Vol.2013-SLDM-160 No.2
Vol.2013-EMB-28 No.2
2013/3/13

メモリ保護対応 RTOS におけるメモリ保護機能のテスト

風間 佳之[†1]　作道 直樹[†2]　段 慧[†3]　木戸脇 有美[†3]　平橋 航[†4]
鴫原 一人[†4]　海上 智昭[†4]　本田 晋也[†4]　高田 広章[†4]

近年の組込みソフトウェアの大規模化, 高機能化のため, RTOS に対するメモリ保護機能の需要が高まっている. RTOS はソフトウェアの中核をなすため, 高い品質が求められるが, メモリ保護機能に対するテストプロセスやテスト手法, テストの規模は明らかになっていない. 本論文では, RTOS のメモリ保護機能に対するテストプロセスやテスト手法, テストの規模, およびテストの効果を明らかにした. メモリ保護機能に対応した RTOS として, AUTOSAR 仕様ベースの RTOS を使用した.

Memory Protection Test for Memory-Protected RTOS

YOSHIYUKI KAZAMA[†1]　NAOKI SAKUDO[†2]　KEI DAN[†3]
AMI KIDOWAKI[†3]　WATARU HIRAHASHI[†4]　KAZUTO SHIGIHARA[†4]
TOMOAKI UNAGAMI[†4]　SHINYA HONDA[†4]　HIROAKI TAKADA[†4]

Needs for RTOS memory protection is growing due to recent trends in largescale, and high functional software. As RTOS compromises the core of software architecture, RTOS engineers are often faced with high levels of quality requirements. Despite the high and complicated requirements, not much has been clarified about the test process and method, or the test scale for memory protection. By adopting a new test method for AUTOSAR specification based RTOS, the present paper introduced the test process for memory protection, test method, as well as the effectiveness of the test procedure.

1. はじめに

近年，組込みシステムの分野においてソフトウェアの大規模化，複雑化が進んでいる．加えて，ソフトウェアの開発期間短縮の要求も高まっており，ソフトウェアのモジュール化が促進された結果として，外部委託開発した複数のアプリケーションを 1 つのハードウェア上で実行させる事例が増えている．また，アプリケーションの不具合に起因する誤作動による致命的な損害の発生を予防するため，リアルタイム OS(以降，RTOS)の保護機能に対する需要が品質および信頼性確保の観点から高まっている．

RTOS が持つ保護機能のうち，代表的なものとして，メモリ保護機能が挙げられる．メモリ保護機能とは，不具合を含む可能性のあるアプリケーションがアクセスを許可されていないメモリ領域へアクセスし，誤作動を引き起こすことを防ぐために, メモリアクセスを制御する機能である. メモリ保護機能を持つ RTOS 仕様として，μITRON4.0/PX 仕様[1]や，OSEK の HIS 拡張仕様[2]，AUTOSAR OS 仕様[3]などが提案されている．これらの RTOS は車載向けシステムとして利用されることを前提としているものが多いため，実装における品質保証のために網羅的なテストが必要となる．しかし，テスト実施の参考になるような，RTOS のメモリ保護機能のテストの事例報告については，我々が知りうる限り存在しない．また，テストの網羅度向上に伴い，テスト開発コストが増大する課題がある．

本論文は，RTOS に対するメモリ保護テストの事例として，AUTOSAR OS 仕様に準拠した RTOS である TOPPERS/ATK2(AuTomotive Kernel version 2;以降, ATK2)[4] のメモリ保護機能に対するテストスイート開発を題材に，テスト手法，プロセス，規模，及び効果を明らかにする．ここで，我々がテストスイート開発を通じて確立した網羅的なメモリ領域へのアクセス実施によるメモリ保護機能のテスト手法の考え方は，特定の RTOS のメモリ保護仕様に依存しないため，他の RTOS のメモリ保護機能のテストに広く適用することが可能である．また，テストの網羅度向上に伴うテスト開発コスト増大の課題に対しては，我々が先に開発したテスト自動化ツールを用い，組み合わせテストケース，テストプログラムの自動生成を実現することで解決した．

