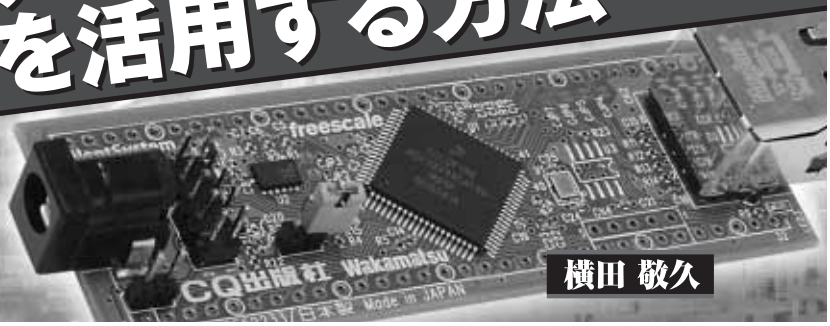
Appendix
CodeWarriorと付属ColdFireマイコン基板1枚だけで
プログラムを実行する

SilentCを消さずにCodeWarriorを活用する方法



関連データ

横田 敬久

● 開発環境とその問題点

本誌2008年9月号では，プログラミング環境としてSilentCとCodeWarriorを，そして10月号ではGCCを用いた開発を解説しました．

9月号で解説しているCodeWarriorは，プログラムの配置アドレスをデフォルト設定（アドレス0番地から配置）で開発しています．しかしこの方法には，いくつかの問題点があります．

この設定のままターゲットとして付属ColdFireマイコン基板を使う場合，Cインタプリタ SilentCを上書きして消してしまうことになります．この問題に対しては，GDBスタブを消してもよいという場合は，配置アドレスをフラッシュROMの後半128Kバイトの位置にずらす方法があります．また，上書きして消えてしまうSilentCやGDBスタブは，本誌のダウンロード・ページから，付属ColdFireマイコン基板に書き込み済みモトローラS形式ファイルが入手できるので，これを使ってフラッシュROMの内容を復旧させることができます．さらに，付属ColdFireマイコン基板を2枚用意して，マスタ/スレーブ接続することによりリカバリも可能です．

もう一つの問題は，アドレス0番地からフラッシュROMに書き込むには，BDMデバッガが必要になる点です．これについても，11月号や今月号の記事で，パラレル・ポートまたはUSB接続版のBDMデバッガを安価に用意可能です．

とはいえ，別途にツールを用意せず，9月号付属ColdFireマイコン基板を1枚だけ使って，SilentCを消さずに後半128Kバイトの領域で実行するプログラムをCodeWarriorで開発したいという要求もあるでしょう．

そこでここでは，CodeWarriorを使ってフラッシュROMの後半128Kの領域に配置するプログラムの作り方と，それをBDMを使用せずに付属ColdFireマイコン基板を1枚だけを使って書き込む方法について紹介します．

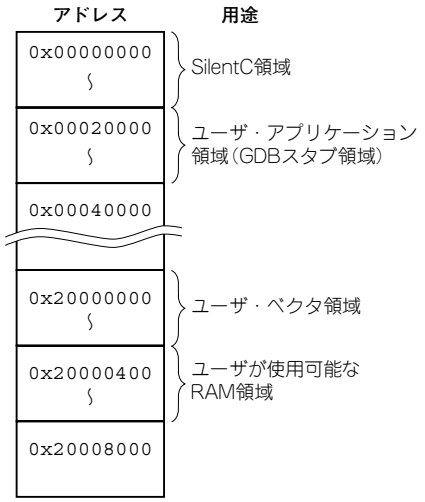
● CodeWarriorの設定

本誌9月号に付属するCD-ROMのCodeWarriorがインストールされていることが前提となります．9月号の説明のようにCodeWarriorをインストールしてください．

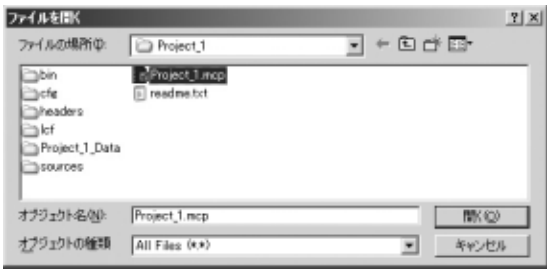
次に，ひな型となるプロジェクト・ファイルProject_1を作成したので，本誌のダウンロード・ページから入手してアーカイブ・ファイルを解凍してください．**表A**に，ファイルの内容（一部）を示します．また，このプロジェクト・ファイルのメモリ・マップを**図B**に示します．プログラムの中身は，10月号の特集 第1章で紹介したLED点滅ルーチンとほぼ同じものです．

