

# 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。



## 目次

前提条件	1
はじめに	1
本書で使うアイコン	2
<ol> <li>課題1:イベントシステムと割り込みの比較</li> <li>ハードウェアの説明</li> <li>1.1 ハードウェアの説明</li> <li>1.2 電力計測を正確に実行するためのCuriosityボードの改変</li> <li>1.3 使用モジュールの説明</li> <li>1.4 ソフトウェア実装プログラムの作成</li> <li>1.5 イベントシステム実装プロジェクトの作成</li> <li>1.6 デバイスのプログラミングと効率の比較</li> </ol>	
<ol> <li>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</li> <li>2.1 ソフトウェア実装プロジェクトの作成</li> <li>2.2 ハードウェア実装プロジェクトの作成</li> <li>2.3 デバイスのプログラミングと電力効率の比較</li> </ol>	28 28 34 41
3. 改訂履歴	46
Microchip社のウェブサイト	47
お客様への通知サービス	47
お客様サポート	47
Microchip社のデバイスコード保護機能	47
法律上の注意点	47
商標	48
品質管理システム	49
各国の営業所とサービス	

### 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 File Edit View Navigate



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

🔀 New Project		×
Steps	Choose Project	
1. Choose Project	Q Filter:	
	Categories: Proje	cts: Standalone Project Existing MPLAB IDE v8 Project Prebuilt (Hex, Loadable Image) Project User Makefile Project Library Project Import START MPLAB Project Import Atmel Studio Project
	Description: Creates a new standalone application project. It us project.	es an IDE-generated makefile to build your
	< Back Net	xt > Finish Cancel Help

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
----------	------

🔀 New Project			×
Steps	Select Device		
Choose Project     Select Device     Select Header     Select Header     Select Plugin Board     Select Compiler	Family:	All Families	
6. Select Project Name and Folder	Device:	AVR128DA48 V	
, older	Tool:	AVR 128DA48 Curiosity Nano-SN: MCH 🗸 🗌 Show All	
MPLAB X IDE			
		< Back Next > Finish Cancel Help	

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

🔀 New Project	×
Steps 1. Choose Project 2. Select Device 3. Select Header 4. Select Plugin Board 5. Select Compiler 6. Select Project Name and Folder	Select Compiler Compiler Toolchains 
MPLAB X IDE	
	< Back Next > Finish Cancel Help

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。

Steps	Select Project Name	and Folder
Choose Project     Select Device     Select Header     Select Plugin Board     Select Compiler     Select Project Name and     Folder	Project Name: Project Location: Project Folder:	avr-da-cnano-low-power-lab-interrupt         C:\Wy stuff\Projects         Browse         stuff\Projects\avr-da-cnano-low-power-lab-interrupt.X
ADLAD	<ul> <li>Overwrite existin</li> <li>Also delete source</li> <li>Set as main proje</li> <li>Use project locate</li> </ul>	g project. ies. ect ion as the project folder
XIDE	Encoding: ISO	D-8859-1

#### 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

System Module	
錜 Easy Setup 📃 Registers	
Clock Control	
Main Clock(Hz):	24000000
Olock Source :	Internal Oscillator
Internal Oscillator Frequency:	1-32MHz internal oscillator
Oscillator Frequency Options:	24 MHz system clock
PLL Enable:	
External Clock(Hz):	1 ≤ 1000000 ≤ 20000000
Prescaler Enable:	
Prescaler:	6X *
Olock Out Enable:	
<ul> <li>Watchdog Timer</li> </ul>	
Brown-out Detector	
<ul> <li>Voltage Level Monitor</li> </ul>	

- 2.2 [System Module]の[Registers]タブで[SEN]を[enabled]にし、[SMODE]を[Standby Mode]にします。
  - Info: [System CTRLA レジ

**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

System Module
🔅 Easy Setup 📃 Registers
WRST disabled ▼
▼ SLPCTRL
Register: SLPCTRLVREGCTRL     0x0       PMODE     AUTO

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。

ij

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。

i

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

VREF	
🔯 Easy Setup 📄 Registers	
<ul> <li>Software Settings</li> </ul>	
API Prefix:     VREF_0	
<ul> <li>Hardware Settings</li> </ul>	
Enable Force ADC Voltage     Reference:	
ADC Voltage Reference:     Internal 2.048V reference	3
Enable Force DAC/AC Voltage     Reference:	
DAC/AC Voltage Reference:     Internal 1.024V reference	3
Enable Force AC Voltage     Reference:	
AC Voltage Reference:     Internal 1.024V reference	3

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

ADC0	
🍪 Easy Setup 📃 Registers	
<ul> <li>Software Settings</li> </ul>	
API Prefix:	ADC0
Result Selection :	12-bit mode
② Differential Mode Conversion :	disabled
② Left Adjust Result :	
<ul> <li>Hardware Settings</li> </ul>	
② Enable ADC:	$\checkmark$
② Sampling Frequency(Hz):	272727 ≤ <b>857142</b> ≤ 923076
ADC Clock(Hz):	12000000
③ Sample Accumulation Number:	No accumulation
Sample Length (# of ADC Clock) :	0 s 1 s 31

#### 図1-12.RUNSTBYに関するADCの設定

ADC0	
	~
SPCONV disabled -	
STCONV disabled -	
▼ Register: CTRLA <sub>0x81</sub>	
CONVMODE disabled -	
INABLE enabled -	
FREERUN     disabled	
Reftadj disabled •	
RESSEL 12-bit mode 💌	
RUNSTBY enabled -	
▼ Register: CTRLB Ox0	
SAMPNUM No accumulation	