2. TOPPERS/ATK2 とは

ATK2 とは，名古屋大学大学院情報科学研究科附属組込みシステム研究センター(NCES)と，複数の企業が参加するコンソーシアム型共同研究において開発した RTOS の名称である. AUTOSAR OS 仕様はメモリ保護機能をサポートするため，ATK2 もメモリ保護機能をサポートした．AUTOSAR OS 仕様には未規定の実装依存仕様が多いため,TOPPERS/HRP2[5](以降，HRP2)の実装を参考に，拡張を行いながら外部仕様書を規定し,外部仕様書をベースに実装を行なった．

†1 日本電気通信システム株式会社
NEC Communication Systems, Ltd.
†2 株式会社サニー技研
Sunny Giken Inc.
†3 富士ソフト株式会社
FUJISOFT INCORPORATED
†4 名古屋大学
Nagoya University

Vol.2013-SLDM-160 No.2
Vol.2013-EMB-28 No.2
2013/3/13

3. ATK2 におけるメモリ保護機能

ATK2 ではメモリ領域を先頭番地，サイズ，アクセス保護属性によって区別する．アクセス保護属性には，読出，書込，実行がある．

ATK2 では OS アプリケーション(以降，OSAP)を保護の単位としてメモリ保護を行う．OSAP とは処理単位の集合体である．処理単位とは，ATK2 によって実行されるプログラムの機能モジュールである．ATK2 の処理単位には，タスク，ISR，フックルーチン(以降，フック)がある．

ATK2 のメモリ保護機能では，メモリ領域をアクセス対象，メモリ保護のパーティション単位である OSAP をアクセス主体とし，メモリ領域に対する読出，書込，実行の操作を制御する．

3.1 OSAP によるメモリ保護の設定

OSAP は信頼 OSAP と非信頼 OSAP に分類される．信頼 OSAP はプロセッサの特権モードで動作する．また，全てのメモリ領域にアクセス可能である．非信頼 OSAP はプロセッサの非特権モードで動作する．非信頼 OSAP は，コンフィギュレーション時に許可されたメモリ領域にのみアクセス可能である．

OSAP によるメモリ保護の設定例を図 1 に示す．

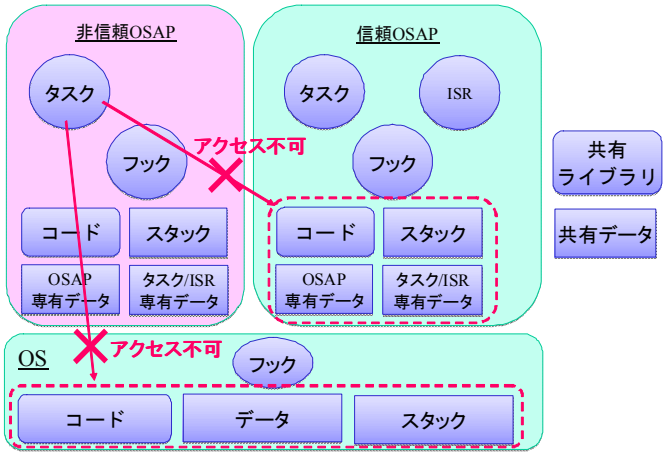

図 1 OSAP によるメモリ保護の設定例

3.2 メモリ領域に対する保護の設定

ATK2 が管理対象とするメモリ領域は以下の 4 つの領域に分類される．

- OSAP のメモリ領域
- 共有メモリ領域
- OS のメモリ領域
- 保護属性が配置されていないメモリ領域

OSAP のメモリ領域，共有メモリ領域，OS のメモリ領域は別々のセクションに分かれ，各セクションにアクセス属性が設定される．なお，無所属のメモリ領域は，非信頼 OSAP からの読出，書込，実行アクセスが禁止される．

表 1 メモリ領域の種別と保護設定

OSAP のメモリ領域	
	スタック (自処理単位からのみ読出，書込可能)
	コード (自処理単位からのみ実行可能)
	専有リードオンリーデータ (自 OSAP からのみ読出可能)
	共有リード専有ライトデータ (自 OSAP からのみ書込可能，他 OSAP からは読出のみ可能)
	専有リードライトデータ (自 OSAP からのみ読出，書込可能)
	専有周辺デバイスの領域 (自 OSAP からのみ読出，書込可能)
共有メモリ領域	
	コード (全 OSAP から読出，書込可能)
	共有リードオンリーデータ (全 OSAP から読出のみ可能，書込不可能)
	共有リードライトデータ (全 OSAP から読出，書込可能)
	共有周辺デバイス (全 OSAP から読出，書込可能)

