TB3209

A/D変換器(ADC)での開始に際して

序説

著者: Victor Berzan, Microchip Technology Inc.

ｱﾅﾛｸﾞ⇒ﾃﾞｼﾞﾀﾙ変換器(A/D変換器:ADC)周辺機能はｱﾅﾛｸﾞ電圧を数値に変換します。この周辺機能は多くのAVR®ﾏｲｸﾛ ｺﾝﾄﾛｰﾗ(MCU)に含まれます。tinyAVR®とmegaAVR®の殆どのMCUで10ﾋﾞｯﾄ ｼﾝｸﾞﾙ ｴﾝﾄﾞADC周辺機能が利用可能な一方で、AVR DA系では12ﾋﾞｯﾄの差動とｼﾝｸﾞﾙ ｴﾝﾄﾞが利用可能なADC周辺機能があります。

この技術概要はmegaAVR® 0系とAVR DAﾏｲｸﾛ ｺﾝﾄﾛｰﾗでA/D変換器単位部がどう動くかを記述します。これは以下の使用事例を網羅します。

- **ADC単独変換:**

  ADCを初期化、変換を開始、変換が終わるまで待ち、ADC結果を読みます。

- **ADC自由走行:**

  ADCを初期化、自由走行(連続変換)動作を許可、変換を開始、無限繰り返し内で変換が終わるまで待ってADC結果を読みます。

- **ADC採取累積器:**

  ADCを初期化、64採取の累積を許可、変換を開始、繰り返し内で変換が終わるまで待ってADC結果を読みます。

- **ADC窓比較器:**

  ADCを初期化、変換窓比較器下側閾値を設定、変換窓動作を許可、自由走行(連続変換)動作を許可、変換を開始、無限繰り返し内で変換が終わるまで待ってADC結果を読み、ADC結果が設定した閾値未満の場合にLEDを点灯します。

- **事象起動ADC:**

  ADCを初期化、実時間計数器(RTC:Real Time Counter)を初期化、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、各ADC変換後にLEDを切り替えます。

注: この文書で記述される各使用事例に対して2つのｺｰﾄﾞ例があります。1つはATmega4809で素の状態で開発され、1つはAVR128DA48で開発されたMPLAB®ｺｰﾄﾞ構成部(MCC)で生成されました。

GitHubでATmega4809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。

GitHubでAVR128DA48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。

目次

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。以下の図はﾋﾟﾝ数の変種とﾒﾓﾘ量を展開して各種系統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互換で同じまたはより多くの機能を提供するため、垂直上方向移植はｺｰﾄﾞ変更なしで可能です。tinyAVR® 1系統ﾃﾞﾊﾞｲｽでの下方向移植はいくつかの周辺機能のより少ない利用可能な実体のため、ｺｰﾄﾞ変更を必要とするかもしれません。
- 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。
- 異なるﾌﾗｯｼｭ ﾒﾓﾘ量を持つﾃﾞﾊﾞｲｽはまた一般的に異なるSRAMとEEPROMの量を持ちます。