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。



#### 図1-13.RTCの設定

RTC	
🔅 Easy Setup 📃 Registers	
<ul> <li>Software Settings</li> </ul>	^ 
API Prefix:	RTC
<ul> <li>Hardware Settings</li> </ul>	
② Enable RTC:	
RTC Clock(Hz):	32768
	Internal 32.768 kHz oscillator
External Clock(Hz):	1 ≤ 32000 ≤ 32000
Prescaling Factor:	RTC Clock / 1
Ompare:	1 s ≤ 0 s ≤ 2 s
Actual Compare:	0 s
Period:	1s ≤ 1s ≤ 2s
Actual Period:	1 s
Periodic Interrupt Timer	
<ul> <li>Interrupt Settings</li> </ul>	
② Compare Match Interrupt Enable:	
Overflow Interrupt Enable:	

#### 図1-14.RUNSTBYに関するRTCの設定

RTC	)	J
Construction     Construction <ul> <li>Register: CMP</li> <li>Ox0</li> <li>Ox0</li></ul>	^	
▼ Register: CNT 0x0		
CORREN disabled •	ſ	
PRESCALER RTC Clock / 1		
RUNSTDBY     enabled		
▼ Register: DBGCTRL <sub>0x0</sub>	L	J
DBGRUN disabled		

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

Interrupt Manager		<b>?</b>
🔯 Easy Setup 📃 Registers		
<ul> <li>Interrupt Setting</li> </ul>		î
Global Interrupt Enable:	$\checkmark$	
Interrupt Priority		
Round-robin Scheduling Enable:		
Interrupt Level Priority:	0	
Interrupt Vector with High Priority:	0	

2.7. [Generate]ボタンを押します。

#### 図1-16.[Generate]ボタン

Projects	Files	Services	Resource Management [MCC] ×	
Tree View	Flat View		_	
Project I	Resource	es Generat	e Import Export	
Interr	unt Manao	lor.		^
intern	upt Manag	jei		n
Pin M	odule			
Syster	n Module			
<ul> <li>Peripher</li> </ul>	als			
💮 🔀	🖌 🕀 🕈	0		
🛞 🔀	🖥 📸 RTC	:		
@ 🔀	🖌 — VRE	F		Ų

- 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。

<pre>#include <avr io.h=""> #include <avr sleep.h=""> #include "mcc_generated_files/mcc.h"</avr></avr></pre>
<pre>void TIMER_interrupt(void);</pre>
<pre>uint16_t result;</pre>
int main(void)
<pre>SYSTEM_Initialize(); RTC_SetOVFIsrCallback(TIMER_interrupt);</pre>
<pre>while (1) {     sleep_cpu();     result = ADC0_GetConversionResult(); }</pre>
}
<pre>void TIMER_interrupt() {</pre>
ADC0_StartConversion(ADC_MUXPOS_TEMPSENSE_gc); }

	i	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。
		<ul> <li>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。</li> </ul>
		<ul> <li>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。</li> </ul>
		• TIMER_interrupt() <b>関数は ADC の変換を開始します</b> 。
		<ul> <li>ADC0_StartConversion() 関数は ADC の温度チャンネルの変換を開始します。</li> </ul>
		- • ADC0_GetConversionResult()は直前の変換結果を返します。
3.2.	<u>[Source Files]</u> コードを以下	
	PORTA.DIR PORTE.DIR PORTC.DIR PORTE.DIR PORTE.DIR PORTF.DIR	= 0xFF; = 0xFF; = 0x3F; = 0xFF; = 0xFF; = 0xFF;
	i	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. "	ールバーから	
	図1-17.クリー	-ンとビルド
	T W	▼ >
		SitHubでサンプルコードを見る フ 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

File Edit View Navigate



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	イプ
-------	----	-----	------	----

🔀 New Project	×
Steps	Choose Project
1. Choose Project 2	Q Filter:
	Categories:       Projects:         Image: Microchip Embedded       Standalone Project         Image: Other Embedded       Existing MPLAB IDE v8 Project         Image: Samples       Prebuilt (Hex, Loadable Image) Project         Image: I
	Description:
	project.
	< Back Next > Finish Cancel Help

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

🔀 New Project		×
Steps	Select Device	2
1. Choose Project 2. Select Device		
Select Header     Select Plugin Board     Select Compiler	Family:	All Families V
6. Select Project Name and Folder	Device:	AVR128DA48 V
	Tool:	AVR 128DA48 Curiosity Nano-SN: MCH 🗸 🔲 Show All
MPLAB		
		<back next=""> Finish Cancel Help</back>

