Vol.2017-ARC-225 No.14
Vol.2017-SLDM-179 No.14
Vol.2017-EMB-44 No.14
2017/3/9

車載システム向けの仮想マシンの割込み応答性向上手法

本田 晋也[1,a)]　岡部 亮[2]　攝津 敦[2]

概要：車載システムの高機能化により，これまで社内で開発したソフトウェアのみを用いて開発していたシステムに対して，外部で作成されたソフトウェア（外部ソフトウェア）を組み合わせてシステムを実現したいという要求が出ている．外部ソフトウェアは安全度水準が低い一方，高い応答性が要求される場合がある．この様な外部ソフトウェアを既存のソフトウェアと分離して実行する方法として，仮想マシンを用いる方法がある．仮想マシンの実現は実行オーバヘッドが問題となるが，近年車載システムにおいてもハードウェア仮想化支援機能を搭載したプロセッサが登場しており，それらをサポートした仮想マシンモニタも開発されている．本研究では，ハードウェア仮想化支援機能を利用した仮想マシンを用いて，外部ソフトウェアの分離と高い割込み応答性を実現する機構について述べる．

A Virtual Machine Monitor to reduce interrupt latency for Automotive System

Shinya Honda[1,a)]　Ryo Okabe[2]　Atsushi Settsu[2]

Abstract: The increasing complexity of automotive system has led to introduction of the outsourcing software in automotive system development which has been consisted of in-house software. The outsourcing software are lower safety integrity level in compare with in-house software, some of which requires high interrupt response time. The integration of different safety integrity level software in the same ecu needs a partitioning mechanism such as virtualization by Virtual Machine Monitor (VMM) The virtualization presents the trade-offs in performance and complexity. To solve this problems, the hardware-assisted virtualization and VMM for automotive system are proposed. This work give a mechanism of full separation between outsourcing and in-house software with high interrupt response time using a VMM with the hardware-assisted virtualization.

1. はじめに

近年，車載システムの高機能化に伴い，ECU 上で実行されるソフトウェア（車載アプリケーション）の複雑化・大規模化が進んでいる．車載システムは高い信頼性が要求されるため，これまではある ECU のプログラムはその ECU を開発しているサプライヤ（ECU サプライヤ）が全てのソースコードを開発するのが通常であった．しかしながら，車載アプリケーションの複雑化・大規模化に伴い，外部で開発されたソフトウェア（外部ソフトウェア）を使用する必要が出てきている．

これまでの車載システムは，メモリ保護や時間保護といった保護機構を持つ RTOS（保護 RTOS）を使用せずに実現されてきた．これは，前述のようにある ECU のプログラムはその ECU のサプライヤが全て同じ安全度水準で開発するためである．一方，外部ソフトウェアは安全度水準が低い場合があり，この場合，外部ソフトウェアの実行結果を監視する安全系のソフトウェアを組み合わせることで信頼性を確保する必要がある．このように，安全度水準の低い外部ソフトウェアと安全度水準の高い安全系のソフトウェアを ECU 内に共存させる場合は，保護 RTOS によるパーティショニングが必要である．

そのため，車載システム向けの標準仕様である AUTOSAR 仕様 [3] では，保護 RTOS として，保護機能付きの OS（保護付き AUTOSAR-OS）を策定しており，その実装を導入する方法がある．しかしながら，保護付き AUTOSAR-OS

