第2章 ColdFireアーキテクチャの最深部を理解する

付属ColdFireマイコン基板へのTOPPERS/JSPの移植

横田 敬久

本章では，μITRON 4.0仕様準拠のOSであるTOPPERS/JSPを付属ColdFireマイコン基板へ移植する．まずはサンプル・プログラムのコンパイルとダウンロードを行い，実際に動いているところを確認してみよう．

そして，移植の実際について解説を行う．ColdFireへの移植にあたってはMC68040版からではなく，あえてSH-2版を参考にしている．その理由も含めて，ColdFireアーキテクチャの一番深い部分である例外周りを中心に解説を行う．（編集部）

ここでは本誌2008年9月号付属ColdFireマイコン基板を対象に，TOPPERS/JSPの移植事例を紹介します．

TOPPERS/JSPは，μITRON 4.0仕様スタンダード・プロファイルに準拠したオープン・ソースのリアルタイムOSです．ColdFire版のTOPPERS/JSPに関してはもともとColdFire評価ボードM52235EVBをターゲットとして移植が進んでいました．今回，付属ColdFireマイコン基板にMCF52233が搭載されることになったため，付属ColdFireマイコン基板に対してTOPPERS/JSPを移植しました．

TOPPERS/JSPを動作させるにあたり，前章までのGNUの開発環境のインストールを前提としています．また，TOPPERS/JSPの動作には必須項目ではありませんが，動作確認をするためにはシリアル・ポートの接続も必要です．

1．TOPPERS/JSPについて

● μITRONとスタンダード・プロファイルとは

TOPPERS/JSPは，μITRON 4.0のスタンダード・プロファイルの仕様に基づいて作られたリアルタイムOSです．μITRON 4.0のスタンダード・プロファイルは，タスクやセマフォ，メールボックス，周期ハンドラ，イベント・フラグ，固定長メモリ・プールなどの機能を備えています．

μITRONでは，アプリケーション・プログラムとOSは，一つの実行ファイルとしてリンクされます（**図1**）．TOPPERS/JSPにおいても例外ではなく，OSの機能はアプリケーションとリンクされた一つの実行ファイル（jsp.exe）が生成されます．アプリケーションとOSが一体化しているので，OSも含めたアプリケーション全体が特権モードで動き，I/Oポートへのアクセスがアプリケーション側からも直接できてしまいます．

μITRONのスタンダード・プロファイルではダイナミック（動的）にカーネルの機能のオブジェクトを生成できず，すべて静的なオブジェクトで扱われます．例えば一般的なOSでは，セマフォなどは必要になったときに確保し，不要になったら開放するというような使い方をしますが，μITRONの場合はコンパイル時にあらかじめ確保しておきます．そのため，あらかじめどのくらいのリソースを使用するのかをコンフィグレーション・ファイル（.cfg）に指定しておく必要があります．コンフィグレーション・ファイルは，コンフィグレータというプログラムによって解釈され，カーネル・オブジェクトの生成に必要なC言語のソー

