# AVRGCC/AVR Studio ビギナーズ ガイド

Rev 1.0

注意 :すべての ソフトウェアの バージョンとURLは 2001年 12月 13日現在のものです。

## AVR Studioでの AVRGCCの使用法とインストール

ここでは、AVR StudioとGNU AVR Cコンパイラ (avg-gcc)の インストールと、これらでの作業方法を判り易く簡単に説明します。 サンプル プログラムの完全な構築 (ビルド)と、選択した目的AVRデバイスの STK500開発基板上での プログラミングを段階的に説明します。
この説明では、以下のものを使用します。

- 2001年 11月 8日 リリースの Ver 3.53以降の AVR Studio (自己展開 ファイル astudio.exe) [5.9M]
  この ファイルは、www.avrfreaks.netからダウンロードできます。
- 2001年 12月 7日以降に AVR freaksが配布した avrgcc200112XXa_AVR freaks.exe [8.1M]
  これには以下のものも含まれています。
  - Flavio Gobberの Elf2Coffコンバータ
  - Volker Othによる gcctest1-9

本説明を完全に実現するには、必要とする正確な追加 ファイル (.batファイル ,メイクファイル )の 全てを含む、www.avrfreaks.netから ダウンロードできる avrgcc200112XXa_AVR freaks.exeを使用することが重要です。

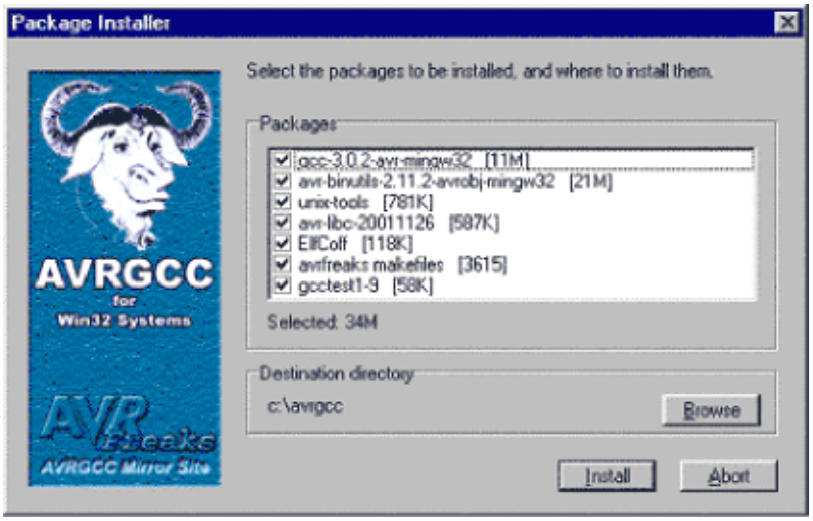
勿論、これらの ソフトウェア以外に STK500開発基板 (若しくは相当品 )、AVRデバイス (例えば AT90S8515)、Windows9x/NT/2000で動作する標準的な パーソナル コンピュータ(PC)が必要です。以降の説明は STK500が既に PC上の COMポートで動作するように設定されていることが前提です。

## AVR Studioの インストール

これは問題ないと思います。自己解凍 ファイル astudio.exeを選択した フォルダに解凍します。解凍完了後、その フォルダ内の install.exe を実行します。その後は画面上の指示に従います。

## AVRGCCの インストール

これはより直接的な手順で、ダウンロードした ファイルを単に実行します。利便性を考慮し、既定 インストール位置の c:\avrgccを選択します。7つの チェック ボックスは、すべて チェック状態のままにします。




一見、UnixToolsが不要と思われますが、これらには make.exe及び rm.exeユーティリティが 含まれていますので必要となります。

全 ライブラリの コンパイルを含む完全な インストールの確認のため、いくつかの検査を行うことが重要です。

- インストール中に DOS窓が現れ、1分間程度 コンパイル出力行が表示されます。
- コンパイル完了時、your pop-up program is ready to run. (ポップアップ プログラム実行可能 )が表示された別の DOS窓が現れます。この DOS窓は Ctrl+Cキー押下で閉じます。

  注 :Windows2000では、この表示が現れません。

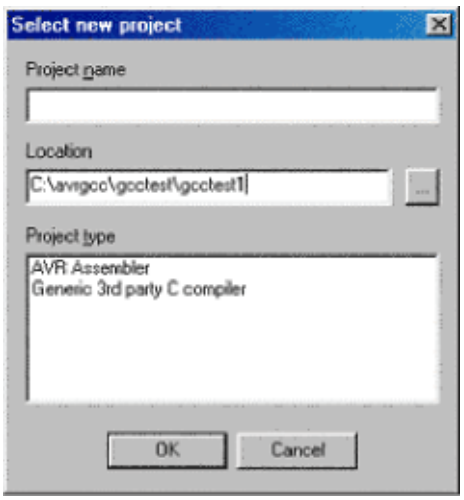
  この表示が現れなかった場合、問題がある可能性があります。オブジェクト ライブラリが適切に コンパイルされなかったかもしれません。この場合、run.batファイルを手動で実行する必要があります。run.batファイルは、avr-gccを インストールした フォルダ内にあります。
- インストール フォルダに既定 フォルダを選択すると c:\avrgccフォルダが作成され、C:\avrgcc\avr\libフォルダに インストールした日付の oファイルが多数存在する筈です。もしも、これらがない場合は、run.batファイルを手動で実行してください。

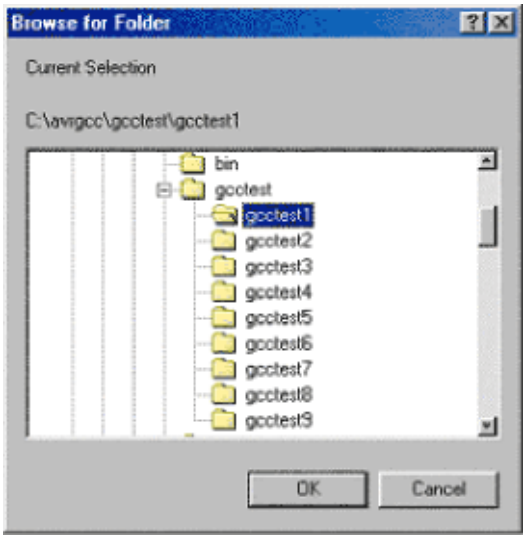
## avr-gccのAVR Studioでの新規プロジェクトの構築

AVR Studioで新規プロジェクト(New project)を開き、プロジェクトにファイルを追加して、プロジェクトの構築に備えてGNU make.exeに必要な手順を与えます。avrfreaksがリリースするavr-gcc に含まれるVolker Othのgcctestファイルが例として参考になります。これらのファイルはインストールしたフォルダ下のgcctestフォルダにあります。