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

New Project		>
Steps	Select Compiler	
<ol> <li>Choose Project</li> <li>Select Device</li> <li>Select Header</li> <li>Select Plugin Board</li> <li>Select Compiler</li> <li>Select Project Name and Folder</li> </ol>	Compiler Toolchains -XC8 -XC8 (v2.20) [C: Program Files (v86) [Vicrochip \vc8 \v2.20 \bin] -XC8 (v2.05) [C: Program Files (v86) [Vicrochip \vc8 \v2.05 \bin] -XC8 (v2.05) [C: Program Files (v86) [Vicrochip \vc8 \v2.05 \bin] -XC8 (v2.05) [C: Program Files (v86) [Vicrochip \vc8 \v2.05 \bin] -XC8 (v2.05) [C: Program Files (v86) [Vicrochip \vc8 \v2.05 \bin] -XC8 (v2.05) [C: Program Files (v86) [Vicrochip \vc8 \v2.05 \bin] -XC8 (v2.05) [C: Program Files (v86) [Vicrochip \vc8 \v2.05 \bin] -XC8 (v2.05) [C: Program Files (v86) [Vicrochip \vc8 \v2.05 \bin] -XC8 (v2.05) [C: Program Files (v86) [Vicrochip \vc8 \v2.05 \bin] -XC8 (v2.05) [C: Program Files (v86) [Vicrochip \vc8 \v2.05 \bin] -XC8 (v2.05) [C: Program Files (v86) [Vicrochip \vc8 \v2.05 \bin] -XC8 (v2.05) [C: Program Files (v86) [Vicrochip \vc8 \v2.05 \bin] -XC8 (v2.05) [C: Program Files (v86) [Vicrochip \vc8 \v2.05 \bin] -XC8 (v2.05) [C: Program Files (v86) [Vicrochip \vc8 \v2.05 \bin] -XC8 (v2.05) [C: Program Files (v86) [Vicrochip \vc8 \v2.05 \bin] -XC8 (v2.05) [C: Program Files (v86) [Vicrochip \vc8 \v2.05 \bin] -XC8 (v2.05) [C: Program Files (v86) [Vicrochip \vc8 \v2.05 \bin] -XC8 (v2.05) [C: Program Files (v86) [Vicrochip \vc8 \v2.05 \bin] -XC8 (v2.05) [C: Program Files (v86) [Vicrochip \vc8 \v2.05 \bin] -XC8 (v2.05) [C: Program Files (v86) [Vicrochip \vc8 \v2.05 \bin] -XC8 (v2.05) [C: Program Files (v86) [Vicrochip \v2.05 \bin] -XC8 (v2.05) [C: Program Files (v86) [Vicrochip \v2.05 \bin] -XC8 (v2.05) [C: Program Files (v86) [Vicrochip \v2.05 \v2.05] -XC8 (v2.05) [Vicrochip \v2.05] [Vicrochip \v2.05 [Vicrochip \v2.05] [Vicrochip \v2.05] -XC8 (v2.05) [Vicrochip \v2.05] [Vicrochi	

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

🔀 New Project			×
Steps	Select Project Name	and Folder	
<ol> <li>Choose Project</li> <li>Select Device</li> <li>Select Header</li> <li>Select Plugin Board</li> <li>Select Compiler</li> <li>Select Compiler</li> <li>Select Project Name and Folder</li> </ol>	Project Name: Project Location: Project Folder:	avr-da-cnano-low-power-lab-evsys       C:\My stuff\Projects       My stuff\Projects\avr-da-cnano-low-power-lab-evsys.X	
MPLAB	Overwrite existing Also delete source Set as main proje Use project location	g project. es. ct on as the project folder	
XIDE	Encoding: ISC	-8859-1 🗸	
		< Back Next > Finish Cancel He	lp

- 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

System Module			
😳 Easy Setup 📃 Registers			
Clock Control			
Main Clock(Hz):	2400000		
Clock Source :	Internal Oscillator		
Internal Oscillator Frequency:	1-32MHz internal oscillator 🔹		
Oscillator Frequency Options:	24 MHz system clock 🔹		
PLL Enable:			
External Clock(Hz):	1 ≤ 1000000 ≤ 20000000		
Prescaler Enable:			
Prescaler:	6X -		
Clock Out Enable:			
<ul> <li>Watchdog Timer</li> </ul>			
Brown-out Detector			
Voltage Level Monitor			

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

System Module	2
Image: Swrstein Stress       Image: Swrstein Stress	^
▼ SLPCTRL	
Register: SLPCTRLCTRLA     0x3       SEN     enabled       SMODE     Standby Mode	

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

· — · —	
VREF	<b>e</b>
🍪 Easy Setup 📃 Registers	
<ul> <li>Software Settings</li> </ul>	
API Prefix:	VREF_0
<ul> <li>Hardware Settings</li> </ul>	
Enable Force ADC Voltage Reference:	
ADC Voltage Reference:	Internal 2.048V reference
Enable Force DAC/AC Voltage Reference:	
OAC/AC Voltage Reference:	Internal 1.024V reference
② Enable Force AC Voltage Reference:	
AC Voltage Reference:	Internal 1.024V reference
1	

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

ADC0		2
🍪 Easy Setup 📃 Registers		
<ul> <li>Software Settings</li> </ul>		Î
API Prefix:	ADC0	
Result Selection :	12-bit mode 🔹	
Oifferential Mode Conversion :	disabled	
② Left Adjust Result :		
<ul> <li>Hardware Settings</li> </ul>		
Enable ADC:		
Sampling Frequency(Hz):	272727 ≤ 857142 ≤ 923076	
ADC Clock(Hz):	12000000	
Sample Accumulation Number:	No accumulation	
Sample Length (# of ADC Clock) :	0 ≤ 1 ≤ 31	

#### 図1-27.RUNSTBYに関するRTCの設定

ADC0	
Easy Setup Registers     Register: COMMAND 0x0	^
SPCONV disabled	
STCONV disabled *	
▼ Register: CTRLA 0x81	
CONVMODE disabled	
ENABLE     enabled     r	
FREERUN disabled	
LEFTADJ disabled	
RESSEL 12-bit mode	
RUNSTBY     enabled	
▼ Register: CTRLB Ox0	
SAMPNUM No accumulation	

#### 図1-28.イベント受信時変換開始に関するADCの設定

ADCO
Easy Setup Registers
▼ Register: CTRLD 0x0
INITDLY Delay 0 CLK_ADC cycles
SAMPDLY 0x0
▼ Register: CTRLE 0x0
WINCM No Window Comparison
▼ Register: DBGCTRL 0x0
OBGRUN disabled -
▼ Register: EVCTRL <sub>0×1</sub>
STARTEI enabled -
▼ Register: INTCTRL 0x0
WCMP disabled -

 [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

RTC	2
💮 Easy Setup 📃 Registers	
<ul> <li>Software Settings</li> </ul>	^ 
API Prefix:	RTC
<ul> <li>Hardware Settings</li> </ul>	
Parallel RTC:	
RTC Clock(Hz):	32768
RTC Clock Source Selection:	Internal 32.768 kHz oscillator
② External Clock(Hz):	1 ≤ 32000 ≤ 32000
Prescaling Factor:	RTC Clock / 1
Ompare:	1 s ≤ 0 s ≤ 2 s
Actual Compare:	0 s
Period:	1 s ≤ 1 s ≤ 2 s
Actual Period:	1 s
Periodic Interrupt Timer	
▼ Interrupt Settings	
Ocmpare Match Interrupt Enable:	
Overflow Interrupt Enable:	

