車載組込みシステムフォーラム（ASIF）
2020年度第2回スキルアップセミナー
2020/10/27

ASIL と QM ソフトウェアの混在のための各種手法の比較

本田　晋也
南山大学 理工学部
TOPPERSプロジェクトシニアテクニカルエキスパート
shonda@nanzan-u.ac.jp

最終更新：2020/10/07

アジェンダ

ASILとQMソフトウェア混在を実現するための各種手法について説明

- ASILとQMソフトウェアの混在
  - 機能パーティショニング
  - ECU統合

- ECU統合手法の比較
  - AUTOSAR OS 保護機能による統合：AUTOSAR
  - ATK2-TPによる統合：ATK2-TP
  - 制御向けプロセッサのマルチコアによる統合：MCU-MC
  - 仮想マシンによる統合：HV

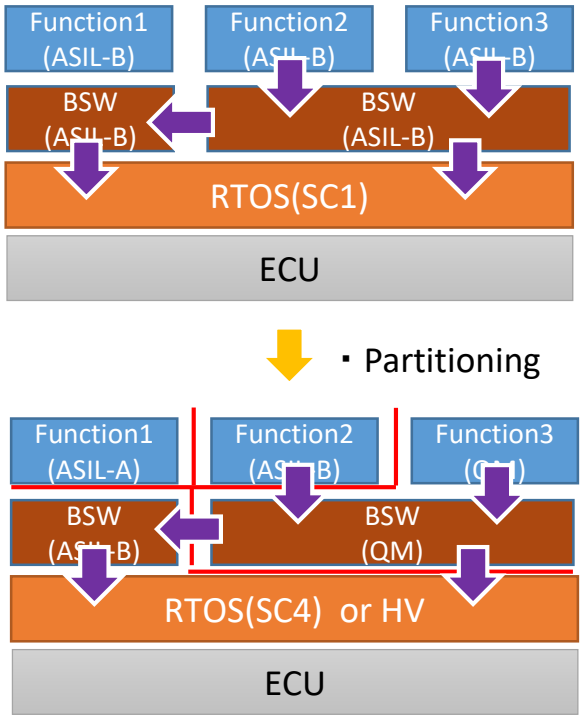
- 機能パーティショニングの実現手法検討

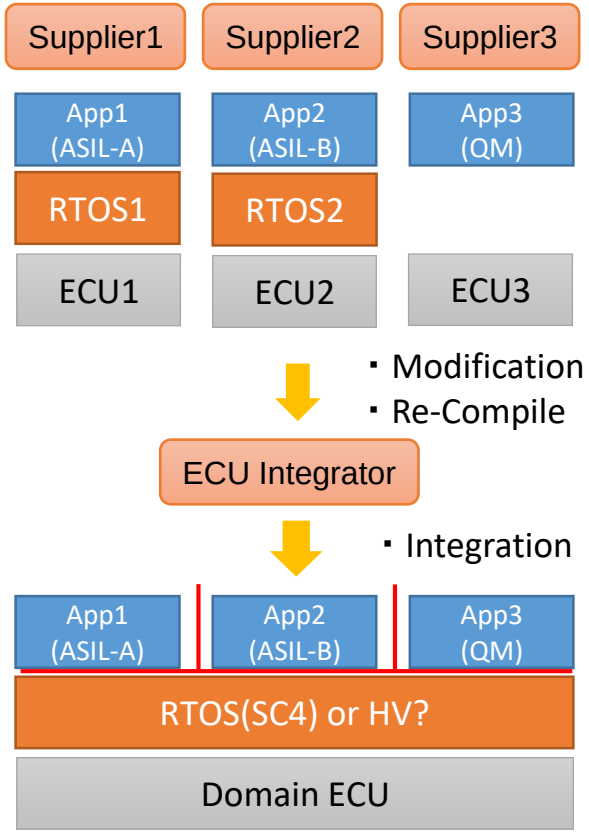
ASILとQMソフトウェアの混在

ASIL と QM ソフトウェアの混在

2種類のユースケース

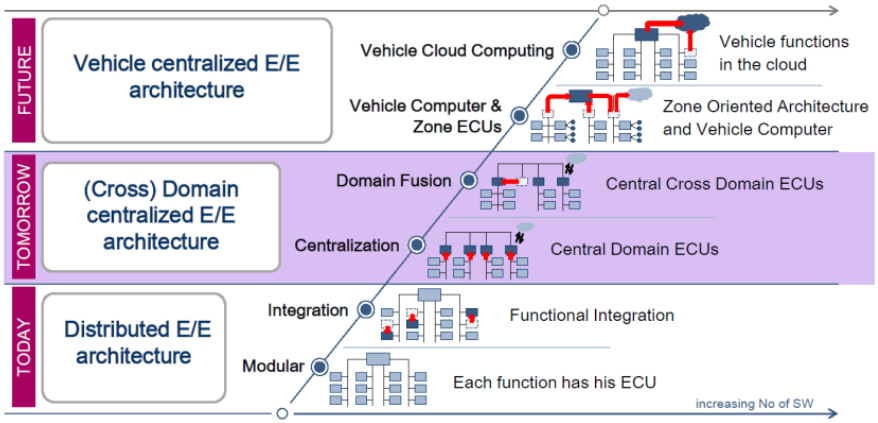
- 機能パーティショニング
  - 既存のシステムにパーテイションニングを適用することで，機能毎の安全度水準で開発可能とし，開発コストを下げる
- ECU統合
  - 安全度水準や開発元が異なるECUを1個のECUに統合する





ECU統合の背景

- 分散リアルタイムシステムの大規模・複雑化
  - 高度な機能，複数のコンピュータが連動した制御を実現するためにコンピュータ数が増加
    - 1機能 = 1ECU = 1サプライヤー
    - 搭載されるソフトウェアも大規模化
- 小型モビリティの実現
- ドメインコントローラ
  - 可能な限り各ECUの機能をまとめる
  - ガソリン，ハイブリッド，ディーゼル



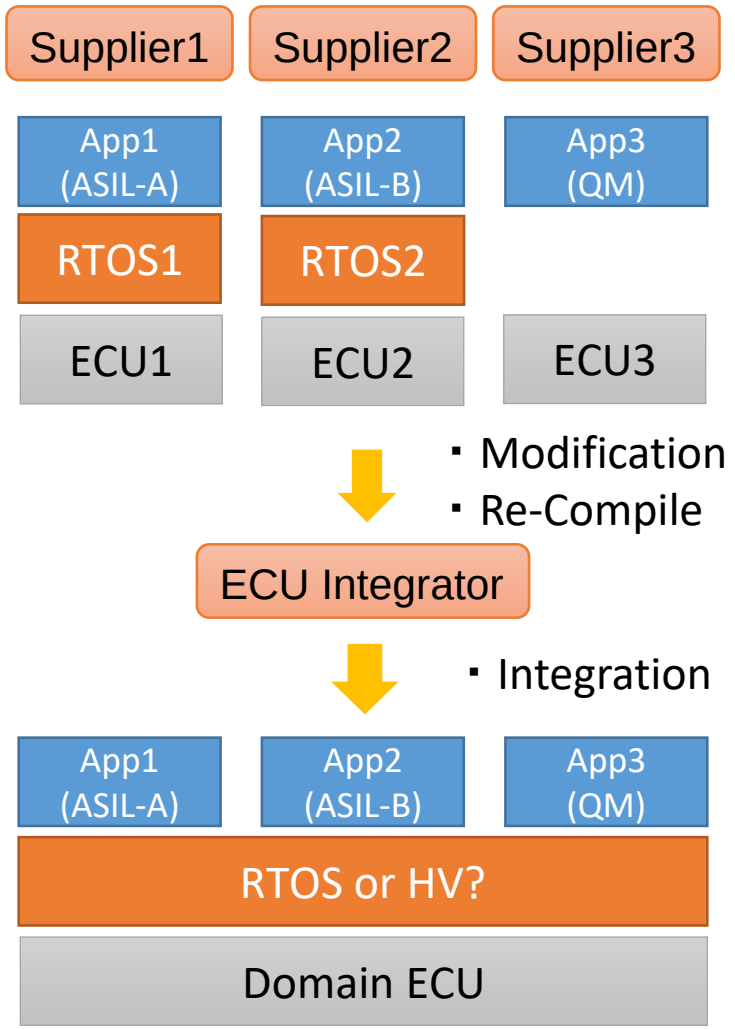
Source: Bosch

- 自動車制御システムにおける課題
  - ECU設置スペースの不足
  - ハードウェアコストの増加
  - ネットワーク帯域の不足

ECUの数を減らすために，単一のECUで複数の車載制御アプリを動作させる**ECU統合**の検討が進んでいる

ECU統合の課題

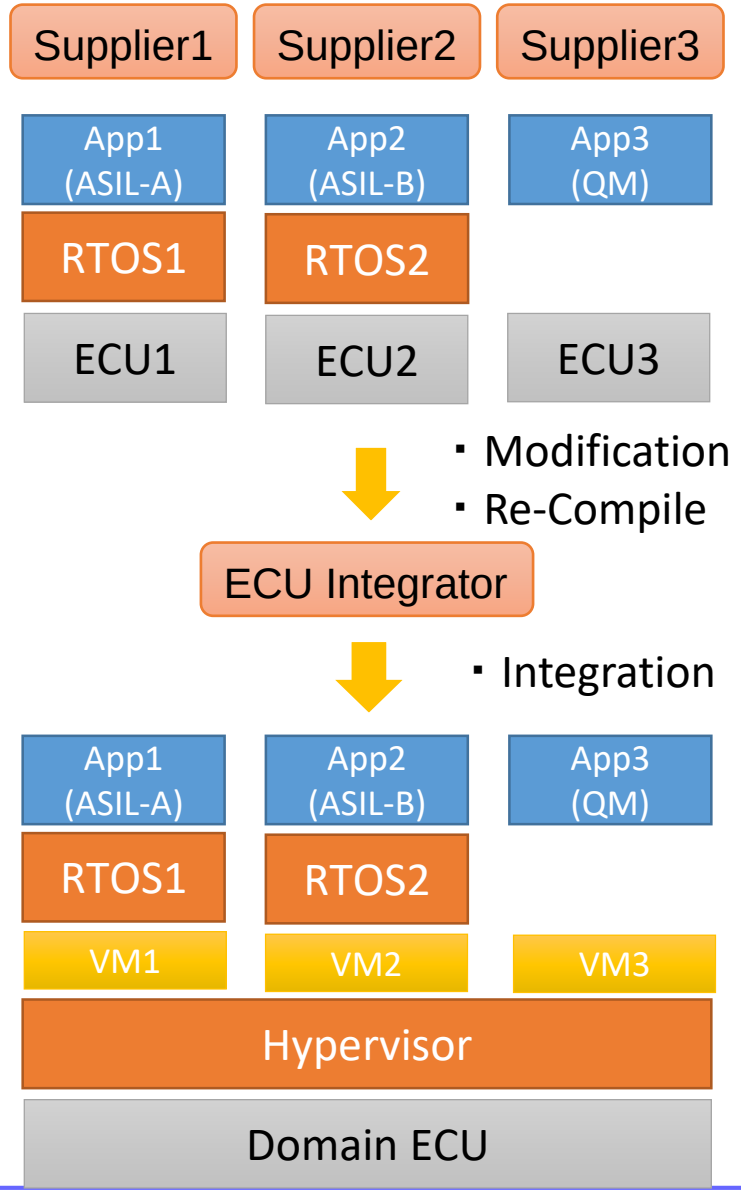
- 統合用ソフトウェアプラットホームの実現
  - パーティショニング
- ハードウェアの共有方法
  - ネットワーク(CAN, Ethernet)
  - データフラッシュ
  - ウオッチドックタイマ
- VM間通信機能
  - ハードウェアの共有方法とも関係
- 設計フローの確立
  - 支援ツール



ECU統合手法の比較

ECU統合のシナリオ

- 車載制御アプリの特徴
  - アプリ毎に別のサプライヤーが開発
  - アプリ毎に安全度水準が異なる
  - サプライヤは全てのコードを持つ
- ECUインテグレータ
  - 統合ECUの開発者
  - OEMまたは特定のサプライヤ
- ECU統合時
  - アプリの設計情報をECUインテグレータに開示したくない
    - ソースコードやタスク構成
  - 軽微なコードの変更は許容される



ECU統合の要件

- 保護要件1（メモリ保護）
  - 各車載制御アプリが使用するメモリ領域（ROM/RAM/周辺回路）がお互い保護されること.
- 保護要件2（時間保護）
  - ある車載制御アプリが，他の車載制御アプリの時間的振る舞いに影響を与えないこと.
- 非機能要件1（実行オーバヘッド）
  - 車載制御アプリの実行オーバヘッドが大きく増加しないこと.
- 非機能要件2（ソースコード変更量）
  - 車載制御アプリのソースコードの変更が少ないこと.
- 非機能要件3（設計情報非開示）
  - サプライヤはECUインテグレータに対して設計情報を開示せずに統合が可能なこと.

