TB3245

変換、累積、起動事象に関する12ビットADCの使用

序説

著者: Stefan Vlad, Microchip Technology Inc.

ｱﾅﾛｸﾞ/ﾃﾞｼﾞﾀﾙ変換器(ADC)は差動とｼﾝｸﾞﾙ ｴﾝﾄﾞの変換が可能な12ﾋﾞｯﾄ逐次比較変換器(SAR:Successive Approximation Register)です。この周辺機能はAVR® DA MCU (AVR DA)ﾏｲｸﾛ ｺﾝﾄﾛｰﾗで利用可能です。

この技術概説はAVR DAﾏｲｸﾛ ｺﾝﾄﾛｰﾗでADC単位部を動かす方法を記述し、以下の使用事例を網羅します。

- **ADC単独変換**:

  ADCを初期化し、変換を開始してそれが完了されるまで待ち、そして繰り返し内でADC結果を読みます。

- **ADC自由走行変換**:

  ADCを初期化し、自由走行動作を許可し、変換を開始してそれが完了されるまで待ち、そして無限繰り返し内でADC結果を読みます。

- **ADC差動変換**:

  ADCを初期化し、差動電圧を読むために2つのﾋﾟﾝを構成設定し、変換を開始してそれが完了されるまで待ち、そして無限繰り返し内でADC結果を読みます。

- **ADC採取累積器**:

  ADCを初期化し、64採取累積を許可し、変換を開始してそれが完了されるまで待ち、そして繰り返し内でADC結果を読みます。

- **ADC窓比較器**:

  ADCを初期化し、変換窓比較器下位閾値を設定し、変換窓動作を許可し、自由走行動作を許可し、変換を開始してそれが完了されるまで待ち、そして無限繰り返し内でADC結果を読みます。ADC結果が設定した閾値未満の場合にLEDがONに切り替えられます。

- **事象起動ADC**:

  ADCを初期化し、実時間計数器(RTC:Real-Time Counter)を初期化し、RTC溢れでADC変換を起動するように事象ｼｽﾃﾑ(EVSYS)を構成設定します。各ADC変換後にLEDがONに切り替えられます。

- **ADC温度測定**:

  ADCを初期化し、内部参照基準を選び、入力として温度感知器を選んで12ﾋﾞｯﾄ、右揃え、ｼﾝｸﾞﾙ ｴﾝﾄﾞ変換を走行することによってﾃﾞｰﾀを採取します。

全ての例に対するADC結果は万能同期/非同期送受信器(USART:Universal Syncronous and Asyncronous Rceiver-Transmitter)を通して送信され、ﾃﾞｰﾀ可視器(Data Visualizer)ﾂｰﾙを使って作図されます。

注: ｺｰﾄﾞ例はAVR128DA48 Curiosity Nanoで開発されました。これらはGitHubで入手可能で、Atmel StudioとMPLAB® Xの両統合開発環境(IDE:Integrated Development Environment)を支援します。

本書は一般の方々の便宜のため有志により作成されたもので、Microchip社とは無関係であることを御承知ください。しおりの[はじめに]での内容にご注意ください。

目次

序説 ····· 1
1. 関連ﾃﾞﾊﾞｲｽ ····· 3
1.1. AVR® DA系概要 ····· 3
2. 概要 ····· 3
3. ﾊｰﾄﾞｳｪｱ構成設定 ····· 4
4. ADC単独変換 ····· 5
4.1. ADC初期化 ····· 5
4.2. 結果送信 ····· 7
4.3. ｺｰﾄﾞ例 ····· 9
5. ADC自由走行変換 ····· 9
5.1. ADC初期化 ····· 9
5.2. 結果送信 ····· 9
5.3. ｺｰﾄﾞ例 ····· 10
6. ADC差動変換 ····· 10
6.1. ADC初期化 ····· 10
6.2. 結果送信 ····· 11
6.3. ｺｰﾄﾞ例 ····· 11
7. ADC採取累積器 ····· 11
7.1. ADC初期化 ····· 11
7.2. 結果送信 ····· 12
7.3. ｺｰﾄﾞ例 ····· 12
8. ADC窓比較器 ····· 12
8.1. ADC初期化 ····· 12
8.2. 結果送信 ····· 13
8.3. ｺｰﾄﾞ例 ····· 14
9. ADC事象起動 ····· 14
9.1. ADC初期化 ····· 14
9.2. 結果送信 ····· 15
9.3. ｺｰﾄﾞ例 ····· 15
10. ADC温度測定 ····· 15
10.1. ADC初期化 ····· 15
10.2. 結果送信 ····· 16
10.3. ｺｰﾄﾞ例 ····· 16
11. ﾃﾞｰﾀ可視器 ····· 17
12. 参考文献 ····· 22
13. 追補 ····· 23
14. 改訂履歴 ····· 39
Microchipｳｪﾌﾞ ｻｲﾄ ····· 40
製品変更通知ｻｰﾋﾞｽ ····· 40
お客様支援 ····· 40
Microchipﾃﾞﾊﾞｲｽ ｺｰﾄﾞ保護機能 ····· 40
法的通知 ····· 40
商標 ····· 41
品質管理ｼｽﾃﾑ ····· 41
世界的な販売とｻｰﾋﾞｽ ····· 42

© 2020 Microchip Technology Inc.

1. 関連ﾃﾞﾊﾞｲｽ

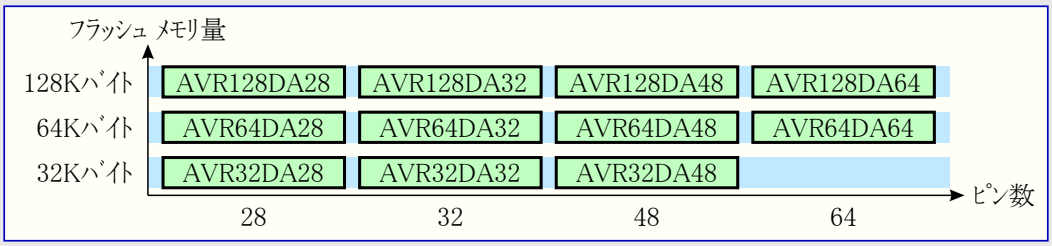
本章はこの文書に関連するﾃﾞﾊﾞｲｽを一覧にします。

1.1. AVR® DA系概要

下図はﾋﾟﾝ数の変種とﾒﾓﾘ量を展開してAVR® DAﾃﾞﾊﾞｲｽを示します。

- これらのﾃﾞﾊﾞｲｽが完全にﾋﾟﾝと機能が互換のため、垂直方向移植はｺｰﾄﾞ変更なしで可能です。
- 左への水平方向移植はﾋﾟﾝ数、従って利用可能な機能を減らします。

図1-1. AVR® DA系概要



異なるﾌﾗｯｼｭ ﾒﾓﾘ量を持つﾃﾞﾊﾞｲｽは一般的にSRAM量も異なります。

2. 概要

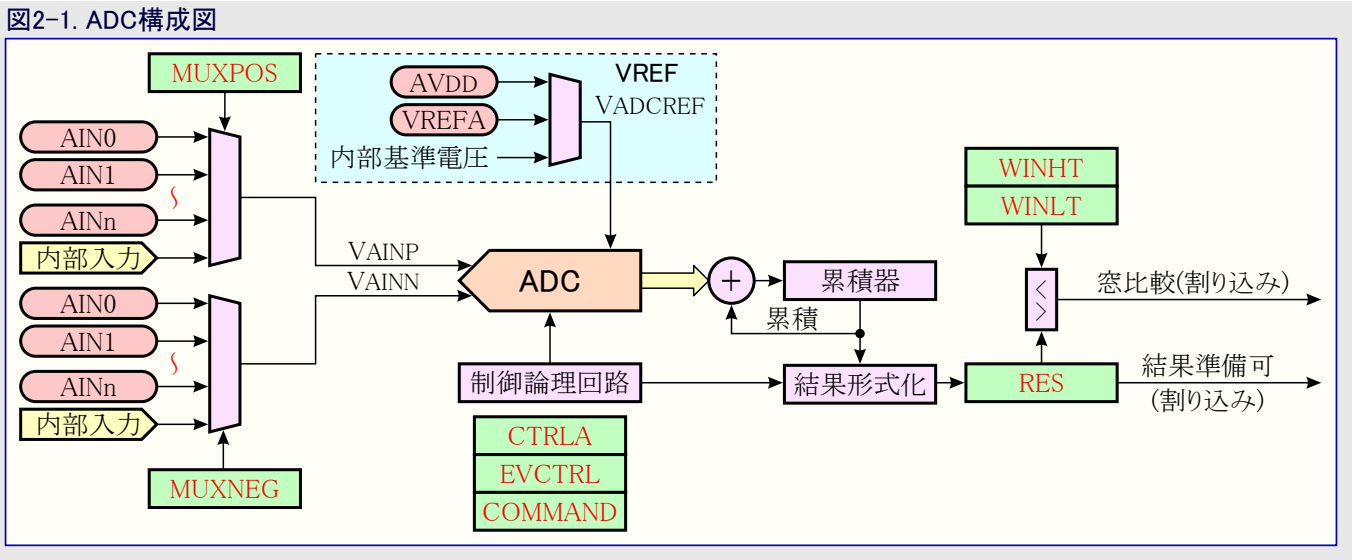
ADCは12ﾋﾞｯﾄ分解能で130kspsまでの採取速度を持つSARです。ADCは多数のｼﾝｸﾞﾙ ｴﾝﾄﾞまたは差動の入力間選択のためのｱﾅﾛｸﾞ入力多重器に接続されます。ｼﾝｸﾞﾙ ｴﾝﾄﾞ変換でADCは選んだ入力ﾁｬﾈﾙと0V(GND)間の電圧を測定します。差動変換では選んだ2つの入力ﾁｬﾈﾙ間の電圧を測定します。選んだADC入力ﾁｬﾈﾙは内部(例えば、電圧参照基準)または外部ｱﾅﾛｸﾞ入力ﾋﾟﾝのどちらかにすることができます。

ADCは構成設定可能な数の変換結果が単一の変換結果に累積される集中での採取、または試料累積を支援します。ADC入力信号は採取中にADCの入力電圧が一定水準で保たれることを保証する採取/保持(S/H)回路を通して供給されます。

内部基準電圧(VREF)周辺機能、AVDD供給電圧、または外部VREF(VREFA)ﾋﾟﾝから選択可能な基準電圧。

ﾃﾞｼﾞﾀﾙ窓比較機能は入力信号を監視するのに利用可能で、必要な最小のｿﾌﾄｳｪｱ介在で、試料が使用者定義された閾値の下(未満)、上(超え)の場合、または使用者定義された窓の内側、外側の場合にだけ割り込みを起動するように構成設定することができます。

図2-1. ADC構成図



ｱﾅﾛｸﾞ入力ﾁｬﾈﾙは多重器正選択(ADCn.MUXPOS)ﾚｼﾞｽﾀの多重器正選択(MUXPOS)ﾋﾞｯﾄ領域に書くことによって選ばれます。ADCへのｼﾝｸﾞﾙ ｴﾝﾄﾞ入力としてADC入力ﾋﾟﾝのどれか、GND、内部入力、または温度感知器を選ぶことができます。ADCは制御A(ADCn.CTRLA)ﾚｼﾞｽﾀのADC許可(ENABLE)ﾋﾞｯﾄに'1'を書くことによって許可されます。電圧参照基準と入力ﾁｬﾈﾙ選択はADCが許可される前に有効になりません。

ADCはENABLEﾋﾞｯﾄが'0'の時に電力を消費しません。ADCは結果(ADCn.RES)ﾚｼﾞｽﾀから読むことができる10または12ﾋﾞｯﾄの右揃えの結果を生成します。

変換式が次に一覧され、ここでのVAINPとVAINNは正と負のADC入力、VADCREFは選んだADC電圧参照基準です。

- ｼﾝｸﾞﾙ ｴﾝﾄﾞ12ﾋﾞｯﾄ変換：$RES = \frac{VAINP}{VADCREF} \times 4096 \in [0,4095]$

- シングル エンド10ビット変換：$RES = \frac{VAINP}{VADCREF} \times 1024 \in [0,1023]$
- 差動12ビット変換　　：$RES = \frac{VAINP-VAINN}{VADCREF} \times 2048 \in [-2048,2047]$
- 差動10ビット変換　　：$RES = \frac{VAINP-VAINN}{VADCREF} \times 512 \in [-512,511]$

3. ハードウェア構成設定

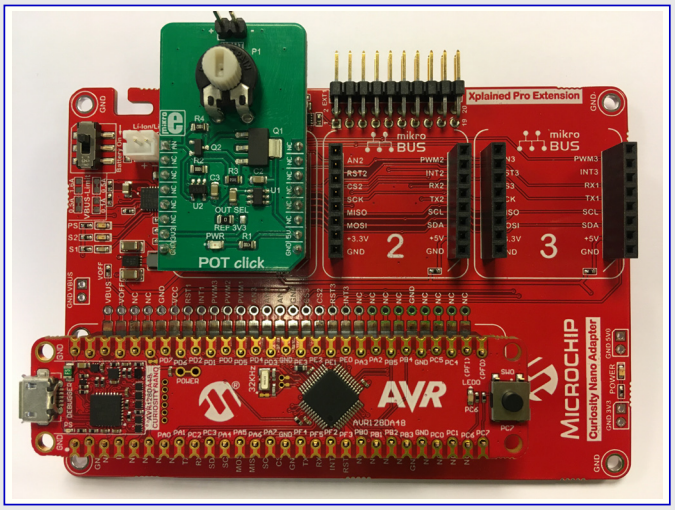
この例に使われるハードウェアは以下に基づきます。

- AVR128DA48 Curiosity Nano (DM164151)
- Click基板用Curiosity Nano BASE (AC164162)
- mikroBus™ POT Click基板 (MIKRO-3402) または
- mikroBus™ POT 2 Click基板 (MIKROE-3325)

この文書で記述される使用事例のために以下の2つの設定が使われます。

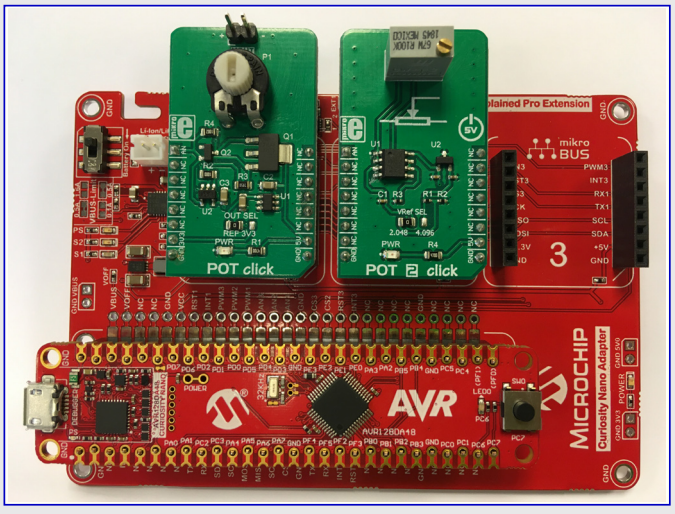
- 最初の構成設定(構成A)はmikroBUSスロット1に挿入されるPOT Clickを使い、AN1をAVR DAデバイスのPD3(AIN3)ピンに接続してください。

図3-1. 1つのPOT Clickでのハードウェア構成A



- 2つ目の構成設定(構成B)はmikroBUSストット1とスロット2に挿入される2つのPOT Clickを使います。スロット1はAN1をAVR DAデバイスのPD3(AIN3)ピンに接続し、スロット2はAN2をAVR DAデバイスのPD4(AIN4)ピンに接続します。

図3-2. 2つのPOT Clickでのハードウェア構成B



4. ADC単独変換

4.1. ADC初期化

この副項はADC単位部を単独変換動作で動くように初期化する方法を披露します。ADC入力ﾋﾟﾝは可能な最高の入力ｲﾝﾋﾟｰﾀﾞﾝｽを持つように、禁止されたﾃﾞｼﾞﾀﾙ入力緩衝部とﾌﾟﾙｱｯﾌﾟ抵抗を持つことが必要です。

ADC単独変換は「3. ﾊｰﾄﾞｳｪｱ構成設定」で説明された構成A設定を使います。

図4-1. ADC0.MUXPOS選択

ﾋﾞｯﾄ	7	6	5	4	3	2	1	0
		MUXPOS6~0						
ｱｸｾｽ種別	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ﾘｾｯﾄ値	0	0	0	0	0	0	0	0

● **ﾋﾞｯﾄ6~0 – MUXPOS6~0 : 多重器正選択** (MUX Selection for Positive ADC Input)

このﾋﾞｯﾄ領域はどのｱﾅﾛｸﾞ入力がADCの正入力に接続されるかを選びます。このﾋﾞｯﾄ領域が変換中に変えられる場合、その変更はこの変換が完了するまで無効です。

値	$00～$15	$40	$42	$48	$49	$4A	$4B	その他
名称	AIN0～AIN21	GND	TEMPSENSE	DAC0	DACREF0	DACREF1	DACREF2	–
説明	ADC入力0～21ﾋﾟﾝ	接地	温度感知器	DAC0	DACREF0	DACREF1	DACREF2	(予約)

ADCﾁｬﾈﾙAIN3(PD3)を選ぶには下のｺｰﾄﾞを使ってください。

```
ADC0.MUXPOS = ADC_MUXPOS_AIN3_gc;
```

ADCｸﾛｯｸ前置分周器はｸﾛｯｸ周波数を分周するのに使うことができます。この例ではｸﾛｯｸが4分周されます。

```
ADC0.CTRLC = ADC_PRESC_DIV4_gc;
```

図4-2. ADC0参照基準電圧選択

ﾋﾞｯﾄ	7	6	5	4	3	2	1	0
	ALWAYSON					REFSEL2~0		
ｱｸｾｽ種別	R/W	R	R	R	R	R/W	R/W	R/W
ﾘｾｯﾄ値	0	0	0	0	0	0	0	0

● **ﾋﾞｯﾄ7 – ALWAYSON : 参照基準常時ON** (Reference Always On)

このﾋﾞｯﾄはADC0の参照基準が常時ONか否かを制御します。

値	0	1
説明	参照基準は必要とされる時に自動的に許可されます。	参照基準は常にONです。

● **ﾋﾞｯﾄ2~0 – REFSEL2~0 : 基準電圧選択** (Reference Select)

このﾋﾞｯﾄ領域はADCに対する参照基準電圧水準を制御します。

値	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
名称	1V024	2V048	4V096	2V500	–	VDD	VREFA	–
説明	内部1.024V参照基準(注)	内部2.048V参照基準(注)	内部4.096V参照基準(注)	内部2.500V参照基準(注)	(予約)	参照基準としてVDD	VREFAﾋﾟﾝからの外部参照基準	(予約)

注: 内部参照基準に対して与えられる値は代表値なだけです。更なる詳細については「電気的特性」章を参照してください。

ADCは正の参照基準に対してVDD、外部または内部の参照基準を使うことができます。この例では内部参照基準が使われます。

```
VREF.ADC0REF = VREF_REFSEL_2V048_gc;                /* 内部2.048V参照基準 */
```

ADC分解能は制御A(ADC0.CTRLA)ﾚｼﾞｽﾀの分解能選択(RESSEL)ﾋﾞｯﾄによって設定されます。ADCはADC0.CTRLAﾚｼﾞｽﾀのADC許可(ENABLE)を設定(1)することによって許可されます。

© 2020 Microchip Technology Inc.

