新連載

最新T-Kernelの活用テクニック

第1回 小規模システム向けμT-Kernelとは何か

由良 修二

近年，μITRONを拡張する形でT-Kernelが誕生した．T-Kernelは，Single One Sourceによるリファレンス実装の提供や強い標準化，拡張機能の提供などの面で注目されている．本連載では，T-Kernelの最新拡張機能を中心に解説を行う．第1回目の今回は，8ビット～16ビットCPUを視野に入れた小型組み込み機器向けOSであるμT-Kernelを取り上げる．（編集部）

1. μT-Kernelとは何か？

2007年3月28日にμT-Kernel（マイクロティー・カーネル）が一般公開されました（**図1**）．μT-Kernelは，T-Kernelと同じITRONの流れを汲むリアルタイムOSです．これは，**図2**に示すようにT-Kernelの小型版という位置付けになります．

今回はμT-Kernelが必要となった背景と，T-KernelやμITRONとの違いについて解説します．

● μT-Kernelの元となったT-Kernelとは

μT-KernelはT-Kernelが元となって開発されていますが，そもそもT-Kernelとはどのような OSなのでしょうか．

T-Kernelは，昨今のハードウェアの進化を前提に，高機能化・複雑化・大規模化する組み込み機器に対応したプラットホームとなるべく開発された組み込みシステム向けOSです．T-Kernelとともに，T-Engineというハードウェア・プラットホームも同時に開発されました．開発プラットホームとして利用できるハードウェアとリアルタイムOSを標準化することでデバイス・ドライバやミドルウェアの流通プラットホームとすることを目標としています．

上位プログラムを複数のプラットホームで流通させるためには，基礎となるソフトウェアであるOSの挙動をできる限り一定にしておく必要があります．そこでT-Kernelでは「Single One Source」という方式を採用しました．Single One Sourceとは，文字通りソース・コードを一つ

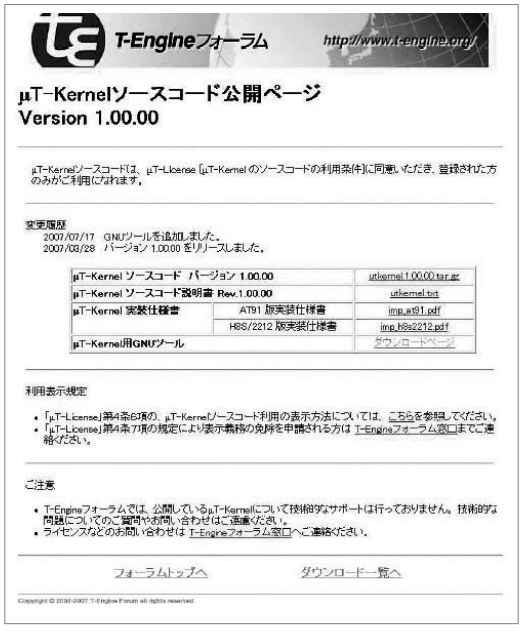

図1 μT-Kernelのダウンロード・ページ

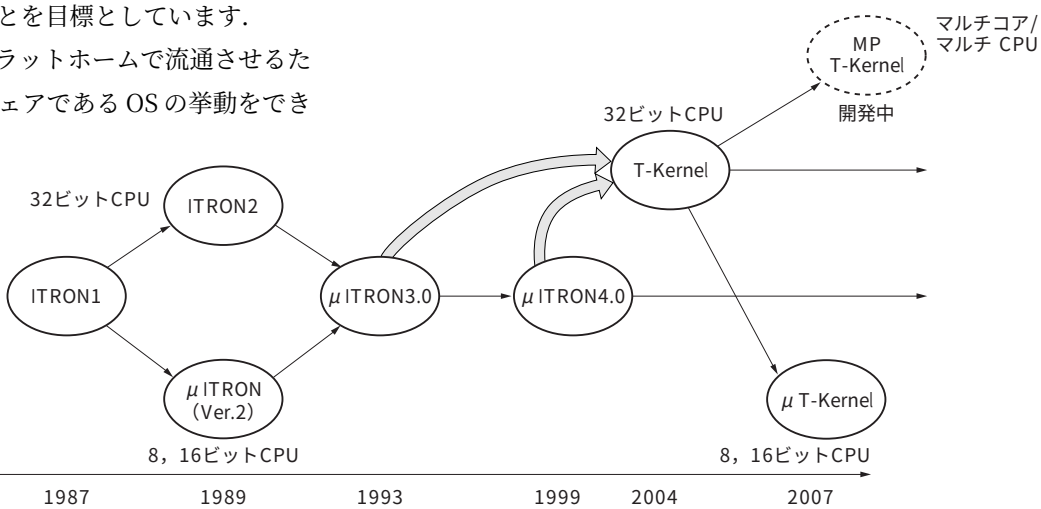

図2 ITRONとT-Kernelの流れ

KEYWORD ── T-Kernel，T-Engine，μT-Engine，μITRON，リアルタイムOS

にすることです．さらにリファレンスとなるソース・コードを無償公開することで，仕様書だけでは規定しきれない挙動も含めてできる限り挙動を一定にしようとしています．しかも，T-Kernelにはサブセットもありません．これにより，T-KernelではITRONで懸案となっていた移植性の問題を解決しています（**図3**，**図4**）．