#### 図1-30.RUNSTBYに関するRTCの設定

RTC	
Easy Setup     ⊟ Registers       ▼ Register: CMP     0x0	^
▼ Register: CTRLA 0x81	
CORREN disabled -	$\square$
PRESCALER RTC Clock / 1	
WRUNSTDBY     enabled	
▼ Register: DBGCTRL Ox0	
DBGRUN disabled	

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ADCOSTART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

EVSYS							<b>?</b>
🛞 Easy Setup 📃 Registers							
<ul> <li>Software Settings</li> </ul>							
API Prefix:	EVSYS						
<ul> <li>Event System Settings</li> </ul>							
Event Generator	Channels						
		ADC0START	CCLLUT0A	CCLLUT0B	CCLLUT1A	CCLLUT1B	CCLLUT2A
RTC_OVF *	CHANNEL0	$\checkmark$					
ADC0_RESRDY -	CHANNEL1						
OFF -	CHANNEL2						
OFF -	CHANNEL3						
OFF •	CHANNEL4						
OFF •	CHANNEL5						
OFF •	CHANNEL6						
OFF -	CHANNEL7						
OFF •	CHANNEL8						
OFF •	CHANNEL9						
<							>

#### 図1-32.EVSYSの設定

EVSYS								<b>?</b>	
🛞 Easy Setup 📃 Registers									
<ul> <li>Software Sett</li> </ul>	▼ Software Settings								
API Prefix:		EVSYS							
<ul> <li>Event System</li> </ul>	n Settings								
CCLLUT5A	CCLLUT5B	EVSYSEVOUTA	EVSYSEVOUTB	EVSYSEVOUTC	EVSYSEVOUTD	EVSYSEVOUTE	EVSYSEVOUTF	EVSYSEVOU	
				$\checkmark$					
<								>	

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

Pin Module									
Selected Package : C	Gessy Setup Registers Selected Package: QFN48								
Pin Name 🔺	Module	Function	Custom Name	OUTPUT	START HIGH	INVEN	PULLUPEN	ISC	
PC2	EVSYS	EVOUTC		$\checkmark$				Sense Both Edges	-

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

Interrupt Manager		0
😳 Easy Setup 📃 Registers		
<ul> <li>Interrupt Setting</li> </ul>		
Global Interrupt Enable:	$\checkmark$	
<ul> <li>Interrupt Priority</li> </ul>		
Round-robin Scheduling Enable:		
Interrupt Level Priority:	0	
Interrupt Vector with High Priority:	0	

2.10. [Generate]をクリックします。

図1-36	[Generate	]ボタン
-------	-----------	------

Projects	Files	Services	Resource Management [MCC] ×	
Tree View	Flat View			
Project F	Resource	es Generat	e Import Export	
▼ System				Â
Interru	upt Manag	er		
Pin M	odule			
Syster	n Module			
<ul> <li>Peripher</li> </ul>	als			
💮 🔀	🖌 🕀 🕈	0		U
💮 🔀	EVS	YS		
<u> </u>	2 🚔 ртс			~

