LSI IP デザイン・アワード応募書類表紙（企業）

タイトル： **オープンソース保護 OS：メモリ保護と時間保護を有する自動車向けリアルタイム OS**

技術分野： 自動車制御用 OS，リアルタイム OS ，メモリ保護 ，時間保護

応募者： 服部博行，大西秀一，片岡歩，松原 豊，高田広章

所属機関： 株式会社ヴィッツ,名古屋大学情報科学研究科

1. 研究・開発の目的・狙い

自動車が搭載している ECU（ Electronic Control Unit ）の数は年々増加し、国内の最高級乗用車が搭載する ECU はついに 100 個に達している。このように ECU 搭載数が増加している理由として、自動車に求められる利用者の要求が年々向上しているためである。すなわち、自動車はより安全に、より快適に、かつ、環境にやさしくい機能および性能が求められている。この要求を満たすために、自動車の各制御に ECU を利用して細密な制御を実現し、かつ、相互に情報を交換することにより連携のとれた制御を実現して、自動車に求められる要求に対応している。

一方で、ECU の増加に伴い、車両内での ECU 搭載空間が枯渇し搭載限界に達している。また、ECU 個数の増加に伴い、ECU 間を接続するワイヤーハーネスの増加とその重量増、さらに、通信接続性の問題やソフトウェア開発量の爆発的な増加など多くの問題を抱えるようになった。

これらの問題を解決及び軽減する方法として ECU 統合は不可欠であると考えられており、事実、国内自動車メーカでは一部の製品において、ECU 統合を実施している。本研究は、この ECU 統合を安全且つ容易に実現できる機能であるメモリ保護と時間保護を研究し、かつ、保護機能を有したリアルタイム OS を開発することにより、自動車制御開発が抱える問題を解決および改善するとともに、自動車制御業界へ貢献することを目的としています。

2. 研究・開発の概要

1） **利用分野：**統合自動車制御用電装部品

2） **特徴：**ECU 統合を安全，かつ，容易に実現可能なリアルタイム OS の研究および開発

3） **種類：**ソフト VC

4） **規模：**メモリ保護 OS　7,655 byte（ROM コードサイズ;M32R-Ⅱ）
時間保護 OS　25,217 byte（ROM コードサイズ;M32C/100）
※ ルネサステクノロジ製 M32R/M32C および同社製コンパイラを使用。

5） **性能：**メモリ保護 OS の RAM メモリ必要量 116 byte ， ROM メモリ必要量 854 byte
時間保護 OS の RAM メモリ必要量 1,198 byte ROM メモリ必要量 4,006 byte

3. 訴求点および効果

次世代の ECU 開発でキーテクノロジとなる保護 OS を世界に類を見ない手法で開発し、オープンソースソフトウェアとして公開する。この保護 OS には、メモリ保護と時間保護の機能を有し、メモリ保護は欧州の HIS（Herstellerinitiative Software ）が提案する OSEK OS Extensions for Protected Applications 仕様と比較し、より自動車制御に求められる性能を有する仕様を独自に策定した。同時に、自動車制御で十分に利用が可能なリアルタイム特性を確保するために、独自に仕様策定した MPU（ Memory Protection Unit ）をマイクロプロセッサに実装（ 仕様を提案し、ルネサステクノロジにてマイクロプロセッサの改造を実施 ）して実現している。また、時間保護は 2 種類の保護機能を実現している。 1 つは HIS が提案するデッドラインモニタリングで機能であり、HIS 仕様をそのまま実現している。しかし、HIS のデッドラインモニタリングは監視機能ではるが、時間保護とは言えない。そこで、2 種目の時間保護として階層型スケジューラ型の時間保護を実装した。この階層型スケジューラは名古屋大学で基礎的な研究を実施している段階であり、製品を意識した OS に搭載されるのは他に例がない。

これらの開発成果は、単なる研究に留めるのではなく、実製品への適用を検討しており、既に、複数の電装部品メーカと共に、実車を利用した実証実験を実施し、本保護 OS の機能的な検証および課題などについても調査および分析をし、実用化に向けた活動を進めている。

尚、本保護 OS は 2008 年 3 月にオープンソースとして公開する予定である。

アピール指標：新規性、実用性、品質

オープンソース保護 OS：メモリ保護と時間保護を有する自動車向けリアルタイム OS

（ 第 10 回 IP アワード応募書類 特徴の説明書 ）

服部博行† 大西秀一† 片岡 歩† 松原 豊‡ 高田広章‡

†株式会社ヴィッツ ‡名古屋大学情報学研究科

E-mail: †{hat,ohnishi,kata}@witz-inc.co.jp, ‡{yutaka,hiro}@ertl.jp

1. 保護 OS とシステム LSI

自動車に求められる基本性能，環境性能，安全性能などは年々高度化し，この要求機能の実現に ECU（ Electronic Control Unit ）を用いた電子制御化が急速に進んでいる．その結果，現在の最高級乗用車の ECU 搭載個数は 100 個を超え，自動車制御システム開発コストの 8 割は制御ソフトウェア開発に費やされている．

