愛知工業大学研究報告
第40号B平成17年

メモリ保護機能を実装したＲＴＯＳに関する提案

Real-time Operating System with Memory Protection Function of Embedded Software

八木　隆行† 加藤　厚生††
Takayuki YAGI† Atsuo KATO††

Abstract: Recently, as the software of embedded systems is getting larger and more complex, keeping the quality and the reliability of the software high becomes very difficult. One of the approaches to solving this problem is to introduce protection functions to real-time operating system used for embedded systems. We added the access protection function of the kernel object to μITRON4.0. Protection function of embedded system is enabled by managing object in the group. This proposition outlined the protection extension of embedded software and describes the result.

1．　はじめに

1・1　研究の背景

近年、マイクロプロセッサ技術の発展により機器組み込みシステムの応用分野は拡大の一途をたどっている。機器組み込みシステムは工場の生産ライン制御をはじめとして産業用途中心に利用されてきた。しかし近年では、携帯電話をはじめとする身の回りの電子機器全てといってよいほど機器組み込みシステムが利用されている。

携帯電話、デジタル家電等の情報機器は日本において普及率が急成長している。これらの情報機器はネットワーク機能を搭載し、相互に情報のやり取りを可能になるなど高性能化は著しく、市場競争も非常に激しいものとなっている。

一般にこのような民生用機器の小規模組み込みシステムは、産業用途の組み込みシステムに代表される規模の大きい組み込みシステムと比べ、生産個数が多く、単価が安いという特徴がある。そこで、必然的に開発コストを下げることよりも、最終的な製品のトータルなコストを下げることが重視される。これまでは、最終コストを下げるためにハードウェア資源を制約しているのが現状であったため、組み込みシステムの開発においては低級言語での開発が中心でOSの使用も避けられていた。しかし、今ではハードウェア資源の制約よりも開発期間の短縮・高性能化を求めることが重要視されている。そのため組み込みシステムにおいて高級言語を使用するケースやリアルタイムOSなどを利用するケースが飛躍的に多くなってきている。

ソフトウェアの複雑化や開発期間の短縮という相反する要求によって、ソフトウェアの品質や信頼性の確保をすることが難しくなっているのが現状である。

†　愛知工業大学　大学院工学研究科　（豊田市）
††　愛知工業大学　工学部　機械学科　（豊田市）

１・２　研究の目的

組み込みシステムにおいて、メモリ管理やメモリ保護機能をカーネルが提供する必要がでてきている。これは組み込みシステム開発プロジェクトの肥大化や、システム開発のライフサイクルが極めて短いことが要因としてあげられる。

組み込みシステムの機能は大きくなる一方であり、製品が小型化・軽量になっていくのと反比例している。そこでは本質的には信頼性のあるプログラムであっても油断はできなく、ＲＯＭに常駐するプログラムにエラーが残っていても不思議はない。このような要求から本研究ではオープンソースとして公開されている TOPPERS/JSP カーネルを使用し、メモリ保護機能を実装した組み込みシステムの開発をする。またソフトウェアの再利用がしやすいカーネルを作ることを目的とする [1]。

１・３　従来の研究

現在、トロン協会ではメモリ保護機能を搭載したＯＳとしてμITRON 仕様保護機能拡張として IIMP カーネル（μITRON4.0/PX 仕様）が公開されている [2)3)]。IIMP カーネルは、セキュリティー保護・信頼性向上・デバッグ支援のための保護機能に対応したカーネルである。この IIMP カーネルはメモリのマッピングや保護を目的としたハードウェア MMU を使いプログラムが許された領域外へアクセスできないようにしている。しかし民生機器の組み込みシステムに使われるプロセッサの多くは MMU 機能を持っていない。MMU 機能を持たないプロセッサにはこのモデルを適応させる事が不可能であり、実装可能なプロセッサを限定してしまう問題がある。