- 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();
}
```

	<ul> <li>Info: SYSTEM_Initialize()はmcc.cで定義されます。</li> <li>ADC0_GetConverisonResult()はadc.cで定義されます。</li> <li>PORTC_PC2_SetInterruptHandler()はpin_manager.cで定義されます。</li> <li>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。</li> </ul>
	<ul> <li>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。</li> </ul>
	<ul> <li>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。</li> </ul>
	• ADC0_GetConversionResult()は変換の最後の結果を返します。
3.2.	
	<pre>PORTA.DIR = 0xFF; PORTB.DIR = 0xFF; PORTC.DIR = 0x3F; PORTD.DIR = 0xFF; PORTE.DIR = 0xFF; PORTF.DIR = 0xFF;</pre>
	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.	ッールバーから[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 <sup>クリックしてリボジトリを閲覧</sup>

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。
  - 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。
- 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

Data	Visualizer 🕫 🗙	•
٢	Configuration	DGi Control Panel
Configuration	Modules 2      External Connection Data Gateway Interface (DGI) Senal Port Visualization      Utilities      Protocols	Connect         Connect           Start         Start           ADP Logging of Autodetect protocols         Show Config search public VIII           Interfaces         Networks
		Senal Port Control Panel 🔹 🗙

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

DGI Control Panel	^ ×
Power Debugger Data Gateway	Connect
100000566	Start
AD2Logging Autodetect perdor	ols 🔽 Reset MCU
Interfaces:	

#### 図1-42.[Power]の設定

DGI Control Panel				* X
Power Debugger Data Gateway		Power Configuration	×	Uisconnect Start
Interfaces:	Power X     Code Profiling X     Code Code Location      Code     Code Location     Code     Code Location     Code     Code Location	Trigger calibration Enable Range Source Lock ChA to High Range Enable Voltage Cutput Voltage Output Averaging No OK Cance	1600 mV ¥	🗌 <u>ADP Logging</u> 🔛 <u>Autodetect protocols</u> 😿 Reset MCU

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

Control Panel	×
Ver Debugger Data Gateway	ct
□ ADP Legging □ Autodatert antocols 🖉 Rest M	си
faces	
SPI 🗘 🗌 USART 🔅 🗌 TMI 🔅 🗍 GPIO 🏠 🧭 Power 🔅 🗋 Code Profiling 🔅	
A Current Care Code Location Care	
1 🌑 A Voltage 🚭	
2 🐲	
3 🚭 🖉	
r Analysis	×
400µ - 280µ - 280µ - 280µ - 160µ - 40µ - - 20µ - - 20µ - - 20µ - - 20µ - - 20µ - - - - - - - - - - - - - -	Control Panel
3907/5 3908.0 3908.5 3909.0 3909.5 3910.0 3910.5 3911.0 3911.5 3912.0 3912.5 3913.0 3913.5 3914.0 3914.5 3915.0 3915.5 3916.0 3916.5 3917.0 Indow Average Instant	

- 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

DGi Control Panel	1	×
Power Debugger Data Gateway	Connec	t
3566000086	Start	
ADP Logging      Autodetect protocols	🖌 Reset N	ICU
Interfaces		

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

DGI Control Panel				* X
Power Debugger Data Gateway		Power Configuration     Enable B Channel	×	Disconnect Start
Interface: SPI ☆ USANT ☆ TWI ☆ GPID ☆ GPID ☆ 1 GPID ☆ 2 GPID ☆ 3 GPID ☆ 3 GPID ☆ 3 GPID ☆	Power      Code Profiling      Code Profiling      Code Location      Code Location	Trigger calibration  Fnoble Range Source Look Chi to High Range E Foshel Voltage Cutput Voltage Cutput Averaging No OK Cancel	1600 mV v	🗌 ÁÐP. Lagging 🔛 Ástadetest protocols 🤿 Reet MCU

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

DGI Control Pa	inel											^ X
Power Deb	bugger Data Ga	teway										Disconnect Start
										ADP Log	ing Autodetect protocols 🗸	Reset MCU
Interfaces:	🗆 USART 🙀	🗆 TWI 🛱	🗆 GPIO 🔅	✓ Power 🙀	Code Profiling 🙀							
<b>@~~</b> 0	<b>@~~</b> ()	<b>@~~</b> 0	0 🐠 🔕	A Current Co	ode Location 🌑							
			1 🌑 💿	A Voltage 🌑								
			2 🌑 💿									
			3 🌑 🖉									
Power Analysis	;											^ X
1200μA - 1050μA - 900μA -											Channel A	> Contro
¥ 750μA ·											Code Profili	ing Pane
600μA -												
Υ 300μΑ											Cursors	
ο 150μA -											✓ Automatical	ly fit Y
0μΑ -							1				Show zero	
		3725	3726	3727	3728	3729	3730	3731	3732	3733	3734	
Ch A	erage Instant	22.4¥										
ର୍ ଭ୍	Scroll the mouse-wheel	vhile pressing and hole	ling the left shift key to zo	om in the time axis. 🗵								

- 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

File Edit View Navigate



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)。

🔀 New Project		Х
Steps	Choose Project	
1. Choose Project	Q Filter:	
	Categories: Microchip Embedded Other Embedded Samples	Projects: Standalone Project Existing MPLAB IDE v8 Project Prebuilt (Hex, Loadable Image) Project User Makefile Project Library Project Import START MPLAB Project Import Atmel Studio Project
	Description:	
	Creates a new standalone application proje project.	ct. It uses an IDE-generated makefile to build your
	< Back	Next > Finish Cancel Help

#### 図2-2.プロジェクトのタイプ

 [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。

a New Project			>
Steps	Select Device	1	
Choose Project     Select Device     Select Header     Select Plugin Board     Select Compiler	Family:	All Families	
<ol> <li>Select Project Name and Folder</li> </ol>	Tool:	AVR128DA48 Curiosity Nano-SN: MCH V	
MPLAB X IDE			

- 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

×	New Project		Х
Ste	ps	Select Compiler	
1. 2. 3. 4. <b>5.</b> 6.	Choose Project Select Device Select Header Select Plugin Board Select Compiler Select Project Name and Folder	Compiler Toolchains -XC8 -XC8 -XC8 (v2.20) [C:\Program Files\/Microchip\xc8\v2.20\pin] -XC8 (v2.10) [C:\Program Files (x86)\/Microchip\xc8\v2.10\pin] -XC8 (v2.05) [C:\Program Files (x86)\/Microchip\xc8\v2.05\pin] -XC8 (v2.05) [C:\Program Files (x86	
	XIDE		
		< Back Next > Finish Cancel Help	

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

Steps	Select Proje	ct Name and Fold	er	
<ol> <li>Choose Project</li> <li>Select Device</li> <li>Select Header</li> <li>Select Plugin Boa</li> <li>Select Compiler</li> <li>Select Project Folder</li> </ol>	Project Nam ard Project Loca Name and Project Fold	e: avr- ntion: C:W er: stuff	da-cnano-low-power-lab-interrupt 1y stuff\Projects ?\Projects\avr-da-cnano-low-power-lab-interr	Browse
MPLA	□ Overwr □ Also de ☑ Set as r □ Use pro	te existing project. lete sources. nain project ject location as the p	project folder	
	Encoding:	ISO-8859-1 me and folder path l ning the project nam	ength are nearing the Windows limit. This near or path.	may cause issues during build or

- 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。



#### 図2-6.システム モジュールの設定

System Module					
錄 Easy Setup 📃 Registers					
Clock Control					
Main Clock(Hz):	24000000				
Clock Source :	Internal Oscillator 🗸				
Internal Oscillator Frequency:	1-32MHz internal oscillator 🗸				
Oscillator Frequency Options:	24 MHz system clock 🔹				
PLL Enable:					
