応用記述

# AT01095：ジョイスティック ゲーム制御器参照基準設計

## 8/16ビット Atmel マイクロ コントローラ

## 要点

- ジョイスティック ゲーム制御器Atmel® ATxmega32A4Uマイクロ コントローラ
- プログラムとデバッグ用インターフェース(PDI:Program and Debug Interface)経由の実装書き込み(ISP:In System Programming)とデバッグ
- アナログ入力
  - 2軸動作を持つ2つのアナログ ジョイスティック
- デジタルB入出力
  - 6つのデジタル押し釦

## 序説

この資料はAtmel ATxmega32A4Uマイクロ コントローラに基づくジョイスティック ゲーム制御器の実装を示します。この設計はマウス カーソルを制御してゲーム制御器のジョイスティック上の押し釦経由でパーソナル コンピュータ(PC)やラップトップPCからの左/右クリックを模倣します。

## 目次

# 1. 用語集

| | |
|---|---|
| ISp | 実装書き込み (In-System Programming) |
| Atmel Studio 6 | Atmel AVR®応用のための統合開発環境(IDE:Integrated Development Environment) |
| USB | 万能直列ﾊﾞｽ (Universal Serial Bus) |
| HID | 人間ｲﾝﾀｰﾌｪｰｽ装置 (Human Interface Device) |

# 2. 概要

この応用はHID装置、この場合は標準PCﾏｳｽの座標に対して得られた値を渡すAtmel ATxmega32A4Uへｱﾅﾛｸﾞ信号を提供する使用者を必要とします。ﾊｰﾄﾞｳｪｱ設計は以下の2つの独立した基板に分かれます。

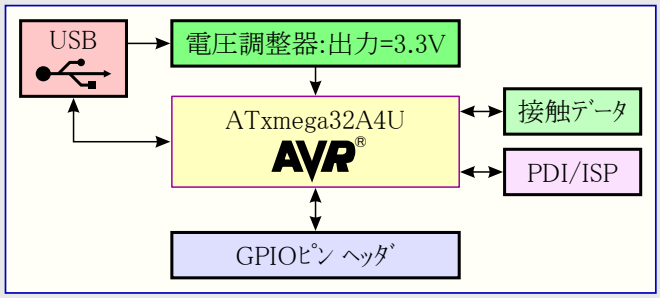
- ATxmega基礎基板 – USBｺﾈｸﾀ、3.3V調整器、適切なﾍｯﾀﾞ ﾋﾟﾝを持つAtmel ATxmegaﾃﾞﾊﾞｲｽを持つ組み立て済み基板
- ｼﾞｮｲｽﾃｨｯｸ子基板 – この基板はATxmega基礎基板の上に取り付けることを意図されました。この基板は押し釦ｽｲｯﾁを装備された2つのｱﾅﾛｸﾞ ｼﾞｮｲｽﾃｨｯｸを持ちます。加えて、XMEGAﾃﾞﾊﾞｲｽの標準入出力に接続された4つのﾀｸﾄ ｽｲｯﾁがあります。

# 3. ATxmega基礎基板

基礎基板の構成図は図3-1.で示されます。この基板自身は以下の構成部分から成ります。

- ATxmega32A4U – ｼﾞｮｲｽﾃｨｯｸ、押し釦、接触ﾊﾟｯﾄﾞのどれかかの使用者入力を処理します。
- ﾏｲｸﾛUSBｺﾈｸﾀ – ｹﾞｰﾑ制御器基板に電源(5V)を提供します。加えて、ｹﾞｰﾑ制御基板とPC間の接続媒体を提供します。
- 電圧調整器 – ATxmega32A4Uﾃﾞﾊﾞｲｽは1.6～3.6V間のVCC範囲で動作します。この調整器はUSBによって提供される5VをATxmega324Uﾃﾞﾊﾞｲｽ用の3.3Vに変換します。
- ISP/PDIﾍｯﾀﾞ – ﾃﾞﾊﾞｲｽ上のﾌｧｰﾑｯｳｪｱを更新するためのISPﾌﾟﾛｸﾞﾗﾐﾝｸﾞ ｲﾝﾀｰﾌｪｰｽ
- 6つの2×5ﾍｯﾀﾞ – これらのﾍｯﾀﾞ ﾋﾟﾝは他の子基板と通信するためのATxmegaﾃﾞﾊﾞｲｽでの入出力用です。加えて、これらは両基板が別の物に接続される時の支えを提供します。

