社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2008－SLDM－134／2008－EMB－8（16）
2008 / 3 / 27

組込みコンポーネントシステム向け保護機構の開発

山田 晋平† 中本 幸一† 安積 卓也†† 高田 広章†† 大山 博司†††

† 兵庫県立大学大学院 応用情報科学研究科 〒650-0044 兵庫県神戸市中央区東川崎町 1-3-3
†† 名古屋大学大学院 情報科学研究科 〒464-8601 愛知県名古屋市千種区不老町
††† オークマ株式会社 愛知県丹羽郡大口町下小口 5-25-1

あらまし 近年，ネットワークや汎用コンピュータシステムとの通信，統合が必要な組込みシステムが増加したため，セキュリティを強化する機構が必要とされている．また，大規模化・複雑化する組込みシステム開発に対して，コンポーネントの使用による開発効率の向上が期待されている．このような背景から，本稿では組込みコンポーネントシステム用にアクセス制御機構とメモリ保護機構を開発する．アクセス制御機構は主に，アクセス制御を行うセキュリティコンポーネントと，アクセスの可否を判断するリファレンスモニタから成る．メモリ保護機構は，複数領域や入れ子構造の保護や，動的なメモリ保護パターンの変更が可能など柔軟な設計となっている．
キーワード コンポーネント，アクセス制御，メモリ保護

Development of Protection Mechanism for Embedded Component System

Shimpei YAMADA†, Yukikazu NAKAMOTO†, Takuya AZUMI††, Hiroaki TAKADA††, and Hiroshi OYAMA†††

† Graduate School of Applied Informatics, University of Hyogo
1-3-3 Higashikawasaki-cho, Chuo-ku, Kobe-shi, Hyogo, 650-0044 Japan
†† Graduate School of Information Science, Nagoya University
Furo-cho, Chikusa-ku, Nagoya-shi, Aichi, 464-8601 Japan
††† OKUMA Corporation
Shimooguchi ,Oguchi-cho, Niwa-gun, Aichi 480-0193 Japan

Abstract Recently, embedded systems connected to network have increased; therefore it is needed to strengthen security. In addition, software component-based development is expected improvement of embedded software in productivity. We present design of security enhancement for embedded component system, access control and memory protection. Firstly, Access control controls what a subject access to an object by the access rule. We design and implement the access control mechanism for embedded component system. The second one is memory protection mechanisms. We also design the generic memory protection mechanism including the nested memory protection and the dynamic protection pattern change.
Key words Component, Access control, Memory protection

1. はじめに

従来の組込みシステムは，クローズドな専用システムであったので，安全性の上で問題となる脅威にさらされる機会が少なかった．しかしながら，近年，車載システムのような人命を左右する組込みシステムにおいても，ネットワークや汎用コンピュータシステムとの通信，統合が必要となり，汎用コンピュータシステムの脅威が組込みシステム側にも大きく影響を及ぼすようになってきている．このような背景から，組込みシステムにもセキュリティを強化する機構を取り入れる必要性が出てきた．特にリアルタイム OS（RTOS）ではまだセキュリティの機構が確立されておらず，対応が求められている．

一方，組込みシステムでは，最近ソフトウェアコンポーネントによる開発が注目されている．コンポーネントベースのソフトウェア開発では，ソフトウェアを機能単位で部品に分解し，得られたソフトウェア部品を組合せることによって開発を進める．コンポーネントベースのソフトウェア開発には，既存のコンポーネントの再利用による開発期間の短縮，開発コストの削

減などの利点がある．

また，近年の組込みシステムではメモリ保護機構も重要になる．従来の組込みシステム，特に電化製品のような小さなシステムではメモリ保護機能は必要とされない．しかしながら近年の大規模な組込みシステムでは，大きく複雑なプログラムが他のプログラム領域を破壊するような信頼性をなくす結果となることや，CPU の高速化によってオーバーヘッドが相対的に小さくなることから，メモリ保護機構が要求されている．