表A　ひな型プロジェクト・ファイルProject_1の内容（一部）

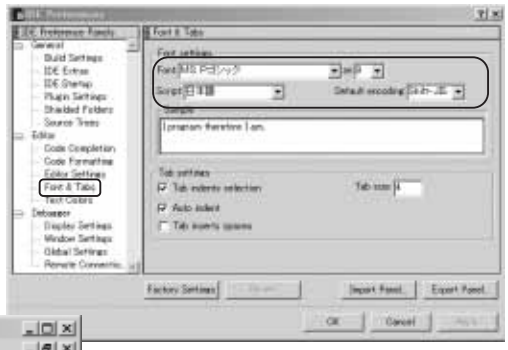
ファイル名	説　明
Project_1.mcp	プロジェクト・ファイル
Project_1_Data	コンパイラのターゲット情報
bin	実行ファイルの出力先 （ELFファイル，Sレコードなど）
headeres	ヘッダ・ファイル
lcf	リンカ・スクリプト
sources	C言語のソース
readme.txt	説明ファイル



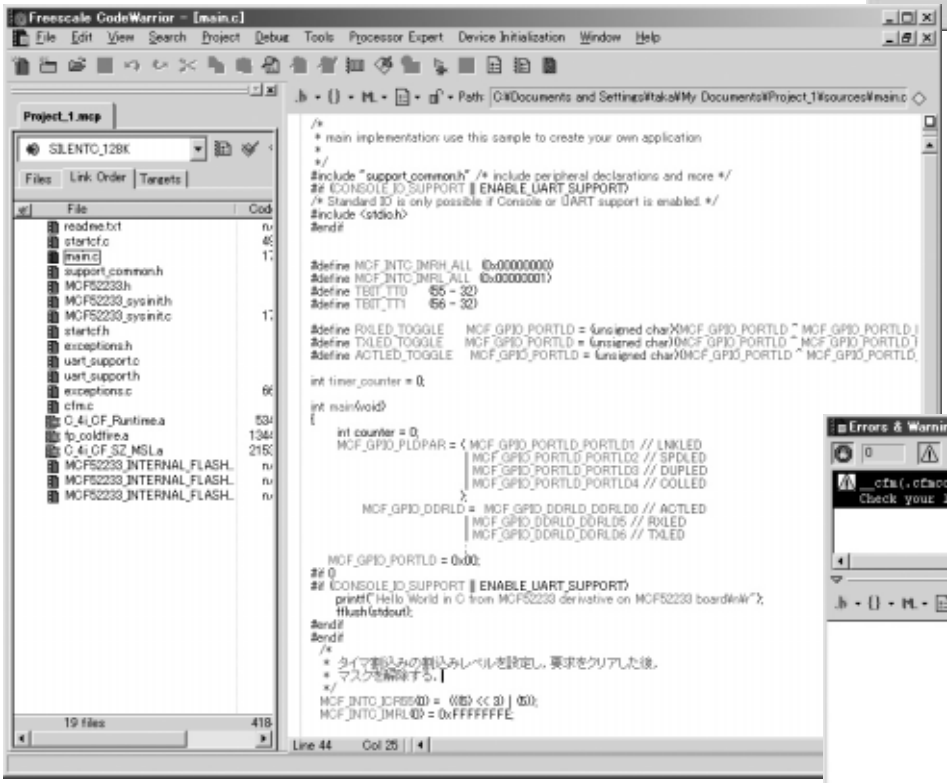
▶ 図B
ひな型プロジェクト・ファイルのメモリ・マップ



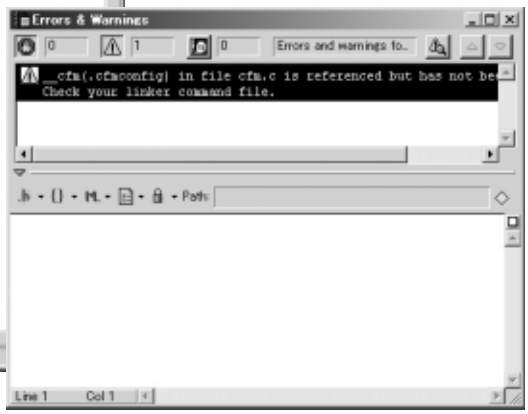
◀図C
プロジェクトのオープン



▲図D
日本語表示設定



図E　Make



図F　ワーニングが出力

まず，CodeWarrior を起動して，プロジェクト・ファイル Project_1.mcp を開きます（**図C**）．

デフォルト状態では，CodeWarrior のエディタは日本語を表示しませんが，**図D**のように IDE Preference Panels の Editor の中の Font&Tabs の項目で日本語フォントを選択し，文字コードを Shift-JIS などに設定することで，日本語も表示できます．

コンパイル・ターゲットは，プログラム配置アドレスが内蔵フラッシュ ROM の後半 128K バイトになっています．スタートアップ部は，既存のルーチンに手を入れてしまっているため共存はできません．しかし，CodeWarrior が出力するソースをベースに作成しているので，それ以外の部分は従来のコンパイラが生成するソースを極力再利用できます（**図E**）．

プロジェクト・インスペクタの Make ボタンを押すことでメイクが始まります．BDM による転送やデバッグの実行はできない前提なので，デバッグや実行ボタンを押す必要はありません．**図F**のようなワーニングが出力されますが，今回のケースでは気にする必要はありません．