株式会社エルミックシステムよりオブジェクト保護機能を搭載したμITRON 仕様の Hyper ITRON が公開されている [4]。この Hyper ITRON はオブジェクト（タスクやイベントフラグなどのカーネルの管理対象）への不正なアクセスを防止するために、従来の ID 番号管理とは別に、グループ内で ID 番号をローカル管理することによってアクセスを制限している。しかし、現存のプログラムに保護機能を追加する場合、コンフィギュレータや ID 番号の割り付けをし直す必要がある。また、ID 番号をローカル管理しているので、この機能はサービスコール全体に影響してしまう。

２．　グループ管理による保護機能拡張

２・１　従来のタスク管理

タスクは ID 番号と優先度で管理されている。従来の ID 番号管理は、システムに一意に決められた番号を割り当て、システムに登録されるタスクは ID 番号を指定して自由にオブジェクトへの要求を行う事によって管理する。しかし、この場合 ID 番号の指定を誤ると本来目的とする処理とは違う動作をしてしまう可能性がある。オブジェクトの ID 番号は、通常１から始まる番号を割り付ける。オブジェクトが追加される場合、未使用の番号を割り付けるか、ID 番号を割り付けなおす。未使用の番号を割り付けていくと、オブジェクトの関連が番号から推測することが難しくなっていく [5)-7)]。

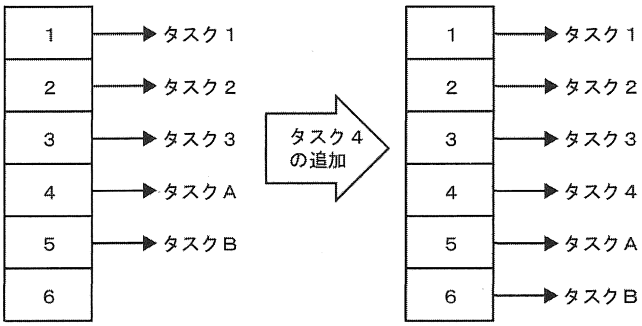

図１：システムにタスクを追加する例

２・２　グループ管理方法

本研究ではオブジェクトへの不正なアクセスを防止するため、オブジェクトをグループ管理することによって保護をする。オブジェクトの ID 番号をローカルに管理するのではなく、オブジェクトは従来のグローバル ID 宣言を使う。ローカル管理したいオブジェクトは、ローカル ID 番号として、ID 番号定義ヘッダファイルに定義する事によって現存のプログラムの書き換えを少なくする。ローカル管理されるオブジェクトは、ID 番号定義ヘッダファイルにユーザーが定義する。サービスコールによって ID 番号を参照する時に、ID 番号の違反を検知することによってオブジェクトの ID 番号の誤りを防ぐ。

２・３　グループ管理を導入した場合のタスク管理

ID 番号の誤りを防ぐために、従来と同じ ID 番号とは別に、ローカル管理されるオブジェクトには、グループのグローバル ID とタスク ID の２つの情報を持たせ、その ID 番号をローカル ID 番号とした。本ソフトウェアグループ管理では、サービスコールで ID 参照の際に、参照した ID 番号と呼び出された ID 番号を比較することによって ID 違反を判別する。オブジェクトへの不正なアクセスの場合、例外処理としてユーザー指定の処理をする。

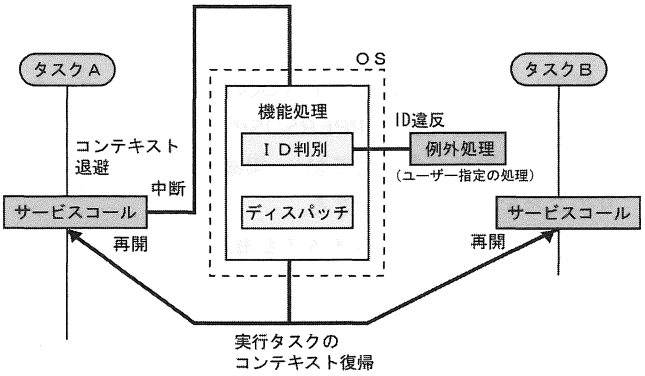

図２：オブジェクト保護の起動

２・４　拡張機能 ID 判別サービスコール

ID 判別サービスコールは、実行中のオブジェクトの ID 番号と、ローカル ID 番号を比べることによって、ID 違反の判別をする。グループ管理による保護機能を使用する場合、呼び出されるサービスコールで ID 判別を行う処理機能を付け加える必要がある。

