注意: この日本語版文書は参考資料としてご利用ください。
最新情報は必ずオリジナルの英語版をご参照願います。

AVR® DAトレーニング マニュアル

Curiosity Nanoによる低消費電力モード

前提条件

- **ハードウェア要件**
  - AVR128DA48 Curiosity Nanoボード(DM164151)に本書に記載する多少の改変を加えたもの
  - Power Debugger (ATPOWERDEBUGGER)
  - 2本のMicro-USBケーブル: 1本はPower Debugger用、もう1本はCuriosity Nanoボード用
  - 2本のピンヘッダおよびショート用ジャンパ
- **ソフトウェア要件**
  - MPLAB® X 5.40
  - MPLAB Xバージョン3.95用MCCプラグイン
  - MCCバージョン2.3.0用AVRライブラリ
  - Atmel Studio最新バージョン

はじめに

本書では、ソフトウェアのみを使ったアプローチと、CIP(コアから独立した周辺モジュール)を使って各種タスクを達成するアプローチの消費電力の違いについて述べます。

この実践トレーニングには以下のトピックが含まれます。

- 課題1: イベントシステムと割り込みの比較
- 課題2: ADC (A/Dコンバータ)におけるソフトウェア積算とハードウェア積算の比較

各課題の冒頭に各課題に関連する基本的な理論を紹介した後、Atmel StudioのData Visualizerを使って機能を検証します。

本書で使うアイコン

本書では、課題の説明を読みやすくするために、以下のアイコンを使います。

	**Info:** 関連する情報を示します。
	**Tip:** 役に立つヒントやテクニックを示します。
	**To do:** 実行する目的を示します。
	**結果:** 課題の手順を実行した結果を示します。
	重要な情報を示します。
	**Execute:** 必要に応じてターゲットの外で実行するアクションを示します。

目次

1. 課題1: イベントシステムと割り込みの比較

本課題では、2種類のアプローチの消費電力を比較します。1つ目のアプローチでは、ADC変換を開始して温度センサを読み出すタイマ オーバーフロー割り込みによってマイクロコントローラがスリープから復帰します。変換が完了すると、結果レディ割り込みがトリガされ、変換結果が読み出されます。

2つ目のアプローチでは、イベントシステム(EVSYS)モジュールを使います。これはイベント経由で各種周辺モジュールを接続できるモジュールです。前者のアプローチをタイマ オーバーフローが温度センサを読み出すADC変換を開始するイベントをトリガするように変更します。変換が終了するとマイクロコントローラがスリープから復帰して結果を読み出します。

1.1 ハードウェアの説明

CuriosityボードはAVR128DA48マイクロコントローラ、Nano組み込みデバッガ、1つのLED、1つのボタンを備えています。Nanoデバッガを使う事で外部ハードウェア不要でボードをプログラムしデバッグ機能を実行できます。また、UART-USBブリッジ機能も提供しています。

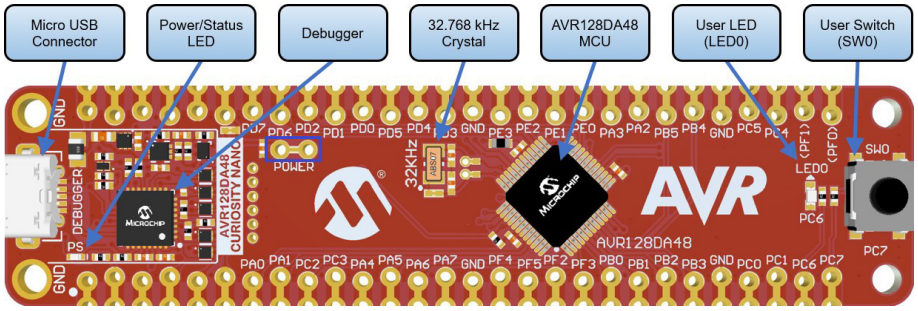
Power Debuggerは電圧を設定可能な電源出力も備えたプログラマで、接続されたデバイスの消費電流を計測するために使えます。

1.2 電力計測を正確に実行するためのCuriosityボードの改変

To do: Curiosity Nano ボードを改変し、トレースを切断して 2 本のピンをはんだ付けします。

micro-USBポートから電力を観測すると、Power Debuggerが多少の電力を消費するため、計測値が不正確になります。

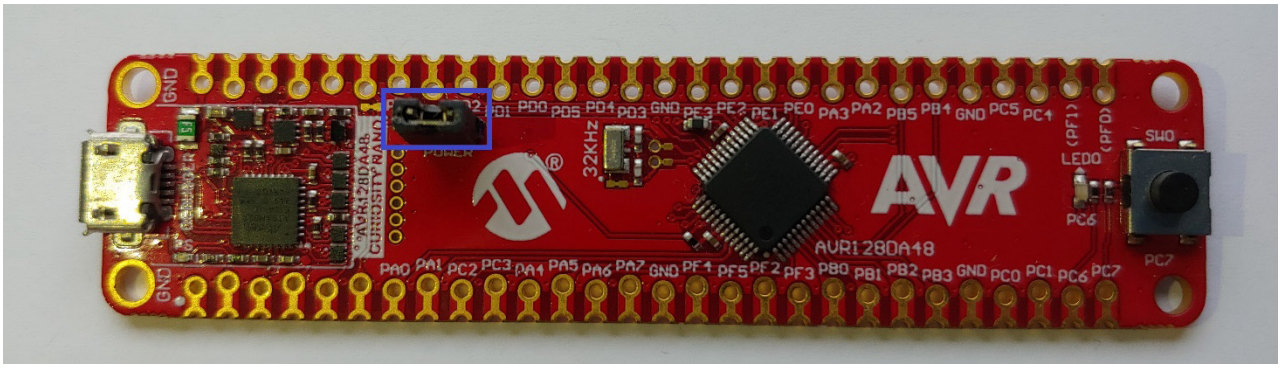
図1-1.POWERトレースの位置



CuriosityボードにはPOWERというラベルの付いた小さなトレースがあります。カッター等で２つのホール間トレースを確実に切断します。絶縁できたことをテスタで確認します。

トレースを切断した後、ホールに2本のピンヘッダをはんだ付けします。以後、ボードをプログラミングする際はこのピンヘッダ間をショートジャンパで接続する必要があります。

図1-2.改変後のCuriosityボード



1.3 使用モジュールの説明

この課題で使うモジュールはリアルタイム クロック(RTC)、ADC、参照電圧(VREF)、イベントシステム(EVSYS)です。

RTCは1秒単位でカウントするタイマとして動作します。

ADCはアナログ電圧を12ビットのデジタル値に変換するモジュールです。EVSYSから変換を開始させる事ができます。また、変換を終了するとイベントと割り込みを発生させます。ADCは内部温度センサの読み出しに使います。

参照電圧は他の周辺モジュールに安定化された電圧を供給する周辺モジュールです。また、2.048 Vの参照電圧をADCに供給するよう設定されている場合、温度センサを有効にします。

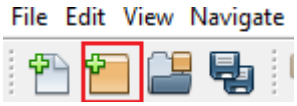
イベントシステムはCPUを介さずに周辺モジュール間の通信を可能にする伝達回路です。通信はある周辺モジュールから別の周辺モジュールにイベントを送信してアクションをトリガする事で行われます。イベントはレイテンシフリーで見逃される事なく、信頼性の高いリアルタイム アプリケーションを実現します。

1.4 ソフトウェア実装プログラムの作成

To do: このプロジェクトで使うハードウェア モジュールを設定し、コードをプロジェクトに追加します。

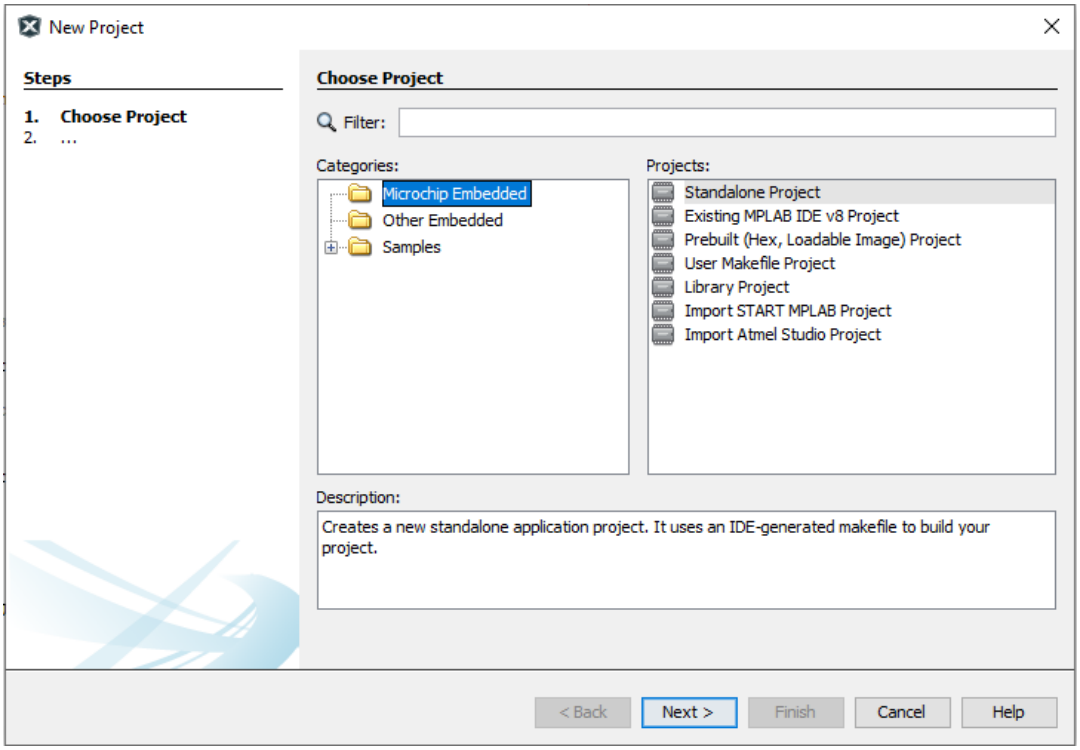
1. AVR128DA48用のMPLAB Xプロジェクトを新規作成します。
   1.1 MPLAB Xを開きます。
   1.2 [File] > [New Project]を選択するか、または**[New Project]**ボタンをクリックします。

図1-3.プロジェクトの新規作成



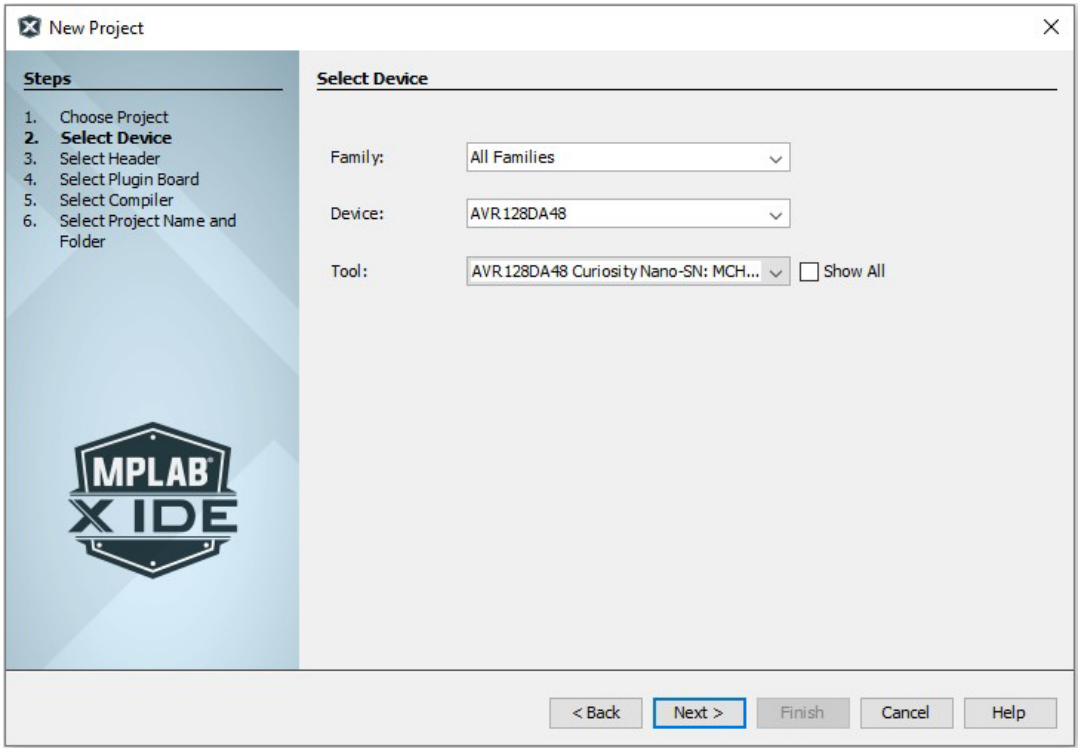
   1.3. **[Next]**をクリックします(既定値では[Microchip Embedded] > [Standalone Project]が選択されています)。

図1-4.プロジェクトのタイプ



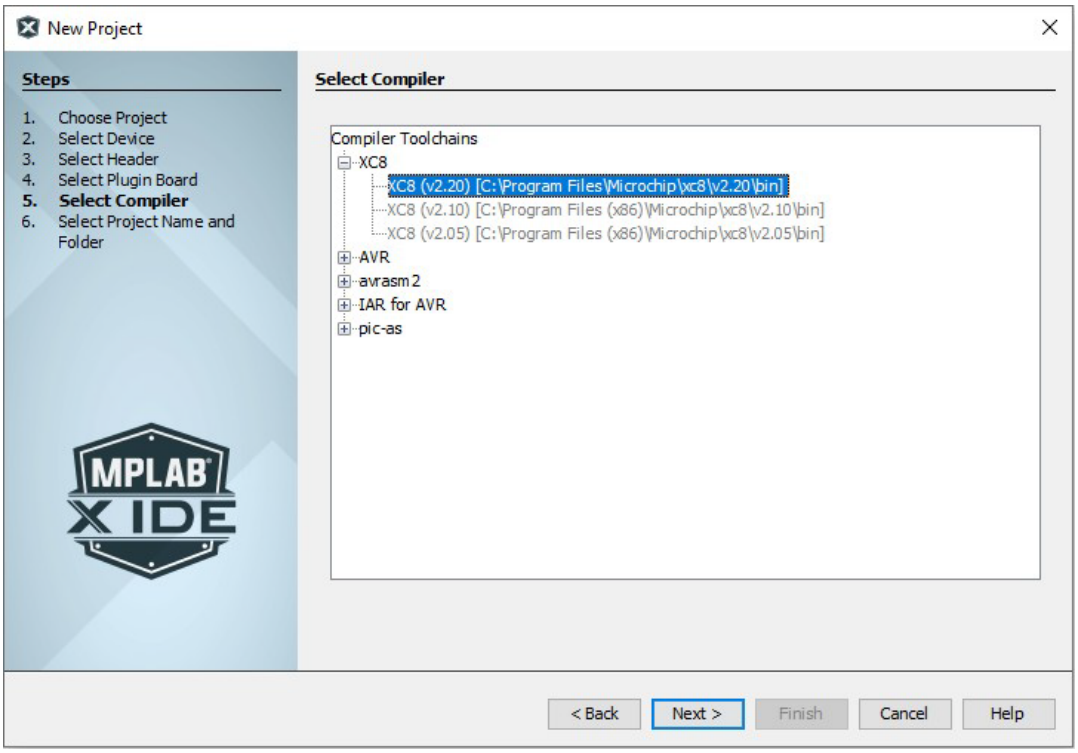
1.4 [Device]フィールドで「AVR128DA48」を検索します。[Tool]カテゴリで、Curiosity Nanoボードがコンピュータに接続されている場合、そのボードを選択します。接続されていない場合は**[None]**を選択します。**[Next]**をクリックします。

図1-5.デバイスの選択



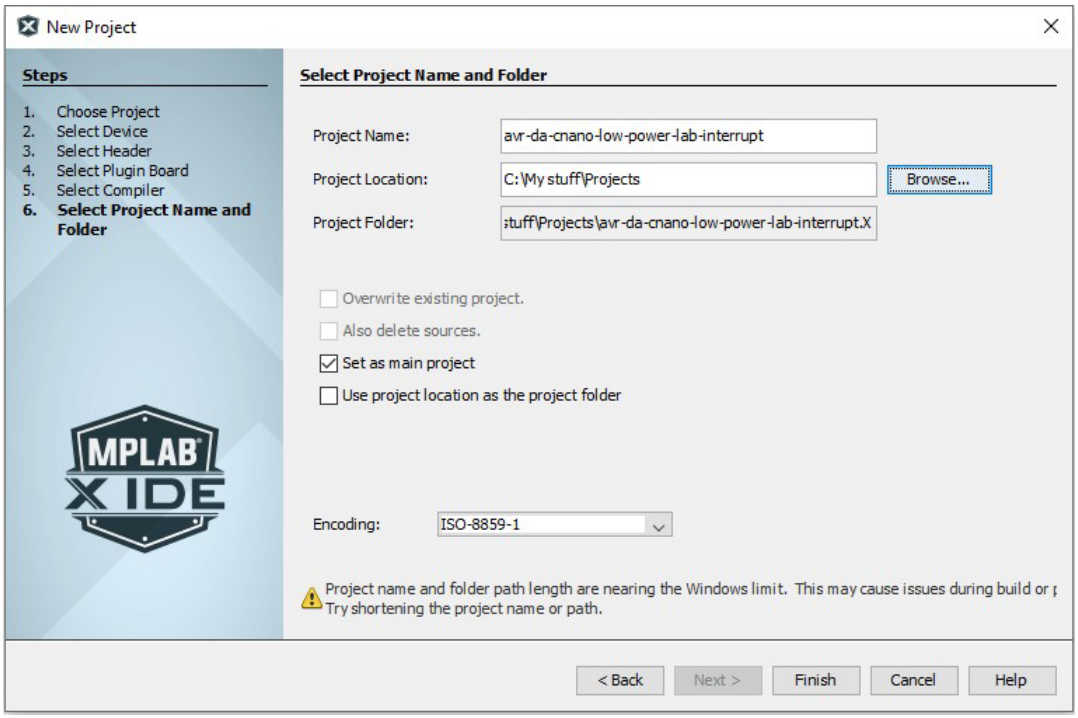
1.5. **[XC8 (2.20)]**コンパイラを選択し、**[Next]**をクリックします。

図1-6.コンパイラの選択



1.6. プロジェクトに名前を付け(保存場所を選択し)、**[Finish]**をクリックします。

図1-7.プロジェクト名

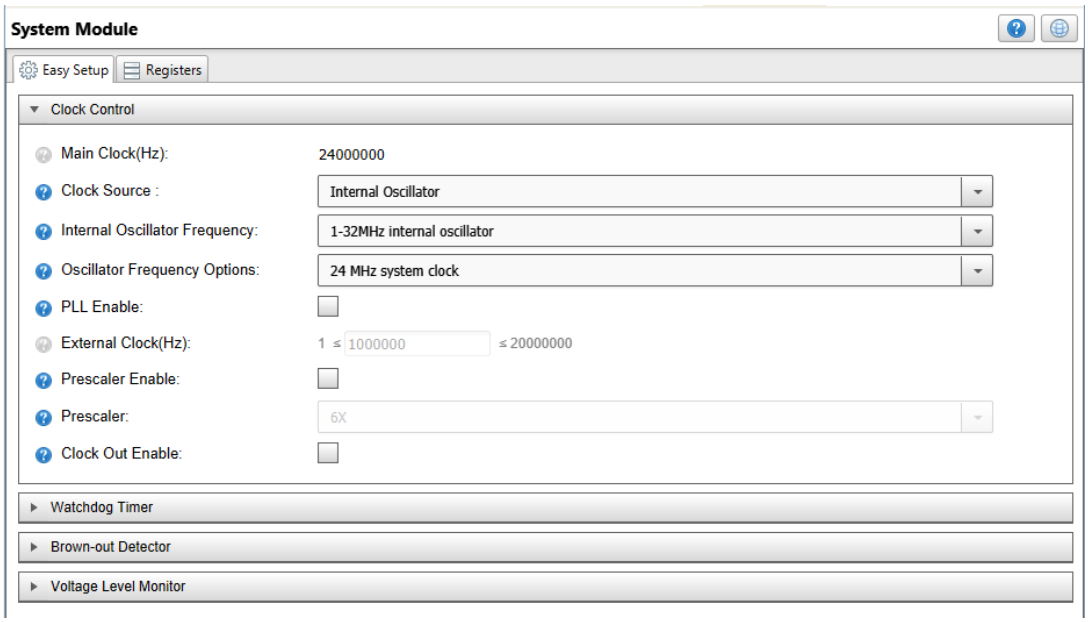


2. MCC (MPLAB Code Configurator)を開き、周辺モジュールを設定します。

2.1 [System Module]で、クロック源として24 MHzクロックの内部オシレータを選択します。

Info: [Clock Source]で[Internal Oscillator]、[Oscillator Frequency Options]で[24 MHz system clock]を選択し、[Prescaler]は無効にします。