[1] 名古屋大学 大学院情報科学研究科
愛知県名古屋市千種区不老町
[2] 三菱電機株式会社 情報技術総合研究所
[a)] honda@ertl.jp

Vol.2017-ARC-225 No.14
Vol.2017-SLDM-179 No.14
Vol.2017-EMB-44 No.14
2017/3/9

表 1　対象システムを構成するプログラム
Table 1　Programs on the Target System

プログラム	安全度水準	OSAPI 呼び出し	割込み応答性	周期	中断
制御系処理	低	無し	短	短	不可
安全系処理	高	有り	長	長	可

は，OS の API（OSAPI）呼び出しの実行オーバヘッドが大きいという問題がある．また，保護付き AUTOSAR-OS を使用したとしても，プロトコルスタック等の BSW と呼ばれる AUTOSAR のモジュールは特権モード動作するためパーティショニングの対象とならない．BSW は安全度水準の低い外部ソフトウェアとする場合があり，この場合，BSW から安全系のソフトウェアを保護できないという同様の問題が発生する．

これらの問題を解決する手法として，仮想マシンモニタ（VMM）と仮想マシン（VM）を用いて大きな単位で保護する手法が挙げられ，車載システムを対象とした VMM（A-VMM）が提案されている [1], [2]．前述の例の場合，VM を用いて外部ソフトウェアを VM 内で実行することにより，安全系のソフトウェアと別の安全度水準とすることが可能である．しかしながら，外部ソフトウェアを VM 内で実行した場合，外部ソフトウェアの高速な割込み応答性を実現することは困難である．具体的には，安全系のソフトウェアの実行時間を保証するため，安全系のソフトウェアが動作中は VM は実行出来ないためである．

本研究ではこの問題に対して，VM 内で実行する外部ソフトウエアの高速な割込み応答性を実現する VMM の機構を提案する．提案する機構は，VM で処理する割込みを VMM で一旦受け取り時間監視を有効にした後，VM 内のハンドラを呼び出すことで，高速な割込み応答時間とパーティショニングを両立する．VM 内のハンドラの呼び出し方法の違いで VM-ISR1-と VM-ISR0 の 2 種類の機構を提案する．

2. 対象とする車載システム

本章では本研究で対象とする車載システムについて説明する．対象システムは通常系処理と安全系処理で構成される．通常系処理は制御系処理や通信系処理として構成される．制御系処理と安全系処理の各プログラムの特性を**表 1** に示す．制御系処理が最もリアルタイム性が要求される処理であるため，ISR として実行され，起動要因となる割込みは 100μsec 周期程度で発生する．制御系処理はドライバを用いた制御処理や，共有メモリを用いた通信などを行う．安全系処理は通常系処理を含むシステム全体の安全性を監視する．

各処理の時間要件を**図 1** に示す．安全系処理は監視周期 (T-CYC) 内で監視処理を T-PET 時間実行する必要がある．T-PET 時間毎の処理は中断されても問題ない．制御系処理は，安全系の時間制約を満たすため，T-CYC と同じ単位時間 (NT-CYC) 内における起動割込み (NT-Int) 発生回数の上限値（NT-MaxInt）が定まっており，NT-MaxInt 回以上割込みが発生した場合はエラーとする必要がある．また，起動毎の最悪実行時間（NT-WCET）も定まっており，NT-WCET 以上実行を継続した場合もエラーとする必要がある．この要求を満たす範囲で制御系処理には可能な限り高速な割込み応答が要求される．また，制御系処理は一旦起動すると処理の中断は不可能である．

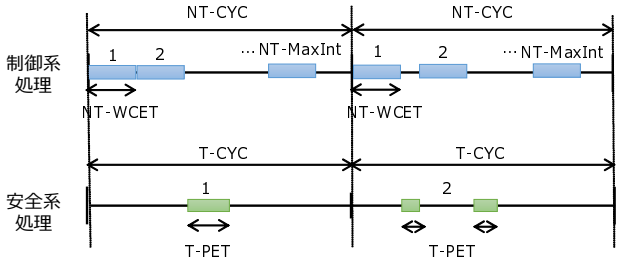

図 1　対象システムの時間要件
Fig. 1　Time Requirements of the Target System

なお，通信系処理は制御系処理や安全系処理の合間に動作し，処理の中断が許容される．

要件をまとめると次のようになる．

- **性能要件 1**：制御系処理の起動レイテンシを可能な限り小さくすること．
- **性能要件 2**：制御系処理は起動後は他の処理 (安全系処理も含む) に中断されないこと．
- **保護要件 1**：制御系処理を低信頼度で開発可能であること．
- **保護要件 2**：安全系処理の単位時間辺りの実行時間が確保されること．