External Clock(Hz):	1 ≤ 1000000 ≤ 20000000				
Prescaler Enable:					
Prescaler:	6X 👻				
Clock Out Enable:					
► Watchdog Timer					
► Brown-out Detector					
► Voltage Level Monitor					

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。

	T.		

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

/REF					
🕃 Easy Setup 📃 Registers					
Software Settings					
API Prefix:	VREF_0				
<ul> <li>Hardware Settings</li> </ul>					
Enable Force ADC Voltage Reference:					
ADC Voltage Reference:	Internal 2.048V reference 💌				
Enable Force DAC/AC Voltage Reference:					
OAC/AC Voltage Reference:	Internal 1.024V reference				
Enable Force AC Voltage Reference:					
AC Voltage Reference:	Internal 1.024V reference 🔹				

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。

F

#### 図2-8.ADCの設定

ADCO					
🔅 Easy Setup 📃 Registers					
<ul> <li>Software Settings</li> </ul>					
API Prefix:	ADC0				
Result Selection :	12-bit mode				
② Differential Mode Conversion :	disabled				
😗 Left Adjust Result :					
<ul> <li>Hardware Settings</li> </ul>					
② Enable ADC:	$\checkmark$				
③ Sampling Frequency(Hz):	272727 ≤ 272727 ≤ 923076				
ADC Clock(Hz):	12000000				
Sample Accumulation Number:	No accumulation				
Sample Length (# of ADC Clock) :	0 ≤ 31 ≤ 31				
Interrupt Settings					
► Select Channels					
▶ Window Settings					

2.4 [Generate]ボタンをクリックします。

### 図2-9.[Generate]ボタン

Projects	Files	Services	Resource Management [MCC]	×	
Tree View	Flat View				
Project F	Resources	Generate	Import Export		
▼ System			•		
Interrupt Manager		r			
Pin Module					
System	n Module				
▼ Peripher	▼ Peripherals				
🛞 🔀 🕀 ADCO					
@ 🔀	🔶 — VREF				

- 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++)</pre>
         {
              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。



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

File Edit View Navigate



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)。

🗙 New Project		
Steps	Choose Project	
1. Choose Project	Q Filter:	
	Categories: Microchip Embedded Other Embedded	Projects: Standalone Project Existing MPLAB IDE v8 Project Prebuilt (Hex, Loadable Image) Project User Makefile Project Library Project Import START MPLAB Project Import Atmel Studio Project
	Description: Creates a new standalone application project.	on project. It uses an IDE-generated makefile to build your
	<	Back Next > Finish Cancel Help

 [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

🔀 New Project		×
Steps	Select Device	e
Choose Project     Select Device     Select Header     Select Plugin Board	Family:	All Families
<ol> <li>Select Compiler</li> <li>Select Project Name and Folder</li> </ol>	Device:	AVR 128DA48 🗸
	Tool:	AVR 128DA48 Curiosity Nano-SN: MCH 🗸 🗌 Show All
MPLAB		
		< Back Next > Finish Cancel Help

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

🔀 New Project		×
<ul> <li>Steps</li> <li>Choose Project</li> <li>Select Device</li> <li>Select Header</li> <li>Select Plugin Board</li> <li>Select Compiler</li> <li>Select Project Name and Folder</li> </ul>	Select Compiler Compiler Toolchains SAC8 Compiler Toolchains Compiler Toolchains Compile	
	< Back Next > Finish Cancel Help	p

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

iteps	Select Project Name	e and Folder	
. Choose Project . Select Device . Select Header	Project Name:	avr-da-cnano-low-power-lab-interrupt	
Select Plugin Board Select Compiler	Project Location:	C: \My stuff\Projects	Browse
<ol> <li>Select Project Name and Folder</li> </ol>	Project Folder:	stuff\Projects\avr-da-cnano-low-power-lab-interrupt.X	
	Overwrite existin	ng project.	
	Also delete sour	ces.	
	Set as main proje	ect	
	Use project locat	tion as the project folder	
	_		
	Encoding: IS	0-8859-1 🗸	
~	Project same and f	alder path length are possing the Windows limit. This may say	use issues during build a
	Try shortening the	project name or path.	use issues during build (

- 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。



#### 図2-16.システム モジュールの設定

System Module					
錄 Easy Setup 📃 Registers					
Clock Control					
Main Clock(Hz):	24000000				
Clock Source :	Internal Oscillator 🗸				
Internal Oscillator Frequency:	1-32MHz internal oscillator 🗸				
Oscillator Frequency Options:	24 MHz system clock 🔹				
PLL Enable:					
External Clock(Hz):	1 ≤ 1000000 ≤ 20000000				
Prescaler Enable:					
Prescaler:	6X 👻				
Clock Out Enable:					
► Watchdog Timer					
► Brown-out Detector					
► Voltage Level Monitor					

#### 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。

F	-		
		×	

**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

System Module	
Easy Setup Registers	
	^
▼ SLPCTRL	
▼ Register: SLPCTRL.CTRLA 0x3	
SEN enabled	
SMODE Standby Mode +	
▼ Register: SLPCTRL.VREGCTRL 0x0	
PMODE AUTO -	

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

VREF		
🍪 Easy Setup 📄 Registers		
<ul> <li>Software Settings</li> </ul>		
API Prefix:	VREF_0	
<ul> <li>Hardware Settings</li> </ul>		
Enable Force ADC Voltage Reference:		
ADC Voltage Reference:	Internal 2.048V reference 🔹	
Enable Force DAC/AC Voltage Reference:		
OAC/AC Voltage Reference:	Internal 1.024V reference 🔹	
Enable Force AC Voltage Reference:		
AC Voltage Reference:	Internal 1.024V reference +	

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。

U	

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

ADC0	
😳 Easy Setup 📃 Registers	
<ul> <li>Software Settings</li> </ul>	
API Prefix:	ADC0
Result Selection :	12-bit mode
Oifferential Mode Conversion :	disabled
(2) Left Adjust Result :	
<ul> <li>Hardware Settings</li> </ul>	
② Enable ADC:	$\checkmark$
③ Sampling Frequency(Hz):	272727 ≤ 272727 ≤ 923076
ADC Clock(Hz):	12000000
Sample Accumulation Number:	128 results accumulated
Sample Length (# of ADC Clock) :	0 ≤ 31 ≤ 31
<ul> <li>Interrupt Settings</li> </ul>	
Result Ready Interrupt Enable:	$\checkmark$
WCMP Interrupt Enable:	
Select Channels	
Window Settings	

#### 図2-20.スタンバイ中動作に関するADCの設定

ADCO	2	Ð
🔅 Easy Setup 📃 Registers		
▼ ADC0		ĥ
Interrupt Enables		
RESRDY		
WCMP		
▼ Register: COMMAND 0x0		
SPCONV disabled *	_	
STCONV disabled -		
Register: CTRLA 0v81		
CONVMODE disabled •		
ENABLE enabled 💌		
FREERUN disabled		
LEFTADJ disabled -		
KESSEL 12-bit mode		
RUNSTBY enabled •		
▼ Register: CTRLB 0x7		
SAMPNUM 128 results accumulated ▼		

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

Interrupt Manager		0
😳 Easy Setup 📃 Registers		
<ul> <li>Interrupt Setting</li> </ul>		î
Global Interrupt Enable:		
Interrupt Priority		
Round-robin Scheduling Enable:		
Interrupt Level Priority:	0	
(?) Interrupt Vector with High Priority:	0	

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。 ADCO\_StartConversion(channel)と ADCO\_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;
```



