第11章

Blackfin DSPへの移植 ――割り込み処理

中村 健真

TOPPERSをBlackfinへ移植するにあたり，第10章ではディスパッチ部分までを解説しました．第11章ではCPU移植でも難関の割り込み部分への移植を解説します．

1. 割り込み処理とそのほかの部分

● μITRON4.0仕様における割り込みの実装

割り込み処理は，簡単そうに見えてなかなか手ごわい部分です．プロセッサのアーキテクチャの割り込み機能を素直に使うだけであれば，単に割り込みを処理して戻ってくるだけなので単純です．

しかし，割り込みからタスク・コンテキストに戻る際，μITRON4.0仕様は必要に応じてタスク例外の処理やディスパッチを行うように要請しています．この機能の実装は難易度が高く，処理系ごとにくふうを凝らす必要があります．したがって，この部分をどのようにするかが割り込み処理の要（かなめ）となります．

また，割り込みはタスク・スタックを多少とはいえ消費します．そのため，アプリケーション・プログラマはアプリケーションの必要量に加えて割り込みで使うぶんもタスク・スタックを用意しなければなりません．ディスパッチャがほぼ透過であったのと比べると対照的です．

以下，このような点を念頭に置きながら割り込み処理部の流れを見ていきましょう．

● 割り込み処理の流れ

先に述べたように，割り込み処理には少しひねった部分もありますが，プログラムの流れそのものは一直線で，ディスパッチャに比べるととても単純です．TOPPERS/JSP for Blackfinの割り込み処理の流れを**図1**に示します．

入り口処理は割り込まれたコンテキストを保存する部分で，おもにレジスタの退避を行います．それに加えて必要に応じてタスク・スタックをイベント・スタックに切り替えます．

共通割り込み処理部は，ユーザ定義の割り込みハンドラを呼び出します．通常，ここは単純に割り込みを起こしたベクタに応じてハンドラを呼び出すだけです．しかしTOPPERS/JSP for Blackfinでは少しくふうを凝らしてユーザが使いやすいようにしています．これについては後で説明します．

出口処理部は入り口処理部と逆の操作を行います．必要に応じてイベント・スタックからタスク・スタックへの切り替えを行った後，レジスタをすべて復帰させます．そして最後に割り込まれた場所に戻ります．

場合によって，出口処理部は単純に元の場所には戻らないことがあります．これはμITRON4.0仕様の要請によるものです．μITRON4.0仕様は割り込みからタスクに戻る場合，ディスパッチ要求が出ているなら

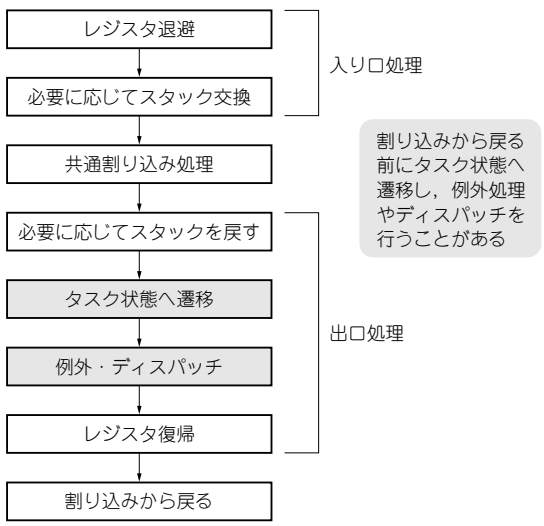

図1　TOPPERS/JSP for Blackfinの割り込み処理

見本

第12章

TCP/IPプロトコル・スタックTINETの実装

阿部 司

本章では，TOPPERSプロジェクト[注1]から配布されている組み込み向けTCP/IPプロトコル・スタックであるTINET（Tomakomai InterNETworkingの略）を例に，組み込み向けのTCP/IPプロトコル・スタックについて解説していきます．TINETは，苫小牧工業高等専門学校 情報工学科で研究・開発したもので，IPv4とIPv6の両ネットワーク層に対応しています．

1. TINETの概要 ――オープン・ソースで組み込み向け

TINETの研究・開発の当初の目的は，研究・教育における利用でした．また，現在，TOPPERSプロジェクトおよび経済産業省 東北経済産業局の委託による「組み込みシステム・オープンプラットフォームの構築とその実用化開発」プロジェクトへ参加し，産業界における広い利用も，研究・開発の目的としています．このため，だれでも自由に利用・変更・配布できるように，オープン・ソースの形態でTINETを配布しています．