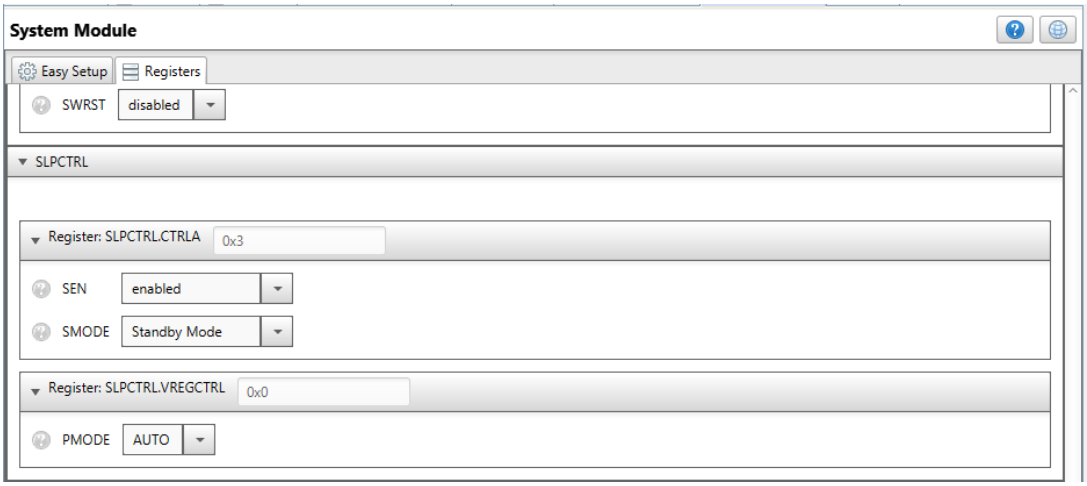
図1-8.システム モジュールの設定



2.2 [System Module]の**[Registers]**タブで[SEN]を[enabled]にし、[SMODE]を[Standby Mode]にします。

Info: [System Module]の**[Registers]**タブを開き、[SLPCTRL]までスクロールします。CTRLA レジスタを変更してスリープを有効にし、モードを選択します。

図1-9.スリープの設定

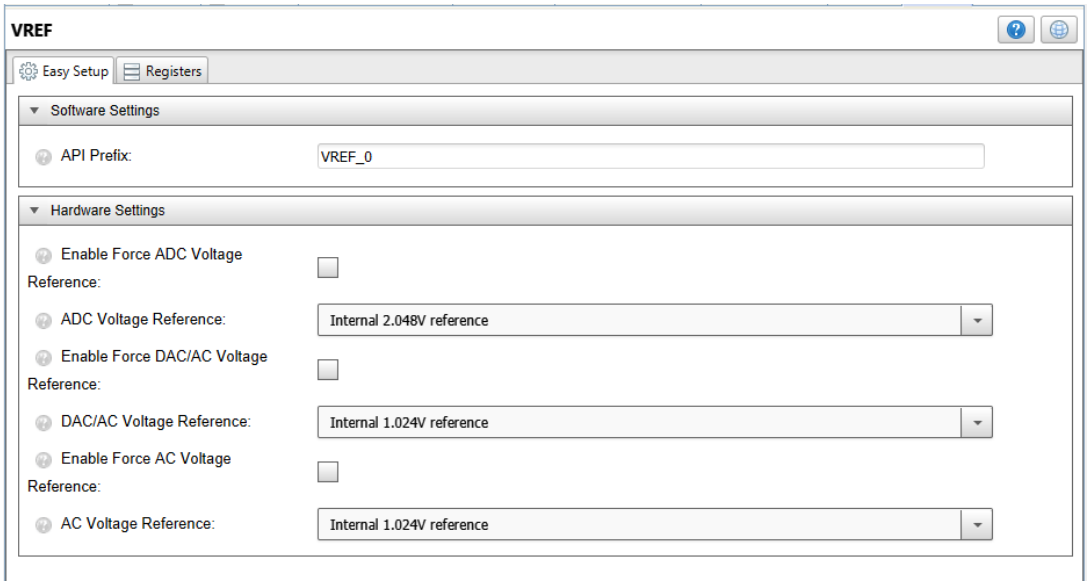


2.3 [Device Resources]からVREFモジュールを追加し、2.048 Vの参照電圧をADCに供給するよう設定します。

Info: ADC 参照電圧は 2.048 V の内部参照電圧に設定する必要があります。
[Enable Force ADC Voltage]にはチェックを入れないでください。

Info: VREF はマイクロコントローラの温度センサを有効にするために使われます。

図1-10.VREFの設定

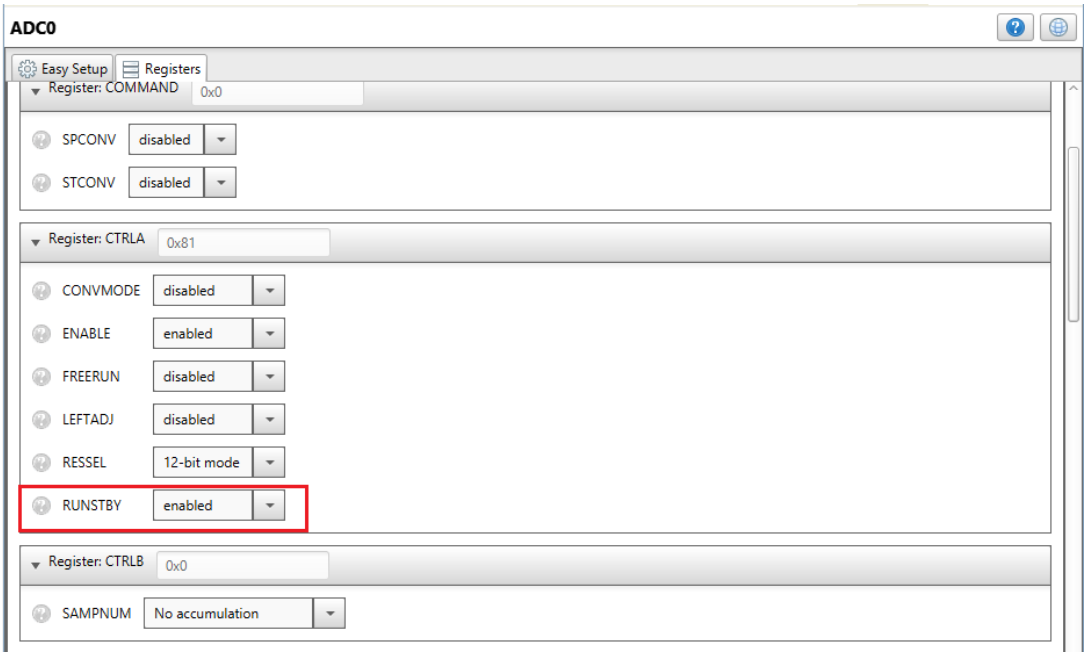


2.4 [Device Resources]からADCモジュールを追加し、12ビットモード、結果は右寄せ、積算なし、サンプル長は1ティック、スタンバイ中動作(RUNSTBY)に設定します。

Info: 左揃え(Left Adjust Result)のオプションは無効にする必要があります。サンプル長とサンプル積算数は**[Hardware Settings]**タブで選択します。12 ビットモードは既定値で設定されています。RUNSTBY ビットは**[Registers]**タブで変更できます。これは CTRLA レジスタにあります。

図1-11.ADCの設定

図1-12.RUNSTBYに関するADCの設定



2.5. [Device Resources]からRTCモジュールを追加し、32.768 kHzの内部オシレータを使い、周期を1秒、プリスケール係数を1、スタンバイ中動作(RUNSTBY)を有効に設定します。

Info: クロック、プリスケール係数、周期は**[Hardware Settings]**タブで設定し、RUNSTBY ビットは CTRLA レジスタのレジスタペインにあります。

図1-13.RTCの設定

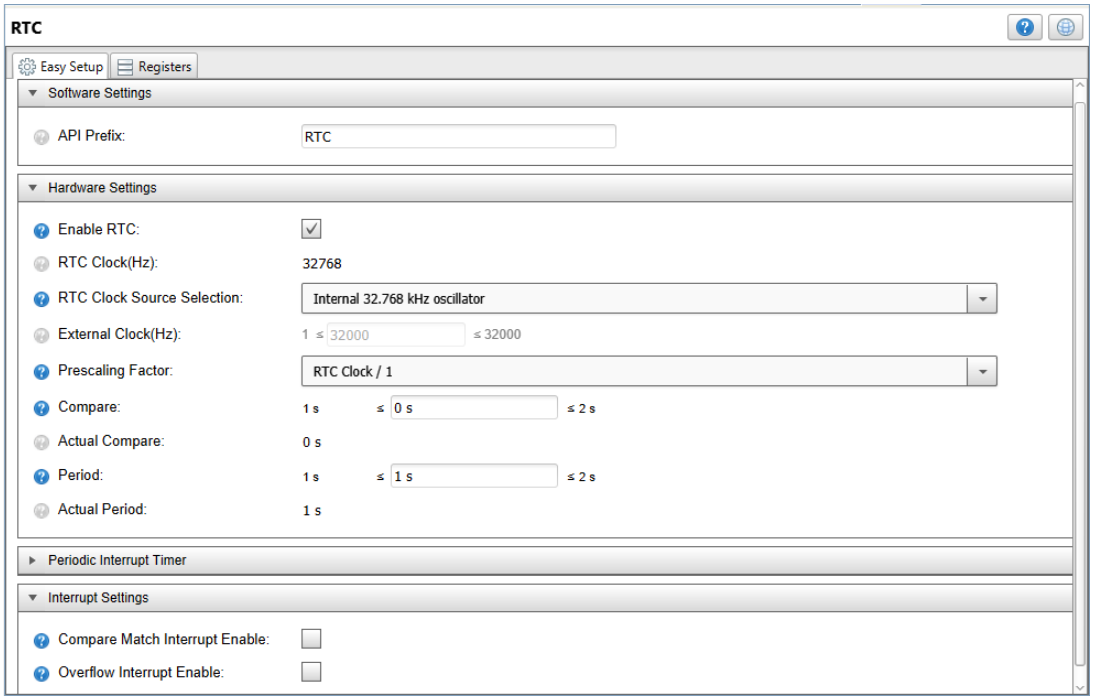
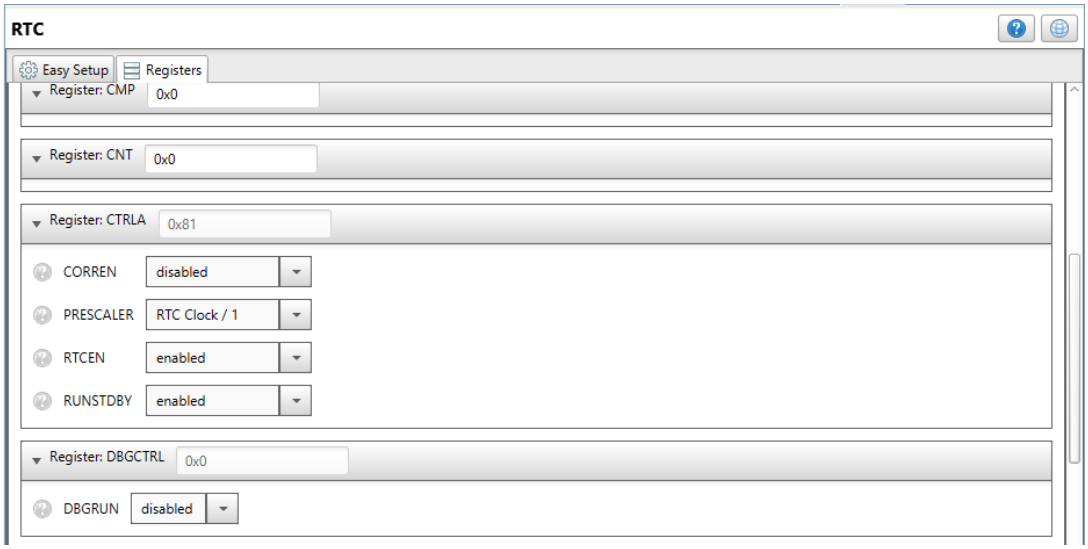


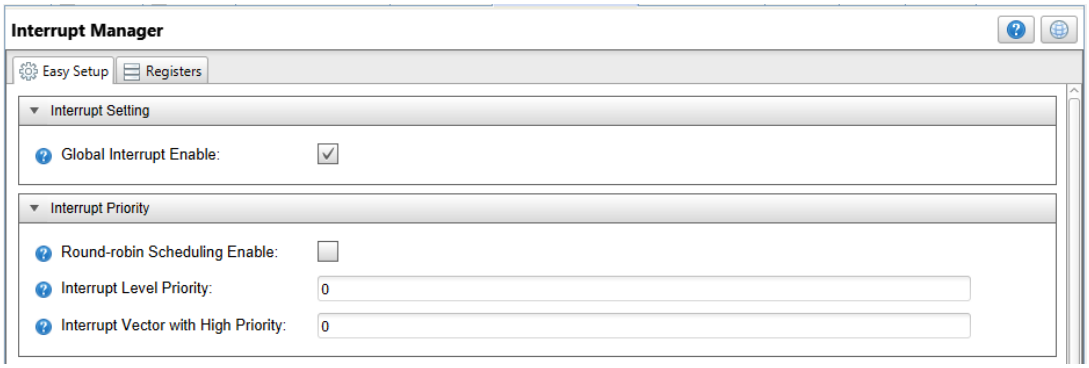
図1-14.RUNSTBYに関するRTCの設定



2.6. グローバル割り込みを有効化し、ADCの結果レディ割り込みとRTCのオーバーフロー割り込みを有効化します。

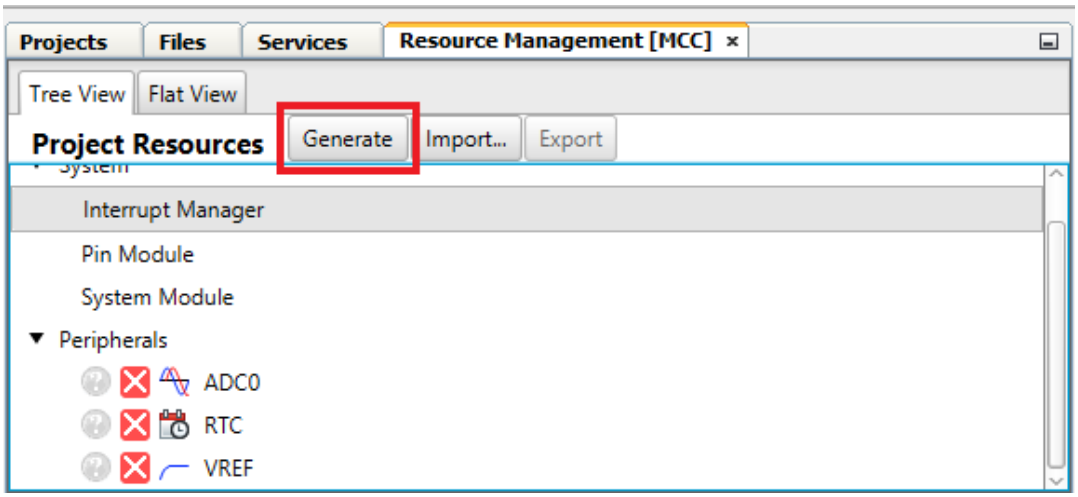
Info: グローバル割り込みの有効化は[Interrupt Manager]で行います。ADC の割り込みは ADC モジュールの**[Interrupt Settings]**タブで、RTC の割り込みは RTC モジュールの**[Interrupt Settings]**タブで確認できます(どちらも上述のステップの画像を参照)。

