

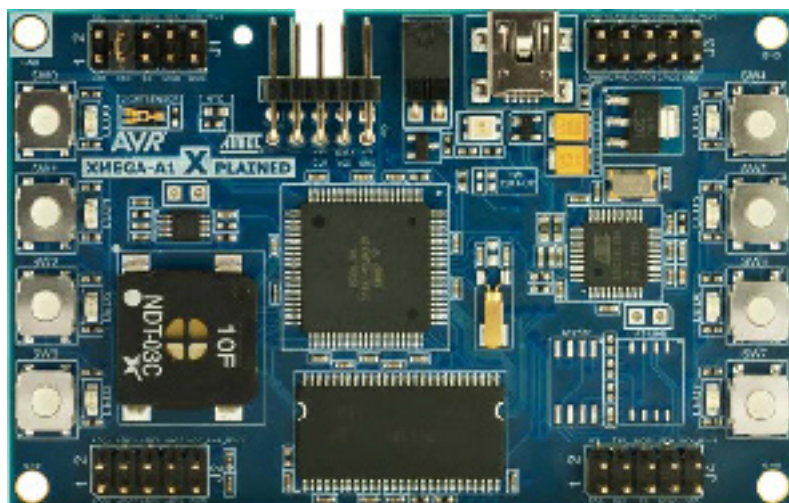
## AVR1512 : XMEGA-A1 Xplain練習 – XMEGA 基礎

## 前提条件

- 必要な知識
  - ・ マイクロコントローラとCプログラミング言語の基礎知識
- ソフトウェア必要条件
  - ・ ATMEL® AVR Studio® 5
- ハードウェア必要条件
  - ・ XMEGA-A1 Xplain評価基板
  - ・ JTAGICEmk3(またはJTAGICEmk II かAVR ONE!)
- 予想完了時間
  - ・ 2時間

## 1. 序説

本応用記述は4つの課題で様々な構想を示すのにI/Oポートを用いてATMEL AVR® XMEGA®の基礎を網羅します。この練習の目的はより効率的で簡素なコードのためにXMEGAヘッダファイルとXMEGAのいくつかの機能を利用する小さなコードの断片で始めさせることです。



## 2. XMEGA用のCコード書き

開発時間の短縮の圧力と同時に電気製品の高品質保証は高位プログラミング言語を必要条件にさせます。それは保守と再利用が容易でより良い可搬性と可読性を与えます。

プログラミング言語の選択だけでは高い可読性と再利用性を確実にせず、それは良いコード書きの流儀です。従ってATMEL AVR® XMEGA®の周辺機能、ヘッダファイルとドライバはこの考えで設計されています。

以降の副項目はXMEGAに対する新しいプログラミング形式のいくつかの概要説明を与えます。より多くの詳細な説明は「AVR1000:XMEGAに対してCコードを書く前に」応用記述で与えられます。

## 2.1. ビット遮蔽とビット群遮蔽

レジスタ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(これは推奨されません。)を用いて操作することができます。予め定義されたビット遮蔽値はビット遮蔽と呼ばれる個別ビットまたはビット群に関連付けられます。予め定義されたビット群遮蔽は短縮で群遮蔽と呼ばれます。

ビット遮蔽は個別ビットの設定と解除の両方の時に使用されます。ビット群遮蔽は主にビット群内の複数ビットを解除する時に用いられます。

あなたがATMEL megaAVR®やATMEL tinyAVR®のAVRマイクロコントローラに慣れているなら、代表的に例1.と同様に事象遅延(EVDLY)ビットを設定(1)するでしょう。

例1. 標準的なATMELのAVRマイクロコントローラでのビット位置使用法

```
TCDO. CTRLD |= (1 << EVDLY);
```



8ビット ATMEL  
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. 8398A-07/11, 8398AJ1-03/14

これはこれらのマイクロコントローラ用のヘッダファイルがビット位置を指示し、そしてシフト演算子(<<)でビット遮蔽を作成するためです。XMEGAヘッダファイルとで、これはビット位置とビット遮蔽が既に定義されているため、より読み易くなります。

### 例2. XMEGAでのビット遮蔽使用法

```
TCDO. CTRLD |= TCO_EVDLY_bm; // ビット遮蔽指示子とでの使用
```

### 例3. XMEGAでのビット位置使用法

```
TCDO. CTRLD |= (1 << TCO_EVDLY_bp); // ビット位置指示子とでの使用
```

例2.の形式を用いることが推奨されますが、両例ともビット4に1の値を設定、即ちレジスタ値は2進値'0001 0000'と論理和(OR)されます。多くの形態設定はビットの群によって制御されます。例えば、タイマ/カウンタ制御レジスタD(CTRLD)(図1.をご覧ください。)で、事象活動(EVACT2~0)と事象元選択(EVSEL3~0)のビットは群化されたビットです。群内のビットの値が形態設定を選びます。群遮蔽はビット群内のビットらと同じ名前と接尾語"\_gm"を用い、一方ビット群の位置は"\_gp"が接尾されます。