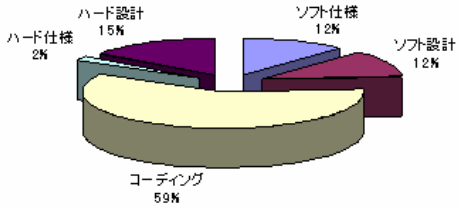

Fig -1 ソフトウェア開発コスト比率　引用[6]

この自動車制御の電子化により ECU の搭載限界 [1]，ワイヤーハーネス重量の増加 [2]，ソフトウェア規模の爆発的な増加などの問題が顕著化している．

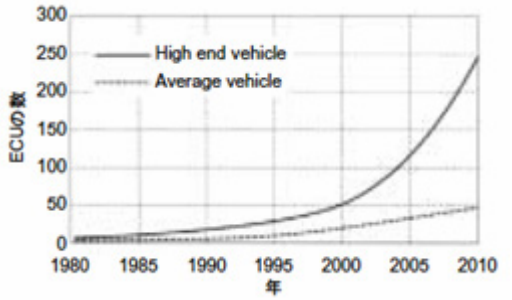

Fig-2　ECU 搭載数の増加　引用[7]

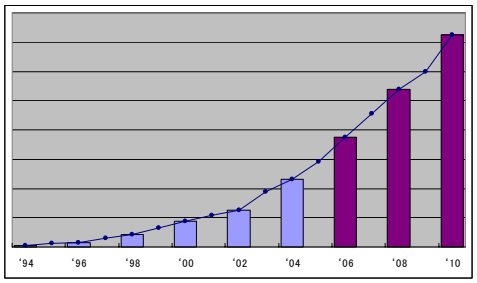

Fig-3　ソフトウェアの増加量　引用[6]

これらの問題を解決もしくは軽減させる方法として自動車メーカ各社 [3]が検討しているのは，標準的なソフトウェアプラットフォーム （ SFP ；Software Plat Form ）の構築と標準SPF上でのECU統合である．すなわち，各ECU供給企業が同一のSPF上で稼動するアプリケーションを開発することにより，自動車メーカはソフトウェアの部品化とECU統合の促進を狙っている．そのため，SPFの構築と標準化は今後の自動車開発のキーテクノロジになる．

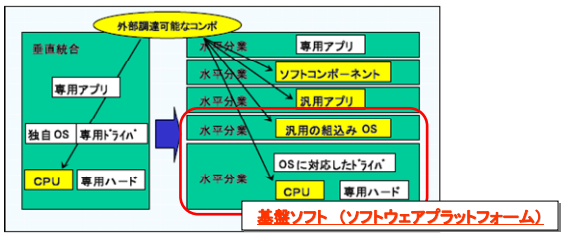

Fig-4 垂直統合から水平分用への移行と SPF　引用[6]

一方，標準的な SPF が構築されたとしても，異なる企業から供給されたソフトウェアを同一 ECU 上で稼動させる，いわゆる ECU 統合は，その動作検証において，相互干渉等が発生しない，リソース競合が発生しないなどの検証を入念に行う必要があり，標準 SPF が構築されても検証にかかるコストは削減できないという問題が残る．この問題を解決もしくは軽減するための技術として，保護機能は有力である．すなわち，SPF に保護機能を導入することにより，アプリケーション間の相互干渉およびリソース干渉などを未然に食い止めることができ，不良ソフトウェアによる被害を最小限に押させることができる．このように，保護機能は，次世代の自動車制御ソフトウェア開発において必要不可欠な機能であり，保護機能を有する保護 OS もしくは保護 OS を含む SPF は不可欠な IP（ ソフトウェア設計資産 ）の一つであると言える．

2. 保護 OS の概要

保護 OS は，自動車制御で事実上業界標準仕様である OSEK/VDX OS 仕様[2] に準拠した TOPPERS/OSEK カーネル上に構築し，自動車アプリケーションの ECU 統合に必要不可欠なメモリ保護機能と時間保護機能を有するリアルタイム OS である．この保護 OS は，ECU 統合を安全且つ容易に実現することを目的とし，アプリケーション間およびアプリケーションと OS 間などのメモリ，メモリ上の各種 IO，プロセッサ時間などの干渉を排除する特徴を持っている．

