the hardware modules that will be used in this project and add code to the project.

1. Create a new MPLAB X project for AVR128DA48.
   1.1. Open MPLAB X.
   1.2. Select *File → New Project* or the **New Project** button.
   Figure 1-3. New Project
   

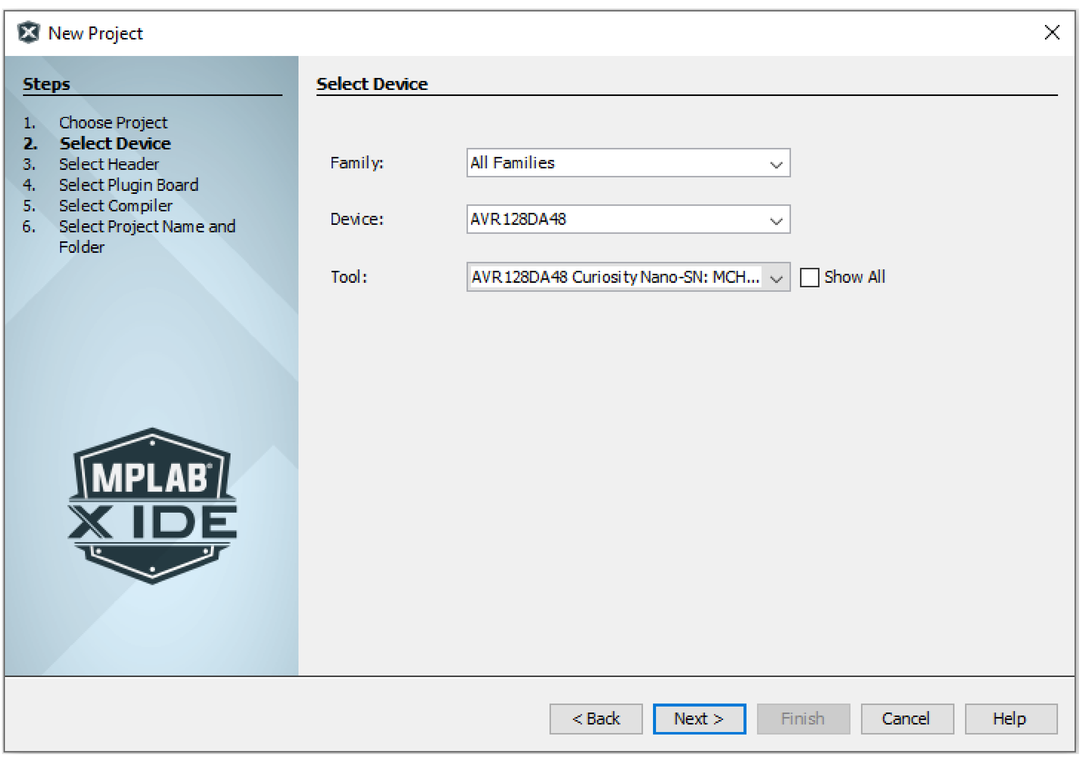
   1.3. Click **Next** (the Microchip Embedded Stand-alone Project is selected by default).

Figure 1-4. Project Type



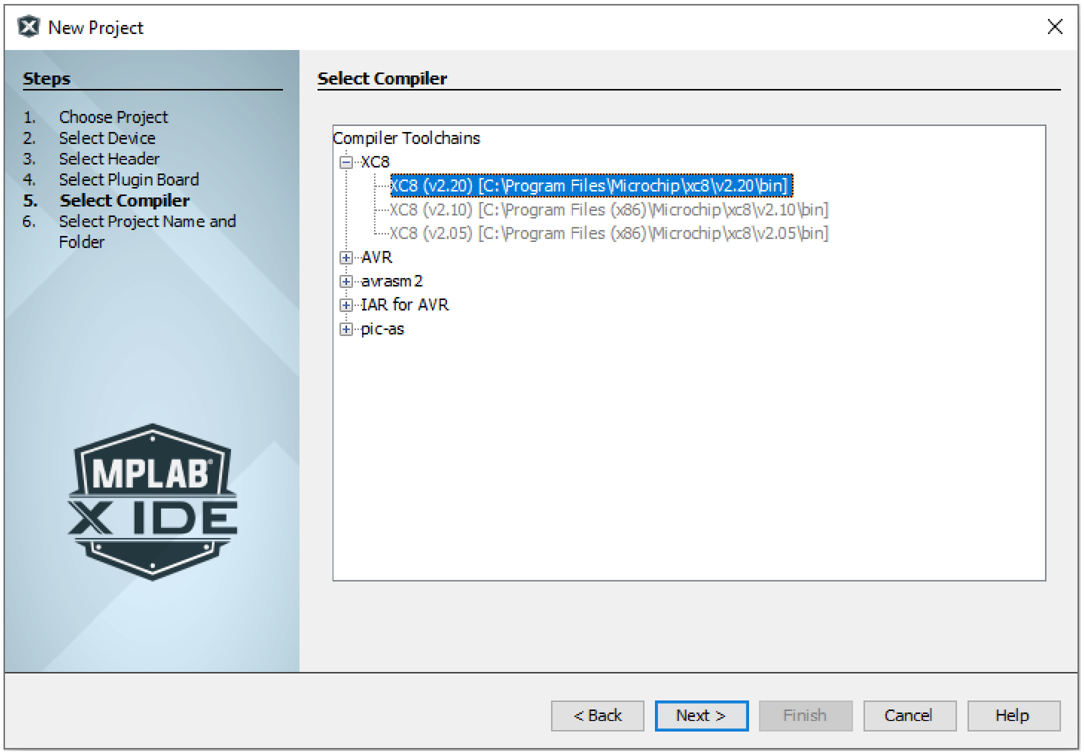
1.4. In the Device field, search for AVR128DA48. In the Tool category, select the Curiosity Nano board if it is connected to the computer, otherwise select **None**. Click **Next**.

Figure 1-5. Device Selection



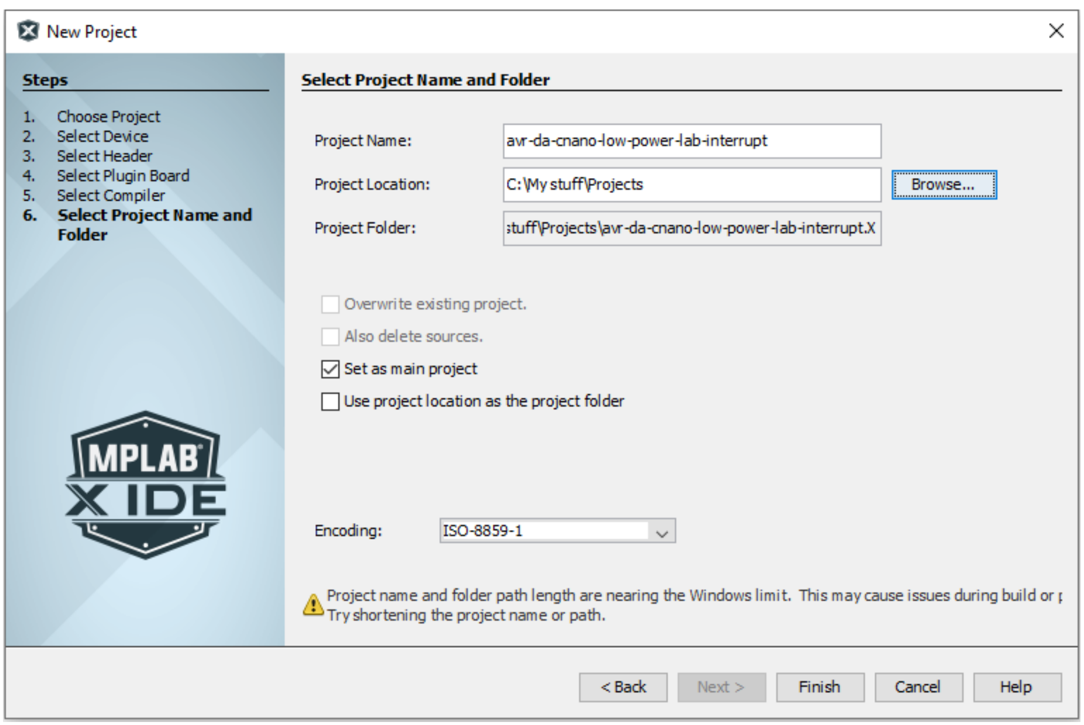
1.5. Select **XC8 2.20 compiler** and click **Next**.

Figure 1-6. Compiler Selection



1.6. Give a name to the project (and the location where to be saved) and click **Finish**.

Figure 1-7. Project Name

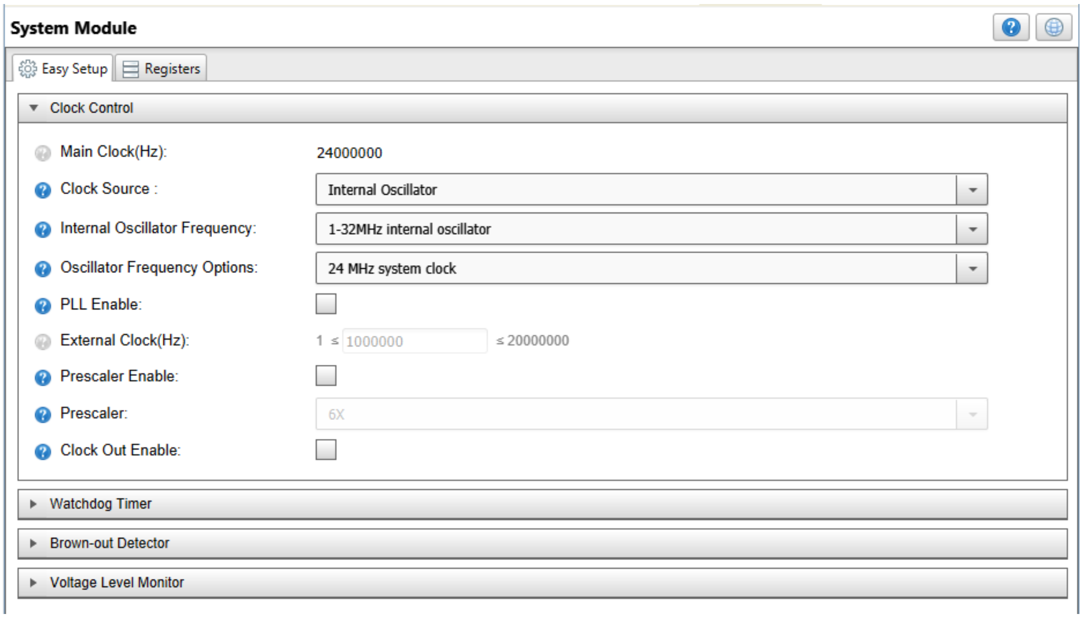


2. Open MPLAB Code Configurator (MCC) and configure the peripherals.

2.1. In System Module, choose the clock source as the internal oscillator with 24 MHz clock.

Info: Choose the Clock source as Internal Oscillator, the Oscillator Frequency Options as the 24 MHz system clock, and disable the Prescaler.

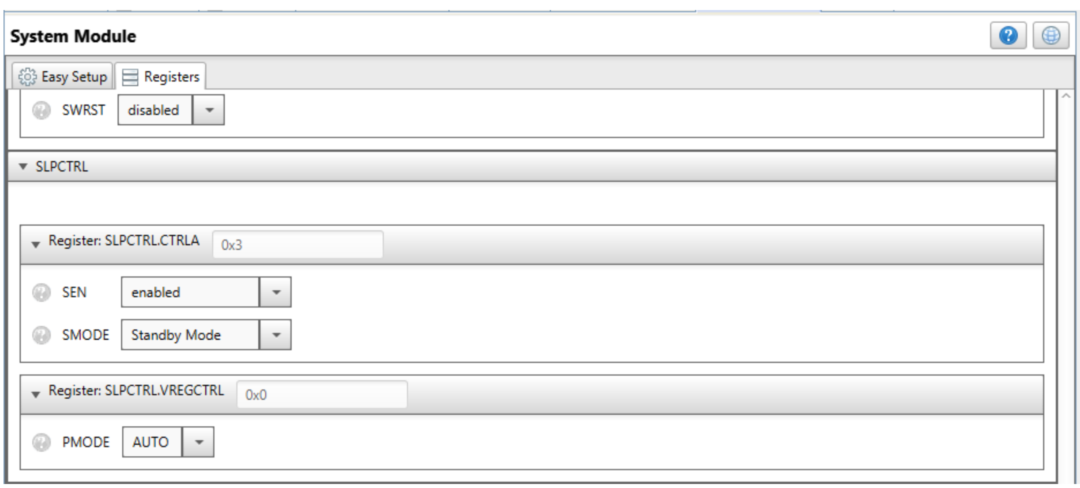
Figure 1-8. System Module Configuration



2.2. In the **Registers** tab of System Module, enable Sleep and set the mode to Standby.

Info: Go to the **Registers** tab in System Module and scroll down to SLPCTRL. Modify the CTRLA register to enable Sleep and select the mode.

Figure 1-9. Sleep Configuration

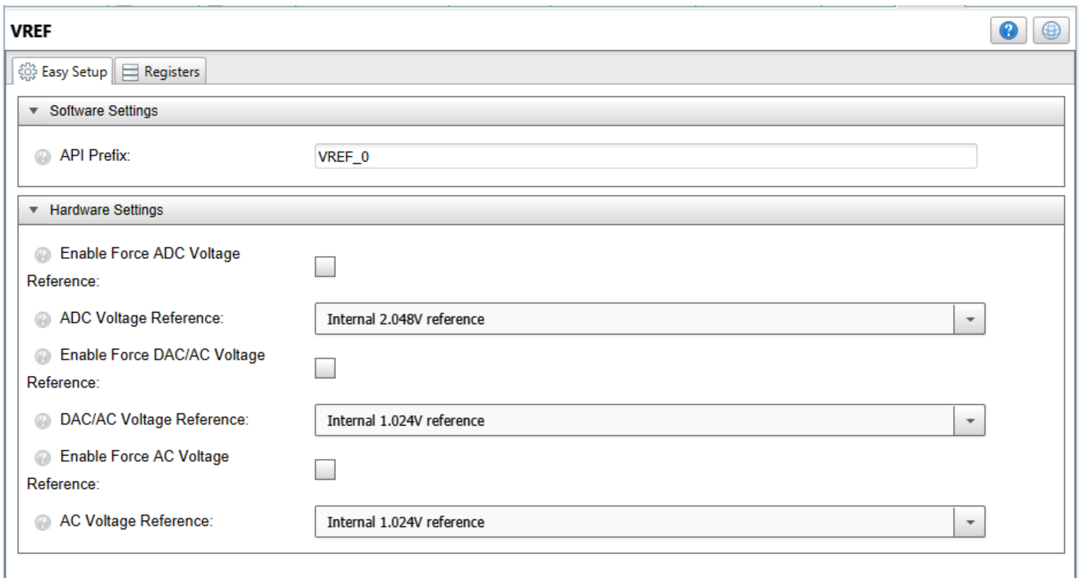


2.3. Add the VREF module from Device resources and configure it to provide 2.048V reference to the ADC.

Info: The ADC voltage reference must be configured to the internal 2.048V reference. Do not enable the Force ADC voltage option.

Info: The VREF is used to enable the temperature sensor of the microcontroller.

Figure 1-10. VREF Configuration



2.4. Add the ADC module from device resources and configure it in 12-bit mode, right-adjusted results, no accumulation, one tick for sample length, and to Run-In Standby (RUNSTBY).

Info: Right-shifted results option must be disabled. The sample length and sample accumulation number are selected in the **Hardware Settings** tab. The 12-bit mode is set by default. The RUNSTBY bit can be changed from the **Registers** tab and it is found in the CTRLA register.

Figure 1-11. ADC Configuration

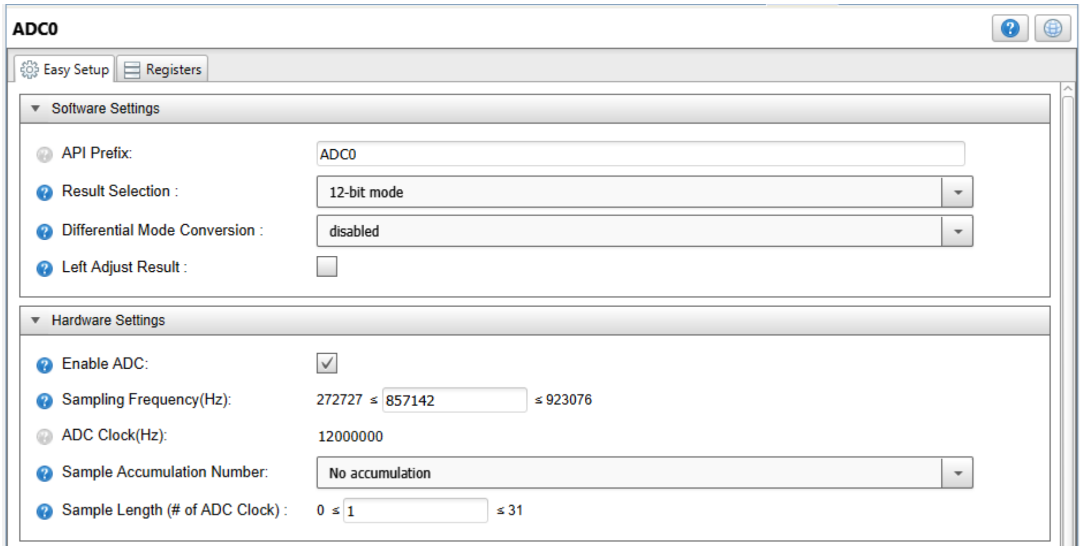
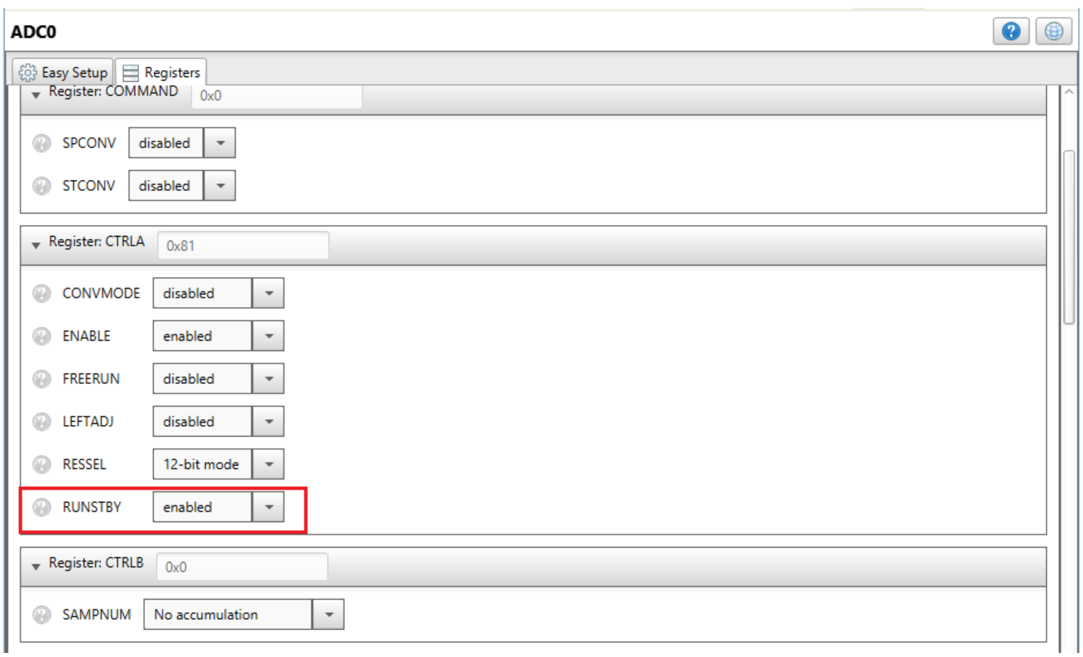


Figure 1-12. RUNSTBY ADC Configuration



2.5. Add the RTC module from device resources and configure it to use the internal 32.768 kHz oscillator, have a period of one second, a prescaling factor 1, and Run-In Standby (RUNSTBY) activated.

