

AVR001 : 条件付アセンブルと可搬性マクロ

要点

- 増した可搬性
- より容易なコード書き
- 簡単化されたI/Oレジスタアクセス
- 改善されたアセンブル状態還元

1. 序説

この応用記述はAVRアセンブラ Ver. 1.74とそれ以降に存在する条件付アセンブル機能を記述します。AVRアセンブラ®はATMELのウェブサイトのAVR部署で得られるAVR Studio®と共に既定でインストールされます。

条件付アセンブルの使用法の例が提供されます。例の1つはデバイス定義ファイルの変更以外の他の修正なしにどのAVR®もアセンブルする標準的なコードを書くことをプログラマに許すマクロ群です。

2. 動作の理屈

条件付アセンブル(CA)はVer. 1.74のAVRアセンブラで導入されました。条件付アセンブルはC言語で利用可能なプリプロセッサ擬似命令と同様の一連の擬似命令に基いています。

Ver. 2.0以降のAVRアセンブラもCのプリプロセッサを模したプリプロセッサを持ちます。これは例えば、使用されるデバイスに対するCヘッダファイルを許します。

通常のプリプロセッサと条件付きアセンブル擬似命令の違いは後者がアセンブル時に評価され、従ってマクロを上書きする意味として使用されるかもしれないことです。通常のプリプロセッサは、例えばデバイス依存コードの形成と選択(例えそれが更に条件付アセンブル擬似命令で行われ得るとしても)に使用されるかもしれませんが。(本応用記述は条件付きアセンブル擬似命令だけを扱います。)

擬似命令の全一覧についてはアセンブラのAVR studioヘルプをご覧ください。

2.1. 条件付アセンブル

アセンブラは条件付アセンブル擬似命令式をアセンブル時に評価し、条件付アセンブル擬似命令によって囲まれたコードが含まれるべきか否かを決めます。

条件付アセンブル擬似命令の一覧は表2-1.で示されます。

表2-1. 条件付アセンブル擬似命令

擬似命令	内容
<code>.ifdef <シンボル></code>	シンボルが定義済みなら、 <code>.ifdef</code> と対応する <code>.else</code> または <code>.endif</code> 間に存在するコードを含めます。 <code>.EQU</code> と <code>.SET</code> によって宣言されたシンボルだけが、 <code>ifdef</code> によって評価され、 <code>.DEF</code> 擬似命令で定義されたシンボルはアセンブラによって違うように扱われ、 <code>ifdef</code> と共に使用できません。
<code>.ifndef <シンボル></code>	シンボルが未定義なら、 <code>ifdef</code> と対応する <code>.else</code> または <code>.endif</code> 間に存在するコードを含めます。 <code>.EQU</code> と <code>.SET</code> によって宣言されたシンボルだけが、 <code>ifndef</code> によって評価され、 <code>.DEF</code> 擬似命令で定義されたシンボルはアセンブラによって違うように扱われ、 <code>ifndef</code> と共に使用できません。
<code>.if <式></code>	式を0以外(真)と評価したなら、 <code>if</code> と対応する <code>.else</code> 、 <code>.elif</code> または <code>.endif</code> 間に存在するコードを含めます。
<code>.elif <式></code>	<code>.elif</code> は <code>if</code> が先行しなければなりません。式を0以外(真)と評価したなら、 <code>elif</code> と対応する <code>.else</code> 、 <code>.elif</code> または <code>.endif</code> 間に存在するコードを含めます。
<code>.else</code>	<code>.else</code> は <code>if</code> または <code>.elif</code> が先行しなければなりません。対応する <code>if</code> または <code>.elif</code> に対して指定された式を0以外(真)と評価したなら、 <code>.else</code> と対応する <code>.endif</code> 間に存在するコードを含めます。
<code>.endif</code>	<code>.endif</code> は <code>ifdef</code> 、 <code>ifndef</code> 、 <code>if</code> または <code>.elif</code> が先行しなければなりません。 <code>.endif</code> は対応する対応する <code>ifdef</code> 、 <code>ifndef</code> 、 <code>if</code> または <code>.elif</code> に対する範囲の終りを定義します。

擬似命令は5段までネストできます。含まれた条件付アセンブル擬似命令に依存してマクロが違ってアセンブルされるために、条件付アセンブル擬似命令は`.MACRO`擬似命令と組み合わせることができます。これは本応用記述に対するコード例で使用されています。



8-bit **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. 2550E-04/08, 2550EJ3-12/13

