

---

---

## 単独動作でのPGA付き12ビット差動ADCの使い方

---

---

### 要点

- ・ 8または12ビット分解能
- ・ 作動とシングルエンドの変換
  - 15個までのアナログ入力
    - ・ 15個の正入力と7つの負入力
- ・ 4つの内部入力
  - GND
  - VDD/10
  - 温度感知器
  - アナログ比較器からのDACREF
- ・ 組み込み内部参照基準と外部参照基準任意選択
- ・ 1～16倍で設定可能な利得増幅器
- ・ 自由走行動作
- ・ 左または右に揃えられた結果
- ・ 任意選択: 事象起動変換
- ・ 構成設定可能な窓比較器

### 序説

著者: Rupali Honrao, Amund Aune, and Egil Rotevatn, Microchip Technology Inc.

この技術概説はtinyAVR® 2系統で特徴とされる12ビットA/D変換器(ADC)での単独動作の使い方を説明します。下のコード例が単独動作を使って与えられます。

- ・ 窓比較器を使う割り込み
- ・ 事象起動変換
- ・ VDD測定
- ・ 内部温度感知器測定

単独動作ではADC変換が起動されると、単一採取に対してADC結果が利用可能です。この動作ではADC分解能を8ビットまたは12ビットに選ぶことができます。

ADC動作形態は次のように3つの群に分けることができます。

- ・ 単独動作 – 8または12ビット変換出力で起動毎に単一変換
- ・ 継続累積動作 – n採取の累積で起動毎に1変換
- ・ 集中累積動作 – 単一起動後に可能な限り速く累積したn採取での集中

他のADC動作の詳細については「[1. 関連文書](#)」章を参照してください。

## 目次

要点	1
序説	1
1. 関連文書	3
2. 構成設定	3
2.1. 単独動作8ビットと12ビット構成設定	3
2.2. 参照基準	3
2.3. シングルエント動作と差動動作	3
2.4. 設定可能な利得増幅器	4
2.5. 割り込み	4
2.6. 窓比較器	4
2.7. 事象	6
3. 入力回路	8
3.1. 入力インピーダンス	8
3.2. 採取持続時間	9
4. 電力とタイミング	9
4.1. クロック	9
4.2. PGAバイアスと出力採取持続時間	9
4.3. 変換時間	9
4.4. 自由走行動作	10
5. 出力処理	10
5.1. 結果範囲	10
5.2. 左揃え	10
5.3. 符号付きと符号なしの出力	11
6. 測定	11
6.1. VDD測定	12
6.2. 温度測定	14
7. GitHubからのコード例取得	16
8. 改訂履歴	16
Microchipウェブ サイト	17
製品変更通知サービス	17
お客様支援	17
Microchipデバイス コード保護機能	17
法的通知	17
商標	18
品質管理システム	18
世界的な販売とサービス	19

## 1. 関連文書

以下の文書がこの技術概説に関連します。

- データシート：製品頁でのtinyAVR 2データシート (.pdf)
  - [www.microchip.com/wwwproducts/en/ATtiny1624](http://www.microchip.com/wwwproducts/en/ATtiny1624)
  - [www.microchip.com/wwwproducts/en/ATtiny1626](http://www.microchip.com/wwwproducts/en/ATtiny1626)
  - [www.microchip.com/wwwproducts/en/ATtiny1627](http://www.microchip.com/wwwproducts/en/ATtiny1627)
- 継続累積動作でのPGA付き12ビット差動ADCの使い方：[www.microchip.com/DS90003257](http://www.microchip.com/DS90003257)
- 集中累積動作でのPGA付き12ビット差動ADCの使い方：[www.microchip.com/DS90003254](http://www.microchip.com/DS90003254)

## 2. 構成設定

### 2.1. 単独動作8ビットと12ビット構成設定

利用可能な2つの単独動作、単独8ビット動作と単独12ビット動作があります。この2つ動作は指令(ADCn.COMMAND)レジスタの動作形態(MODE)ビットを書くことによって選ぶことができます。下は単独動作の構成設定を示しているコード例です。

```
/* 8ビット */
ADC0.COMMAND = ADC_MODE_SINGLE_8BIT_gc;
/* 12-bit */
ADC0.COMMAND = ADC_MODE_SINGLE_12BIT_gc;
```

### 2.2. 参照基準

- 外部参照基準
- 内部参照基準
  - 1.024V
  - 2.048V
  - 2.500V
  - 4.096V
  - VDD

ADC用の参照基準電圧(VREF)はADCの変換範囲を制御します。外部参照基準と5つの内部参照基準が利用可能です。

```
ADC0.CTRL0 = ADC_REFSEL_1024MV_gc; /* 参照基準選択 1.024V */
```

VDDを除き、内部参照基準電圧は内部バンドギャップ参照基準から生成されます。VDDは選んだバンドギャップ参照基準電圧よりも最低0.5V高くなければなりません。

変換進行中の参照基準変更は出力を不正にします。自由走行動作使用時の安全な入力または参照基準の変更のため、何れかの変更を行うのに先立って、自由走行動作を禁止して変換完了を待ってください。次の変換を始める前に自由走行動作を許可してください。

```
ADC0.CTRL0 &= ~ADC_FREERUN_bm; /* 自由走行禁止 */
while(!(ADC0.INTFLAGS & ADC_SAMPRDY_bm)); /* 変換終了待ち */
ADC0.CTRL0 = ADC_REFSEL_VDD_gc; /* 参照基準としてVDDに構成設定 */
ADC0.CTRL0 |= ADC_FREERUN_bm; /* 自由走行許可 */
```

### 2.3. シングルエンド動作と差動動作

シングルエンド動作ではADCが選択可能な単一入力元の電圧を読む一方で、差動動作ではADCが2つの入力元間の電圧差を読みます。

差動動作は下で示されるように差動(DIFF)ビットに'1'を書くことによって構成設定されます。

```
/* 差動動作構成設定 */
ADC0.COMMAND |= ADC_DIFF_bm;
```

シングルエンド動作は下で示されるように差動(DIFF)ビットに'0'を書くことによって構成設定されます。

```
/* シングルエンド動作構成設定 */
ADC0.COMMAND &= ~ADC_DIFF_bm;
```

## 2.4. 設定可能な利得増幅器

ADCへの入力信号を増幅するのに設定可能な利得増幅器(PGA:Programmable Gain Amplifier)を使うことができます。利用可能な範囲は1~16倍利得です。PGAはPGA制御(ADCn.PGACTRL)レジスタでPGA許可(PGAEN)ビットに'1'を書いて利得(GAIN)ビット領域を構成設定することによって許可されます。

```
ADC0.PGACTRL |= ADC_GAIN_16X_gc | ADC_PGAEN_bm; /* 16倍利得でPGA許可 */
```

注: PGA制御は非0リセット値を持つAVRレジスタの1つです。これはレジスタの一部だけを構成設定する場合に考慮されなければなりません。

PGAが許可されると、正と負の多重器(ADCn.MUXPOSとADCn.MUXNEG)レジスタの経由(VIA)ビット領域の構成設定が必要とされます。VIAビットは共有され、故にMUXPOSとMUXNEGのVIAビット領域に書かれた値は両レジスタで更新されます。従って、1つの入力にPGAを使って他にPGAを使わないことはできません。

```
ADC0.MUXPOS |= ADC_VIA_gm; /* VIA許可 */
```

## 2.5. 割り込み

ADCは3つの独立した割り込みベクタが特徴です。割り込み条件の1つが起こると、割り込み要求フラグが設定され、CPUは通知されて対応する割り込み処理ルーチン(ISR:Interrupt Service Routine)へ向けられます。以下の表は利用可能なADC用割り込みベクタを示します。

表2-1. 利用可能な割り込みベクタと供給元

名前	ベクタ説明	割り込み要求フラグ	条件
ERROR	異常割り込み	TRIGOVR	1つが進行中に新しい変換が起動される。
		SAMPOVR	新しい変換がADCn.SAMPLEで未読試料を上書き
		RESOVR	新しい変換または累積がADCn.RESULTで未読結果を上書き
SAMPRDY	採取準備可割り込み	SAMPRDY	ADCn.SAMPLEで試料が利用可能
		WCMP	ADCn.CTRLDのWINSRCとWINCMによって定義されたとおりの時
RESRDY	結果準備可割り込み	RESRDY	ADCn.RESULTで結果が利用可能
		WCMP	ADCn.CTRLDのWINSRCとWINCMによって定義されたとおりの時

