TB3209

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です。

この技術概要はmegaAVR® 0系ﾏｲｸﾛ 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: ｺｰﾄﾞ例はATmega4809 Xplained Pro (ATMEGA4809-XPRO)で開発されました。

目次

1. 関連ﾃﾞﾊﾞｲｽ

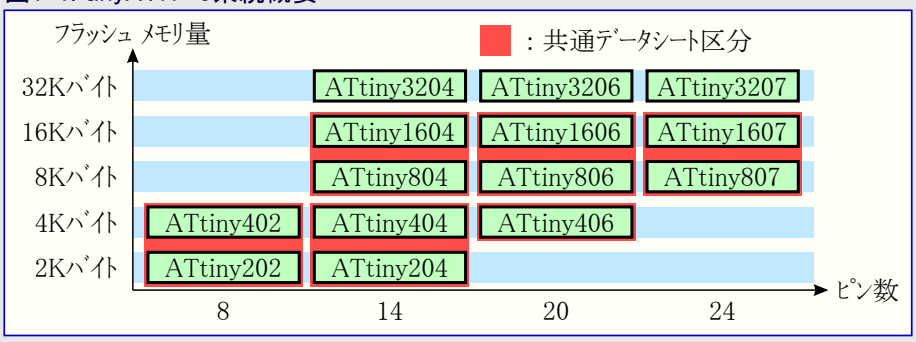
本章はこの資料に関連するﾃﾞﾊﾞｲｽを一覧にします。

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1-1. tinyAVR® 0系統概要



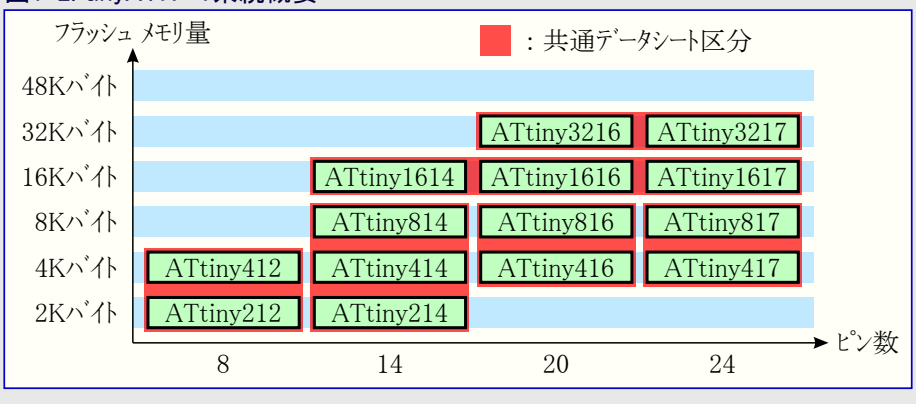
異なるﾌﾗｯｼｭ ﾒﾓﾘ量を持つﾃﾞﾊﾞｲｽは一般的に異なるSRAMとEEPROMの量を持ちます。

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1-2. tinyAVR® 1系統概要



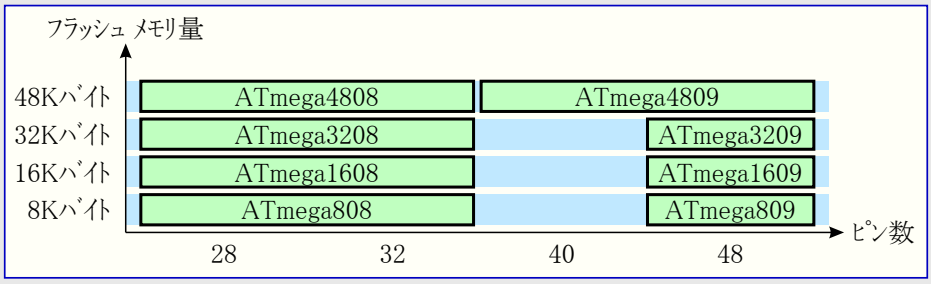
異なるﾌﾗｯｼｭ ﾒﾓﾘ量を持つﾃﾞﾊﾞｲｽは一般的に異なるSRAMとEEPROMの量を持ちます。

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1-3. 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。

