第2章

GDBの使い方からGDBスタブの移植の基本まで

GDBによるクロス・デバッグ環境の構築

プログラムのミスを見つけるためにはデバッガが必要になる．また，シリアル・ケーブルを使ってホストとターゲットを接続する場合，ターゲットにGDBスタブが必要になる．本稿ではGDBの使い方や，リモート・シリアル・プロトコルについて，さらにCPUに依存しない部分のGDBスタブの動作についても解説する．（編集部）

山際 伸一

第1章の解説で，ターゲットCPUで動作するプログラムを作成する環境は整いました．しかし，書いたプログラムがバグもなく一発で動作することはまずないでしょう．また，プログラムの挙動がどうも思い通りにならない，ということもあると思います．そのような場合，手軽にプログラムの実行とデバッグを可能にするのがGNUツールに含まれるデバッガのGDBです．

本稿ではGDBを使った組み込みプログラムの実行環境を構築し，これを利用するための基本事項を解説します．さらにアーキテクチャに依存しない部分のターゲットCPU側のROMモニタの作成方法についても触れます．

1. GDBによるデバッグ環境とは

まずはGDBによるプログラム実行環境を構築する上での基本事項を押さえましょう．

● GDBを使ったデバッグ環境の仕組み

GDBは，CPUに依存しないプログラム・デバッグ環境を作ろうという目標を掲げて作成されたデバッガ・ソフトウェアです．従って，階層化されたプログラム・ソースの最下層に位置する命令ニーモニックの定義や通信手段，通信プロトコルを変更するとどのようなCPUにも対応できるようになっています．

GDBはホスト・マシンの通信ポートを介して接続されたターゲット・マシンと通信することでプログラムのダウンロード，実行，停止を可能とします．ターゲット・マシンはホスト・マシンと通信するために決められたプロトコルを解釈できなければなりません．この通信プロトコルはJTAGのようなハードウェアで実装されたり，ROMモニタとしてターゲットCPUが実行している場合もあります．この通信プロトコルがCPUごとに異なると，CPUごとにGDBの最下位ルーチンの通信手順を作成しなければならず，多大な労力が必要となります．そこでGDBには，リモート・シリアル・プロトコルと呼ばれる通信プロトコルが定義されています．ターゲット・マシンがこの通信プロトコルに従う限り，GDBはどのようなCPUとも会話できるようになります．

通信ケーブルについても，たとえターゲットCPUがJTAGポートを持っているとしても，そのJTAGコマンドはベンダの未公開のものが多く，GDBのサポートは困難です（仕様が公開されているARMプロセッサのJTAGポートなどについては，オープン・ソースのJTAG制御ソフトウェアも存在するが…）．従って，RS-232-Cのシリアル・ケーブルで接続された通信方法を用いるのが，GDBでは当然の選択となっています．

