TB3254

集中累積動作でのPGA付き12ビット差動ADCの使い方

要点

- 12ﾋﾞｯﾄ分解能
- 1024採取までの採取累積
- 作動とｼﾝｸﾞﾙ ｴﾝﾄﾞの変換
  - 15個までのｱﾅﾛｸﾞ入力
    - 15個の正入力と7つの負入力
- 4つの内部入力
  - GND
  - VDD/10
  - 温度感知器
  - ｱﾅﾛｸﾞ比較器からのDACREF
- 組み込み内部参照基準と外部参照基準任意選択
- 1～16倍で設定可能な利得増幅器
- 自由走行動作
- 左または右に揃えられた結果
- 任意選択: 事象起動変換
- 構成設定可能な窓比較器

序説

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

この技術概説はtinyAVR® 2系統で特徴とされる12ﾋﾞｯﾄA/D変換器(ADC)での集中動作の使い方を説明します。

下のｺｰﾄﾞ例が集中動作を使って与えられます。

- 窓比較器を使う割り込み
- 事象起動集中変換
- 分解能を増すためのADC過採取
- ADC分解能を増すための尺度調整付き集中累積

集中動作では単一起動後にADCが可能な限り速く構成設定された採取数を採取します。変換結果は単一ADC結果として累積されます。単一ADC変換起動で最大1024採取を累積することができます。

ADC動作形態は次のように3つの群に分けることができます。

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

他のADC動作の詳細については「1. 関連文書」章を参照してください。

目次

1. 関連文書

以下の文書がこの技術概説に関連します。

- ﾃﾞｰﾀｼｰﾄ：製品頁でのtinyAVR 2ﾃﾞｰﾀｼｰﾄ (.pdf)
  - www.microchip.com/wwwproducts/en/ATtiny1624
  - www.microchip.com/wwwproducts/en/ATtiny1626
  - www.microchip.com/wwwproducts/en/ATtiny1627
- 単独動作でのPGA付き12ﾋﾞｯﾄ差動ADCの使い方：www.microchip.com/DS90003256
- 継続累積動作でのPGA付き12ﾋﾞｯﾄ差動ADCの使い方：www.microchip.com/DS90003257

2. 構成設定

2.1. 集中動作構成設定

利用可能な2つの集中動作、集中累積と尺度調整付き集中累積があります。両動作に於いて、ADCは起動後に設定された採取数変換し、各変換結果で採取(ADCn.SAMPLE)ﾚｼﾞｽﾀを更新します。採取は自動的に累積されて集中での最後の変換が完了された時に結果(ADCn.RESULT)ﾚｼﾞｽﾀに置かれます。集中累積動作では累積された結果がそのまま提示される一方で、尺度調整付き集中累積位では「5.2.2. 尺度調整付き集中累積」項で示されるように結果が出力処理を容易にするため尺度調整されます。

2つの動作は指令(ADCn.COMMAND)ﾚｼﾞｽﾀの動作形態(MODE)ﾋﾞｯﾄを書くことによって選ぶことができます。下は集中動作の構成設定を示すｺｰﾄﾞ例です。

```
/* 集中累積動作 */
ADC0.COMMAND = ADC_MODE_BURST_gc;
/* 尺度調整付き集中累積動作 */
ADC0.COMMAND = ADC_MODE_BURST_SCALING_gc;
```

2.2. 参照基準

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

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

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

VDDを除き、内部参照基準電圧は内部ﾊﾞﾝﾄﾞｷﾞｬｯﾌﾟ参照基準から生成されます。VDDは選んだﾊﾞﾝﾄﾞｷﾞｬｯﾌﾟ参照基準電圧よりも最低0.5V高くなければなりません。

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

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

2.3. ｼﾝｸﾞﾙ ｴﾝﾄﾞ動作と差動動作

ｼﾝｸﾞﾙ ｴﾝﾄﾞ動作ではADCが選択可能な単一入力元の電圧を読む一方で、差動動作ではADCが2つの入力元間の電圧差を読みます。

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

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

ｼﾝｸﾞﾙ ｴﾝﾄﾞ動作は下で示されるように差動(DIFF)ﾋﾞｯﾄに'0'を書くことによって構成設定されます。

```
/* ｼﾝｸﾞﾙ ｴﾝﾄﾞ動作構成設定 */
ADC0.COMMAND &= ~ADC_DIFF_bm;
```