図1-15.割り込みマネージャの設定



2.7. **[Generate]**ボタンを押します。

図1-16.[Generate]ボタン



3. 生成されたファイルにコードを追加します。

3.1. `main.c`ファイルに以下のコードを追加します。

```
#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()`は`mcc.c`で定義されます。`RTC_SetOVFIsrCallback()`は`rtc.c`で定義されます。`ADC0_StartConversion()`と`ADC0_GetConversionResult()`は`adc.c`で定義されます。

- `SYSTEM_Initialize()`は CPU と周辺モジュール用の全てのコンフィグレーションレジスタを設定します。この関数は MCC によって生成されます。
- `RTC_SetOVFIsrCallback()`は main 内で使い、`TIMER_interrupt()`関数にコールバック(割り込み発生時に呼び出される関数)を設定する関数です。
- `TIMER_interrupt()`関数は ADC の変換を開始します。
- `ADC0_StartConversion()`関数は ADC の温度チャンネルの変換を開始します。
- `ADC0_GetConversionResult()`は直前の変換結果を返します。

3.2. [Source Files] > [MCC Generated Files] > [src]にある`pin_manager.c`ファイルで、ピンの方向に関するコードを以下に置き換えます。

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

Info: これは電力読み値に干渉するフローティングのピンを防ぐために行います。

3.3. ツールバーから**[Clean and Build]**ボタンを押して、プログラムがエラーなくビルドされる事を確認します。

図1-17.クリーンとビルド

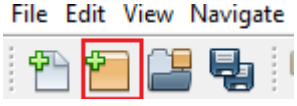
GitHubでサンプルコードを見る
クリックしてリポジトリを閲覧

1.5 イベントシステム実装プロジェクトの作成

To do: このプロジェクトで使うハードウェア モジュールを設定し、コードをプロジェクトに追加します。

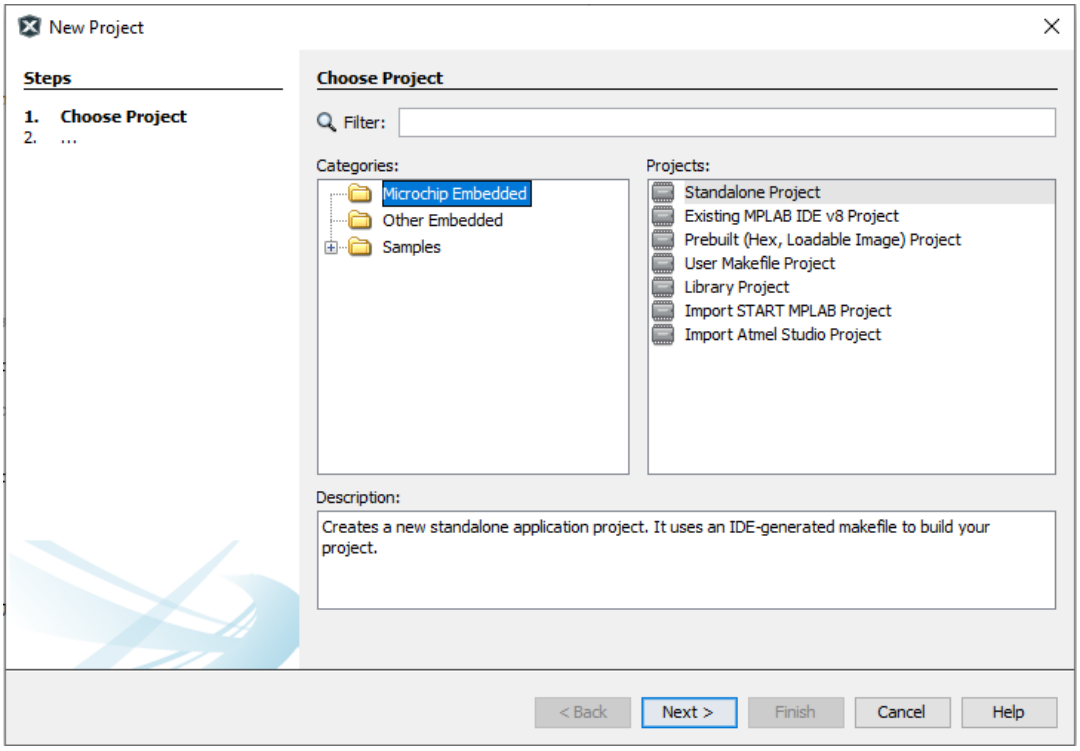
1. AVR128DA48用のMPLAB Xプロジェクトを新規作成します。
   1.1. MPLAB Xを開きます。
   1.2. [File] > [New Project]を選択するか、または**[New Project]**ボタンをクリックします。

図1-18.プロジェクトの新規作成



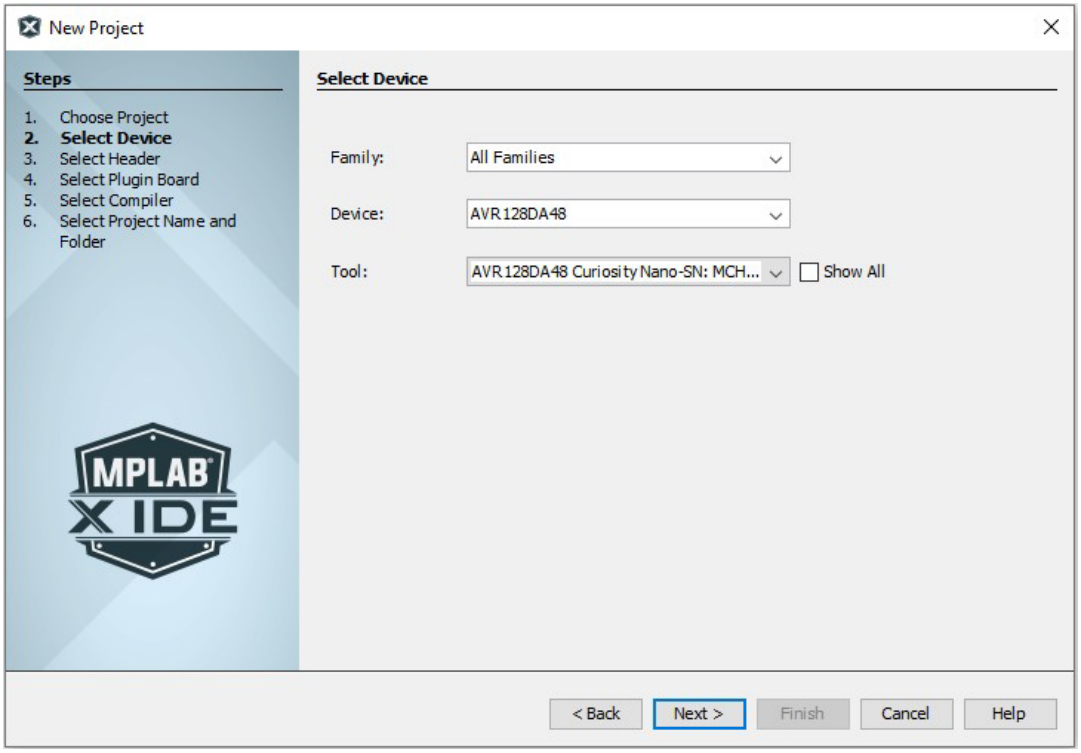
1.3. **[Next]**をクリックします(既定値では[Microchip Embedded] > [Standalone Project]が選択されています)。

図1-19.プロジェクトのタイプ



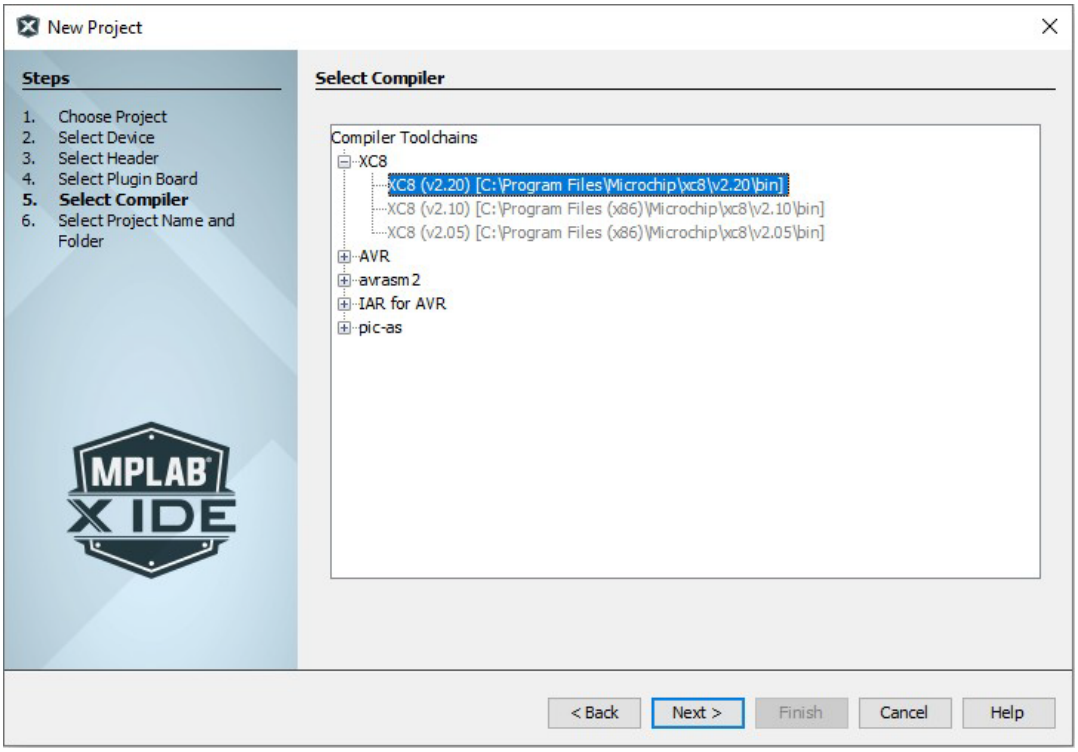
1.4. [Device]フィールドで「AVR128DA48」を検索します。[Tool]カテゴリで、Curiosity Nanoボードがコンピュータに接続されている場合、そのボードを選択します。接続されていない場合は**[None]**を選択します。**[Next]**をクリックします。

図1-20.デバイスの選択



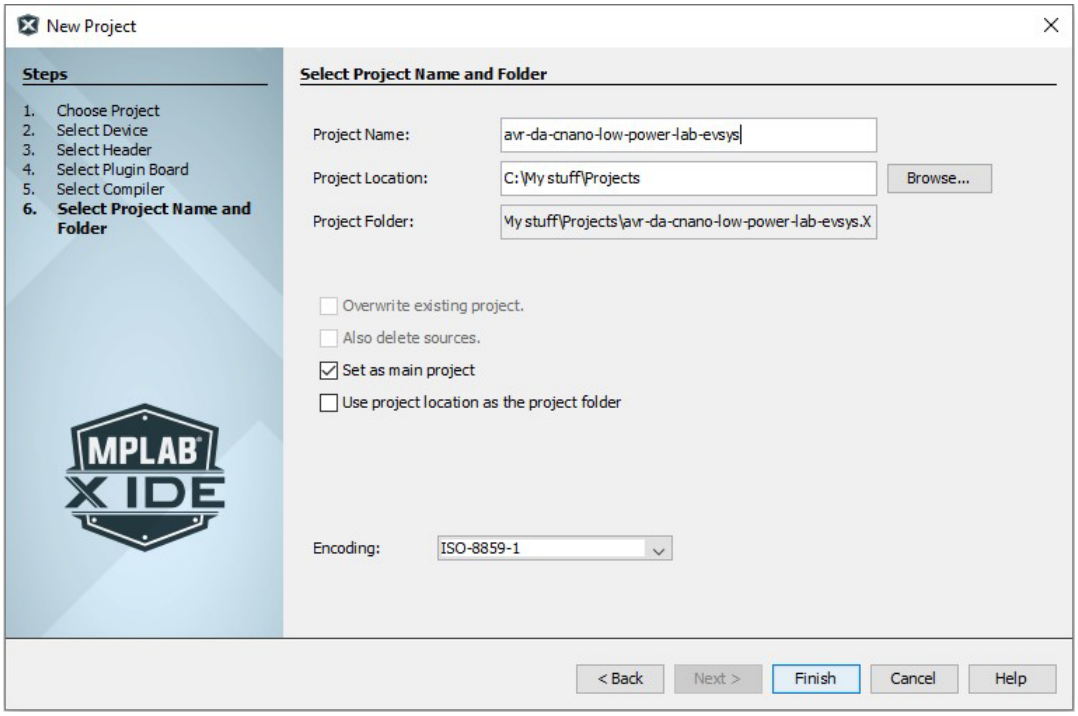
1.5. **[XC8 (2.20)]**コンパイラを選択し、**[Next]**をクリックします。

図1-21.コンパイラの選択



1.6. プロジェクトに名前を付け(保存場所を選択し)、**[Finish]**をクリックします。

図1-22.プロジェクト名

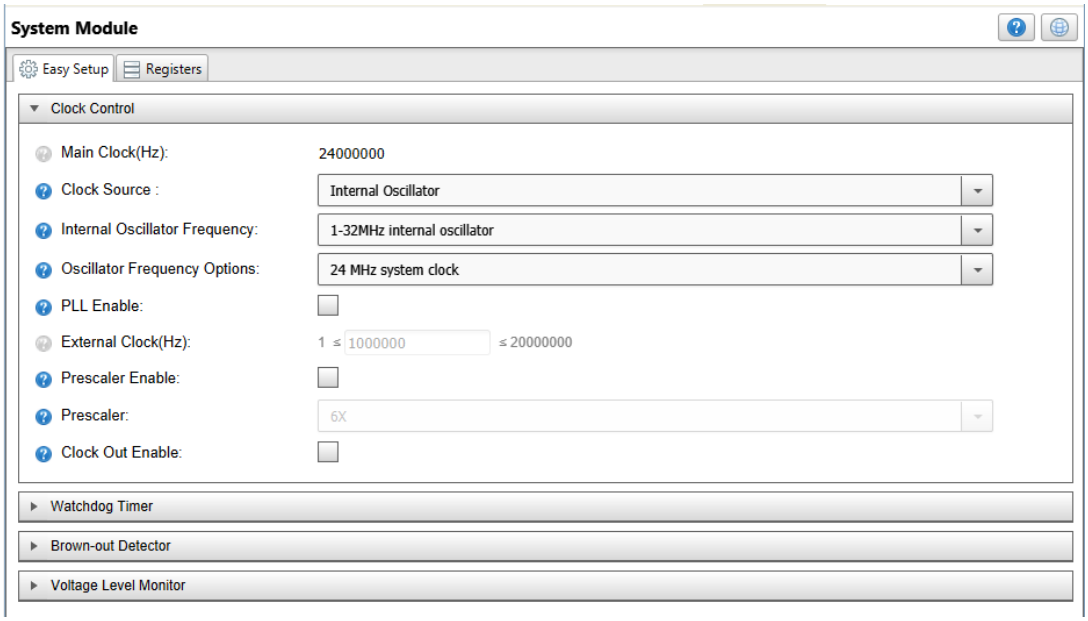


2. MCC (MPLAB Code Configurator)を開き、周辺モジュールを設定します。

2.1. [System Module]で、クロック源として24 MHzクロックの内部オシレータを選択します。

Info: [Clock Source]で[Internal Oscillator]、[Oscillator Frequency Options]で[24 MHz system clock]を選択し、[Prescaler]は無効にします。

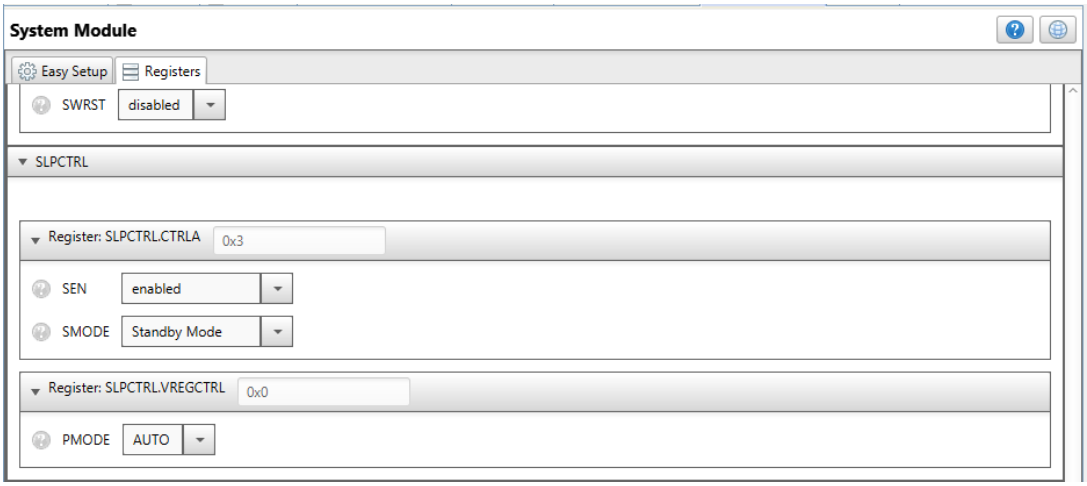
図1-23.システム モジュールの設定



2.2. [System Module]の**[Registers]**タブでスリープを有効にし、モードをスタンバイに設定します。

Info: [System Module]の**[Registers]**タブを開き、[SLPCTRL]までスクロールします。CTRLA レジスタを変更してスリープを有効にし、モードを選択します。

図1-24.スリープの設定

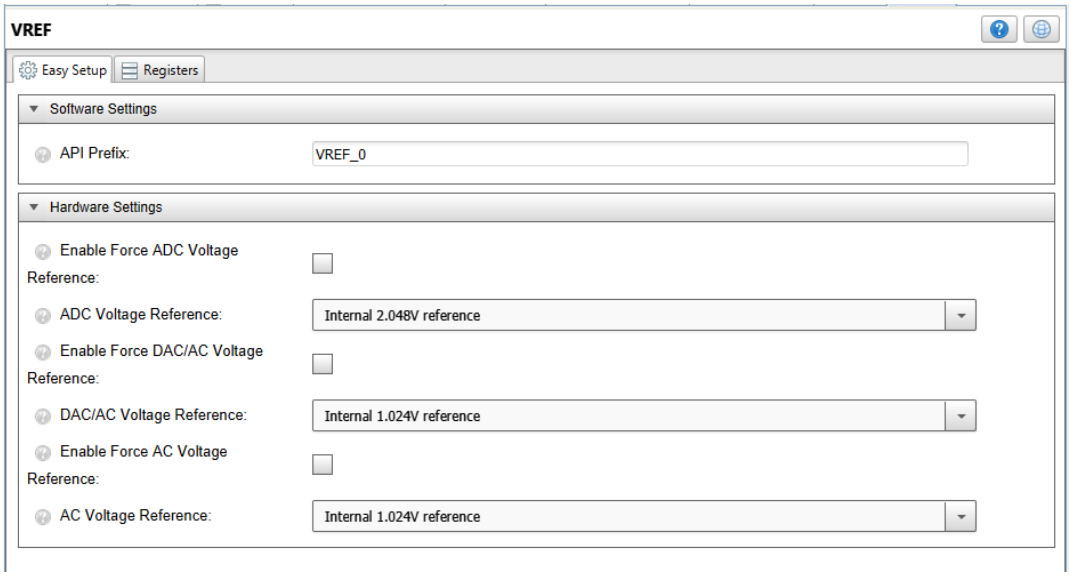


2.3. [Device Resources]からVREFモジュールを追加し、2.048 Vの参照電圧をADCに供給するよう設定します。

Info: ADC 参照電圧は 2.048 V の内部参照電圧に設定する必要があります。[Enable Force ADC Voltage]にはチェックを入れないでください。

Info: VREF はマイクロコントローラの温度センサを有効にするために使われます。

図1-25.VREFの設定



2.4. [Device Resources]からADCモジュールを追加し、12ビットモード、結果は右寄せ、積算なし、サンプル長は1ティック、スタンバイ中動作(RUNSTBY)に設定します。また、イベント受信時に変換が開始されるようにSTARTEIビットも有効にします。

Info: 左揃え(Left Adjust Result)のオプションは無効にする必要があります。サンプル長とサンプル積算数は**[Hardware Settings]**タブで選択します。12 ビットモードは既定値で設定されています。RUNSTBY ビットは**[Registers]**タブで変更できます。これは CTRLA レジスタにあります。STARTEI ビットは**[Registers]**タブの EVCTRL レジスタにあります。

図1-26.ADCの設定

図1-27.RUNSTBYに関するRTCの設定

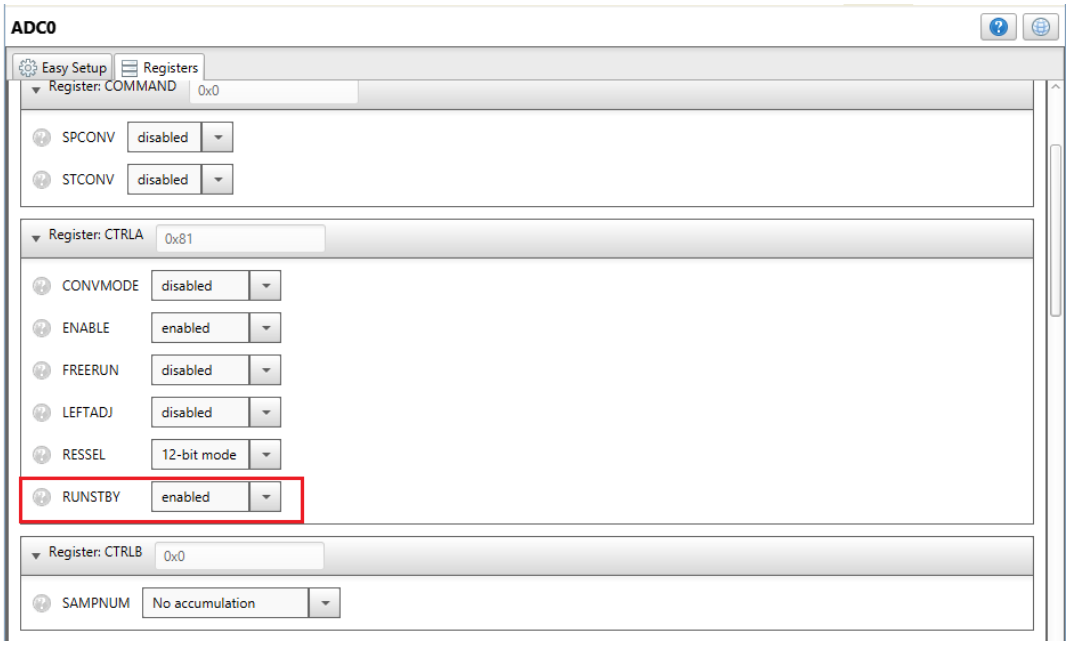
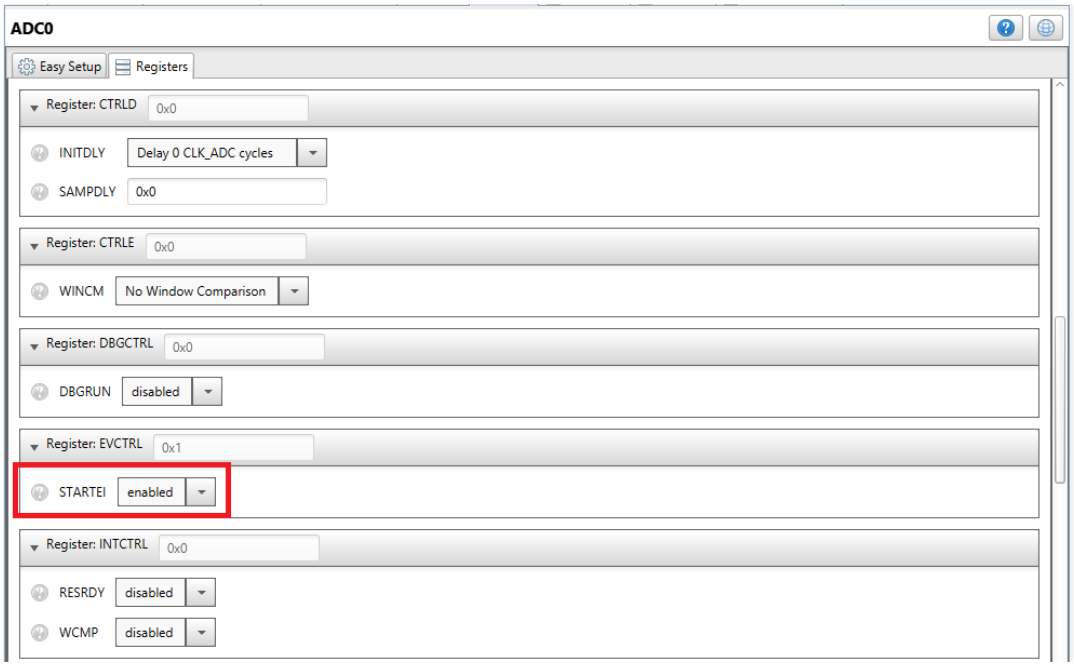


図1-28.イベント受信時変換開始に関するADCの設定



2.5. [Device Resources]からRTCモジュールを追加し、32.768 kHzの内部オシレータを使い、周期を1秒、プリスケール係数を1、スタンバイ中動作(RUNSTBY)を有効に設定します。

Info: クロック、プリスケール係数、周期は**[Hardware Settings]**タブで設定し、RUNSTBY ビットは CTRLA レジスタのレジスタペインにあります。オーバーフロー割り込みが有効になっている場合、無効にします。

図1-29.RTCの設定

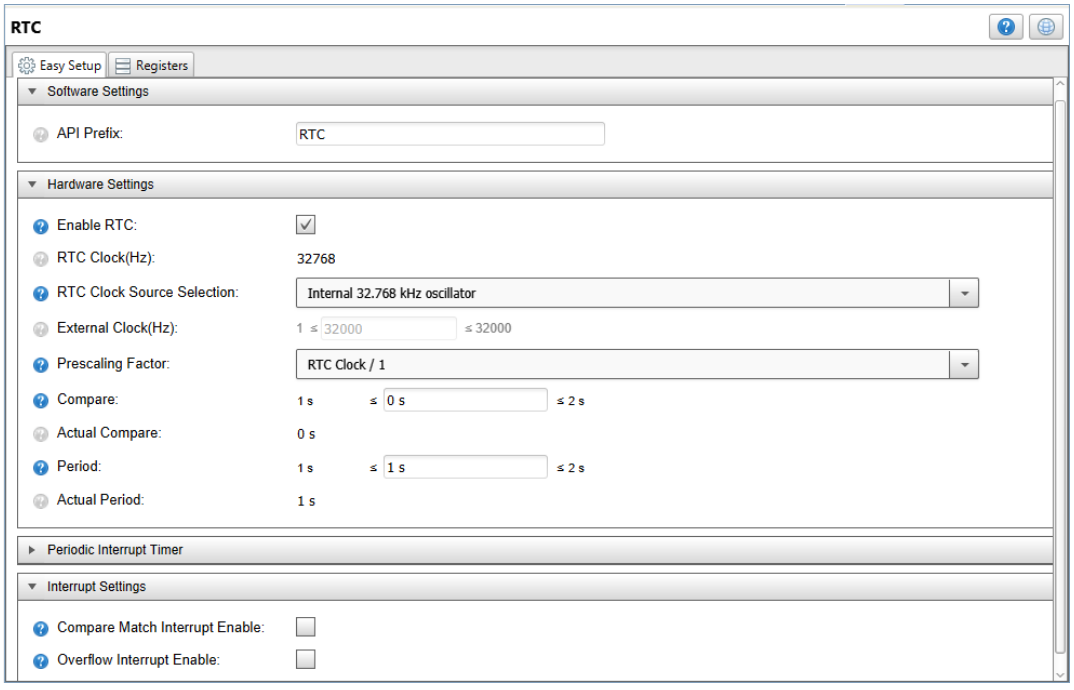
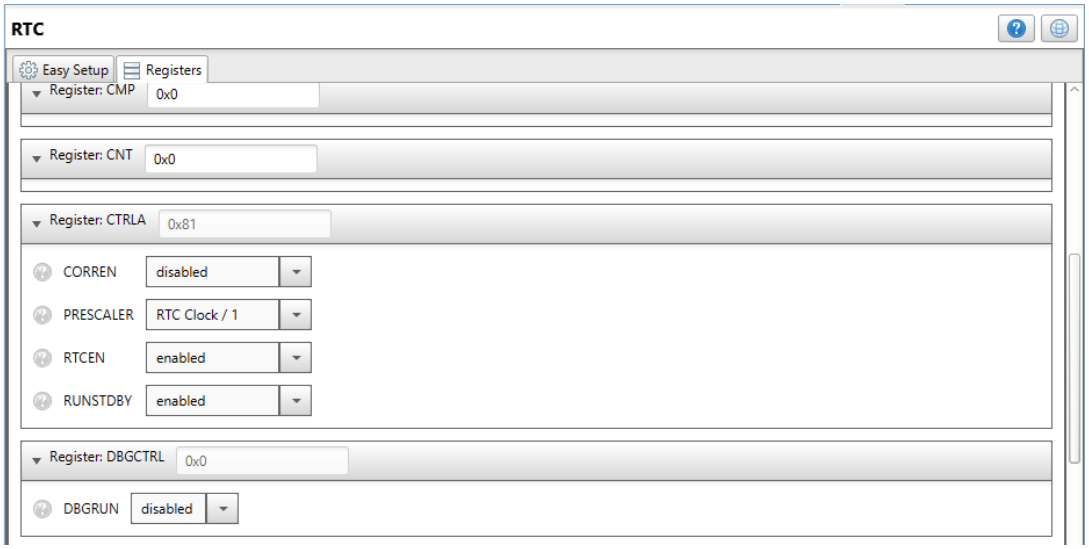


図1-30.RUNSTBYに関するRTCの設定



2.6. EVSYSモジュールを追加します。チャンネル0のイベント ジェネレータをRTC_OVFに設定し、トリガされるイベントをADC0STARTに設定します。チャンネル1のイベント ジェネレータをADC0_RESRDYに設定し、トリガされるイベントをEVSYSEVOUTCに設定します。

図1-31.EVSYSの設定

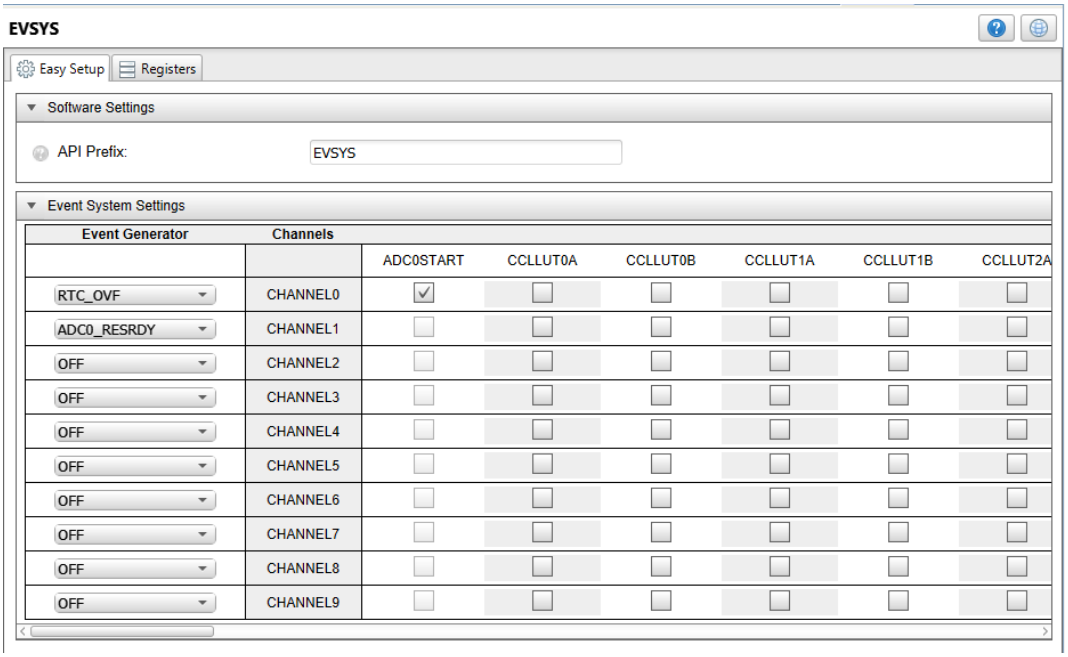
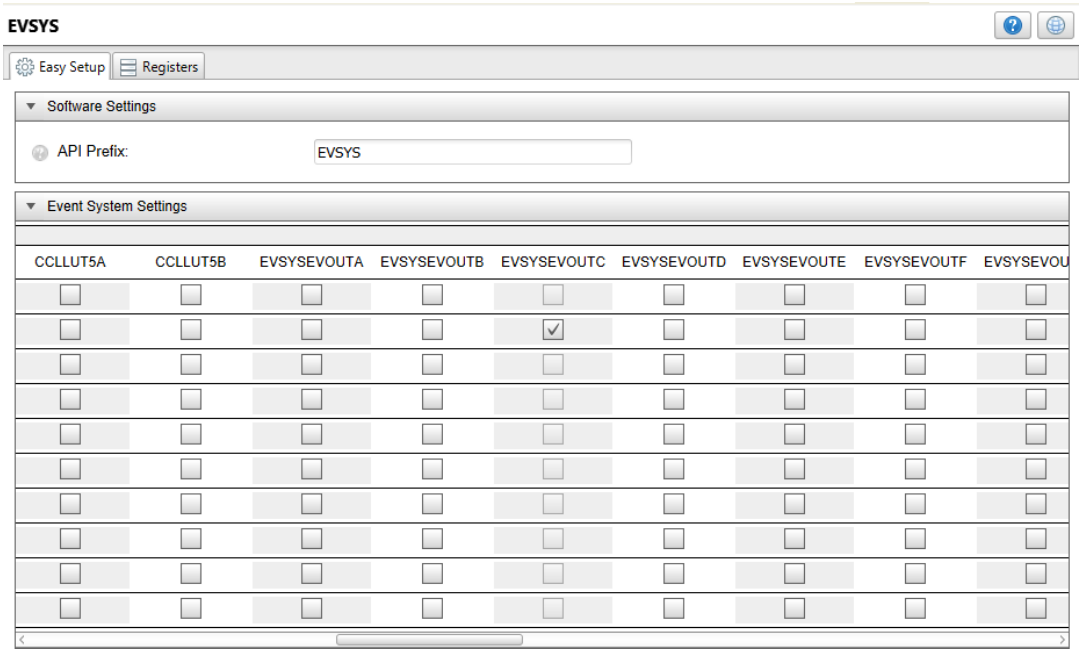
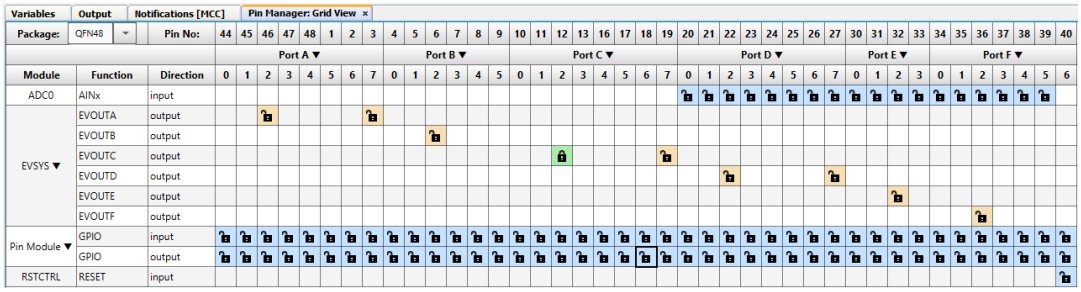


図1-32.EVSYSの設定



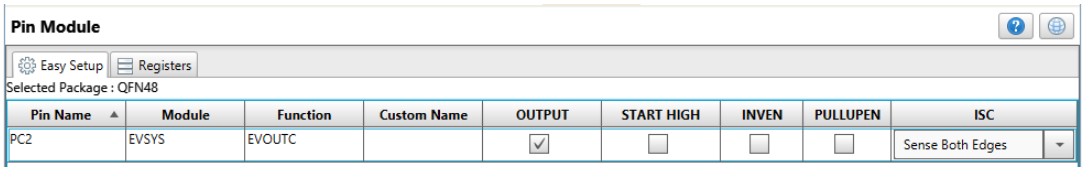
2.7. [Pin Manager: Grid View]で、EVSYSのEVOUTCとして**[PC2]**を選択します。

図1-33.[Pin Manager: Grid View]



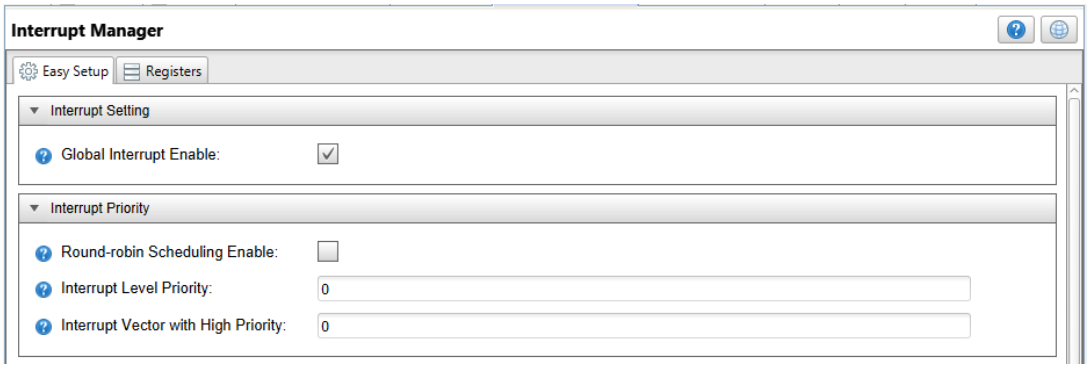
2.8. [Pin Module]で、PC2ピンの状態変化割り込みを両エッジ検出で有効にします。

図1-34.ピンモジュールの設定



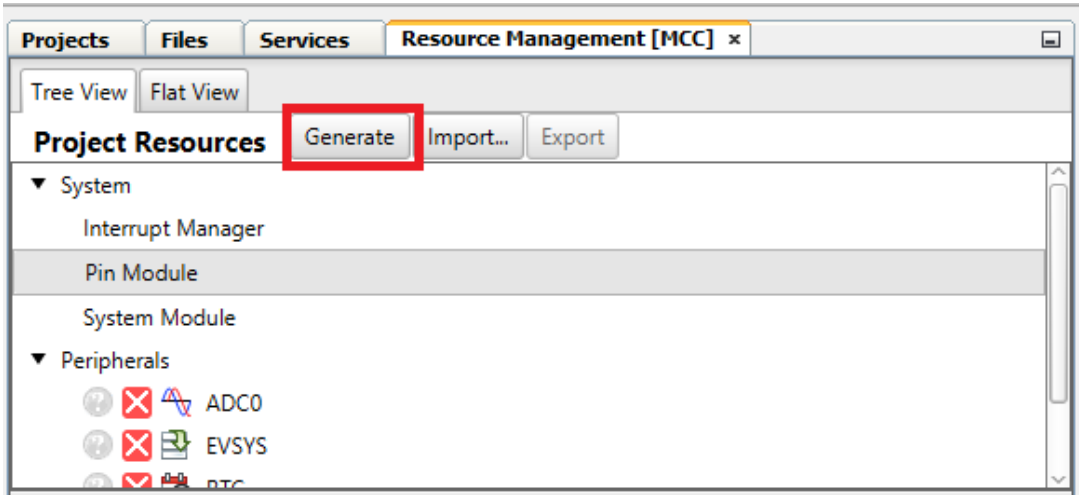
2.9. [Interrupt Manager]でグローバル割り込みを有効にします。

図1-35.割り込みマネージャの設定



2.10. **[Generate]**をクリックします。

図1-36.[Generate]ボタン



3. プロジェクトにコードを追加します。

3.1. `main.c`ファイルに以下のコードを追加します。