ECU統合の手法

- AUTOSAR OS 保護機能による統合：AUTOSAR
  - AUTOSAR OSのメモリ保護・時間保護による統合

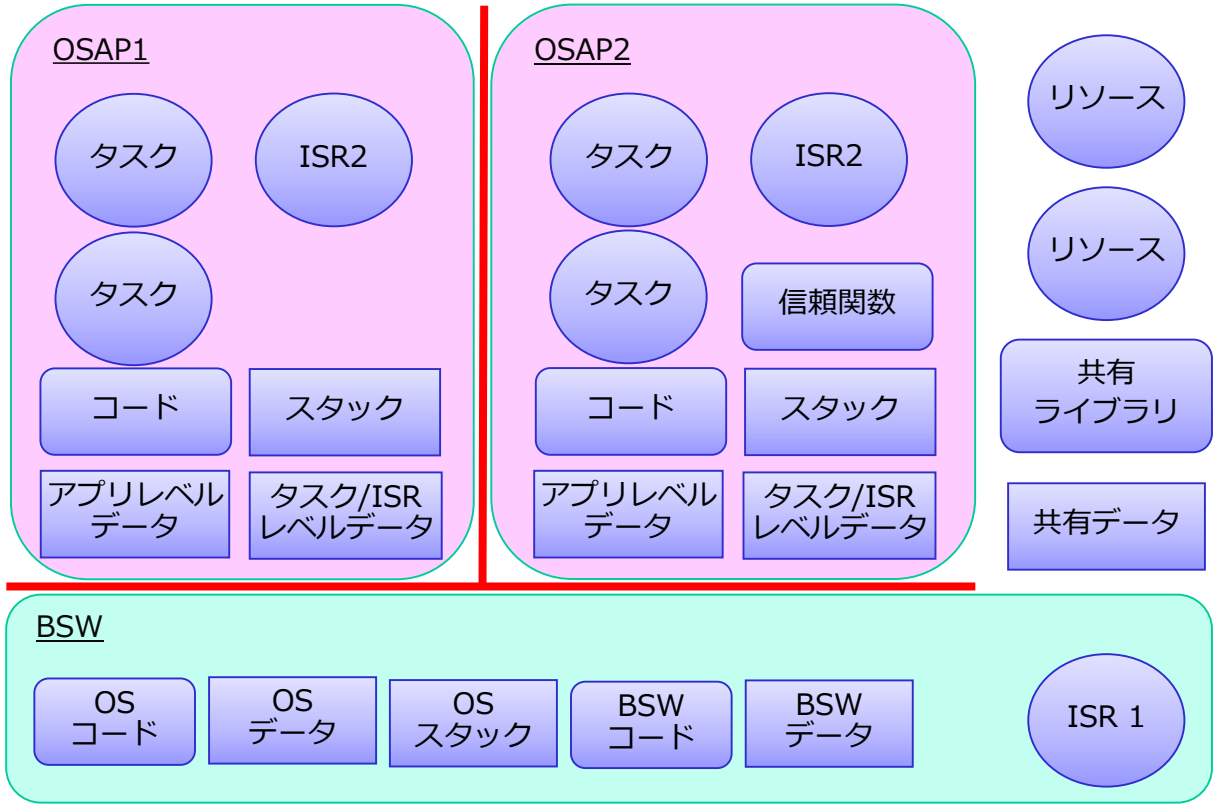
- ATK2-TPによる統合：ATK2-TP
  - AUTOSAR OSを拡張してTDMAスケジューリングを導入

- 制御向けプロセッサのマルチコアによる統合：MCU-MC
  - 各コアで統合前のアプリを動作させる

- 仮想マシンによる統合：HV
  - 仮想マシンにより各VMで統合前のアプリを実行

AUTOSAR：保護機能

- 2種類の保護機能を提供
  - メモリ・アクセス保護
  - タイミング保護



AUTOSAR：メモリ保護

概要

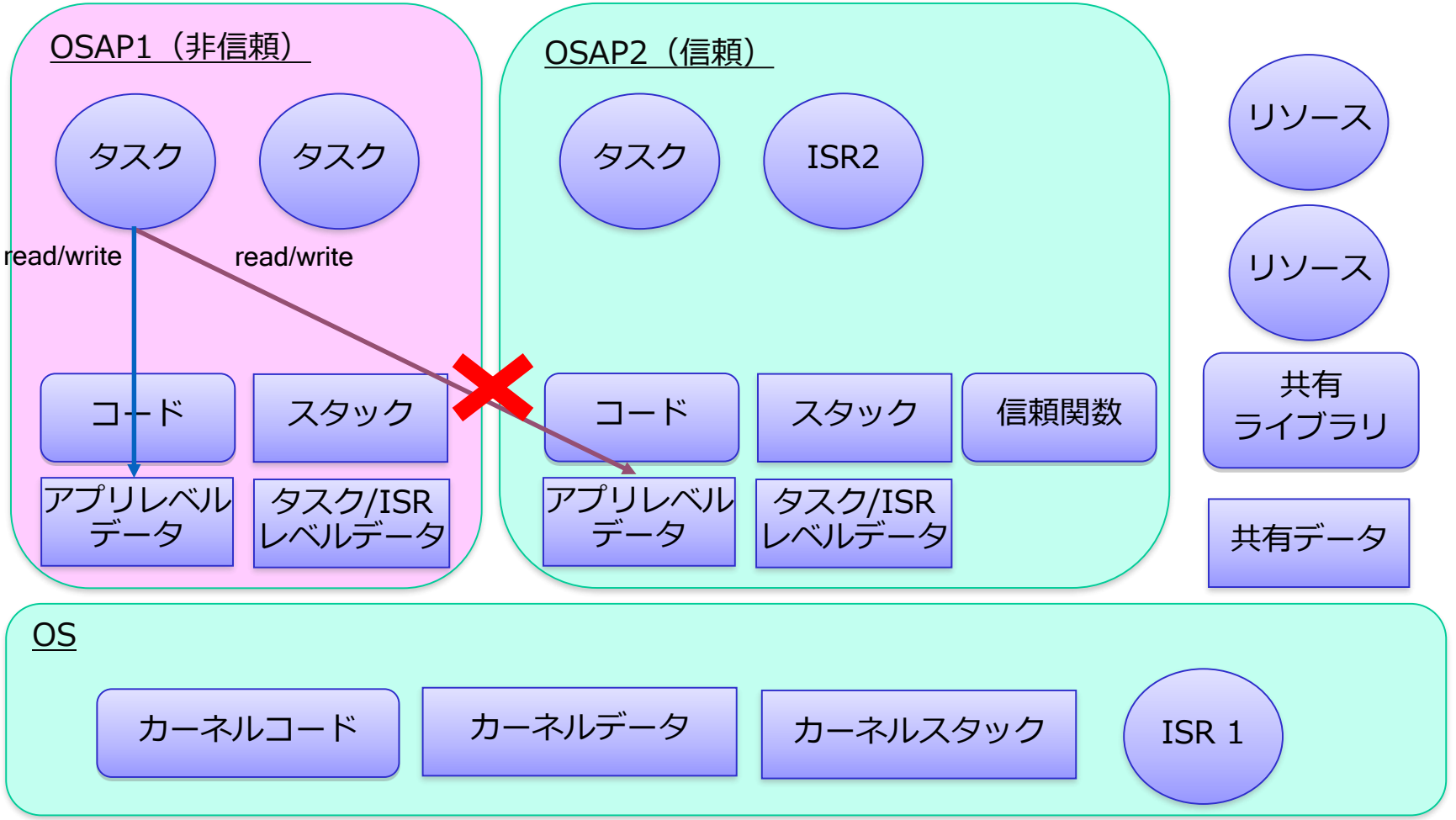
- AUTOSAR OS仕様のスケーラビリティクラス3と4で実装を要求

基本的な考え方

- OS仕様にアプリケーション（OSAP）の概念を導入
  - すべてのシステムオブジェクト（タスク，ISR，リソースなど）は，いずれかのアプリケーションに所属
- アプリを信頼アプリと非信頼アプリに分類
  - 非信頼アプリケーションのアクセスから，スタック領域，データ領域，コード領域，I/O領域を保護
- 信頼関数の導入
- ハードウェアによるサポートを前提
  - プロセッサ動作モード，MMU/MPU
- 保護違反発生時の動作を規定
  - 保護違反が発生した場合には，アプリごとにもつProtectionHook を呼出して，アプリケーションに通知
  - 不正なAPI呼び出しに対してはService Protectionを規定