● ベースはFreeBSDとKAMEプロジェクト

TINETは，FreeBSDのTCP/IPプロトコル・スタックとKAMEプロジェクトによるIPv6ソフトウェアをベースに研究・開発を行いました．その理由を以下に示します．

(1) FreeBSDのベースとなったBSD UNIXのTCP/IPプロトコル・スタックに実装されている各種アルゴリズムは，長年のインターネットにおけるプロトコル開発の成果が反映され，枯れたソフトウェアとなっており，事実上，ほかのシステムの見本となっている

(2) KAMEプロジェクトによるIPv6ソフトウェアは，フィールドで十分に検証されており，組み込みシステムのIPv6実装のベースとしても最適である

(3) BSDライセンスで配布されているFreeBSDは，ソース・コードおよびドキュメントなどにBSDライセンスを明記することで，だれでも自由に利用・変更・配布が可能な，制約の少ないライセンス形態である

(4) TINETの研究・開発では，人的リソースに限りがあり，まったくゼロからの研究・開発が困難だった

● 組み込み用途への対応

組み込みシステムの厳しいリソース制約の中で，TINETはメモリ容量の制約，とくに，少ないRAMメモリ容量への対応を最重点課題としています．また，リアルタイムOS（RTOS）を利用することから，リアルタイム性の制約も考慮しています．

しかし，TINETのベースとなったFreeBSDとKAMEが多用している動的割り当て構造体とリスト構造は，メモリ容量やリアルタイム性の制約から，組み込みソフトウェアではできるだけ使用すべきではありません．このため，TINETでは，TCPの再構成キューなど必要不可欠な部分を除いて，動的割り当て構造体とリスト構造を使用しないように，FreeBSDとKAMEのTCP/IPプロトコル・スタックに変更を加えています．

これに対応するリアルタイムOSは，TOPPERSプロジェクトで開発されたJSPカーネルです．JSPカーネルは，μITRON4.0仕様のスタンダード・プロファイルに準拠したOSであり，同様のリアルタイムOSであれば，TINETを移植することは容易であると思われます．

アプリケーション・プログラムとのインターフェースとなるAPIには，ITRON TCP/IP APIを実装して

注1：http://www.toppers.jp/

見本

第13章

TINETによるアプリケーション・プログラムの実装例

阿部 司

本章では，第12章で解説したTINETを使ったアプリケーション・プログラムの実装例を示します．筆者が使用しているTINETのターゲット・プロセッサは，ルネサス テクノロジ製のH8/300Hシリーズの H8/3048F，H8/3068FおよびH8/3069Fです．ボードとしては，秋月電子通商製のAKI-H8/3069Fフラッシュ・マイコンLANボード(以下，AKI-H8/3069Fボード)を利用しています．このため，以下での解説は，本ボードを使用することを前提にしています．また，開発環境はWindows 2000/XP上のCygwinで，コンパイラなどは，

- binutils-2.11.2
- gcc-2.95.3
- newlib-1.9.0

を使用しました．

1. サンプル・プログラムの生成と実行

ます，TINETに付属するサンプル・プログラムnservの生成と実行方法について解説します．

● JSPカーネルとTINETの入手方法

TOPPERS/JSPカーネルとTINET，および簡易モニタを入手してください．いずれもTOPPERSプロジェクト[注1]か，宮城県産業技術総合センター[注2]からダウンロードできます．リリース・バージョンはTOPPERS/JSPカーネルが1.4.1，TINETが1.2.1，簡易モニタが1.12です．

● JSPカーネルとTINETの展開とコンフィギュレータの生成

作業のようすを**リスト1**に示します．最初に，簡易モニタを展開し(1行目)，秋月電子通商製のフラッシュ・メモリ書き込みプログラムを使用して，AKI-H8/3069Fボード上のCPU内のフラッシュ・メモリに書き込みます(2行目)．TOPPERS/JSPカーネルを展開し(3行目)，さらにTOPPERS/JSPカーネルのルート・ディレクトリjspでTINETを展開します(4～5行目)．次に，ディレクトリcfgで，TOPPERS/JSPカーネルのコンフィギュレータを生成し(6～7行目)，ディレクトリtinet/cfgで，TINETのコンフィギュレータを生成します(8～9行目)．