図1. ATMEL XMEGA A手引書で描かれるようなタイマ/カウンタ制御レジスタ

ビット	7	6	5	4	3	2	1	0	
+\$03	EVACT2~0			EVDLY	EVSEL3~0				CTRLD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
初期値	0	0	0	0	0	0	0	0	

ビット群遮蔽は新しい値を書く前にビット群の旧形態設定を最初に解除(=0)することが意図されています。ビット群位置は数値係数、例えばPLL用の倍率係数を設定する時に有用です。

### 例4. XMEGAでのビット位置使用法

```
TCDO. CTLD &= ~(TCO_EVACT_gm); // 群遮蔽での群ビット解除
```

それらがXMEGAヘッダファイル内にあるので、遮蔽とビットの関連を調べることで、上記が何を行うかを見ます。

```
#define TCO_EVACT_gm 0xE0 /* 事象活動群遮蔽 */
#define TCO_EVACT_gp 5 /* 事象活動群位置 */

#define TCO_EVACT0_bm (1<<5) /* 事象活動ビット0遮蔽 */
#define TCO_EVACT0_bp 5 /* 事象活動ビット0位置 */
#define TCO_EVACT1_bm (1<<6) /* 事象活動ビット1遮蔽 */
#define TCO_EVACT1_bp 6 /* 事象活動ビット1位置 */
#define TCO_EVACT2_bm (1<<7) /* 事象活動ビット2遮蔽 */
#define TCO_EVACT2_bp 7 /* 事象活動ビット2位置 */
```

TCO\_EVACT\_gmは1110 0000の2進値を持ち、それはビット群を解除するのに非常に有用です。

## 2.2. ビット群形成遮蔽

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、多数の群形成遮蔽値が利用可能にされています。群形成の名前は接尾語"\_gc"を持ちます。

図2. 群形成名合成



図2.からUSART部署で受信完了割り込みレベル(RXCINTLVL)ビットに対して使用される群形態設定を見ることができます。この指定群形態設定は高(HI)割り込みレベルを選択します。

ビット群を新しい形態設定に変更するには、旧形態設定が先に消去されるのを保証するため、代表的にビット群形成がビット群遮蔽値と連結して使用されます。

### 例5.

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

上の例は共に使用される群遮蔽と形成遮蔽を示します。例の最初の部分(USARTC0. CTRLA & ~USART\_RXCINTLVL\_gm)は例4.で示されるものと同様の方法でUSARTC0レジスタ内のRCXINTLVLビットを解除(=0)します。最後の部分(USART\_RXCINTLVL\_MED\_gc)は中位割り込みレベルを得るように新しい値を設定します。

このコードはレジスタ内の他のビットに影響を及ぼさず、USARTC0受信完了割り込みレベルを中に再形態設定するのに用いられます。多数の応用記述と共にやって来るドライバの多くでこれと同様のコードを見ましょう。

## 2.3. 部署レジスタと部署アドレス

ATMEL XMEGAのI/O割り当ては与えられた周辺機能部署に関する全てのレジスタが1つの連続するメモリ部に配置されるように構成されます。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。

以下のコードは設定可能な多段割り込み制御器(PMIC)に関する各種レジスタがATMEL XMEGAヘッダファイルでどう定義されるかを示します。

### 例. XMEGAヘッダファイル(iox128a1.h)での定義

```
typedef struct PMIC_struct {
    register8_t STATUS;           /* 状態レジスタ */
    register8_t INTPRI;          /* 割り込み優先権 */
    register8_t CTRL;           /* 制御レジスタ */
} PMIC_t
```

このコードはPMIC部署で利用可能なレジスタを定義します。構造体は指定メモリアドレスでPMICを定義するのに使用されます。

### 例. 周辺機能部署定義

```
#define PMIC (*(PMIC_t *) 0x00A0)
```

上の例は部署事例定義が部署事例基準アドレスと符合して、メモリに於いて間接参照ポインタを絶対アドレスへどう使用するかを示します。部署ポインタはXMEGAヘッダファイルで予め定義され、従ってあなたのソースファイルにそれらの定義を追加する必要はありません。

上の例が理解できなくても、心配しないでください。

知る必要があるのはこれらの定義をどう使用するかです。上の定義はATMEL AVR XMEGAヘッダファイルの一部で、以下の例で示されるように、“.” (ドット)構文規則で部署内のどのレジスタでもアクセスすることができます。