割り込み元は下のコード断片で示されるように割り込み制御(ADCn.INTCTRL)レジスタで対応するビットを書くことによって許可または禁止されます。

```
ADC0.INTCTRL = ADC_RESRDY_bm; /* 結果準備可割り込み許可 */
```

割り込み要求フラグは下のコード断片で示されるように割り込み要求フラグ(ADCn.INTFLAGS)レジスタでそのビット位置に'1'を書くことによって解除(0)されます。

```
ADC0.INTFLAGS = ADC_RESRDY_bm; /* 結果準備可割り込み要求フラグ解除 */
```

SAMPRDYとRESRDYの割り込み要求フラグは各々、採取(ADCn.SAMPLE)と結果(ADCn.RESULT)のレジスタを読むことによっても解除(0)することができます。

## 2.6. 窓比較器

ADCは変換または累積の出力が或る閾値越えと/または未満の時に割り込み要求フラグ(ADCn.INTFLAGS)レジスタの窓比較器割り込み要求(WCMP)フラグを立てて割り込み(WCMP)を要求することができます。利用可能な動作形態は次のとおりです。

- ABOVE - 値が閾値未満
- BELOW - 値が閾値超え
- INSIDE - 値が窓の内側(下側閾値以上、しかし上側閾値以下)
- OUTSIDE - 値が窓の外側(下側閾値未満または上側閾値超え)

閾値は窓比較器下側/上側閾値(ADCn.WINLTとADCn.WINHT)レジスタに書くことによって設定されます。使う窓動作は制御D(ADCn.CTRLD)レジスタの窓比較器動作(WINCM)ビット領域によって選ばれます。

制御D(ADCn.CTRLD)レジスタの窓動作元(WINSRC)ビットは比較が結果(ADCn.RESULT)レジスタの下位16ビットまたは採取(ADCn.SAMPLE)レジスタのどちらで行われるかを選びます。WCMPフラグに対して割り込み要求が許可される場合、WINSRCはRESRDYまたはSAMPRDYのどちらの割り込みベクタを要求するか選びます。

複数採取累積時、窓比較元が結果レジスタの場合、結果と閾値間の比較は最後の変換完了後に起きます。供給元が採取レジスタの場合、比較は毎回の変換後に起きます。

以下のコードは窓比較器の閾値構成設定方法と結果レジスタに対してINSIDE動作での比較の構成設定方法を示します。

```
ADC0.WINHT = 200; /* 窓上側閾値 */
ADC0.WINLT = 100; /* 窓下側閾値 */
ADC0.CTRLD |= ADC_WINCM_INSIDE_gc | ADC_WINSRC_RESULT_gc; /* 窓比較器元として結果レジスタ */
```

## 2.6.1. コード例

下のコード例は2000未満または3000超えのADC読み取りが、無効な信号はみ出しと見做される場合の応用例を示します。窓比較器は信号が閾値の内側の時にだけSAMPRDY割り込みを起動することによってそれらの出力の濾波に使われます。信号の電圧はSAMPRDY割り込み処理ルーチン(ISR)で計算されます。

```
#define F_CPU 3333333u1

#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <math.h>

#define TIMEBASE_VALUE      ((uint8_t) ceil(F_CPU*0.000001))
#define ADC_MAX_VALUE      ((1 << 12) - 1) /* シングルエンド動作で最大値は4095 */

/* デバッグ体験を改善するための揮発性変数 */
static volatile uint16_t adc_reading;
static volatile float voltage;

/*****
ADC初期化
*****/
void adc_init()
{
    ADC0.CTRLA = ADC_ENABLE_bm;
    ADC0.CTRLB = ADC_PRESC_DIV2_gc; /* fCLK_ADC=3.333333/2MHz */
    ADC0.CTRLC = ADC_REFSEL_VDD_gc | (TIMEBASE_VALUE << ADC_TIMEBASE_gp);
    ADC0.CTRLE = 17; /* (SAMPDUR+0.5)×fCLK_ADC=10.5µs採取持続時間 */

    ADC0.MUXPOS = ADC_MUXPOS_AIN6_gc; /* ADCチャンネルAIN6⇒PA6 */

    ADC0.WINHT = 3000; /* 窓上側閾値 */
    ADC0.WINLT = 2000; /* 窓下側閾値 */
    /* 窓比較器動作:内側。窓比較器元としてSAMPLEレジスタを使用 */
    ADC0.CTRLD = ADC_WINCM_INSIDE_gc | ADC_WINSRC_SAMPLE_gc;
    ADC0.INTCTRL = ADC_WCMP_bm; /* 窓比較割り込み許可 */

    ADC0.COMMAND = ADC_MODE_SINGLE_12BIT_gc; /* 単独12ビット動作 */
}

/*****
窓比較割り込み: この例では採取が或る窓の外側の時にこれが無効な信号はみ出しと見做されます。
信号が窓の内側で検出される時にだけ窓比較割り込みが起動します。そのようにしてはみ出しが無視されます。
*****/
ISR(ADC0_SAMPRDY_vect)
{
    ADC0.INTFLAGS = ADC_WCMP_bm; /* WCMPフラグ解除 */

    adc_reading = ADC0.SAMPLE; /* ADC採取読み込み */
    /* ADCピンの電圧計算、VDD=3.3V、12ビット分解能 */
    voltage = (float)(adc_reading * 3.3) / ADC_MAX_VALUE;
}

int main(void)
{
    adc_init();
    sei(); /* 全体割り込み許可 */

    while(1)
    {
        /* 1ms毎に1度変換を開始 */
    }
}
```

```

ADC0.COMMAND |= ADC_START_IMMEDIATE_gc;
    _delay_ms(1);
}
}

```

## 2.7. 事象

ADCは事象システムに接続することができます。事象システムはCPUの介在なしに周辺機能を通信させ、CPUに他の作業の実行や休止動作に留まることを許します。ADCは別の周辺機能への信号を提供する事象生成部、または別の周辺機能からの信号に基づいて作業を実行する事象使用部のどちらとしても接続することができます。

下表は利用可能なADCからの各種事象生成部を示します。

表2-2. ADC事象生成部

生成部名		説明	事象型	生成クロック領域	事象長
周辺機能	事象				
TCAn	RES	結果準備可	ハルス	CLK_PER	1 CLK_PER周期
	SAMP	採取準備可			
	WCMP	窓比較一致			

下は事象チャンネル1を通してEVOUT事象使用部に接続したADC0\_RES事象生成部の構成設定を示し、この場合、事象をPB2に出力します。

- ・ 事象生成部：ADC0 RES
- ・ 事象使用部：EVOUT (PB2)

```

EVSYS.CHANNEL1 = EVSYS_CHANNEL1_ADC0_RES_gc; /* ADC結果準備可 */
EVSYS.USEREVSYSSEVOUTB = EVSYS_USER_CHANNEL1_gc; /* 非同期事象チャンネル1 */

```

ADCは入力事象で検出して働くための1つの事象使用部を持ちます。下表は事象使用部と関連する機能を記述します。

表2-3. ADC事象使用部と利用可能な事象活動

使用部名		説明	入力検出	同期/非同期
周辺機能	入力			
ADCn	START	事象でADC開始	端	非同期

START事象活動は下のコード断片で示されるように指令(ADCn.COMMAND)レジスタの開始(START)ビット領域にEVENT\_TRIGGER設定が書かれる場合に起動することができます。

```

ADC0.COMMAND = ADC_START_EVENT_TRIGGER_gc;

```

下はRTC溢れに反応する事象使用部としてADC0.STARTの構成設定を示しているコード断片です。

```

EVSYS.CHANNEL0 = EVSYS_CHANNEL0_RTC_OVF_gc; /* 実時間計数器溢れ */
EVSYS.USERADC0START = EVSYS_USER_CHANNEL0_gc; /* 非同期事象チャンネル0 */

