AVR1016 :Xplain QTouch AVR訓練

事前必要条件

- 知識水準: 中
- PC基盤: Windows® 2000,XP,Vista
- ﾊｰﾄﾞｳｪｱ必要条件:
  - JTAGICEmk Ⅱ
  - Xplainｷｯﾄ
  - ATMEL® QTouch® Xplain基板
- ｿﾌﾄｳｪｱ必要条件:
  - ATMEL® AVR Studio® 4 (最終版)
  - (ATMEL QTouch Studio, 任意)
  - WinAVRの最終版
  - ATMEL QTouch® ﾗｲﾌﾞﾗﾘ (最終版)
- 消費時間: ～3時間

序説

この訓練の目的はATMEL QTouchﾗｲﾌﾞﾗﾘ4.0とあなた自身の応用にそれを使用する方法に慣れることです。

この手順はいくつかの課題から成ります。最初の課題は全ての物を接続する方法を示してﾊｰﾄﾞｳｪｱが正しく動くことを確実にします。2つ目の課題は簡単なATMEL QTouch釦のﾌﾟﾛｼﾞｪｸﾄを構成設定して形態設定する方法を説明します。次の課題はもっと高度な感知器の形態設定方法と、QTouch Studioのﾗｲﾌﾞ ﾃﾞﾊﾞｯｸﾞ機能の利点を取り入れるためにﾃﾞﾊﾞｯｸﾞ ｲﾝﾀｰﾌｪｰｽを使用する方法を示します。

ATMEL AVR Studio、WinAVRの基本やﾂｰﾙの使い方は網羅されません。

資料概要

手順は多数の異なる指示に分けられます、各指示は理解を単純化するため、更に個別項目に分割されます。

この資料を通していくつかの特別なｱｲｺﾝを見つけるでしょう。これらのｱｲｺﾝは指示の異なる項目を識別して複雑さを容易にするために用いられます。

情報
特定の話題について流れに関する情報を提供

助言
有用な助言と技術を強調します。

行うべきこと
完了されるべき物を強調します。

結果
課題段階の予測される結果を強調します。

警告
重要な情報を示します。

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. 8304A-04/10, 8304AJ1-03/14

目次

1. 必要条件

1.1. ｿﾌﾄｳｪｱ

以下の最終版のｲﾝｽﾄｰﾙを確実にしてください。

- AVR Studio
- ATMEL QTouch Studio 4.0.1またはより新しい版(任意)
- WinAVR GCC
- ATMEL QTouchﾗｲﾌﾞﾗﾘ