3. 既存のパーティショニング機構

本章では対象システムを実現するための機構として，保護付き AUTOSAR-OS と A-VMM について説明する．

3.1 保護付き AUTOSAR OS

保護付き AUTOSAR-OS の保護機構と制御系処理の実現に必要な割込み機構について説明する．なお，保護機能を持たない AUOTSAR OS は，保護無し AUTOSAR-OS と呼ぶ．

3.1.1 保護機能

保護付き AUTOSAR-OS では，プログラムは，OS アプリケーション（OSAP）と呼ばれるグループに分けられる．OSAP には保護が有効な非信頼 OSAP と，無効な信頼 OSAP がある．非信頼 OSAP はユーザモードで動作させ，信頼 OSAP と OS を含む BSW は特権モードで動作させる．

Vol.2017-ARC-225 No.14
Vol.2017-SLDM-179 No.14
Vol.2017-EMB-44 No.14
2017/3/9

表 2 AUTOSAR OS の割込みハンドラの種類
Table 2 ISR Types on AUTOSAR OS

名称	OSAPI 呼び出し	OS 実行中の割込み	保護	起動オーバヘッド
ISR1	不可能	可能	なし	小
ISR2	可能	不可能	なし	小
非特権 ISR2	可能	不可能	あり	大

メモリ保護に関しては，非信頼 OSAP からアクセス可能なメモリや I/O を制限する．そのため，非信頼 OSAP から BSW の機能（例えば OS の API 呼び出し）を使用するためには，ソフトウェア割込みによる呼び出しが必要となる．時間保護に関しては，タスクや割込みハンドラ（ISR）毎に実行時間や起動間隔を監視する機構を提供する．

3.1.2 割込み機構

表 2 に示す 3 種類の割込み機構を提供する．ISR1 は OSAPI を呼び出すことが出来ないが，OS 実行中に割り込むことが可能である．ISR2 は通常の ISR であり，非特権 ISR2 は保護付き AUTOSAR-OS において，タスクと同様に保護が有効となった状態で実行される ISR である．そのため，他の ISR と比較して起動オーバヘッド（プロセッサが割込みを受け付けた後，ユーザ記述のハンドラを呼び出すまでの時間）が大きい．

3.2 車載システム向け VMM

本研究では，RH850 ベースの評価用コアに搭載されているハードウェア仮想化支援機能（HAV:Hardware-asisted Virtualization）[5] を用いた A-VMM を用いる．

3.2.1 RH850 評価コアにおける HAV

RH850 は車載システム向けのプロセッサであり，パワートレイン等のリアルタイム性や高い性能が要求される用途で用いられている．HAV は RH850 評価コアに導入されたハードウェア仮想化支援機能である．概要を**図 2** に示す．HAV では既存のプロセッサが持つ動作モードに加えて新たな動作モードを追加している．既存の動作モードが， OS と OS 上で動作するアプリケーションとを区別するのに対し，新しい動作モードは， CPU 本来の機能をすべて利用できるネイティブマシン（NM）と，一部の機能だけを利用できる仮想マシン（VM）とを区別する．

動作モードが VM の間は，特定の機能の設定レジスタなどのプロセッサリソースを NM のリソースではなく， VM 用に独立した異なるリソースを参照して動作する．これらのリソースを VM コンテキストと呼ぶ．VM コンテキストは，プロセッサの用途に応じて 1 組以上の数が搭載される．

NM から VM への遷移は，ステータスレジスタに VM に遷移することを示すビットをセットして例外リターン命令を実行することで行う．VM から NM への遷移は，コールゲートとして，後述する割込みや，例外発生時，NM 呼び出し命令実行時に遷移するのみとしている