### AVR Studio 3.53での新規プロジェクト作成

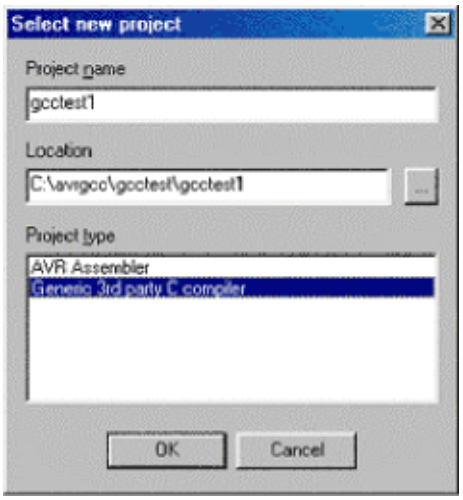
AVR Studioのメニューバーから Projectを選択し、プロジェクトメニューからNewを選択すると、Select new project ダイアログが現れます。




新規プロジェクト名をProject nameに入力し、Locationテキスト ボックス右側の閲覧(ブラウズ)ボタンをクリックしてプロジェクトフォルダ(この場合 gcctest1)を選択します。選択されたプロジェクト フォルダは、avr-gcc及びAVR Studioにより生成されたすべてのファイルの出力場所になります。これはユーザのすべてのプロジェクト ファイルも同様に保持する場所であることを意味します。

これら操作時、OKボタンをクリックする前にProjyect typeでGeneric 3rd party compilerが選択されていることを確認します。AVRGCCはGeneric 3rd party compilerに該当します。この選択はAVR Studioのマニュアルで上級者のみに推奨されていますが、これを選択します。




OKボタンのクリックで、このプロジェクトを保存します。

### プロジェクトの概念とGNU make ユーティリティの使用

各プロジェクトに新しいメイク ファイルを使用するための割り当て概念に近づけるために、各プロジェクト毎にAVR Studioのプロジェクトを使用して割り当てます。すべてのプロジェクトファイルは同一フォルダ内に保持します。このフォルダは上記の手順内で選択したフォルダになります。

これはAVR Studioが Generic 3rd party compilerの適切な動作をサポートするために必要なことです。

配布版avr-gcc付属のGNU makeツールは、プロジェクトをMCU内に取得するための完全なhexファイルにコンパイル、リンクするためにメイク ファイル内の規定を使用します。これは作業を系統立てる、整然且つ明瞭な判り易い方法です。

メイク ファイル用テンプレートとして、gcctest群内に含まれるメイク ファイルを使用します。これらはgcctestフォルダ内の各メイク ファイルのことです。これらは殆ど同じですが、この内の幾つかは僅かの拡張 (gcctest1からgcctest2へコピーしないなど)が成されています。

注：gcctest1～9を含むAVR freaksの配布版をダウンロードしなかった場合、本資料の指示に従ってメイク ファイルを編集し、比較対照を行わなければなりません。

### プロジェクトへの ファイル登録 (追加)

プロジェクトを構成するファイルを登録する必要があります。この小さなテストプロジェクトの場合には、以下が必要です。

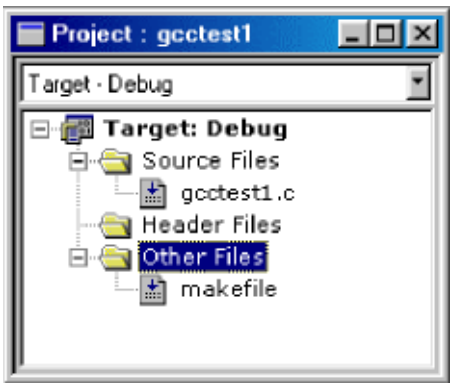
- いくつかの ソース コード
- メイク ファイル

Cの ソース ファイル gcctest1.cをプロジェクトに含めることから始めます。

1. AVR Studioのプロジェクト ウィンドウ内の Source Files上で右 クリックします。
2. Add fileを選択し、プロジェクト フォルダ内の gcctest1.cファイルを見つけます。
3. このファイルをダブル クリックするか開くボタンをクリックします。

このプロジェクト用のメイク ファイルを登録 (追加)します。

1. AVR Studioのプロジェクト ウィンドウ内の Other Files上で右 クリックします。
2. Add fileを選択し、makefileファイルを見つけます。見つからない場合は、ファイルの種類のドロップ ダウン リスト内の All filesを選択します。
3. このファイルをダブル クリックするか開くボタンをクリックします。




プロジェクトに登録されたすべてのファイルは、プロジェクト ウィンドウのフォルダ ツリー内の対応表示のダブル クリックで、そのファイルを開き編集ができます。

## メイク ファイル (makefile)

avr-gccをインストールしたフォルダの サブフォルダ avrfreaks内に含まれるVolker Othによるテンプレート メイク ファイルの内容を以下に示します。

```
# Simple Makefile Volker Oth (c) 1999
# edited by AVRfreaks.net nov.2001

########### change these lines according to your project ##################

# put the name of the target mcu here (at90s8515,at90s8535,attiny22,atmega603 etc.)
            MCU = at90s8515

# put the name of the target file here (without extension)
            TRG = gcctest1

# put your C sourcefiles here
            SRC = $(TRG).c

#put additional assembler source file here
            ASRC =

#additional libraries and object files to link
            LIB =

#additional includes to compile
            INC =

#compiler flags
            CPFLAGS = -g -Os -Wall -Wstrict-prototypes -Wa,-ahlms=$(<:.c=.lst)

#linker flags
            LDFLAGS = -Wl,-Map=$(TRG).map,--cref

########### you should not need to change the following line #############
include $(AVR)/avrfreaks/avr_make

###### dependecies, add any dependencies you need here ###################
$(TRG).o : $(TRG).c
```

```
#簡素なメイク ファイル VolkerOth
(c) 1999 # 2001年 11月
AVRfreaks.netにより編集
## プロジェクトに応じてこれらの
行を変更 ##
#対象 AVRデバイス名

#対象 ファイル名 (拡張子を除く)

#C ソース ファイル名

#付加アセンブラ ソース ファイル名

# リンクする付加 ライブラリ、
オブジェクト ファイル名

# コンパイルする付加 インクルード
ファイル名

# コンパイラ用 フラグ

# リンカ用 フラグ

##次行の変更の必要はありま
せん。 ##

##従属物 (必要な従属物を追
加 ) ##
```