1.2. ﾊｰﾄﾞｳｪｱ

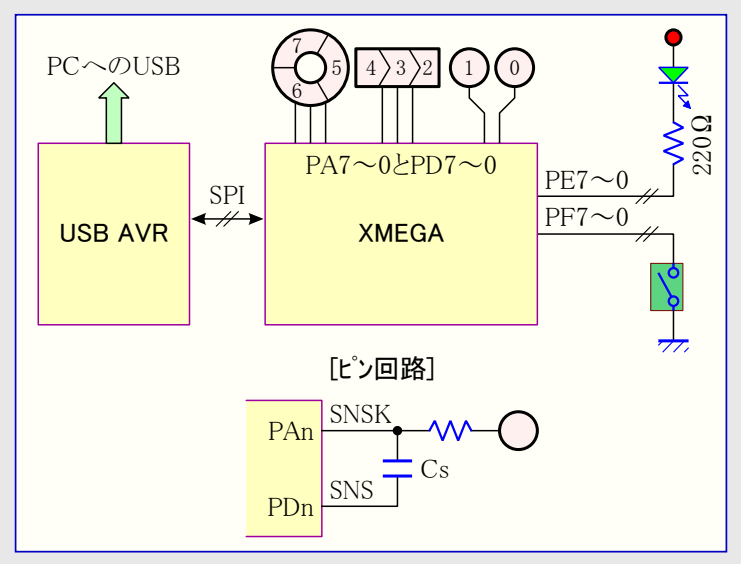
この実施に於いて以下を使用します。

- XplainｷｯﾄとUSBｹｰﾌﾞﾙ
- Xplain QTouch上乗せｶｰﾄﾞ
- JTAGICEmk IIとUSBｹｰﾌﾞﾙ

1.3. Xplain基板

XplainｷｯﾄはATxmega128Aを試験する簡単な方法を設計者に許すために設計された小さく単純な訓練ｷｯﾄです。

1.3.1. Xplain基板回路



2. 課題

2.1. 課題1: ツール接続と試験ﾌﾟﾛｼﾞｪｸﾄ走行

2.1.1. 序説

この課題は全てのツールが接続されて意図するように動くのを確実にするように設計されています。この課題に関してｺｰﾄﾞを書くことは全く必要とされません。

2.1.2. 作業

ｿﾌﾄｳｪｱの準備

1. この練習に必要とされるｿﾌﾄｳｪｱをｲﾝｽﾄｰﾙしてください。
   a. ATMEL AVR Studio (最終版をもっていなければ)
   b. ATMEL QTouch Studio (任意)
   c. ATMEL QTouchﾗｲﾌﾞﾗﾘ
   d. WinAVR
2. この訓練資料に付随する書庫ﾌｧｲﾙをPCのﾌｫﾙﾀﾞに解凍してください。

Xplain基板のﾌﾟﾛｸﾞﾗﾐﾝｸﾞ

1. ﾌﾞﾘｯｼﾞ ﾁｯﾌﾟ ﾌｧｰﾑｳｪｱはXplain基板上のAT90USB1287ﾃﾞﾊﾞｲｽにﾌﾟﾛｸﾞﾗﾐﾝｸﾞされる必要があります。これは2つの方法で行うことができます。
   a. JTAGICEmk Ⅱ 書き込み器でAT90USB1287をﾌﾟﾛｸﾞﾗﾐﾝｸﾞしてください。Xplain基板のJTAG USBと記されたﾎﾟｰﾄに書き込み器を接続してください。Firmwareﾌｫﾙﾀﾞ内のUSB_Bride_V3.3_646.a90ﾌｧｲﾙを探し出して、このﾌｧｲﾙをUSB装置に書いてください。**警告**: JTAGを用いてUSBﾌﾞﾘｯｼﾞ ﾌｧｰﾑｳｪｱをﾌﾟﾛｸﾞﾗﾐﾝｸﾞした場合、ﾌﾞｰﾄﾛｰﾀﾞを無効にします。元のﾌﾟﾛｸﾞﾗﾑのﾊﾞｯｸｱｯﾌﾟを作成することが勧告されます。
   b. ATMEL Flipｿﾌﾄｳｪｱとﾌﾞｰﾄﾛｰﾀﾞを用いてAT90USB1287をﾌﾟﾛｸﾞﾗﾐﾝｸﾞしてください。既定で存在するﾌﾞｰﾄﾛｰﾀﾞをUSB装置が持つなら、書き込み器を使用することなくﾌﾟﾛｸﾞﾐﾝｸﾞすることができます。代わりにATMELのｳｪﾌﾞｻｲﾄから無料でﾀﾞｳﾝﾛｰﾄﾞすることができるATMEL Flipを使用することができます。Flipを開始する前に、JTAG USBﾍｯﾀﾞの1と2のﾋﾟﾝを短絡することに注意してください。これはﾌﾞｰﾄﾛｰﾀﾞを許可します。USB_Bride_V3.3_646.a90ﾌｧｲﾙを探し出して、このﾌｧｲﾙをUSB装置へ書くのにFlipを用いてください。これはﾌﾞｰﾄﾛｰﾀﾞを保護します。
2. 今やXplain基板上のATMEL® AVR® XMEGAﾃﾞﾊﾞｲｽは例応用ﾌｧｰﾑｳｪｱでﾌﾟﾛｸﾞﾗﾐﾝｸﾞされることを必要とします。JTAG&PDI XMEGAﾎﾟｰﾄ経由でATxmega128A1に書き込み器を接続し、その後にExample.hexﾌｧｲﾙでﾌﾟﾛｸﾞﾗﾐﾝｸﾞしてください。この段階は書き込み器を必要とし、ﾌﾞｰﾄﾛｰﾀﾞで行えないことに注意してください。

Xplain QTouch基板とPCの接続

1. Xplain QTouch基板をXplainに接続してください。向きが正しいことを確実にし、その場合に両基板の印刷は同じ向きになるべきです。Xplain QTouch基板上部の裏側のﾍｯﾀﾞはAVR XMEGAﾃﾞﾊﾞｲｽに接続されるJTAGﾍｯﾀﾞです。
2. USBｲﾝﾀｰﾌｪｰｽを接続することによって基板に通電し、QTouch Studioを開始してください。