```
#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()`は`mcc.c`で定義されます。
`ADC0_GetConverisonResult()`は`adc.c`で定義されます。
`PORTC_PC2_SetInterruptHandler()`は`pin_manager.c`で定義されます。

- `SYSTEM_Initialize()`は CPU と周辺モジュール用の全てのコンフィグレーションレジスタを設定します。この関数は MCC によって生成されます。
- イベントシステムは即座に変換を開始し、その時点でチャンネルを指定する方法がないため、レジスタ ADC0.MUXPOS は main で設定されます。
- `PORTC_PC2_SetInterruptHandler(GPIO_Interrupt)`は PC2 ピンの状態変化割り込みがトリガされた時に呼び出される関数を設定します。
- `ADC0_GetConversionResult()`は変換の最後の結果を返します。

3.2. [Source Files] > [MCC Generated Files] > [src]にある`pin_manager.c`ファイルで、ピンの方向に関するコードを以下に置き換えます。

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

Info: これは電力読み値に干渉するフローティングのピンを防ぐために行います。

3.3. ツールバーから**[Clean and Build]**ボタンを押して、プログラムがエラーなくビルドされる事を確認します。

図1-37.クリーンとビルド

GitHubでサンプルコードを見る
クリックしてリポジトリを閲覧

1.6 デバイスのプログラミングと効率の比較

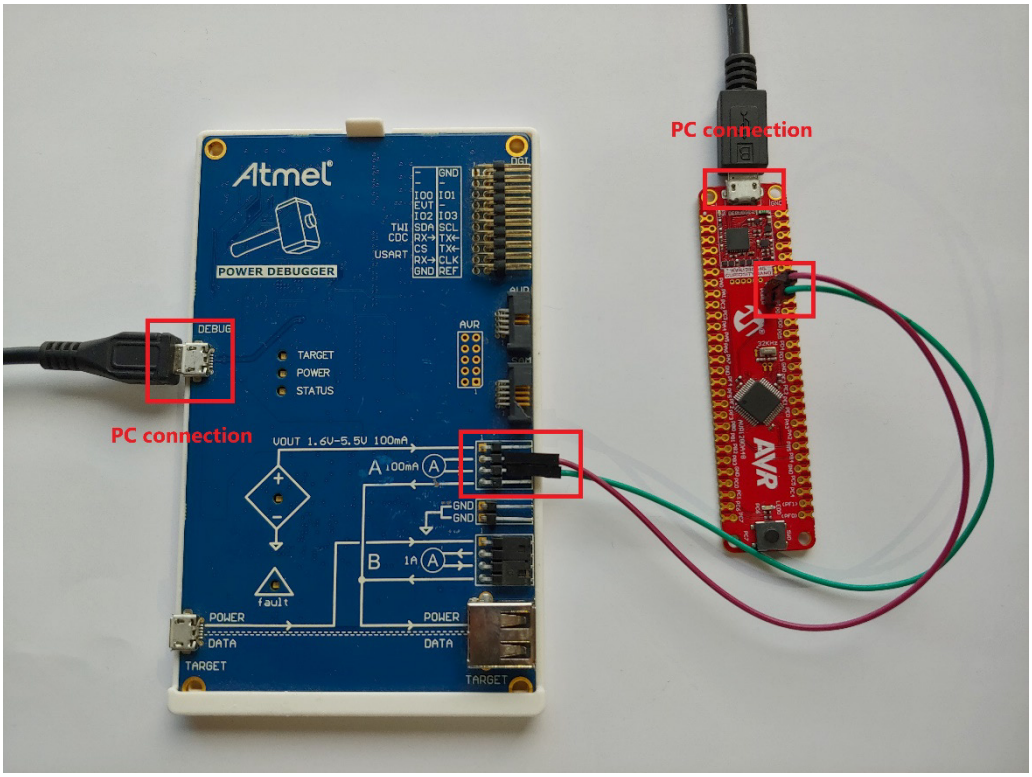
To do: マイクロコントローラにプロジェクトを読み込み、電力読み値を取得します。

1. ソフトウェア実装プログラムから消費電力の計測値を取得します。
   1.1. Curiosity Nanoボードをコンピュータと、はんだ付けした2本のピンに接続します。
   1.2. ソフトウェア実装プログラムをメインプログラムに設定して、ツールバーから**[Make and Program Device]**ボタンを押します。プログラマとして使うツールを選択するためのポップアップが表示される場合があります。AVR128DA48 Curiosity Nanoボードを選択します。

図1-38.デバイスのmakeとプログラミング

1.3. Power Debuggerをコンピュータに接続します。1本目のピン(Nano組み込みデバッガに近い方)をAチャンネル電流計の内向きの矢印に接続し、2本目のピン (マイクロコントローラに近い方)をAチャンネル電流計の外向きの矢印に接続します。

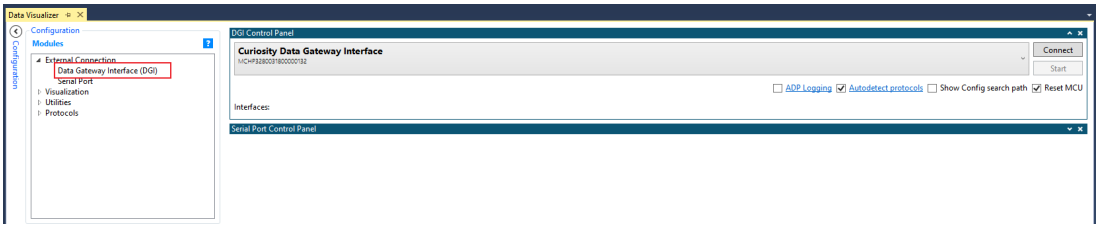
図1-39.ハードウェアの接続



1.4. Atmel Studioで[Tools] > [Data Visualizer]を選択します。

1.5. 2つのボックスが表示されます。**[DGI (Data Gateway Interface) Control Panel]**と書いてある方をチェックします。表示されない場合、左側の[Configuration]メニューにアクセスします。[Modules] > [External Connection]セクションの[Data Gateway Interface]にあります。

図1-40.DGI

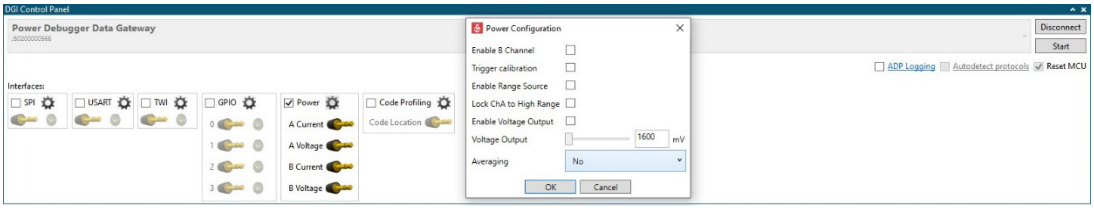


1.6. **[Connect]**をクリックすると、[Power]というボックスが表示されます。チェックを入れて歯車アイコンをクリックすると、メニューが開きます。Bチャンネルを無効にします。