図4-3. ADC0.CTRLA分解能選択

ﾋﾞｯﾄ	7	6	5	4	3	2	1	0
	RUNSTDBY		CONVMODE	LEFTADJ	RESSEL1,0		FREERUN	ENABLE
ｱｸｾｽ種別	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W
ﾘｾｯﾄ値	0	0	0	0	0	0	0	0

● **ﾋﾞｯﾄ3,2 – RESSEL1,0 : 分解能選択** (Resolution Selection)

このﾋﾞｯﾄ領域はADC分解能を選びます。分機能を12ﾋﾞｯﾄから10ﾋﾞｯﾄに変えると、変換時間は13.5 CLK_ADC周期から11.5 CLK_ADC周期に減らされます。

値	0 0	0 1	その他
説明	12ﾋﾞｯﾄ分解能	10ﾋﾞｯﾄ分解能	(予約)

これは次のｺｰﾄﾞになります。

```
ADC0.CTRLA = ADC_ENABLE_bm              /* ADC許可: 許可 */
           | ADC_RESSEL_12BIT_gc;       /* 12ﾋﾞｯﾄ動作 */
```

ADC変換は指令(ADC0.COMMAND)ﾚｼﾞｽﾀの変換開始(STCONV)ﾋﾞｯﾄを設定(1)することによって開始されます。

図4-4. ADC0.COMMAND – 変換開始

ﾋﾞｯﾄ	7	6	5	4	3	2	1	0
							SPCONV	STCONV
ｱｸｾｽ種別	R	R	R	R	R	R	R/W	R/W
ﾘｾｯﾄ値	0	0	0	0	0	0	0	0

● **ﾋﾞｯﾄ1 – SPCONV : 変換停止** (Stop Conversion)

このﾋﾞｯﾄへの'1'書き込みは現在の測定を終えます。このﾋﾞｯﾄは変換開始(STCONV)ﾋﾞｯﾄに優先します。

● **ﾋﾞｯﾄ0 – STCONV : 変換開始** (Start Conversion)

このﾋﾞｯﾄへの'1'書き込みは進行中のどの変換も完了されて直ぐに変換を開始します。自由走行動作の場合、これは初回変換を開始します。STCONVは変換が進行中である限り'1'として読みます。変換が完了すると、このﾋﾞｯﾄは自動的に解除(0)されます。このﾋﾞｯﾄへの'0'書き込みは無効です。

これは次のｺｰﾄﾞになります。

```
ADC0.COMMAND = ADC_STCONV_bm;
```

変換が終わると、ﾊｰﾄﾞｳｪｱによって割り込み要求ﾌﾗｸﾞ(ADC0.INTFLAGS)ﾚｼﾞｽﾀの結果準備可割り込み要求ﾌﾗｸﾞ(RESRDY)ﾋﾞｯﾄが設定(1)されます。

図4-5. ADC0.INTFLAGS – ﾊｰﾄﾞｳｪｱ設定RESRDYﾋﾞｯﾄ

ﾋﾞｯﾄ	7	6	5	4	3	2	1	0
							WCOMP	RESRDY
ｱｸｾｽ種別	R	R	R	R	R	R	R/W	R/W
ﾘｾｯﾄ値	0	0	0	0	0	0	0	0

● **ﾋﾞｯﾄ1 – WCOMP : 窓比較器割り込み要求ﾌﾗｸﾞ** (Window Comparator Interrupt Flag)

この窓比較器割り込み要求ﾌﾗｸﾞは測定が完了して結果が制御E(ADCn.CTRLE)ﾚｼﾞｽﾀの窓比較器動作(WINCM)によって定義される選んだ窓比較器動作に合致した場合に設定(1)されます。比較は変換の最後で行われます。このﾌﾗｸﾞはこのﾋﾞｯﾄ位置への'1'書き込み、または結果(ADCn.RES)ﾚｼﾞｽﾀ読み込みのどちらかによって解除(0)されます。このﾋﾞｯﾄへの'0'書き込みは無効です。

● **ﾋﾞｯﾄ0 – RESRDY : 結果準備可割り込み要求ﾌﾗｸﾞ** (Result Ready Interrupt Flag)

結果準備可割り込み要求ﾌﾗｸﾞは測定が完了して新しい結果の準備が整った時に設定(1)されます。このﾌﾗｸﾞはこのﾋﾞｯﾄ位置への'1'書き込み、または結果(ADCn.RES)ﾚｼﾞｽﾀ読み込みのどちらかによって解除(0)されます。このﾋﾞｯﾄへの'0'書き込みは無効です。

© 2020 Microchip Technology Inc.

使用者は別の変換を開始する前に結果(ADC0.RES)ﾚｼﾞｽﾀを読むことによって結果準備可割り込み要求ﾌﾗｸﾞ(RESRDY)ﾋﾞｯﾄを解除(0)しなければなりません。

```
/* 結果読み込みによって割り込み要求ﾌﾗｸﾞを解除(0) */
return ADC0.RES;
```

変換結果は結果(ADC0.RES)ﾚｼﾞｽﾀから読むことができます。

```
adcVal = ADC0.RES;
```

4.2. 結果送信

ﾃﾞｰﾀ可視器(Data Visualizer)ﾌﾟﾗｸﾞｲﾝを使ってADC結果を可視化するため、使用者は先にUSARTを通して結果を送らなければなりません。ﾎﾞｰﾚｰﾄを構成設定するのにUSARTのﾎﾞｰﾚｰﾄ(BAUD)ﾚｼﾞｽﾀが使われなければなりません。これは下図で提示されます。

図4-6. USARTﾎﾞｰﾚｰﾄ ﾚｼﾞｽﾀ

名称 : BAUD (BAUDH,BAUDL)
変位 : +$08
ﾘｾｯﾄ : $0000
特質 : -

USARTn.BAUDHとUSARTn.BAUDLのﾚｼﾞｽﾀ対は16ﾋﾞｯﾄ値のUSARTn.BAUDを表します。下位ﾊﾞｲﾄ[7～0](接尾辞L)は変位原点でｱｸｾｽ可能です。上位ﾊﾞｲﾄ[15～8](接尾辞H)は変位+1でｱｸｾｽすることができます。

送信部と受信部の進行中の転送はﾎﾞｰﾚｰﾄが変更される場合に不正にされます。このﾚｼﾞｽﾀへの書き込みはﾎﾞｰﾚｰﾄ前置分周器の即時更新を起動します。ﾎﾞｰﾚｰﾄの設定方法のより多くの情報については**表25-1. ﾎﾞｰﾚｰﾄ ﾚｼﾞｽﾀ設定計算用の式**をご覧ください。

ﾋﾞｯﾄ	15	14	13	12	11	10	9	8
	BAUD15~8							
ｱｸｾｽ種別	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ﾘｾｯﾄ値	0	0	0	0	0	0	0	0

ﾋﾞｯﾄ	7	6	5	4	3	2	1	0
	BAUD7~0							
ｱｸｾｽ種別	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ﾘｾｯﾄ値	0	0	0	0	0	0	0	0

● **ﾋﾞｯﾄ15~8 - BAUD15~8 : ﾎﾞｰﾚｰﾄ上位ﾊﾞｲﾄ** (USART Baud Rate high byte)
これらのﾋﾞｯﾄは16ﾋﾞｯﾄ ﾎﾞｰﾚｰﾄ ﾚｼﾞｽﾀの上位ﾊﾞｲﾄを保持します。

● **ﾋﾞｯﾄ7~0 - BAUD7~0 : ﾎﾞｰﾚｰﾄ下位ﾊﾞｲﾄ** (USART Baud Rate low byte)
これらのﾋﾞｯﾄは16ﾋﾞｯﾄ ﾎﾞｰﾚｰﾄ ﾚｼﾞｽﾀの下位ﾊﾞｲﾄを保持します。

以下のｺｰﾄﾞが使われなければなりません。

```
USART1.BAUD = (uint16_t)USART1_BAUD_RATE(115200);
```

USART_BAUD_RATEﾏｸﾛはBAUDﾚｼﾞｽﾀに対応する値を計算するのに使われます。これは下で提示されます。

```
#define USART1_BAUD_RATE(BAUD_RATE) (((float)F_CPU*64/(16 *(float)BAUD_RATE))+0.5)
```

USART送信は下で提示される制御B(CTRLB)ﾚｼﾞｽﾀを使って許可することができます。

図4-7. 制御Bﾚｼﾞｽﾀ

名称 : CTRLB
変位 : +$06
ﾘｾｯﾄ : $00
特質 : -

ﾋﾞｯﾄ	7	6	5	4	3	2	1	0
	RXEN	TXEN		SFDEN	ODME	RXMODE1,0		MPCM
ｱｸｾｽ種別	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
ﾘｾｯﾄ値	0	0	0	0	0	0	0	0

● **ﾋﾞｯﾄ6 - TXEN : 送信許可** (Transmitter Enable)
このﾋﾞｯﾄはUSART送信部が許可されるか否かを制御します。より多くの情報については「25.3.2.3.1. 送信部禁止」項を参照してください。

値	0	1
説明	RUSART送信部が禁止されます。	USART送信部が許可されます。

以下のｺｰﾄﾞが使われなければなりません。

```
USART1.CTRLB = USART_TXEN_bm;                    /* 送信部許可 */
```

文字の大きさは制御C(CTRLC)ﾚｼﾞｽﾀを使って構成設定することができます。

図4-8. 制御Cﾚｼﾞｽﾀ

名称 ：CTRLC
変位 ：+$07
ﾘｾｯﾄ ：$03
特質 ：-

このﾚｼﾞｽﾀ記述は主装置SPI動作を除く全動作に対して有効です。このﾚｼﾞｽﾀのUSART通信動作(CMODE)ﾋﾞｯﾄが'MSPI'を書かれた時の正確な記述については「制御C(CTRLC) - 主装置SPI動作」ﾚｼﾞｽﾀをご覧ください。

ﾋﾞｯﾄ	7	6	5	4	3	2	1	0
	CMODE1,0		PMODE1,0		SBMODE		CHSIZE2~0	
ｱｸｾｽ種別	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ﾘｾｯﾄ値	0	0	0	0	0	0	1	1

● **ﾋﾞｯﾄ2~0 - CHSIZE2~0 : 文字ﾋﾞｯﾄ数** (Character Size)

これらのﾋﾞｯﾄはﾌﾚｰﾑ内のﾃﾞｰﾀ ﾋﾞｯﾄ数を選びます。受信部と送信部は同じ設定を使います。9ﾋﾞｯﾄ文字に対しては、受信ﾃﾞｰﾀ(RXDATA)または送信ﾃﾞｰﾀ(TXDATA)の下位または上位で先に読み書きするﾊﾞｲﾄ順を構成設定することができます。

値	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
名称	5BIT	6BIT	7BIT	8BIT	-	-	9BITL	9BITH
説明	5ﾋﾞｯﾄ	6ﾋﾞｯﾄ	7ﾋﾞｯﾄ	8ﾋﾞｯﾄ	(予約)	(予約)	9ﾋﾞｯﾄ(下位ﾊﾞｲﾄ先行)	9ﾋﾞｯﾄ(上位ﾊﾞｲﾄ先行)

以下のｺｰﾄﾞが使われなければなりません。

```
USART1.CTRLC = USART_CHSIZE_8BIT_gc;             /* 文字の大きさを8ﾋﾞｯﾄに構成設定 */
```

USARTを通して1ﾊﾞｲﾄを送信するため、使用者はUSART緩衝部が送信する準備が整っているかを調べ、その後にﾃﾞｰﾀを送信ﾃﾞｰﾀ下位ﾊﾞｲﾄ(TXDATAL)ﾚｼﾞｽﾀに書かなければなりません。

ﾃﾞｰﾀ可視器(Data Visualizer)によってﾃﾞｰﾀが正しく解釈されるため、指定形式に従わなければなりません。12ﾋﾞｯﾄ分機能のADC結果は(符号ﾋﾞｯﾄが拡張された)16ﾋﾞｯﾄ形式を使って提供され、結果の2ﾊﾞｲﾄは下で記述されるように送信されます。

図4-9. ADC結果送信形式

フレーム開始通票(1ﾊﾞｲﾄ)	ADC 12ﾋﾞｯﾄ結果下位(1ﾊﾞｲﾄ)	ADC 12ﾋﾞｯﾄ結果上位(1ﾊﾞｲﾄ)	フレーム終了通票(1ﾊﾞｲﾄ)

ﾌﾚｰﾑ開始通票(START_TOKEN)とﾌﾚｰﾑ終了通票(END_TOKEN)のﾏｸﾛは下で記述されるように定義されます。

```
#define START_TOKEN 0x03                         /* フレーム開始通票 */
#define END_TOKEN 0xFC                           /* フレーム終了通票 */
```

無限繰り返しの内側に於いてADC結果が読まれ、指定形式を使ってUSARTを通して送信され、変換が再び開始されます。

USARTを通してADC結果を送信するために以下のｺｰﾄﾞが主無限繰り返しに追加されなければなりません。

```
while (1)
{
    /* ADC変換開始 */
    ADC0_start();

    /* ADC結果読み込み */
    adcVal = ADC0_read();

    /* ﾃﾞｰﾀ可視器を使って作図されるべきADC結果を送信 */
    USART1_Write(START_TOKEN);
    USART1_Write(adcVal &amp; 0x00FF);
```

```
    USART1_Write(adcVal >> 8);
    USART1_Write(END_TOKEN);
}
```

4.3. コード例

Atmel Studioを使って開発したコード例は下で入手可能です。

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。

MPLAB X IDEを使って開発したコード例は下で入手可能です。

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。

助言: 完全なコード例は「追補」で利用可能です。

5. ADC自由走行変換

5.1. ADC初期化

この副項はADC単位部を自由走行変換動作で動くように初期化する方法を提示します。この応用は「3. ハードウェア構成設定」で説明した構成A設定を使います。この動作を有効にするには標準ADC初期化に加えて、制御A(ADC0.CTRLA)の自由走行(FREERUN)ビットが設定(1)されなければなりません。ADCを自由走行動作で構成設定すると、直前の変換後、次の変換が直ちに開始します。

図5-1. ADC0.CTRLA – FREERUNビット設定

ビット	7	6	5	4	3	2	1	0
	RUNSTDBY		CONVMODE	LEFTADJ	RESSEL1,0		FREERUN	ENABLE
アクセス種別	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W
リセット値	0	0	0	0	0	0	0	0

● **ビット1 – FREERUN : 自由走行** (Free Running)

このビットへの'1'書き込みはデータ取得に関して自由走行動作を許可します。最初の変換は指令(ADCn.COMMAND)レジスタの変換開始(STCONV)ビットへの'1'書き込みによって始められます。自由走行動作では、直前の変換周回が完了された直後または直ぐに新しい変換が開始されます。これは割り込み要求フラグ(ADCn.INTFLAGS)レジスタの結果準備可割り込み要求(RESRDY)フラグによって合図されます。

5.2. 結果送信

ADC変換結果を送信するため、無限繰り返し内に以下の算法が実装されなければなりません。

```
while (1)
{
    /* ADC変換開始 */
    ADC0_start();

    /* ADC結果読み込み */
    adcVal = ADC0_read();

    /* データ可視器を使って作図されるべきADC結果を送信 */
    USART1_Write(START_TOKEN);
    USART1_Write(adcVal &amp; 0x00FF);
    USART1_Write(adcVal >> 8);
    USART1_Write(END_TOKEN);
}
```