2.2. デバッグ疑似命令

コード開発時、各種の還元形式はアセンブル時に有用です。この目的に対して2つの新しい疑似命令が追加されました。1つは出会った場合にメッセージ ウィンドウへメッセージを出力し、もう1つは(異常メッセージを伴う)異常を発行します。デバッグ疑似命令は表2-2.で記述されます。

表2-2. デバッグ疑似命令

疑似命令	内容
<code>.message "文字列"</code>	コードのアセンブル時に <code>.message</code> 疑似命令に出会うと、メッセージ ウィンドウに"文字列"が書かれます。
<code>.error "文字列"</code>	コードのアセンブル時に <code>.error</code> 疑似命令に出会うと、アセンブル異常が発行され、メッセージ ウィンドウに"文字列"が書かれます。

これらの疑似命令はアセンブル状態についての情報を提供するために、(例えば条件付アセンブル)疑似命令と組み合わせることができます。`.message`疑似命令の使い方は本応用記述に対するコード例に含まれています。AVRアセンブラ 1.77.1が`.error`疑似命令に出会った時に更なるアセンブルを止めることに注意してください。

2.3. 例1: 条件付アセンブル疑似命令の一般的な使い方

或るプロジェクトから次へのコード部再使用は度々大いなる時間の節約になり得ます。異なる使用目的に仕立てることを必要とするとき、コード部の再使用はプリプロセッサ疑似命令も、この場合は条件付アセンブル疑似命令もなしで、結果としてコード部の異なる色々な版を生じるでしょう。条件付アセンブル疑似命令を使用する1つの版は様々な目的デバイスに対して使用することができます。

例としてUART通信に使用する入出力ピンの初期化を考察してください。

```

    .EQU      ATmega128  =1          ; ATmega128シンボル宣言
;
    .EQU      ATmega16   =1          ; ATmega16シンボル宣言(現状注釈)
    .EQU      UART      =0          ; UART番号指定(0=UART0または1=UART1)

    .ifdef ATmega128
        .message "UART部をATmega128用にアセンブル." ; ATmega128なら、
        .if UART == 0 ; ATmega128使用表示
            .message "UART0使用." ; UART0指定なら、
                SBI      DDRE, PE1 ; UART0の使用表示
                ; UART0のTxDを出力として設定
        .elif UART == 1 ; UART1指定なら、
            .message "UART1使用." ; UART1使用表示
                SBI      DDRD, PD3 ; UART1のTxDを出力として設定
        .else ; UART0,1以外で、
            .error "無効UART番号" ; 無効UART番号表示
        .endif ;
    .elif ATmega16
        .message "UART部をATmega16用にアセンブル." ; ATmega16なら、
        .if UART == 0 ; ATmega16使用表示
            .message "UART0使用." ; UART0指定なら、
                SBI      DDRD, PD1 ; UART0の使用表示
                ; UART0のTxDを出力として設定
        .else ; UART0以外で、
            .error "無効UART番号" ; 無効UART番号表示
        .endif ;
    .endif ;

```

このコード例で見られるように、デバイス特有の初期化(例えばUART)を1つのファイルに含めることができます。これは再使用するコード部のより良い制御を可能にします。

2.4. 例2: マクロでの条件付アセンブル

本応用記述のコード例は正しくアセンブルするためにAVRアセンブラ Ver. 1.77またはそれ以降が必要です。AVRアセンブラはAVR Studioに含まれ、それはATMELのウェブサイトのAVR部署からダウンロードできます。

"macros.inc"ファイルはI/Oとデータ空間で容易にビットとバイトをアクセスするためのいくつかのマクロを含みます。マクロは表2-3.に於いて注釈付きで一覧されます。

表2-1. "macros.inc"ファイル内のマクロ定義

マクロ名	内容
SETB [アドレス, ビット番号, レジスタ]	"ビット設定" - I/O空間の'アドレス'位置で'ビット番号'のビットを設定(1)します。'レジスタ'はR16~R31が使用できます。
CLRB [アドレス, ビット番号, レジスタ]	"ビット解除" - I/O空間の'アドレス'位置で'ビット番号'のビットを解除(0)します。'レジスタ'はR16~R31が使用できます。
SKBS [アドレス, ビット番号, レジスタ]	"ビット=1ならスキップ" - I/O空間の'アドレス'位置で'ビット番号'によって指定されたビットが設定(1)なら、マクロに後続する命令をスキップします。
SKBC [アドレス, ビット番号, レジスタ]	"ビット=0ならスキップ" - I/O空間の'アドレス'位置で'ビット番号'によって指定されたビットが解除(0)なら、マクロに後続する命令をスキップします。
STORE [アドレス, レジスタ]	"レジスタ格納" - I/O空間の'アドレス'位置に'レジスタ'の内容を格納します。
LOAD [レジスタ, アドレス]	"レジスタ格納" - I/O空間の'アドレス'位置から内容を'レジスタ'に取得します。