⚠ WARNING	[Power]フィールドが表示されない場合、ボードを取り外して接続し直します。

図1-41.DGIの接続

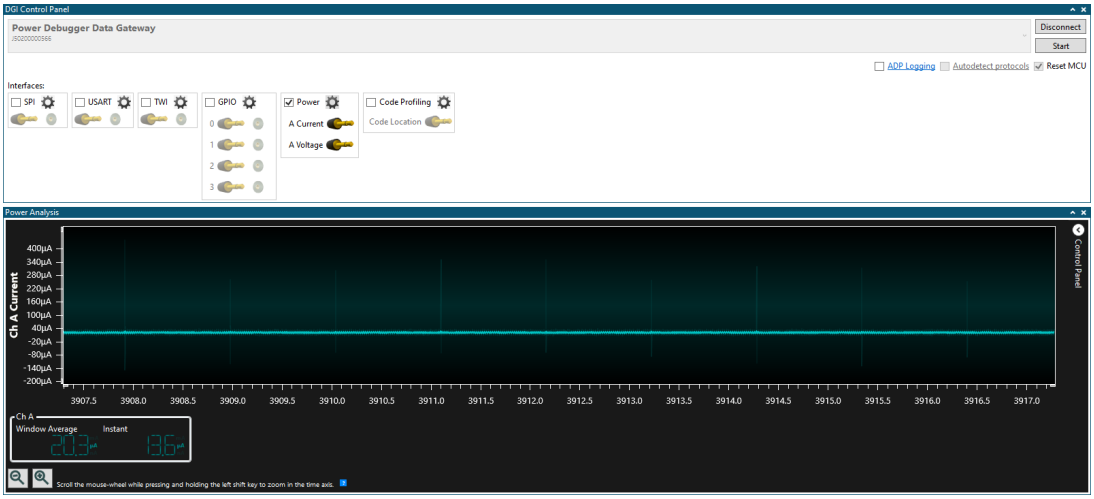
図1-42.[Power]の設定



1.7. 全ての設定が正しく完了したら**[Start]**をクリックすると、消費電力をリアルタイムで表示する新しいボックスが表示されます。

Tip: 表示を調整するにはボックスの左側で[Control Panel]を開きます。[Show zero]のチェックを外すと見やすくなります。

図1-43.監視の開始



1.8. 読み値が安定したら、**[Stop]**を押してから切断します。電力計測が表示されたボックスは開いたままになります。

2. イベントシステム実装プログラムから消費電力の計測値を取得します。

2.1. Curiosityボード上のはんだ付けされた2本のピンを互いに接続します。

2.2. イベントシステム実装プロジェクトをメインプログラムに設定して、MPLAB Xのツールバーから**[Make and Program Device]**ボタンをクリックします。

図1-44.デバイスのmakeとプログラミング

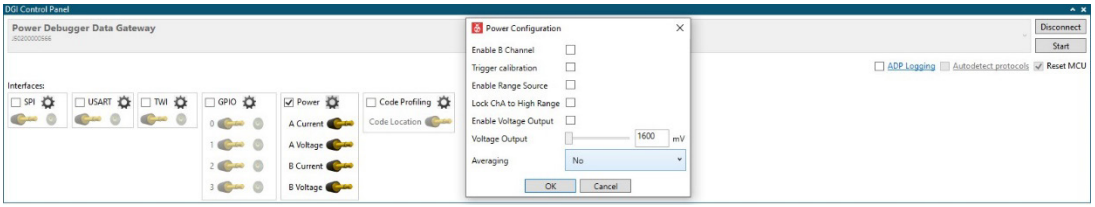
2.3. Curiosity NanoボードをPower Debuggerに接続します。詳細と画像は本章のステップ1.3を参照してください。

2.4. Atmel StudioのData Visualizerに戻り、[DGI Control Panel]で**[Connect]**をクリックします。

図1-45.DGI

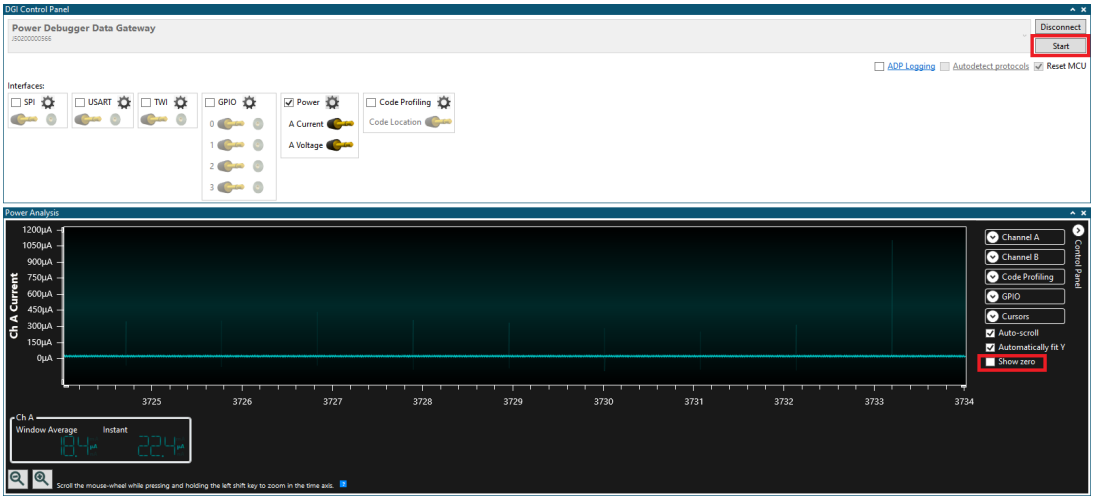
2.5. [Power]ボックスにチェックを入れ、設定を編集してBチャンネルを無効にします。

図1-46.[Power]の設定



2.6. **[Start]**をクリックすると、消費電力をリアルタイムで表示する新しいボックスが表示されます。[Show zero]を無効にすると見やすくなります。

図1-47.監視の開始

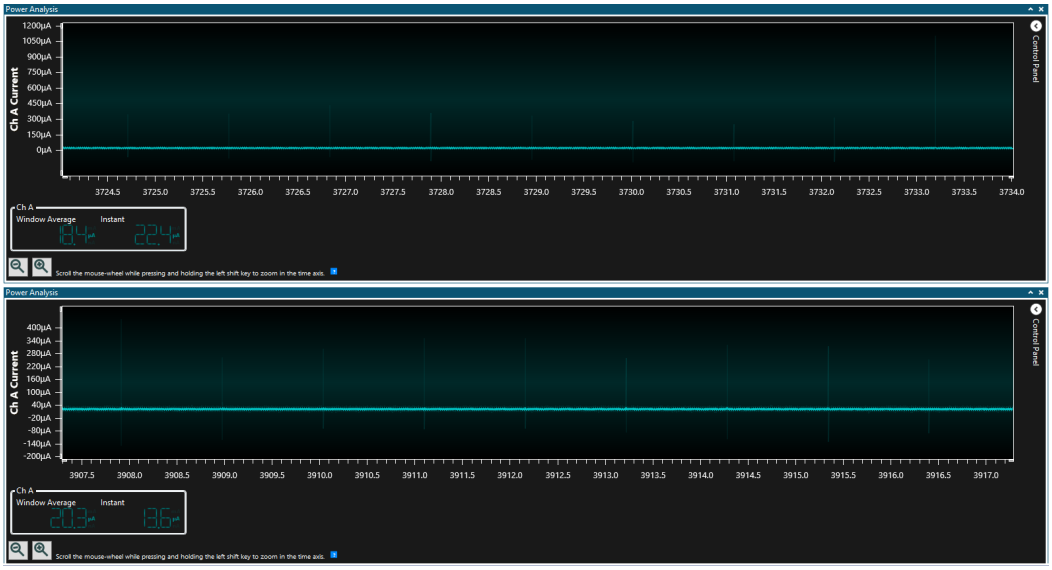


2.7. 読み値が安定したら、**[Stop]**を押して固定された結果を得ます。

3. 2つの計測値を比較します。

3.1. 両方の消費電力が同じ画面に表示されます。これらを解析する事でプログラムの挙動を観察できます。

図1-48.ソフトウェア実装とEVSYSの比較



Info: 上側にイベントシステム実装の消費電力、下側に割り込み実装の消費電力が表示されています。画像から分かるように、その差は 1.5 µA とかなり小さくなっています。これはマイクロコントローラがスリープしている時間が長いためです。

2. 課題2: ADCのソフトウェア積算とハードウェア積算の比較

この課題では、2種類のADC積算手法の消費電力を比較します。ADCで結果を積算する事には、結果の精度を向上し、ノイズの影響を最小限に抑えるという役割があります。

1つ目の手法では、128回に達するまで繰り返し変換をトリガします。結果はアキュムレータとして機能する変数に追加されます。

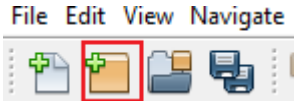
2つ目の手法では、ADCの内蔵アキュムレータを使います。ADCの積算変換モードを使って変換を128回実行し、その結果をハードウェア アキュムレータに追加します。

2.1 ソフトウェア実装プロジェクトの作成

To do: このプロジェクトで使うハードウェア モジュールを設定し、コードをプロジェクトに追加します。

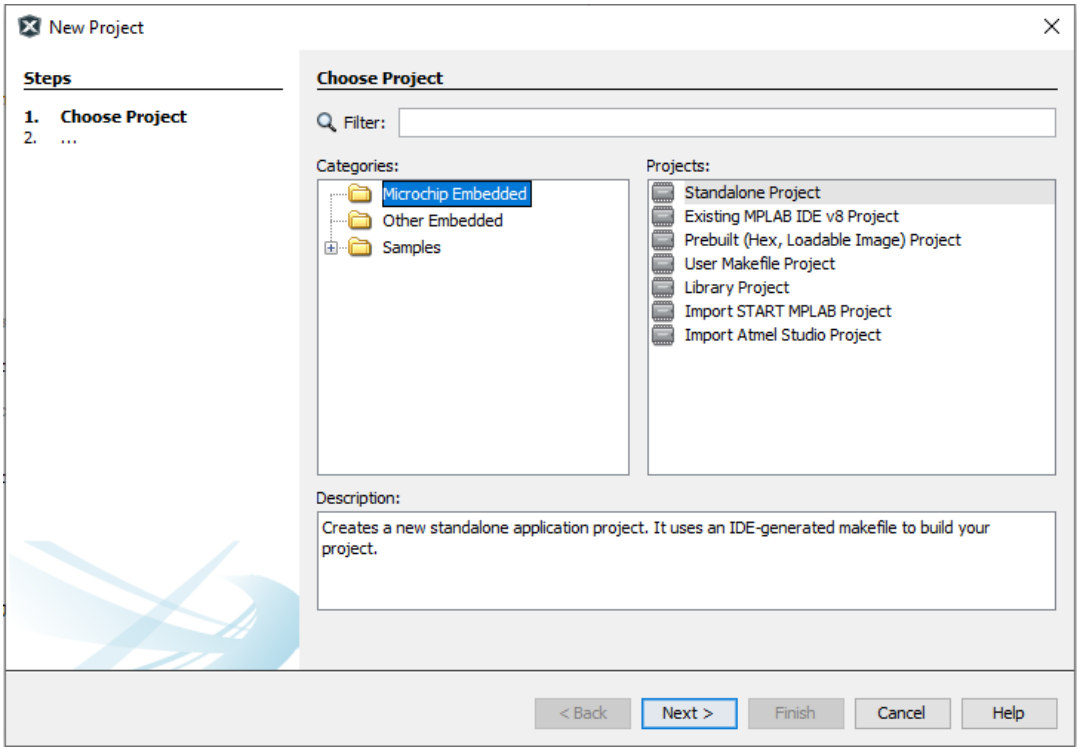
1. AVR128DA48用のMPLAB Xプロジェクトを新規作成します。
   1.1. MPLAB Xを開きます。
   1.2. [File] > [New Project]を選択するか、または**[New Project]**ボタンをクリックします。

図2-1.プロジェクトの新規作成



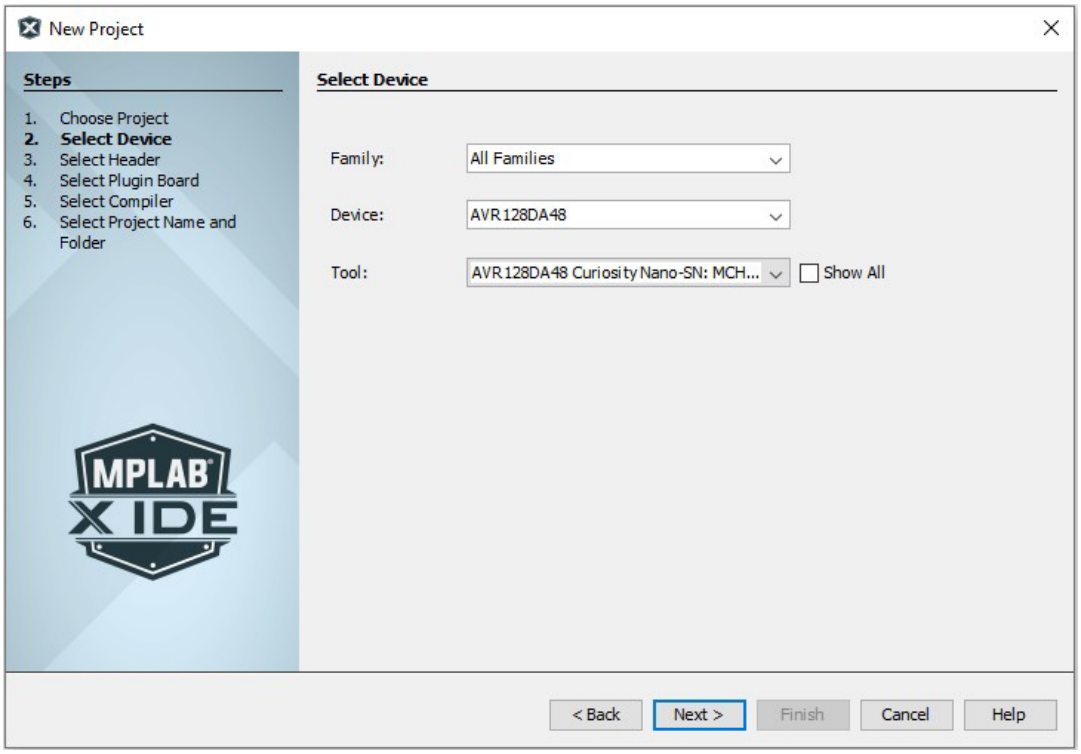
   1.3. **[Next]**をクリックします(既定値では[Microchip Embedded] > [Standalone Project]が選択されています)。

図2-2.プロジェクトのタイプ



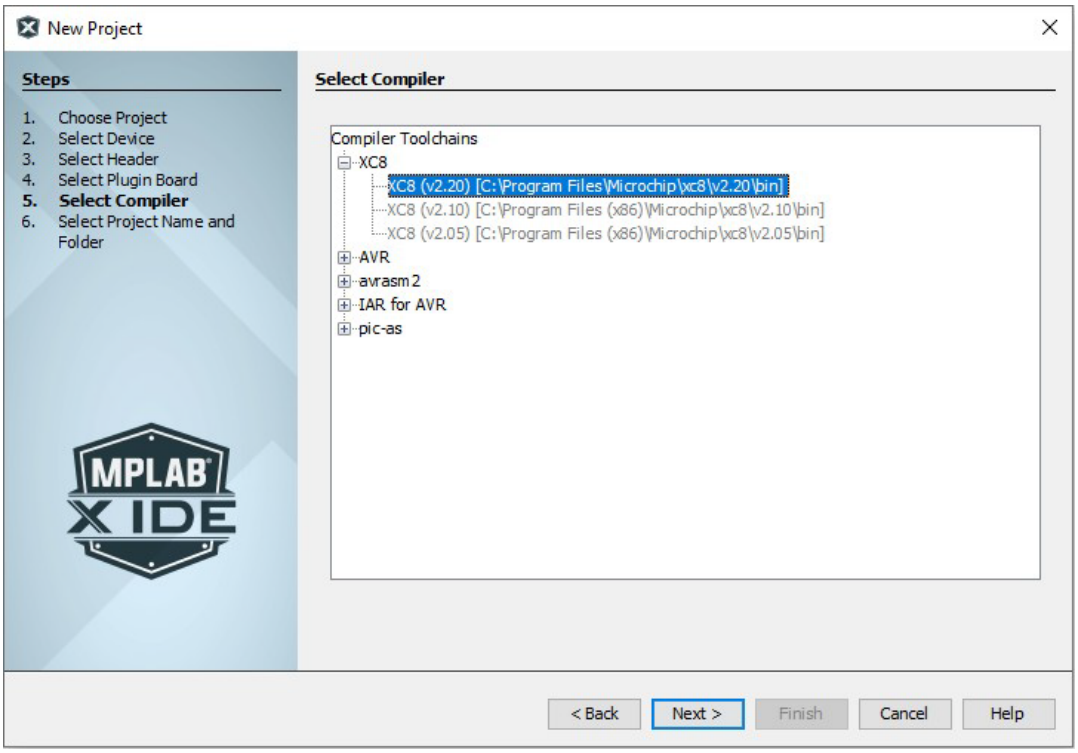
1.4. [Device]フィールドで「AVR128DA48」を検索します。[Tool]カテゴリで、Curiosity Nanoボードがコンピュータに接続されている場合、その**[Curiosity Nano]**ボードを選択します。接続されていない場合は**[None]**を選択します。**[Next]**をクリックします。

図2-3.デバイスの選択



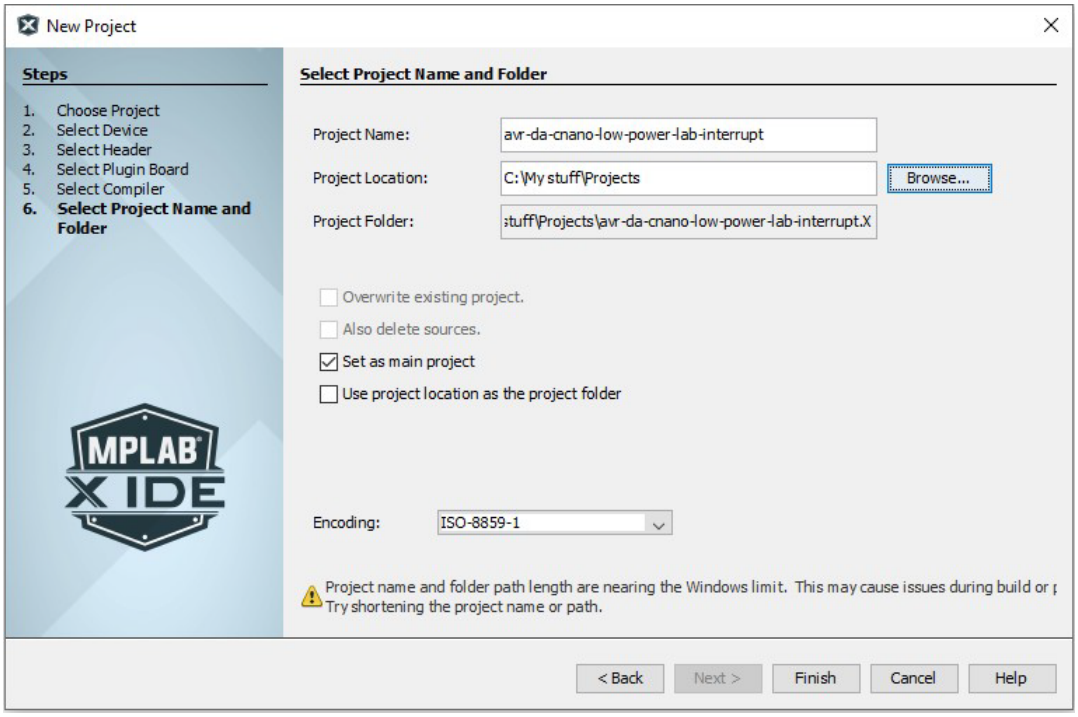
1.5. **[XC8 2.20]**コンパイラを選択して**[Next]**をクリックします。

