時間保護のための組込みシステム向け階層型リアルタイム OS

―リアルタイムアプリの統合を実現する OS―

1. 背景

近年，組込みシステムの高性能化・高機能化により，一つのプロセッサに複数のアプリケーションを統合したシステムが増加している．アプリケーションの大規模化により，システムに搭載するアプリケーションを複数のグループやメーカにより，別々に設計・開発・検証することが多い．このようなアプリケーションを従来のリアルタイム OS を搭載したプロセッサ上に混在させると，設計・スケジューリング方針の違いやアプリケーション間に保護機能がないなどの理由から，あるアプリケーションの実行時間の遅れがシステム全体へ波及してしまう．この問題は，ハードリアルタイム性を要求されるシステムでは致命的となるため，個々のアプリケーションのプロセッサ時間を保護する機能（これを時間保護機能と呼ぶ）が必要となる．時間保護機能により，異なる設計・検証レベルで開発されたアプリケーションを一つのプロセッサ上に容易に統合することが可能となる．我々は，時間保護機能の理論的なスケジューリングアルゴリズムの研究をベースに，2005 年度未踏ソフトウェア事業の支援を受けて，階層型スケジューラを搭載した「時間保護機能をもつリアルタイム OS」を開発した．

2. 目的

今回の未踏ソフトウェア事業では，2005 年度に開発したμITRON 仕様準拠の時間保護 OS の完成度を向上させるため，OS の構造を整理し，機能を容易に追加できるよう構造化を進めた．さらに，現在までの研究・開発経験から得られた問題点・改良点を考慮し，時間保護 OS の適用範囲の拡大と実用性の向上に重点を置いた機能を追加した．なお，本プロジェクトでは TOPPERS/JSP カーネルの開発で蓄積された実装技術及び，自動車業界からの要求事項を積極的に取り入れ，実用を念頭に置いた基盤的リアルタイム OS の開発を目指した．

3. 開発の内容

本プロジェクトでは、2005 年度に開発した時間保護 OS をベースに実用性の向上を目的とした機能拡張を行った．プロセッサ時間の保護に関する従来手法の問題点についてはこれまでの開発の成果により，ある程度解決できたと考えている．今回は，これまで開発した時間保護 OS の課題を解決するために必要な機能を開発していくことに主眼を置いた．以下に主な開発内容を挙げる．

<u>階層型スケジューラの構造化</u>

昨年度開発したリアルタイム OS の階層型スケジューラでは，その設計上，アプリケーションスケジューラとタスクスケジューラ間のデータ依存性が高く，互いのデータ構造を参照しあっている（図 1 の BEFORE 参照）．リアルタイム OS 内のモジュール化を進めるため，アプ

リケーションスケジューラとタスクスケジューラ間のインタフェース（これを SPI：Scheduler Program Interface と呼んでいる）を標準化し，互いの独立性を高めるような構造化を行った（図 1 の AFTER 参照）．構造化により，アプリケーションスケジューラはタスクスケジューラの構造に依存することなくアプリケーションをスケジューリングすることができるようになった．その結果，アプリケーションスケジューラは，ある種のアプリケーションモニタのような位置づけとなる．スケジューラ間インタフェースを定めることにより，タスクスケジューラを追加できるようになった．