このような背景から本稿では，組込みコンポーネントシステムのデータや機能へのセキュリティ強化を行う．コンポーネントを保護する手段として，コンポーネントをアクセス制御やメモリ保護が挙げられる [1]．アクセス制御とは，あるサブジェクトがあるオブジェクトにアクセスすることを，アクセスルールに従って制御する機能のことである．コンポーネントベースの組込みシステムにおいて，オブジェクトである資源はコンポーネントによって扱われる．そのため，資源を扱うコンポーネントへのアクセスを制御することで，資源そのものを保護できると考えられる．この考えを基に設計したアクセス制御機構は，実際にアクセス制御を行うセキュリティコンポーネントと，アクセスルールを持ちアクセスの可否を判断するリファレンスモニタ [2]，この 2 つをコンポーネントベースのアプリケーションに適切に付加するセキュリティエンハンサから成る．これらの実装と評価を行った．メモリ保護に関しては，入れ子構造や複数領域の保護，動的なメモリ保護パターンの変更が可能なメモリ保護機構の提案を行う．

2. TOPPERS 組込みコンポーネント仕様

組込みソフトウェアのオープンソースプロジェクトである TOPPERS プロジェクトでも，TOPPERS Embedded Component System (TECS) として，組込みシステムに適したコンポーネントシステムを検討している．TECS を用いることで，時間制約やメモリ制約を考慮した，コンポーネントベースの組込みシステム向けアプリケーションを開発することができる．TECS は以下を目的とする，組込みシステムに適したコンポーネントシステムである．

- 組込みソフトウェア部品の流通のための標準化
- 大規模化，複雑化する組込みアプリケーション開発への対応
- 組込みシステムの分散処理のためのフレームワーク

2.1 コンポーネントモデル

TECS のコンポーネントは図 1 に示すように，自身の機能を提供するためのインタフェースである受け口と，他のコンポーネントの機能を呼び出すためのインタフェースである呼び口を持つ．これらを静的に結合することで，アプリケーションを作成する．コンポーネントが持つ機能は C 言語の関数として実装され，あるコンポーネントが他のコンポーネントの機能を呼び出すとき，関数呼び出しが行われる．

TECS における開発の流れを図 2 に示す．初めにコンポーネントの定義とコンポーネント間の結合の定義，コンポーネントが提供する機能のコードが定義された実装ファイルを用意する．これらの定義を読み込み，コンポーネント間の結合に必要なテーブルなどを出力するのがインタフェースジェネレータである．インタフェースジェネレータから出力されたファイルを実装ファイルとともにコンパイルすることで，アプリケーションが生成される．

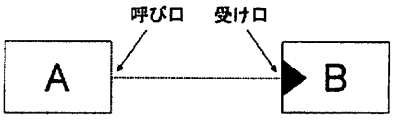

図 1 コンポーネント図

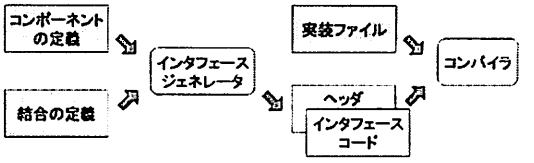

図 2 TECS における開発の流れ

TECS には，コンポーネント同士の結合部分にコンポーネントを付加し，フックをかける through という仕組みが存在する．through には既存のコンポーネントに影響を与えずにコンポーネントを付加できる特徴があり，分散環境における通信チャネルや，コンポーネント間の呼び出しのログ取りなどに用いることが考えられる．

3. アクセス制御機構

アクセス制御とは，アクセスの主体であるサブジェクトが，アクセスの対象であるオブジェクトにアクセスすることを，アクセスルールに従って制御する機能のことである．コンポーネントベースの組込みシステムにおいて，オブジェクトである資源はコンポーネントによって扱われる．そのため，資源を扱うコンポーネントへのアクセスを制御することで，資源そのものを保護できると考えられる．この考えを基にアクセス制御機構を設計した．図 3 に示す例を用いて説明する．

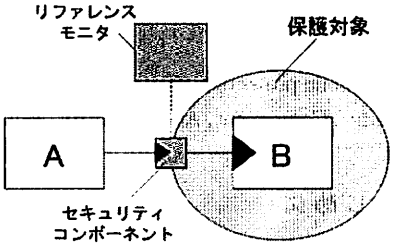

図 3 アクセス制御機構を用いた例

保護対象は，保護すべき資源を扱うコンポーネントのことである（図 3 ではコンポーネント B）．この保護対象にアクセスする部分に対して，アクセス制御機構を付加することでアクセス制御を行う．アクセス制御機構はセキュリティコンポーネン

