UserManual
AMD – MICROSAR 4
AUTOSAR Monitoring and Debugging

Version 1.3.0
English

Imprint

Vector Informatik GmbH
Ingersheimer Straße 24
D-70499 Stuttgart

Contents

1 Introduction

In this chapter you will find the following information:

1.1 About This User Manual

To Find information quickly

This user manual provides you with the following access help:

> At the beginning of each chapter you will find a summary of the contents

> The header shows in which chapter of the manual you are

> The footer shows the version of the manual

> At the end of the user manual you will find a glossary to look-up used technical terms

> At the end of the user manual an index will help you to find information quickly

Conventions

In the two tables below you will find the notation and icon conventions used throughout the manual.

Style	Utilization
bold	Fields/blocks, user/surface interface elements, window- and dialog names of the software, special emphasis of terms. **[OK]** Push buttons in square brackets **File**|**Save** Notation for menus and menu entries
MICROSAR	Legally protected proper names and marginal notes.
`Source Code`	File and directory names, source code, class and object names, object attributes and values
Hyperlink	Hyperlinks and references.
<CTRL>+<S>	Notation for shortcuts.

Symbol	Utilization
i	This icon indicates notes and tips that facilitate your work.
!	This icon warns of dangers that could lead to damage.
➡	This icon indicates more detailed information.
▲	This icon indicates examples.
👣	This icon indicates step-by-step instructions.
✎	This icon indicates text areas where changes of the currently described file are allowed or necessary.
✘	This icon indicates files you must not change.

Symbol	Utilization
🎞	This icon indicates multimedia files like e.g. video clips.
☰	This icon indicates an introduction into a specific topic.
🧱	This icon indicates text areas containing basic knowledge.
🎓	This icon indicates text areas containing expert knowledge.
🔄	This icon indicates that something has changed.

1.1.1 Certification

Quality Management System

Vector Informatik GmbH has ISO 9001:2010 certification. The ISO standard is a globally recognized standard.

1.1.2 Warranty

Restriction of warranty

We reserve the right to modify the contents of the documentation or the software without notice. Vector disclaims all liabilities for the completeness or correctness of the contents and for damages which may result from the use of this documentation.

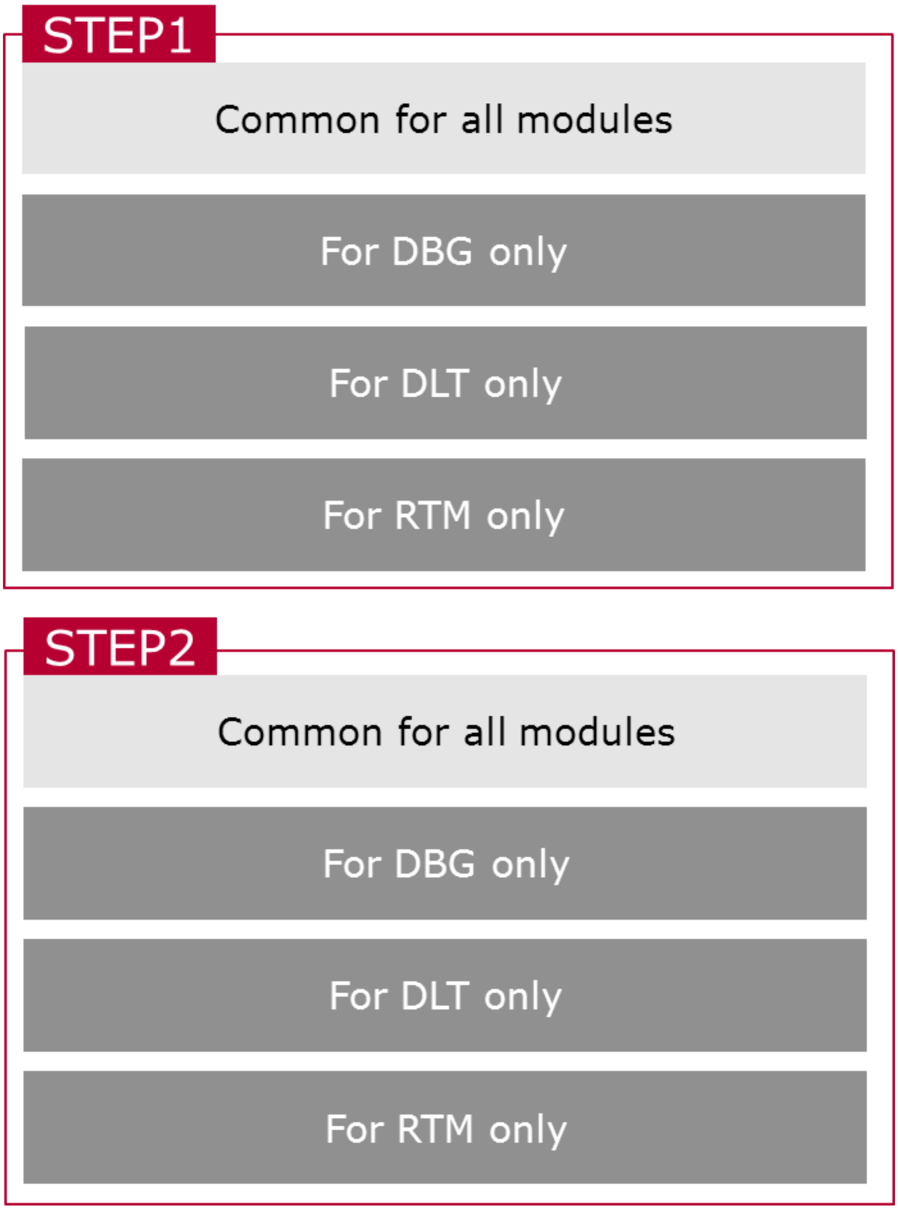
1.1.3 Trademarks

Protected trademarks

All brand names in this documentation are either registered or non-registered trademarks of their respective owners.

1.2 How to Use This Step-By-Step Introduction

Follow the steps in this manual to get AMD working fast and easily. The document describes how to integrate all modules of AMD in parallel or only one, or two. The reader can only read the parts necessary for the module of interest and omit the other parts.



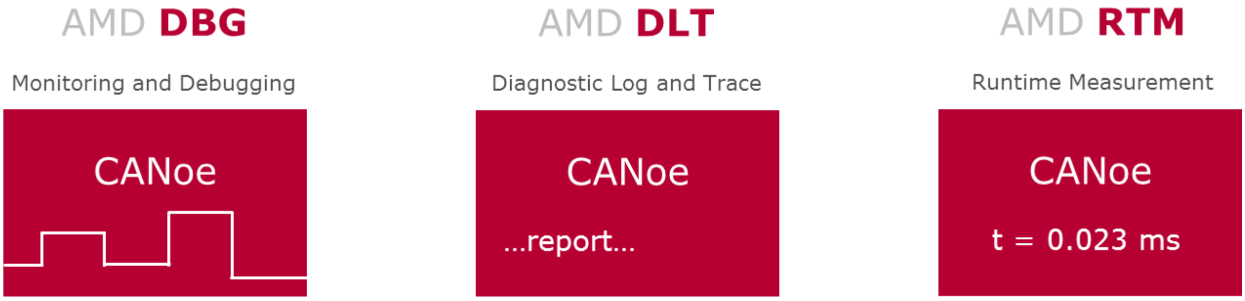
2 AMD Overview

AMD – AUTOSAR Monitoring and Debugging

The AMD – the AUTOSAR Monitoring and Debugging – comprises the modules DBG (**De**Bu**G**ging) and DLT (**D**iagnostic **L**og and **T**race) as defined by AUTOSAR and is extended by the module RTM (**R**un**T**ime **M**easurement) by Vector. The tasks are:

- **Visualize paramater values of an ECU in CANoe at runtime.**
  Observe in CANoe when a parameter of your software changes in the ECU.
- **DET and DEM errors**
  Display DET or DEM error textually.
- **User defined Verbose and Non-Verbose messages**
  Send free messages either predefined at configuration time or freely programmed in your application.
- **Measure runtime and CPU load of predefined BSW module functions**
  Figure out how long e.g. `CAN Rx Interrupt` is running or many other service functions of the BSW modules.
- **Measure runtime and CPU load of your application code**
  Mark any application code with start and stop and get the runtime of it.
- **Measure the overall CPU load**
  Measure the CPU load of the whole ECU software running on the target.

Note: MICROSAR AMD requires CANoe.AMD. DBG and DLT functionality is also available with Vector's CANape (or other XCP tools).



Multimedia link: You find the video AMD – AUTOSAR Monitoring and Debugging in the YouTube channel of Vector. It describes the DBG and the DLT module based on the MICROSAR 3 solution.

http://www.youtube.com/watch?v=c9Im9fM70To

To use AMD you need:

- MICROSAR AMD
- MICROSAR XCP or VX1000 debug port connectivity
- Network communication supporting XCP or a direct access via VX1000
- CANoe.AMD
- An already running ECU project with a configured XCP module

Note: Different than defined by AUTOSAR, DLT additionally supports XCP as communication protocol. The advantage is that the XCP protocol is standardized and tools with XCP support are widespread.

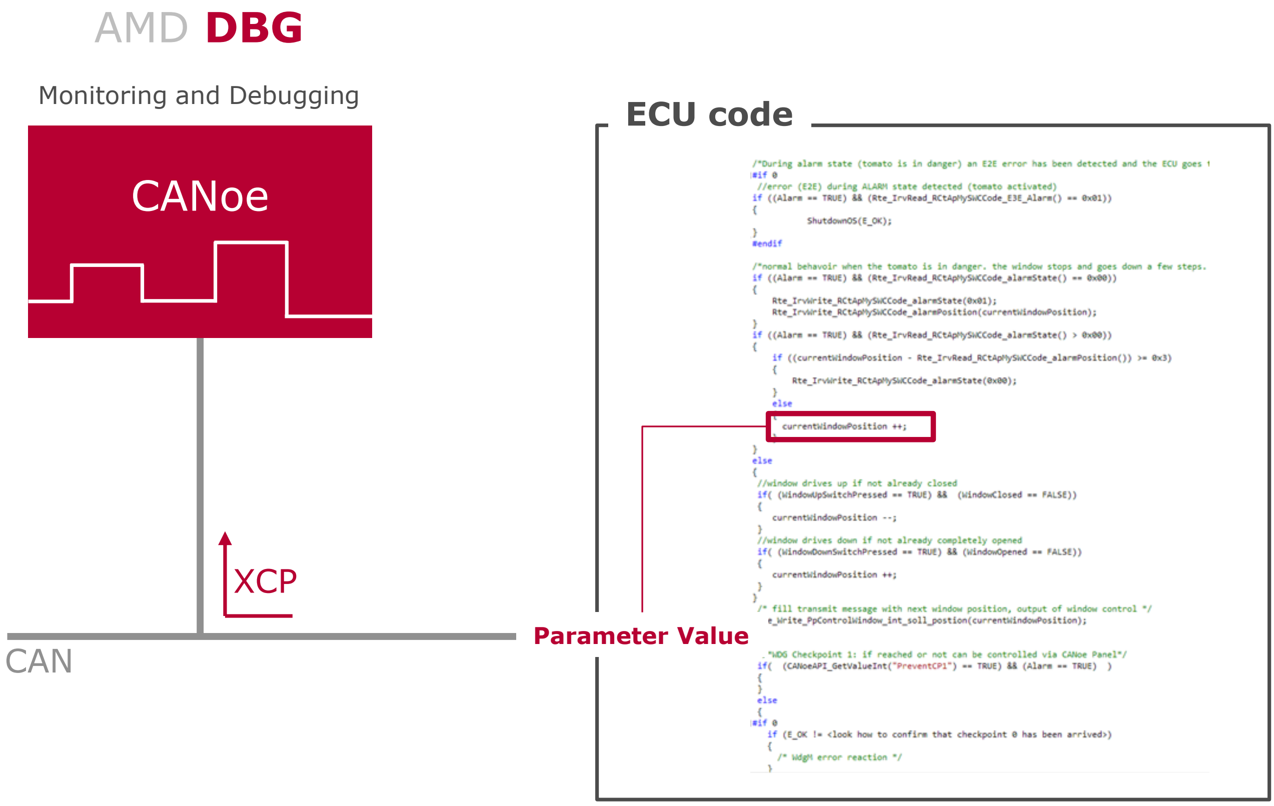
Reference: This document covers the steps for DLT on XCP. A detailed description of AUTOSAR DLT with communication layer can be found in TechnicalReference_Dlt.pdf.

3 Modules of AMD

In This Chapter You Will Find the Following Information:

3.1 AMD Debugging

With AMD Debugging (DBG) you can observe internal parameters of your AUTOSAR software without the need of a debugger.