2.4. 設定可能な利得増幅器

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

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

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

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

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

2.5. 割り込み

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

表2-1. 利用可能な割り込みﾍﾞｸﾀと供給元

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

割り込み元は下のｺｰﾄﾞ断片で示されるように割り込み制御(ADCn.INTCTRL)ﾚｼﾞｽﾀで対応するﾋﾞｯﾄを書くことによって許可または禁止されます。

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

割り込み要求ﾌﾗｸﾞは下のｺｰﾄﾞ断片で示されるように割り込み要求ﾌﾗｸﾞ(ADCn.INTFLAGS)ﾚｼﾞｽﾀでそのﾋﾞｯﾄ位置に'1'を書くことによって解除(0)されます。

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

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

2.6. 窓比較器

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

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

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

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

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

以下のｺｰﾄﾞは窓比較器の閾値構成設定方法と結果ﾚｼﾞｽﾀに対してINSIDE動作での比較の構成設定方法を示します。

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

2.6.1. ｺｰﾄﾞ例

下のｺｰﾄﾞ例は2000未満または3000超えのADC読み取りが、無効な信号はみ出しと見做される場合の応用例を示します。窓比較器は信号が閾値の外側の時にSAMPRDY割り込みで採取累積を再始動することによってそれらの出力の濾波に使われます。信号の電圧は全ての採取が累積された時にRESRDY割り込み処理ﾙｰﾁﾝ(ISR)で計算されます。

```
#define F_CPU 3333333ul

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

#define TIMEBASE_VALUE        ((uint8_t) ceil(F_CPU*0.000001))
#define ADC_MAX_VALUE         ((1 << 12) - 1)   /* ｼﾝｸﾞﾙ ｴﾝﾄﾞ動作で最大値は4095 */

/* 容易なADC累積構成設定のために定義 */
#define ADC_SAMPNUM_CONFIG    ADC_SAMPNUM_ACC256_gc
/* 累積採取数で結果を左移動(1 << SAMPNUM) */
#define ADC_SAMPLES           (1 << ADC_SAMPNUM_CONFIG)

/* ﾃﾞﾊﾞｯｸﾞ体験を改善するための揮発性変数 */
static volatile uint32_t adc_reading;
static volatile float voltage;

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

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

    ADC0.WINHT = 3000;                          /* 窓上側閾値 */
    ADC0.WINLT = 2000;                          /* 窓下側閾値 */
    /* 窓比較器動作:外側。窓比較器元としてSAMPLEﾚｼﾞｽﾀを使用 */
    ADC0.CTRLD = ADC_WINCM_OUTSIDE_gc | ADC_WINSRC_SAMPLE_gc;
    /* 窓比較と結果準備可割り込みを許可 */
    ADC0.INTCTRL = ADC_WCMP_bm | ADC_RESRDY_bm;

    ADC0.COMMAND = ADC_MODE_BURST_gc;           /* 集中累積動作 */
}

/*****************************************************************************
窓比較割り込み: この例では採取が或る窓の外側の時にこれが無効な信号はみ出しと見做されます。
このようなはみ出しが検出されると、集中変換を再始動することによって累積されたADC結果が無視されます。
*****************************************************************************/
ISR(ADC0_SAMPRDY_vect)
{
    ADC0.INTFLAGS = ADC_WCMP_bm;                /* WCMPﾌﾗｸﾞ解除 */

    /* 進行中の集中を停止 */
    ADC0.COMMAND = ADC_START_STOP_gc;
    /* 新しい集中累積開始 */
    ADC0.COMMAND = ADC_MODE_BURST_gc | ADC_START_IMMEDIATE_gc;
}
```

```
/*****************************************************************************
結果準備可割り込み: はみ出しが検出されなかった場合、結果が読まれて対応する電圧が計算されます。
*****************************************************************************/
ISR(ADC0_RESRDY_vect)
{
    ADC0.INTFLAGS = ADC_RESRDY_bm;              /* RESRDYﾌﾗｸﾞ解除 */

    /* 最後の採取が窓の内側だったか調査 */
    if(!(ADC0.INTFLAGS & ADC_WCMP_bm))
    {
        adc_reading = ADC0.RESULT;              /* ADC結果読み込み */
        /* ADCﾋﾟﾝの電圧計算、VDD=3.3V、12ﾋﾞｯﾄ分解能、256採取 */
        voltage = (float)(((adc_reading * 3.3) / ADC_MAX_VALUE) / ADC_SAMPLES);
    }
}

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

    while(1)
    {
        /* 100ms毎に1度変換を開始 */
        ADC0.COMMAND |= ADC_START_IMMEDIATE_gc;
        _delay_ms(100);
    }
}
```