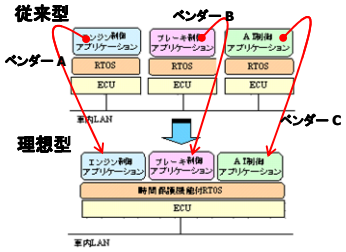

Fig-5 理想統合の図

---

1 通常，ECU は動作環境の問題から助手席の足元など限られた場所に搭載されるため，ECU 搭載空間には限界がある．

2 通常，ECU には各種センサやアクチュエータが接続されて機械制御を行っている．また，ECU 間で相互に情報を交換してより細密な制御を実現している．これらの機器間を接続するのがワイヤーハーネスであり，現在の車には 100kg 以上のワイヤーハーネスが利用されている．

3 国内の自動車メーカ各社は，各社で共通で利用可能な SPF を検討するために，国内の標準化団体 JasPar (Japan Automotive Software Platform Architecture ; http://www.jaspar.jp)を設立して検討している．また，欧州を中心として国際的な標準化団体 AUTOSAR (Automotive Open System Architecture ; http://www.autosar.org/) においても，標準 SPF の検討は進んでいる

この保護 OS を導入することにより，アプリケーションでの不良動作が他のアプリケーションおよび OS に伝播しない．すなわち，同一 ECU 上に配置されたアプリケーションであっても，物理的に異なる ECU 上で稼動していると同等の独立性を保つことができる．このことは，アプリケーション A の不具合動作により，アプリケーション B において問題が発覚するなど，従来の ECU 統合が抱えていた問題を解決し，かつ，従来の ECU 統合が抱えるプロセッサ利用時間の保証やスケジューリング問題等を解決している．

ターゲット環境として，現時点では，ルネサステクノロジ社の M32R-Ⅱを利用している．この理由として，メモリ保護機能は，汎用 OS のメモリ保護が利用している MMU（ Memory Management Unit ）に類似した MPU（ Memory Protection Unit ）を利用する必要がある．しかし，MMU および MPU は既存の組込み向けマイクロプロセッサは有していない装置であるため，MPU の仕様から保護 OS に必要な要求事項を検討し，ルネサステクノロジ社に MPU 要求仕様を開示し，開発していただいたため，現在のところ M32R-Ⅱの FPGA 版でのみ稼動可能である．

尚、本保護 OS は研究目的で開発したばかりでなく，実用としての検討も進んでおり，2007 年 10 月に実車を用いた実証実験を行い，メモリ保護および時間保護機能の機能的な有効性を確認し，実用検討も進んでいる。

この保護OSは，TOPPERSプロジェクト[4]から 2008 年 3 月末よりTOPPERSプロジェクト会員向けに早期リリースし，最終的にオープンソースソフトウェアとして公開する予定である．研究用途や組込み利用を想定して，これらのソフトウェア部品は，フリーソフトウェアとしても特に柔軟なTOPPERSライセンス[5]による配布とし，製品への利用，販売等に制限を与えない

3. 保護 OS の機能および特徴

本保護OSは大きく 2 種類の保護機能から構成される．一つはメモリ保護機能であり，二つ目は時間保護機能である．さらに，時間保護機能はHIS[6]が提案する，デッドラインモニタリング機能[1]と提案者らが考案した階層型スケジューラの二種類を開発した．

3.1 メモリ保護の機能および特徴

メモリ保護機能は OSEK 仕様 OS にメモリ保護処理部を追加して実現している．また，メモリ保護処理部は，マイコン内に独自に追加した MPU を利用し，メモリ保護を実現している．すなわち，メモリ保護機能はソフトウェアのみで実現しているのではなく，ハードウェアのサポートを得て実現している．これは，組込み装置に使われる OS はリアルタイム特性が必要であり，メモリ保護をソフトウェアのみで実現するとリアルタイム特性を損なうためである．また，独自の MPU を開発しているのは，汎用 OS が利用する MMU を利用していては，ハードリアルタイム特性を維持できないと判断したためである．なぜなら，MMU は一種のキャッシュメモリを利用している関係上，ハードリアルタイム特性を維持できないためである．そのような理由から，本メモリ保護は組込みシステムの特徴を利用した静的アドレスマッピング方式の MPU を開発および利用して実現している．