3.3 メモリ保護機能の実現方法

ATK2 では，MPU(Memory Protection Unit)機能を用いてメモリ保護機能を実現した．MPU はプロセッサからのメモリアクセスを保護領域レジスタの設定値を元に判定して，禁止されているメモリアクセスと判定した場合には，そのアクセスを拒否した上で，プロセッサに対してメモリアクセス違反の CPU 例外を発生させるハードウェアの機能である．開発用のターゲットとして，Altera 社の Nios2 プロセッサを利用して，ターゲット依存部の実装とテストを行った．

MPU を利用した ATK2 のメモリ保護の概要を図 2 に示す．ユーザはシステム設計にて OSAP 間の保護設定を XML 形式のコンフィギュレーションファイルに記述する．コンフィギュレーション情報の生成には，コンフィギュレータという外部ツールを利用する．コンフィギュレータは，ユーザのコンフィギュレーション情報を入力として，タスクや ISR の設定情報の他，メモリ配置情報や MPU 設定情報を出力する．ATK2 は，コンフィギュレータの生成コードを参照して，ディスパッチャや非信頼 OSAP のフックの出入り口といった処理単位の切換え時に，MPU や特権/非特権モードを切換える処理を実施する．処理単位がメモリアクセスを実行すると，MPU はまず，アクセス先のメモリアドレスとアクセス種別が静的にコンフィギュレーションを行った保護設定に違反しないかチェックする．アクセス権限への違反時は，保護違反が起きた際に呼び出されるフックであるプロテクションフックが呼び出される．プロテクションフックでは，保護違反を起こした処理単位の強制終了や，OS シャットダウンなどの処理を行うことができる．

Vol.2013-SLDM-160 No.2
Vol.2013-EMB-28 No.2
2013/3/13

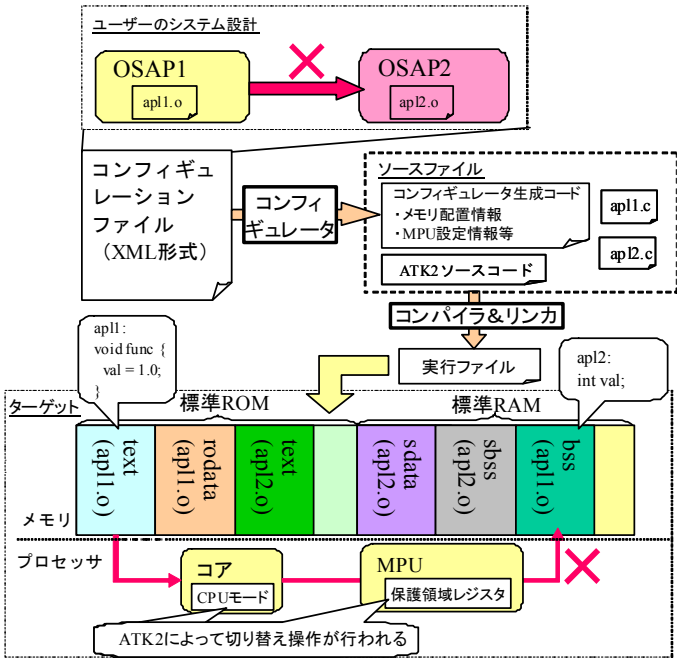

図 2　メモリ保護の概要図

4. ATK2 に対するメモリ保護テスト

4.1 テスト概要

ATK2 に対するメモリ保護テストで確認すべきことは，ターゲット上で動作するプログラムが，ユーザがアクセスを制約したメモリ領域へアクセスした場合，メモリアクセス保護違反が検知され，保護違反時処理を実施されることである．また，ユーザがアクセスを制約していない領域へのアクセスは，メモリ保護機能を使用しない場合と同様に実施できることも，確認すべきである．