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。
  - 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 <u>Connection]</u>セクションの[Data Gateway Interface]にあります。

Configuration	DGI Control Panel	^
A External Connection     Data Gateway Interface (DGI)	Curiosity Data Gateway Interface	Connect Start
Serial Port		agging 🖌 Autodetect protocols 🗌 Show Config search path 🖌 Reset M
Utilities     Protocols	Interfaces:	
	Serial Port Control Panel	v

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。

	[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2-26.DGI

Power Debugger Data Gateway	onnect Start
AD2 Logging □ Autodetert protocols Ø	leset MCU

#### 図2-28.[Power]の設定

DGi Control Panel							
Power Debugger	r Data Gateway				Bower Configuration	×	Disconnect
.5.2000.500					Enable B Channel		Start
					Trigger calibration		ADP Logging Autodetect protocols 🖌 Reset MCU
Interfaces:					Enable Range Source		
🗆 SPI 🏠 🗌	USART 🏠 🗌 TWI 🏠	🗆 GPIO 🏠	Power 🔯	🗌 Code Profiling 🏠	Lock ChA to High Range		
<b>~</b> 0 <b></b>	0 -0	0 000 0	A Current	Code Location	Enable Voltage Output		
		1 6 0	A Voltage		Voltage Output	1600 mV	
		2 🕬 🔕	B Current		Averaging No	~	
		3 🌑 🚳	B Voltage 🌑		OK Canc	el	

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

DGI Control Panel		^ X
Power Debugger Data Gateway		Disconnect
		Start
	ADP Logging	Autodetect protocols 🔽 Reset MCU
Interfaces:		
SPI 🛱 USART 🛱 🗆 TWI 🛱 🗆 GPIO 🛱	Power	
	A Current Code Location	
1 💕 🙂	A Voltage 💕	
2 💕 🖉		
3 💕 🖉		
Power Analysis		
6360µA		Channel A
6330µA		Channel B
5000µA -		Code Profiling
6240µA		GPIO
		Cursors
		Auto-scroll
6150µA -		Show zero
6090µA -		
14139 14140 141	4141 14142 14143 14144 14145 14146 14147 14148	
ChA]		
Window Average Instant		
007W,3* 007 ( 1*		
Scroll the mouse-wheel while pressing and holding the left shift key to a	zoom in the firme axis.	

- 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	
DGI Cont	ol Panel	^ X
Powe	Debugger Data Gateway	Connect
15020000	10	Start
	C ADPLogging Audodetet.scotocols	Reset MCU
Interface	s .	

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

DGI Control Panel								^ X
Power Debugger Data Gateway				B Power Configuration		×	Disconnect	
35220000566				Enable B Channel				Start
				Trigger calibration				ADP Logging Autodetect protocols 🗸 Reset MCU
Interfaces:				Enable Range Source				
SPI 🛱 🗆 USART 🙀 🗆 TWI 🛱	🗆 GPIO 🏠	Power	🗌 Code Profiling 🇳	Lock ChA to High Rang	e 🗆			
<b>@</b> ~ 0 <b>@</b> ~ 0	0 🕬 🔘	A Current	Code Location	Enable Voltage Output				
	1 🕬 🔕	A Voltage 🌍		Voltage Output	0	1600 m	v	
	2 🕬 🔕	B Current		Averaging	No	,	×	
	3 🌑 🙆	B Voltage 🌑		OK	Cano	cel		

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

DGI Control Panel	i .					A
Power Debugger Data Gateway						Disconnect
15020000566						Start
					ADP Logging	Autodetect protocols 🖌 Reset MC
Interfaces:						
SPI 🔅 USART 🔅 🗆 TWI 🔅 🗆 GPIO 🔅	Power 🔯 🗌 Code Profiling 🛱					
C-0 C-0 C-0 . C- 0	A Current Code Location					
1000	A Voltage					
	r tonge					
2	2					
3 💕 🖉	2					
Power Analysis						Ý
Power Analysis						•
3800µA -						
3700µА -						
🙀 3600μΑ -						2
2 3500µА -						ā
3400μΑ -						
■ 3300µA -						
3200µA -						
300004						And a start of the start of the
14319 14320 14	4321 14322	14323 14324	14325	14326	14327	14328
Ch A Window Average Instant						
Scroll the mouse-wheel while pressing and holding the left shift key	to zoom in the time axis. 📕					

- 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。

その他の商標は各社に帰属します。

© 2023, Microchip Technology Incorporated and its subsidiaries.

All Rights Reserved.