トとリファレンスモニタから成る．セキュリティコンポーネントは保護対象外から保護対象に対するアクセスが発生したときに，実際にアクセス制御を行うものである．リファレンスモニタは，アクセス制御の際にアクセスの可否を判定するためのアクセスルールを持っており，アクセスの状況に応じて応答を返す．保護対象外から保護対象にアクセスする部分全てにアクセス制御機構を追加する作業は，セキュリティエンハンサが行う．

アクセス制御機構を追加することで，以下の効果が期待される．

- システムの検証がしやすくなる

従来はアプリケーション内にセキュリティの機構のコードが埋め込まれていたため，セキュリティ設計の客観的な評価が難しかった．セキュリティコンポーネントは元々存在するコンポーネントとは独立しているため，セキュリティコンポーネントの有無や，セキュリティコンポーネントの内容をレビューすることで容易に評価できる．

- 一括してアクセス制御機構を付加できるようになる

保護対象が決まれば，セキュリティエンハンサよってセキュリティコンポーネントが一括して付加されるため，アクセス制御の抜けがなくなる．

- コンポーネントを安全に再利用できるようになる

他のアプリケーションで使用されたコンポーネントは，必ずしも安全とは限らない．しかしながら，アクセス制御機構が存在することで，安心して他のコンポーネントを再利用することができる．

- コンポーネントベースのアプリケーションのテスト，デバッグに利用できる

コンポーネント間の呼び出しをチェックできるため，クローズドな組込みシステムにおいても，コンポーネントベースのアプリケーションのテスト，デバッグに用いることができる．

アクセス制御機構の動作の流れを図 4 に示す．ここではサブジェクト X と Y がコンポーネント A から保護対象のコンポーネント B の機能を呼び出して，資源であるファイルにアクセスすることを考える．B はファイルを読み込むための read と，ファイルに書き込むための write という 2 つの機能を持っていると仮定する．アクセス制御機構が付加されていない状態では，A から B に直接アクセスできるが，付加されている場合，A から B にアクセスする前に，まず，セキュリティコンポーネントでアクセス制御が行われる (1)．セキュリティコンポーネントは保護対象にアクセスしようとしているサブジェクトと，呼び出そうとしている機能を基にリファレンスモニタに問い合わせを行い (2)，リファレンスモニタはアクセスルールを基にセキュリティコンポーネントにアクセスの可否を返す (3)．アクセスが許可された場合のみ保護対象の機能が実行され (4)，却下された場合は呼び出し元に対してエラーを返す (5)．この例では X が read か write を，Y が read を呼び出す場合はアクセスでき，Y が write を呼び出す場合はアクセスできない．組込みシステムでのセキュリティポリシーは多様であると考えられる．その一例を [3] に示す．

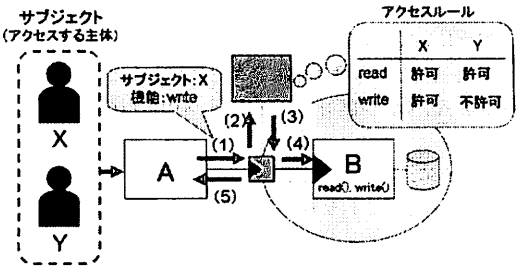

図 4　アクセス制御の動作の流れ

3.1 実　　装

コンポーネントベースのアプリケーションにアクセス制御機構が付加される流れを図 5 に示す．

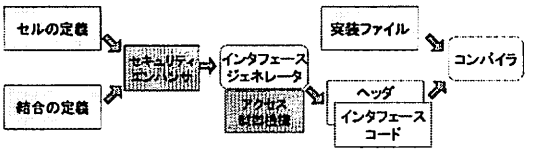

図 5　アクセス制御機構を付加する流れ

3.1.1 セキュリティエンハンサ

セキュリティエンハンサは保護対象のコンポーネントにアクセスする結合すべてにセキュリティコンポーネントを付加する機能である．セキュリティエンハンサはコンポーネントの定義，コンポーネント間の結合の定義，保護対象の定義を読み込み，セキュリティコンポーネントを付加する場所の情報を出力する．

3.1.2 セキュリティコンポーネント

セキュリティコンポーネントは，保護対象にアクセスする結合部分に付加され，アクセス制御を行うコンポーネントである．セキュリティコンポーネントの付加は，セキュリティエンハンサによって出力された付加する場所の情報を基に，インタフェースジェネレータの through 機能を用いて行われ，フックの処理としてアクセス制御が行われる．セキュリティコンポーネントがない場合のセル間での機能呼び出しは図 6(1) のようになるが，セキュリティコンポーネントが through によって付加された状態では図 6(2) のように，本来呼び出されるセル B の機能の代わりにセキュリティコンポーネントがもつ同名の機能が呼び出される．この機能の中で，現在アクセスしてきているサブジェクトや，アクセス先の機能名（現在実行している機能名と同じ）を基にアクセス制御を行う．