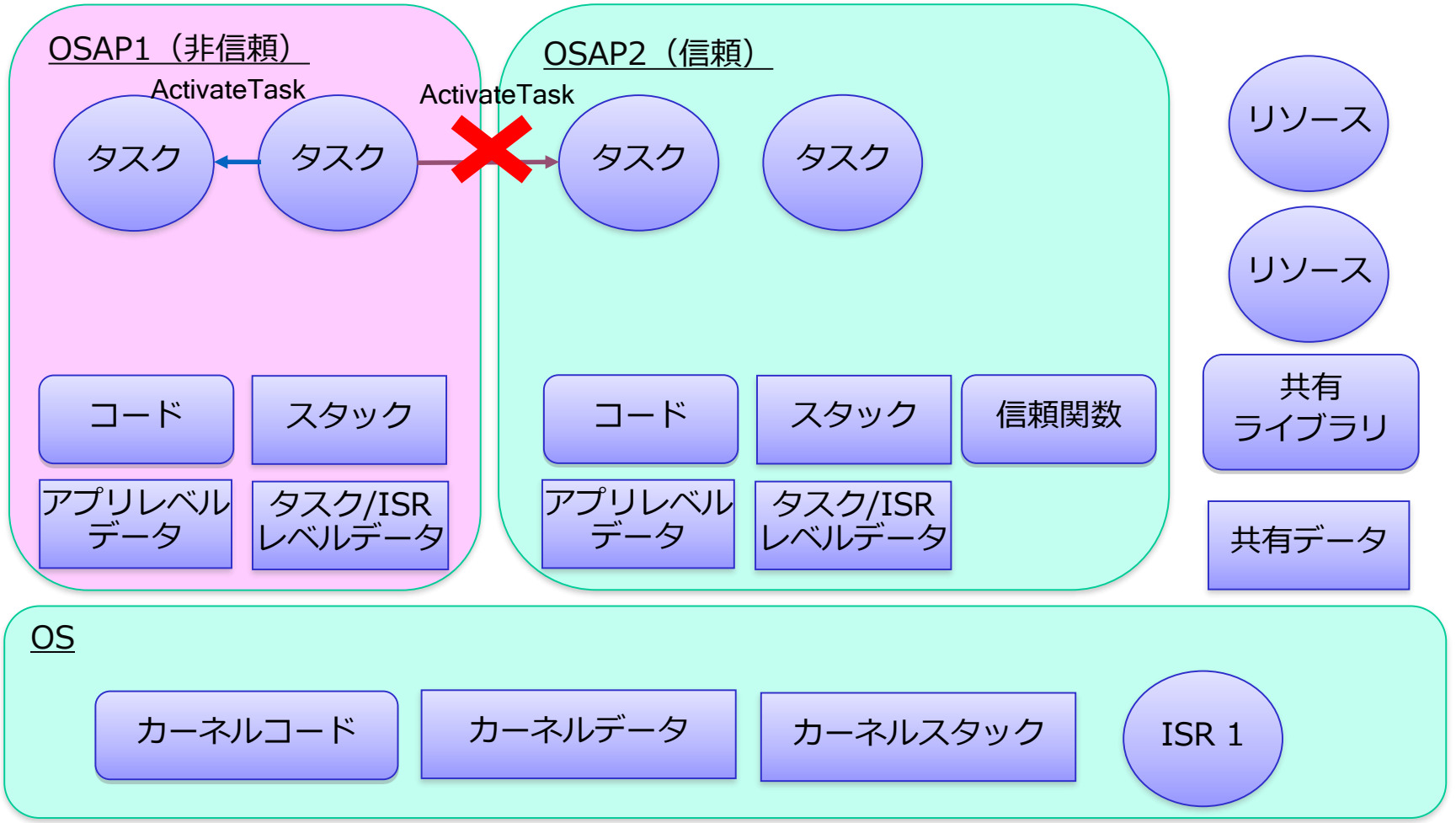
AUTOSAR：メモリ保護

- 非信頼OSAPは他のOSAPやOSのメモリにアクセスできない
  - MPU/MMUにより実現



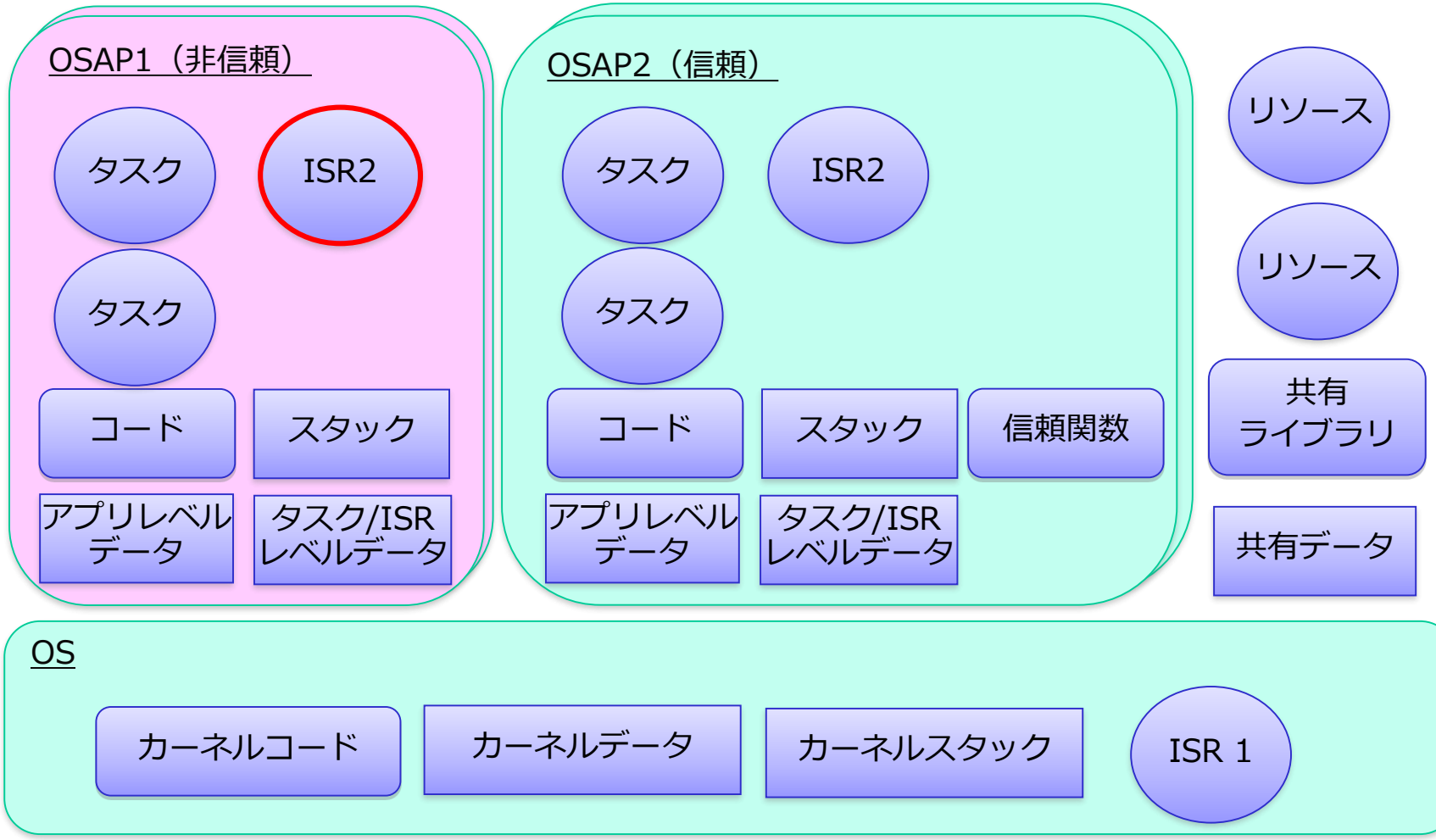
AUTOSAR：アクセス保護

- 非信頼OSAPは他のOSAPのオブジェクトに対してAPIを発行できない
  - OSによりAPI発行元と発行先をチェック



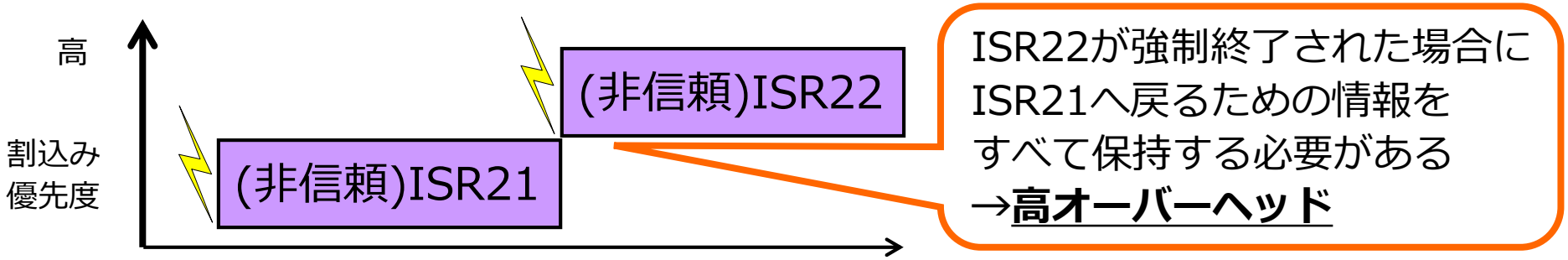
AUTOSAR：非特権ISR2

- 保護が有効なISR2



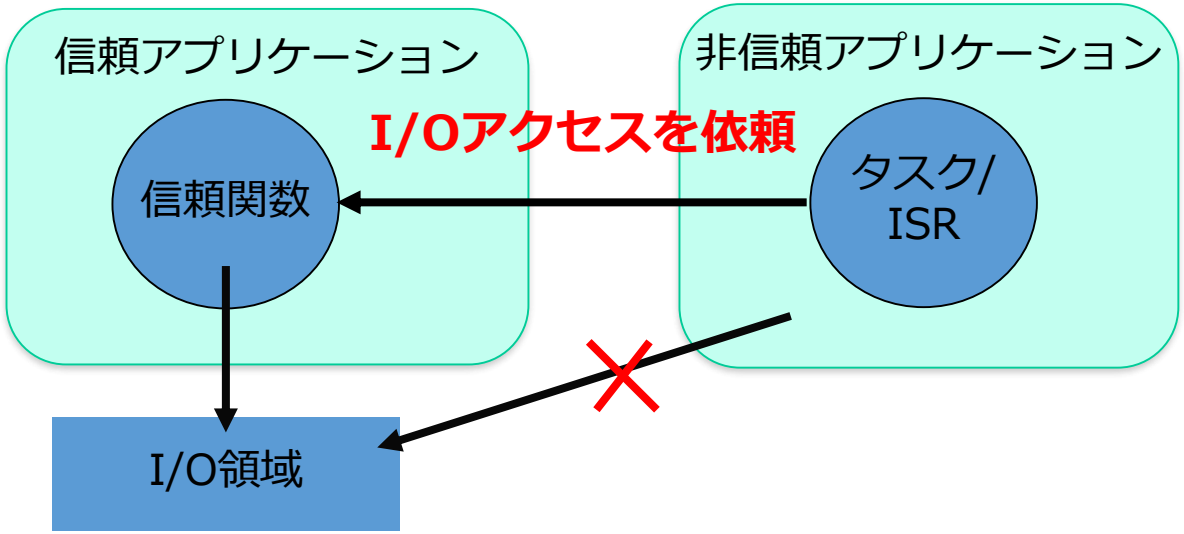
AUTOSAR：非特権ISR2

- 実行オーバヘッドが大きい
  - 非信頼OSAPに所属するISRを実現する場合，割込み発生後，MPUを非特権モードに切り換えメモリ保護属性を設定する必要がある
  - 非信頼ISRからシステムサービスを呼び出す場合，ソフトウェア割込み等で特権に切り換える必要がある
  - ISRが強制終了される場合に備え，多重割込み発生時は，元のISRに関する情報を保持しておく必要がある
  - ISRの強制終了を無効にしても，OSAP全体を強制終了される場合に，同様の問題が発生する



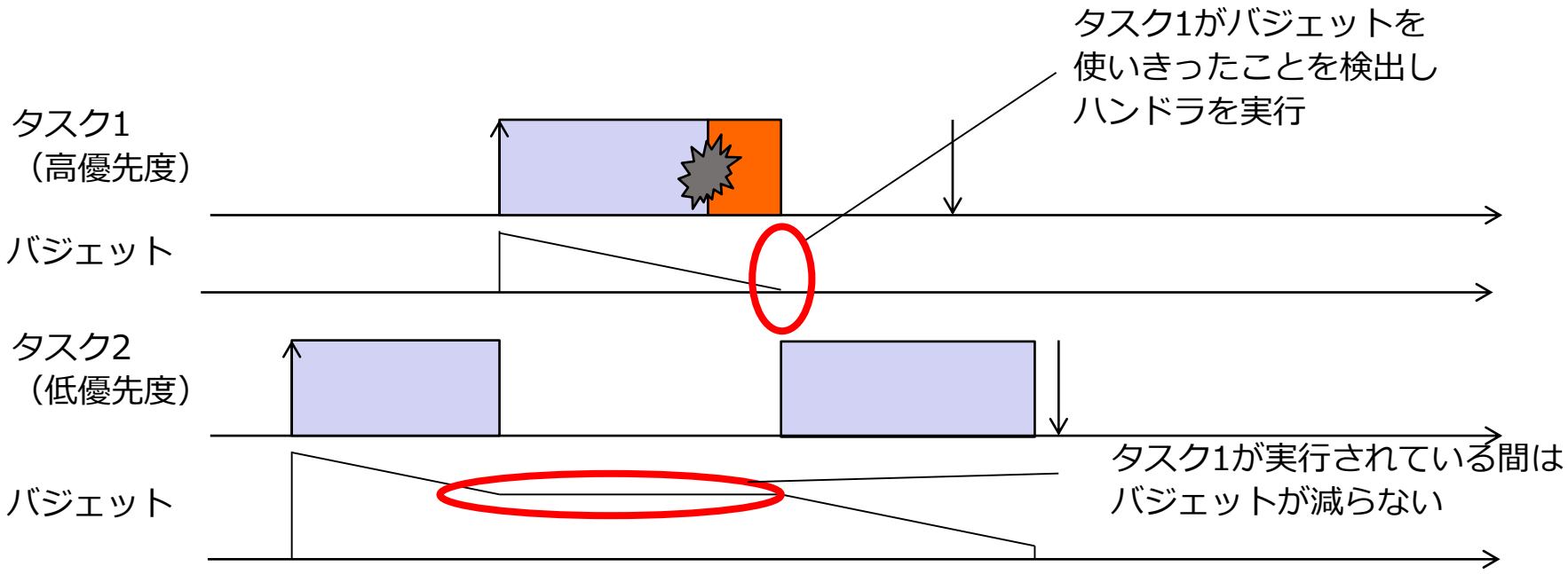
AUTOSAR：信頼関数

- ハードウェアへのアクセスなど，制限された機能を処理するための関数
- 他の信頼／非信頼アプリケーションから呼出し可能
  - 非信頼アプリケーションからの呼出しでは，関数のパラメータ（オブジェクトID，アドレス）をチェック
    - スタックのアドレスは特に危険



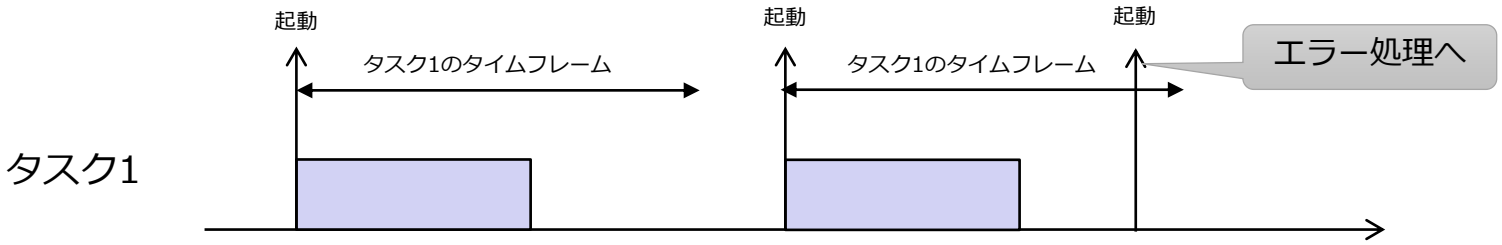
AUTOSAR：タイミング保護

- タスク/ISRの実行時間監視
  - タスクごとに，動作できる最大時間をバジェットとして割当てる
  - 実行時間監視では，対象タスクが実行中にバジェットを使い切ったことを検出する
    - バジェットを使い切ったタスクに根本的な原因がある
  - 検出した後の動作は，アプリケーション開発者または，システムインテグレータが決定する

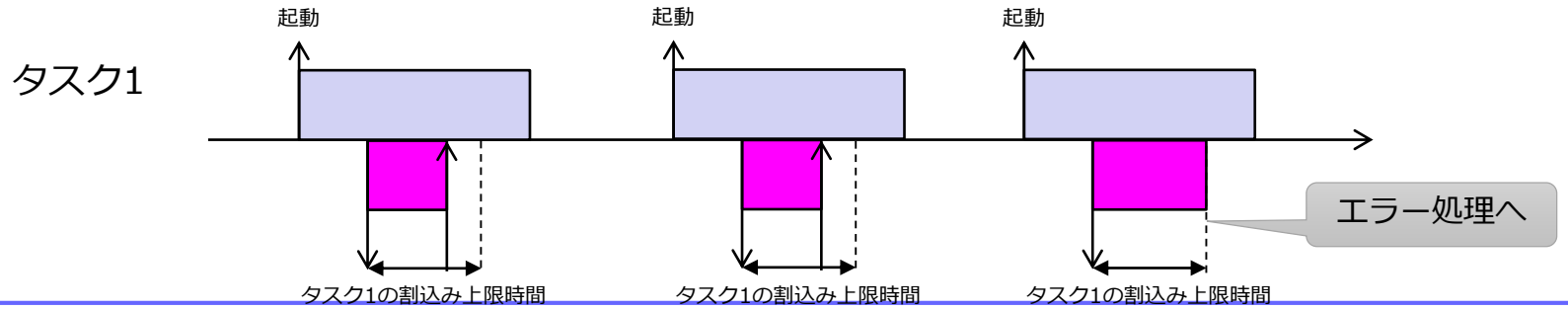


AUTOSAR：タイミング保護

- 到着間隔の監視
  - タスク/ISRの起動間隔が指定して間隔以下になればエラーとする



- ロック時間監視
  - リソースの取得時間や割込み禁止時間が指定された時間以上ならエラーとする



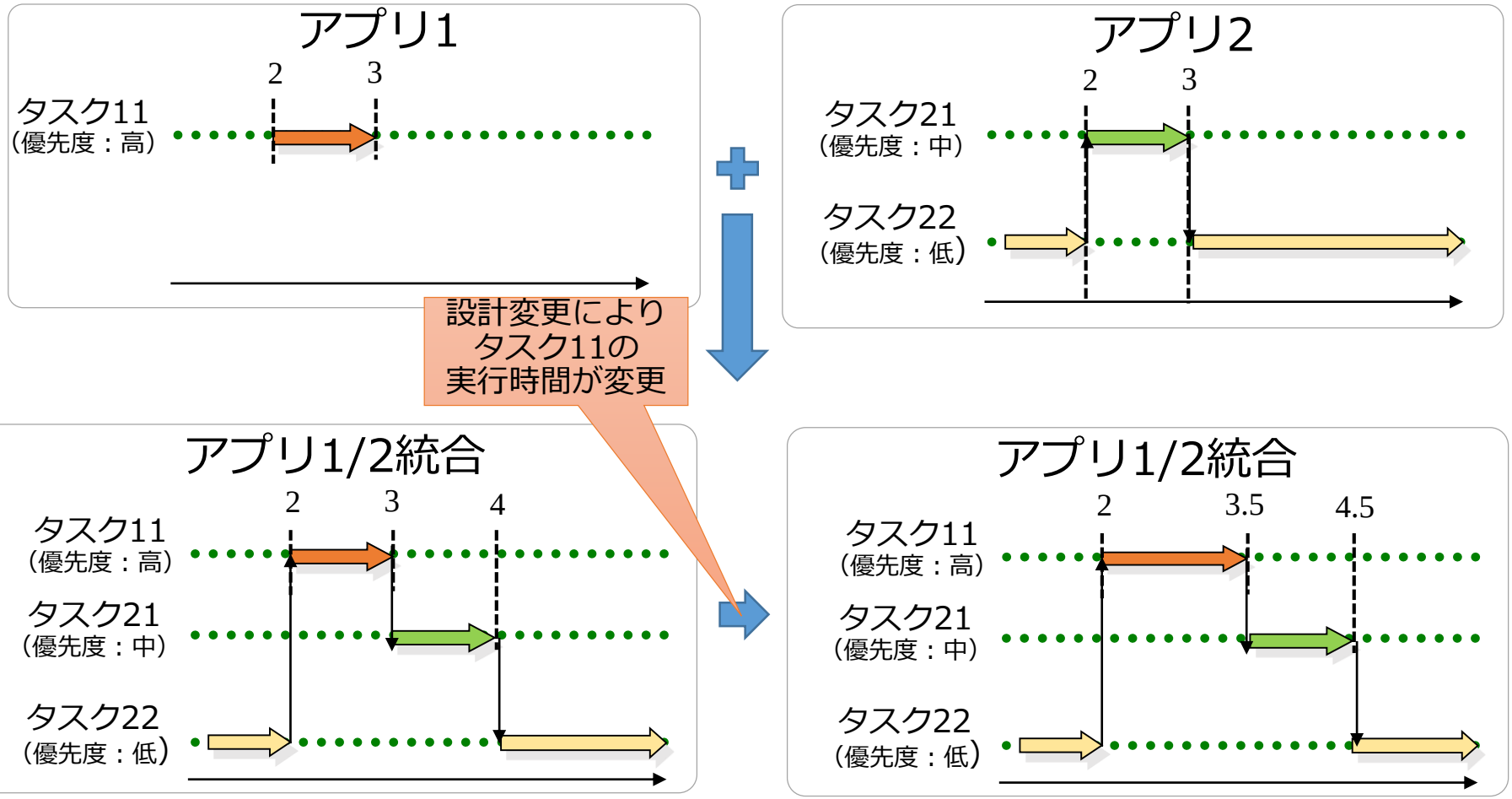
AUTOSAR：ECU統合への適用性

- 保護要件1（メモリ保護）
  - メモリ保護機能により保護可能

- 保護要件2（時間保護）
  - 全アプリのタスクをまとめて優先度ベーススケジューリングするため，パーティショニングとしては用いることができない.
    - タイミング保護はタスクのエラー検出を目的としている.
  - 問題1：不具合の波及問題
    - あるアプリの高優先度タスクが実行を継続し続けると，優先度が低い他のアプリのタスクも動作できない
  - 問題2：設計変更の波及問題
    - あるアプリの処理の実行時間が変わると，優先度が低い他のアプリのタスクの実行タイミングが変化する
  - 問題3：バックグラウンド処理の統合の問題
    - 最低優先度で動作するバックグラウンド処理が各アプリに存在すると，どれかのアプリのバックグランド処理しか動作しない.