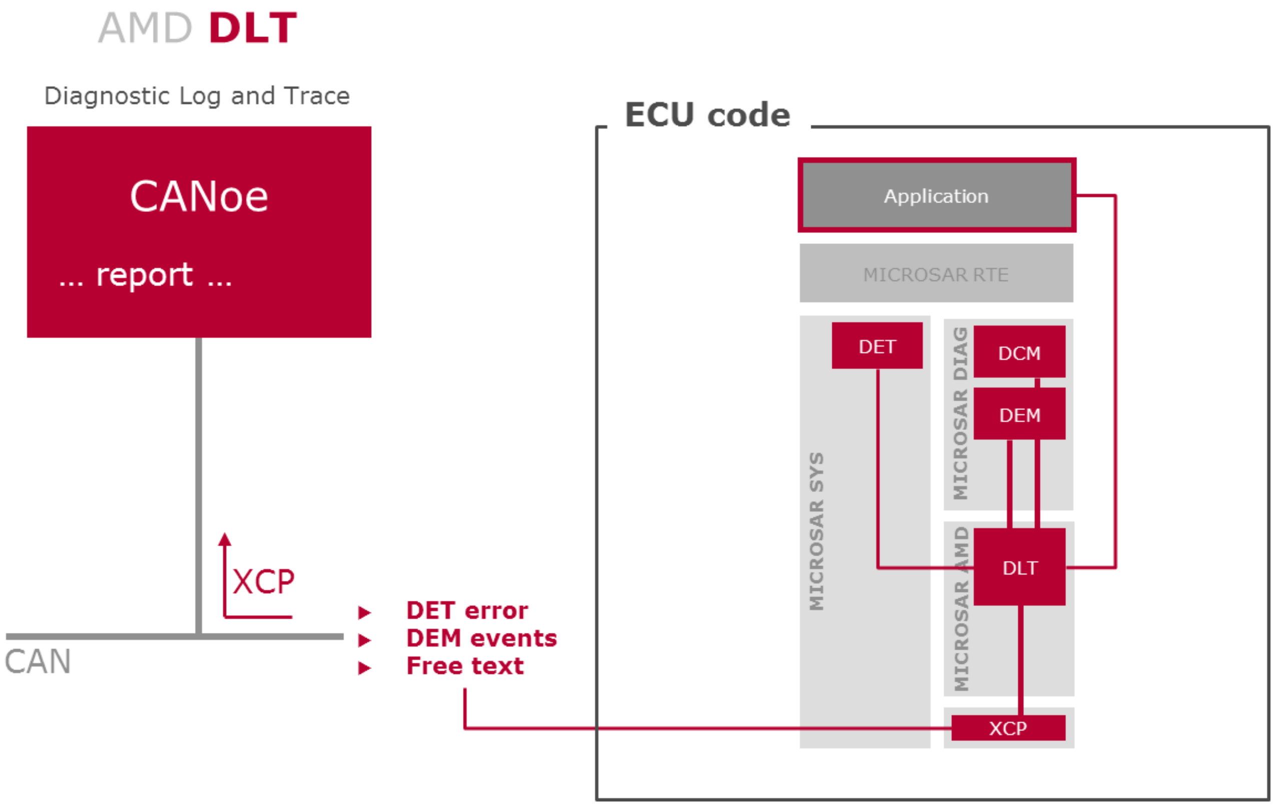


```
/*During alarm state (tomato is in danger) an E2E error has been detected and the ECU goes t
#if 0
 //error (E2E) during ALARM state detected (tomato activated)
if ((Alarm == TRUE) && (Rte_IrvRead_RCtApMySWCCode_E3E_Alarm() == 0x01))
{
          ShutdownOS(E_OK);
}
#endif

/*normal behavoir when the tomato is in danger. the window stops and goes down a few steps.
if ((Alarm == TRUE) && (Rte_IrvRead_RCtApMySWCCode_alarmState() == 0x00))
{
    Rte_IrvWrite_RCtApMySWCCode_alarmState(0x01);
    Rte_IrvWrite_RCtApMySWCCode_alarmPosition(currentWindowPosition);
}
if ((Alarm == TRUE) && (Rte_IrvRead_RCtApMySWCCode_alarmState() > 0x00))
{
    if ((currentWindowPosition - Rte_IrvRead_RCtApMySWCCode_alarmPosition()) >= 0x3)
    {
        Rte_IrvWrite_RCtApMySWCCode_alarmState(0x00);
    }
    else
    {
      currentWindowPosition ++;
    }
}
else
{
 //window drives up if not already closed
 if( (WindowUpSwitchPressed == TRUE) &&  (WindowClosed == FALSE))
 {
    currentWindowPosition --;
 }
 //window drives down if not already completely opened
 if( (WindowDownSwitchPressed == TRUE) && (WindowOpened == FALSE))
 {
    currentWindowPosition ++;
 }
}
 /* fill transmit message with next window position, output of window control */
   e_Write_PpControlWindow_int_soll_postion(currentWindowPosition);

  "WDG Checkpoint 1: if reached or not can be controlled via CANoe Panel"/
 if(  (CANoeAPI_GetValueInt("PreventCP1") == TRUE) && (Alarm == TRUE)  )
 {
 }
 else
 {
#if 0
   if (E_OK != <look how to confirm that checkpoint 0 has been arrived>)
   {
     /* WdgM error reaction */
   }
```

The parameter values that you can select freely out of a broad collection is transported to CANoe via XCP and then displayed in the State Tracker window of CANoe.

3.2 AMD Diagnostic Log and Trace

With AMD Diagnostic Log and Trace debugging and error tracing becomes very easy and comfortable.



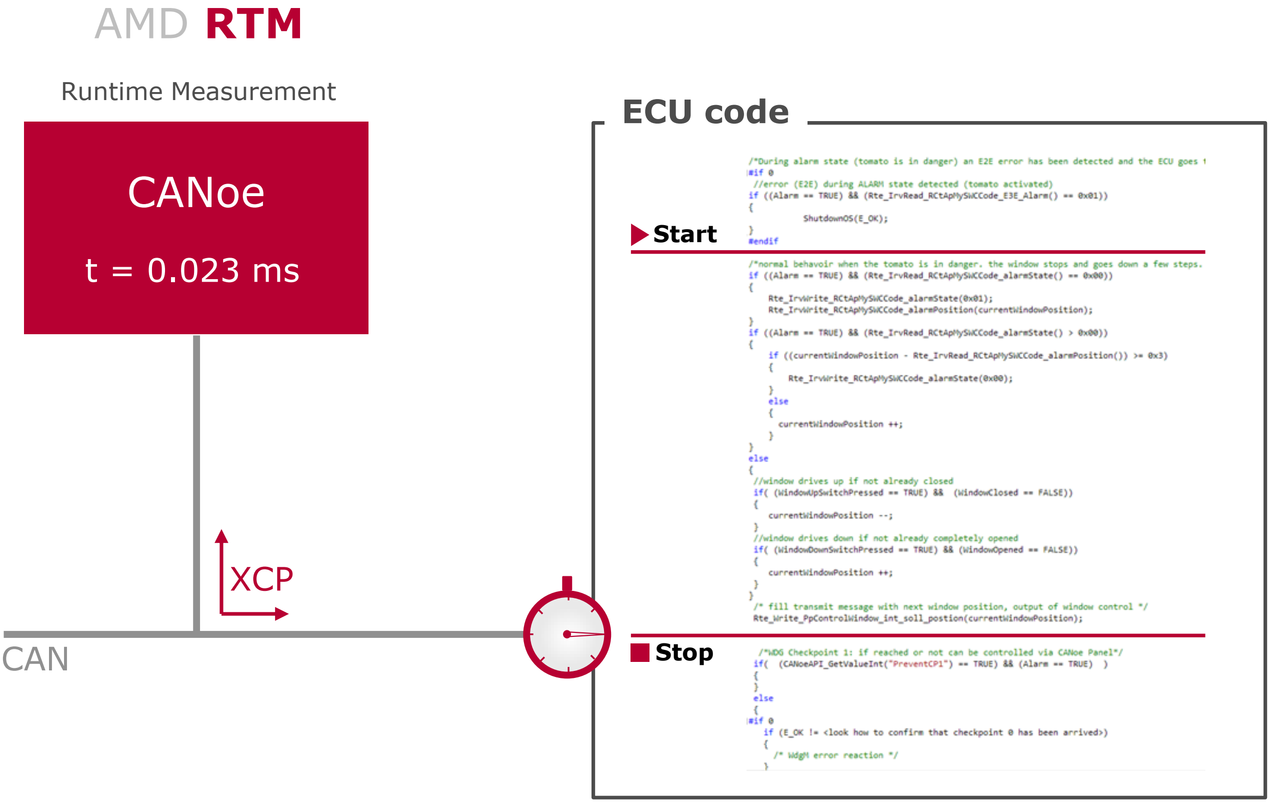
DET errors, DEM fault memory status changes and customizable text from your application can be sent to CANoe via XCP. CANoe can be used to visualize or automatically process the received data (e.g. as part of a test environment).

3.3 AMD Runtime Measurement

RTM – Runtime measurement

With RTM you can measure the runtime and CPU load of

- Predefined functions of the BSW modules
- Your application code



```
/*During alarm state (tomato is in danger) an E2E error has been detected and the ECU goes t
#if 0
 //error (E2E) during ALARM state detected (tomato activated)
if ((Alarm == TRUE) && (Rte_IrvRead_RCtApMySWCCode_E3E_Alarm() == 0x01))
{
           ShutdownOS(E_OK);
}
#endif
/*normal behavoir when the tomato is in danger. the window stops and goes down a few steps.
if ((Alarm == TRUE) && (Rte_IrvRead_RCtApMySWCCode_alarmState() == 0x00))
{
    Rte_IrvWrite_RCtApMySWCCode_alarmState(0x01);
    Rte_IrvWrite_RCtApMySWCCode_alarmPosition(currentWindowPosition);
}
if ((Alarm == TRUE) && (Rte_IrvRead_RCtApMySWCCode_alarmState() > 0x00))
{
    if ((currentWindowPosition - Rte_IrvRead_RCtApMySWCCode_alarmPosition()) >= 0x3)
    {
        Rte_IrvWrite_RCtApMySWCCode_alarmState(0x00);
    }
    else
    {
      currentWindowPosition ++;
    }
}
else
{
 //window drives up if not already closed
 if( (WindowUpSwitchPressed == TRUE) &&  (WindowClosed == FALSE))
 {
    currentWindowPosition --;
 }
 //window drives down if not already completely opened
 if( (WindowDownSwitchPressed == TRUE) && (WindowOpened == FALSE))
 {
    currentWindowPosition ++;
 }
}
 /* fill transmit message with next window position, output of window control */
 Rte_Write_PpControlWindow_int_soll_postion(currentWindowPosition);
  /*WDG Checkpoint 1: if reached or not can be controlled via CANoe Panel*/
 if(  (CANoeAPI_GetValueInt("PreventCP1") == TRUE) && (Alarm == TRUE)  )
 {
 }
 else
 {
#if 0
   if (E_OK != <look how to confirm that checkpoint 0 has been arrived>)
   {
     /* WdgM error reaction */
   }
```

Mechanism of runtime measurement

How Does Measurement Work

At a start point the measurement is started, at a stop point, it is stopped. In your code you can define as many start-stop couples as you want. You can also use the predefined measurements of the BSW modules.

Define the complete duration of a measurement and get the following results:

Count:
Counts how often the defined start-stop sections had been passed.

Max:
Shows the absolute longest runtime of each start-stop section.

Min:
Shows the absolute shortest runtime of each start-stop section.

Average:
Shows the average runtime of each start-stop section.

Average CPU load:
Shows the average CPU load caused by each start-stop section.

4 AMD Step by Step

In This Chapter You Will Find the Following Information:

4.1 Pre-Conditions

Base is an already running system with XCP

This manual assumes an already running ECU project including XCP communication.

4.2 STEP Configuration

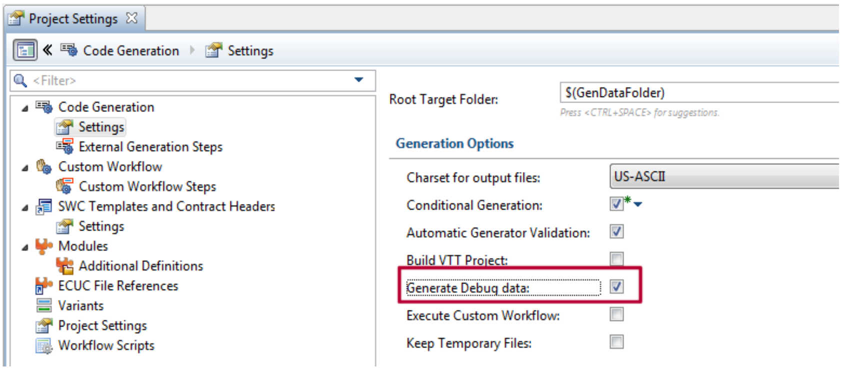
4.2.1 Configuration of DBG

Note: If only BSW internal variables shall be measured and the feature of triggering debug functions in the ECU is not used, it is not necessary to add the module DBG to the configuration. Nevertheless, the step to enable generation of debug data is mandatory in any case.

DBG in DaVinci Configurator Pro

Use checkbox **Generate Debug data** on **Project|Project Settings|Code Generation|Settings** to switch generation of debug variables on or off.

Switched off the generation time will be reduced a lot. Use this feature if there is no new generation of the debug variables necessary and switch it on on demand only.



Root Target Folder: $(GenDataFolder)

Press <CTRL+SPACE> for suggestions.

Generation Options

Charset for output files: US-ASCII

Conditional Generation: ☑

Automatic Generator Validation: ☑

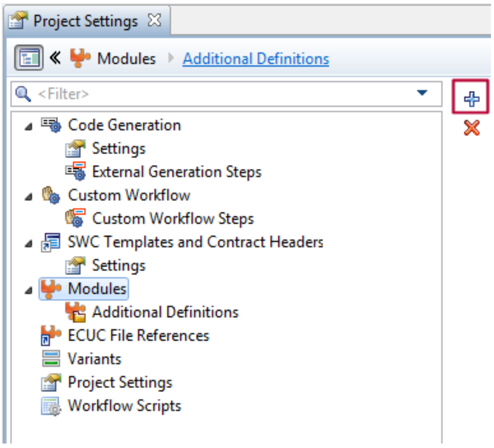
Build VTT Project: □

Generate Debug data: ☑

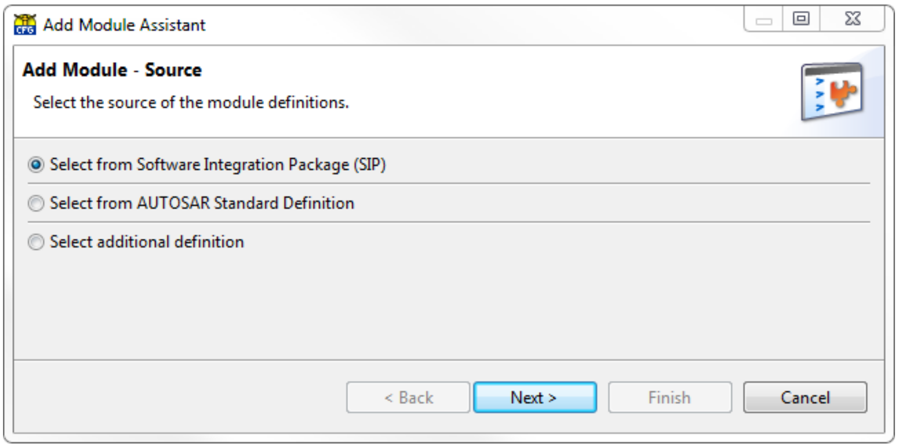
Execute Custom Workflow: □

Keep Temporary Files: □

For the basic feature of measuring BSW internal variables this step is enough. For further features you have to add and configure the module DBG.



