応用記述

AT1886 : AVRGCCでのｱｾﾝﾌﾞﾘとCの混合

8ﾋﾞｯﾄ AVR ﾏｲｸﾛ ｺﾝﾄﾛｰﾗ

要点

- Atmel® Studio 6での容易なCとｱｾﾝﾌﾞﾘ ｿｰｽの結合
  - Cはｱｾﾝﾌﾞﾘ ﾙｰﾁﾝを呼び出し可
  - ｱｾﾝﾌﾞﾘはC関数を呼び出し可
- Cとｱｾﾝﾌﾞﾘ間での変数とﾎﾟｲﾝﾀを渡す。
- 全域変数の共用をCとｱｾﾝﾌﾞﾘに許す。

序説

この応用記述はAtmel Studio 6 IDEを用いてAVRGCCﾌﾟﾛｼﾞｪｸﾄでCとｱｾﾝﾌﾞﾘの両ｺｰﾄﾞを結合する方法を記述します。この応用記述はCが選択言語で必要または便利のどちらかであるｱｾﾝﾌﾞﾘ言語が解決策でｲﾝｸﾙｰﾄﾞされる考え方で書かれます。

Studio 6(と以前の版)は同じﾌﾟﾛｼﾞｪｸﾄ内でｱｾﾝﾌﾞﾘ言語とCﾌｧｲﾙの両方での作業を許します。疑問はｿﾌﾄｳｪｱ技術者がｱｾﾝﾌﾞﾘ ｺｰﾄﾞとCｺｰﾄﾞをどう構成するかです。ﾊﾟﾗﾒｰﾀ、変数、それとｱｾﾝﾌﾞﾘ ｺｰﾄﾞとCｺｰﾄﾞ間で渡された同様のものはどうですか?

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. 42055B-11/2012, 42055BJ1-12/2013

目次

1. 事前必要条件

この資料で検討される解決策は、以下の技能と技術の基本的な習熟が必要です。より多くを学ぶため、11.章 参照を参照してください。

- Atmel Studio 6
- ATMELのﾃﾞﾊﾞｯｶﾞ JTAGICE mk Ⅱ またはJTAGICE 3
- ATMEL STK® 600ｽﾀｰﾀ ｷｯﾄ

2. ﾌﾟﾛｼﾞｪｸﾄ構成

ﾌﾟﾛｼﾞｪｸﾄを構成するのに特別な必要条件はありません。Cｺｰﾄﾞまたはｱｾﾝﾌﾞﾘのどちらであろうとも、ﾍｯﾀﾞ ﾌｧｲﾙとして識別されたﾌｧｲﾙはﾌﾟﾛｼﾞｪｸﾄのﾍｯﾀﾞ ﾌｧｲﾙ ﾌｫﾙﾀﾞに置かれます。ｿｰｽ ﾌｧｲﾙとして識別されたﾌｧｲﾙはﾌﾟﾛｼﾞｪｸﾄのｿｰｽ ﾌｧｲﾙ ﾌｫﾙﾀﾞに置かれます。

例え厳密に必要とされなくても、全てのｱｾﾝﾌﾞﾘ言語ｿｰｽ ﾌｧｲﾙは".S"拡張子を持つべきです。これは必要な時に自動的にｱｾﾝﾌﾞﾗまたはﾘﾝｶを呼び出すことをCｺﾝﾊﾟｲﾗ前置処理部に許します。加えて、Cﾌﾟﾘﾌﾟﾛｾｯｻは抽象的な定数(ｼﾝﾎﾞﾙ定数)の使用の自動的な許諾が実施されます。

ｿｰｽ ﾌｧｲﾙの1つはﾘﾝｶが応用を開始する場所を知るよう、ﾘﾝｶに対して"main"部署を生成することが必要です。最も一般的なのは"main"と呼ばれる関数を持つCｺｰﾄﾞです。けれども、"main"と名付けられて(".global"擬似命令を使用して)全域宣言されたｻﾌﾞﾙｰﾁﾝを持つｱｾﾝﾌﾞﾘ ﾌｧｲﾙも"main"と名付けられた単位部を生成します。