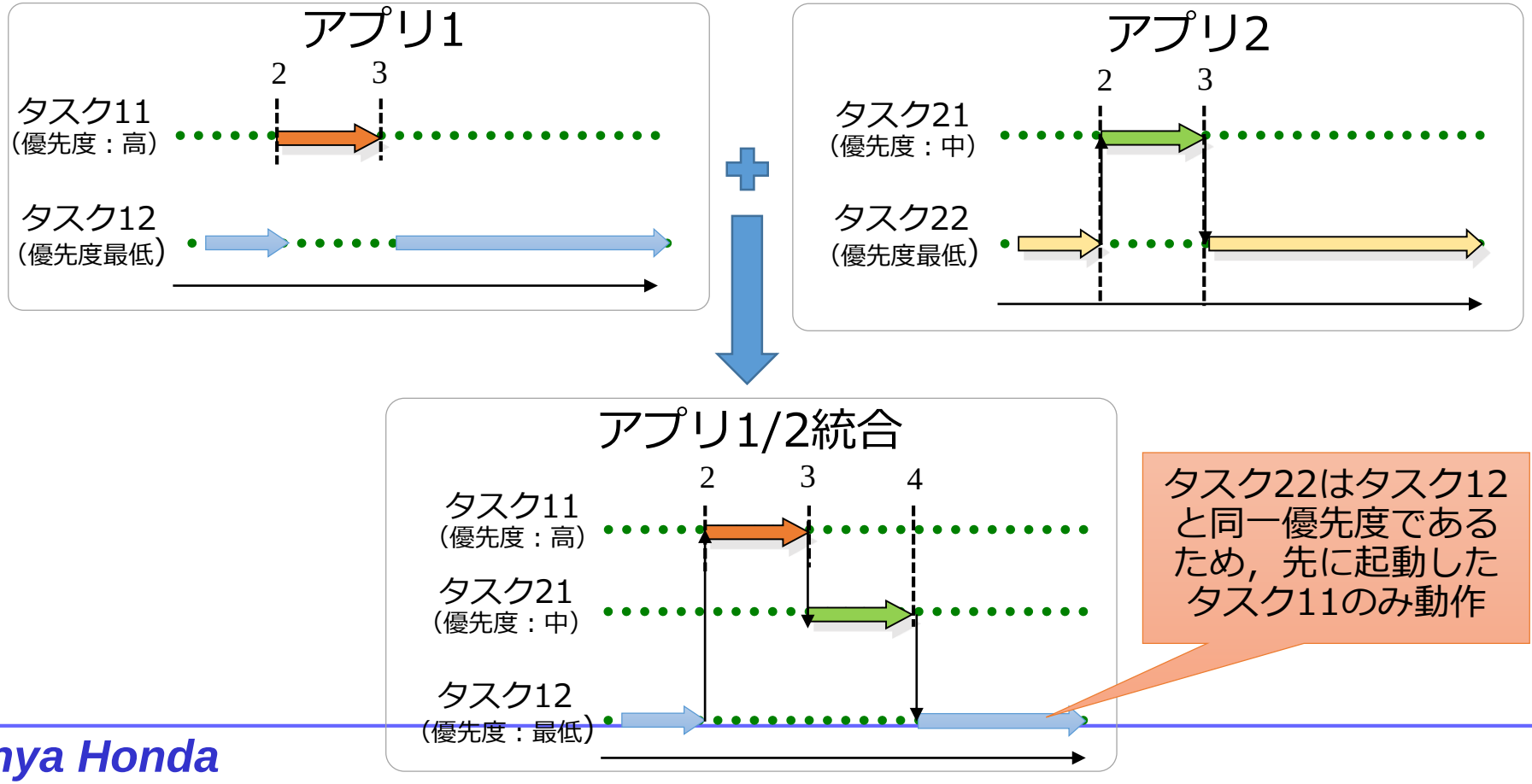
AUTOSAR：ECU統合への適用性

- あるアプリの設計変更が他のアプリに波及する
  - あるアプリの処理の実行時間が変わると，優先度が低い他のアプリのタスクの実行タイミングが変化する



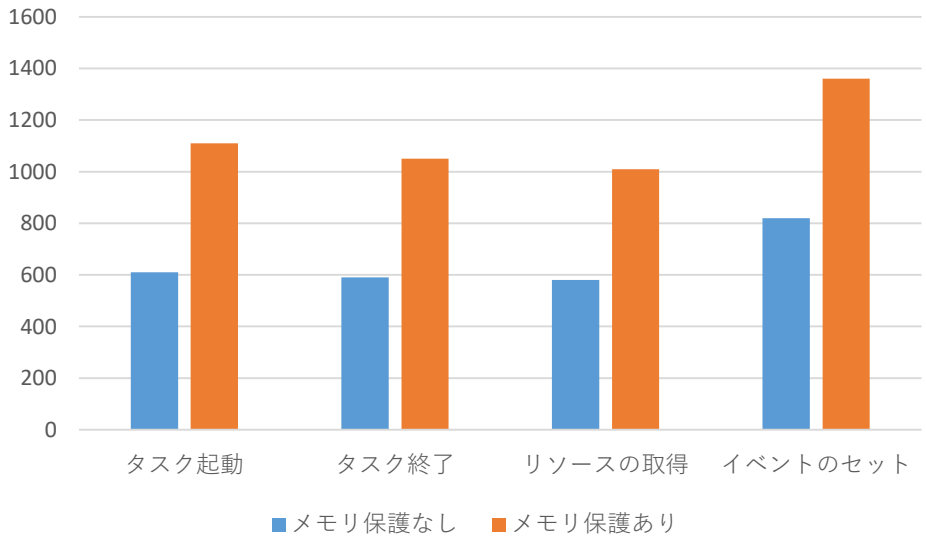
AUTOSAR：ECU統合への適用性

- 問題3の例
  - バックグラウンド処理の統合
    - 最低優先度で動作するバックグラウンド処理が各アプリに存在すると，どれかのアプリのバックグランド処理しか動作しない.



AUTOSAR：ECU統合への適用性

- 非機能要件1（実行オーバヘッド）
  - OSやデバイスドライバの呼び出しオーバヘッドが発生
    - 非特権から特権への切替オーバヘッド
    - 同性能を得るには高速なプロセッサが必要
      → ハードウェアコストの増大
  - 品質確保の問題
    - 保護無しの場合から コード 1.2倍・テストケース 11倍
      →大規模であるため品質確保が困難

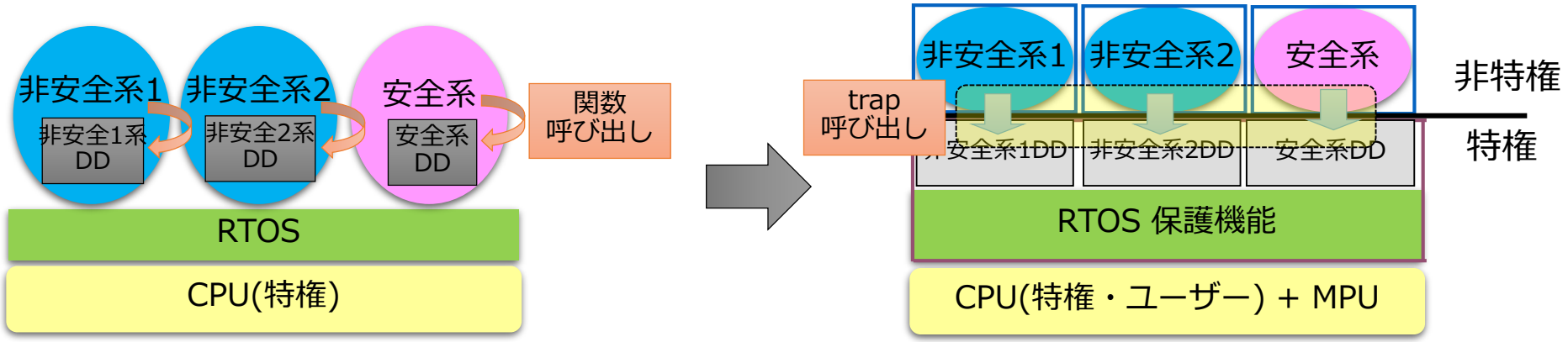


コード量・検証項目数

OS種類	コード量(行数)	検証項目数
メモリ保護無し	10,315	10,982
メモリ保護有り	19,027	112,051

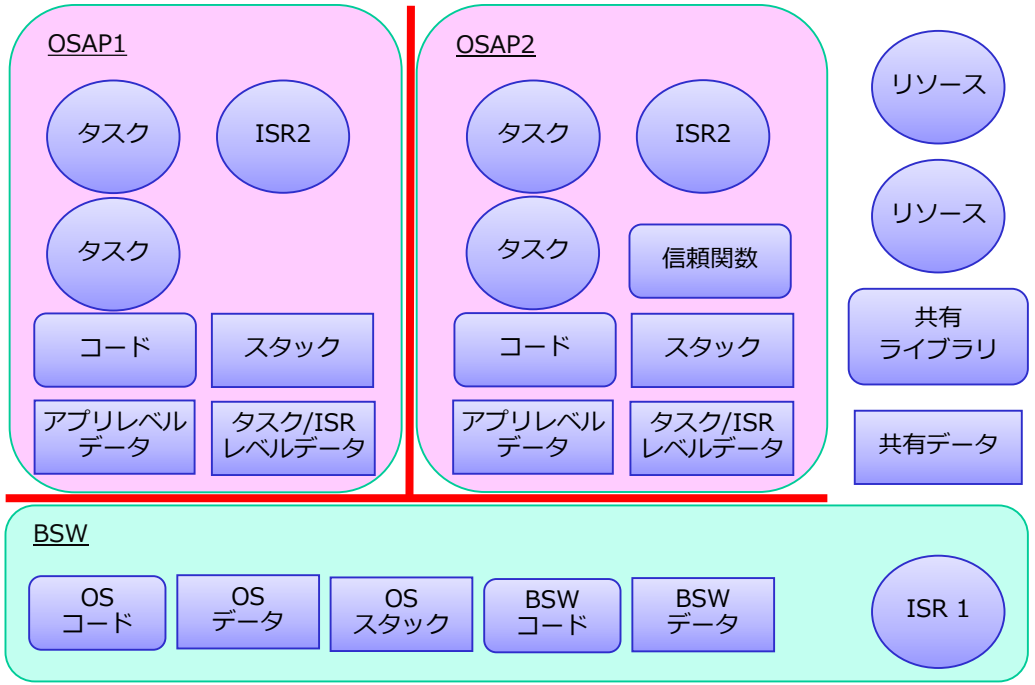
AUTOSAR：ECU統合への適用性

- 非機能要件2（ソースコード変更量）
  - 既存ソフトウェアの変更量の問題
    - デバイスドライバ（DD）はアプリケーションと切り離して特権モードで動作する別モジュールとする必要がある.
  - デバイスドライバの品質確保の問題
    - デバイスドライバは特権モードで動作し保護が有効とならない
    - 非安全系のデバイスドライバであっても安全系と同レベルの品質を確保する必要がある
  - OSレスや使用していたOSベンダが異なる場合



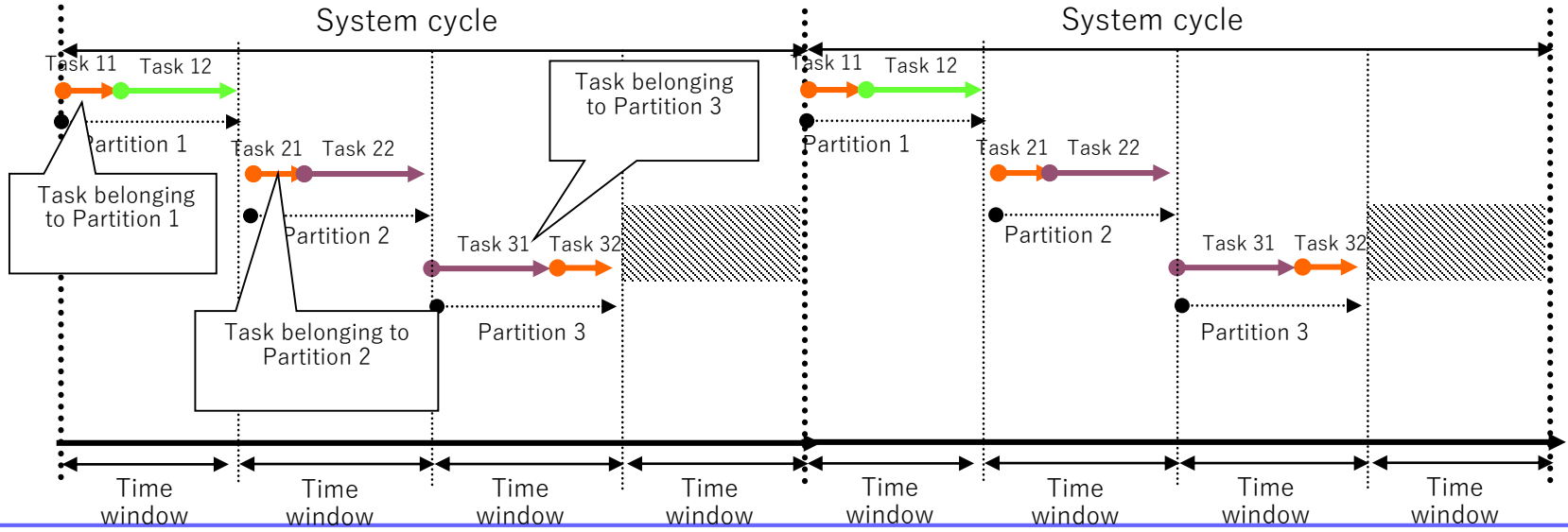
AUTOSAR：ECU統合への適用性

- 非機能要件3（設計情報非開示）
  - 設計情報を開示する必要がある
    - 各サプライヤーはXML形式の設定ファイルをECUインテグレータに提供する必要がある
      - AUTOSAR OS ではOSAPの内構成（タスクやカーネルオブジェクト）は，XML形式の設定ファイルにより記載して，最終的なリンク時に使用される