● TINETのディレクトリ構成とドキュメント

リスト2にTINETのディレクトリ構成を示します．また，tinet/docには**表1**に示すTINETのドキュメント類が入っています．

● サンプル・プログラムの生成

TINETに付属するサンプル・プログラムnservを生成します．まず，IPアドレスを設定します．**リスト3**に示すファイルnserv/tinet_app_config.hの106～108行目にあるIPv4アドレス，サブネット・マスク，デフォルト・ゲートウェイのIPv4アドレス

リスト1　JSPカーネルとTINETの展開とコンフィギュレータの生成

```
1$ tar xvfz h8mon-1.12.tar.gz
2$ h8write -3069 -f20 h8mon/mon3069/mon3069.mot com1
3$ tar xvfz jsp-1.4.1.tar.gz
4$ cd jsp
5$ tar xvfz ../tinet-1.2.1.tar.gz
6$ cd cfg
7$ make
8$ cd ../tinet/cfg
9$ make
```

注1：http://www.toppers.jp/
注2：http://www.mit.pref.miyagi.jp/

見本

第14章

ダイナミック・ローディング対応 TOPPERS/RLL

河合 孝夫

1. TOPPERS/RLL開発の背景

● ネットワークの普及

いままでの組み込み機器は，ソフトウェアが小規模であったため，品質や信頼性を確保することが比較的容易でした．また，一度出荷された機器のソフトウェアは基本的には変更しないことも前提でした．ところが最近では，携帯電話機でとくに顕著に見られるように，機器の多機能化・ネットワーク化により，組み込まれているソフトウェアの大規模化および複雑化が急速に進み，品質・信頼性の確保が問題になっています．

● 1リンク・モデルだったITRON仕様OS

いままでの組み込み機器では，ソフトウェアを最小限のハードウェア資源で効率良く動かすことに主眼を置いたリアルタイムOS（RTOS）が使われてきました．このリアルタイムOSとして国内で大きなシェアを占めているのがITRON仕様OSです．ITRON仕様では，組み込みソフトウェアを更新するローダ機能は仕様策定範囲に含めていませんでした．

これは，いままでの組み込み機器は，人がROMにOSとアプリケーション・プログラムをリンクして書き込み，ROMでプログラムが実行されるという前提があったからです．このROMに書かれたプログラムはROM交換以外にソフトウェアを更新する手段を持ち合わせていませんでした．しかし，フラッシュ・メモリ・デバイスの低価格化，高集積化に伴い，組み込み機器においてもOS自身によるソフトウェアのバージョンアップが可能となってきました．

● バージョンアップの必要性

新機種モデルや新製品をいち早く市場に投入したいというビジネス面の要求は，ますます大きくなってきています．そのため，一定の品質基準・機能完成度をもって出荷することが難しくなってきています．このような状況下で，一度出荷した機器に不具合があった場合は機器を回収してソフトウェアの修正を行わねばならず，大きな保守コストが必要になります．

さらに，来たるべきユビキタス時代においては，マイクロコンピュータを使い，通信・ネットワーク機能をもったさまざまな機器が家庭やオフィス，工場などのあらゆるところに出現すると想像されます．「どこでもコンピューティング」環境においては，機器が相互にネットワーク接続されていることから，機器の購入後にユーザがソフトウェアのバージョンアップを行うことが不可欠になるでしょう．

2. 基本スキームについて

ダイナミック・ローディング機能をITRONで初めて実現したのが，TOPPERS/RLLカーネル（旧称：TOPPERS/IDLカーネル）です．TOPPERS/RLLカーネルは，機器に組み込まれているソフトウェアの不具合または機能更新・追加が必要な場合に，機器ユーザがネットワーク機能を使って部分的にモジュールをダウンロードし，バージョンアップする機能を備えています．目的は，今後の携帯電話機や無線通信技術を応用したユビキタス機器に対応する新しい保守用機能を提案することです．また，現在の組み込みソフトウェアの不具合対策および品質問題に対して，即効的な解決策を提供することも目的です．

TOPPERS/RLLカーネルでは，メモリ管理の手法として，すべてのプログラムで単一の物理アドレス空間を共有し，実行位置に依存したプログラムを動的にロード実行するしくみを採用しました．これにより，多くのプロセッサや組み込みシステムに適応することが可能であると考えました．

見本