<AUTOSAR RTE ソフトウェア>

RTE リリースノート

（ver1.1.0）

1. はじめに

本製品は､AUTOSAR RTE ソフトウェア機能仕様に準拠した RTE モジュールを提供する。RTE モジュールは、AUTOSAR メタモデルを利用した設計データに応じた実行コードとして実現することが求められることから、RTE モジュールを生成するツール「RTE ジェネレータ」として提供を行う。

1.1. 前提文書

前提となるドキュメントを以下に記載する。

[1] RTE モジュール仕様
・JASPAR 基盤ソフトウェア RTE ソフトウェア機能仕様書 Ver 1.0

1.2. 用語定義

ID	用語	正式表記	意味
1	パーティション	Software Partition	1 つ以上のソフトウェアモジュールが格納される区画
2	モジュール、Module	Software Module	対象環境向けに実装されたソフトウェアモジュール
5	RTE	Runtime Environment	JASPAR 基盤ソフトウェアにおけるソフトウェア部品の実行基盤環境
10	BSW	Basic Software	JASPAR 基盤ソフトウェアにおける ECU プラットフォームの基盤機能を提供するモジュール群
11	ランナブル、Runnable	Runnable Entity	AUTOSAR 規格におけるソフトウェア部品の動作単位
12	SchM	Schedule Manager	RTE の一機能として提供される BSW の実行スケジューラ機能
13	BswSchedulable Entity	Bsw Schedulable Entity	BSW の実行スケジューラにおけるスケジュールの実行単位
14	OS	Operating System	JASPAR 基盤ソフトウェアにおける OS モジュール
15	COM	COM	JASPAR 基盤ソフトウェアにおける ECU 間通信モジュール

2. リリース機能

以下の仕様に対応した RTE ジェネレータツールをリリースする。

対応仕様	JASPAR 基盤ソフトウェア RTE ソフトウェア機能仕様 Ver 1.0 補足： 本仕様は、AUTOSAR R4.0 rev002 をベースとして、JASPAR で定められた機能安全要件を満たすサブセットとして規定されたものである。

2.1. RTE ジェネレータツール

本ツールでは、RTE モジュール、および SchM モジュールの生成機能をサポートする。

【補足】
RTE 上で動作する各コンポーネントのヘッダファイルのみを生成する機能は提供しない。代わりに RTE モジュール生成時に生成されるコンポーネント用ヘッダファイルを用いることで代替することができる。

2.2. RTE モジュール

本ツールで生成される RTE モジュールは、JASPAR 基盤ソフトウェア RTE ソフトウェア機能仕様をサポートする。
提供機能の概要は以下のとおり。

■RTE モジュール
周期スケジューリングで動作するアプリケーションに必要となる機能セットを提供する。

機能分類	説明
パーティショニング	・ソフトウェアコンポーネントのパーティショニングをサポートする ※利用可能なパーティション数は OS の利用可能リソース数に依存する。詳細は、OS 仕様を参照のこと。
ソフトウェア コンポーネント	・シングルインスタンス生成のみサポートする ・サポートするコンポーネント種別は以下のとおり。 - ApplicationSoftwareComponentType, - SensorActuatorComponentType - ComplexDeviceDriverComponentType - ECUAbstractionComponentType
コンポーネント間連携	・Sender-Receiver 連携（ECU 内／ECU 間）をサポートする。 ・Client-Server 連携（ECU 内の同期連携）をサポートする。

機能分類	説明
	・トリガ連携（外部／内部トリガ）をサポートする。 ・モード連携（非同期モード遷移手順）をサポートする。
ランナブル動作トリガ	以下の起動イベントをサポートする。 ・周期イベント - TimingEvent (OsAlarm で起動) ・Client-Server 連携に伴うサーバ呼び出しイベント - OperationInvokedEvent ・モード連携に伴うモード切替イベント（※１） - ModeSwitchEvent また、API によるランナブル起動（外部／内部イベント）をサポートする。