3. 釦、摺動子、輪の押下を試み、Xplain基板上の光が正しく応答することを確認してください。

4. QTouch Studioを開始してﾃﾞﾊﾞｲｽからのﾗｲﾌﾞﾃﾞﾊﾞｯｸﾞﾃﾞｰﾀを報告する画像ｲﾝﾀｰﾌｪｰｽを確認してください。

5. ﾃﾞｰﾀが更新されるなら、ｷｯﾄは動いており、次の作業を続けることができます。

2.2. 課題2: 簡単な釦

2.2.1. 序説

この課題ではQTouch釦をより真直で眺めて、無からﾌﾟﾛｼﾞｪｸﾄを形態設定します。(task2ﾌｫﾙﾀﾞ内の)Task2ﾌﾟﾛｼﾞｪｸﾄを開くことで始めます。このﾌﾟﾛｼﾞｪｸﾄは単にｼｽﾃﾑ周波数を設定して空の無限繰り返しに運ぶ非常に簡単なﾌﾟﾛｸﾞﾗﾑを含みます。このﾌﾟﾛｼﾞｪｸﾄの何処にもQTouch参照基準がありませんが、Xplainの光を点滅するのに2つの接触ｷｰを用いるように構成設定を行います。

QTouchﾗｲﾌﾞﾗﾘ ﾌｧｲﾙが存在しているかを検査

この作業を開始する前に、ｲﾝｽﾄｰﾙしたQTouchﾗｲﾌﾞﾗﾘから以下のﾌｧｲﾙをﾌﾟﾛｼﾞｪｸﾄ ﾌｫﾙﾀﾞへ複写してください。

libavrxmega7g1-8qt-k-0rs.a
touch_api.h
touch_qt_config.h
qt_asm_xmega.s

2.2.2. ｺｰﾄﾞ構成設定

ﾌﾟﾛｼﾞｪｸﾄ ﾌｧｲﾙ読み込み

最初に. AVR Studioを開始して課題2と連携するﾌﾟﾛｼﾞｪｸﾄを読み込んでください。ｺｰﾄﾞを考察するのに少し時間をかけて単純な特徴に慣れてください。未だ接触感知に直接関連するｺｰﾄﾞがないことに注意してください。

QTouchﾗｲﾌﾞﾗﾘのﾍｯﾀﾞ、ﾊﾞｲﾅﾘ、ｿｰｽのｲﾝｸﾙｰﾄﾞ

1. 接触ｷｰを使用するため、最初にｲﾝｸﾙｰﾙ ﾌｧｲﾙの一覧に #include ″touch_api.h″ の追加が必要です。これはQTouchﾗｲﾌﾞﾗﾘAPIを利用可能にします。
2. 今やAPIを持ち、そして予めｺﾝﾊﾟｲﾙされたQTouchｺｰﾄﾞを含むﾗｲﾌﾞﾗﾘ ﾌｧｲﾙへの参照が必要です。Project Optinsを開き、Libraries下でlibavrxmega7g!-8qt-k-0rs.aﾗｲﾌﾞﾗﾘ ﾌｧｲﾙを選択してAdd Librayをｸﾘｯｸしてください。
3. 最後にqt_asm_xmega.sﾌｧｲﾙをﾌﾟﾛｼﾞｪｸﾄに追加することが必要でしょう。これはﾌﾟﾛｼﾞｪｸﾄと共にｺﾝﾊﾟｲﾙされる必要がある最適化ﾙｰﾁﾝを含む特別なｱｾﾝﾌﾞﾘ言語ｿｰｽです。AVR StudioのAVR GCCﾀﾌﾞに於いて、ﾌﾟﾛｼﾞｪｸﾄを右ｸﾘｯｸしてAdd Existing Fileを選び、そしてｱｾﾝﾌﾞﾘ言語ｿｰｽ ﾌｧｲﾙを選択してください。代わりに、ﾌﾟﾛｼﾞｪｸﾄ名上にｿｰｽ ﾌｧｲﾙをﾄﾞﾗｯｸﾞ&ﾄﾞﾛｯﾌﾟすることもできます。

QTouchﾗｲﾌﾞﾗﾘ形態設定

QTouchﾗｲﾌﾞﾗﾘは現在の機能のために定義された少しのﾌﾟﾘﾌﾟﾛｾｯｻ定義が必要です。これらは全てのﾌｧｲﾙに対して設定が必要で、それ故、Project Optionsﾒﾆｭｰのｺﾝﾊﾟｲﾗ任意選択にそれらを入力する良い場所があります。簡単化のために正しい設定が構築形態設定として既にﾌﾟﾛｼﾞｪｸﾄ内に入力されています。それらが正しく設定されているのを確認するため、Project Optionsｳｨﾝﾄﾞｳを開き、Generalﾀﾌﾞでactive Configurationが″libavrxmega7g!-8qt-k-0rs″に設定されていることを確認してください。

