

AUTOSAR レスから AUTOSAR 使用の マイグレーションガイドライン

Ver.1.0.0

2017/03/15

Copyright (C) 2016-2017 by Center for Embedded Computing Systems

Graduate School of Information Science, Nagoya Univ., JAPAN

Copyright (C) 2016-2017 by FUJI SOFT INCORPORATED, JAPAN

Copyright (C) 2016-2017 by NEC Communication Systems, Ltd., JAPAN

Copyright (C) 2016-2017 by SCSK Corporation, JAPAN

Copyright (C) 2016-2017 by SUZUKI MOTOR CORPORATION

Copyright (C) 2016-2017 by TOSHIBA CORPORATION, JAPAN

Copyright (C) 2016-2017 by Witz Corporation

上記著作権者は、以下の (1)~(3)の条件を満たす場合に限り、本ドキュメント（本ドキュメントを改変したものを含む。以下同じ）を使用・複製・改変・再配布（以下、利用と呼ぶ）することを無償で許諾する。

- (1) 本ドキュメントを利用する場合には、上記の著作権表示、この利用条件および下記の無保証規定が、そのままの形でドキュメント中に含まれていること。
- (2) 本ドキュメントを改変する場合には、ドキュメントを改変した旨の記述を、改変後のドキュメント中に含めること。ただし、改変後のドキュメントが、TOPPERS プロジェクト指定の開発成果物である場合には、この限りではない。
- (3) 本ドキュメントの利用により直接的または間接的に生じるいかなる損害からも、上記著作権者および TOPPERS プロジェクトを免責すること。また、本ドキュメントのユーザまたはエンドユーザからのいかなる理由に基づく請求からも、上記著作権者および TOPPERS プロジェクトを免責すること。

本ドキュメントは、AUTOSAR (AUTomotive Open System ARchitecture) 仕様にに基づいている。上記の許諾は、AUTOSAR の知的財産権を許諾するものではない。AUTOSAR は、AUTOSAR 仕様に基づいたソフトウェアを商用目的で利用する者に対して、AUTOSAR パートナーになることを求めている。

本ドキュメントは、無保証で提供されているものである。上記著作権者および TOPPERS プロジェクトは、本ドキュメントに関して、特定の使用目的に対する適合性も含めて、いかなる保証も行わない。また、本ドキュメントの利用により直接的または間接的に生じたいかなる損害に関しても、その責任を負わない。

<目次>

1. 概要	1
1.1 本文書の目的	1
1.2 本文書で示す範囲	1
1.2.1 想定する開発ケース	1
1.2.2 想定する利用方法	1
1.2.3 AUTOSAR を適用する範囲	1
1.3 本文書の構成	2
1.4 参考文献	3
1.5 環境	4
2. AUTOSAR の導入の全体概要	6
2.1 AUTOSAR の導入の全体的な流れ	6
2.2 AUTOSAR の導入パターンの選定	7
2.2.1 AUTOSAR のメリットの意識	7
2.2.2 プラットフォーム部分から導入した場合のメリットの活用度合いの比較	7
2.2.3 アプリケーション部分から導入した場合のメリットの活用度合いの比較	8
3. 検討詳細	9
3.1 検討の流れ	9
3.2 モデルカーの詳細	9
3.3 システム構成	11
3.4 AUTOSAR レスのソフトウェア詳細	13
3.4.1 ソフトウェア構成	13
3.4.2 プラットフォーム	14
3.4.3 アプリケーション	15
3.4.4 各機能の処理タイミング	15
4. プラットフォーム部分からの導入	16
4.1 AUTOSAR BSW の一部導入	16
4.1.1 概要	16
4.1.2 始めに OS を導入する場合	17
4.1.3 始めに COM スタックを導入する場合	20
4.2 AUTOSAR BSW の全導入	23
4.2.1 概要	23
4.2.2 OS と COM スタックの導入	23
4.2.3 OS と COM スタックと RTE を導入する場合	25
4.2.4 Complex Drivers の対応	28

4.3	アプリケーションを一部 SW-C 化.....	29
4.3.1	概要	29
4.3.2	RTE を導入していない場合	30
4.3.3	すでに RTE を導入している場合.....	34
4.4	アプリケーションを全 SW-C 化	36
4.4.1	概要	36
4.4.2	AUTOSAR SW-C 後のソフトウェア構成.....	37
4.4.3	SW-C の設計	38
5.	アプリケーション部分からの導入	39
5.1	アプリケーションを一部 SW-C 化.....	39
5.1.1	概要	39
5.1.2	AUTOSAR SW-C 後のソフトウェア構成.....	40
5.1.3	RTE の生成	40
5.2	アプリケーションを全 SW-C 化	43
5.2.1	概要	43
5.2.2	AUTOSAR SW-C 後のソフトウェア構成.....	44
5.3	AUTOSAR BSW の一部導入	45
5.3.1	概要	45
5.3.2	OS から導入する場合.....	46
5.3.3	COM スタックから導入する場合.....	47
5.4	AUTOSAR BSW の全導入.....	48
5.4.1	概要	48
5.4.2	導入後のソフトウェア構成.....	48
	変更履歴.....	49
図 1-1	本文書の構成.....	2
図 2-1	AUTOSAR の導入の全体的な流れ.....	6
図 3-1	全体構成の概要.....	10
図 3-2	システム構成 – AUTOSAR レス	11
図 3-3	システム構成 – AUTOSAR 使用	12
図 3-4	既存のソフトウェアのアーキテクチャ.....	13
図 3-5	各機能の定期処理タイミングの関係.....	15
図 4-1	OS 導入後のソフトウェア構成.....	17
図 4-2	AUTOSAR で定義されているスタートアップシーケンス.....	18
図 4-3	COM スタック導入後のソフトウェア構成.....	20
図 4-4	OS と COM スタック導入後のソフトウェア構成.....	24

図 4-5 OS と COM スタックと RTE 導入後のソフトウェア構成.....	25
図 4-6 Wrapper SW-C の設計図	26
図 4-7 一部を AUTOSAR SW-C 化した後のソフトウェア構成.....	30
図 4-8 一部を AUTOSAR SW-C 化した後の SW-C 設計図.....	31
図 4-9 一部を AUTOSAR SW-C 化した後のソフトウェア構成.....	34
図 4-10 一部を AUTOSAR SW-C 化した後の SW-C 設計図	35
図 4-11 全て AUTOSAR SW-C 化した後のソフトウェア構成	37
図 4-12 全て AUTOSAR SW-C 化した後の SW-C 設計図	38
図 5-1 一部を AUTOSAR SW-C 化した後のソフトウェア構成.....	40
図 5-2 全て AUTOSAR SW-C 化した後のソフトウェア構成.....	44
図 5-3 OS 導入後のソフトウェア構成	46
図 5-4 COM スタック導入後のソフトウェア構成.....	47
図 5-5 OS と COM スタック導入後のソフトウェア構成.....	48
表 1-1 参考文献	3
表 1-2 マイコン情報.....	4
表 1-3 構成要素	4
表 1-4 検討で使ったソフトウェア	5
表 2-1 プラットフォーム部分から導入した場合のメリットの活用度合いの比較	7
表 2-2 アプリケーション部分から導入した場合のメリットの活用度合いの比較	8
表 3-1 プラットフォーム部分の各機能	14
表 3-2 各アプリケーションの制御内容	15
表 4-1 スタブとして実装したファイルと処置内容.....	22
表 4-2 AUTOSAR OS への依存箇所と処置検討結果.....	22
表 4-3 システムディスクリプションファイルの記述内容.....	27
表 4-4 BSW モジュールディスクリプションファイルの記述内容.....	27
表 4-5 システムディスクリプションファイルの記述内容.....	32
表 4-6 BSW モジュールディスクリプションファイルの記述内容.....	32
表 4-7 システムディスクリプションファイルの記述変更内容	36

1. 概要

1.1 本文書の目的

本文書は、「AUTOSAR レスから AUTOSAR 使用のマイグレーション」のアプローチ方法において、部分的、且つ段階的な導入手順をまとめたガイドラインである。

AUTOSAR はソフトウェア制御のみでなく、開発手法も仕様規定されており、また VFB (Virtual Functional Bus) と RTE (Runtime Environment) の特徴的な仕様もあり、従来から使用されている既存の開発手法やソフトウェアに適用していくには、まず導入ハードルが高い。本文書は、その導入ハードルを下げ AUTOSAR の普及を促進することを目的として作成したものである。

本文書では、実際の車載ソフトウェア開発に役立つよう、より現実的なものとするため、モデルカーを題材にして、導入方法を検討した際の導入手順や要検討事項などを記載する。

1.2 本文書で示す範囲

1.2.1 想定する開発ケース

本文書では実際の車載ソフトウェア開発において、開発期間やコストの観点から現実的であり、且つ大多数のケースと想定される、既存のソフトウェアに対し部分的、且つ段階的に AUTOSAR を導入する方法について記載する。

新製品などのソフトウェアの新規構築や既存のソフトウェアのソースコードを使用しないようなりファクタリングなどは本文書の範囲外とする。

1.2.2 想定する利用方法

本文書は、実際の車載ソフトウェア開発において AUTOSAR の導入ハードルを下げることを目的として作成したものであるため、以下の利用方法を想定する。

- ・ 開発サイクル毎に部分的、且つ段階的に AUTOSAR を導入予定でこういった手順を進めるかを決める場合
- ・ AUTOSAR の導入を進める中での注意点を確認する場合
- ・ 組織や部門に AUTOSAR を使用した開発経験者が相対的に少なく、社内教育を実施する必要性が高く、その教育の題材として使用する場合

1.2.3 AUTOSAR を適用する範囲

本文書は、ガイドラインを策定する上で NPO 法人 TOPPERS プロジェクトから公開されている開発成果物を使用したものであり、AUTOSAR の導入範囲は以下となる。

- ・ AUTOSAR BSW は OS, COM スタックのみ導入
- ・ RTE の導入
- ・ アプリケーションの SW-C 化

1.3 本文書の構成

次章以降の説明の流れを図 1-1 に示す。

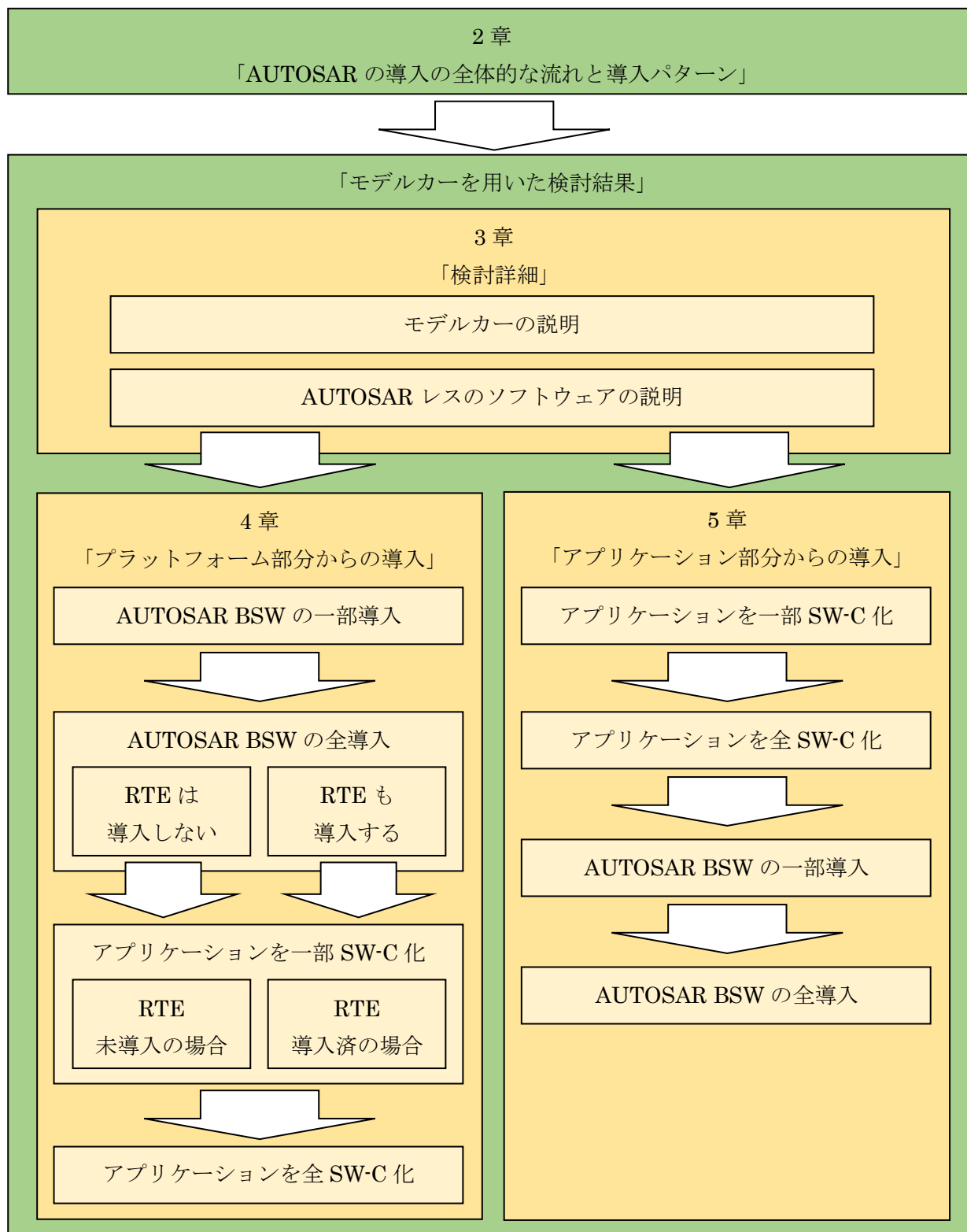


図 1-1 本文書の構成

1.4 参考文献

本文書から参照している文書，または本文書を理解するために必要な文書を表 1-1 に示す．内容は本文書に包含されていない．

表 1-1 参考文献

文書名	バージョン
ATK2 外部仕様書	Ver.1.2.1
次世代車載システム向け RTE 外部仕様書	Ver.1.3.0
次世代車載システム向け COM 外部仕様書	Ver.1.1.0
1.開発対象アプリケーション (※1)	なし
2.開発環境 (※1)	なし
システム設計書 (※1)	なし
モデルカー構築マニュアル (※1)	なし

※1: NPO 法人 TOPPERS プロジェクトから公開されているモデルカー制御プログラムに含まれる．
http://dev.toppers.jp/trac_user/contrib/browser/rc_autosar_rh850/trunk/doc

1.5 環境

モデルカーのマイコン情報を表 1-2 に示す.

表 1-2 マイコン情報

マイコン	開発環境
RH850/F1L	CCRH

モデルカーの構成要素を表 1-3 に示す.

表 1-3 構成要素

名前	説明
ESC(モータ)	モデルカーの動力.
サーボ	制御信号によってモータの回転軸の角度や速度をコントロールすることが可能.
ECU × 3 個	モデルカーを制御するためのアプリケーションが動作する基盤.
SBDBT5V	5V 入出力対応マイコン基盤. USB の Bluetooth アダプタを使用すると通信モジュールとして使用可能.
Bluetooth USB アダプタ	SBDBT5V に接続することで PS3 コントローラからのデータを受信することが可能.
プレイステーション 3(以下 PS3)コントローラ	モデルカーを操作することが可能.
車載カバー	車体を覆うカバー. ライトが接続されている.
タミヤ製 TLU-01	LED ライトユニット. 接続された LED ライトの点灯を制御する.
タミヤ製 ランプ用 LED	ヘッドランプ用 LED(白色)
	フォグランプ用 LED(白色または黄色)
	ウィンカー用 LED(オレンジ色)
	バックランプ用 LED(白色)
	ブレーキランプ用 LED(赤色)
ブザー	起動音, バック音用ブザー
バッテリー	バッテリーにより給電可能.
バッテリー充電器	バッテリーを充電するため.