※１：モードの遷移（OnEntry）契機の起動のみをサポート

■SchM モジュール

BSW を構成する各基盤モジュールを周期スケジューリングで起動するために必要となる機能セットを提供する。

機能分類	説明
BswSchedulableEntity 動作トリガ	・周期イベント - TimingEvent (OsAlarm で起動) ・モード連携に伴うモード切替イベント（※１） - ModeSwitchEvent

※１：モードの遷移（OnEntry）契機の起動のみをサポート

3. 動作環境

以下では、ツールおよびモジュールの動作環境について記載する。

3.1. ツール動作環境

RTE ジェネレータツールを利用する場合は、以下のソフトウェアが必要になる。

１）Java SE Development Kit 6 以降

参考：

2012/12 月時点では、下記 URL から取得可能。

http://java.sun.com/javase/ja/6/download.html より「JDK6 Update3」を選択

3.2. RTE モジュール動作環境

RTE ジェネレータツールから生成される RTE モジュール（ソースコード）は、特定のマイコンに依存したコードを利用しておらず、32Bit の各種マイコンで動作可能である。

ただし、JASPAR 基盤ソフトウェアに対する機能安全要件の検証環境として、対象マイコンが定められていることから、ハードウェア環境の前提を規定する。

■ ハードウェア環境
SH7227 マイコン環境を対象とする。

■ 開発環境
RTE モジュール（ソースコード）を利用する場合は、ハードウェア環境に応じて以下のものを準備する必要がある。

1） マイコン向け開発環境
例：SH7227 マイコン向けの場合：HEW

2） BSW モジュール
RTE モジュールの利用にあたり、JASPAR 基盤ソフトウェアとして定められた下記モジュールが必要となる。
・COM モジュール
・OS モジュール

4. 注意事項

4.1. RTE モジュール利用時の注意事項

以下では、RTE モジュールのインテグレーションに関連する注意事項を記載する。

4.1.1. RTE の初期化・終了処理

RTE の初期化、および終了を行う場合は、以下の RTE API を利用して、インテグレータ側で処理を実装する必要がある。

分類	対象	備考
ヘッダファイル	Rte_Main.h	RTE モジュール生成時に作成される。
API	Std_ReturnType Rte_Start(void)	RTE の初期化
	Std_ReturnType Rte_Stop(void)	RTE の終了処理

[参考]
RTE の動作基盤となる OS モジュールの初期化・終了処理は、以下の API を利用して行う。

分類	対象	備考
ヘッダファイル	Os.h	OS モジュールから提供される。
API	void StartOS(AppModeType Mode)	OS モジュールの初期化と開始
	void ShutdownOS(StatusType Error)	OS モジュールの終了

■RTE の初期化手順例
RTE の初期化が実行されていない場合は、RTE API を呼び出した際の処理結果は保障されない。インテグレータが適切な手段で RTE の初期化処理を実装する必要がある。
参考のため、RTE の初期化処理例として 2 パターンの実装例を記載する。

A）OS のスタートアップタスクを利用した初期化
OS のコンフィグレーションで、OS の開始（StartOS API 呼び出し）時に動作するタスクを設定する。RTE の初期化は、OS 起動時の動作タスク内から行う。

```
/*
 * OS スタートアップ時に一度だけ動作するタスク
*/
TASK(StartupTask){
        :
    /* RTE の初期化 */
    Rte_Start();
        :
  }
```

B) ECU のスタートアップルーチンを利用した初期化

ECU 上で動作する実行プログラムのスタートアップルーチンから RTE の初期化を実行する。

```
#include “Os.h”
/*
 * アプリケーションのスタートアップ処理
 */
int main() {
        :
    /* アプリケーションのデータ領域、利用デバイスの初期化 */
        :
    /* RTE の初期化 */
    Rte_Start();
        :
    /* 実行制御モジュール（AUTOSAR OS）の開始 */
    StartOs(APPMODE);
        :
}
```

