AVR1000 : XMEGA用のCｺｰﾄﾞを書く前に

要点

- 命名規定
  - ﾚｼﾞｽﾀ名
  - ﾋﾞｯﾄ名
- Cｺｰﾄﾞ名
  - ﾋﾞｯﾄと群の遮蔽
  - 群形成の遮蔽
- ﾚｼﾞｽﾀのｱｸｾｽ方法
- 再使用可能な部署関数の書き方

1. 序説

電気製品での高品質と短い開発時間の必要条件が高位ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ言語を必要にさせます。主な理由はより良い可搬性と可読性のために高位言語がｺｰﾄﾞの再使用と保守を容易にすることです。

ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ言語の選択だけが可読性と再使用の可能性、良いｺｰﾄﾞ書きの流儀を行うことを保証する訳ではありません。XMEGA®の周辺機能、ﾍｯﾀﾞ ﾌｧｲﾙ、ﾄﾞﾗｲﾊﾞはこの考えで設計されています。

最も広範囲に使用されているAVR®ﾏｲｸﾛ ｺﾝﾄﾛｰﾗ用の高位言語はCで、従って本応用記述はCﾌﾟﾛｸﾞﾗﾐﾝｸﾞに集中します。AVRのCｺﾝﾊﾟｲﾗの殆どを支援するために利用可能なｺｰﾄﾞ例は可能な限りANSI Cで書かれています。少しの例がIAR Embedded Workbench®に特化していますが、概念と方法は小変更で他のｺﾝﾊﾟｲﾗに対して使用できます。IAR指定例は明確に記されています。

2. XMEGA部署

AVR XMEGAはAVR CPUｺｱ、SRAM、ﾌﾗｯｼｭ、EEPROM、幾らかの周辺機能部署の、多数の構成物で構成されています。この構成物は"部署型(Module type)"と呼ばれます。XMEGAは与えられた部署型の実物を1つ以上持つことができます。或る部署型の全ての実物は同じ特性と機能を持ちます。

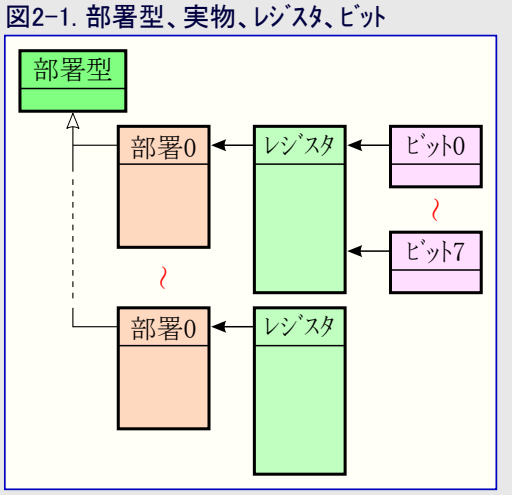
いくつかの部署型は別の部署型の下位型で有り得ます。これらは上位型の機能(とﾚｼﾞｽﾀ)の下位型を受け継ぎ、受け継いだ機能の全ては完全に互換です。これは例えばﾀｲﾏ/ｶｳﾝﾀとI/Oﾎﾟｰﾄに適用します。ﾀｲﾏ/ｶｳﾝﾀに対する部署型の下位型は完全なﾀｲﾏ/ｶｳﾝﾀ部署よりも少ない比較と捕獲のﾁｬﾈﾙを持つことを意味します。同様にI/Oﾎﾟｰﾄは8ﾋﾟﾝよりも少ないﾋﾟﾝを持つかもしれません。

部署型は"USART"で有り得、一方部署実物は例えば"USARTC0"で、ここで接尾語"C0"はこの実物が"ﾎﾟｰﾄCでのUSART番号0"であることを示します。単純化のため、これらを区別する必要がなければ、本資料を通じて部署実物は部署として参照されます。

各部署は制御と状態のﾋﾞｯﾄを含む幾つかのﾚｼﾞｽﾀを持ちます。与えられた型の全ての部署はﾚｼﾞｽﾀの同じ組(または下位の組)を含み、これらのﾚｼﾞｽﾀの全てが制御と状態のﾋﾞｯﾄの同じ組(または下位の組)を含みます。

各部署はI/Oﾒﾓﾘ割り当て内に固定の基準ｱﾄﾞﾚｽを持ち、部署内に含まれる全てのﾚｼﾞｽﾀは部署基準ｱﾄﾞﾚｽに相対する固定の差分ｱﾄﾞﾚｽを持ちます。この方法での各ﾚｼﾞｽﾀはI/Oﾒﾓﾘ空間内の絶対ｱﾄﾞﾚｽではなく、その差分によって定義される相対ｱﾄﾞﾚｽを持ちます。ﾚｼﾞｽﾀ差分ｱﾄﾞﾚｽは部署型の全ての実物に対して等しく、ﾄﾞﾗｲﾊﾞ書きの簡単化した作業は指定型の全部署に対して使用することができます。