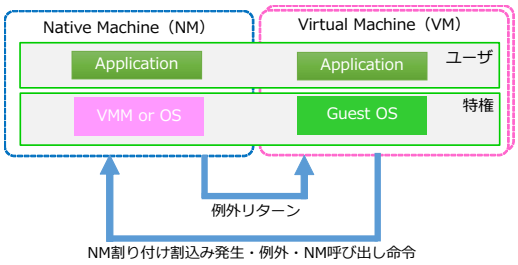

図 2 RH850 評価コアにおけるハードウェア仮想化支援機能
Fig. 2 HAV of RH850 evaluation core

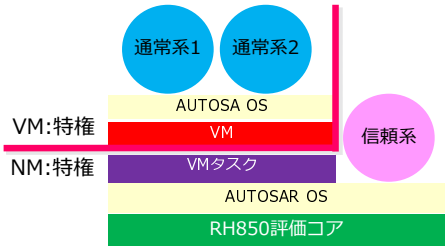

図 3 車載システム向け VMM の概要
Fig. 3 Overipew of Automotive System VMM

割込みは割込みチャネル毎に NM ないし任意の VM に割り付け，優先度制御は VM 毎に行う．割込みが発生した場合の振る舞いを図 2 に示す． VM で実行中に NM 割り付けの割込み（NM-ISR）が発生すると即座に NM に遷移する．また VM 実行中は NM 割り付けの割込みは禁止出来ないため，NM 側で実行するプログラムの時間保護を実現することが出来る．ただし，NM で割込み優先度を上げた状態で VM に遷移すると，その優先度以下の NM-ISR は受け付けられない．

3.2.2 HAV を用いた車載システム向け VMM

A-VMM の構成は，必要なハードウェアリソースが少ないホスト型となっている（**図 3**）．ホスト OS には AUTOSAR OS を用いる．保護が必要ないプログラムはホスト OS 上で実行する（図 3 中の信頼系）．

VM はホスト OS 上に VM を実行するタスク（VM タスク）がスケジューリングされると実行される．ホスト OS において VM タスクの実行を制御することで VM の実行を制御することが可能である．VM タスクも他のタスクと同様にスケジューリングされる．VM タスクのコードの例を**図 4** に示す．起動後に任意のコードを実行して必要に応じて VM に遷移する関数（vm_start()）を呼び出す．VM への遷移後，前述のように NM 割込みが発生して VM から NM に処理が移った後，VM タスクに再びディスパッチすると vm_start() からリターンし，VM タスク中のユーザーコードが実行される．そのため，VM のスケジューリングをユーザコード（図 4 中の Post/Pre User Code）により制御可能である．

Vol.2017-ARC-225 No.14
Vol.2017-SLDM-179 No.14
Vol.2017-EMB-44 No.14
2017/3/9

```
1:  TASK(VMTask) {
2:      /* Pre User Code */
3:      vm_start(p_vminib->rbase); /* Call VM */
4:      /* Post User Code */
5:      TerminateTask();
6:  }
```