そしてCustom Optionsﾀﾌﾞに切り換えて、全体ｺﾝﾊﾟｲﾗ ﾌﾗｸﾞを眺めてください。以下のﾌﾗｸﾞが存在すべきです。

-D_QTOUCH_
-DQT_NUM_CHANNELS=8
-DSNSK1=A
-DSNS1=D
-DQT_DELAY_CYCLES=1

これはいくつかのことをﾗｲﾌﾞﾗﾘに告げます。ここでは(QMatrix®と対立する)QTouch計測技術を使用するつもりです。そして最大8ﾁｬﾈﾙを持つつもりです。それらは1遅延周期だけで、測定はﾎﾟｰﾄAとDで行われるべきです。これらのﾊﾟﾗﾒｰﾀのいくつかの正確な意味は今それ程重要ではありませんが、更なる情報を望むなら、ｵﾝﾗｲﾝのQTouch使用者の手引きをご覧ください。

これらの設定はQTouchﾗｲﾌﾞﾗﾘを使用する、どのﾌﾟﾛｼﾞｪｸﾄにも入力が必要です。

QTouchライブラリ形態設定をもっと追加

次にいくつかの走行時全体QTouchパラメータの構成設定が必要です。これが行われるべき関数init_qt_globals()が提供されます。この関数を見つけてそれに以下のコードを追加してください。

```
qt_config_data.qt_di                = DEF_QT_DI;
qt_config_data.qt_neg_drift_rate    = DEF_QT_NEG_DRIFT_RATE;
qt_config_data.qt_pos_drift_rate    = DEF_QT_POS_DRIFT_RATE;
qt_config_data.qt_max_on_duration   = DEF_QT_MAX_ON_DURATION;
qt_config_data.qt_drift_hold_time   = DEF_QT_DRIFT_HOLD_TIME;
qt_config_data.qt_recal_threshold   = DEF_QT_RECAL_THRESHOLD;
```

これらのパラメータが正確に何を指定するのかは今重要ではなく、後で更に詳細を説明します。勿論、ここの助言は上の塊からあなたのプロジェクトへコードを複写して張り付けることです。

今や最初の釦を形態設定する時です。

釦を形態設定するのに以下のAPI呼び出しを使用します。

```
qt_enable_key( CHANNEL_0, AKS_GROUP_1, 10u, HYST_6_25 );
```

init_system呼び出し後にこれを追加してください。ここではこれを"感知器0許可"と言います。これはチャネル0がキーとして扱われるべきであることをQTouchに告げます。少し後で最後の3つのパラメータを考察します。

ライブラリ初期化

この初期形態設定後、QTouchライブラリの初期化が必要です。これは単にqt_init_sensing()を呼び出すことによって行われます。ラベルの"initialize touch sensing"下にこの関数呼び出しを追加してください。

時間を格納するための何処か

今や測定の取得開始の準備が殆ど整っています。単一測定はqt_measure_sensors(uint16_t current_time_ms)呼び出しによって行われます。しかしお分かりのように、それは引数を求めます。この引数は測定された値に於いて時間上での偏移に関する補償を許すための時間の経緯を保つのに使用されます。この偏移に関わらないのなら、この引数を0として渡すことができます。

この応用に関して、時間の経緯を保つのに簡単な計数器を追加するつもりで、故に静的変数下に次の変数を宣言してください。

```
static volatile uint16_t current_time_ms_touch = 0u;
```

今や接触測定を実行する準備が整っています。

主繰り返しの内側の"measure touch sensors"下に以下の呼び出しを追加してください。

```
qt_measure_sensors(current_time_ms_touch);
```

これが現在実装される方法で、常に計数器は0に留まりますが、それは当分OKです。

主繰り返し速度低下

今や首尾よくポーリングでの接触感知器を持ちます。けれども、この繰り返しは全速でのポーリングを維持します。これは必要ではなく、故にこれをかなり遅くすることができます。主繰り返しの内側にアイドル遅延を追加してください。

```
_delay_ms( 10 );
current_time_ms_touch += 10;
```