5.3. ｺｰﾄﾞ例

Atmel Studioを使って開発したｺｰﾄﾞ例は下で入手可能です。

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

MPLAB X IDEを使って開発したｺｰﾄﾞ例は下で入手可能です。

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

助言: 完全なｺｰﾄﾞ例は「追補」で利用可能です。

6. ADC差動変換

6.1. ADC初期化

この副項はADC単位部を差動変換動作で動くように初期化する方法を提示します。この応用は「3. ﾊｰﾄﾞｳｪｱ構成設定」で説明した構成B設定を使います。

ADC差動変換動作は制御A(ADC0.CTRLA)ﾚｼﾞｽﾀの変換動作(CONVMODE)ﾋﾞｯﾄを設定(1)することによって許可されます。

図6-1. ADC0.CTRLA – CONVMODEﾋﾞｯﾄ設定

ﾋﾞｯﾄ	7	6	5	4	3	2	1	0
	RUNSTDBY		CONVMODE	LEFTADJ	RESSEL1,0		FREERUN	ENABLE
ｱｸｾｽ種別	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W
ﾘｾｯﾄ値	0	0	0	0	0	0	0	0

● **ﾋﾞｯﾄ5 – CONVMODE : 変換動作** (Conversion Mode)

このﾋﾞｯﾄはADCがｼﾝｸﾞﾙ ｴﾝﾄﾞ動作または差動動作のどちらで動くかを定義します。

値	0	1
名称	SINGLEENDED	DIFF
説明	ADCは正入力だけが使われるｼﾝｸﾞﾙ ｴﾝﾄﾞ動作で動きます。ADC結果は符号なし値として表されます。	ADCは正と負の両入力が使われる差動動作で動きます。ADC結果は符号付き値として表されます。

これは次のｺｰﾄﾞになります。

```
ADC0.CTRLA |= ADC_CONVMODE_bm;
```

差動変換については、POT ClickがAVR DAﾃﾞﾊﾞｲｽのPD3(AIN3)ﾋﾟﾝに接続され、POT 2がPD4(AIN4)ﾋﾟﾝに接続されます。

図6-2. 負のADC入力用多重器選択

ﾋﾞｯﾄ	7	6	5	4	3	2	1	0
		MUXNEG6~0						
ｱｸｾｽ種別	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ﾘｾｯﾄ値	0	0	0	0	0	0	0	0

● **ﾋﾞｯﾄ6~0 – MUXNEG6~0 : 多重器負選択** (MUX Selection for Negative ADC Input)

このﾋﾞｯﾄ領域はどのｱﾅﾛｸﾞ入力がADCの負入力に接続されるかを選びます。このﾋﾞｯﾄ領域が変換中に変えられる場合、その変更はこの変換が完了するまで無効です。

値	$00～$0F	$40	$48	その他
名称	AIN0～AIN15	GND	DAC0	–
説明	ADC入力0～15ﾋﾟﾝ	接地	DAC0	(予約)

これは次のｺｰﾄﾞになります。

```
ADC0.MUXNEG = ADC_MUXNEG_AIN4_gc;
```

© 2020 Microchip Technology Inc.

その後、ADCの結果はwhile繰り返しで読むことができます。

6.2. 結果送信

ADC差動変換結果は下で一覧にするｺｰﾄﾞで提示されるように、読んで送信することができます。

```
while (1)
{
    if (ADC0_conversionDone())
    {
        /* ADC結果読み込み */
        adcVal = ADC0_read();
        /* ﾃﾞｰﾀ可視器を使って作図されるべきADC結果を送信 */
        USART1_Write(START_TOKEN);
        USART1_Write(adcVal &amp; 0x00FF);
        USART1_Write(adcVal >> 8);
        USART1_Write(END_TOKEN);
    }
}
```

6.3. ｺｰﾄﾞ例

Atmel Studioを使って開発したｺｰﾄﾞ例は下で入手可能です。

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

MPLAB X IDEを使って開発したｺｰﾄﾞ例は下で入手可能です。

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

助言: 完全なｺｰﾄﾞ例は「追補」で利用可能です。

7. ADC採取累積器

7.1. ADC初期化

この副項はADC単位部を採取累積器機能を使うように初期化する方法を提示します。この応用は「3. ﾊｰﾄﾞｳｪｱ構成設定」で説明した構成A設定を使います。この動作でADCは累積ﾚｼﾞｽﾀで128採取までを加算することができ、従って信号の濾過と雑音の減少は滑らかな信号が必要とされるｱﾅﾛｸﾞ感知器ﾃﾞｰﾀ読み取り時に有用です。ｿﾌﾄｳｪｱでのこれらの読み取り加算の代わりにﾊｰﾄﾞｳｪｱ累積器を使うことにより、CPU負荷を減らします。

この動作を有効にするには標準ADC初期化に加えて、制御B(ADC0.CTRLB)ﾚｼﾞｽﾀの採取累積数(SAMPNUM)が設定されなければなりません。

図7-1. ADC0.CTRLB – SAMPNUMﾋﾞｯﾄ領域設定

ﾋﾞｯﾄ	7	6	5	4	3	2	1	0
						SAMPNUM2~0		
ｱｸｾｽ種別	R	R	R	R	R	R/W	R/W	R/W
ﾘｾｯﾄ値	0	0	0	0	0	0	0	0

● **ﾋﾞｯﾄ2~0 – SAMPNUM2~0 : 採取累積数選択** (Sample Accumulation Select)

これらのﾋﾞｯﾄは自動的にどれくらいの連続するADC採取結果が累積されるかを選びます。このﾋﾞｯﾄが0よりも大きな値を書かれると、1回の完全な変換で対応する数の連続するADC採取結果がADC結果(ADCn.RES)ﾚｼﾞｽﾀに累積されます。

値	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
名称	NONE	ACC2	ACC4	ACC8	ACC16	ACC32	ACC64	ACC128
説明	累積なし	2回の累積	4回の累積	8回の累積	16回の累積	32回の累積	64回の累積	128回の累積

これは次のｺｰﾄﾞに変換されます。

```
ADC0.CTRLB = ADC_SAMPNUM_ACC64_gc;
```

採取(試料)は結果(ADCn.RES)ﾚｼﾞｽﾀで合計されます。制御B(ADC0.CTRLB)ﾚｼﾞｽﾀで指定された採取数が取得された後に割り込み要求ﾌﾗｸﾞ(ADC0.INTFLAGS)ﾚｼﾞｽﾀの結果準備可割り込み要求ﾌﾗｸﾞ(RESRDY)ﾋﾞｯﾄが設定(1)されます。

使用者は平均値を得るためにその値を読んでそれを採取数で除算することができます。累積された採取(試料)が16よりもっとある場合、その結果はﾊｰﾄﾞｳｪｱで16ﾋﾞｯﾄ変数に切り下げられ、それは16で除算されることが必要です。

```
adcVal = ADC0_read();                   /* 結果読み込みによって割り込み要求ﾌﾗｸﾞを解除(0) */
adcVal = adcVal >> ADC_SHIFT_DIV16;     /* 採取数>16の場合に採取数または16で除算 */
```

7.2. 結果送信

ADC結果は下で提示されるように、読まれ、採取数で除算され、USARTを通して送信されなければなりません。

```
while (1)
{
    /* ADC結果読み込み */
    adcVal = ADC0_read();
    /* 採取数>16の場合に採取数または16で除算 */
    adcVal = adcVal >> ADC_SHIFT_DIV16;
    /* ﾃﾞｰﾀ可視器を使って作図されるべきADC結果を送信 */
    USART1_Write(START_TOKEN);
    USART1_Write(adcVal &amp; 0x00FF);
    USART1_Write(adcVal >> 8);
    USART1_Write(END_TOKEN);
}
```

7.3. ｺｰﾄﾞ例

Atmel Studioを使って開発したｺｰﾄﾞ例は下で入手可能です。

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

MPLAB X IDEを使って開発したｺｰﾄﾞ例は下で入手可能です。

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

助言: 完全なｺｰﾄﾞ例は「追補」で利用可能です。

8. ADC窓比較器

8.1. ADC初期化

窓比較器構成設定ではADC結果が指定閾値未満か、または超えるかをﾃﾞﾊﾞｲｽが検出します。これは指定範囲内側で維持されることが必要とされる信号を監視する時や、電池低下/過充電を合図するのに有用です。

ADC窓比較器は「3. ﾊｰﾄﾞｳｪｱ構成設定」で説明した構成A設定を使います。

この例では窓比較器が自由走行動作で使われます。監視される信号は継続的な採取が必要とされ、自由走行動作は各変換に対して手動の開始を必要としないことによってCPU負荷を減らします。この例に対しては窓比較器下側閾値(ADC0.WINLT)ﾚｼﾞｽﾀで$100の閾値が設定されます。

図8-1. ADC0.WINLT – 下位側閾値設定

ビット	15	14	13	12	11	10	9	8
	WINLT15~8							
アクセス種別	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
リセット値	0	0	0	0	0	0	0	0

ビット	7	6	5	4	3	2	1	0
	WINLT7~0							
アクセス種別	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
リセット値	0	0	0	0	0	0	0	0

● **ビット15~8 – WINLT15~8：窓比較器下側閾値上位バイト** (Window Comparator Low Threshold high byte)
これらのビットは16ビット レジスタの上位バイトを保持します。

● **ビット7~0 – WINLT7~0：窓比較器下側閾値下位バイト** (Window Comparator Low Threshold low byte)
これらのビットは16ビット レジスタの下位バイトを保持します。

下のコード断片では0x100に対してWINDOW_CMP_LOW_TH_EXAMPLEマクロが定義されて設定され、窓比較器用下側閾値として提供されます。

```
#define WINDOW_CMP_LOW_TH_EXAMPLE       (0x100)
ADC0.WINLT = WINDOW_CMP_LOW_TH_EXAMPLE;
```

ADC窓比較器動作は制御E(ADC0.CTRLE)レジスタで設定されます。

図8-2. ADC0.CTRLE 0 窓比較器動作設定

ビット	7	6	5	4	3	2	1	0
						WINCM2~0		
アクセス種別	R	R	R	R	R	R/W	R/W	R/W
リセット値	0	0	0	0	0	0	0	0

● **ビット2~0 – WINCM2~0：窓比較器動作** (Window Comparator Mode)
この領域は窓比較器動作で割り込み要求フラグが設定(1)される時を許可して定義します。下表で結果は16ビット累積器の結果です。WINLTとWINHTは各々16ビットの下側閾値と16ビットの上側閾値です。

値	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	その他
名称	NONE	BELOW	ABOVE	INSIDE	OUTSIDE	–
説明	窓比較なし(既定)	結果<WINLT	結果>WINHT	WINLT≦結果≦WINHT	結果<WINLTまたは結果>WINHT	(予約)

次のコード行は結果に対する閾値がWINLT未満を設定します。

```
ADC0.CTRLE = ADC_WINCM_BELOW_gc;
```

ADCの結果が前に設定した閾値未満の場合、ハードウェアによって割り込み要求フラグ(ADC0.INTFLAGS)レジスタの窓比較器割り込み要求フラグ(WCMP)ビットが設定(1)され、それはこれに'1'を書く、または結果(RES)から変換結果を読むことによって解除(0)されることが必要です。

8.2. 結果送信

ADC結果は下で提示されるように、読まれ、採取数で除算され、USARTを通して送信されなければなりません。

```
while (1)
{
    while (!ADC0_resultReady());

    if (ADC0_resultBelowTreshold())
    {
        LED0_on();
    }
    else
    {
        LED0_off();
```

```
    }

    adcVal = ADC0_read();

    /* データ可視器を使って作図されるべきADC結果を送信 */
    USART1_Write(START_TOKEN);
    USART1_Write(adcVal &amp; 0x00FF);
    USART1_Write(adcVal >> 8);
    USART1_Write(END_TOKEN);
}
```

8.3. コード例

Atmel Studioを使って開発したコード例は下で入手可能です。

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。

MPLAB X IDEを使って開発したコード例は下で入手可能です。

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。

助言: 完全なコード例は「追補」で利用可能です。

9. ADC事象起動

9.1. ADC初期化

ADC変換は事象によっても起動することができ、これは事象制御(ADCn.EVCTRL)レジスタの事象入力開始(STARTEI)ビットへの'1'書き込みによって許可されます(この例ではADC0実体が使われます)。

ADC事象起動使用事例は「3. ハードウェア構成設定」で説明した構成A設定に基づきます。

図9-1. ADC0.EVCTRL – STARTEIビット設定

ビット	7	6	5	4	3	2	1	0
								STARTEI
アクセス種別	R	R	R	R	R	R	R	R/W
リセット値	0	0	0	0	0	0	0	0

- **ビット0 – STARTEI : 事象入力開始** (Start Event Input)

このビットは変換開始用の起動元として事象入力の使用を許可します。このビットに'1'が書かれると、事象上昇端がADC変換を起動します。

次のコード行は事象上昇端によって起動されることをADCに許可します。

```
ADC0.EVCTRL |= ADC_STARTEI_bm;
```

事象システム(EVSYS)を通してADCに配線されてやって来るどの事象もADC変換を起動します。事象起動入力は端感知です。事象が起こると、指令(ADCn.COMMAND)の変換開始(STCONV)が設定(1)され、STCONVは変換完了時に解除(0)されます。

例えば、RTC溢れでADC変換を開始するには以下の設定が行われなければなりません。

1. RTC溢れ事象が事象システムのチャネル0に繋がれなければなりません。
2. 事象使用部のADC0はチャネル0からの入力を取るように構成設定されなければなりません。
3. ADCの事象制御(ADC0.EVCTRL)レジスタの事象入力開始(STARTEI)ビットは事象によって起動されることをADC変換に許すように設定されなければなりません。

```
EVSYS.CHANNEL0 = EVSYS_GENERATOR_RTC_OVF_gc;    /* 実時間計数器溢れ */
EVSYS.USERADC0 = EVSYS_CHANNEL_CHANNEL0_gc;     /* 使用部を事象チャネル0に接続 */
ADC0.EVCTRL |= ADC_STARTEI_bm;                  /* 事象起動変換を許可 */
```

9.2. 結果送信

ADC結果はUSARTを通して送信されます。割り込みのためにadcVal値が不正になるのを避けるため、下で提示されるように結果は別の変数を使って格納されます。

```
while (1)
{
    if (adcResultReady == 1)
    {
        /* 割り込みのためにadcValが変わるのを避けるために結果を格納。これの操作は非分断でなければなりません。 */
        cli();
        result = adcVal;
        sei();

        /* フラグ値更新 */
        adcResultReady = 0;
        /* LED交互切替 */
        LED0_toggle();

        /* データ可視器を使って作図されるべきADC結果を送信 */
        USART1_Write(START_TOKEN);
        USART1_Write(result & 0x00FF);
        USART1_Write(result >> 8);
        USART1_Write(END_TOKEN);
    }
}
```

9.3. コード例

Atmel Studioを使って開発したコード例は下で入手可能です。

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。

MPLAB X IDEを使って開発したコード例は下で入手可能です。

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。

助言: 完全なコード例は「追補」で利用可能です。

10. ADC温度測定

10.1. ADC初期化

AVR DAチップ上の温度感知器を使い、この事例について、ADC温度測定はAVR128DA48 Curiosity Nano基板を除き、余分なハードウェアは必要とされません。それに加えて、多重器正選択(ADC0.MUXPOS)レジスタでの入力が温度感知器に設定されることが必要です。

図10-1. ADC0.MUXPOS選択

ビット	7	6	5	4	3	2	1	0
		MUXPOS6~0						
アクセス種別	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
リセット値	0	0	0	0	0	0	0	0

- **ビット6~0 – MUXPOS6~0 : 多重器正選択** (MUX Selection for Positive ADC Input)

このビット領域はどのアナログ入力がADCの正入力に接続されるかを選びます。このビット領域が変換中に変えられる場合、その変更はこの変換が完了するまで無効です。

値	$00～$15	$40	$42	$48	$49	$4A	$4B	その他
名称	AIN0～AIN21	GND	TEMPSENSE	DAC0	DACREF0	DACREF1	DACREF2	-
説明	ADC入力0～21ﾋﾟﾝ	接地	温度感知器	DAC0	DACREF0	DACREF1	DACREF2	(予約)

より高い精度の読み取りを達成するため、温度感知器測定の結果はﾃﾞﾊﾞｲｽ製造または使用者校正からの補償値を使って応用ｿﾌﾄｳｪｱで処理されなければなりません。

温度は次式によって計算され、これはｹﾙﾋﾞﾝで表されます。

$$T = \frac{(\text{変位}-\text{ADC結果}) \times \text{利得}}{4096}$$

温度感知器校正値はﾁｯﾌﾟ上の温度感知器での温度測定用の修正係数を含みます。温度感知器校正0(SIGROW.TEMPSENSE0)は利得/傾斜に対する(符号なし)修正係数で温度感知器校正1(SIGROW.TEMPSENSE1)は変位に対する(符号付き)修正係数です。

識票列からの補償値を使う時に以下の断片ｺｰﾄﾞが推奨されます。

```
uint16_t sigrow_offset = SIGROW.TEMPSENSE1;
uint16_t sigrow_slope = SIGROW.TEMPSENSE0;

/* ADC0.RES読み込みによって割り込み要求ﾌﾗｸﾞを解除(0) */
temp = sigrow_offset - ADC0.RES;      /* 変換結果の変位(ｵﾌｾｯﾄ)補正 */
temp *= sigrow_slope;                 /* 変換結果の利得補正(結果は16ﾋﾞｯﾄ変数を溢れます。) */
temp += 0x0800;                       /* 下の除算で正しい丸めを得るために4096/2を加算 */
temp >>= 12;                          /* 2^12(4096)で除算することによって最も近いｹﾙﾋﾞﾝ温度に四捨五入 */
return temp - 273;                    /* ｹﾙﾋﾞﾝから摂氏に変換(0ｹﾙﾋﾞﾝ - 273.15 = -273.1℃) */
```