単一ﾌﾟﾛｼﾞｪｸﾄでのCとｱｾﾝﾌﾞﾘの組み合わせは応用の必要に応じて様々な疑問を引き起こします。それらの疑問は以下に含まれるかもしれません。

- C関数がｱｾﾝﾌﾞﾘ ﾙｰﾁﾝを呼ぶことができるように、ｱｾﾝﾌﾞﾘ ﾙｰﾁﾝはCｺﾝﾊﾟｲﾗに見えるようにどう作ることができますか?
- 同様に、ｱｾﾝﾌﾞﾘ ﾙｰﾁﾝがC関数を呼ぶことができるように、C関数はｱｾﾝﾌﾞﾘ ﾙｰﾁﾝに見えるようにどう作ることができますか?
- 変数はｱｾﾝﾌﾞﾘ ｺｰﾄﾞにどう渡されますか?
- 変数はC関数にどう渡されますか?
- ｱｾﾝﾌﾞﾘ ｺｰﾄﾞとCは同じ全域変数を使うことができますか?

3. 関数の視界

C言語関数はｱｾﾝﾌﾞﾗによって"見える"ためにｱｾﾝﾌﾞﾘ ｺｰﾄﾞ内で外部(external)として宣言されることが必要です。

```
.extern my_C_funtion
```

ｱｾﾝﾌﾞﾘ言語ﾙｰﾁﾝはCｺﾝﾊﾟｲﾗで可視であるためにｱｾﾝﾌﾞﾘ ｺｰﾄﾞ内で全域(global)として宣言されることが必要です。

```
.global my_assembly_fct
```

加えて、ｱｾﾝﾌﾞﾘ言語ﾙｰﾁﾝを呼ぶことを意図されるCﾌｧｲﾙは外部であるｱｾﾝﾌﾞﾘ言語ﾙｰﾁﾝの関数原型宣言を持つことが必要です。

```
extern unsigned char my_assembly_fct (unsigned char, unsigned int);
```

4. 変数

Cｺｰﾄﾞとｱｾﾝﾌﾞﾘ ｺｰﾄﾞの両方は独立して変数をｱｸｾｽすることができます。実際問題として、Cｺｰﾄﾞが変数を管理して値または参照のどちらかによってﾊﾟﾗﾒｰﾀをｱｾﾝﾌﾞﾘ ｺｰﾄﾞへ渡すのが得策です。5. ﾚｼﾞｽﾀの使い方と6. ﾊﾟﾗﾒｰﾀの渡し方の章はﾚｼﾞｽﾀの組がCｺﾝﾊﾟｲﾗによってどう使用され、ﾊﾟﾗﾒｰﾀがどう渡されるのかを記述します。

ｱｾﾝﾌﾞﾘとCの両方に対して同じ全域変数をｱｸｾｽすることが可能です。そのような変数はCｺｰﾄﾞ内で全域変数であり、ｱｾﾝﾌﾞﾘ ｺｰﾄﾞ内で外部(external)として宣言されることが必要です。全域が意図される"my_value"変数を考察してください。Cｺｰﾄﾞに於いて、それは以下のように、何れかの他の変数のように何れかの関数の外側で宣言されます。

```
unsigned char my_value;
```

ｱｾﾝﾌﾞﾘではそれが以下のようにｺｰﾄﾞ化されます。

```
.extern my_value
```

5. ﾚｼﾞｽﾀの使い方

Cｺｰﾄﾞと結合するためのｱｾﾝﾌﾞﾘ言語ﾙｰﾁﾝを書くにはｺﾝﾊﾟｲﾗがﾚｼﾞｽﾀをどう使うかの知識が必要です。

- R0は一時ﾚｼﾞｽﾀでｺﾝﾊﾟｲﾗが生成したｺｰﾄﾞによって使用され得ます。このﾚｼﾞｽﾀを使用してC関数を呼ぶｱｾﾝﾌﾞﾘ ｺｰﾄﾞを書く場合、ｺﾝﾊﾟｲﾗがこのﾚｼﾞｽﾀを使用するかもしれないので、それを保存して回復することが必要です。
- R1はｺﾝﾊﾟｲﾗによって常に0を含むと仮定されます。このﾚｼﾞｽﾀを使用するｱｾﾝﾌﾞﾘ ｺｰﾄﾞはｺﾝﾊﾟｲﾗが生成した何れかのｺｰﾄﾞに戻るまたは呼ぶ前にこのﾚｼﾞｽﾀを解除(=0)すべきです。
- R2～R17,R28,R29はC関数呼び出しがそれらのﾚｼﾞｽﾀを未変化のままにすべきことを意味する"呼び出し保存"ﾚｼﾞｽﾀです。これらのﾚｼﾞｽﾀを使用するCから呼ばれるｱｾﾝﾌﾞﾘ言語ﾙｰﾁﾝは使用するこれらのどのﾚｼﾞｽﾀの内容も保存して回復することが必要です。
- R18～R27,R30,R31はﾚｼﾞｽﾀがどのｺｰﾄﾞに対しても使用可能を意味する"呼び出し使用"ﾚｼﾞｽﾀです。ｺﾝﾊﾟｲﾗが生成したｺｰﾄﾞは使用するそれらのどのﾚｼﾞｽﾀも保存しないため、C関数を呼ぶｱｾﾝﾌﾞﾘ ｺｰﾄﾞはｱｾﾝﾌﾞﾘ ｺｰﾄﾞによって使用されるそれらのﾚｼﾞｽﾀのどれをも保存することが必要です。