お分かりのように、これは計時計数器を増加する便宜点でもあります。

これはかなり速い10ms(100Hz)毎に接触キーをポーリングします。25msがもっと一般的です。

注: もっと複雑な応用では接触測定が殆ど計時器によって起動されるようです。ここでは簡単化のために多忙(ポーリング)繰り返しを用いています。

接触データからキー状態読み込み

今や10ms毎に更新される利用可能で有効な接触データを持ちます。釦のON/OFF押下状況は以下のこの構造体を読むことによって得られます。

```
char key_states = qt_measure_data.qt_touch_status.sensor_states[0];
```

これを主繰り返しの"read key states"下に加えてください。key_statesは最初の8つの感知器の2進状態で一般化したものを取得します。1つの感知器(キー)だけが許可され、故にそれの状態はkey_statesでのビット0であるべきです。

簡単な応用の追加

キー押下時に光を点滅するようにいくつかのｺｰﾄﾞを追加しましょう。key_statesが読まれた後で主繰り返しに以下のこれを追加してください。

```
/* 全光OFF */
PORTE.OUT = 0xFF;
if(key_states & 1){
    /* 全光ON */
    PORTE.OUT = 0xF0;
}
```

これはｷｰ押下時に左列の光がON/OFF交互切り替わり、それはｷｰ0とXplain QTouch基板上の最も上部の2つのｷｰです。

任意選択: 2つ目のキー追加

ﾁｬﾈﾙ1で2つ目のｷｰを追加することができるかどうかやって見てみてください。ここは右側の光をONにするいくつかのｺｰﾄﾞで、これは有用かもしれません。

```
/* 右側光ON */
PORTE.OUT = 0x0F;
```

2.2.3. 更なる説明

この作業で使用される関数呼び出しの殆どはかなり簡単で自己説明的です。最も複雑な行はqt_enable_key()関数で、故にこれでより密接に考察してみましょう。

```
void qt_enable_key( channel_t channel, aks_group_t aks_group, threshold_t detect_threshold,
                    hysteresis_t detect_hysteresis );
```

ﾊﾟﾗﾒｰﾀは次の通りです。

- channel : ｷｰ感知器が使用する接触ﾁｬﾈﾙ
- aks_group : (どれかの)感知器が入力の場合のAKS群
- detect_threshold : 感知器検出閾値
- detect_hysteresis : 感知器検出ﾋｽﾃﾘｼｽ値

Tip 隣接ｷｰ消去(AKS:Adjacent Key Suppression)群は同時に検出されることができないｷｰまたは感知器の集まりです。従って2つのｷｰがPCB上でお互いに近く、偶発的に押されるのを望まないなら、それら2つのｷｰを同じAKS群に置くことが良い考えかもしれません。群内で押された最初のｷｰが押下に留まります。

2.3. 課題3: 摺動子とﾃﾞﾊﾞｯｸﾞ ｲﾝﾀｰﾌｪｰｽの追加

2.3.1. 序説

この課題はAVR QTouch Studioに接触ﾃﾞｰﾀと統計を出力ための摺動子とﾃﾞﾊﾞｯｸﾞ ｲﾝﾀｰﾌｪｽの構成設定法を示します。ﾃﾞﾊﾞｯｸﾞ ｲﾝﾀｰﾌｪｰｽに興味がなければ、この作業の部分を安全に飛ばすことができます。

AVR QTouch Studioと正しく接続するために、XplainのAT90USB1287ﾃﾞﾊﾞｲｽはUSBﾌﾞﾘｯｼﾞ ﾁｯﾌﾟ ﾌｧｰﾑｳｪｱでﾌﾟﾛｸﾞﾗﾐﾝｸﾞされていることが必要です。このﾌｧｰﾑｳｪｱは練習ﾌｧｲﾙと共に供給されるべきです。

QTouchﾗｲﾌﾞﾗﾘ ﾌｧｲﾙ存在検査

この作業を始める前に、ﾌﾟﾛｼﾞｪｸﾄ ﾌｫﾙﾀﾞに以下の4つのﾌｧｲﾙが存在することを確認してください。

libavrxmega7g1-8qt-k-2rs.a
touch_api.h
touch_qt_config.h
qt_asm_xmega.s

それらが無い場合、ｲﾝｽﾄｰﾙしたQTouchﾗｲﾌﾞﾗﾘからそれらを複写してください。

2.3.2. 摺動子の追加

摺動子は単純なｷｰ以外の感知器のより精巧な形式です。それらは度々配置に於いて直線状形式を取り、使用者は摺動子値を変更するために指を上下に移動することができます。摺動子値は変化する分解能を持つことができます。例の摺動子は8ﾋﾞｯﾄで、故に256の異なる値を持ちます。