図3-1. ATxmega基礎基板、構成図

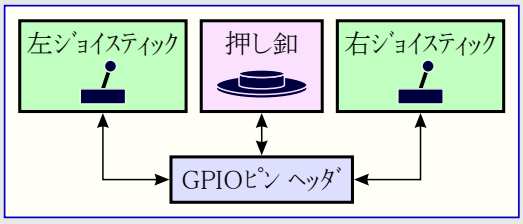



# 4. ｼﾞｮｲｽﾃｨｯｸ基板

ｼﾞｮｲｽﾃｨｯｸ子基板の構成図は図4-1.で示され、以下の構成部分から成ります。

- 2つのｱﾅﾛｸﾞ ｼﾞｮｲｽﾃｨｯｸ
  - 方向移動は単に各軸に対して1つで2つの可変抵抗器(10kΩ)です。
  - ｼﾞｮｲｽﾃｨｯｸはｼﾞｮｲｽﾃｨｯｸが押下される時に作動させる選択釦を持ちます。
- 4つのﾀｸﾄ押し釦
  - これらの押し釦はAtmel ATxmegaﾃﾞﾊﾞｲｽのﾎﾟｰﾄDとEに接続されるﾍｯﾀﾞに直接接続されます。

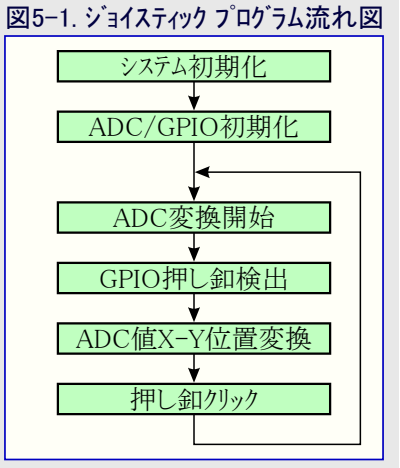
図3-2. ｼﾞｮｲｽﾃｨｯｸ基板、構成図




# 5. ｿﾌﾄｳｪｱ実装

## 5.1. ｼﾞｮｲｽﾃｨｯｸ ﾌﾟﾛｸﾞﾗﾑ流れ図

図5-1. ｼﾞｮｲｽﾃｨｯｸ ﾌﾟﾛｸﾞﾗﾑ流れ図



## 5.2. システム初期化

Atmel ATxmega32A4Uはそれから選ぶための多数のクロック元を持ちます。最速のADC変換時間と同時に参照基準基板上の空間を節約するために、コードはデバイス上の32MHz内部RC発振器を初期化して校正します。内部発振器は動作するのにどんな外部部品も必要としません。内部発振器の特性と精度に関するより多くの詳細についてはデバイスのデータシートを参照してください。

ATxmega32A4UのGPIOポートはジョイスティックと押し釦からやって来る信号を処理するために正しく構成設定されることが必要です。マイクロ コントローラの各ポート ピンは選択可能な駆動部と引っ張り設定を持つ入力または出力として構成設定することができます。この応用に関しては、2つのジョイスティックと4つのタクト押し釦から合計4つのアナログ信号と6つのデジタル信号を持ちます。タクト押し釦に関してはATxmega32A4Uの入出力ピンが許可された内部プルアップ付き入力として設定されます。

## 5.3. ADC初期化

この参照基準設計は2つのジョイスティックによって生成されるアナログ信号を処理するためにAtmel ATxmega32A4Uのデバイス上のADCを利用します。各ジョイスティックはジョイスティック位置の比較に於いて値を増減する2つの可変抵抗器(XとYの位置)を含みます。ADCはジョイスティックからの信号が正しく翻訳されることを保証するために以下の動作に構成設定されます。