2.7. 事象

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

下表は利用可能なADCからの各種事象生成部を示します。

表2-2. ADC事象生成部

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

下は事象ﾁｬﾈﾙ1を通してEVOUT事象使用部に接続したADC0_RES事象生成部の構成設定を示し、この場合、事象をPB2に出力します。

- 事象生成部：ADC0 RES
- 事象使用部：EVOUT (PB2)

```
EVSYS.CHANNEL1 = EVSYS_CHANNEL1_ADC0_RES_gc;        /* ADC結果準備可 */
EVSYS.USEREVSYSEVOUTB = EVSYS_USER_CHANNEL1_gc;     /* 非同期事象ﾁｬﾈﾙ1 */
```

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

表2-3. ADC事象使用部と利用可能な事象活動

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

START事象活動は下のｺｰﾄﾞ断片で示されるように指令(ADCn.COMMAND)ﾚｼﾞｽﾀの開始(START)ﾋﾞｯﾄ領域にEVENT_TRIGGER設定が書かれる場合に起動することができます。

```
ADC0.COMMAND = ADC_START_EVENT_TRIGGER_gc;
```

下はRTC溢れに反応する事象使用部としてADC0_STARTの構成設定を示しているｺｰﾄﾞ断片です。

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

2.7.1. ｺｰﾄﾞ例

下は事象生成部と事象使用部としてADCの構成設定を示しているｺｰﾄﾞ例です。

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

```
#define F_CPU 3333333ul

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

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

/* 容易なADC累積構成設定のために定義 */
#define ADC_SAMPNUM_CONFIG      ADC_SAMPNUM_ACC8_gc
/* 累積採取数で結果を左移動(1 << SAMPNUM) */
#define ADC_SAMPLES             (1 << ADC_SAMPNUM_CONFIG)

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

/* ﾃﾞﾊﾞｯｸﾞ体験を改善するための揮発性変数 */
static volatile int32_t adc_reading;
static volatile float voltage;

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

    EVSYS.CHANNEL0 = EVSYS_CHANNEL0_RTC_OVF_gc;              /* RTC溢れ ⇒ ﾁｬﾈﾙ0 */
    EVSYS.USERADC0START = EVSYS_USER_CHANNEL0_gc;            /* ﾁｬﾈﾙ0 ⇒ ADC0開始 */

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

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

