# T-Kernel/Standard Extensionとは?

豊山 祐一（とよやま ゆういち）
YRPユビキタス・ネットワーキング研究所

T-Kernelの機能を拡張するT-Kernel/Standard Extension（以下、T-Kernel/SE）の一般公開がいよいよ近づいてきました。本稿では、T-Kernel/SEとはいったいどのようなソフトウェアか、その概要を解説します。

## T-Kernelを拡張するExtension

T-Kernelは新世代の組込みシステム向けのリアルタイムOSです。組込みシステムの世界で長年にわたり使用され多くの実績を持つμITRONの技術を継承しつつ、日増しに大規模化・高機能化していく組込みシステムに対応すべく新たに設計されました。

組込みシステムのOSに要求される機能は、μITRONが設計され普及した時代とは比べものにならないくらい多くなっています。かつては、一部のシステムで使われるのみであったファイルシステムやネットワークの機能も一般的になりつつあります。大規模化したソフトウェアを管理するために、高度なメモリ管理や資源管理の機能も要求されるようになりました。しかし、これらの機能をただT-Kernelに組み込んで行くことは、OS自体の肥大化につながり、μITRONの長所であった軽量・高応答性を損なうことにもなります。組込みシステムの製品は非常に幅が広く、高機能なものはファイルシステムやネットワークなどの機能を要求する一方で、昔ながらの軽いシステムを要求するものも多々あります。

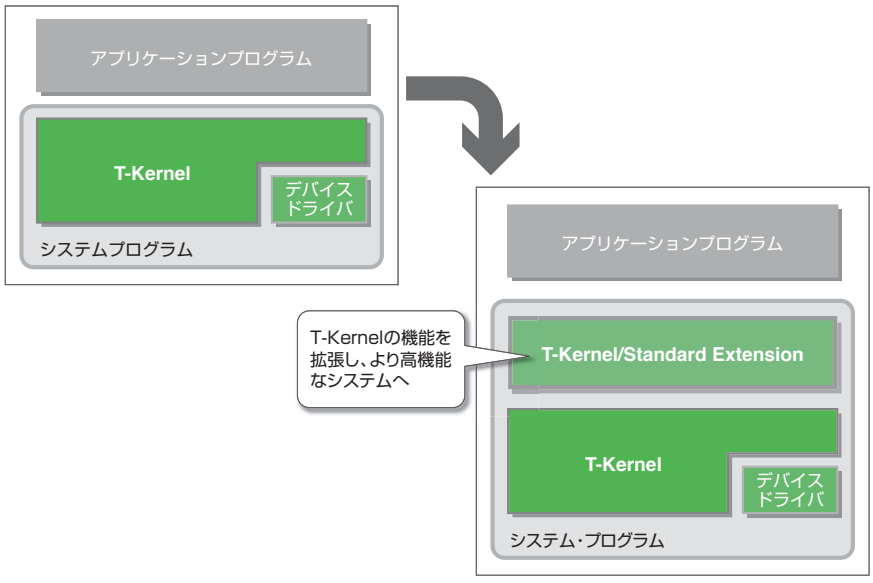



図1　Extensionによりシステムを拡張

これらの要求に応えるためにT-Kernelでは、カーネル本体にはリアルタイムOSとして基本的な機能のみを実装し、ファイルシステムなどの拡張機能はT-Kernel Extensionという形で提供することとしました。これは、T-Kernelをマイクロカーネル的に使用して、Extensionと合わせて高機能なシステムを構築するととらえることもできます（図1）。

T-Kernelにはサブシステムという自身のOS機能を拡張するソフトウェアモジュールをサポートする機能があります。通常、Extensionはこのサブシステムの機能を利用して実現されます。実際にはExtensionの実体は、複数のサブシステムから構成されることでしょう。

また、以上のしくみからもわかるように、T-KernelのExtensionは交換が可能です。異なったExtensionを利用すればT-Kernel上に別の高機能なシステムを構築できます。将来的には、さまざまなExtensionの中から構築するシステムに応じて最適なExtensionを利用する、といったことも可能となるかもしれません。

Extensionの中でも、T-Engineフォーラムにより標準的なExtensionとして開発されたものを、Native Extensionと呼びます。本稿で紹介するT-Kernel/SEはこのNative Extensionのひとつとなります。