■RTE の終了処理

RTE を明示的に停止する場合は、終了処理の割り込みハンドラ等を実装し、ハンドラ内から Rte_Stop 関数を呼び出す処理を定義する。

【補足】AUTOSAR 規格における ECU の初期化／終了処理

・AUTOSAR では、ECU State Manager から ECU の停止／開始、および COM、OS の初期化や終了通知を受け取り RTE の初期化／終了処理を行う仕様となる。

ECU State Manager または、ECU の状態通知機能を利用せずに ECU の実行プログラムを構成する場合は、上記方法で初期化／終了処理を実現する必要がある。

4.1.2. コンポーネントの初期化・終了処理

RTE 上で動作する各種ソフトウェアコンポーネントの初期化、および終了を行う場合は、インテグレータ側で処理を実装する必要がある。

参考のため、ソフトウェアコンポーネントの初期化処理例を記載する。

例：OS のスタートアップタスクを利用した初期化

```
/*
 * OS スタートアップ時に一度だけ動作するタスク
*/
TASK(StartupTask){
        :
    /*  コンポーネントの実装に利用している変数を初期化  */
    Component1_internalValue = 0;
        :
    /* RTE の初期化  */
    Rte_Start();
        :
}
```

RTE の初期化に伴い、RTE API を経由して取得するデータ値も初期化が実行されるため、考慮する必要はない。コンポーネントの実装で追加した独自の変数等が、コンポーネントの初期化に必要な処理内容となる。

また、Rte_Start API が復帰した場合、RTE 内のランナブルは動作可能となることから、Rte_Start API を呼び出す前にコンポーネントの初期化処理を定義する必要がある。

終了処理も同様に、RTE の終了処理の後に各コンポーネントの終了処理を定義することで実現する必要がある。

4.2. ECU コンフィグレーションに関する注意事項

以下では、RTE モジュール、および関連する BSW モジュールのコンフィグレーションに関連する注意事項を記載する。

4.2.1. COM モジュールのコールバック設定

【RTE ジェネレータ仕様】
RTE モジュールは、以下のコールバック関数を利用して通信層からの通知を受け取る。

分類	COM コールバック <sn >:COM Signal 名 <ipdu>:COM Signal が格納される PDU 名	COM からの通知内容
データ送信	Rte_COMCbkTAck_<sn >	データの送信成功
	Rte_COMCbkTErr_<sn >	データの送信失敗（COM エラー）
	Rte_COMCbkTxTOut_<sn >	データの送信失敗（タイムアウト）
データ受信	Rte_COMCbk_<sn >	データの受信
	Rte_COMCbkInv_<sn>	データの無効化受信
	Rte_COMCbkRxTOut_<sn>	データの受信失敗（タイムアウト）
	Rte_COMCbkCounterErr_< ipdu > (※1)	データ到達順序の異常検出

(※1) COM モジュールからのコールバック通知 IF に合わせ、Rte_COMCbkCounterErr のみ IF に引数を持つ。 Rte_COMCbkCounterErr_<ipud> (PduIdType, uint8, uint8)

■注意事項
RTE モジュールが COM モジュールからのコールバック通知を正しく解釈するためには上記命名側に即した COM コンフィグレーションが行われている必要がある。