```
    while(RTC.STATUS > 0);          /* 全レジスタ同期待ち */
}

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

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

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

    while(1)
    {
        if(ADC0.INTFLAGS & ADC_RESRDY_bm)                   /* ADC採取準備可か調査 */
        {
            /* 累積したADC結果読み込み、割り込み要求フラグ解除 */
            adc_reading = ADC0.RESULT;
            /* ADCピンの電圧計算、VDD=3.3V、12ビット分解能で8採取 */
            voltage = (float)(((adc_reading * 3.3) / ADC_MAX_VALUE) / ADC_SAMPLES);
        }
    }
}
```

助言: 参照基準としてVDDを使い累積した採取の平均を取る時にADCを効率的に電源雑音濾波器として使うことができます。

3. 入力回路

3.1. 入力インピーダンス

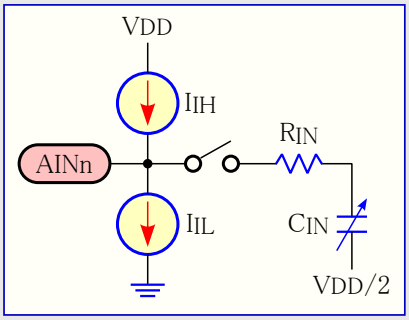
ピンに与えられた電圧水準が試料化される時にそれは最初に採取/保持コンデンサ(C_{IN})によって捕獲されます。これはADCが信号を試料化する間その電圧が変わらないことを保証します。C_{IN}を或る電圧水準に充放電するのにかかる時間は入力抵抗(R_{IN})によって制限されます。次式は時定数τと入力インピーダンス間の比例関係を示します。

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

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

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

図3-1. 内部アナログ入力回路の模式



$$VMSb = VREF - \frac{VREF}{4096 \times 利得}$$

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

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

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

入力回路に対するｲﾝﾊﾟﾙｽ応答は次式によって与えられます。

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

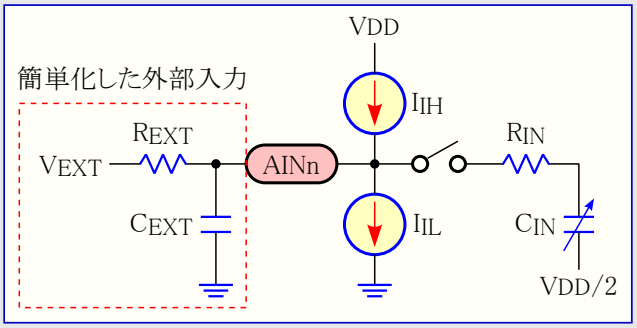
V_{IN}が100%であるVMSbに対する2つの例を解決すると、以下の安定時間が得られます。

$t利得なし = 8.29\tau$
$t16倍利得 = 10.81\tau$

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

外部ｲﾝﾋﾟｰﾀﾞﾝｽの特性は整定時間計算をどれ位複雑にするかを決めます。けれども、これはこの技術概説によって網羅されません。

図3-2. 外部信号付き内部ｱﾅﾛｸﾞ入力回路の模式



3.1.1. PGA

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

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

3.2. 採取持続時間

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

このADCに対する採取持続時間は最大256 ADCｸﾛｯｸ(CLK_ADC)周期にすることができ、制御E(ADCn.CTRLE)ﾚｼﾞｽﾀの採取持続時間(SAMPDUR)ﾋﾞｯﾄ領域を使って構成設定されます。採取持続時間はPGAが禁止されている時にSAMPDUR+0.5(CLK_ADC)周期、PGAが許可されている時にSAMPDUR+1(CLK_ADC)周期です。入力ｲﾝﾋﾟｰﾀﾞﾝｽが非常に高い場合、更なる採取持続時間増加にADC前置分周器増加使うこともできます。

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

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

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

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

4. 電力とﾀｲﾐﾝｸﾞ

4.1. ｸﾛｯｸ

ADCｸﾛｯｸ(CLK_ADC)は周辺機能ｸﾛｯｸ(CLK_PER)から下げられます。これは制御B(ADCn.CTRLB)ﾚｼﾞｽﾀの前置分周器(PRESC)ﾋﾞｯﾄ領域によって構成設定することができます。

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

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

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

ADC0.CTRLC = (TIMEBASE_VALUE << ADC_TIMEBASE_gp);
```

4.2. PGAﾊﾞｲｱｽと出力採取持続時間

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

推奨されるf_{CLK_ADC}の組み合わせとPGABIASSELとADCPGASAMPDURについてはﾃﾞｰﾀｼｰﾄでこれらのﾋﾞｯﾄ領域に対するﾚｼﾞｽﾀ説明をご覧ください。

構成設定例が下で示されます。

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

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

4.3. 変換時間

単一結果に対する総変換時間は以下によって計算されます。

$$\text{総変換時間} = \text{初期化} + \frac{(\text{SAMPDUR} + 13.5) \times \text{SAMPNUM} + 2}{f_{CLK\_ADC}}$$

例えば、初期化=60µs、SAMPDUR=2、SAMPNUM=32、f_{CLK_ADC}= 1MHzが与えられると、総変換時間は次によって与えられます。

$$\text{総変換時間} = 60\mu s + \frac{(2 + 13.5) \times 32 + 2}{1\ (\text{MHz})} = 558\mu s$$

制御A(ADCn.CTRLA)ﾚｼﾞｽﾀで'1'を書かれた低遅延(LOWLAT)ﾋﾞｯﾄでは初期化時間がADC許可で一度だけ必要とされます。その後、上の例は498µsの総変換時間を与えます。

ADCの採取期間は(SAMPDUR+0.5)CLK_ADC周期として制御E(ADCn.CTRLE)ﾚｼﾞｽﾀの採取持続時間(SAMPDUR)ﾋﾞｯﾄ領域を通して構成設定されます。

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

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

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

4.4. 自由走行動作

自由走行動作では直前の累積が完了したら直ぐに新しい変換が開始されます。

下のｺｰﾄﾞ断片で示されるように制御F(ADCn.CTRLF)ﾚｼﾞｽﾀの自由走行(FREERUN)ﾋﾞｯﾄに'1'を書くことによって構成設定されます。

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

新しい変換は結果が結果(ADCn.RESULT)ﾚｼﾞｽﾀで利用可能になった直後に開始することができます。これは割り込み要求ﾌﾗｸﾞ(ADCn.INTFLAGS)ﾚｼﾞｽﾀの結果準備可(RESRDY)によって合図されます。