ADC結果はwhile繰り返しで読むことができます。

10.2. 結果送信

温度はUSARTを通して送信されます。ADC値読み込み後、結果は対応する摂氏温度を得るために変換されなければなりません。

```
while (1)
{
    /* ADC結果読み込み */
    adcVal = ADC0_read();
    /* ADC結果を摂氏温度に変換 */
    temp_C = temperatureConvert(adcVal);

    /* ﾃﾞｰﾀ可視器を使って作図されるべきADC結果を送信 */
    USART1_Write(START_TOKEN);
    USART1_Write(temp_C & 0x00FF);
    USART1_Write(temp_C >> 8);
    USART1_Write(END_TOKEN);
}
```

10.3. ｺｰﾄﾞ例

Atmel Studioを使って開発したｺｰﾄﾞ例は下で入手可能です。

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

MPLAB X IDEを使って開発したｺｰﾄﾞ例は下で入手可能です。

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

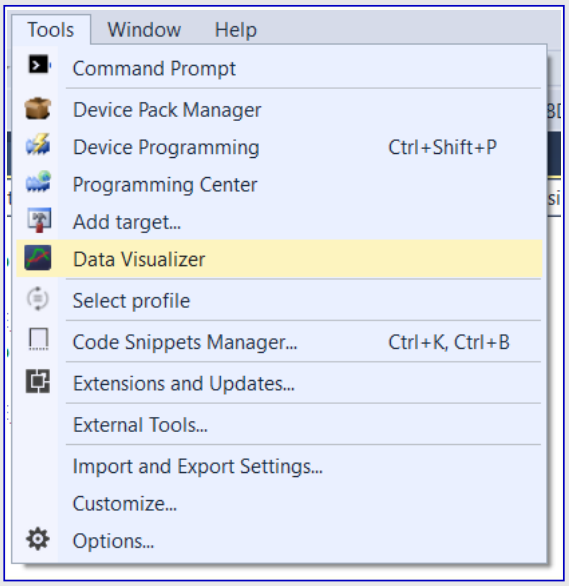
助言: 完全なｺｰﾄﾞ例は「追補」で利用可能です。

11. データ可視器

ADC結果を可視化して分析するため、使用者は以下のこれらの手順に従わなければなりません。

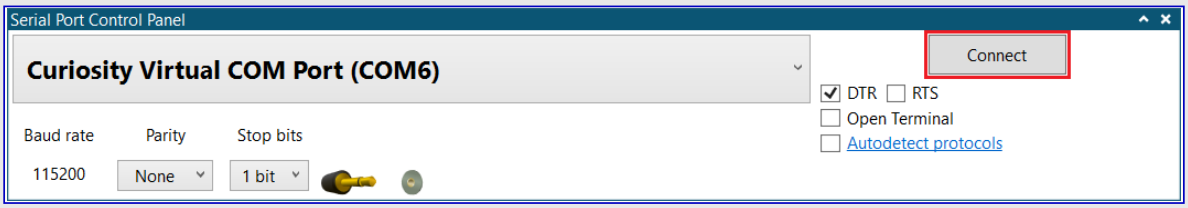
1. Tools(ツール)⇒Data Visualizer(データ可視器)を選ぶことによってデータ可視器(Data Visualizer)ツールを開いてください。

図11-1. データ可視器を開く



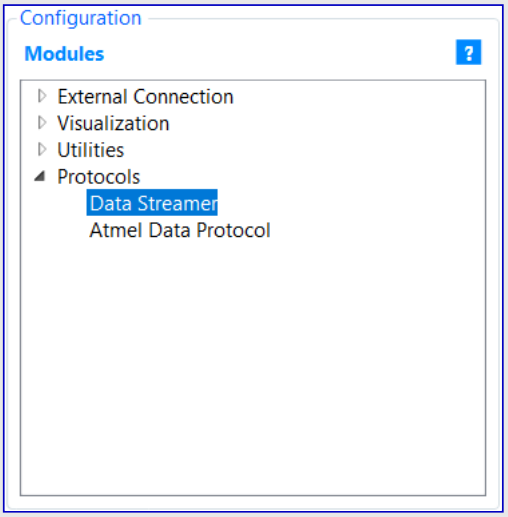
2. 下図で描かれるようにCuriosity NanoのCOMnポートを接続してください。

図11-2. 望む通信ポートへ接続



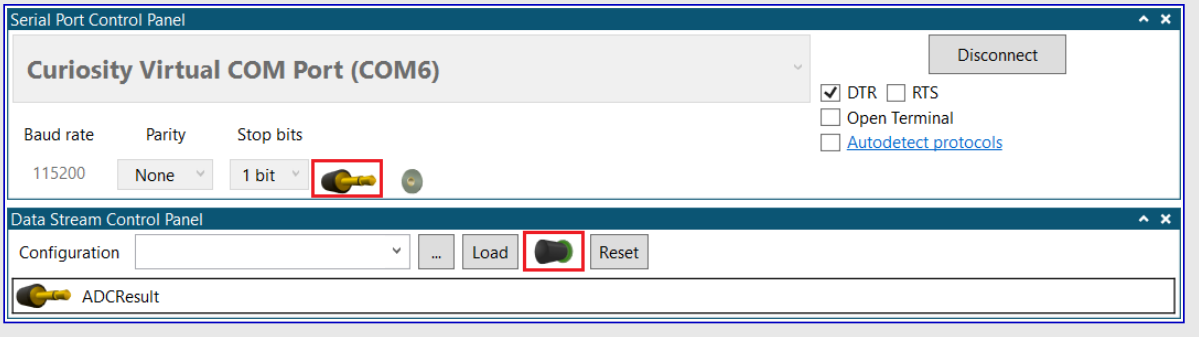
3. Configuration(構成設定)⇒Protocols(規約)からダブル クリックすることによってData Streamer(データ流し)を開いてください。

図11-3. 新しいデータ流し規約を開く



4. ﾄﾞﾗｯｸﾞ&ﾄﾞﾛｯﾌﾟすることによってCOMnﾎﾟｰﾄ供給元をData Stream Control Panel(ﾃﾞｰﾀ流し制御盤)入力に接続してください。

図11-4. 通信ﾎﾟｰﾄをﾃﾞｰﾀ流しへ接続



5. 構成設定ﾌｧｲﾙを選んでLoad(読み込み)をｸﾘｯｸしてください。

図11-5. 構成設定ﾌｧｲﾙ検索

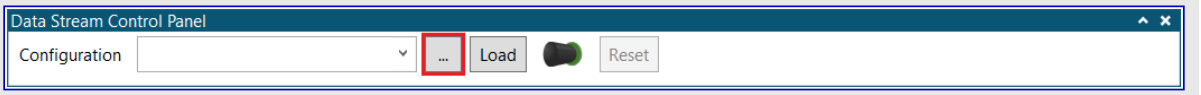


図11-6. 構成設定ﾌｧｲﾙ選択

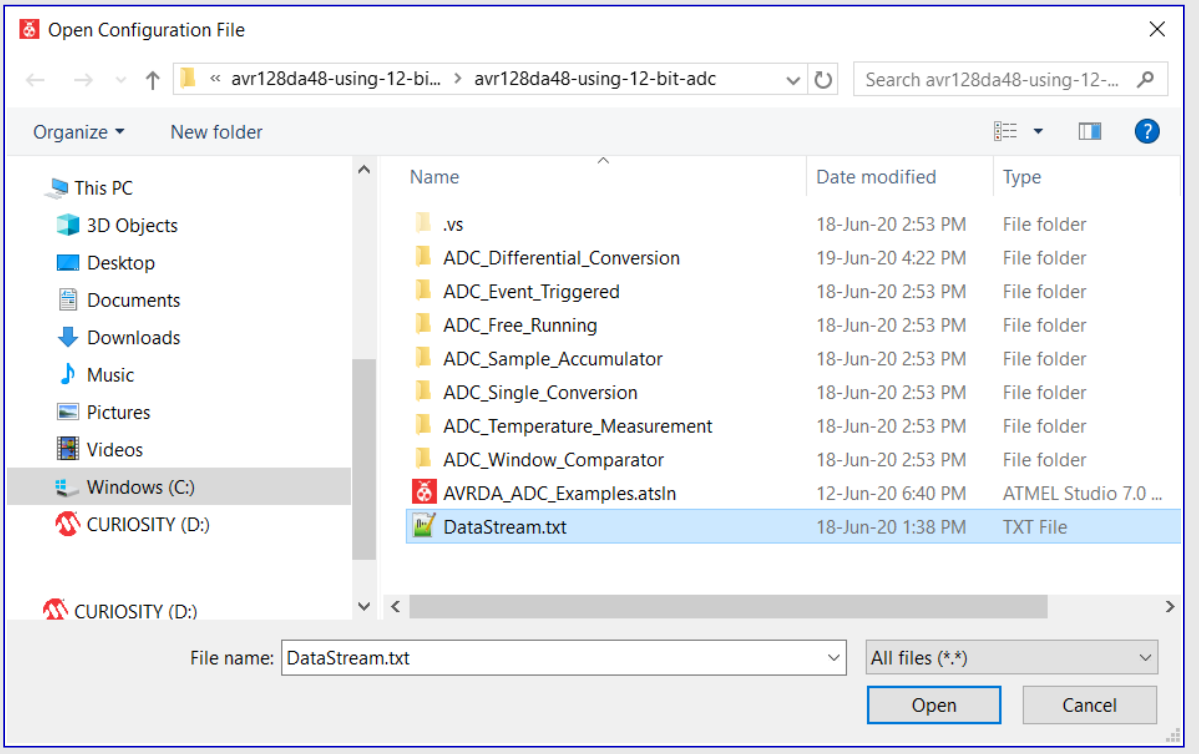
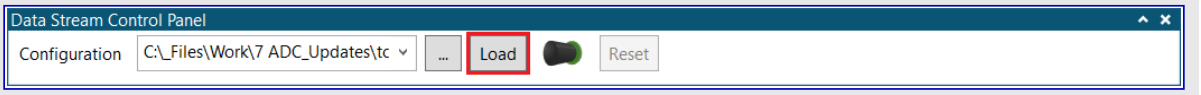
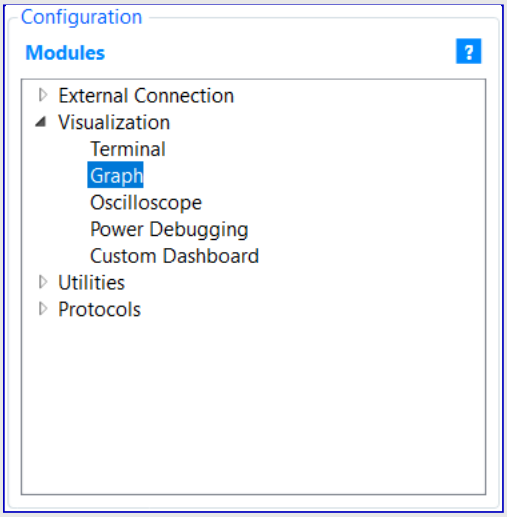


図11-7. 構成設定ﾌｧｲﾙ読み込み



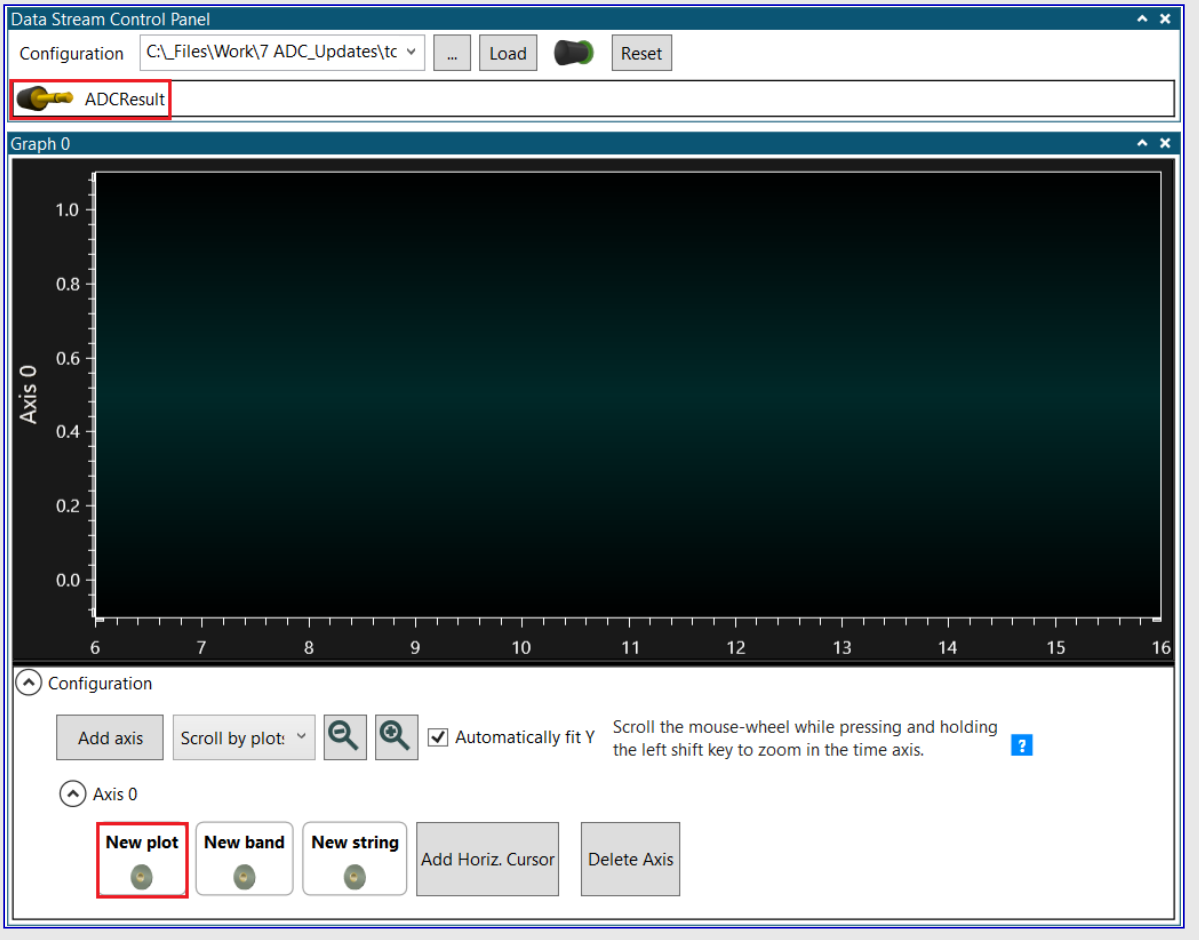
6. Configuration(構成設定)⇒Visualization(可視化)からﾀﾞﾌﾞﾙ ｸﾘｯｸすることによって新しいGraph(図表)を開いてください。

図11-8. 新しい図表を開く



7. ﾄﾞﾗｯｸﾞ&ﾄﾞﾛｯﾌﾟすることによって、ADCResult(ADC結果)供給元をGraph(図表)(Configuration(構成設定)⇒Axis 0(軸0)⇒New plot(新規作図))入力に接続してください。

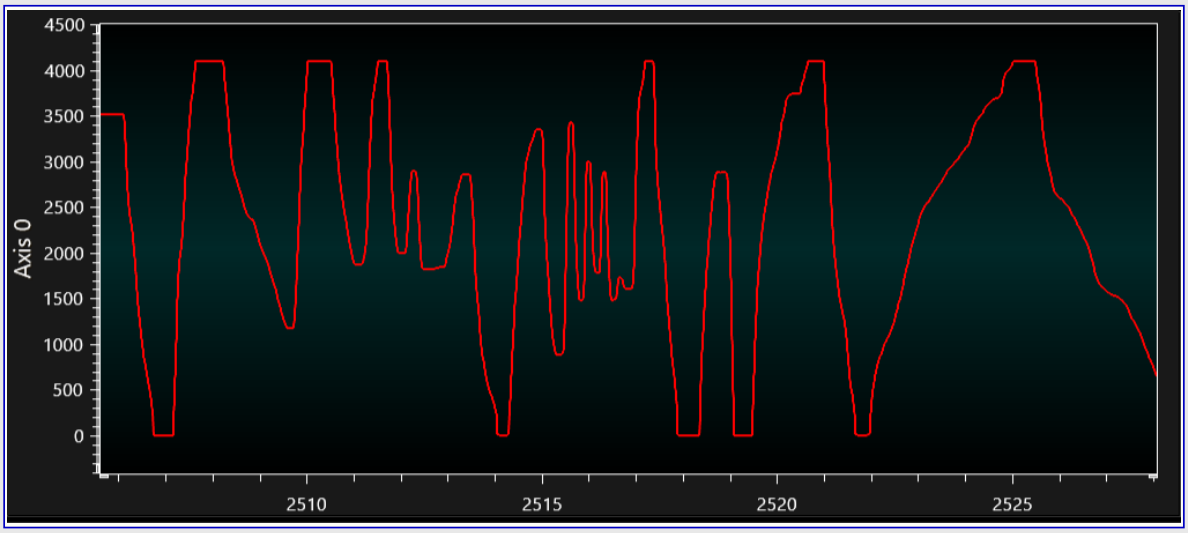
図11-9. ADC結果を図表上の作図へ接続



全ての使用事例に対する結果が下図で提示されます。この結果は可変抵抗器の回転子を無作為に調整することによって得られました。

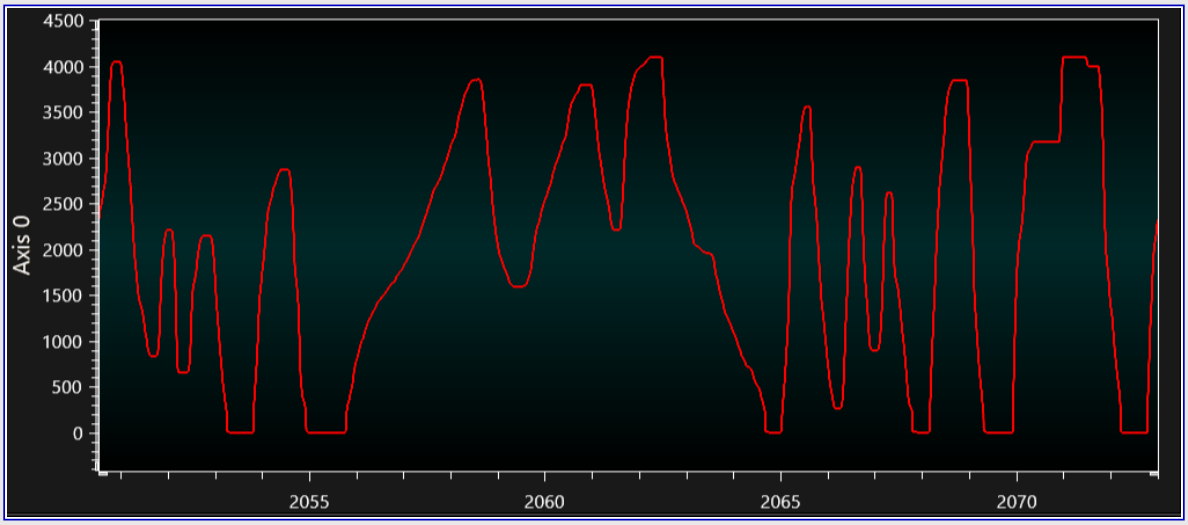
ADC単独変換使用事例は下図に従って作図されます。この結果は12ビット分解能値に対して予期されるように[0～4095]の範囲です。