また，メモリ保護の概念は，大規模 OS や汎用 OS などでは一般的に利用されているが，組込みシステムでは利用されていない．なぜなら，リアルタイム特性などが維持できないおよび低機能マイコンでの実現が難しいなどの理由がある．一方，現在の自動車システムにおいてメモリ保護は利用されていないが，ECU 統合などの技術が進むと必要不可欠となることが予想され，事実 HIS においてメモリ保護仕様[1]は検討されている．本メモリ保護は，HIS 仕様を参考にしたものの，HIS 仕様では疎結合マルチプロセッサシステムの ECU 統合には不十分と考え，HIS 仕様を拡張したメモリ保護仕様を策定している．尚，自動車システムの標準化団体 AUTOSAR も HIS の仕様[1]を拡張したメモリ保護仕様を 2007 年に公開したが，本仕様と類似した仕様となっており，本仕様のアプローチが正しいことを述べておく．

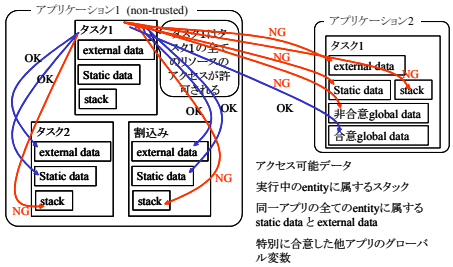

Fig-6 HIS のメモリ保護（ 保護は書込みのみ)

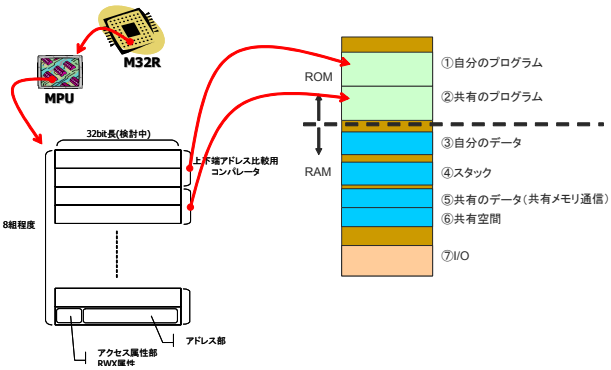

Fig-7 本メモリ保護の構成図

本メモリ保護を利用した場合の優位点を以下に述べる．

・ ハードリアルタイムシステムへの利用

自動車制御システムなどのハードリアルタイムシステムへのメモリ保護の利用は，その保護機能を含む全ての OS 処理時間やサービス提供時間などが予測可能でなければ利用することができない．すなわち，ハードリアルタイムシステムでの利用は，サービス等の提供時間が予測可能であることが重要となる．

一方，汎用 OS 等で実現されているメモリ保護は，比較的高機能プロセッサには内包されている MMU を利用して，プログラム動作時にアクセスするメモリ領域のアクセス可否を判定している．この MMU はアプリケーション等の動的再配置が前提とされているため，仮想アドレス空間から実アドレス空間に変換するためのテーブルを利用いている．このテーブルは TLB(Translation Look aside Buffer) と呼ばれ，キャッシュメモリと同等の処理により実現されている．すなわち，変換対象となるメモリ領域情報がこのテーブル上に存在する場合は，高速なアドレス変換が可能となるが，テーブルに存在しない，すなわち，ミスヒットの場合は，テーブル情報の再ロードが必要となり，アドレス変換時間が予測できない．加えて，組込みシステムの特徴として，処理単位は比較的小さく，その処理単位が短時間に切替ることもあり，当該情報がテーブル上に存在する確立も低くなる．このような理由により，汎用 OS と同等の手法では，ハードリアルタイム特性が必要なシステムへのメモ

---

4 http://www.toppers.jp/

5 http://www.toppers.jp/license.html

6 Herstellerinitiative Software　http://www.automotive-his.de/

リ保護が利用できないため，普及も遅れている．

本メモリ保護は，組込みシステムの特徴を利用することにより，アドレス変換を行わないメモリ保護を実現する．すなわち，組込みシステム，特に，ハードリアルタイム特性を必要とするシステムでは，動的再配置は利用せず，リンケージにアドレスが固定される静的配置方式を利用している．そのため，仮想アドレスは利用せず，実アドレスのみで処理が可能となる点を利用し，実アドレスを利用したメモリ保護ユニットを開発することにより，変換テーブルを利用しない，すなわち，MMU を利用しない，処理時間予測可能なメモリ保護が実現できる．

そのため，提案者らはハードリアルタイムの実現に必要な MPU の要求をまとめ，メモリ保護機能の開発を実現した．このように，保護 OS を開発する上で，実製品への応用を意識した性能を得るために，ハードウェアの仕様まで策定している点は，新規性，有用性が高いといえる．また，本 MPU を搭載したマイコンは 2008 年初旬にワンチップマイコンとしてサンプル出荷されるとの報告をうけており，ハードウェア面からも新規性，有用性を認められている．