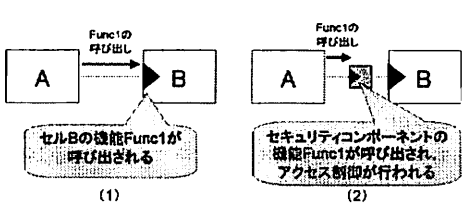

図 6　セキュリティコンポーネントの動き

3.1.3 リファレンスモニタ

リファレンスモニタは，セキュリティコンポーネントから渡される機能名と，保護対象にアクセスしようとしているサブジェクト名から，保護対象へのアクセスの可否を判定し，その結果をセキュリティコンポーネントに返す．アクセスルールは図7のように，ユーザが記述したルール記述ファイルからC言語のソースに変換されたものを用いる．ルール記述ファイルは，1つのルールを1行にサブジェクト名，機能名，アクセスの可否（accept または deny）の順にカンマ区切りで記述し，複数ルールを列挙したテキストファイルになっている．アクセスの可否情報は，全てのサブジェクトと，保護対象の持つ関数の組の数だけ存在するため，アクセスルールのサイズはサブジェクト数と保護対象の持つ関数の数に比例して増加する．

アクセスの可否の判定は，check 関数として実装されている．check 関数は，現在アクセスして来ているサブジェクト番号，アクセスしようとしている機能の番号を元に，アクセスルールの配列を参照し，アクセスの可否の情報を取り出す．

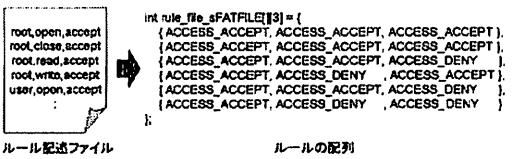

図7 アクセスルールの変換

3.2 評　価

アクセス制御機構の性能評価を行うため，アクセス制御機構を付加しない場合とする場合の実行時間を測定することで，アクセス制御機構によるオーバーヘッドを調べた．評価に用いた環境を表1に示す．

表1 動作環境

プロセッサ	SH3（SH7727 [4]）
ボード	MS7727CP02
動作クロック	96MHz
PC カードコントローラ	MR-SHPC-01 V2T-F
PC カードアダプタ	SDAD-38-J60（SanDisk）
メディア（CF）	32MB RCF-X （BUFFALO）
コンパイラ	gcc（3.3）
コンパイルオプション	-O2

評価用アプリケーションのコンポーネント図を図8に示す．このアプリケーションでは，動作環境であるターゲットボードに搭載されたディスク内のファイルを保護すべき資源として扱う．ファイル操作を行うために，組込みシステム向けの汎用FAT ファイルシステムである，FatFS [5] をコンポーネント化したものを用いる．

実行時間を測定した結果を表2に示す．実行時間は各機能を10000回呼び出した平均値である．この結果は，アクセス制御機構の実行時間が比較的短い事を示す．close の実行時のオーバーヘッドが大きな値になっているが，絶対的な実行時間の差

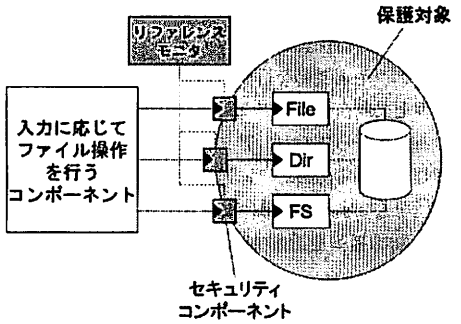

図8 評価用アプリケーションのコンポーネント図

表2 実行時間の測定結果

	アクセス制御機構なし（μs）	アクセス制御機構あり（μs）	オーバーヘッド（%）
open	47	50	6.38
close	7	9	28.57
read	2554	2586	1.25
write	4016	4047	0.77