ADCは次の様に構成設定されます。

- シングル エンド動作
- 符号なし12ビット分解能の結果
- ポートA入力チャネル
- 外部3.3V VREF
- 手動変換起動

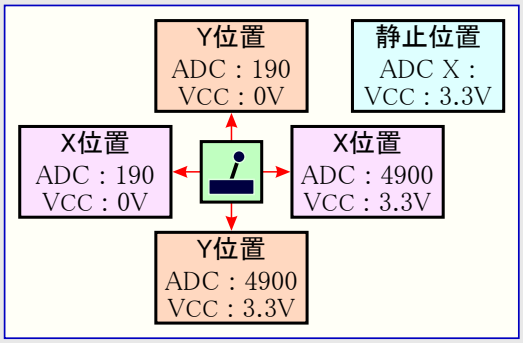
main.cのadc_hw_init関数下で完全なADC初期化を見つけることができます。使われる各入力チャネルが正しく初期化されなければなりません。

コードは次の様に定義された入力ポート チャネルを持ちます。

```
// アナログ入力ポート チャネル
#define JOYSTICK_INPUT_0   ADCCH_POS_PIN1      // PA1
#define JOYSTICK_INPUT_1   ADCCH_POS_PIN2      // PA2
#define JOYSTICK_INPUT_2   ADCCH_POS_PIN3      // PA3
#define JOYSTICK_INPUT_3   ADCCH_POS_PIN4      // PA4
```

アナログ値を受け取るための正しいADC構成設定に加え、ATxmega32A4Uはジョイスティック移動が正しく表されていいることを確実にすることができる定義済み閾値を持ちます。加えて、閾値を適用することにより、ジョイスティックが一方向で完全に加速されつつある時の高速カーソル移動を使用者に与えます。ジョイスティックにVCCとGNDを印加することにより、電圧水準と適切に変換された値が図5-2.で見ることができます。

図5-2. ジョイスティック閾値




## 5.4. ADC採取/変換開始

一旦ADCが正しく構成設定されてしまうと、その後はジョイスティックの動きを検出することができます。ADCはマウス カーソルの実時間位置で使用者に絶えず続く更新を提供するために自由走行として構成設定されます。

図5-2.で、ADC閾値はジョイスティックが全てで使われない(静止の)時の閾値と共に各位置軸で与えられます。ADCは入力ポート上の値を継続的に採取してジョイスティックがどれかの方向に移動される時にマウス カーソルに対する実時間移動更新を提供します。

押し釦実装はかなり簡単です。各押し釦はAtmel ATxmega32A4Uへの入出力ポートに接続されます。押し釦が活性にされると、入出力ピンは釦押下活性を合図するLowに引かれます。ジョイスティック上に置かれた2つの釦はマウスの標準的な左と右のクリックを模倣します。

ADC値とGPIO検出は標準マウス インターフェースを処理するUI(使用者インターフェース部)に渡されます。この部分はこの資料の後で詳細に検討されます。

# 6. 使用者インターフェース(UI)

## 6.1. HID記述子ファイル

一旦システム初期化が完了されてしますと、その後に使用者はゲーム制御器を開始してジョイスティック/接触パッドでカーソル移動のカーソル模倣を始めます。HID(Human Interface Device、人間インターフェース装置)の実装に関して以下のソース ファイルが詳細に検討されます。

- udi_d_mouse.c
- udi_hid_mouse_desc.c
- ui.c

PCにHID装置が接続されると、(マウス)装置は予め定義された塊または定期的な間隔でコンピュータに返す報告のデータを送ります。これらの報告はどの釦が押されたかや、ジョイスティックがどの位移動したかについての重要な情報を含みます。けれども、データ形式を理解するPCのために、デバイス(Atmel ATxmega)は"記述子ファイル"と呼ぶホストへの報告を"記述"しなければなりません。udi_hid_mouse.cに於いて図6-1.で記述子ファイルが作成されて初期化されます。