そこで，ATK2 のメモリ保護テストでは，表 1 で示したような全種類のメモリ保護領域が生成情報に含まれるように，OSAP の設定情報を固定したコンフィギュレーションファイルを用意した．そして，実際にターゲット上で動作するテストプログラムの処理単位を切換えることで，MPU の設定や，特権/非特権モードを切換えながら，全種類のメモリ保護領域へのアクセスを実施し，アクセス保護属性に応じた処理が行われることを確認するテストシナリオを検討した．テストシナリオをベースに，メモリアクセスを実行するメモリ領域とアクセス種別，処理単位の切換えのそれぞれで，組み合わせパターンを検討し，テストケースのバリエーションを検討した．以下に詳細を述べる．

4.2 メモリアクセスを実行するメモリ領域とアクセス種別

OSAP の構成から生成されるメモリ配置情報より，メモリアクセスを実行するメモリ領域として，OSAP のメモリ領域，共有メモリ領域，OS のメモリ領域のどこをアクセスするかを検討した．さらに，アクセス種別として，読出，書込．実行を検討した．読出，書込アクセスは，メモリ領域の先頭アドレス，終端アドレスのそれぞれに対して実施するものとした．例を表 2 に示す．

表 2　テスト対象の OSAP の処理単位の組み合わせ(抜粋)

対象領域	メモリ領域の種類	メモリアクセス実行処理単位	アクセス種別 読出(先頭)	読出(終端)	書込(先頭)	書込(終端)	実行
ROM 領域	非信頼 OSAP 1 の専有コード領域	非信頼 OSAP1	○	○	×	×	○
		非信頼 OSAP2	×	×	×	×	×
		信頼 OSAP，OS	○	○	○	○	○
	非信頼 OSAP 1 の専有リードオンリー領域	非信頼 OSAP1	○	○	×	×	×
		非信頼 OSAP2	×	×	×	×	×
		信頼 OSAP，OS	○	○	○	○	○

○：アクセス許可　×：アクセス禁止

4.3 処理単位の切換え

ATK2 が「MPU の設定を切換える箇所」と「特権/非特権モードを切換える箇所」に着目し，MPU の設定や，特権/非特権モードの切換えが発生する処理単位の組み合わせパターンを検討した．OSAP，OS の単位で，必要な処理単位の組み合わせを検討した(表 3)．組み合わせ制約として，MPU,特権/非特権モードが切り替わらない組み合わせ，仕様上処理単位の切換えが起こらない組み合わせなどは除外した．テストケース設計では，表 3 でテスト対象とした OSAP，OS に所属する処理単位の組み合わせを検討した．

表 3　テスト対象の処理単位切換えパターン

		切換元 信頼 OSAP1 の処理単位	非信頼 OSAP1 の処理単位	OS の処理単位
切換先	信頼 OSAP の処理単位	×	○	×
	非信頼 OSAP1 の処理単位	○	○※	○
	非信頼 OSAP2 の処理単位	×	○	×
	OS の処理単位	×	○	×

○：組み合わせテスト対象　×：組み合わせテスト対象外
※：スタックに対する MPU の切換えのみ確認

5. メモリ保護テストにおける課題

テストケース設計を終えた段階で，予想される組み合わせテストケースの数は約 3 万件となった．これにより，テスト実装工程では以下の課題が発生すると予想された．

5.1 膨大なテストパターンの作成

テストシナリオで実現する処理単位の切換え，アクセス領域，アクセス種別などの組み合わせパターンは，仕様制約などを考慮すると複雑であり，漏れのない組み合わせテストパターンを生成するには，相当の工数がかかる．

5.2 テストプログラム実装工数

テストプログラム 1 件あたりの実行数は 200 行を超えた．全ての組み合わせパターンに該当するテストプログラムを手動で実装することは，工数から考えて現実的でない．更に，複数人でテストプログラムを実装した場合，テストプログラムの記述がばらつきやすいため，レビューおよび後戻り修正のコストによる作業の遅延がリスク発生する．

5.3 テストのポータビリティ

ATK2 では，ポータビリティ確保のため，ターゲット依存部と，非依存部を，明確に切り分けることで，ポータビリティを確保している．ATK2 を別ターゲットに移植した上で，メモリ保護テストを実施する場合を考慮すると，テストのポータビリティを確保する必要があるが，メモリ保護機能は，対象とするメモリ領域の種別やアクセス可否など，テストケースがターゲットに依存するため，ターゲット依存部と非依存部を明確に切り分けることが困難である．