図2-1. 部署型、実物、ﾚｼﾞｽﾀ、ﾋﾞｯﾄ



8ﾋﾞｯﾄ AVR® マイクロ コントローラ

応用記述

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、ATMEL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。

Rev. 8075A-02/08, 8075AJ3-03/14

2.1. レジスタ命名規則

レジスタは大雑把に制御、状態、データのレジスタに分けて表し、そしてレジスタ命名はこれを反映します。部署の一般目的制御レジスタはCTRLと名付けられます。部署内に複数の一般目的制御レジスタが存在する場合、それらはCTRLA, CTRLB, CTRLC、以下同様に名付けられます。これはSTATUSレジスタにも適用します。

特殊機能を持つレジスタについては名前がその機能を反映します。例えば、部署の割り込みレベルを制御する制御レジスタはINTCTRLと名付けられます。

AVRのデータ バス幅が8ビットなので、より大きなレジスタは多数の8ビット レジスタを使用して実装されます。16ビット レジスタについては、上位と下位のバイトがそのレジスタ名に各々"H"と"L"を追補することによってアクセスされます。例えば、16ビット タイマ/カウンタ計数レジスタはCNTと名付けられます。2つのバイトはCNTHとCNTLと命名されます。

16ビットより大きなレジスタについては、バイトが最下位バイトから数値付けされます。例えば、32ビットADC校正レジスタはCALと命名されます。4つのバイトは(最下位バイトから最上位バイトへ)CAL0, CAL1, CAL2, CAL3と名付けられます。

殆どのCコンパイラは複数バイト レジスタに対するアクセスの自動的な扱いを提供します。その場合では、"H"または"L"なしの名前CNTがタイマ/カウンタ計数レジスタに対する16ビット アクセスの実行に使用できます。これは32ビット レジスタに対する場合もです。

2.2. ビット命名規則

レジスタのビットは個別機能または連帯機能を持つビット群の一部を持ち得ます。個別ビットは部署の許可ビット、例えばUSARTのENABLEビットで有り得ます。ビット群は2つ以上のビットから成り得、それらが属する部署の特定形成を連帯して選択します。ビット群は2^n選択までを提供し、ここでのnはビット群のビット数です。USART受信完了割り込みレベルを制御する2つのビット、RXINTLVL1,0はビット群の例です。これら2つのビットは右の選択を提供します。

表2-1. RXINTLVLビットと対応する割り込みレベル選択

RXINTLVL1	RXINTLVL0	割り込みレベル選択
0	0	割り込みOFF
0	1	低レベル割り込み
1	0	中レベル割り込み
1	1	高レベル割り込み

群の一部であるビットは常に数値接尾語を持ちます。群の一部でないビットは決して数値接尾語を持ちません。タイマ/カウンタ制御レジスタDはEVACTとEVSELの2つのビット群を持ちます。これらの群内のビットは数値接尾語を持ち、一方EVDLYはこれがビット群の一部でなく、数値接尾語を持ちません。

表2-2. タイマ/カウンタ制御レジスタD(CTRLD)内のビットに対するビット群とビット名

ビット群	EVACT			-	EVSEL			
ビット名	EVACT2	EVACT1	EVACT0	EVDLY	EVSEL3	EVSEL2	EVSEL1	EVSEL0
ビット番号	7	6	5	4	3	2	1	0

3. XMEGA用Cコードの書き方

後続項はXMEGA用のCコードの書き方に焦点を当てます。例は異なるXMEGAデバイス間の可搬性と高い可読性のコードの作成方法を示します。例は検証と保守が容易なコードを書く指針にも使用できます。

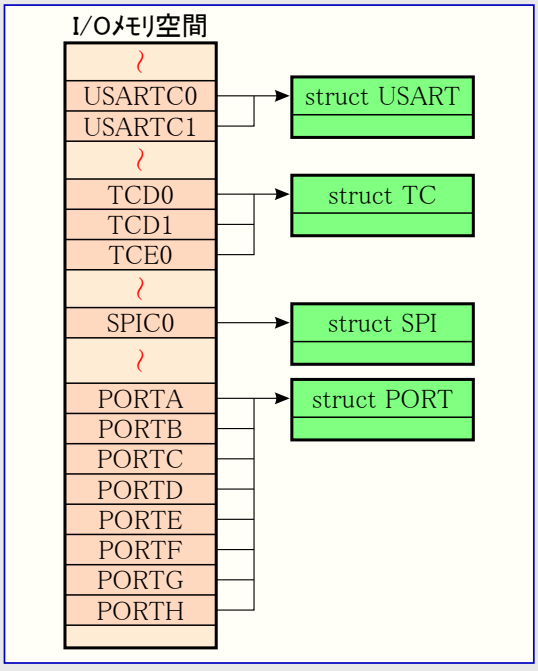
XMEGAの部署はメモリ空間で専用の連続した塊で配置され、カプセル内の単位群として見ることができます。これはCコード書き時の部署がアクセスされる方法を反映しています。部署はC構造体を使用してカプセル化され、そしてそれには部署の全レジスタが含まれます。図3-1.はこの図式を示します。