図 4 VM タスクの例
Fig. 4 Example of VM Task

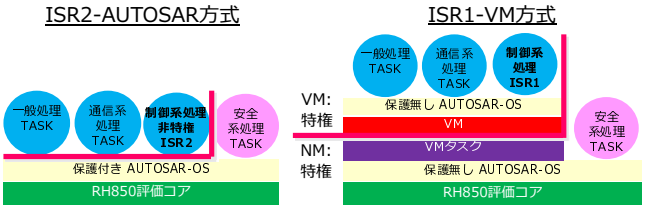

図 5 既存のパーティショニング機構による実現
Fig. 5 Realize with Existing Partitioning Method

4. 既存のパーティショニング機構による実現

3 章で述べたパーティショニング機構による対象システムの実現方法を図 **5** に示す．

4.1 保護付き AUTOSAR-OS による実現（ISR2-AUTOSAR）

保護付き AUTOSAR-OS を用いて，制御系処理を非特権 ISR2 で実現し，安全系処理は特権 OSAP に所属するタスクとして実行する方法である．

非特権 ISR2 は 3 章で述べたように起動オーバヘッドが大きいことや，OS 実行中は受け付けられないため，性能要件 1 を満たせない可能性がある．

また，制御系処理とする非特権 ISR2 の優先度はタスクである安全系処理より高いため，制御系処理の起動回数や実行時間を監視する機構が必要となる．保護付き AUTOSAR-OS の時間保護機構は実行時間監視と実行間隔監視である．前者は使用することが可能であるが，後者では起動回数の監視は不可能である．起動回数の監視を行うには OS の ISR の出入口処理を変更する必要がある．

4.2 車載システム向け VMM による実現（ISR1-VM）

A-VMM を用いて，制御系処理をゲスト OS 上の ISR1 として，安全系処理をホスト OS のタスクとして実行する方法である．どちらの OS も保護無し AUTOSAR-OS を用いる．

4.2.1 安全系処理の時間要件の実現手法

A-VMM による保護要件 2 の実現方法について説明する．システム構成とタイムチャートを図 **6** に示す．

タスクは VM タスクと安全系処理タスクで，VM タスクを高優先度とする．VM タスクは起動されると VM を実行する．その後，NM に割り付けた割込みが発生して VM から NM に遷移した後，再びスケジューリングされて実行が再開されると VM が終了するコードとする．タイマは T-CYC タイマと T-PET タイマを用いる．T-CYC タイマは安全系処理の T-CYC を実現するタイマであり，周期動作させる．タイムアウトすると T-CYC ISR が実行される．T-PET タイマは，安全系処理の T-PET 分の実行時間を計測する．タイムアウトすると VM タスクを起動する．

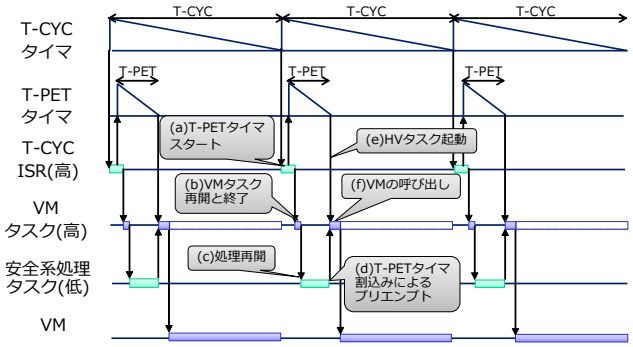

図 6 安全系処理の時間保護機構
Fig. 6 Time Protection Mechanism for Safety Process with VMM

これらの機構による実行を図 6 を用いて示す．まず，T-CYC タイマがタイムアウトして T-CYC ISR が実行されると T-PET タイマをスタートさせる（図中 a）．T-CYC ISR からリターンすると VM タスクの実行が再開され VM タスクは終了する（図中 b）．次に安全系処理タスクの実行が再開される（図中 c）．T-PET 時間経過後 T-PET タイマがタイムアウトして安全系処理タスクがプリエンプトされた後（図中 d），VM タスクを起動する（図中 e）．VM タスクは起動すると VM を呼び出す（図中 f）．この動作を繰り返すことにより，安全系処理の保護要件 2 を満たす．

4.2.2 その他の要件の実現

保護要件 2 以外について満たしているか確認する．

保護要件 1 に関しては，制御系処理をゲスト OS 上で実行するため，安全系処理のメモリへのアクセスを制限することが可能であるため，保護要件 1 を満たすことが可能である．性能要件 1 に関しては，制御系処理を起動する割込みは VM に割りつけられているため，安全系処理のタスクが実行している間は受け付けることは出来ないため，満たすことが出来ない．また，性能要件 2 に関しても同様に VM 実行中はホスト OS の割込みを禁止することが出来ないため満たすことは出来ない．