### 使用例

```
Unsigned char temp;
temp = PMIC.STATUS;           // tempに状態レジスタ読み込み
PMIC.CTRL |= PMIC_PMRRPE_bm; // 制御レジスタのPMRRPEビットを設定(1)
```

上で示された部署レジスタと部署アドレスを使用する主な利点はATMELのXMEGA系統の各種デバイス間と実際の周辺機能と無関係に(例えばUASRT1とUSART4の両方で動く)、ドライバの作成ができることです。

## 3. 概要

ここはこの練習に於ける課題の短い概要です。

この練習は4つの課題で様々な構想を示すのにI/Oポートを用いて、XMEGAの基礎を網羅します。この練習の目的はより効率的で簡素なコードのためにXMEGAヘッダファイルとXMEGAのいくつかの機能を利用する小さなコードの断片で始めさせることです。

### 課題1. 基本LED制御

この課題は可搬可能なコードを作成するためにXMEGAヘッダファイルから#defineと部署名を使用する方法とI/Oポートの操作方法を示します。

### 課題2. 一般的なドライバ

この課題は一般的なドライバコードを作るための周辺機能部署へのポインタ使用法と、スイッチ読み込みとLED出力の方法を示します。

### 課題3. 出力とプルアップ/ダウンの形態設定

この課題はレジスタ内のビット領域を効率的に変更するためにその群遮蔽と群形態設定値とでXMEGAヘッダファイルを使用する方法を示します。

### 課題4. 複数形態設定

この課題は同時に1つよりも多くのピンを形態設定するための複数形態設定レジスタの使用法を示します。

**注:** 基準フォルダ内の“board.h”が使用する評価基板を定義します。このファイルは練習コードで使用される全ての独自定義を含みます。

ご幸運を!

## 4. 課題1: 基本LED制御

あなたの開発基板上で点滅するLEDほど面白いものはありません! この課題はまさにそれで、他に何もありません。独自化が容易なコードを作るためにATMELのXMEGAヘッダファイルを利用する方法を示します。それはXMEGAのI/Oポート機能のいくつかも示します。

この課題の目的は以下です。

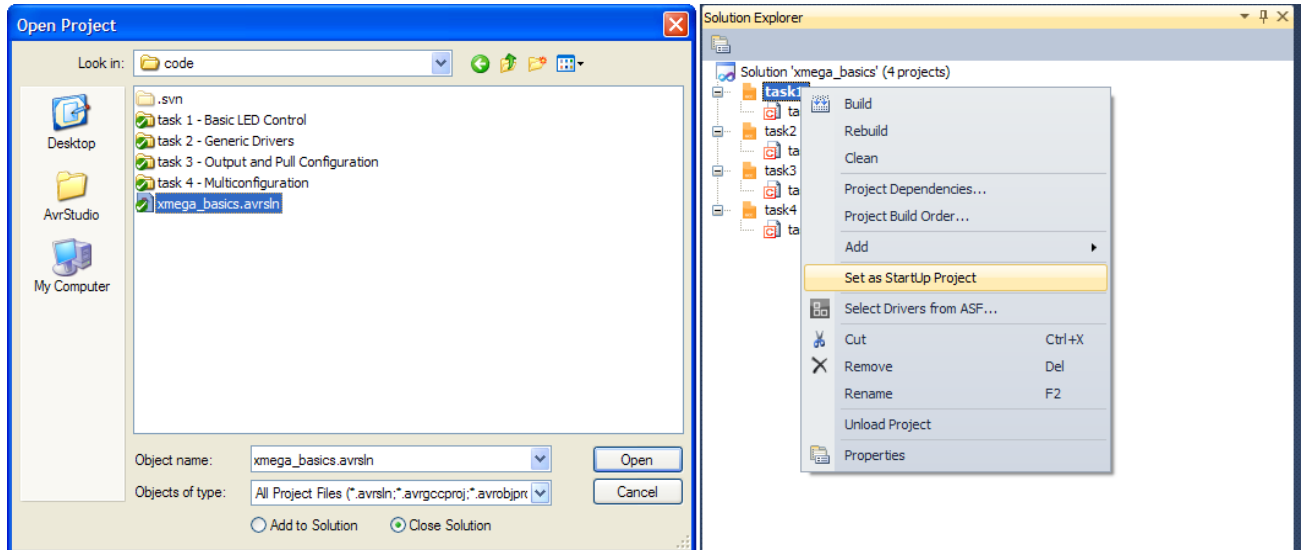
- 1つの#defineだけでLEDを使用するためにポートを変更する方法を知る。
- I/Oピンの方向を形態設定して出力値を設定する方法を知る。
- 1つの書き込みアクセスだけで既存のポート出力値を変更する方法を知る。