いくつかのレジスタが部署と直接の関連を持たないことに注意してください。構造体が部署に関連するレジスタに使用されるため、これらは構造体でカプセル化されません。

より大きなコード作業に関しては部署構造が可読性ではなく、コンパイラが部署ドライバの再使用をできるため、それによって非常に簡潔なコード作成の利点を提供します。これは後でもっと詳細に記述されます。

本資料は命令規則とAVRプログラミングの熟練者が使用するのとは異なるアクセス方法を導入しますが、レジスタ アクセスの"旧来の方法"も未だヘッダ ファイルによって支援されていることも留意されるべきです。これはビット レベルにも適用します。

図3-1. I/Oメモリ空間の専用部に配置された部署



3.1. XMEGA ヘッダ ファイル

各XMEGAデバイスに対して専用のヘッダ ファイルが利用可能です。目的デバイスが(IAR EWAVR用IDEでそれの使用と仮定する)プロジェクト設定で指定されている場合、コード リスト3-1.で示されるようにデバイス ファイルがインクルードされるなら、IARコンパイラは正しいヘッダ ファイルを自動的にインクルードします。

コード リスト3-1. IARヘッダ ファイルのインクルード

```
#include <ioavr.h>
```

この利点は目的デバイス変更の場合にソース ファイル変更の必要がなく、プロジェクト設定の変更だけなことです。

3.2. 部署のレジスタ

I/O割り当ては与えられた周辺機能部署に関する全レジスタが1つの連続するメモリ部に配置されるように展開されます。違う部署に属するレジスタは混ぜられません。これは全ての周辺機能部署をC構造体に構成することを可能にし、ここで構造体のアドレスはその部署の基準アドレスを定義します。部署に属する全レジスタがその部署構造体の要素です。

例は設定可能な多段割り込み制御器(PMIC)部署です。この部署に対する構造体宣言はコード リスト3-2.で示され、この例はコード リスト3-3.で使用されます。コード リスト3-3.の例はPMICと名付けられたPMIC_t型との仮定であることに注意してください。これは本資料に於いて後で網羅されます。

コード リスト3-2. 部署構造体宣言

```
typedef struct PMIC_struct {
    unsigned char STATUS;           // 状態レジスタ
    unsigned char INTPRI;           // 割り込み優先権
    unsigned char CTRL;             // 制御レジスタ
 } PMIC_t;
```

コード リスト3-3. 部署構造体の使い方

```
unsigned char temp;
temp = PMIC.STATUS;                 // tempへ状態レジスタ読み込み
PMIC.CTRL |= PMIC_PMRRPE_bm;        // 制御レジスタ内のPMRRPEビット設定(1)
```

3.2.1. 部署構造体内の複数語レジスタ

いくつかのレジスタは16または32ビットを表すために他のレジスタと連結して使用されます。例としてコード リスト3-4.で示されるADC構造体宣言を見ることができます。

コード リスト3-4. ADC構造体宣言

```
typedef struct ADC_struct {
    unsigned char CH0MUXCTRL;       // チャネル0多重器(MUX)制御レジスタ
    unsigned char CH1MUXCTRL;       // チャネル1多重器(MUX)制御レジスタ
    unsigned char CH2MUXCTRL;       // チャネル2多重器(MUX)制御レジスタ
    unsigned char CH3MUXCTRL;       // チャネル3多重器(MUX)制御レジスタ
    unsigned char CTRLA;            // 制御レジスタA
    unsigned char CTRLB;            // 制御レジスタB
    unsigned char REFCTRL;          // 基準電圧制御レジスタ
    unsigned char EVCTRL;           // 事象制御レジスタ
    WORDREGISTER(CH0RES);           // チャネル0結果レジスタ
    WORDREGISTER(CH1RES);           // チャネル1結果レジスタ
    WORDREGISTER(CH2RES);           // チャネル2結果レジスタ
    WORDREGISTER(CH3RES);           // チャネル3結果レジスタ
    unsigned char reserved_0x10;
    unsigned char reserved_0x11;
    unsigned char CH0INTCTRL;       // チャネル0割り込み制御レジスタ
    unsigned char CH1INTCTRL;       // チャネル1割り込み制御レジスタ
    unsigned char CH2INTCTRL;       // チャネル2割り込み制御レジスタ
    unsigned char CH3INTCTRL;       // チャネル3割り込み制御レジスタ
    unsigned char INTFLAGS;         // 割り込み要求フラグ レジスタ
    WORDREGISTER(CMP);              // 比較値レジスタ
    unsigned char PRESCALER;        // クロック前置分周レジスタ
    unsigned char reserved_0x1A;
    unsigned char reserved_0x1B;
    unsigned char CALCTRL;          // 校正制御レジスタ
    DWORDREGISTER(CAL);             // 校正値レジスタ
} ADC_t;
```