また、各ベンダーが独自に開発しているものをPorted Extensionと呼びます。たとえば、MontaVista社が発表したT-LinuxはこのPorted Extensionです。

## T-Kernelの標準Extension

T-Kernel/SEは、Standard Extensionという名前のとおり、T-Kernelの標準のExtensionとして設計・開発されました（図2）。

対象としては、情報家電や次世代携帯電話などの組込みシステムの中でも高度な機能が要求される分野を想定しています。この分野は、PCやサーバで使用される情報系OSに近い機能が要求される一方で、組込みシステム本来のリアルタイム性能も要求されます。また、従来のμITRONでは力不足がささやかれている分野でもあり、T-Kernel/SEが期待されるところでもあります。

表1にT-Kernel/SEの主な機能を示します。

T-Kernel/SEの重要な機能として、第一に大規模なプログラムの開発に対応できるメモリ、資源の管理機能があります。T-Kernel/SE上で動作するアプリケーションプログラムはプロセスと呼ばれる単位で管理されます。個々のプロセスは独立した論理アドレス空間上で動作します。またさまざまな資源もプロセスの単位で管理されます。T-Kernel自体でもMMUを用いてタスクごとに異なった論理アドレス空間で実行したり、資源をリソースグループとして管理する機能はありました。ただし、T-Kernel単体でのこれらの機能はごく基本的なもので、そのままユーザーのアプリケーションから使用するには機能不足です。T-Kernel/SEはこのT-Kernelの基本機能をもとに、プロセス管理といった上位の機能を実現しています。プロセス管理に付随して、プロセス間のメッセージ機能やグローバル名機能、タスク間同期・通信機能などがあります。プロセスに関しては次項でもう少し詳しく説明したいと思います。

第二に重要な機能として、ファイルシステムがあります。ファイルシステムには、TRONのプロジェクトにおいて研究・開発されてきた実身／仮身をベースとしたTRONファイルシステムを標準でサポートしました。このファイルシステムは、一般的な階層ディレクトリ構造を持たず、より柔軟なファイル間のリンクによるネットワーク構造を持ちます。

ただし、他のシステムとの互換性を維持するため、組込みシステムで多く利用されているFATファイルシステムやCD-ROMファイルシステム（ISO9660）などにも対応しました。さらにその他のファイルシステムの追加も容易にできるしくみとなっています。また、複数のファイルシステムが混在した環境で、それぞれの違いを極力意識することなくファイル操作できるように、標準入出力管理の機能を持ちます。

なお、T-Kernel/SEではファイルシステムの使用は必須ではありません。組込みシステムでは外部記録装置を持

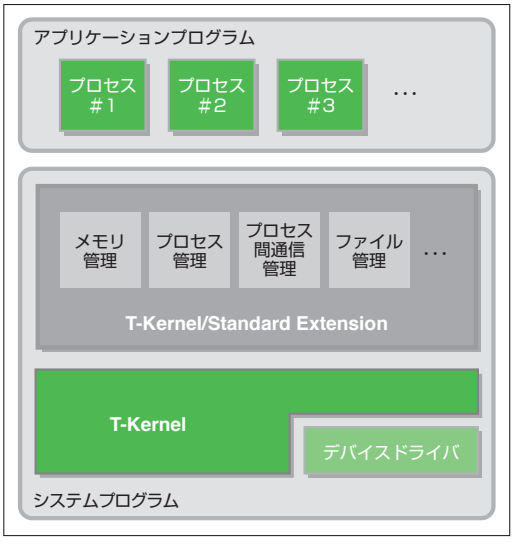



図2　T-Kernel/SEを用いたシステムの概略

表1　T-Kernel/SEの主な機能

| | |
|---|---|
| メモリ管理機能 | ファイル管理 |
| プロセス／タスク管理機能 | デバイス管理機能 |
| プロセス間メッセージ機能 | イベント管理機能 |
| グローバル名機能 | 時計管理機能 |
| タスク間同期・通信機能 | システム管理 |
| 共有ライブラリ機能 | |

たない製品も存在します。よってT-Kernel/SEでは、たとえばディスクレスのシステムでもROMベースで動作可能なように考えられています（図3）。

このほかにデバイス制御に関連して、デバイスの直接の制御を行うデバイス機能、デバイスからの非同期な情報を統一的に扱うイベント管理機能などがあります。イベント管理機能は、T-Kernel/SE上にユーザーインタフェースを構築するために重要な機能です。ただし、T-Kernel/SEの対応する範囲にはユーザーインタフェースは含みません。