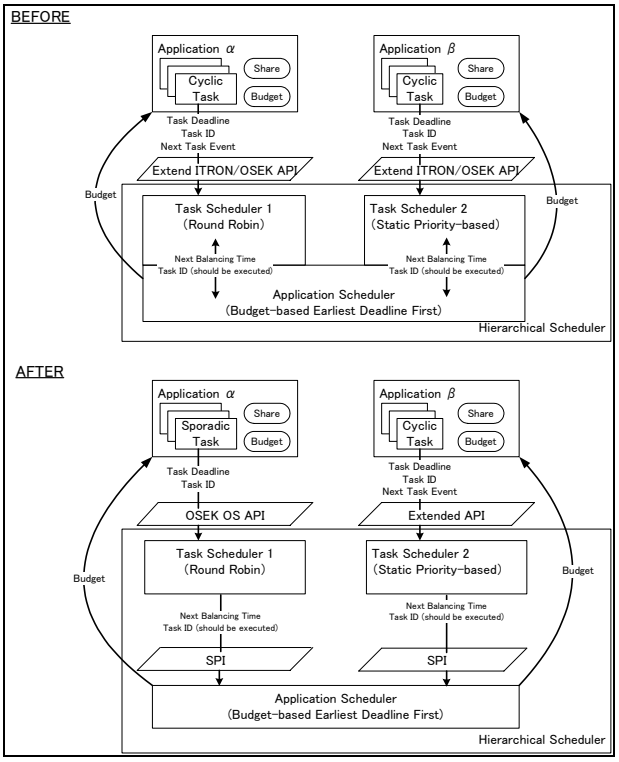

図 1：RTOS の内部構造の整理

割込み処理機構

2005 年度の成果では，アプリケーション毎に割り込みを扱う機能をもっていなかった．組込みシステムで動作するアプリケーションの多くは，さまざまなハードウエアからの割込み要求に応じた処理をもち，クリティカルセクションの実現に割込み許可・禁止の操作を行うこともある．そこで，図2に示すような割り込み処理機構を導入した．この割込み処理機構では，プロセッサ依存領域とアプリケーション領域の間に，アプリケーション毎に割込み許可・禁止する機能をハードウエア（ソフトコアの修正）もしくは，ソフトウェアにより実現する．これを導入することで，アプリケーションからはハードウエアを独占しているように見える．OS は割込み要求が発生した時点で，その割込みの受付けを許可しているアプリケーションの割り込みハンドラのみ実行する．

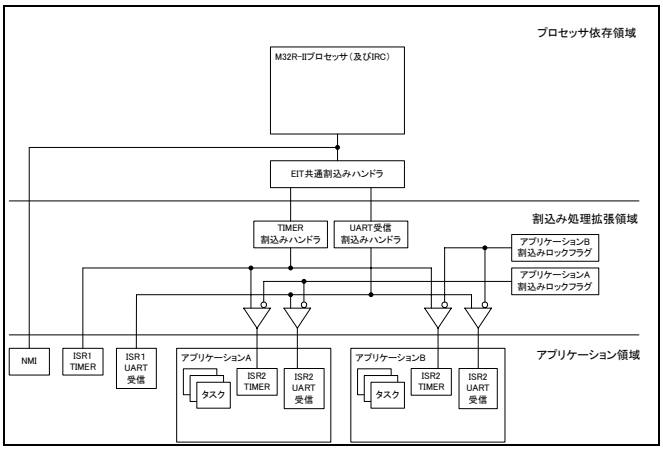

図2：割込み処理機構

非リアルタイムアプリケーション対応

今回の開発では，ハードリアルタイムアプリケーションだけではなく，非リアルタイムアプリケーションも容易に統合できるように階層型スケジューラを拡張した．図 1 の AFTER 図のように，非リアルタイムアプリケーション（左）とハードリアルタイムアプリケーション（右）が混在させることができる．ハードリアルタイムアプリケーションの場合には，これまでに開発したアルゴリズムを採用し，アプリケーションスケジューラに対してアプリケーションの時間制約を満たせる適切なタイミングでプロセッサ時間を要求する（SPI を利用する）．一方，非リアルタイムアプリケーションの場合には，設定されたプロセッサ利用率を保証すれば良いので，アプリケーションスケジューラ側で非リアルタイムアプリケーションに，定期的にプロセッサ時間を割り当てる機能を追加した．

4．従来の技術（または機能）との相違

リアルタイムアプリケーションを対象とした組込みシステム向けの時間保護機能をもつ RTOS に関しては，実用段階のものは未だに存在していない．ここでは，関連する研究段階の技術を紹介する．

デッドライン監視機能

リアルタイムアプリケーションを対象とした組込みシステム向けの時間保護を搭載したリアルタイム OS に関しては、実用段階のものは未だに存在しておらず，研究段階においても仕様が提案されている程度である．例えば，HIS(Hersteller Initiative Software:欧州の主要自動車メーカ・グループ)により公開されている OSEK/VDX 仕様を拡張した時間保護機能仕様は，実行可能状態にあるタスクがデッドラインまでに実行を完了することを監視するモニタリング機能に留まり，アプリケーションのプロセッサ時間の保護としては不十分である。開発した RTOS では，アプリケーションが実質的に使用したプロセッサ時間を管理し，別のアプリケーションのプロセッサ時間に影響することを防ぐことができる．

リソース管理手法の問題点

プロセッサ時間だけでなく，ネットワーク，メモリ領域などのリソースをアプリケーション（もしくはタスク，プロセス）毎に管理する手法が数多く提案されている．これらの手法では，アプリケーション毎にプロセッサ利用率を設定し，その利用率を越えるプロセッサ時間が得られないようにする仕組みを提供している．リソース管理手法では，アプリケーションに所属するタスクがデッドラインを満たすことは保証できず，ハードリアルタイム性を要求されるシステムに適用するには不向きである．
一方，我々の提案する時間保護機能では，アプリケーションごとのプロセッサ時間を保護するとともに，統合する前の環境で時間制約を満たすタスクは統合後にも時間制約を満たすことを保証している．また，今回のプロジェクトでは，非リアルタイム OS にも対応した．これらの点で，我々の開発した RTOS の方がハードリアルリアルタイムシステムに適用しやすく，適用範囲が広いと考えている．