実際に本導入検討で使用したソフトウェアを表 1-4 に示す。

表 1-4 検討で使用したソフトウェア

使用用途	フォルダ	AUTOSAR 導入部分 (○：導入，－：未導入)				
		OS	スタートアップ	COM スタック	RTE	SW-C 化
検討のベースプログラム						
	既存のソフトウェア	src/ecu1/100_Base	－	－	－	－
		src/ecu2/200_Base	－	－	－	－
プラットフォーム部分からの導入検討						
AUTOSAR BSW の 一部導入		src/ecu1/110_Platform_Ss	○	－	－	－
		src/ecu1/111_Platform_SsInit	○	○	－	－
		src/ecu1/120_Platform-Cs	－	－	○	－
		src/ecu2/220_Platform-Cs	－	－	○	－
AUTOSAR BSW の全導入						
	RTE も導入する場合	src/ecu1/121_Platform-CsRte	○	－	○	○
アプリケーションを一部 SW-C 化						
	RTE 未導入の場合	src/ecu1/130_Platform_Swc	○	○	○	○
	RTE 導入済の場合	src/ecu1/131_Platform_SwcRte	○	○	○	○
	アプリケーションを 全 SW-C 化	src/ecu1/140_Platform_AllSwc	○	○	○	○
アプリケーション部分からの導入検討						
	アプリケーションを 一部 SW-C 化	src/ecu1/310_Application_Swc	－	－	－	○
	アプリケーションを 全 SW-C 化	src/ecu1/320_Application_AllSwc	－	－	－	○
AUTOSAR BSW の 一部導入		src/ecu1/330_Application_Ss	○	－	－	○
		src/ecu1/331_Application_SsInit	○	○	－	○
AUTOSAR BSW の 全導入		src/ecu1/340_Application_Ss-Cs	○	○	○	○

※2 : AUTOSAR OS について未導入となっているものは、本来、AUTOSAR OS を使用していない想定だが、本導入検討では便宜上、ATK2-SC1 を使用し擬似的に実現している。

2. AUTOSAR の導入の全体概要

2.1 AUTOSAR の導入の全体的な流れ

既存のソフトウェアに対し部分的、且つ段階的に AUTOSAR を導入する全体的な流れを図 2-1 に示す。

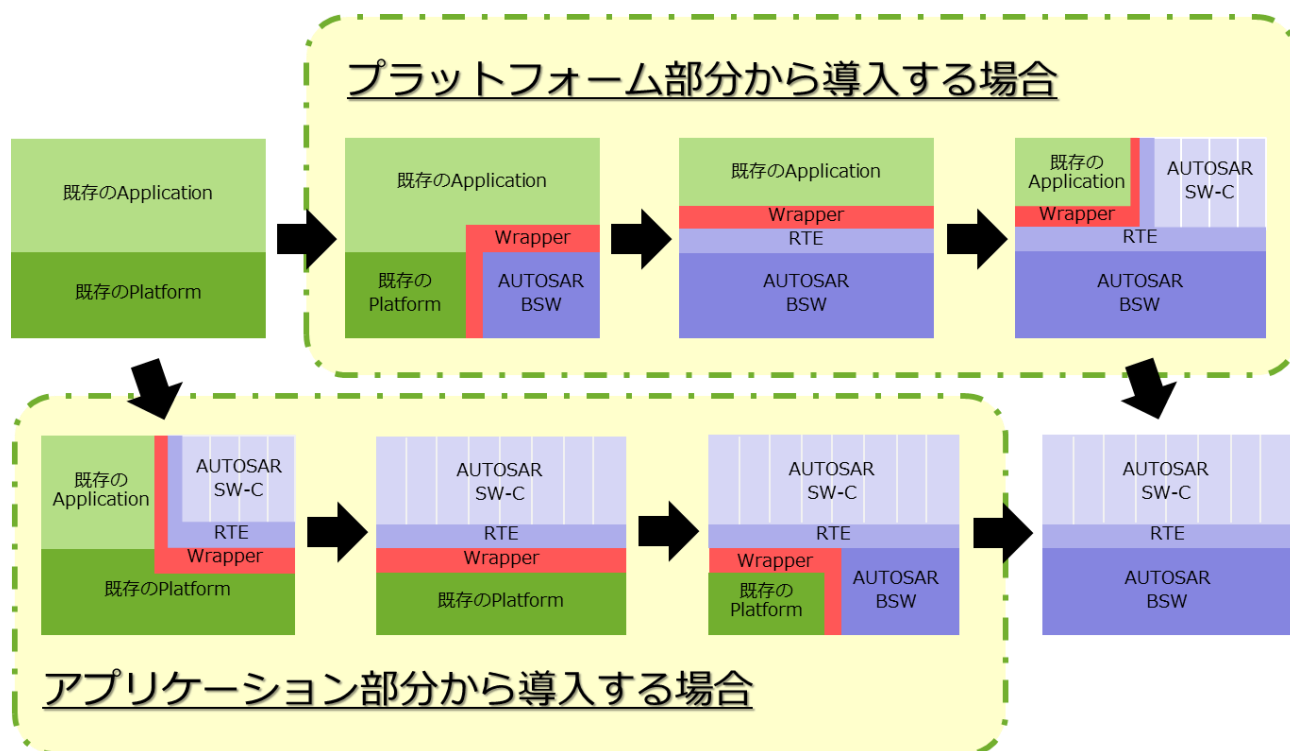


図 2-1 AUTOSAR の導入の全体的な流れ

この図では、まず「プラットフォーム部分から導入する場合」と「アプリケーション部分から導入する場合」に大きく分けており、従来から使用されている既存のプラットフォームを「既存の Platform」、既存のアプリケーションを「既存の Application」と表現している。また、AUTOSAR に適用したソフトウェアを「AUTOSAR BSW」、「RTE」、「AUTOSAR SW-C」と表現している。

また、導入途中段階であっても既存のソフトウェアがそのまま使用できるよう、必要に応じて作成するソフトウェアを「Wrapper」と表現している。

本導入検討における結果として、「プラットフォーム部分から導入する場合」と「アプリケーション部分から導入する場合」で「Wrapper」部分を除いて、対応内容に大きな違いはなかった。「Wrapper」部分においても、(当たり前ではあるが) 作成方法に違いはあるものの、作成規模に大きな違いはなかった。

目的にあった導入手順を行うことが重要であると言える。

2.2 AUTOSAR の導入パターンの選定

2.2.1 AUTOSAR のメリットの意識

AUTOSAR を導入することにより、車載ソフトウェア開発には様々なメリットがもたらされることになるが、限られた開発期間やコストの中で最大限のメリットが活かせるよう、メリットを意識した AUTOSAR の導入パターンを選定を推奨する。

本文書では「プラットフォーム部分から導入する場合」と「アプリケーション部分から導入する場合」で、AUTOSAR の導入でもたらされるメリットの活用度合いがどのように移り変わっていくかを相対的に比較したものを示すため、参考にして頂きたい。

2.2.2 プラットフォーム部分から導入した場合のメリットの活用度合いの比較

プラットフォーム部分から導入する場合に、導入が進むにつれ AUTOSAR の各メリットの活用度合いがどのようになるかを表 2-1 に示す。

表 2-1 プラットフォーム部分から導入した場合のメリットの活用度合いの比較

No.	メリット	Platform部分からの導入(左から導入順に記載)				
		OSのみ導入	COMのみ導入	BSWの全導入	一部SW-C化	全SW-C化
①	VFB(Virtual Function Bus)を用いることで設計の初期段階で詳細な検証が可能	×	×	×	△	◎
②	VFBを用いることでECUを意識せずにソフトウェア開発可能	×	×	×	○	◎
③	VFBとSW-CによりECUの再構築が容易	×	×	×	○	◎
④	開発プロセスの全領域をツールによってサポート	△	△	△	○	◎
⑤	自動車メーカーとサプライヤなどとのやりとりを各種標準規格で統一	×	×	×	△	◎
⑥	再利用/汎用品含む外部調達品の利用などの可能性・幅が拡大。	△	△	◎	◎	◎
⑦	自動コード生成の利用	△	△	△	○	◎

◎ : 十分活用できる ○ : 活用できる
△ : 少し活用できる × : 活用できない

2.2.3 アプリケーション部分から導入した場合のメリットの活用度合いの比較

アプリケーション部分から導入する場合に、導入が進むにつれ AUTOSAR の各メリットの活用度合いがどのようになるかを表 2-2 に示す。

表 2-2 アプリケーション部分から導入した場合のメリットの活用度合いの比較

No.	メリット	Application部分からの導入(左から導入順に記載)				
		一部 SW-C化	全SW-C 化	OSのみ 導入	COMの み導入	BSWの 全導入
①	VFB(Virtual Function Bus)を用いることで設計の初期段階で詳細な検証が可能	△	◎	◎	◎	◎
②	VFBを用いることでECUを意識せずにソフトウェア開発可能	○	◎	◎	◎	◎
③	VFBとSW-CによりECUの再構築が容易	△	○	○	○	◎
④	開発プロセスの全領域をツールによってサポート	△	○	○	○	◎
⑤	自動車メーカーとサプライヤなどとのやりとりを各種標準規格で統一	△	◎	◎	◎	◎
⑥	再利用/汎用品含む外部調達品の利用などの可能性・幅が拡大。	△	△	○	○	◎
⑦	自動コード生成の利用	○	○	○	○	◎

◎：十分活用できる ○：活用できる
△：少し活用できる ×：活用できない

3. 検討詳細

3.1 検討の流れ

モデルカーを使用し、以下の流れでソフトウェアを作成・変更し、AUTOSAR の導入検討を実施した。

- ① まずは一般的な ECU のソフトウェア構成 (OS レス) を仮定し、既存のソフトウェアを作成。
- ② 既存のソフトウェアを段階的、且つ部分的に AUTOSAR に適用。
- ③ 最終的に全て AUTOSAR を適用したソフトウェアを作成。

次章以降、モデルカーの説明と導入検討のベースとした既存のソフトウェアの説明を行う。

3.2 モデルカーの詳細

本文書の作成において、AUTOSAR の導入方法の検討の題材としたモデルカーの説明を行う。

本節は、「1.開発対象アプリケーション.pdf」から抽出して記載する。

題材としたモデルカーは、車両に搭載されている ECU を使用し、PS3 コントローラからの指示を制御することで操作を実現しているものである。

モデルカーの機能は以下 3 種類に分類される。

- ・ 操作系
 - ・ PS3 コントローラからの指示を解析して制御系に送る
 - ・ PS3 コントローラからは SBDBT と Bluetooth により接続され、SBDBT は PS3 コントローラからの指示を RCB3 と呼ばれる形式に変換し、UART を介して操作系を実現する ECU に送る
- ・ 制御系
 - ・ 操作系からの指示により、ボディ制御(各ランプ/ブザー)や車両制御(操舵角/車速)を決定
 - ・ ボディ制御の情報はボディ系に送る
 - ・ 決定した車両制御の情報を元にサーボと ESC(モータ)の制御を行う
 - ・ 制御系を実行するマイコンボードは、モデルカーとサーボと ESC(モータ)に接続されて速度・操舵角を制御可能
- ・ ボディ系
 - ・ 制御系からの指示によりボディの各ランプ/ブザーの ON/OFF を制御する

モデルカーに搭載されている 2 個の ECU に操作系・制御系・ボディ系を配置した時の全体構成の概要を図 3-1 に示す。ECU 間通信は CAN を使用する。

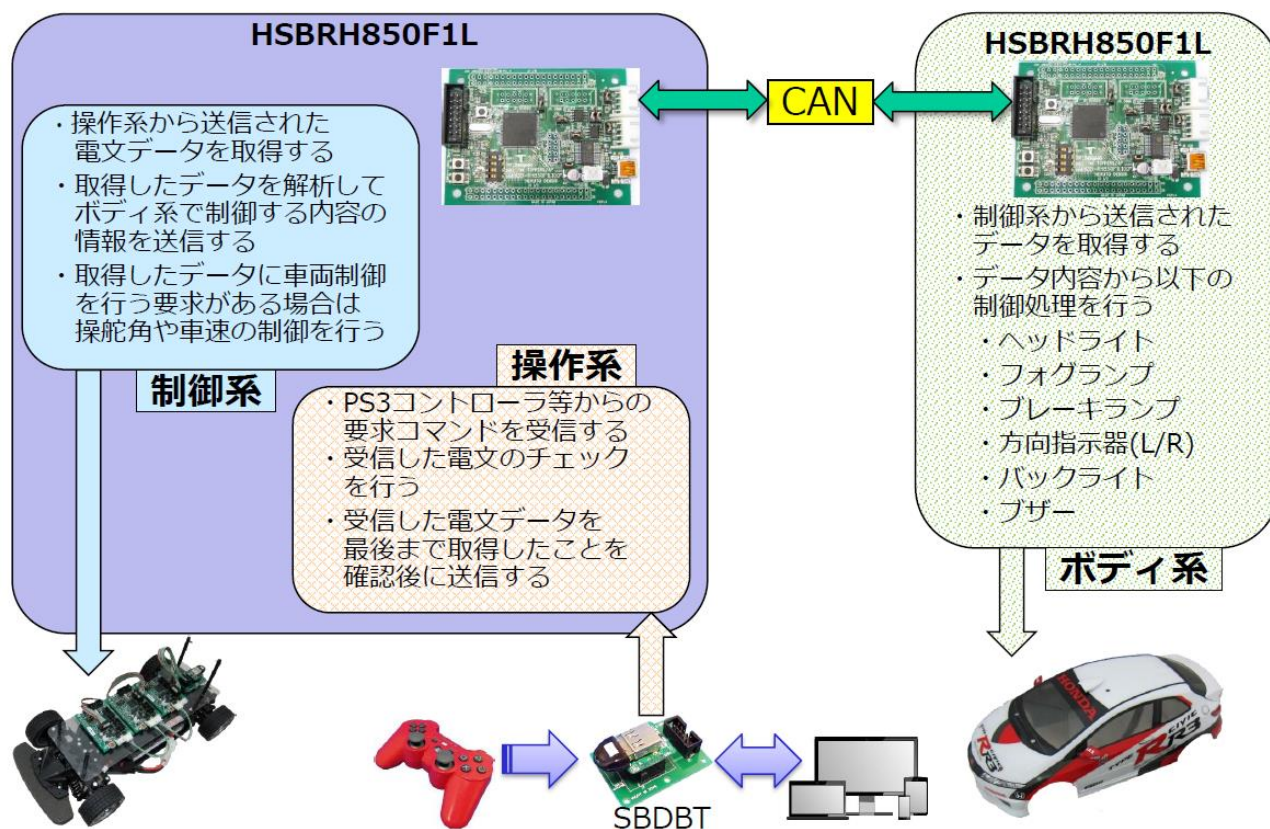


図 3-1 全体構成の概要

3.3 システム構成

モデルカーのシステム構成を記載する。

システム構成は、既存のソフトウェアが搭載されている「AUTOSAR レス」のシステム構成と全て AUTOSAR を適用したソフトウェアが搭載されている「AUTOSAR 使用」のシステム構成を記載する。

既存のソフトウェアにおいても、ハードウェアの制御や ECU 間の通信制御は、通信モジュールやドライバソフトウェアなどのプラットフォームに位置するソフトウェアが直接的な制御を行うことを想定している。