としては僅かである．

また，作成したプログラムのコードやデータのサイズを，アクセス制御機構がある場合とない場合について調べて比較した結果を表3に示す．データ部の違いは，セキュリティコンポーネントが持つ関数テーブルとリファレンスモニタのアクセスルールのテーブルによるものであり，保護対象の数と保護対象が提供する機能の数，保護対象にアクセスするサブジェクトの数に比例して増える．プログラムの全体的なサイズに対するオーバーヘッドは2% 程度と小さく，実行時間の測定結果と合わせて，組込みシステムに適用しても問題ない結果となっている．

表3 プログラムサイズの測定結果

	アクセス制御機構なし（KB）	アクセス制御機構あり（KB）	オーバーヘッド（%）
Text	54166	55750	2.92
Data	1592	1700	6.78
BSS	21184	21184	0
Total	76942	78634	2.02

4. メモリ保護機構

コンポーネントを保護する手段としてアクセス制御とメモリ保護が考えられ，前章ではアクセス制御機構についての説明を行った．本章ではメモリ保護機構の提案を行う．

近年の組込みシステムでは OS でメモリ保護機構が要求されている．この理由として，カーナビや高機能な携帯電話，情報家電のような大規模な組込みシステムにおける，大きく複雑なプログラムが，他のプログラム領域を破壊するような信頼性をなくす結果をもたらしてしまうことが挙げられる．また，組込みシステムの CPU がより高速になり，オーバーヘッドが相対的に小さくなることも挙げられる．PC やサーバといった汎用

のコンピュータシステムでは，多くのセキュリティ機構がメモリ保護やアクセス制御を持っている [6]．セキュリティ機構は，CPU や，Java 実行環境のような言語ベースの実行機構における Memory Management Unit(MMU) 機能を用いて実装される．Linux に提供された典型的なメモリ保護は，システムレベルとユーザレベルの 2 レベルのメモリ保護によりカーネル領域を保護する．各プロセスの領域は複数の仮想アドレス空間によって保護される．

本稿では，コンポーネントの保護を目的としながらも，より一般的なメモリ保護機構（GMPE）を提案する．

4.1 要　　求

- 複数領域のメモリ保護

汎用コンピュータシステムでは，保護されるべき資源は OS やサーバプロセスに配置される．しかしながら，汎用コンピュータシステムと異なり，組込みシステムでは保護されるべき資源が，OS からミドルウェアやアプリケーションにまで分散している．そのような資源には，メモリ領域や管理データにマップされたデバイス制御レジスタを含む．組込みシステムでは，ミドルウェアやアプリケーションがデータを直接管理するため，複数領域のメモリ保護が必要とされる．

- 入れ子構造のメモリ保護

TCP/IP プロトコルスタックや電子商取引プロトコルスタックのように，2 つまたはそれ以上の原始的なプログラム（デバイスドライバを含む）からなる複合プログラムのような，粒度の大きいミドルウェアを仮定する．複合されたコンポーネントを 1 つのコンポーネントとして扱うことで複雑さが減る．このような入れ子構造では，内側のプログラムは外側のプログラムから保護されるようにする．

- メモリ保護の動的な変更

アプリケーションのコンポーネントから呼び出されたシステムのコンポーネントは，アプリケーションのコンポーネントにアクセスできる（OS は信頼できるという考えの下）．同じレベルを持つコンポーネント同士では，互いへのアクセスは保護されるべきである