摺動子は多数のﾁｬﾈﾙを使用します。Xplain QTouch基板上の摺動子は3つのﾁｬﾈﾙ、2,3,4を使用します。

この課題を開始するための課題3ﾌﾟﾛｼﾞｪｸﾄ ﾌｧｲﾙ読み込み

一旦読み込まれたなら、次にProject Optionsを開いてLibrariesﾀｸﾞを見つけてください。課題2で行ったようではなく、今回はQTouchﾗｲﾌﾞﾗﾘ ﾌｧｲﾙのk-2rs版を読み込みます。これはﾗｲﾌﾞﾗﾘが単に課題2でのｷｰのようでなく、ｷｰと2つの回転子または摺動子を支援することを意味します。k-2rsﾗｲﾌﾞﾗﾘはk-0rsﾗｲﾌﾞﾗﾘよりも僅かに大きいです。

それに慣れを感じるまで新しいｺｰﾄﾞを調べるために時間を取ってください。

この課題は課題2で構成したものに非常に似ていますが、いくつかの顕著な違いがあります。例えば、今や25ms間隔計時器と測定実行時に結果を出すためのﾌﾗｸﾞを使用します。

ﾃﾞﾊﾞｯｸﾞ ｲﾝﾀｰﾌｪｰｽを参照する少しのｺｰﾄﾞもあり、それのいくつかは注釈にされています。あまりこれを気にしないでください。後で考察します。

ｺｰﾄﾞのｺﾝﾊﾟｲﾙと試験

2つのｷｰが活性であるべきで、上側のｷｰは上側4つの光を点灯するでしょう。そして下側のｷｰは下側4つの光を制御します。

摺動子形態設定はかなり容易で、1行の追加だけが必要です。

ﾗﾍﾞﾙ"enable sensor 2"の下に以下のｺｰﾄﾞを追加してください。

```
qt_enable_slider( CHANNEL_2, CHANNEL_4, AKS_GROUP_1, 16u, HYST_6_25, RES_8_BIT, 0u  );
```

これはﾁｬﾈﾙ2,3,4を用いて摺動子を許可します。これ、RES_8_BIT引数が摺動子の分解能であることに注意してください。他の引数は今のところあまり重要ではありません。

それはそれで、摺動子は今やｷｰと同時にﾎﾟｰﾘﾝｸﾞされて計算されます。

応用での摺動子値の使用

ｷｰとは違い、摺動子は押下有り/なしの押下状態のﾌﾞｰﾙ値に加えて、ｽｶﾗ値も持ちます。摺動子は0～255のどの値も持てます。応用に於いてこれは殆ど何にでも使用することができ、音量制御のような連想機能が容易です。もっと先を望むなら、使用者が下を押した時に値を記録してその位置から相対的な上下調整することができます。

この例に対してもっとより簡単な何かを行うつもりで、それは光を上下に移動するつもりです。

光を交互切り替えする2つのif文の後に以下のｺｰﾄﾞを追加してください。

```
if(s_states & (1<<2)){
    uint16_t sliderval = qt_measure_data.qt_touch_status.rotor_slider_values[0];
    val = 1<<(sliderval/32);
}
```

お分かりのように、摺動子上で接触があることを意味する、それが検出されているかを見るため、最初に感知器番号2(ｷｰは各々感知器0と1)に対する感知器の状態を検査します。

そうならば、qt_measure_dataから現在の摺動子値を読みます。摺動子は最初の回転子または摺動子で、故にそれはrotor_slider_valuesrotor_slider_values配列内で指標0を持ちます。

その後にXplain基板上で正しい光を点灯するためにいくつかのﾋﾞｯﾄ移動の小細工を行います。

任意選択: 光の方向修正

摺動子を滑らせる時に、光は直感的なものと逆方向になります。これは摺動子での0が頂上だからです。

いくつかの簡単なｺｰﾄﾞでこれを修正することができますか?。

任意選択: 回転子追加

回転子は基本的に摺動子で、円で周回するだけです。Xplain QTouch基板上のﾁｬﾈﾙ5～7が回転子に割り当てられています。回転子も許可するようにいくつかのｺｰﾄﾞを追加することができますか?。それは摺動子を初期化した方法と殆ど同じで、単に"摺動子(slider)"を"回転子(rotor)"に取り替えます。それがどのように行われるのか不思議に思うのなら、何時でも例のﾌﾟﾛｼﾞｪｸﾄを見ることができます。