```

### 2.7.1. コード例

下は事象生成部と事象使用部としてADCの構成設定を示しているコード例です。

- ・ 事象使用部：RTC溢れ事象によって起動されるADC変換
  - RTCは望むADC採取速度でRTC溢れ事象を生成するように構成設定されます。この例での採取速度は100Hzです。
  - ADC変換は100Hzの速度で起動され、結果は割り込み要求フラグ(ADCn.INTFLAGS)レジスタの結果準備可(RESRDY)ビットが設定(1)された時に読まれます。
- ・ 事象生成部：ADC結果が準備可の時にPB2ピンが事象(ハルス)を出力

```

#define F_CPU 3333333u1

#include <avr/io.h>
#include <math.h>

#define TIMEBASE_VALUE ((uint8_t) ceil(F_CPU*0.000001))
#define ADC_MAX_VALUE (((1 << 12) / 2) - 1) /* 差動動作での最大値は2047 */

/* 容易なRTC事象周波数構成設定のために定義 */
#define ADC_SAMPLING_FREQ 100 /* Hz */
#define RTC_CLOCK 32768 /* Hz */

```

```

#define RTC_PERIOD          (RTC_CLOCK / ADC_SAMPLING_FREQ)

/* デバッグ体験を改善するための揮発性変数 */
static volatile int32_t adc_reading;
static volatile float voltage;

/*****
EVSYS初期化: チャンネル0:  事象システム生成部: RTC溢れ
                  事象システム使用部: ADC0
                チャンネル1: 事象システム生成部: ADC0結果準備可
                  事象システム使用部: EVOUTB (PB2ピン)
*****/
void event_system_init(void)
{
    PORTB.DIRSET = PIN2_bm;          /* EVOUTBを出力に構成設定 */

    EVSYS.CHANNEL0 = EVSYS_CHANNEL0_RTC_OVF_gc;          /* RTC溢れ ⇒ チャンネル0 */
    EVSYS.USERADCOSTART = EVSYS_USER_CHANNEL0_gc;      /* チャンネル0 ⇒ ADC0開始 */

    EVSYS.CHANNEL1 = EVSYS_CHANNEL1_ADC0_RES_gc;       /* ADC RESRDY ⇒ チャンネル1 */
    EVSYS.USEREVSYSEVOUTB = EVSYS_USER_CHANNEL1_gc;    /* チャンネル1 ⇒ EVOUTB (PB2) */
}

/*****
RTC初期化
*****/
void rtc_init(void)
{
    while(RTC.STATUS > 0);          /* 全レジスタ同期待ち */
    RTC.CTRLA = RTC_PRESCALER_DIV1_gc | RTC_RTCEN_bm;    /* RTC許可、前置分周なし */
    RTC.CLKSEL = RTC_CLKSEL_INT32K_gc;                   /* 32.768kHz内部RC発振器選択 */
    RTC.PER = RTC_PERIOD;
    while(RTC.STATUS > 0);          /* 全レジスタ同期待ち */
}

/*****
ADC初期化
*****/
void adc_init()
{
    ADC0.CTRLA = ADC_ENABLE_bm;
    ADC0.CTRLB = ADC_PRESC_DIV2_gc; /* fCLK_ADC=3.333333/2MHz */
    ADC0.CTRLC = ADC_REFSEL_VDD_gc | (TIMEBASE_VALUE << ADC_TIMEBASE_gp);
    ADC0.CTRLE = 17;                /* (SAMPDUR+0.5)×fCLK_ADC=10.5µs採取持続時間 */

    ADC0.MUXPOS = ADC_MUXPOS_AIN6_gc;          /* ADCチャンネルAIN6⇒PA6 */
    ADC0.MUXNEG = ADC_MUXNEG_AIN7_gc;         /* ADCチャンネルAIN7⇒PA7 */
    /* 事象起動でADC変換開始 */
    ADC0.COMMAND = ADC_DIFF_bm | ADC_MODE_SINGLE_12BIT_gc | ADC_START_EVENT_TRIGGER_gc;
}

int main(void)
{
    event_system_init();
    rtc_init();
    adc_init();

    while(1)
    {
        if(ADC0.INTFLAGS & ADC_RESRDY_bm)      /* ADC採取準備可か調査 */
        {

```

```

adc_reading = ADC0.RESULT; /* ADC結果読み込み、割り込み要求フラグ解除 */
/* ADCピンの電圧計算、VDD=3.3V、12ビット分解能 */
voltage = (float)((adc_reading * 3.3) / ADC_MAX_VALUE);
}
}
}

```

### 3. 入力回路

#### 3.1. 入力インピーダンス

ピンに与えられた電圧水準が試料化される時にそれは最初に採取/保持コンデンサ(C<sub>IN</sub>)によって捕獲されます。これはADCが信号を試料化する間その電圧が変わらないことを保証します。C<sub>IN</sub>を或る電圧水準に充放電するのにかかる時間は入力抵抗(R<sub>IN</sub>)によって制限されます。次式は時定数τと入力インピーダンス間の比例関係を示します。

$$\tau = R_{IN} \times C_{IN}$$

ADCの入力特性の詳細についてはデータシートの「電気的特性」章を参照してください。

ADCの12ビット分解能(と任意選択の利得)は測定が正しいことを確実にするため、最終電圧の99.9%を超えて安定される入力回路のインパルス応答を必要とします。以下の利得なしと16倍利得付きの計算例は12ビット分解能で正しく採取するため、ADCに対して必要な信号の安定度を示します。

$$V_{MSb} = V_{REF} - \frac{V_{REF}}{4096 \times \text{利得}}$$

$$V_{MSb\%} = \left(1 - \frac{1}{4096 \times \text{利得}}\right) \times 100\%$$

$$V_{MSb\% \text{利得なし}} = \left(1 - \frac{1}{4096 \times 1}\right) \times 100\% = 99.975\%$$

$$V_{MSb\% 16 \text{倍利得}} = \left(1 - \frac{1}{4096 \times 16}\right) \times 100\% = 99.998\%$$

入力回路に対するインパルス応答は次式によって与えられます。

$$V(t) = V_{IN} \times (1 - e^{-t/\tau})$$

V<sub>IN</sub>が100%であるV<sub>MSb</sub>に対する2つの例を解決すると、以下の安定時間が得られます。

$$t_{\text{利得なし}} = 8.29\tau$$

$$t_{16 \text{倍利得}} = 10.81\tau$$

外部信号のインピーダンスは右図で示されるように回路をもっと複雑な系に拡大して安定時間を計算する時にも考慮されるべきです。

外部インピーダンスの特性は安定時間計算をどれ位複雑にするかを決めます。けれども、これはこの技術概説によって網羅されません。

図3-1. 内部アナログ入力回路の模式

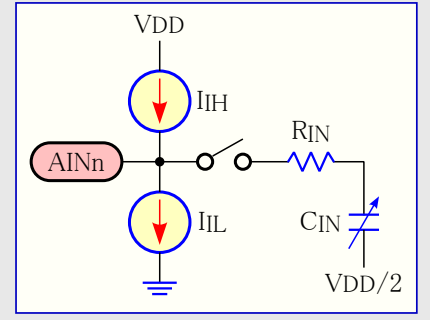
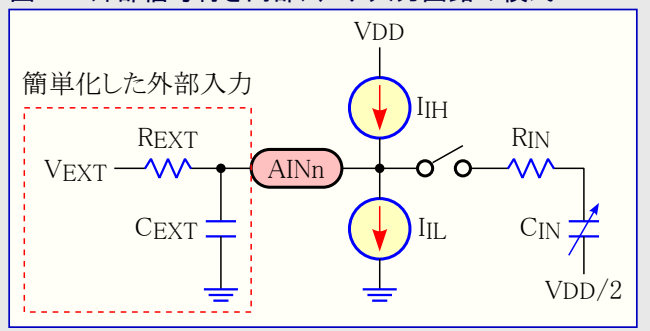


