

● 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.

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

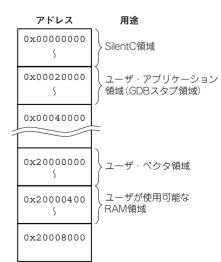
とはいえ、別途にツールを用意せず、9月号付属 ColdFire マイコン基板を1枚だけ使って、SilentC を消さずに後半 128K バイトの領域で実行するプログラムを CodeWarrior で開発したいという要求もあるでしょう.

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。



▶ 図 B ひな型プロジェクト・ファイル のメモリ・マップ

- P -

elib I e

E Fout It Take

eret 🖺 I 🎚

Build Settings IDE Entres IDE Sherke Phages Settings Sheided Folder



File Edit View Search Project Debug Tools Processor Expert Device Initialization Window Help

#include (stdio.h) #endif

Adefine RXLED_TOGGLE Adefine TXLED_TOGGLE Adefine ACTLED_TOGGLE

int timer_counter = 0;

jnt main(void)

int counter = 0; MCF_GPIO_PLDPAR = { MCF_GPIO_PORTLD_PI

/* * タイマ割込みの割込みしべルを設定し、要求をクリアした後、 * マスクを解除する。

MOF GPIO DDRLD = MOF GPIO PORTLD = 0x00:

MOF INTO JORESOD = (VE) << 30 J (SO); MOF INTO IMPLOD = 0xFFFFFFFE

Line 44 Gol 25 [4]

* main implementation use this sample to create your own application

#include "support commonth" /* include peripheral declarations and more */
(CONSOLE IO SUPPORT || ENABLE UART SUPPORT)

警告は目のシストの名を御御御 (季覧を) ■ 日泊 (数

• E &

@Freescale CodeWarrior - [main.c]

Project_1.mcp

♠ SILENTO 128K

readme.txt
startcf.c
startcf.c
main.c
support.co
MCF52233
MCF52233
MCF52233 readments
startof.c
|menc|
support_commonh
MCF52233.h
MCF52233_sysinith
MCF52233_sysinitc
startofh
support_commonh

exceptions? uart supporto uart_supporth exceptions.c

ormanie.

C.4.CF Runtime.a

to.coldire.a

C.4.CF SZ MSLa

MCF82233 INTERNAL FLASH.
MOF82233 INTERNAL FLASH.
MOF82233 INTERNAL FLASH.

Files Link Order Tangets

■図C プロジェクトのオープン

ve on MCF52233 boardVnW">



図 E Make

19 files

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).

図F ワーニングが出力

Line 1 Col 1 +

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.



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:\prec{4}>tftp -i 192.168.1.10 put COLDFIRE.

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 $G \sim 2$ 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

Interface Dec. 2008