・ メモリの保護属性

本メモリ保護は，保護属性として書込，読込，実行の 3 属性を区別して保護を行う．すなわち，当該メモリ空間の属性が上記 3 属性を許している設定であれば，実行プログラムからのアクセスは全て許可される．仮に，上記メモリ空間の属性が，読込，実行を許している設定の場合，プログラムは上記メモリ空間のプログラムを実行することと内容の参照は許されるが，書込みは許されない．このようにメモリ空間毎にアクセス属性が設定できるために，各空間の利用方法に沿った設定が可能となる．

一方，本保護 OS は各種 IO などのリソースを個別に保護してはいない．しかし，組込み向けプロセッサの多くは，各種 IO リソースはメモリ空間に割り付けていることが多いため，このメモリ保護機能を利用して，アプリケーション単位での IO 等のアクセス制限をかけることが可能となる．

また，現在の自動車制御システムは CAN 等のネットワークを介した疎結合マルチプロセッサシステムであると言える．このシステムを ECU 統合する場合，CAN 通信などの受信を特定メモリ領域に容易に置き換えることが可能である．すなわち,データ生成元は特定メモリ領域に書込権限を持ち，他の複数アプリケーションは読込権限とすることにより，同時に複数のアプリケーションが同時にデータ受信したと同等の効果を得ることが可能となる．

・ 複数のメモリ領域を保護対象として管理可能

本メモリ保護は複数のメモリ領域を保護対象として登録することが可能であり，ルネサステクノロジ社製プロセッサは 8 組のメモリ領域を保護領域として登録することが可能である．また，この領域はメモリの重なりを許可している．複数の領域を管理できることにより，それぞれの領域に意味を持たせることが可能となるため，制御アプリ毎の管理ポリシーを任意に決めることが可能となる．一般的な管理領域としては，自己プログラム領域，共有プログラム領域，自己データ領域，自己スタック領域，共有データ（ 共有メモリ通信空間 ），共有データ領域，I/O 領域などを想定している．保護領域として登録しない領域は全て他のアプリケーション領域と判断し，アクセス不可として扱う．この方法により，自己所有領域空間の認識と，自己アプリケーションがアクセスできる領域を適格に把握することが可能となる．

また，領域指定は，メモリ領域の上下端アドレスを設定することにより指定する．この設定は，アプリケーションが切替るタイミングで，アプリケーションごとの登録内容に従って，MPU レジスタに登録する．MPU は登録されたアドレス情報を用いて，メモリアクセス違反の有無をチェックする．

・ 保護粒度の柔軟性

本保護 OS は，保護対象のメモリ領域サイズを任意に変更できるばかりでなく，アプリケーション粒度も任意に指定することができる．一方，メモリは通常アプリケーションが所有していると言える．このアプリケーションは，タスクやリソースなどから構成されるものであり，その規模はアプリケーションにより異なる．また，実行単位毎に個別の保護領域を設定したい場合やライブラリなど複数のアプリケーションで共有したい場合が考えられる．これらの場合を考慮し，アプリケーションの単位は，たとえタスク 1 つであっても，アプリケーションとして登録することが可能であり，柔軟な管理単位で保護制御が可能となる．なお，アプリケーション登録は OIL（ OSEK Implementation Language ）と呼ばれる定義言語で簡単に記述できるように仕様を作成した．

・ OS 処理のオーバヘッドは少ない

本メモリ保護機能を実現するに当たり，OS 内部でメモリ保護専用の実行コードは少ない．メモリアクセス時に自アプリケーションがアクセス可能な領域か否かの判定は，ハードウェア（MPU）で判断する．すなわち，ソフトウェアによる判定処理を必要としないため，判定処理部位を OS 内部に実装する必要はない．また，保護属性に従ったアクセス判定もハードウェアにて，現在どのような種別のアクセスが発生したかを認識し，各領域に登録されている属性情報に基づいて種別判定も同時に行なっている．

このようにメモリアクセス時の判定処理はハードウェアにて実行しているために，判定オーバヘッドは無い。唯一のオーバヘッドとして，アプリケーション毎に異なるメモリ保護領域情報を MPU レジスタに設定する処理である．

尚，各領域の属性情報はアドレス情報の上位ビットを用いて設定するために，属性情報設定用のオーバヘッドは発生しない．

・ 特定機能部位の隔離

メモリ保護機能を利用することにより，各アプリケーション間や OS との間に壁を設けることができるため，アプリケーションの動作が他のアプリケーションに影響を及ぼすことはない．そのため，アプリケーションは分離されていると言える．一方，この性質はアプリケーション内の特定部位を分離することも可能となる．近年のアプリケーションはアプリケーション内部の処理単位で，その役割や求められる信頼性は異なってきている（ 例えば，ブレーキ処理においても，純粋にブレーキ機能を担う部位と他の ECU との通信や付加機能を担う部位では厳密な意味で信頼度は異なる ）．すなわち，本メモリ保護を利用することにより，同一アプリケーションであっても，信頼度による機能隔離が可能となる．