ネットワーク機能も今回のT-Kernel/SEのリリースには含まれません。T-Kernel/SE上で動作するTCP/IPプロトコルスタックは開発中であり、追って公開される予定です。

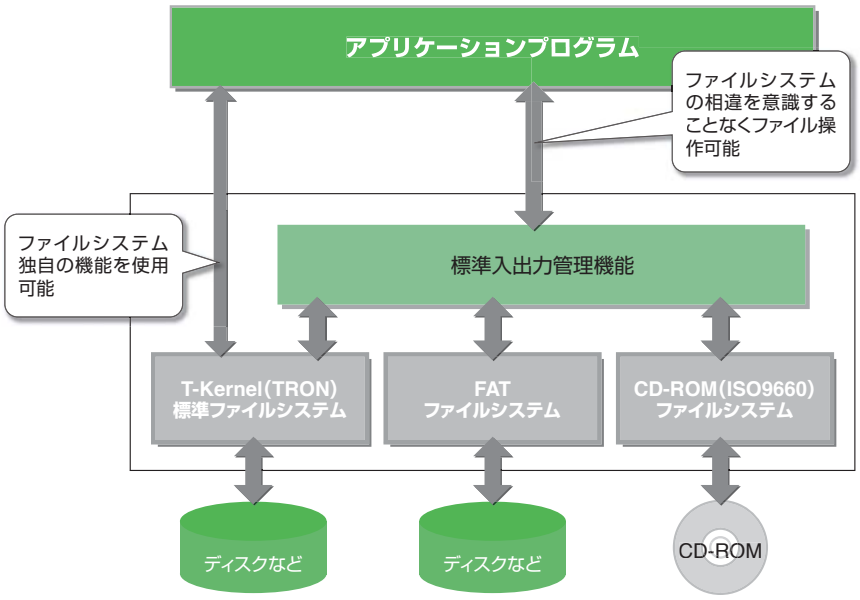



図3　T-Kernel/SEのファイルシステム

## プロセスとタスク

T-Kernel/SE上で動作するアプリケーションプログラムは、プロセスから構成され、さらにプロセスはタスクから構成されます。

T-Kernel/SEのプロセスとタスクについて説明する前に、プロセスモデルとスレッドモデルについて簡単に説明します。この言葉はT-Kernelの用語ではありませんが、一般にプログラムの並列実行のしくみを表すのに使われます。プロセスとスレッドというのは、どちらも並列実行するプログラムの単位です。両者の大きな違いは、プロセスモデルではメモリなどを含む資源がプロセスごとに独立しているのに対し、スレッドモデルではこれらがスレッド間で共有されていることです。

具体的には、プロセスは通常、独自のメモリ空間で動作します。他のプロセスのメモリ空間は、より低水準なレベルで保護されており、勝手にアクセスすることができません。プロセス間での情報のやり取りは、OSが提供するプロセス間の通信機能を使用するしかありません。Linuxなどの情報系のOSで主に使用されているのが、このプロセスモデルです。

これに対して、スレッドは、共通のメモリ空間で動作します。よって、スレッド間で情報はある程度自由に共有することができます。従来の組込みシステムでよく用いられたのがこのスレッドモデルです。μITRONのタスクは、スレッドと同等と言えます（図4）。

プロセスモデルとスレッドモデルはそれぞれ利点、欠点があります。プロセスモデルの利点はプロセスの独立性の高さにあります。プロセスはOSの提供する機能を用いてのみ他のプロセスと結合していますので、モジュールとしての独立性はとても高くなります。また、メモリ空間が異なっているため、たとえあるプロセスが暴走したとしても他のプロセスへの被害は最小に抑えることができます。プロセスモデルの欠点は、プロセスの切り替えやプロセス間通信の際のオーバーヘッド時間が、スレッドモデルに比べて大きくなることがあげられます。

スレッドモデルの利点、欠点はそのままプロセスモデルの利点、欠点の裏返しになります。スレッドは切り替えがプロセスに比べて素早く、情報の共有が簡単ですが、そのぶん、スレッド間の結びつきが強くなりやすく、またお互いの影響を簡単に受けます。