2. 概要

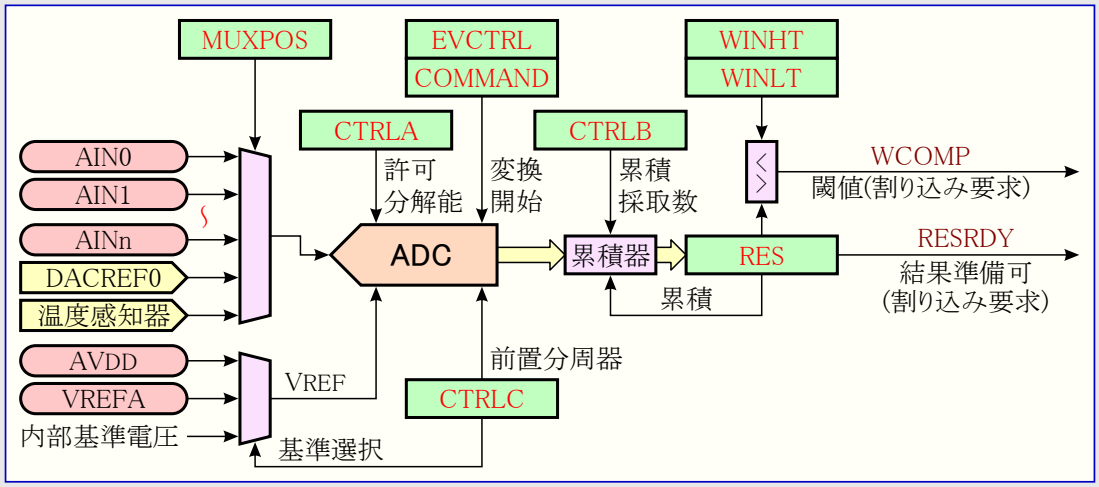
A/D変換器(ADC)周辺機能は10ﾋﾞｯﾄの結果を生じます。ADC入力は内部(例えば、基準電圧)またはｱﾅﾛｸﾞ入力ﾋﾟﾝを通す外部のどちらかにすることができます。ADCは多数のｼﾝｸﾞﾙ ｴﾝﾄﾞ電圧入力を許すｱﾅﾛｸﾞ多重器に接続されます。ｼﾝｸﾞﾙ ｴﾝﾄﾞ電圧入力は0V(GND)を参照基準にします。

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)ﾋﾟ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)ﾋﾞｯﾄに書くことによって選ばれます。ADC入力ﾋﾟﾝのどれか、GND、内部基準電圧(VREF)、または温度感知器をADCへのｼﾝｸﾞﾙ ｴﾝﾄﾞ入力として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。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ﾚｼﾞｽﾀの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ﾋﾞｯﾄの結果を生成します。この結果は右揃えで表されます。10ﾋﾞｯﾄ変換に対する結果は右式として与えられます。

$$RES = \frac{V_{IN} \times 1023}{V_{REF}}$$

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

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

ビット	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

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

これらのビットは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)

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

これらのビットは周辺クロック(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レジスタのRESSELビットによって設定されます。ADCはADCn.CTRLAレジスタのENABLEビットを設定(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

ビット	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

● **ビット2 – RESSEL : 分解能選択** (Resolution Selection)

このビットはADC分解能を選びます。

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。

図3-4. ADCn.COMMAND – 変換開始

ﾋﾞｯﾄ	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

● **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、このﾋﾞｯﾄは自動的に解除(0)されます。

変換進行中のこのﾋﾞｯﾄへの'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ﾋﾞｯﾄがﾊｰﾄﾞ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ﾋﾞｯﾄが設定(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。

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

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;
```

変換結果はADCn.RESﾚｼﾞｽﾀから読むことができます。

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

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**」章でも利用可能です。

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();
    }
}
```

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」章でも利用可能です。

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	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	–
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。

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

採取(試料)はADCn.RESレジ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フラグが設定(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;
```

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**」章でも利用可能です。

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デバイスが検出します。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レジ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ﾚｼﾞｽﾀで設定されます。

図6-2. ADCn.CTRLE – 窓比較器動作構成設定

ﾋﾞｯﾄ	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

● **ﾋﾞｯﾄ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ｸﾞが設定(1)される時を許可して定義します。RESULTは16ﾋﾞｯﾄ累積器の結果です。WINLTとWINHTは各々16ﾋﾞｯﾄの下側閾値と16ﾋﾞ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、ﾊｰﾄﾞｳｪｱによってADCn.INTFLAGSﾚｼﾞｽﾀのWCOMPﾋﾞｯﾄが設定(1)されます。

図6-3. ADC0.INTFLAGS – WCOMPﾋﾞｯﾄ ﾊｰﾄﾞｳｪｱ設定

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。

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

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

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**」章でも利用可能です。

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レジスタのSTARTEIビ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 */
```