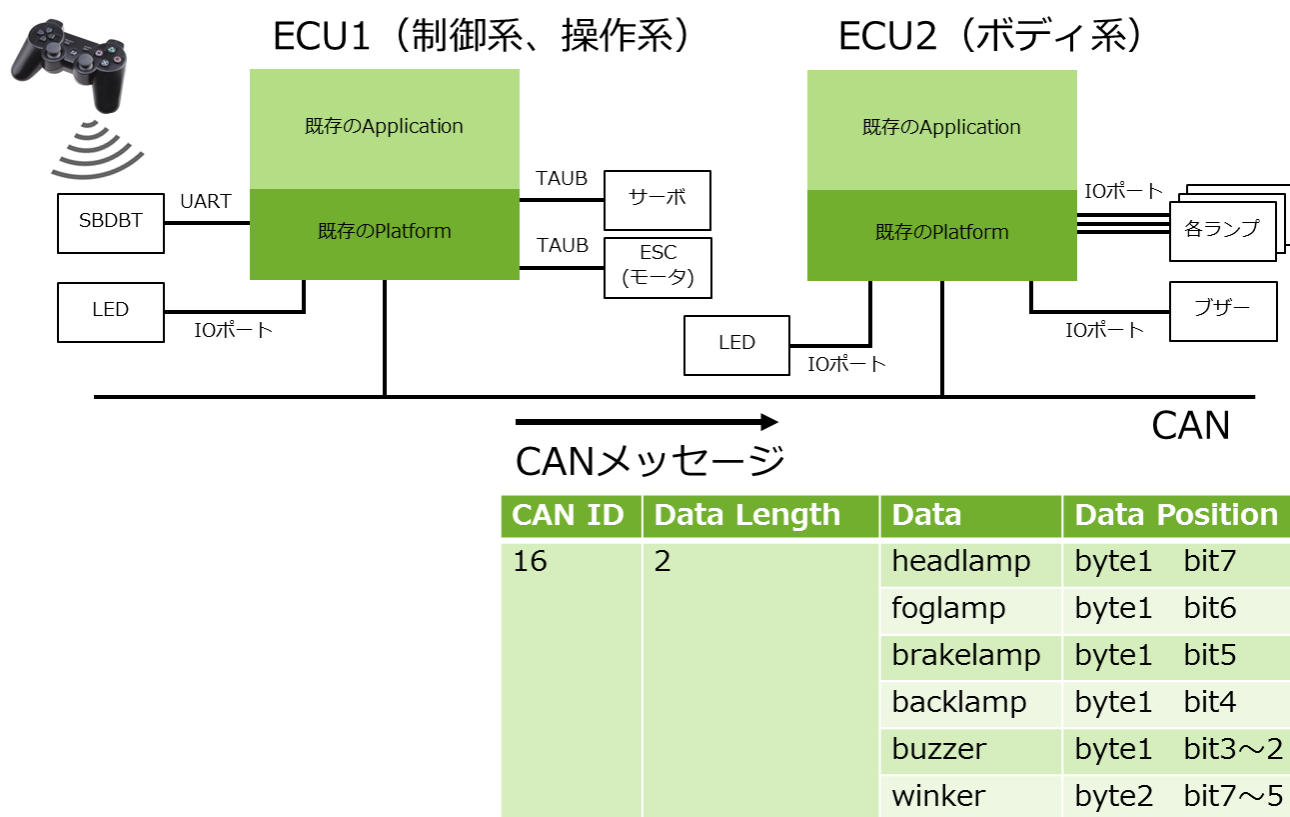


図 3-2 システム構成 — AUTOSAR レス

「AUTOSAR レス」のシステム構成と「AUTOSAR 使用」のシステム構成は同じとなる。

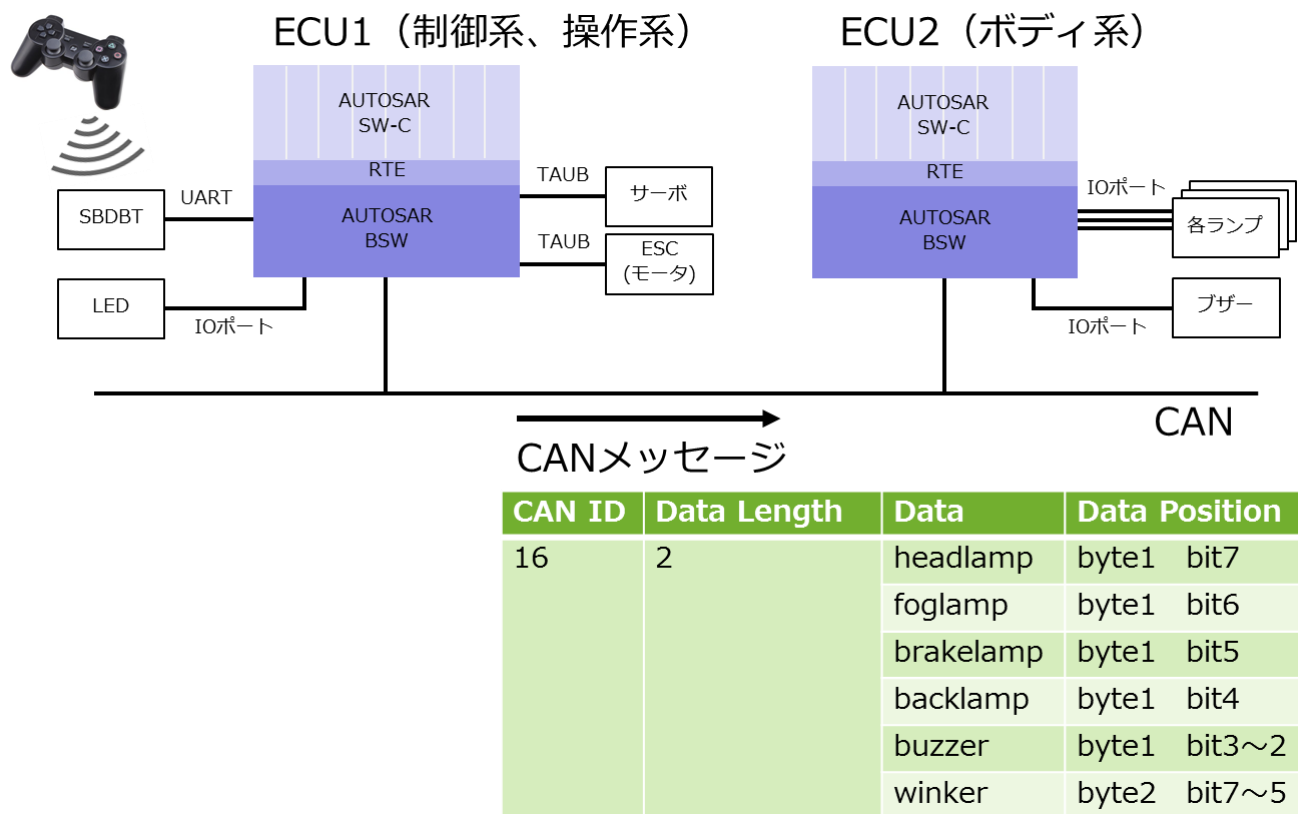


図 3-3 システム構成 - AUTOSAR 使用

3.4 AUTOSAR レスのソフトウェア詳細

3.4.1 ソフトウェア構成

一般的な ECU のソフトウェア構成 (OS レス) を仮定し作成した既存のソフトウェアの構成を図 3-4 に記載する。

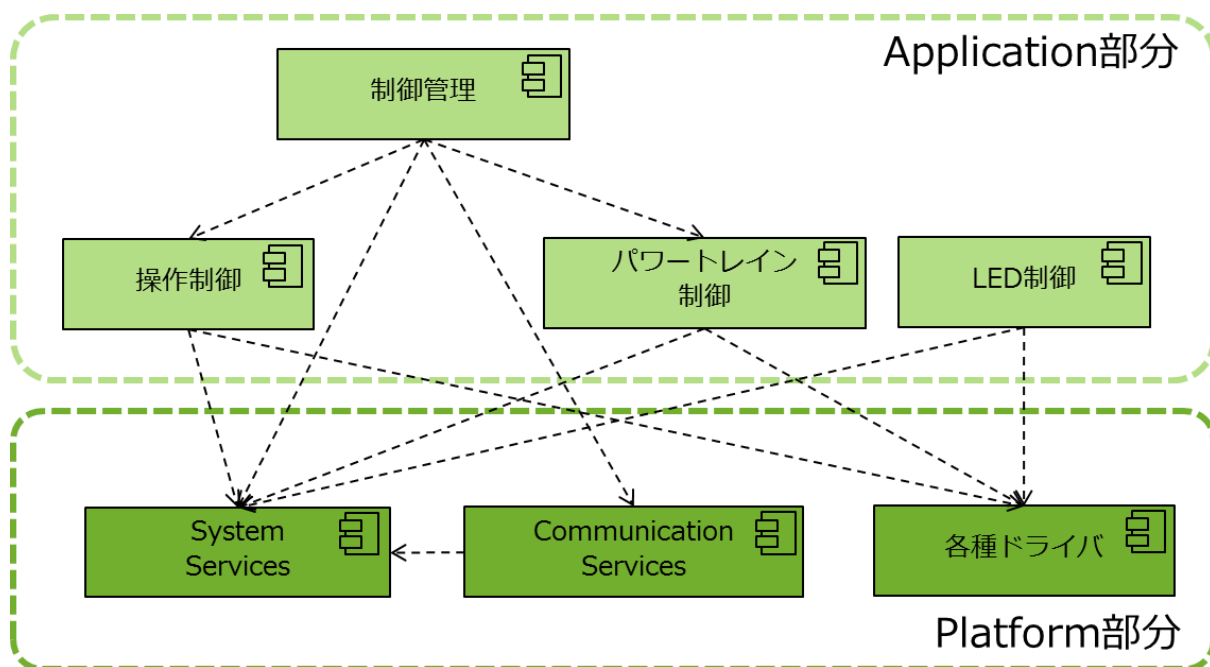


図 3-4 既存のソフトウェアのアーキテクチャ

3.4.2 プラットフォーム

プラットフォーム部分の各機能を表 3-1 に記載する.

表 3-1 プラットフォーム部分の各機能

コンポーネント名	機能	備考
System Services	スタートアップルーチンの実行	
	Hook の提供	
	初期化	本導入検討では ATK2-SC1 を使用し擬 似的に System Services の機能を作成.
	定期 ・ 高周期タスク (10ms 周期起動) ・ 定周期タスク (100ms 周期起動)	
	CPU のスリープ前・ウェイクアップ後	本検討では対象外
Communication Services	信号の送受信 IF の提供 (送信要求 IF / 受信値参照 IF)	
	ネットワークマネージメント	本検討では対象外
Memory Services	—	本検討では対象外
Diagnostic Services	—	本検討では対象外
各種ドライバ	UART ドライバ	
	RCB 解析ライブラリ	
	TAUB 用 PWM ドライバ	

3.4.3 アプリケーション

各アプリケーションの制御内容を表 3-2 に記載する.

表 3-2 各アプリケーションの制御内容

コンポーネント名	制御内容
操作制御	<ul style="list-style-type: none"> ・高周期タスク上で定期処理を実行. ・定期処理で PS3 コントローラからの 入力判定処理を実行.
制御管理	<ul style="list-style-type: none"> ・高周期タスク上で定期処理を実行. ・定期処理で操作制御からの入力を処理し, パワートレイン制御/ECU2 (ボディ制御) への要求を実行. (ECU2 への要求は CAN 通信を使用)
パワートレイン制御	<ul style="list-style-type: none"> ・高周期タスク上で定期処理を全割り込み禁止状態で実行. ・定期処理で制御管理からの要求に従って, サーボと ESC (モータ) を制御.
LED 制御	<ul style="list-style-type: none"> ・低周期タスク上で定期処理を実行. ・定期処理で動作確認用にマイコンボードの LED を点滅させる.

3.4.4 各機能の処理タイミング

各機能の定期処理が呼び出される関係を図 3-5 に記載する.

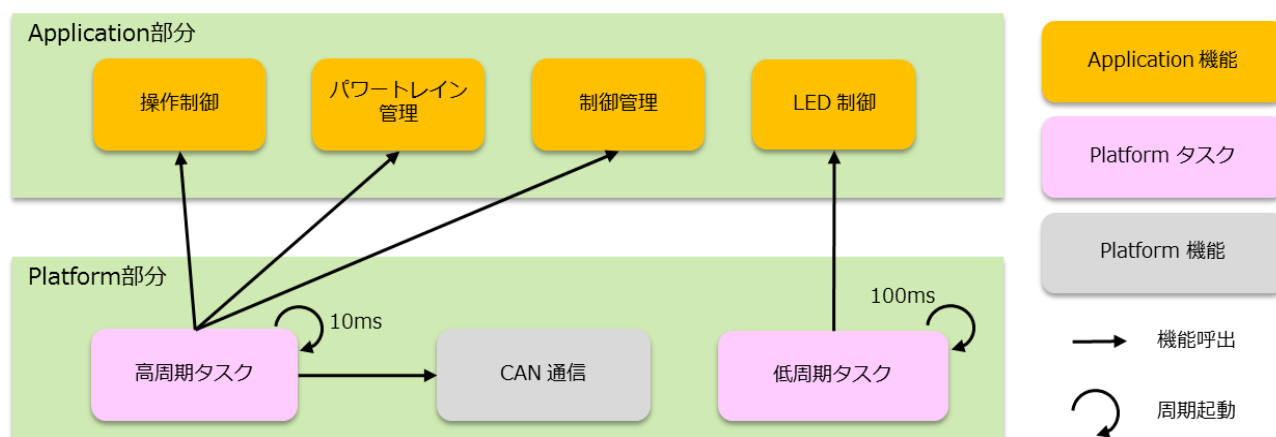


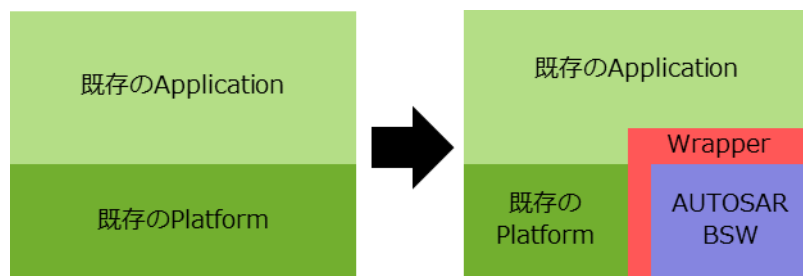
図 3-5 各機能の定期処理タイミングの関係

4. プラットフォーム部分からの導入

ECU1（制御系，操作系）の既存のソフトウェアに対しプラットフォーム部分から AUTOSAR の導入を実施する場合の手順や要検討事項を記載する。

4.1 AUTOSAR BSW の一部導入

4.1.1 概要



既存のプラットフォームの一部を AUTOSAR BSW に置き換える場合について、ポイントとしては以下となる。

- ・ 既存のソフトウェアにおいて、動作上、影響を受ける箇所への対応。
- ・ Wrapper ソフトウェアを作成することにより影響を局所化。
- ・ Complex Drivers に相当する既存のソフトウェアの各種ドライバの対応についてはそのドライバを使用するアプリケーションの SW-C 化と併せて対応する。（理由は 4.2.4 を参照）

本文書では以下のパターンについての手順を説明する。

- ・ 始めに OS を導入する場合
- ・ 始めに COM スタックを導入する場合

4.1.2 始めに OS を導入する場合

4.1.2.1 導入後のソフトウェア構成

OS の導入後のソフトウェア構成を図 4-1 に記載する。

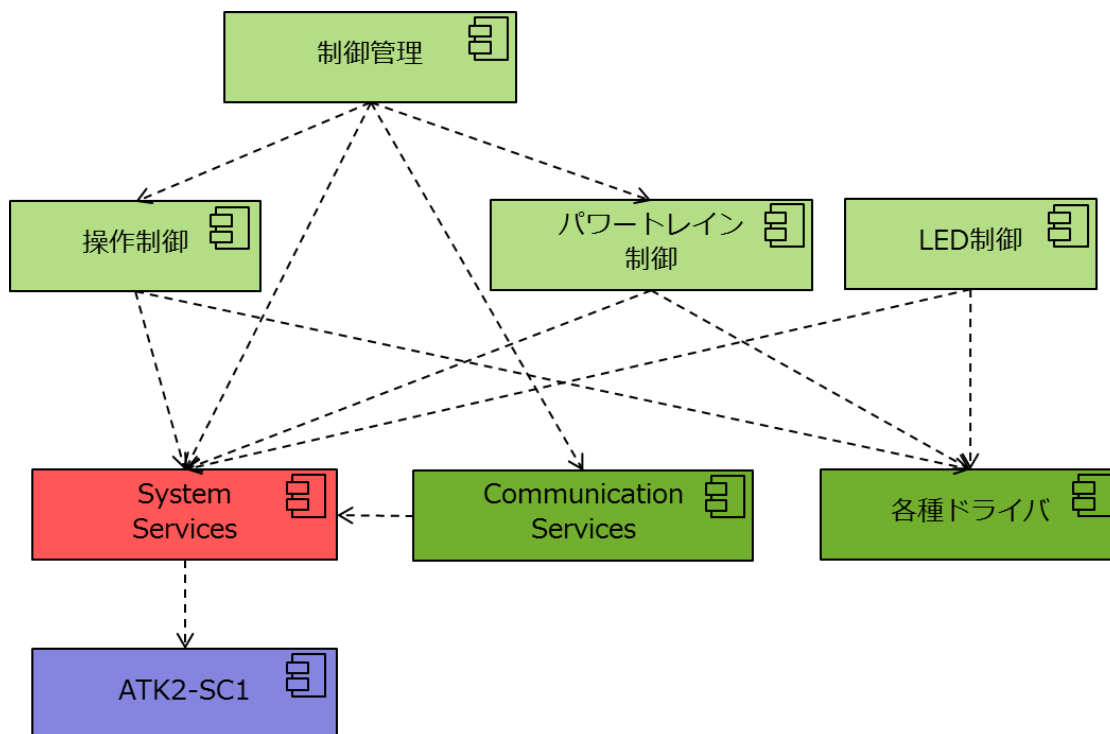


図 4-1 OS 導入後のソフトウェア構成

対応概要としては、以下となる。

- ・ 導入に対する影響をなるべく抑えるため、既存のソフトウェアの System Services とその他の結合部分は変更しないようにする。（System Services を Wrapper ソフトウェアとして実装）
- ・ 既存のソフトウェアの System Services が提供していた機能を AUTOSAR OS（本導入検討では ATK2-SC1 を使用）で実現する。