マイクロカーネルを利用したハイブリッド型 OS の問題

マイクロカーネル上で複数のリアルタイム OS を動作させることで，各リアルタイム OS 上のアプリケーションのリアルタイム性を保証する手法がいくつか提案されている．しかし，多くのマイクロカーネルでは単位時間ごとに実行する OS を切り替えたり，プロセッサ利用率を割り当てて OS を切り替えたりするスケジューリング手法が採用されている．このような場合，リソース管理手法と同じ問題が発生すると考えられる．また，リソース制約の厳しい組込みシステムにおける時間保護機能の観点では，OS 自体を階層化するよりも，スケジューラのみを階層化するほうがメモリ使用量や，アプリケーションを切り替える際のオーバヘッドは少なくて済むと考えられる．

5．期待される効果

近年の組込みソフトウェアの大規模化・複雑化により，複数のベンダがアプリケーション・ソフトウエアを開発することが多い．本プロジェクトの開発成果を導入することにより，システム設計者はプロセッサ性能に応じて各アプリケーションのプロセッサ利用率を割り当てるだけで，ハードリアルタイムアプリケーションや非リアルタイムアプリケーションを容易に単一のプロセッサ上に統合することが可能になるため，開発コストを削減し開発期間を短縮することが期待できる．また，組込みシステム向け（特に，自動車制御システム）のオープンソースソフトウエアという文化の確立し，組込みシステムの発展に貢献していきたいと考えている．

自動車制御システムへの適用

自動車制御システムに時間保護を適用することにより，ECU と ECU 間通信のネットワークケーブル（ハーネス）の数を削減することが可能になる．例えば，エンジン制御用 ECU と AT 制御用 ECU などを一つ ECU に置き換えることが考えられる（図 3）．自動車制御の分野では，ECU 数の削減や制御アプリケーションの流動性向上を目的として，自動車メーカ，自動車部品メーカ，半導体メーカなどが中心となり，ソフトウエアプラットホームの標準化を検討している．例えば，欧州では AUTOSAR，国内では JasPar などの団体がある．提案する

時間保護は，アプリケーションの統合に必須の機能であることを述べた．そこで，時間保護機能を自動車制御システムに適用することが十分可能であると考えている．現段階では，自動車メーカとの共同研究により，本ソフトウェアに対する需要を確認している．今後はさらに，自動車制御システムへの具体的な適用を検討していきたいと考えている．

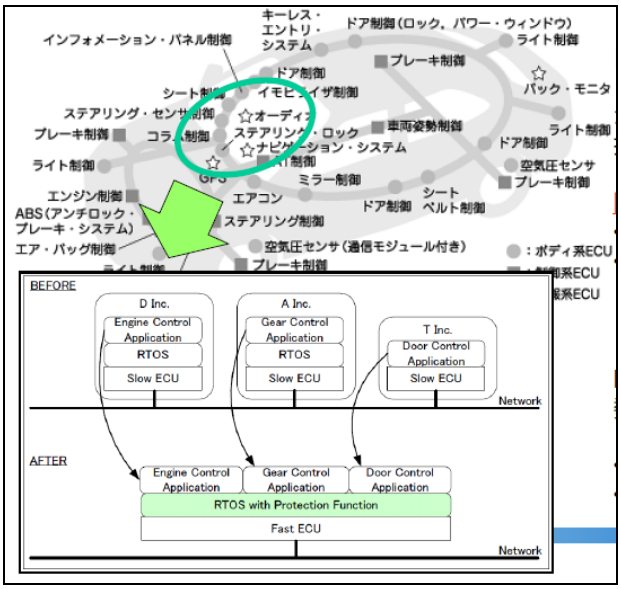

図 3：自動車制御システムにおける ECU の統合

6．普及（または活用）の見通し

現在，自動車メーカとの共同研究により，本ソフトウェアの有用性が確認されている．今後は，JasPar を通して他の自動車メーカや車載ソフトウエアメーカに対して本ソフトウェアを提案していく．また，実際に使用されているアプリケーションを用いた評価を検討していく．また，2006 年 3 月には TOPPERS プロジェクトの技術検討会議にて，本ソフトウェアの有用性を多方面の技術者と検討した．さらに，ソースコードレビューを行って完成度を向上させ，オープンソースソフトウエアとしての早期公開を目指す．

7．開発者名（所属）

松原豊（名古屋大学大学院情報科学研究科）
（参考）開発者URL：http://www.ertl.jp/~yutaka/