ローカル管理される ID には、グループのグローバル ID とタスク ID の情報を持っている。このローカル ID 情報と実行中のオブジェクト ID を比較することにより ID 違反を調べる。ID 違反をしていた場合、例外処理としてユーザー指定の処理を行う。ID 違反ではない場合、本来の処理機能を実行する。

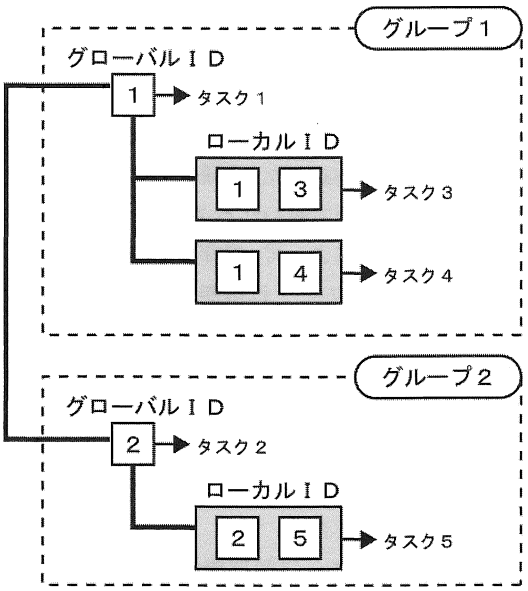

図 3:グループ ID 番号の管理例

タスク1のアクセス

アクセス可能	アクセス不可
タスク3、タスク4	タスク5
タスク2	

タスク2のアクセス

アクセス可能	アクセス不可
タスク5	タスク3、タスク4
タスク1	

表１：グループ管理のタスクのアクセス

ローカル ID 番号を割り付けたオブジェクトはそのグループに所属するタスクからのアクセスしかできなくすることによってアクセスの制限をする。

３．　従来との互換性

本研究のソフトウェアグループ管理では、従来のグローバル ID 番号をそのまま使って管理しているので、現存のプログラムに組み込む際にコンフィギュレータファイルの書き換えや ID 番号を割り当てなおす必要はない。また、グループ番号を意識するのは、ID 定義ファイルにオブジェクトを登録する時とサービスコールでオブジェクトを指定するときだけである。ID 違反の判別をサービスコールとして登録しているので取り外しが容易となる。

４．　グループ管理の効果

４・１　デバッグ効率の向上

オブジェクトがアクセスしてはいけない領域にアクセスした場合、カーネルが不正アクセスとして検出して例外処理ルーチンを起動する。例外処理ルーチンで不正処理を行ったプログラムの情報を取り出すことでプログラム開発者は迅速に、信頼性の高いプログラムを書くことができる。プログラムが大規模になるにつれて、不具合の発見には時間と労力が必要となるが、保護機能を有効に活用することでプログラムのテスト段階でバグとして検出でき、デバッグ効率が向上する。

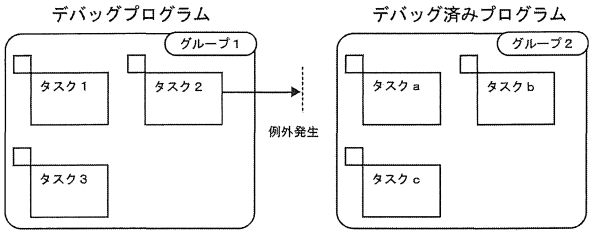

図４：デバッグ効率の向上

４・２　基幹システムの保護

ユーザーアプリケーションをグループとして登録することにより、基盤システムを構成するオブジェクトをユーザーアプリケーションから保護する事が可能となり、システムの信頼性を向上させることができる。