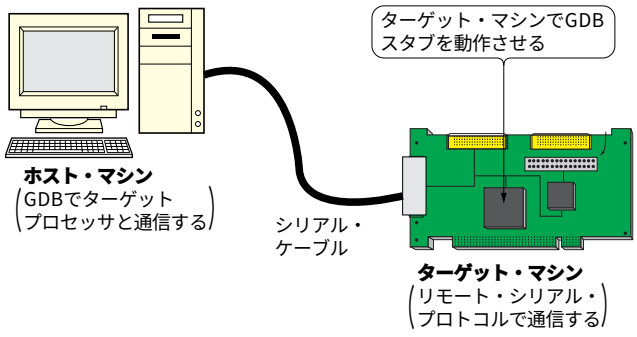

図1　GDB＋GDBスタブによるリモート・デバッグ環境

KEYWORD ―デバッガ，GDB，リモート・シリアル・プロトコル，GDBスタブ，例外ハンドラ，ROM化，スタートアップ・ルーチン，リンカ・スクリプト

表1　パケットに用いられるコマンド一覧

!	拡張モード	コマンド解釈モードを拡張モードに変更する．一度拡張モードになると，ノーマル・モードに変更できない．デバッグ対象のプログラムを再起動するには‘ R ’パケットを使う． 返答：‘ OK ’リモート・ターゲットを拡張モードに変更完了
?	ラスト・シグナル	ターゲットの中止の理由を表示する．返答はステップ，コンティニュと同じ
a	予約	将来使用するための予約
Aarglen,argnum,arg,...	プログラムの引き数をセットする（予約）	初期化された argv[ ]をプログラムの中に渡す．arglen は arg のバイト数を 16 進数で指定する．詳細は，gdbserver のマニュアルを参照
caddr	コンティニュ	addr は再開するためのアドレス．addr を省略すると，現在の実行アドレスから再開する． 返答：停止返答パケットの表を参照
Csig;addr	コンティニュ （シグナル付き）	シグナル sig は 16 進数．addr を省略すると，現在の実行アドレスから再開する． 返答：停止返答パケットの表を参照
D	デタッチ	GDB をリモート・システムから切り離す．デタッチ・コマンドを通じて GDB を分離する前にこのパケットが送られる． 返答：なし．GDB はこのパケットの送信後の返答のためのチェックを行わない
e，E，f	予約	将来使用するための予約
FRC,EE,CF;XX	F パケット・ターゲットへの返答	このパケットは，ターゲットによって送られたリクエスト・パケット F に対する GDB からの返答．これは，ファイル I/O リモート・プロトコルの一部なので無視してよい
g	レジスタの読み出し	レジスタを読み取る． 返答：‘ XX... ’レジスタ・データのそれぞれのバイトは，二つの hex のけたにより表され，バイト・オーダはターゲット・システムのものに準じる．レジスタのサイズや名前などは，GDB の内部で DEPRECATED_REGISTER_RAW_SIZE と REGISTER_NAME のマクロで定義されている
GXX...	レジスタへの書き込み	XX...のフォーマットは読み出しと同じ．返答：‘ OK ’成功，‘ ENN ’エラー
n	予約	将来使用するための予約
Hct...	スレッドへのコマンド送信	スレッドへパケット操作（m, M, g, G など）を送信する．‘ c ’はステップとコンティニュのために，‘ g ’はほかのオペレーションのために付加される．t...に *t* スレッド番号を指定する．－1 を指定するとすべてのスレッドという意味． 返答：‘ OK ’成功，‘ ENN ’エラー
j，J	予約	将来使用するための予約
k	リクエストを消す	——
K，l，L	予約	将来使用するための予約
maddr,length	メモリの読み出し	メモリを addr アドレスから length バイト読み取る．addr，length は 16 進数． 返答：‘ XX... ’はメモリの内容を 16 進数で返す．‘ ENN ’ NN はエラー番号
Maddr,length:XX...	メモリへの書き込み	メモリの addr 番地から length バイトに XX...のデータを書き込む．addr，length，XX... は 16 進数．返答：‘ OK ’成功，‘ ENN ’エラー
n，N，o，O	予約	将来使用するための予約
Pn=r	レジスタ書き込み（個別）	レジスタ番号 *n* に値 *r* を書き込む

以上の理由から，ホスト・マシンとターゲット・マシンがシリアル・ケーブルで接続され，ターゲット・マシンにリモート・シリアル・プロトコル実装用の ROM モニタが実行されていれば，どのようなターゲット CPU でも GDB で組み込みプログラムのデバッグ環境を構築することが可能になります．このリモート・シリアル・プロトコルを実装する ROM モニタのことを GDB スタブと呼びます．

従って，本稿で前提とする環境は，**図1**に示すようにホスト・マシンとターゲット・マシンがシリアル・ケーブルでつながれているものです．ホスト・マシンでは GDB が起動されていて，ターゲット・マシンでは GDB スタブが実行されている環境を考えます．

2. GDB リモート・シリアル・プロトコル

GDB の機能の恩恵を受ける際には，GNU ツールでコンパイルされた ELF などの実行形式ファイルを作成する環境と GDB スタブを必要とします．GNU ツールについては前章で説明したので，ここでは GDB スタブについて説明します．まず，GDB スタブを作成する上で重要なリモート・シリアル・プロトコルについて解説します．

● プロトコルの概要

リモート・シリアル・プロトコルは GDB ドキュメントに規定されています．

```
http://sources.redhat.com/gdb/download/onlinedocs/gdb_33.html#SEC655
```


News Flash ――仕様記述や合意方法を標準化する「発注者ビューガイドライン（画面編）」を公開
NTT データなど 9 社は，情報システムの仕様記述や合意方法を共通化する「発注者ビューガイドライン（画面編）」を公開した．発注者と開発者の間の誤った理解を防ぐ狙いがある．富士通，NEC，日立製作所，構造計画研究所，東芝ソリューションなどが策定に参加した．