また，T-Kernelでは上位のソフトウェアはカーネルから分離してT-Kernel/Extensionとして実装します．これにより，リアルタイム性を損なわずに任意の機器に適した構成への拡張が可能となります．

特に，大規模システムにおいて標準的に利用されるプロセス制御やファイル・システムといった機能は，T-Kernel/Standard Extension（以下，T-Kernel/SE．**図5**）としてT-Engineフォーラムが無償でリファレンス・コードを公開しています．T-Kernel/SEを利用すると，ITRONでは標準化できなかった高レベルのAPI標準化も可能です．T-

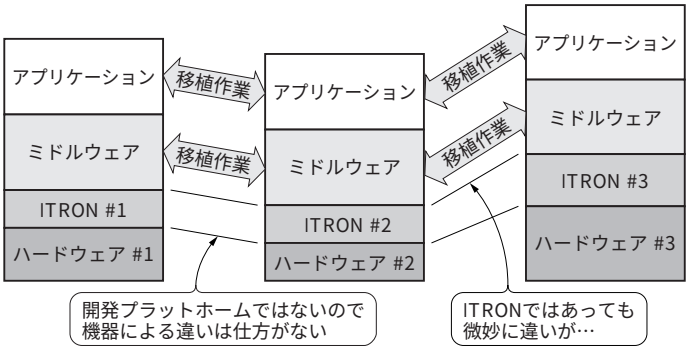

図3　ITRONにおける移植

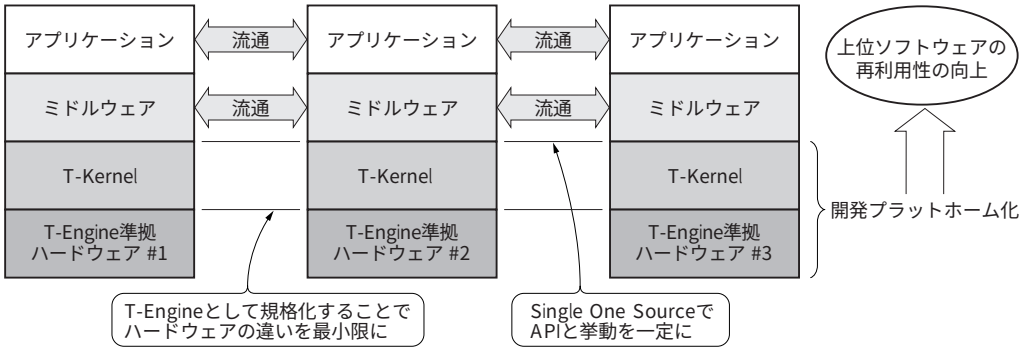

図4
T-Engineにおける移植性の向上

コラム1「弱い標準化」から「強い標準化」へ

ITRONの仕様書は社団法人 トロン協会から公開されていますが，その実装は各社（個人の場合もある）に任されています．

- TRON関連資料
  `http://www.assoc.tron.org/jpn/itron-doc.html`
- ITRON仕様準拠製品登録制度 登録製品一覧
  `http://www.assoc.tron.org/jpn/product/list.html`

このため，同じITRONと呼ばれるOSであっても，実装ごとに少しずつ挙動に違いがあります．その上，ITRON仕様では対象とするシステムに合わせてサービス・コールのサブセット化も可能です．これは「弱い標準化」という考え方に基づいた実装方法です．

ITRONの初期の仕様が決められたのが1987年，今から20年も前（パソコンでさえ16ビットCPUが出始めたころ）であり，当時の組み込み機器のCPUパワーを考えると，実用的な機器を開発するためには必要最低限の内容だけを決める弱い標準化という考え方は妥当であったといえます．

ところが，この弱い標準化には移植性を阻害するという弊害があります．ITRONが普及していろいろなソフトウェアが作られるようになりましたが，それを別のITRONに移植しようとすると余計な作業が発生するため，ソフトウェアの再利用が思うように進みませんでした．

一方，近年の組み込み機器は機能の複雑化と納期の短縮が進み，ソフトウェアの生産性の向上が至上命題となってきました．

T-Engineプロジェクトではソフトウェアの再利用性を向上させることでこの問題を解決しようと考えています．ハードウェアの仕様（T-Engine）からOS（T-Kernel）までをトータルに標準化してソフトウェアの移植の手間を最小限に抑え，再利用性を向上させることにしました．これを「強い標準化」と呼び，同プロジェクトの基本的なコンセプトになっています．