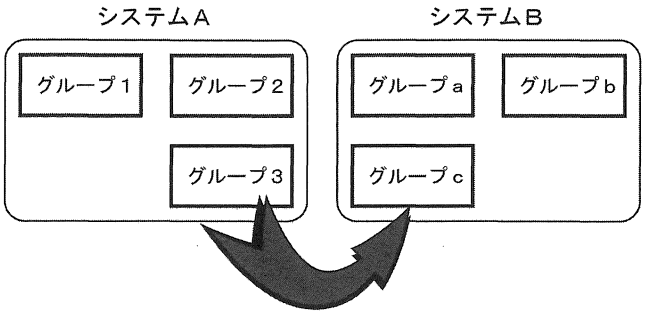

図５：基幹システムの保護

４・３　処理機能ごとの保護機能

サービスコールで保護関数を呼び出す事によって不正なアクセスを防止しているため、サービスコールそれぞれにグループ管理機能の使用をユーザー自身が決めることができる。たとえば、タスクの起動(act_tsk)には保護機能を持たせ不正なアクセスを防止し、タスクの強制終了(ter_tsk)では保護機能を持たせないことによって、すべてのタスクにアクセスする事が可能となる。サービスコールごとに保護機能の有無を決することによって、デバッグ効率の向上や、システムの例外処理などのユーザープログラムの自由度が増す。

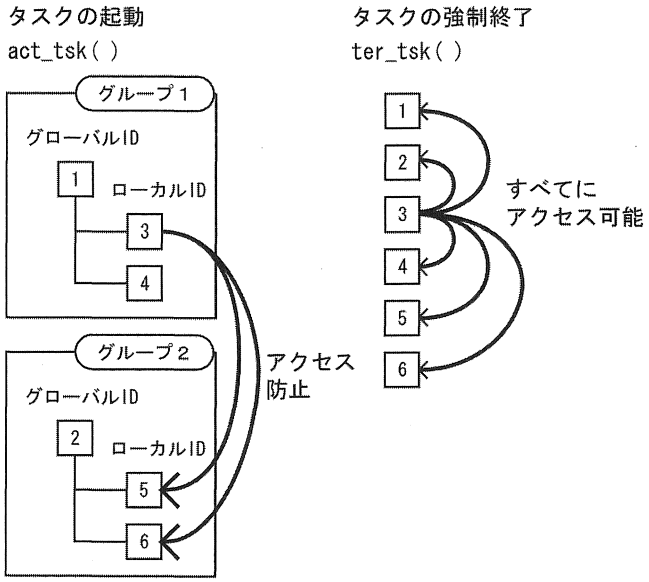

図６：サービスコールごとの保護機能例

５．　動作確認

拡張した保護機能の動作を TOOPPERS/JSP カーネルを使用し確認した。TOPPERS/JSP カーネルにはシミュレーション環境が含まれており、実機がなくてもひととおりのことを試すことができる。

三つの並列実行されるタスクを用意し、並列実行されるタスクは、タスクが実行中であることをあらわすメッセージを表示することによってタスクの状態を確認することが出来る。実験ではグループ登録されたタスクの起動(act_tsk)の ID 違反の判別をし、アクセス制限の動作を確認した。