図3-2. 外部信号付き内部アナログ入力回路の模式



##### 3.1.1. PGA

PGAは選んだ利得設定に応じた入力インピーダンスでアナログ入力とADC間に接続されます。PGAの入力特性の詳細については「電気的特性」章を参照してください。上の式はPGAインピーダンス値を使う時に最適な採取持続時間を計算するのに対しても同じです。

PGAが使われると、継続的に採取され、ADCがPGAを採取する時にだけ保持状態です。変換間の時間が必要とする採取時間よりも長い場合、これは支援される値を最小にするため、採取持続時間(SAMPDUR)を設定することによって総変換時間をより短くするのに利用することができます。



## 3.2. 採取持続時間

適切な採取持続時間は「3.1. 入力インピーダンス」項で示されるように回路のインパルス応答に基づいて計算されるか、またはADC変換からの安定な出力が達成されるまでファームウェアで採取持続時間を調整することによって見つけるかのどちらかで行うことができます。

このADCに対する採取持続時間は最大256 ADCクロック(CLK\_ADC)周期にすることができ、制御E(ADCn. **CTRL E**)レジスタの採取持続時間(SAMPDUR)ビット領域を使って構成設定されます。採取持続時間はPGAが禁止されている時にSAMPDUR+0.5(CLK\_ADC)周期、PGAが許可されている時にSAMPDUR+1(CLK\_ADC)周期です。入力インピーダンスが非常に高い場合、更なる採取持続時間増加にADC前置分周器増加使用することもできます。

最小採取持続時間は以下のコード断片で示されるように構成設定されます。計算は禁止されたPGA、16MHzで走行するCPUクロックに基づきます。

```
ADC0. CTRLB = ADC_PRESC_DIV2_gc; /* ADCクロック: 8MHz */
ADC0. CTRL E = 0; /* 採取持続時間: (0+0.5)/8MHz=0.06μs */
```

最大採取持続時間は以下のコード断片で示されるように構成設定されます。計算は禁止されたPGA、16MHzで走行するCPUクロックに基づきます。

```
ADC0. CTRLB = ADC_PRESC_DIV40_gc; /* ADCクロック: 400kHz */
ADC0. CTRL E = 255; /* 採取持続時間: (255+0.5)/400kHz=639μs */
```

## 4. 電力とタイミング

### 4.1. クロック

ADCクロック(CLK\_ADC)は周辺機能クロック(CLK\_PER)から下げられます。これは制御B(ADCn. **CTRL B**)レジスタの前置分周器(PRESC)ビット領域によって構成設定することができます。

```
ADC0. CTRLB = ADC_PRESC_DIV20_gc; /* CLK_ADC=CLK_PER/20 */
```

ADCとPGAの内部タイミングのいくつかはCLK\_ADCと無関係です。ADCクロック周波数に関わらずに正しい内部タイミングを保証するため、制御C(ADCn. **CTRL C**)レジスタの時間基準(TIMEBASE)ビット領域に(CLK\_PER周期で与えられる)1μs時間基準が書かれなければなりません。時間基準は最も近い整数値に丸められなければなりません。以下のコード断片はceil関数を使ってこれがどう行われるかを示します。

```
#include <math.h>
#define F_CPU 3333333u1
#define TIMEBASE_VALUE ((uint8_t) ceil(F_CPU*0.000001))

ADC0. CTRL C = (TIMEBASE_VALUE << ADC_TIMEBASE_gp);
```

### 4.2. PGAバイアスと出力採取持続時間

ADC PGA制御(ADCn. **PGA CTRL**)レジスタのPGAバイアス選択(PGABIASSEL)ビット領域はADCクロック周波数に応じて消費電力を減らすように構成設定することができます。ADC PGA採取持続時間(ADCPGASAMPDUR)ビット領域はADCがPGAの出力を採取するCLK\_ADC周期数を減らすように構成設定することができます。これもADCクロック周波数と無関係です。

推奨される $f_{CLK\_ADC}$ の組み合わせとPGABIASSELとADCPGASAMPDURについてはデータシートでこれらのビット領域に対するレジスタ説明をご覧ください。

構成設定例が下で示されます。

```
ADC0. PGACTRL = ADC_GAIN_16X_gc | /* 16倍利得 */
ADC_PGABIASSEL_100PCT_gc | /* 100%バイアス電流 */
ADC_ADCPGASAMPDUR_32CLK_gc | /* 32周期採取のPGA */
ADC_PGAEN_bm; /* PGA許可 */
```

**注:** PGA制御は非0セット値を持ついくつかのAVRレジスタの1つです。これはレジスタの一部だけを構成設定する場合に考慮されなければなりません。

### 4.3. 変換時間

単一結果に対する総変換時間は以下によって計算されます。

$$\text{総変換時間(12ビット)} = \text{初期化} + \frac{\text{SAMPDUR} + 15.5}{f_{\text{CLK\_ADC}}}$$

$$\text{総変換時間(8ビット)} = \text{初期化} + \frac{\text{SAMPDUR} + 11.5}{f_{\text{CLK\_ADC}}}$$

例えば、初期化=60μs、SAMPDUR=2、 $f_{\text{CLK\_ADC}}$ =1MHzが与えられると、8ビット総変換時間は次によって与えられます。

$$\text{総変換時間(8ビット)} = 60\mu\text{s} + \frac{2 + 11.5}{1 \text{ (MHz)}} = 73.5\mu\text{s}$$

制御A(ADCn.CTRLA)レジスタで'1'を書かれた低遅延(LOWLAT)ビットでは初期化時間がADC許可で一度だけ必要とされます。その後、上の例は13.5µsの総変換時間を与えます。

ADCの採取期間は(SAMPDUR+0.5)CLK\_ADC周期として制御E(ADCn.CTRLE)レジスタの採取持続時間(SAMPDUR)ビット領域を通して構成設定されます。

```
ADC0.CTRLE = 2; /* 2に構成設定された採取持続時間 */
```

PGAが使われる場合、入力採取持続時間は(SAMPDUR+1)CLK\_ADC周期である一方で、PGA制御(ADCn.PGACTRL)レジスタのADC PGA採取持続時間(ADCPGASAMPDUR)はADCがPGAをどれ位の長さ採取するかを制御します。

```
ADC0.PGACTRL = ADC_ADCPGASAMPDUR_15CLK_gc; /* 15 CLK_ADC周期 */
```

#### 4.4. 自由走行動作

自由走行動作では直前の変換が完了したら直ぐに新しい変換が開始されます。

下のコード断片で示されるように制御F(ADCn.CTRLF)レジスタの自由走行(FREERUN)ビットに'1'を書くことによって構成設定されます。

```
ADC0.CTRLF = ADC_FREERUN_bm; /* ADC自由走行動作許可 */
```

新しい変換は結果が結果(ADCn.RESULT)レジスタで利用可能になった直後に開始されます。これは割り込み要求フラグ(ADCn.INTFLAG)レジスタの結果準備可(RESRDY)によって合図されます。単独12ビットでの自由走行変換速度は次によって与えられます。

$$f_{\text{CONV}} = \frac{f_{\text{CLK\_ADC}}}{\text{SAMPDUR} + 15.5}$$

例えば、SAMPDUR=2で $f_{\text{CLK\_ADC}}=3.33\text{MHz}$ が与えられると、変換速度は188.571kHzです。

単独8ビットでの自由走行変換は次によって与えられます。

$$f_{\text{CONV}} = \frac{f_{\text{CLK\_ADC}}}{\text{SAMPDUR} + 11.5}$$

例えば、SAMPDUR=2で $f_{\text{CLK\_ADC}}=3.33\text{MHz}$ が与えられると、変換速度は246.666kHzです。

## 5. 出力処理

### 5.1. 結果範囲

ADC変換からの出力は次式によって与えられます。

$$\text{シングル エント}12\text{ビット変換} = \frac{V_{\text{INP}} \times \text{利得}}{V_{\text{REF}}} \times 4096 \in [0, 4095]$$

$$\text{シングル エント}8\text{ビット変換} = \frac{V_{\text{INP}} \times \text{利得}}{V_{\text{REF}}} \times 256 \in [0, 255]$$

$$\text{差動}12\text{ビット変換} = \frac{(V_{\text{INP}} - V_{\text{INN}}) \times \text{利得}}{V_{\text{REF}}} \times 2048 \in [-2048, 2047]$$

$$\text{差動}8\text{ビット変換} = \frac{(V_{\text{INP}} - V_{\text{INN}}) \times \text{利得}}{V_{\text{REF}}} \times 128 \in [-128, 127]$$