Open **Project Settings** and click **Modules**. Add the module DBG by clicking on the blue plus button. The **Add Module Assistant** is opened.



Add Module Assistant

Add Module - Source

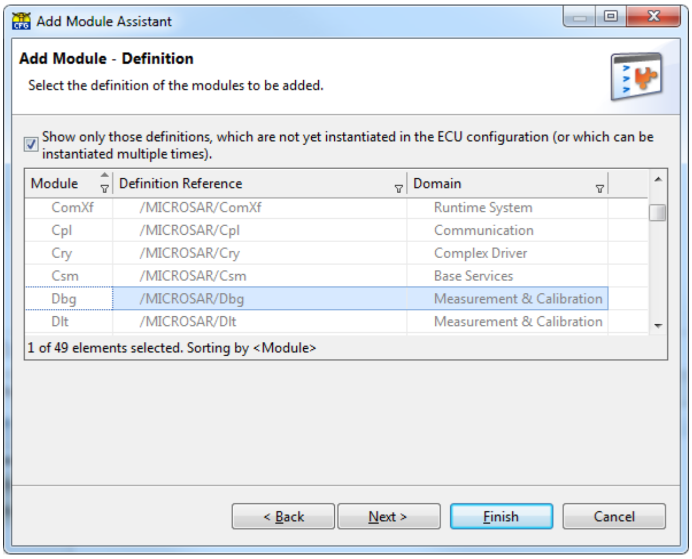
Select the source of the module definitions.

● Select from Software Integration Package (SIP)

○ Select from AUTOSAR Standard Definition

○ Select additional definition

Click **Select from Software Integration Package (SIP)** and then **[Next]**.



Add Module Assistant

Add Module - Definition

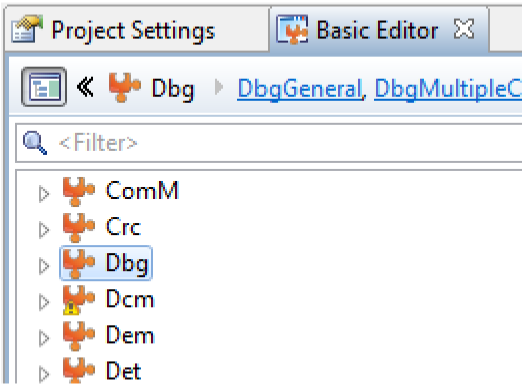
Select the definition of the modules to be added.

☑ Show only those definitions, which are not yet instantiated in the ECU configuration (or which can be instantiated multiple times).

Module	Definition Reference	Domain
ComXf	/MICROSAR/ComXf	Runtime System
Cpl	/MICROSAR/Cpl	Communication
Cry	/MICROSAR/Cry	Complex Driver
Csm	/MICROSAR/Csm	Base Services
Dbg	/MICROSAR/Dbg	Measurement & Calibration
Dlt	/MICROSAR/Dlt	Measurement & Calibration

1 of 49 elements selected. Sorting by <Module>

Search for the DBG module, select it and click **[Finish]**.

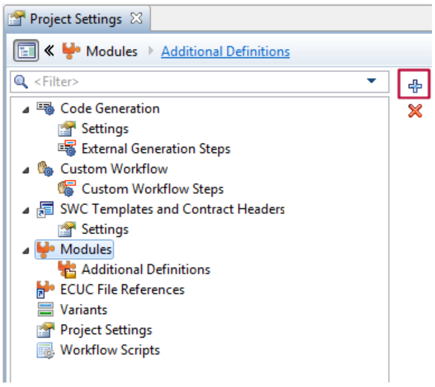


You can find the DBG module in the list of modules of the Basic Editor.

4.2.2 Configuration of DLT

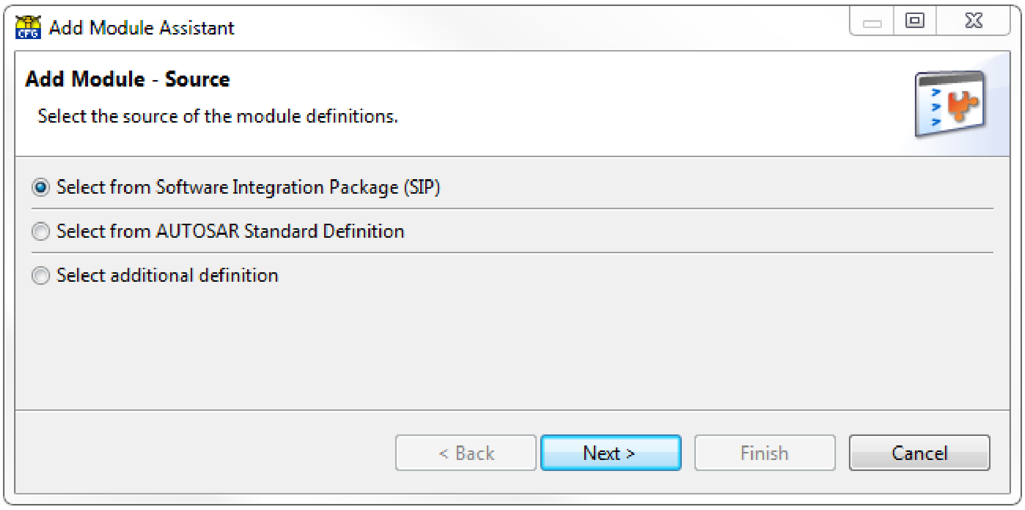
DLT in DaVinci Configurator Pro

Activate the module DLT.



- Code Generation
  - Settings
  - External Generation Steps
- Custom Workflow
  - Custom Workflow Steps
- SWC Templates and Contract Headers
  - Settings
- Modules
  - Additional Definitions
- ECUC File References
- Variants
- Project Settings
- Workflow Scripts

Open **Project Settings Editor** and click **Modules.** And blue plus to add a further module. The **Add Module Assistant** window opens.


Add Module Assistant

Add Module - Source

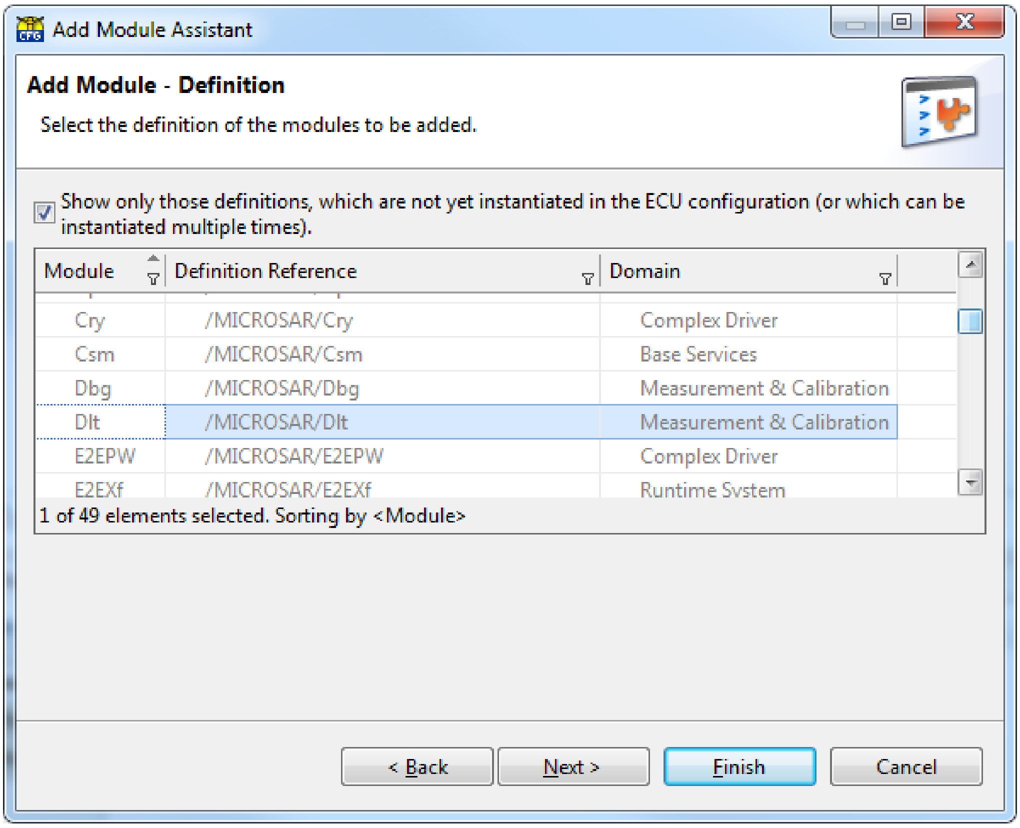
Select the source of the module definitions.

● Select from Software Integration Package (SIP)

○ Select from AUTOSAR Standard Definition

○ Select additional definition

Click **Select from Software Integration Package (SIP)** and the **[Next]**.



Add Module Assistant

Add Module - Definition

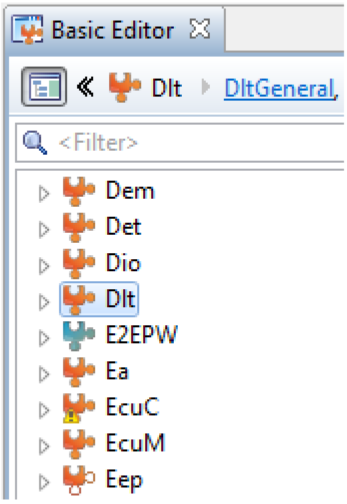
Select the definition of the modules to be added.

☑ Show only those definitions, which are not yet instantiated in the ECU configuration (or which can be instantiated multiple times).

Module	Definition Reference	Domain
Cry	/MICROSAR/Cry	Complex Driver
Csm	/MICROSAR/Csm	Base Services
Dbg	/MICROSAR/Dbg	Measurement & Calibration
Dlt	/MICROSAR/Dlt	Measurement & Calibration
E2EPW	/MICROSAR/E2EPW	Complex Driver
E2EXf	/MICROSAR/E2EXf	Runtime System

1 of 49 elements selected. Sorting by <Module>

Now search for the DLT module in the list, mark it and click **[Finish]**.



Find the DLT module now in the list of modules.

4.2.2.1 Non-Verbose Messages

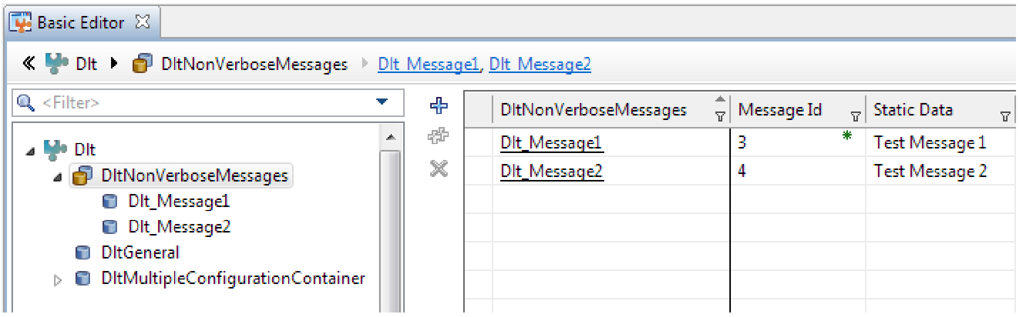
What is a non-verbose message?

A non-verbose message is a predefined message that is illustrated in CANoe after a predefined event in the ECU has been triggered. Non-verbose messages are runtime efficient as only an abstract identifier is transmitted over the network. CANoe then translates this identifier into the defined message. As a consequence the message content cannot be altered at runtime.

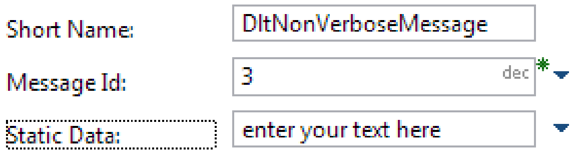
Create non-verbose message

How to create a non-verbose message

Open the **DLT** module in the **Basic Editor**.



DltNonVerboseMessages	Message Id	Static Data
Dlt_Message1	3	Test Message 1
Dlt_Message2	4	Test Message 2



Short Name: DltNonVerboseMessage

Message Id: 3 dec

Static Data: enter your text here

Name the message, define a unique **Message Id** and enter the textual information into the **Static Data** field. This is the actual textual message that will be displayed.

4.2.2.2 Verbose Messages

What is a verbose message?

In contrast to the non-verbose messages verbose messages are created dynamically at runtime by the ECU application. The content of the message can thus contain dynamic information such as signal values for debugging purposes. The drawback is that the message needs to be assembled and transmitted over the network. A maximum message length has to be defined at configuration time. To reduce the runtime required for transmitting the message it is recommended to reduce the maximum message length as much as possible.

To use the verbose mode, you have to

> Set **Implement Verbose Mode** in **DltGeneral**

> Set **Max Message Length** of verbose messages in **dltMultipleConfigurationContainer | DltProtocol**

Note: It is possible to use verbose and non-verbose messages at the same time in order to combine the advantages of both.

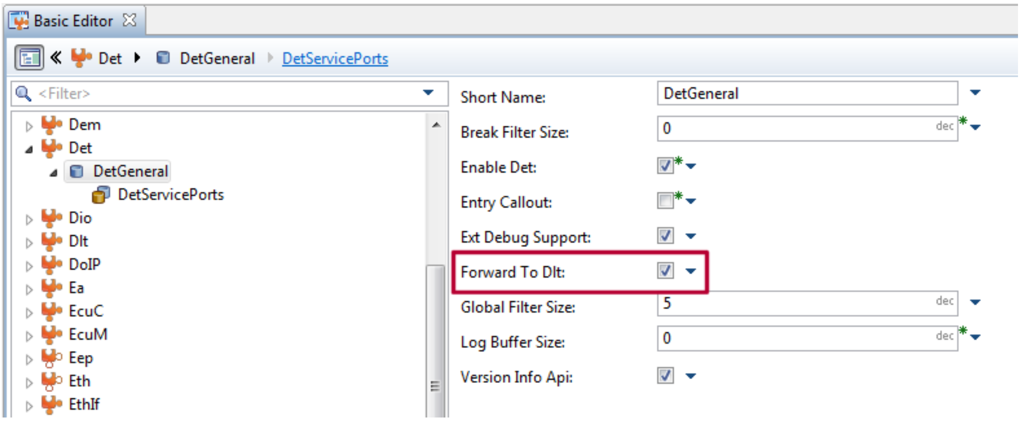
4.2.2.3 Forward Messages of DEM and DCM

Error Messages of DEM and DCM

Any DEM and DCM error reports are forwarded to DLT when the parameter DemGeneral/DemTriggerDltReports of the DEM module is activated.