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.オーストラリア・シドニー Tel: 61-2-9868-6733インド・バンガロール Tel: 91-80-3090-4444オース Tel: 42Chandler, AZ 85224-6199 Tel: 480-792-7200中国 - 北京 Tel: 86-10 -8569-7000インド・ニューデリー Tel: 91-11-4160-8631Fax: 4 インド・ブネTel: 480-792-7277 Fax: 480-792-7277 Fax: 480-792-7277 Fax: 480-792-7277 Fax: 480-792-7277 Fax: 640-792-7277 Fax: 640-792-7277 Fax: 480-792-7277 Fax: 640-792-7277 Fax: 640-792-7277 Fax: 640-792-7277 Fax: 640-792-7277 Fax: 640-792-7277 Fax: 66-792-7277 Fax: 678-957-9614 Fel: 86-23-8980-9588 Fel: 86-23-8980-9588 PIE - 恵弗 Tel: 86-23-8980-9588 PIE - 恵希 Tel: 86-769-8702-9880 PIE - 広州 Tel: 86-769-8702-9880 PIE - 広州 Tel: 86-20-8755-8029インド - パン インド - ブネ Tel: 81-3-6880-3770Tel: 36- 7 - グ Tel: 36- Tel: 31-3-6880-3770Tel: 512-257-3370 ポース - イン、TX Tel: 86-571-8792-8115 Tel: 86-25-8473-2460Fax - 京原 マレーシア - クアラルンプール Tel: 442 Tel: 460-37651-7906Fax: 37 Y Tel: 442 Tel: 442 Tel: 442 Tel: 442 Tel: 60-4-227-8870Fax: 442 Tel: 443 Tel: 442 Tel: 443 Tel: 444 Tel: 4	ストリア・ヴェルス         3-7242-2244-39         3-7242-2244-393         アーク・コペンハーゲン         5-4485-5910         5-4485-2829         パランド・エスポー         58-9-4520-820         パス・パリ         3-1-69-53-63-20         3-1-69-30-90-79         パーガーヒンク         9-8931-9700         パーン         9-2129-3766400         パーパノブロン         0-7131 72400
本社オーストラリア・シドニーインド・バンガロールオース2355 West Chandler Blvd.Tel: 61-2-9868-6733Tel: 91-80-3090-4444Tel: 44Chandler, AZ 85224-6199中国 - 北京インド・ニューデリーFax: 4Tel: 480-792-7200Tel: 86-10 - 8569-7000Tel: 91-11-4160-8631デンマFax: 480-792-7277中国 - 成都インド・ブネTel: 44技術サポート:http://www.microchip.com/support中国 - 直慶Tel: 91-20-4121-0141URL:Tel: 86-28-8665-5511Tel: 91-20-4121-0141Fax: 4ウ国 - 重慶Tel: 86-23-8980-9588Tel: 81-6-6152-7160Tel: 33アトランタTel: 86-769-8702-9880Tel: 81-3-6880-3770Tel: 33Duluth, GA中国 - 広州Tel: 86-20-8755-8029Tel: 81-3-6880-3770Tel: 33Fax: 678-957-9614Tel: 86-571-8792-8115Tel: 82-253-744-4301ドイツFax: 678-957-9614Tel: 86-571-8792-8115Tel: 82-2554-7200FイツFax: 678-957-9614Tel: 86-571-8792-8115Tel: 82-2554-7200FイツFax: 678-957-9614Tel: 86-571-8792-8115Tel: 82-2554-7200FイツFax: 678-957-9614Tel: 852-2943-5100Fd ツTel: 45Fel: 512-257-3370中国 - 充州Tel: 86-25-8473-2460Tel: 60-3-7651-7906Fd ツWestborough, MATel: 86-25-8473-2460Tel: 60-3-7651-7906Fd ワFel: 774-760-0087Tel: 86-25-8473-2460Tel: 60-4-227-8870Tel: 45Fax: 774-760-0088PE - 青島Tel: 60-4-227-8870Tel: 45	トリア - ヴェルス         3-7242-2244-39         3-7242-2244-393         アーク - コペンハーゲン         5-4485-5910         5-4485-5910         5-4485-2829         アランド - エスポー         58-9-4520-820         アス - パリ         3-1-69-53-63-20         3-1-69-30-90-79         ア・ガーヒンク         9-8931-9700         ア・ハーン         9-2129-3766400         ア・ハイルブロン         9-7131-72400
シカゴ Itasca, IL Tel: 86-322-8502-7355フィリビン-マニラ Tel: 63-2634-9065Tel: 64 ドイン Tel: 63-2634-9065Tel: 74 F4 Y Tel: 63-2634-9065Tel: 74 F4 Y Tel: 74Tel: 63-263-0071 Fax: 630-285-0075Tel: 86-21-3326-8000ンガボールTel: 74 Fax: 44 Fax: 44 <b>ダラス</b> Tel: 72-818-7423Tel: 76-234-2334-2829台湾 - 新竹 Tel: 72-818-7423Fax: 74 Tel: 86-755-8664-2200Tel: 72-818-7423 Tel: 86-755-8664-2200Tel: 86-35-77-8366Tel: 74 75-71-78306Tel: 74 75-71-77-8300Tel: 72-818-7423 Tel: 86-755-8664-2200Tel: 886-7-213-77800Tel: 97 76-71Tel: 248-648-4000Tel: 86-186-6233-1526台湾 - 64x74.91 75-86600Tel: 248-648-4000Tel: 86-755-8664-2200台湾 - 64x74.91 75-8730Tel: 248-648-4000Tel: 86-755-8664-2200台湾 - 64x74.91 75-8730Tel: 248-648-4000Tel: 86-756-32100Fax: 37 75-8752Tel: 86-22-898-833-725274.71.71.71.71.71.71.71.71.71.71.71.71.71.	y - n - n - n - n - n - n - n - n - n -