図1-1. tinyAVR® 0系統概要

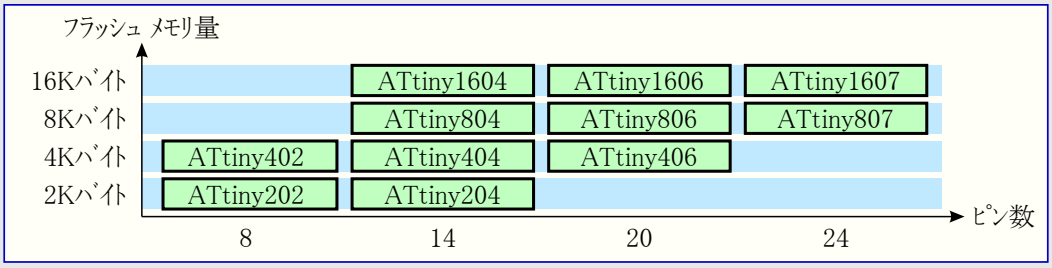


図1-2. tinyAVR® 1系統概要

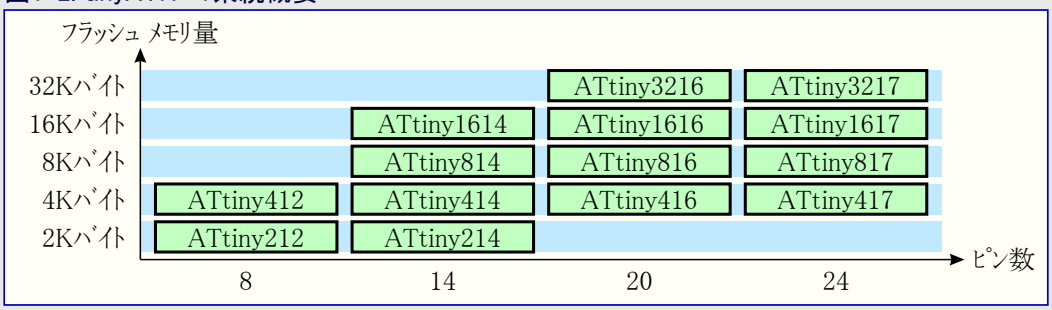


図1-3. megaAVR® 0系統概要

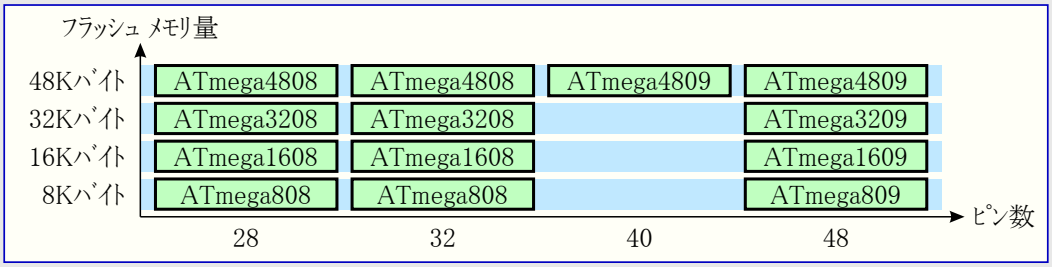
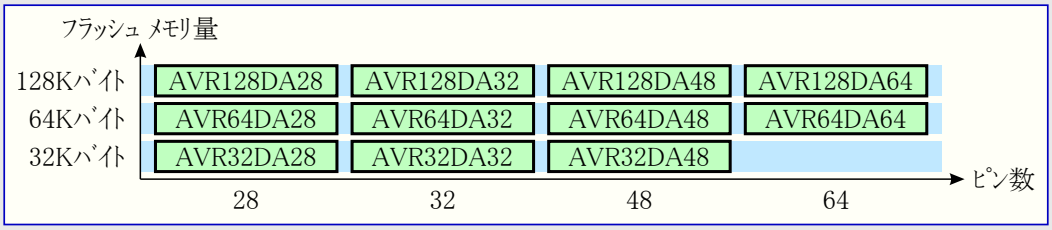


図1-4. AVR® DA系統概要

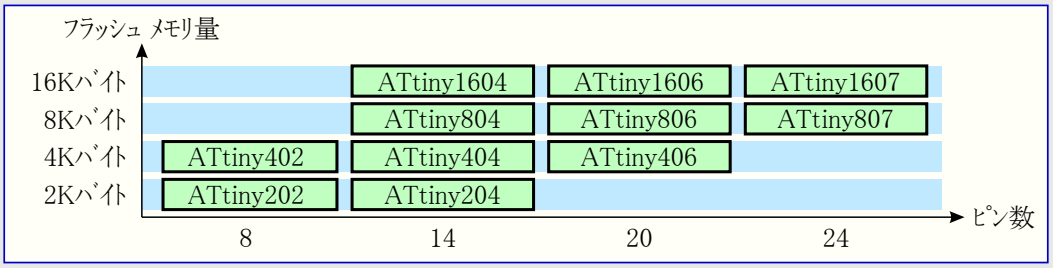


1.1. tinyAVR® 0系統

下図はﾋﾟﾝ数の変種とﾒﾓﾘ量を展開してtinyAVR® 0系統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1-5. tinyAVR® 0系統概要



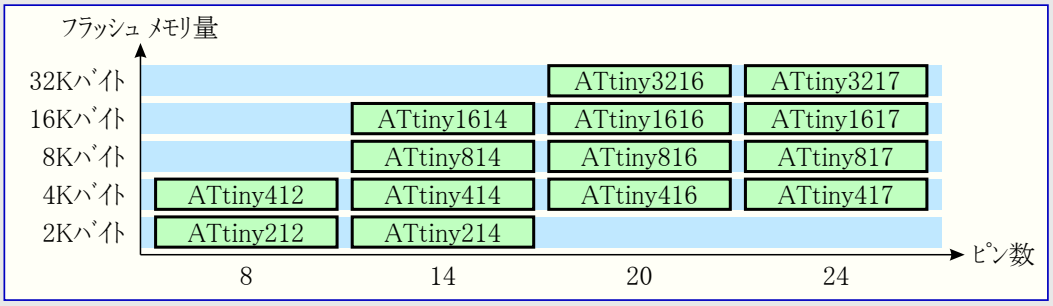
異なるﾌﾗｯｼｭ メモリ量を持つﾃﾞﾊﾞｲｽは一般的に異なるSRAMとEEPROMの量を持ちます。

1.2. tinyAVR® 1系統

下図はﾋﾟﾝ配置変種とﾒﾓﾘ量を展開してtinyAVR® 1系統ﾃﾞﾊﾞｲｽを示します。

- これらのﾃﾞﾊﾞｲｽがﾋﾟﾝ互換で同じまたはより多くの機能を提供するため、垂直上方向移植はｺｰﾄﾞ変更なしに可能です。下方向移植はより少ない利用可能ないくつかの周辺機能の実体のためにｺｰﾄﾞ変更が必要かもしれません。
- 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1-6. tinyAVR® 1系統概要



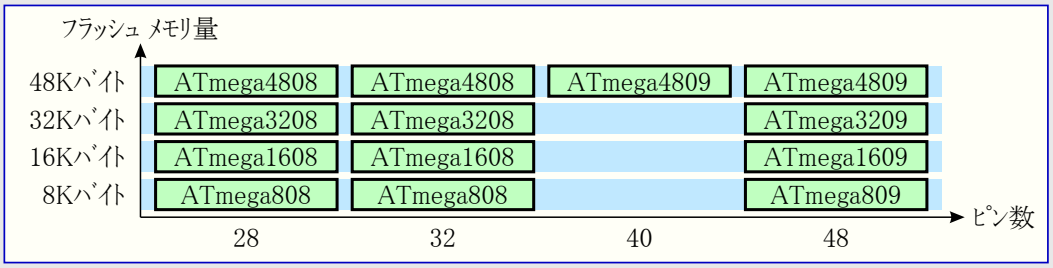
異なるﾌﾗｯｼｭ メモリ量を持つﾃﾞﾊﾞｲｽは一般的に異なるSRAMとEEPROMの量を持ちます。

1.3. megaAVR® 0系統

下図はﾋﾟﾝ配置変種とﾒﾓﾘ量を展開してmegaAVR® 0系統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1-7. megaAVR® 0系統概要