コード リスト3-4.ではADCチャネル結果レジスタCH0RES, CH1RES, CH2RES, CH3RESと比較レジスタCMPが16ビット値です。コード リスト3-5.で示されるWORDREGISTERマクロを使用して宣言されます。校正レジスタCALは32ビット値で、コード リスト3-6.で示されるDWORDREGISTERマクロを使用して宣言されます。

コード リスト3-5. WORDRGISTERマクロ

```
#define WORDREGISTER(regname) ¥
    union { ¥
        unsigned short regname; ¥
        struct { ¥
            unsigned char regname ## L; ¥
            unsigned char regname ## H; ¥
        }; ¥
    }
```

コード リスト3-6. DWORDRGISTERマクロ

```
#define DWORDREGISTER(regname) ¥
    union { ¥
        unsigned long regname; ¥
        struct { ¥
            unsigned char regname ## 0; ¥
            unsigned char regname ## 1; ¥
            unsigned char regname ## 2; ¥
            unsigned char regname ## 3; ¥
        }; ¥
    }
```

ここで見られるように、WORDREGISTERマクロは上位と下位のバイトに対して各々"H"と"L"の接尾語を使用します。WORDREGISTERマクロはバイト順を示すために数値接尾語を使用します。16ビットと32ビットの両レジスタはコード リスト3-7.で示されるように、接尾語なしのレジスタ名を使用することによって、16ビット/32ビット動作でアクセスすることができます。

コード リスト3-7. 大きさの違うレジスタのアクセス

```
unsigned char tempByte;
unsigned int  tempWord;
unsigned long tempDword;

tempByte = ADCA0.CTRLA;              // 制御レジスタA読み込み
tempWord = ADCA0.CH0RES;             // チャネル0の16ビット結果読み込み
tempDword = ADCA0.CAL;               // 32ビット校正値読み込み
```

コード リスト3-7.は単一バイト レジスタCTRLAの読み方、16ビット操作を使用する2つのCH0RESH/Lレジスタの読み方、32操作を使用する4つのCAL3/2/1/0レジスタの読み方を示します。Cコンパイラは複数バイト レジスタを自動的に扱います。けれども或る状態で、不正を避けるために非分断操作での読み書きが必要とされるかもしれないことに注意してください。この場合は割り込み処理ルーチンが複数バイト アクセスを妨害しないことを保証するために、複数バイト アクセスの間、割り込みが禁止されなければなりません。AVR1306応用記述はXMEGAのタイマ/カウンタ部署に対して行われるレジスタの非分断アクセス法の例を含みます。

3.3. 部署アドレス

全ての周辺機能部署の定義はXMEGAに関して利用可能なデバイス ヘッダ ファイルで得られます。部署のアドレスは利用可能な殆どのCコンパイラと互換にするためにANSI Cで指定されています。コード リスト3-8.はポートAでのADC 0がどう定義されているかを示します。

コード リスト3-8. 周辺機能部署定義

```
#define ADCA    (*(volatile ADC_t *) 0x0200)
```

コード リスト3-8.は部署実物の基準アドレスと一致するメモリ内の絶対アドレスへの直参照位置指示子を使用する部署実物定義法を示します。部署の位置指示子はXMEGAヘッダ ファイルで予め定義されており、従ってソース コードでこれらの定義を追加する必要はありません。

3.4. ビット遮蔽値とビット群遮蔽値

レジスタのビットは予め定義された遮蔽値、または代わりにビット位置値を使用して操作することができます。ビット位置は殆どの作業に対して推奨されません。予め定義されているビット遮蔽値はビット マスクまたはビット群、ビット群マスクまたは短縮して群マスクと呼ばれる、個別ビットに関連したどれかです。

ビット遮蔽値は個別ビットの設定(1)と解除(0)を行う時の両方で使用されます。ビット群遮蔽値は主にビット群内の複数ビットを解除(0)する時に使用されます。ビット群の部分である複数ビットの設定(1)は3.5.項で網羅されます。