5. VMM 向け高速割込み機構

4 章で述べた通り，既存のパーティショニング機構では対象システムの要件を満たすことは出来ない．そこで，VMM を拡張して各要件を満たす機構を提案する．提案機構は

Vol.2017-ARC-225 No.14
Vol.2017-SLDM-179 No.14
Vol.2017-EMB-44 No.14
2017/3/9

図 7 VM-ISR の優先度
Fig. 7 Priority of VM-ISR

VM-ISR1-と VM-ISR0 の 2 種類の ISR 機構である．両者をまとめて VM-ISR と呼ぶ．VM-ISR の外部仕様と実装，提案機構によるシステムの実現方法について説明する．

5.1 外部仕様

VM-ISR のシステムのその他の処理単位との優先度の関係を図 **7** に示す．VM-xxx はゲスト OS 上の処理単位，NM-xxx はホスト OS 上の処理単位を示す．例えば NM-ISR1 は NM 上で動作するホスト OS の ISR1 を示す．NM-TASK_VM は VM タスクであり，NM-TASK_HIGH はホスト OS 中で VM タスクより優先度が高いタスク，NM-TASK_LOW は低いタスクを示す．

VM 内の処理単位（VM-xxx）は対応する NM の処理単位がホスト OS で実行されると，VM 内で実行される．VM タスク（NM-TASK_VM）がスケジューリングされると，ゲスト OS 上の処理単位（VM-ISR1,VM-ISR2,VM-TASK）が実行される．VM-ISR は NM-ISR1_VM が動作すると実行され，ホスト OS の ISR（NM-ISR1/2）より高い優先度で実行する．一方，NM-ISR0 よりは優先度は低い．NM-ISR0 は後述する VM-ISR の時間保護のための ISR であり，VM-ISR が定められた実行時間を超過して実行された場合に発生する．このような優先度とすることで，性能要件 1,2 を満たす．また，安全系処理は NM-TASK_ LOW として実行し，4.2.1 節で述べた方法により保護用件 2 を実現する．

保護要件 1,2 を実現するため，VM-ISR の実行に関しては次の監視を設ける．エラー発生時は NM-ISR0 を起動する．

- 割込み発生回数監視
  単位時間内 (NT-CYC) に VM-ISR の要因となる割込みの発生回数に上限を設けて監視し，上限値以上発生した場合はエラーとする．
- 実行時間監視
  VM-ISR の一回毎の実行時間に上限を設けて上限時間以上実行を継続した場合はエラーとする．

なお，実行時間監視と共に，VM-ISR1-/VM-ISR0 呼び出し時には T-PET タイマの停止と再開を行う．つまり，安全系処理の実行中に VM-ISR が実行された場合，VM-ISR の実行時間は安全系処理の実行時間に含めない．これにより安全系処理の実行時間を保証する．