図11-10. ADC単独変換



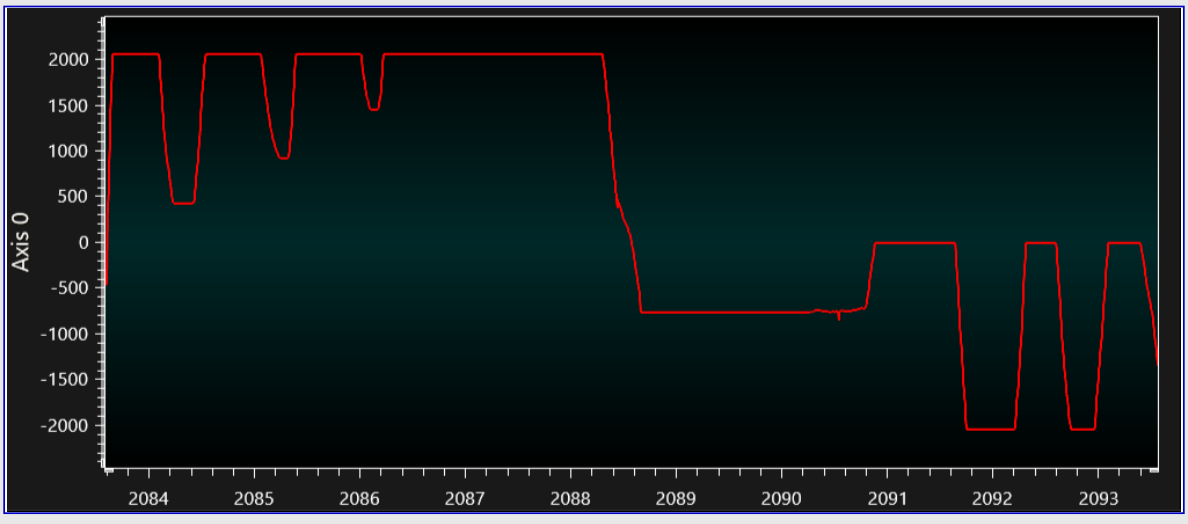
ADC自由走行使用事例結果が下で提示されます。これらは単独変換結果と同じ様に見えます。

図11-11. ADC自由走行変換



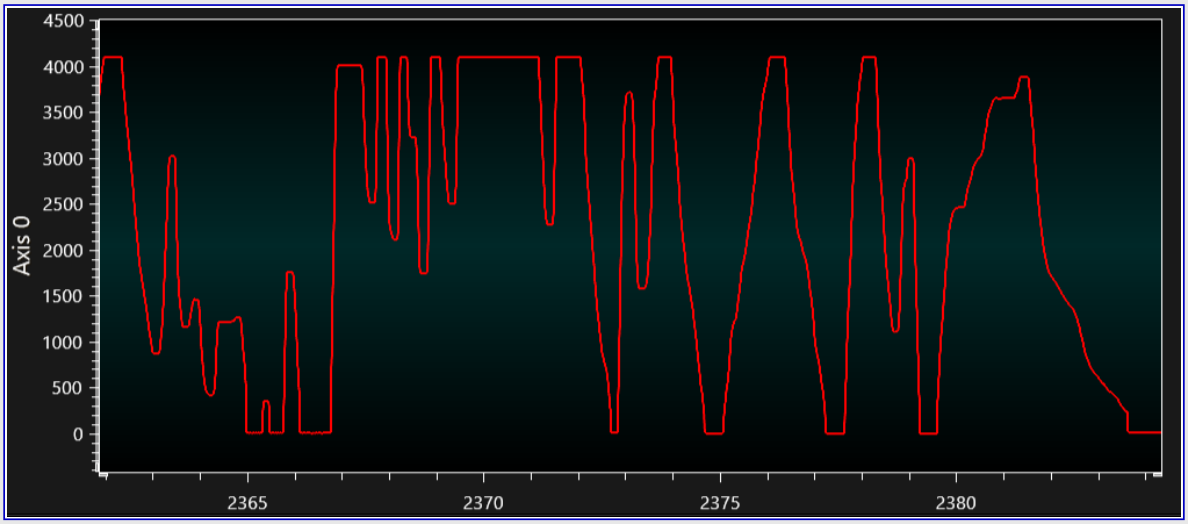
ADC差動変換結果が下で提示されます。この結果は12ビット符号付き値に対して予期されるように[-2048～+2047]の範囲です。

図11-12. ADC差動変換



© 2020 Microchip Technology Inc.

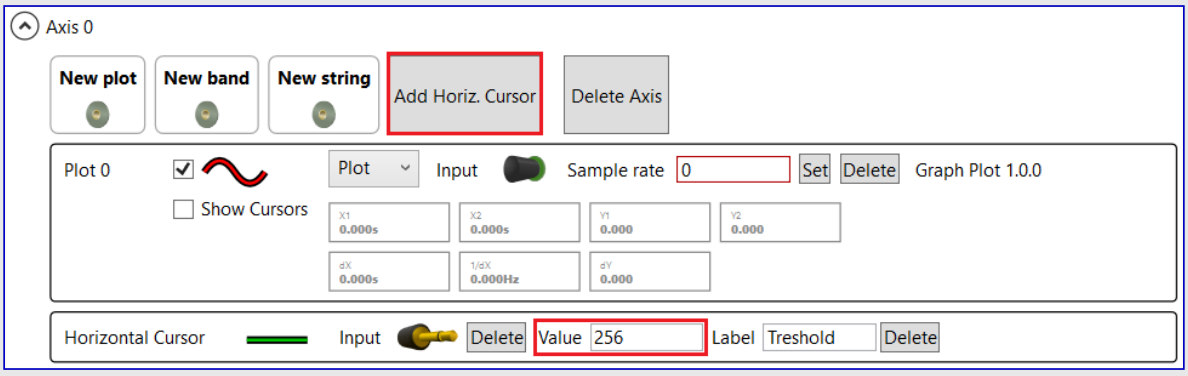
図11-13. ADC採取累積器



ADC窓比較器使用事例について、望む閾値を測定するために水平カーソルを追加することができます。この手順は下図で描かれます。

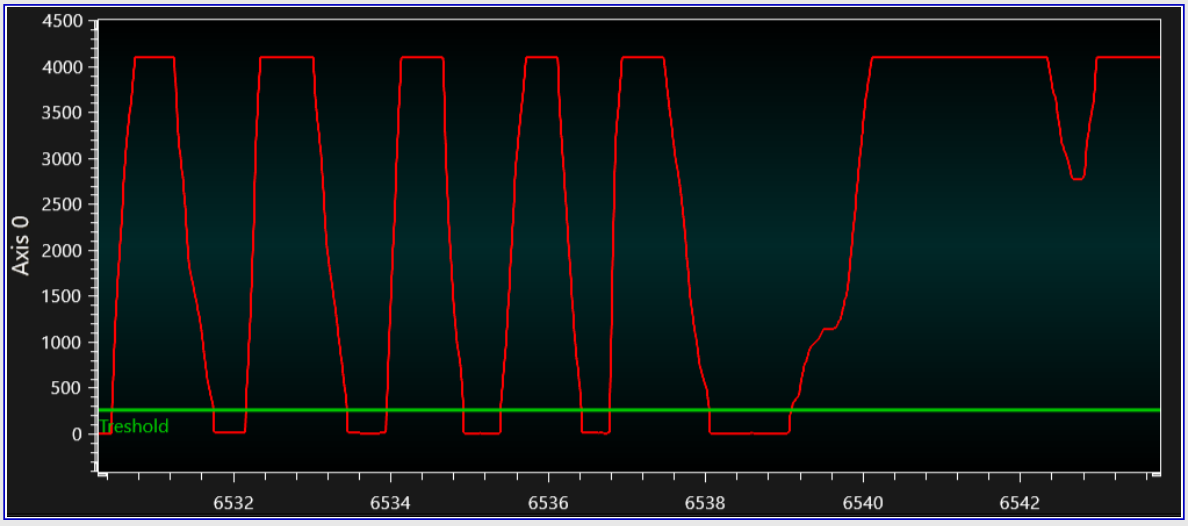
この使用事例に於いて可変抵抗器の回転はLED0がONまたはOFFに切り替わるまでADC値を変更します。閾値(LED0変更状態に従うADC変換結果の値)は水平カーソルによって強調表示されます。