この標準記述子ファイルは装置がPCに接続されている限り定期的に送られます。これはマウスがそれの移動、釦、加速に対応するそれを送るデータの枠組みです。

図6-1. 記述子ファイル実装

```
udi_hid_mouse.c | udi_hid_mouse_desc.c | ui.c | udi_hid.c | conf_board.h | QTouch Analyzer
udi_hid_mouse_report_desc.B
UDC_DESC_STORAGE udi_hid_mouse_report_desc_t udi_hid_mouse_report_desc = {
  {
        0x05, 0x01, /* Usage Page (Generic Desktop),       */
        0x09, 0x02, /* Usage (Mouse),                     */
        0xA1, 0x01, /*  Collection (Application),          */
        0x09, 0x01, /*   Usage (Pointer),                 */
        0xA1, 0x00, /*  Collection (Physical),            */
        0x05, 0x09, /*    Usage Page (Buttons),           */
        0x19, 0x01, /*    Usage Minimum (01),             */
        0x29, 0x03, /*    Usage Maximum (03),             */
        0x15, 0x00, /*    Logical Minimum (0),            */
        0x25, 0x01, /*    Logical Maximum (1),            */
        0x75, 0x01, /*    Report Size (1),                */
        0x95, 0x03, /*    Report Count (3),               */
        0x81, 0x02, /*    Input (Data, Variable, Absolute) */
        0x75, 0x05, /*    Report Size (5),                */
        0x95, 0x01, /*    Report Count (1),               */
        0x81, 0x01, /*    Input (Constant),               */
        0x05, 0x01, /*    Usage Page (Generic Desktop),   */
        0x09, 0x30, /*    Usage (X),                      */
        0x09, 0x31, /*    Usage (Y),                      */
        0x09, 0x38, /*    Usage (Scroll),                 */
        0x15, 0x81, /*    Logical Minimum (-127),         */
        0x25, 0x7F, /*    Logical Maximum (127),          */
        0x75, 0x08, /*    Report Size (8),                */
        0x95, 0x03, /*    Report Count (3),               */
        0x81, 0x06, /*    Input (Data, Variable, Relative) */
        0xC0,   /*  End Collection,                       */
        0xC0,   /* End Collection                         */
  }
100 %
```

## 6.2. マウス操作

標準マウスは位置、左マウス釦、右マウス釦の3つの主な出力を持ちます。ui.cファイルはこれらの出力がどう定義されて実装されるのかを記述します。

カーソルの位置については標準デカルト(直交)座標が用いられます。ui.cファイルはカーソル位置に対するADC値を正しく表現するのに使われる関数を含みます。図6-2.は完全な関数宣言を示します。

図6-2. カーソル位置定義

```
static bool udi_hid_mouse_move(int8_t pos, uint8_t index_report)
{
    int16_t s16_newpos;

    irqflags_t flags = cpu_irq_save();

    // Add position in HID mouse report
    s16_newpos = (int8_t) udi_hid_mouse_report[index_report];
    s16_newpos += pos;
    if ((-127 > s16_newpos) || (127 < s16_newpos)) {
        cpu_irq_restore(flags);
        return false;   // Overflow of report
```

この関数は2つの入力を取って新しい位置が送られた時に真(true)を返します。最初の入力のposはADC採取を通して得られた実際の位置です。2つ目の入力のindex_reportはどの軸の移動が更新されるかを決めるためです。"1"の値はX軸に沿った移動を参照し、一方"2"の値はY軸に沿った移動を参照します。一旦位置と軸が設定されてしまうと、更新された座標が記述子形式でホスト(PC)へ送られます。

先に言及したように、装置は"完全移動"が発行されつつある時や、"微小移動"が望まれた場合を検出する閾値が設定されています。次の図6-3.は左アナログ ジョイスティック(PA4)の閾値検査と対応する移動を示します。