3.4.1. ビット遮蔽値

タイマ/カウンタ制御レジスタD(CTRLD)を考察してください。このレジスタのビット群、ビット名、ビット位置、ビット遮蔽値は**表3-1.**で見ることができます。

表3-1. タイマ/カウンタ制御レジスタD(CTRLD)内のビットに対するビット群、ビット名、ビット遮蔽値

ビット群	EVACT			-	EVSEL			
ビット名	EVACT2	EVACT1	EVACT0	EVDLY	EVSEL3	EVSEL2	EVSEL1	EVSEL0
ビット番号	7	6	5	4	3	2	1	0
ビット遮蔽値	$80	$40	$20	$10	$08	$04	$02	$01

ビットの名前がそれらを扱うためにコンパイラに対して固有であることが必要なので、全てのビットはそれらが属する部署型で接頭語を付けられています。多くの場合で、部署型名は略名にされています。タイマ/カウンタ部署に関連する全てのビット定義に関しては、ビット名が"TC_"の接頭語を付けられています。

ビット遮蔽値とビット位置を区別するために接尾語も追補されます。ビット遮蔽値に対する接尾語は"_bm"です。従ってEVDLYビットに対するビット遮蔽値名はTC_EVDLY_bmです。**コード リスト3-9.**はビット遮蔽値の代表的な使い方を示します。タイマ/カウンタD0のCTRLDレジスタ内のEVDLYビットが設定(1)され、レジスタ内の他の全てのビットを無変化のままにします。

コード リスト3-9. ビット遮蔽値の使い方

```
TCD0.CTRLD |= TC_EVDLY_bm;              // ビット遮蔽値指定での設定(1)
```

3.4.2. ビット群遮蔽値

多くの関数がビット群によって制御されています。タイマ/カウンタCTRLDレジスタのEVACT2～0とEVSEL3～0は群化されたビットです。群内のビット値が特定形成を選択します。

ビット群内のビット変更時に新しい値を指示する前にビット群を解除することが度々必要とされます。別の方法でそれにするには、設定(1)すべきビットを設定(1)することは不充分で、解除(0)すべきビットを解除(0)することも必要とされます。これを容易に行うためにビット群遮蔽値が定義されています。群遮蔽値はビット群内のビットと同じ名前を使用し、"_gm"の接尾語を付けられています。

コード リスト3-10.は群遮蔽値がビット遮蔽値とどう関連するかを示します。現実には群遮蔽値がヘッダ ファイルで予め計算されていて、故にコンパイラは何度も何度も同じ定数を計算する必要がありません。

コード リスト3-10. 群遮蔽値とビット遮蔽値の関連

```
#define TC_EVACT_gm   (TC_EVACT2_bm | TC_EVACT1_bm | TC_EVACT0_bm)
```

ビット群遮蔽値は新しい値を書く前にビット群の旧形成を解除することを主に意図されています。**コード リスト3-11.**はこれがどう行われ得るかを示します。コードはタイマ/カウンタD0のCTRLDレジスタ内のEVACTビット群を解除します。この構成はそれ自身であまり有用ではありません。群遮蔽値は代表的に群形成遮蔽値と連結して使用され、これは3.5.項で網羅されます。

コード リスト3-11. 群遮蔽値の使い方

```
TCD0.CTRLD &= ~(TC_EVACT_gm);           // 群遮蔽値での群ビット解除
```

3.5. ビット群形成遮蔽値

望む形成にビット群を設定する時に何のビット模様が使用される必要があるかを調査するためにデータシートを調べることが度々必要とされます。これはコードを読むまたはデバッグする時にも適用されます。可読性を増してビット群での不正なビット設定の可能性を最小とするために、多数の群形成遮蔽値が利用可能にされています。

群形成遮蔽値の名前は部署型、群名、形成内容と接尾語"_gc"との連結で、これが群形成を示します。例については**図3-2.**をご覧ください。

図3-2. 群形成名合成(例としてUSART受信完了割り込みレベル ビットを使用)

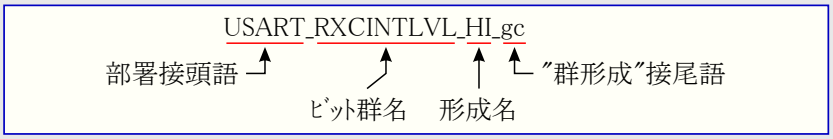


図3-2.の群形成を調査すると、USART部署でRXCINTLVLﾋﾞｯﾄに対する形成を選択するのにそれが使用されているのを見ることができます。この指定群形成は高(HI)割り込みﾚﾍﾞﾙを選択します。