Info: The clock, prescaling factor, and period are set from the **Hardware Settings** tab, while the RUNSTBY bit is found in the register pane in the CTRLA register.

Figure 1-13. RTC Configuration

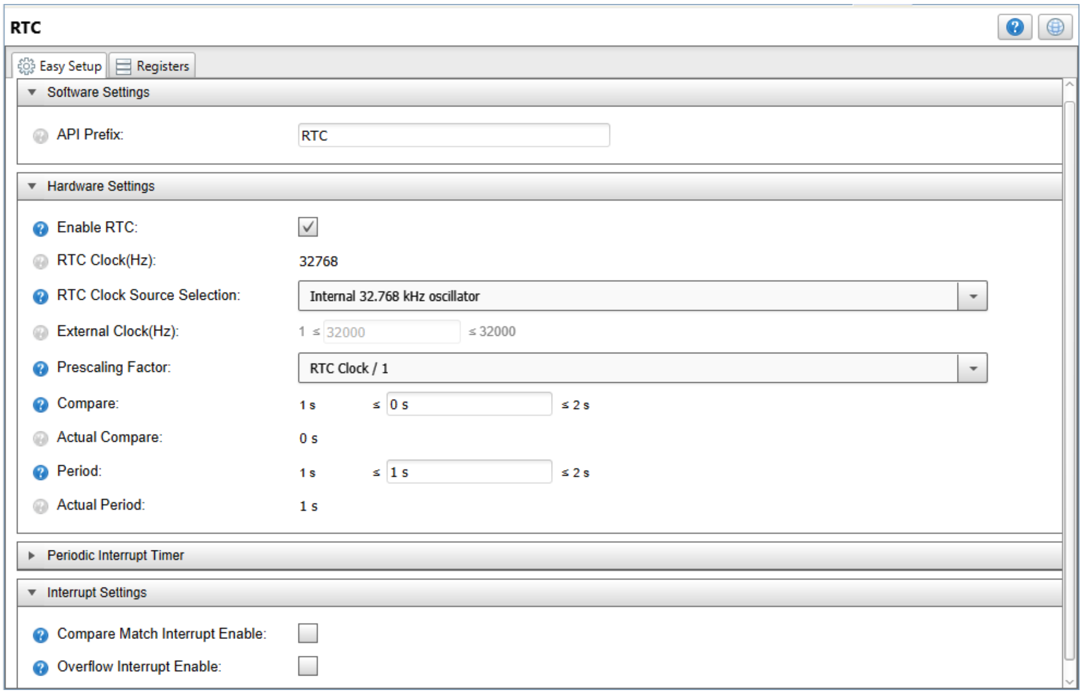
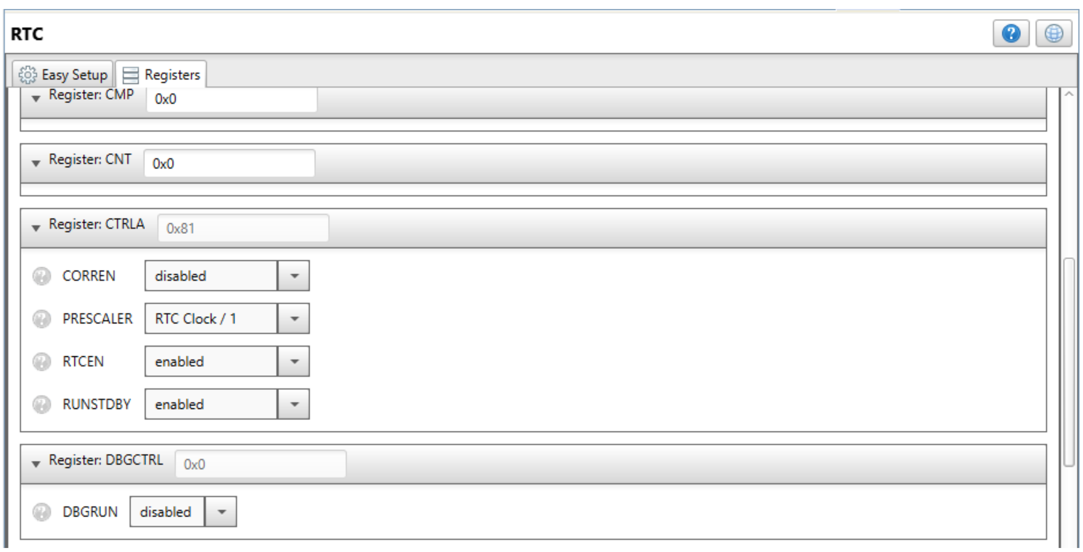


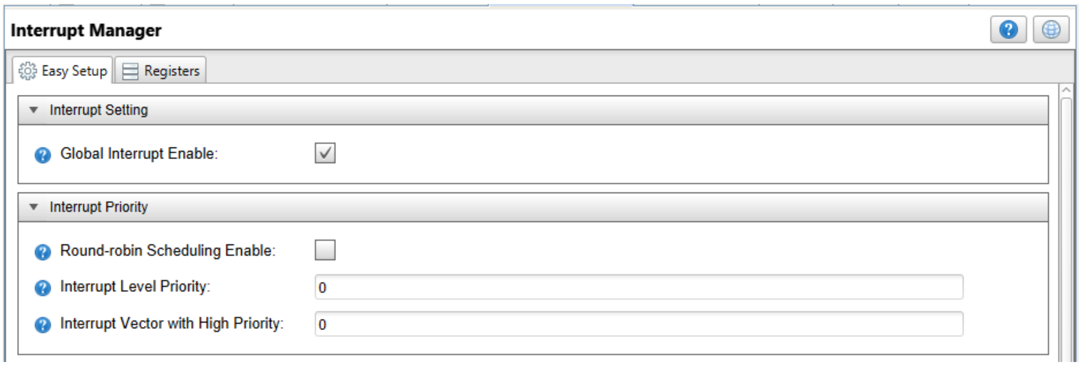
Figure 1-14. RUNSTBY RTC Configuration



2.6. Enable the Global Interrupts and enable the Result Ready Interrupt for the ADC and the overflow interrupt for the RTC.

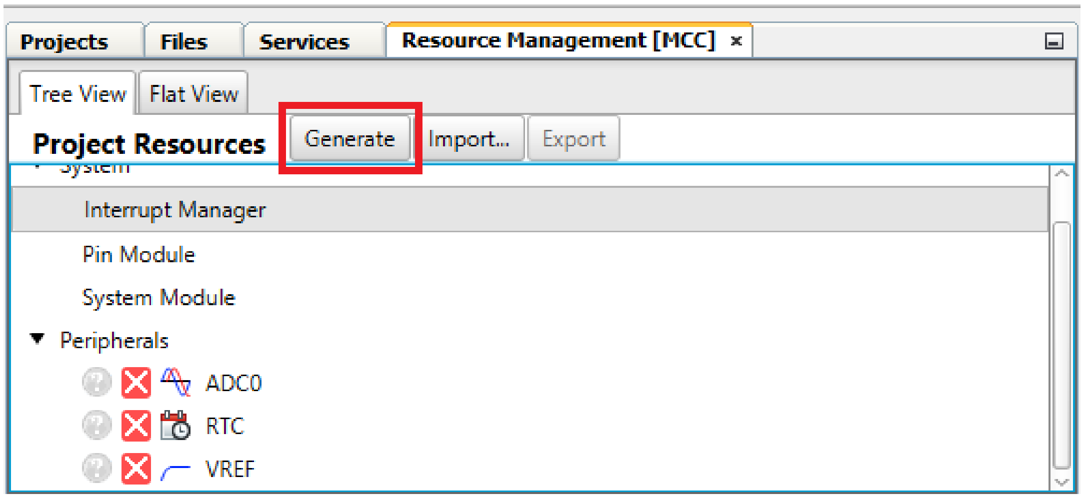
Info: The Global Interrupts are enabled from the Interrupt Manager, the ADC interrupt is enabled from the **Interrupt Settings** tab of the ADC module, and the RTC interrupt is enabled from the **Interrupt Settings** tab of the RTC module (seen in the pictures for the previous steps).

Figure 1-15. Interrupt Manager Configuration



2.7. Press the **Generate** button.

Figure 1-16. Generate Button



3. Add code to the generated files.

3.1. Add the following code to the `main.c` file.

```
#include <avr/io.h>
#include <avr/sleep.h>
#include "mcc_generated_files/mcc.h"

void TIMER_interrupt(void);

uint16_t result;

int main(void)
{

    SYSTEM_Initialize();
    RTC_SetOVFIsrCallback(TIMER_interrupt);

    while (1){
        sleep_cpu();
        result = ADC0_GetConversionResult();
    }
}

void TIMER_interrupt()
{
    ADC0_StartConversion(ADC_MUXPOS_TEMPSENSE_gc);
}
```

Info: `SYSTEM_Initialize()` is defined in `mcc.c`, `RTC_SetOVFIsrCallback()` is defined in `rtc.c` and `ADC0_StartConversion()` and `ADC0_GetConversionResult()` are defined in `adc.c`.

- `SYSTEM_Initialize()` sets all the configuration registers for the CPU and peripherals. The function is generated by MCC.
- `RTC_SetOVFIsrCallback()` is a function that sets the callback (the function that will be called when the interrupt is triggered) to the `TIMER_interrupt()` functioned defined in the main.
- The `TIMER_interrupt()` function starts the conversion for the ADC.
- The `ADC0_StartConversion()` function starts the conversion on the temperature channel of the ADC.
- The `ADC0_GetConversionResult()` returns the last result of the conversion.

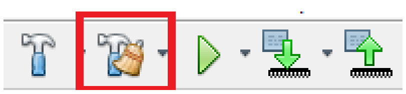
3.2. In the `pin_manager.c` file found in *Source Files → MCC Generated files → src* replace the code for the direction of the pins with:

```
PORTA.DIR = 0xFF;
PORTB.DIR = 0xFF;
PORTC.DIR = 0x3F;
PORTD.DIR = 0xFF;
PORTE.DIR = 0xFF;
PORTF.DIR = 0xFF;
```

Info: This is done to prevent floating pins that will disturb the power readings.

3.3. Press the **Clean and Build** button from the toolbar and verify that the program builds without errors.

Figure 1-17. Clean and Build



View Code Examples on GitHub
Click to browse repositories

1.5 Creating the Event System Implementation Project

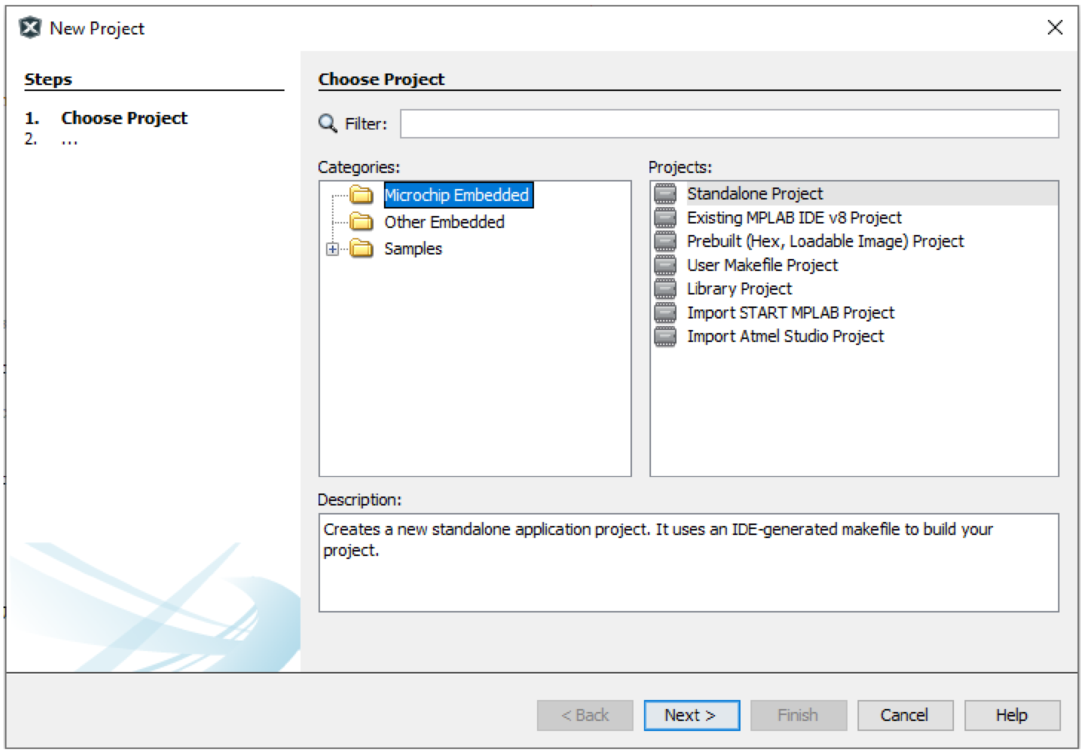
To do: Configure the hardware modules that will be used in this project and add code to the project.

1. Create a new MPLAB X project for AVR128DA48.
   1.1. Open MPLAB X.
   1.2. Select *File → New Project* or the **New Project** button.

Figure 1-18. New Project

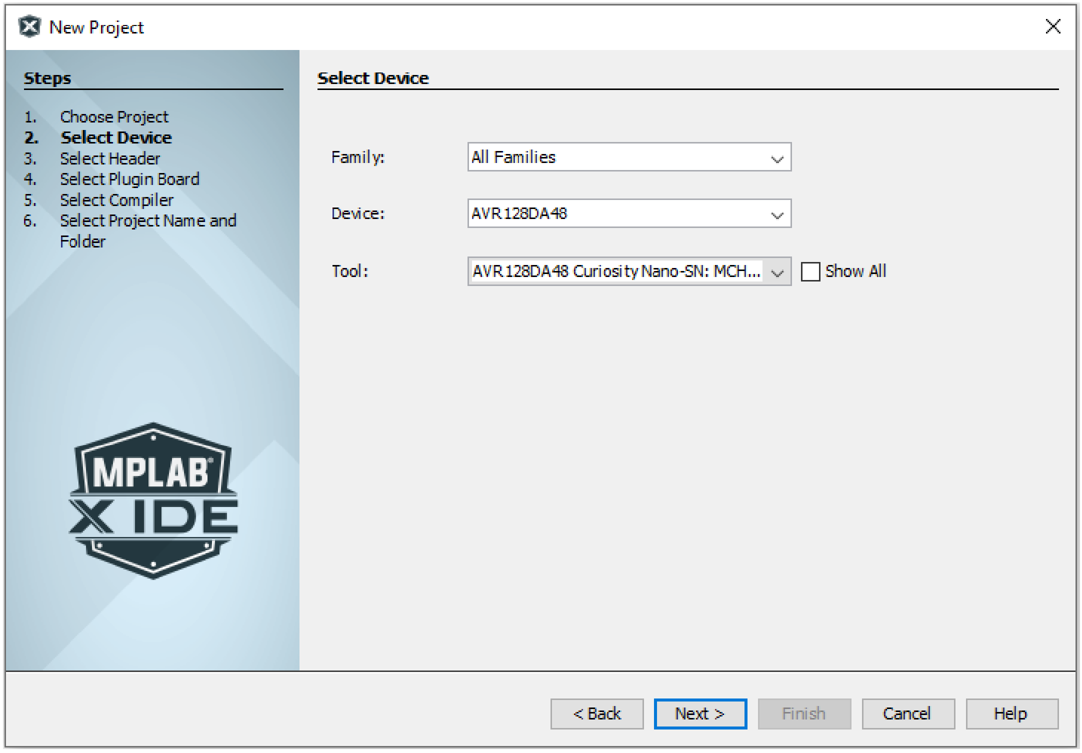
1.3. Click **Next** (the Microchip Embedded Stand-alone Project is selected by default).

Figure 1-19. Project Type



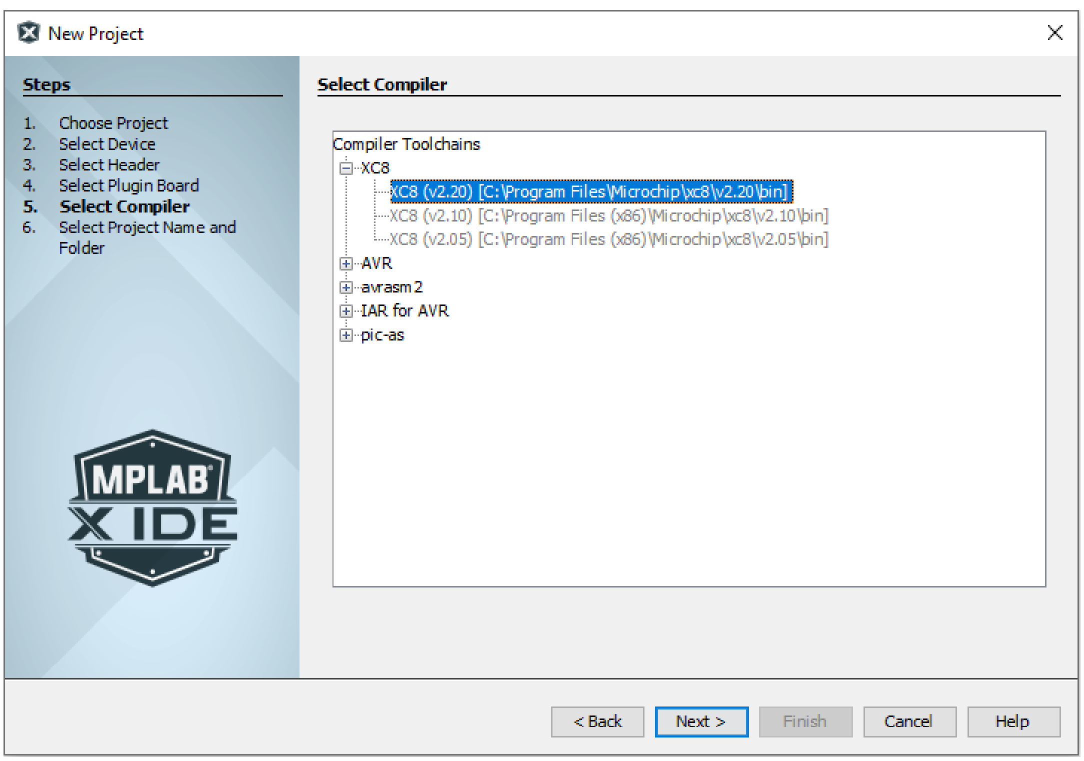
1.4. In the Device field search for AVR128DA48. In the Tool category, select the Curiosity Nano board if it is connected to the computer, otherwise select **None**. Click **Next**.