図11-14. 水平カーソル追加



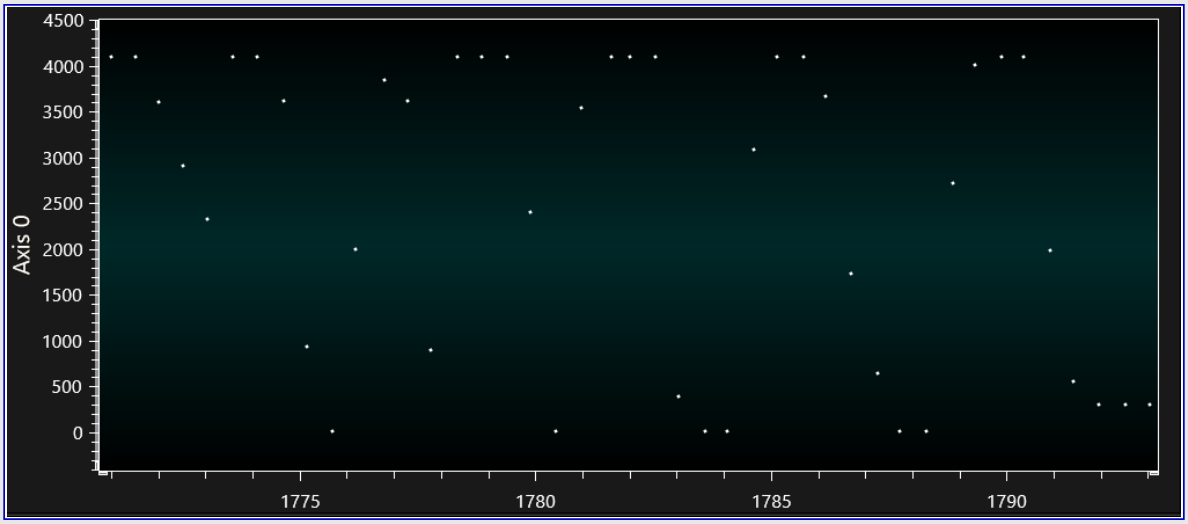
作図は下図で見ることができます。

図11-15. ADC窓比較器



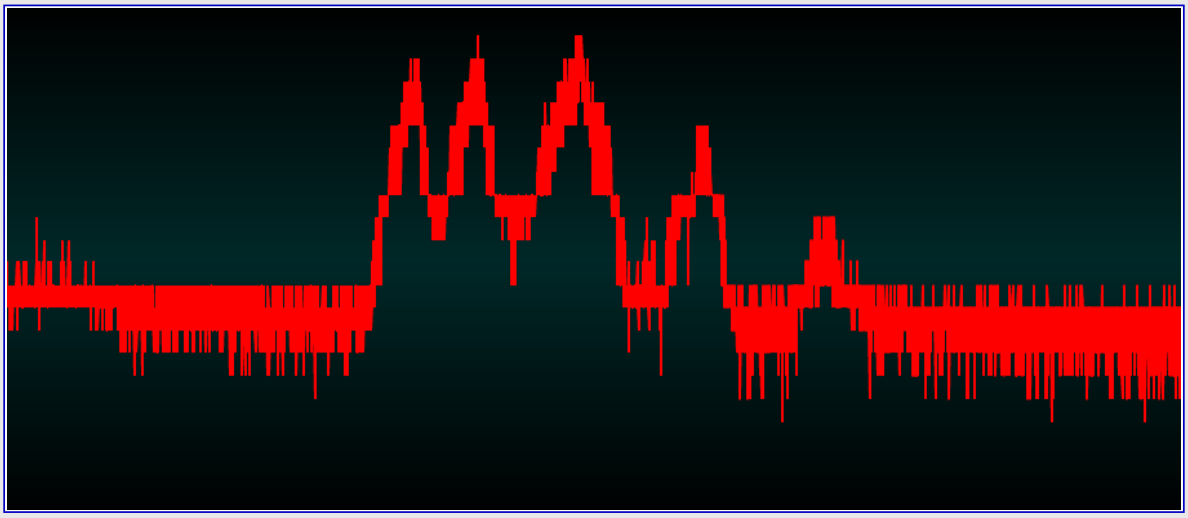
ADC事象起動使用事例結果は下図で提示されます。

図11-16. ADC事象起動



ADC温度測定使用事例結果は下図で提示されます。これらの結果を得るため、構成環境に小さな温度変動が導入されました。

図11-17. ADC温度測定



12. 参考文献

1. AVR128DA28/328/48/64暫定ﾃﾞｰﾀｼｰﾄ
2. AVR128DA48 Cutiosity Nano使用者の手引き
3. ﾃﾞｰﾀ可視器(Data Visualizer)ｿﾌﾄｳｪｱ使用者の手引き

13. 追補

例13-1. ADC単独変換ｺｰﾄﾞ例

```
/*  ¥file  main.c
    ¥brief      ADC単独変換
    (c) 2019 Microchip Technology Inc.とその子会社
    これらの条件を遵守するのを条件にMicrochipｿﾌﾄｳｪｱとMicrochip製品でのどの占的派生物も使うことができます。
    Microchipｿﾌﾄｳｪｱに付随し得る(ｵｰﾌﾟﾝ ｿｰｽ ｿﾌﾄｳｪｱを含む)第三者ｿﾌﾄｳｪｱの使用に対して適用できる第三者許諾条項を
    遵守するにはあなたの責任です。
    このｿﾌﾄｳｪｱはMicrochipによって「現状そのまま」で供給されます。Microchipは非侵害、商品性、特定目的に対する適合
    性の何れの黙示的保証を含め、このｿﾌﾄｳｪｱに適用する、明示的、黙示的、法的であろうとなかろうと保証しません。
    如何なる場合においても、Microchipはｿﾌﾄｳｪｱに関連するあらゆる種類の間接的、特別的、懲罰的、偶発的または結果的
    な損失、損害、費用または経費に対して責任を負わないものとします。法律で認められている最大限の範囲で、ｿﾌﾄｳｪｱに
    関連する全ての請求に対するMicrochipの全責任は、もしあれば、ｿﾌﾄｳｪｱのためにMicrochipへ直接支払った料金を超え
    ないものとします。
*/
#define F_CPU             4000000UL             /* 主ｸﾛｯｸ周波数 */
#define START_TOKEN       0x03                  /* ﾌﾚｰﾑ開始通票 */
#define END_TOKEN         0xFC                  /* ﾌﾚｰﾑ終了通票 */
/* ﾎﾞｰﾚｰﾄ計算 */
#define USART1_BAUD_RATE(BAUD_RATE) (((float)F_CPU * 64 / (16 * (float)BAUD_RATE)) + 0.5)
#include <avr/io.h>
#include <stdbool.h>
#include <avr/cpufunc.h>
void CLKCTRL_init(void);
void PORT_init(void);
void VREF0_init(void);
void ADC0_init(void);
void USART1_init(void);
uint16_t ADC0_read(void);
void ADC0_start(void);
void USART1_Write(const uint8_t data);
/* この関数はCLKCTRL単位部を初期化します。 */
void CLKCTRL_init(void)
{
    /* 周波数選択(FREQSEL) 4MHz */
    ccp_write_io((void*)&(CLKCTRL.OSCHFCTRLA), (CLKCTRL.OSCHFCTRLA | CLKCTRL_FREQSEL_4M_gc));
}
/* この関数はPORT単位部を初期化します。 */
void PORT_init(void)
{
    /* PC0をUSART1 TX用出力として構成設定 */
    PORTC.DIRSET = PIN0_bm;

    /* PD3で割り込みとﾃﾞｼﾞﾀﾙ入力緩衝部を禁止 */
    PORTD.PIN3CTRL &= ~PORT_ISC_gm;
    PORTD.PIN3CTRL |= PORT_ISC_INPUT_DISABLE_gc;

    /* ﾌﾟﾙｱｯﾌﾟ抵抗を禁止 */
    PORTD.PIN3CTRL &= ~PORT_PULLUPEN_bm;
}
/* この関数はVREF単位部を初期化します。 */
void VREF0_init(void)
{
    VREF.ADC0REF = VREF_REFSEL_2V048_gc;        /* 内部2.048V参照基準 */
}
/* この関数はADC単位部を初期化します。 */
void ADC0_init(void)
{
    ADC0.CTRLC = ADC_PRESC_DIV4_gc;             /* 4分周したCLK_PER */
    ADC0.CTRLA = ADC_ENABLE_bm                  /* ADC許可: 許可 */
```

```
                    | ADC_RESSEL_12BIT_gc;            /* 12ﾋﾞｯﾄ動作 */
    ADC0.MUXPOS = ADC_MUXPOS_AIN3_gc;            /* ADCﾁｬﾈﾙAIN3(PD3)選択 */
}
/* この関数はUSART単位部を初期化します。 */
void USART1_init(void)
{
    /* ﾎﾞｰﾚｰﾄを115200に構成設定 */
    USART1.BAUD = (uint16_t)USART1_BAUD_RATE(115200);
    USART1.CTRLB = USART_TXEN_bm;                /* 送信部許可 */
    USART1.CTRLC = USART_CHSIZE_8BIT_gc;         /* 文字の大きさを8ﾋﾞｯﾄに構成設定 */
}
/* この関数はADC変換結果を返します。 */
uint16_t ADC0_read(void)
{
    /* ADC結果準備可待ち */
    while (!(ADC0.INTFLAGS & ADC_RESRDY_bm));
    /* 結果読み込みによって割り込み要求ﾌﾗｸﾞを解除(0) */
    return ADC0.RES;
}
/* この関数はADC変換を開始します。 */
void ADC0_start(void)
{
    /* ADC変換開始 */
    ADC0.COMMAND = ADC_STCONV_bm;
}
/* この関数はUSARTを通して1ﾊﾞｲﾄを送信します。 */
void USART1_Write(const uint8_t data)
{
    /* USART緩衝部がﾃﾞｰﾀ送信準備可か調査 */
    while (!(USART1.STATUS & USART_DREIF_bm));
    /* TXDATALﾚｼﾞｽﾀを使ってﾃﾞｰﾀを送信 */
    USART1.TXDATAL = data;
}
int main(void)
{
    uint16_t adcVal;

    /* 全ての周辺機能を初期化 */
    CLKCTRL_init();
    PORT_init();
    VREF0_init();
    ADC0_init();
    USART1_init();

    while (1)
    {
        /* ADC変換開始 */
        ADC0_start();

        /* ADC結果読み込み */
        adcVal = ADC0_read();

        /* ﾃﾞｰﾀ可視器を使って作図されるべきADC結果を送信 */
        USART1_Write(START_TOKEN);
        USART1_Write(adcVal & 0x00FF);
        USART1_Write(adcVal >> 8);
        USART1_Write(END_TOKEN);
    }
}
```

例13-2. ADC自由走行変換ｺｰﾄﾞ例

```
/*  ¥file  main.c
    ¥brief      ADC自由走行
    (c) 2019 Microchip Technology Inc.とその子会社
    これらの条件を遵守するのを条件にMicrochipｿﾌﾄｳｪｱとMicrochip製品でのどの占的派生物も使うことができます。
    Microchipｿﾌﾄｳｪｱに付随し得る(ｵｰﾌﾟﾝ ｿｰｽ ｿﾌﾄｳｪｱを含む)第三者ｿﾌﾄｳｪｱの使用に対して適用できる第三者許諾条項を
    遵守するにはあなたの責任です。
    このｿﾌﾄｳｪｱはMicrochipによって「現状そのまま」で供給されます。Microchipは非侵害、商品性、特定目的に対する適合
    性の何れの黙示的保証を含め、このｿﾌﾄｳｪｱに適用する、明示的、黙示的、法的であろうとなかろうと保証しません。
    如何なる場合においても、Microchipはｿﾌﾄｳｪｱに関連するあらゆる種類の間接的、特別的、懲罰的、偶発的または結果的
    な損失、損害、費用または経費に対して責任を負わないものとします。法律で認められている最大限の範囲で、ｿﾌﾄｳｪｱに
    関連する全ての請求に対するMicrochipの全責任は、もしあれば、ｿﾌﾄｳｪｱのためにMicrochipへ直接支払った料金を超え
    ないものとします。
*/
#define F_CPU             4000000UL             /* 主ｸﾛｯｸ周波数 */
#define START_TOKEN       0x03                  /* ﾌﾚｰﾑ開始通票 */
#define END_TOKEN         0xFC                  /* ﾌﾚｰﾑ終了通票 */
/* ﾎﾞｰﾚｰﾄ計算 */
#define USART1_BAUD_RATE(BAUD_RATE) (((float)F_CPU * 64 / (16 * (float)BAUD_RATE)) + 0.5)
#include <avr/io.h>
#include <stdbool.h>
#include <avr/cpufunc.h>
void CLKCTRL_init(void);
void PORT_init(void);
void VREF0_init(void);
void ADC0_init(void);
void USART1_init(void);
uint16_t ADC0_read(void);
void ADC0_start(void);
void USART1_Write(const uint8_t data);
/* この関数はCLKCTRL単位部を初期化します。 */
void CLKCTRL_init(void)
{
    /* 周波数選択(FREQSEL) 4MHz */
    ccp_write_io((void*)&(CLKCTRL.OSCHFCTRLA), (CLKCTRL.OSCHFCTRLA | CLKCTRL_FREQSEL_4M_gc));
}
/* この関数はPORT単位部を初期化します。 */
void PORT_init(void)
{
    /* PC0をUSART1 TX用出力として構成設定 */
    PORTC.DIRSET = PIN0_bm;

    /* PD3で割り込みとﾃﾞｼﾞﾀﾙ入力緩衝部を禁止 */
    PORTD.PIN3CTRL &= ~PORT_ISC_gm;
    PORTD.PIN3CTRL |= PORT_ISC_INPUT_DISABLE_gc;

    /* ﾌﾟﾙｱｯﾌﾟ抵抗を禁止 */
    PORTD.PIN3CTRL &= ~PORT_PULLUPEN_bm;
}
/* この関数はVREF単位部を初期化します。 */
void VREF0_init(void)
{
    VREF.ADC0REF = VREF_REFSEL_2V048_gc;      /* 内部2.048V参照基準 */
}
/* この関数はADC単位部を初期化します。 */
void ADC0_init(void)
{
    ADC0.CTRLC = ADC_PRESC_DIV4_gc;           /* 4分周したCLK_PER */
    ADC0.CTRLA = ADC_ENABLE_bm                /* ADC許可: 許可 */
               | ADC_RESSEL_12BIT_gc          /* 12ﾋﾞｯﾄ動作 */
               | ADC_FREERUN_bm;              /* 自由走行動作許可 */
```

```
    ADC0.MUXPOS = ADC_MUXPOS_AIN3_gc;           /* ADCチャネルAIN3(PD3)選択 */
}
/* この関数はUSART単位部を初期化します。 */
void USART1_init(void)
{
    /* ﾎﾞｰﾚｰﾄを115200に構成設定 */
    USART1.BAUD = (uint16_t)USART1_BAUD_RATE(115200);
    USART1.CTRLB = USART_TXEN_bm;               /* 送信部許可 */
    USART1.CTRLC = USART_CHSIZE_8BIT_gc;        /* 文字の大きさを8ﾋﾞｯﾄに構成設定 */
}
/* この関数はADC変換結果を返します。 */
uint16_t ADC0_read(void)
{
    /* ADC結果準備可待ち */
    while (!(ADC0.INTFLAGS & ADC_RESRDY_bm));
    /* 結果読み込みによって割り込み要求ﾌﾗｸﾞを解除(0) */
    return ADC0.RES;
}
/* この関数はADC変換を開始します。 */
void ADC0_start(void)
{
    /* ADC変換開始 */
    ADC0.COMMAND = ADC_STCONV_bm;
}
/* この関数はUSARTを通して1ﾊﾞｲﾄを送信します。 */
void USART1_Write(const uint8_t data)
{
    /* USART緩衝部がﾃﾞｰﾀ送信準備可か調査 */
    while (!(USART1.STATUS & USART_DREIF_bm));
    /* TXDATALﾚｼﾞｽﾀを使ってﾃﾞｰﾀを送信 */
    USART1.TXDATAL = data;
}
int main(void)
{
    uint16_t adcVal;

    /* 全ての周辺機能を初期化 */
    CLKCTRL_init();
    PORT_init();
    VREF0_init();
    ADC0_init();
    USART1_init();

    /* ADC変換開始 */
    ADC0_start();

    while (1)
    {
        /* ADC結果読み込み */
        adcVal = ADC0_read();

        /* ﾃﾞｰﾀ可視器を使って作図されるべきADC結果を送信 */
        USART1_Write(START_TOKEN);
        USART1_Write(adcVal & 0x00FF);
        USART1_Write(adcVal >> 8);
        USART1_Write(END_TOKEN);
    }
}
```

例13-3. ADC差動変換ｺｰﾄﾞ例

```
/*  ¥file  main.c
    ¥brief      ADC差動変換
    (c) 2019 Microchip Technology Inc.とその子会社
    これらの条件を遵守するのを条件にMicrochipｿﾌﾄｳｪｱとMicrochip製品でのどの占的派生物も使うことができます。
    Microchipｿﾌﾄｳｪｱに付随し得る(ｵｰﾌﾟﾝ ｿｰｽ ｿﾌﾄｳｪｱを含む)第三者ｿﾌﾄｳｪｱの使用に対して適用できる第三者許諾条項を
    遵守するにはあなたの責任です。
    このｿﾌﾄｳｪｱはMicrochipによって「現状そのまま」で供給されます。Microchipは非侵害、商品性、特定目的に対する適合
    性の何れの黙示的保証を含め、このｿﾌﾄｳｪｱに適用する、明示的、黙示的、法的であろうとなかろうと保証しません。
    如何なる場合においても、Microchipはｿﾌﾄｳｪｱに関連するあらゆる種類の間接的、特別的、懲罰的、偶発的または結果的
    な損失、損害、費用または経費に対して責任を負わないものとします。法律で認められている最大限の範囲で、ｿﾌﾄｳｪｱに
    関連する全ての請求に対するMicrochipの全責任は、もしあれば、ｿﾌﾄｳｪｱのためにMicrochipへ直接支払った料金を超え
    ないものとします。
*/
#define F_CPU             4000000UL             /* 主ｸﾛｯｸ周波数 */
#define START_TOKEN       0x03                  /* ﾌﾚｰﾑ開始通票 */
#define END_TOKEN         0xFC                  /* ﾌﾚｰﾑ終了通票 */
/* ﾎﾞｰﾚｰﾄ計算 */
#define USART1_BAUD_RATE(BAUD_RATE) (((float)F_CPU * 64 / (16 * (float)BAUD_RATE)) + 0.5)
#include <avr/io.h>
#include <stdbool.h>
#include <avr/cpufunc.h>
int16_t adcVal;
void CLKCTRL_init(void);
void PORT_init(void);
void VREF0_init(void);
void ADC0_init(void);
void USART1_init(void);
void ADC0_start(void);
bool ADC0_conversionDone(void);
int16_t ADC0_read(void);
void USART1_Write(const uint8_t data);
void SYSTEM_init(void);

void CLKCTRL_init(void)
{
    /* 周波数選択(FREQSEL) 4MHz */
    ccp_write_io((void*)&(CLKCTRL.OSCHFCTRLA), (CLKCTRL.OSCHFCTRLA | CLKCTRL_FREQSEL_4M_gc));
}
void PORT_init(void)
{
    /* PC0をUSART1 TX用出力として構成設定 */
    PORTC.DIRSET = PIN0_bm;

    /* PD3で割り込みとﾃﾞｼﾞﾀﾙ入力緩衝部を禁止 */
    PORTD.PIN3CTRL &= ~PORT_ISC_gm;
    PORTD.PIN3CTRL |= PORT_ISC_INPUT_DISABLE_gc;
    /* PD4で割り込みとﾃﾞｼﾞﾀﾙ入力緩衝部を禁止 */
    PORTD.PIN4CTRL &= ~PORT_ISC_gm;
    PORTD.PIN4CTRL |= PORT_ISC_INPUT_DISABLE_gc;

    /* ﾌﾟﾙｱｯﾌﾟ抵抗を禁止 */
    PORTD.PIN3CTRL &= ~PORT_PULLUPEN_bm;
    PORTD.PIN4CTRL &= ~PORT_PULLUPEN_bm;
}
void VREF0_init(void)
{
    VREF.ADC0REF = VREF_REFSEL_2V048_gc;        /* 内部2.048V参照基準 */
}
void ADC0_init(void)
{
```

```
    ADC0.CTRLC = ADC_PRESC_DIV4_gc;              /* 4分周したCLK_PER */
    ADC0.CTRLA = ADC_ENABLE_bm                   /* ADC許可: 許可 */
                | ADC_RESSEL_12BIT_gc            /* 12ビット動作 */
                | ADC_CONVMODE_bm                /* 差動変換 */
                | ADC_FREERUN_bm;                /* 自由走行動作許可 */
    ADC0.MUXPOS = ADC_MUXPOS_AIN3_gc;            /* ADCチャネルAIN3(PD3)選択 */
    ADC0.MUXNEG = ADC_MUXNEG_AIN4_gc;            /* ADCチャネルAIN4(PD4)選択 */
}
void USART1_init(void)
{
    /* ボーレートを115200に構成設定 */
    USART1.BAUD = (uint16_t)USART1_BAUD_RATE(115200);
    USART1.CTRLB = USART_TXEN_bm;                /* 送信部許可 */
    USART1.CTRLC = USART_CHSIZE_8BIT_gc;         /* 文字の大きさを8ビットに構成設定 */
}
int16_t ADC0_read(void)
{
    /* 結果読み込みによって割り込み要求フラグを解除(0) */
    return ADC0.RES;
}
void ADC0_start(void)
{
    /* ADC変換開始 */
    ADC0.COMMAND = ADC_STCONV_bm;
}
bool ADC0_conversionDone(void)
{
    /* 変換終了か調査 */
    return (ADC0.INTFLAGS & ADC_RESRDY_bm);
}
void USART1_Write(const uint8_t data)
{
    /* USART緩衝部がデータ送信準備可か調査 */
    while (!(USART1.STATUS & USART_DREIF_bm));
    /* TXDATALレジスタを使ってデータを送信 */
    USART1.TXDATAL = data;
}
void SYSTEM_init(void)
{
    CLKCTRL_init();
    PORT_init();
    VREF0_init();
    ADC0_init();
    USART1_init();
}
int main(void)
{
    SYSTEM_init();
    ADC0_start();

    while (1)
    {
        if (ADC0_conversionDone())
        {
            /* ADC結果読み込み */
            adcVal = ADC0_read();
            /* データ可視器を使って作図されるべきADC結果を送信 */
            USART1_Write(START_TOKEN);
            USART1_Write(adcVal & 0x00FF);
            USART1_Write(adcVal >> 8);
            USART1_Write(END_TOKEN);
```

```
        }
    }
}
```