多分、編集しなければならない行は以下のものだけです。

- **MCU = at90s8515**の行。使用するAVRデバイスが反映されるように編集します。
- **TRG = gcctest1**の行。これは対象名です。この対象名は希望する何にでも変更できますが、gcctest1は適切な名前です。このgcctest群では、ソース ファイルが核となる対象名と同一名を持つ必要があるようにメイク ファイルが記述されています。そうしない場合は、makeがソース ファイルを全く見つけられないことになります。このことは次の行から理解できます。

  **SRC = $(TRG).C**

  注 :対象名には何れの拡張子も使用してはいけません。

メイク ファイル中の **include $(AVR)/avrfreaks/avr_make**行に注意してください。この行は、メイク手順のための多くの従属物やコマンドを含むavr-makeファイルからの処理を行います。このファイルはAVR freaksからの配布版avr-gccに含まれています。このファイル内容の概要は、本資料の付録 Aを参照してください。

## 連結動作のための設定

ここまでで、すべての実行準備が整った、開かれたﾌﾟﾛｼﾞｪｸﾄとｿｰｽ ﾌｧｲﾙ、ﾒｲｸ ﾌｧｲﾙがありますが、これらすべてを共に動作させるには、まだ少し作業が残っています。これらの段階完了時、後のための比較的簡単な作業継続方法が得られます。

この時点で了解しておかなければいけない幾つかを以下に示します。

- AVR Studioはavr-gccに対して関与しません。これはｺﾝﾊﾟｲﾙ作業開始後、AVR Studioは何処で何をするかについては関与しないということです。
- ﾌﾟﾛｼﾞｪｸﾄ形式としてGeneric 3rd party compilerを選択した場合、何れの ｲﾍﾞﾝﾄ(各内部 ﾌﾟﾛｸﾞﾗﾑ)も初期化されませんので、結果的にAVR Studioはｺﾝﾊﾟｲﾗや使われ方を不意に知ることとなります。
- avr-gccは、ｱｾﾝﾌﾞﾗ、ﾘﾝｶ及びunixﾂｰﾙのような他のGNU AVRﾂｰﾙと同様にｺﾏﾝﾄﾞ行から実行します。これらは快適なﾕｰｻﾞｲﾝﾀｰﾌｪｰｽを持っていませんが、ｺﾏﾝﾄﾞ行から実行されるmakeﾌﾟﾛｸﾞﾗﾑにより、最も効果的に制御されます。

それではどうやってAVR Studioにavr-gccを使用させるかです。
やり方は、

- 環境変数によりAVR Studioがｸﾞﾉｰﾂｰﾙへの ﾊﾟｽを認識できるようにすること。
- 正しいﾒｲｸ ﾌｧｲﾙでmakeを実行すること。

で、従って以下となります。

- 正しいﾌｫﾙﾀ位置を指示し、makeを実行する batﾌｧｲﾙを記述します。
- 対応するﾌｧｲﾙをAVR Studioに知らせます。

この手順はWin9xとWin2000/XPで僅かに異なります。

注 :以下はavr-gccがc:\Avrgccにｲﾝｽﾄｰﾙされている場合です。

### Windows9x

このﾌｧｲﾙもavr-gccをｲﾝｽﾄｰﾙしたﾌｫﾙﾀ下のavrfreaksﾌｫﾙﾀ内にあります。ﾃｷｽﾄ ｴﾃﾞｨﾀで開き、以下の行で構成されるﾌｧｲﾙを作成します。

```
@echo -------- begin --------
@set AVR=c:\AVRGCC
@set CC=avr-gcc
@set PATH=c:\AVRGCC\bin
make %1
@echo -------- end --------
```

このﾌｧｲﾙをgcc_cmp.batとして既知のWindowsﾊﾟｽ内 (例えばc:\windows)に保存します。

このﾌｧｲﾙではmakeﾕｰﾃｨﾘﾃｨ用に幾つかの重要な環境変数を設定し、makeを実行します。%1引数はAVR Studioにより設定されるﾌﾟﾛｼﾞｪｸﾄﾌｫﾙﾀへのﾊﾟｽです。既定のmakeﾕｰﾃｨﾘﾃｨは、このﾌｫﾙﾀ内でﾌｧｲﾙを検索し、makefileを呼び出します。

### Windows2000/XP

Win2000では、ｺﾝﾊﾟｲﾙを開始するためにstartﾌｧｲﾙが必要になります。これらのﾌｧｲﾙの両方とも、avr-gccをｲﾝｽﾄｰﾙしたﾌｫﾙﾀ下のavrfreaks\win2000ﾌｫﾙﾀ内にあります。

```
@echo -------- begin --------
@start /MIN /wait cmd /c gcc_cmp2.bat %1
@type c:\tmpout.txt
@del c:\tmpout.txt
@echo -------- end --------
```

このﾌｧｲﾙをgcc_cmp.batとして、c:\winntまたは既知の利用可能なﾊﾟｽ内に保存します。

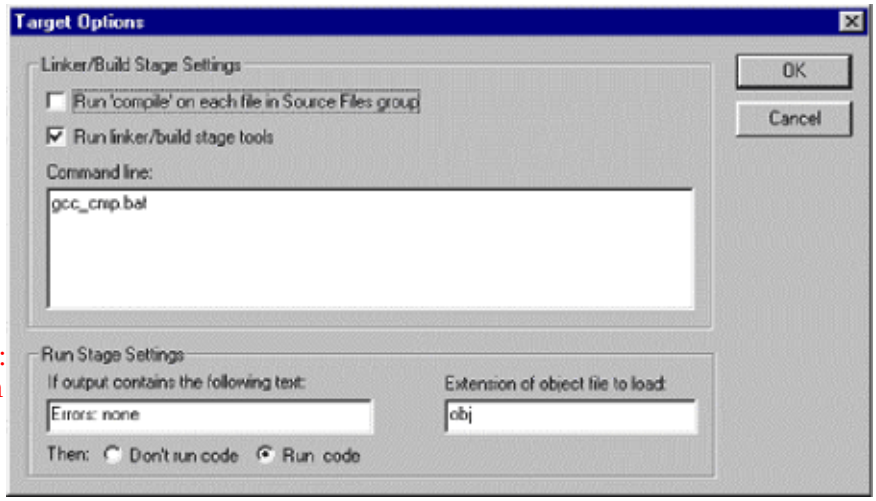
次に以下の行で構成されるﾌｧｲﾙを作成します。

```
@set AVR=c:\avrgcc
@set CC=avr-gcc
@set PATH=c:\avrgcc\bin
make.exe %1 >c:\tmpout.txt 2>&1
```