図2-4.コンパイラの選択



1.6. プロジェクトに名前を付け(保存場所を選択し)、**[Finish]**をクリックします。

図2-5.プロジェクト名

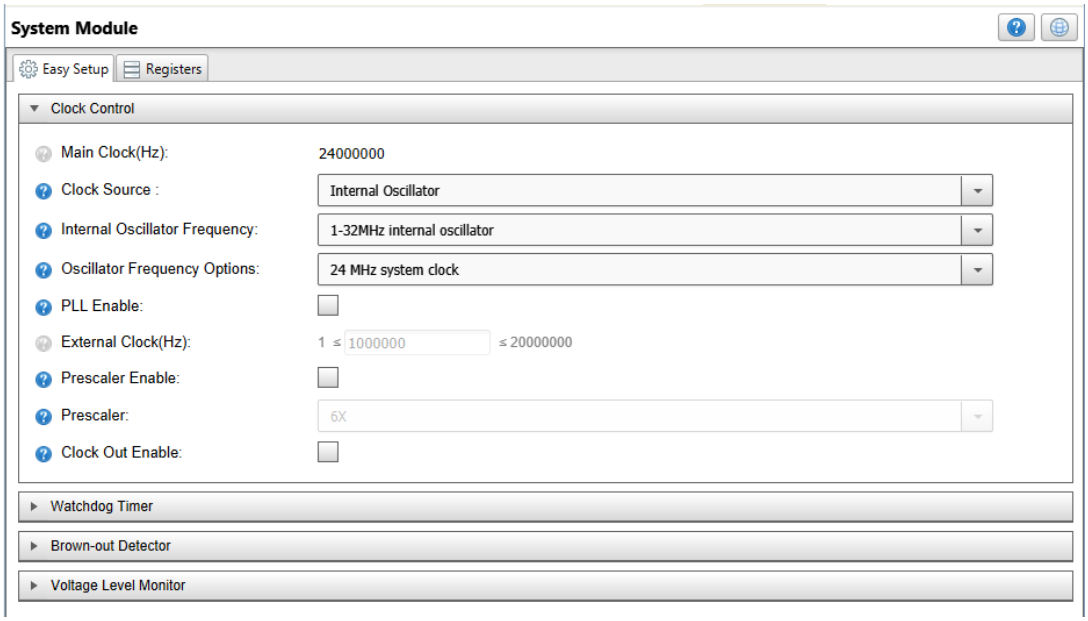


2. MCC (MPLAB Code Configurator)を開き、周辺モジュールを設定します。

2.1. [System Module]を24 MHzの内部オシレータで動作するよう設定します。

Info: クロック源は内部オシレータにし、周波数は 24 MHz にします。プリスケーラのオプションは無効にする必要があります。

図2-6.システム モジュールの設定

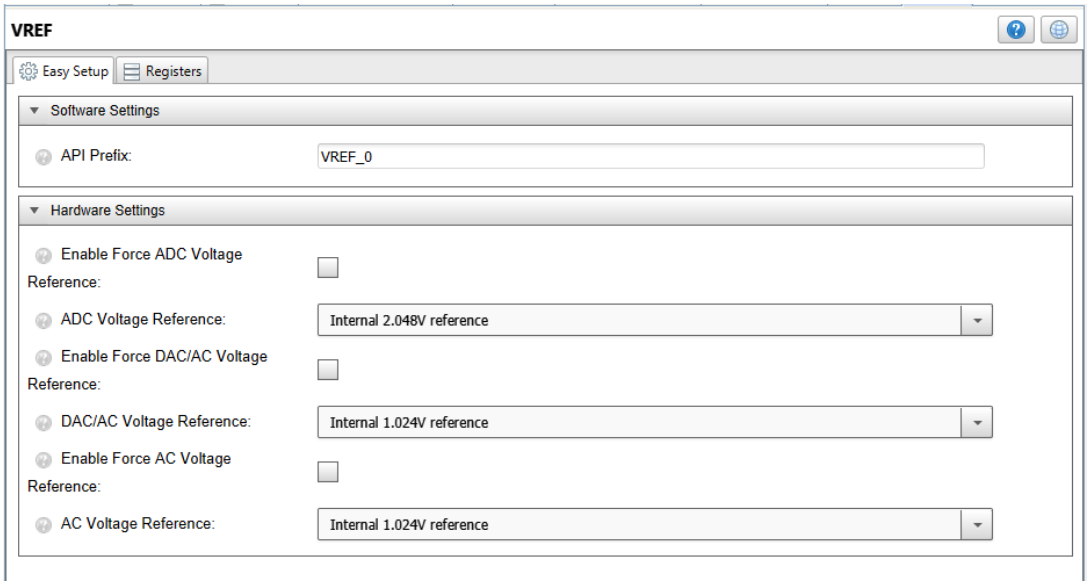


2.2 [Device Resources]からVREFモジュールを追加し、2.048 Vの参照電圧をADCに供給するよう設定します。

Info: ADC 参照電圧は 2.048 V の内部参照電圧に設定する必要があります。[Enable Force ADC Voltage]にはチェックを入れないでください。

Info: VREF はマイクロコントローラの温度センサを有効にするために使われます。

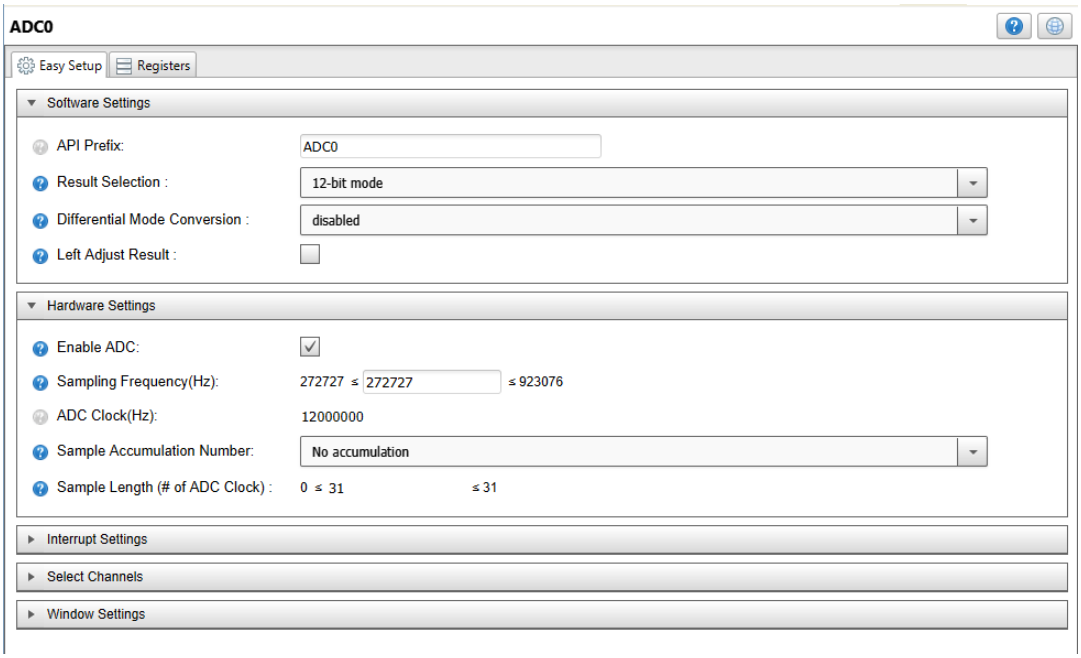
図2-7.VREFの設定



2.3 [Device Resources]からADCモジュールを追加し、12ビットモード、結果は右寄せ、積算なし、サンプル長は31ティックに設定します。

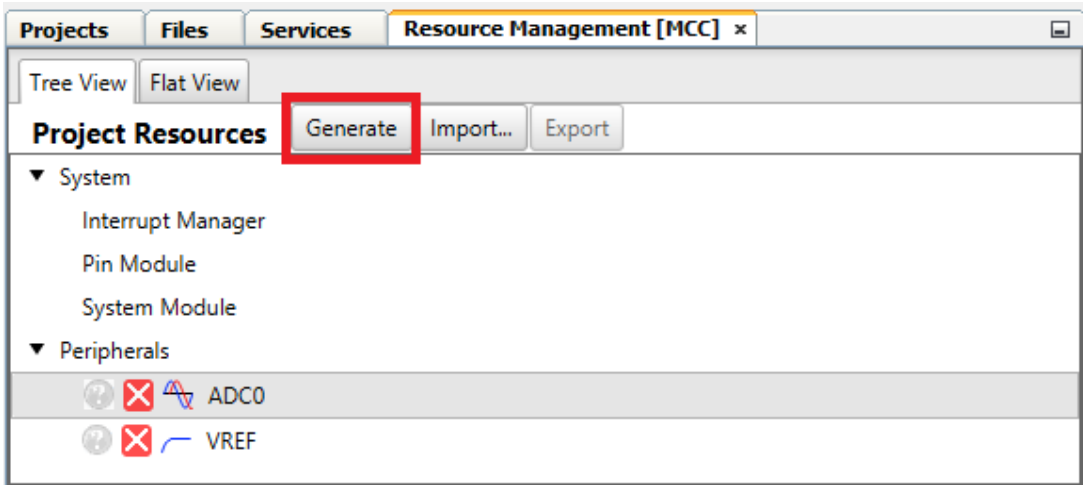
Info: 左揃え(Left Adjust Result)のオプションは無効にする必要があります。サンプル長は **[Hardware Settings]**タブで選択します。12 ビットモードは既定値で設定されています。[No accumulation]の選択も同様です。

