Using RTE or BRE without AUTOSAR OS

Version 1.1.1

2016-06-08

Application Note AN-ISC-8-1166

Author Hannes Haas

Restrictions Customer Confidential – Vector decides

Abstract This application note describes how to configure the MICROSAR BRE (Basic Runtime Environment) or a minimum RTE without using an AUTOSAR OS.

Table of Contents

1 Overview

The RTE and BRE implementation assume the existence of an AUTOSAR OS or an OSEK OS as it very much depends on it functionalities and configuration description. If a different OS or a basic scheduler (non-AUTOSAR OS) is used, the provided interfaces must be similar to the interfaces defined by AUTOSAR OS. Interfaces in this context include.

> C APIs and header files
> ECUC configuration data exchange
> API behavior

The intention of this document is to provide an overview of the configuration process and the required APIs. An exact description of the API behavior is not scope of this document. Please refer to the AUTOSAR OS specification for details.

The focus of this document is set on a minimum RTE feature set that is also provided by the BRE. For more advanced RTE features the usage of an AUTOSAR OS is highly recommended.

1.1 Limitations

This document describes a RTE/BRE configuration for an ECU project without enhanced functionality such as

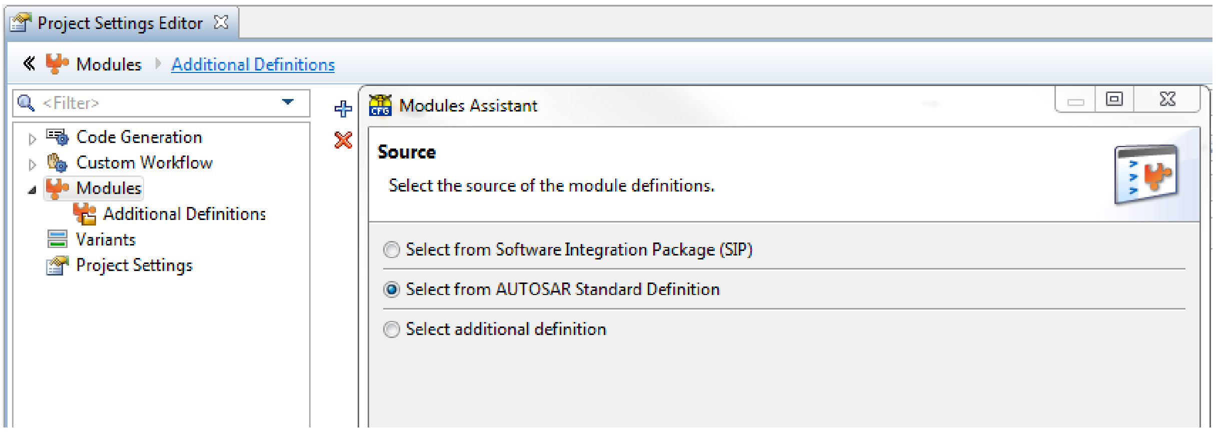
> Multi-core
> Application partitioning using MPU
> Safety requirements

These assumptions are vital as otherwise the dependency between BRE and operating system becomes too complex to be managed and documented in such an application note.

2 OS Configuration in DaVinci Configurator

BRE requires the existence of an AUTOSAR OS in the ECUC project as it will e.g. read the tasks configuration from the OS configuration.

In order to add an OS, launch DaVinci Configurator Pro and open the **Project Settings** page. Add the AUTOSAR OS as new BSW module. As the OS is not part of your delivery choose **Select from AUTOSAR Standard Definition**. Now you can add the OS to your project.



Source

Select the source of the module definitions.

Select from Software Integration Package (SIP) [✓ Select from AUTOSAR Standard Definition] Select additional definition

Figure 2-1 Project Settings Editor in DaVinci Developer Pro

It is not required to configure the OS completely, thus the OS configuration may contain validation errors as long as all information required for BRE is available. The AUTOSAR OS configuration must match with the behavior of your non-AUTOSAR OS with respect to the following aspects. Please configure these properties using DaVinci Configurator Pro:

> **Tasks**: Add a container for at least those tasks that are handled by BRE. The BRE will implement the tasks that are used to map BSW main functions to.
> **TaskPriority**: Configure the task priority of the tasks implemented by BRE

> **TaskSchedule**: Attribute defines the preempt ability of the task. If this Task cannot be preempted by other Tasks select NON. If this Task can be preempted by other Tasks with higher TaskPriority select FULL.