このﾌｧｲﾙをgcc_cmp2.batとして、上記ﾌｧｲﾙと同一ﾌｫﾙﾀに保存します。この起動設定により、make.exeが開始され、ｺﾝﾊﾟｲﾗ出力表示は直接ﾌｧｲﾙに一時保存された後、AVR Studioのｺﾝﾊﾟｲﾗ出力として表示され、この一時ﾌｧｲﾙはその後削除されます。

ここは重要な段階です。
AVR Studioの target optionsを設定します。

- ﾌﾟﾛｼﾞｪｸﾄｳｨﾝﾄﾞｳ内のTarget:debugを右ｸﾘｯｸし、ﾎﾟｯﾌﾟｱｯﾌﾟﾒﾆｭｰからsettingsを選択します。
- Run "compile" ･･･のﾁｪｯｸを解除します。
- Run linker/build ･･･をﾁｪｯｸします。
- Command line:ﾃｷｽﾄﾎﾞｯｸｽに先に作成したﾊﾞｯﾁﾌｧｲﾙ名 gcc_cmp.batを入力します。
- Run Stage Settings欄のThenをRun codeに切り替え、If output contains the following text:ﾃｷｽﾄﾎﾞｯｸｽにErrors:noneと入力し、Extension of object file to load:ﾃｷｽﾄﾎﾞｯｸｽにobjと入力します。



## AVR Studioとavr-gccでの最初のﾌﾟﾛｼﾞｪｸﾄ構築

ここまでで全ての実行準備が整いました。AVR Studioのﾌﾟﾛｼﾞｪｸﾄｳｨﾝﾄﾞｳ内のmakefileｱｲｺﾝをﾀﾞﾌﾞﾙｸﾘｯｸし、前記別項で示されたように入力されているかを確認します。特に対象名を設定する行が正しいか注意してください。以下の行からも明らかなように、対象名はCのｿｰｽ ﾌｧｲﾙ名と同一でなければいけません。

ﾌﾟﾛｼﾞｪｸﾄを構築するには、

- AVR Studioのﾌﾟﾛｼﾞｪｸﾄｳｨﾝﾄﾞｳ内のTarget:debugを右ｸﾘｯｸし、ﾎﾟｯﾌﾟｱｯﾌﾟﾒﾆｭｰのbuildを選択します。

これまでの全ての段階を正しく完了しているならば、このﾌﾟﾛｼﾞｪｸﾄは正しく構築される筈です。AVR Studio内部のｳｨﾝﾄﾞｳに表示されるmake出力を見て、ｴﾗｰが報告されていなければ全て完了です。

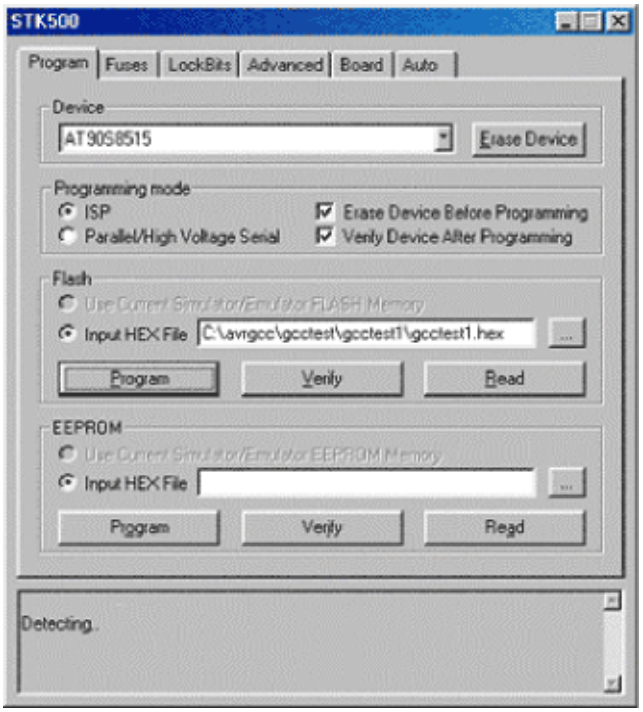
```
-------- begin --------

C:\avrgcc\gcctest\gcctest1\>make
avr-gcc -c -g  -Os -Wall -Wstrict-prototypes -Wa,-ahlms=gcctest1.lst -mmcu=at90s8515 -l. gcctest1.c -o gcctest1.o
avr-gcc  gcctest1.o  -Wl,-Map=gcctest1.map,--cref -mmcu=at90s8515 -o gcctest1.elf
avr-objcopy -O avrobj -R .eeprom gcctest1.elf gcctest1.obj
avr-objcopy -O ihex -R .eeprom gcctest1.elf gcctest1.rom
elfcoff gcctest1.elf coff gcctest1.cof gcctest1.sym
Ended
cp coff\gcctest1.cof .
cp coff\\*sym .
cp coff\\*S .
avr-objcopy -j .eeprom --set-section-flags=.eeprom="alloc,load" --change-section-lma .eeprom=0 -O ihex gcctest1.elf gcctest1.eep
Errors: none
-------- end --------
```

## STK500でのAVRﾌﾟﾛｸﾞﾗﾐﾝｸﾞ

STK500開発ｷｯﾄ梱包を解き、ｲﾝｽﾄｰﾙし、障害対策の手助けを行うことはこの資料の範囲外です。この件で問題を抱えている場合、2001年12月にSean Ellisにより書かれたAVR freaks net上の記事を参照してください。但し、その前に以下の項目の確認を常に行ってください。

- STK500が正しく接続され、電源が入っていることを確認します。
- 本ﾌﾟﾛｼﾞｪｸﾄ用に10ﾋﾟﾝ ｼﾞｬﾝﾊﾟｹｰﾌﾞﾙでPORTBとLEDSｺﾈｸﾀを接続します。

AVRﾃﾞﾊﾞｲｽ内に新しく作成したｺｰﾄを設定するために、AVR Studioのﾂｰﾙﾊﾞｰ上のAVRﾁｯﾌﾟ表示ｱｲｺﾝをｸﾘｯｸするか、ﾒﾆｭｰﾊﾞｰのToolsのﾒﾆｭｰからSTK500/..を選択し、STK500ﾀﾞｲｱﾛｸﾞを開きます。