$V_{\text{INP}}$ と $V_{\text{INN}}$ はADCに対する正と負の入力で、 $V_{\text{REF}}$ は選んだ参照基準電圧です。利得はPGAで構成設定されるように1~16倍で、PGAが使われない場合は1倍です。

ADCは2つの出力レジスタ、採取(ADCn.SAMPLE)と結果(ADCn.RESULT)のレジスタを持ちます。16ビットの採取レジスタは常に最後のADC変換出力(1採取)で更新されます。単独変換動作では採取(ADCn.SAMPLE)と結果(ADCn.RESULT)の両レジスタは同じです。

シングル エント12ビット変換ではアナログ ピンに印加された平均電圧が次によって計算されます。

$$V_{\text{INP}} = \frac{\text{ADCn.SAMPLE} \times V_{\text{REF}}}{4096 \times \text{利得}}$$

### 5.2. 左揃え

制御F(ADCn.CTRLF)レジスタの左揃え(LEFTADJ)はこれが支援される動作で出力データの左移動を許します。許可される場合、これは結果と採取の両レジスタからの出力を左移動します。これは以下のコード断片で示されるように構成設定されます。

```
ADC0.CTRLF = ADC_LEFTADJ_bm; /* 左揃えビット許可 */
```

左揃えは単独変換12ビット、ADC動作1で利用可能です。

以下の表は左揃え機能がシングル エント動作と差動動作で結果レジスタの出力形式にどう影響を及ぼすかを示します。

表5-1. RESULTレジスタ - シングル エント動作 - ADC動作1 (単独変換12ビット)

LEFTADJ	RES31~24	RES23~16	RES15~12	RES11~8	RES7~0
0	\$00			変換11~0	
1	\$00		変換11~0 << 4		

表5-2. RESULTレジスタ - 差動動作 - ADC動作1 (単独変換12ビット)

LEFTADJ	RES31~24	RES23~16	RES15~12	RES11~8	RES7~0
0	符号拡張			符号付き変換11~0	
1	符号拡張		符号付き変換11~0 << 4		

以下の表はシングル エント動作と差動動作で左揃え機能が採取レジスタ出力形式にどう影響を及ぼすかを示します。

表5-3. SAMPLEレジスタ - シングル エント動作/差動動作 - ADC動作1 (単独変換12ビット)

LEFTADJ	DIFF	SAMPLE15~12	SAMPLE11~8	SAMPLE7~0
0	0	\$00	変換11~0	
	1	符号拡張	符号付き変換11~0	
1	0	変換11~0 << 4		
	1	符号付き変換11~0 << 4		

例えば、左揃え機能が禁止され、ADCn.SAMPLE値が\$0FFFの場合、左揃えが許可される時の対応するADCn.SAMPLE値は\$FFF0です。

### 5.3. 符号付きと符号なしの出力

シングル エント動作での採取に対するデータ形式は\$0000が0を表し、\$0FFFが最大数値を表す符号なし1の補数です。アナログ入力がADCの参照基準よりも高い場合、12ビットADC出力は\$0FFFの最大値に等しくなります。同様に、入力が0V未満の場合、ADC出力は\$0000です。

差動動作についてデータ形式は符号拡張付き2の補数です。

#### 採取レジスタ出力

採取変数のデータ型はシングル エント動作使用時にuint16\_tです。

採取変数のデータ型は差動動作使用時にint16\_tです。

例えば、12ビット動作でシングル エント動作使用時、単一採取の電圧は下のコード断片で示されるように解釈され得ます。

```
uint16_t sample_variable = ADCn.SAMPLE;
float sample_voltage = (sample_variable * VREF) / 4095;
```

12ビット動作で差動動作使用時、単一採取の電圧は下のコード断片で示されるように解釈され得ます。

```
int16_t sample_variable = ADCn.SAMPLE;
float sample_voltage = (sample_variable * VREF) / 2047;
```

#### 結果レジスタ出力

結果変数のデータ型はシングル エント動作使用時にuint32\_tです。

結果変数のデータ型は差動動作使用時にint32\_tです。

例えば、12ビット動作でシングル エント動作使用時、SAMPNUM個累積された採取の電圧は下のコード断片で示されるように解釈され得ます。

```
uint32_t result_variable = ADCn.SAMPLE;
float result_voltage = ((result_variable * VREF) / SAMPNUM) / 4095;
```

12ビット動作で差動動作使用時、SAMPNUM個累積された採取の電圧は下のコード断片で示されるように解釈され得ます。

```
int32_t result_variable = ADCn.SAMPLE;
float result_voltage = ((result_variable * VREF) / SAMPNUM) / 2047;
```

## 6. 測定

単独動作では変換起動受け取りで単独採取結果がADC結果(ADCn.RESULT)と採取(ADCn.SAMPLE)レジスタで利用可能です。結果の準備が整うと、割り込み要求フラグ(ADCn.INTFLAGS)レジスタでRESRDYとSAMPRDYのビットが設定(1)されます。

単独動作ではADCn.RESULTとADCn.SAMPLEでの結果は等価です。

下のコード断片はADC変換を開始して変換終了まで待ち、採取レジスタから結果を読む方法を示します。

```
ADC0.COMMAND |= ADC_START_IMMEDIATE_gc;          /* ADC変換開始 */
while(!ADC0.INTFLAGS & ADC_SAMPDRY_bm);         /* 変換終了待ち */
adc_result = ADC0.SAMPLE;                        /* 採取(変換結果)読み込み */
```

同様に、下のコードは上と同じものを示しますが、今回は結果レジスタを読みます。

```
ADC0.COMMAND |= ADC_START_IMMEDIATE_gc;          /* ADC変換開始 */
while(!ADC0.INTFLAGS & ADC_SAMPDRY_bm);         /* 変換終了待ち */
adc_result = ADC0.RESULT;                        /* 結果(変換結果)読み込み */
```

## 6.1. VDD測定

下のコード断片はADCの初期化方法とデバイスに給電している電圧を測定するために内部入力としてVDDを使う方法を示します。データ可視器(Data Visualizer)によって図表にされ得るような方法で結果を送信するためUSARTも使われます。コードをダウンロードする方法の指示については「7. GitHubからのコード例取得」を、データ可視器で図表を構成設定する方法の指示については「6.1.1. データ可視器での図表作図」をご覧ください。

```
#define F_CPU 3333333ul

#include <avr/io.h>
#include <util/delay.h>
#include <math.h>

#define TIMEBASE_VALUE ((uint8_t) ceil(F_CPU*0.000001))
#define ADC_MAX_VALUE ((1 << 12) - 1) /* 単独動作での最大値は4095 */
#define BAUD_RATE 9600
#define BAUD_REG_VAL ((float)(64 * F_CPU / (16 * (float)BAUD_RATE)) + 0.5)

static uint16_t adc_reading;
static float voltage;

/*****
ADC初期化
*****/
void adc_init()
{
    ADC0.CTRLA = ADC_ENABLE_bm;
    ADC0.CTRLB = ADC_PRESC_DIV2_gc; /* fCLK_ADC=3.333333/2MHz */
    ADC0.CTRLC = ADC_REFSEL_1024MV_gc | (TIMEBASE_VALUE << ADC_TIMEBASE_gp);
    ADC0.CTRLE = 17; /* (SAMPDUR+0.5)×fCLK_ADC=10.5µs採取持続時間 */

    ADC0.MUXPOS = ADC_MUXPOS_VDDDIV10_gc; /* ADCチャンネル VDD/10 */
    ADC0.COMMAND = ADC_MODE_SINGLE_12BIT_gc; /* 単独12ビット動作 */
}

/*****
USART初期化
*****/
void usart_init()
{
    PORTB.DIRSET = PIN2_bm; /* TxDを出力に設定 */
    USART0.CTRLB = USART_TXEN_bm; /* USART送信部許可 */
    USART0.BAUD = BAUD_REG_VAL; /* ボーレート設定 */
}

/*****
USART経由で浮動値をデータ可視器へ送出
*****/
void USART_send_DV(float *float_ptr)
{
    uint8_t *byte_ptr = (uint8_t *) float_ptr;
```

```

while(!(USART0.STATUS & USART_DREIF_bm));
USART0.TXDATAL = 0x33; /* データ流れ開始バイト送出 */

for(uint8_t i = 0; i < sizeof(float); i++)
{
    while(!(USART0.STATUS & USART_DREIF_bm));
    USART0.TXDATAL = byte_ptr[i];
}

while(!(USART0.STATUS & USART_DREIF_bm));
USART0.TXDATAL = ~0x33; /* データ流れ停止バイト送出 */
}

int main(void)
{
    adc_init();
    usart_init();

    while(1)
    {
        ADC0.COMMAND |= ADC_START_IMMEDIATE_gc; /* ADC変換開始 */
        while(!(ADC0.INTFLAGS & ADC_SAMPRDY_bm)); /* 変換終了待ち */

        adc_reading = ADC0.SAMPLE; /* ADC採取読み込み、フラグ解除 */
        /* VDD計算、VREF=1.024V、12ビット分解能。入力チャンネルがVDD/10なので10倍 */
        voltage = (float)(adc_reading * 1.024 * 10) / ADC_MAX_VALUE;

        USART_send_DV(&voltage); /* データ可視器へ電圧送出 */

        _delay_ms(500);
    }
}