1. Atmel XMEGA-Basicsフォルダに位置付けてください。xmega\_basics.avrln解決ファイルを開き、始動(Startup)プロジェクトとしてそれを選択することによってTask1を活性に設定してください。そしてtask1.cを探してください(図3をご覧ください)。

すること

図3. AVR Studio 5でプロジェクトを開く



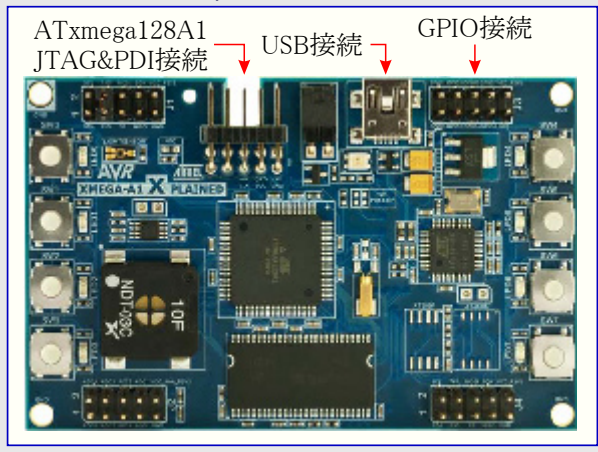
2. 図4は別のLEDポートを望む場合にLEDPORT定義を変更する容易な方法を示します。LEDPORT定義は"board.h"ファイルで見つかります。

図4. LEDポート変更

```
// file, it is enough to define t
// #defines for all registers for
// IO port to use for LED output
#define LEDPORT PORTE
volatile unsigned int counter;
```

3. JTAGICEmk3を評価基板のXMEGA JTAGヘッダに接続してください(図5をご覧ください)。そして対応するUSBケーブルで基板をコンピュータに接続してください。USBケーブルでJTAGICEmk3をコンピュータに接続してそれをONにしてください。

図5. XMEGA-A1 Xplain



- プロジェクトを構築(Ctrl-Shift-Bを押)してください(図6をご覧ください)。
- 開始鈕を押下することによってデバッグを始めてください(図7をご覧ください)。

図6. プロジェクト構築(Ctrl-Shift-B)

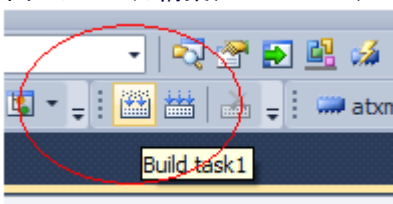
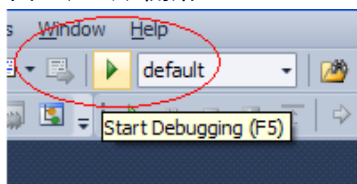


図7. デバッグ開始



- LEDポートのためにI/Oウィンドウを開いてください(図8をご覧ください)。I/Oピンを扱う、かなり多くのレジスタがあります。レジスタはコードをより簡潔にし、開発者に多くの任意選択を提供します。

図8. LEDポートでのレジスタ

Name	Address	Value	Bits
DIR	0x680	0xFF	11111111
DIRSET	0x681	0xFF	11111111
DIRCLR	0x682	0xFF	11111111
DIRTGL	0x683	0xFF	11111111
OUT	0x684	0xFF	11111111
OUTSET	0x685	0xFF	11111111
OUTCLR	0x686	0xFF	11111111
OUTTGL	0x687	0xFF	11111111
IN	0x688	0xFF	00000000
INTCTRL	0x689	0x00	00000000
INT0MASK	0x68A	0x00	00000000
INT1MASK	0x68B	0x00	00000000
INTFLAGS	0x68C	0x00	00000000
PIN0CTRL	0x690	0x00	00000000
PIN1CTRL	0x691	0x00	00000000
PIN2CTRL	0x692	0x00	00000000
PIN3CTRL	0x693	0x00	00000000
PIN4CTRL	0x694	0x00	00000000
PIN5CTRL	0x695	0x00	00000000
PIN6CTRL	0x696	0x00	00000000
PIN7CTRL	0x697	0x00	00000000

- コード全体を1行実行(F11を押)して目的対象基板とI/OウィンドウでLEDを観測してください(図9をご覧ください)。
- 1行実行(F11)してI/Oウィンドウを見ながら、各種ポートレジスタを理解してみてください。
- LEDを点滅するコードの最終部を見るためにコードを走行(F5を押)してください。