次章以降で各対応の詳細を記載する。

4.1.2.2 スタートアップルーチンの実行

Wrapper ソフトウェア（System Services）でスタートアップルーチンの処理を実装する。

- ・ Boot 処理の実装。
- ・ Main 関数の実装。また、Main 関数内でアプリケーションモードの設定と StartOS の呼び出しを実行。

<Main 関数の実装例>

```
sint32
main(void)
{
    AppModeType crt_app_mode = MainApp;
    /*
     * OS 起動
     */
    StartOS(crt_app_mode);
    while (1) {
    }
}
```

尚、必須ではないが、スタートアップ処理も AUTOSAR に適用する場合は C Init Code / EcuM を作成する。

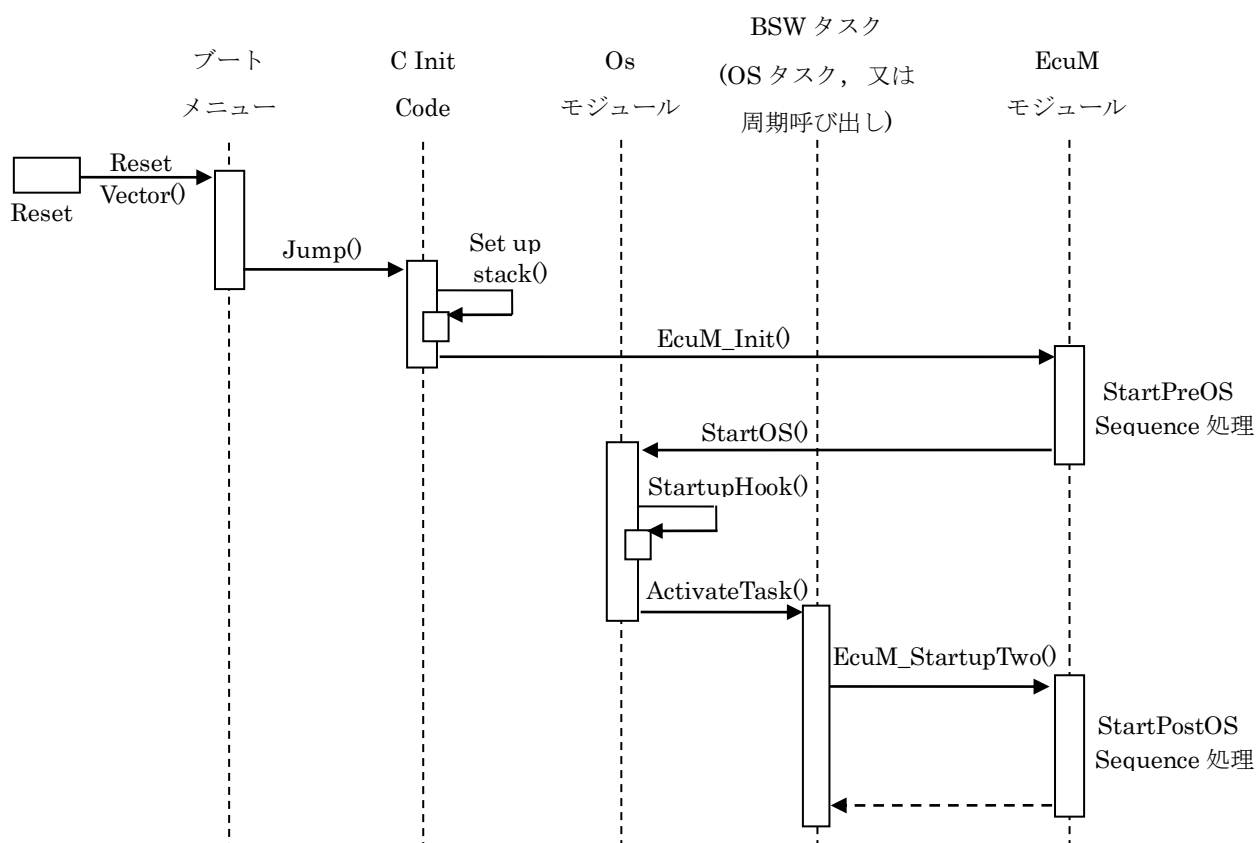


図 4-2 AUTOSAR で定義されているスタートアップシーケンス

4.1.2.3 Hook の提供

初期化

AUTOSAR OS が提供する StartupHook, 又は初期化タスク内で実行し, Wrapper ソフトウェア (System Services) を経由して, 他のコンポーネントへ提供する.

<初期化 Hook の実装例>

```
TASK(ECU_INIT_TASK)
{
    /* 初期化フックの呼び出し */
    InitHook();
    TerminateTask();
}
```

定期

OS タスクを使用し, Wrapper ソフトウェア (System Services) を経由して, 他のコンポーネントへ提供する. 優先度次第では OS アラームでも実現可能となる.

<定期 Hook の実装例>

```
TASK(HighTask)
{
    /*高周期タスク (10ms 周期起動) */
    HighTaskHook();
    TerminateTask();
}
```

4.1.2.4 割り込み処理のサポート

C1ISR, C2ISR を使用する.

タイミングを刻んでいるだけの処理の場合は OS タスク, OS アラームで実現可能となる.

4.1.2.5 割り込み禁止／解除用 IF の提供

AUTOSAR OS が提供する IF を使用する.

4.1.2.6 その他

各種 OS フックルーチン (シャットダウンフック／エラーフックなど) を必要に応じて実装する.

4.1.3 始めに COM スタックを導入する場合

4.1.3.1 導入後のソフトウェア構成

COM スタックの導入後のソフトウェア構成を図 4-3 に記載する。

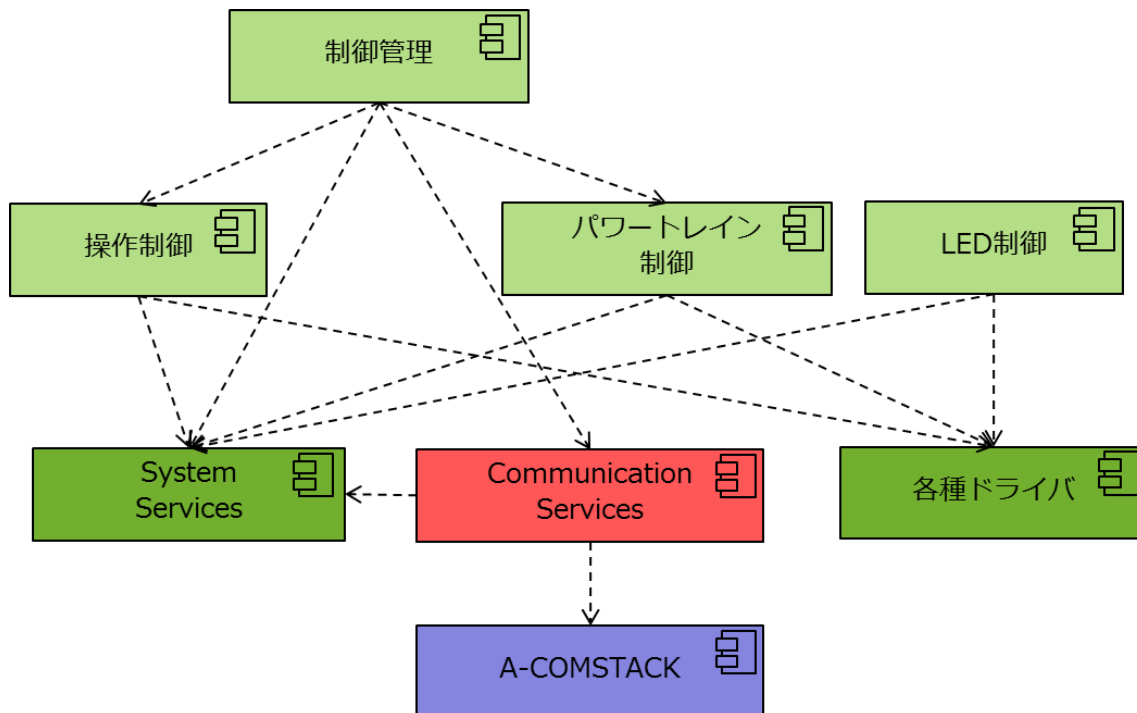


図 4-3 COM スタック導入後のソフトウェア構成

対応概要としては、以下となる。

- ・ 導入に対する影響をなるべく抑えるため、既存のソフトウェアの Communication Services とその他の結合部分は変更しないようにする。（Communication Services を Wrapper ソフトウェアとして実装）
- ・ 既存のソフトウェアの Communication Services が提供していた機能を AUTOSAR COM スタック（本導入検討では A-COMSTACK を使用）で実現する。

次章以降で各対応の詳細を記載する。

4.1.3.2 初期化处理

既存のソフトウェアの System Services が提供する初期化 Hook に Can, CanIf, Com モジュールの初期化处理を追加する。

4.1.3.3 定期処理

既存のソフトウェアの System Services が提供する定期 Hook に Can, CanIf, Com モジュールの定期処理を追加する。(A-COMSTACK では Com の定期処理 Com_MainFunctionTx0, Com_MainFunctionRx0の追加のみ)

4.1.3.4 信号の送受信 IF の提供

既存のソフトウェアの Communication Services が提供していた送信要求 IF／受信値参照 IF の関数内を Com_SendSignal0／Com_ReceiveSignal0に置き換える処理を追加する。

ただし、既存のソフトウェアの Communication Services が提供していた送信要求 IF／受信値参照 IF の使用箇所が少ない場合は、Wrapper ソフトウェア (Communication Services) を経由せず、使用箇所を直接変更しても良い。どちらを選択するかは影響範囲、コストを考慮して検討する。(本導入検討では Wrapper ソフトウェア (Communication Services) は省略した。)

<送信要求 IF 内の実装例>

```
void
既存の信号送信要求関数(uint8 value)
{
    Com_SendSignal(ComConf_ComSignal_ComSignal0, &value);
}
```

<受信値参照 IF 内の実装例>

```
sint8
既存の受信値参照関数(void)
{
    Std_ReturnType ercd;
    IDT_Buzzer      *p_data;
    ercd = Com_ReceiveSignal(ComConf_ComSignal_ComSignal1, &buzzer);
    if (ercd != E_OK) {
        前回値を戻り値に設定;
    } else {
        変数 buzzer 値を戻り値に設定;
    }
    . . .
}
```

4.1.3.5 他の AUTOSAR BSW モジュールとの関連箇所の対応

Can, CanIf, Com モジュールはその他の AUTOSAR BSW モジュールと関連があるが、実際にはそのモジュールがないため、その関連箇所への対応を実施する。

以下、本導入検討で実施した処置内容を記載する。

スタブの作成

以下、スタブとして実装したファイルと処置内容を表 4-1 に記載する。

表 4-1 スタブとして実装したファイルと処置内容

ファイル	処置内容
EcuM.h	CanIf で使用する型定義を追加。(EcuM_WakeupSourceType)
EcuM_Cbk.h	CnaIf で使用する関数を実装。(EcuM_ValidateWakeupEvent(), EcuM_CheckWakeup())
Rte_Cbk.h	Com でインクルードするファイルを追加。(空ファイル)
SchM_Can.h	・ Can, CanIf, Com で使用する排他処理を実装。(空マクロ)
SchM_CanIf.h	・ SchM_Enter_XXX_Reentrant_Y(), SchM_Exit_XXX_Reentrant_Y()
SchM_Com.h	(XXX は Can/CanIf/Com, Y は数値又は省略)

AUTOSAR OS に依存する箇所への対応

本導入検討では System Services の機能を ATK2-SC1 を使用し擬似的に作成したため、対応していないが、依存する箇所の洗い出しを実施し対応方法を検討したため、その内容を表 4-2 に記載する。

表 4-2 AUTOSAR OS への依存箇所と処置検討結果

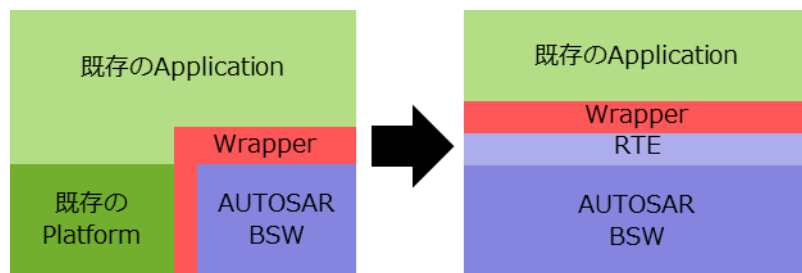
依存箇所	処置検討結果
基本型定義	既存のソフトウェアの定義と重複しないよう基本型定義を追加する。
環境依存部分 (メモリ、レジスタアクセス)	既存のソフトウェアの定義、処置に置き換えるか、又は個別に定義する。
CAN 送受信割り込み	既存のソフトウェアの System Services でサポートされている割り込み処理を使用するか、又は個別に割り込みの設定を実施する。
ハードウェアカウンタ (CAN コントローラの状態監視処理で使用)	既存のソフトウェアで使用しているハードウェアカウンタに置き換える。

4.1.3.6 ECU コンフィギュレーション

Can, CanIf, Com モジュールのコンフィギュレーションを行い、ジェネレータを実行しソースコードを生成する。(システム/BSW モジュールディスクリプションファイルは不要。)

4.2 AUTOSAR BSW の全導入

4.2.1 概要



既存のプラットフォームを全て AUTOSAR BSW に置き換える場合について、ポイントとしては以下となる。

- ・ RTE を導入しない場合は Wrapper ソフトウェアの作成とその処置内容は一部を AUTOSAR BSW に置き換える場合とほぼ同じ対応内容となる。
- ・ RTE を導入する場合は既存のソフトウェアを Wrapper SW-C として扱い対応する。
- ・ Complex Drivers に相当する既存のソフトウェアの各種ドライバの対応についてはそのドライバを使用するアプリケーションの SW-C 化と併せて対応する。(理由は 4.2.4 を参照)

本文書では以下のパターンについての手順を説明する。

- ・ OS と COM スタックを導入する場合
- ・ OS と COM スタックと RTE を導入する場合

4.2.2 OS と COM スタックの導入

4.2.2.1 導入後のソフトウェア構成

OS と COM スタックの導入後のソフトウェア構成を図 4-4 に記載する。

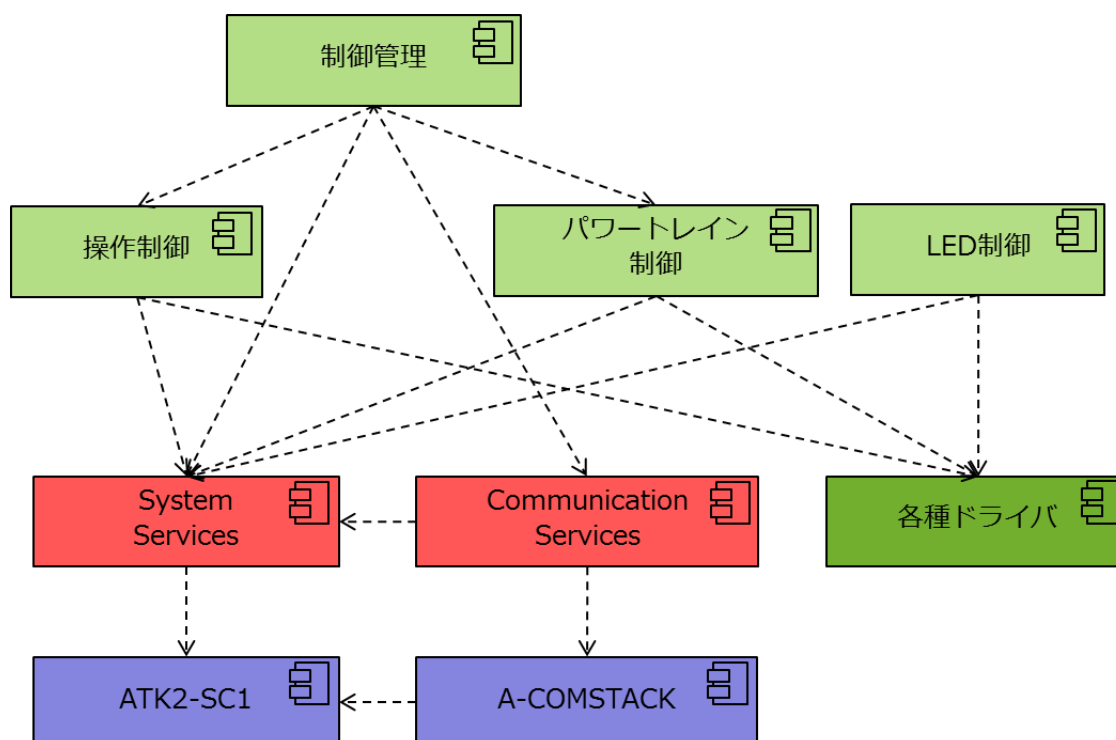


図 4-4 OS と COM スタック導入後のソフトウェア構成

対応概要としては、以下となる。

- ・ 処置内容は既存のプラットフォームの一部を AUTOSAR BSW に置き換える場合とほぼ同じとなる。

次章で既存のプラットフォームの一部を AUTOSAR BSW に置き換える場合との相違点、その他検討事項を記載する。

4.2.2.2 既存のプラットフォームの一部を AUTOSAR BSW に置き換える場合との相違点

AUTOSAR OS のみ、又は AUTOSAR COM スタックのみを導入した場合との相違点を記載する。

- ・ 始めに COM スタックを導入する場合に必要な「表 4-2 AUTOSAR OS への依存箇所と処置検討結果」の対応は不要となる。
- ・ 始めに COM スタックを導入する場合に既存のソフトウェアの System Services が提供する機能を使用して実行していた以下の処理は AUTOSAR OS から直接呼び出す形にしても良い。
 - ・ Can, CanIf, Com モジュールの初期化处理
 - ・ Can, CanIf, Com モジュールの定期処理