Figure 1-20. Device Selection



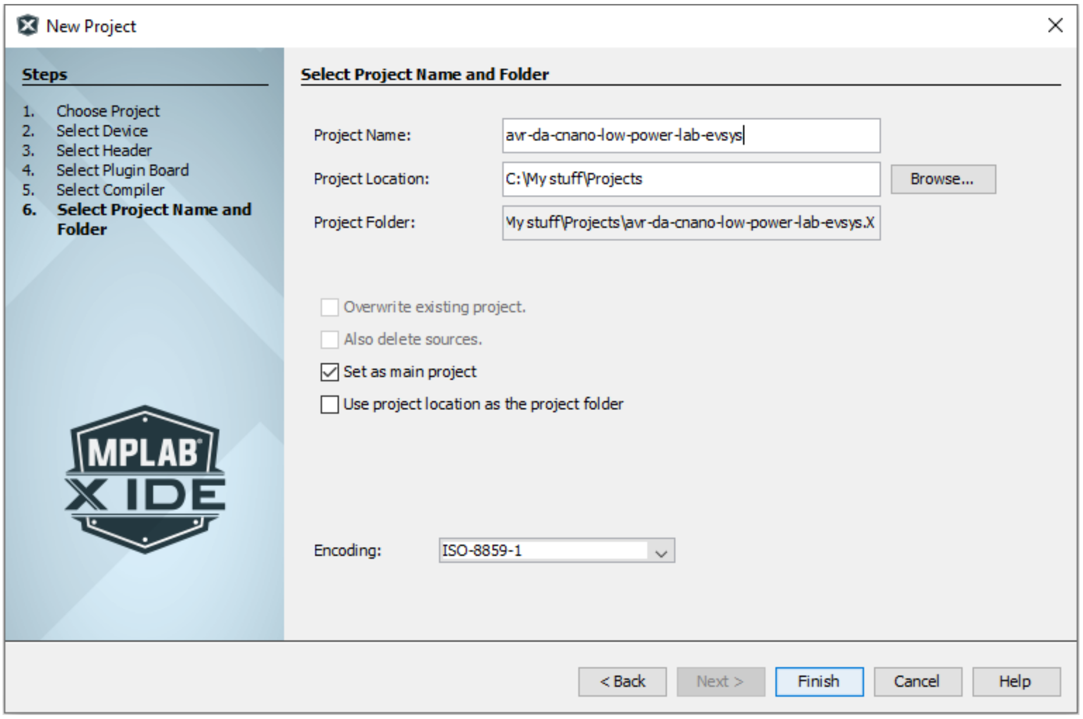
1.5. Select **XC8 2.20 compiler** and click **Next**.

Figure 1-21. Compiler Selection



1.6. Give a name to the project (and the location where to save) and click **Finish**.

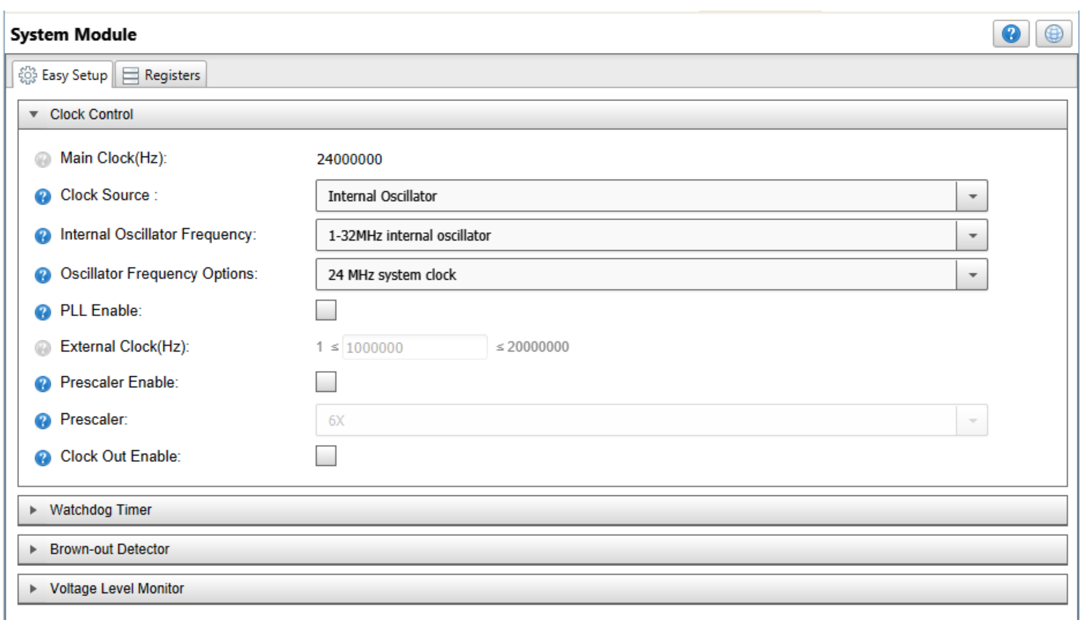
Figure 1-22. Project Name



2. Open MPLAB Code Configurator (MCC) and configure the peripherals.
   2.1. In System Module, choose the clock source as the internal oscillator with 24 MHz clock.

> **Info:** Choose the Clock source as Internal Oscillator, the Oscillator Frequency Options as the 24 MHz system clock, and disable the Prescaler.

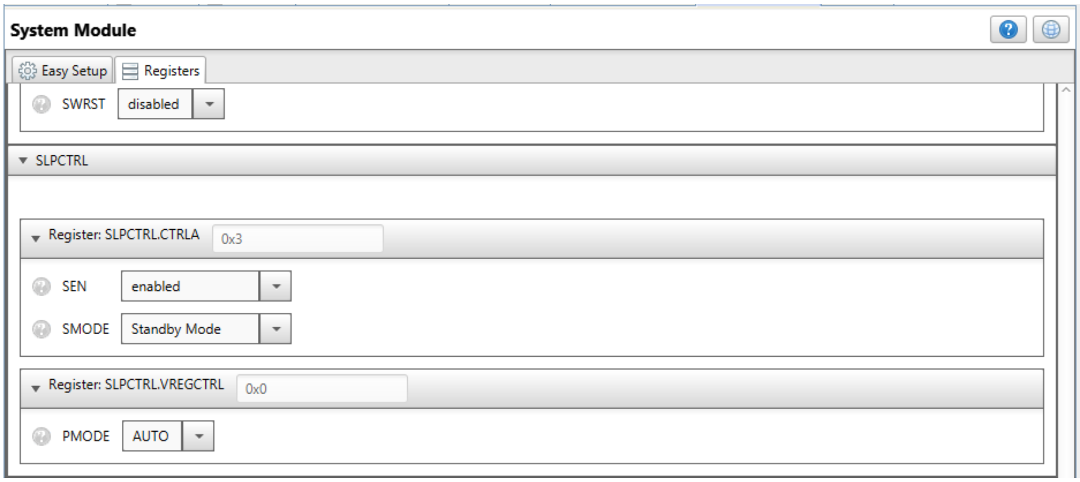
Figure 1-23. System Module Configuration



   2.2. In the **Registers** tab of System Module, enable Sleep and set the mode to Standby.

Info: Go to the **Registers** tab in System Module and scroll down to SLPCTRL. Modify the CTRLA register to enable Sleep and select the mode.

Figure 1-24. Sleep Configuration

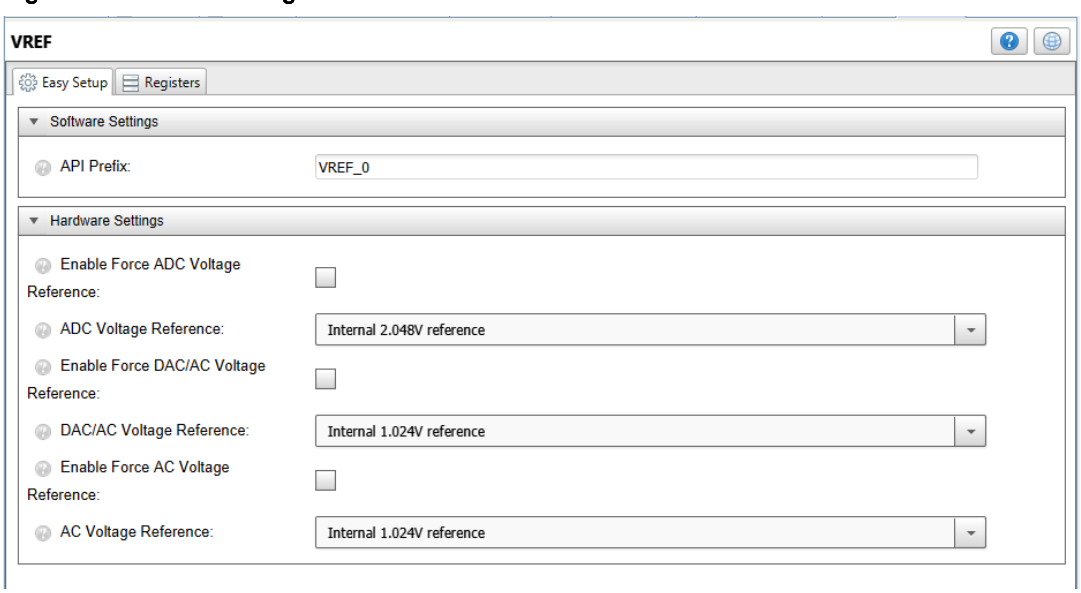


2.3. Add the VREF module from Device resources and configure it to provide 2.048V reference to the ADC.

Info: The ADC voltage reference must be configured to the internal 2.048V reference. Do not enable the Force ADC voltage option.

Info: The VREF is used to enable the temperature sensor of the microcontroller.

Figure 1-25. VREF Configuration



2.4. Add the ADC module from device resources and configure it in 12-bit mode, right-adjusted results, no accumulation, one tick for sample length and to Run-In Standby (RUNSTBY). Also enable the STARTEI bit so that a conversion starts when the event is received.

Info: The right-shifted results option must be disabled. The sample length and sample accumulation number are selected in the **Hardware Settings** tab. The 12-bit mode is set by default. The RUNSTBY bit can be changed from the **Registers** tab and is found in the CTRLA register. The STARTEI bit is found in the EVCTRL register in the **Registers** tab.

Figure 1-26. ADC Configuration

Figure 1-27. RUNSTBY RTC Configuration

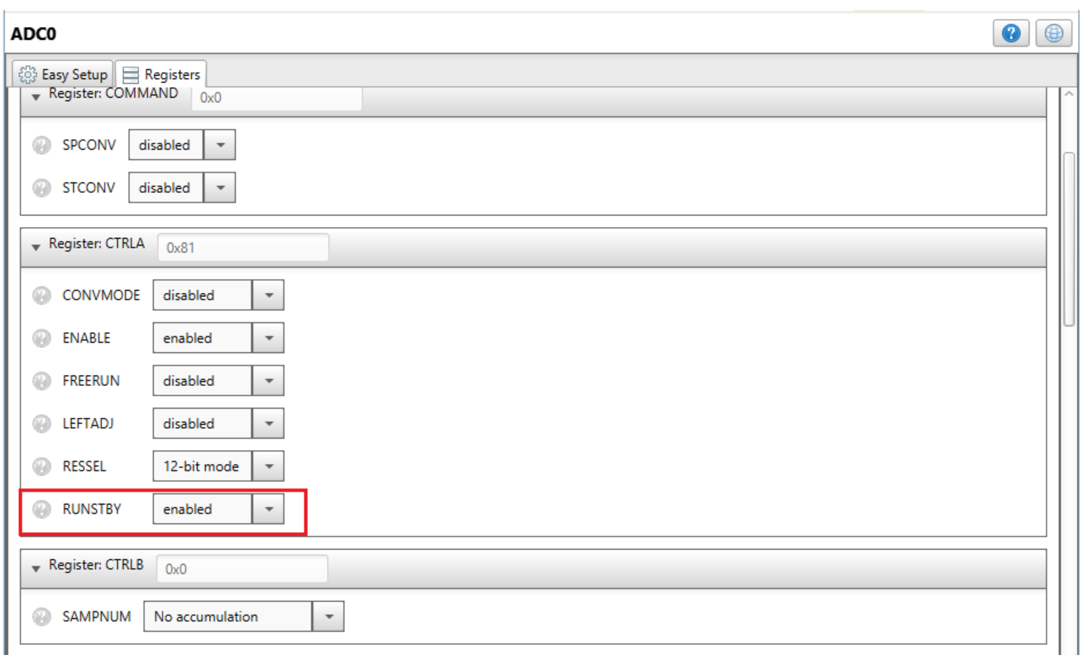
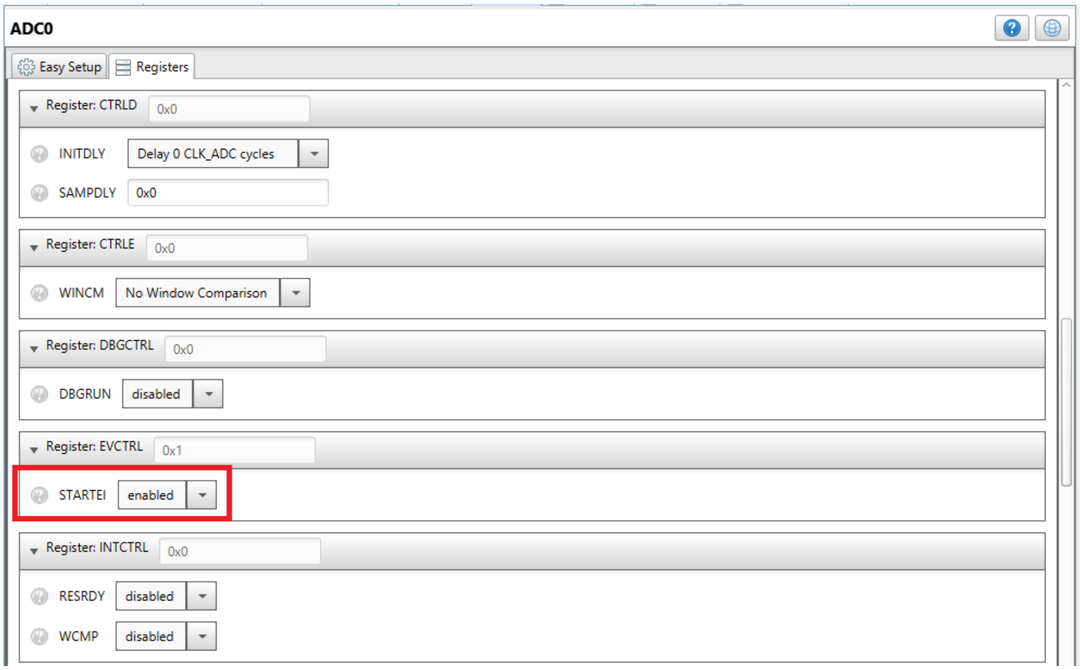


Figure 1-28. Start Conversion on Event ADC Configuration



2.5. Add the RTC module from device resources and configure it to use the internal 32.768 kHz oscillator, have a period of one second, prescaling factor 1 and Run-In Standby (RUNSTBY) activated.

Info: The clock, prescaling factor and period are set from the **Hardware Settings** tab, while the RUNSTBY bit is found in the register pane in the CTRLA register. Disable the Overflow Interrupt if it is enabled.

Figure 1-29. RTC Configuration

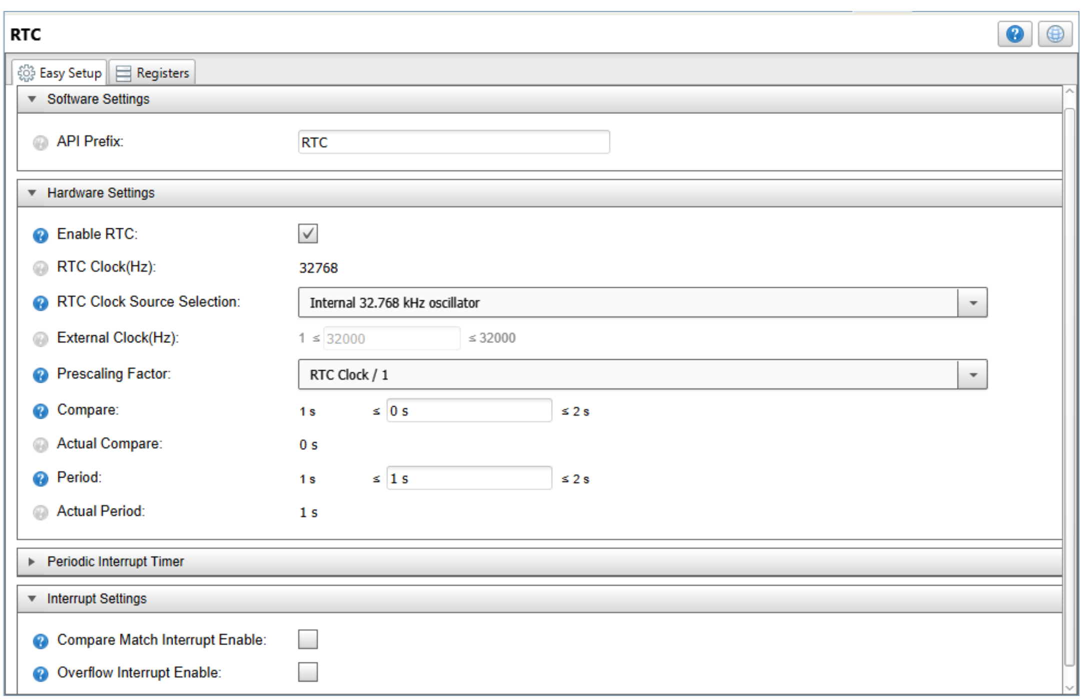
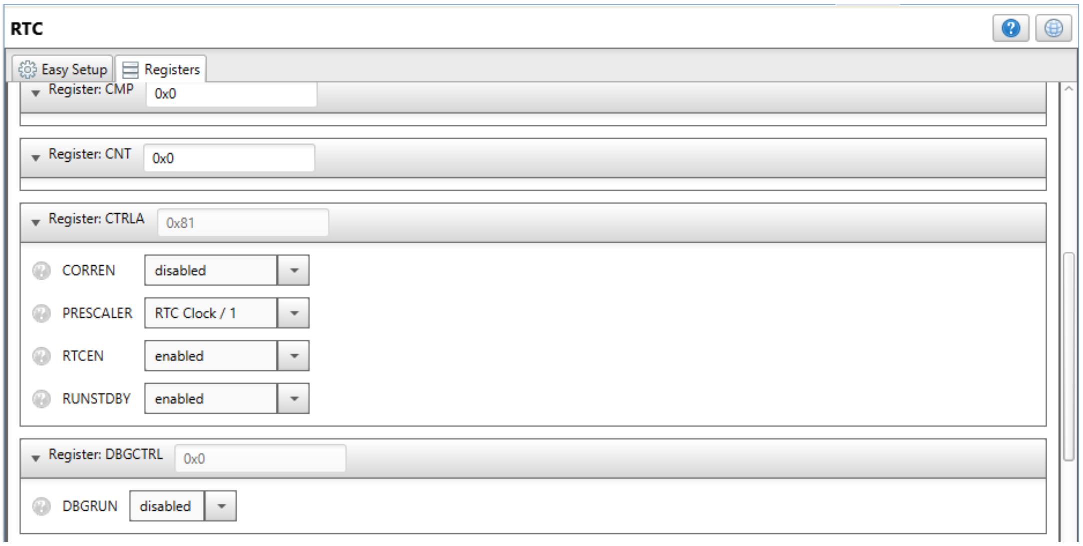


Figure 1-30. RUNSTBY RTC Configuration



2.6. Add the EVSYS module. Configure channel 0 event generator to be RTC_OVF and the event that will be triggered ADC0START. Configure channel 1 event generator to be ADC0_RESRDY and the event that will be triggered EVSYSEVOUTC.