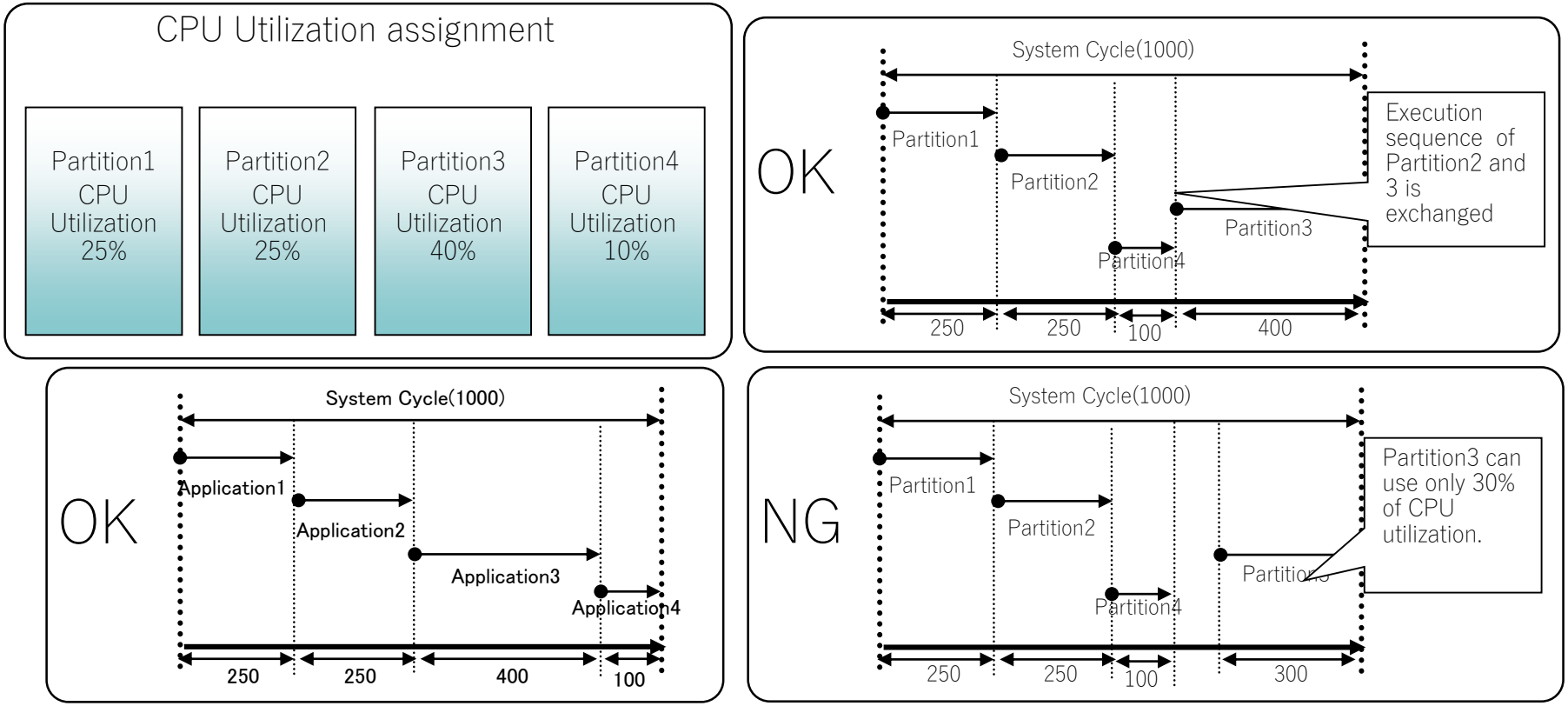
ATK2-TP（Timing Protection）

- AUTOSAR OSに時間保護機能を追加した実装(AUTOSAR準拠ではない)
- 時間保護機能
  - システム周期を定め，各パーティションにシステム周期内を複数のタイムウィンドウに分割し，そのタイムウィンドウをパーティションに割り付ける．OSはこのタイムウィンドウの設定と割り付けに従って各パーティションを実行．
  - TDMAスケジューリングと呼ぶ場合が多い．
  - 厳密さの段階がある
    - "CPU利用率保護","実行順序保護","実行タイミング保護“
  - 航空機向けのOS仕様であるARIN653で採用されている



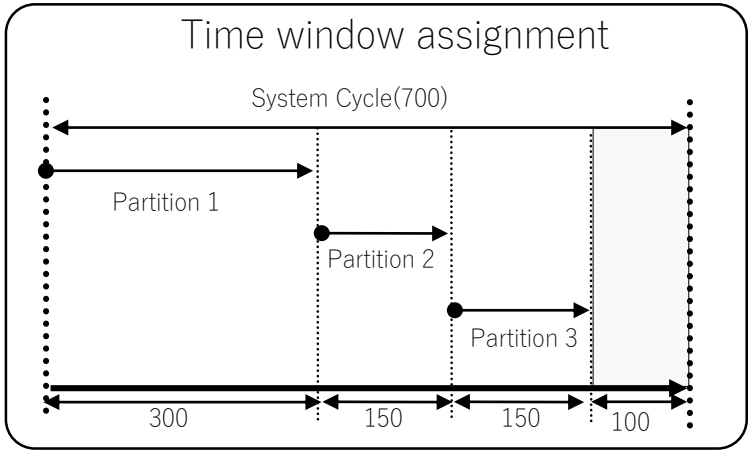
ATK2-TP：レベル1：CPU利用率保護

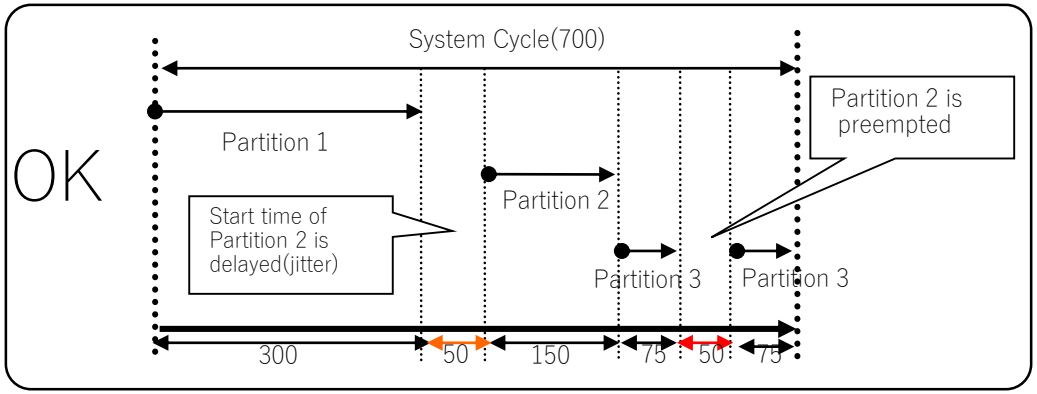
- 設計時に各パーティションにCPU利用率を割り当て，割り当てたCPU利用率を満足するようにパーティションを実行すること．CPU利用率が守られていれば，パーティションの実行順序の入れ替えや実行の中断は許容する．

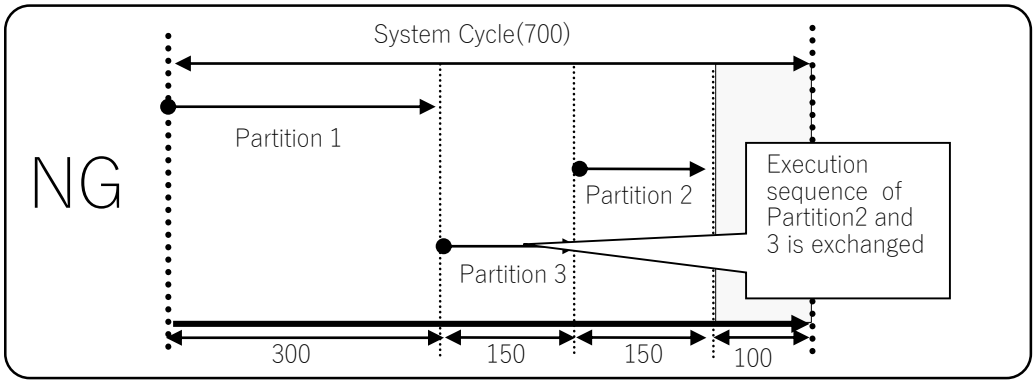


ATK2-TP：レベル2：実行順序保護

- 設計時に設定したパーティションの実行順序を実行時に守ること.
- 実行順序はシステム周期と呼ぶ任意の一定時間内の時間を各パーティションに割り付ける.
- 実行順序が守られていれば，開始時間の遅れ（ジッター）や，実行の中断は許容するが，許容値をユーザーに明示させること.

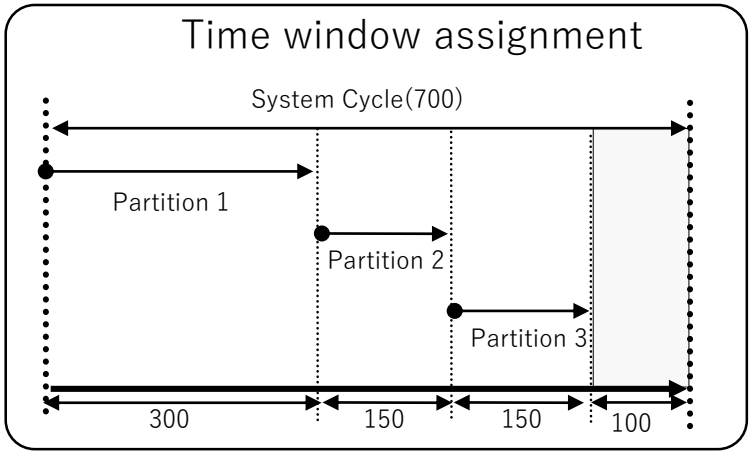


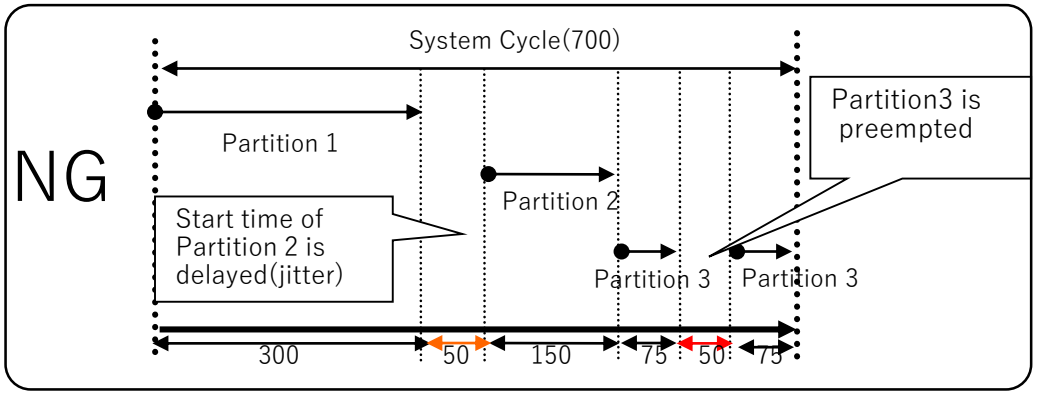




ATK2-TP：レベル3：実行タイミング保護

- 設計時に設定したパーティションの実行順序と実行タイミングに沿って実行
- 開始時間の遅れ（ジッター）や，実行の中断も許容しない.
- ARINC653はこの方式





ATK2-TP：割込みのサポート

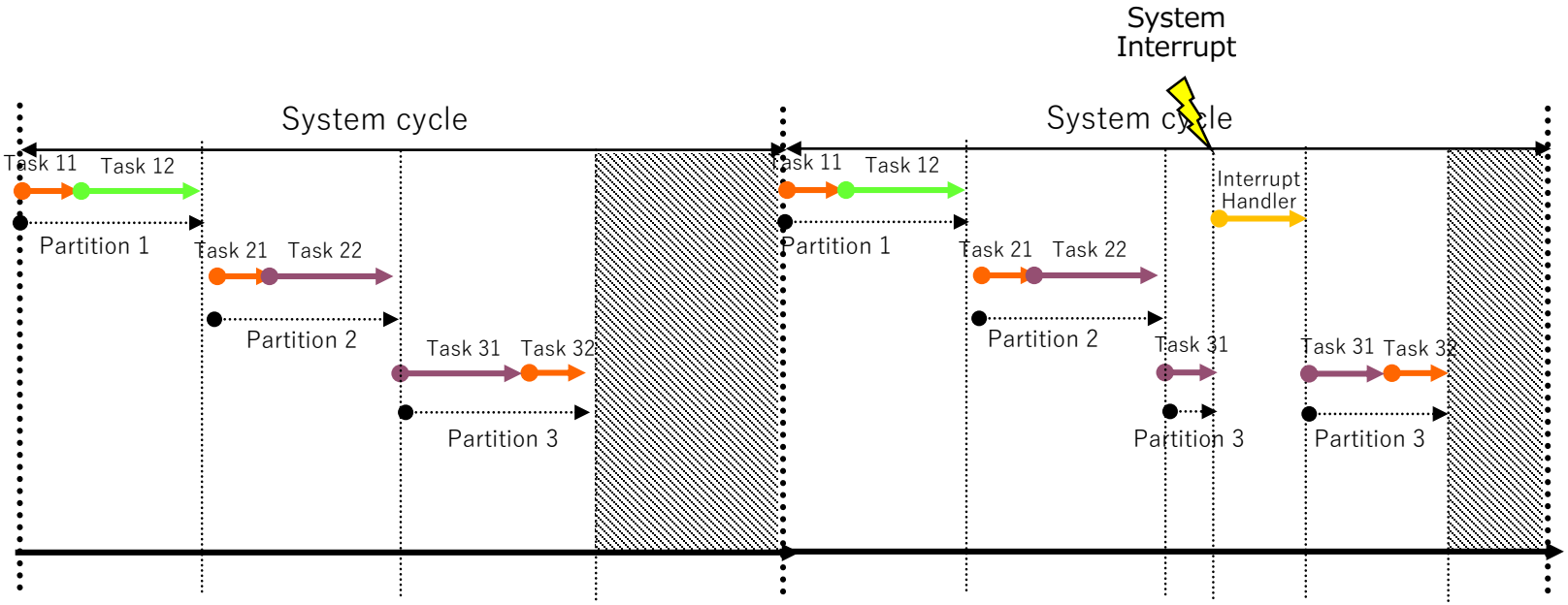
- ARINC653からの拡張
  - 航空機制御では割込みは使用しないが，車載制御システムは割込みは必須であるため拡張