受信完了割り込みﾚﾍﾞﾙに対するﾋﾞｯﾄ群は2ﾋﾞｯﾄのRXINTLVL1～0から成ります。表3-2.はこのﾋﾞｯﾄ群に対して利用可能な群形成を示します。

形成名は"OFF"、"LO"、"MED"、"HI"です。これらの名前は特定の形成遮蔽値が何を形成するかを理解するための非常に小さな努力を必要とすることで、ｺｰﾄﾞを書きそして保守することを大変容易にします。

表3-2. RXINTLVLﾋﾞｯﾄと対応する部署割り込みの割り込みﾚﾍﾞﾙ

RXINTLVL1	RXINTLVL0	割り込みﾚﾍﾞﾙ	群形成遮蔽値
0	0	割り込みOFF	USART_RXCINTLVL_OFF_gc
0	1	低ﾚﾍﾞﾙ割り込み	USART_RXCINTLVL_LO_gc
1	0	中ﾚﾍﾞﾙ割り込み	USART_RXCINTLVL_MED_gc
1	1	高ﾚﾍﾞﾙ割り込み	USART_RXCINTLVL_HIGH_gc

ﾋﾞｯﾄ群を新しい形成に変更するには、旧形成が先に消去されるのを保証するために、ﾋﾞｯﾄ群形成が代表的にﾋﾞｯﾄ群遮蔽値と連結して使用されます。ｺｰﾄﾞ ﾘｽﾄ3-12.はUSART C0受信完了割り込みﾚﾍﾞﾙを中ﾚﾍﾞﾙに再形成するのに群遮蔽値と形成遮蔽値がどう使用され得るかを示します。

ｺｰﾄﾞ ﾘｽﾄ3-12. ﾋﾞｯﾄ群形成の変更

```
USARTC0.INTCTRL = (USARTC0.INTCTRL & ~USART_RXCINTLVL_gm ) | USART_RXCINTLVL_MED_gc;
```

それにも関わらず、ｺｰﾄﾞ ﾘｽﾄ3-12.のｺｰﾄﾞを2つの独立したｺｰﾄﾞ行、ﾋﾞｯﾄ群を解除する1つと新しい値を設定する1つに分割する誘惑に駆られるかもしれないことに注意してください。これは推奨されません。INTCTRLﾚｼﾞｽﾀがvolatileとして定義されているので、2つの独立したｺｰﾄﾞ行は1度に代えて2度のINTCTRLﾚｼﾞｽﾀ読み書きを起動します。ｺｰﾄﾞを非効率にすることに加えて、これは周辺機能を予期せぬ状態に置き得ます。

ﾋﾞｯﾄ解除のための群遮蔽値の使用は常に必要な訳ではありません。ｺｰﾄﾞ ﾘｽﾄ3-13.はUSART C0の全ての割り込みﾚﾍﾞﾙが同時にどう形成され得るかを示します。受信完了、送信完了、USARTﾃﾞｰﾀ ﾚｼﾞｽﾀ空の割り込みﾚﾍﾞﾙは各々中(MEDIUM)、なし(OFF)、低(LOW)の割り込みﾚﾍﾞﾙに設定されます。

ｺｰﾄﾞ ﾘｽﾄ3-13. ﾚｼﾞｽﾀ内の全形成同時設定

```
USARTC0.INTCTRL = USART_RXCINTLVL_MED_gc |
                  USART_TXCINTLVL_OFF_gc |
                  USART_DREINTLVL_LO_gc;
```

3.5.1. 群形成遮蔽値の列挙

ﾋﾞｯﾄ遮蔽値や群遮蔽値のようではなく、ﾋﾞｯﾄ形成遮蔽値はCの列挙子を使用して定義されています。1つの列挙子は各ﾋﾞｯﾄ群に対して定義されています。USART TXCINTLVLﾋﾞｯﾄに対する列挙子はｺｰﾄﾞ ﾘｽﾄ3-14.で示されます。

ｺｰﾄﾞ ﾘｽﾄ3-14. USART TXCINTLVL列挙子

```
typedef enum {
    USART_RXCINTLVL_OFF_gc = (0x00 << 4);
    USART_RXCINTLVL_LO_gc  = (0x01 << 4);
    USART_RXCINTLVL_MED_gc = (0x02 << 4);
    USART_RXCINTLVL_HI_gc  = (0x03 << 4);
} USART_RXCINTLVL_t;
```

ｺｰﾄﾞ ﾘｽﾄ3-14.で見えるように、列挙子の名前は部署型(USART)、ﾋﾞｯﾄ群(RXCINTLVL)、接尾語(_t)と結び付き、これはﾃﾞｰﾀ型を示します。

