プロローグ

TCP/IP プロトコル・スタック？ FAT ファイル・システム？ RTOS ？

GCC でプログラムを作ろう！

編集部

● C プログラミング/デバッグ環境

本誌 2008 年 9 月号付属 ColdFire マイコン基板（以降付属基板，**写真 1**）搭載の CPU には，C インタプリタ SilentC が内蔵フラッシュ ROM にあらかじめ書き込まれています．パソコン上で C インタプリタ用ソース・コードを書いて，TFTP コマンドでソース・プログラムを転送するだけで，付属基板上で手軽に動作確認ができました．

しかし，C インタプリタはコンパイラでコンパイルしたプログラムと比較して一般的に実行速度が遅く，付属基板搭載のものはファイル領域のサイズが小さいので，ファイル・サイズの大きなプログラムを実行できませんでした．

また先月号では CPU ベンダ純正ツールともいうべき C コンパイラとして，CodeWarrior Development Studio for ColdFire Architectures V7.0 Special Edition（**写真 2**，以降 CodeWarrior）を取り上げました．CD-ROM も付属しているので，インストール後すぐに C プログラムをコンパイルでき，ColdFire 上でネイティブ・コードを実行可能です．しかし CodeWarrior を使ってプログラムをデバッグする場合，BDM デバッガと呼ばれる ColdFire 用の CPU デバッグ機器が必要です（**写真 3**）．

さらに付属基板には，BDM 端子と呼ばれる BDM デバッガを接続するための端子は用意されていません．BDM 端子に必要な信号が拡張用スルーホールまで出力されているので，26 ピン・コネクタの BDM 端子（**写真 4**）を用意して，拡張用スルーホールとコネクタの間を配線しなければなりません．なお，**写真 5** に示す付属基板対応拡張ベースボードを使えば，拡張ベースボード上に用意された BDM 端子を使って，すぐに BDM デバッガを接続できます．

いずれにせよ，CodeWarrior を使ったデバッグには，幾つかの機材の追加が必要です．

● GNU ツールによる C コンパイラとデバッガを用意

そこで今月号の特集記事では，GNU（GNU is Not Unix）の世界で使われているコンパイラの GCC（GNU Compiler Collection）と GDB（GNU Debugger）を取り上げます．GNU と聞くと Linux などの UNIX 系 OS が必要と思われるかもしれません．しかし，Windows 上に UNIX 環境を実現する Cygwin を使うことで，Windows マシンさえ 1 台あれば GCC や GDB を使えます（**図 1**）．

Cygwin をインストールしたとはいえ，Windows マシンは CPU が 8086 系です．けれども今回プログラムを走らせようと考えている CPU は ColdFire です．CPU が異なると，CPU に対する命令も全く異なります．

そこで今回は，8086 系の CPU で，ColdFire 向けのプログラムをコンパイルする C コンパイラを用意しました．異

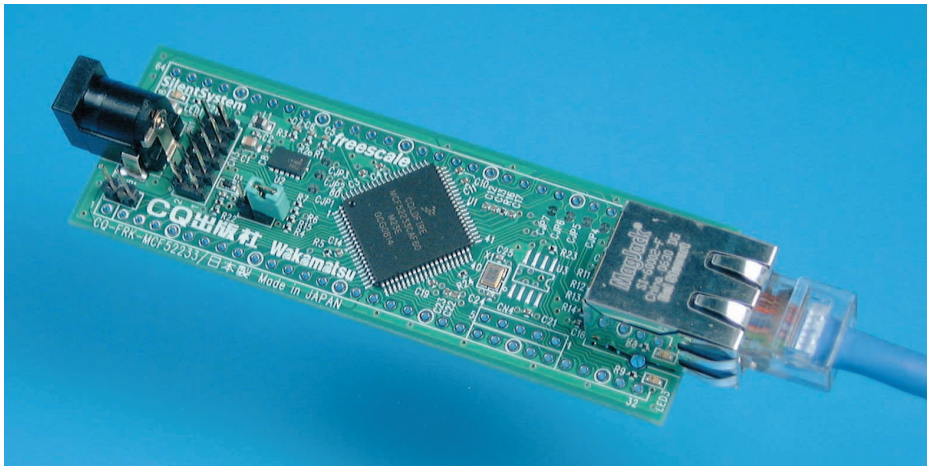

写真 1　Interface2008 年 9 月号付属 ColdFire マイコン基板

写真 2　Interface 2008 年 9 月号付属 CD-ROM（CodeWarrior Development Studio for ColdFire Architectures V7.0 Special Edition）