これまで、組込みシステムでは主にスレッドモデルが、情報系ではプロセスモデルが使われてきたのは、以上のような特徴を踏まえてのことです。た

だし、情報系の多くのOSでもプロセスとともにスレッドも使用できるようになってきました。この場合、あるプロセスの中に複数のスレッドが存在し、そのプロセスのメモリ空間や資源を共有して動作します。つまりプロセスの中のプログラムの実行単位としてスレッドが存在していると言えます。

T-Kernel/SEに話を戻しますと、T-Kernel/SEのプロセスは、ここまで説明してきたプロセスモデルのプロセスそのものです。それぞれのプロセスは独立したメモリ空間で実行されます。そして、T-Kernel/SEでスレッドの相当するものがタスクです。

T-Kernel/SEではタスクはプロセス内におけるプログラムの実行単位であり、1つのプロセスの中に必ず1つ以上のタスクが存在します。プロセスを生成すると自動的にメインタスクと呼ばれるタスクが生成されます。この状態では、プロセスとそのメインタスクは同一と考えて良いでしょう。つまりそのプロセスを実行するということはメインタスクを実行することとなります。T-Kernel/SEのサービスコールを用いると、プロセスの中にさらにタスクを生成することができます。後から生成されたタスクは、サブタスクと呼ばれ、同一のプロセス内のタスクとメモリ空間や資源を共有します。これは、前述の情報系OSにおいて、プロセス内に複数のスレッドが存在するモデルと同等と考えられます（図5）。

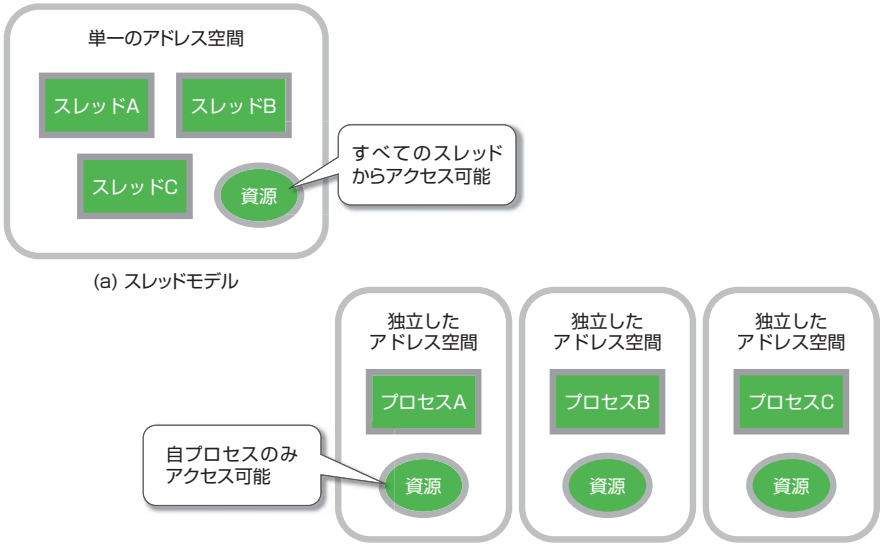



図4　プロセスモデルとスレッドモデル

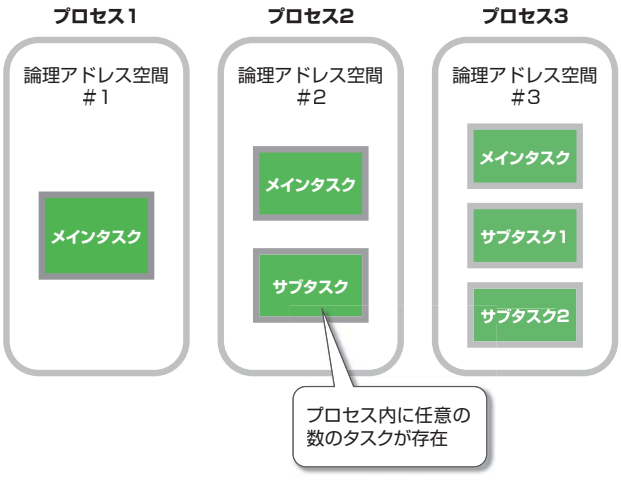



図5　T-Kernel/SEにおけるプロセスとタスク

## タスクとスケジューリング

前項では一般的なプロセスとスレッドのモデルの観点から、T-Kernel/SEのプロセスとタスクを説明してきました。ここではT-Kernelの観点から、T-Kernel/SEのプロセスとタスクを見ていきたいと思います。