Having set up the OS in this way, the BRE can now be configured by e.g. mapping BSW main functions (`MainFunctions`) to the tasks.

If no OSEK OS is used, only basic tasks shall be used as otherwise the interface to the OS will get too complex. In contrast to extended tasks, basic tasks can be integrated with a non-AUTOSAR and non OSEK-OS scheduler with reasonable effort. This documentation is therefore limited to the usage of basic tasks.

To cause the BRE to utilize basic tasks only, it is only allowed to map Runnables (BSW MainFunctions) with same trigger conditions to one task (cyclic with same cycle time and offset). If there are different cycle times to be used, one task for each cycle time has to be defined. The same applies if different offset times shall be used. Mixing different trigger conditions can cause the BRE to utilize complex OS features such as extended tasks and schedule tables.

2.1 BRE Configuration Requirements for the OS

Once the BRE is configured, click **Validate** or **Generate** in DaVinci Configurator Pro. Activate at least the **RTE generator** (which is also used to generate the BRE code). The BRE will now update parts of the OS configuration based on the current configuration. To verify that only basic tasks are required by the BRE, check that no events are created in ‘Os/OsEvent’.

If you do not use an OSEK configuration tool you will find the required OS configuration in the OS configuration within DaVinci Configurator Pro. The following OS configuration options added by the BRE are of importance for your non-AUTOSAR OS configuration.

2.2 Call Cycle of Tasks

For basic tasks the BRE implementation (see Rte.c) will invoke SetRelAlarm indicating the required cycle time (this is also the time configured for the main functions mapped to that task). Configure your non-AUTOSAR OS in a way that it invokes the task in that cycle time.

3 Mandated OS APIs

Assuming the BRE is configured as above, basic tasks will be used only.

When using basic tasks, the following APIs are required by BRE and must be provided by the non-AUTOSAR OS. The BRE implementation assumes that these APIs match the AUTOSAR OS standard:

> **OS Tasks:** As defined by the AUTOSAR OS, the BRE will utilize the `TASK(x)` macro when implementing the task. `x` is thereby the function name as defined in the OS configuration. If your OS does not provide this macro, define this or a similar macro within `Os.h`.

Example
#define TASK(x) void x##func(void)

To invoke the task from your OS call: <x>func();

> Provision of `SuspendAllInterrupts(void)` and `ResumeAllInterrupts(void)` for critical section handling. These APIs shall support nested calls (i.e. lock interrupts the first time `SuspendAllInterrupts` is called, count the nesting and unlock interrupts the last time `ResumeAllInterrupts` is invoked). The API is used if Rte|RteBswModuleInstance|RteBswExclusiveAreaImpl|RteExclusiveAreaImplMechanism is set to "ALL_INTERRUPT_BLOCKING". This is the recommended setting and the default choice.
> Depending on the BRE configuration the APIs `DisableAllInterrupts(void)` and `EnableAllInterrupts(void)` are used that directly lock the interrupts without nesting.

> SetRelAlarm: Will be used to set up the cycle time and offset for (basic) tasks. You can use the API to enable the main function scheduling if required. If you have a simple scheduler that does not require dynamic alarm configuration, function may be defined to nothing within your `Os.h` implementation. Ensure the tasks are not called before the call of this API.

Example
```
#define SetRelAlarm (x, y, z)
```

CancelAlarm: API will be used to disable calling the main functions during ECU shutdown. You can use the API to disable the main function scheduling if required. If you have a simple scheduler this function may be defined to nothing within your `Os.h` implementation.

Example
```
#define CancelAlarm (x)
```

> **TerminateTask**: Will be called at the end of a basic task. If you have a simple scheduler this function may be defined to nothing within your `Os.h` implementation.

Example
```
#define TerminateTask()
```

> OS Functionality used by BRE must be published in the header `Os.h`.

4 Calling of Tasks

After the stack has been initialized (earliest after SchM_Init calling the `SetRelAlarm()` API) the Tasks have to be called by the non-AUTOSAR OS according to the defined cycle time.

5 Additional Resources

AUTOSAR STANDARD

AUTOSAR_SWS_OS.pdf AUTOSAR specification of OS

AUTOSAR_SWS_RTE.pdf AUTOSAR specification of RTE

TECHNICAL REFERENCE

TechnicalReference_Bre.pdf MICROSAR BRE technical reference (available in SIP if BRE is being used)

6 Contacts

For a full list with all Vector locations and addresses worldwide, please visit http://vector.com/contact/.