例13-4. ADC採取累積器ｺｰﾄﾞ例

```
/*  ¥file  main.c
    ¥brief      ADC採取累積器
    (c) 2019 Microchip Technology Inc.とその子会社
    これらの条件を遵守するのを条件にMicrochipｿﾌﾄｳｪｱとMicrochip製品でのどの占的派生物も使うことができます。
    Microchipｿﾌﾄｳｪｱに付随し得る(ｵｰﾌﾟﾝ ｿｰｽ ｿﾌﾄｳｪｱを含む)第三者ｿﾌﾄｳｪｱの使用に対して適用できる第三者許諾条項を
    遵守するにはあなたの責任です。
    このｿﾌﾄｳｪｱはMicrochipによって「現状そのまま」で供給されます。Microchipは非侵害、商品性、特定目的に対する適合
    性の何れの黙示的保証を含め、このｿﾌﾄｳｪｱに適用する、明示的、黙示的、法的であろうとなかろうと保証しません。
    如何なる場合においても、Microchipはｿﾌﾄｳｪｱに関連するあらゆる種類の間接的、特別的、懲罰的、偶発的または結果的
    な損失、損害、費用または経費に対して責任を負わないものとします。法律で認められている最大限の範囲で、ｿﾌﾄｳｪｱに
    関連する全ての請求に対するMicrochipの全責任は、もしあれば、ｿﾌﾄｳｪｱのためにMicrochipへ直接支払った料金を超え
    ないものとします。
*/
#define F_CPU               4000000UL               /* 主ｸﾛｯｸ周波数 */
#define START_TOKEN         0x03                    /* ﾌﾚｰﾑ開始通票 */
#define END_TOKEN           0xFC                    /* ﾌﾚｰﾑ終了通票 */
/* ﾎﾞｰﾚｰﾄ計算 */
#define USART1_BAUD_RATE(BAUD_RATE) (((float)F_CPU * 64 / (16 * (float)BAUD_RATE)) + 0.5)
#include <avr/io.h>
#include <stdbool.h>
#include <avr/cpufunc.h>
void CLKCTRL_init(void);
void PORT_init(void);
void VREF0_init(void);
void ADC0_init(void);
void USART1_init(void);
uint16_t ADC0_read(void);
void ADC0_start(void);
void USART1_Write(const uint8_t data);
/* この関数はCLKCTRL単位部を初期化します。 */
void CLKCTRL_init(void)
{
    /* 周波数選択(FREQSEL) 4MHz */
    ccp_write_io((void*)&(CLKCTRL.OSCHFCTRLA), (CLKCTRL.OSCHFCTRLA | CLKCTRL_FREQSEL_4M_gc));
}
/* この関数はPORT単位部を初期化します。 */
void PORT_init(void)
{
    /* PC0をUSART1 TX用出力として構成設定 */
    PORTC.DIRSET = PIN0_bm;

    /* PD3で割り込みとﾃﾞｼﾞﾀﾙ入力緩衝部を禁止 */
    PORTD.PIN3CTRL &= ~PORT_ISC_gm;
    PORTD.PIN3CTRL |= PORT_ISC_INPUT_DISABLE_gc;

    /* ﾌﾟﾙｱｯﾌﾟ抵抗を禁止 */
    PORTD.PIN3CTRL &= ~PORT_PULLUPEN_bm;
}
/* この関数はVREF単位部を初期化します。 */
void VREF0_init(void)
{
    VREF.ADC0REF = VREF_REFSEL_2V048_gc;        /* 内部2.048V参照基準 */
}
/* この関数はADC単位部を初期化します。 */
void ADC0_init(void)
```

```
{
    ADC0.CTRLC = ADC_PRESC_DIV4_gc;              /* 4分周したCLK_PER */
    ADC0.CTRLA = ADC_ENABLE_bm                   /* ADC許可: 許可 */
               | ADC_RESSEL_12BIT_gc             /* 12ﾋﾞｯﾄ動作 */
               | ADC_FREERUN_bm;                 /* 自由走行動作許可 */
    ADC0.MUXPOS = ADC_MUXPOS_AIN3_gc;            /* ADCﾁｬﾈﾙAIN3(PD3)選択 */
    ADC0.CTRLB = ADC_SAMPNUM_ACC64_gc;           /* 64採取累積に設定 */
}
/* この関数はUSART単位部を初期化します。 */
void USART1_init(void)
{
    /* ﾎﾞｰﾚｰﾄを115200に構成設定 */
    USART1.BAUD = (uint16_t)USART1_BAUD_RATE(115200);
    USART1.CTRLB = USART_TXEN_bm;                /* 送信部許可 */
    USART1.CTRLC = USART_CHSIZE_8BIT_gc;         /* 文字の大きさを8ﾋﾞｯﾄに構成設定 */
}
/* この関数はADC変換結果を返します。 */
uint16_t ADC0_read(void)
{
    /* ADC結果準備可待ち */
    while (!(ADC0.INTFLAGS & ADC_RESRDY_bm));
    /* 結果読み込みによって割り込み要求ﾌﾗｸﾞを解除(0) */
    return ADC0.RES;
}
/* この関数はADC変換を開始します。 */
void ADC0_start(void)
{
    /* ADC変換開始 */
    ADC0.COMMAND = ADC_STCONV_bm;
}
/* この関数はUSARTを通して1ﾊﾞｲﾄを送信します。 */
void USART1_Write(const uint8_t data)
{
    /* USART緩衝部がﾃﾞｰﾀ送信準備可か調査 */
    while (!(USART1.STATUS & USART_DREIF_bm));
    /* TXDATALﾚｼﾞｽﾀを使ってﾃﾞｰﾀを送信 */
    USART1.TXDATAL = data;
}
int main(void)
{
    uint16_t adcVal;

    /* 全ての周辺機能を初期化 */
    CLKCTRL_init();
    PORT_init();
    VREF0_init();
    ADC0_init();
    USART1_init();

    /* ADC変換開始 */
    ADC0_start();

    while (1)
    {
        /* ADC結果読み込み */
        adcVal = ADC0_read();
        /* 採取数>16の場合に採取数または16で除算 */
        adcVal = adcVal >> ADC_SHIFT_DIV16;
        /* ﾃﾞｰﾀ可視器を使って作図されるべきADC結果を送信 */
        USART1_Write(START_TOKEN);
        USART1_Write(adcVal & 0x00FF);
```

```
        USART1_Write(adcVal >> 8);
        USART1_Write(END_TOKEN);
    }
}
```

例13-5. ADC窓比較器ｺｰﾄﾞ例

```
/*  ¥file  main.c

    ¥brief      ADC窓比較器

    (c) 2019 Microchip Technology Inc.とその子会社

    これらの条件を遵守するのを条件にMicrochipｿﾌﾄｳｪｱとMicrochip製品でのどの占的派生物も使うことができます。
    Microchipｿﾌﾄｳｪｱに付随し得る(ｵｰﾌﾟﾝ ｿｰｽ ｿﾌﾄｳｪｱを含む)第三者ｿﾌﾄｳｪｱの使用に対して適用できる第三者許諾条項を
    遵守するにはあなたの責任です。

    このｿﾌﾄｳｪｱはMicrochipによって「現状そのまま」で供給されます。Microchipは非侵害、商品性、特定目的に対する適合
    性の何れの黙示的保証を含め、このｿﾌﾄｳｪｱに適用する、明示的、黙示的、法的であろうとなかろうと保証しません。

    如何なる場合においても、Microchipはｿﾌﾄｳｪｱに関連するあらゆる種類の間接的、特別的、懲罰的、偶発的または結果的
    な損失、損害、費用または経費に対して責任を負わないものとします。法律で認められている最大限の範囲で、ｿﾌﾄｳｪｱに
    関連する全ての請求に対するMicrochipの全責任は、もしあれば、ｿﾌﾄｳｪｱのためにMicrochipへ直接支払った料金を超え
    ないものとします。 */

#define WINDOW_CMP_LOW_TH_EXAMPLE     (0x100)
#define F_CPU                         4000000UL       /* 主ｸﾛｯｸ周波数 */
#define START_TOKEN                   0x03            /* ﾌﾚｰﾑ開始通票 */
#define END_TOKEN                     0xFC            /* ﾌﾚｰﾑ終了通票 */
/* ﾎﾞｰﾚｰﾄ計算 */
#define USART1_BAUD_RATE(BAUD_RATE) (((float)F_CPU * 64 / (16 * (float)BAUD_RATE)) + 0.5)

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

void CLKCTRL_init(void);
void PORT_init(void);
void VREF0_init(void);
void ADC0_init(void);
void USART1_init(void);
uint16_t ADC0_read(void);
uint8_t ADC0_resultReady(void);
void ADC0_start(void);
uint8_t ADC0_resultBelowTreshold(void);
void USART1_Write(const uint8_t data);
void LED0_init(void);
void LED0_on(void);
void LED0_off(void);

/* この関数はCLKCTRL単位部を初期化します。 */
void CLKCTRL_init(void)
{
    /* 周波数選択(FREQSEL) 4MHz */
    ccp_write_io((void*)&(CLKCTRL.OSCHFCTRLA), (CLKCTRL.OSCHFCTRLA | CLKCTRL_FREQSEL_4M_gc));
}

/* この関数はPORT単位部を初期化します。 */
void PORT_init(void)
{
    /* PC0をUSART1 TX用出力として構成設定 */
    PORTC.DIRSET = PIN0_bm;
```

```
    /* PD3で割り込みとﾃﾞｼﾞﾀﾙ入力緩衝部を禁止 */
    PORTD.PIN3CTRL &= ~PORT_ISC_gm;
    PORTD.PIN3CTRL |= PORT_ISC_INPUT_DISABLE_gc;

    /* ﾌﾟﾙｱｯﾌﾟ抵抗を禁止 */
    PORTD.PIN3CTRL &= ~PORT_PULLUPEN_bm;
}

/* この関数はVREF単位部を初期化します。 */
void VREF0_init(void)
{
    VREF.ADC0REF = VREF_REFSEL_2V048_gc;            /* 内部2.048V参照基準 */
}

/* この関数はADC単位部を初期化します。 */
void ADC0_init(void)
{
    ADC0.CTRLC = ADC_PRESC_DIV4_gc;                 /* 4分周したCLK_PER */
    ADC0.CTRLA = ADC_ENABLE_bm                      /* ADC許可: 許可 */
               | ADC_RESSEL_12BIT_gc                /* 12ﾋﾞｯﾄ動作 */
               | ADC_FREERUN_bm;                    /* 自由走行動作許可 */
    ADC0.MUXPOS = ADC_MUXPOS_AIN3_gc;               /* ADCﾁｬﾈﾙAIN3(PD3)選択 */
    ADC0.WINLT = WINDOW_CMP_LOW_TH_EXAMPLE;         /* 変換窓比較器下側閾値設定 */
    ADC0.CTRLE = ADC_WINCM_BELOW_gc;                /* 変換窓比較動作設定 */
}

/* この関数はUSART単位部を初期化します。 */
void USART1_init(void)
{
    /* ﾎﾞｰﾚｰﾄを115200に構成設定 */
    USART1.BAUD = (uint16_t)USART1_BAUD_RATE(115200);
    USART1.CTRLB = USART_TXEN_bm;                   /* 送信部許可 */
    USART1.CTRLC = USART_CHSIZE_8BIT_gc;            /* 文字の大きさを8ﾋﾞｯﾄに構成設定 */
}

/* この関数はADC変換結果を返します。 */
uint16_t ADC0_read(void)
{
    /* 結果読み込みによって割り込み要求ﾌﾗｸﾞを解除(0) */
    return ADC0.RES;
}

uint8_t ADC0_resultReady(void)
{
    return (ADC0.INTFLAGS & ADC_RESRDY_bm);
}

/* この関数はADC変換を開始します。 */
void ADC0_start(void)
{
    /* 変換開始 */
    ADC0.COMMAND = ADC_STCONV_bm;
}

uint8_t ADC0_resultBelowTreshold(void)
{
    return (ADC0.INTFLAGS & ADC_WCMP_bm);
}

/* この関数はUSARTを通して1ﾊﾞｲﾄを送信します。 */
void USART1_Write(const uint8_t data)
```

```
{
    /* USART緩衝部がデータ送信準備可か調査 */
    while (!(USART1.STATUS & USART_DREIF_bm));
    /* TXDATALレジスタを使ってデータを送信 */
    USART1.TXDATAL = data;
}

/* この関数はLEDピンを初期化します。 */
void LED0_init(void)
{
    /* ピンを出力として構成設定 */
    PORTC.DIRSET = PIN6_bm;
    /* 出力をHigh(LED=OFF)に構成設定 */
    PORTC.OUTSET = PIN6_bm;
}

void LED0_on(void)
{
    /* 出力をLow(LED=ON)に構成設定 */
    PORTC.OUTCLR = PIN6_bm;
}

void LED0_off(void)
{
    /* 出力をHigh(LED=OFF)に構成設定 */
    PORTC.OUTSET = PIN6_bm;
}

int main(void)
{
    uint16_t adcVal;

    /* 全ての周辺機能を初期化 */
    CLKCTRL_init();
    PORT_init();
    VREF0_init();
    ADC0_init();
    USART1_init();
    LED0_init();

    /* ADC変換開始 */
    ADC0_start();

    while (1)
    {
        while (!ADC0_resultReady());

        if (ADC0_resultBelowTreshold())
        {
            LED0_on();
        }
        else
        {
            LED0_off();
        }

        adcVal = ADC0_read();

        /* データ可視器を使って作図されるべきADC結果を送信 */
        USART1_Write(START_TOKEN);
        USART1_Write(adcVal & 0x00FF);
```

```
        USART1_Write(adcVal >> 8);
        USART1_Write(END_TOKEN);
    }
}
```

例13-6. ADC事象起動ｺｰﾄﾞ例

```
/*  ¥file  main.c
    ¥brief      RTC溢れによるADC事象起動
    (c) 2019 Microchip Technology Inc.とその子会社
    これらの条件を遵守するのを条件にMicrochipｿﾌﾄｳｪｱとMicrochip製品でのどの占的派生物も使うことができます。
    Microchipｿﾌﾄｳｪｱに付随し得る(ｵｰﾌﾟﾝ ｿｰｽ ｿﾌﾄｳｪｱを含む)第三者ｿﾌﾄｳｪｱの使用に対して適用できる第三者許諾条項を
    遵守するにはあなたの責任です。
    このｿﾌﾄｳｪｱはMicrochipによって「現状そのまま」で供給されます。Microchipは非侵害、商品性、特定目的に対する適合
    性の何れの黙示的保証を含め、このｿﾌﾄｳｪｱに適用する、明示的、黙示的、法的であろうとなかろうと保証しません。
    如何なる場合においても、Microchipはｿﾌﾄｳｪｱに関連するあらゆる種類の間接的、特別的、懲罰的、偶発的または結果的
    な損失、損害、費用または経費に対して責任を負わないものとします。法律で認められている最大限の範囲で、ｿﾌﾄｳｪｱに
    関連する全ての請求に対するMicrochipの全責任は、もしあれば、ｿﾌﾄｳｪｱのためにMicrochipへ直接支払った料金を超え
    ないものとします。
*/
/* RTC周期 */
#define RTC_PERIOD       (511)                  /* RTC period */
#define F_CPU            4000000UL              /* 主ｸﾛｯｸ周波数 */
#define START_TOKEN      0x03                   /* ﾌﾚｰﾑ開始通票 */
#define END_TOKEN        0xFC                   /* ﾌﾚｰﾑ終了通票 */
/* ﾎﾞｰﾚｰﾄ計算 */
#define USART1_BAUD_RATE(BAUD_RATE) (((float)F_CPU * 64 / (16 * (float)BAUD_RATE)) + 0.5)
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/cpufunc.h>
#include <stdbool.h>
volatile uint16_t adcVal;
volatile bool adcResultReady = 0;
void CLKCTRL_init(void);
void PORT_init(void);
void VREF0_init(void);
void ADC0_init(void);
void LED0_init(void);
void USART1_init();
void LED0_toggle(void);
void USART1_Write(const uint8_t data);
void RTC_init(void);
void EVSYS_init(void);
/* この関数はCLKCTRL単位部を初期化します。 */
void CLKCTRL_init(void)
{
    /* 周波数選択(FREQSEL) 4MHz */
    ccp_write_io((void*)&(CLKCTRL.OSCHFCTRLA), (CLKCTRL.OSCHFCTRLA | CLKCTRL_FREQSEL_4M_gc));
}
/* この関数はPORT単位部を初期化します。 */
void PORT_init(void)
{
    /* PC0をUSART1 TX用出力として構成設定 */
    PORTC.DIRSET = PIN0_bm;

    /* PD3で割り込みとﾃﾞｼﾞﾀﾙ入力緩衝部を禁止 */
    PORTD.PIN3CTRL &= ~PORT_ISC_gm;
    PORTD.PIN3CTRL |= PORT_ISC_INPUT_DISABLE_gc;

    /* ﾌﾟﾙｱｯﾌﾟ抵抗を禁止 */
    PORTD.PIN3CTRL &= ~PORT_PULLUPEN_bm;
}
```

```
/* この関数はVREF単位部を初期化します。 */
void VREF0_init(void)
{
    VREF.ADC0REF = VREF_REFSEL_2V048_gc;        /* 内部2.048V参照基準 */
}
/* この関数はADC単位部を初期化します。 */
void ADC0_init(void)
{
    ADC0.CTRLC = ADC_PRESC_DIV4_gc;             /* 4分周したCLK_PER */
    ADC0.CTRLA = ADC_ENABLE_bm                  /* ADC許可: 許可 */
               | ADC_RESSEL_12BIT_gc;           /* 12ﾋﾞｯﾄ動作 */
    ADC0.MUXPOS = ADC_MUXPOS_AIN3_gc;           /* ADCﾁｬﾈﾙAIN3(PD3)選択 */
    ADC0.INTCTRL |= ADC_RESRDY_bm;              /* 割り込み許可 */
    ADC0.EVCTRL |= ADC_STARTEI_bm;              /* 事象起動変換許可 */
}
/* この関数はLEDﾋﾟﾝを初期化します。 */
void LED0_init(void)
{
    /* ﾋﾟﾝを出力として構成設定 */
    PORTC.DIRSET = PIN6_bm;
    /* 出力をHigh(LED=OFF)に構成設定 */
    PORTC.OUTSET = PIN6_bm;
}
/* この関数はUSART単位部を初期化します。 */
void USART1_init()
{
    /* ﾎﾞｰﾚｰﾄを115200に構成設定 */
    USART1.BAUD = (uint16_t)USART1_BAUD_RATE(115200);
    USART1.CTRLB = USART_TXEN_bm;               /* 送信部許可 */
    USART1.CTRLC = USART_CHSIZE_8BIT_gc;        /* 文字の大きさを8ﾋﾞｯﾄに構成設定 */
}
/* この関数はLEDﾋﾟﾝを交互切り替えします。 */
void LED0_toggle(void)
{
    PORTC.OUTTGL = PIN6_bm;
}
/* この関数はUSARTを通して1ﾊﾞｲﾄを送信します。 */
void USART1_Write(const uint8_t data)
{
    /* USART緩衝部がﾃﾞｰﾀ送信準備可か調査 */
    while (!(USART1.STATUS & USART_DREIF_bm));
    /* TXDATALﾚｼﾞｽﾀを使ってﾃﾞｰﾀを送信 */
    USART1.TXDATAL = data;
}
ISR(ADC0_RESRDY_vect)
{
    /* 結果読み込みによって割り込み要求ﾌﾗｸﾞを解除(0) */
    adcVal = ADC0.RES;
    /* ﾌﾗｸﾞを更新 */
    adcResultReady = 1;
}
/* この関数はRTC単位部を初期化します。 */
void RTC_init(void)
{
    /* RTC初期化 */
    while (RTC.STATUS > 0)
    {
        ; /* 全ﾚｼﾞｽﾀ同期待ち */
    }
    RTC.CTRLA = RTC_PRESCALER_DIV32_gc          /* 32分周 */
              | RTC_RTCEN_bm                    /* 許可: 許可 */
```

```
                | RTC_RUNSTDBY_bm;                    /* スタンバイで走行: 許可 */
    RTC.PER = RTC_PERIOD;                           /* 周期設定 */
    RTC.CLKSEL = RTC_CLKSEL_OSC32K_gc;              /* 32.768kHz内部発振器(OSC32K) */
}
/* この関数はEVSYS単位部を初期化します。 */
void EVSYS_init(void)
{
    /* 実時間計数器溢れ */
    EVSYS.CHANNEL0 = EVSYS_CHANNEL0_RTC_OVF_gc;
    /* 使用部を事象チャネル0に接続 */
    EVSYS.USERADC0START = EVSYS_USER_CHANNEL0_gc;
}
int main(void)
{
    uint16_t result;

    /* 全ての周辺機能を初期化 */
    CLKCTRL_init();
    PORT_init();
    VREF0_init();
    ADC0_init();
    LED0_init();
    USART1_init();
    RTC_init();
    EVSYS_init();

    /* 全体割り込み許可 */
    sei();

    while (1)
    {
        if (adcResultReady == 1)
        {
            /* 割り込みのためのadcVal変わりを避けるため結果を格納。この操作は非分断でなければなりません。 */
            cli();
            result = adcVal;
            sei();

            /* フラグ値を更新 */
            adcResultReady = 0;
            /* LEDを交互切り替え */
            LED0_toggle();

            /* データ可視器を使って作図されるべきADC結果を送信 */
            USART1_Write(START_TOKEN);
            USART1_Write(result & 0x00FF);
            USART1_Write(result >> 8);
            USART1_Write(END_TOKEN);
        }
    }
}
```