T-Kernel/SEのタスクは、基本的にはT-Kernelのタスクと同一のものです。T-KernelでもT-Kernel/SEでも、プログラムを実行する基本単位はあくまでタスクであり、スケジューリングはタスクに対して行われます。ではプロセスは何かと言えば、メモリ空間を含む資源管理の単位ととらえることができます。

T-Kernel単体の機能として、タスク

固有のメモリ空間とリソースグループを、個々のタスクに割り当てる機能があります。T-Kernel/SEではこの機能を利用して、プロセスごとにタスク固有空間とリソースグループを生成し、そのプロセスに属するタスクに割り当てています。

つまり、T-Kernel/SEのプロセスはT-Kernelから見れば、1つのタスク固有空間と1つのリソースグループ、そしてそれに属する1つないし複数のタスクととらえることができます（図6）。

すでに述べたようにT-Kernel/SEでは、スケジューリングはタスクの単位で行われ、プロセスは関係しません。プロセスに設定された優先度は、メインタスクの優先度となります。プロセスに1つしかタスクが存在しない場合は特に意識することはありませんが、プロセスに複数のタスクが存在する場合のスケジューリングは、タスクの単位で行われることを意識する必要があります。

たとえば、プロセスAの優先度が5で、プロセスBの優先度が10であったとします。両プロセスが実行可能であった場合、まず実行されるのはプロセスAのメインタスクです。しかし、プロセスBに優先度3のサブ・タスクが存在した場合、このサブタスクのほうが実行されます。スケジューリングを決めるのは、各タスクの優先度ということになります。

T-Kernel/SEのタスクスケジューリングについて、もう少し詳しく説明します。

T-Kernel/SEのタスクは、T-Kernelと基本的に同一であり、タスクのスケジューリングはT-Kernelのスケジューリングの機能をベースに実現されています。ただし、T-Kernel/SEによる機能の拡張も行われているため、両者は完全に同じではありません。最も大きな違いは、T-Kernelではタスクのスケジューリングは、絶対優先度によるもののみですが、T-Kernel/SEでは時分割によるラウンドロビンスケジューリングも対応します。

ラウンドロビンスケジューリングは、CPU時間を分割してタスクに割り振り、各タスクを順番に実行していきます。情報系のOSで一般的に使われ、並列動作するタスク（プロセス）を独立したものとして扱うという点でプロセスモデルと相性が良いと言えます。しかし、非同期に発生するイベントに対する応答性に問題があり、リアルタイム制御を必要とする組込みシステムには不向きとも言えます。

一方、組込みシステムのリアルタイムOSでは絶対優先度によるスケジューリングが一般的です。こちらはリアルタイム性に優れる反面、上位のアプリケーションのように比較的独立したプログラムを並列で動作させるには不向きといえます。実際にこのような場合は、絶対優先度によるスケジューリングを行うOSでも、タスクに同一の優先度を割り当て、タイムイベントなどを利用しラウンドロビンを行うのが普通です（図7）。

T-Kernel/SEでは、タスクに割り振られた優先度により、絶対優先度グループ、ラウンドロビングループ1、ラウンドロビングループ2の3つのグループに分けます。グループ間の優先度は、絶対優先度グループ

＞ラウンドロビングループ1

＞ラウンドロビングループ2

となり、それぞれの優先度グループの中で、対応したスケジューリングが行われます（図8）。

よってT-Kernel/Extensionでは、リアルタイム性の高い処理を行うタスクは絶対優先度グループに、逆にリアルタイム性を要求しないアプリケーションなどはラウンドロビングループに割り振るといったことが可能となります。

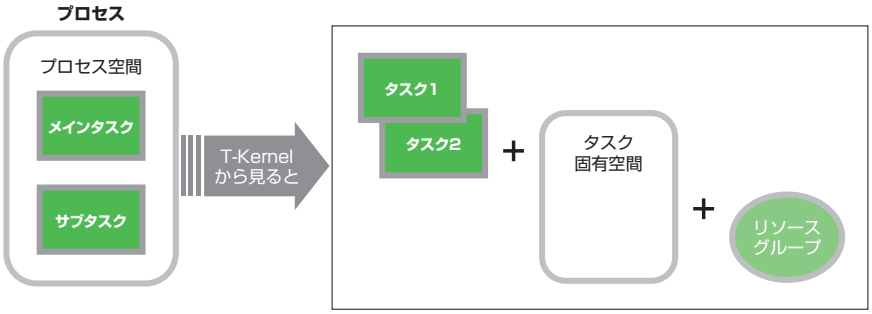



図6　T-Kernel/SEのプロセス

## T-Kernelとの親和性・互換性

ここまでは、T-Kernel/SEがT-Kernelの機能を、リアルタイム性能を維持したまま、情報系OSに匹敵するまで拡張することを説明してきました。T-Kernel単体やμITRONでは対応の