加えて，メモリ保護機能では，プロセッサの特権モードを利用する必要がある（ 特権モードを持ったプロセッサが必要 ）．異常発生時はこの特権モードを利用して，本当に必要な処理をアプリケーションと切り離して実行することも可能であり，従来の OS 機能と比較して高い信頼性を有しているといえる．

3.2 時間保護の機能および特徴

本時間保護機能は種類の異なる 2 タイプの時間保護もしくは監視を実装している．現在の自動車システムを構築する場合，要求さ

れたタイミングからある種の計測もしくは演算を経て，その結果を元に機器を動作させるなどの処理は比較的多く，かつ，その時間的制約と処理の終了性は動作決定において重要である．このような制御において，要求タイミングから動作指令までの間に，ある種の処理実行が完了しているかどうかを判断する手段は重要である．この観点を重視し，HIS は Time Protection 機能として，デッドラインモニタリングの機能[1]を規定し，仕様化している．本時間保護機能は，このデッドラインモニタリング機能を一つ目の時間保護機能として導入している．一方，上記デッドラインモニタリングは，単に監視を実行し，規定時間内に処理の完了が否かを判断しているのみであり，これは純粋に監視であり保護とは呼べない．なぜなら，処理完了しない原因を特定できない，もしくは，他への影響するのを防いではいないためである．すなわち，他のアプリケーションの不具合で，たまたま優先度の低い処理がデッドラインオーバーをしても，問題となる部位を特定できない．本保護 OS は，デッドライン監視の有効性を認めているものの，保護機能としては不十分であるため，現世代システムへの適応をデッドラインモニタリングで対応し，次世代の ECU 統合の基本機能として階層型スケジューラを用いた新しい考え方の時間保護を提案する．

3.2.1 デッドラインモニタリングの機能および特徴

デッドラインモニタリング機能は，HIS にて機能策定された時間監視機能である．将来的には真の意味での時間保護が必要となるものの，現時点では，必要となる処理が必要となる時間までに完了していることが，制御品質の向上には重要である．

このデッドラインモニタリングは，機能要求された処理単位が，処理単位が処理を完了しなくてはならない時間までに処理を完了することを監視する．すなわち，処理単位の起動・終了時にタイマの操作を行いデッドラインを判定するものである．

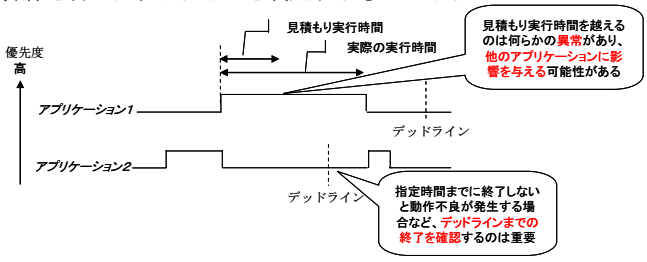

Fig-8 デッドラインモニタリング説明図

このデッドラインモニタリングには以下の特徴を有する．

・ **シンプルな監視でオーバヘッドが少ない**

デッドラインモニタリング機能を実現するに当たり，OS 内部に新たに追加するべき処理部位は非常に小さく，また，時間計測用には OS 機能が保有するアラーム機能を利用するため，新たなタイマなどのリソースを必要としない．現状の OSEK OS に親和性の高い仕様である．

具体的には，OS が処理単位（この場合はタスクを示す）を実行したタイミングで，OS の既存機能であるアラームにデッドライン時間を指定して稼動させる．その後，処理単位が終了した時点で OS は当該アラームを停止する．アラーム停止処理前に，当該アラームが満了した場合，当該処理単位はデッドラインオーバーしたと判断することができる．その結果，OS 内部での追加処理は，処理単位起動処理にアラーム設定及び動作指令の追加と処理単位終了処理にアラーム機能の停止処理を追加することが主な追加処理であり，この処理に加えて，デッドラインミス発生時するアラーム処理にデッドラインミス用のハンドラ部位を追加するのみで実現することができる．また，HIS の仕様では，ミスを発生した処理単位を含むアプリケーションの停止および再起動の手順について規定されているため，現状の ECU 開発に必要な各種タスクおよびアプリケーションの停止および起動に必要な API は用意した．ただし，これらの処理はデッドラインモニタ専用ではなく，保護 OS として必要な機能として実現している．