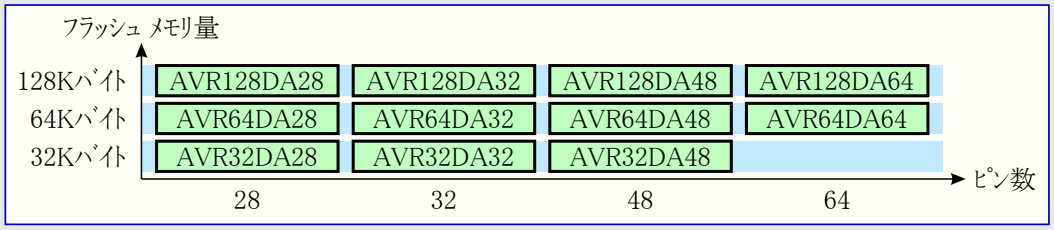
異なるﾌﾗｯｼｭ メモリ量を持つﾃﾞﾊﾞｲｽは一般的に異なるSRAMとEEPROMの量を持ちます。

1.4. AVR® DA系概要

次図はﾋﾟﾝ配置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1-8. AVR® DA系統概要



異なるフラッシュ メモリ量を持つデバイスは一般的に異なるSRAM量を持ちます。

2. 概要

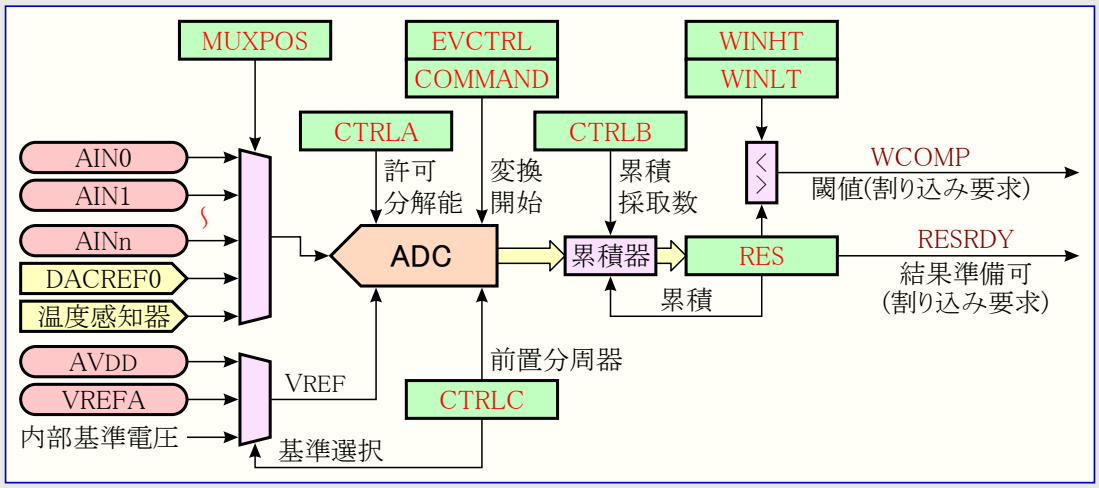
A/D変換器(ADC)周辺機能はtinyAVRとmegaAVRのマイクロ コントローラで10ビットの結果、AVR DAマイクロ コントローラで10ビット/12ビットの結果を生じます。ADC入力は内部(例えば、基準電圧)またはアナログ入力ピンを通す外部のどちらかにすることができます。ADCは多数のシングル エンド電圧入力を許すアナログ多重器に接続されます。シングル エンド電圧入力は0V(GND)を参照基準にします。

ADCは構成設定可能な変換結果数が単一ADC結果内に累積される集中での採取(採取累積)を支援します。ADC入力信号は採取中にADC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、VDD供給電圧、または外部VREF(VREFA)ピンから選択可能な基準電圧。

窓比較機能は入力信号を監視するのに使用可能で、必要とされる最小のソフトウェア介在で窓の下(未満)、上(超え)、内側、外側に対する使用者定義閾値で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



アナログ入力チャネルは多重器正選択(ADCn.MUXPOS)レジスタの多重器正選択(MUXPOS)ビットに書くことによって選ばれます。ADC入力ピンのどれか、GND、内部基準電圧(VREF)、または温度感知器をADCへのシングル エンド入力として選ぶことができます。ADCは制御A(ADCn.CTRLA)レジスタのADC許可(ENABLE)ビットに'1'を書くことによって許可されます。基準電圧と入力チャネルの選択はADCが許可される前に有効になりません。ADCはADCn.CTRLAレジスタのENABLEビットが'0'の時に電力を消費しません。

ADCはtinyAVRとmegaAVRのマイクロ コントローラで10ビットの結果を、AVR DAマイクロ コントローラで10ビット/12ビットの結果を生成します。これは結果(ADCn.RES)レジスタから読むことができます。この結果は右揃えで表されます。10ビット シングル エンド変換に対する結果は次のように与えられます。

$$\mathrm{RES} = \frac{V_{IN} \times 1024}{V_{REF}}$$

ここでのV_{IN}は選択した入力ピンの電圧で、V_{REF}は選択した基準電圧です。

12ビット シングル エンド変換に対する結果は次のように与えられます。

$$\mathrm{RES} = \frac{V_{IN} \times 4096}{V_{REF}}$$

3. ADC単独変換

ADCを使う最も簡単な動作は単独変換にすることです。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。この例ではADC入力にPD6/AIN6ﾋﾟﾝが使われます。

図3-1. ADCn.MUXPOS選択

ﾋﾞｯﾄ	7	6	5	4	3	2	1	0
				MUXPOS4~0				
ｱｸｾｽ種別	R	R	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4~0 – MUXPOS4~0：多重器正選択** (MUXPOS)

このﾋﾞｯﾄ領域はADCにどのｼﾝｸﾞﾙ ｴﾝﾄﾞ ｱﾅﾛｸﾞ入力が接続されるかを選びます。これらのﾋﾞｯﾄ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有効ではありません。