図9. 1行実行(F10)開始

```
volatile unsigned int counter;

int main( void )
{
    LEDPORT.DIR = 0xff; // Set all pin
    LEDPORT.OUT = 0xf0; // Set upper f
    LEDPORT.OUTSET = 0x0f; // Set lower f
    LEDPORT.OUTCLR = 0xf0; // Clr upper f
    LEDPORT.OUTTGL = 0xff; // Toggle all
    LEDPORT.OUTTGL = 0xff; // Toggle all
    LEDPORT.OUTTGL = 0xAA; // Toggle some
    LEDPORT.OUTTGL = 0xAA; // Toggle some
    LEDPORT.OUTTGL = 0xAA; // Toggle some
    LEDPORT.OUTTGL = 0xAA; // Toggle some

    // Now, just for fun, we toggle all I
```

## 5. 課題2: 一般的なドライバ

正しいレジスタを選ぶのに“switch/case”や“if else”宣言の必要なしにどのI/Oポート(またはどのA/D変換器、またはどのD/A変換器、またはどのタイマ/カウンタ)をもアクセスすることができる1つの関数をこれまでに望みましたか?。この課題はI/Oポート部署へのポインタを取るコードを作るのにATMELのXMEGAのI/O割り当ての配置を利用し、そして正しいレジスタをアクセスするのに一般的なコードを使用する方法を示します。この方法はI/Oポート、A/D変換器、D/A変換器、タイマ/カウンタなどに対する一般的なドライバの作成に使用することができます。

この課題の目的は以下です。

- ポインタ変数を作成してそれを何れかのI/Oポート部署へ位置付ける方法を知る。
- 関数のパラメータとして部署ポインタを使用する方法を知る。
- 部署ポインタを通して部署レジスタをアクセスする方法を知る。



すること

1. **Atmel XMEGA-Basics**フォルダ位置付けてください。 **xmega\_basics.avrsln**解決ファイルを開き、始動(Startup)プロジェクトとしてそれを選択することによって**Task2**を活性に設定してください。
2. LEDとスイッチに他のポートの使用を望む場合に**ledPort**と**switchPort**のポインタ割り当てが容易に変更できることを観察してください。
3. プロジェクトを構築してください(**F7/Ctrl-Shift-B**)。
4. デバッグを開始してください。
5. コード内を1行実行(**F11**を押)してLEDを観測してください。 **GetSwitches**関数の内側の時に各種スイッチを押してみてください。
6. 実時間でスイッチ状態をLEDに複写するコードを見るためにコードを走行(**F5**を押)してください。

図10. 関数の1行実行

```
// Prepare pointers to the peripheral port modules we want t
PORT_t * ledPort = &LEDPOR;
PORT_t * switchPortL = &SWITCHPORTL;
PORT_t * switchPortH = &SWITCHPORTH;

// Enable pullups for switches. MPCMASK will be explained in
PORTCFG.MPCMASK = SWITCHPORTL_MASK_gc;
switchPortL->PIN0CTRL = PORT_OPC_PULLUP_gc;

PORTCFG.MPCMASK = SWITCHPORTH_MASK_gc;
switchPortH->PIN6CTRL = PORT_OPC_PULLUP_gc;

// Now copy switch state to LEDs again and again and again..
// Because the switches are split between two ports on the X
// the value statement combines bits from each port and aran
while(1)
{
    value = GetSwitches( switchPortL, SWITCHPORTL_MASK_gc )
    SetLEDs( ledPort, value );
}
}
```

## 6. 課題3: 出力とプルアップ/ダウンの形態設定

ATMEL AVR XMEGAでは全ての部署の全てのビット領域に対する全ての形態設定任意選択が1つのヘッダファイル(例えば**iox128a1.h**)で名付けられた定数として利用可能です。この課題はヘッダファイルから群遮蔽と群形成値を使用する方法と、データシートでそれらを探す方法、そしてあなたのコードでそれらを効果的且つ簡潔にそれらを使用する方法を示します。また、この課題はXMEGAのI/Oポートの機能のいくつかの洞察を与えます。

この課題の目的は以下です。

- レジスタ内のビット領域を変更するために効率的且つ簡潔なコードを書く方法を知る。
- データシートとヘッダファイルから群遮蔽と群形成の値を探して使用する方法を知る。
- XMEGAのI/Oポートの出力とプルアップ/ダウンの各種設定について知る。



すること

1. **Atmel XMEGA-Basics**フォルダ位置付けてください。 **xmega\_basics.avrsln**解決ファイルを開き、始動(Startup)プロジェクトとしてそれを選択することによって**Task3**を活性に設定してください。
2. プロジェクトを構築、**F7/Ctrl-Shift-B**を押して、デバッグ作業を開始してください。
3. XMEGA A手引書を開いて**PINnCTRL**レジスタ形態設定を探し出してしてください。レジスタ説明項で出力とプルアップ/ダウン形態設定(OPC)値を考察してください。
4. 構築ツリーで依存性(Dependencies)を展開して**iox128a1.h**を開いてください(図11をご覧ください)。 **PORT\_OPC\_WIRED\_AND\_gc**のOPC形態設定が見つかりましたか?。(助言:“WIRED”で検索してください。)