4.2.3 OS と COM スタックと RTE を導入する場合

4.2.3.1 導入後のソフトウェア構成

OS と COM スタックと RTE の導入後のソフトウェア構成を図 4-5 に記載する。

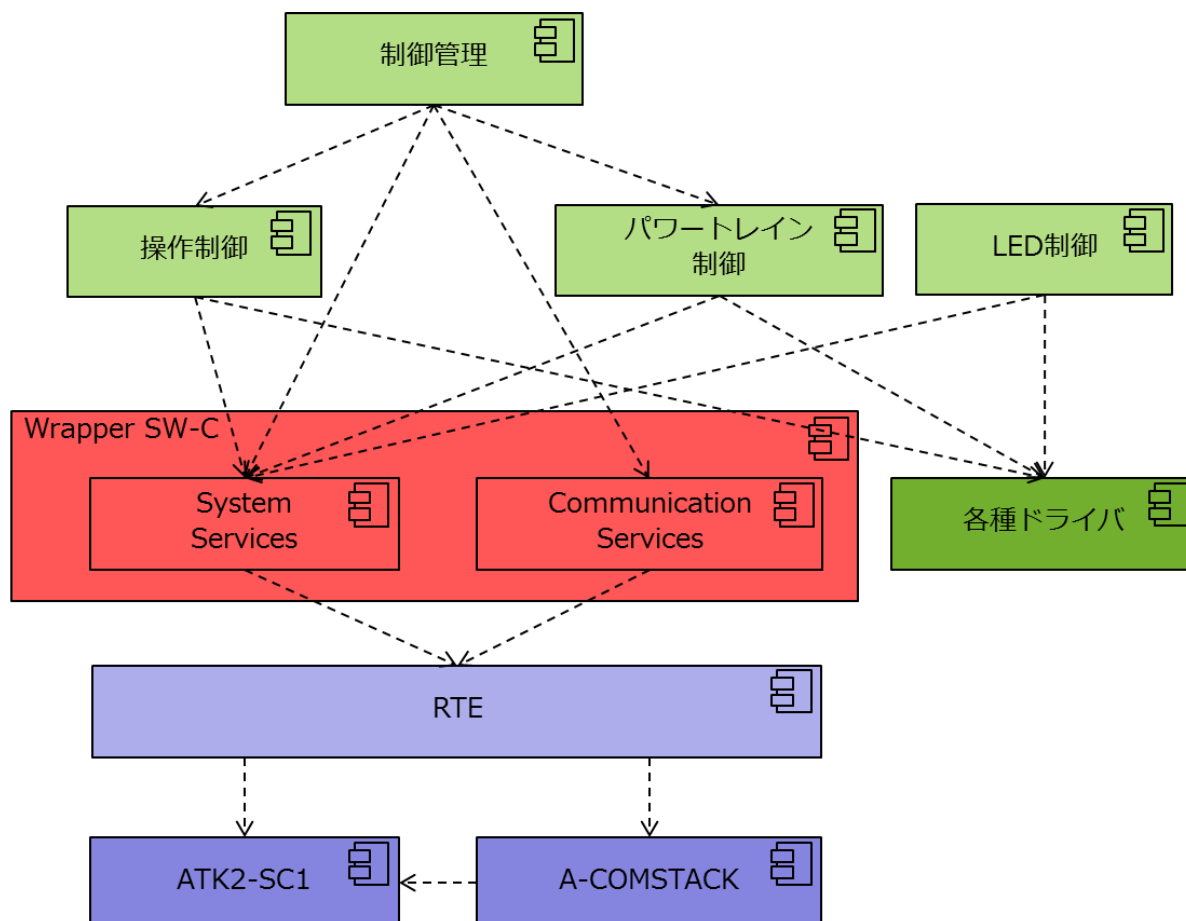


図 4-5 OS と COM スタックと RTE 導入後のソフトウェア構成

対応概要としては、以下となる。

- ・ 既存のソフトウェアとの結合部分を変更しないようにすることや既存のソフトウェアが提供していた機能を AUTOSAR BSW で実現する考え方については、既存のプラットフォームの一部を AUTOSAR BSW に置き換える場合とほぼ同じとなる。
- ・ RTE API を利用するため既存のソフトウェアに対する Wrapper SW-C を作成し、システム／BSW モジュールディスクリプションファイルを作成する。

次章以降で RTE を導入した場合に必要な対応を記載する。

4.2.3.2 SW-C の設計

既存のソフトウェアに対する Wrapper SW-C を作成し、既存のソフトウェアの System Services や Communication Services で実行していた定期処置などや ECU 間連携を抽出し、SW-C の設計を行う。

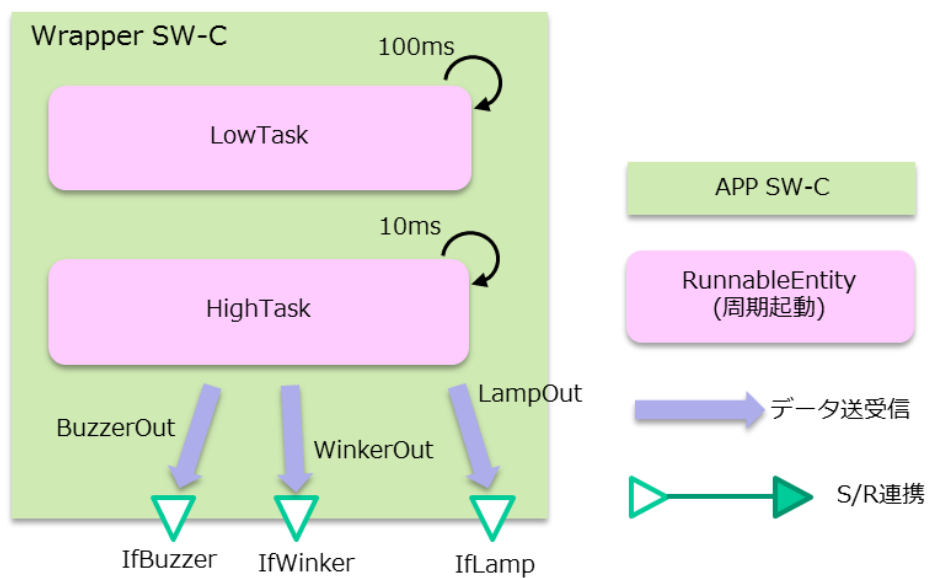


図 4-6 Wrapper SW-C の設計図

4.2.3.3 システム／BSW モジュールディスクリプションファイルの作成

SW-C の設計を元にシステム／BSW モジュールディスクリプションファイルを作成する。
本導入検討で行った概要を表 4-3、表 4-4 に記載する。

表 4-3 システムディスクリプションファイルの記述内容

対応項目	記述内容
SW-C の定義	<ul style="list-style-type: none">・ LowTask, HighTask のランナブルを定義・ ECU 間連携用のポートを定義（他 ECU の部分は設計しなくても良い）・ RTE イベントを定義
インターフェースの定義	<ul style="list-style-type: none">・ ECU 間連携用のインターフェースを定義
ECU の定義	<ul style="list-style-type: none">・ 対象の ECU を定義（その他 ECU の定義は不要）
ネットワークの定義	<ul style="list-style-type: none">・ 他 ECU との通信データ（SystemSignal）の定義・ iSignal の定義と SystemSignal への参照・ iSignal の PDU へのマッピング
各種マッピング	<ul style="list-style-type: none">・ SW-C の ECU へのマッピング・ SW-C のポート情報と通信データ（SystemSignal）のマッピング
その他	<ul style="list-style-type: none">・ ECU 構成に依存しないデータ型を定義

表 4-4 BSW モジュールディスクリプションファイルの記述内容

対応項目	記述内容
COM スタックの使用	<ul style="list-style-type: none">・ Can, CanIf, Com モジュールの BSW のエントリー、排他エリア, RTE イベントを定義

4.2.3.4 定期処理

「4.1.2 始めに OS を導入する場合」に行った「4.1.2.3 Hook の提供」では、OS タスクを使用し Wrapper ソフトウェア（System Services）を経由して、他のコンポーネントへ定期 Hook を提供していたが、RTE を導入した場合は Wrapper SW-C のランナブルとして呼び出すことになる。

4.2.3.5 信号の送受信 IF の提供

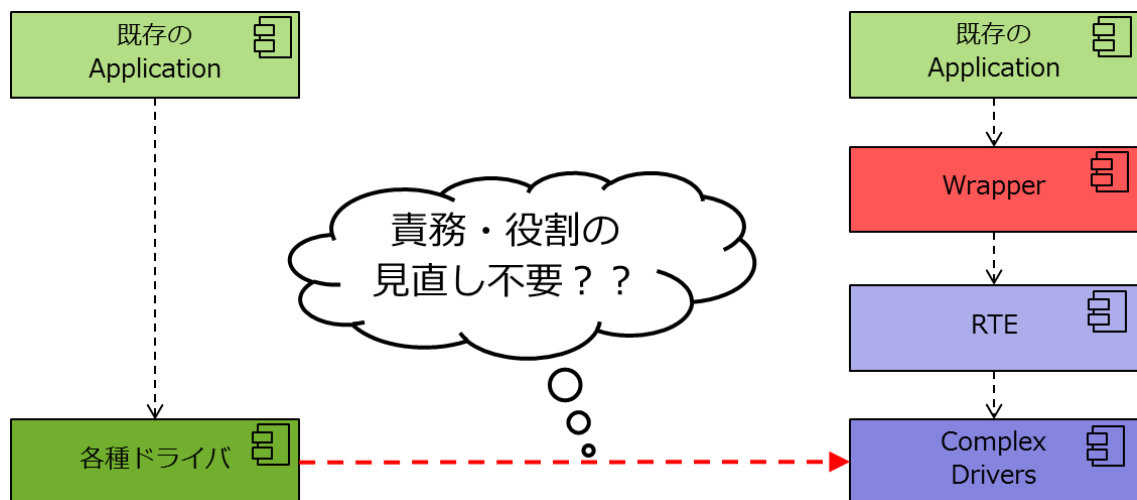
「4.1.3 始めに COM スタックを導入する場合」に行った「4.1.3.4 信号の送受信 IF の提供」では、Com モジュールの Com_SendSignal0/Com_ReceiveSignal0を直接使用していたが、RTE を導入した場合は RTE API を使用する。

4.2.4 Complex Drivers の対応

Complex Drivers に相当する既存のソフトウェアの各種ドライバの対応について、以下観点より、既存のソフトウェアの各種ドライバをそのまま Complex Driver SW-C に置き換えるのはデメリットがある。

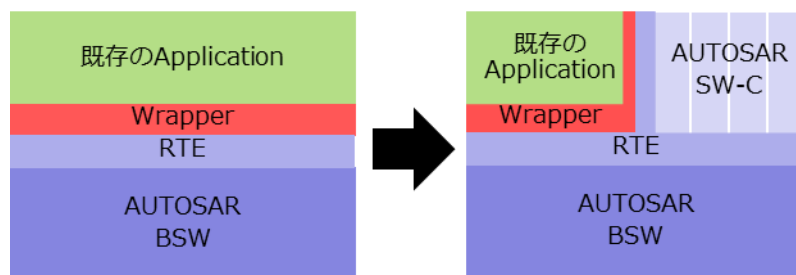
- ・ 既存のソフトウェアの各種ドライバがハードウェアの抽象化を完全にできていない場合がある。
(そのドライバを作成した時の背景やアーキテクチャの思想により、AUTOSAR のレイヤーの考え方と合わない可能性があるため。)
- ・ Complex Driver SW-C の定義や設計が必要になるため、システムディスクリプションファイルを作成しなければならないが、上記理由により、後でシステムディスクリプションファイルの作り直しが発生する可能性がある。

よって、Complex Drivers の対応はアプリケーションの SW-C 化と併せて対応する。



4.3 アプリケーションを一部 SW-C 化

4.3.1 概要



既存のアプリケーションの一部を AUTOSAR SW-C 化する場合について、ポイントとしては以下となる。

機能間連携のポイント

- ・ AUTOSAR SW-C 化したアプリケーションと既存のソフトウェアのアプリケーション間は Wrapper SW-C を経由して連携を行う。
- ・ 既存のソフトウェアのアプリケーションが AUTOSAR BSW の機能を使用する場合、RTE を経由せず使用することは可能。

機能分割のポイント

既存のアプリケーションの AUTOSAR SW-C への機能分割は、以下の理由からマイコン（もしくはコア）を跨ることがない機能単位が最小限とする。

- ・ SW-C の単位が小さいと RTE 経由の連携が多くなり、変更規模が増加し、さらに処理効率の低下やリソース消費量の増加にもつながる。
- ・ SW-C の単位が大きいと、SW-C を変更する際に分担が難しくなり開発効率が低下、また、ソフトウェアシステム設計の粒度が粗くなり設計の初期段階での検証レベルが低下することが考えられ、AUTOSAR の導入でもたらされるメリットが活かせない。

本文書では以下のパターンについての手順を説明する。

- ・ RTE を導入していない場合
- ・ すでに RTE を導入している場合

4.3.2 RTE を導入していない場合

4.3.2.1 AUTOSAR SW-C 後のソフトウェア構成

既存のアプリケーションの一部を AUTOSAR SW-C 化した後のソフトウェア構成を図 4-7 に記載する。

本導入検討では機能間の機能間／ECU 間の連携があり、また、Complex Drivers に相当する既存のソフトウェアの各種ドライバを使用しているアプリケーションを選定し、AUTOSAR SW-C 化を実施した。

AUTOSAR SW-C 化した制御

- ・ 操作制御
- ・ 制御管理

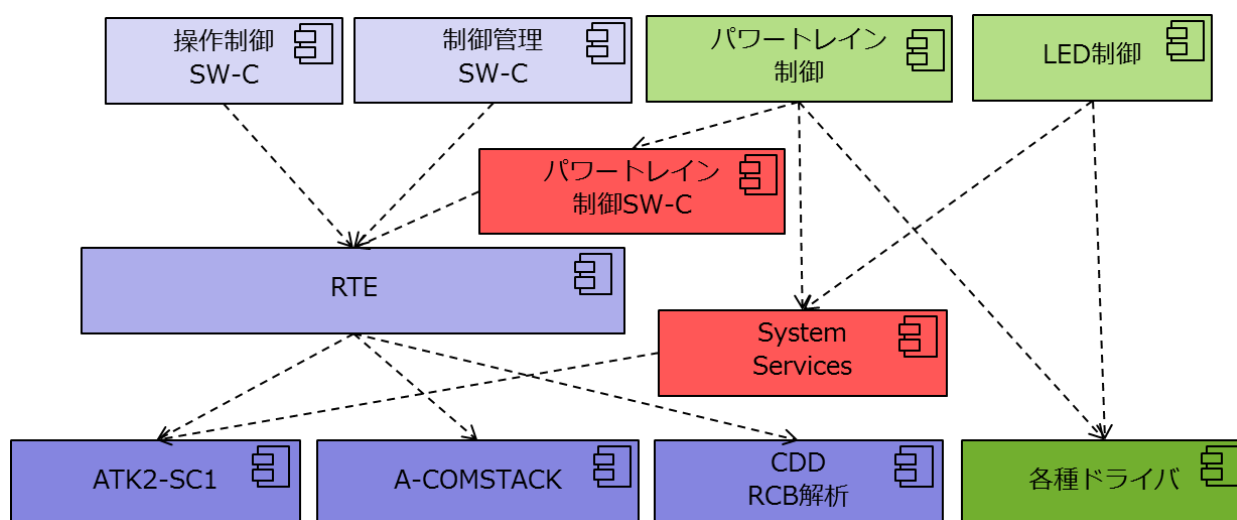


図 4-7 一部を AUTOSAR SW-C 化した後のソフトウェア構成

対応概要としては、以下となる。

- ・ 既存のソフトウェアのパワートレイン制御に影響を与えないよう、AUTOSAR SW-C 化した制御管理との結合部分はパワートレイン制御用の Wrapper SW-C を作成する。
- ・ AUTOSAR SW-C 化の対象となる以下ソフトウェアの全ての SW-C 設計を行い、システム／BSW モジュールディスクリプションファイルを作成する。
 - ・ AUTOSAR SW-C 化する制御自体
 - ・ AUTOSAR SW-C 化する制御と関係する Complex Drivers に相当する既存のソフトウェアの各種ドライバ
 - ・ AUTOSAR SW-C 化する制御と関係する既存のソフトウェア用の Wrapper SW-C