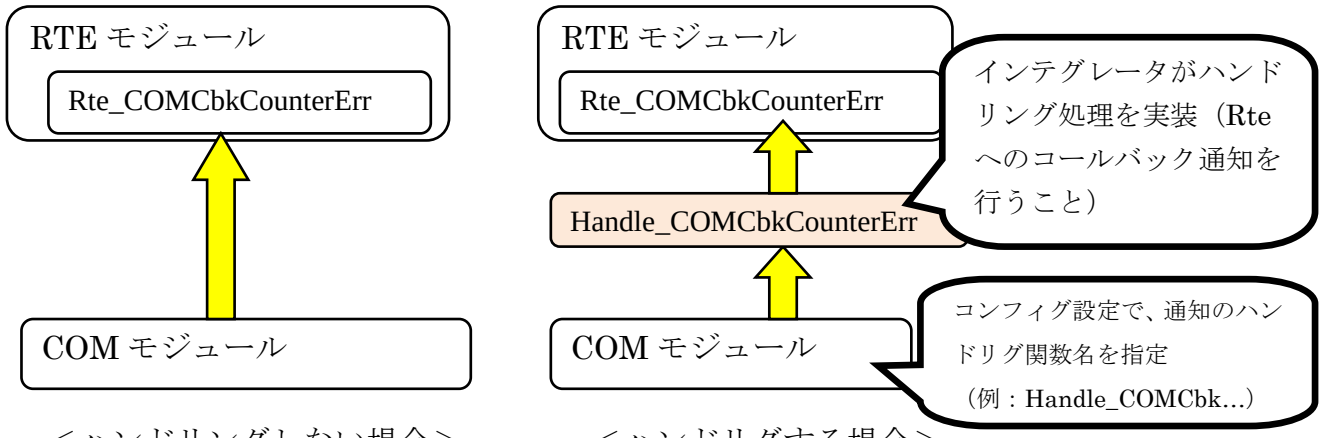
なお、RTE は、COM のコンフィグレーションにおいて、ComIPduCounter が定義された PDU に含まれる COM シグナルについては、データ到達順序の異常通知を受け取るための Rte_COMCbkCounterErr コールバックが有効になることを期待する。

■参考：データ到達順序の異常検出をトリガとした処理の実装方法について

データ到達順序の異常検出をハンドリングして処理するためには、COM モジュールのコンフィグレーション設定をカスタマイズし、到達順序の異常を通知するコールバック関数名をハンドリング処理関数の名前に設定する。

この設定により、COM がデータ到達順序の異常を検出した場合は、ハンドリング関数が呼び出されるようになるため、異常検出をトリガとした処理を定義することができる。

○データ到達順序の異常検出ハンドリングの動作イメージ



4.2.2. ランナブルの動作コンテキスト設定

【RTE ジェネレータ仕様】
本 RTE ジェネレータツールは、種別が ECU Abstraction／ComplexDeviceDriver で定義されたコンポーネントに含まれるランナブルは、RTE コンフィグレーションで OS タスクが割り当てられない場合に、タスク以外のコンテキストで動作するものとして扱う仕様である。

タスク以外のコンテキストで動作する扱いとなるランナブルについては、RTE モジュールからの呼び出しは行われない。

■注意事項
該当するランナブルを実行するためにはインテグレータが、以下の方法でランナブルの呼び出し処理を実装する必要がある。
・外部／内部トリガ API を用いてランナブルを起動する。
・ランナブルのエントリポイントとなる関数を直接呼び出す

例）ランナブルの呼び出し処理の定義例

```
/ *
 * 定義した割り込みハンドラ（SomeIsr）から、ランナブルのエントリポイント
 * を呼出すことで実行する場合の実装例
 */
ISR(SomeIsr)
{
    runnable_on_isr2();
}
```

4.2.3. パーティション間通信の OS 設定

【RTE ジェネレータ仕様】
本 RTE ジェネレータツールが生成する RTE モジュールは、ECU 内に定義される 各 Trusted Partition 毎に、1 つの TrustedFunction を生成する仕様である。
（本 TrustedFunction は、RTE の内部処理やパーティション間の通信に用いられる）。

パーティション間通信のための TrustedFunction のシンボル名は以下の仕様となる。
TRUSTED_Rte_Function_<pid>（<pid> は EcucPartition の ShortName 値）
※ EcucPartition は、ECU コンフィグレーションで定義されたパーティション要素