MUXPOS	00000～01111	10000～11011	11100	11101	11110	11111
名称	AIN0～AIN15	–	DACREF0	–	TEMPSENSE	GND
入力	ADC入力0～15ﾋﾟﾝ	(予約)	AC0のDACREF	(予約)	温度感知器	GND

```
ADC0.MUXPOS = ADC_MUXPOS_AIN6_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。特にこの例ではｸﾛｯｸが4分周されます。ADCは可能な参照基準に対してVDD、外部参照基準、または内部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。

図3-2. ADCn.CTRLC参照基準電圧選択

ﾋﾞｯﾄ	7	6	5	4	3	2	1	0
		SAMPCAP	REFSEL1,0			PRESC2~0		
ｱｸｾｽ種別	R	R/W	R/W	R/W	R	R/W	R/W	R/W
ﾘｾｯﾄ値	0	0	0	0	0	0	0	0

● **ﾋﾞｯﾄ5,4 – REFSEL1,0：基準電圧選択** (Reference Selection)

これらのﾋﾞｯﾄはADCに対する基準電圧を選びます。

値	0 0	0 1	1 0	1 1
名称	INTERNAL	VDD	VREFA	–
説明	内部基準電圧	VDD	外部基準電圧(VREFA)	(予約)

● **ﾋﾞｯﾄ2~0 – PRESC2~0：前置分周器** (Prescaler)

これらのﾋﾞｯﾄは周辺機能ｸﾛｯｸ(CLK_PER)からADCｸﾛｯｸ(CLK_ADC)への整数分周比を定義します。

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	DIV2	DIV4	DIV8	DIV16	DIV32	DIV64	DIV128	DIV256
説明	CLK_PER/2	CLK_PER/4	CLK_PER/8	CLK_PER/16	CLK_PER/32	CLK_PER/64	CLK_PER/128	CLK_PER/256

```
ADC0.CTRLC |= ADC_PRESC_DIV4_gc;
ADC0.CTRLC |= ADC_REFSEL_INTREF_gc;
```

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

図3-3. ADCn.CTRLA分解能選択

ﾋﾞｯﾄ	7	6	5	4	3	2	1	0
	RUNSTDBY					RESSEL	FREERUN	ENABLE
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2 – RESSEL：分解能選択** (Resolution Selection)

このﾋﾞｯﾄはADC分解能を選びます。

図3-3 (続き). ADCn.CTRLA分解能選択

値	0	1
説明	全10ビット分解能。10ビットADCの結果はADC結果(ADCn.RES)レジスタに累積または格納されます。	8ビット分解能。変換結果はそれらがADC結果(ADCn.RES)レジスタに累積または格納される前に(上位)8ビットに切り詰められます。下位2ビットは破棄されます。

● **ビット0 – ENABLE : ADC許可** (ADC Enable)

値	0	1
説明	ADCは禁止されます。	ADCは許可されます。

```
ADC0.CTRLA |= ADC_RESSEL_10BIT_gc;
ADC0.CTRLA |= ADC_ENABLE_bm;
```

ADC変換はADCn.COMMANDレジスタのSTCONVビットを設定(1)することによって開始されます。

図3-4. ADCn.COMMAND – 変換開始

ビット	7	6	5	4	3	2	1	0
								STCONV
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 – STCONV : 変換開始** (Start Conversion)

このビットへの'1'書き込みは単一の変換を開始します。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'として読まれます。変換が完了すると、このビットは自動的に解除(0)されます。

変換進行中のこのビットへの'0'書き込みはその変換を中止します。

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

変換が終了されると、ADCn.INTFLAGSのRESRDYビットがハードウェアによって設定(1)になります。使用者はADC結果を読むのに先立ってこのビットが設定(1)になるのを待たなければなりません。

図3-5. ADCn.INTFLAGS – ハードウェア設定RESRDYビット

ビット	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

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

結果準備可割り込み要求フラグ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。このフラグはこのビット位置への'1'書き込み、または結果(ADCn.RES)レジスタ読み込みのどちらかによって解除(0)されます。このビットへの'0'書き込みは無効です。

```
while (!(ADC0.INTFLAGS & ADC_RESRDY_bm))
{
    ;
}
```

使用者は別の変換を開始する前にRESRDYビットに'1'を書くことによってRESRDYビットを解除(0)しなければなりません。

```
ADC0.INTFLAGS = ADC_RESRDY_bm;
```

変換結果はADCn.RESレジスタから読むことができます。

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

助言: 完全なコード例は「**追補**」章でも利用可能です。

GitHubでATmega4809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。

本章で記述されたのと同じ機能を持つAVR128DA48用のMCC生成されたｺｰﾄﾞ例は以下のここで見つけることができます。

GitHubでAVR128DA48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。

4. ADC自由走行

ADCを自由走行動作で構成設定すると、前の変換が完了した直後に次の変換が開始します。この動作を活性にするには通常のADC初期化に加えてADCn.CTRLAのFREERUNﾋﾞｯﾄが設定(1)されなければなりません。

図4-1. ADCn.CTRLA – FREERUNﾋﾞｯﾄ設定

ﾋﾞｯﾄ	7	6	5	4	3	2	1	0
	RUNSTDBY					RESSEL	FREERUN	ENABLE
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1 – FREERUN : 自由走行** (Free Running)

このﾋﾞｯﾄへの'1'書き込みはﾃﾞ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(COMMAND)ﾚｼﾞｽﾀの変換開始(STCONV)ﾋﾞｯﾄへの'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ｸﾞ(INTFLAGS)ﾚｼﾞｽﾀの結果準備可割り込み要求(RESRDY)ﾌﾗｸﾞによって合図されます。

```
ADC0.CTRLA |= ADC_FREERUN_bm;
```

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

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

その後にADC結果はWhile繰り返し内で読むことができます。