4.2 仕　　様

日本における組込みシステム用の事実上の標準 RTOS である ITRON 仕様 [9] を基にした，GMPE のための計算とメモリモデルを示す．

タスクは組込みシステムにおける実行単位であり，汎用 OS におけるプロセスが持つようなメモリ管理やファイル管理のような資源管理データを持たない．本稿ではメモリ管理の考え方として，ITRON 仕様の保護拡張 [10] のドメインを用いる．ドメインは保護のための単位で，タスクの実行モードのための「読み，書き，実行許可」のようなメモリ管理情報を持つ．実行モードは OS を保護するために提供され，少なくともカーネルモードとユーザモードの 2 つの実行モードがある．システムの実行と I/O 命令はカーネルモードにおいてのみ許可される．もしドメインがユーザモードで書き込み許可されず，カーネルモードでは許可される場合，ユーザモードのプログラムはドメインにデータを書き込むことができず，カーネルモードの方は書き込みできる，この場合，ドメインはユーザプログラムに対して書込み保護を持つ．よって，カーネルドメインとユーザドメインの 2 つのドメインを仮定する．リアルタイム OS やデバイスドライバはカーネルドメインに属し，アプリケーションタスクはユーザドメインに属す．リアルタイム OS のメモリオブジェクトやタスク，プログラムはそれぞれカーネルドメインやユーザドメインに属す，と言うことができる．本稿で提案する GMPE のために，[7], [8] における，ドメインの用いるマルチプロテクションページテーブルの機能を拡張し，タスクに関連するリージョンを取り入れる．リージョンは保護ドメインや，タスク実行時におけるドメインの動的変更を含む．ドメインの変更は，タスクがドメイン A の関数を実行するときに他のドメイン B の関数を呼ぶことで起こる．ドメイン A からドメイン B への他のアクセスは禁止される．図 9 を用いてドメインの変更を説明する．アプリケーションプログラム A のタスク T がミドルウェア M の関数 m を呼び，関数 m はデバイスドライバ D1 の関数 d1 を呼び，それからデバイスドライバ D2 の関数 d2 を呼ぶと仮定する．プログラム A はユーザモードで実行され，ミドルウェア M，デバイスドライバ D1，D2 がカーネルモードで実行されると仮定する．図 9 における影つきの部分は，タスクがアクセスを許可された領域である．A のタスク T が実行されるとき，T は A の領域へのアクセスが唯一許可される (1)．タスク T は M で実行され，T は M と A へのアクセスを許可される (2)．タスク T が d1 を呼んだ後，T は D1，M，A へのアクセスを許可される (3)．D1 と D2 はデバイスドライバなので，このようなドライバの保護レベルはほとんどの OS において同じである．しかしながら，この例では，D2 の管理データを破壊するような D1 の誤動作を阻止するため，GMPE において T は D2 の領域にアクセスできない．D2 で実行する場合も，D1 の場合と同様である (5)．

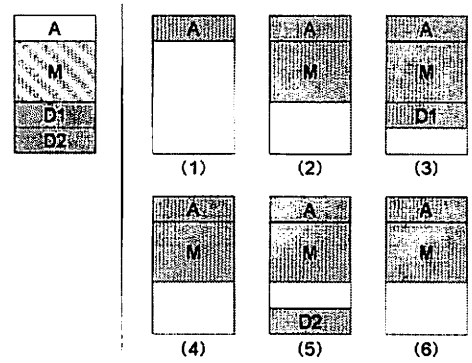

図 9　メモリ保護の変更

メモリ保護機構を利用するために，以下の OS の機能が必要である．

- パーミッションマトリクス

パーミッションマトリクスはドメイン間のアクセスパーミッションを定義する．図 9 の例では，ドメイン M がドメイン A へのアクセスパーミッションを許可し，ドメイン D1 はドメイン A

と M へのアクセスパーミッションを許可する．しかしながら D1 はドメイン D2 へのパーミッションを与えない．ドメイン D2 から D1 もまた同様である．

- ドメイン変更機能

ドメイン変更機能は，パーミッションマトリクスに基づき，タスクが関連付けられるリージョンのページテーブルを更新する．ドメイン変更機能は，カーネルモードを除く実行モードでドメインが変更される時に，システムコールかソフトウェアトラップで実装される．もしカーネルモードで発生するならば，ドメイン変更機能は関数呼び出しによって実装される．

4.3 設計と実装

前節で説明したメモリ保護機能は，MMU を用いて実装される．MMU は仮想アドレスから物理アドレスへのマッピングを実行し，ページテーブルを用いてメモリアクセスを調べる．ページテーブルのエントリは，仮想アドレスに対応する物理アドレスとメモリ保護情報を持つ．

仮想アドレスと，対応するページテーブルエントリ情報 (物理アドレスと保護情報を含む) をエントリが持つ MMU の Translation Lookaside Buffer(TLB) は，効率良くアドレス変換するのに用いられる．組込みシステムで用いられる RISC プロセッサでは，TLB の管理はソフトウェアで行い [11]，TLB エントリの中身を修正し，TLB エントリを効率よく置き換えることで対処する．本稿で対象としている SH3 RISC プロセッサ (SH7727) の TLB は図 10 のように，4 ウェイ 32 エントリの構成になっており，エントリ中に仮想ページ番号，物理ページ番号，保護キーデータが含まれている．参照する TLB エントリは仮想ページ番号の一部とアドレス空間識別子 (ASID) の排他的論理和によって決定でき，同じ仮想ページに対するアクセスでも異なる TLB エントリを参照できる．そのため，例えば異なるエントリにあらかじめメモリ保護パターンを書き込んでおけば，ASID の変更だけでメモリ保護パターンを変更でき，少ないオーバーヘッドで仮想空間のパーミッションを変更することができると考えられる．