```

### 6.1.1. データ可視器で図表作図

以下指示はデータ流れ規約を使うことによってデータ可視器でUSARTデータを作図する方法を示します。

**注:** データ可視器(Data Visualizer)の詳細な情報については[データ可視器使用者の手引き](#)を参照してください。

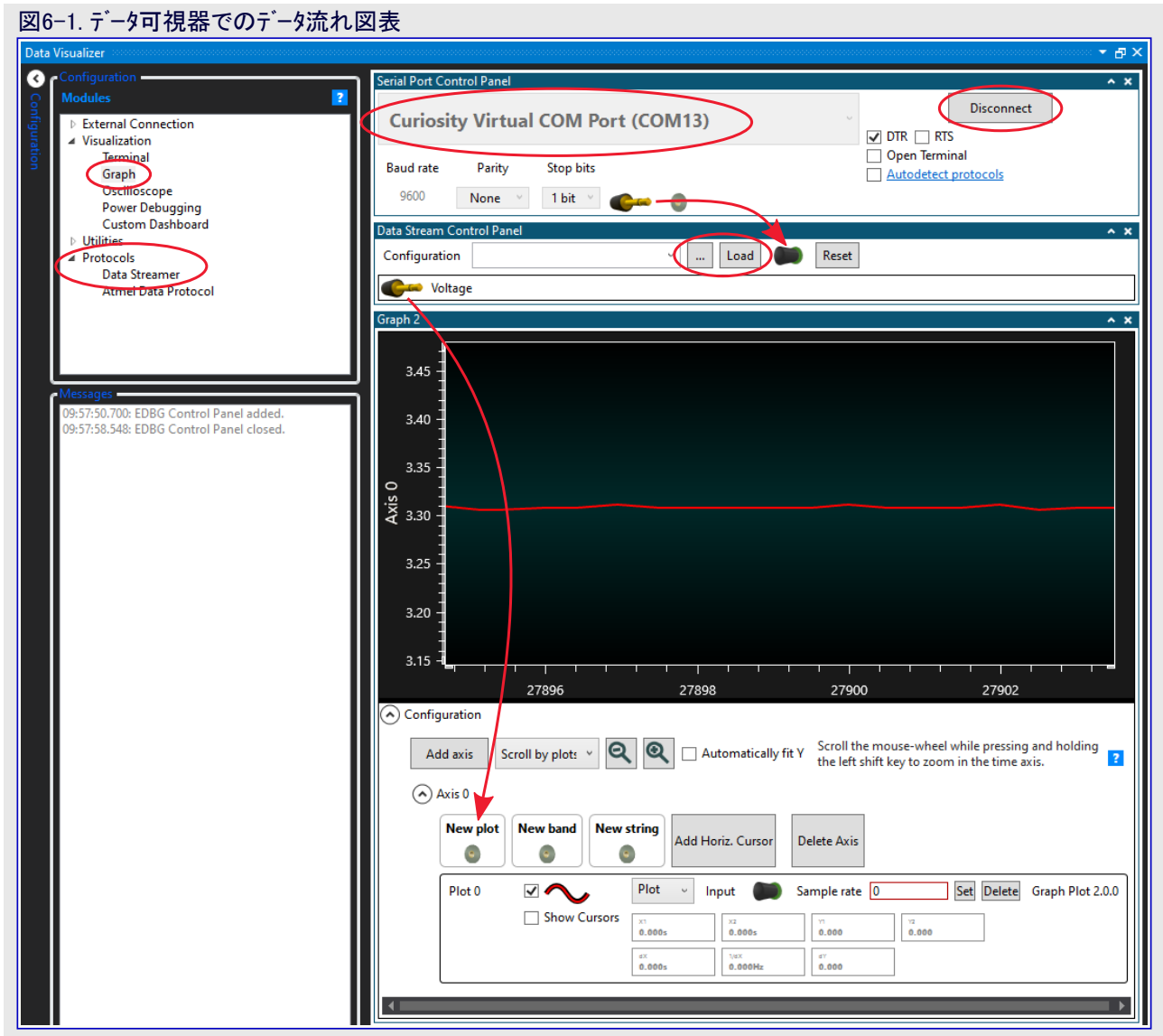
1. Data Visualizerを開いてください。
2. Data VisualizerでConfiguration(構成設定)⇒External Connection(外部接続)⇒Serial Port(シリアルポート)を開いてください。
3. Curiosity Virtual COM portを選び、Baud rate:を9600に、その後にConnect(接続)を選んでください。
4. Configuration(構成設定)⇒Protocols(規約)⇒Data Streamer(データ流れ)を開いてください。
5. Data Stream Control Panel(データ流れ制御盤)に於いてConfiguration(構成設定)下で構成設定ファイルを探してその後にLoad(読み込み)を選んでください。

**注:** この場合、構成設定ファイルはsingle\_measuring\_VDD/single\_VDD\_voltage.txtで、例ソースコードプロジェクトフォルダで見つけることができます。

**注:** データ流れ規約のより多くの詳細については[データ可視器使用者の手引き](#)、[データ流れ規約項](#)を参照してください。

1. Configuration(構成設定)⇒Visualization(可視化)⇒Graph(図表)を開いてください。
2. 図表を作図するために下図の赤矢印で示されるように接続を引き摺ってください。

図6-1. データ可視器でのデータ流れ図表



図表でY軸を調整するには以下の手順に従ってください。

1. Graph(図表)のConfiguration(構成設定)下でAutomatically Fit Y(Y自動適合)を選択解除してください。
2. 作図領域内側の何処かをクリックしてください。
3. Ctrlキーを押している、または押し続けている間にマウス ホイールをスクロールしてください。