列挙子定数の各々はそれ自体で使用される時に大抵通常の定数のように振舞います。けれども、列挙子を使用することはそれが新しいﾃﾞｰﾀ型を作成することの利点を持ちます。USART_RXCINTLVL_tは丁度intやcharのように型名として使用できます。列挙子定数は整数(integer)として直接使用できますが、整数の列挙型割り当てはｺﾝﾊﾟｲﾗの警告を起動するでしょう。これは書き手の利点として使用することができます。USART部署に対する受信完了割り込みﾚﾍﾞﾙ設定を受け入れる関数を思い浮かべてください。関数が整数型(例えばunsigned char)として割り込みﾚﾍﾞﾙを受け入れる場合、正当または不当などんな値も関数を通り得ます。関数が代わりにUSART_RXCINTLVL_t型のﾊﾟﾗﾒｰﾀを受け入れるなら、USART_RXCINTLVL_tで予め定義された4つの定数だけがその関数を通ることができます。他の何かの通過はｺﾝﾊﾟｲﾗの警告に終ります。

ｺｰﾄﾞ ﾘｽﾄ3-14.の定数が実際のﾋﾞｯﾄ位置にｼﾌﾄされていることに注意してください。これは列挙子定数がﾚｼﾞｽﾀに書かれるべき実際の値であることを意味します。追加のｼﾌﾄは必要ありません。

3.6. 関数呼び出しと部署ﾄﾞﾗｲﾊﾞ

複数の実物を持つ部署型用のﾄﾞﾗｲﾊﾞを書くとき、全ての実物が同じﾚｼﾞｽﾀ ﾒﾓﾘ割り当てを持つ事実は、その部署型の全ての実物に対して再使用可能なﾄﾞﾗｲﾊﾞを作成することで利用可能になります。ﾄﾞﾗｲﾊﾞが適切な部署実物を指示するﾎﾟｲﾝﾀ引数を取る場合、そのﾄﾞﾗｲﾊﾞはこの型の全部署に対して使用することができます。可搬性を考慮するとき、これは偉大な優位性を表します。

8つのﾀｲﾏ/ｶｳﾝﾀを持つﾃﾞﾊﾞｲｽを考察してください。ﾀｲﾏ/ｶｳﾝﾀ部署を初期化してｱｸｾｽするための関数は全ての部署実物で共用できます。例え部署ﾎﾟｲﾝﾀを関数へ渡す小さなｵｰﾊﾞｰﾍｯﾄﾞがあるとは言え、各部署型の全ての実物に対してｺｰﾄﾞが再使用されるので、総ｺｰﾄﾞ量は頻繁に低減されるでしょう。更に重要なのは、開発時間、保守費用、可搬性がこの方法を使用することによって大いに改善され得ることです。

ｺｰﾄﾞ ﾘｽﾄ3-15.は何れかのﾀｲﾏ/ｶｳﾝﾀ部署に対してｸﾛｯｸ元を選択するために部署ﾎﾟｲﾝﾀを使用する関数を示します。

ｺｰﾄﾞ ﾘｽﾄ3-15. 部署実物ﾎﾟｲﾝﾀを使用する関数の例

```
void TC_ConfigClockSource(volatile TC_t * tc, TC_CLKSEL_t clockSelection)
{
        tc->CTRLA = tc->CTRLA & ~TC_CLKSEL_gm | clockSelection;
}
```

この関数は部署ﾎﾟｲﾝﾀの型TC_tと群形成型TC_CLKSEL_tの2つの引数を取ります。関数内のｺｰﾄﾞはtcﾊﾟﾗﾒｰﾀを通して提供されるﾀｲﾏ/ｶｳﾝﾀ部署に対して、新しいｸﾛｯｸ選択を設定するためのCTRLAﾚｼﾞｽﾀのｱｸｾｽに、ﾀｲﾏ/ｶｳﾝﾀ部署ﾎﾟｲﾝﾀを使用します。

ｺｰﾄﾞ ﾘｽﾄ3-16.は異なるｸﾛｯｸ選択で異なるﾀｲﾏ/ｶｳﾝﾀを形成するのに**ｺｰﾄﾞ ﾘｽﾄ3-15.**の関数がどう使用され得るかを示します。

ｺｰﾄﾞ ﾘｽﾄ3-16. ﾊﾟﾗﾒｰﾀとして部署ﾎﾟｲﾝﾀを取る関数の呼び出し

```
TC_ConfigClockSource (&TCC0, TC_CLKSEL_OFF_gc);
TC_ConfigClockSource (&TCC1, TC_CLKSEL_DIV1_gc);
TC_ConfigClockSource (&TCD0, TC_CLKSEL_DIV2_gc);
TC_ConfigClockSource (&TCD1, TC_CLKSEL_DIV1024_gc);
```