2.3.3. ﾃﾞﾊﾞｯｸﾞ ｲﾝﾀｰﾌｪｰｽの使用

AVR QTouch Studioが走行しているｺﾝﾋﾟｭｰﾀと様々な接触感知器情報を通信するためにﾃﾞﾊﾞｯｸﾞ ｲﾝﾀｰﾌｪｰｽが使用されます。XMEGAで見られるようなﾃﾞﾊﾞｯｸﾞ ｲﾝﾀｰﾌｪｰｽがﾌｧｰﾑｳｪｱで完全に実装され、そのｺｰﾄﾞはmainﾌｧｲﾙの終り近くで得られます。ｺｰﾄﾞを調査するなら、それが非常に簡単なｼﾘｱﾙ転送であることを知るでしょう。

ﾃﾞﾊﾞｯｸﾞ情報用保持部作成

ﾃﾞﾊﾞｯｸﾞ ｲﾝﾀｰﾌｪｰｽを実装する最初の段階は送信前のﾃﾞﾊﾞｯｸﾞ ﾃﾞｰﾀを保持するためのいくつかの変数を作成して配置を形態設定することで、従ってｷｰ、回転子、または摺動子の何を使うのかをAVR QTouch Studioに知らせることができます。

全体変数下にこれら2つの宣言を追加します。

```
uint8_t sensor_config[QT_NUM_CHANNELS];
board_info_t board_info;
```

sensor_configは、驚くことはなく、感知器がどう形態設定されているかの情報を保持し、boad_infoは基板についての全般情報を保持します。

感知器がどう形態設定されているかをﾃﾞﾊﾞｯｸﾞ ｲﾝﾀｰﾌｪｰｽに通知

AVR QTouch Studioはﾁｬﾈﾙがどう形態設定されているのかを知ることが必要です。どれが使用され、そしてどれが何の感知器を構成するか?。ここでは感知器0に対するﾃﾞｰﾀを形態設定することによって始めるつもりで、この場合は上部ｷｰでﾁｬﾈﾙ0です。

ﾗﾍﾞﾙ"configure the degug data"下に以下の行を追加します。

```
sensor_config[0] = SENSOR_CONFIG( CHANNEL_0, CHANNEL_0, SENSOR_TYPE_KEY );
```

sensor_configの指標0に値を割り当てるのにSENSOR_CONFIGﾏｸﾛを用い、これは感知器0に対応します。

上のﾊﾟﾗﾒｰﾀはﾁｬﾈﾙ0から始まりﾁｬﾈﾙ0で終わるﾁｬﾈﾙを使用する、ｷｰ(key)型の感知器を持つことを意味します。

2つ目のｷｰを形態設定するための行を追加してください。今度はCHANNEL_1を使用してsensor_config[1]に格納することを憶えて置いてください。

感知器はそれらが形態設定された順に番号付けされ、よってこれに関して注意してください。qt_enable_keyとqt_enable_sliderの呼び出し順が問題になります。

形態設定を望む最後の感知器は感知器番号2の摺動子です。以下のこの行を追加することによってそれを形態設定します。

```
sensor_config[2] = SENSOR_CONFIG( CHANNEL_2, CHANNEL_4, SENSOR_TYPE_SLIDER );
```

今回は型が変更され、開始と終了のﾁｬﾈﾙは摺動子が使用する3つ全てのﾁｬﾈﾙを含むことに注意してください。

感知器が使用されないことに関してﾃﾞﾊﾞｯｸﾞ ｲﾝﾀｰﾌｪｰｽへ明示的に告げるのは良い習慣です。この基板は8つのﾁｬﾈﾙ、故に(全てがｷｰならば)最大8つの感知器を持ちます。残りの感知器を不活性としてそれらの行を形態設定に追加してください。

```
sensor_config[3] = SENSOR_CONFIG( CHANNEL_0, CHANNEL_0, SENSOR_TYPE_UNASSIGNED );
sensor_config[4] = SENSOR_CONFIG( CHANNEL_0, CHANNEL_0, SENSOR_TYPE_UNASSIGNED );
sensor_config[5] = SENSOR_CONFIG( CHANNEL_0, CHANNEL_0, SENSOR_TYPE_UNASSIGNED );
sensor_config[6] = SENSOR_CONFIG( CHANNEL_0, CHANNEL_0, SENSOR_TYPE_UNASSIGNED );
sensor_config[7] = SENSOR_CONFIG( CHANNEL_0, CHANNEL_0, SENSOR_TYPE_UNASSIGNED );
```