図表でX軸を調整するには以下の手順に従ってください。

1. 作図領域内側の何処かをクリックしてください。
2. Shiftキーを押している、または押し続けている間にマウス ホイールをスクロールしてください。

**注:** Data Visualizer(データ可視器)⇒Graph(図表)のより多くの詳細についてはデータ可視器使用者の手引きの図表項を参照してください。

## 6.2. 温度測定

下のコード断片は内部温度感知器を使って温度を測定する方法を示します。

```
#define F_CPU 3333333u1

#include <avr/io.h>
#include <math.h>
#include <util/delay.h>

#define TIMEBASE_VALUE ((uint8_t) ceil(F_CPU*0.000001))
```

```

#define TEMPSENSE_SAMPDUR    ((uint8_t) ceil(F_CPU*0.000032/2)) /* TEMPSENSEに対してSAMPDURは
                                                                    ≥32μs×f_ADC≒32μs×1.67MHz≒54でなければなりません。 */

/* デバッグ体験を改善するための揮発性変数 */
static volatile uint16_t adc_reading;
static volatile uint16_t temperature_in_K;
static volatile int16_t temperature_in_degC;

/*****
ADC初期化
*****/
void adc_init()
{
    ADC0.CTRLA = ADC_ENABLE_bm;
    ADC0.CTRLB = ADC_PRESC_DIV2_gc; /* fCLK_ADC=3.333333/2MHz */
    ADC0.CTRLC = ADC_REFSEL_1024MV_gc | (TIMEBASE_VALUE << ADC_TIMEBASE_gp);
    ADC0.CTRLE = TEMPSENSE_SAMPDUR;

    ADC0.MUXPOS = ADC_MUXPOS_TEMPSENSE_gc; /* ADC内部温度感知器 */
    ADC0.COMMAND = ADC_MODE_SINGLE_12BIT_gc; /* 単独変換12ビット動作 */
}

int main(void)
{
    adc_init();

    int8_t sigrow_offset = SIGROW_TEMPSENSE1; /* 識票列から符号付き変位読み込み */
    uint8_t sigrow_gain = SIGROW_TEMPSENSE0; /* 識票列から符号なし利得/傾斜読み込み */

    while(1)
    {
        ADC0.COMMAND |= ADC_START_IMMEDIATE_gc; /* ADC変換開始 */
        while(!(ADC0.INTFLAGS & ADC_RESRDY_bm)); /* 変換終了待ち */

        /* データシートで説明されるような校正補償 */
        adc_reading = ADC0.RESULT >> 2; /* 1.024V内部参照基準でのADC結果の上位10ビット */
        uint32_t temp = adc_reading - sigrow_offset;
        temp *= sigrow_gain; /* 結果(10+8ビット)が16ビット変数を溢れるかもしれません。 */
        temp += 0x80; /* 下の除算で正しい整数丸めを得るために256/2を加算 */
        temp >>= 8; /* ケルビンで処理した温度を得るために結果を256で除算 */
        temperature_in_K = temp;
        temperature_in_degC = temperature_in_K - 273;

        _delay_ms(500);
    }
}

```

## 7. GitHubからのコード例取得

コード例は図画的ユーザーインターフェース(GUI)を通して応用コードを提供するウェブに基づくサーバーであるGitHubを通して入手可能です。コード例はAtmel StudioとMPLAB Xの両方で開くことができます。MPLAB XでAtmel Studioプロジェクトを開くにはメニューでFile(ファイル)⇒Import(インポート)⇒Atmel Studio project(Atmel Studioプロジェクト)で.cprojファイルへ誘導してください。

GitHubウェブ頁：[GitHub](#)

### コード例

tinyAVR 2系列のデバイス用コード例を探すのは、GitHub例閲覧部でデバイス名、例えば、ATtiny1627に対して検索することによって行うことができます。



GitHubでコード例を見てください。  
貯蔵庫を閲覧するにはクリックしてください。

Clone(複製)またはDownload(ダウンロード)鈕をクリックすることによってGitHubの例頁から.zipファイルとしてコードをダウンロードしてください。

## 8. 改訂履歴

改訂	日付	注釈
A	2020年7月	初版文書公開



---

## Microchipウェブ サイト

---

Microchipは[www.microchip.com/](http://www.microchip.com/)で当社のウェブ サイト経由でのオンライン支援を提供します。このウェブ サイトはお客様がファイルや情報を容易に利用可能にするのに使われます。利用可能な情報のいくつかは以下を含みます。

- **製品支援** – データシートと障害情報、応用記述と試供プログラム、設計資源、使用者の手引きとハードウェア支援資料、最新ソフトウェア配布と保管されたソフトウェア
- **一般的な技術支援** – 良くある質問(FAQ)、技術支援要求、オンライン検討グループ、Microchip設計協力課程会員一覧
- **Microshipの事業** – 製品選択器と注文の手引き、最新Microchip報道発表、セミナーとイベントの一覧、Microchip営業所の一覧、代理店と代表する工場

---

## 製品変更通知サービス

---

Microchipの製品変更通知サービスはMicrochip製品を最新に保つのに役立ちます。加入者は指定した製品系統や興味のある開発ツールに関連する変更、更新、改訂、障害情報がある場合に必ず電子メール通知を受け取ります。

登録するには[www.microchip.com/pcn](http://www.microchip.com/pcn)へ行って登録指示に従ってください。

---

## お客様支援

---

Microchip製品の使用者は以下のいくつかのチャネルを通して支援を受け取ることができます。

- 代理店または販売会社
- 最寄りの営業所
- 組み込み解決技術者(ESE:Embedded Solutions Engineer)
- 技術支援

お客様は支援に関してこれらの代理店、販売会社、またはESEに連絡を取るべきです。最寄りの営業所もお客様の手助けに利用できます。営業所と位置の一覧はこの資料の後ろに含まれます。

技術支援は[www.microchip.com/support](http://www.microchip.com/support)でのウェブ サイトを通して利用できます。

---

## Microchipデバイスコード保護機能

---

Microchipデバイスでの以下のコード保護機能の詳細に注意してください。

- Microchip製品はそれら特定のMicrochipデータシートに含まれる仕様に合致します。
- Microchipは意図した方法と通常条件下で使われる時に、その製品系統が今日の市場でその種類の最も安全な系統の1つであると考えます。
- コード保護機能を破るのに使われる不正でおそらく違法な方法があります。当社の知る限りこれらの方法の全てはMicrochipのデータシートに含まれた動作仕様外の方法でMicrochip製品を使うことが必要です。おそらく、それを行う人は知的財産の窃盗に関与しています。
- Microchipはそれらのコードの完全性について心配されているお客様と共に働きたいと思います。
- Microchipや他のどの半導体製造業者もそれらのコードの安全を保証することはできません。コード保護は当社が製品を”破ることができない”として保証すると言うことを意味しません。

コード保護は常に進化しています。Microchipは当社製品のコード保護機能を継続的に改善することを約束します。Microchipのコード保護機能を破る試みはデジタル ミレニアム著作権法に違反するかもしれません。そのような行為があなたのソフトウェアや他の著作物に不正なアクセスを許す場合、その法律下の救済のために訴権を持つかもしれません。

---

## 法的通知

---

デバイス応用などに関してこの刊行物に含まれる情報は皆さまの便宜のためにだけ提供され、更新によって取り換えられるかもしれません。皆さまの応用が皆さまの仕様に合致するのを保証するのは皆さまの責任です。Microchipはその条件、品質、性能、商品性、目的適合性を含め、明示的にも黙示的にもその情報に関連して書面または表記された書面または黙示の如何なる表明や保証もしません。Microchipはこの情報とそれの使用から生じる全責任を否認します。生命維持や安全応用でのMicrochipデバイスの使用は完全に購入者の危険性で、購入者はそのような使用に起因する全ての損害、請求、訴訟、費用からMicrochipを擁護し、補償し、免責にすることに同意します。他に言及されない限り、Microchipのどの知的財産権下でも暗黙的または違う方法で許認可は譲渡されません。

## 商標

Microchipの名前とロゴ、Mmicrochipロゴ、Adaptec、AnyRate、AVR、AVRロゴ、AVR Freaks、BesTime、BitCloud、chipKIT、chipKITロゴ、CryptoMemory、CryptoRF、dsPIC、FlashFlex、flexPWR、HELDO、IGLOO、JukeBlox、KeeLoq、Kleer、LANCheck、LinkMD、maXStylus、maXTouch、MediaLB、megaAVR、Microsemi、Microsemiロゴ、MOST、MOSTロゴ、MPLAB、OptoLyzer、PackeTime、PIC、picoPower、PICSTART、PIC32ロゴ、PolarFire、Prochip Designer、QTouch、SAM-BA、SenGenuity、SpyNIC、SST、SSTロゴ、SuperFlash、Symmetricom、SyncServer、Tachyon、TempTracker、TimeSource、tinyAVR、UNI/O、Vectron、XMEGAは米国と他の国に於けるMicrochip Technology Incorporatedの登録商標です。