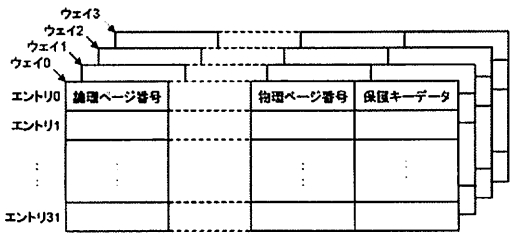

図 10 SH3 の TLB 構成

GMPE の設計は制限を持つことに注意しなければならない．それらの制限の一つは，カーネルモードで実行されるプログラムの悪意のある書き換えを防ぐことができないことである．書き換えを避けるために，プログラムはより安全な領域に移動されなければならない．

5. 結　論

本稿では，TECS 用のアクセス制御機構を開発し，実装と評価を行った．その結果，実行時間，プログラムサイズの面から見て，組込みシステムに適用しても問題のない範囲に収まっていることを示した．現状では，リファレンスモニタの内容であるアクセスルールを，全てのサブジェクトや機能に関して記述しなければならない．しかしながら，アクセスルールのサイズはサブジェクトや機能の数に比例して増えてしまうため，大きなシステムにアクセス制御機構を導入する場合は，記述する量が増えて手間がかかってしまう．そのため，今後の課題として，システム全体のセキュリティポリシーを記述し，ポリシーを基にアクセスルールを生成する仕組みを作ることで，設定にかかる作業量を軽減することが考えられる．

また，組込みシステムのための一般的なメモリ保護機構を提案した．メモリ保護機構は，複数の領域や入れ子になった領域の保護，メモリ保護領域の動的な変更機能を提供する．特徴は，非常に柔軟で一般的，かつ組込みアプリケーションの多くの種類に適していると考えられることである．組込みコンポーネントシステムとアクセス制御機構をメモリ保護機構と統合することで，より安全で安心な組込みプログラム実行環境を構築できる．今後の課題として，提案した一般的なメモリ保護機構の実装と評価を行うことが挙げられる．

謝　辞

本稿は IPA の 2006 年度下期未踏ソフトウェア創造事業の支援によるものである．また，本稿の一部は派遣型高度人材育成協同プランにおいて実施されたものである．

文　献

[1] IPA, “アクセス制御に関するセキュリティポリシーモデルの調査 報告書,” http://www.ipa.go.jp/security/fy16/reports/access_control/policy_model.html, 2004.

[2] 山田 晋平, 安積 卓也, 大山 博司, 中本 幸一, 高田 広章, “組込みソフトウェアコンポーネントにおけるセキュリティフレームワークの設計,” 第 51 回システム制御情報学会, May. 2007.

[3] 中本 幸一, 安積 卓也, 山田 晋平, 大山 博司, 高田 広章, “組込みシステム向けアクセス制御ポリシーについて,” 第 22 回電子情報通信学会アシュアランスシステム研究会, Nov. 2007.

[4] ルネサステクノロジ, “SH3-DSP SH7727 ハードウェアマニュアル,” Rev.5.00, Dec. 2005.

[5] ChaN, “汎用 FAT ファイルシステム・モジュール,” http://elm-chan.org/fsw/ff/00index_j.html.

[6] A. Silbeschatz, P. Galvin, and G. Gagne, “Operating System Concepts,” John Wiley & Sons, Inc., 6th edition, 2002.

[7] 品川 高廣, 河野 健二, 高橋 雅彦, 益田 隆司, “拡張コンポーネントのためのカーネルによる細粒度軽量保護ドメインの実現,” 情報処理学会, 第 40 巻, 第 6 号, pp.2596-2606, June 1999.

[8] M. Takahashi, K. Kono, and T. Masuda, “Efficient kernel support of fine-grained protection domains for mobile code,” Proc. 19th IEEE International Conference on Distributed Computing Systems, pp.64-73, May. 1999.

[9] 坂村 健, “μITRON4.0 仕様 Ver.4.00.00,” トロン協会, 1999.

[10] バージョンアップ WG, “μITRON4.0 仕様 保護機能拡張 Ver.1.00.00,” トロン協会, 2002.

[11] A. S. Tanenbaum, “Modern Operating System,” Prentice Hall, 3rd edition, 2007.