5.4 テスト実施時間

メモリ保護テストでは，メモリ保護違反時の処理として ATK2 のシャットダウンが行われるケースが存在する．ATK2 はシャットダウン後，無限ループ処理を実施するため，テストケース 1 件毎に個別のバイナリデータを用意してテストする必要があった．この場合，ターゲットのロードおよび実行を繰り返すことになり，1 件あたりのテスト実行時間の平均は，12 秒程度であるので，全てのテストケースを実施した場合の実施時間の見積もりは約 3.5 日以上となり，テスト実施に膨大な時間がかかる．

6. ツールによる課題の解決

上述の課題を解決するため，ATK2 のメモリ保護テストでは，我々が先に開発したツールの利用を検討し，テストの自動化を行った．

6.1 組み合わせテストケース生成ツールによるテストケースの自動生成[6]

組み合わせテストケース生成ツールである PictMaster[7]に，処理単位の切換えパターン，メモリアクセスのアクセスパターン，そして，それらの組み合わせから仕様で決定される期待動作としての保護違反時処理の振る舞いを入力することで，全テストケースに必要な組み合わせパターンを自動的に網羅した(課題 5.1 の解決)．PictMaster で出力するテストケースは形式化されており，次節 6.2 で説明するツールに入力可能である．

6.2 AKTG によるテスト実装の自動化

形式化されたテストケースからテストプログラムを自動生成するツールである AKTG(Automotive Kernel Test Generator)[6]を利用し，メモリ保護テストのプログラムの自動生成を実施できるようにした(課題 5.2 の解決)．

AKTG を利用するために，YAML を使った階層的なデータ記述記法である TESRY 記法[8]によって，自然言語で記述されたテストシナリオを形式言語でスクリプト化することを検討した．TESRY 記法で記述されたテストケースは，TESRY データと呼ばれる．TESRY 記法では，複数の処理，後状態の組み合わせを記述することができるので，1 つ目の処理で「処理単位の切換え」，2 つ目の処理で「テスト対象のメモリアクセス」を記述することで，メモリ保護テストを実現した．「処理単位の切換え」の TESRY データの例を図 3 に示す．

```
pre_condition:
  #切換え対象の処理単位の前状態を設定
  TASK1:
    type: TASK
    tstat: RUNNING
    osap: TRUSTED_OSAP1
  TASK2:
    type: TASK
    tstat: READY
    osap: NON_TRUSTED_OSAP1
  #保護違反時処理用のフックの設定
  SHUTDOWN_HOOK1:
    type: SHUTDOWN_HOOK
    hookstat: AK_STOP

#処理単位切換え処理
#信頼 OSAP のタスクから非信頼 OSAP のタスクに切換える
do_0:
  id: TASK1
  syssrv: TerminateTask()
  rettype: StatusType
  retval: E_OK

#処理単位切換え確認
post_condition_0:
  TASK1:
    tstat: SUSPENDED
  TASK2:
    tstat: RUNNING

#以降，TASK2 がテスト対象のメモリアクセスを実施し，保護違反時処理が発生する場合，フックの起動を確認する
```

図 3　処理単位の切換えの TESRY データ

6.3 メモリアクセス処理の関数化によるターゲット依存部分の切り離し

テストケースにおいて，ターゲット依存部となる部分は，メモリアクセスを行う部分である．そこで，メモリアクセスを行う処理は，ターゲット依存のテストライブラリとして関数化し，引数を変えるだけで特定の領域にメモリアクセスできるようにした．これにより，メモリアクセス処理を行う関数をターゲット非依存，関数に渡す引数のバリエーションをターゲット依存として切り分けた(課題 5.3 の解決)．以下に概要を示す(図 4).