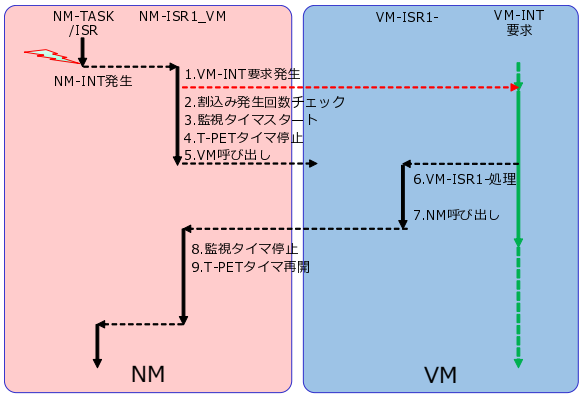

図 8 VM-ISR1-の実行シーケンス
Fig. 8 Execution Order of VM-ISR1-

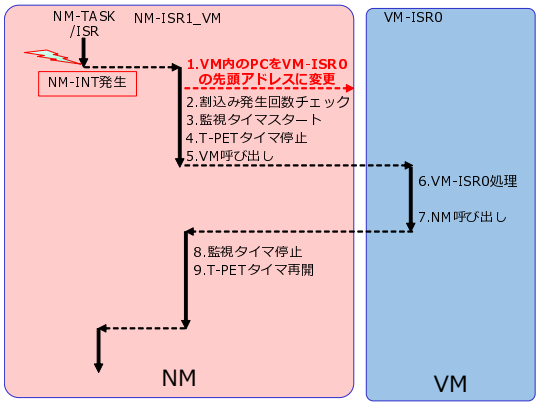

図 9 VM-ISR0 の実行シーケンス
Fig. 9 Execution Order of VM-ISR0

5.2 実装

前述の設計を実現するための実装について述べる．

5.2.1 VM-ISR1-

VM-ISR1-を起動する制御系の割込みは NM に割り付け優先度の設定は NM-ISR1 や NM-ISR2 より高く設定する．この割込みを NM-INT と呼び，起動されるハンドラを NM-ISR1_VM とする．

NM から発生可能な割込みを 1 つ VM に割り付ける（VM-INT）．VM-INT が発生すると，VM-ISR1-とするハンドラを実行するように VM 内の AUTOSAR OS をコンフィギュレーションする．

これらを用いた VM-ISR1-起動までの処理の流れを図 **8** に示す．NM-INT が発生すると NM で NM-ISR1_VM が起動される．NM-ISR1_VM では，VM 内で VM-INT が発生するように割込みを発生させ（図中 1），NM-INT の発生回数のチェックをした後（図中 2），VM-ISR1-の監視のためのタイマをスタートさせる（図中 3）．さらに T-PET タイマが動作していれば停止する（図中 4）．最後に VM を呼び出し（図中 5），VM に処理が移る．VM に処理が移ると，

VM-ISR1-が実行され（図中 6），処理が終了すると NM に呼び出しが戻る（図中 7）．NM に戻ると NM-ISR1_VM が再び実行され監視タイマを停止する（図中 8）．最後に T-PET タイマが起動時に動作していれば再開する．VM-ISR が不具合等でリターンしない場合には，監視タイマがタイムアウトして NM-ISR0 が実行される．この保護と，図中 2 の割込み発生回数のチェックにより，保護要件 1,2 を満たす．

性能要件 2 に関しては，NM-ISR1_VM は NM-ISR1 や NM-ISR2 より高い優先度に設定しており，VM-ISR1-はその延長で動作するため満たすことが可能である．性能要件 1 に関しては実装して計測することにより評価する．

5.2.2 VM-ISR0

前述の VM-ISR1-は，ISR を VM 内の AUTOSAR OS の ISR1 として実装して，NM から VM に対して割込みを発生させることで実現している．この方法は，既存の ISR1 からの変更量が少ないというメリットはあるが，VM 内で ISR1 を禁止している場合は即座に受け付けられない．そこで，VM 内の AUTOSAR OS がどの様な状態でも割込みを受け付ける VM-ISR0 を提案する．VM-ISR0 の実行シーケンスを**図 9** に示す．

VM-ISR1-との違いは，VM-ISR0 には VM 内の割込みは特に割り付けず，NM-ISR1_VM において，VM 内の PC を VM-ISR0 の先頭アドレスに変更し（図中 1），その後 VM 呼び出しをすることで（図中 5），VM に遷移すると VM-ISR0 が実行される．

6. 評価

評価として，各割込みハンドラの起動オーバヘッドについて評価する．次に，割込み応答時間を比較する．評価に用いた環境は，プロセッサ:RH850 評価コア 120Mhz，コンパイラ:GHS 2016.5.3 である．OS は TOPPERS プロジェクトの ATK2-SC1/SC3 1.4.0[4] を用いた．