次章以降で各対応の詳細を記載する。

4.3.2.2 SW-C の設計

AUTOSAR SW-C 化の対象となるソフトウェアの操作制御，制御管理，その操作制御と関係する Complex Drivers の RCB 解析，パワートレイン制御用の Wrapper SW-C に対し，既存のソフトウェアの System Services や Communication Services で実行していた定期処置などや機能間連携／ECU 間連携を抽出し，SW-C の設計を行う．

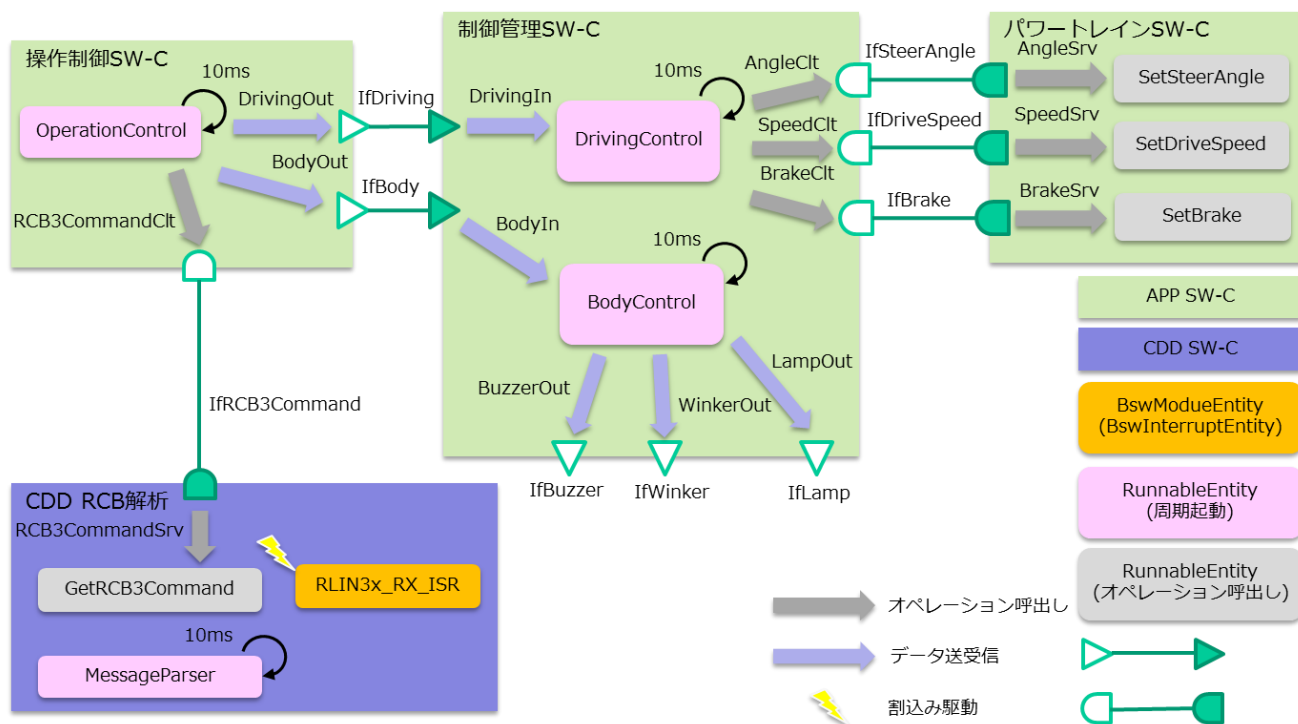


図 4-8 一部を AUTOSAR SW-C 化した後の SW-C 設計図

4.3.2.3 システム/BSW モジュールディスクリプションファイルの作成

SW-C の設計を元にシステム/BSW モジュールディスクリプションファイルを作成する。
本導入検討で行った概要を表 4-5、表 4-6 に記載する。

表 4-5 システムディスクリプションファイルの記述内容

対応項目	記述内容
SW-C の定義	<ul style="list-style-type: none">・ 操作制御 SW-C, 制御管理 SW-C, CDD RCB 解析, パワートレイン Wrapper SW-C で使用するランナブルを定義・ SW-C 間連携用のポートを定義・ ECU 間連携用のポートを定義 (他 ECU の部分は設計しなくても良い)・ RTE イベントを定義
インターフェースの定義	<ul style="list-style-type: none">・ SW-C 間連携用のインターフェースを定義・ ECU 間連携用のインターフェースを定義
ECU の定義	<ul style="list-style-type: none">・ 対象の ECU を定義 (その他 ECU の定義は不要)
ネットワークの定義	<ul style="list-style-type: none">・ 他 ECU との通信データ (SystemSignal) の定義・ iSignal の定義と SystemSignal への参照・ iSignal の PDU へのマッピング
各種マッピング	<ul style="list-style-type: none">・ SW-C の ECU へのマッピング・ SW-C のポート情報と通信データ (SystemSignal) のマッピング
その他	<ul style="list-style-type: none">・ ECU 構成に依存しないデータ型を定義

表 4-6 BSW モジュールディスクリプションファイルの記述内容

対応項目	記述内容
COM スタックの使用	<ul style="list-style-type: none">・ Can, CanIf, Com モジュールの BSW のエントリー, 排他エリア, RTE イベントを定義

4.3.2.4 SW-C の実装

各 SW-C の実装内容を記載する.

操作制御 SW-C ・ 制御管理 SW-C ・ CDD RCB 解析

- ・ 定期処理, SW-C 間のインターフェースをランナブルに変更する. (呼び出し箇所の変更)
- ・ SW-C 間のデータ送受信, 関数呼び出しを RTE API に変更する.
- ・ ECU 間通信データの送受信を RTE API に変更する.

パワートレイン Wrapper SW-C

- ・ SW-C の設計で定義したランナブル (SetSteerAngle, SetDriveSpeed, SetBrake) で既存のソフトウェアのパワートレイン制御の関数コールを実装する.

4.3.2.5 ECU コンフィギュレーション

AUTOSAR OS/COM スタックに加え RTE のコンフィギュレーションを行い, ジェネレータを実行しソースコードを生成する.

既存のソフトウェアの System Services の定期 Hook 内などで制約があった場合にはランナブル化することによって呼び出しタイミングが変わる可能性があるため, 各ランナブルの呼び出し優先度や順番のタイミング検証が必要となる.

4.3.3 すでに RTE を導入している場合

4.3.3.1 AUTOSAR SW-C 後のソフトウェア構成

既存のアプリケーションの一部を AUTOSAR SW-C 化した後のソフトウェア構成を図 4-9 に記載する。

本導入検討では機能間の機能間／ECU 間の連携があり、また、Complex Drivers に相当する既存のソフトウェアの各種ドライバを使用しているアプリケーションを選定し、AUTOSAR SW-C 化を実施した。

AUTOSAR SW-C 化した制御

- ・ 操作制御
- ・ 制御管理

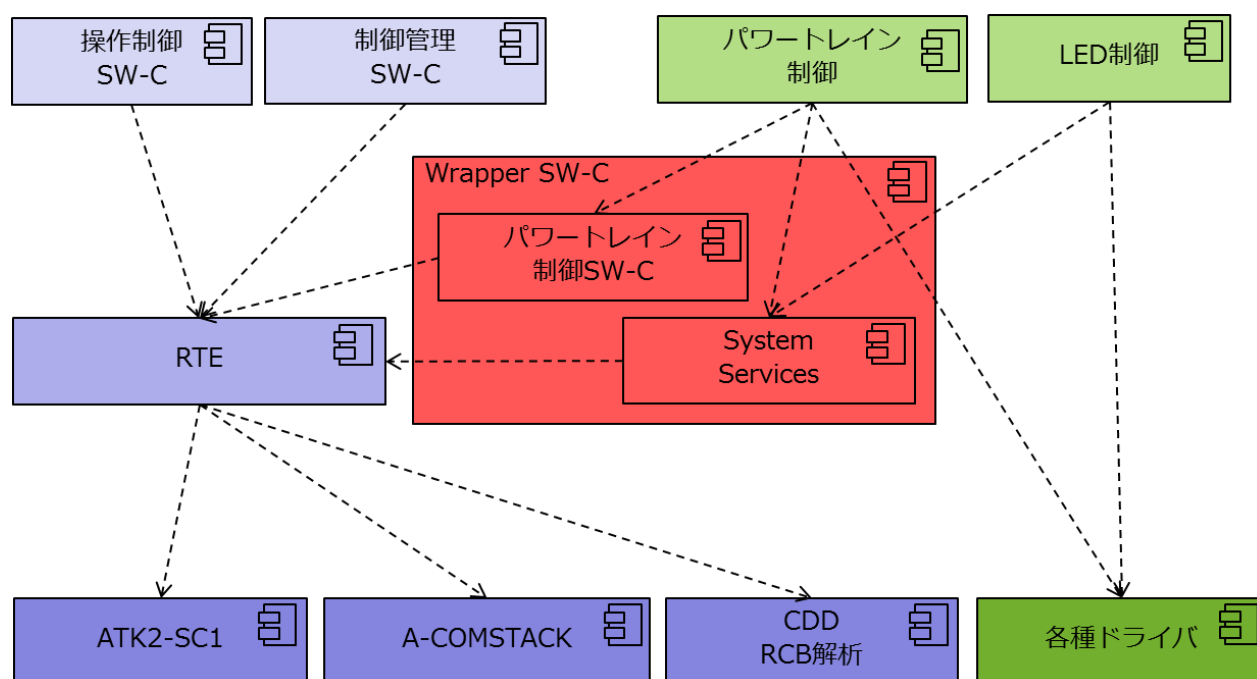


図 4-9 一部を AUTOSAR SW-C 化した後のソフトウェア構成

対応概要としては、以下となる。

- ・ 既存のソフトウェアのパワートレイン制御に影響を与えないよう、AUTOSAR SW-C 化した制御管理との結合部分はパワートレイン制御用の Wrapper SW-C を作成する。
- ・ AUTOSAR SW-C 化の対象となる以下ソフトウェアの全ての SW-C 設計を行い、システムディスクリプションファイルを変更する。
 - ・ AUTOSAR SW-C 化する制御自体
 - ・ AUTOSAR SW-C 化する制御と関係する Complex Drivers に相当する既存のソフトウェアの各種ドライバ

次章以降で各対応の詳細を記載する。

4.3.3.2 SW-C の設計

AUTOSAR SW-C 化の対象となるソフトウェアの操作制御，制御管理，その操作制御と関係する Complex Drivers の RCB 解析，パワートレイン制御用の Wrapper SW-C に対し，既存のソフトウェアの System Services や Communication Services で実行していた定期処置などや機能間連携／ECU 間連携を抽出し，SW-C の設計を行う．

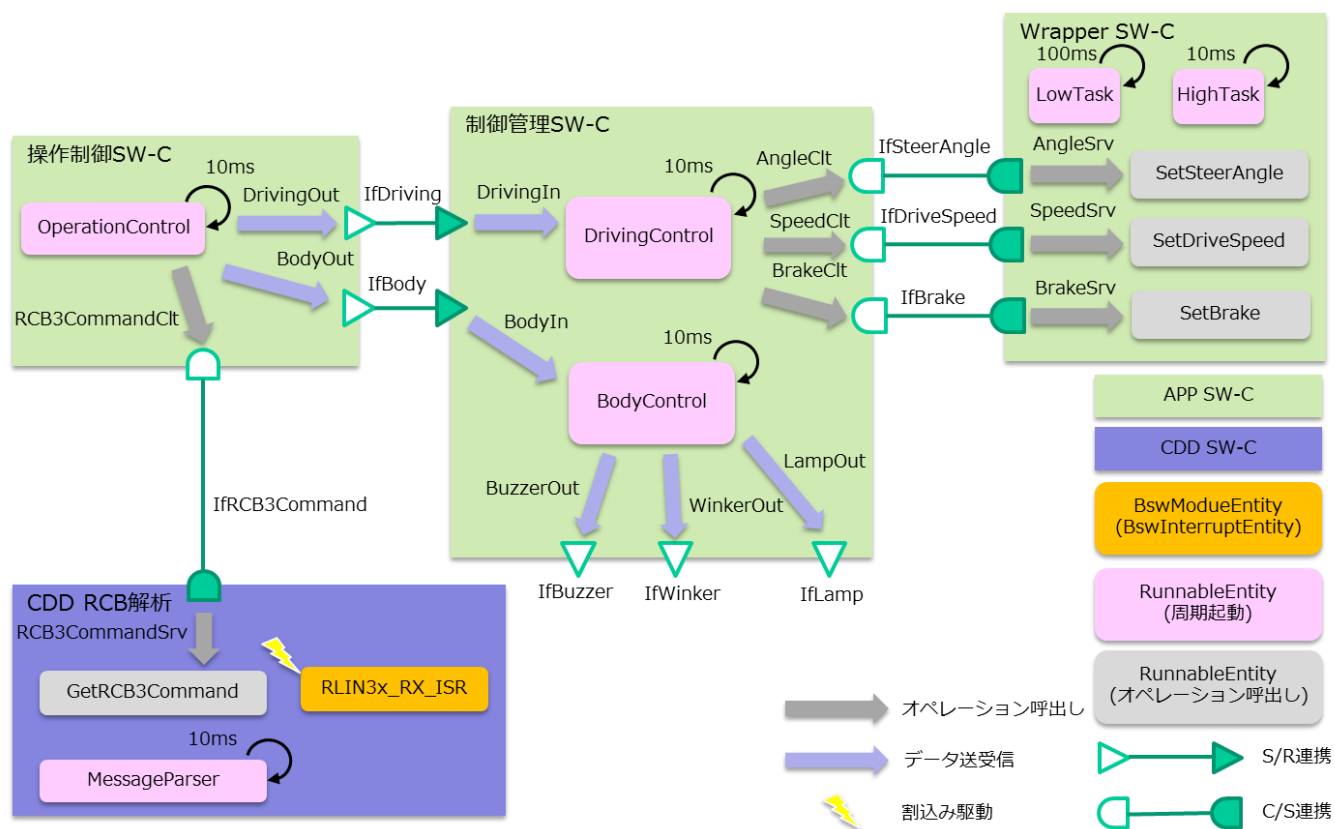


図 4-10 一部を AUTOSAR SW-C 化した後の SW-C 設計図

4.3.3.3 システム/BSW モジュールディスクリプションファイルの作成

SW-C の設計を元にシステムディスクリプションファイルの変更を行う。
本導入検討で行った概要を表 4-7 に記載する。

表 4-7 システムディスクリプションファイルの記述変更内容

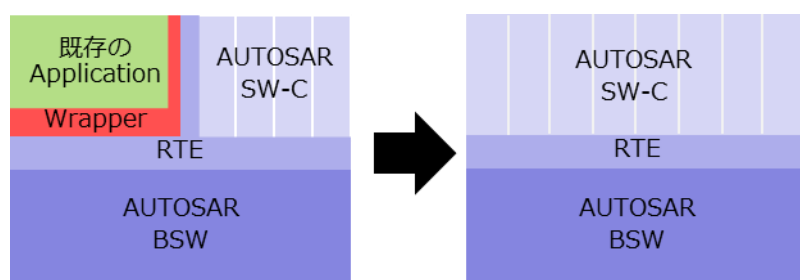
対応項目	記述内容
SW-C の定義	<ul style="list-style-type: none"> ・ 操作制御 SW-C, 制御管理 SW-C, CDD RCB 解析, パワートレイン Wrapper SW-C で使用するランナブルの定義を追加 ・ SW-C 間連携用のポートの定義を定義 ・ RTE イベントの定義を追加
インターフェースの定義	<ul style="list-style-type: none"> ・ SW-C 間連携用のインターフェースの定義を追加
各種マッピング	<ul style="list-style-type: none"> ・ SW-C の ECU へのマッピング ・ SW-C のポート情報と通信データ (SystemSignal) のマッピング
その他	<ul style="list-style-type: none"> ・ ECU 構成に依存しないデータ型を追加

