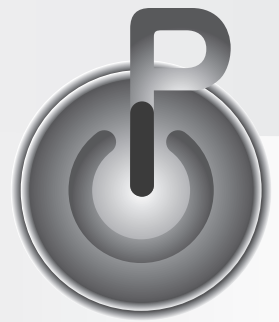


# 実践的 PowerPC 活用テクニック



## 第13回 GDB スタブの移植と GDB によるプログラムのデバッグ

坂井 弘亮

より本格的なプログラムの開発にはデバッグ環境も重要だ。ここでは PowerPC 用 GDB スタブを作成し、組み込みシステム開発評価キット (通称 BLANCA) のオプションとして発売されている、CPU カード/PowerPC (MPC5200) へ移植する。また作成した GDB デバッグ環境を使ってサンプル・プログラムを動かす。

(編集部)

組み込みシステムでは、ホスト・パソコンとターゲット・ボードをシリアル・ケーブルで接続してデバッグする「リモート・デバッグ」が一般的です。このためには、ターゲット・ボードにホスト・パソコンで動作する GDB と通信するプログラム「GDB スタブ」を実装する必要があります。

GDB スタブはシリアル通信によってホスト・パソコンと通信し、ホスト・パソコン上で動作している GDB からの命令 (GDB コマンド) を受信して必要な動作を行い、GDB に対して応答します。コマンドの内容はレジスタ値の取得や特定アドレスのメモリの値の読み書きなど、非常に単純なものばかりです。シンボルの解析などの複雑な動作は、ホスト・パソコン上の GDB が行います。このためスタブは非常に小さなプログラムとして実装できます。

たとえば、不正メモリ・アクセスなどの例外発生時には、例外ハンドラから GDB スタブを呼び出し、GDB スタブに処理を渡します (つまり、GDB スタブは割り込み処理の延長で動作する)。GDB スタブは例外発生をシリアル経由でホスト・パソコン上の GDB に通知し、自身はシリアル受信待ちのビジー・ループに入って GDB からの指示を待ちます。GDB は例外発生をユーザに通知し、ユーザのコマンド入力に応じて GDB スタブに指示を出し、GDB スタブからの応答を受けてユーザに応答を返します。

表1 gdb-6.8 に付属するスタブのサンプル・ファイル名と対応 CPU

i386-stub.c	i386
sh-stub.c	SH
m32r-stub.c	M32R
sparc-stub.c	SPARC
m68k-stub.c	680x0

### 1. GDB スタブの実装

#### ● GDB スタブのソース・コード

GDB のソース・コードには、各 CPU ごとの GDB スタブのサンプルが付属しています (表 1)。ライセンス条項については各ファイルの先頭部分を参照してください。GDB 自体のライセンスは GPL (General Public License) ですが、スタブに関しては組み込みの利用が前提のため、多くは別のライセンス表示になっているようです。

表 1 でわかるように PowerPC 用のスタブは付属していません。そこで i386 用のスタブである `i386-stub.c` をひな型として、PowerPC に移植することにします。

#### ● PowerPC 用スタブの作成

まずは PowerPC 用スタブとして `ppc-stub.c` を追加します。今回作成した PowerPC 用のスタブをリスト 1 に示します。PowerPC 対応のための変更点は次の通りです。

- レジスタの定義と保存方法
- 割り込みベクタ番号からシグナル番号への変換方法
- 例外発生時のホスト・パソコンへの例外通知方法
- ステップ実行時のレジスタ設定
- ブレークポイントでのトラップ命令実行

レジスタの種類は `regnames` で定義されます。当然ながらこの定義は CPU ごとに異なります。必要なレジスタと定義の順番は CPU ごとに決められており、GDB のソース・コードの `regformats` というディレクトリに記述があります。PowerPC 用は `reg-ppc.dat` というファイルに記述されているので、それを元にして定義しています。

`computeSignal()` では例外発生時にベクタ番号から GDB に通知するシグナル番号を計算しています。通常はメモリ・アクセス違反などでは SIGBUS か SIGSEGV とし