- 多分、多くの設定が既定値のままになります。
- ﾃﾞﾊﾞｲｽに正しいﾊﾞｲﾅﾘが設定されることを確認します。ﾀﾞｲｱﾛｸﾞﾎﾞｯｸｽのFlash欄にﾌﾟﾛｼﾞｪｸﾄﾌｫﾙﾀ内のgcctest1.hexを指定します。閲覧ﾎﾞﾀﾝでの選択で対応ﾌｧｲﾙが見えない場合は、ﾌｧｲﾙ形式のﾄﾞﾛｯﾌﾟﾀﾞｳﾝﾘｽﾄからother filesを選択します。
- Programﾎﾞﾀﾝをｸﾘｯｸします。
- ﾀﾞｲｱﾛｸﾞ下側のﾃｷｽﾄﾎﾞｯｸｽの出力で、全てが正常に完了したことを確認します。

STK500基板上のLEDは、再びｸﾘｽﾏｽがやって来たように点滅する筈です。正常に動作しない場合、ｺﾝﾊﾟｲﾗの最適化ﾚﾍﾞﾙに起因する可能性があります。

### メイク ファイルの修正

メイク ファイルの コンパイラ フラグ行を見てみます。

```
#compiler flags
    CPFLAGS = -g -Os -Wall -Wstrict-prototypes -Wa,-ahlms=$(<:.c=.lst)
```

-Oフラグは コンパイラの最適化 レベルを示します。この設定はかなり厳しい設定になっています。Cの ソース ファイルを調べると、LED出力間が ループ カウンタによる遅延で行われていることが判ります。この タイマは特定の遅延を保証するための複雑な タイマではありません。このため、最適化により、この遅延が消滅した訳です。コンパイラは、この遅延を無用と判断し、排除したという訳です。

この遅延を動作させるには、AVR Studioの プロジェクト ウィンドウ内のmakefileアイコンを ダブル クリックし、メイク ファイルを編集します。次の行を変更します。

```
#compiler flags
    CPFLAGS = -g -Os -Wall -Wstrict-prototypes -Wa,-ahlms=$(<:.c=.lst)
```

⇓

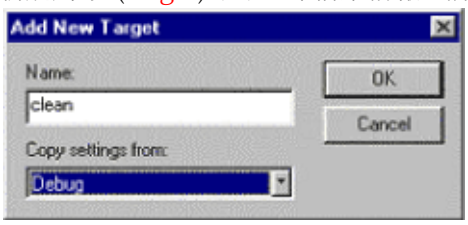
```
#compiler flags
    CPFLAGS = -g -O1 -Wall -Wstrict-prototypes -Wa,-ahlms=$(<:.c=.lst)
```

これで最適化 レベル 1に設定されます。変更前の最適化指示 sは コード サイズ最適化を指示しています。より詳細な情報は avr-gccの manを参照してください。

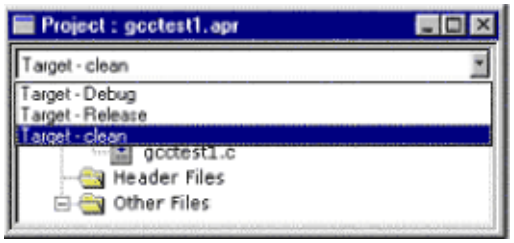
### 構築情報の解除

重要注意：新しい設定で構築を行う場合、直前の構築情報を削除する必要があります。これを行わないと、makeは メイク ファイル内の変更を検出するように設定されていないため、再構築の必要がないと判断してしまいます。これらの ファイルを手動で解除する代わりに、AVR Studio内で新規対象 (Target)を追加し、構築情報を消去します。

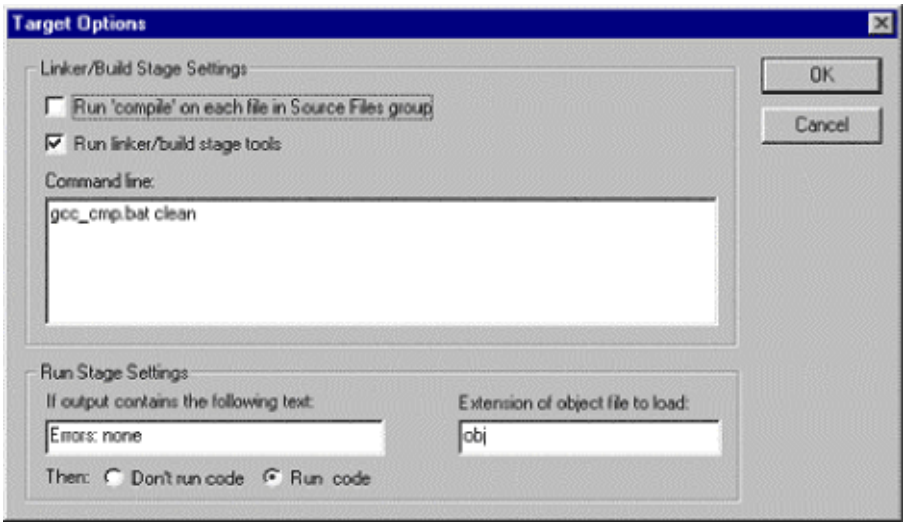
- AVR Studioの プロジェクト ウィンドウ内の Target: debugを右 クリックし、メニューから Target->add... を選択すると、Add New Targetダイアログが現れます。Name欄に新規対象として cleanを入力し、Copy settings from欄の ドロップ ダウン リストから Debugを選択します。




- プロジェクト ウィンドウの対象 ドロップ ダウン リストから対象 cleanが利用可能になります。
- この対象 (Target: clean)を選択し、右 クリックから settings..を選択します。




- これで対象 Debugから設定が継承されたことが判ります。バッチ ファイル名の後にキーワード cleanを追加します。




- 対象 cleanを構築します。

これで メイク作業での作成物が一掃されます。cleanは avr-gcc内に含まれる avr-make メイク ファイル内で定義された対象 (Target)です。この ファイルについては付録 Aを参照してください。

- 対象をDebugに戻します。
- 再構築後、AVRに プログラム (書き込み)し、結果を確認してください。

何れかの変更を行わずに一度を越えてこの プロジェクトの再構築を行おうとしても、単にmakeの構築が失敗するだけです。これはmakeがmakefileを通して行われ、如何なる時も プロジェクトの再構築に必要な部分のみを再構築するように設定されているためです。そのため、makeが最新の出力 ファイルでないと認識した場合に、すべてのものを再構築すると判断されます。しかし、メイク ファイル (makefile)自体は個別のものとして プロジェクトに含まれていませんので、この ファイルの変更では自動的に再構築されるようにはなりません。然しながら、ソース ファイルの変更には対応しますので、充分合理的です。