5. 出力処理

5.1. 結果範囲

ADC変換からの出力は次式によって与えられます。

$$\text{ｼﾝｸﾞﾙ ｴﾝﾄﾞ変換} = \frac{\text{VINP}\times\text{利得}}{\text{VREF}}\times 4096 \in [0,4095]$$

$$\text{差動変換} = \frac{(\text{VINP}-\text{VINN})\times\text{利得}}{\text{VREF}}\times 2048 \in [-2048,2047]$$

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

ADCは2つの出力ﾚｼﾞｽﾀ、採取(ADCn.SAMPLE)と結果(ADCn.RESULT)のﾚｼﾞｽﾀを持ちます。16ﾋﾞｯﾄの採取ﾚｼﾞｽﾀは常に最後のADC変換出力(1採取)で更新されます。集中累積動作ではSAMPNUMの採取数が終了された後、結果ﾚｼﾞｽﾀが累積された結果で更新されます。

ｼﾝｸﾞﾙ ｴﾝﾄﾞ変換ではｱﾅﾛｸﾞ ﾋﾟﾝに印加された平均電圧が次によって計算されます。

$$\text{VINP} = \frac{\text{ADCn.RESULT}\times\text{VREF}}{4096\times\text{利得}\times\text{SAMPNUM}}$$

5.2. 累積

ADCは構成設定可能な変換結果数が自動的にADC結果に累積される集中での採取を支援します。最大1024採取を累積することができます。採取累積は制御F(ADCn.CTRLF)ﾚｼﾞｽﾀの採取累積数選択(SAMPNUM)ﾋﾞｯﾄ領域を書くことによって構成設定されます。

採取は支援される全ての累積構成設定に対して溢れを避けるため、充分な幅の内部採取累積器で累積されます。累積された結果は選んだ採取数が終了された時に32ﾋﾞｯﾄ結果(ADCn.RESULT)ﾚｼﾞｽﾀに転送され、割り込み要求ﾌﾗｸﾞ(ADCn.INTFLAGS)ﾚｼﾞｽﾀの結果準備可(RESRDY)ﾋﾞｯﾄが設定(1)されます。

下のｺｰﾄﾞ断片は1024採取を累積するための構成設定を示します。

```
ADC0.CTRLF = ADC_SAMPNUM_ACC1024_gc;
```

5.2.1. 過採取

集中累積動作はより高いADC分解能を達成するのに使うことができます。例えば、1024個の12ﾋﾞｯﾄの採取を使うことにより、17ﾋﾞｯﾄの結果を達成することができます。

分解能を増やすため、ADC分解能(n)の各追加ﾋﾞｯﾄに対して、信号は4^n採取個、過採取されなければなりません。17ﾋﾞｯﾄADCを達成するには追加5ﾋﾞｯﾄ分解能が必要とされます。従って、信号は4^5=1024回採取されなければなりません。その結果を望む分解能に縮小するにはその結果を2^nで除算することに等しいnによって右移動されなければなりません。

5.2.1.1. ｺｰﾄﾞ例

以下のｺｰﾄﾞ例は過採取を使うことによって12ﾋﾞｯﾄから17ﾋﾞｯﾄにADC分解能を増す方法とADCﾁｬﾈﾙAIN6に印加された電圧を計算する方法を示します。

```
#define F_CPU 3333333ul

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

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

/* ADC累積構成設定用定義 */
#define OVERSAMPLING_BITS       5               /* 5ﾋﾞｯﾄ追加 */
#define OVERSAMPLING_MAX_VALUE  ((uint32_t) ((1 << 12) - 1) << OVERSAMPLING_BITS) /* 12+5=17ﾋﾞｯﾄ */
#define ADC_SAMPNUM_CONFIG      (OVERSAMPLING_BITS << 1) /* SAMPNUMﾋﾞｯﾄ領域設定はこの式に合致 */
#define ADC_SAMPLES             (1 << ADC_SAMPNUM_CONFIG) /* 5ﾋﾞｯﾄ=1024採取 */

/* ﾃﾞﾊﾞｯｸﾞ体験を改善するための揮発性変数 */
static volatile uint32_t adc_reading;
static volatile float voltage;

/*********************************************************************************
```