- システム割込み
  - 特定のパーティションに属さず，タイムウィンドウによらずに受け付けられる割込みをシステム割込みと呼ぶ
  - システム割込みは，特権モードで実行される
    - つまり，システム割込みは（基本的には）保護の対象ではなく，システム内の最高信頼度で実装しなければならない

- システム割込みとタイムウィンドウ
  - あるタイムウィンドウの途中でシステム割込みが処理された場合，そのタイムウィンドウの終了時刻は，システム割込みの処理時間分，遅くなる
    - タイムウィンドウの正味の長さは変わらない
  - タイムウィンドウの終了時刻が遅くなった分，後続のタイムウィンドウの開始時刻および終了時刻も遅くなる

時間パーティショニングでの割込みのサポート

- システム割込みとアイドルウィンドウ
  - システム周期の最後に置かれたアイドルウィンドウの長さは，システム周期内で処理されるシステム割込みの合計処理時間以上でなければならない
  - あるシステム周期内で処理されるシステム割込みが，次のシステム周期の開始時刻に影響を及ぼすことはない
- 時間保護のレベル
  - レベル2 実行順序保護となる.



ATK2-TP：ECU統合への適用性

- 保護要件1（メモリ保護）
  - メモリ保護機能により実現（AUTOSARと同様）

- 保護要件2（時間保護）
  - TDMAスケジューリングにより解決
  - 保護レベルは使用するシステムの要件に合わせて選択する必要あり

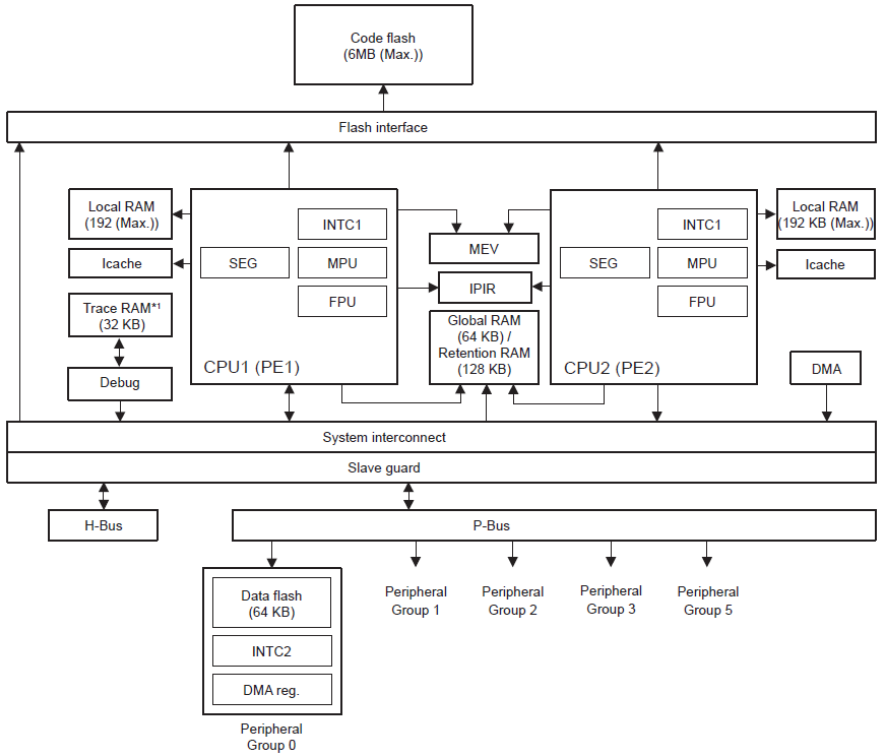
- 非機能要件1（実行オーバヘッド）
  - メモリ保護を行うため，API実行時間が増加する（AUTOSARと同様）

- 非機能要件2（ソースコード変更量）
  - メモリ保護を行うため，ドライバ呼び出しの書き換えが発生（AUTOSARと同様）

- 非機能要件3（設計情報非開示）
  - コンフィギュレーション情報の開示が必要（AUTOSARと同様）

パーティショニングの実現：MCU-MC

- 制御システム向けのマルチコアマイコンを用いて，コア毎に既存システムを実行する．
- 制御システム向けのマルチコアマイコン
  - コア間の性能的な干渉要因が少ないマルチコア（AMPタイプ）
    - コア間でキャッシュを共有しない
    - コア毎にローカルメモリを持つ
  - 各バスマスタ毎に周辺回路やメモリの保護機能を持つ（バスガード）
    - 名称はマイコンベンダ毎に異なる
    - 多くの仕様では起動時にアクセス権を設定した後，設定をロックする．
    - もしくはコア0のみ設定を変更可能

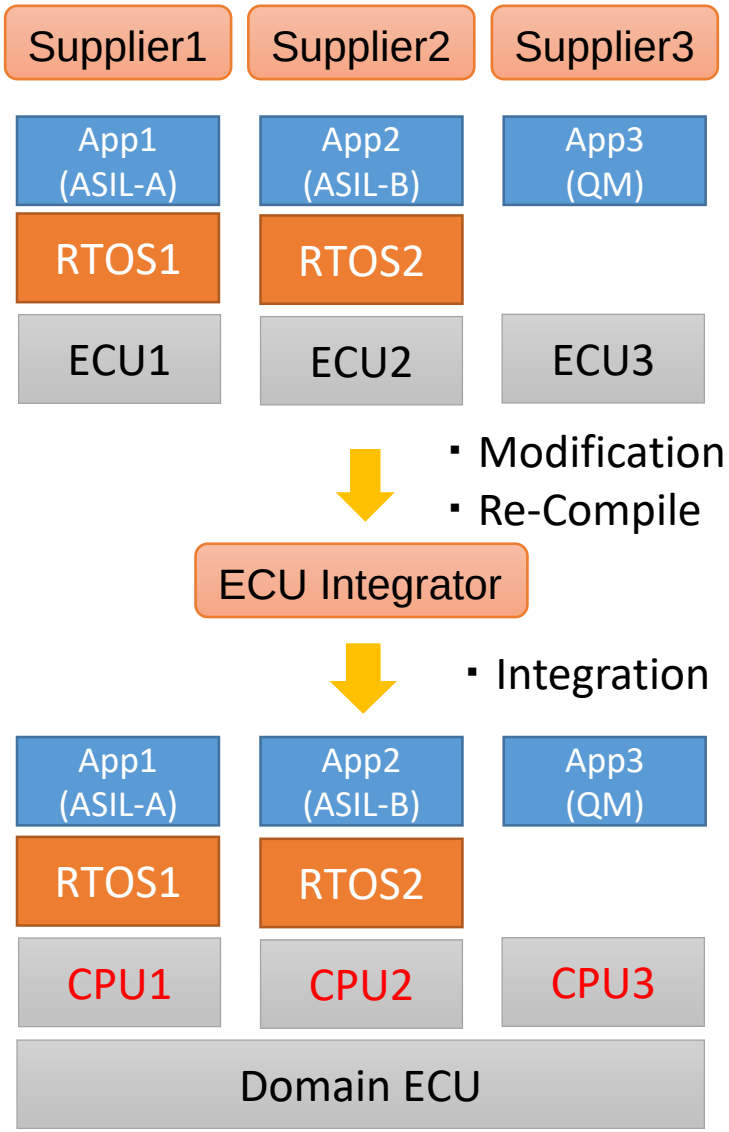


出典：IAR Webサイト

パーティショニングの実現：MCU-MC

- コア毎に既存システムを実行する.
  - 各コアで使用するメモリや周辺回路以外は，バスガードにより保護
  - 各コアのバイナリは独立.
  - クロック等の初期化は特定のコアで実施

- オーソドックスな統合方法
  - Linux + RTOSでも使用されている方法
  - Hypervisorは必要ない
  - 車載システムに対する適用も論文発表されている
    - BoschによるエンジンECUとビークルコントロールの統合[1]
  - Hypervisorと組み合わせているケースもある
    - BMWとETASによる複数のAUTOSAR環境の実行[2]



[1]Arun Kumar Sundar Rajan, Armin Feucht, Lothar Gamer, Idriz Smaili, Nirmala Devi M., Hypervisor for consolidating real-time automotive control units: Its procedure, implications and hidden pitfalls, Journal of Systems Architecture, Volume 82, 2018, Pages 37-48, ISSN 1383-7621.

[2]Reinhardt, Dominik, and Gary Morgan. "An embedded hypervisor for safety-relevant automotive E/E-systems." Proceedings of the 9th IEEE International Symposium on Industrial Embedded Systems (SIES 2014), IEEE, 2014.

MCU-MC：ECU統合への適用性

- 保護要件1（メモリ保護）
  - バスガードにより保護

- 保護要件2（時間保護）
  - 統合前のアプリ毎にコアを割り付けるため保護可能
    - コア間の性能的な干渉が少ない制御システム向けのマルチコアマイコンが前提

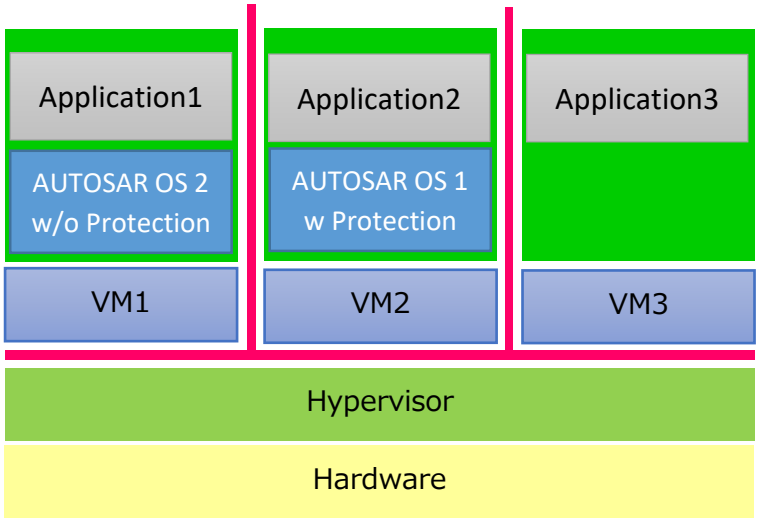
- 非機能要件1（実行オーバヘッド）
  - コア間の干渉以外の実行オーバヘッドは発生しない

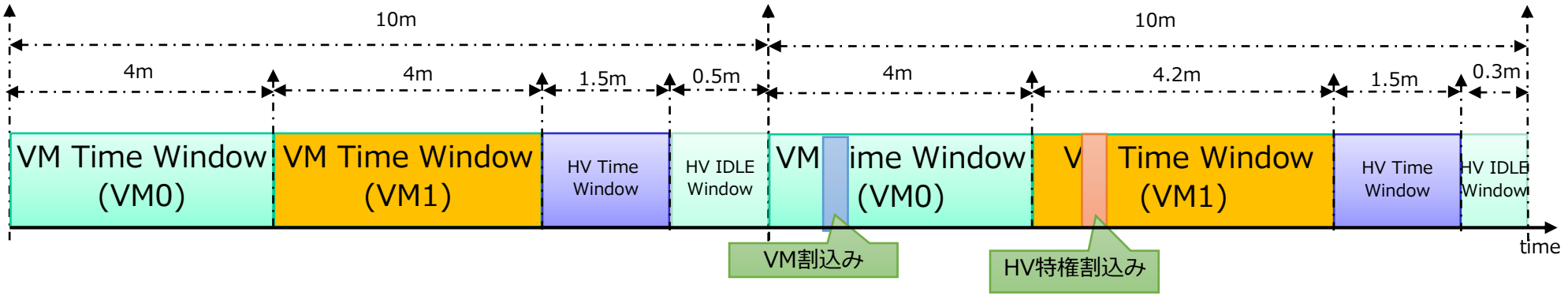
- 非機能要件2（ソースコード変更量）
  - 占有ハードウェアのコードは変更する必要なし
    - 各ソフトウェアは統合前と同じ特権モードで動作
  - 共有ハードウェアが課題（他の手法も同様）

- 非機能要件3（設計情報非開示）
  - サプライヤーはバイナリのみをECUインテグレータに渡せばよい

パーティショニングの実現：HV

- HVの一般論として以下の前提で比較する
  - スケジューリングはTDMA
    - ATK2-TPと同等
  - マルチコアのサポート
  - バスガートは使用可能
  - ハードウェア仮想化支援機能を使用





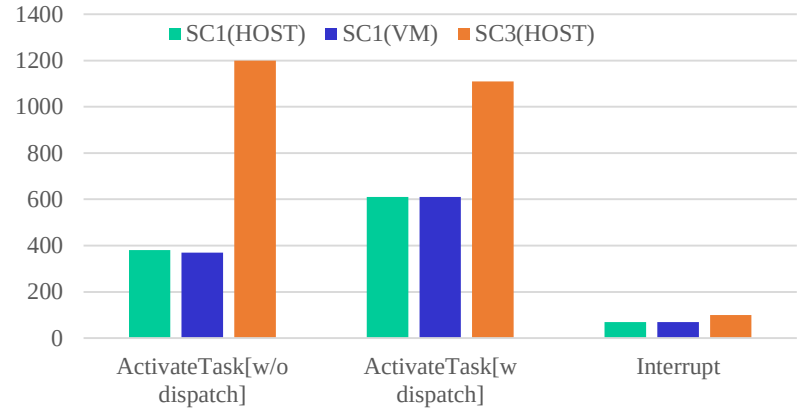
HV：ECU統合への適用性

- 保護要件1（メモリ保護）
  - メモリ保護機能により実現可能
  - 周辺回路についてはスレーブガードにより保護
- 保護要件2（時間保護）
  - TDMAによる時間保護
- 非機能要件1（実行オーバヘッド）
  - VM内では保護なしOSが使用出来るため実行オーバヘッドは増加しない
  - VM切り替えのオーバヘッドは発生する
    - 仮想化支援によりタスク切り替えと同等レベルとなっている（後述）
- 非機能要件2（ソースコード変更量）
  - 占有ハードウェアのコードは変更する必要なし
  - 共有ハードウェアが課題（他の手法も同様）
- 非機能要件3（設計情報非開示）
  - サプライヤーはバイナリのみをECUインテグレータに渡せばよい