4.2.2.4 Forward DET Messages to DLT

To activate the forwarding of DET messages to the DLT module you can use the parameter DetGeneral/DetForwardToDlt in the configuration of DET.



Short Name: DetGeneral

Break Filter Size: 0

Enable Det: ☑

Entry Callout: □

Ext Debug Support: ☑

Forward To Dlt: ☑

Global Filter Size: 5

Log Buffer Size: 0

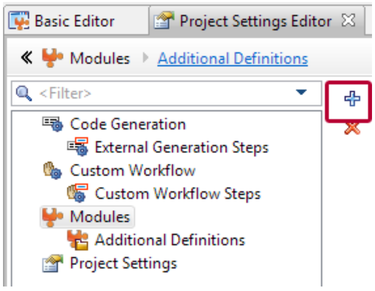
Version Info Api: ☑

4.2.3 Configuration of RTM

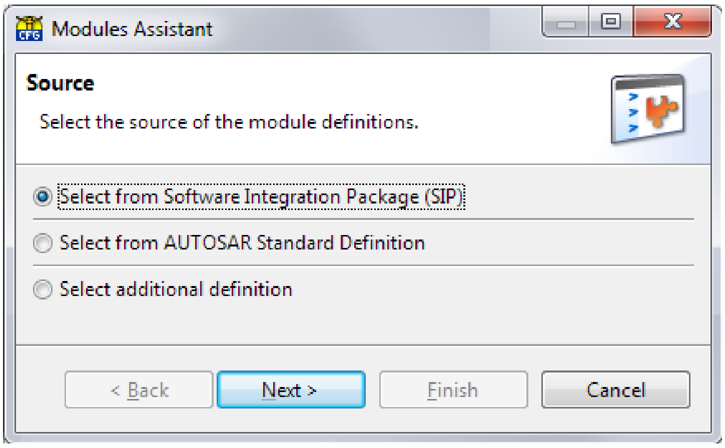
RTM in DaVinci Configurator Pro

Activate the RTM module.

Select **Project | Project Settings** and then click **Modules**.



Click the Plus to open the **Add Module Assistant**.



Source

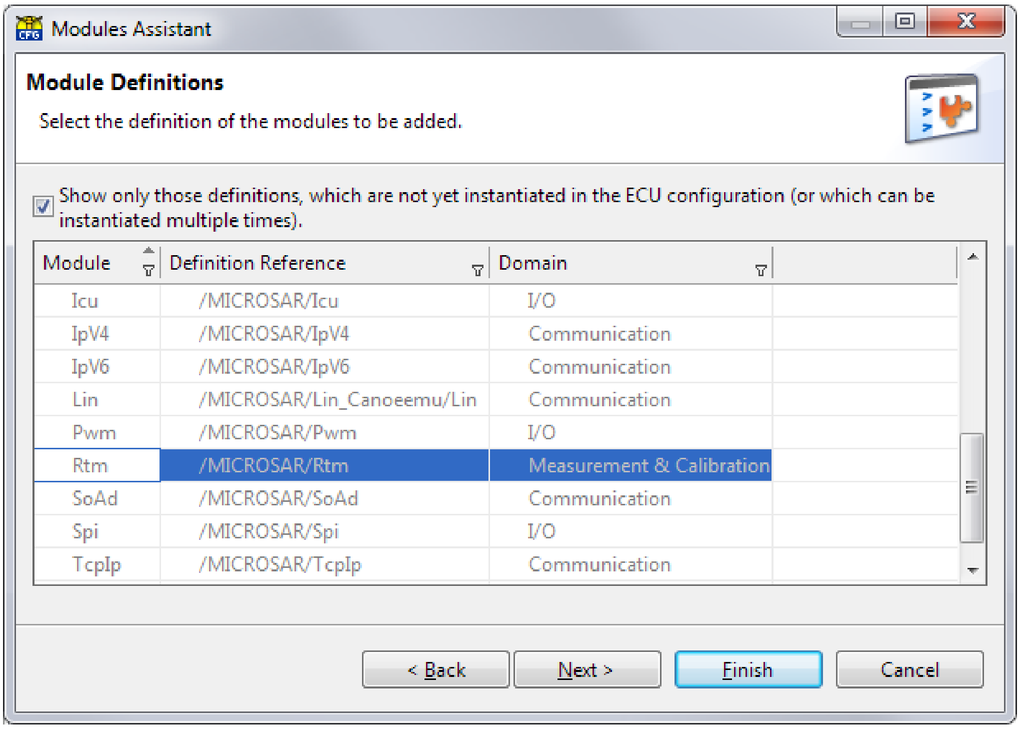
Select the source of the module definitions.

● Select from Software Integration Package (SIP)

○ Select from AUTOSAR Standard Definition

○ Select additional definition

Click **Select from Software Integration Package (SIP)** and then **[Next]**.



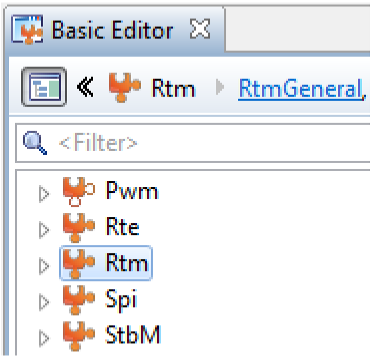
Module Definitions

Select the definition of the modules to be added.

☑ Show only those definitions, which are not yet instantiated in the ECU configuration (or which can be instantiated multiple times).

Module	Definition Reference	Domain
Icu	/MICROSAR/Icu	I/O
IpV4	/MICROSAR/IpV4	Communication
IpV6	/MICROSAR/IpV6	Communication
Lin	/MICROSAR/Lin_Canoeemu/Lin	Communication
Pwm	/MICROSAR/Pwm	I/O
Rtm	/MICROSAR/Rtm	Measurement & Calibration
SoAd	/MICROSAR/SoAd	Communication
Spi	/MICROSAR/Spi	I/O
TcpIp	/MICROSAR/TcpIp	Communication

Search the RTM module in the list, mark it and click **[Finish]**.



Now you should see the RTM module in the list of modules.

4.2.3.1 Enable Measurement Points of BSW Modules

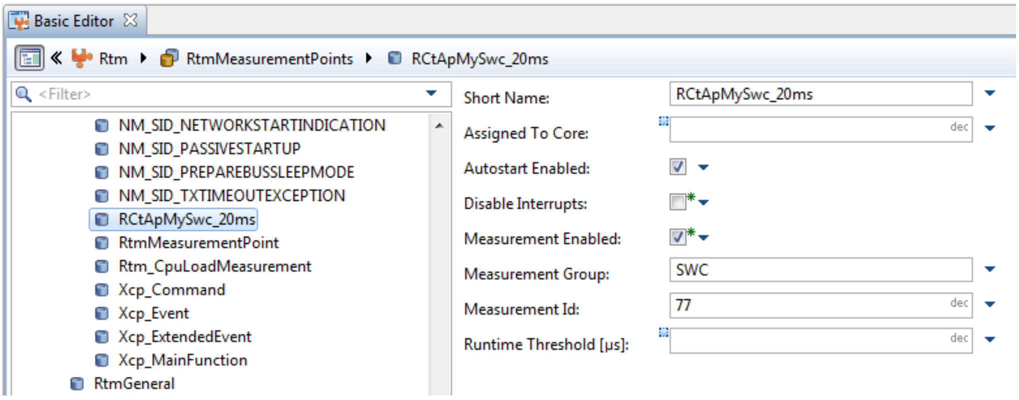
Some MICROSAR modules can automatically add measurement points for functions of interest, e.g. the main function. The configuration parameter of a BSW module supporting the RTM is called Runtime Measurement Support. Enabling the parameter automatically adds the relevant measurement points.

Example: FRNM
Parameter:
FrNmGlobalConfig/FrNmVendorSpecific/FrNmRuntimeMeasurementSupport

4.2.3.2 Configure Your Own Runtime Measurement Section

Add own measurement points

Go to the **Basic Editor** and open the **Rtm** tree.



Basic Editor

« Rtm ▸ RtmMeasurementPoints ▸ RCtApMySwc_20ms

<Filter>

- NM_SID_NETWORKSTARTINDICATION
- NM_SID_PASSIVESTARTUP
- NM_SID_PREPAREBUSSLEEPMODE
- NM_SID_TXTIMEOUTEXCEPTION
- RCtApMySwc_20ms
- RtmMeasurementPoint
- Rtm_CpuLoadMeasurement
- Xcp_Command
- Xcp_Event
- Xcp_ExtendedEvent
- Xcp_MainFunction
- RtmGeneral

Short Name: RCtApMySwc_20ms

Assigned To Core: dec

Autostart Enabled: ☑

Disable Interrupts: ☐

Measurement Enabled: ☑

Measurement Group: SWC

Measurement Id: 77 dec

Runtime Threshold [µs]: dec

Select **RtmMeasurementPoints** and add a new measurement point via a click on the blue plus. Select the new **RtmMeasurementPoint** and enter the following values.

Short Name:
Name of the measurement

Autostart Enabled:
The measurement should be started with the start of the software as XCP will not be up and running when the code is executed? Then check the box.

Disable Interrupts:
The interrupts can be disabled during the measurement. Check the box.

Measurement Enabled:
Enables the measurement point. If disabled, the start and stop commands will be generated as empty macros.

Measurement Group:
Organize your measurements in CANoe and in the report.

Measurement Id:
Unique handle ID of the measurement point.

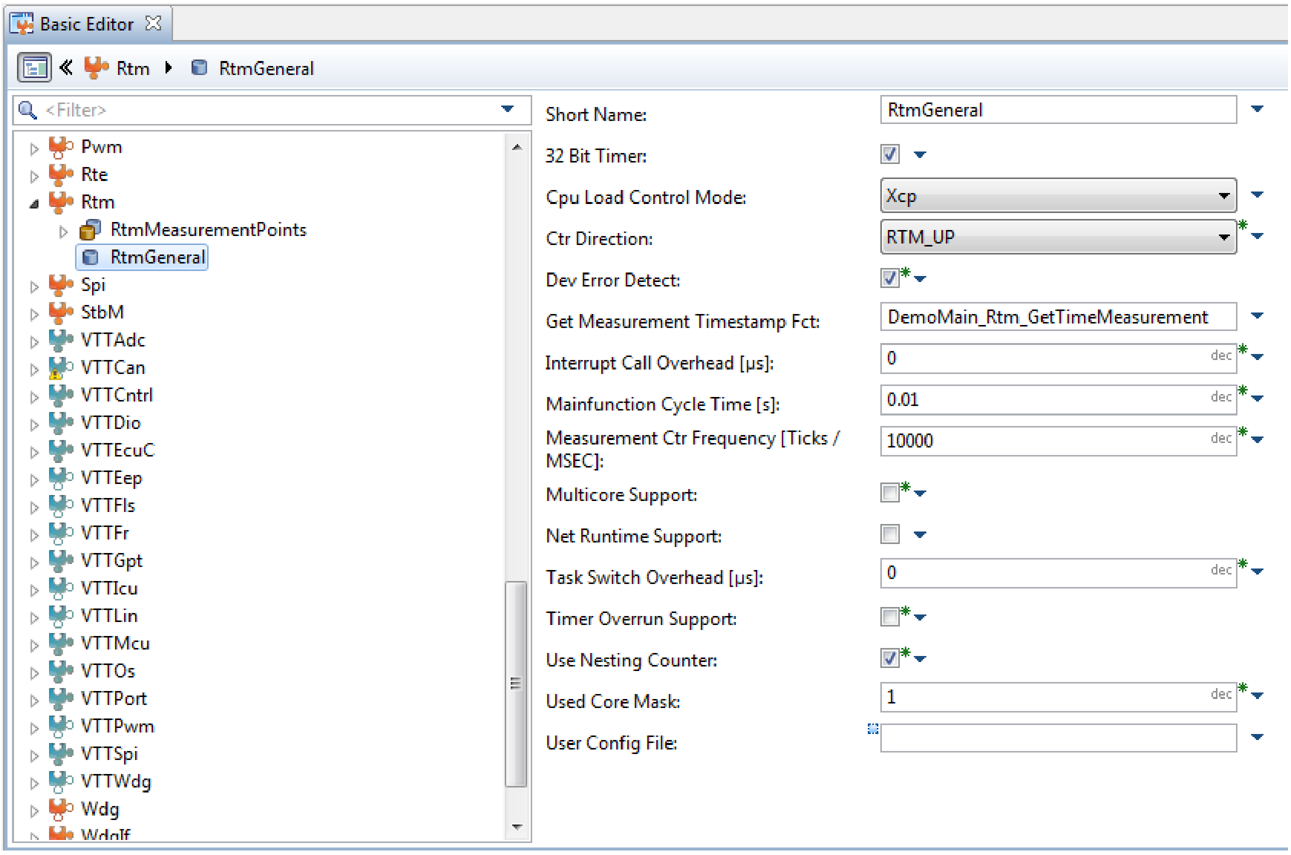
Runtime Threshold [µs]:
Define the runtime threshold of the measurement. If the current runtime of this measurement section exceeds the threshold, a callback function is called. Set value to 0 to inactivate the runtime threshold.

4.2.3.3 Free-Running Timer

Note: You have to provide a free running timer and implement the function to read the current time stamp value of the timer (**Get Measurement Timestamp Fct**).

Use an existing one from e.g. FlexRay or define a new one

To be able to measure very short time periods the recommended frequency of the timer is 10000/ms (**Measurement Ctr Frequency**). You can use an already existing timer (e.g. from FlexRay) or configure a new one.