Figure 1-31. EVSYS Configuration

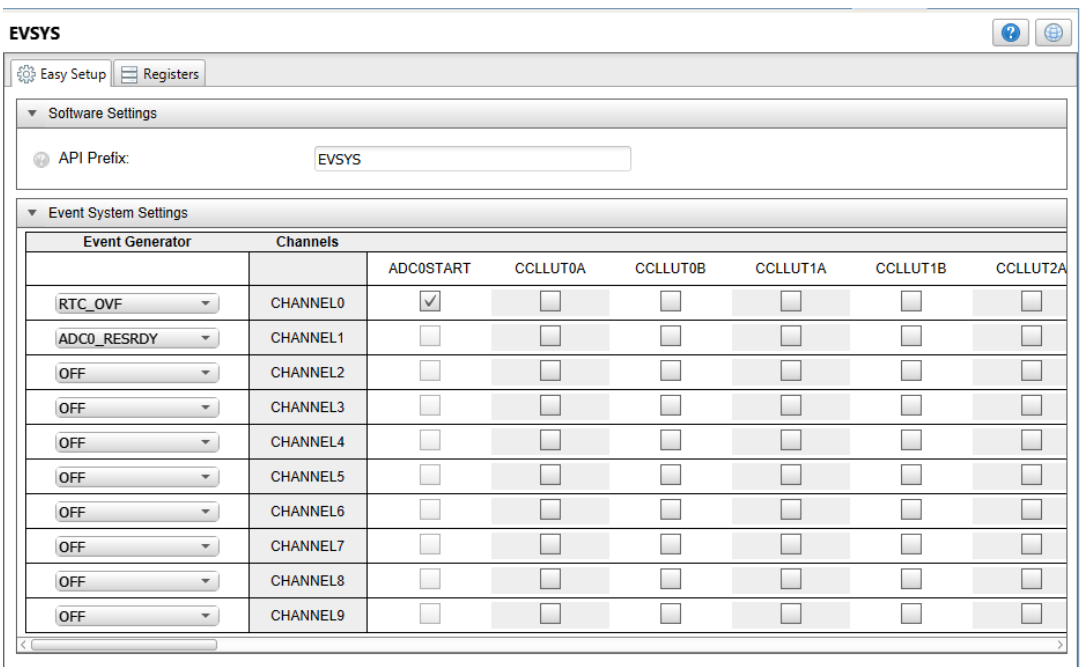
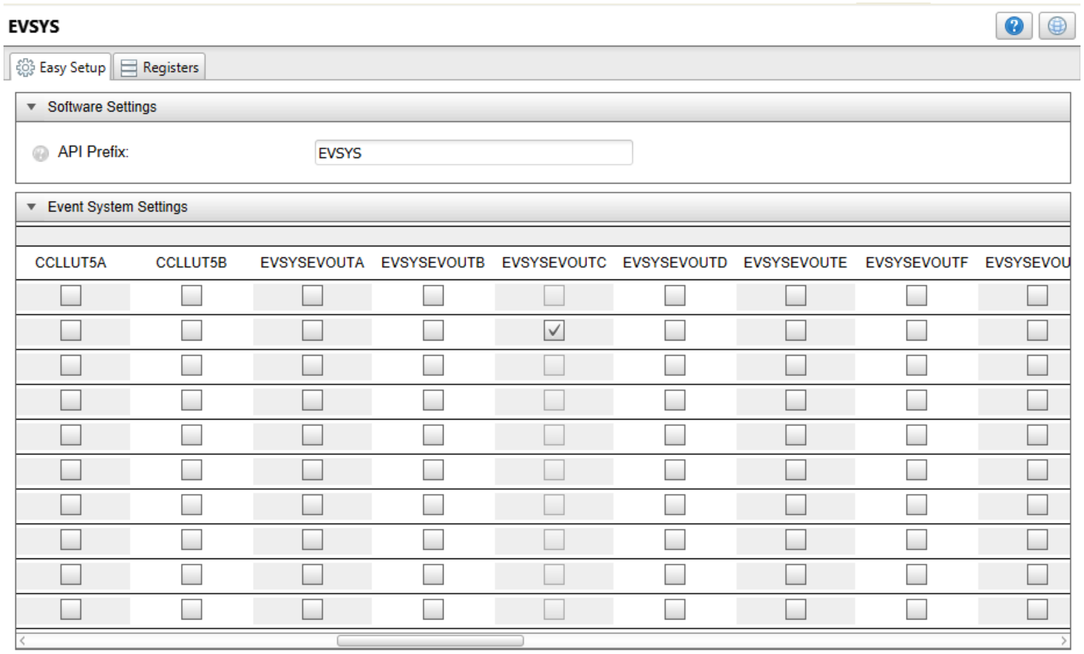
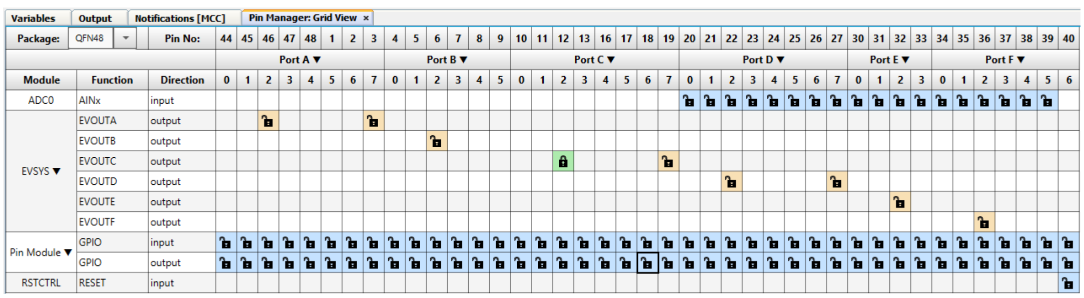


Figure 1-32. EVSYS Configuration



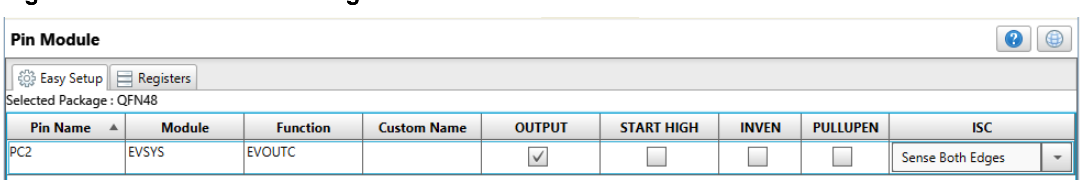
2.7. In Pin Manager: Grid View select **PC2** as EVOUTC for the EVSYS.

Figure 1-33. Pin Manager: Grid View



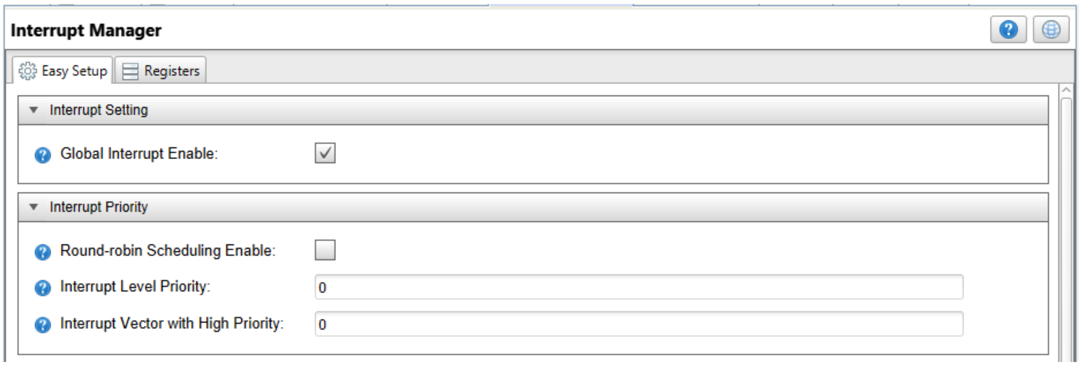
2.8. In the Pin Module, enable the interrupt-on-change of the PC2 pin to sense both edges.

Figure 1-34. Pin Module Configuration



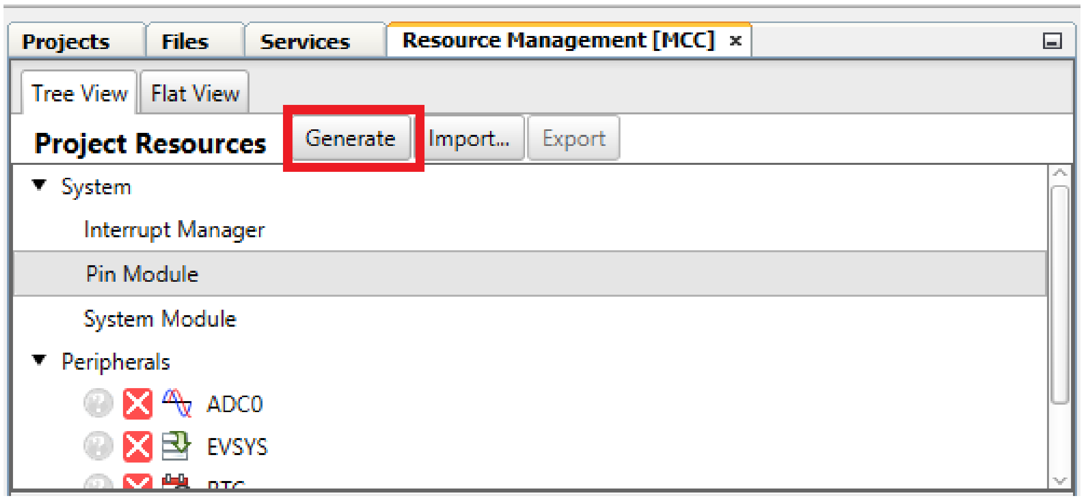
2.9. Enable Global Interrupts from Interrupt Manager.

Figure 1-35. Interrupt Manager Configuration



2.10. Press **Generate**.

Figure 1-36. Generate Button



3. Add code to the project.

3.1. Add code to the `main.c` file:

```
#include "mcc_generated_files/mcc.h"
#include <avr/sleep.h>


void GPIO_Interrupt(void);

uint16_t volatile result;

int main(void)
{

    SYSTEM_Initialize();
    ADC0.MUXPOS = ADC_MUXPOS_TEMPSENSE_gc;
    PORTC_PC2_SetInterruptHandler(GPIO_Interrupt);

    while (1){

        sleep_cpu();

    }
}

void GPIO_Interrupt(void)
{
    result = ADC0_GetConversionResult();
}
```

Info: `SYSTEM_Initialize()` is defined in `mcc.c`, `ADC0_GetConverisonResult()` is defined in `adc.c`, `PORTC_PC2_SetInterruptHandler()` is defined in `pin_manager.c`.

- `SYSTEM_Initialize()` sets all the configuration registers for the CPU and peripherals. The function is generated by MCC.
- The register ADC0.MUXPOS is set in main because the Event System starts the conversion immediately and there is no way to give a channel at that time.
- `PORTC_PC2_SetInterruptHandler(GPIO_Interrupt)` sets the function that will be called when the interrupt-on-change on pin PC2 is triggered.
- The `ADC0_GetConversionResult()` returns the last result of the conversion.

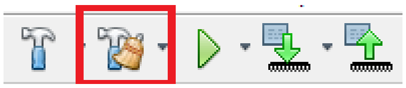
3.2. In the `pin_manager.c` file found in *Source Files → MCC Generated files → src* replace the code for the direction of the pins with:

```
PORTA.DIR = 0xFF;
PORTB.DIR = 0xFF;
PORTC.DIR = 0x3F;
PORTD.DIR = 0xFF;
PORTE.DIR = 0xFF;
PORTF.DIR = 0xFF;
```

Info: This is done to prevent floating pins that will disturb the power readings.

3.3. Press the **Clean and Build** button from the toolbar and verify that the program builds without errors.

Figure 1-37. Clean and Build



1.6 Programming the Device and Comparing Efficiency

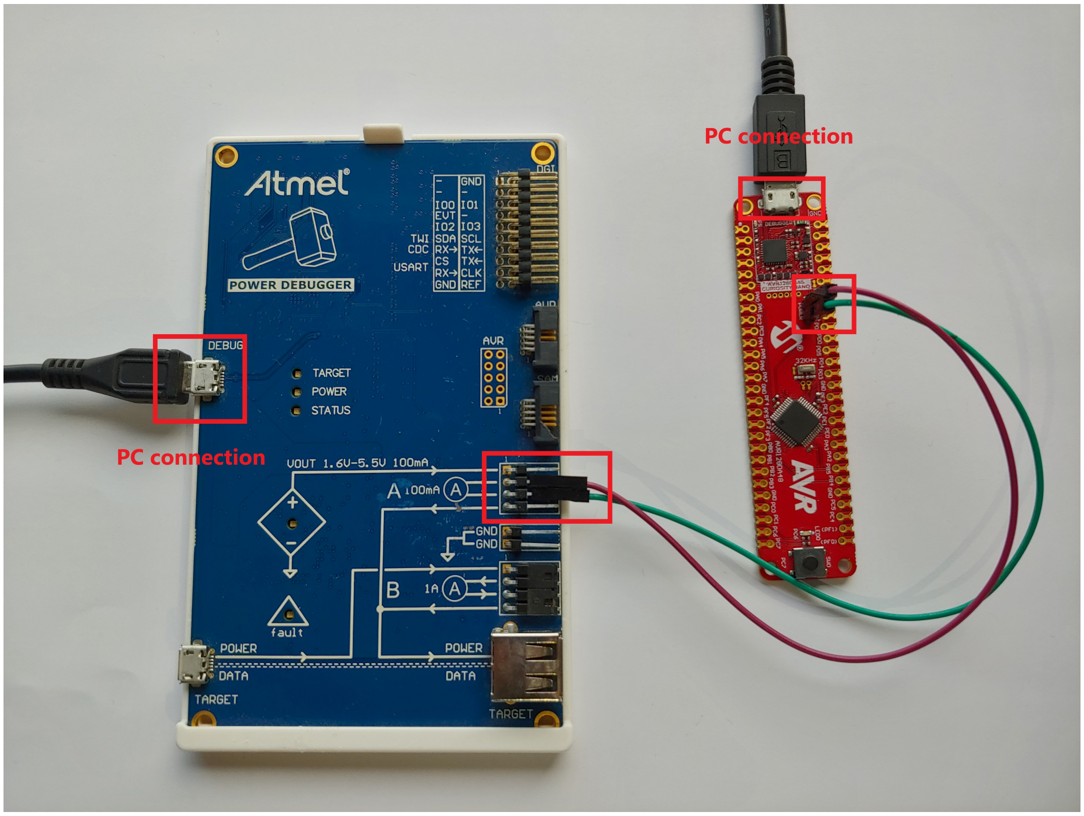
To do: Load the project onto the microcontroller and get power readings.

1. Get power consumption readings from the software implementation program.
   1.1. Connect the Curiosity Nano board to the computer and the two soldered pins together.
   1.2. With the software implementation program set as the main program, press the **Make and Program Device** button from the toolbar. A pop-up might appear that asks to select the tool that will be used as a programmer. Select the AVR128DA48 Curiosity Nano board.

Figure 1-38. Make and Program the Device

1.3. Connect the Power Debugger to the computer. Connect the first pin (the one closest to the Nano embedded debugger) to the inward arrow of the A channel ammeter and the second pin (the one closer to the microcontroller) to the outward pointing arrow of the A channel ammeter.

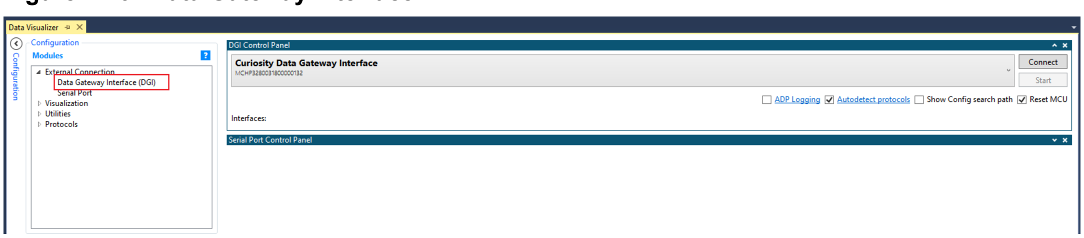
Figure 1-39. Hardware Connections



1.4. In Atmel Studio, go to *Tools → Data Visualizer*.

1.5. Two boxes will appear. Check the one called **DGI (Data Gateway Interface) Control Panel**. If this does not show up, the Configuration menu on the left side can be accessed, and in the *Modules → External Connection* section, the Data Gateway Interface can be found.

Figure 1-40. Data Gateway Interface



1.6. Press **Connect** and a box labeled 'power' will appear. Select it and then press the wheel to open the menu. Disable the B channel.

⚠WARNING In case the Power field does not appear, disconnect the board and connect it again.