HV：ECU統合への適用性

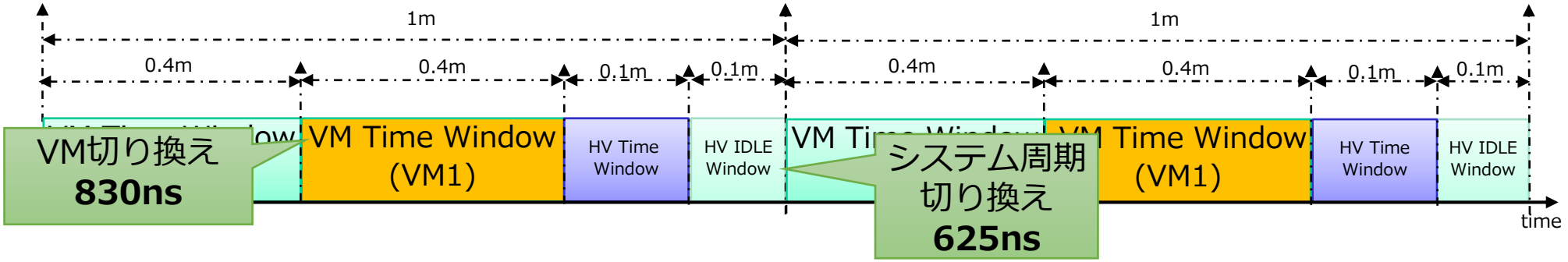
- HVの使用による実行オーバヘッドの見積もり
  - システム周期 1m, VM x 2
  - 1システム周期での実行オーバヘッド
    - 830ns x 3 + 625ns = 3,115ns
    - 3,115ns / 1ms * 100 = 0.3115%
  - SC3のタスク切り替えの増加分：500ns
    - 7回切り替えるとHVの実行オーバヘッダを上回る

単位：nsec



- SC3のタスク切り替えより高速な理由
  - VM切替　　：HVの割込みが契機でエラーチェックが少ない
  - タスク切替　：ユーザーからのAPI呼び出しが契機であるためエラーチェックが多い

RTOSのタスク切り替え時間



パーティショニングの実現：比較

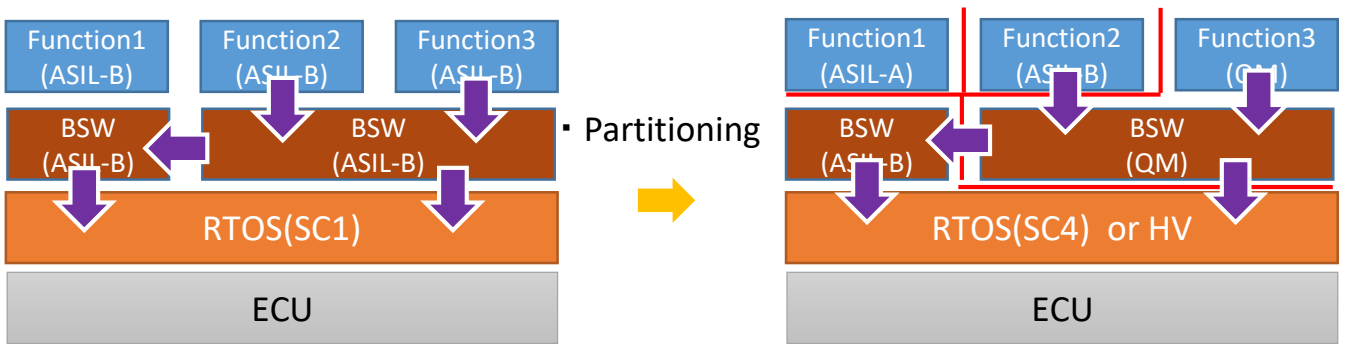
	AUTOAR	ATK2-TP	HV	MCU-MC
メモリ保護	○	○	○	○
時間保護	×	○	○	○
実行オーバヘッド	×	×	○	○
ソースコード変更量	×	×	△	△
ECU統合時の設計情報非開示	×	×	○	○
メモリ使用量	○	○	×*1	×*1
ハードウェアの汎用性	○	○	×*2	△*3
デバイス共有	○	△*4	×*4*5	×*5
検証容易性	×	△	△-○*6	○*7
拡張性・柔軟性	○	○	○	×*8

- *1 アプリ毎にRTOSやBSWが必要
- *2 仮想化支援機能が必要
- *3 統合対象のシステム数分のコアが必要
- *4 TDMAスケジューリングでのデバイス共有手法が課題
- *5 MCALレベルでの共有が必要
- *6 VM実行中の他VMやHVの割込みを許可すると検証工数は増える
- *7 コア間の干渉が少ない制御用マイコンの場合
- *8 アプリ数分のコアが必要

機能パーティショニングの実現手法検討

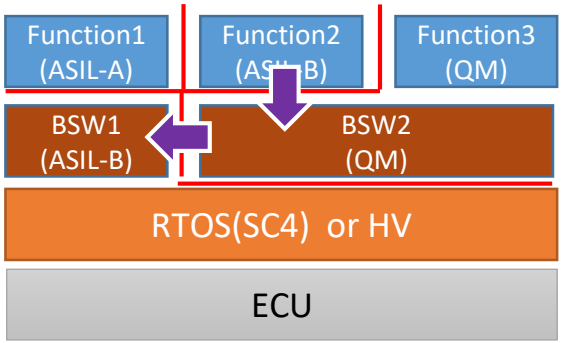
機能パーティショニング

- ECU統合との違い
  - パーティショニングした機能間の呼び出しを実現する必要がある
  - ECU統合と異なり，全体を単一のサプライヤーが開発することになるため，FFIは低くても検証は可能と考えられる.
  - バイナリの分離やスケジューリングの分離が必要かは分離したい機能の性質に依存



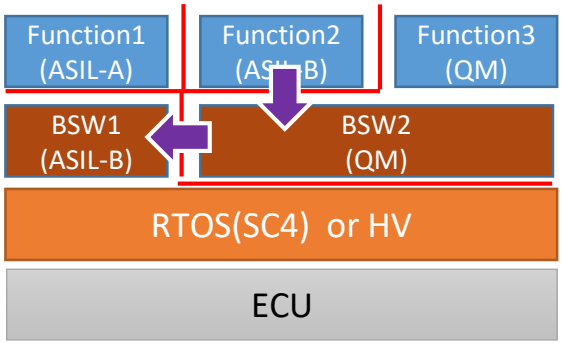
機能パーティショニング：AUTOSARでの実現

- AUTOSARの保護機構を使うのが一般的な解決方法
- メリット
  - 汎用的なマイコンで実現可能
  - AUTOSAR準拠
- デメリット
  - ソフトウェアの書き換えが必要
  - 呼び出しのオーバヘッドが発生
    - 下図の例
      - ✓Function2(ユ) → OS(特) → BSW2(ユ) → OS(特) → BSW1(ユ)



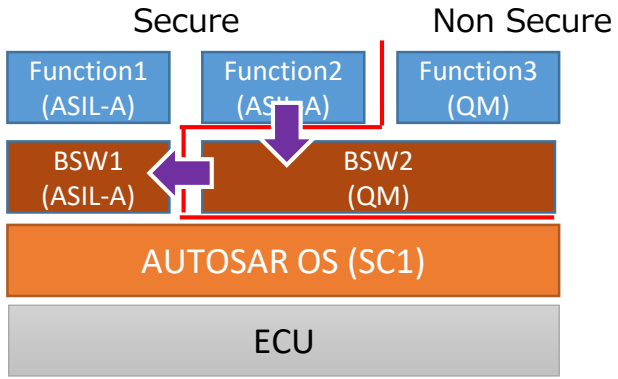
機能パーティショニング：その他の手法

- ATK2-TP/HV/MC
- 呼び出しのオーバヘッドが大きい
  - HVの例
    - Function2(VM1) → HV → BSW2(VM2) → HV → BSW1(VM3)
  - MCU-MCの例
    - Function2(CORE0) → コア間通信 →BSW2(CORE1) → コア間通信 → BSW1(CORE3)
- スケジューリングの問題
  - TDMAスケジューリングとどう組み合わせるか
    - 呼び出し元の時間で実行
    - 呼び出された機能の時間で実行



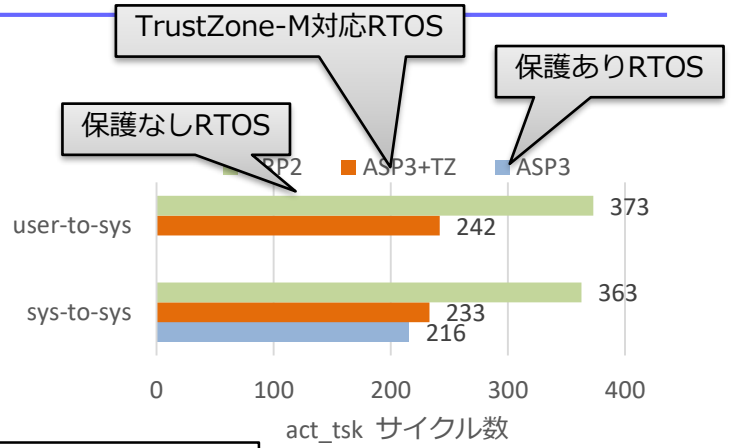
機能パーティショニング：ASILとQMへの分割

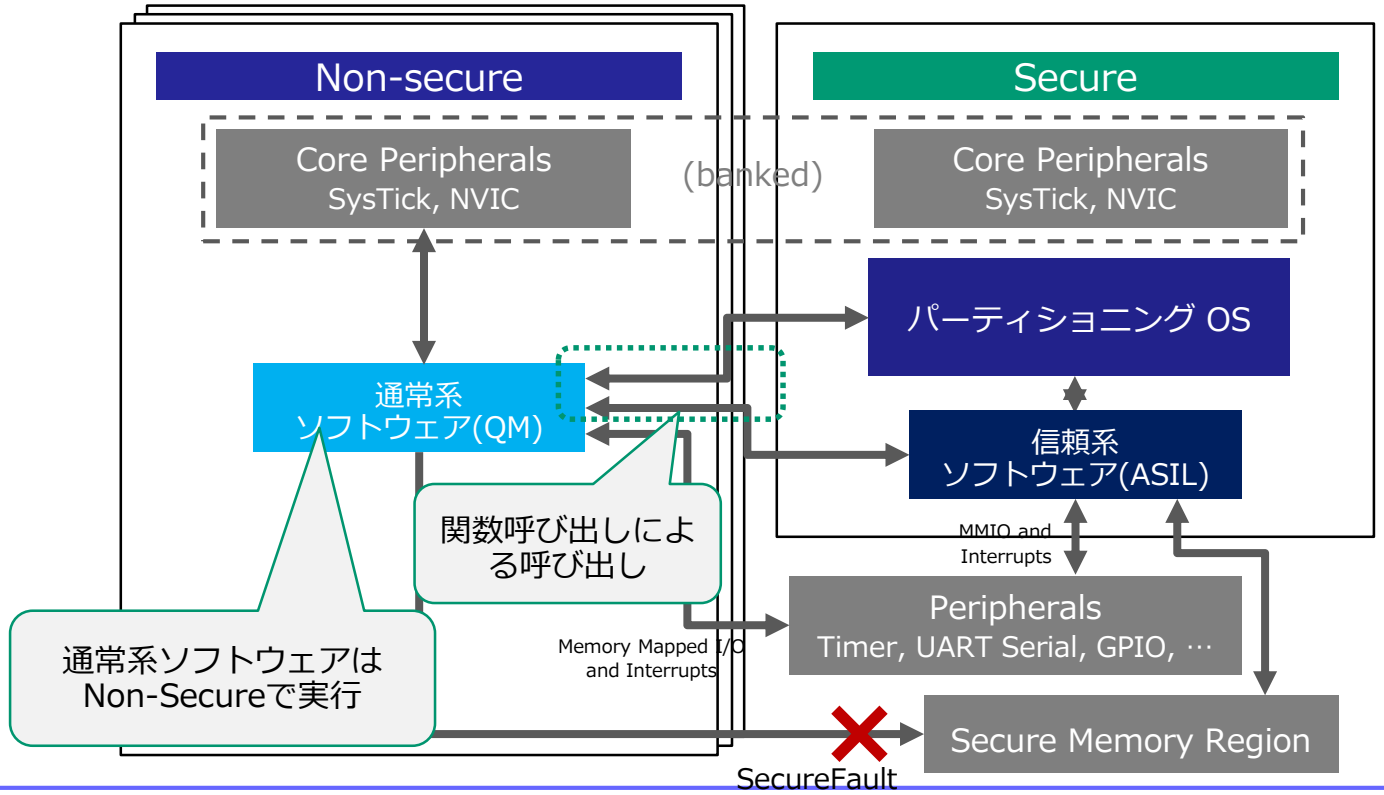
- 分割をASILとQMの2種類のみに分割することでよければ，Cortex-M TrustZoneを用いる方法が有力
- Cortex-M TrustZone の特徴
  - システムをSecureとNon-Secureに分割
    - 多くのリソースは2個持つ
  - Secure・Non-Secure間は関数呼び出しで移動可能
- Cortex-M TrusZoneの適用
  - 保護機構なしのAUTOSAR OS SC1を使用
  - ASILの機能をSecureへ，QMをNon Secureで実行
    - ITRON系のOSの評価では，保護なしOSと近い実行オーバヘッドで呼び出し可能