4.3.3.4 SW-C の実装と ECU コンフィギュレーション

各 SW-C の実装内容と ECU コンフィギュレーションについては、「4.3.2 RTE を導入していない場合」と同様の内容となる。

4.4 アプリケーションを全 SW-C 化

4.4.1 概要



既存のアプリケーションを全て AUTOSAR SW-C 化する場合について、「4.3 アプリケーションを一部 SW-C 化」に記載した手順と同じ手法で順次 AUTOSAR SW-C 化を実施する。

SW-C 化の対応内容

- ・ SW-C への機能分割
- ・ SW-C の設計
- ・ システム/BSW モジュールディスクリプションファイルへの定義追加
- ・ SW-C の実装
- ・ ECU コンフィグレーションへの定義追加（各ランナブルの呼び出し優先度や順番のタイミング検証も必要）

4.4.2 AUTOSAR SW-C 後のソフトウェア構成

既存のアプリケーションの全て AUTOSAR SW-C 化した後のソフトウェア構成を図 4-11 に記載する。

AUTOSAR SW-C 化した制御

- ・ パワートレイン制御
- ・ LED 制御

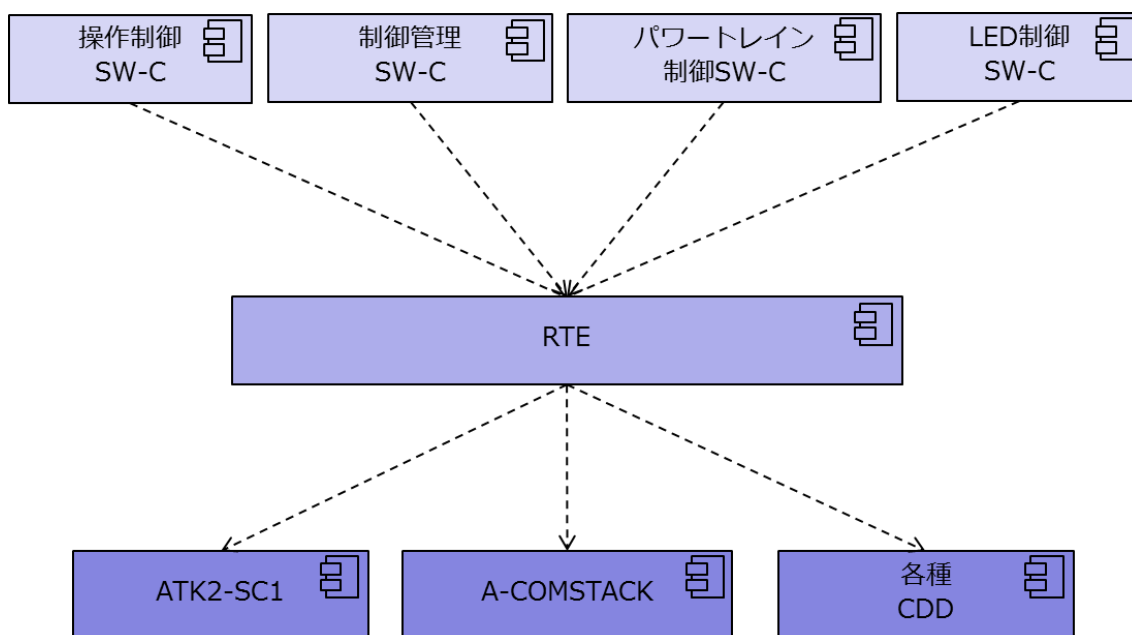


図 4-11 全て AUTOSAR SW-C 化した後のソフトウェア構成

4.4.3 SW-C の設計

既存のアプリケーションの全て AUTOSAR SW-C 化した後の SW-C の設計図を図 4-12 に記載する。

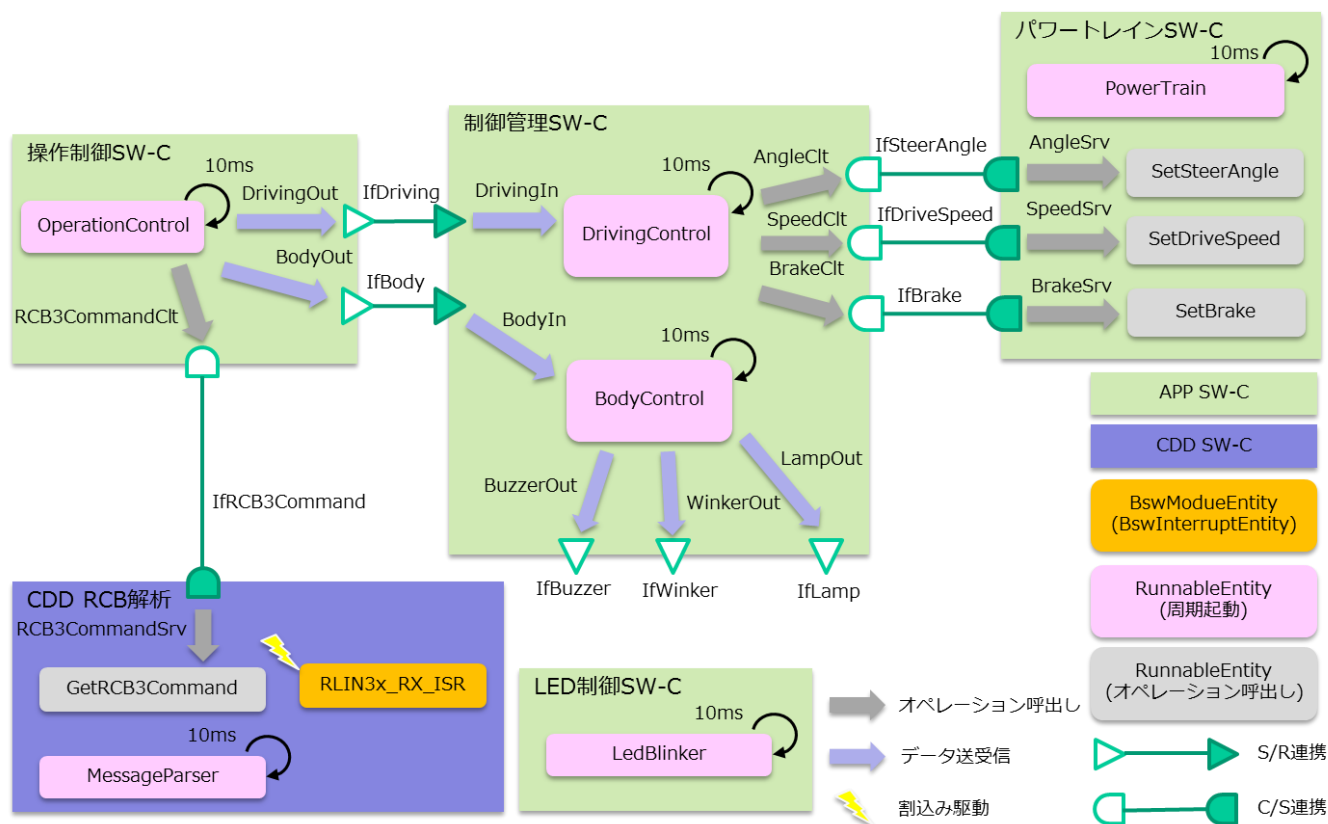


図 4-12 全て AUTOSAR SW-C 化した後の SW-C 設計図

※本導入検討時の未対応事項

パワートレイン SW-C と LED 制御 SW-C と関係する Complex Drivers の設計について、本導入検討では省略した。

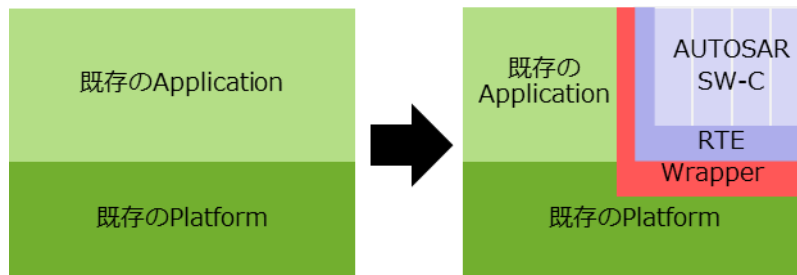
また、実装において、TOPPERS プロジェクトから公開されているモデルカー制御プログラムを流用したことから、パワートレイン SW-C と LED 制御 SW-C は Complex Drivers としてシステムディスクリプションファイルには定義されている。

5. アプリケーション部分からの導入

ECU1（制御系，操作系）の既存のソフトウェアに対しアプリケーション部分から AUTOSAR の導入を実施する場合の手順や要検討事項を記載する。

5.1 アプリケーションを一部 SW-C 化

5.1.1 概要



既存のアプリケーションの一部を AUTOSAR SW-C 化する場合について、ポイントとしては以下となる。

機能間連携のポイント

- ・ AUTOSAR SW-C 化したアプリケーションと既存のソフトウェアのアプリケーション間は Wrapper SW-C を経由して連携を行う。
- ・ Complex Drivers に相当する既存のソフトウェアの各種ドライバの対応についてはそのドライバを使用するアプリケーションの SW-C 化と併せて対応する。（理由は 4.2.4 を参照）
- ・ AUTOSAR SW-C 化したアプリケーションが必要とする AUTOSAR BSW の機能は、AUTOSAR BSW が存在するものとして RTE を生成し、AUTOSAR BSW の Wrapper ソフトウェアを作成する。

機能分割のポイント

「4.3 アプリケーションを一部 SW-C 化」の「4.3.1 概要」に記載した「機能分割のポイント」同じとなる。

既存のアプリケーションを AUTOSAR SW-C 化する対応内容は、「4.3 アプリケーションを一部 SW-C 化」に記載した手順と同じ手法で実施する。

SW-C 化の対応内容

- ・ SW-C への機能分割
- ・ SW-C の設計
- ・ システム／BSW モジュールディスクリプションファイルへの定義追加
- ・ SW-C の実装
- ・ ECU コンフィグレーションへの定義追加（各ランナブルの呼び出し優先度や順番のタイミング検証も必要）

本節では以下のポイントについての説明を記載する。(以下以外のポイントについては「4 プラットフォーム部分からの導入」と同じ内容となるため省略)

- ・ RTE の自動生成

5.1.2 AUTOSAR SW-C 後のソフトウェア構成

既存のアプリケーションの一部を AUTOSAR SW-C 化した後のソフトウェア構成を図 5-1 に記載する。

本導入検討では「4.3 アプリケーションを一部 SW-C 化」と同様に、機能間の機能間／ECU 間の連携があり、また、Complex Drivers に相当する既存のソフトウェアの各種ドライバを使用しているアプリケーションを選定し、AUTOSAR SW-C 化を実施した。

AUTOSAR SW-C 化した制御

- ・ 操作制御
- ・ 制御管理

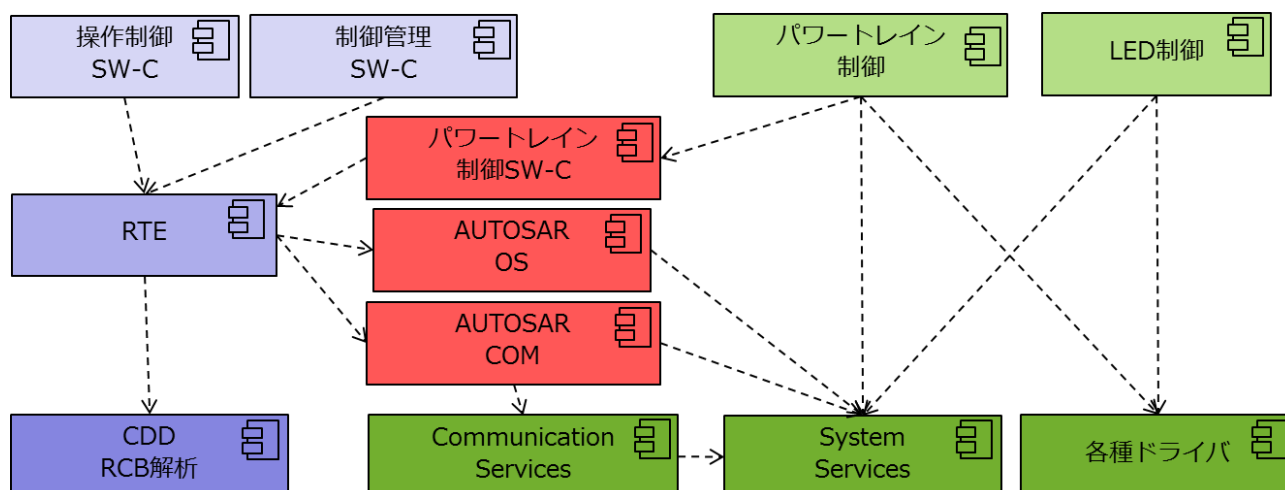


図 5-1 一部を AUTOSAR SW-C 化した後のソフトウェア構成

操作制御や制御管理には OS タスクや ECU 間連携などが必要となるため、AUTOSAR OS/Com モジュールの Wrapper ソフトウェアを作成する。

5.1.3 RTE の生成

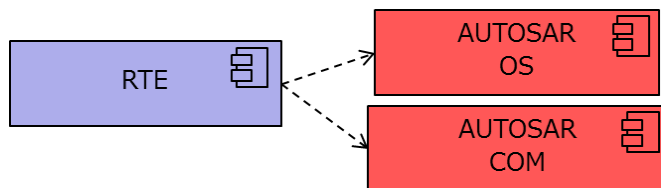
AUTOSAR SW-C は RTE に依存した関係となるため、RTE の生成方法の推奨案とその他の案について記載する。

推奨案

本導入検討ではこの案を採用。

AUTOSAR OS/Com モジュールが存在するものとして RTE を生成し、AUTOSAR OS/Com モジュールの Wrapper ソフトウェアを作成し、RTE と Wrapper ソフトウェアを接続する。実際の必要となる AUTOSAR OS/Com モジュールの機能は既存のソフトウェアを使用して同等の機能を実現する。

Wrapper ソフトウェアを作成すること以外は「4.3.2 RTE を導入していない場合」と同じ手順での AUTOSAR SW-C 化を実施する。



<RTE のコンフィギュレーション例の一部>

SWCI_DriveManager:

DefinitionRef: RteSwComponentInstance

RteSoftwareComponentInstanceRef: /RcCar/Composition_RcCar/DriveManager

IB_DM_TimingEvent_DrivingControl_10ms:

DefinitionRef: RteEventToTaskMapping

RteActivationOffset: 0.0

RtePositionInTask: 5

RteEventRef: /RcCar/DriveManager/IB_DM/DrivingControl_timingEvent

RteMappedToTaskRef: /Ecu_CO/Os/ECU_CO_TASK_10ms

RteUsedOsAlarmRef: /Ecu_CO/Os/ECU_CO_ALARM_10ms

<Com モジュールのコンフィギュレーション例の一部 (RTE 用) >

ComSignal0:

DefinitionRef: ComSignal

ComHandleId: 0

ComBitPosition: 0

ComBitSize: 8

ComSignalType: UINT8

ComTransferProperty: TRIGGERED_ON_CHANGE

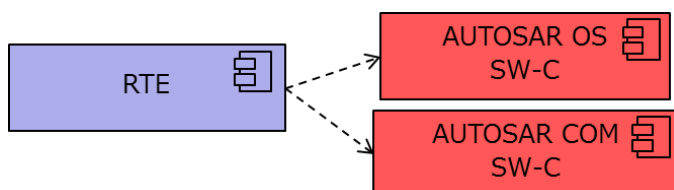
ComSignalEndianness: LITTLE_ENDIAN

ComSystemTemplateSystemSignalRef:

/Network/ISignalIPdu_Winker/iSignalToIPduMapping_Winker_state

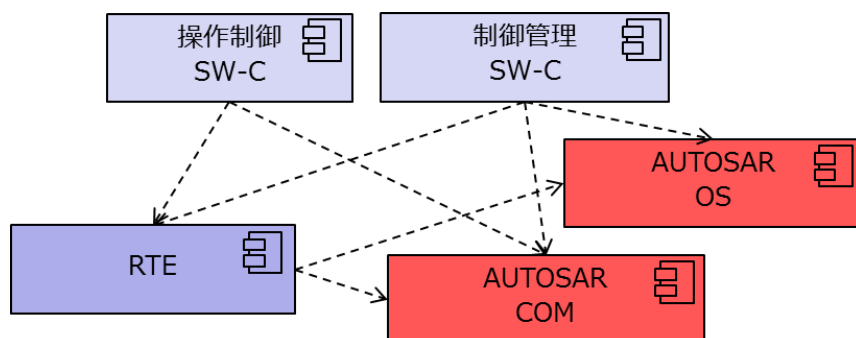
その他の案

- 既存のソフトウェアとの接続部分をすべて SW-C として RTE を生成する.