Figure 1-41. Connect DGI

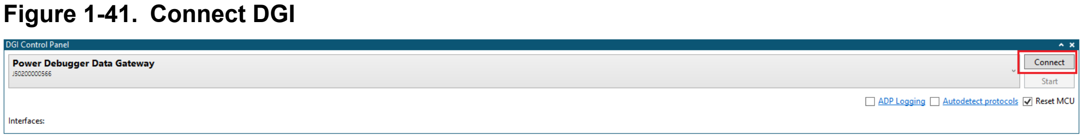
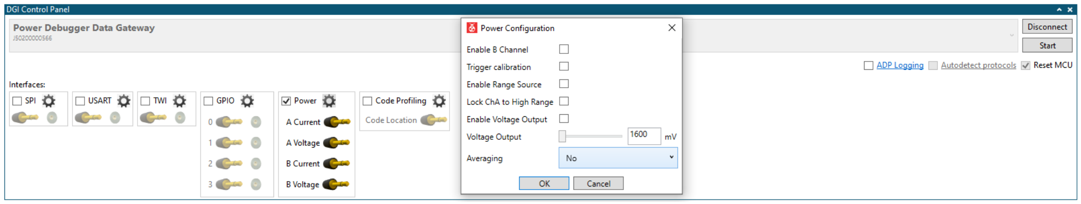


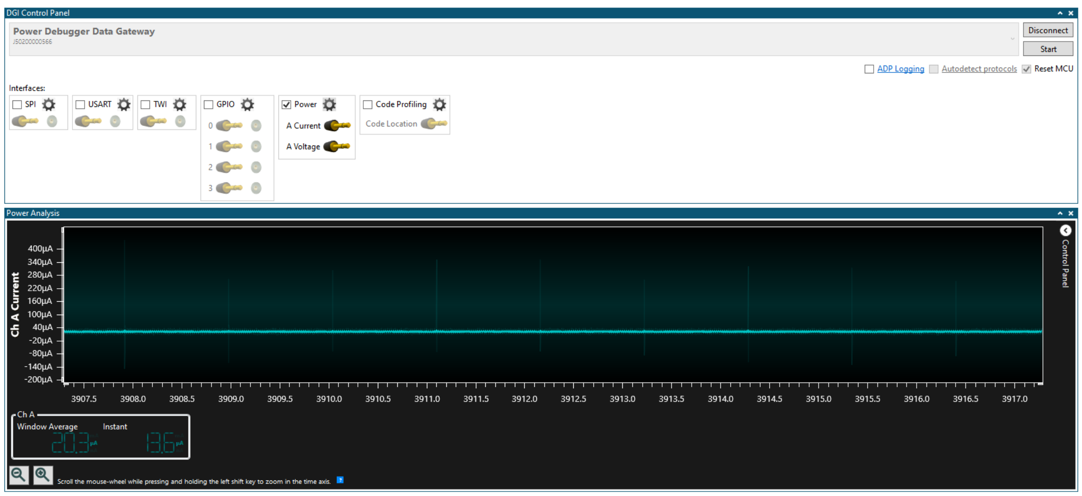
Figure 1-42. Power Settings



1.7. When everything is correctly configured, press **Start** and a new box that displays the power consumption in real time will appear.

Tip: Open the control panel on the left side of the box to adjust the display. Unchecking Show Zero will provide a clearer picture.

Figure 1-43. Start Monitoring



1.8. After the reading has stabilized, press **Stop** and then disconnect. The box with the power measurement will remain open.

2. Get power consumption readings from the Event System implementation program.

2.1. Connect the two soldered pins on the Curiosity board together.

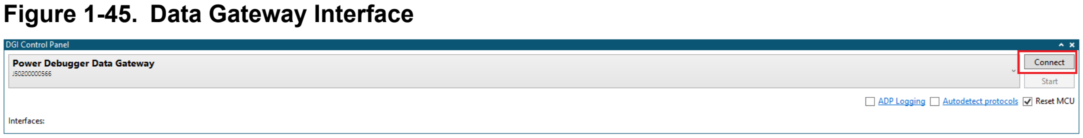
2.2. With the Event System implementation project set as main program, press the **Make and Program Device** button on the toolbar in MPLAB X.

Figure 1-44. Make and Program the Device

2.3. Connect the Curiosity Nano Board to the Power Debugger. See Step 1.3 in this chapter for details and pictures.

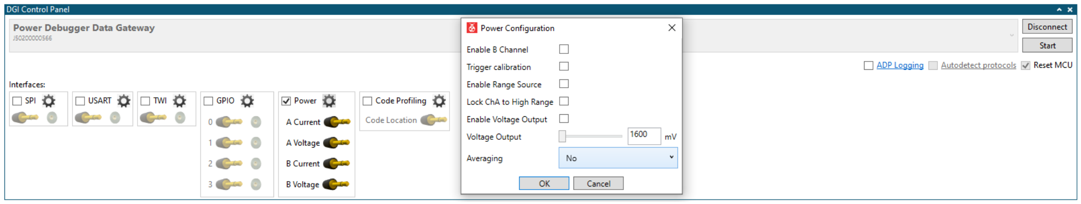
2.4. Go back to the Data Visualizer in Atmel Studio and in the DGI panel press **Connect**.

Figure 1-45. Data Gateway Interface



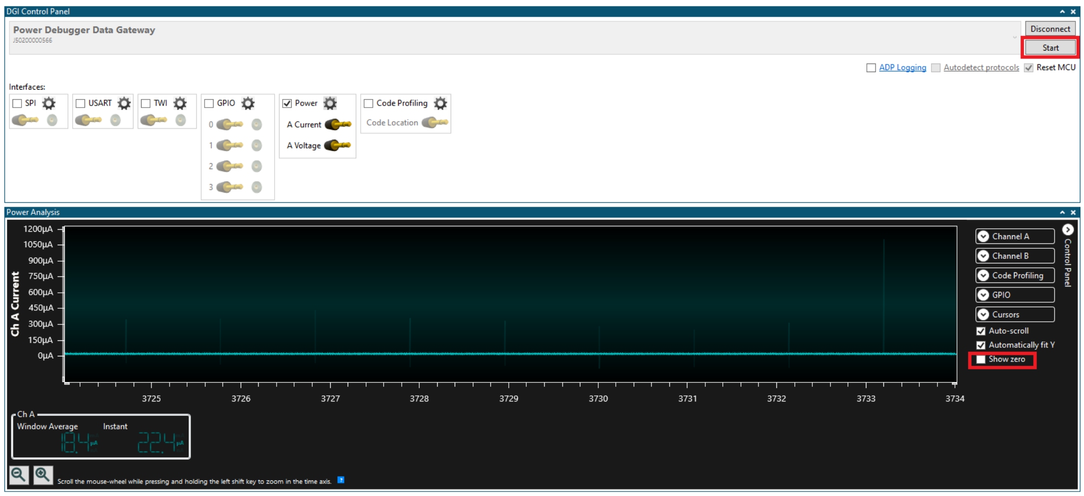
2.5. Check the power box and edit the settings to disable the B channel.

Figure 1-46. Power Settings



2.6. Press **Start** and a new box that displays the real-time power consumption will appear. Disable Show zero for a better picture.

Figure 1-47. Start Monitoring

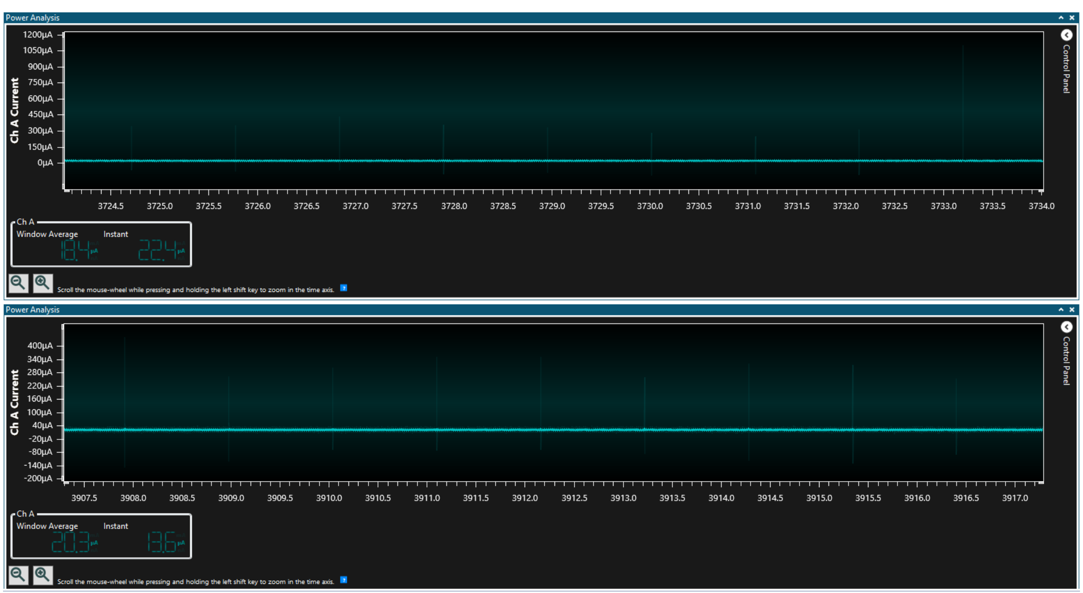


2.7. When the reading has stabilized, press **Stop** to have a fixed result.

3. Compare the two measurements.

3.1. Both power consumption graphs are open on the same screen. They can now be analyzed to observe the way the programs behaved.

Figure 1-48. Software Implementation Compared to EVSYS



Info: The top part displays the power consumption of the Event System implementation, while the bottom part displays the power consumption of the Interrupts implementation. As can be seen from the picture, the difference is fairly small of about 1.5 uA. This is due to the long amount of time the microcontroller spends in Sleep.

2. Assignment 2: Software Accumulation Compared to Hardware Accumulation for ADC

In this assignment, the power consumption of two different ADC accumulation techniques will be compared. Accumulating results in an ADC has the role of increasing the accuracy of the results and minimizing the effect of noise.

The first approach will trigger conversions again and again until 128 of these are completed. The results will be added to a variable that acts as an accumulator.

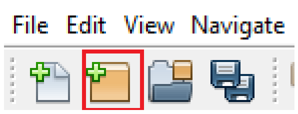
The second approach takes advantage of the built-in accumulator for the ADC. It will use the accumulated conversion mode of the ADC to run 128 conversions and add the results to the hardware accumulator.

2.1 Creating the Software Implementation Project

To do: Configure the hardware modules that will be used in this project and add code to the project.

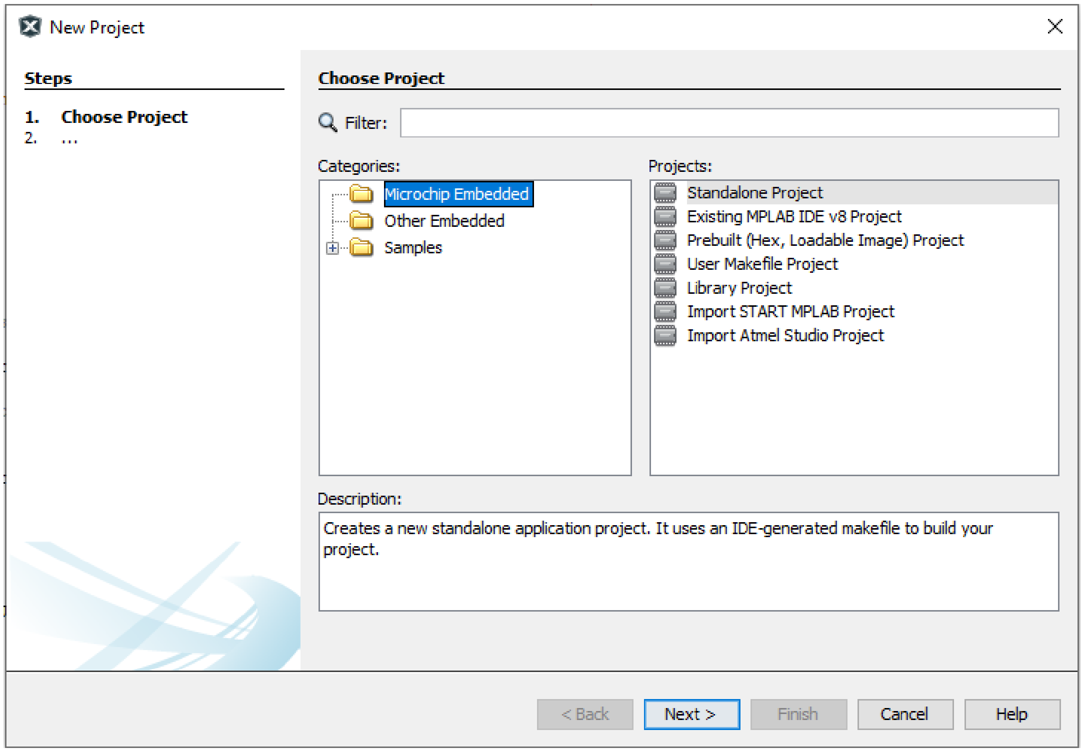
1. Create a new MPLAB X project for AVR128DA48.
   1.1. Open MPLAB X.
   1.2. Select *File → New Project* or the **New Project** button.

   Figure 2-1. New Project

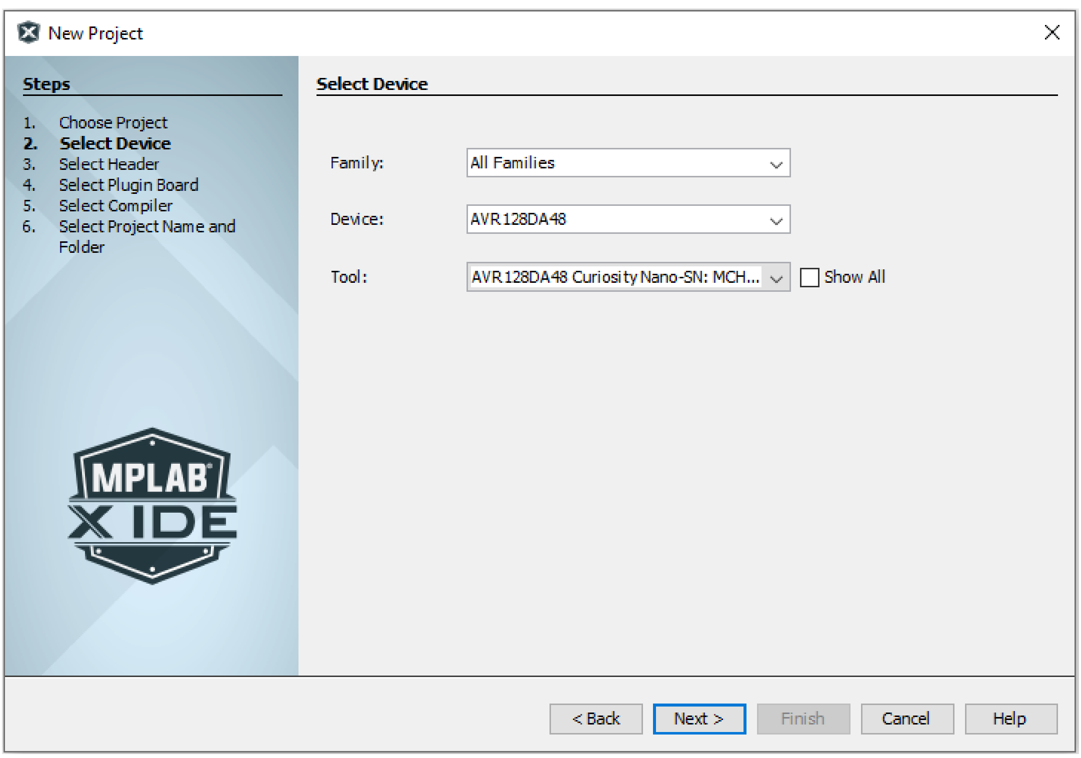
   1.3. Click **Next** (Microchip Embedded Stand-alone Project is selected by default).

Figure 2-2. Project Type



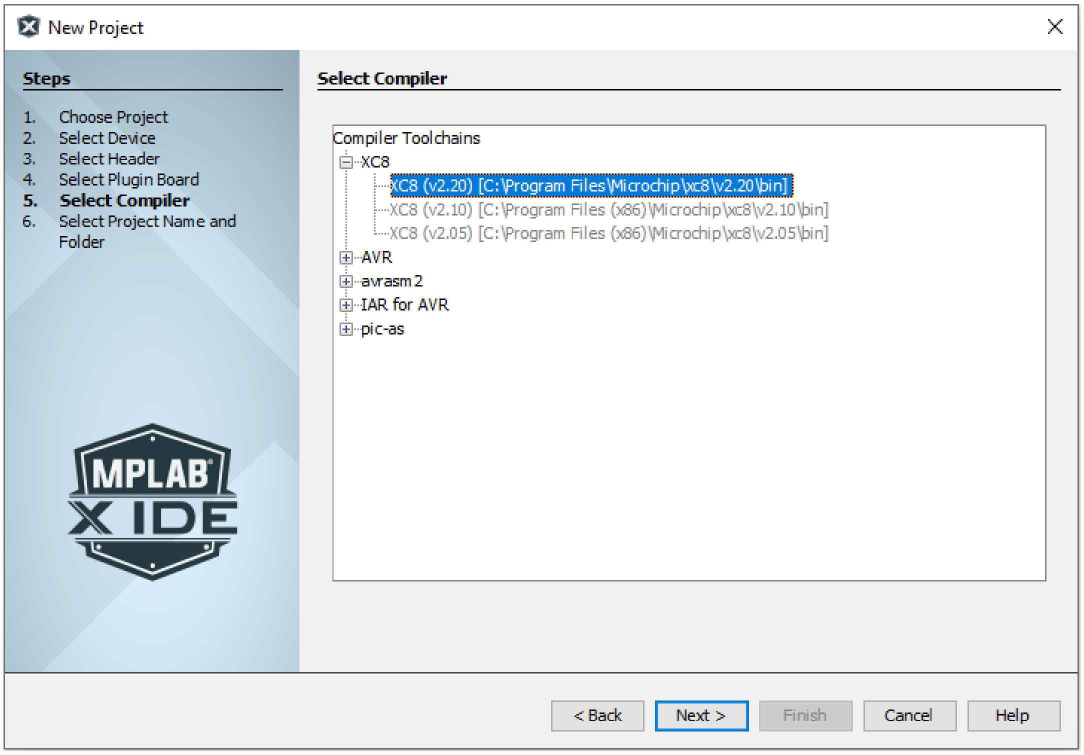
1.4. In the Device field search for: AVR128DA48. In the Tool category, select the **Curiosity Nano board** if it is connected to the computer, otherwise select **None**. Click **Next**.