難しかった高機能・大規模な組込み分野でもT-Kernel/SEを使用することにより対応が可能となります。

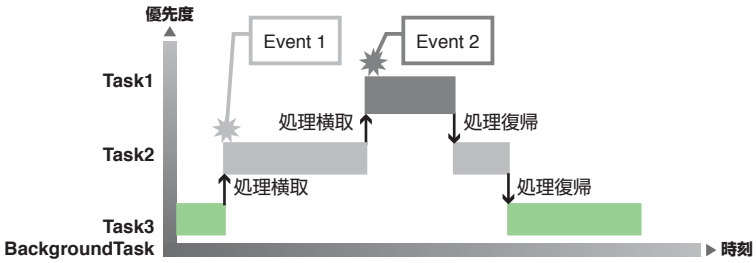
ここでもうひとつ、T-Kernel/SEの重要な特徴として、T-Kernelとの親和性、互換性を重視して開発されているということがあげられます。これは、T-Kernel/SEを用いたシステムと、これから説明するハイブリット型の組込みシステムOSとの大きな違いでもあります。

ハイブリット型の組込みシステムOSとは、情報系OSとリアルタイムOSを組み合わせて使用することにより、両者の良いところを利用し、欠点を補おうというものです。よくある例では、Linuxとμ ITRONのハイブリットなどがあります。リアルタイム性が必要な部分にはμITRONを使用し、さほどリアルタイム性が重要でない上位のアプリケーション部分などはLinuxを使用するわけです。

このようなハイブリット型OSは、ある面ではとても実用的と言えます。既存のLinuxのソフトウェアを使用したのですが、Linuxだけではリアルタイム制御ができない、というような場合にはまさにうってつけとも言えます(図9)。

ハイブリット型OSのアプローチは、T-KernelとT-Kernel/SEに似ている部分もあります。しかし、大きな違いはハイブリット型OSはまったく異なった2つのOSを動かそうとしている点です。両者のOSでは、APIも、プログラムモデルも、ちょっとした作法までが異なります。たとえばデータの型定義からエラーコードの体系など細々としたことまで異なるのです。1つのシステムを作り上げるには両方のOSの知識が必

**絶対優先度スケジューリング**　ITRON、T-Kernelなどの一般的なスケジューリング




**ラウンドロビンスケジューリング**　PC、サーバなどの情報系OSの一般的なスケジューリング

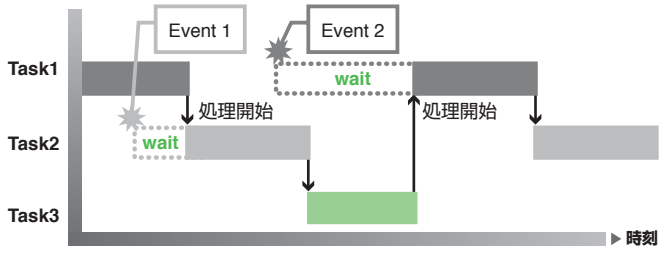



図7　タスクのスケジューリング


図8　T-Kernel/SEのスケジューリンググループ

要となり、プログラム開発の負荷を増大させる可能性があります。また長期的にみれば、2つのOSをメンテナンスし続けなければならないデメリットも小さくはありません。

この点、T-Kernel/SEは、T-Kernelの純粋な機能拡張として同一の体系上に作られており、タスク間の同期・通信機能やデバイスの管理機能など、T-Kernel/SEとT-Kernelは極力互換性を保つように設計されています。よってT-Kernel単体上に作られたタスクをT-Kernel/SEに移植することは簡単です。さらに、そもそもT-Kernel/SE自体がT-Kernelの上で動作していますので、デバイスドライバやサブシステムは、T-Kernelと同じものが使用可能です。多くのミドルウェアもそのまま使用が可能となります。