図6-3. 閾値検査

```
else if(joystick_input_3 <JOYSTICK_LOWER_VALUE){

        if (joystick_input_3 < JOYSTICK_MIN_MOVE) {
            udi_hid_mouse_moveY(joystick_input_3); // full move up
        }

        else
            udi_hid_mouse_moveY(((joystick_input_3-JOYSTICK_MIDDLE_VALUE)>>MOUSE_OFFSET));
```

ui.cに於いて、考慮に入れなければならない閾値を確か定義しています。最初に、ジョイスティックが移動/押下されつつない時にADCは未だこの”中心”位置に対応する値を艇庫湯しています。joystick_lower_valueの値は”中心位置”の中央範囲設定です。joystick_lower_valueとjoystick_higher_value間のどの値もカーソルの移動に影響を及ぼしません。どれかの値が下位側中央値よりも低いと、変更された位置が使用者によって求められます。joystick_min_moveの閾値は使用者が新しく得られた値の位置でHID移動関数を呼ぶ”完全移動”をしたいと思うかを決めます。けれども、下側閾値が満足されない場合、”微小移動”が要求されつつあり、HID移動関数にそれを渡す前に異なる位置が計算されます。この実装は全ての軸移動に対してほぼ同じです。考慮に入れるのが必要な唯一の値は閾値で、それらがどの軸(XまたはY)にどう関連するかです。

同様に、釦の実装は同様の方法で動作します。標準マウスは左と右の両方の釦クリック機能と共に出荷されて来ます。ui.cでの釦関数は図6-4.で定義されます。

図6-4. マウス釦実装

```
static bool udi_hid_mouse_btn(bool b_state, uint8_t btn)
{
    // Modify buttons report
    if (HID_MOUSE_BTN_DOWN == b_state)
        udi_hid_mouse_report[0] |= btn;
    else
        udi_hid_mouse_report[0] &= ~btn;
    // Use mouse move routine
    return udi_hid_mouse_move(0, 1);
}
```

この関数に関してはb_stateとbtnの2つの関数があります。1つ目の入力はマウス釦が実際に押されているかどうかを調べます。2つ目の入力はどの釦が許可されているかを決めます(0x01=左釦、0x02=右釦)。このプロジェクトに対しては左右の釦として各々PE3とPE4が割り当てられます。けれども、これは使用者が好むかもしれないどれにでも再割り当てすることができます。

現在、コンピュータのマウスは応用で違う様に使うことができる更にいくつかの釦/機能を持ちます。スクロールのような機能は既にこの記述子ファイルの適切な位置で実装することができます。マウスのスクロール機能用関数はudi_hid_mouse_cソース ファイルで見つけることができます。同様に、スクロール関数は図6-3.で記述される同じマウス移動を呼びます。

## 6.3. 定義と原型

表6-1.はゲーム制御器応用で使われる重要な関数原型を示します。表6-2.は閾値水準を示します。

注: 以下の閾値は各システムに依存し、それによって調節されるべきです。

表6-1. 関数定義

| 原型 | 使用目的 |
|---|---|
| udi_hid_mouse_move(int8_t pos, uint8_t index_report) | カーソル位置更新 |
| udi_hid_mouse_btn(bool b_state, uint8_t btn) | マウス釦検出 |
| sysclk_init(void) | CLK初期化 |
| adc_hw_init(int ch,int pin) | ADC初期化 |
| board_init(void) | GPIO初期化 |