写真 3
ColdFire 対応 BDM デバッガ

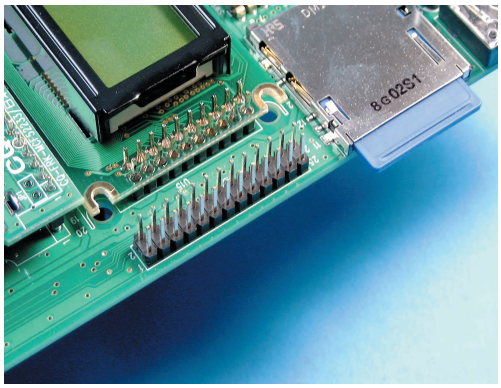

写真 4
BDM 端子

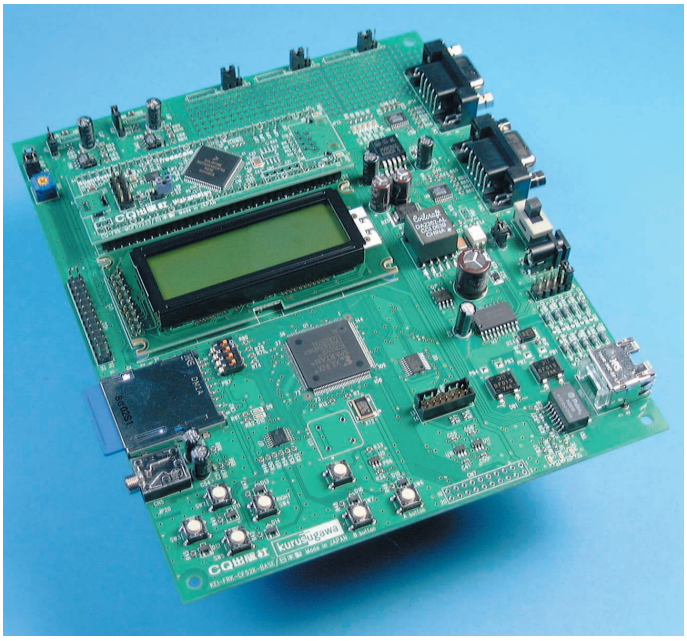

写真 5　ColdFire マイコン基板対応拡張ベースボード（量産版）
入手方法は第 5 章（pp.116-125）を参照．

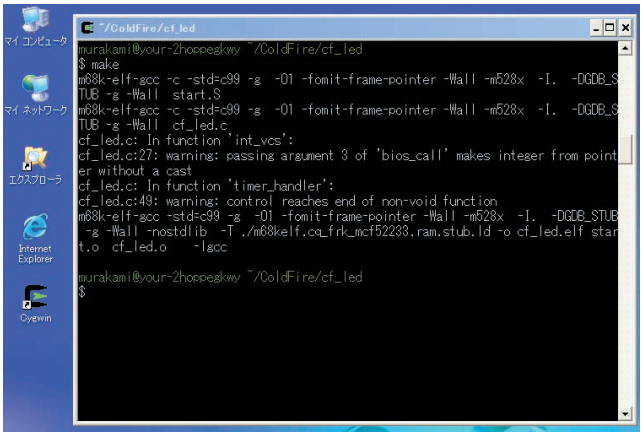

図 1　Windows 環境に Cygwin をインストールすることで，GNU ツールを使える環境を用意する

なる CPU 向けのプログラムをコンパイルすることを，クロス・コンパイルといいます．つまりここで用意した C コンパイラは，（CPU は 8086 系の）Cygwin 環境で動作する，ColdFire 用のプログラムをコンパイルするクロス・コンパイラというわけです．

プログラムをコンパイルするだけでなく，正しく動くかどうかの動作確認も必要です．このときに使用するツールがデバッガ GDB です．プログラムの動作がおかしいとき，各種変数の値が想定している値になっているかどうか表示（ウォッチ）したり，分岐するはずの `if` 文で正しくジャンプしているか確認したり，ステップ実行したり，ブレークポイントを設定したりなどの操作をします．

ターゲット CPU ボードと開発パソコン間の接続は，LAN ケーブル 1 本で OK です．今回は BDM デバッガなどは使わずに，LAN ケーブルを使ってプログラムを付属基板に転送