図2-8.ADCの設定



2.4 **[Generate]**ボタンをクリックします。

図2-9.[Generate]ボタン



3. プロジェクトにコードを追加します。

3.1 `main.c`ファイルに以下のコードを追加します。

```
#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()`は`mcc.c`で定義されます。`ADC0_StartConversion(channel)`、`ADC0_IsConversionDone()`、`ADC0_GetConversionResult()`は`adc0.c`で定義されます。

- `SYSTEM_Initialize()`は MCC で設定された内容に従ってマイクロコントローラと周辺モジュールを設定する関数です。この関数は MCC により生成され、プログラムの最初で呼び出される必要があります。
- `ADC0_StartConversion(channel)`は指定されたチャンネルで ADC 変換を開始します。
- `ADC0_IsConversionDone()`は変換のステータスを返します。0 は処理中、1 は終了済みです。
- `ADC0_GetConversionResult()`は最後の変換の結果を返します。
- for ループは変換を開始して変換の終了を待機しその結果をアキュムレータに追加するステップを 128 回反復します。

3.2 [Source Files] > [MCC Generated Files] > [src]にある`pin_manager.c`ファイルで、ピンの方向に関するコードを以下に置き換えます。

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

Info: これは電力読み値に干渉するフローティングのピンを防ぐために行います。

3.3 ツールバーから**[Clean and Build]**ボタンを押して、プログラムがエラーなくビルドされる事を確認します。

図2-10.クリーンとビルド

GitHubでサンプルコードを見る
クリックしてリポジトリを閲覧

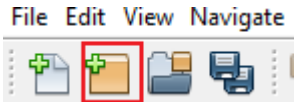
2.2 ハードウェア実装プロジェクトの作成

To do: このプロジェクトで使うハードウェア モジュールを設定し、コードをプロジェクトに追加します。

1. AVR128DA48用のMPLAB Xプロジェクトを新規作成します。

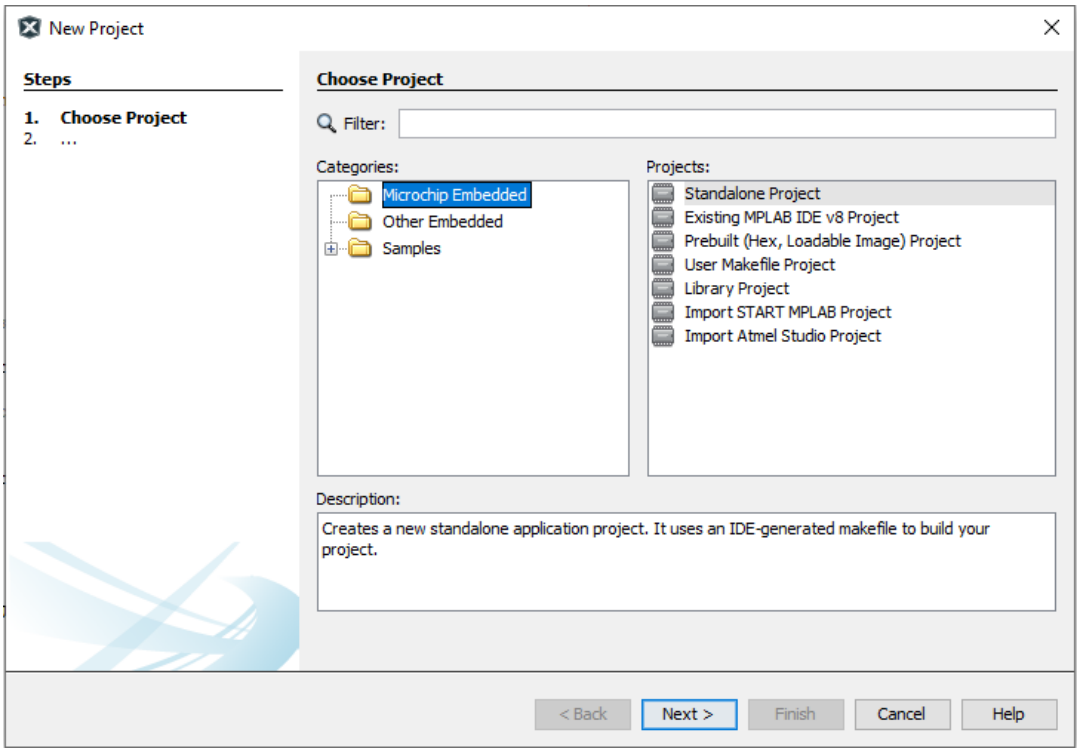
1.1. MPLAB Xを開きます。
1.2. [File] > [New Project]を選択するか、または**[New Project]**ボタンをクリックします。

図2-11.プロジェクトの新規作成



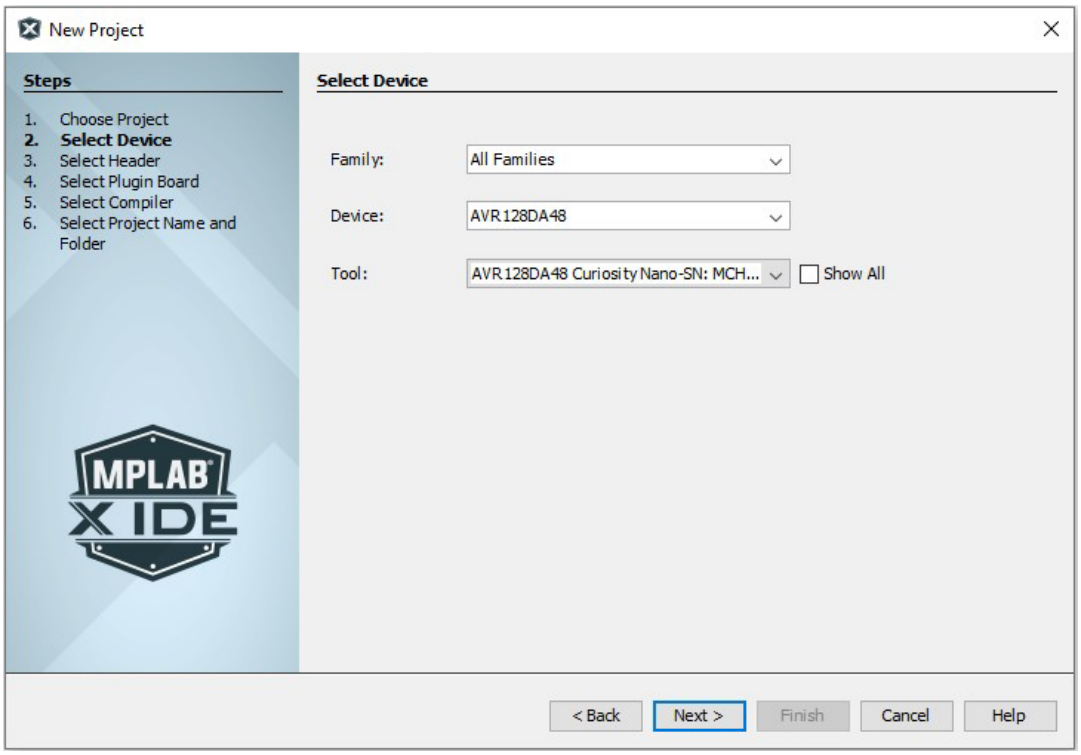
1.3. **[Next]**をクリックします(既定値では[Microchip Embedded] > [Standalone Project]が選択されています)。

図2-12.プロジェクトのタイプ



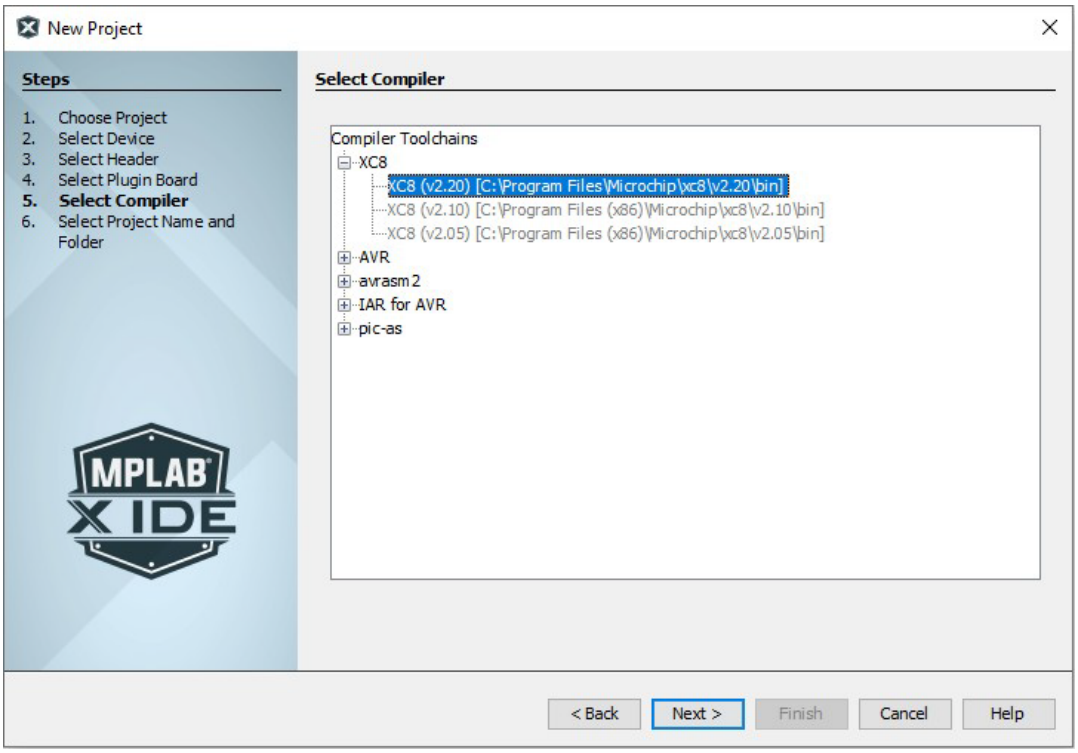
1.4. [Device]フィールドで「AVR128DA48」を検索します。[Tool]カテゴリで、Curiosity Nanoボードがコンピュータに接続されている場合、その**[Curiosity Nano]**ボードを選択します。接続されていない場合は**[None]**を選択します。**[Next]**をクリックします。

図2-13.デバイスの選択



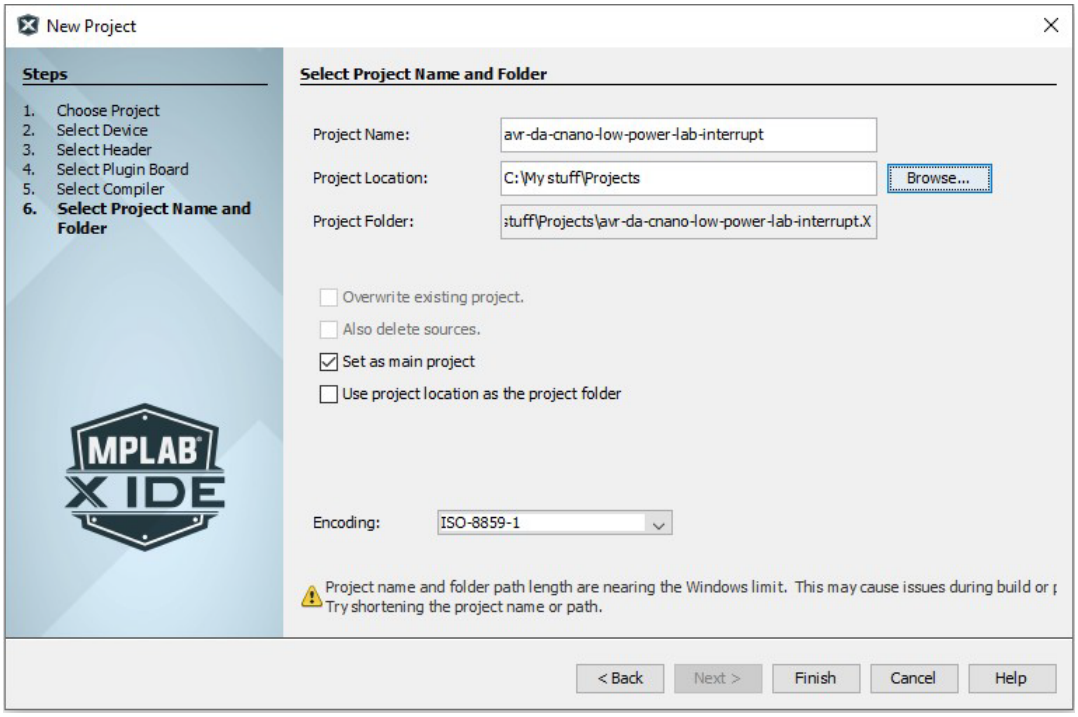
1.5. **[XC8 2.20]**コンパイラを選択して**[Next]**をクリックします。

図2-14.コンパイラの選択



1.6. プロジェクトに名前を付け(保存場所を選択し)、**[Finish]**をクリックします。

図2-15.プロジェクト名

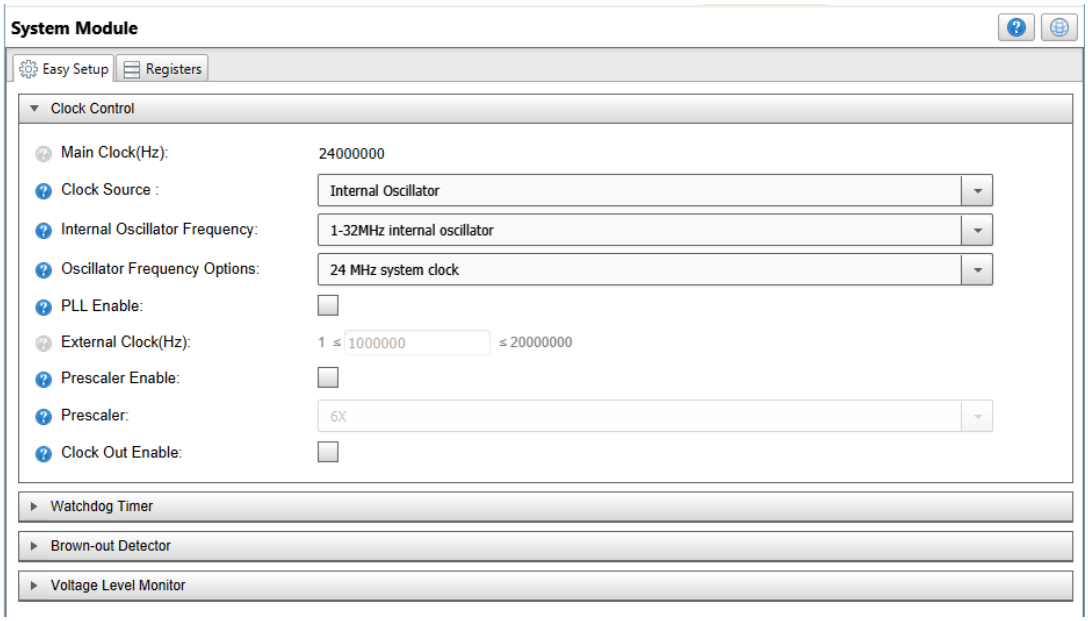


2. MCC (MPLAB Code Configurator)を開き、周辺モジュールを設定します。

2.1. [System Module]を24 MHzの内部オシレータで動作するよう設定します。

Info: クロック源は内部オシレータにし、周波数は 24 MHz にします。プリスケーラのオプションは無効にする必要があります。

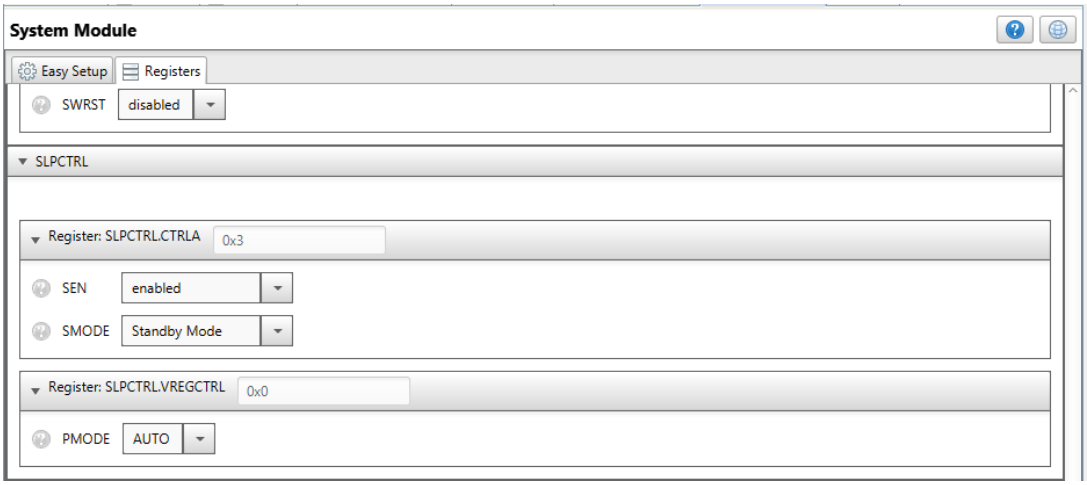
図2-16.システム モジュールの設定



2.2. スリープを有効にしてモードがスタンバイになるようにスリープ オプションを設定します。

Info: [System Module]の**[Registers]**ページを開き、[SLPCTRL]までスクロールします。SLPCTRL.CTRLA レジスタのオプションを変更して有効にし、[Standby Mode]に設定します。

図2-17.スリープの設定

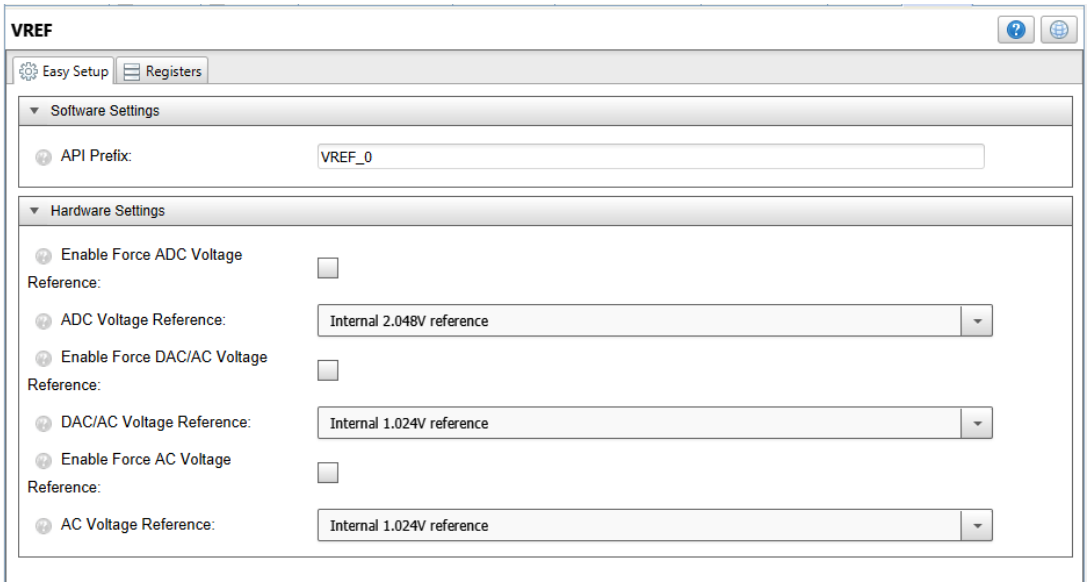


2.3. [Device Resources]からVREFモジュールを追加し、2.048 Vの参照電圧をADCに供給するよう設定します。

Info: ADC 参照電圧は 2.048 V の内部参照電圧に設定する必要があります。[Enable Force ADC Voltage]にはチェックを入れないでください。

Info: VREF はマイクロコントローラの温度センサを有効にするために使われます。

図2-18.VREFの設定



2.4. [Device Resources]からADCモジュールを追加し、12ビットモード、結果は右寄せ、128個の結果を積算する、サンプル長は31ティック、スタンバイ中動作(RUNSTBY)に設定します。

Info: 左揃え(Left Adjust Result)のオプションは無効にする必要があります。サンプル長とサンプル積算数は**[Hardware Settings]**タブで選択します。12 ビットモードは既定値で設定されています。RUNSTBY ビットは[Registers]タブで変更できます。これは CTRLA レジスタにあります。

図2-19.ADCの設定

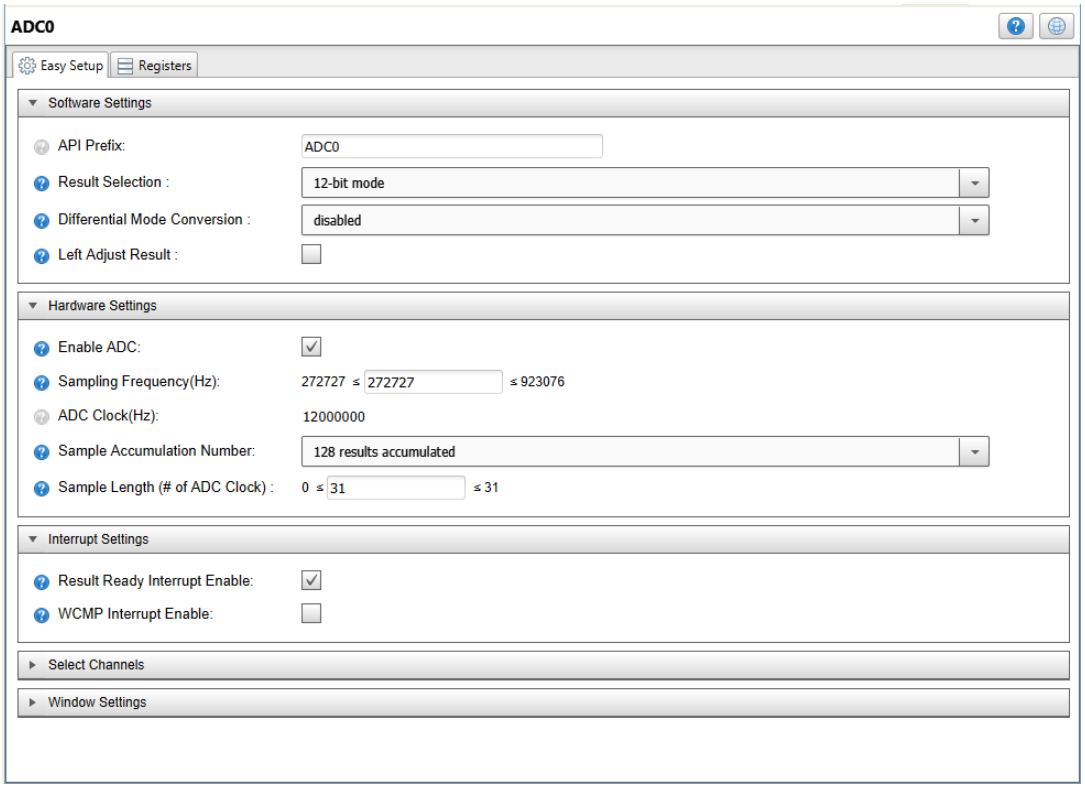
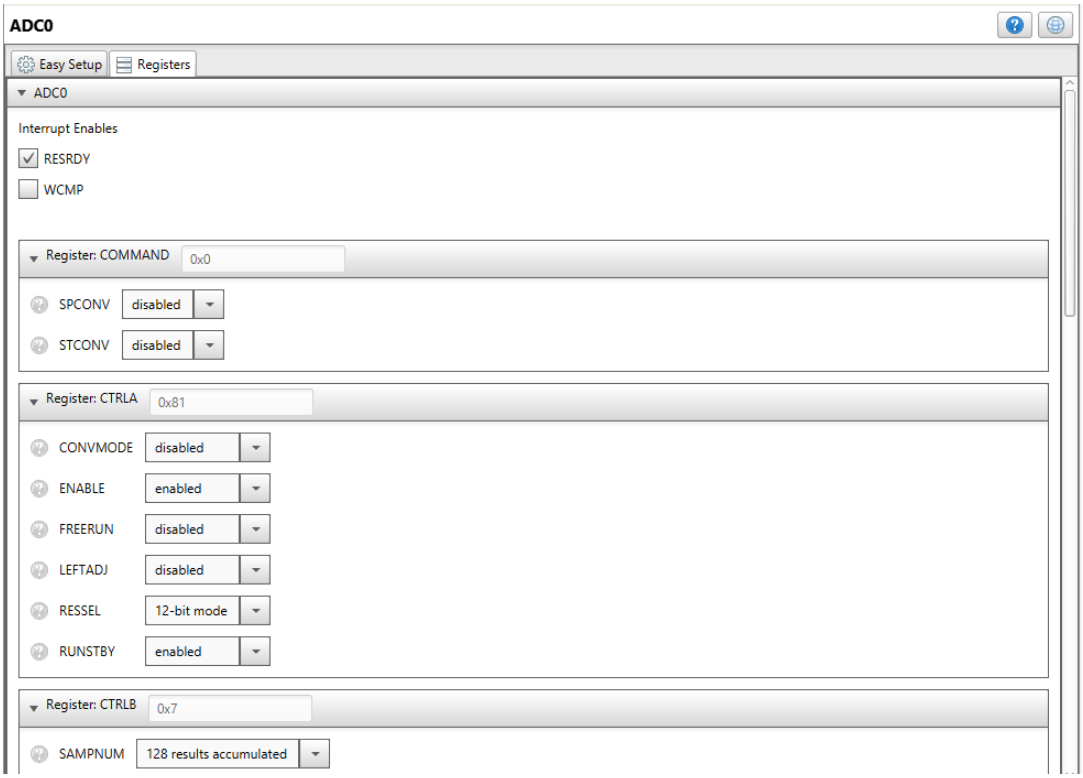


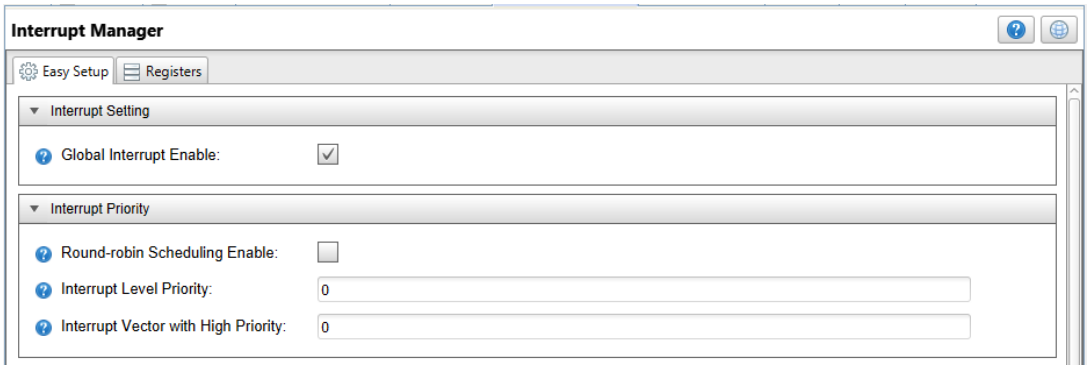
図2-20.スタンバイ中動作に関するADCの設定



2.5. ADCの結果レディ割り込みとグローバル割り込みを有効にします。

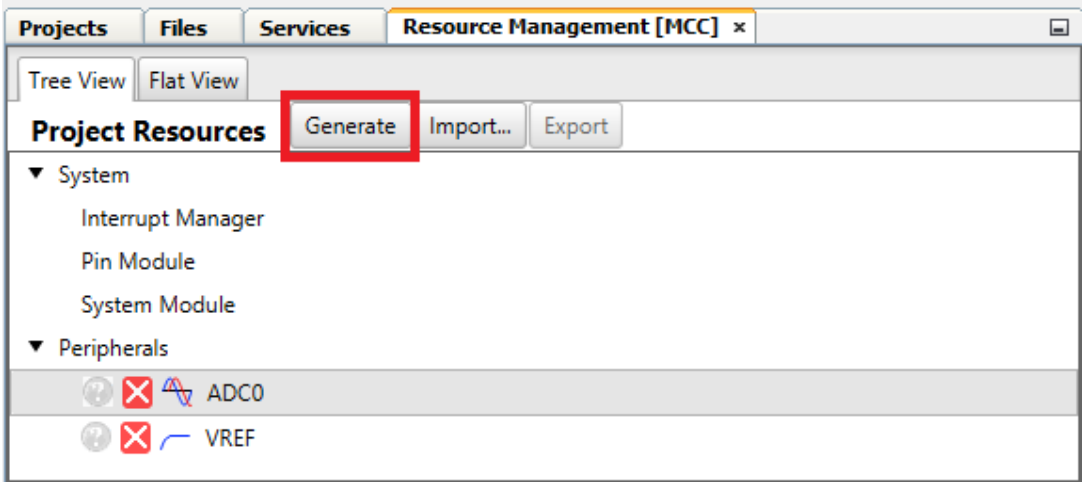
Info: ADC0 モジュールの[Interrupt Setting]で[Result Ready Interrupt Enable]にチェックを入れます。この設定は前のステップの画像にあります。[Interrupt Manager]でグローバル割り込みを有効にします。

図2-21.割り込みマネージャの設定



2.6. **[Generate]**ボタンを押します。

図2-22.[Generate]ボタン



3. プロジェクトにコードを追加します。

3.1. `main.c`ファイルに以下のコードを追加します。

```
#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()`は`mcc.c`で定義されます。`ADC0_StartConversion(channel)`と`ADC0_GetConversionResult()`は`adc0.c`で定義されます。

- `SYSTEM_Initialize()`はMCCの設定に従ってマイクロコントローラと周辺モジュールを設定する関数です。この関数はMCCにより生成され、プログラムの最初で呼び出される必要があります。
- `ADC0_StartConversion(channel)`は指定されたチャンネルでADC変換を開始します。
- `ADC0_GetConversionResult()`は最後の変換の結果を返します。
- プログラムは変換を開始した後、スリープ状態に入ります。結果レディ割り込みによって復帰させられ、結果を読み出した後、新しい変換を開始します。

3.2. [Source Files] > [MCC Generated Files] > [src]にある`pin_manager.c`ファイルで、ピンの方向に関するコードを以下に置き換えます。

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

Info: これは電力読み値に干渉するフローティングのピンを防ぐために行います。

3.3. ツールバーから**[Clean and Build]**ボタンを押して、プログラムがエラーなくビルドされる事を確認します。

図2-23.クリーンとビルド

GitHubでサンプルコードを見る
クリックしてリポジトリを閲覧

2.3 デバイスのプログラミングと電力効率の比較

To do: マイクロコントローラにプロジェクトを読み込み、電力読み値を取得します。

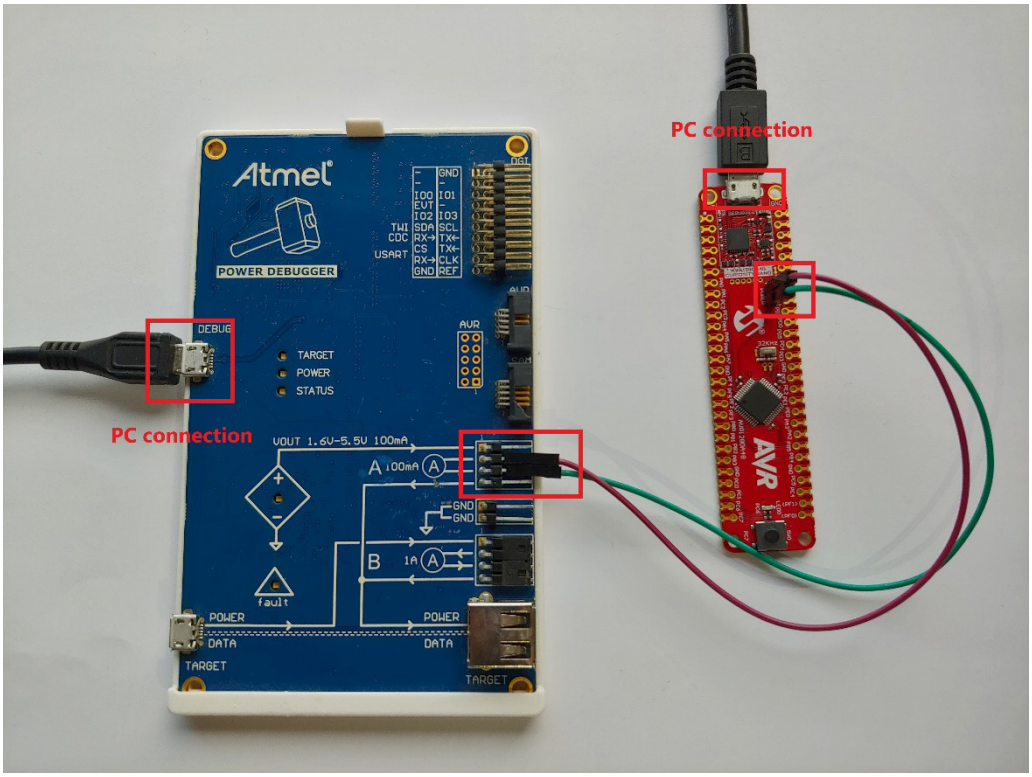
1. デバイスをプログラミングし、ソフトウェア実装プログラムから消費電力の計測値を取得します。

1.1. Curiosity Nanoボードをコンピュータに接続します。はんだ付けされた2本のピンを互いに接続します。ソフトウェア積算プログラムをメインプログラムに設定して、ツールバーから**[Make and Program Device]**ボタンを押します。プログラミングするツールを選択するためのポップアップが表示される場合があります。AVR128DA48 Curiosity Nanoボードを選択します。

図2-24.デバイスのmakeとプログラミング

1.2. Atmel Studioを開き、Power Debuggerをコンピュータに接続します。Curiosity NanoボードをPower Debuggerのチャンネル A電流計に接続します。Nano組み込みデバッガに近い方のピンをA電流計の内向きの矢印に接続し、マイクロコントローラに近い方のピンを電流計の外向きの矢印に接続します。

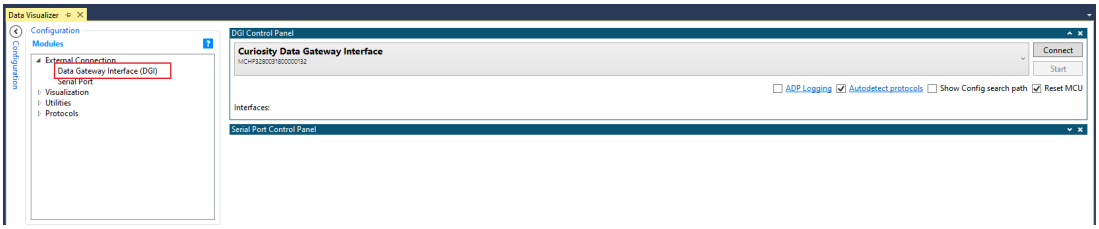
図2-25.ハードウェアのセットアップ



1.3. Atmel Studioで[Tools] > [Data Visualizer]を選択します。

1.4. 2つのボックスが表示されます。[DGI (Data Gateway Interface) Control Panel]と書いてある方をチェックします。表示されない場合、左側の[Configuration]メニューにアクセスします。[Modules] > [External Connection]セクションの[Data Gateway Interface]にあります。