APT、ClockWorks、The Embedded Control Solutions Company、EtherSynch、FlashTec、Hyper Speed Control、HyperLight Load、IntelliMOS、Liberio、motorBench、mTouch、Powermite 3、Precision Edge、ProASIC、ProASIC Plus、ProASIC Plusロゴ、Quiet-Wire、SmartFusion、SyncWorld、Temux、TimeCesium、TimeHub、TimePictra、TimeProvider、Vite、WinPath、ZLは米国に於けるMicrochip Technology Incorporatedの登録商標です。

Adjacent Key Suppression、AKS、Analog-for-the-Digital Age、Any Capacitor、AnyIn、AnyOut、BlueSky、BodyCom、CodeGuard、CryptoAuthentication、CryptoCompanion、CryptoController、dsPICDEM、dsPICDEM.net、Dynamic Average Matching、DAM、ECAN、EtherGREEN、In-Circuit Serial Programming、ICSP、INICnet、Inter-Chip Connectivity、JitterBlocker、KleerNet、KleerNetロゴ、memBrain、Mindi、MiWi、MPASM、MPF、MPLAB Certifiedロゴ、MPLAB、MPLINK、MultiTRAK、NetDetach、Omniscient Code Generation、PICDEM、PICDEM.net、PICkit、PICtail、PowerSmart、PureSilicon、QMatrix、REALICE、Ripple Blocker、SAM-ICE、Serial Quad I/O、SMART-I.S.、SQI、SuperSwitcher、SuperSwitcher II、Total Endurance、TSHARC、USBCheck、VariSense、View Sense、WiperLock、Wireless DNA、ZENAは米国と他の国に於けるMicrochip Technology Incorporatedの商標です。

SQTPは米国に於けるMicrochip Technology Incorporatedの役務標章です。

Adaptecロゴ、Frequency on Demand、Silicon Storage Technology、Symmcomは他の国に於けるMicrochip Technology Inc.の登録商標です。

GestICは他の国に於けるMicrochip Technology Inc.の子会社であるMicrochip Technology Germany II GmbH & Co. KGの登録商標です。

ここで言及した以外の全ての商標はそれら各々の会社の所有物です。

© 2020年、Microchip Technology Incorporated、米国印刷、不許複製

## 品質管理システム

Microchipの品質管理システムに関する情報については[www.microchip.com/quality](http://www.microchip.com/quality)を訪ねてください。

日本語© HERO 2020.

本技術概説はMicrochipのTB3256技術概説(DS90003256A-2020年7月)の翻訳日本語版です。日本語では不自然となる重複する形容表現は省略されている場合があります。日本語では難解となる表現は大幅に意識されている部分もあります。必要に応じて一部加筆されています。頁割の変更により、原本より頁数が少なくなっています。

必要と思われる部分には( )内に英語表記や略称などを残す形で表記しています。

青字の部分はリンクとなっています。一般的に赤字の0,1は論理0,1を表します。その他の赤字は重要な部分を表します。



**MICROCHIP**

## 世界的な販売とサービス

米国	亜細亜/太平洋	亜細亜/太平洋	欧州
<b>本社</b> 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 技術支援: <a href="http://www.microchip.com/support">www.microchip.com/support</a> ウェブアドレス: <a href="http://www.microchip.com">www.microchip.com</a> <b>アトランタ</b> Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 <b>オースチン TX</b> Tel: 512-257-3370 <b>ボストン</b> Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 <b>シカゴ</b> Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 <b>ダラス</b> Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 <b>デトロイト</b> Novi, MI Tel: 248-848-4000 <b>ヒューストン TX</b> Tel: 281-894-5983 <b>インディアナポリス</b> Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 <b>ロサンゼルス</b> Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 <b>ローリー NC</b> Tel: 919-844-7510 <b>ニューヨーク NY</b> Tel: 631-435-6000 <b>サンホセ CA</b> Tel: 408-735-9110 Tel: 408-436-4270 <b>カナダ - トロント</b> Tel: 905-695-1980 Fax: 905-695-2078	<b>オーストラリア - シドニー</b> Tel: 61-2-9868-6733 <b>中国 - 北京</b> Tel: 86-10-8569-7000 <b>中国 - 成都</b> Tel: 86-28-8665-5511 <b>中国 - 重慶</b> Tel: 86-23-8980-9588 <b>中国 - 東莞</b> Tel: 86-769-8702-9880 <b>中国 - 広州</b> Tel: 86-20-8755-8029 <b>中国 - 杭州</b> Tel: 86-571-8792-8115 <b>中国 - 香港特別行政区</b> Tel: 852-2943-5100 <b>中国 - 南京</b> Tel: 86-25-8473-2460 <b>中国 - 青島</b> Tel: 86-532-8502-7355 <b>中国 - 上海</b> Tel: 86-21-3326-8000 <b>中国 - 瀋陽</b> Tel: 86-24-2334-2829 <b>中国 - 深圳</b> Tel: 86-755-8864-2200 <b>中国 - 蘇州</b> Tel: 86-186-6233-1526 <b>中国 - 武漢</b> Tel: 86-27-5980-5300 <b>中国 - 西安</b> Tel: 86-29-8833-7252 <b>中国 - 廈門</b> Tel: 86-592-2388138 <b>中国 - 珠海</b> Tel: 86-756-3210040	<b>インド - ハンガロール</b> Tel: 91-80-3090-4444 <b>インド - ニューデリー</b> Tel: 91-11-4160-8631 <b>インド - フネー</b> Tel: 91-20-4121-0141 <b>日本 - 大阪</b> Tel: 81-6-6152-7160 <b>日本 - 東京</b> Tel: 81-3-6880-3770 <b>韓国 - 大邱</b> Tel: 82-53-744-4301 <b>韓国 - ソウル</b> Tel: 82-2-554-7200 <b>マレーシア - クアラルンプール</b> Tel: 60-3-7651-7906 <b>マレーシア - ペナン</b> Tel: 60-4-227-8870 <b>フィリピン - マニラ</b> Tel: 63-2-634-9065 <b>シンガポール</b> Tel: 65-6334-8870 <b>台湾 - 新竹</b> Tel: 886-3-577-8366 <b>台湾 - 高雄</b> Tel: 886-7-213-7830 <b>台湾 - 台北</b> Tel: 886-2-2508-8600 <b>タイ - バンコク</b> Tel: 66-2-694-1351 <b>ベトナム - ホーチミン</b> Tel: 84-28-5448-2100	<b>オーストラリア - ウェルズ</b> Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 <b>デンマーク - コペンハーゲン</b> Tel: 45-4485-5910 Fax: 45-4485-2829 <b>フィンランド - エスポー</b> Tel: 358-9-4520-820 <b>フランス - パリ</b> Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 <b>ドイツ - ガルヒング</b> Tel: 49-8931-9700 <b>ドイツ - ハーン</b> Tel: 49-2129-3766400 <b>ドイツ - ハイムブロン</b> Tel: 49-7131-72400 <b>ドイツ - カールスルーエ</b> Tel: 49-721-625370 <b>ドイツ - ミュンヘン</b> Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 <b>ドイツ - ローゼンハイム</b> Tel: 49-8031-354-560 <b>イスラエル - ラーナナ</b> Tel: 972-9-744-7705 <b>イタリア - ミラノ</b> Tel: 39-0331-742611 Fax: 39-0331-466781 <b>イタリア - ハドバ</b> Tel: 39-049-7625286 <b>オランダ - デルフト</b> Tel: 31-416-690399 Fax: 31-416-690340 <b>ノルウェー - トロンハイム</b> Tel: 47-72884388 <b>ポーランド - ワルシャワ</b> Tel: 48-22-3325737 <b>ルーマニア - ブカレスト</b> Tel: 40-21-407-87-50 <b>スペイン - マドリッド</b> Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 <b>スウェーデン - イェテボリ</b> Tel: 46-31-704-60-40 <b>スウェーデン - ストックホルム</b> Tel: 46-8-5090-4654 <b>イギリス - ウォーキングム</b> Tel: 44-118-921-5800 Fax: 44-118-921-5820