4. ｺｰﾄﾞ書きの代替法

利便性とﾌﾟﾛｸﾞﾗﾐﾝｸﾞの流儀変更をAVRﾌﾟﾛｸﾞﾗﾐﾝｸﾞの熟練者に強いるのを避けるため、構造体を伴わないﾌﾟﾛｸﾞﾗﾐﾝｸﾞの流儀が未だ使用可能です。それはmegaAVR®に対して使用されるﾋﾞｯﾄ名形式を使用することも可能です。本項はﾋﾞｯﾄ名の使用とﾚｼﾞｽﾀ ｱｸｾｽの代替方法を簡単に記述します。

4.1. ﾚｼﾞｽﾀ名

部署構造体を使用しないでどのﾚｼﾞｽﾀもｱｸｾｽすることが可能です。ﾚｼﾞｽﾀを直接参照するには、部署実物名とﾚｼﾞｽﾀ名を下線(_ :ｱﾝﾀﾞｰｽｺｱ)で連結してください。同じ命名規則がｱｾﾝﾌﾞﾘ言語ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ時に使用されます。

例: ﾀｲﾏ/ｶｳﾝﾀC0のCTRLAﾚｼﾞｽﾀにｱｸｾｽするには、名前としてTCC0_CTRLAを使用してください。

4.2. ﾋﾞｯﾄ位置

ﾋﾞｯﾄを設定(1)または解除(0)するのにﾋﾞｯﾄ遮蔽値を使用することが可能です。ﾚｼﾞｽﾀ内のﾋﾞｯﾄ位置はﾋﾞｯﾄ位置に対する追加の接尾語"_bp"で、ﾋﾞｯﾄ遮蔽値と同じ名前を使用して定義されています。**ｺｰﾄﾞ ﾘｽﾄ4-1.**はﾚｼﾞｽﾀ形成にﾋﾞｯﾄ位置がどう使用され得るかを示します。

ｺｰﾄﾞ ﾘｽﾄ4-1. ﾚｼﾞｽﾀ ｱｸｾｽ代替ｺｰﾄﾞ

```
PORTB_OUT = (1 << PORTB_OUT0_bp);    // ﾎﾟｰﾄB出力(PORTB_OUT)のﾋﾞｯﾄ0を設定(1)
```

ﾋﾞｯﾄ位置定義は互換性の理由でｲﾝｸﾙｰﾄﾞされています。これらはﾋﾞｯﾄ番号を使用する命令に対してｱｾﾝﾌﾞﾘ言語でﾌﾟﾛｸﾞﾗﾐﾝｸﾞする時にも必要とされます。

5. 要約

参照用に、ﾋﾞｯﾄ形成を扱う時に使用される各種接尾語の概要が**表5-1.**で一覧されます。

XMEGAに対してCｺｰﾄﾞを書くのに提唱された方法を使用することは、必須を意味しませんが、堅強、可搬性、再使用性、高い可読性のｺｰﾄﾞを保証するために、提供される利点の使用が考慮されるべきです。より大きな企画とより多くの機能のﾃﾞﾊﾞｲｽはより大きな優位性を持ちます。小さな企画に関しては恩恵がないと主張し得ますが、その主張をひっくり返すと不利もまたないと言うことができます。

1つ確かなことがあります。それは良いｺｰﾄﾞの流儀は常に利点です。

表5-1. 接尾語の概要

接尾語	意味	例
_gm	群遮蔽値	TC_CLKSEL_gm
_gc	群形成値	TC_CLKSEL_DIV1_gc
_bm	ﾋﾞｯﾄ遮蔽値	TC_CCAEN_bm
_bp	ﾋﾞｯﾄ位置	TC_CCAEN_bp

本社

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
TEL 1(408) 441-0311
FAX 1(408) 487-2600

国外営業拠点

Atmel Asia
Unit 1-5 & 16, 19/F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
Hong Kong
TEL (852) 2245-6100
FAX (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-
Yvelines Cedex
France
TEL (33) 1-30-60-70-00
FAX (33) 1-30-60-71-11

Atmel Japan
104-0033 東京都中央区
新川1-24-8
東熱新川ﾋﾞﾙ 9F
ｱﾄﾒﾙ ｼﾞｬﾊﾟﾝ株式会社
TEL (81) 03-3523-3551
FAX (81) 03-3523-7581

製品窓口

ｳｪﾌﾞｻｲﾄ
www.atmel.com

技術支援
avr@atmel.com

販売窓口
www.atmel.com/contacts

文献請求
www.atmel.com/literature

本応用記述はATMELのAVR1000応用記述(doc8075.pdf Rev.8075A-02/08)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。