図2-26.DGI

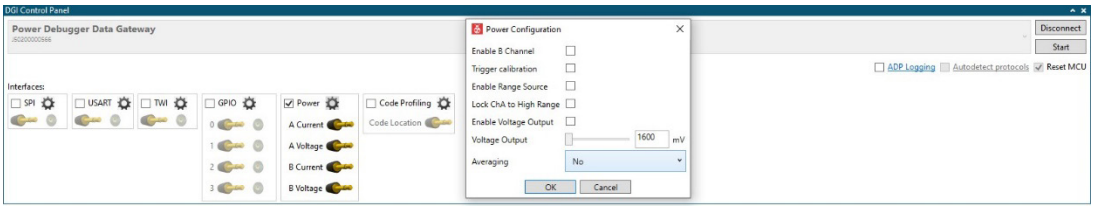


1.5. **[Connect]**をクリックすると、[Power]というボックスが表示されます。チェックを入れて歯車アイコンをクリックすると、メニューが開きます。Bチャンネルを無効にします。

⚠ WARNING	[Power]フィールドが表示されない場合、ボードを取り外して接続し直します。

図2-27.DGIの接続

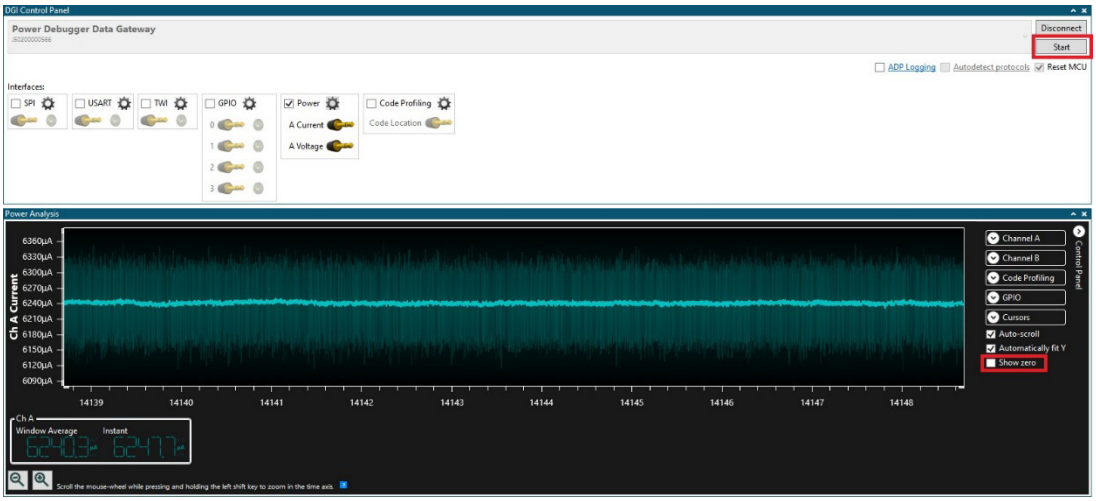
図2-28.[Power]の設定



1.6. 全ての設定が正しく完了したら**[Start]**をクリックすると、消費電力をリアルタイムで表示する新しいボックスが表示されます。

Tip: 表示を調整するにはボックスの左側で[Control Panel]を開きます。[Show zero]のチェックを外すと見やすくなります。

図2-29.監視の開始



1.7. 読み値が安定したら、**[Stop]**を押してから切断します。電力計測が表示されたボックスは開いたままになります。

2. ハードウェア積算実装プログラムから消費電力の計測値を取得します。

2.1. Curiosityボード上のはんだ付けされた2本のピンを互いに接続します。
2.2. ハードウェア積算プロジェクトをメインプログラムに設定して、MPLAB Xのツールバーから**[Make and Program Device]**ボタンをクリックします。

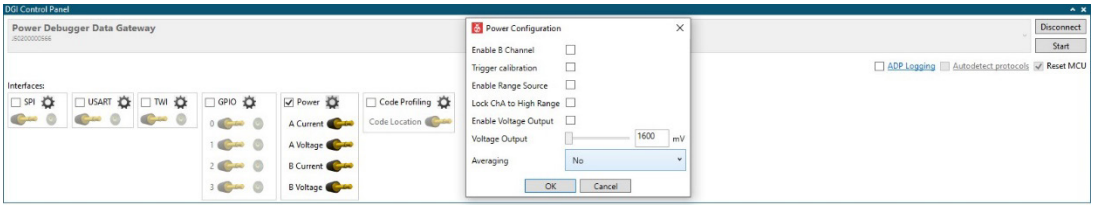
図2-30.デバイスのmakeとプログラミング

2.3. Curiosity Nanoボード上のはんだ付けされた2本のピンをPower Debuggerに接続します。ステップ1.2を参照してください。
2.4. Atmel StudioのData Visualizerに戻り、[DGI Control Panel]で**[Connect]**をクリックします。

図2-31.DGIの接続

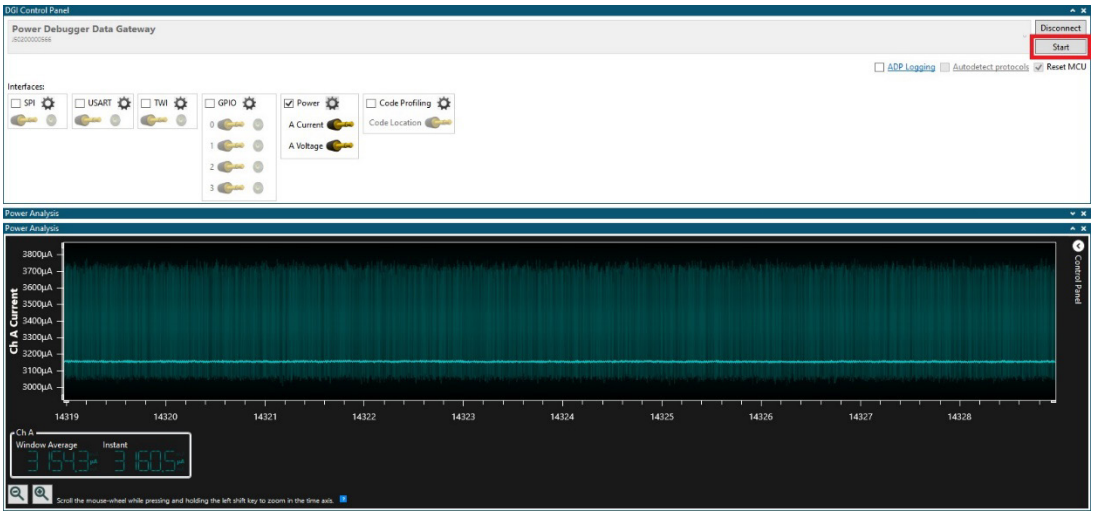
2.5. [Power]ボックスにチェックを入れ、設定を編集してBチャンネルを無効にします。

図2-32.[Power]の設定



2.6. **[Start]**をクリックすると、消費電力をリアルタイムで表示する新しいボックスが表示されます。[Show zero]を無効にすると見やすくなります。

図2-33.監視の開始

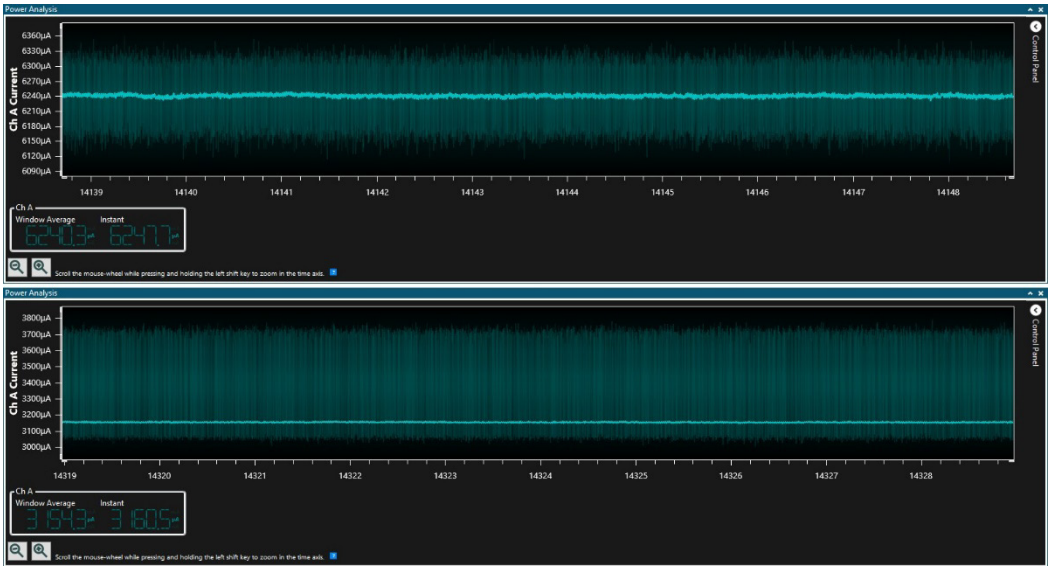


2.7. 読み値が安定したら、**[Stop]**を押して固定された結果を得ます。

3. 2つの計測値を比較します。

3.1. 両方の消費電力が同じ画面に表示されます。これらを解析する事でプログラムの挙動を観察できます。

図2-34.ソフトウェア積算とハードウェア積算の比較



Info: 上側がソフトウェア積算、下側がハードウェア積算を示しています。画像から分かるように、両者には 3.1 mA の差異があり、ハードウェア実装では効率がほぼ 2 倍です。

3. 改訂履歴

リビジョン	日付	説明
A	2020 年 8 月	初版

Microchip社のウェブサイト

Microchip社はウェブサイト(www.microchip.com)を通してオンライン サポートを提供しています。当ウェブサイトでは、お客様に役立つ情報やファイルを提供しています。以下を含む各種の情報をご覧になれます。

- **製品サポート** - データシートとエラッタ、アプリケーション ノートとサンプル プログラム、設計リソース、ユーザガイドとハードウェア サポート文書、最新のソフトウェアと過去のソフトウェア
- **技術サポート** - FAQ(よく寄せられる質問)、技術サポートのご依頼、オンライン ディスカッション グループ、Microchip社のデザイン パートナー プログラムおよびメンバーリスト
- **ご注文とお問い合わせ** - 製品セレクタと注文ガイド、最新プレスリリース、セミナー/イベントの一覧、お問い合わせ先(営業所/正規代理店)の一覧

お客様への通知サービス

Microchip社のお客様への通知サービスは、お客様にMicrochip社製品の最新情報をお届けする配信サービスです。ご興味のある製品ファミリまたは開発ツールに関する変更、更新、リビジョン、エラッタ情報をいち早くメールにてお知らせします。

http://www.microchip.com/pcnにアクセスし、登録手続きをしてください。

お客様サポート

Microchip社製品をお使いのお客様は、以下のチャンネルからサポートをご利用頂けます。

- 正規代理店
- 技術サポート

サポートは正規代理店にお問い合わせください。本書の最後のページに各国の営業所の一覧を記載しています。

技術サポートは以下のウェブページからもご利用頂けます。
www.microchip.com/support

Microchip社のデバイスコード保護機能

Microchip 社製品のコード保護機能について以下の点にご注意ください。

- Microchip社製品は、該当するMicrochip 社データシートに記載の仕様を満たしています。
- Microchip社では、通常の条件ならびに動作仕様書の仕様に従って使った場合、Microchip 社製品のセキュリティレベルは、現在市場に流通している同種製品の中でも最も高度であると考えています。
- Microchip社はその知的財産権を重視し、積極的に保護しています。Microchip 社製品のコード保護機能の侵害は固く禁じられており、デジタル ミレニアム著作権法に違反します。
- Microchip社を含む全ての半導体メーカーで、自社のコードのセキュリティを完全に保証できる企業はありません。コード保護機能とは、Microchip 社が製品を「解読不能」として保証するものではありません。コード保護機能は常に進化しています。Microchip 社では、常に製品のコード保護機能の改善に取り組んでいます。

法律上の注意点

本書および本書に記載されている情報は、Microchip 社製品を設計、テスト、お客様のアプリケーションと統合する目的を含め、Microchip 社製品に対してのみ使う事ができます。それ以外の方法でこの情報を使う事はこれらの条項

に違反します。デバイス アプリケーションの情報は、ユーザの便宜のためにのみ提供されるものであり、更新によって変更となる事があります。お客様のアプリケーションが仕様を満たす事を保証する責任は、お客様にあります。その他のサポートはMicrochip 社正規代理店にお問い合わせ頂くか、https://www.microchip.com/en-us/support/design-help/client-support-servicesをご覧ください。

Microchip 社は本書の情報を「現状のまま」で提供しています。Microchip 社は明示的、暗黙的、書面、口頭、法定のいずれであるかを問わず、本書に記載されている情報に関して、非侵害性、商品性、特定目的への適合性の暗黙的保証、または状態、品質、性能に関する保証をはじめとするいかなる類の表明も保証も行いません。

いかなる場合もMicrochip 社は、本情報またはその使用に関連する間接的、特殊的、懲罰的、偶発的または必然的損失、損害、費用、経費のいかんにかかわらず、またMicrochip 社がそのような損害が生じる可能性について報告を受けていた場合あるいは損害が予測可能であった場合でも、一切の責任を負いません。法律で認められる最大限の範囲を適用しようとも、本情報またはその使用に関連する一切の申し立てに対するMicrochip 社の責任限度額は、使用者が当該情報に関連してMicrochip 社に直接支払った額を超えません。

Microchip 社の明示的な書面による承認なしに、生命維持装置あるいは生命安全用途にMicrochip社の製品を使う事は全て購入者のリスクとし、また購入者はこれによって発生したあらゆる損害、クレーム、訴訟、費用に関して、Microchip 社は擁護され、免責され、損害をうけない事に同意するものとします。特に明記しない場合、暗黙的あるいは明示的を問わず、Microchip社が知的財産権を保有しているライセンスは一切譲渡されません。

商標

Microchip 社の名称とロゴ、Microchip ロゴ、Adaptec、AVR、AVRロゴ、AVR Freaks、BesTime、BitCloud、CryptoMemory、CryptoRF、dsPIC、flexPWR、HELDO、IGLOO、JukeBlox、KeeLoq、Kleer、LANCheck、LinkMD、maXStylus、maXTouch、MediaLB、megaAVR、Microsemi、Microsemi ロゴ、MOST、MOST ロゴ、MPLAB、OptoLyzer、PIC、picoPower、PICSTART、PIC32 ロゴ、PolarFire、Prochip Designer、QTouch、SAM-BA、SenGenuity、SpyNIC、SST、SST ロゴ、SuperFlash、Symmetricom、SyncServer、Tachyon、TimeSource、tinyAVR、UNI/O、Vectron、XMEGA は米国とその他の国におけるMicrochip Technology Incorporated の登録商標です。

AgileSwitch、APT、ClockWorks、The Embedded Control SolutionsCompany、EtherSynch、Flashtec、Hyper Speed Control、HyperLightLoad、Libero、motorBench、mTouch、Powermite 3、Precision Edge、ProASIC、ProASIC Plus、ProASIC Plus ロゴ、Quiet-Wire、SmartFusion、SyncWorld、Temux、TimeCesium、TimeHub、TimePictra、TimeProvider、TrueTime、ZL は米国におけるMicrochip Technology Incorporated の登録商標です。

Adjacent Key Suppression、AKS、Analog-for-the-Digital Age、Any Capacitor、AnyIn、AnyOut、Augmented Switching、BlueSky、BodyCom、Clockstudio、CodeGuard、CryptoAuthentication、CryptoAutomotive、CryptoCompanion、CryptoController、dsPICDEM、dsPICDEM.net、Dynamic Average Matching、DAM、ECAN、Espresso T1S、EtherGREEN、GridTime、IdealBridge、In-Circuit Serial Programming、ICSP、INICnet、Intelligent Paralleling、IntelliMOS、Inter-Chip Connectivity、JitterBlocker、Knob-on-Display、KoD、maxCrypto、maxView、memBrain、Mindi、MiWi、MPASM、MPF、MPLAB Certified ロゴ、MPLIB、MPLINK、MultiTRAK、NetDetach、Omniscient Code Generation、PICDEM、PICDEM.net、PICkit、PICtail、PowerSmart、PureSilicon、QMatrix、REAL ICE、RippleBlocker、RTAX、RTG4、SAM-ICE、Serial Quad I/O、simpleMAP、SimpliPHY、SmartBuffer、SmartHLS、SMART-I.S.、storClad、SQI、SuperSwitcher、SuperSwitcher II、Switchtec、SynchroPHY、TotalEndurance、Trusted Time、TSHARC、USBCheck、VariSense、VectorBlox、VeriPHY、ViewSpan、WiperLock、XpressConnect、ZENAは米国とその他の国におけるMicrochip Technology Incorporated の商標です。

SQTP は米国におけるMicrochip Technology Incorporated のサービスマークです。

Adaptec ロゴ、Frequency on Demand、Silicon Storage Technology、Symmcom はその他の国におけるMicrochip Technology Incorporatedの登録商標です。

GestIC は、その他の国における Microchip Technology Germany II GmbH & Co. KG (Microchip Technology Incorporated の子会社) の登録商標です。

ISBN: 978-1-6683-2433-2

品質管理システム

Microchip社の品質管理システムについてはwww.microchip.com/qualityをご覧ください。

各国の営業所とサービス

南北アメリカ	アジア/太平洋	アジア/太平洋	欧州
本社 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 技術サポート: http://www.microchip.com/support URL: www.microchip.com **アトランタ** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **オースティン、TX** Tel: 512-257-3370 **ボストン** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **シカゴ** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **ダラス** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **デトロイト** Novi, MI Tel: 248-848-4000 **ヒューストン、TX** Tel: 281-894-5983 **インディアナポリス** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **ロサンゼルス** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **ローリー、NC** Tel: 919-844-7510 **ニューヨーク、NY** Tel: 631-435-6000 **サンノゼ、CA** Tel: 408-735-9110 Tel: 408-436-4270 **カナダ - トロント** Tel: 905-695-1980 Fax: 905-695-2078	**オーストラリア - シドニー** Tel: 61-2-9868-6733 **中国 - 北京** Tel: 86-10 -8569-7000 **中国 - 成都** Tel: 86-28-8665-5511 **中国 - 重慶** Tel: 86-23-8980-9588 **中国 - 東莞** Tel: 86-769-8702-9880 **中国 - 広州** Tel: 86-20-8755-8029 **中国 - 杭州** Tel: 86-571-8792-8115 **中国 - 香港SAR** Tel: 852-2943-5100 **中国 - 南京** Tel: 86-25-8473-2460 **中国 - 青島** Tel: 86-532-8502-7355 **中国 - 上海** Tel: 86-21-3326-8000 **中国 - 瀋陽** Tel: 86-24-2334-2829 **中国 - 深圳** Tel: 86-755-8864-2200 **中国 - 蘇州** Tel: 86-186-6233-1526 **中国 - 武漢** Tel: 86-27-5980-5300 **中国 - 西安** Tel: 86-29-8833-7252 **中国 - 厦門** Tel: 86-592-2388138 **中国 - 珠海** Tel: 86-756-3210040	**インド - バンガロール** Tel: 91-80-3090-4444 **インド - ニューデリー** Tel: 91-11-4160-8631 **インド - プネ** Tel: 91-20-4121-0141 **日本 - 大阪** Tel: 81-6-6152-7160 **日本 - 東京** Tel: 81-3-6880-3770 **韓国 - 大邱** Tel: 82-53-744-4301 **韓国 - ソウル** Tel: 82-2-554-7200 **マレーシア - クアラルンプール** Tel: 60-3-7651-7906 **マレーシア - ペナン** Tel: 60-4-227-8870 **フィリピン - マニラ** Tel: 63-2-634-9065 **シンガポール** Tel: 65-6334-8870 **台湾 - 新竹** Tel: 886-3-577-8366 **台湾 - 高雄** Tel: 886-7-213-7830 **台湾 - 台北** Tel: 886-2-2508-8600 **タイ - バンコク** Tel: 66-2-694-1351 **ベトナム - ホーチミン** Tel: 84-28-5448-2100	**オーストリア - ヴェルス** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **デンマーク - コペンハーゲン** Tel: 45-4485-5910 Fax: 45-4485-2829 **フィンランド - エスポー** Tel: 358-9-4520-820 **フランス - パリ** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **ドイツ - ガーヒンク** Tel: 49-8931-9700 **ドイツ - ハーン** Tel: 49-2129-3766400 **ドイツ - ハイルブロン** Tel: 49-7131-72400 **ドイツ - カールスルーエ** Tel: 49-721-625370 **ドイツ - ミュンヘン** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **ドイツ - ローゼンハイム** Tel: 49-8031-354-560 **イスラエル - ラーナナ** Tel: 972-9-744-7705 **イタリア - ミラノ** Tel: 39-0331-742611 Fax: 39-0331-466781 **イタリア - パドヴァ** Tel: 39-049-7625286 **オランダ - ドリューネン** Tel: 31-416-690399 Fax: 31-416-690340 **ノルウェー - トロンハイム** Tel: 47-7288-4388 **ポーランド - ワルシャワ** Tel: 48-22-3325737 **ルーマニア - ブカレスト** Tel: 40-21-407-87-50 **スペイン - マドリッド** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **スウェーデン - ヨーテボリ** Tel: 46-31-704-60-40 **スウェーデン - ストックホルム** Tel: 46-8-5090-4654 **イギリス - ウォーキンガム** Tel: 44-118-921-5800 Fax: 44-118-921-5820