```
while(1)
{
    if (ADC0_conersionDone())
    {
        adcVal = ADC0_read();
    }
}
```

助言: 完全なｺｰﾄﾞ例は「**追補**」章でも利用可能です。

GitHubでATmega4809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。

本章で記述されたのと同じ機能を持つAVR128DA48用のMCC生成されたｺｰﾄﾞ例は以下のここで見つけることができます。

GitHubでAVR128DA48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。

5. ADC採取累積器

採取累積器動作ではADCが累積器ﾚｼﾞｽﾀで64採取まで加算することができ、従って信号を濾波して雑音を減らします。これ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を読み取り時に有用です。ｿﾌﾄｳｪｱでのそれら読み取りの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。この動作を活性にするには通常のADC初期化に加えてADCn.CTRLBﾚｼﾞｽﾀで採取累積数が設定されなければなりません。

図5-1. ADCn.CTRLB – SAMPNUMﾋﾞｯﾄ設定

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連続するADC採取結果がどれくらい自動的に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回の完全な変換で連続するADC採取結果の対応する数がADC結果(ADCn.RES)ﾚｼﾞｽﾀに累積されます。

値	000	001	010	011	100	101	110	111
名称	NONE	ACC2	ACC4	ACC8	ACC16	ACC32	ACC64	–
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	(予約)

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

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

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

採取(試料)はADCn.RESﾚｼﾞｽﾀで加算されます。ADCn.CTRLBで指定した採取数が取得された後にINTFLAGS.RESRDYﾌﾗｸﾞが設定(1)されます。

```
while (!(ADC0.INTFLAGS & ADC_RESRDY_bm))
{
    ;
}
```

使用者は平均値を得るためにその値を読んで採取数で除算することができます。

```
adcVal = ADC0.RES;
adcVal = adcVal >> ADC_SHIFT_DIV64;
```

使用者は別の変換を開始する前にRESRDYﾋﾞｯﾄに'1'を書くことによってRESRDYﾋﾞｯﾄを解除(0)しなければなりません。

```
ADC0.INTFLAGS = ADC_RESRDY_bm;
```

助言: 完全なｺｰﾄﾞ例は「追補」章でも利用可能です。

GitHubでATmega4809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。

本章で記述されたのと同じ機能を持つAVR128DA48用のMCC生成されたｺｰﾄﾞ例は以下のここで見つけることができます。

GitHubでAVR128DA48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。

6. ADC窓比較器

窓比較器動作ではADC結果が指定した閾値の下または上かをﾃﾞﾊﾞｲｽが検出します。これは指定範囲内を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/過充電などを合図するのに有用です。窓比較器は自由走行動作と単独変換動作の両方で使うことができます。監視される信号が継続的な採取を必要とし、自由走行動作が各変換に対する手動的な開始を必要としないことによってCPU負荷を減らすため、この例では窓比較器が自由走行動作で使われます。

この例についてはADCn.WINLTﾚｼﾞｽﾀで下側閾値が設定されます。

図6-1. ADCn.WINLT – 下側閾値設定

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。

```
ADC0.WINLT = WINDOW_CMP_LOW_TH_EXAMPLE;
```

変換窓動作形態はADCn.CTRLEレジスタで設定されます。

図6-2. ADCn.CTRLE – 窓比較器動作構成設定

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)

これら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。RESULT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)

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

ADC結果が設定した閾値未満の場合、ハードウェアによってADCn.INTFLAGSレジスタのWCOMPビットが設定(1)されます。

図6-3. ADC0.INTFLAGS – WCOMPビット ハードウェア設定

ビット	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

- **ビット1 – WCOMP：窓比較器割り込み要求フラグ** (Window Comparator Interrupt Flag)

この窓比較器割り込み要求フラグは測定が完了して結果が制御E(ADCn.CTRLE)レジスタの窓比較器動作(WINCM)によって定義されて選んだ窓比較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。このフラグはこのビット位置への'1'書き込み、または結果(ADCn.RES)レジスタ読み込みのどちらかによって解除(0)されます。このビットへの'0'書き込みは無効です。

使用者はこれに'1'を書くことによってこのビットを解除しなければなりません。

```
ADC0.INTFLAGS = ADC_WCMP_bm;
```

助言: 完全なコード例は「追補」章でも利用可能です。

GitHubでATmega4809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。

本章で記述されたのと同じ機能を持つAVR128DA48用のMCC生成されたｺｰﾄﾞ例は以下のここで見つけることができます。

GitHubでAVR128DA48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。

7. 事象起動ADC

ADC変換は事象によって起動することができます。これは事象制御(ADCn.EVCTRL)ﾚｼﾞｽﾀの事象入力開始(STARTEI)ﾋﾞｯﾄに'1'を書くことによって許可されます。

図7-1. ADCn.EVCTRL – STARTEIﾋﾞｯﾄ許可

ﾋﾞｯﾄ	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

● **ﾋﾞｯﾄ0 – STARTEI : 事象入力開始** (Start Event Input)

このﾋﾞｯﾄは変換開始用の供給元として事象入力を使うことを許可します。

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

事象ｼｽﾃﾑ(EVSYS)を通してADCに配線されるどの到着事象も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、ADCn.COMMANDのSTCONVが設定(1)されます。STCONVはその変換が完了する時に解除(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溢れ事象は事象ｼｽﾃﾑのﾁｬﾈﾙ0に繋げら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EVCTRLﾚｼﾞｽﾀのSTARTEIﾋﾞｯﾄは事象によって起動されるADC変換を許可するために設定(1)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;                 /* 事象起動変換許可 */
```

助言: 完全なｺｰﾄﾞ例は「**追補**」章でも利用可能です。

GitHubでATmega4809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。

本章で記述されたのと同じ機能を持つAVR128DA48用のMCC生成されたｺｰﾄﾞ例は以下のここで見つけることができます。

GitHubでAVR128DA48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。

8. 参考文献

1. AN2573 – tinyAVR® 0と1系及びmegaAVR® 0系でのADC基礎
2. AVR128DA48製品頁：www.microchip.com/wwwproducts/en/AVR128DA48
3. AVR128DA48 Curiosity Nano評価ｷｯﾄ製品頁：https://www.microchip.com/Developmenttools/ProductDetails/DM164151
4. AVR128DA28/32/48/64ﾃﾞｰﾀｼｰﾄ
5. AVR® DA系での開始に際して
6. ATmega4809製品頁：www.microchip.com/wwwproducts/en/ATMEGA4809
7. megaAVR® 0系ﾃﾞｰﾀｼｰﾄ
8. megaAVR® 0系ATmega809/1609/3209/4809 – 48ﾋﾟﾝ ﾃﾞｰﾀｼｰﾄ
9. ATmega4809 Xplained Pro製品頁：https://www.microchip.com/developmenttools/ProductDetails/atmega4809-xpro