表5-1.はCとｱｾﾝﾌﾞﾘ間のﾚｼﾞｽﾀ ｲﾝﾀｰﾌｪｰｽを要約します。

表5-1. Cとｱｾﾝﾌﾞﾘ間のﾚｼﾞｽﾀ ｲﾝﾀｰﾌｪｰｽの要約

ﾚｼﾞｽﾀ	説明	Cから呼ばれるｱｾﾝﾌﾞﾘ ｺｰﾄﾞ	Cｺｰﾄﾞを呼ぶｱｾﾝﾌﾞﾘ ｺｰﾄﾞ
R0	一時	使用するなら、保存して回復	使用するなら、保存して回復
R1	常に0	戻る前に解除(=0)しなければなりません。	呼ぶ前に解除(=0)しなければなりません。
R2～R17	″呼び出し保存″	使用するなら、保存して回復	自由に使用することができます。
R28			
R29			
R18～R27	″呼び出し使用″	自由に使用することができます。	使用するなら、保存して回復
R30			
R31			

6. ﾊﾟﾗﾒｰﾀの渡し方

固定引数一覧内の引数はR25～R8のﾚｼﾞｽﾀを通して左から右に割り当てられます。全ての引数は偶数のﾚｼﾞｽﾀを使用します。これはchar引数が2つのﾚｼﾞｽﾀを消費することに帰着します。ﾚｼﾞｽﾀ内に適応する範囲を超えた追加の引数はｽﾀｯｸで渡されます。

変数引数一覧内の引数は右から左への順でｽﾀｯｸに押し込まれます。char引数は2ﾊﾞｲﾄを消費します。

戻り値は戻り値の大きさに依存してR25～R18のﾚｼﾞｽﾀを使用します。ﾚｼﾞｽﾀとﾊﾞｲﾄ順間の関連は表6-1.で示されます。

表6-1. ﾚｼﾞｽﾀとﾊﾞｲﾄ順間の関連

ﾚｼﾞｽﾀ	R19	R18	R21	R20	R23	R22	R25	R24
ﾊﾞｲﾄ順	ﾊﾞｲﾄ7	ﾊﾞｲﾄ6	ﾊﾞｲﾄ5	ﾊﾞｲﾄ4	ﾊﾞｲﾄ3	ﾊﾞｲﾄ2	ﾊﾞｲﾄ1	ﾊﾞｲﾄ0

7. 実演の構築と走行の方法

実演ｺｰﾄﾞはC内からｱｾﾝﾌﾞﾘ言語ﾙｰﾁﾝを呼び出して2つの言語間でﾊﾟﾗﾒｰﾀを渡すことを説明するための簡単な関数を含みます。加えて、ｱｾﾝﾌﾞﾘ言語割り込み処理ﾙｰﾁﾝが含まれます。

このｺｰﾄﾞは新しい前置分周器値をこのﾙｰﾁﾝに渡してｸﾛｯｸ前置分周器を変更する″change_clook″ｱｾﾝﾌﾞﾘ言語ﾙｰﾁﾝを呼ぶことによって始めます。

ｺｰﾄﾞはその後に渡された2つのﾊﾟﾗﾒｰﾀを共に加算し、加算の結果をCに返すｱｾﾝﾌﾞﾘ言語ﾙｰﾁﾝを呼びます。

ｺｰﾄﾞはその後に周期的な割り込みを生成するためにﾀｲﾏ/ｶｳﾝﾀ0を初期化し、その後にｺｰﾄﾞは″while″繰り返しに留まります。割り込み発生時、ｱｾﾝﾌﾞﾘ言語割り込み処理ﾙｰﾁﾝはﾎﾟｰﾄDのﾋﾞｯﾄ0(PD0)を交互切り替えします。