- プロジェクトを保存します。

## AVR Studioでの .coffファイルによるﾃﾞﾊﾞｯｸﾞ

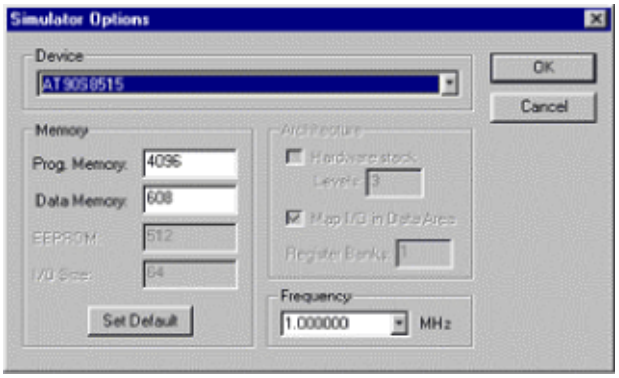
avr-gccが生成する .objファイルには、AVR Studioで行うﾃﾞﾊﾞｯｸﾞ中の変数監視に必要な情報が含まれていません。標準 gccの .elfﾌｧｲﾙには、この全ての情報が含まれていますが、AVR Studioがこの ELF形式をｻﾎﾟｰﾄしていません。しかし、変数監視ができるCOFF形式がｻﾎﾟｰﾄされています。

これに対応するAVR freaks net登録ﾕｰｻﾞの Flavio Gobberi作の ELF→COFFｺﾝﾊﾞｰﾀがあります。これは利便性を考慮し、20011203a以降の AVR freaksの配布版 avr-gccに統合され含まれています。これらの配布版では作業ﾌｫﾙﾀに .coffﾌｧｲﾙが出力されるように設定されています。単にAVR Studio内にこの .coffﾌｧｲﾙを開くことで、ﾌﾞﾚｰｸﾎﾟｲﾝﾄや監視項目の追加や削除が行えます。

参考：Elf2coffは AVR freaks net (http://www.avrfreaks.net/AVRGCC/download.php)からでも取得できます。この場合、そのままでは avr-gccで使用できませんので、付録 Bを参照して設定を行わなければなりません。

例：gcctest1ﾌﾟﾛｼﾞｪｸﾄの構築成功後、ﾌﾟﾛｼﾞｪｸﾄﾌｫﾙﾀ内に出力された .coffﾌｧｲﾙを開きます。

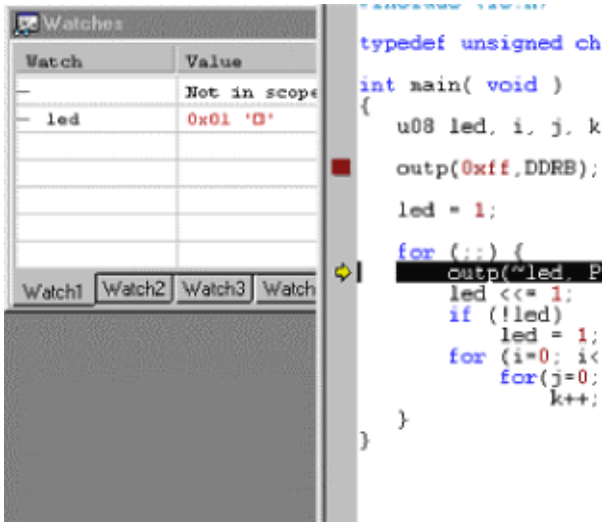
- 少しの擬似命令と多くのｱｾﾝﾌﾞﾘｺｰﾄﾞの .coffの内容が表示される新規ｳｨﾝﾄﾞｳが現れ、ﾌﾟﾛｼﾞｪｸﾄｳｨﾝﾄﾞｳが隠されます。
- Simulator optionsﾀﾞｲｱﾛｸﾞが現れますので、ﾃﾞﾊﾞｲｽ(Device)、ｸﾛｯｸ周波数 (Frequency)を選択します。




- ﾌﾟﾛｼﾞｪｸﾄを再開し、Cの ｿｰｽ ﾌｧｲﾙをﾀﾞﾌﾞﾙｸﾘｯｸして開きます。
- main関数の次の行までｽｸﾛｰﾙします。

```
outp(0xff,DDRB);
```

- F9ｷｰを押下し、ここにﾌﾞﾚｰｸﾎﾟｲﾝﾄを設定します。
- ﾒﾆｭｰﾊﾞｰのWatchからAdd watchを選択し、Watchesｳｨﾝﾄﾞｳを開き、右ｸﾘｯｸで ledを監視項目に追加します。
- Shift+F5ｷｰ押下でﾃﾞﾊﾞｯｸﾞのﾘｾｯﾄを行うと、有効なｳｨﾝﾄﾞｳが coffﾌｧｲﾙｳｨﾝﾄﾞｳに変更されます。F5ｷｰ押下で実行すると、有効なｳｨﾝﾄﾞｳがｿｰｽ ｺｰﾄﾞｳｨﾝﾄﾞｳに変更され、上記で指定した行で一時停止します。
- F10ｷｰ押下で 1命令実行を行うと、監視変数 ledが 0x01・・・と変化します。




以上が avr-gccとAVR Studioによる AVRﾌｧﾐﾘの開発法です。本資料の各段階を通して提供された設定で、暫くの間は充分でしょう。例となる gcctest1～9の実行や、http://www.avrfreaks.net/AVRGCC/からﾀﾞｳﾝﾛｰﾄﾞできるRich Neswoldの AVRﾏｲｸﾛｺﾝﾄﾛｰﾗ用 GNU開発環境 (GNU development Environment for AVR micro-controller)が良き導入、参考となります。

多くの ﾌﾟﾛｼﾞｪｸﾄを組み合わせる作業を開始する場合や、より多くの経験を積むと、この設定の変更や拡張が必要になるかもしれません。その時には多分、必要な変更や調整の方法についての多くを知ることになります。

最後に、

- AVRﾏｲｸﾛｺﾝﾄﾛｰﾗ用 GNU開発環境 (GNU development Environment for AVR micro-controller)を未だ入手していない場合は、直ぐに入手してください。
- AVR freaks netの AVRﾌｫｰﾗﾑを度々訪れてください。