9. 改訂履歴

文書改訂	日付	注釈
A	2019年5月	初版文書公開
B	2021年3月	GitHub貯蔵庫ﾘﾝｸ、「参考文献」、使用事例章更新。「AVR® DA系概要」項と「改訂履歴」章追加。各使用事例に対してAVR128DA48で動くMCC版を追加。その他些細な修正。

10. 追補

例10-1. ADC単独変換ｺｰﾄﾞ例

```
/* RTC周期 */
#define RTC_PERIOD   (511)

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

uint16_t adcVal;

void ADC0_init(void);
uint16_t ADC0_read(void);

void ADC0_init(void)
{
    /* ﾃﾞｼﾞﾀﾙ入力緩衝部禁止 */
    PORTD.PIN6CTRL &= ~PORT_ISC_gm;
    PORTD.PIN6CTRL |= PORT_ISC_INPUT_DISABLE_gc;

    /* ﾌﾟﾙｱｯﾌﾟ抵抗禁止 */
    PORTD.PIN6CTRL &= ~PORT_PULLUPEN_bm;

    ADC0.CTRLC = ADC_PRESC_DIV4_gc        /* CLK_PERを4分周 */
               | ADC_REFSEL_INTREF_gc;    /* 内部参照基準 */

    ADC0.CTRLA = ADC_ENABLE_bm            /* ADC許可:許可 */
               | ADC_RESSEL_10BIT_gc;     /* 10ﾋﾞｯﾄ動作 */

    /* Select ADC channel */
    ADC0.MUXPOS = ADC_MUXPOS_AIN6_gc;
}

uint16_t ADC0_read(void)
{
    /* ADC変換開始 */
    ADC0.COMMAND = ADC_STCONV_bm;

    /* ADC変換終了まで待機 */
    while ( !(ADC0.INTFLAGS & ADC_RESRDY_bm) )
    {
        ;
    }

    /* 1書き込みによって割り込み要求ﾌﾗｸﾞ解除(0) */
    ADC0.INTFLAGS = ADC_RESRDY_bm;

    return ADC0.RES;
}

int main(void)
{
    ADC0_init();

    adcVal = ADC0_read();

    while (1)
    {
        ;
    }
}
```

例10-2. ADC自由走行ｺｰﾄﾞ例

```
#include <avr/io.h>
#include <stdbool.h>

uint16_t adcVal;

void ADC0_init(void);
uint16_t ADC0_read(void);
void ADC0_start(void);
bool ADC0_conersionDone(void);

void ADC0_init(void)
{
    /* ﾃﾞｼﾞﾀﾙ入力緩衝部禁止 */
    PORTD.PIN6CTRL &= ~PORT_ISC_gm;
    PORTD.PIN6CTRL |= PORT_ISC_INPUT_DISABLE_gc;

    /* ﾌﾟﾙｱｯﾌﾟ抵抗禁止 */
    PORTD.PIN6CTRL &= ~PORT_PULLUPEN_bm;

    ADC0.CTRLC = ADC_PRESC_DIV4_gc              /* CLK_PERを4分周 */
               | ADC_REFSEL_INTREF_gc;          /* 内部参照基準 */

    ADC0.CTRLA = ADC_ENABLE_bm                  /* ADC許可:許可 */
               | ADC_RESSEL_10BIT_gc;           /* 10ﾋﾞｯﾄ動作 */

    /* ADCﾁｬﾈﾙ選択 */
    ADC0.MUXPOS = ADC_MUXPOS_AIN6_gc;

    /* 自由走行動作許可 */
    ADC0.CTRLA |= ADC_FREERUN_bm;
}

uint16_t ADC0_read(void)
{
    /* 1書き込みによって割り込み要求ﾌﾗｸﾞ解除(0) */
    ADC0.INTFLAGS = ADC_RESRDY_bm;

    return ADC0.RES;
}

void ADC0_start(void)
{
    /* 変換開始 */
    ADC0.COMMAND = ADC_STCONV_bm;
}

bool ADC0_conersionDone(void)
{
    return (ADC0.INTFLAGS & ADC_RESRDY_bm);
}

int main(void)
{
    ADC0_init();
    ADC0_start();

    while(1)
    {
        if (ADC0_conersionDone())
        {
```

例10-2 (続き). ADC自由走行ｺｰﾄﾞ例

```
            adcVal = ADC0_read();
            /* 自由走行動作では次の変換が自動的に始まります。 */
        }
    }
}
```

例10-3. ADC採取累積器ｺｰﾄﾞ例

```
#define ADC_SHIFT_DIV64  (6)

#include <avr/io.h>

uint16_t adcVal;

void ADC0_init(void);
uint16_t ADC0_read(void);

void ADC0_init(void)
{
    /* ﾃﾞｼﾞﾀﾙ入力緩衝部禁止 */
    PORTD.PIN6CTRL &= ~PORT_ISC_gm;
    PORTD.PIN6CTRL |= PORT_ISC_INPUT_DISABLE_gc;

    /* ﾌﾟﾙｱｯﾌﾟ抵抗禁止 */
    PORTD.PIN6CTRL &= ~PORT_PULLUPEN_bm;

    ADC0.CTRLC = ADC_PRESC_DIV4_gc          /* CLK_PERを4分周 */
               | ADC_REFSEL_INTREF_gc       /* 内部参照基準 */

    ADC0.CTRLA = ADC_ENABLE_bm              /* ADC許可:許可 */
               | ADC_RESSEL_10BIT_gc        /* 10ﾋﾞｯﾄ動作 */

    /* ADCﾁｬﾈﾙ選択 */
    ADC0.MUXPOS  = ADC_MUXPOS_AIN6_gc;

    /* 64採取を累積するように累積器動作を設定 */
    ADC0.CTRLB = ADC_SAMPNUM_ACC64_gc;
}

uint16_t ADC0_read(void)
{
    /* ADC変換開始 */
    ADC0.COMMAND = ADC_STCONV_bm;

    /* ADC変換終了まで待機 */
    while ( !(ADC0.INTFLAGS & ADC_RESRDY_bm) )
    {
        ;
    }

    /* 1書き込みによって割り込み要求ﾌﾗｸﾞ解除(0) */
    ADC0.INTFLAGS = ADC_RESRDY_bm;

    return ADC0.RES;
}

int main(void)
{
    ADC0_init();
```