AUTOSAR OS/Com モジュールの Wrapper ソフトウェアも SW-C として扱うため、システムディスクリプションファイルの定義が必要となり、また、AUTOSAR OS/Com モジュールを導入した際にそのシステムディスクリプションファイルの定義の削除が必要となるデメリットがある。

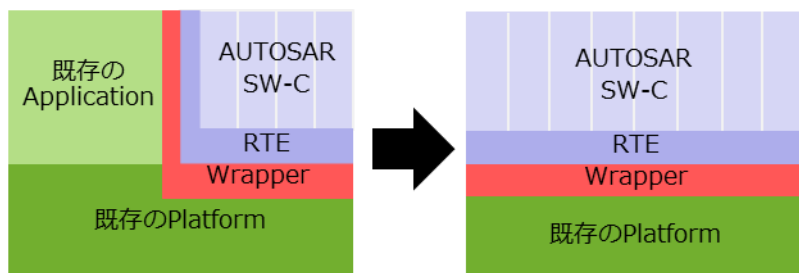
- SW-C 間の連携のみ RTE の自動生成を利用し、AUTOSAR OS/Com モジュールとの接続部分は実装を行う。



実装を行う部分については、AUTOSAR のメリットのひとつである「自動コード生成の利用」が活用できないデメリットがある。

5.2 アプリケーションを全 SW-C 化

5.2.1 概要



既存のアプリケーションを全て AUTOSAR SW-C 化する場合について、「5.1 アプリケーションを一部 SW-C 化」に記載した手順と同じ手法で順次 AUTOSAR SW-C 化を実施する。

SW-C 化の対応内容

- ・ SW-C への機能分割
- ・ SW-C の設計
- ・ システム/BSW モジュールディスクリプションファイルへの定義追加
- ・ SW-C の実装
- ・ ECU コンフィグレーションへの定義追加（各ランナブルの呼び出し優先度や順番のタイミング検証も必要）

5.2.2 AUTOSAR SW-C 後のソフトウェア構成

既存のアプリケーションの全て AUTOSAR SW-C 化した後のソフトウェア構成を図 5-2 に記載する。

AUTOSAR SW-C 化した制御

- ・ パワートレイン制御
- ・ LED 制御

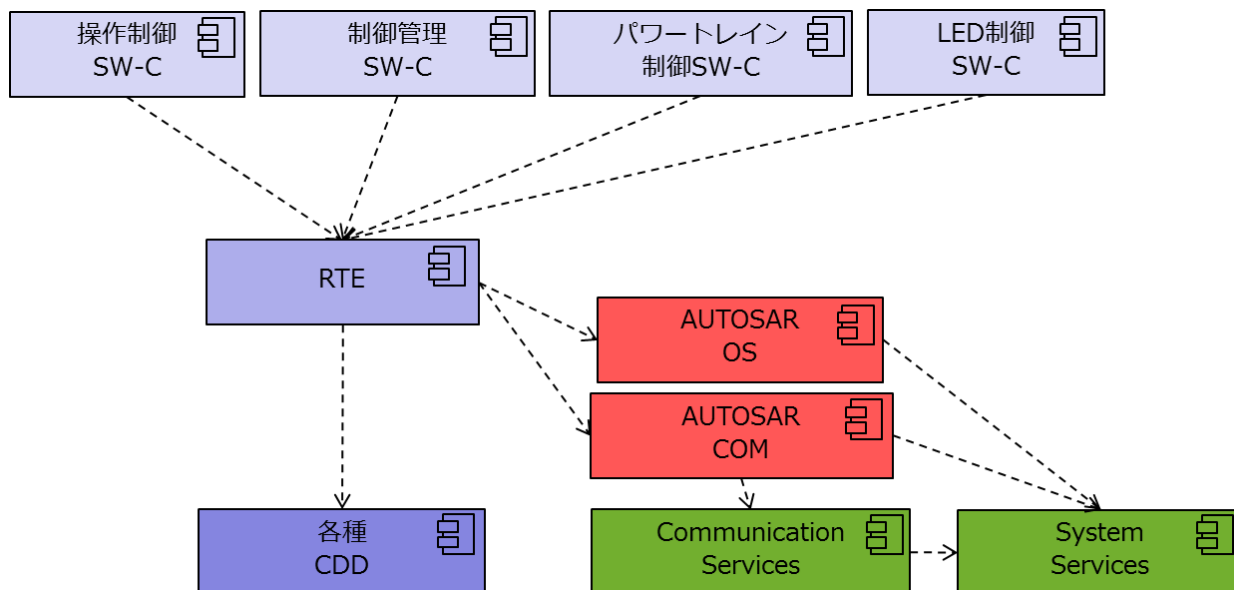


図 5-2 全て AUTOSAR SW-C 化した後のソフトウェア構成

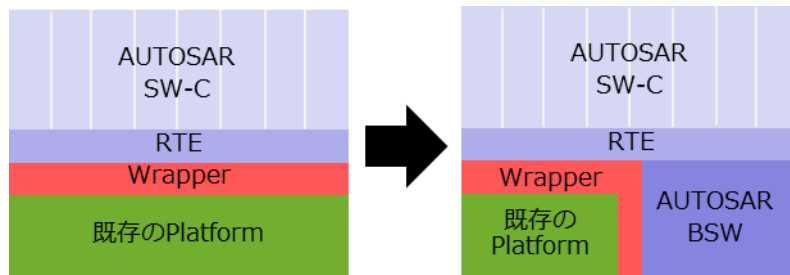
※本導入検討時の未対応事項

パワートレイン SW-C と LED 制御 SW-C と関係する Complex Drivers の設計について、本導入検討では省略した。

また、実装において、TOPPERS プロジェクトから公開されているモデルカー制御プログラムを流用したことから、パワートレイン SW-C と LED 制御 SW-C は Complex Drivers としてシステムディスクリプションファイルには定義されている。

5.3 AUTOSAR BSW の一部導入

5.3.1 概要



既存のプラットフォームの一部を AUTOSAR BSW に置き換える場合について、「4 プラットフォーム部分からの導入」の「4.1.2 始めに OS を導入する場合」に記載した手順とほぼ同じ対応内容となる。

本文書では以下のパターンについての手順を説明する。

- OS から導入する場合
- COM スタックから導入する場合

5.3.2 OS から導入する場合

5.3.2.1 導入後のソフトウェア構成

OS の導入後のソフトウェア構成を図 5-3 に記載する。

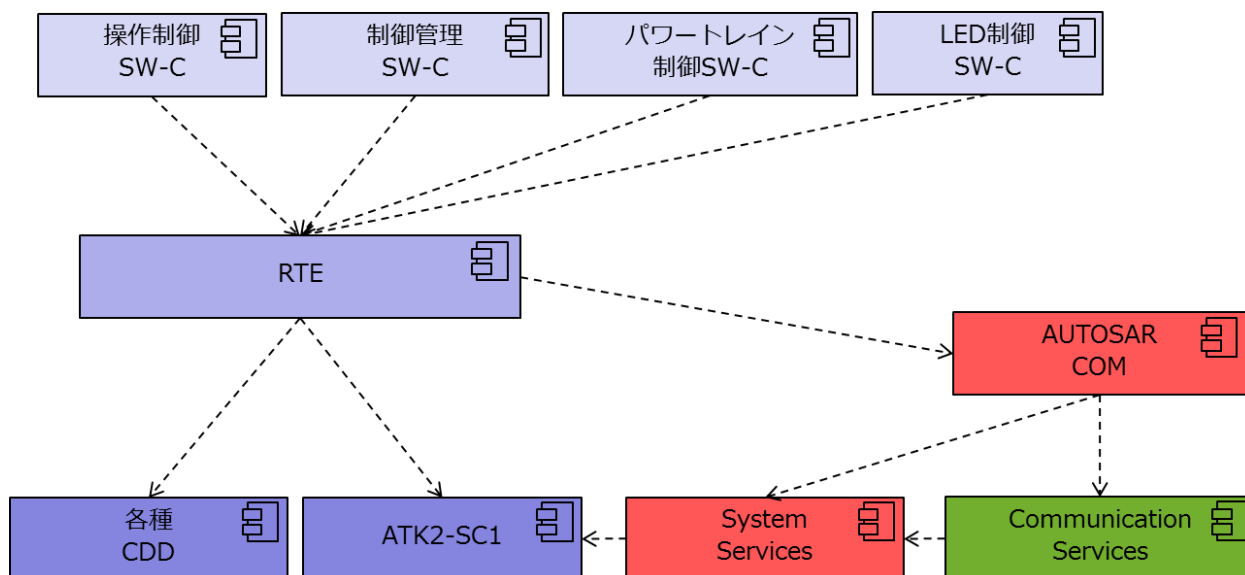


図 5-3 OS 導入後のソフトウェア構成

対応内容

- ・ スタートアップルーチンの実行
- ・ AUTOSAR BSW に置き換えていない既存のプラットフォームに対する対応
 - ・ Hook の提供（初期化／定期）
 - ・ 割り込み処理のサポート
 - ・ 割り込み禁止／解除用 IF の提供
- ・ 各種 OS フックルーチン（シャットダウンフック／エラーフックなど）を必要に応じて実装

5.3.3 COM スタックから導入する場合

5.3.3.1 導入後のソフトウェア構成

COM スタックの導入後のソフトウェア構成を図 5-4 に記載する。

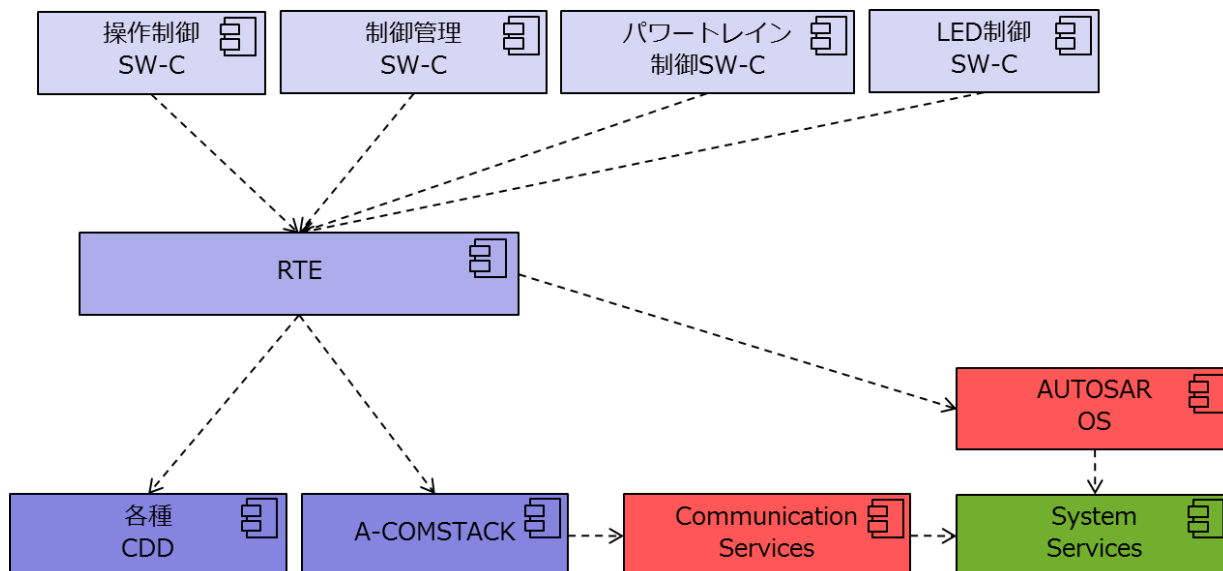


図 5-4 COM スタック導入後のソフトウェア構成

対応内容

- ・ Can, CanIf, Com モジュールの初期化处理の追加
- ・ AUTOSAR BSW に置き換えていない既存のプラットフォームに対する対応
 - ・ 信号の送受信 IF の提供
- ・ 他の AUTOSAR BSW モジュールとの関連箇所の対応
- ・ スタブの作成
- ・ AUTOSAR OS に依存する箇所への対応

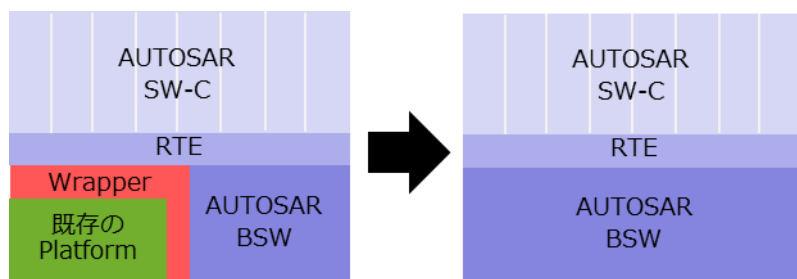
5.3.3.2 プラットフォーム部分から導入する場合との相違点

プラットフォーム部分からの導入時に COM スタックを導入した場合との相違点を記載する。

- ・ BSW モジュールディスクリプションファイルの作成。（対応内容は「表 4-4 BSW モジュールディスクリプションファイルの記述内容」と同じ）
- ・ Can, CanIf, Com モジュールの定期処理の追加。（RTE で自動生成するため）

5.4 AUTOSAR BSW の全導入

5.4.1 概要



既存のプラットフォームを全て AUTOSAR BSW に置き換える場合について、「5.3 AUTOSAR BSW の一部導入」記載した手順と「AUTOSAR BSW に置き換えていない既存のプラットフォームに対する対応」を除き、ほぼ同じ対応内容となる。

5.4.2 導入後のソフトウェア構成

OS と COM スタックの導入後のソフトウェア構成を図 5-5 に記載する。

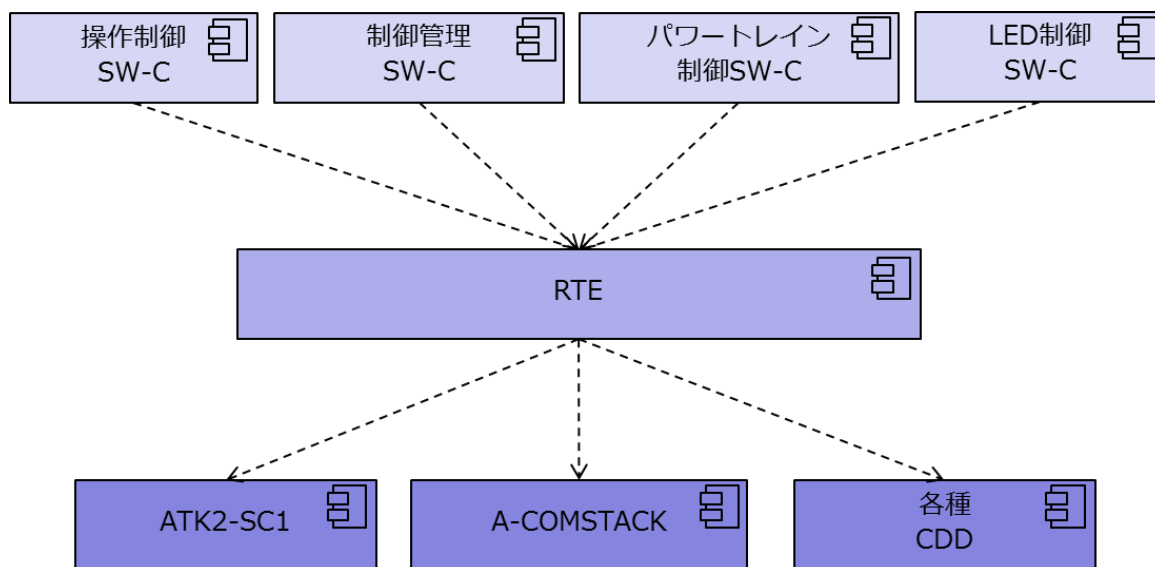


図 5-5 OS と COM スタック導入後のソフトウェア構成

変更履歴

Version	Date	Detail	Editor
1.0.0	2017/03/15	・ 新規作成	NCES
		・	