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

    ADC0.MUXPOS = ADC_MUXPOS_AIN6_gc;           /* ADCチャネルAIN6⇒PA6 */
    ADC0.COMMAND = ADC_MODE_BURST_gc;           /* 集中累積動作 */
}

int main(void)
{
    adc_init();

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

        /* 技術概要で説明されたような過採取補償 */
        adc_reading = ADC0.RESULT >> OVERSAMPLING_BITS; /* 追加ﾋﾞｯﾄ数を右移動することによって累積結果尺
                                                           度調整 */
        voltage = (float)((adc_reading * 3.3) / OVERSAMPLING_MAX_VALUE); /* VDD=3.3Vで17ﾋﾞｯﾄ分解能
                                                                            を使って電圧計算 */
    }
}
```

5.2.2. 尺度調整付き集中累積

集中累積動作でのように、この動作でのSAMPNUM数の採取は単一結果に累積されます。尺度調整付き集中累積での違いは結果が自動的に16ﾋﾞｯﾄ値に縮小されます。尺度調整は常に最後の採取累積後に適用され、SAMPNUM-4ﾋﾞｯﾄで累積結果を右移動することによって実行されます。

この動作は単純に累積して自動的に平均結果を得るのに使うことができます。最低4^4=256採取で累積して自動的に16ﾋﾞｯﾄに縮小することによって16ﾋﾞｯﾄ分解能を得るため、「5.2.1. 過採取」項で記述されるような過採取技法との組み合わせでも使われるかもしれません。

5.2.2.1. ｺｰﾄﾞ例

以下のｺｰﾄﾞ例は尺度調整付き集中累積動作、16倍利得でのPGA、16ﾋﾞｯﾄ分解能を得るために過採取を使って差動測定を実行する方法を示します。

```
#define F_CPU 3333333ul

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

#define TIMEBASE_VALUE           ((uint8_t) ceil(F_CPU*0.000001))
#define ADC_DIFF_MAX_VALUE       (((1 << 12) / 2) - 1)   /* 差動動作での最大値は2047 */
#define ADC_DIFF_MAX_VALUE_16BIT ((uint32_t) ADC_DIFF_MAX_VALUE << 4)    /* 差動動作での16ﾋﾞｯﾄ結果の
                                                                            最大値は32752 */

/* 容易なADC累積構成設定のための定義 */
#define ADC_SAMPNUM_CONFIG  ADC_SAMPNUM_ACC256_gc
/* 累積採取数で結果左移動(1 << SAMPNUM) */
#define ADC_SAMPLES          (1 << ADC_SAMPNUM_CONFIG)

/* ﾃﾞﾊﾞｯｸﾞ体験を改善するための揮発性変数 */
static volatile int32_t adc_reading;
```

```
static volatile float voltage;
static volatile float current;

/*********************************************************************************
ADC初期化
*********************************************************************************/
void adc_init()
{
    ADC0.CTRLA = ADC_ENABLE_bm;
    ADC0.CTRLB = ADC_PRESC_DIV2_gc;              /* fCLK_ADC=3.333333/2MHz */
    ADC0.CTRLC = ADC_REFSEL_1024MV_gc | (TIMEBASE_VALUE << ADC_TIMEBASE_gp);
    ADC0.CTRLE = 17;      /* (SAMPDUR+0.5)×fCLK_ADC=10.5μs採取持続時間 */
    ADC0.CTRLF = ADC_LEFTADJ_bm | ADC_SAMPNUM_CONFIG;    /* 累積<16採取なら、左調整許可 */

    ADC0.MUXPOS = ADC_VIA_PGA_gc | ADC_MUXPOS_AIN6_gc;   /* ADCチャネルAIN6⇒PA6 */
    ADC0.MUXNEG = ADC_VIA_PGA_gc | ADC_MUXNEG_AIN7_gc;   /* ADCチャネルAIN7⇒PA7 */
    ADC0.COMMAND = ADC_DIFF_bm | ADC_MODE_BURST_SCALING_gc;       /* 尺度調整付き集中累積 */

    /* 16倍利得でPGA許可、高速採取用全バイアス電流設定、データシートに従ってADCPGASAMPDUR構成設定 */
    ADC0.PGACTRL = ADC_GAIN_16X_gc | ADC_PGABIASSEL_1X_gc | ADC_ADCPGASAMPDUR_6CLK_gc |
                   ADC_PGAEN_bm;
}

int main(void)
{
    adc_init();