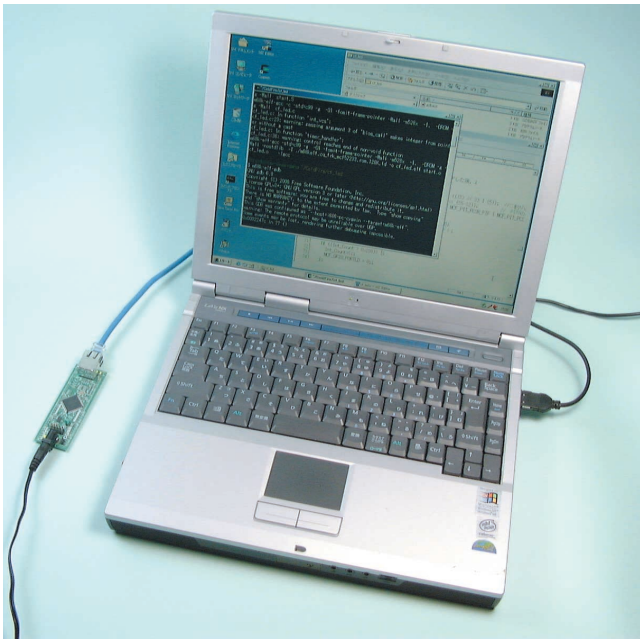

写真 6　GNU によるクロス開発環境をインストールしたパソコンと ColdFire マイコン基板を Ethernet で接続

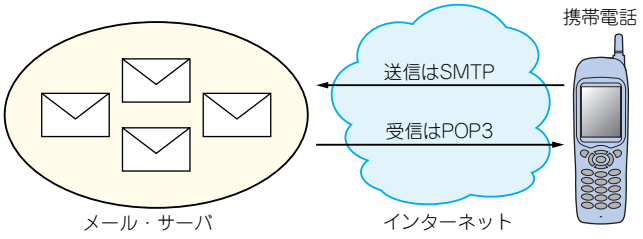

図2　メールは SMTP および POP3

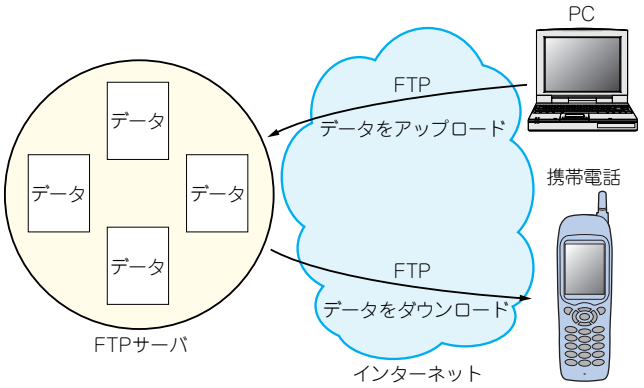

図4　ファイルの転送には FTP

(ダウンロード)してデバッグ可能です.

写真 6 に示すように，Cygwin や GCC/GDB をインストールしたパソコンと付属基板を，LAN のクロス・ケーブルで接続するだけで，C プログラムのコンパイルからデバッグまで行える，立派な開発環境が用意できます.

● プロトコル・スタックって何？

さて，GCC や GDB があれば，C インタプリタ SilentC のサンプル・プログラムのように，UDP 通信をしたりメールを送受信するなど，簡単にネットワーク対応プログラムを作れるかというと，実はそう簡単ではありません.

1 本の LAN ケーブルで，メールが送信できたり Web サイトを見られるのは，送信するデータがメール用なのか，Web 用なのかを示すプロトコルが決められているからです．代表的なプロトコルとして，メールの場合は SMTP および POP3 (**図 2**)，Web では HTTP (**図 3**)，ファイルの転送には FTP (**図 4**) などがあります．SilentC を操作する場合に使った TELNET も，プロトコルの一種です.

メールを送ったり Web をブラウズするには，これらのプロトコルを処理するルーチンが必要です．これをプロトコル・スタックと呼びます.

SilentC で簡単にメールなどが送受信できたのは，SilentC がこれらプロトコル・スタックを内蔵したシステムだからです．SilentC を使わずに GCC を使ってネットワーク・プログラムを作ろうとすると，各種プロトコルを処理するプロトコル・スタックが必要になります.

そこで今回は，TINET と μIP という 2 種類のプロトコル・スタックを付属基板に移植してみました.