助言: ｺｰﾄﾞを複写&貼り付けしてください。

今はﾃﾞﾊﾞｯｸﾞ ｲﾝﾀｰﾌｪｰｽを形態設定しただけで、これは感知器が実際にどう動くのかとは関係ありません。

基板情報形態設定

ﾃﾞﾊﾞｯｸﾞ ｼｽﾃﾑへ基板についていくつかのより多くの情報の追加が必要です。これはAVR QTouch Studioが基板を認証できてXplain QTouch基板用の適切な配置を表示するために行われます。

感知器形態設定の直後に以下を追加します。

```
board_info.qt_max_num_rotors_sliders_board_id = ( ( QT_MAX_NUM_ROTORS_SLIDERS << 4 ) | BOARD_ID );
board_info.qt_num_channels = QT_NUM_CHANNELS;
```

ﾃﾞﾊﾞｯｸﾞ報告送出開始

今やﾃﾞﾊﾞｯｸﾞ形態設定を完了し、報告を送る準備が整いました。1つの報告はreport_degug_data()呼び出しによって送られ、故に感知器を測定した直後でこれの呼び出しを追加してください。ここでこれは"ﾎｽﾄへのﾃﾞﾊﾞｯｸ ﾃﾞｰﾀ戻し報告(report degug data back to host)"と言います。

ﾃﾞﾊﾞｯｸﾞ報告の送出は少しの時間がかかり得るので、接触測定毎に1回だけ行われるべきです。従ってreport_debug_data呼び出しがqt_measure_sensorsと同じif部の内側で、それが測定呼び出し自体の後であることを絶対に保証してください。

ｺﾝﾊﾟｲﾙ前にreoprt_debug_data関数の注釈を外すことを確実に行ってください。

これはそれの周囲の/* comment block */を取り去ることによって行われます。これが正しく行われなかった場合、"undefined reference to 'report_debug_data'('report_debug_data'参照未定義)"異常になります。

それはそれとして、ﾃﾞﾊﾞｯｸﾞ ｲﾝﾀｰﾌｪｰｽの形態設定を終えました。ｺｰﾄﾞをｺﾝﾊﾟｲﾙして走らせてください。

AVR QTouch Studio開始、受信ﾃﾞｰﾀ確認

全てのことが正しく構成されたなら、今やQTouch Studioはﾃﾞﾊﾞｲｽからの生ﾃﾞｰﾀを報告するでしょう。
おめでとう御座います。今やあなたはあなた自身のQTouch応用の開発とﾃﾞﾊﾞｯｸﾞを始める準備ができました。

3. 要約

この実地での主な到達点はQTouchﾗｲﾌﾞﾗﾘを構成設定して使用する方法を学ぶことです。

- ﾁｬﾈﾙ初期化法
- 予めｺﾝﾊﾟｲﾙされたﾗｲﾌﾞﾗﾘの使い方
- 摺動子と輪(回転子)の使い方と調整

この全てに加えて、あなたはCｺｰﾄﾞを書いてｺｰﾄﾞをﾃﾞﾊﾞｯｸﾞしてﾃﾞﾊﾞｲｽにﾌﾟﾛｸﾞﾗﾐﾝｸﾞするためのAVR Studio GUIの使い方を学びました。

4. ATMEL技術支援ｾﾝﾀ

ATMELは以下のように利用可能な多数の支援経路を持っています。

○ ｳｪﾌﾞ入口：http://suppoty.atmel.no/　全てのATMELﾏｲｸﾛ ｺﾝﾄﾛｰﾗ
○ E-mail　：avr@atmel.com　全てのAVR製品
○ E-mail　：avr32@atmel.com　全てのAVR32製品

以下のｻｰﾋﾞｽへのｱｸｾｽを得るためにｳｪﾌﾞの入口で登録してください。

○ 豊富なFAQﾃﾞｰﾀﾍﾞｰｽへのｱｸｾｽ
○ 技術的な支援要請の容易な依頼
○ あなたの過去の支援要請の履歴
○ ATMELﾏｲｸﾛ ｺﾝﾄﾛｰﾗの時事通信を受け取るための登録
○ 利用可能が練習と練習材料についての情報取得

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AVR1016応用記述(doc8304.pdf Rev.8304A-04/10)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。