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

        adc_reading = ADC0.RESULT;              /* 16ビット縮小または左調整した結果を読み込み */

    /* VREF=1.024V、16ビット分解能、16倍利得で作動電圧計算 */
    voltage = (float)((adc_reading * 1.024) / ADC_DIFF_MAX_VALUE_16BIT) / 16;
    //current = voltage / 5;      /* 電源と直列の5Ωを渡って測定する場合にこの行の注釈を外してください。 */

    _delay_ms(500);
    }
}
```

上の使用事例は電流測定です。下の画像で示されるように、デバイスの消費電流を測定するようにできるには些細なハードウェア変更が行われなければなりません。電流測定用の抵抗の大きさ選択は乗率に依存します。例えば、抵抗が高い場合、応用に給電する電池に短い電池寿命を引き起こすかもしれない大きな電圧低下を誘引するかもしれません。一方で、抵抗が低い場合、より低い分解能を引き起こすかもしれません。この例での5Ω抵抗器は400nAの測定段階量と400nA～12.8mAの範囲になります。

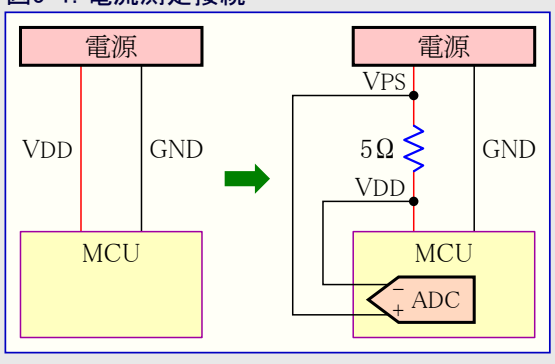
1. VPSとVDD間に抵抗器を接続してください。
2. PA6をVPSへ、PA7をVDDに接続してください。

消費電流測定を読むため、ソフトウェアで以下が行われなければなりません。

1. コードのmain()関数で電圧計算の下の行の注釈を外してください。
2. コードに中断点を追加してデバッグを開始するためにStart Debugging(デバッグ開始)をクリックしてください。
3. プログラムが中断点で停止された時にcurrent変数上を右クリックしてAdd Watch(監視に追加)を選んでください。今やcurrent変数は監視(Watch)ウィンドウで見ることができます。
4. 値(注)を更新するためにContinue(継続)をクリックしてください。

注: マルチメータで測定を管理している場合、結果がずれるかもしれんません。ADCが不活動の時にもマルチメータが測定するため、通常、より低い消費電流を示すでしょう。

図5-1. 電流測定接続



5.3. 左揃え

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