割込みハンドラの起動オーバヘッドとして，プロセッサが割込みを受け付けた後，ユーザ記述のハンドラを呼び出すまでの時間を計測した．測定結果を**表 3**（a）に示す．NM-ISR1 は OSAPI 呼び出しが出来ない仕様であるため，ハンドラ呼び出しまでに OS 内部の変数等の設定が必要ないため，最も起動オーバヘッドが小さい．一方，VM-ISR1-や VM-ISR0 は監視タイマのスタート処理や各種コンテキストの保存・復帰処理が必要となるため，通常の ISR と比較して起動オーバヘッドが大きい．

割込み応答時間は，前述の起動オーバヘッドと割込み禁止区間を加算した値となる．表 3（b）に割込み禁止区間として，最長の OSAPI の実行時間を示す．割込み禁止区間に関しては，NM-非特権 ISR2 が最も大きい．これは，NM-非特権 ISR2 をサポートする保護付き AUTOSAR-OS においては，非特権 OSAP からは API をソフトウェア割込みで呼び出すためである．NM-ISR1 は原則 OSAPI 実行中でも割り込むことが可能であるが，プロセッサの仕様上 NM-ISR2 の入口処理では ISR1 を含む全 ISR を禁止する必要があるため，その値を記載した．VM-ISR1-及び VM-ISR0 においても同様に禁止される区間が存在する．特に VM-ISR1-に関しては，ホスト OS とゲスト OS の禁止区間が連続する場合があるため，VM-ISR0 より長い結果となっている．

表 3 測定結果
Table 3 Evaluation Result

ハンドラ	（a）起動オーバヘッド	（b）割込み禁止区間	（c）割込み応答時間
NM-ISR1	600ns	850ns	1,450ns
NM-ISR2	1,400ns	2,900ns	4,300ns
NM-非特権 ISR2	1,670ns	5,900ns	7,570ns
VM-ISR1-	2,283ns	1,783ns	4,066ns
VM-ISR0	1,466ns	933ns	2,399ns

以上より，最終的な割込み応答時間は表 3（c） に示す通り，VM-ISR1-及び VM-ISR0 に関しては，NM-ISR2 および NM-非特権 ISR2 以下の値となっている．この結果から提案機構は，性能要件 1 を満たしていると言える

7. おわりに

本研究では，対象とする車載システムで要求されるパーティショニングと高速な割込み応答を実現する A-VMM の割込み機構について提案し，実装と評価を行った．設計と評価の結果，提案機構は性能要件及び保護要件を満たしていることを確認した．今後の課題としては，提案機構の実システムへの適用と評価を挙げることができる．

謝辞 本研究の一部は JSPS 科研費 JP26330062 の助成を受けたものです．

参考文献

[1] D. Reinhardt and G. Morgan, ”An embedded hypervisor for safety-relevant automotive E/E-systems,” Proceedings of the 9th IEEE International Symposium on Industrial Embedded Systems (SIES 2014), Pisa, pp. 189-198 2014.

[2] 本田晋也, 鈴木均, 樋口正雄, 福井昭也, ”車載システム向けハードウェア仮想化支援機能による RTOS 一体型仮想マシンモニタ,” 情報処理学会 OS 研究会，Mar. 2017.

[3] AUTOSAR (online), available from ⟨http://www.autosar.org/⟩ (accessed 2015-10-24).

[4] TOPPERS/ATK2 (online),available from ⟨http://www.toppers.jp/atk2.html⟩ (accessed 2015-10-24).

[5] リアルタイム制御システムに適した V850 CPU 向け仮想化技術 (online), available from ⟨https://www.renesas.com/ja-jp/about/press-center/news/2010/news20100929.html⟩ (accessed 2017-01-19).