I/O空間(と拡張I/O空間)をアクセスするのにこれらのマクロを使用する理由は、アクセスするレジスタがI/O空間の何処に配置されているかをコードの書き手が考慮する必要がないからです。全ての命令がI/O空間の全アドレスに届く訳ではないので、(既存)マクロを使用しない場合に、これが必要でしょう。故に利点は多数あります。

- ・ 著者はI/O配置を知る必要がなく、只レジスタの名前だけです。
- ・ レジスタ名とビット名に対する標準定義ファイルが使用できます。
- ・ レジスタアクセスに最もコード量効率的な命令が使用されます。
- ・ アセンブリコードがコード修正なしでどのデバイスにも移転できます。

2.4.1. ビット設定マクロ

全てのマクロが本質的に同様のため、SETBだけが詳細に記述されます。

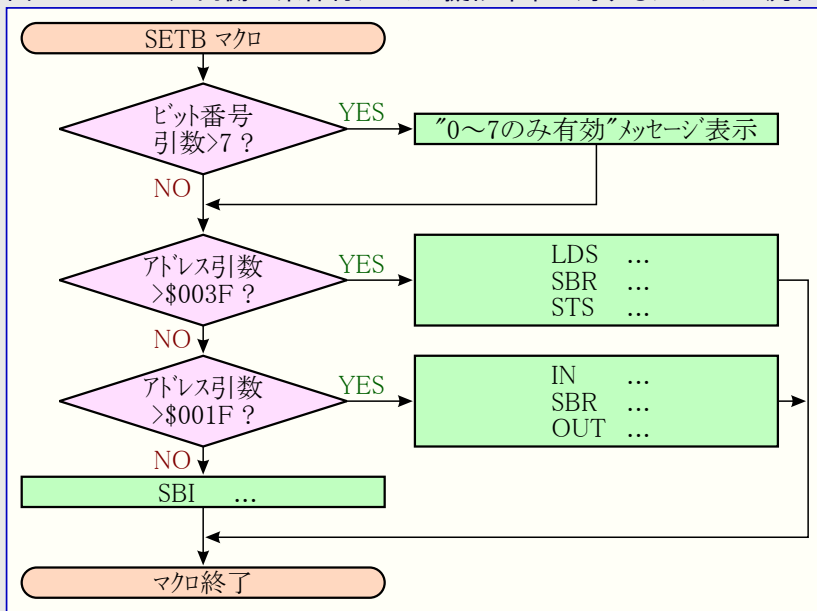
3つの引数、目的アドレス、ビット番号、レジスタがSETBマクロに対して指定されます。レジスタはアドレスが\$001Fよりも大きな場合にだけ使用されますが、正しいアセンブル結果と最良の可搬性の機会を保証するために、いつも指定することが推奨されます。

ビット番号の範囲は0~7の間であることを検証され、条件違反の場合は.errorr擬似命令を使用して誤りが発行されます。

アドレスが\$001F以下の場合はビットを設定(1)するのにSBI命令が使用されます。アドレスが\$0020~\$003F間なら、そのアドレスのアクセスにIN命令とOUT命令が使用されます。最後に、アドレスが\$0040以上なら、LDS命令とSTS命令が使用されます。

図2-1.は条件付アセンブル擬似命令を使用するSETBマクロをアセンブラがどう取り扱うかを示します。

図2-1. SETBマクロ内側の条件付アセンブル擬似命令に対するアセンブルの流れ



2.5. 改良

YまたはZレジスタの1つの犠牲で、これらのレジスタの1つが間接アクセス用に予約されてLDS命令とSTS命令がLDD命令とSTD命令に置換できるなら、LOADとSTOREのマクロはより速い実行とともっと簡潔なコードに改良できます。これはコードの書き手を圧迫するため、提供した実装に付加されていません。



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
東熱新川ビル 9F
アトメル ジャパン株式会社
TEL (81) 03-3523-3551
FAX (81) 03-3523-7581

製品窓口

ウェブサイト

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

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、または利益の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。

© Atmel Corporation 2008. 全権利予約済 ATMEL®、ロゴとそれらの組み合わせ、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。

© HERO 2013.

本応用記述はATMELのAVR001応用記述(doc2550.pdf Rev.2550E-04/08)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。