このようにT-KernelとT-Kernel/SEは、ソフトウェア資産やノウハウを共有することができます。比較的小規模な組込みシステムはT-Kernelのみで、規模の大きなシステムはT-Kernel/SEを使う、といったスケーラビリティのある対応も可能となるのです（図10）。

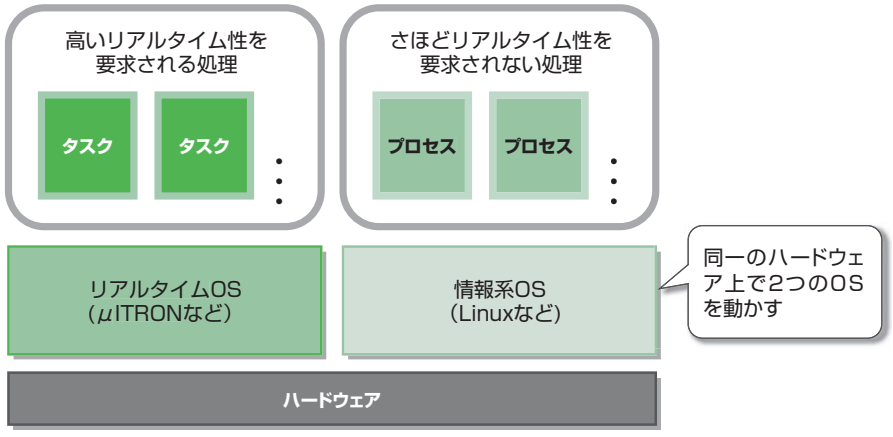



図9　ハイブリット型の組込みOS

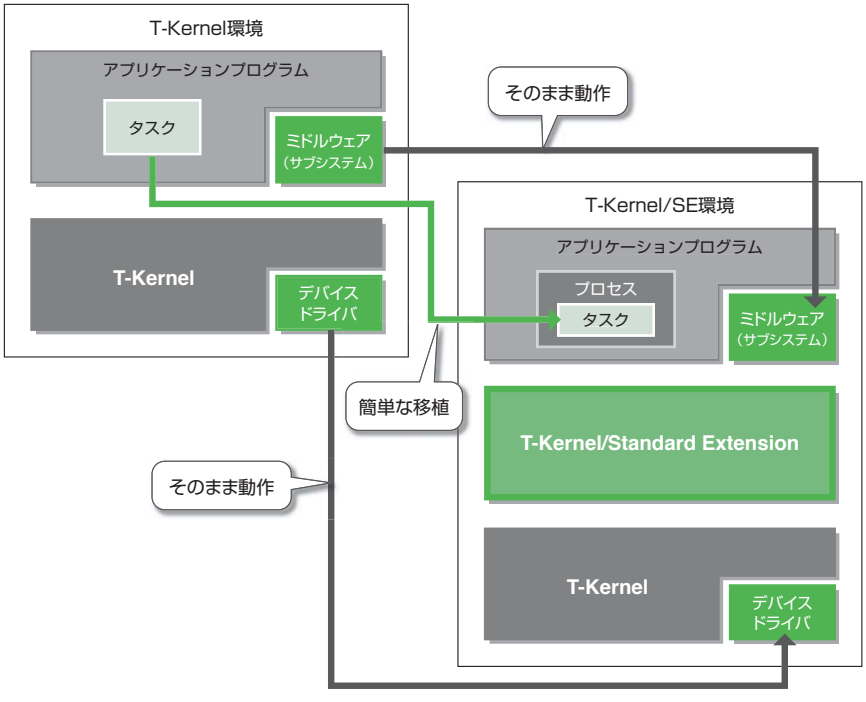



図10　T-KernelからT-Kernel/SEへ

## 終わりに

T-Kernel/SEはすでに2年近くT-Engineフォーラムにおいてベータ版の評価が行われてきました。この間に、実際に製品への適応を含めた数々の試みが行われ、その結果が反映され改善されています。そして、いよいよ一般公開が間近となりました。

T-Kernel/SEの一般公開は、T-Kernelに準じて誰でもが自由に使えるオープンなものになる予定です。今まで、T-Engine、T-Kernelを使い続けてきた方々、またT-Kernel単体やμITRONに機能的に不満を感じていた方々もぜひこの機会にT-Kernel/SEを試してみてください。