機能パーティショニング：ASILとQMへの分割

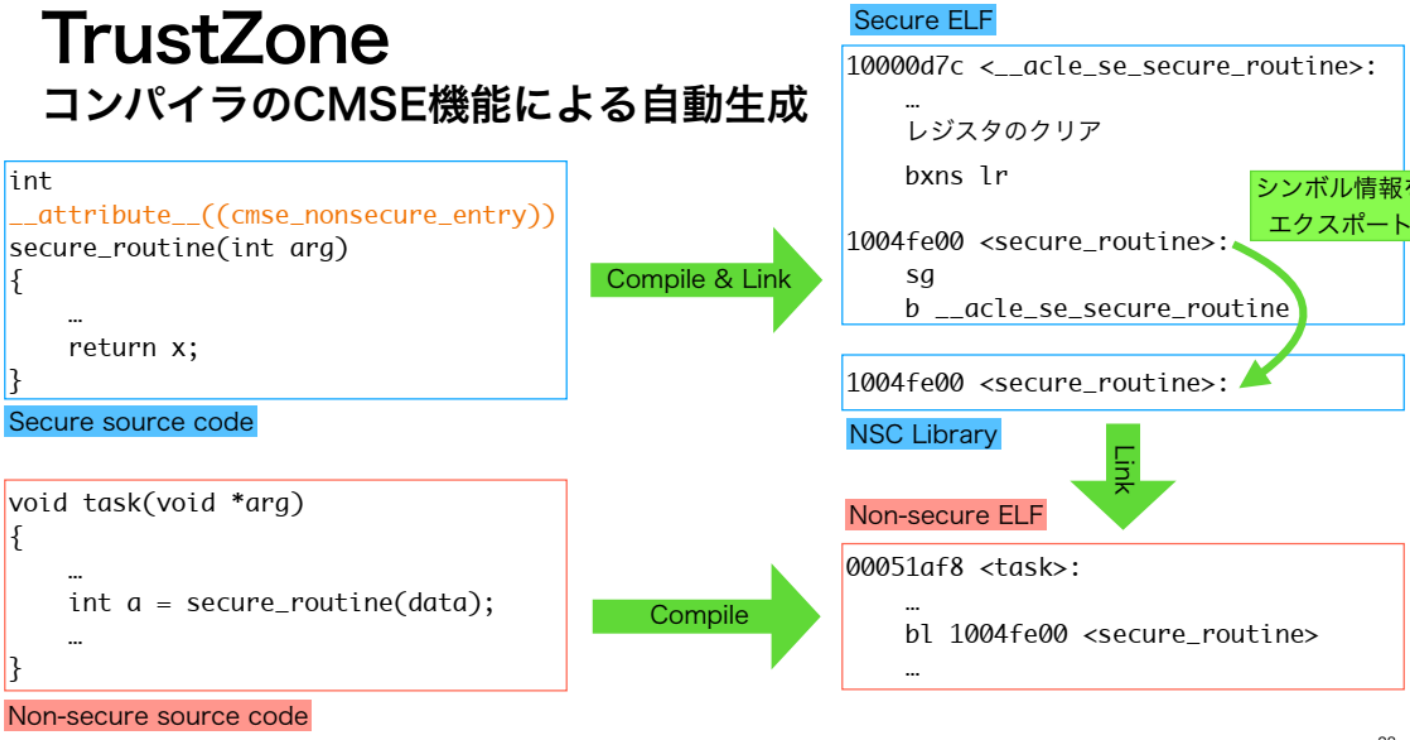
- 機能間の呼び出しオーバヘッドが低い
  - ITRON系のOSの評価では，保護なしOSと近い実行オーバヘッドで呼び出し可能





機能パーティショニング：ASILとQMへの分割

- ソースコードの書き換えが少ない
  - コンパイラの機能(CMSE機能)によりプラグマを追加のみで呼び出し可能
  - パラメータのチェックは必要に応じて追加が必要
    - ポインタの妥当性等



ハイパーバイザーの機能検討

ハイパーバイザーの機能検討

- 車載システム向けのハイパーバイザーは発展途上であり検討事項が多く存在
  - TDMAスケジューリングの拡張
  - マルチコアでの実現方法
  - VM間通信/HyperVisorサービス
  - デバイス共有の実現方法

TDMAスケジューリングの拡張

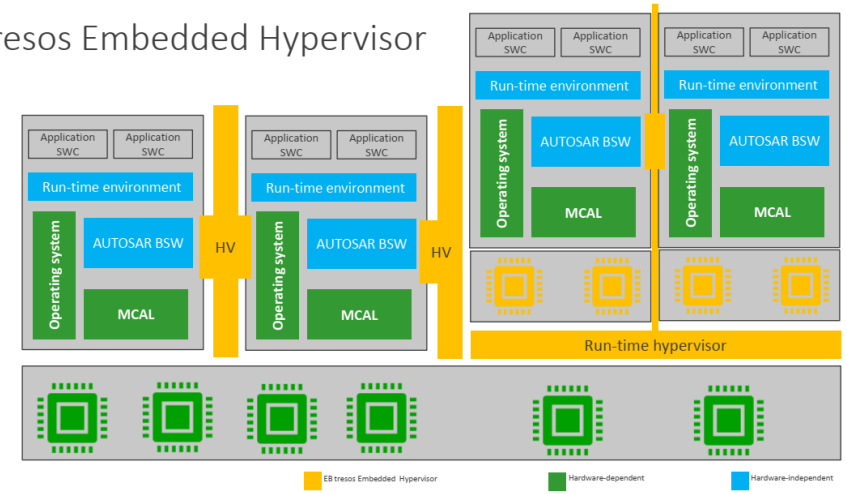
- TAMAスケジューリングの課題
  - 各VMに固定的にCPU時間を割り当てるため，負荷変動に対応出来ない
    - 検証性及び実行オーバヘッドとのトレードオフ
  - 各社拡張を実施
    - ETAS RTA Lightweight Hypervisor
      - ✓空きタイムウィンドウを用意しておき，処理量の増加リクエストがあったVMを実行時に割り付けて実行する拡張
      - ✓(研究)fixed priority (FP) サーバであるDeferrable Server(DS)を適用[1]
        - »非同期イベントを処理
    - EB tresos Embedded Hypervisor
      - ✓タイムウィンドウを一定時間延長する機能
    - SYSGO PikeOS
      - ✓優先度を導入
        - »優先度を低くすると他のパーティションがIDLE時に実行
        - »優先度を高くすると要求があれば他のパーティションを止めて実行

[1]D. Dasari, M. Pressler, A. Hamann, D. Ziegenbein and P. Austin, "Applying Reservation-based Scheduling to a μC-based Hypervisor: An industrial case study," *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Grenoble, France, 2020, pp. 987-990, doi: 10.23919/DATE48585.2020.9116385.

マルチコアでの実現方法

- EB tresos Embedded Hypervisor
  - 1つのVMで占有するコアの導入
    - 切り替えオーバヘッドが発生しない
- SysGo PikeOS
  - 時間クリティカルな処理は他のコアでは何も実行しないというスケジューリングも可能
  - パーティション切り換え時にキャシュとTLBをフラッシュしている.

EB tresos Embedded Hypervisor



出典:EB tresos資料

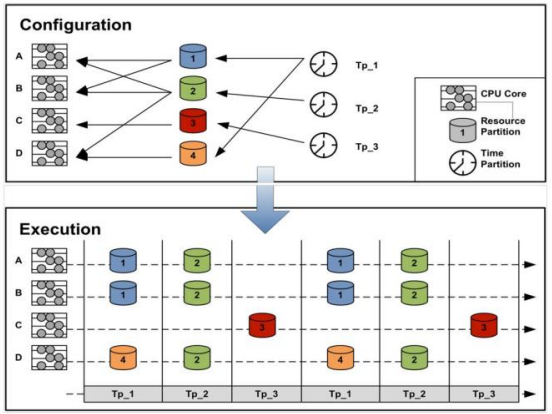
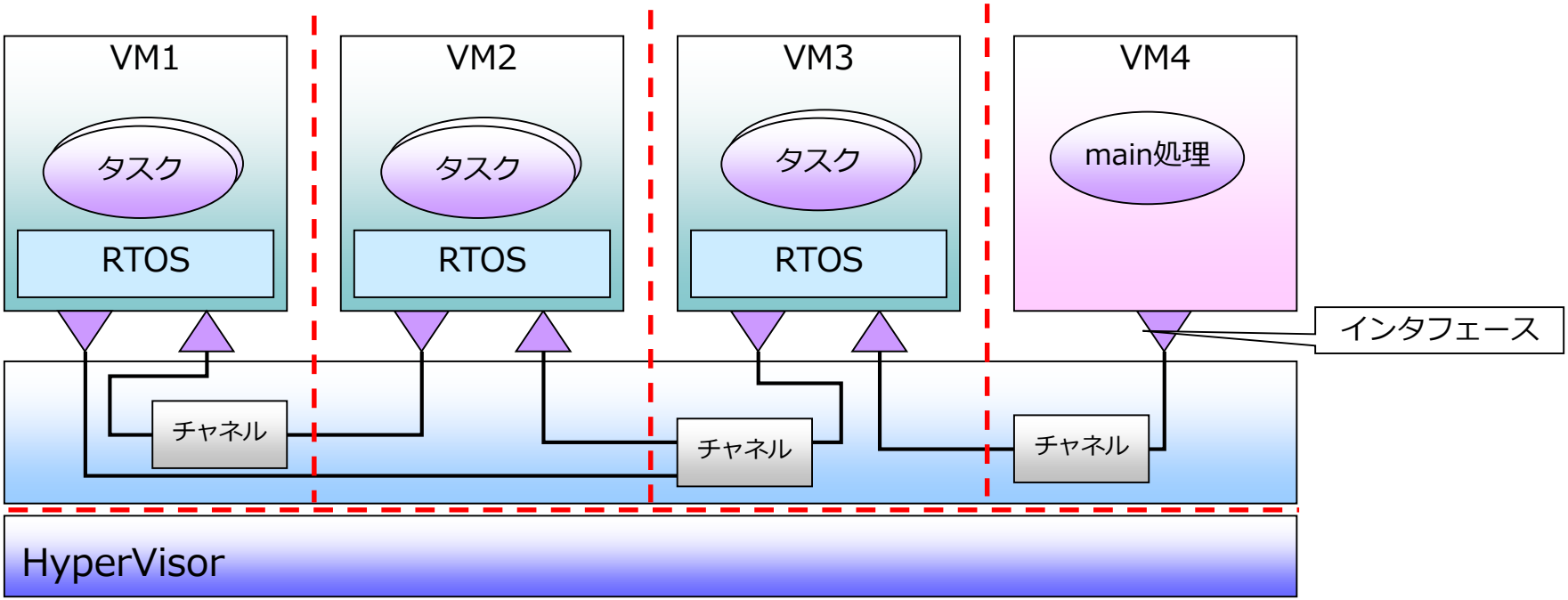


Figure 6: Example of resource and time partitioning in PikeOS SMP mode.

出典 : S. Saidi, R. Ernst, S. Uhrig, H. Theiling and B. D. de Dinechin, "The shift to multicores in real-time and safety-critical systems," *2015 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, Amsterdam, 2015, pp. 220-229, doi: 10.1109/CODESISSS.2015.7331385.

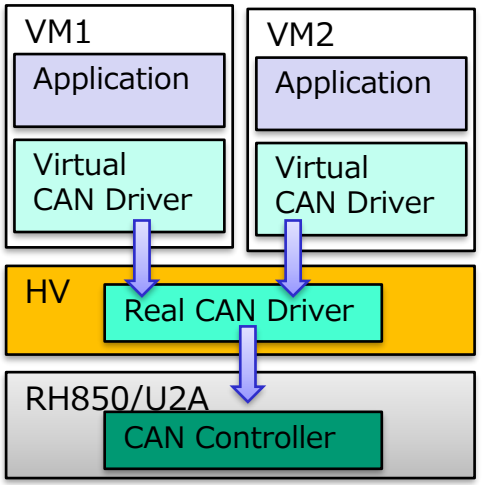
VM間通信機能/Hypvervisorサービス

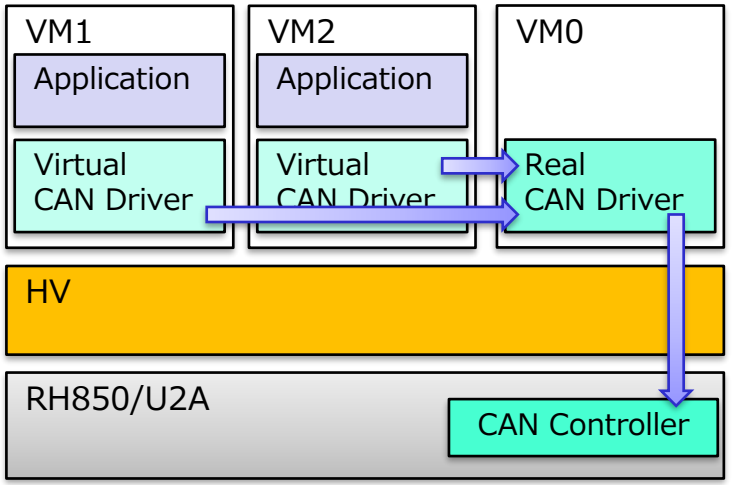
- デバイス共有を実現するためにも必要
- VM間通信機能
  - メッセージ送信や共有メモリ
- Hypvervisorサービス
  - TDMAを拡張するならスケジューリング関連の操作
  - VMの状態操作
  - アトミックメモリコピー
  - デバイスの操作



デバイス共有の実現方法

- 共有対象のデバイス候補
  - ネットワークデバイス（CAN, Ethernet等）
- 課題
  - どの様にデバイスを共有するか
    - メモリ保護・時間保護を保ちつつ共有をどう実現するか
  - 何処でデバイスドライバやプロトコルスタックを実行するか
    - HVで実行：最高レベルの安全度水準で開発する必要がある
    - 特定のVMで実行：VM間通信が必要．実行オーバヘッドが大きい?





まとめ

まとめ

ASILとQMソフトウェア混在を実現するための各種手法について説明

- 手法
  - AUTOSAR OS 保護機能による統合：AUTOSAR
  - ATK2-TPによる統合：ATK2-TP
  - 制御向けプロセッサのマルチコアによる統合：MCU-MC
  - 仮想マシンによる統合：HV

- 機能パーティショニングやECU統合といった，ASILとQMソフトウェアの混在を実現する方法として，マイコンの高性能化とハードウェア仮想化支援機能を用いたHypervisorを用いる方法が有力

- デイバスの共有のための仮想化技術も重要

- この領域の勉強会を企画中，興味のある方はお知らせ下さい.