・ **現車両開発に利用可能な標準的仕様**

本デッドラインモニタリングは，必要な処理が必要な時間内に完了していることを前提としている現在の ECU 開発において重要な監視機能である．現状の開発では，必要時間内での完了は，各種テスト，処理時間の計測及び予測などから導いている．しかし，実稼動時全ての検証は事実上不可能であり，ECU 開発において、万が一必要時間内で処理が終了しない場合でも動作に影響の無い様設計されている．これらのフェールセーフ処理は，ソフトウェアの複雑度を増し，複雑化する機能との相乗効果でソフトウェアの爆発的な増加を促進さえている要員でもある．本デッドラインモニタを利用することにより，デッドラインミスが判定できるため，フェールセーフ処理の削減や，ミス発生時の対処が的確に行うことができ，より細密な制御をすることが可能である．

また，本デッドラインモニタリングは，HIS[1]にて規定された仕様であるため，世界標準として利用しやすい仕様である．

3.2.2 階層型スケジューラの機能および特徴

階層型スケジューラは，今後の EUC 開発に必要不可欠となる ECU 統合を安全且つ容易に実現するための基本となる機能の一つである．

OS 内部に処理の実行順序などを決定するスケジューラ機能を階層型に組み合わせたものを階層型スケジューラと呼び，本時間保護では 2 段の階層型スケジューラを採用している．各スケジューラは，アプリケーションスケジューラ(メインスケジューラ)とタスクスケジューラ(サブスケジューラ)と呼ばれる．タスクスケジューラは各アプリケーション内のスケジュール管理を行う．このアプリケーションスケジューラのスケジュールポリシーは各アプリケーションの設計方針に依存しても良く，階層型スケジューラでは特に限定はしない．ここでは組込みで一般的に利用されているプリエンプト可能な優先度ベーススケジューラにて説明する．タスクスケジューラはアプリケーション内で次に実行するべき処理単位を特定し，その処理単位の優先度，デッドライン時間などアプリケーションスケジューラが必要な情報と共に，アプリケーションスケジューラに起動の要求をする．アプリケーションスケジューラは複数のアプリケーションから要求される処理単位と処理単位と共に通達される情報を元に，次に動作するべきアプリケーション（処理単位）を決定する．アプリケーションスケジューラのスケジューリングポリシーは，EDF（Earliest Deadline First）方式で，最も早くデッドラインを迎える処理を優先して実行する．また，特定アプリケーションばかりが実行され，他のアプリケーションに影響を与えないように，バジェットと呼ぶ単位時間当たりに各アプリケーションが実行してもよい時間を管理し，特定アプリケーションが動作しすぎないように管理する．

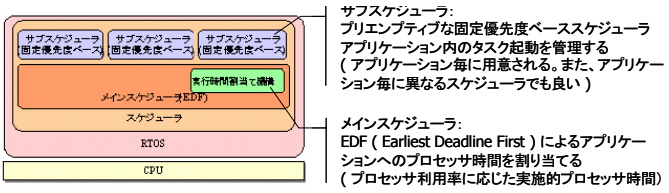

Fig-9 階層型スケジューラ構成図

この階層型スケジューラには以下の特徴を有する.

・ バジェットの管理で ECU 統合時の時間問題を解決

ECU 統合を行う上で，複数の異なる企業で開発されたアプリケーションが同一アプリ上で時間的に動作できるかどうかを判断するのは難しい．例を挙げて説明すると，10MHz（アプリ A），20MHz (アプリ B), 30MHz (アプリ C) のプロセッサで時間的制約を守って動作している各アプリケーションを 60MHz で動作するプロセッサに統合して正しく動作するかの判断は難しい. なぜなら, 各アプリケーションの最大負荷が同時に発生する場合など負荷状況の見積りが困難となるからである. 本階層型スケジューラでは, バジェットを管理することにより，同一プロセッサにおいても，個別プロセッサで処理されると同じ時間的環境を作り出すことにより，アプリケーション間の時間的衝突を回避し，動作検証不要を目的としている.

具体的には，各アプリケーションの動作割合（ 上記例では，プロセッサ能力比率より, アプリケーション A は 16% ， B は 33% ， C は 50%の占有比率が認められる ）から単位時間当たりの動作可能時間（ バジェット ）を定め，そのバジェット時間分だけの動作を許可する管理を行う．その結果，各アプリケーションは自己がプロセッサを占有できる時間を越えて，動作を許可されないために，他のアプリケーションが動作するべき時間を消費しない. すなわち, 同一プロセッサ上で稼動しながら, 各アプリケーションのプロセッサ占有時間は完全に分離されている．この特徴は，各アプリケーションが低速度プロセッサで確実に動作できるのであれば, そのプロセッサ能力に相当する時間を割り当てれば，確実に動作し，かつ，他に影響を与えないために，ECU 統合時のプロセッサ時間に関する検証は不要となる．すなわち，真の意味での時間保護が可能となることを意味する.