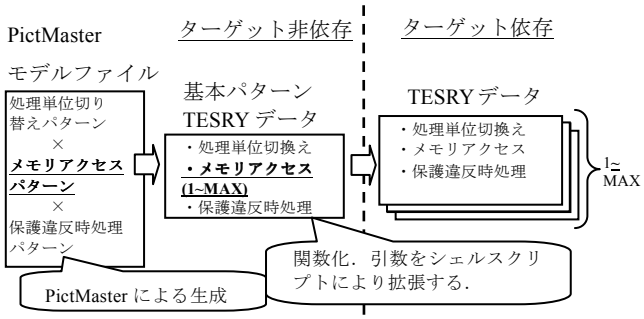

図 4　ターゲット依存部処理の関数化による非依存部との切り分け

PictMaster が出力するターゲット非依存となる基本パターンの TESRY データには，メモリアクセス関数のみを記載した．ターゲット依存部となるアクセス関数の引数の部分は, シェルスクリプトに基本パターンを入力することで，TESRY データのバリエーションを増やした．

6.4 シャットダウンの回避によるテスト連続実行の実現

シャットダウン処理の最後に実施されるターゲット依存の終了処理関数にて，メモリ保護テスト専用のコンパイルスイッチによって，無限ループによるリセット待ちに入らず，main 関数に戻って OS を再起動する処理を実装した(図 5).

AKTG は，メモリ保護テストを実施する際，上述のコンパイルスイッチを定義し，処理を有効にすることで，保護違反発生時も，ATK2 の再起動を可能にした．また，AKTG が定義するテスト用の main 関数では, 再起動回数の管理を行い，回数に応じて実行するテストケースを分岐させた．これにより，保護違反時処理を起こすメモリ保護テストの連続実行を実現して，テスト実施時間を短縮した(課題 5.4 の解決)．このような対応は，ATK2 およびテストプログラムが BSS セクションの初期化に依存せずに動作するように設計しているために可能であった．

```
/*
 *  ターゲット依存の終了処理
 */
void
target_exit(void)
{
    ～略～
#ifdef ENABLE_RETURN_MAIN
    /* 再起動時に必要となる変数の設定 */
    p_ctxosap = NULL;
    kerflg = FALSE;

    /* main 関数にジャンプ */
    Asm("movia r2, _ostkpt");
    Asm("ldw    sp, 0(r2)");
    Asm("call   main");
#endif /* ENABLE_RETURN_MAIN */
    ～略～
    /* 無限ループ*/
    while (1) {
    }
```

図 5　ATK2 終了処理の変更による再起動の対応

7. テストの実施と評価

7.1 実施したテストケース数

本論 6 で述べた手法を使用して，PictMaster のエクセルシート 30 枚からテストケースを自動生成すると，27,332 件になった．テストケースから AKTG を用いて約 1,100 万行のテストプログラムを自動生成し，テストを実施した．この規模のテストケースを手動で生成，保守するのは困難であるが，開発したツールの適用による自動化によって，テスト工程の大幅な軽減を実現した．

7.2 テスト実施時間

6.4 の対応前には約 88.4 時間程度かかっていたテスト実施時間を約 95%削減して，約 4.3 時間にまで短縮した．

7.3 検出した不具合

テスト実施により発見した ATK2 の不具合は，実装の不具合が 2 件，コンフィギュレータの不具合が 3 件の合計 5 件であった．また，外部仕様書の不具合を 4 件発見した．

実装では，非信頼 OSAP のフックから別の非信頼 OSAP のフックに切換えるテストケースにおいて, MPU の切換えが行われず，メモリ保護違反を検出できない不具合を検出した．コンフィギュレータでは，メモリ領域の保護属性を正しく生成できず，書込できないメモリ領域に書込ができてしまう不具合を検出した．また，組み合わせテストの設計中，外部仕様書に振る舞いが規定されていない不具合を検出した．

このように，仕様上想定される処理単位の切換えパターンを網羅的に組み合わせたことで，OS 内部のパスを網羅

的に実行することが可能になり，結果として実装上の不具合の検出につながった．同時に，コンフィギュレータが出力する静的なコンフィギュレーションファイルの不具合も検出した．また，組み合わせテストの観点をテスト設計に用い，マトリクスなどにテスト仕様をまとめたことで，未規定の仕様も検出した．

8. 他 RTOS への適用検討

今回検討したメモリ保護テストを，メモリ保護機能をもつ他 RTOS へ適用する場合の課題の検討を行う．本章では，メモリ保護の仕様差分に対するメモリ保護テストの適用課題に着目するため，ターゲットに依存する差分は言及しない．

