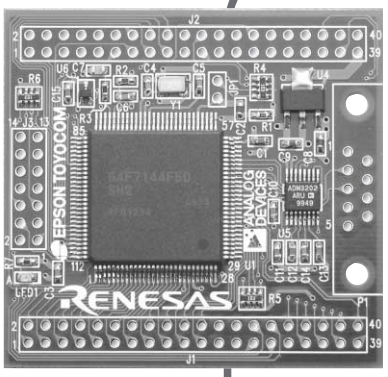
第5章

新しいCPUへのカーネル移植のポイント

TOPPERS/OSEKカーネル移植作業の実際

森川 聡久/片岡 歩/鵜飼 敬幸/杉本 明加/橋本 昌伸/大西 秀一



前章で紹介したTOPPERS/OSEKカーネルの移植作業について解説する．これまでTOPPERS/OSEKカーネルはSH-2には対応していなかったが，今回の記事のために移植作業を行うことになった．割り込み部分など，CPUアーキテクチャに依存する部分の移植方法について重点的に解説する．（編集部）

本章ではOSEK仕様OSのフリーな実装であるTOPPERS/OSEKカーネルをSH-2付録基板に移植，そして検証した作業の全容を紹介します．

TOPPERS/OSEKカーネルの概要を説明したので，移植を行うカーネルについて理解できると思います．

移植作業では，依存部と呼ばれるシステム・ターゲット個別に対応が必要な部分を説明します．また，システムを構築する際に必要となるシステム・コンフィグレーションの方法をまとめます．続いて本カーネルに含まれるサンプル・アプリケーションを動作させるための手順と利用方法を紹介し，最後に検証環境について説明します．

なお，本文中の用語などはカーネルに含まれるTOPPERS/OSEKカーネル外部仕様書を参考にすると，より理解が深まるでしょう．

TOPPERS/OSEKカーネルとは

μITRON準拠OSであるTOPPERS/JSPカーネルは数多くのターゲット・システムをサポートしています．OSEK仕様のTOPPERS/OSEKカーネルでもこの思想は受け継がれており，JSPカーネルを手本としてほかのターゲットへの移植を容易にするくふうをしています．

現在TOPPERSで公開されている「TOPPERS/OSEKカーネルVersion1.1」では，ルネサス テクノロジ製M32C/80シリーズ，M16C/20シリーズ，H8S/2600シリーズ，H8/300Hシリーズがサポート対象となっていますが，筆者らは，ルネサス テクノロジ製M16C/60シリーズ，R8C/Tinyシリーズ，NEC製V850Eシリーズ，ARMコア・シリーズなどへ移植した経験があります．しかし，今回ターゲットとするSH-2コアは初めて移植しました．

本章では，TOPPERS/OSEKの移植性の良さをアピールしながら，実際の移植作業を紹介していきます．

TOPPERS/OSEKカーネルの移植作業

● TOPPERS/OSEKの開発環境

今回の移植にあたって，以下の開発環境を用意しました．

- Cygwin
- binutils-2.16
- gcc-3.4.5
- newlib-1.14.0

Cygwinを採用したのは，GCCによるクロス・コンパイル環境が簡単に手に入るという理由からです．またこれまでの本誌にもCygwinを利用した開発環境の構築の記事があり，こちらも参考にして開発環境を構築しました．

● TOPPERS/OSEKカーネル移植

OSの移植というと難しそうな印象がありますが，CPUのアーキテクチャやアセンブラ仕様を理解していれば，後は既存ターゲットの実装を読みながら学んでいくことで比較的容易に移植を行うことができます．今回の解説でTOPPERS/OSEKカーネルについて学んだあと，得意とするCPUに移植してみてはいかがでしょうか．

それでは，早速，移植作業について説明していきますが，その前に，**図1**に本カーネルの構成を示します．

図1からわかるように，本カーネルは，共通部とターゲット依存部に分離されています．共通部とターゲット依存部は，CPUアーキテクチャや動作環境に関係なく実装できるかどうかで切り分けられています．共通部がカーネル全体の80～90%となっており，容易に移植することが可能です．