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**」章でも利用可能です。

8. 参照

ADC動作形態についてより多くの情報は以下の応用記述で見つけることができます。

1. megaAVR® 0系手引書 (DS40002015)
2. AN2573 – tinyAVR® 0と1系及びmegaAVR® 0系でのADC基礎 (DS00002573)

9. 追補

例9-1. ADC単独変換コード例

```
/* 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)
{
    /* デジタル入力緩衝部禁止 */
    PORTD.PIN6CTRL &= ~PORT_ISC_gm;
    PORTD.PIN6CTRL |= PORT_ISC_INPUT_DISABLE_gc;

    /* プルアップ抵抗禁止 */
    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ビット動作 */

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

uint16_t ADC0_read(void)
{
```

例9-1 (続き). ADC単独変換ｺｰﾄﾞ例

```
    /* 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9-2. 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;
}
```

例9-2 (続き). ADC自由走行ｺｰﾄﾞ例

```
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())
        {
            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9-3. 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 */
```

例9-3 (続き). ADC採取累積器ｺｰﾄﾞ例

```
    /* 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();

    while (1)
    {
        adcVal = ADC0_read();

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

例9-4. 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 */
```

例9-4 (続き). ADC窓比較器ｺｰﾄﾞ例

```
    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;

    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;
```

例9-4 (続き). ADC窓比較器ｺｰﾄﾞ例

```
    /* 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();
            }
            else
            {
                LED0_off();
            }
            adcVal = ADC0_read();
        }
    }
}
```

例9-5. 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 */
```

例9-5 (続き). 事象起動ADCｺｰﾄﾞ例

```
    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;
}

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 */
    /* 結果発振器禁止 */
    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 */
    }
```

例9-5 (続き). 事象起動ADCコード例

```
    /* SEL=0 (外部クリスタルを使用) */
    temp = CLKCTRL.XOSC32KCTRLA;
    temp &= ~CLKCTRL_SEL_bm;
    /* 保護されたレジ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;
    /* 保護されたレジ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)
    {
        ; /* 全てのレジ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;                /* スタンバイで走行:許可 */

    /* 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;

    /* デバッグで走行: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ウェブ サイト

Microchipはhttp://www.microchip.com/で当社のウェブ サイト経由でのオンライン支援を提供します。このウェブ サイトはお客様がファイルや情報を容易に利用可能にする手段として使われます。お気に入りのインターネット ブラウザを用いてアクセスすることができ、ウェブ サイトは以下の情報を含みます。

- **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相談役プログラム員一覧
- **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

お客様への変更通知サービス

Microchipのお客様通知サービスは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http://www.microchip.com/でMicrochipのウェブ サイトをアクセスしてください。”Support”下で”Customer Change Notification”をクリックし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
- 現場応用技術者(FAE:Field Aplication 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、または現場応用技術者(FAE)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http://www.microchip.com/supportでのウェブ サイトを通して利用できます。

Microchipデバイス コード保護機能

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

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

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

法的通知

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

商標

DNVによって認証された品質管理ｼｽﾃﾑ

ISO/TS 16949

Microchipはその世界的な本社、ｱﾘｿﾞﾅ州のﾁｬﾝﾄﾞﾗｰとﾃﾝﾍﾟ、ｵﾚｺﾞﾝ州ｸﾞﾗｼｬﾑの設計とｳｪﾊｰ製造設備とｶﾘﾌｫﾙﾆｱとｲﾝﾄﾞの設計ｾﾝﾀｰに対してISO/TS-16949:2009認証を取得しました。当社の品質ｼｽﾃﾑの処理と手続きはPIC® MCUとdsPIC® DSC、KEELOQ符号飛び回りﾃﾞﾊﾞｲｽ、直列EEPROM、ﾏｲｸﾛ周辺機能、不揮発性ﾒﾓﾘ、ｱﾅﾛｸﾞ製品用です。加えて、開発ｼｽﾃﾑの設計と製造のためのMicrochipの品質ｼｽﾃﾑはISO 9001:2000認証取得です。

日本語© *HERO* 2020.

本技術概説はMicrochipのTB3209技術概説(DS90003209A-2019年1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: http://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 - 廈門** 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 - ヴｪﾙｽ** 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: 45-4450-2828 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: 49-7131-67-3636 **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