Short Name:	RtmGeneral
32 Bit Timer:	☑
Cpu Load Control Mode:	Xcp
Ctr Direction:	RTM_UP
Dev Error Detect:	☑
Get Measurement Timestamp Fct:	DemoMain_Rtm_GetTimeMeasurement
Interrupt Call Overhead [µs]:	0
Mainfunction Cycle Time [s]:	0.01
Measurement Ctr Frequency [Ticks / MSEC]:	10000
Multicore Support:	□
Net Runtime Support:	□
Task Switch Overhead [µs]:	0
Timer Overrun Support:	□
Use Nesting Counter:	☑
Used Core Mask:	1
User Config File:	

In DaVinci Configurator Pro you define a function to get the current timer value (**Get Measurement Timestamp Fct**) that is called by RTM. The function has to be implemented by you and has to return a `uint32` value containing the current timer value. If configuration parameter **32 Bit Timer** is disabled, a `uint16` value has to be returned.

Additionally set the counter direction **Ctr Direction** to be **RTM_UP** or **RTM_DOWN** in accordance to the timer.

4.3 STEP Static and Generated Files

Generate files with DaVinci Configurator Pro. The following files are generated and accomplish the already existing static files of the modules DBG, DLT and RTM.

4.3.1 Files of DBG

DBG
static files

The DBG mechanism is integrated into each BSW module. There are additional source files available for AMD DBG that are used when the DBG module is added to the list of modules for further features.

- Dbg.c
- Dbg.h

DBG
generated files

- Dbg_Cfg.c
- Dbg_Cfg.h
- DbgPanel.xvp
- McData.a2l
- McData_Events.a2l

4.3.2 Files of DLT

DLT
static files

- Dlt.c
- Dlt.h

DLT
generated files

- Dlt_Cbk.c
- Dlt_Cbk.h
- Dlt_Cfg.h
- DltCom.c
- DltCom.h
- McData.a2l
- McData_Events.a2l

4.3.3 Files of RTM

RTM
static files

- Rtm.c
- Rtm.h

RTM
generated files

- Rtm_Canoe.can
- Rtm_Cbk.h
- Rtm_Cfg.c
- Rtm_Cfg.h
- Rtm_Canoe.xml

- McData.a2l
- McData_Events.a2l

4.4 STEP Integration

What is to do to use AMD

To use the module DBG, no integrations steps are necessary.

To use the module RTM a few integration preparations must be done like:

- Including the necessary header files to your source code
- Initialization of the modules
- Call functions cyclically (main functions of the modules)
- Necessary adaptations of your application to use the modules.

To use DLT, also some preparations must be done

- Connect your application software components with the DLT service component
- Trigger the different messages (verbose, non-verbose, DCM, DEM, DET) at appropriate place in your code

4.4.1 Integration of DBG

#include files of DBG

There is no file to be included to use DBG.

Initialization of DBG

There is no initialization for DBG.

Main Function of DBG

There is no main function for DBG.

Adapt your application for DBG

Nothing has to be adapted in your application for DBG.

Reference: If the DBG module is added to the list of modules for enabling further features, the DBG module has to be integrated. See TechnicalReference_Asr_Dbg.pdf for details.

4.4.2 Integration of DLT

#include files of DLT

To use the DLT module you have to include:

```
#include “Det.h”
#include “Dlt.h”
#include “Rte_Dlt_Type.h”
```

Initialization of DLT

To use DLT in your code, you have to initialize it via

- `Dlt_Init(DLT_INIT_POINTER)`

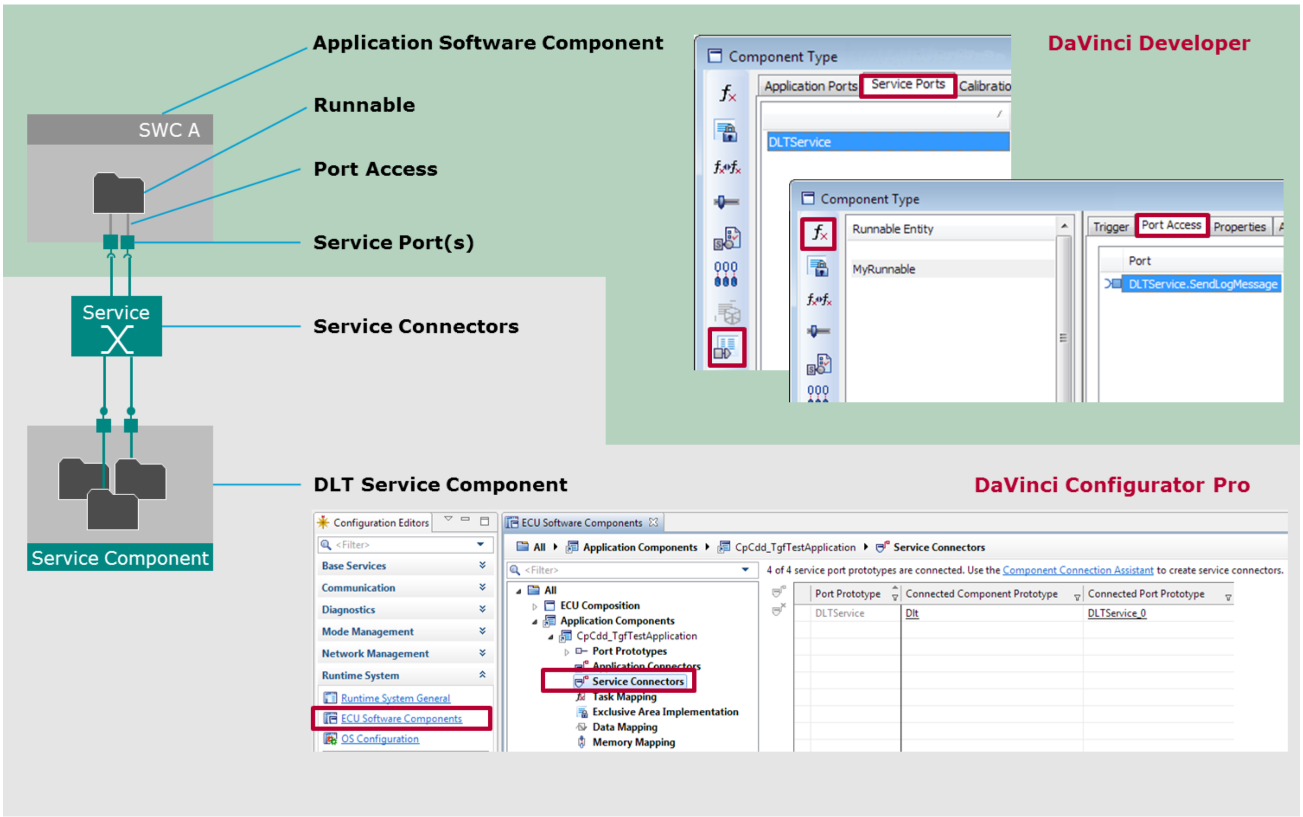
at start-up of your system, e.g. in the start-up lists of the ECUM.

Main Function of DLT

The main function Dlt_MainFunction has to be called cyclically by the SCHM. The cycle time of the main function can be set in the configuration container DltGeneral of the DLT module (DaVinci Configurator Pro).

4.4.2.1 DLT Service Component

The DLT is a service component. It provides the service to send verbose and non-verbose messages. If only DET and DEM status reports are required, the service component is not required to be connected with your application.



Service Port

Open the DaVinci Developer and the open your software component below **Application Component Type**, click **Port Prototype List** and select the tab **Service Ports**. Add a new **Port Interface** of Type DLTService. Its **Direction** should be **Client**.

Port Access

Switch to the **Runnable Entity List** and select the runnable that need access to the DLT or create a new one (think of a suitable trigger of a new runnable). Select the tab **Port Access**. Click **New** at the bottom of the dialog and select **Invoke Operations…**. Select the DLTService.

Save the project and close the DaVinci Developer.

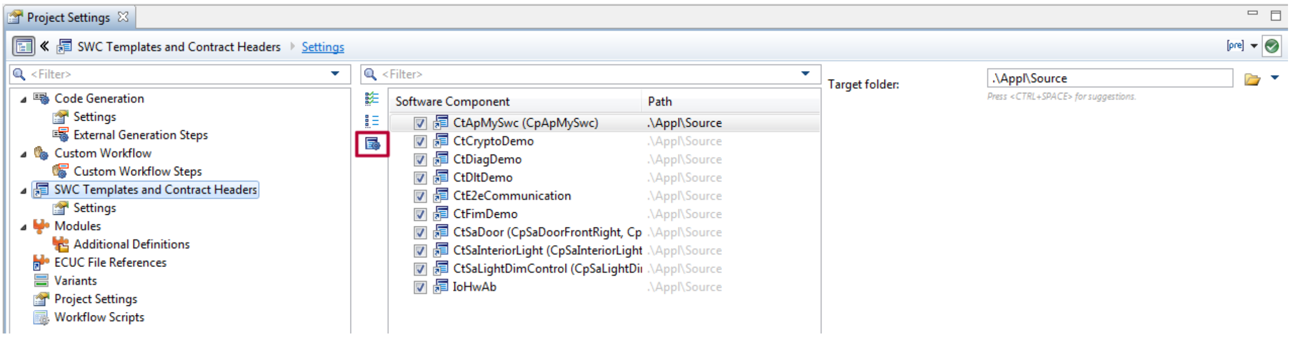
DaVinci Configurator Pro

Open the DaVinci Configurator Pro.

Open **Runtime System** and **ECU Software Components**. Navigate to your **Application Component** and select it. A tree opens below your component. Select

Service Connectors. Right-click to connect the port prototypes of your application with the ones of the DLT service component.

Note: If you have introduced a new runnable, you also have to map the runnable to a task. Click Task Mapping and map your runnable to an appropriate task.



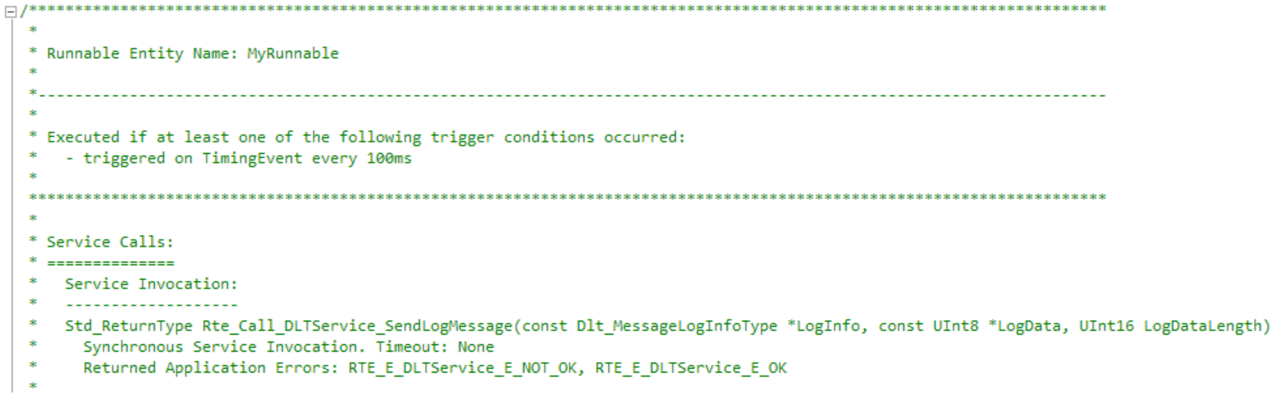
Software Component	Path
☑ CtApMySwc (CpApMySwc)	.\Appl\Source
☑ CtCryptoDemo	.\Appl\Source
☑ CtDiagDemo	.\Appl\Source
☑ CtDltDemo	.\Appl\Source
☑ CtE2eCommunication	.\Appl\Source
☑ CtFimDemo	.\Appl\Source
☑ CtSaDoor (CpSaDoorFrontRight, Cp	.\Appl\Source
☑ CtSaInteriorLight (CpSaInteriorLight	.\Appl\Source
☑ CtSaLightDimControl (CpSaLightDi	.\Appl\Source
☑ IoHwAb	.\Appl\Source

Target folder: .\Appl\Source

Press <CTRL+SPACE> for suggestions.

Generate runnable templates

Do not forget to start the template generator to generate the new runnable skeletons.



```
/**********************************************************************************************************************
 *
 * Runnable Entity Name: MyRunnable
 *
 *---------------------------------------------------------------------------------------------------------------------
 *
 * Executed if at least one of the following trigger conditions occurred:
 *   - triggered on TimingEvent every 100ms
 *
 **********************************************************************************************************************
 *
 * Service Calls:
 * ==============
 *   Service Invocation:
 *   -------------------
 *   Std_ReturnType Rte_Call_DLTService_SendLogMessage(const Dlt_MessageLogInfoType *LogInfo, const UInt8 *LogData, UInt16 LogDataLength)
 *     Synchronous Service Invocation. Timeout: None
 *     Returned Application Errors: RTE_E_DLTService_E_NOT_OK, RTE_E_DLTService_E_OK
 *
```

Note: Make sure to `#include` the header file `Dlt_Cfg.h` at the beginning of the file within a section (include and declaration area), that is protected against overwriting via a new generation process.

DLT service interface available

The result is that your runnable now has access to the service of the DLT via `Rte_Call_DltService_SendLogMessage`.

4.4.2.2 Implement Non-Verbose Messages

Non-verbose messages

The non-verbose messages are predefined in the configuration tool and can be used on demand. To decide, which of the defined messages should be sent, you just give the **Message Id**, that is covered behind a speaking define.