表6-2. 閾値

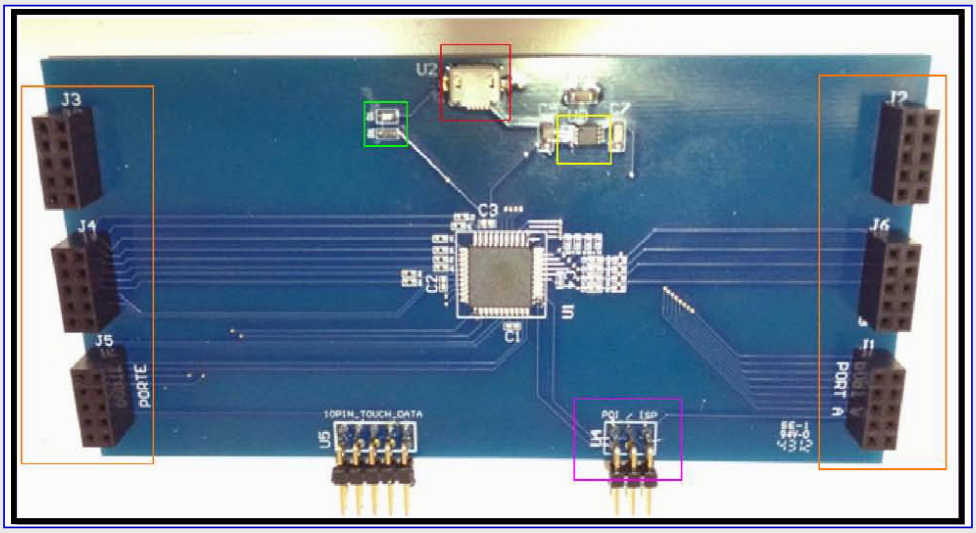
| パラメータ定義 | | | 使用目的 |
|---|---|---|---|
| #define | JOYSTICK_MIDDLE_VALUE | 2900 | 静止(REST)位置用閾値 |
| #define | JOYSTICK_LOWER_VALUE | 2450 | 静止(REST)位置用閾値 |
| #define | JOYSTICK_MAX_MOVE | 4000 | 完全移動用閾値 |
| #define | JOYSTICK_MIN_MOVE | 290 | 完全移動用閾値 |

# 7. ハードウェア設計

## 7.1. Atmel ATxmega基礎基板

図7-1.はATxmega基礎基板を示します。

図7-1. ATxmega基礎基板

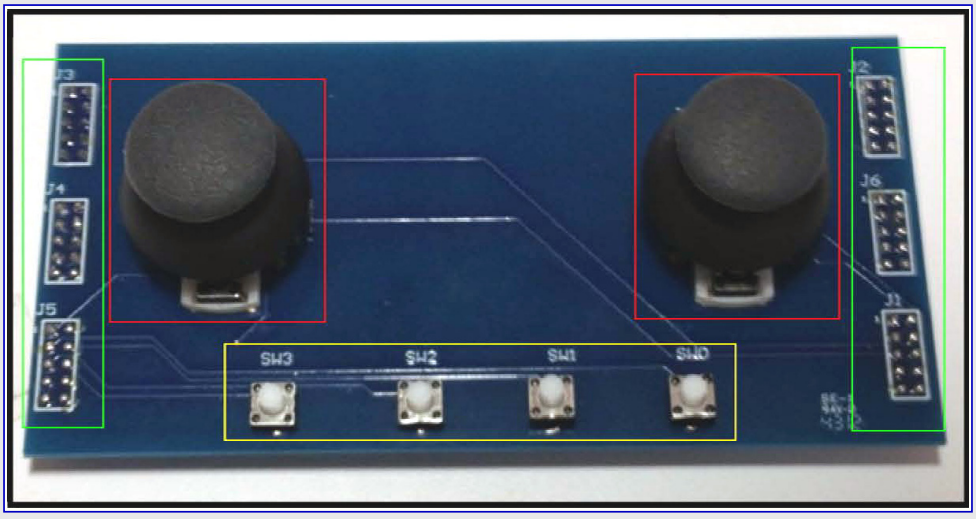



- **マイクロUSB** (U2/赤)
  これはホスト(PC)とAtmel ATxmega32A4Uデバイス間の中間接続です。加えて、これは基礎基板に5VでVCC供給電圧を提供します。
- **電圧調整器** (U3/黄)
  この調整器はデバイスの動作電圧が1.8～3.3Vのため、ATxmega用にUSBコネクタによって提供され電圧を落とすことが必要とされます。
- **電源LED** (D1/緑)
  基板へのVCCが正しく提供されたことを確かめる電源LED
- **PDI/ISP** (U4/紫)
  デバイスのファームウェアを更新するためのATxmega32A4U用プログラミング ヘッダ。実装中にICをプログラミングするのにAtmel JTAGICE3のような書き込みツールを使うことができます。
- **I/Oヘッダ** (J1～6/橙)
  これらの基板ヘッダは2つの目的に役立ちます。1つ目は基礎基板と子基板が結合される時に安定性を加えるために基礎基板に対して子基板を支えることで、2つ目はジョイスティックからATxmega32A4Uへの混合信号出力を正しく配線することです。