Figure 2-3. Device Selection



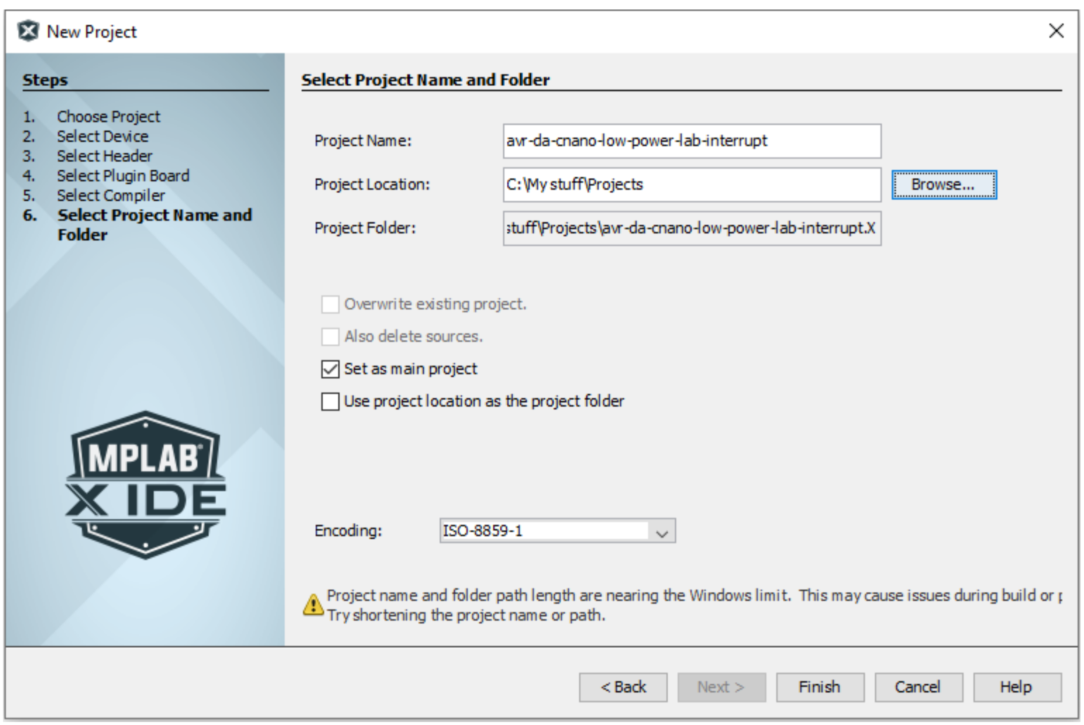
1.5. Select the **XC8 2.20 compiler** and click **Next**.

Figure 2-4. Compiler Selection



1.6. Give a name to the project (and the location where to be saved) and click **Finish**.

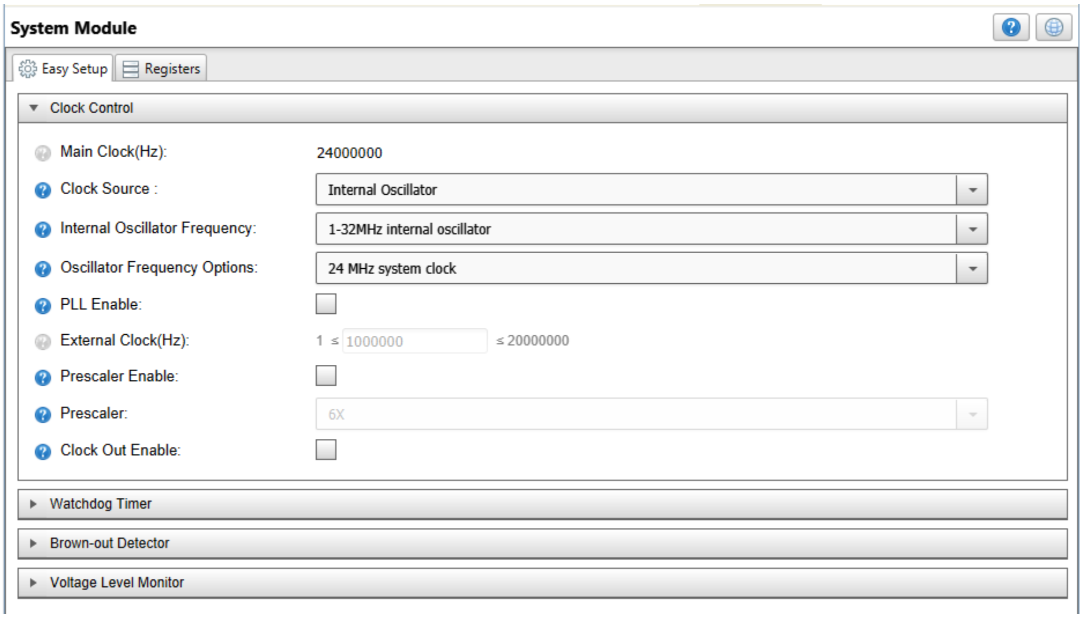
Figure 2-5. Project Name



2. Open MPLAB Code Configurator (MCC) and configure the peripherals.
   2.1. Configure the System Module to run on the internal oscillator at 24 MHz.

Info: The clock source will be the internal oscillator and the frequency 24 MHz. The prescaler option must be disabled.

Figure 2-6. System Module Configuration

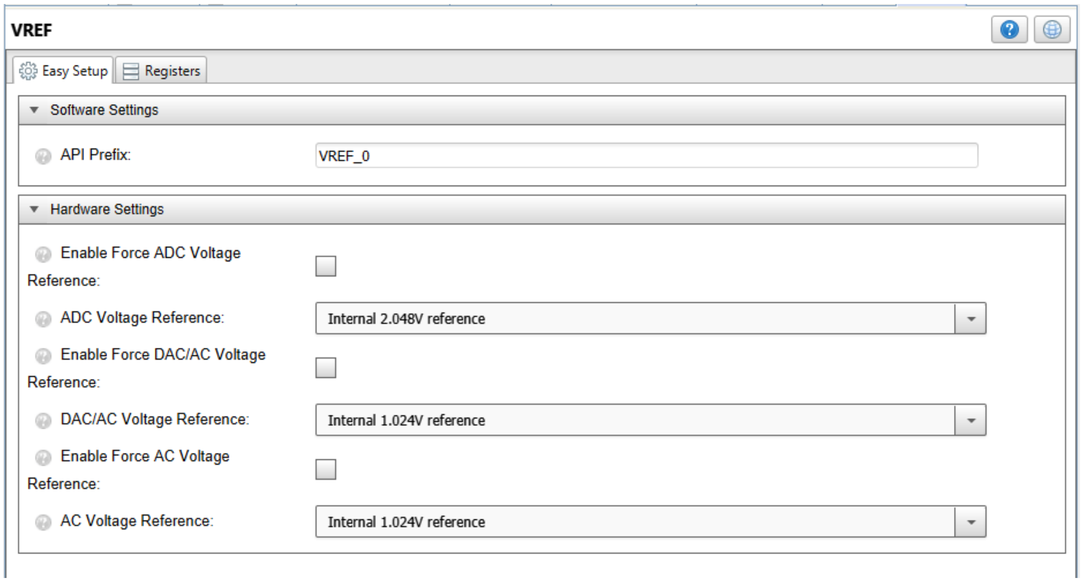


2.2. Add the VREF module from Device resources and configure it to provide 2.048V reference to the ADC.

Info: The ADC voltage reference must be configured to the internal 2.048V reference. Do not enable the Force ADC Voltage option.

Info: The VREF is used to enable the temperature sensor of the microcontroller.

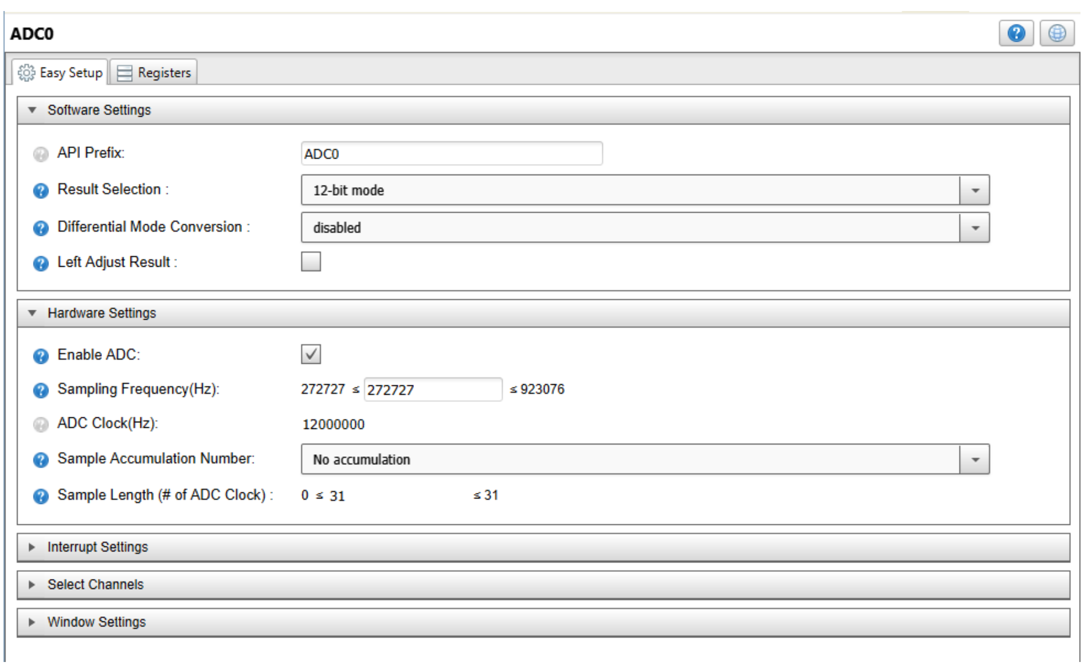
Figure 2-7. VREF Configuration



2.3. Add the ADC module from device resources and configure it in 12-bit mode, right-adjusted results, no accumulation and 31 ticks for sample length.

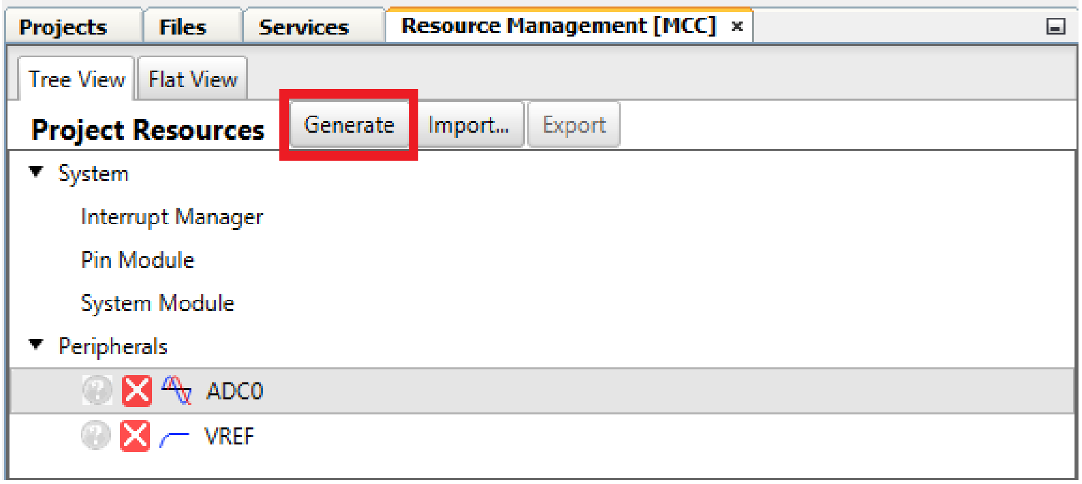
Info: Right-shifted results option must be disabled. The sample length is selected in the **Hardware Settings** tab. The 12-bit mode is set by default. The same is with the no accumulation selection.

Figure 2-8. ADC Configuration



2.4. Press the **Generate** button.

Figure 2-9. Generate Button



3. Add code to the project.

   3.1. Add the following code to the `main.c` file.

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

uint32_t accumulator = 0;

int main(void)
{

    SYSTEM_Initialize();

    while (1){
        accumulator = 0;
        for(uint8_t i = 0; i<128; i++)
        {
            ADC0_StartConversion(ADC_MUXPOS_TEMPSENSE_gc);
            while(!ADC0_IsConversionDone())
            {
                ;
            }
            accumulator += ADC0_GetConversionResult();
        }
    }
}
```

Info: `SYSTEM_Initialize()` is defined in `mcc.c`, `ADC0_StartConversion(channel)`, `ADC0_IsConversionDone()`, and `ADC0_GetConversionResult()` are defined in `adc0.c`.

- `SYSTEM_Initialize()` is a function that configures the microcontroller and peripherals according to what was set in MCC. It is generated by MCC and needs to be called at the beginning of the program.
- `ADC0_StartConversion(channel)` starts an ADC conversion on the given channel
- `ADC0_IsConversionDone()` returns the status of the conversion: 0 for in progress and 1 for finished
- `ADC0_GetConversionResult()` returns the result of the last conversion
- The for loop repeats the steps for starting a conversion, waiting for it to end and adding the result to the accumulator 128 times

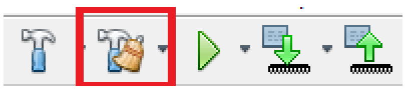
3.2. In the `pin_manager.c` file found in *Source Files → MCC Generated files → src* replace the code for the direction of the pins with:

```
PORTA.DIR = 0xFF;
PORTB.DIR = 0xFF;
PORTC.DIR = 0x3F;
PORTD.DIR = 0xFF;
PORTE.DIR = 0xFF;
PORTF.DIR = 0xFF;
```

Info: This is done to prevent floating pins that will disturb the power reading.

3.3. Press the **Clean and Build** button from the toolbar and verify that the program builds without errors.

Figure 2-10. Clean and Build



View Code Example on GitHub
Click to browse repositories

2.2 Creating the Hardware Implementation Project

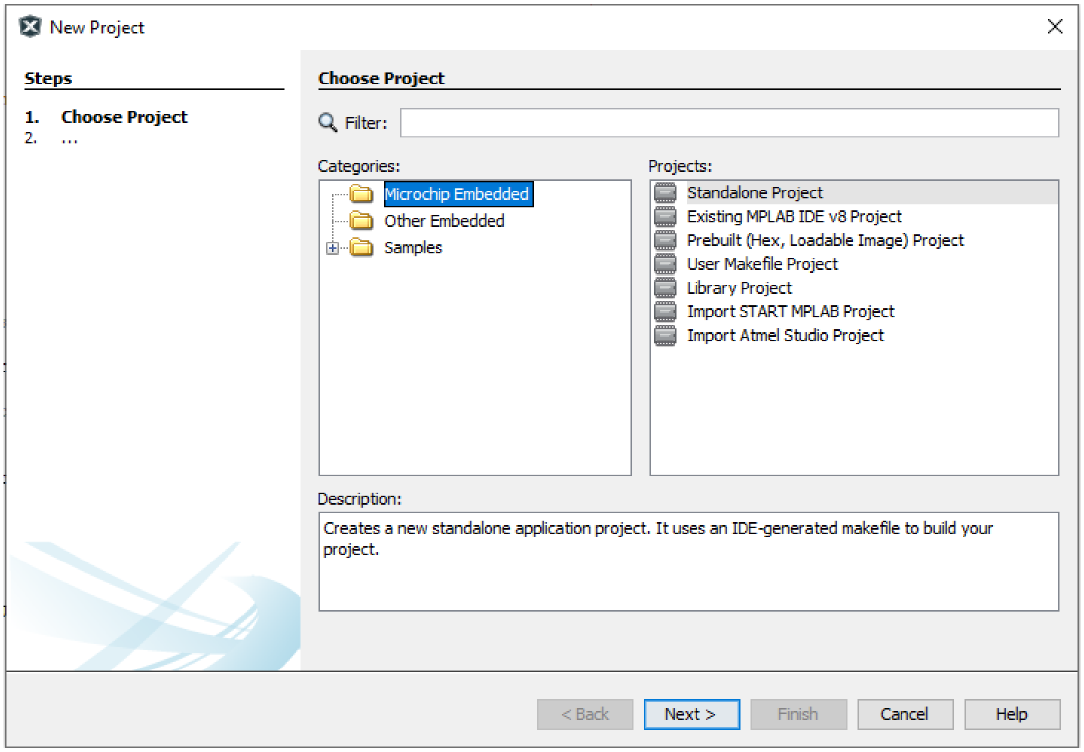
To do: Configure the hardware modules that will be used in this project and add code to the project.

1. Create a new MPLAB X project for AVR128DA48.
   1.1. Open MPLAB X.
   1.2. Select *File → New Project* or the **New Project** button.

   Figure 2-11. New Project

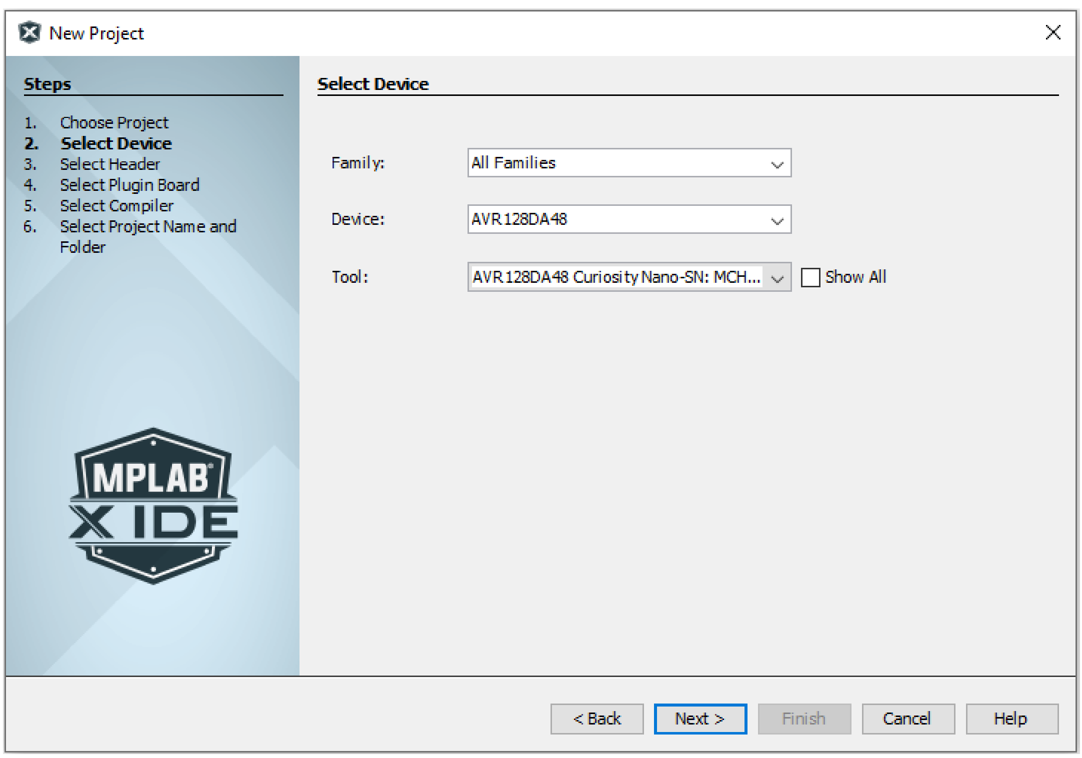
   1.3. Click **Next** (Microchip Embedded Stand-alone Project is selected by default).

Figure 2-12. Project Type



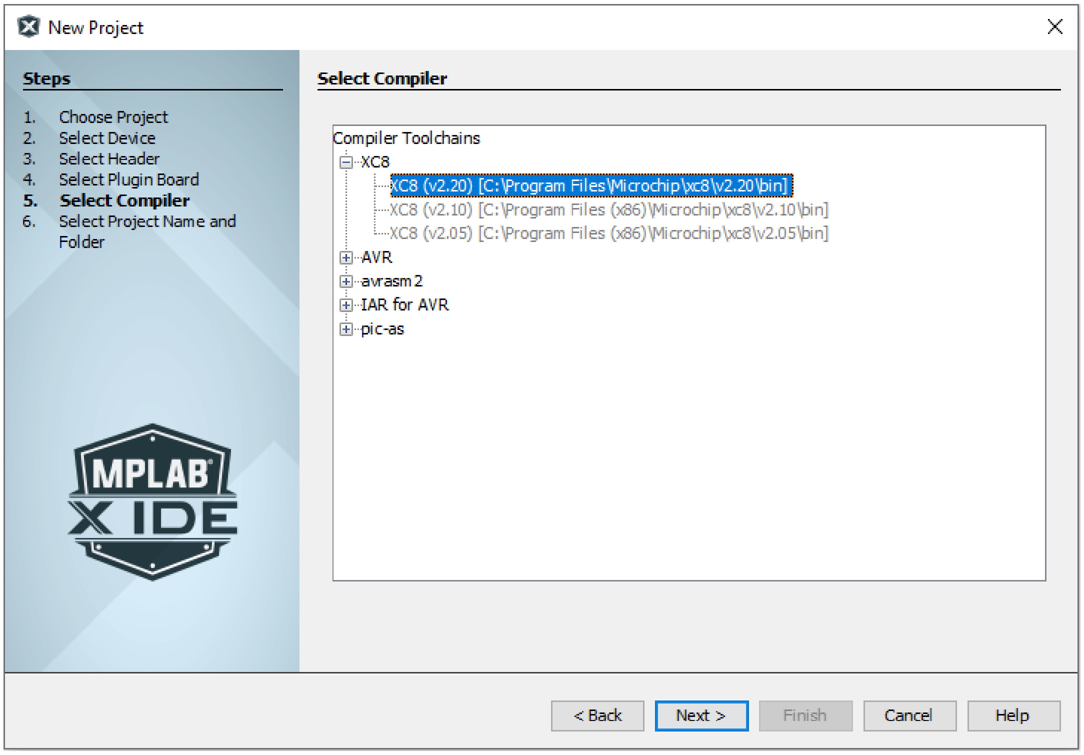
1.4. In the Device field, search for: AVR128DA48. In the Tool category, select the **Curiosity Nano board** if it is connected to the computer, otherwise select **None**. Click **Next**.

Figure 2-13. Device Selection



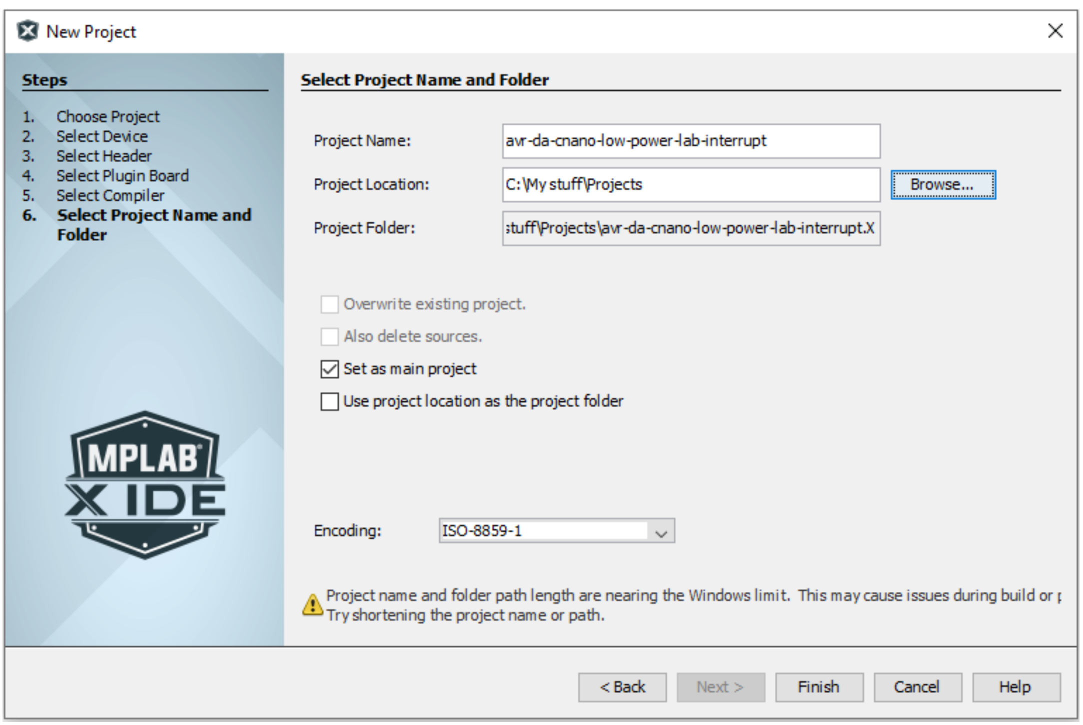
1.5. Select the **XC8 2.20 compiler** and click **Next**.

Figure 2-14. Compiler Selection



1.6. Give a name to the project (and the location where to be saved) and click **Finish**.

Figure 2-15. Project Name

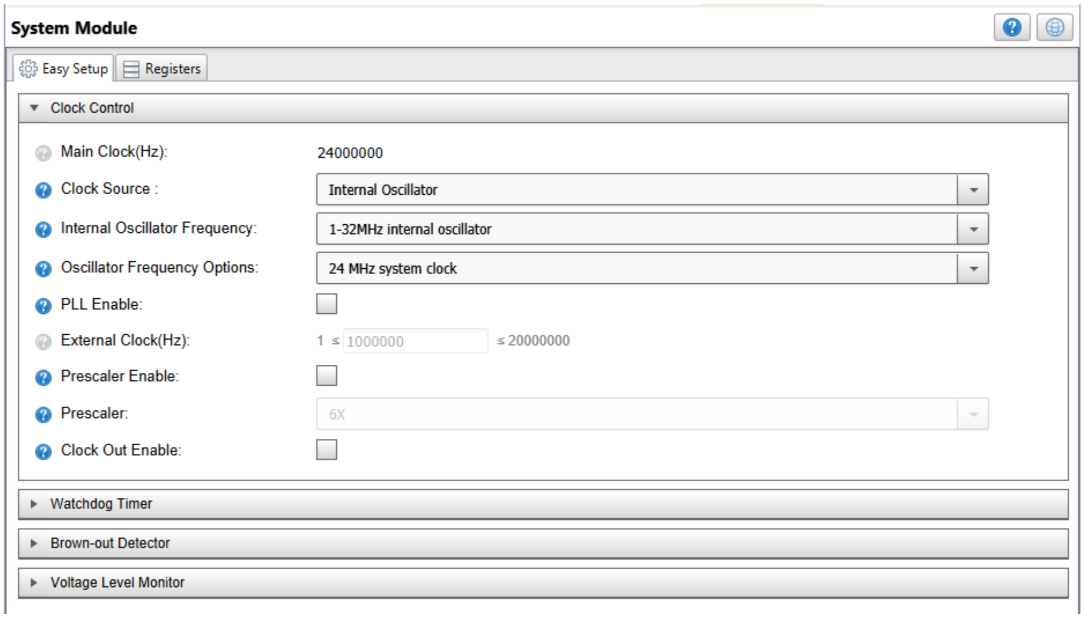


2. Open MPLAB Code Configurator (MCC) and configure the peripherals.

2.1. Configure the System Module to run on the internal oscillator at 24 MHz.

Info: The clock source will be the internal oscillator and the frequency 24 MHz. The prescaler option must be disabled.

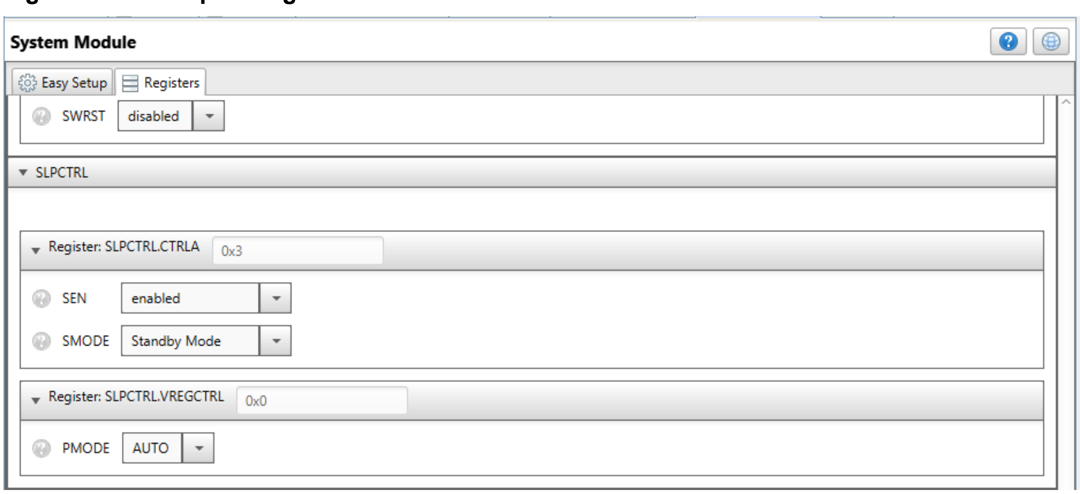
Figure 2-16. System Module Configuration



2.2. Configure the Sleep options so that Sleep is enabled, and the mode is Standby.

Info: Go to the **Registers** page of the System Module and scroll down until reaching SLPCTRL. In the SLPCTRL.CTRLA register, modify the options to be enabled and Standby mode.

Figure 2-17. Sleep Configuration

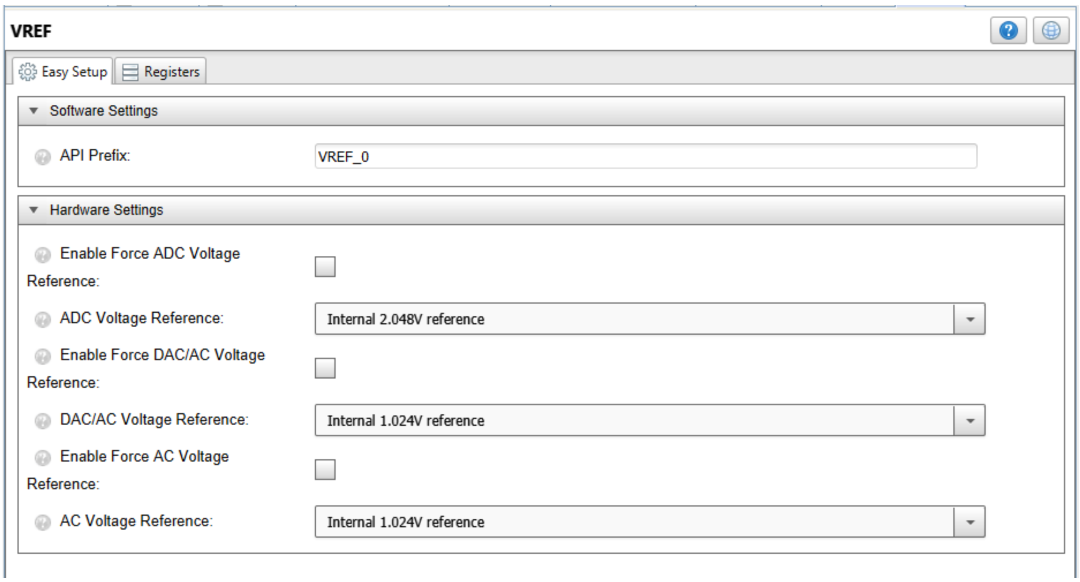


2.3. Add the VREF module from Device resources and configure it to provide 2.048V reference to the ADC.

Info: The ADC voltage reference must be configured to the internal 2.048V reference. Do not enable the Force ADC Voltage option.

Info: The VREF is used to enable the temperature sensor of the microcontroller.

Figure 2-18. VREF Configuration



2.4. Add the ADC module from device resources and configure it in 12-bit mode, right-adjusted results, 128 results accumulation, 31 ticks for sample length, and to Run-In Standby (RUNSTBY).

Info: The right-shifted results option must be disabled. The sample length and sample accumulation number are selected in the **Hardware Settings** tab. The 12-bit mode is set by default. The RUNSTBY bit can be changed from the register view and is found in the CTRLA register.

Figure 2-19. ADC Configuration

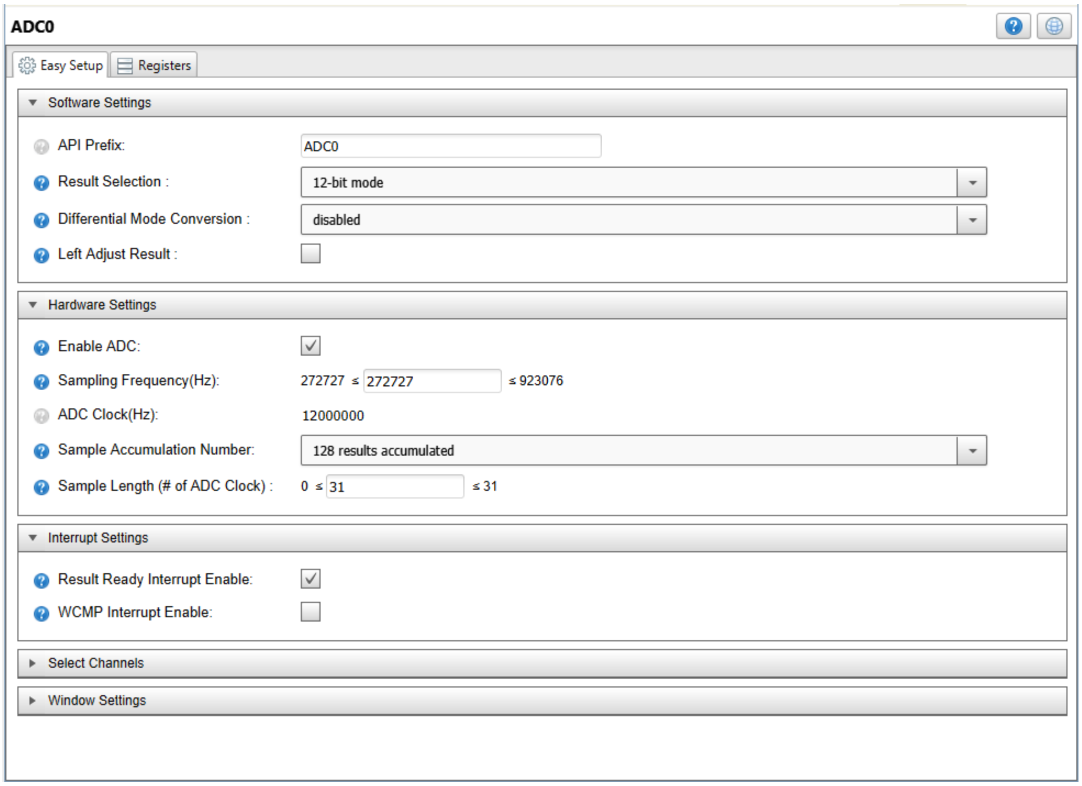
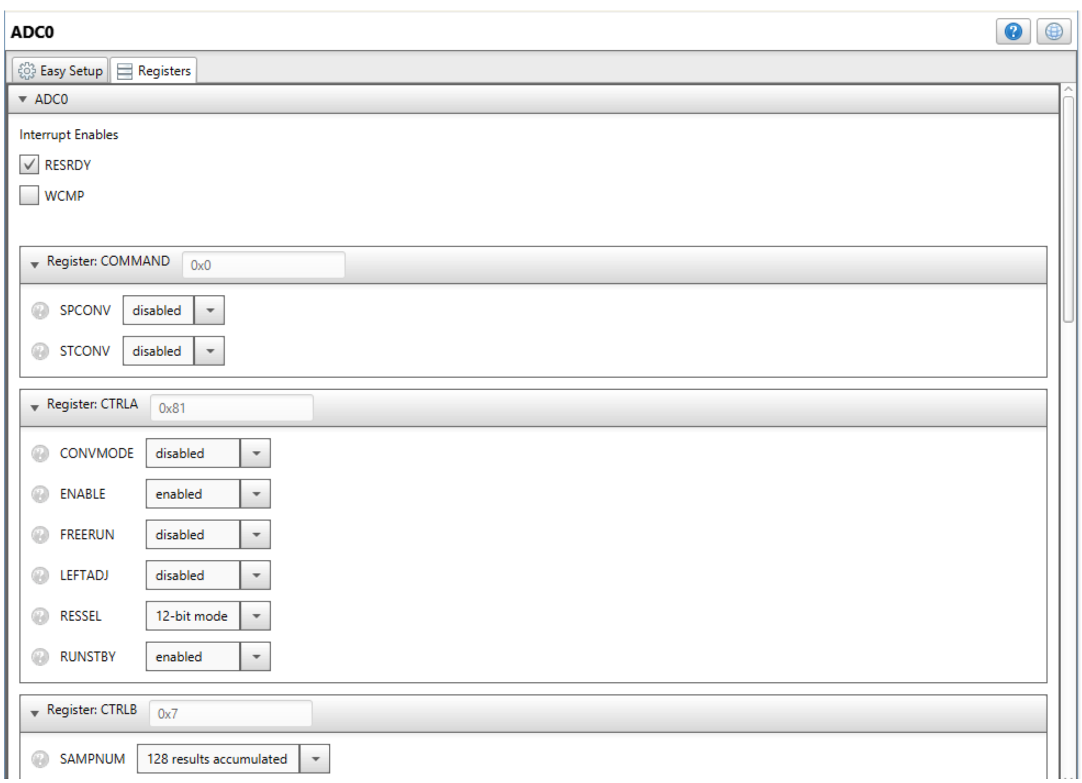


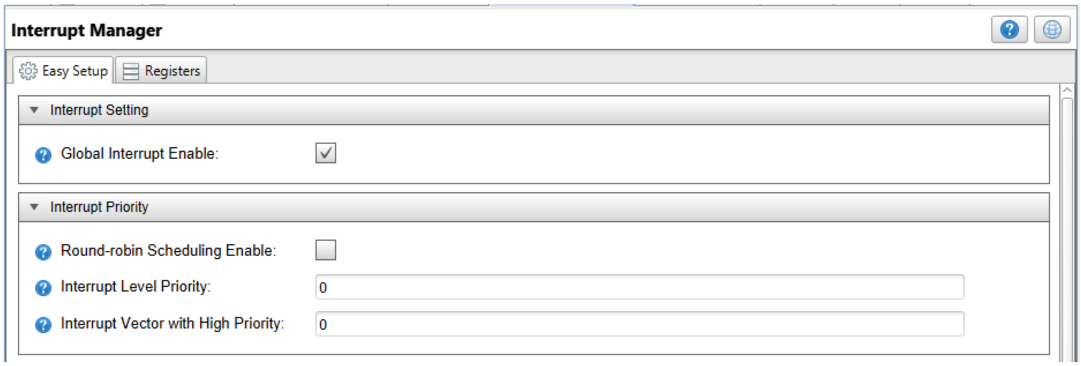
Figure 2-20. Run in Standby ADC Configuration



2.5. Enable the Result Ready Interrupt for the ADC and the Global Interrupts.

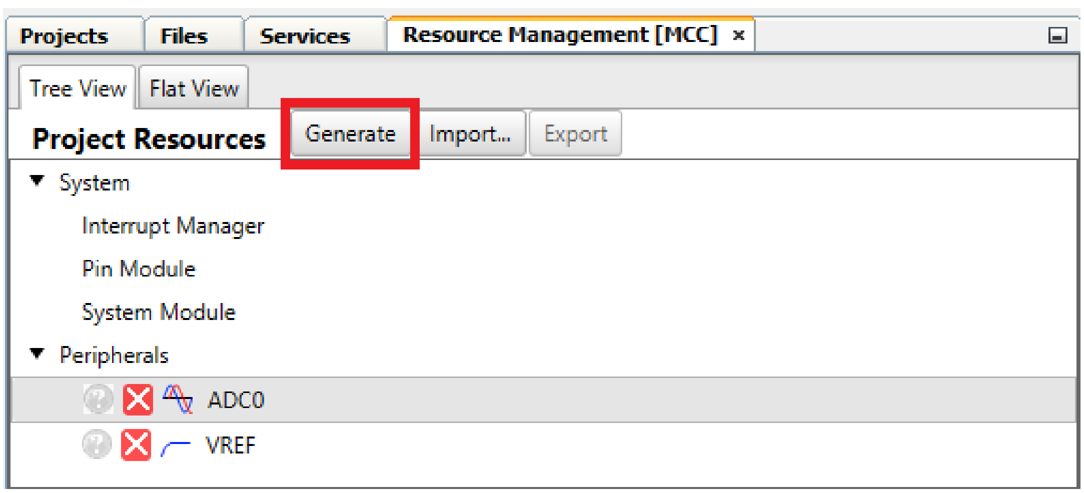
Info: In the ADC0 module, in the Interrupt Setting, activate Result Ready Interrupt Enable. This setting can be observed in the figure for the previous step. In the Interrupt Manager, enable Global Interrupt.

Figure 2-21. Interrupt Manager Configuration



2.6. Press the **Generate** button.

Figure 2-22. Generate Button



3. Add code to the project.

3.1. Add the following code to the `main.c` file:

```
#include "mcc_generated_files/mcc.h"
#include <avr/sleep.h>

uint16_t result;

int main(void)
{

    SYSTEM_Initialize();
    while (1){
        ADC0_StartConversion(ADC_MUXPOS_TEMPSENSE_gc);
        sleep_cpu();
        result = ADC0_GetConversionResult();
    }
}
```

Info: `SYSTEM_Initialize()` is defined in `mcc.c`, `ADC0_StartConversion(channel)` and `ADC0_GetConversionResult()` are defined in `adc0.c`.

- `SYSTEM_Initialize()` is a function that configures the microcontroller and peripherals according to the settings in MCC. It is generated by MCC and needs to be called at the beginning of the program.
- `ADC0_StartConversion(channel)` starts an ADC conversion on the given channel.
- `ADC0_GetConversionResult()` returns the result of the last conversion.
- The program initiates a conversion and then enters Sleep. It is woken up by the Result Ready Interrupt, reads the result and then starts a new one.

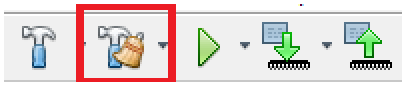
3.2. In the `pin_manager.c` file found in *Source Files → MCC Generated files → src*, replace the code for the direction of the pins with:

```
PORTA.DIR = 0xFF;
PORTB.DIR = 0xFF;
PORTC.DIR = 0x3F;
PORTD.DIR = 0xFF;
PORTE.DIR = 0xFF;
PORTF.DIR = 0xFF;
```

Info: This is done to prevent floating pins that will disturb the power readings.

3.3. Press the **Clean and Build** button from the toolbar and verify that the program builds without errors.

Figure 2-23. Clean and Build



2.3 Programming the Device and Comparing Power Efficiency

To do: Load the project onto the microcontroller and get power readings.

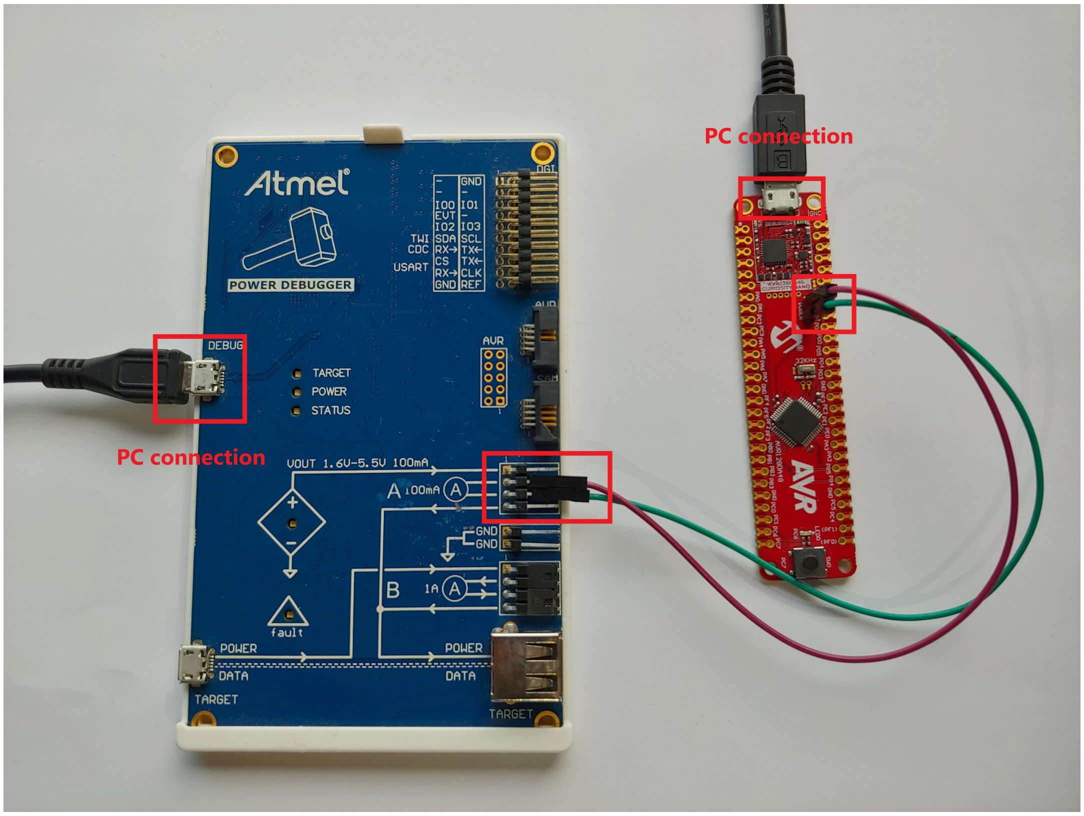
1. Program the device and get power consumption readings from the software implementation program.
   1.1. Connect the Curiosity Nano board to the computer. Connect the two soldered pins together. With the software accumulation program set as main program, press the **Make and Program Device** button from the toolbar. A pop-up might appear that asks to select the tool that will be programmed. Select the AVR128DA48 Curiosity Nano board.

   Figure 2-24. Make and Program the Device

   1.2. Open Atmel Studio and connect the Power Debugger to the computer. Connect the Curiosity Nano board to the channel A ammeter of the Power Debugger. Connect the pin closest to the Nano embedded debugger to the inward pointing arrow of the A ammeter and the pin closest to the microcontroller to the outward pointing arrow of the ammeter.

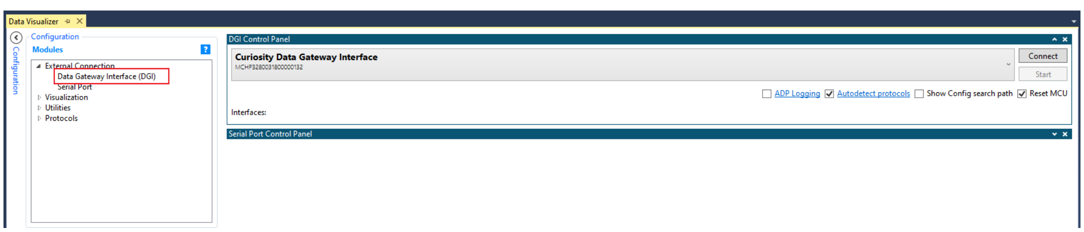
Figure 2-25. Hardware Setup



1.3. In Atmel Studio, go to *Tools → Data Visualizer*.

1.4. Two boxes will appear. Check the one called DGI (Data Gateway Interface) Control Panel. If this does not show up, the Configuration menu on the left side can be accessed and, in the *Modules → External Connection* section, the Data Gateway Interface can be found.

Figure 2-26. Data Gateway Interface



1.5. Press **Connect** and a box labeled 'power' will appear. Select it and then press the wheel to open the menu. Disable the B channel.

⚠WARNING In case the Power field does not appear, disconnect the board and connect it again.

Figure 2-27. DGI Connect

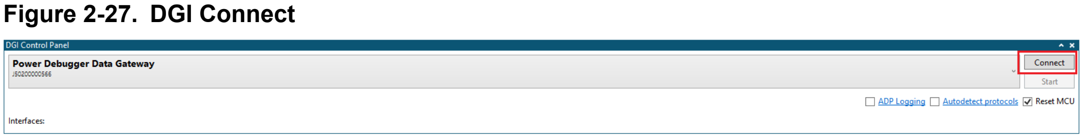
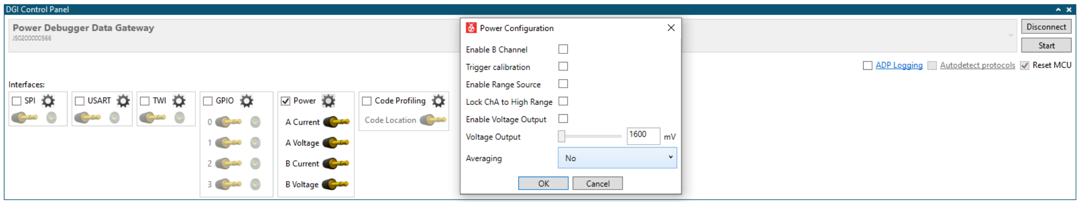


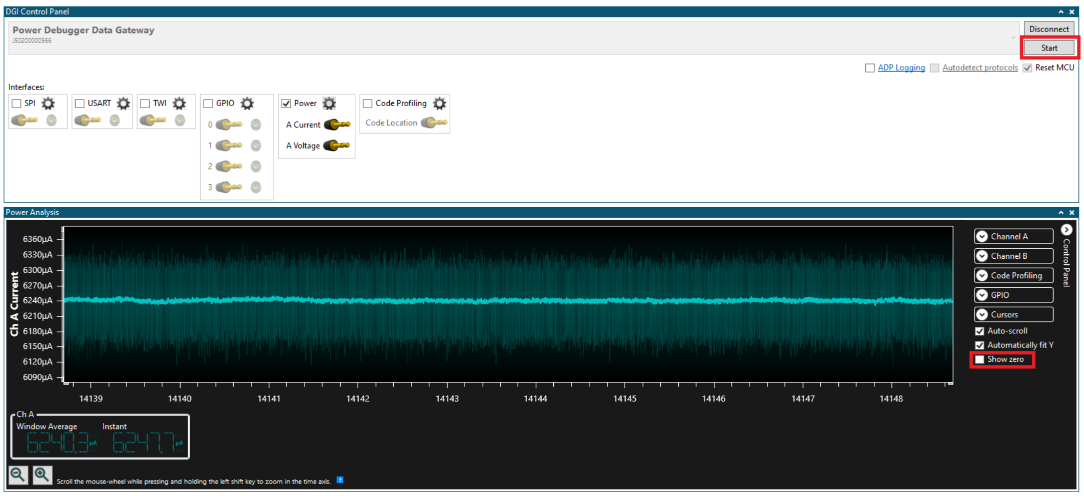
Figure 2-28. Power Settings



1.6. When everything is correctly configured, press **Start** and a new box that displays the power consumption in real time will appear.

Tip: To adjust the display, open the control panel on the left side of the box. Unchecking Show zero will provide a clearer picture.

Figure 2-29. Start Monitoring



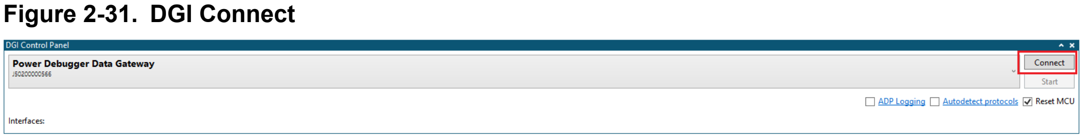
1.7. After the reading has stabilized, press **Stop** and then disconnect. The box with the power measurement will remain open.

2. Get power consumption readings from the hardware accumulation implementation program.
   2.1. Connect the two soldered pins on the Curiosity board together.
   2.2. With the hardware accumulation project set as main program, press the **Make and Program Device** button on the toolbar in MPLAB X.

Figure 2-30. Make and Program the Device

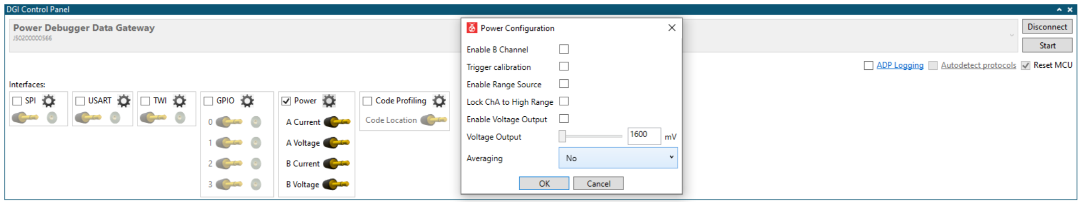
   2.3. Connect two soldered pins of the Curiosity Nano Board to the Power Debugger. See Step 1.2.
   2.4. Go back to the Data Visualizer in Atmel Studio and in the DGI panel press **Connect**.

Figure 2-31. DGI Connect



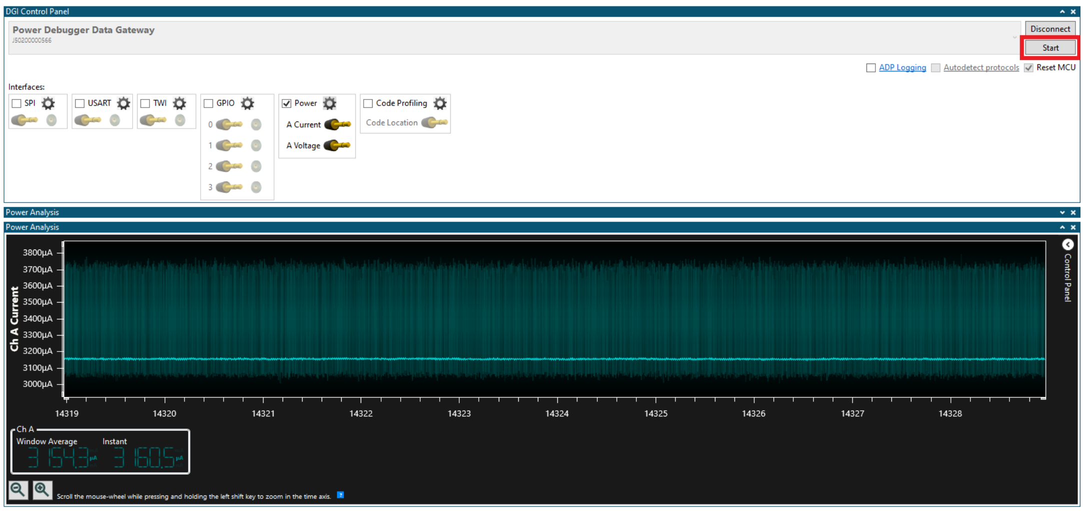
   2.5. Check the power box and edit the settings to disable the B channel.

Figure 2-32. Power Settings



2.6. Press **Start** and a new box that displays the real-time power consumption will appear. Disable Show zero for a better picture.

Figure 2-33. Start Monitoring

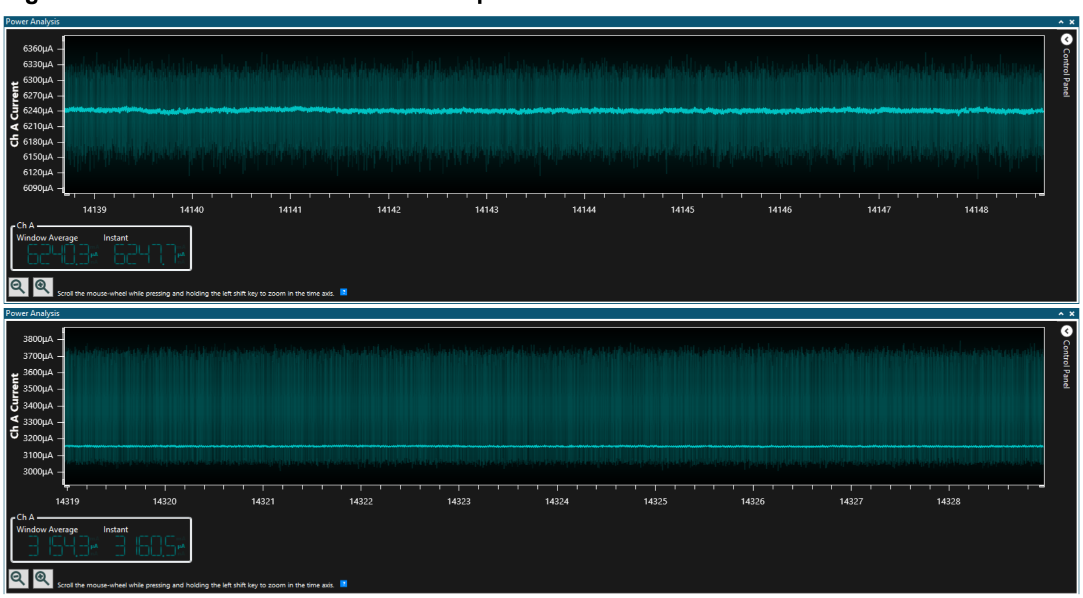


2.7. When the reading has stabilized, press **Stop** to have a fixed result.

3. Compare the two measurements.

3.1. Both power consumption graphs are open on the same screen. They can now be analyzed to observe the way the programs behaved.

Figure 2-34. Software Accumulation Compared to Hardware Accumulation



Info: The top part displays the software accumulation, while the bottom part, the hardware accumulation. As can be seen from the picture, there is a 3.1 mA difference between the two, making the hardware implementation almost twice as efficient.

3. Revision History

Revision	Date	Description
A	8/2020	Initial document release

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

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, TempTrackr, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

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, Vite, 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, BlueSky, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, INICnet, Inter-Chip Connectivity, JitterBlocker, KleerNet, KleerNet logo, 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, SAM-ICE, Serial Quad I/O, SMART-I.S., SQI, SuperSwitcher, SuperSwitcher II, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, 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: 978-1-5224-6615-4

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