```
ADC0.CTRLF = ADC_LEFTADJ_bm;                    /* 左揃えﾋﾞｯﾄ許可 */
```

左揃えは尺度調整付き集中累積、ADC動作5で利用可能です。

以下の表は左揃え機能がｼﾝｸﾞﾙ ｴﾝﾄﾞ動作と差動動作で結果ﾚｼﾞｽﾀの出力形式にどう影響を及ぼすかを示します。

表5-1. RESULTﾚｼﾞｽﾀ – ｼﾝｸﾞﾙ ｴﾝﾄﾞ動作 – ADC動作5

LEFTADJ	RES31～24	RES23～16	RES15～12	RES11～8	RES7～0
0	$00			縮小された累積15～0	
1	$00			縮小された累積15～0 (注)	

表5-2. RESULTﾚｼﾞｽﾀ – 差動動作 – ADC動作5

LEFTADJ	RES31～24	RES23～16	RES15～12	RES11～8	RES7～0
0	符号拡張			符号付き縮小された累積15～0	
1	符号拡張			符号付き縮小された累積15～0 (注)	

SAMPNUM<4の場合、結果はﾋﾞｯﾄ15が最上位ﾋﾞｯﾄになるように4(SAMPNUMﾋﾞｯﾄ分)左移動されます。SAMPNUM>4の場合、ADC動作の縮小機能がSAMPNUM-4ﾋﾞｯﾄによって結果を右移動します。

例1: SAMPNUM=3 (8採取での累積)

左揃えﾋﾞｯﾄが'0'、全ての採取に対してADCn.SAMPLE=$FFF、8採取が累積されると、ADCn.RESULTに累積された値は8×4096=$7FF8です。

左揃えﾋﾞｯﾄ許可後、結果は4-SAMPNUM=1ﾋﾞｯﾄ(SAMPNUMは8採取を累積するため3)によって左移動されます。故に1ﾋﾞｯﾄでの$7FF8左移動によってADCn.RESULT=$FFF0

例2: SAMPNUM=5 (32採取での累積)

左揃えﾋﾞｯﾄが'0'、全ての採取に対してADCn.SAMPLE=$FFF、32採取が累積されると、ADCn.RESULTに累積された値は32×4096=$1FFE0です。

縮小動作で、結果はSAMPNUM-4=1ﾋﾞｯﾄ(SAMPNUMは32採取を累積するため5)によって右移動されます。故に1ﾋﾞｯﾄでの$1FFE0右移動によってADCn.RESULT=$FFF0

以下の表はｼﾝｸﾞﾙ ｴﾝﾄﾞ動作と差動動作で左揃え機能が採取ﾚｼﾞｽﾀ出力形式にどう影響を及ぼすかを示します。

表5-3. SAMPLEﾚｼﾞｽﾀ – ｼﾝｸﾞﾙ ｴﾝﾄﾞ動作/差動動作 – ADC動作5

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

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

5.4. 符号付きと符号なしの出力

ｼﾝｸﾞﾙ ｴﾝﾄﾞ動作での採取に対するﾃﾞｰﾀ形式は$0000が0を表し、$0FFFが最大数値を表す符号なし1の補数です。ｱﾅﾛｸﾞ入力がADCの参照基準よりも高い場合、12ﾋﾞｯﾄADC出力は$0FFFの最大値に等しくなります。同様に、入力が0V未満の場合、ADC出力は$0000です。

差動動作についてﾃﾞｰﾀ形式は符号拡張付き2の補数です。

採取ﾚｼﾞｽﾀ出力

採取変数のﾃﾞｰﾀ型はｼﾝｸﾞﾙ ｴﾝﾄﾞ動作使用時にuint16_tです。

採取変数のﾃﾞｰﾀ型は差動動作使用時にint16_tです。

例えば、12ﾋﾞｯﾄ動作でｼﾝｸﾞﾙ ｴﾝﾄﾞ動作使用時、単一採取の電圧は下のｺｰﾄﾞ断片で示されるように解釈され得ます。

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

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

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

結果ﾚｼﾞｽﾀ出力

結果変数のﾃﾞｰﾀ型はｼﾝｸﾞﾙ ｴﾝﾄﾞ動作使用時にuint32_tです。

結果変数のﾃﾞｰﾀ型は差動動作使用時にint32_tです。

例えば、12ﾋﾞｯﾄ動作でｼﾝｸﾞﾙ ｴﾝﾄﾞ動作使用時、SAMPNUM個累積された採取の電圧は下のｺｰﾄﾞ断片で示されるように解釈され得ます。

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

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

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

6. GitHubからのｺｰﾄﾞ例取得

ｺｰﾄﾞ例は図画的使用者ｲﾝﾀｰﾌｪｰｽ(GUI)を通して応用ｺｰﾄﾞを提供するｳｪﾌﾞに基づくｻｰﾊﾞｰであるGitHubを通して入手可能です。ｺｰﾄﾞ例はAtmel StudioとMPLAB Xの両方で開くことができます。MPLAB XでAtmel Studioﾌﾟﾛｼﾞｪｸﾄを開くにはﾒﾆｭｰでFile(ﾌｧｲﾙ)⇒Import(ｲﾝﾎﾟｰﾄ)⇒Atmel Studio project(Atmel Sudioﾌﾟﾛｼﾞｪｸﾄ)で.cprojﾌｧｲﾙへ誘導してください。

GitHubｳｪﾌﾞ頁：GitHub

ｺｰﾄﾞ例

tinyAVR 2系列のﾃﾞﾊﾞｲｽ用ｺｰﾄﾞ例を探すのは、GitHub例閲覧部でﾃﾞﾊﾞｲｽ名、例えば、ATtiny1627に対して検索することによって行うことができます。

GitHubでｺｰ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。

Clone(複製)またはDownload(ﾀﾞｳﾝﾛｰﾄﾞ)釦をｸﾘｯｸすることによってGitHubの例頁から.zipﾌｧｲﾙとしてｺｰﾄﾞをﾀﾞｳﾝﾛｰﾄﾞしてください。

7. 改訂履歴

改訂	日付	注釈
A	2020年7月	初版文書公開

Microchipウェブ サイト

Microchipはwww.microchip.com/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

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TB3254技術概説(DS90003254A-2020年7月)の翻訳日本語版です。日本語では不自然となる重複する形容表現は省略されている場合があります。日本語では難解となる表現は大幅に意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 - 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 - 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