## 7.2. ジョイスティック子基板

図7-2.はジョイスティック子基板を示します。

図7-2. ジョイスティック子基板

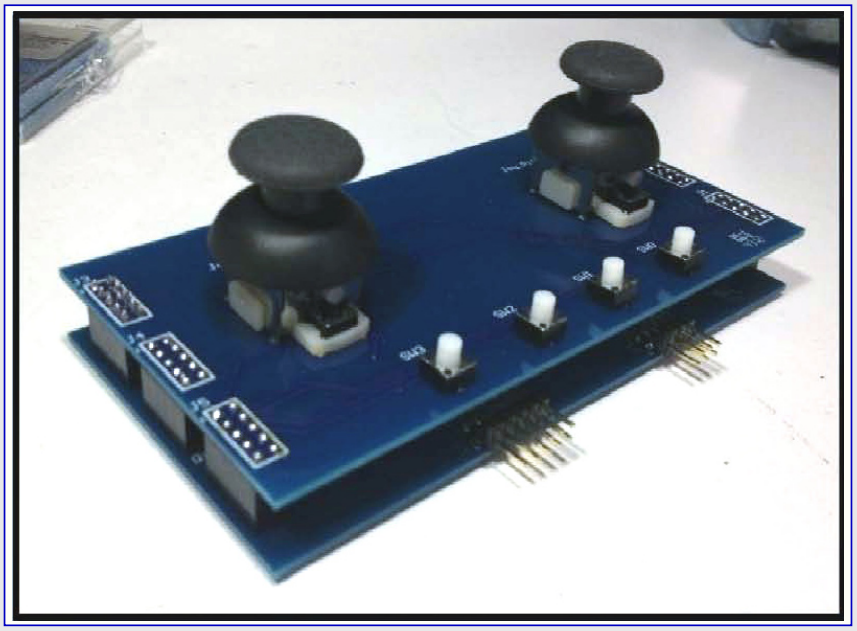


- **ｱﾅﾛｸﾞ ｼﾞｮｲｽﾃｨｯｸ** (Joy_LeftとJoy_Right/赤)
  押し釦ｽｲｯﾁを持つ2軸ｼﾞｮｲｽﾃｨｯｸ
- **押し釦** (SW0～3/黄)
  ﾀｸﾄ押し釦ｽｲｯﾁ
- **I/Oﾍｯﾀﾞ** (J1～6/緑)
  基礎基板と同様。これらのﾍｯﾀﾞは基礎基板とﾃﾞﾊﾞｲｽ(ATxmega32A4U)への混合信号の配線を提供し、基板が別の基板に接続される時の安定性を提供します。

### 7.3. 基礎基板と子基板の配置

図7-3.では基礎基板と子基板が単一単位部として示されます。

図7-3. 基礎基板と子基板



## 8. 参考資料


[1]. Atmel Studio 6 : http://www.atmel.com/Microsite/atmel_studio6/default.aspx

[2]. USB HID報告記述子ﾁｭｰﾄﾘｱﾙ : http://www.frank-zhao.com/cache/hid_tutorial_1.php


## 9. 改訂履歴

| 文書改訂 | 日付 | 注釈 |
|---|---|---|
| 42059A | 2013年1月 | 初版文書公開 |

Atmel | Enabling Unlimited Possibilities®


*Atmel Corporation*
1600 Technology Drive
San Jose, CA 95110
USA
TEL (+1)(408) 441-0311
FAX (+1)(408) 487-2600
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 G.K.*
141-0032 東京都品川区
大崎1-6-4
新大崎勧業ﾋﾞﾙ 16F
ｱﾄﾒﾙ ｼﾞｬﾊﾟﾝ合同会社
TEL (+81)(3)-6417-0300
FAX (+81)(3)-6417-0370




---



本応用記述はAtmelのAT01095応用記述(Rev.42059A-01/2013)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。