8. 必要とされるﾊｰﾄﾞｳｪｱ

- ATMEL STK600ｽﾀｰﾀ ｷｯﾄ
- (STK600と共に含まれる)STK600-ATmega2560ｶｰﾄﾞ
- ATMELのﾃﾞﾊﾞｯｶﾞ(JTAGICE mkⅡ、JTAGICE 3、ATMEL AVR Dragon™、など)
- Atmel Studio 6
- (STK600と共に含まれる)10ﾋﾟﾝIDCﾘﾎﾞﾝ ｹｰﾌﾞﾙ

9. ﾊｰﾄﾞｳｪｱ構成設定

STK600基板上で以下の接続を行ってください。

- STK600上にSTK600-ATmega2560を装着してください。
- ﾎﾟｰﾄD(PORTD)とLEDﾍｯﾀﾞ間に10ﾋﾟﾝ ﾘﾎﾞﾝ ｹｰﾌﾞﾙを接続してください。
- STK600上でﾃﾞﾊﾞｲｽのJTAGﾍｯﾀﾞにﾃﾞﾊﾞｯｶﾞのｹｰﾌﾞﾙを接続してください。

10. 手順

1. www.atmel.comからAT1886.zipをﾀﾞｳﾝﾛｰﾄﾞしてください。
2. 作業ﾃﾞｨﾚｸﾄﾘ内にﾌｧｲﾙを解凍してください。
3. Atmel Studio 6でAVR1886.atslnﾌﾟﾛｼﾞｪｸﾄ ﾌｧｲﾙを開いてください。
4. Studioのﾎﾞｯｸｽであなたのﾃﾞﾊﾞｯｸﾞ ﾂｰﾙを選んでください。
5. 目的対象ﾃﾞﾊﾞｲｽ(Target device)としてATmega2560を指定してください。
6. ﾌﾟﾛｼﾞｪｸﾄを構築してください。
7. ﾃﾞﾊﾞｯｸﾞ動作へ移行してください。
8. 応用を走らせてください。
9. 監視(Watch)ｳｨﾝﾄﾞｳに"val_1"、"val_2"、"val_3"を追加してください。
10. mainで"val_1="指定上に中断点(ﾌﾞﾚｰｸﾎﾟｲﾝﾄ)を設定してください。
11. 右のﾌﾟﾛｾｯｻ(Processor)ﾀﾌﾞで、ﾚｼﾞｽﾀ(Registers)ｳｨﾝﾄﾞｳを拡大してください。
12. 走行(Run)釦をｸﾘｯｸしてください。ｺｰﾄﾞは"change_clook"関数を実行してその後に中断すべきです。
13. "init_timer_0"関数呼び出しに達するまで、次の命令を通して段階実行するために単一段階実行(Single step)機能を使用してください。ﾊﾟﾗﾒｰﾀがﾚｼﾞｽﾀ経由でCから"add_two"ｱｾﾝﾌﾞﾘ ﾙｰﾁﾝに渡されるのを観察することができるでしょう。
14. 一旦"init_timer_0"関数に達したなら、(Run)釦をｸﾘｯｸしてください。これはCを使用して計時器を初期化して計時器からの周期的な割り込みを許可します。割り込み処理はｱｾﾝﾌﾞﾘ言語です。PD0のLED点滅を見るでしょう。

11. 参照

1. Atmel Studio 6 – www.atmel.com
2. ATMEL AVR JTAGICE mkⅡ – www.atmel.com
3. ATMEL JTAGICE 3 – www.atmel.com
4. ATMEL STK600 – www.atmel.com

12. 改訂履歴

資料改訂	日付	注釈
42055A	2012年11月	初版資料公開
42055B	2012年11月	資料はAVR1886からAT1886へ改名

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

© 2012 Atmel Corporation. 全権利予約済 / 改訂:42055B-AVR-11/2012

Atmel®、ﾛｺﾞとそれらの組み合わせ、Enabling Unlimited Possibilities®、AVR®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のｳｪﾌﾞ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* 2013.

本応用記述はAtmelのAT1886応用記述(doc42055.pdf Rev.42055B-11/12)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。