例13-7. ADC温度測定コード例

```
/*  ¥file  main.c
    ¥brief      ADC温度測定
    (c) 2019 Microchip Technology Inc.とその子会社
    これらの条件を遵守するのを条件にMicrochipソフトウェアとMicrochip製品でのどの占的派生物も使うことができます。
    Microchipソフトウェアに付随し得る(オープン ソース ソフトウェアを含む)第三者ソフトウェアの使用に対して適用できる第三者許諾条項を
    遵守するにはあなたの責任です。
    このソフトウェアはMicrochipによって「現状そのまま」で供給されます。Microchipは非侵害、商品性、特定目的に対する適合
    性の何れの黙示的保証を含め、このソフトウェアに適用する、明示的、黙示的、法的であろうとなかろうと保証しません。
```

```
    如何なる場合においても、Microchipはｿﾌﾄｳｪｱに関連するあらゆる種類の間接的、特別的、懲罰的、偶発的または結果的な損失、損害、費用または経費に対して責任を負わないものとします。法律で認められている最大限の範囲で、ｿﾌﾄｳｪｱに関連する全ての請求に対するMicrochipの全責任は、もしあれば、ｿﾌﾄｳｪｱのためにMicrochipへ直接支払った料金を超えないものとします。
*/
#define F_CPU               4000000UL               /* 主ｸﾛｯｸ周波数 */
#define START_TOKEN         0x03                    /* ﾌﾚｰﾑ開始通票 */
#define END_TOKEN           0xFC                    /* ﾌﾚｰﾑ終了通票 */
/* ﾎﾞｰﾚｰﾄ計算 */
#define USART1_BAUD_RATE(BAUD_RATE) (((float)F_CPU * 64 / (16 * (float)BAUD_RATE)) + 0.5)
#include <avr/io.h>
#include <stdbool.h>
#include <avr/cpufunc.h>
void CLKCTRL_init(void);
void VREF0_init(void);
void ADC0_init(void);
void USART1_init(void);
uint16_t ADC0_read(void);
int16_t temperatureConvert(uint16_t data);
void ADC0_start(void);
void USART1_Write(const uint8_t data);
/* この関数はCLKCTRL単位部を初期化します。 */
void CLKCTRL_init(void)
{
    /* 周波数選択(FREQSEL) 4MHz */
    ccp_write_io((void*)&(CLKCTRL.OSCHFCTRLA), (CLKCTRL.OSCHFCTRLA | CLKCTRL_FREQSEL_4M_gc));
}
/* この関数はPORT単位部を初期化します。 */
void PORT_init(void)
{
    /* PC0をUSART1 TX用出力として構成設定 */
    PORTC.DIRSET = PIN0_bm;
}
/* この関数はVREF単位部を初期化します。 */
void VREF0_init(void)
{
    VREF.ADC0REF = VREF_REFSEL_2V048_gc;        /* 内部2.048V参照基準 */
}
/* この関数はADC単位部を初期化します。 */
void ADC0_init(void)
{
    ADC0.CTRLC = ADC_PRESC_DIV4_gc;             /* 4分周したCLK_PER */
    ADC0.CTRLA = ADC_ENABLE_bm                  /* ADC許可: 許可 */
               | ADC_RESSEL_12BIT_gc;           /* 12ﾋﾞｯﾄ動作 */
               | ADC_FREERUN_bm;                /* 自由走行動作 */
    ADC0.MUXPOS = ADC_MUXPOS_TEMPSENSE_gc;      /* 温度ADCﾁｬﾈﾙを選択 */
}
/* この関数はUSART単位部を初期化します。 */
void USART1_init(void)
{
    /* ﾎﾞｰﾚｰﾄを115200に構成設定 */
    USART1.BAUD = (uint16_t)USART1_BAUD_RATE(115200);
    USART1.CTRLB = USART_TXEN_bm;               /* 送信部許可 */
    USART1.CTRLC = USART_CHSIZE_8BIT_gc;        /* 文字の大きさを8ﾋﾞｯﾄに構成設定 */
}
/* この関数はADC変換結果を返します */
uint16_t ADC0_read(void)
{
    /* 結果準備可待ち */
    while (!(ADC0.INTFLAGS & ADC_RESRDY_bm));
    /* 結果読み込みによって割り込み要求ﾌﾗｸﾞを解除(0) */
```

```
    return ADC0.RES;
}
/* この関数は摂氏(℃)での温度値を返します */
int16_t temperatureConvert(uint16_t data)
{
    uint16_t sigrow_offset = SIGROW.TEMPSENSE1;
    uint16_t sigrow_slope = SIGROW.TEMPSENSE0;
    int32_t temp;
    /* 変換結果の変位(ｵﾌｾｯﾄ)補正 */
    temp = sigrow_offset - data;
    /* 変換結果の利得補正(結果は16ﾋﾞｯﾄ変数を溢れます。) */
    temp *= sigrow_slope;
    /* 下の除算で正しい丸めを得るために4096/2を加算 */
    temp += 0x0800;
    /* 2^12(4096)で除算することによって最も近いｹﾙﾋﾞﾝ温度に四捨五入 */
    temp >>= 12;
    /* ｹﾙﾋﾞﾝから摂氏に変換(0ｹﾙﾋﾞﾝ - 273.15 = -273.1℃) */
    return temp - 273;
}
/* この関数はADC変換を開始します。 */
void ADC0_start(void)
{
    /* 変換開始 */
    ADC0.COMMAND = ADC_STCONV_bm;
}
/* この関数はUSARTを通して1ﾊﾞｲﾄを送信します。 */
void USART1_Write(const uint8_t data)
{
    /* USART緩衝部がﾃﾞｰﾀ送信準備可か調査 */
    while (!(USART1.STATUS & USART_DREIF_bm));
    /* TXDATALﾚｼﾞｽﾀを使ってﾃﾞｰﾀを送信 */
    USART1.TXDATAL = data;
}
int main(void)
{
    int16_t temp_C;
    uint16_t adcVal;

    /* 全ての周辺機能を初期化 */
    CLKCTRL_init();
    PORT_init();
    VREF0_init();
    ADC0_init();
    USART1_init();

    /* ADC変換開始 */
    ADC0_start();

    while (1)
    {
        /* 変換結果を読み込み */
        adcVal = ADC0_read();
        /* ADC結果を摂氏温度に変換 */
        temp_C = temperatureConvert(adcVal);

        /* ﾃﾞｰﾀ可視器を使って作図されるべきADC結果を送信 */
        USART1_Write(START_TOKEN);
        USART1_Write(temp_C & 0x00FF);
        USART1_Write(temp_C >> 8);
        USART1_Write(END_TOKEN);
    }
}
```

14. 改訂履歴

文書改訂	日付	注釈
A	2020年2月	初版文書公開
B	2020年3月	貯蔵庫ﾘﾝｸ更新 最新の商標により、AVR-DAをAVR® MCU DA (AVR-DA)に更新
C	2020年5月	最新の商標により、AVR® MCU DA (AVR-DA)をAVR® DA MCUに、AVR-DAをAVR DAに更新
D	2020年10月	GitHubﾛｺﾞ更新 ﾃﾞｰﾀ可視器(Data Visualozer)機能説明と手順を追加 ADC変換結果を送信するようにｺｰﾄﾞ例を更新

Microchipウェブ サイト

Microchipは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
- **全般的な技術支援** – 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、その製品系統が安全であると考えます。
- Microchipデバイスのコード保護機能を破ろうとする試みに使われる不正でおそらく違法な方法があります。当社はこれらの方法が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とされると確信しています。これらのコード保護機能を破ろうとする試みは、おそらく、Microchipの知的財産権に違反することなく達成することはできません。
- Microchipはそれのコードの完全性について心配されている何れのお客様とも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。コード保護は製品が”破ることができない”ことを当社が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

この刊行物に含まれる情報はMicrochip製品を使って設計する唯一の目的のために提供されます。デバイス応用などに関す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。

この情報はMicrochipによって「現状そのまま」で提供されます。Microchipは非侵害、商品性、特定目的に対する適合性の何れの黙示的保証やその条件、品質、性能に関する保証を含め、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。

如何なる場合においても、Microchipは情報またはその使用に関連するあらゆる種類の間接的、特別的、懲罰的、偶発的または結果的な損失、損害、費用または経費に対して責任を負わないものとします。法律で認められている最大限の範囲で、情報またはその使用に関連する全ての請求に対するMicrochipの全責任は、もしあれば、情報のためにMicrochipへ直接支払った料金を超えないものとします。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の名前とﾛｺﾞ、Mcicrochipﾛｺﾞ、Adaptec、AnyRate、AVR、AVRﾛｺﾞ、AVR Freaks、BesTime、BitCloud、chipKIT、chipKITﾛｺﾞ、CryptoMemory、CryptoRF、dsPIC、FlashFlex、flexPWR、HELDO、IGLOO、JukeBlox、KeeLoq、Kleer、LANCheck、LinkMD、maXStylus、maXTouch、MediaLB、megaAVR、Microsemi、Microsemiﾛｺﾞ、MOST、MOSTﾛｺﾞ、MPLAB、OptoLyzer、PackeTime、PIC、picoPower、PICSTART、PIC32ﾛｺﾞ、PolarFire、Prochip Designer、QTouch、SAM-BA、SenGenuity、SpyNIC、SST、SSTﾛｺﾞ、SuperFlash、Symmetricom、SyncServer、Tachyon、TimeSource、tinyAVR、UNI/O、Vectron、XMEGAは米国と他の国に於けるMicrochip Technology Incor poratedの登録商標です。

AgileSwitch、APT、ClockWorks、The Embedded Control Solutions Company、EtherSynch、FlashTec、Hyper Speed Control、Hyper Light Load、IntelliMOS、Libero、motorBench、mTouch、Powermite 3、Precision Edge、ProASIC、ProASIC Plus、ProASIC Plusﾛｺﾞ、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、Augmented Switching、BlueSky、BodyCom、CodeGuard、CryptoAuthentication、CryptoAutomotive、CryptoCompanion、CryptoController、dsPICDEM、dsPICDEM.net、Dynamic Average Matching、DAM、ECAN、Espresso T1S、EtherGREEN、IdealBridge、In-Circuit Serial Programming、ICSP、INICnet、Intelligent Paralleling、Inter-Chip Connectivity、JitterBlocker、maxCrypto、maxView、memBrain、Mindi、MiWi、MPASM、MPF、MPLAB Certifiedﾛｺﾞ、MPLIB、MPLINK、MultiTRAK、NetDetach、Omniscient Code Generation、PICDEM、PICDEM.net、PICkit、PICtail、PowerSmart、PureSilicon、QMatrix、REAL ICE、Ripple Blocker、RTAX、RTG4、SAM-ICE、Serial Quad I/O、simpleMAP、SimpliPHY、SmartBuffer、SMART-I.S.、storClad、SQI、SuperSwitcher、SuperSwitcher II、Switchtec、SynchroPHY、Total Endurance、TSHARC、USBCheck、VariSense、VectorBlox、VeriPHY、ViewSpan、WiperLock、XpressConnect、and ZENA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ﾛｺﾞ、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を訪ねてください。

日本語© *HERO* 2020.

本技術概説はMicrochipのTB3245技術概説(DS90003245D-2020年10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訳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。

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
本社 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 技術支援: www.microchip.com/support ｳｪﾌﾞｱﾄﾞﾚｽ: www.microchip.com **ｱﾄﾗﾝﾀ** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **ｵｰｽﾁﾝ** TX Tel: 512-257-3370 **ﾎﾞｽﾄﾝ** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **ｼｶｺﾞ** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **ﾀﾞﾗｽ** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **ﾃﾞﾄﾛｲﾄ** Novi, MI Tel: 248-848-4000 **ﾋｭｰｽﾄﾝ** TX Tel: 281-894-5983 **ｲﾝﾃﾞｱﾅﾎﾟﾘｽ** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **ﾛｻﾝｾﾞﾙｽ** 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 **ｻﾝﾎｾ** CA Tel: 408-735-9110 Tel: 408-436-4270 **ｶﾅﾀﾞ - ﾄﾛﾝﾄ** Tel: 905-695-1980 Fax: 905-695-2078	**ｵｰｽﾄﾗﾘｱ - ｼﾄﾞﾆｰ** 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 - 香港特別行政区** 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	**ｲﾝﾄﾞ - ﾊﾝｶﾞﾛｰﾙ** Tel: 91-80-3090-4444 **ｲﾝﾄﾞ - ﾆｭｰﾃﾞﾘｰ** Tel: 91-11-4160-8631 **ｲﾝﾄﾞ - ﾌﾟﾈｰ** 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 - ｸｱﾗﾙﾝﾌﾟｰﾙ** Tel: 60-3-7651-7906 **ﾏﾚｰｼｱ - ﾍﾟﾅﾝ** Tel: 60-4-227-8870 **ﾌｨﾘﾋﾟﾝ - ﾏﾆﾗ** Tel: 63-2-634-9065 **ｼﾝｶﾞﾎﾟｰﾙ** 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 - ﾊﾞﾝｺｸ** Tel: 66-2-694-1351 **ﾍﾞﾄﾅﾑ - ﾎｰﾁﾐﾝ** Tel: 84-28-5448-2100	**ｵｰｽﾄﾘｱ - ｳﾞｪﾙｽ** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **ﾃﾞﾝﾏｰｸ - ｺﾍﾟﾝﾊｰｹﾞﾝ** Tel: 45-4485-5910 Fax: 45-4485-2829 **ﾌｨﾝﾗﾝﾄﾞ - ｴｽﾎﾟｰ** Tel: 358-9-4520-820 **ﾌﾗﾝｽ - ﾊﾟﾘ** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **ﾄﾞｲﾂ - ｶﾞﾙﾋﾝｸﾞ** Tel: 49-8931-9700 **ﾄﾞｲﾂ - ﾊｰﾝ** Tel: 49-2129-3766400 **ﾄﾞｲﾂ - ﾊｲﾙﾌﾞﾛﾝ** Tel: 49-7131-72400 **ﾄﾞｲﾂ - ｶｰﾙｽﾙｰｴ** Tel: 49-721-625370 **ﾄﾞｲﾂ - ﾐｭﾝﾍﾝ** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **ﾄﾞｲﾂ - ﾛｰｾﾞﾝﾊｲﾑ** 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 - ﾊﾟﾄﾞﾊﾞ** Tel: 39-049-7625286 **ｵﾗﾝﾀﾞ - ﾃﾞﾙｰﾈﾝ** Tel: 31-416-690399 Fax: 31-416-690340 **ﾉﾙｳｪｰ - ﾄﾛﾝﾊｲﾑ** Tel: 47-72884388 **ﾎﾟｰﾗﾝﾄﾞ - ﾜﾙｼｬﾜ** Tel: 48-22-3325737 **ﾙｰﾏﾆｱ - ﾌﾞｶﾚｽﾄ** Tel: 40-21-407-87-50 **ｽﾍﾟｲﾝ - ﾏﾄﾞﾘｰﾄﾞ** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **ｽｳｪｰﾃﾞﾝ - ｲｪｰﾃﾎﾞﾘ** Tel: 46-31-704-60-40 **ｽｳｪｰﾃﾞﾝ - ｽﾄｯｸﾎﾙﾑ** Tel: 46-8-5090-4654 **ｲｷﾞﾘｽ - ｳｫｰｷﾝｶﾞﾑ** Tel: 44-118-921-5800 Fax: 44-118-921-5820