今回は，ターゲット依存部，サンプル・プログラムの移植，およびSG（システム・ジェネレータ）への対応を行いました．

● ターゲット依存部の構成

ターゲット依存部のおもな機能は以下のとおりです．

- ディスパッチャ

KEYWORD ── TOPPERS/OSEK，OSEK/VDX，CPU移植，割り込み処理，適合試験仕様MODISTARC

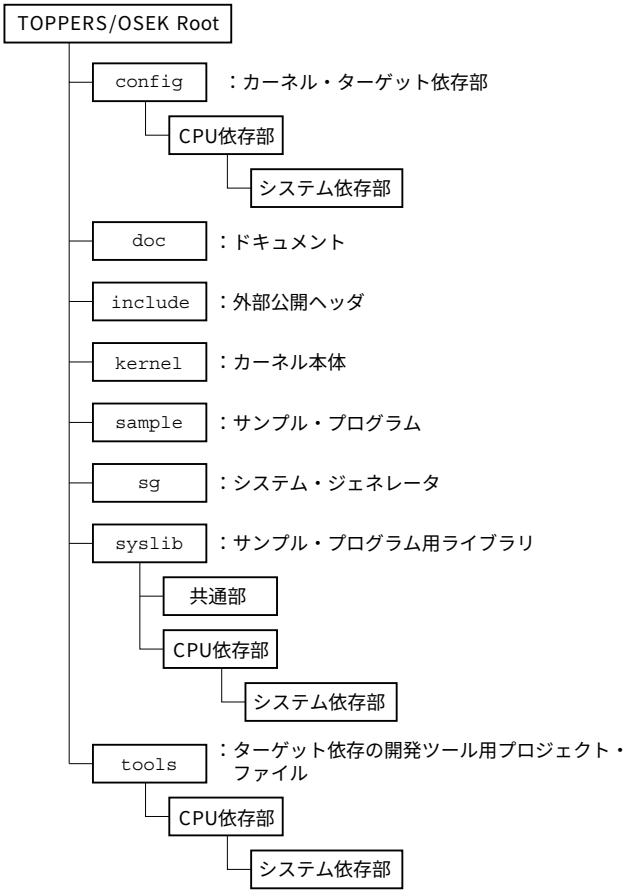


図1　OSEKディレクトリ階層

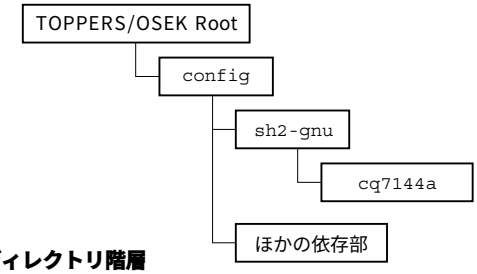


図2
今回作成したディレクトリ階層

表1　CPU依存部で作成するファイル一覧

ファイル名	内　容
cpu_config.h	プロセッサ依存モジュール・ヘッダ
cpu_config.c	プロセッサCPU依存モジュール
cpu_support.S	プロセッサCPU依存モジュール・アセンブラ・ファイル
cpu_insn.h	低レベル操作ルーチン
start.S	スタートアップ
cpu_defs.h	CPU依存の定義ヘッダ
sh7144.h	SFR定義ファイル
tool_config.h	開発環境依存モジュール
tool_defs.h	ツール依存の定義ファイル
Makefile.config	CPU依存部用Makefile

表2　システム依存部にて作成するファイル一覧

ファイル名	内　容
sys_config.h	システム依存モジュール・ヘッダ
sys_config.c	システム依存モジュール
sys_support.S	システムCPU依存モジュール・アセンブラ・ファイル
sys_def.h	システム依存の定義ファイル
sh7144f.sgt	SH7144F用SGテンプレート・ファイル
Makefile.config	システム依存部用Makefile
shelf.ld	GNUリンカ・スクリプト

- 割り込み処理
- 割り込みレベル操作
- スタートアップ
- CPUやシステム，ツールに依存した定義や処理

誌面の都合上，本節ではディスパッチャ，割り込み処理，割り込みマスク制御を中心に解説します．

▶ 作成，修正するディレクトリ，ファイル

configディレクトリにCPU依存部としてSH-2用のディレクトリを作成し，さらにSH-2用のディレクトリにシステム依存部を作成します．

基本的な命名ルールとしては，CPU依存部は「CPUコア名称-ツール名称」，システム依存部は「ボード名」となっています．今回は**図2**のようにディレクトリを作成しました．

CPU依存部に作成するファイルは**表1**のとおりです．なお，cpu_support.SとstartS.はアセンブラ・ファイルで，開発環境ごとに拡張子が異なるので注意してください．GCCでは.Sを使用します．また，sh7144.hはファイル名がCPUと関連しているため，ほかのターゲットではターゲットCPUに合わせて命名します．

システム依存部に作成するファイルは**表2**のとおりです．ほかにも必要なファイルがある場合は，適宜追加します．sys_support.Sについては先に説明したとおり，開発環境ごとに拡張子が異なります．sh7144f.sgtはファイル名がCPUと関連しているため，ほかのターゲットではターゲットCPUに合わせて命名します．

実際にこの作業を行う場合には，すでに移植済みのターゲットをコピーし，リネーム・修正を行っていくことが多いでしょう．

● 移植前の事前調査

▶ タスク・コンテキストの内容

ディスパッチャの実装のためにタスク・コンテキスト（タスクを管理する構造体で保存すべき情報）として扱う情報を決める必要があります．

SH-2では，以下がタスク・コンテキストとなります．

- 汎用レジスタ
  R0～R15までの16本のレジスタがある
- スタック・ポインタ（以下SP）
  タスクのスタック位置を示す．SH-2ではR15に格納される