5. task.cに於いて、次のこの行に位置付けて1行実行を行ってください。

```
SWITCHPORTL.PIN0CTRL = (SWITCHPORTL.PIN0CTRL & ~PORT_OPC_gm) | PORT_OPC_WIREDANDPULL_gc;
```

2.2節を参照して上のビット形成遮蔽を理解してみてください。(ワイヤードAND形態設定は次の課題で詳述されます。)

6. I/Oウィンドウを考察すると同時に更に1行実行を行ってください。手引書の形態設定レジスタ説明と比較してOPC形態設定を検証してください。

図11. iox128a1.hを開く

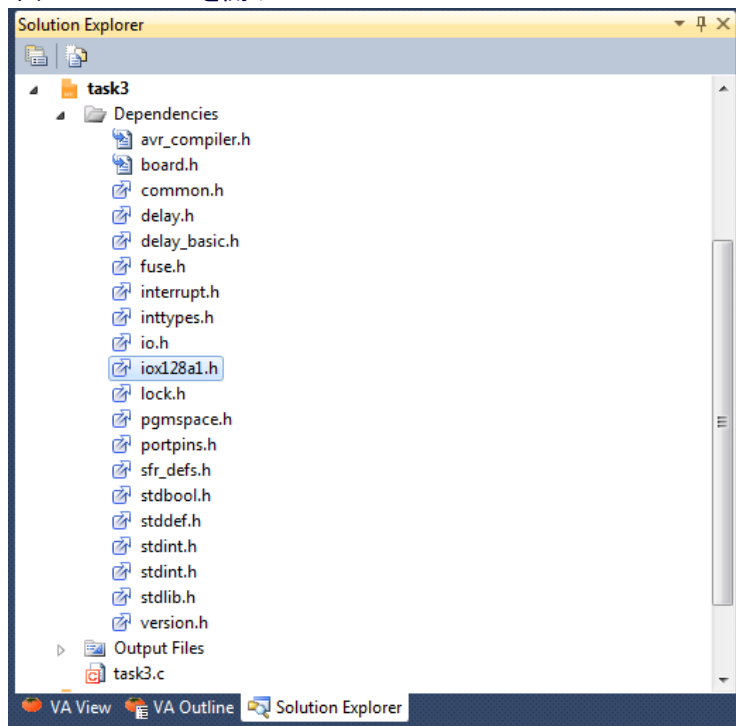
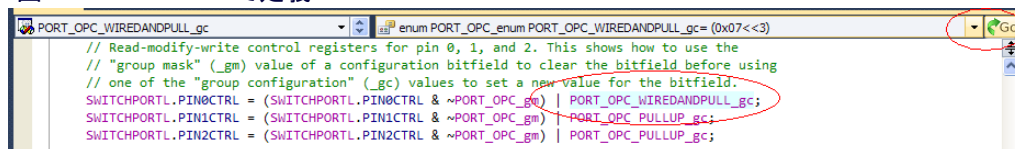


図12. AVR Studio 5で定義へ



## 7. 課題4: 複数形態設定

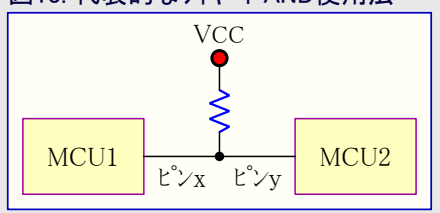
直前の課題で各入出力ピンに対して1つの形態設定レジスタがあることを学びました。同時に多数のピンの形態設定を望む場合はどうでしょうか。複数形態設定で、ポート内の多数のピンを同時に形態設定することができます。

各ピンに対して形態設定レジスタを持つことは、単一ポートを形態設定するために必要とする操作数が増加することを意味します。書き込み操作数は全てのピンに対して共通の全体的な複数ピン形態設定許可(MPCMASK)レジスタの導入によって減らされます。MPCMASKはピン形態設定レジスタに対してビット遮蔽を設定するのに使用することができます。MPCMASKでビットnを設定すると、PINnCTRLはピン形成遮蔽が追加されます。ポートピン形態設定レジスタのどれかへの次の書き込みの間に、遮蔽(値)によって設定される全てのポートピン形態設定レジスタは同じ値が書かれます。MPCMASKレジスタはピン形態設定レジスタが完了されるための書き込み操作後に自動的に解除されます。