例10-3 (続き). ADC採取累積器ｺｰﾄﾞ例

```
    while (1)
    {
        adcVal = ADC0_read();

        /* 64で除算 */
        adcVal = adcVal >> ADC_SHIFT_DIV64;
    }
}
```

例10-4. ADC窓比較器ｺｰﾄﾞ例

```
#define WINDOW_CMP_LOW_TH_EXAMPLE     (0x100)

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

uint16_t adcVal;

void ADC0_init(void);
uint16_t ADC0_read(void);
void ADC0_start(void);
bool ADC0_conersionDone(void);
bool ADC0_resultBelowTreshold(void);
void ADC0_clearWindowCmpIntFlag(void);
void LED0_init(void);void LED0_on(void);
void LED0_off(void);

void ADC0_init(void)
{
    /* ﾃﾞｼﾞﾀﾙ入力緩衝部禁止 */
    PORTD.PIN6CTRL &= ~PORT_ISC_gm;
    PORTD.PIN6CTRL |= PORT_ISC_INPUT_DISABLE_gc;

    /* ﾌﾟﾙｱｯﾌﾟ抵抗禁止 */
    PORTD.PIN6CTRL &= ~PORT_PULLUPEN_bm;

    ADC0.CTRLC = ADC_PRESC_DIV4_gc              /* CLK_PERを4分周 */
               | ADC_REFSEL_INTREF_gc           /* 内部参照基準 */

    ADC0.CTRLA = ADC_ENABLE_bm                  /* ADC許可:許可 */
               | ADC_RESSEL_10BIT_gc            /* 10ﾋﾞｯﾄ動作 */

    /* ADCﾁｬﾈﾙ選択 */
    ADC0.MUXPOS = ADC_MUXPOS_AIN6_gc;

    /* 変換窓比較器下側閾値設定 */
    ADC0.WINLT = WINDOW_CMP_LOW_TH_EXAMPLE;

    /* 変換窓動作設定 */
    ADC0.CTRLE = ADC_WINCM_BELOW_gc;

    /* 自由走行動作許可 */
    ADC0.CTRLA |= ADC_FREERUN_bm;
}

uint16_t ADC0_read(void)
{
    /* 1書き込みによって結果準備可割り込み要求ﾌﾗｸﾞ解除(0) */
    ADC0.INTFLAGS = ADC_RESRDY_bm;
```

例10-4 (続き). ADC窓比較器ｺｰﾄﾞ例

```
    return ADC0.RES;
}

void ADC0_start(void)
{
    /* 変換開始 */
    ADC0.COMMAND = ADC_STCONV_bm;
}

bool ADC0_conersionDone(void)
{
    return (ADC0.INTFLAGS & ADC_RESRDY_bm);
}

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

void ADC0_clearWindowCmpIntFlag(void)
{
    /* 1書き込みによって窓割り込み要求ﾌﾗｸﾞ解除(0) */
    ADC0.INTFLAGS = ADC_WCMP_bm;
}

void LED0_init(void)
{
    /* 出力値High(OFF)指定 */
    PORTB.OUT |= PIN5_bm;
    /* 出力に設定 */
    PORTB.DIR |= PIN5_bm;
}

void LED0_on(void)
{
    /* 出力値Low(ON)指定 */
    PORTB.OUT &= ~PIN5_bm;
}

void LED0_off(void)
{
    /* 出力値High(OFF)設定 */
    PORTB.OUT |= PIN5_bm;
}

int main(void)
{
    ADC0_init();
    LED0_init();

    ADC0_start();

    while(1)
    {
        if (ADC0_conersionDone())
        {
            if(ADC0_resultBelowTreshold())
            {
                LED0_on();
                ADC0_clearWindowCmpIntFlag();
```

例10-4 (続き). ADC窓比較器ｺｰﾄﾞ例

```
            }
            else
            {
                LED0_off();
            }

            adcVal = ADC0_read();
        }
    }
}
```

例10-5. 事象起動ADCｺｰﾄﾞ例

```
/* RTC周期 */
#define RTC_PERIOD   (511)

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

volatile uint16_t adcVal;

void ADC0_init(void);
void LED0_init(void);
void LED0_toggle(void);
void RTC_init(void);
void EVSYS_init(void);

void ADC0_init(void)
{
    /* ﾃﾞｼﾞﾀﾙ入力緩衝部禁止 */
    PORTD.PIN6CTRL &= ~PORT_ISC_gm;
    PORTD.PIN6CTRL |= PORT_ISC_INPUT_DISABLE_gc;

    /* ﾌﾟﾙｱｯﾌﾟ抵抗禁止 */
    PORTD.PIN6CTRL &= ~PORT_PULLUPEN_bm;

    ADC0.CTRLC = ADC_PRESC_DIV4_gc          /* CLK_PERを4分周 */
               | ADC_REFSEL_INTREF_gc       /* 内部参照基準 */

    ADC0.CTRLA = ADC_ENABLE_bm              /* ADC許可:許可 */
               | ADC_RESSEL_10BIT_gc        /* 10ﾋﾞｯﾄ動作 */

    /* ADCﾁｬﾈﾙ選択 */
    ADC0.MUXPOS = ADC_MUXPOS_AIN6_gc;

    /* ADC割り込み許可 */
    ADC0.INTCTRL |= ADC_RESRDY_bm;

    /* 事象起動変換許可 */
    ADC0.EVCTRL |= ADC_STARTEI_bm;
}

void LED0_init(void)
{
    /* 出力値High(OFF)指定 */
    PORTB.OUT |= PIN5_bm;
    /* 出力に設定 */
    PORTB.DIR |= PIN5_bm;
}
```