・ ECU 統合後のリアルタイム応答性確保

汎用 OS は時間保護機能を有していない．しかし，複数のアプリケーションを並列実行する機能（ マルチタスク ）は有している．多くの OS では，この機能を実現するにあたり，タイムスライスという概念で実現している．すなわち，同時実行中のアプリケーションの処理を少しずつ，時分割して実行している．この制御の場合，緊急処理が必要なアプリケーションがあった場合であっても, アプリケーションの動作タイミングまでその処理が遅延する問題があるため，リアルタイム制御が可能な OS とは呼ぶことができない.

一方リアルタイム OS は, プリエンプト可能かつ優先度ベースのスケジューリングを行うことにより, リアルタイム特性を獲得している. ECU 統合を行う場合の OS は, アプリケーションを超えて，このような即時応答性を備えた性能が必要となる. 階層型スケジューラは, 各アプリケーション内で即時応答が必要な処理単位をアプリケーションスケジューラに通達することにより，例え，他のアプリケーションが実行中の処理があったとしても, 即時応答が必要なアプリケーションの処理単位を実行する機能を有している. すなわち，ECU 統合をした後であっても，全アプリケーションを対象とした即時応答処理が可能である特徴を有する性能を持っている.

4. 動作環境と開発環境

本保護 OS は, 1 章で述べた通り, ルネサステクノロジ製 M32R-Ⅱに MPU を追加した特殊プロセッサをターゲットプロセッサとしている．しかし，2008 年初旬には，本 MPU を搭載したプロセッサのサンプル出荷が開始されるため, 今後は実プロセッサの利用が可能となる．尚，MPU を必要とするのはメモリ保護機能であるため, 時間保護機能は一般的なプロセッサであれば動作可能である.

開発環境は，ルネサステクノロジ製コンパイラ（ NC308 Version 5.20 Release 02 ）および gcc コンパイラを用いて開発し，同社製デバッガ(KD3083 Version 3.30 Release 1)，同社製簡易エミュレータ（ M3A-0655 FoUSB ）を用いてデバッグを実施した.

5. 開発成果について

応募代表者は，保護 OS の全機能について，アイシン精機株式会社および株式会社東海理化の協力を得て, 実車による実証実験を実施し，保護 OS に必要な機能面の評価を行っている.

また，この実証実験の結果を基にして，製品適応についても，いくつかの会社が検討を開始しており, 国内メーカからの問い合わせ及び評価を多数いただいている.

6. 今後の計画

本保護 OS は，経済産業省の平成 17・18 年度地域新生コンソーシアム研究開発事業（ 中部地区 ）:「自動車統合制御用組込み OS の開発」にて研究開発したものであり，2008 年 3 月末を目標にオープンソースとしても公開を計画している．また，本保護 OS の仕様は，JasPar に提案し，将来的には，AUTOSAR のメンバ社を通じて AUTOSAR でも検討していただけるように提案したいと考えている.

7. 謝辞

本保護 OS を開発するにあたり, トヨタ自動車統合システム開発部の細谷様をはじめ，アイシン精機，東海理化の方々にご意見およびご協力をいただきまいた．ここに記して謝意を表します.

文　献

[1] HIS，OSEK OS Extensions for Protected Applications，HIS，July 24th 2003

[2] OSEK/VDX, OSEK/VDX Operating System Version 2.2.1 , OSEK VDX，January 16th 2003

[3] 松原豊，本田晋也，冨山宏之，高田広章，"時間保護のためのリアルタイムスケジューリングアルゴリズム", 情報処理学会論文誌コンピューティングシステム，Vol.48,No.SIG 8(ACS 18), pp. 192-202, May 2007

[4] 松原豊，本田晋也，冨山宏之，高田広章，"OSEK アプリケーション統合のための柔軟なスケジューリングフレームワーク", 組込みシステムシンポジウム 2007，Vol.2007,No.8，pp. 33-41, Oct 2007.

[5] 松原豊，冨山宏之，高田広章，階層型スケジューラによる自動車制御システム向け時間保護環境，組込みソフトウェアシンポジウム 2005 論文集, pp. 110--116，東京都, Sep 2005

[6] 谷川浩，自動車制御ソフト・車載 LAN 標準化活動，Automotive Technology Days 2004 Autumn，自動車制御ソフト開発の効率化予稿集，pp. 61-74，Nov 1st 2004

[7] 大倉勝徳，デンソーテクニカルレビュー Vol.10 No.2，2005 ，P11