この課題は出力プルアップ/ダウン形態設定(OPC)の1つ、ワイヤードANDプルアップの実演でもあります。下図は共に接続された2つのマイクロコントローラと外部プルアップを図解します。図の右の表に於いてデータ出力値(OUT)レジスタの値に応じて入出力ピン駆動部がどう反応するかが分かります。

この課題では1つのマイクロコントローラを持ち、故にジャンパで2つのピンを共に接続し、内部プルアップを使用します。

図13. 代表的なワイヤードAND使用法



ワイヤードAND時の入出力ポートの動きは次のように形態設定されます。

OUTレジスタ	入出力ピン出力
0	Low(0)引き込み
1	Hi-Z

この課題の目的は以下です。

- 複数ピン形態設定許可レジスタを使用する方法を知る。
- ATMEL XMEGAの入出力ポートの出力とプルアップ/ダウンについてもっと理解する。
- 指定のワイヤードAND例を理解する。



すること

1. Atmel XMEGA-Basicsフォルダ位置付けてください。xmega\_basics.avrsln解決ファイルを開き、始動(Startup)プロジェクトとしてそれを選択することによってTask4を活性に設定してください。
2. 評価基板の右下に配置されたTESTPORTヘッダのピン0とピン1を接続するためにジャンパを使用してください。
3. ATMELのAVR Studioでプロジェクトを構築してデバッグを開始してください。
4. 図14.で示されるようにwhile繰り返しの前に中断点を追加して走行するためにF5を押してください。中断点を配置するには中断点を欲するコード行にカーソルを置いてF9を押してください。

図14. while(1)繰り返し前に中断点追加

```
// same value will be written to all the port's pin confi
PORTCFG.MPCMASK = 0xFF;

// The WIREDANDPULL enables the internal pull-up.
// Also, when the port is configured as output (dir=1),
// setting a pin low will drive low,
// but setting a pin high will set the pin in tri-state
TESTPORT.PINCTRL = (TESTPORT.PINCTRL & ~PORT_OPC_gm) |

// The MPCMASK register is cleared automatically after th
// configuration registers is finished.

// Connect TESTPORT pin 0 and 1 with a jumper on the board

while(1)
{
    // Set pins 0 and 1 high
    TESTPORT.OUTSET = PIN0_bm;
    TESTPORT.OUTSET = PIN1_bm;
    // Now the pins are in tri-state
```

5. I/Oウィンドウを開いてTESTPORTレジスタ視野を見てください。1行実行(F10)する時にPINnCTRLピンに何が起きますか?。
6. ワイヤードAND設定が何を行うか見てみましょう。コードはピン0とピン1の両方をHighに設定し、INレジスタでその結果がどうかを調べてください。その後、ピン0をLowに設定し、そしてピン1に関してINレジスタでその結果を調べてください。
7. while繰り返しの1行実行してI/OウィンドウでTESTPORT.INレジスタを観測してください。ピン0がLowに設定された時にピン1のIN値で何が起きますか?。
8. 2つのピンを共に接続(それは短絡回路)して標準的な形態設定を持つ両ピンで1つのピンをHighに他方をLowに形態設定する場合に、ピン出力駆動部が違う方向に駆動し、ピン間に強い電流の流れを生じることにご注意してください。例示のようにワイヤードAND形態設定を用いると、これが避けられます。

図15. ピン0のLow駆動がピン1をLowに駆動

Name	Address	Value	Bits
DIR	0x660	0xFF	■ ■ ■ ■ ■ ■ ■ ■
DIRSET	0x661	0xFF	■ ■ ■ ■ ■ ■ ■ ■
DIRCLR	0x662	0xFF	■ ■ ■ ■ ■ ■ ■ ■
DIRTGL	0x663	0xFF	■ ■ ■ ■ ■ ■ ■ ■
OUT	0x664	0x02	□ □ □ □ □ □ □ □
OUTSET	0x665	0x02	□ □ □ □ □ □ □ □
OUTCLR	0x666	0x02	□ □ □ □ □ □ □ □
OUTTGL	0x667	0x02	□ □ □ □ □ □ □ □
IN	0x668	0x00	□ □ □ □ □ □ □ □
INTCTRL	0x669	0x00	□ □ □ □ □ □ □ □
INTOMASK	0x66A	0x00	□ □ □ □ □ □ □ □
INTIMASK	0x66B	0x00	□ □ □ □ □ □ □ □
INTFLAGS	0x66C	0x00	□ □ □ □ □ □ □ □
PIN0CTRL	0x670	0x38	□ □ ■ ■ □ □ □ □
PIN1CTRL	0x671	0x38	□ □ ■ ■ □ □ □ □
PIN2CTRL	0x672	0x38	□ □ ■ ■ □ □ □ □
PIN3CTRL	0x673	0x38	□ □ ■ ■ □ □ □ □
PIN4CTRL	0x674	0x38	□ □ ■ ■ □ □ □ □
PIN5CTRL	0x675	0x38	□ □ ■ ■ □ □ □ □
PIN6CTRL	0x676	0x38	□ □ ■ ■ □ □ □ □
PIN7CTRL	0x677	0x38	□ □ ■ ■ □ □ □ □