8.1 他の AUTOSAR OS 仕様に準拠した RTOS に対するメモリ保護テストの適用

他の AUTOSAR OS 仕様に準拠した RTOS に対してメモリ保護テストを適用する場合，テストシナリオおよびテストケースを流用可能である．なお，AUTOSAR OS 仕様のメモリ保護仕様は未規定の部分が多いため，ATK2 では多くのメモリ保護仕様を独自規定している．よって，純粋に AUTOSAR OS 仕様に則ったテストを行う場合，ATK2 で独自規定した仕様に対するテストケースを省いたテストを行う必要がある．

8.2 µITRON などのメモリ保護の仕様が異なる RTOS への適用

今回検討したメモリ保護テストを，µITRON などのメモリ保護の仕様が異なる RTOS に適応する際の課題について考察する．

8.2.1 テストシナリオの適用

テストシナリオは，メモリ保護の実現方法(MPU，MMU)や，処理単位，保護粒度に依存しないため，どの RTOS のメモリ保護機能に対しても適用可能であると考える．

8.2.2 テストケースの適用

AUTOSAR OS 仕様と他の RTOS 仕様では，処理単位やメモリ保護で対象とするメモリ領域の保護粒度の仕様などが異なるため，仕様差分に応じてテストケースを再設計した上で適用する必要がある．再設計が必要なポイントは，処理単位の切換えパターンと，保護違反時処理の内容である．

8.2.3 自動化ツールの適用

テストシナリオは特定の RTOS のメモリ保護仕様に依存しないため，TESRY 記法によってテストケースをスクリプト化する手法は適用可能である．同様に，PictMaster を利用した組み合わせパターンの自動生成も適用可能である．AKTG を使って TESRY データからテストプログラムを生成する手法は，他 RTOS のコンフィギュレーション仕様が AUTOSAR OS 仕様と異なるため，AKTG のコード生成部を改変の上，適用する必要がある．

なお，他 RTOS が保護違反時処理として RTOS をシャットダウンする仕様である場合，テスト実施時間を短縮するために，今回と同様のシャットダウン回避策を導入することは実装上容易であると考えるが，前提として，RTOS が BSS セクションの初期化に依存せずに動作することが必須である．

9. おわりに

本論文では，ATK2 のメモリ保護テストとして，網羅的なメモリ領域へのアクセスを実施することで，メモリ保護機能のテスト手法を提案した．我々は，過去に開発したテストツールを流用または拡張して，テストケース，テストプログラムの自動生成を実施した．結果，27,332 件のテストケースを自動生成した上で，連続実行して，合計 9 件の不具合を検出した．本テスト手法の適用により，ATK2，コンフィギュレータの不具合の検出，テストプログラム実装およびテスト実施工数の削減を実現した．これにより，テスト手法の有効性が確認された．また，メモリ保護機能のテストの規模も明確にした．

謝辞 本手法の検討に参加し，さらに本論執筆にご協力頂いた NCES 研究員の皆様に謹んで感謝の意を表します．

参考文献

1) 高田広章編: µITRON4.0 仕様保護機能拡張(Ver1.00.00),トロン協会(2002)
2) HIS: OSEK OS Extensions for Protected Applications Version 1.0 (2003)
3) AUTOSAR，入手先(http://www.autosar.org/) (参照 2013-02-06)
4) TOPPERS/ATK2，入手先(http://www.toppers.jp/atk2.html) (参照 2013-02-06)
5) TOPPERS/HRP2，入手先 (http://www.toppers.jp/hrp2-kernel.html) (参照 2013-02-06)
6) 風間佳之,平橋 航,鴫原 一人,海上智昭,本田晋也,高田広章:AUTOSAR OS に対するテストケースおよびテストプログラムの自動生成,ソフトウェアテストシンポジウム 2012 予稿集, pp17-24(2012)
7) PictMaster，入手先 (http://sourceforge.jp/projects/pictmaster/) (参照 2013-02-06)
8) 鴫原一人，眞弓友宏，森孝夫，本田晋也，高田広章: µITRON ベースの RTOS 向けテストプログラム生成ツール，電子情報通信学会論文誌 D, Vol.J95-D, No.4, pp.874-880, (2012)