AUTOSAR SWC API

```
Dlt_NonVerboseMsgType nonVerboseMsg;

nonVerboseMsg.MessageId =
DltConf_DltNonVerboseMessage_Dlt_Message1;
```

```
Rte_Call_DLTService_LogOnXcp_SendLogMessage(&logInfo,
(P2CONST(UInt8, AUTOMATIC, RTE_DLT_APPL_DATA))&nonVerboseMsg,
0))
```

also see C API example

```
Dlt_NonVerboseMsgType nonVerboseMsg = { {0 /* HeaderType */, 0 /*
MessageCounter */, 0 /* Length */} /* StdHeader */,
DltConf_DltNonVerboseMessage_Dlt_Message1 /* MessageId */ };
         Dlt_SendLogMessage( 0 /* SessionId */, &msgLogInfoType /*
LogInfo */, (P2CONST(uint8, AUTOMATIC, DLT_APPL_VAR)) &nonVerboseMsg, 0
);
```

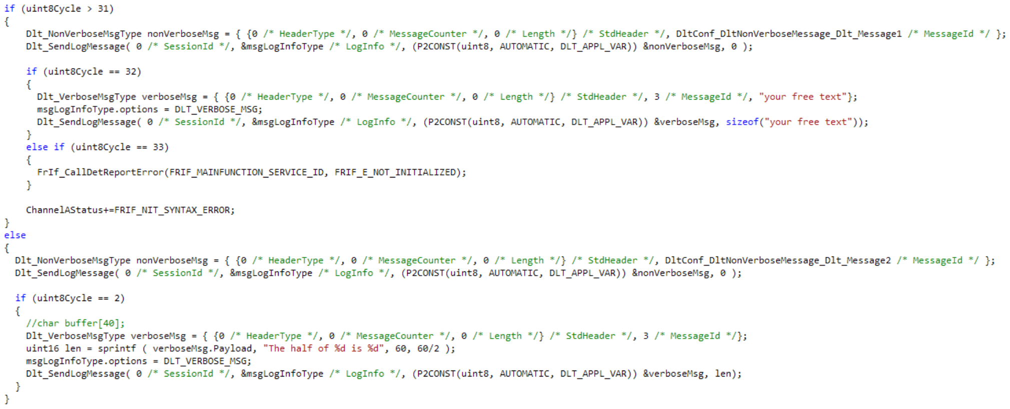
4.4.2.3 Implement Verbose Messages

Verbose messages

Similar to the non-verbose messages, you can also send verbose messages, free text in any length (regard defined maximum length). Look at the two examples in the code screenshot below.

```
Dlt_VerboseMsgType verboseMsg;

payload = (uint16)sprintf((P2VAR(char, AUTOMATIC,
AUTOMATIC))verboseMsg.Payload, "Received Flexray signal value
0x%x.", frData);

(void)Rte_Call_DLTService_LogOnXcp_SendLogMessage(&logInfo,
(P2CONST(uint8, AUTOMATIC, RTE_DLT_APPL_DATA))&verboseMsg,
payload);
```

C API



```
if (uint8Cycle > 31)
{
    Dlt_NonVerboseMsgType nonVerboseMsg = { {0 /* HeaderType */, 0 /* MessageCounter */, 0 /* Length */} /* StdHeader */, DltConf_DltNonVerboseMessage_Dlt_Message1 /* MessageId */ };
    Dlt_SendLogMessage( 0 /* SessionId */, &msgLogInfoType /* LogInfo */, (P2CONST(uint8, AUTOMATIC, DLT_APPL_VAR)) &nonVerboseMsg, 0 );

    if (uint8Cycle == 32)
    {
      Dlt_VerboseMsgType verboseMsg = { {0 /* HeaderType */, 0 /* MessageCounter */, 0 /* Length */} /* StdHeader */, 3 /* MessageId */, "your free text"};
      msgLogInfoType.options = DLT_VERBOSE_MSG;
      Dlt_SendLogMessage( 0 /* SessionId */, &msgLogInfoType /* LogInfo */, (P2CONST(uint8, AUTOMATIC, DLT_APPL_VAR)) &verboseMsg, sizeof("your free text"));
    }
    else if (uint8Cycle == 33)
    {
      FrIf_CallDetReportError(FRIF_MAINFUNCTION_SERVICE_ID, FRIF_E_NOT_INITIALIZED);
    }

    ChannelAStatus+=FRIF_NIT_SYNTAX_ERROR;
}
else
{
  Dlt_NonVerboseMsgType nonVerboseMsg = { {0 /* HeaderType */, 0 /* MessageCounter */, 0 /* Length */} /* StdHeader */, DltConf_DltNonVerboseMessage_Dlt_Message2 /* MessageId */ };
  Dlt_SendLogMessage( 0 /* SessionId */, &msgLogInfoType /* LogInfo */, (P2CONST(uint8, AUTOMATIC, DLT_APPL_VAR)) &nonVerboseMsg, 0 );

  if (uint8Cycle == 2)
  {
    //char buffer[40];
    Dlt_VerboseMsgType verboseMsg = { {0 /* HeaderType */, 0 /* MessageCounter */, 0 /* Length */} /* StdHeader */, 3 /* MessageId */};
    uint16 len = sprintf ( verboseMsg.Payload, "The half of %d is %d", 60, 60/2 );
    msgLogInfoType.options = DLT_VERBOSE_MSG;
    Dlt_SendLogMessage( 0 /* SessionId */, &msgLogInfoType /* LogInfo */, (P2CONST(uint8, AUTOMATIC, DLT_APPL_VAR)) &verboseMsg, len);
  }
}
```

4.4.3 Integration of RTM

#include files of RTM

To be able to use RTM functionality in your C code, you have to include:

```
#include "Rtm.h".
```

Initialization of RTM

To use AMD in your code, you have to call

- `Rtm_InitMemory() (if start-up code does not initialize RAM)` and
- `Rtm_Init()`

at start-up of your system, e.g. in the start-up lists of the ECUM.

Main Function of RTM

The main function of AMD with name `Rtm_Mainfunction()` has to be called cyclically by the SCHM, within the configured cycle time **Main function cycle time** in DaVinci Configurator Pro | RtmGeneral (Mainfunction Cycle time [s]).

Adapt your application for RTM

Now figure out the code part you want to measure. Every measurement has to be initialized first.

Start the Measurement

Initialize each measurement

To start the measurement use the function below and place it just before the code section you want to measure.

```
Rtm_Start(<Measurement Name>);
```

Stop the Measurement

Set marker to start and stop the measurement of your code

To stop the measurement use the function below and place it just behind the code section you want to measure.

```
Rtm_Stop(<Measurement Name>);
```

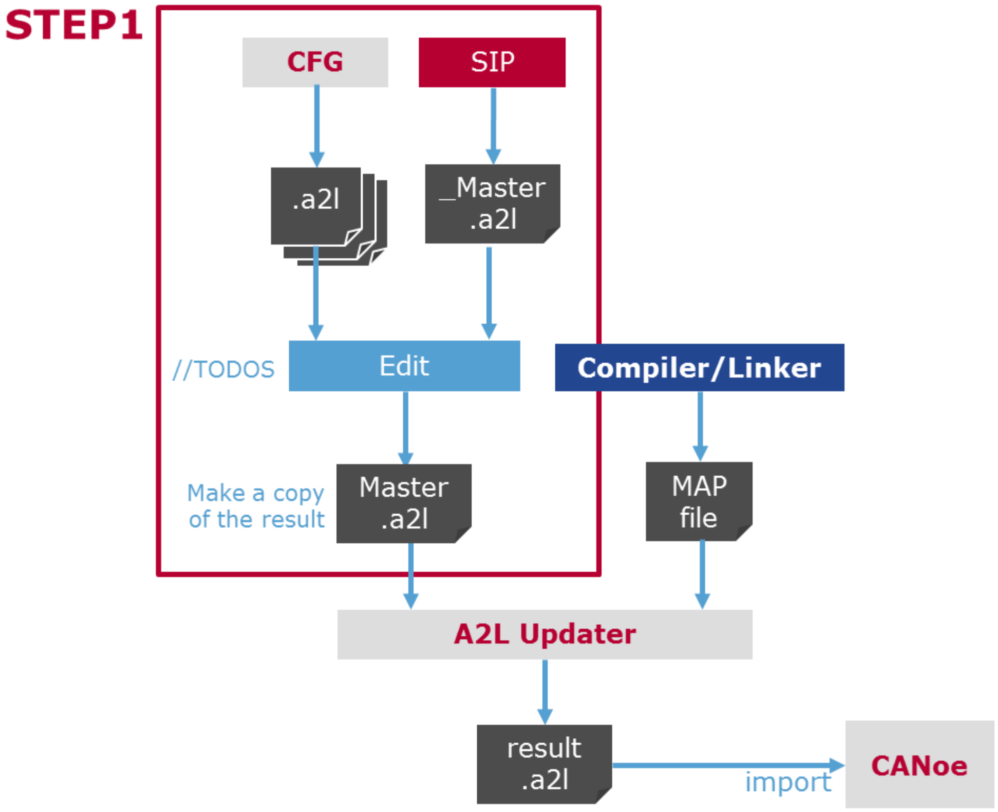
Note: If you place start and stop around a function call, you also measure the time to call the function and to come back from the function. To measure the pure runtime of the function, place start and stop into the function.

4.5 STEP A2l Address Update

The target of the A2L address update is to tailor the master.a2l file to your needs and then to create a result.a2l file where the address information of your project (map file) is considered.

This chapter is divided up into two steps, the preparation of the _master.a2l and the actual address update.

The illustration below shows the complete workflow where step 1 is highlighted.



4.5.1 STEP1 Edit _Master.a2l

Note: The **_Master.a2l** can be found in the delivery folder **..\Misc\McData.**

Location of _master.a2l

The **Edit** step is a manual one.

Adapt the template _Master.a2l file as follows:

- Save it as **Master.a2l** file in folder <ProjectDir>\Config\McData (without underscore, to show that this file is modified)
- Open it with a suitable editor
- Search for <**// TODO**> and adapt the includes for AMD
  - /include will be used (one backslash)

- //include will be ignored (two backslashes)
- include path must be relative to the location of the file Master.a2l

“First” TODOs for AMD

There are two parts of TODOs in the master.a2l to be adapted:

DBG and/or RTM

```
//include "McData\McData_Events.a2l" // TODO: Adapt path of A2L
file accordingly. Remove if not required.
```

To use the DBG and RTM module, **McData_Events.a2l** must be included. The file is generated to the specified McData folder defined for your project. By default this is <ProjectDir>\Config\McData

“Second” TODO for AMD

Second part of TODOs:

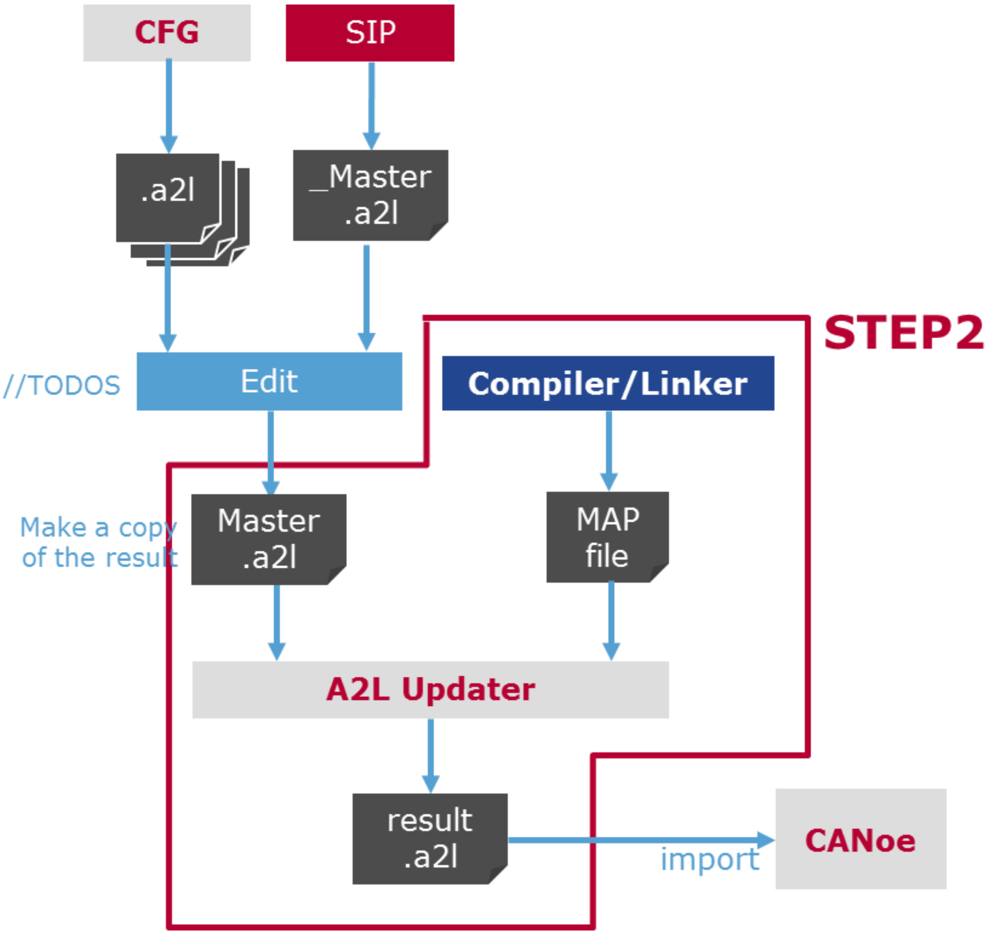
DBG and/or RTM

```
//include " McData\McData.a2l" // TODO: Adapt path of A2L file
accordingly. Remove if not required.
```

To use the DBG and RTM module, **McData.a2l** must be included. The file is generated to the specified McData folder defined for your project. By default this is <ProjectDir>\Config\McData.

4.5.2 STEP2 Perform Address Update

Now perform the address update. Address update means to create the **result.a2l** file out of the files **Master.a2l** and your **map** file by adding addresses of the measurement variables to the A2L-file.



Caution: Please always use a copy of your manually modified **Master.a2l** file because the address update tools may resolve the /includes with the content to be included. Afterwards, updates of the A2L fragments may no more be incorporated.

Note: The A2L updater can be found in the CANoe installation (**Start|Programs|Vector CANwin|Tools**)

Reference: For details of the A2L address update please refer to the UserManual_AUTOSAR_Calibration.pdf.

Go on with CANoe

Now all integration steps are done and we can go on with configuring CANoe to communicate with the AMD modules.

4.6 STEP Configure CANoe

Note: In this user manual CANoe 9.0 is used. Other versions of CANoe can be similarly configured with minor differences in the configuration dialogs.