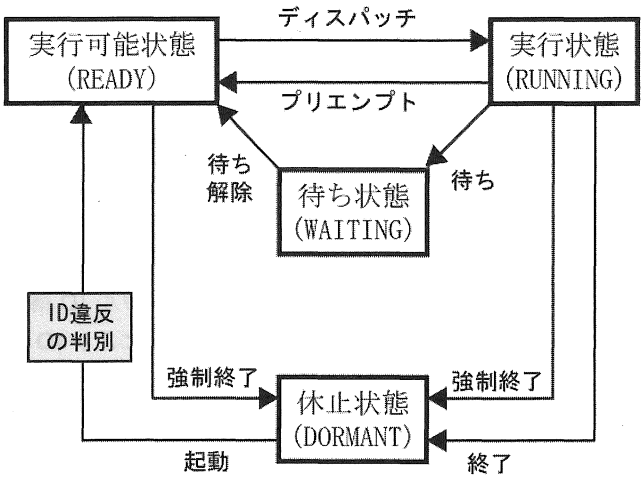

図７：ID 違反判別のアクセス制限

６．　考察

サンプルプログラムでは、タスク起動(act_tsk)の ID 違反によるアクセス制限を確認した。本研究のグループ管理機能は、従来のタスク管理で使われるグローバル ID 番号をそのまま使用することによって、現存のプログラムの書き換えをなくしている。このことは機能を利用するユーザー側からみると、コンフィギュレータや ID 番号の書き換えがないので簡単に現存のプログラムにオブジェクト保護機能を追加できる。また、新しいタスクを追加する場合、未使用の ID 番号を割り付けたとしても、ID 番号定義ヘッダファイルを見ることによってオブジェクトの関連が番号から推測することができる。

サービスコールごとにグループ管理機能を追加できるようにしたことは、ユーザープログラムの自由度を広げたといえる。すべてのオブジェクトにアクセスできるサービスコールは、例外処理などでローカルに管理されるタスクを強制終了させたい時などに利用できる。また、本研究のグループ管理機能は、タスク管理機能のサービスコール以外にも ID によって起動するオブジェクト（イベントフラグ

やメールボックス、セマフォ）にも同じ考えで保護することができる。

サービスコールごとにID違反の判別をすることは、すべてのサービスコールに保護機能を搭載した場合、オーバーヘッドが大きくなってしまうのが問題である。デバッグ支援のための保護機能として使用するならば、保護機能は簡単に取り外せるので問題はない。しかし、信頼性向上のための保護機能として使用する場合、オーバーヘッドが大きくなりリアルタイム性が失われる可能性がある。ハードリアルタイムなどの厳格な制限時間をもつシステムでは、オーバーヘッドを少なくし処理を早くする必要があるため、OSの使用者は注意が必要である。OSのリアルタイム性能のチェックは、OSの使用者の責任に任されているのが現状である。使用者は、組み込み製品のシステムに対する要求を十分に理解し、さらにどの要求を優先すべきかという要求の重みも十分に考えなければならない。

7． 結論

本研究ではオブジェクトのグループ管理による保護機能をJSPカーネルをベースとし拡張した。オブジェクトのグループ管理によって、研究目的のすべてを達成することができたといえる。

ＭＭＵを持たない低価格なプロセッサでの保護機能を追加したことによって、システム全体の信頼性をあげるとともに組み込みシステムのトータルなコストを下げることにも期待できる。またサービスコールによってオブジェクト保護機能を呼び出すことにより現存のプログラムの書き換えをすることなく、従来のITRONを使用したシステムへの導入が容易になったといえる。

参考文献

1)TOPPERSプロジェクト
http://www.toppers.jp
2) IIMPプロジェクト
http://www.assoc.tron.org/iimp/
3) 高田広章　μITRON4.0仕様 保護拡張機能(μITRON4.0/PX 仕様)Ver.1.00.00(社)トロン協会バージョンアップWG ,2002
4)株式会社エルミックシステム Hyper ITRON
http://www.elmic.co.jp/
5) 坂村　健／高田　広章　μITRON4.0仕様 Ver.4.02.00(社)トロン協会 ITRON仕様検討グループ ,1999 - 2004
)金田一勉　ITRONプログラミング入門 ,2004
7) Interface ,2003.9～,CQ出版社

(受理 平成17年3月17日)