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。

## 付録A：avr-makeﾌｧｲﾙ

以下のﾘｽﾄはavr-makeﾌｧｲﾙの内容です。このﾌｧｲﾙは2000年1月にVolker Othが作成したmake1～2ﾌｧｲﾙを元に作られました。このﾌｧｲﾙは、配布版avr-gccがc:\avrgccﾌｫﾙﾀにｲﾝｽﾄｰﾙされた場合、c:\avrgcc\avrfreaksﾌｫﾙﾀにあります。簡単な参照情報が(BLOCK 1･･･)などのBLOCK識別で、このﾌｧｲﾙ内に挿入されています。各ﾌﾞﾛｯｸの説明は後で行います。これらを通して読むことで、この動作の幾つかの概念を容易に得ることができます。

```
#------------------ START OF FILE ------------------------------------------------
# GCC-AVR standard Makefile part 3
# Based on Volker Oth's makefiles of jan.2000
# Modified and merged by AVRfreaks.net for smoother integration with AVR Studio,
# and easier comprehension for the average user (nov.2001). Minor errors corrected.
# -------------------------------------------------------------------------------

###### BLOCK 1) define some variables based on the AVR base path in $(AVR) #######

    CC = avr-gcc
    AS = avr-gcc -x assembler-with-cpp
    RM = rm -f
    RN = mv
    OUT = coff
    ELFCOF = elfcoff
    CP = cp
    BIN = avr-objcopy
    SIZE = avr-size
    INCDIR = .
    LIBDIR = $(AVR)/avr/lib
    SHELL = $(AVR)/bin/sh.exe

###### BLOCK 2) output format can be srec, ihex (avrobj is always created) #######

    FORMAT = ihex

###### BLOCK 3) define all project specific object files ######

    OBJ = $(ASRC:.s=.o) $(SRC:.c=.o)
    CPFLAGS += -mmcu=$(MCU)
    ASFLAGS += -mmcu=$(MCU)
    LDFLAGS += -mmcu=$(MCU)

###### BLOCK 4) this defines the aims of the make process ######

all: $(TRG).obj $(TRG).elf $(TRG).hex $(TRG).cof $(TRG).eep $(TRG).ok

###### BLOCK 5) compile: instructions to create assembler and/or object files from C source ######

%.o : %.c
        $(CC) -c $(CPFLAGS) -I$(INCDIR) $< -o $@
%.s : %.c
        $(CC) -S $(CPFLAGS) -I$(INCDIR) $< -o $@

###### BLOCK 6) assemble: instructions to create object file from assembler files ######

%.o : %.s
        $(AS) -c $(ASFLAGS) -I$(INCDIR) $< -o $@

###### BLOCK 7) link: instructions to create elf output file from object files ######

%.elf: $(OBJ)
        $(CC) $(OBJ) $(LIB) $(LDFLAGS) -o $@

###### BLOCK 8) create avrobj file from elf output file ######

%.obj: %.elf
        $(BIN) -O avrobj -R .eeprom $< $@
```

次ﾍﾟｰｼﾞへ続く

```
###### BLOCK 9) create bin (.hex and .eep) files from elf output file ######

%.hex: %.elf
	$(BIN) -O $(FORMAT) -R .eeprom $< $@

%.eep: %.elf
	$(BIN) -j .eeprom --set-section-flags=.eeprom="alloc,load" --change-section-lma .eeprom=0 -O $(FORMAT) $< $@

%.cof: %.elf
	$(ELFCOF) $< $(OUT) $@ $*.sym
	$(CP) $(OUT)\$@ .
	$(CP) $(OUT)\\*sym .
	$(CP) $(OUT)\\*S .

###### BLOCK 10) If all other steps compile ok then echo "Errors: none" ######

%ok:
	$(SIZE) $(TRG).elf
	@echo "Errors: none"

###### BLOCK 11) make instruction to delete created files ######

clean:
	$(RM) $(OBJ)
	$(RM) $(SRC:.c=.s)
	$(RM) $(SRC:.c=.lst)
	$(RM) $(TRG).map
	$(RM) $(TRG).elf
	$(RM) $(TRG).cof
	$(RM) $(TRG).obj
	$(RM) $(TRG).a90
	$(RM) $(TRG).hex
	$(RM) *.bak
	$(RM) *.log

size:
	$(SIZE) $(TRG).elf

#----------------- END OF FILE ---------------------------------------------------------------
```

注：ｺﾏﾝﾄﾞに先行するﾀﾌﾞ(TAB)は、makeﾕｰﾃｨﾘﾃｨが依存関係からｺﾏﾝﾄﾞを認識するための重要な要素ですので、削除しないように注意してください。逆に変数宣言内の余分な空白（ｽﾍﾟｰｽ）などは削除してください。これらは、実行できない、またはﾊﾟﾗﾒｰﾀが認識されないなどの原因となる可能性があります。

## 内容説明

### Block 1

この部分では、make動作で必要なユーティリティやフォルダの幾つかの付加的な位置を指定します。c:¥avrgccにインストールした場合の変数AVRを呼び出しに対する再呼び出しは次のように記述します。

```
LIBDIR = $(AVR)/avr/lib
```

これは単に変数LIBDIRにc:¥avrgcc¥avr¥libを設定するだけです。この部分は設定における基準点を変更する唯一の部分です。

### Block 2

バイナリ出力形式オプションとして、INTEL HEX形式を設定します。

### Block 3

最初の行は、含まれるすべてのソースファイル(アセンブリ、C言語の両方)から作成されるオブジェクト名を定義します。これらはmakefile内で使用されます。同様に$(MCU)変数は、makefile内で設定されるAVRマイクロコントローラの種類(例えばAT90S8515)です。

### Block 4

この行では生成する一連の対象ファイル(.obj, .elf, .hexなど)を指示します。ファイルの順番は任意ですが、メイクファイル内の次の部分は違います。

### Block 5

この部分は2行で1つの意味を表し、最初の行が対象:依存元を示し、規定で呼び出される行です。次の行がmake規定のコマンドです。すべてのコマンドに先行するタブ(TAB)は、メイ動作における識別ですので、このタブがない場合、そのコマンドは省かれ実行されません。

ここではすべての.cファイルに対応する.oファイルの依存を規定します。%はワイルドカード(全てを表現)です。従って、次行はすべての.cファイルを対応する.oファイルにコンパイルすることを指示します。

```
%.o : %.c
    $(CC) -c $(CPFLAGS) -I$(INCDIR) $< -o $@
```

$<と$@はmakeの自動化変数で、各々依存元ファイル名、本規定の対象ファイル名の意味になります。本規定の対象はオブジェクトファイル.oになります。従って、本行実行時にソースgcctest1.cがgcctest1.oにコンパイルされます。