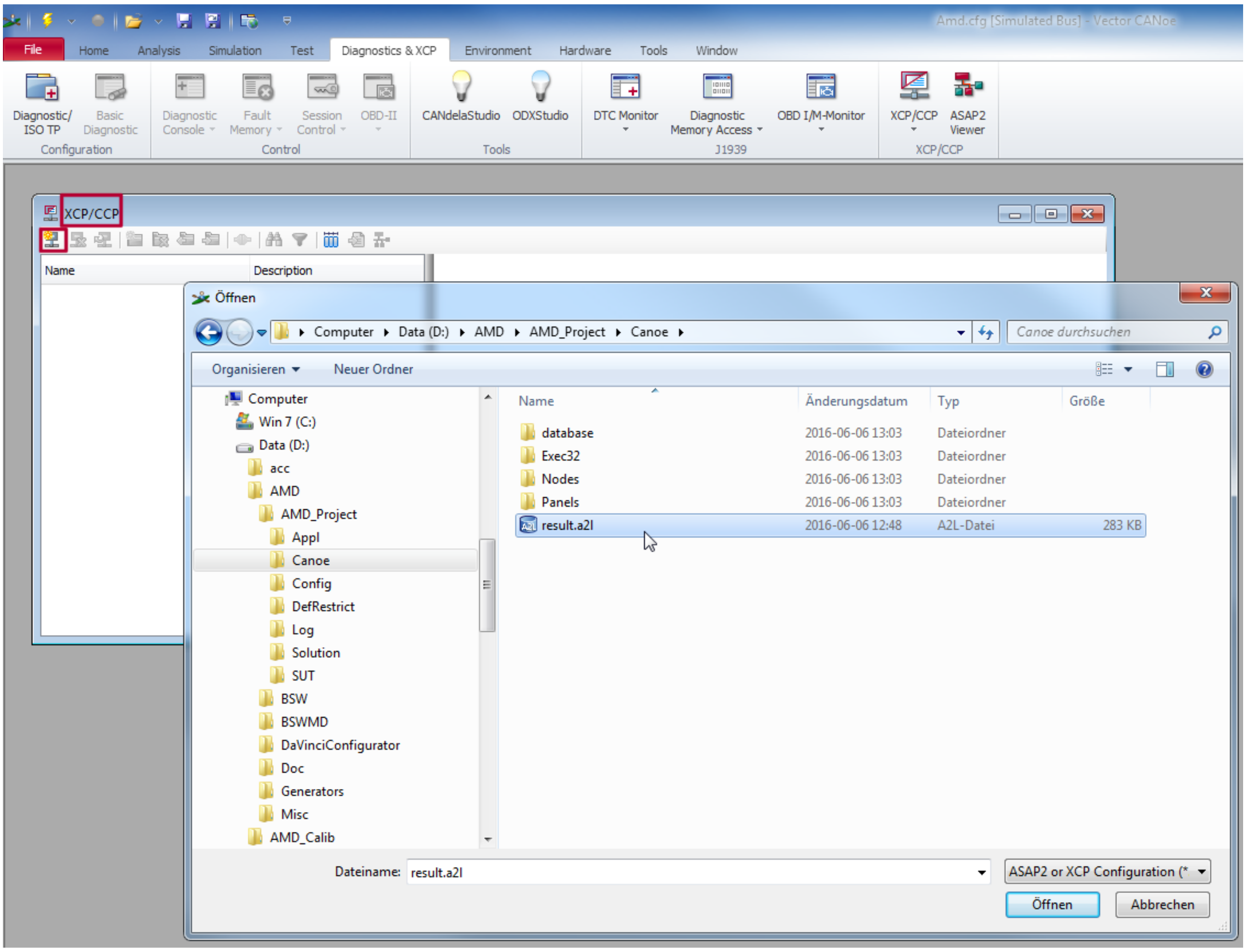
A2L file

Open CANoe.

Click **Diagnostics & XCP | XCP/CCP** to open the **XCP/CCP** window.

Click **[Add Device]**, select the **result.a2l** file and the **Device Configuration** window opens. Enter **Rtm** as **ECU Qualifier**. Confirm with **[OK]**.

Note: If RTM is not configured, you are free to choose any name as ECU qualifier.

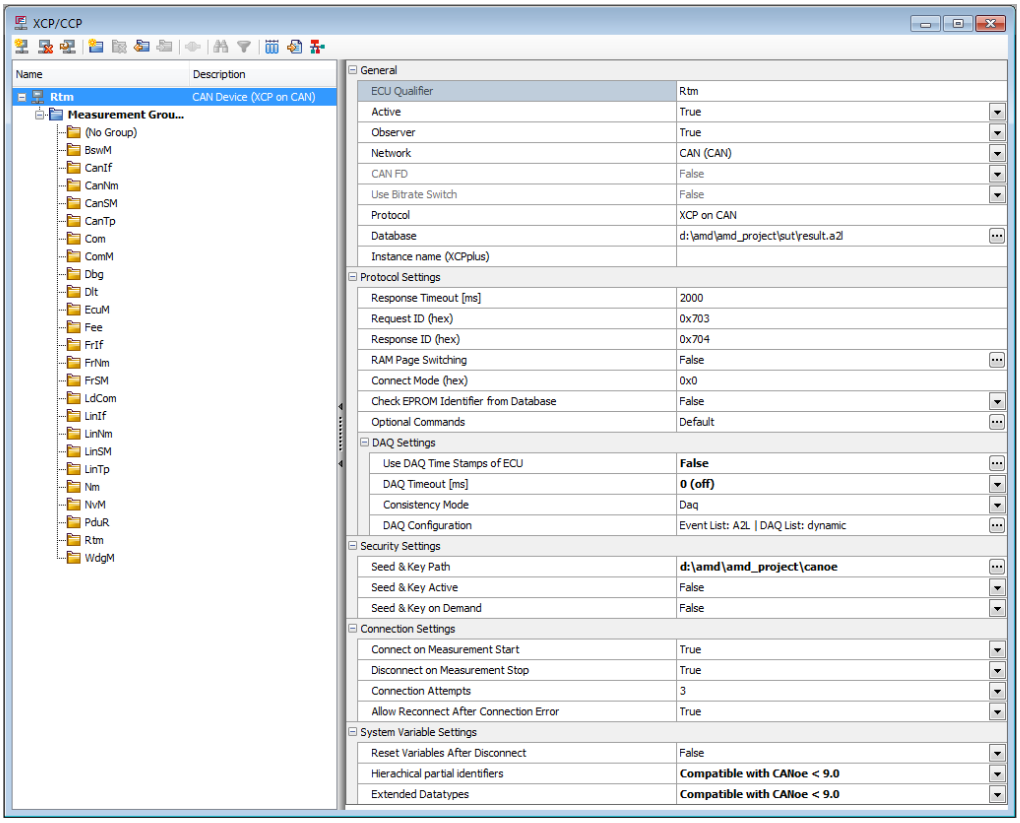


Name	Änderungsdatum	Typ	Größe
database	2016-06-06 13:03	Dateiordner	
Exec32	2016-06-06 13:03	Dateiordner	
Nodes	2016-06-06 13:03	Dateiordner	
Panels	2016-06-06 13:03	Dateiordner	
result.a2l	2016-06-06 12:48	A2L-Datei	283 KB

Dateiname: result.a2l

Device Configuration

Select the device **Rtm** and make sure that **DAQ Timeout** is set to **0 (off)**, and that both **Hierarchical partial identifiers** and **Extended Datatypes** is set to **Compatible with CANoe < 9.0**.

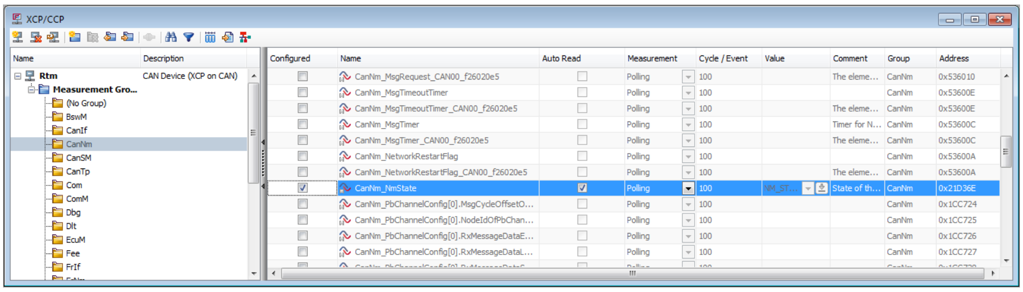


Name	Description
Rtm	CAN Device (XCP on CAN)
Measurement Grou...	
(No Group)	
BswM	
CanIf	
CanNm	
CanSM	
CanTp	
Com	
ComM	
Dbg	
Dlt	
EcuM	
Fee	
FrIf	
FrNm	
FrSM	
LdCom	
LinIf	
LinNm	
LinSM	
LinTp	
Nm	
NvM	
PduR	
Rtm	
WdgM	

General	
ECU Qualifier	Rtm
Active	True
Observer	True
Network	CAN (CAN)
CAN FD	False
Use Bitrate Switch	False
Protocol	XCP on CAN
Database	d:\amd\amd_project\sut\result.a2l
Instance name (XCPplus)	
Protocol Settings	
Response Timeout [ms]	2000
Request ID (hex)	0x703
Response ID (hex)	0x704
RAM Page Switching	False
Connect Mode (hex)	0x0
Check EPROM Identifier from Database	False
Optional Commands	Default
DAQ Settings	
Use DAQ Time Stamps of ECU	**False**
DAQ Timeout [ms]	**0 (off)**
Consistency Mode	Daq
DAQ Configuration	Event List: A2L | DAQ List: dynamic
Security Settings	
Seed & Key Path	**d:\amd\amd_project\canoe**
Seed & Key Active	False
Seed & Key on Demand	False
Connection Settings	
Connect on Measurement Start	True
Disconnect on Measurement Stop	True
Connection Attempts	3
Allow Reconnect After Connection Error	True
System Variable Settings	
Reset Variables After Disconnect	False
Hierachical partial identifiers	**Compatible with CANoe < 9.0**
Extended Datatypes	**Compatible with CANoe < 9.0**

4.6.1 CANoe DBG

Select the measurement variables in the XCP/CCP window you want to observe. Set these variables to **Configured** and select **Auto Read**. As measurement mode select **Polling** with the desired cycle time (in ms).



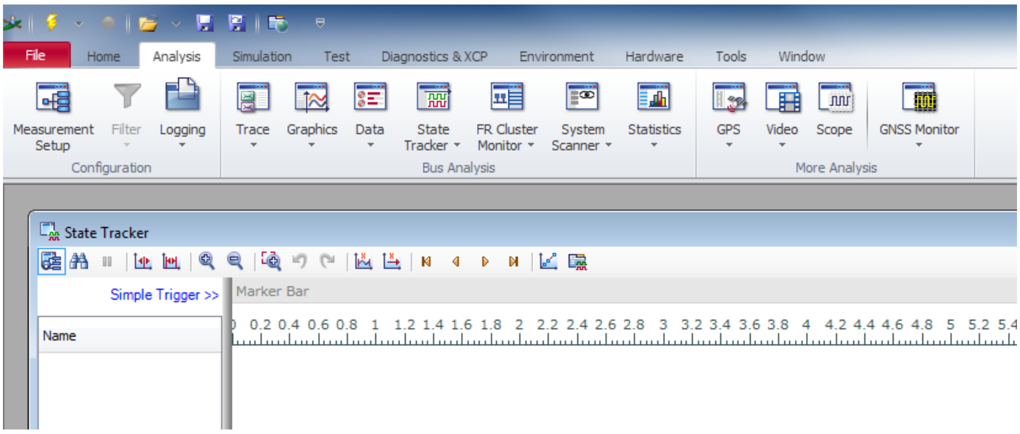
XCP/CCP

Name	Description
Rtm	CAN Device (XCP on CAN)
Measurement Gro...	
(No Group)	
BswM	
CanIf	
CanNm	
CanSM	
CanTp	
Com	
ComM	
Dbg	
Dlt	
EcuM	
Fee	
FrIf	

Configured	Name	Auto Read	Measurement	Cycle / Event	Value	Comment	Group	Address
☐	CanNm_MsgRequest_CAN00_f26020e5	☐	Polling	100		The eleme...	CanNm	0x536010
☐	CanNm_MsgTimeoutTimer	☐	Polling	100			CanNm	0x53600E
☐	CanNm_MsgTimeoutTimer_CAN00_f26020e5	☐	Polling	100		The eleme...	CanNm	0x53600E
☐	CanNm_MsgTimer	☐	Polling	100		Timer for N...	CanNm	0x53600C
☐	CanNm_MsgTimer_CAN00_f26020e5	☐	Polling	100		The eleme...	CanNm	0x53600C
☐	CanNm_NetworkRestartFlag	☐	Polling	100			CanNm	0x53600A
☐	CanNm_NetworkRestartFlag_CAN00_f26020e5	☐	Polling	100		The eleme...	CanNm	0x53600A
☑	CanNm_NmState	☑	Polling	100	NM_ST...	State of th...	CanNm	0x21D36E
☐	CanNm_PbChannelConfig[0].MsgCycleOffsetO...	☐	Polling	100			CanNm	0x1CC724
☐	CanNm_PbChannelConfig[0].NodeIdOfPbChan...	☐	Polling	100			CanNm	0x1CC725
☐	CanNm_PbChannelConfig[0].RxMessageDataE...	☐	Polling	100			CanNm	0x1CC726
☐	CanNm_PbChannelConfig[0].RxMessageDataL...	☐	Polling	100			CanNm	0x1CC727

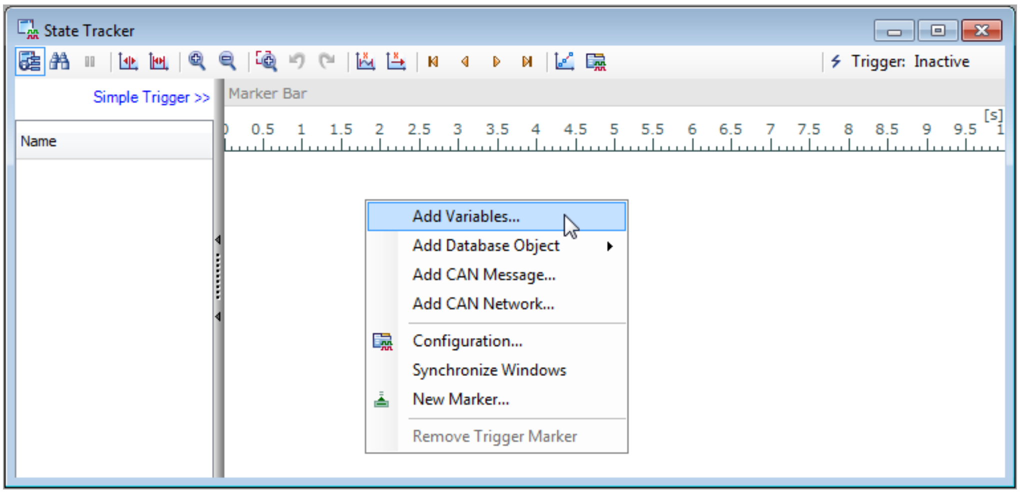
State Tracker Window

Open the **State Tracker Window** of CANoe.