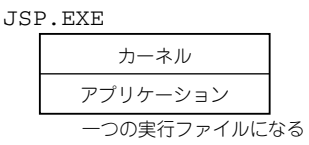

図1　μITRONの実行ファイル

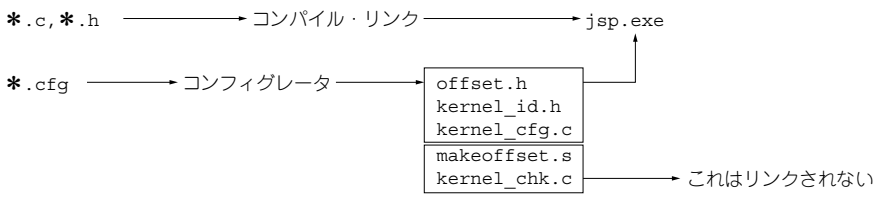

図2　コンパイルとコンフィグレーションの流れ

KEYWORD —— ColdFire，MC68000，OS移植，μITRON，TOPPERS/JSP

```
$ m68k-elf-gdb
GNU gdb 6.8
Copyright (C) 2008 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=i686-pc-cygwin --target=m68k-elf".
(gdb) target remote udp:192.168.1.10:6000
warning: The remote protocol may be unreliable over UDP.
Some events may be lost, rendering further debugging impossible.
Remote debugging using udp:192.168.1.10:6000
0x0002247c in ?? ()
(gdb) load jsp.exe
Loading section .text, size 0x3128 lma 0x28000
Loading section .rodata, size 0x780 lma 0x2b128
Start address 0x28000, load size 14504
Transfer rate: 35 KB/sec, 204 bytes/write.
(gdb) c                    ← continueの意味
Continuing.
```

図4　jsp.exeのダウンロード

```
(TOPPERS/JSPのルート・ディレクトリに移動)
cd cfg
make    ← コンフィグレータのコンパイル
..      ← 次の作業のためにJSPのルート・ディレクトリに戻る
cd sample1_led
make depend
make
```

図3　LED点滅のサンプル・プログラムのコンパイル

ス・ファイルやヘッダ・ファイルを生成します(**図2**).

2. LED点滅のサンプル・プログラムを使ってみる

● 2種類のサンプル・プログラム

百聞は一見にしかずというので，まず，付属ColdFireマイコン基板へ移植したTOPPERS/JSPを動かしてみましょう.

TOPPERS/JSPをインストールする際に誰もが動かすであろう標準添付プログラムのsample1に加えて，UARTがない環境でもTOPPERS/JSPの動作を確認できることを考慮して，ここではLEDの点滅プログラムをサンプル・プログラムとして用意しました．今回のサンプル・プログラムは，周期ハンドラを利用してACT_LED(LED2)とLNK_LED(LED3)を単純に2秒ごとに点滅させることができます.

● LED点滅のサンプルをコンパイルする

本誌のWebページから第2章のサンプル・プログラムをダウンロードし，解凍してください．次に**図3**のようにLED点滅のサンプル・プログラムをコンパイルします.

コンパイルが正常に終了すると，jsp.exeとjsp.srecが生成されます．第1章でコンパイル後のメモリ配置として3種類の方法が選べると説明しました．しかし，このLED点滅用サンプル・プログラムは，デフォルト状態で後半128Kバイトからの GDBスタブからダウンロードして起動するメモリ配置(SILENT_STUB)になっています.

jsp.exeを付属ColdFireマイコン基板にダウンロードさせるためにGDBを起動します(**図4**).

いかがでしょうか？　実行がうまくいくと，2秒おきにACT_LEDとLNK_LEDが点滅します.

● ソース・プログラムの解説

それではソースを見てみましょう．**リスト1**にプログラム本体sample1.cを，**リスト2**にコンフィグレーション・ファイルsample1.cfgを示します．**リスト1**を見れば気付くと思いますが，このLED点滅のアプリケーションをはじめとして，μITRONではC言語でいうmain関数からプログラムが開始するとは決まっていません．アプリケーションに相当する部分は，今回の場合ならmain_taskから始まります.

リスト2はコンフィグレーション・ファイルで，カーネル・オブジェクトを定義しています．この場合CRE_TSKでmain_task関数をタスクとして作成しています．タスクの属性にTA_ACTを付けているので，TOPPERS/JSPカーネルが起動すると同時にタスクが起動します．main_taskでsample1.cfgに定義されている周期ハンドラ(一定周期で処理を繰り返す)をスタートさせます．周期ハンドラは2秒後ごとの周期でcyclic_handler関数を呼び出し，LEDを点滅させます.

このように，付属ColdFireマイコン基板ではTOPPERS/JSPを活用できます．しかしデバッグをする際や，最も基本的な標準アプリケーションであるsample1を実行する