■注意事項
Trusted Partition を利用した構成で、RTE モジュールを利用する場合は、OS モジュールに対して以下の 2 点のコンフィグレーションを行う必要がある。
・TRUSTED_Rte_Function を OS の TrustedFunction として登録すること
・TRUSTED_Rte_Function の呼び出しのためのインデックス番号の定数として、下記名称の定数が定義されていること
Rte_Function_<pid>（<pid> は EcucPartition のショートネーム）
※ EcucPartition は、ECU コンフィグレーションで定義されたパーティション要素

例）TRUSTED_Rte_Function の呼び出しのためのインデックス番号の定数定義
#define Rte_Function_BswPartition ((TrustedFunctionIndexType) 0)

OS モジュールのコンフィグレーション方法については、OS モジュールの添付ドキュメントを参照のこと。

4.2.4. ランナブルの最小実行間隔を制御する場合の OS 設定

【RTE ジェネレータ仕様】
ランナブルの最小実行間隔に 0 以上が指定され、最小実行間隔制御が有効となるランナブルについては、以下の OS コンフィグレーションを期待する。
・ランナブルが動作するタスクは、該当ランナブルの動作以外に利用されない
・ランナブルを起動する周期イベントの周期、およびオフセット設定と、
ランナブルが動作するタスクを起動する OS アラームの実行周期、および起動オフセットが等しい。

■注意事項
ランナブルの最小実行間隔制御を有効にした場合は、RTE ジェネレータ仕様に準拠した OS コンフィグレーションを適用する必要がある。

4.3. 設計上の注意事項

以下では、RTE モジュール、および SchM モジュール上で動作するランナブルの構成に関する注意事項を記載する。

4.3.1. ランナブルの起動イベント数

【RTE ジェネレータ仕様】
ランナブルを起動するためのイベント定義は以下の仕様とする。
あるランナブルに対して、同じ種類のイベントを複数設定するような構成はサポートしない。

○未サポートとなる構成の例１：
・ランナブル１に対して、周期イベント１（300ms）、および周期イベント 2(500ms)で起動するように構成する

○未サポートとなる構成の例２：
・１つの外部トリガ API が呼び出された際に動作対象となるランナブル１に対して、複数回の外部トリガイベントが発生して、ランナブル１がイベント数だけ呼び出されるように構成する。

■注意事項
複数のイベントから同一ランナブルの実行を行うような場合は、ランナブルをイベントの数だけ定義し、それぞれのイベント毎に別のランナブルが起動されるよう構成を行い、各ランナブルの実装内容を共通にして実現する必要がある。

補足：
異なるイベント（例：周期イベントと、外部トリガイベント）により、同一ランナブルを起動するように構成することは可能である。

例：ランナブル１に対して、周期イベント１（300ms）、および外部トリガイベントで起動するように構成する。

5. 未実装機能・制限事項

以下では、RTE モジュールの未実装機能、および制限事項について記載する。

5.1. 未実装機能

なし

5.2. 制限事項

なし

6. リリース履歴

以下では、RTE モジュールのリリース履歴について記載する。

6.1. Ver1.0.0

リリース日	2011/12/28
リリース概要	初回の正式リリース版
変更履歴	なし

6.2. Ver1.0.1

リリース日	2012/01/31
リリース概要	正式リリース FIX 版
変更履歴	Ver1.0.0 の制限事項の解除 BSW Scheduler の排他 API 名に排他リソース名を付与できるように修正 Rte_COMCbkCounterErr の IF を COM 側のコールバック IF に合わせるように修正

6.3. Ver1.1.0

リリース日	2012/11/30
リリース概要	2012 年度正式リリース 版
変更履歴	機能追加 • モード連携機能のサポート Ver1.0.1 の拡張機能の廃止 • Rte_TerminatePartition の廃止 Ver1.0.1 の制限事項の解除 • Non Trusted パーティション-Non Trusted パーティション間の S/R 連携をサポート