### Block 6

これは上記同様のアセンブリ言語のソースファイル用の規定です。

注:avr-gccがCのソースファイルのコンパイル時に$(AS)変数として他の擬似命令と共に呼び出すため、他のフラグが含まれる場合があります。

次からの数ブロック(Block)は同様の規定です。

### Block 10

すべての規定はファイル内に現れた順番で順に処理されます。処理が中止せず、ここに到達したならば、すべてが完了したことになります。従ってAVR Studioに Error: noneを通知します。この部分はAVR Studioが直前までの段階を正常に完了したことを認識するために必要です。更にavr-size.exeを呼び出し、実行します。この出力はelfファイルの容量ですので、これからプログラム(コード)変更による結果のコード容量が分かります。

### Block 11

これらの幾つかの対象物は現実の対象ファイルを持たないため、完全な型規則の対象:依存元形式に対応していません。これらの規定の結果として、現実のファイルは構築されません。このような規定を偽規定と呼びます。これらの呼び出し時、この規定に対応する全てのファイルが削除されますので、構築の最初などに使用します。この部分は自動的に実行されませんので、make cleanのように明示的に呼び出し、実行します。

Block4の記述が完全なメイク処理の定義となる既定で、特別な呼び出しがない場合、常に実行されます。これも擬似対象です。依存関係の1つとしてcleanを含めることで、中間ファイルの削除などが行えます。

注:変更(例えばmakefile内のコンパイラフラグの変更)を行った場合、完全な再構築が必要です。しかし、makefileは何れのプロジェクトの依存関係も共通に考慮しないので、これに何れの規定も含めることはできません。

例えば、上記のBlock5と6内にアセンブラとオブジェクトファイルに対し、依存関係として含めると、全部のプロジェクトは常にmakefileが変更されたとして(例えば、いつもmakefileがソースファイルやオブジェクトファイルより新しいとmakeに通知され)構築されます。これは普通ではありません。

makefile内を変更した場合に、最も簡単で安全なのは、全てのプロジェクトに対してmake cleanを実行してフォルダを一掃し、makeで構築し直すことです。windows環境で作業中の場合は、makeの手動操作により、すべてのファイル(Block11の全ての拡張子)を削除することもできます。

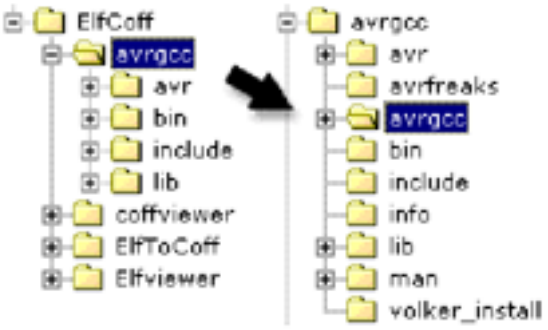
## 付録 B:AVR Studioとmakeでの elf2coff用設定

gcc標準の .elfファイルは、デバック時の変数監視用の全情報を含んでいますが、AVR Studioは ELF形式をサポートしていません。しかし、変数監視可能な COFF形式がサポートされています。.elfから .cofへの変換は、AVR freaksユーザの Flavio Gobber作の Elf2Coff ユーティリティで行えます。この ファイルは、デバック用に AVR Studio内へ直接読み込めます。

AVR Studioと avg-gccでの Elf2Coff実行用の設定は、どうすればよいかが判れば、非常に難しい訳ではありません。これが難しいようでしたら、やり方の情報 (知る限り、AVR freaks netの AVR-GCCフォーラムの単独の ソースを入手してください。これは編集されており、快適さと便利さを提供します。

注 : AVR freaks netから 2001年 12月 3日以降の配布版 avr-gccを既にダウンロードしている場合は、Elf2Coffが既に統合され含まれていますので、これ以降を続行する必要はありません。Coffファイルは プロジェクト フォルダへ自動的に出力されます。以降の使用方法は、AVR Studioの .coffファイルでの デバックを参照してください。

### 設定方法




- www.AVRfreaks.net/AVRGCCから Elf2Coffをダウンロードします。
- 何れかの フォルダに解凍 (Unzip)します。
- 解凍後構成を調べます (右図参照 )。重要な部分は avrgccフォルダです。
- この フォルダ下の サブフォルダも含め、avr-gccを インストールした フォルダ (例えば c:\avrgcc\)に コピーします。4つの サブフォルダを含む新しい フォルダ (例えの場合、c:\avrgcc\avrgcc\)ができます。
- \elfcoff\avrgcc\binフォルダの elfcoff.exeファイルを、avr-gccを インストールした フォルダ下の \binフォルダ (例えば c:\avrgcc\bin)内に コピーします。

avr-makeファイル内の幾つかを変更しなければなりません。(付録 A参照 )

- Block1内に次の行を追加します。(注 ) この部分の他の行に合せて タブ (TAB)を使用します。

```
ELFCOF = elfcoff
OUT = Coff
CP = cp
```

- Block4内に .cof対象 ファイル $(TRG).cofを追加します。追加後の行は次のようになります。

```
all: $(TRG).obj $(TRG).elf $(TRG).cof $(TRG).hex $(TRG).eep $(TRG).ok
```

- Block9内に次の新規定を追加します。(注 ) タブ (TAB)に注意。

```
%.cof: %.elf
	$(ELFCOF) $< $(OUT) $@ $*sym
	$(CP) $(OUT)\$@ .
	$(CP) $(OUT)\\*sym .
	$(CP) $(OUT)\\*S .
```

- 最後に Block11の clean:内に次行を追加します。

```
	$(RM) $(TRG).cof
```

注 : メイク ファイルにキーワードを追加する場合、不必要なすべての スペースを省くことに注意してください。不慮の タブや スペースは、make 作業が失敗する原因になります。この結果の コンパイル出力は以下のようになります。

```
-------- begin --------
C:\WINDOWS\Desktop\AVRprojects\testgcc\gcctest1>make
avr-gcc -c -g  -Os -Wall -Wstrict-prototypes -Wa,-ahlms=gcctest1.lst -mmcu=At90s8515 -I, gcctest1.c -o gcctest1.o
avr-gcc gcctest1.o  -Wl,-Map=gcctest1.map,--cref -mmcu=at90s8515 -o gcctest1.elf
avr-objcopy -O avrobj -R .eeprom gcctest1.elf gcctest1.obj
elfcoff gcctest!.eLf Coff gcctest1.cof gcctest1.sym
Ended
cp Coff\gcctest1.cof .
cp Coff\\*sym .
cp Coff\\*S .
avr-objcopy -O ihex -R .eeprom gcctest1.elf gcctest1.rom
avr-objcopy -j .eeprom --set-section-flags=.eeprom="alloc,load" --change-section-lma .eeprom=0 -O ihex gcctest1.elf gccte...
Errors: none
-------- end --------
```