例10-5 (続き). 事象起動ADCｺｰﾄﾞ例

```
void LED0_toggle(void)
{
    PORTB.IN |= PIN5_bm;
}

ISR(ADC0_RESRDY_vect)
{
    /* 1書き込みによって結果準備可割り込み要求ﾌﾗｸﾞ解除(0) */
    ADC0.INTFLAGS = ADC_RESRDY_bm;
    adcVal = ADC0.RES;
    LED0_toggle();
}

void RTC_init(void)
{
    uint8_t temp;

    /* 32.768kHz発振器初期化 */
    /* 発振器禁止 */
    temp = CLKCTRL.XOSC32KCTRLA;
    temp &= ~CLKCTRL_ENABLE_bm;
    /* 保護されたﾚｼﾞｽﾀへの書き込みを許可 */
    CPU_CCP = CCP_IOREG_gc;
    CLKCTRL.XOSC32KCTRLA = temp;

    while(CLKCTRL.MCLKSTATUS & CLKCTRL_XOSC32KS_bm)
    {
        ; /* XOSC32KSが0になるまで待機 */
    }

    /* SEL=0 (外部ｸﾘｽﾀﾙを使用) */
    temp = CLKCTRL.XOSC32KCTRLA;
    temp &= ~CLKCTRL_SEL_bm;
    /* 保護されたﾚｼﾞｽﾀへの書き込みを許可 */
    CPU_CCP = CCP_IOREG_gc;
    CLKCTRL.XOSC32KCTRLA = temp;

    /* 発振器許可 */
    temp = CLKCTRL.XOSC32KCTRLA;
    temp |= CLKCTRL_ENABLE_bm;
    /* 保護されたﾚｼﾞｽﾀへの書き込みを許可 */
    CPU_CCP = CCP_IOREG_gc;
    CLKCTRL.XOSC32KCTRLA = temp;

    /* RTC初期化 */
    while (RTC.STATUS > 0)
    {
        ; /* 全てのﾚｼﾞｽﾀが同期化されるのを待機 */
    }

    RTC.CTRLA = RTC_PRESCALER_DIV32_gc          /* 32前置分周 */
              | RTC_RTCEN_bm                    /* 許可:許可 */
              | RTC_RUNSTDBY_bm;                /* ｽﾀﾝﾊﾞｲで走行:許 */

    /* 周期設定 */
    RTC.PER = RTC_PERIOD;

    /* 32.768kHz外部ｸﾘｽﾀﾙ用発振器(XOSC32K) */
    RTC.CLKSEL = RTC_CLKSEL_TOSC32K_gc;
```

例10-5 (続き). 事象起動ADCｺｰﾄﾞ例

```
    /* ﾃﾞﾊﾞｯｸﾞで走行:許可 */
    RTC.DBGCTRL |= RTC_DBGRUN_bm;
}

void EVSYS_init(void)
{
    /* 実時間計数器溢れ */
    EVSYS.CHANNEL0 = EVSYS_GENERATOR_RTC_OVF_gc;
    /* 使用部を事象ﾁｬﾈﾙ0に接続 */
    EVSYS.USERADC0 = EVSYS_CHANNEL_CHANNEL0_gc;
}

int main(void)
{
    ADC0_init();
    LED0_init();
    RTC_init();
    EVSYS_init();

    /* 全体割り込み許可 */
    sei();

    while (1)
    {
        ;
    }
}
```

Microchipｳｪﾌﾞ ｻｲﾄ

Microchipはwww.microchip.com/で当社のｳｪﾌﾞ ｻｲﾄ経由でのｵﾝﾗｲﾝ支援を提供します。このｳｪﾌﾞ 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** – ﾃﾞｰﾀｼｰﾄと障害情報、応用記述と試供ﾌﾟﾛｸﾞﾗﾑ、設計資源、使用者の手引きとﾊｰﾄﾞｳｪｱ支援資料、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、ｵﾝﾗｲﾝ検討ｸﾞﾙｰﾌﾟ、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、ｾﾐﾅｰとｲﾍﾞﾝﾄの一覧、Microchip営業所の一覧、代理店と代表する工場

製品変更通知ｻｰﾋﾞｽ

Microchipの製品変更通知ｻｰﾋﾞｽは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でのｳｪﾌﾞ ｻｲﾄを通して利用できます。

Microchipﾃﾞﾊﾞｲｽ ｺｰﾄﾞ保護機能

Microchipﾃﾞﾊﾞｲｽでの以下のｺｰﾄﾞ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ﾃﾞ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ﾃﾞﾊﾞｲｽのｺｰﾄﾞ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のﾃﾞ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。これらのｺｰﾄﾞ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はそれのｺｰﾄﾞ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ﾄﾞの安全を保証することはできません。ｺｰﾄﾞ保護は製品が”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。ｺｰﾄﾞ保護は常に進化しています。Microchipは当社製品のｺｰﾄﾞ保護機能を継続的に改善することを約束します。Microchipのｺｰﾄﾞ保護機能を破る試みはﾃﾞｼﾞﾀﾙ 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。ﾃﾞﾊﾞ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ﾃﾞﾊﾞ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

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。

本技術概説はMicrochipのTB3209技術概説(DS90003209B-2021年3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