コンパイルしたバイナリは，ELF フォーマットとして bin ディレクトリに COLDFIRE.elf が出力されます．しかし，CodeWarrior の IDE からリンカが出力するバイナリ・イメージでは，作成したものが正しく動作しないようです．そのため，11 月号で紹介した GNU binutils の objcopy で ELF ファイルをバイナリ・フォーマットに変更し，COLDFIRE.BIN を生成します．

図 G　CF Init の CPU 選択画面（M52235 を選択）

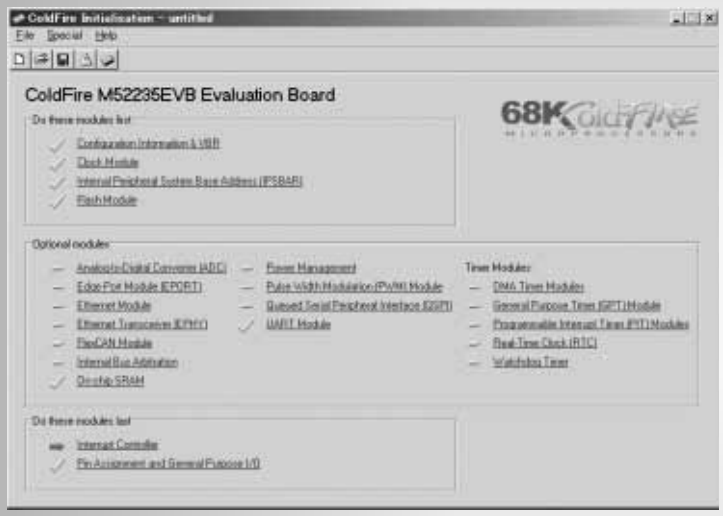


図 H　CF Init の周辺機能選択画面

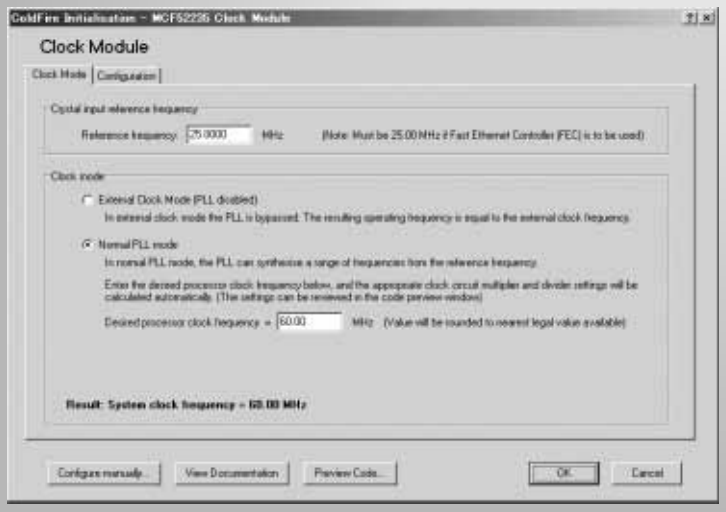


図 I　CF Init のクロック・モジュール周波数設定画面

Cygwin のコマンド・プロンプトを起動して，Project_1/bin までディレクトリを移動してください．そこで，次のコマンドを実行します．

```
$ m68k-elf-objcopy.exe -O binary
COLDFIRE.elf COLDFIRE.BIN
```

これで COLDFIRE.BIN が作成されます．

フラッシュ ROM への書き込みは，GCC の場合の手順と同じです．付属 ColdFire マイコン基板の JP2 をオープン状態で電源を入れ，SilentC を起動します．そして，Windows のコマンド・プロンプトを起動して，先ほど生成した COLDFIRE.BIN を次のコマンドで転送します．

```
C:¥>tftp -i 192.168.1.10 put COLDFIRE.
BIN
```

転送が終わったら JP2 をショートして，CPU をリセットまたは電源を再投入すると，後半 128K バイトに転送した CodeWarrior でコンパイルしたプログラムを実行できます．うまく LED が点滅したでしょうか．

よこた・たかひさ
来栖川電工（有）

コラム A
ColdFire の周辺機能の初期化ルーチン生成ツール CF Init

ColdFire マイコンは，さまざまな周辺機能を内蔵しています．これをデータ・シートだけ参照して設定し，プログラムを作成するのはなかなか大変です．

しかし，世の中には便利なツールがあります．ColdFire には，“CF Init”という各種 ColdFire シリーズの周辺機能の初期化ルーチンを生成できる GUI ツールが存在します．

これを使うと，ColdFire の周辺機能の初期化コードを容易に生成できます（**図 G ～図 I**）．筆者も，設定が複雑な周辺機能についてはこのツールを使ってコードを生成し，その内容をデータ・シートと照らし合わせて解読しながら，周辺機能の使い方を理解しています．

CF Init は，次の URL から入手できます．

http://www.microapl.co.uk/CFInit/cfinit_main.html