

GNUを使ったSTM32の開発ツールの製作

山際伸一

本誌 2008 年 5 月号付属の ARM 基板を使って組み込みソフトウェア開発を行うには Keil 社のツール μ Vision を用いた。 μ Vision は、Cortex-M3 向けのコンフィグレーションや最適化などが自動的に行われるため、Cortex-M3 でのプログラム開発に適した環境である。今回は ARM 専用の開発ツールではなく、GNU が提供するフリーの開発環境を使い、プログラムをコンパイルし、バイナリ・ファイルをダウンロードして実行する方法について解説する。 (筆者)

1. Cortex-M3 を GNU ツールで動かす

● GNU 開発環境を用意する

まず、GNU (GNU's Not UNIX) ツールを用いた C 言語開発環境をそろえます。ここで用意する開発環境は Cygwin (GNU の開発ツールを含む UNIX のさまざまなフリー・ソフトウェアを Windows に移植したもの) で動作します。Cygwin のサイト (<http://www.cygwin.com/>) から Cygwin をインストールします。

次に、本誌の Web サイト (<http://www.cqpub.co.jp/dwm/>) から「download_files_yamagiwa.zip」をダウンロードします。そのファイルを適当なフォルダに解凍します。以下の四つのフォルダが含まれています。

① gcc_sample

後述する GNU ツールを用いた Cortex-M3 向けサンプル・プログラムです。

② gdb-stub

後述する GNU ツールを用いた Cortex-M3 向け GDB スタ

ブです。

③ GNU_tools_binary

ARM 開発向けにコンパイル済みの GNU ツールです。

④ GNU_tools_src

GNU ツールのソース・ファイルです。

ここでは、GNU ツールをセットアップするので、③ GNU_tools_binary フォルダの「arm-tools.tar.bz2」を Cygwin のホーム・ディレクトリ (デフォルトの Cygwin のインストールでは C:\cygwin\home\ユーザ名) にコピーします。Cygwin のコンソール画面を開き、以下のコマンドを実行します。

```
$ cd %usr%local
$ tar jxvf ~/arm-tools.tar.bz2
```

これで、コンパイル済みの C コンパイラ、アセンブラ、リンカなどが %usr%local\arm-tools フォルダに展開されます。これで開発環境の準備は完了です。

これから解説するサンプル・プログラムが、ダウンロード・ファイルの gcc_sample\cortex_m3 フォルダにあるので、そのフォルダをホーム・ディレクトリにコピーしておきます。

```
$ cp -R download_files_yamagiwa\gcc_sample
                                         %cortex_m3 %
```

先ほど展開した GNU 開発環境が正しく動作するかをテストするため、以下のコマンドを実行してみます。

```
$ cd cortex_m3
$ make clean
$ make
```

Keyword

μ Vision, GNU, Cygwin, C コンパイラ, アセンブラ, リンカ, GCC, T バリエーション, インテル HEX, モトローラ S, DfuSe, GDB スタブ, スタートアップ・ルーチン

ここで、GCCやGASなどのコマンドが実行されてエラーが発生しなければ開発の準備は完了です。

もし、何かしらの原因でGCCなどが実行できない場合には、**コラム1** (下記参照) のソースからのコンパイルを実行してください。

● GNU ツールで Cortex-M3 向けプログラムを開発する

ここではGNU ツールを使って、ARM 基板用に以下のような機能を持つアプリケーション・プログラムを開発する方法を解説します。

- ① A-D コンバータからの値を RS-232-C シリアル接続を介して1秒ごとに出力する
- ② RS-232-C シリアル接続からの入力文字が「0」のとき、ARM 基板上のLED1を消灯する
- ③ RS-232-C シリアル接続からの入力文字が「1」のとき、ARM 基板上のLED1を点灯する

①は無限ループで出力し続け、②と③の操作はいつでも受け付けるようにします。これを実装するためには割り込み機能を使う必要があります。

Keil 社の μ Vision では、GUIが必要な設定項目を表示しています。これだけの仕様が固まっていれば、チェックを忘れなければスムーズに開発が進みます。GNU ツールをベースとした開発では、ターゲットとなるCortex-M3プロ

セッサのアーキテクチャをきちんと理解していないと、メモリ・マップや初期化ルーチンなどを正しく記述できません。そこでまずは、Cortex-M3 向けのプログラムを作成する際の注意点を最初に考えてみます。

(1) Cortex-M3 および STM32 アーキテクチャの注意点

ARM 基板の STM32 に搭載されている Cortex-M3 プロセッサの注意点に関して考えてみます。

Cortex-M3 向けのプログラムを開発する際、これまでの ARM アーキテクチャと異なる以下の事柄があります。

① Thumb 命令のみを実行する

Cortex-M3 プロセッサは、ARM7-M と呼ばれる ARM プロセッサの派生CPUですが、サポートする命令はThumb、または、Thumb-2のみです。従って、GCCで開発をする際には、必ずThumb 命令を出力するようにコマンド・ライン・オプションを追加する必要があります。本稿ではThumb 命令を用いる方法を解説していきます。

② Thumb 命令の中でサポートされていないものがある

Thumb 命令を実行するとはいえ、Thumb 命令のいくつかはCortex-M3ではサポートされていません。例えば、BLX 命令 (32ビット・アドレス分岐+LRに振り番地を格納する命令) はサポートされておらず、BX 命令のみを用いて32ビット・アドレス分岐を実現する必要があります。

コラム1 クロス開発環境をコンパイルする

何らかの理由で arm-tools.tar.gz を展開したコマンドが動作しない場合、ソース・コードからクロス開発環境を作る方法を説明します。

本誌 Web ページから提供されるダウンロード・ファイル download_files_yamagiwa.zip を解凍した GNU_tools_src フォルダの中には、

```
binutils-2.16.tar.bz2
gcc-3.4.5.tar.bz2
insight-6.4.tar.bz2
```

の三つのファイルが含まれています。

これらのファイルをホーム・ディレクトリにコピーします。

リストAに示す手順でARM向けのクロス開発環境を構築します。クロス環境を構築する際には順番があり、binutilsというパッケージ、GCCパッケージという順で構築が必要です。さらに、GCCを構築するにはPATH環境変数にbinutilsで構築したコマンドへのパスを追加する必要があります。

最後に後に説明するInsightの構築をしています。

約1時間から2時間くらい静かに見守っていると、一連のコンパイルが完了します。

リストA ARM向けクロス開発環境の構築

```
#!/bin/sh
tar jxvf binutils-2.16.tar.bz2
cd binutils-2.16
./configure --target=arm-elf --
prefix=/usr/local/arm-tools
make
make install
cd ..

export PATH=/usr/local/arm-tools/bin:$PATH
tar jxvf gcc-3.4.5.tar.bz2
cd gcc-3.4.5
./configure --target=arm-elf
--prefix=/usr/local/arm-tools --enable-languages=c
make
make install
cd ..

tar jxvf insight-6.4.tar.bz2
cd src
./configure --target=arm-elf --
prefix=/usr/local/arm-tools/gdb
make
make install
cd ..
```