9. TESTPORTからジャンパを取り外してください。



## 8. 要約

これはあなたがこの課題の最中に学んだ主な特徴/機能のいくつかです。

- ATMELのXMEGA用のCコードの書き方
- 基本的なポート形態設定
- 標準的なドライバの作り方
- 出力とプルアップ/ダウンの形態設定
- ピンの複数形態設定
- ワイヤードAND

## 9. 資料

- XMEGAの手引書とデータシート
  - <http://www.atmel.com/xmega>
- ATMELのヘルプ ファイル付きAVR studio
  - <http://www.atmel.com/products/AVR/>
- WINAVR GCCコンパイラ
  - <http://winavr.sourceforge.net/>
- ATMEL用IAR Embedded Workbench<sup>®</sup>コンパイラ
  - <http://www.iar.com/>

## 10. ATMEL技術支援センター

ATMELは以下の利用可能な多数の支援チャネルを持ちます。

- ウェブ入り口 : <http://www.atmel.no/> 全てのATMELマイクロ コントローラ
- Eメール : [avr@atmel.com](mailto:avr@atmel.com) 全てのATMEL AVR製品
- Eメール : [avr32@atmel.com](mailto:avr32@atmel.com) 全ての32ビットAVR製品

以下のサービスへのアクセスを得るにはウェブ入り口で登録してください。

- 豊富なFAQデータベースへのアクセス
- 技術支援要請の容易な依頼
- あなたの過去の全支援要請の履歴
- ATMELマイクロ コントローラ時事通信の受信のための登録
- 利用可能な練習と練習材料についての情報取得



#### *Atmel Corporation*

2325 Orchard Parkway  
San Jose, CA 95131  
USA  
TEL (+1)(408) 441-0311  
FAX (+1)(408) 487-2600  
[www.atmel.com](http://www.atmel.com)

#### *Atmel Asia Limited*

Unit 01-5 & 16, 19F  
BEA Tower, Millennium City 5  
418 Kwun Tong Road  
Kwun Tong, Kowloon  
HONG KONG  
TEL (+852) 2245-6100  
FAX (+852) 2722-1369

#### *Atmel Munich GmbH*

Business Campus  
Parking 4  
D-85748 Garching b. Munich  
GERMANY  
TEL (+49) 89-31970-0  
FAX (+49) 89-3194621

#### *Atmel Japan*

141-0032 東京都品川区  
大崎1-6-4  
新大崎勸業ビル 16F  
アトメル ジャパン合同会社  
TEL (+81)(3)-6417-0300  
FAX (+81)(3)-6417-0370

#### © 2011 Atmel Corporation. 全権利予約済

ATMEL®、ATMELロゴとそれらの組み合わせ、それとXMEGA®、AVR Studio®、AVR®、AVR®ロゴとその他はATMEL Corporationの登録商標または商標またはその付属物です。他の用語と製品名は一般的に他の商標です。

**お断り:** 本資料内の情報はATMEL製品と関連して提供されています。本資料またはATMEL製品の販売と関連して承諾される何れの知的所有権も禁反言あるいはその逆によって明示的または暗示的に承諾されるものではありません。ATMELのウェブサイトに位置する販売の条件とATMELの定義での詳しい説明を除いて、商品性、特定目的に関する適合性、または適法性の暗黙保証に制限せず、ATMELはそれらを含むその製品に関連する暗示的、明示的または法令による如何なる保証も否認し、何ら責任がないと認識します。たとえATMELがそのような損害賠償の可能性を進言されたとしても、本資料を使用できない、または使用以外で発生する(情報の損失、事業中断、または利益と損失に関する制限なしの損害賠償を含み)直接、間接、必然、偶然、特別、または付随して起こる如何なる損害賠償に対しても決してATMELに責任がないでしょう。ATMELは本資料の内容の正確さまたは完全性に関して断言または保証を行わず、予告なしでいつでも製品内容と仕様の変更を行う権利を保留します。ATMELはここに含まれた情報を更新することに対してどんな公約も行いません。特に別の方法で提供されなければ、ATMEL製品は車載応用に対して適当ではなく、使用されるべきではありません。ATMEL製品は延命または生命維持を意図した応用での部品としての使用に対して意図、認定、または保証されません。

#### © HERO 2014.

本応用記述はATMELのAVR1512応用記述(doc8398.pdf Rev.8398A-07/11)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識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。