● FAT ファイル・システムって何？

付属基板に搭載されている CPU は，内蔵フラッシュ ROM が 256K バイトしかありません．例えばネットワーク

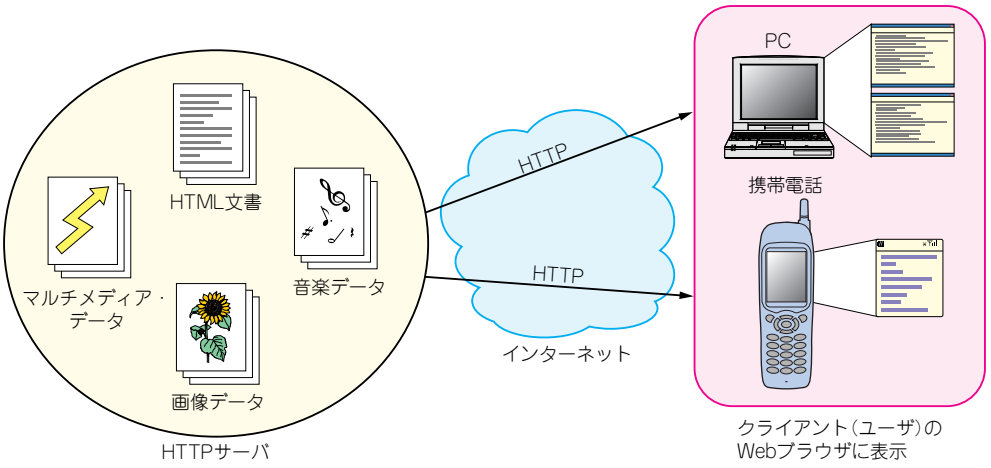

図3　Web は HTTP

経由でデータを転送する場合，内蔵フラッシュ ROM に記録すると，あっという間にデータがいっぱいになってしまいます．

そこで，デジタル・カメラや携帯電話などで普及している小型フラッシュ・メモリ・カードを使って，これにデータを記録することを考えます．今回は SD カードとも互換性のある MMC カード（**写真 7**）を使用しました．

ネットワークにプロトコルというルールがあるように，フラッシュ・メモリ・カードにもプロトコルに相当する約束事があります．これを，ファイルを記録するストレージ・メディアの場合はファイル・フォーマットと呼びます．MMC カードの場合は，ファイル・フォーマットとして FAT ファイル・システムが採用されています．

FAT ファイル・システムに対応できると，Windows マシンに接続したフラッシュ・メモリ・カード・リーダを使ってファイルをカードに転送し，それを付属基板で読み出すことも，またその逆も可能です．

● リアルタイム OS って何？

例えばフラッシュ・メモリ・カードに入ったファイルを友人にメールで送ろうとした場合，ファイル・アクセスとネットワークへのアクセスの両方を同時に動かす必要があります．このように複数の処理を同時・並列にプログラムするには，オペレーティング・システム（OS）があると便利です．

OS というと，Windows などのように画面表示やキーボード入力のあるマシンを想像する人も多いでしょう．しかし ColdFire のような組み込み向けのマイコン・システムであっても，OS があると CPU をより効率良く動かせます．

ただし，組み込みシステムで使われる OS は，Windows と比較するとある特徴があります．例えば工場などで動くシステムは，緊急停止ボタンが押されたら即時の機械停止が保障されている必要があります．Windows のように砂時計が表示されて，停止するまで数十秒かかるようなシステムでは使いものになりません．このように，一定時間以内に応答できる仕組みが用意されている OS を，リアルタイム OS と呼びます．

リアルタイム OS にもいろいろな種類があります．ColdFire マイコンのようにメモリが数十 K バイトから数百 K バイト程度の CPU で多く使われる OS として，μITRON

写真 7　MMC カード（左）と RS-MMC カード（右）

があります．μITRON は OS の固有名詞ではなく，OS 仕様の一般名称であり，この仕様に準拠したリアルタイム OS が幾つか発表されています．この中から今回は TOPPERS/JSP を付属基板に移植しました．

● すべてオープン・ソース！

今回，特集記事で用意したコンパイラやデバッガなどの開発環境はもちろん，付属基板に移植したリアルタイム OS や TCP/IP プロトコル・スタック，FAT ファイル・システムなどのミドルウェアは，すべてがオープン・ソースとなっています．

これにより，リアルタイム OS やミドルウェアの中身がどうプログラムされているか，実際にソースを見て確認できます．OS やミドルウェアの学習用としても最適です．

＊　　＊

先月号の C インタプリタでは満足できなかった人は，ぜひ今月号のこの環境で，思う存分プログラミングを楽しんでください！