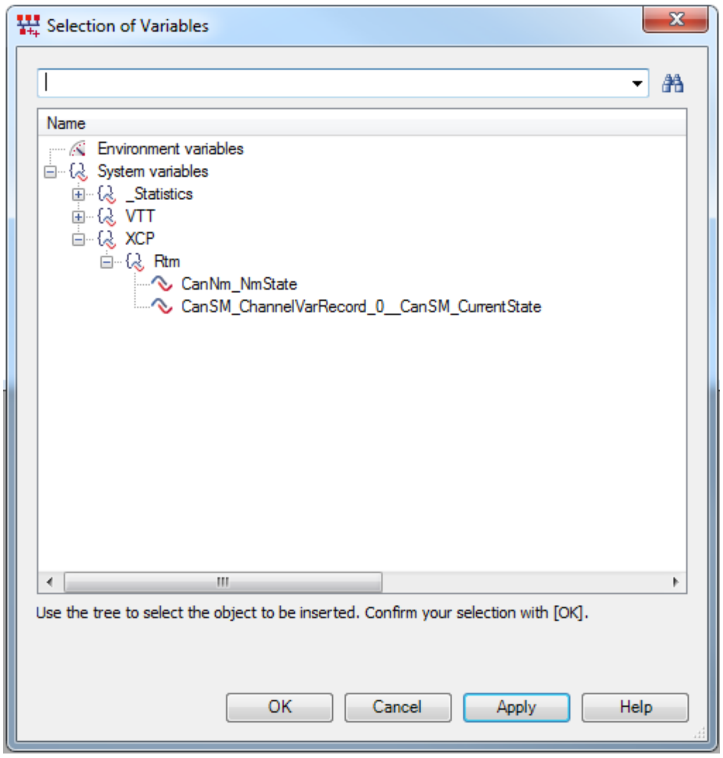


Add system variables of XCP

Right-click into the **State Tracker** window and click **Add Variables …** from the context menu to open the **Selection of Variables** window.

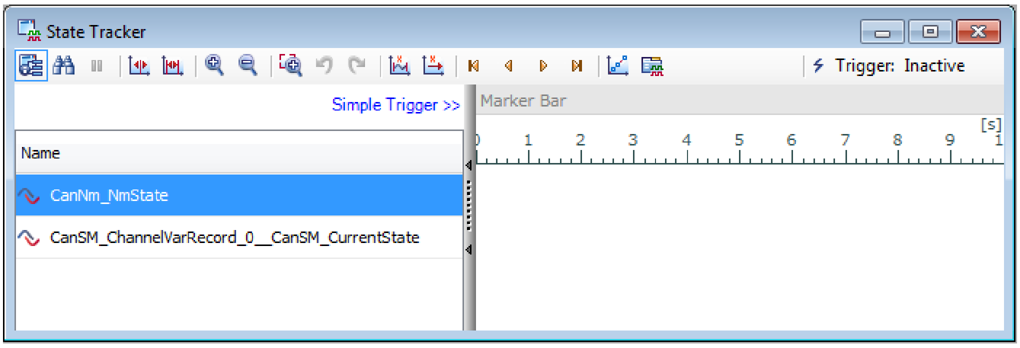


Selection of Variables



Use the tree to select the object to be inserted. Confirm your selection with [OK].

Scroll down to the XCP entry and find your previously selected variables, again. Select them and confirm with **[OK]**. Now they will be displayed on the left side of the **State Tracker** window and when the measurement is started, you can see the current values of the variables on the right side of the **State Tracker** window.



4.6.2 CANoe DLT

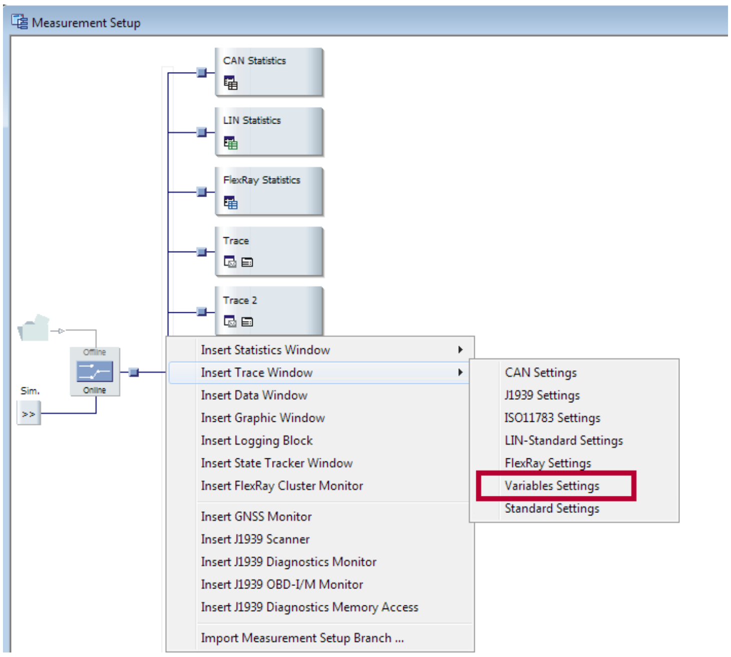
DLT variables in XCP

In the window **XCP/CCP** select all DLT measurement variables you want to be displayed. Activate the selected measurement signals by using the Configured checkbox.

DAQ

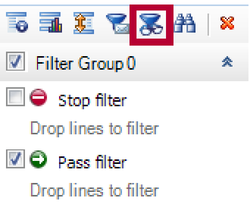
Activate **Auto Read**, select **DAQ** as **Measurement** and select **Dlt_EvtDem, Dlt_EvtDet, Dlt_EvtMsg** and **Dlt_EvtVMsg** for DEM, DET, non-verbose and verbose message reporting respectively.

Create DET window in CANoe



Filter settings

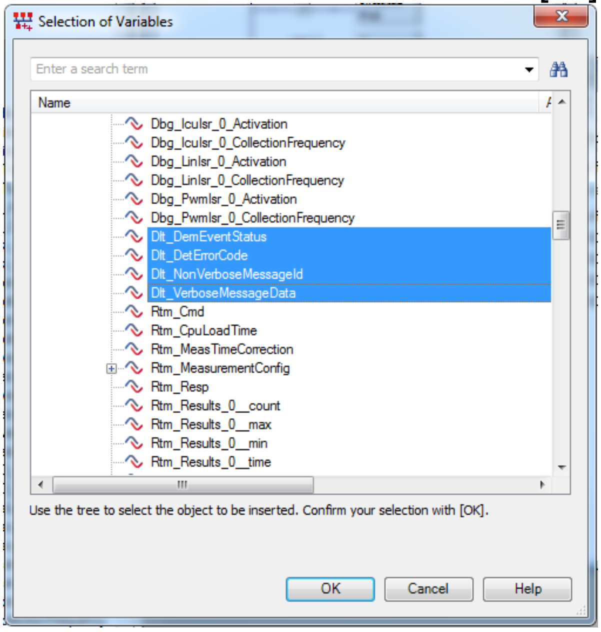
Define a Pass filter in the trace window.



Open the context menu of the pass filter to **add variables.**

Selection of Environmental Data

Select the DLT variables and confirm with [OK].



Dbg_IcuIsr_0_Activation
Dbg_IcuIsr_0_CollectionFrequency
Dbg_LinIsr_0_Activation
Dbg_LinIsr_0_CollectionFrequency
Dbg_PwmIsr_0_Activation
Dbg_PwmIsr_0_CollectionFrequency
Dlt_DemEventStatus
Dlt_DetErrorCode
Dlt_NonVerboseMessageId
Dlt_VerboseMessageData
Rtm_Cmd
Rtm_CpuLoadTime
Rtm_MeasTimeCorrection
Rtm_MeasurementConfig
Rtm_Resp
Rtm_Results_0__count
Rtm_Results_0__max
Rtm_Results_0__min
Rtm_Results_0__time

Use the tree to select the object to be inserted. Confirm your selection with [OK].

4.6.3 CANoe RTM

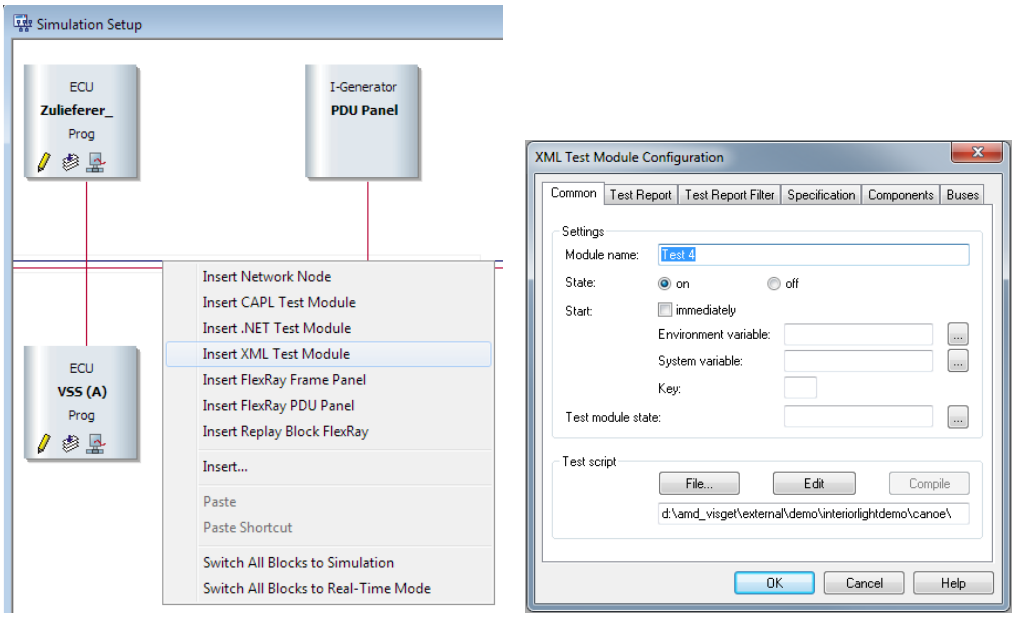
XCP Configuration

In the XCP/CCP window activate all RTM variables using the **Configured** checkbox. Activate **Auto Read** for **Rtm_Resp**, set **Measurement** to **DAQ** and set the **Cycle/Event** to **Rtm_Evt**.

Note: For **live measurement** set **all** signal entries to **DAQ**.

Simulation Setup

Open the **Simulation Setup** window via **Simulation|Simulation Setup**. Right-click to open the context menu and select **Insert XML Test Module**.



Settings

Module name: Test 4

State: ● on ○ off

Start: □ immediately

Environment variable:

System variable:

Key:

Test module state:

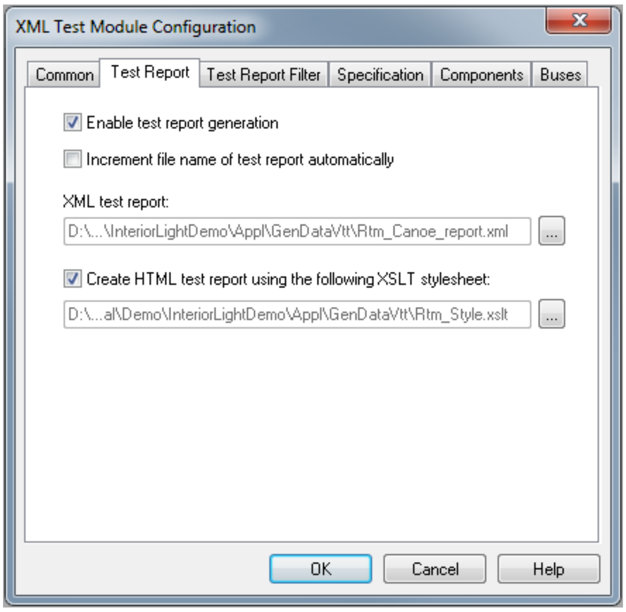
Test script

d:\amd_visget\external\demo\interiorlightdemo\canoe\

Enter a **Module Name** and click on **[File…]** to browse for the **Rtm_Canoe.xml** in the folder **GenData** of your project.

Report

Enable report generation. The stylesheet **Rtm_Style.xslt** is in the **GenData** folder of your project.



XML Test Module Configuration

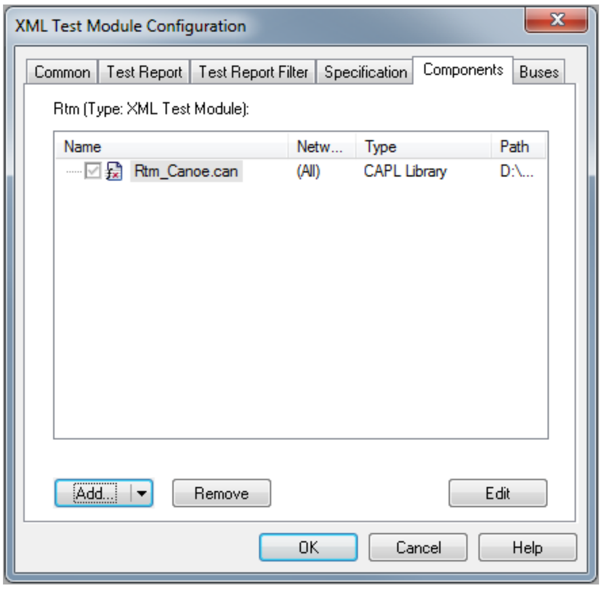
☑ Enable test report generation

□ Increment file name of test report automatically

XML test report: D:\...\InteriorLightDemo\Appl\GenDataVtt\Rtm_Canoe_report.xml

☑ Create HTML test report using the following XSLT stylesheet: D:\...al\Demo\InteriorLightDemo\Appl\GenDataVtt\Rtm_Style.xslt

Components In **Components** add the generated file **Rtm_Canoe.can** from your **GenData** folder.

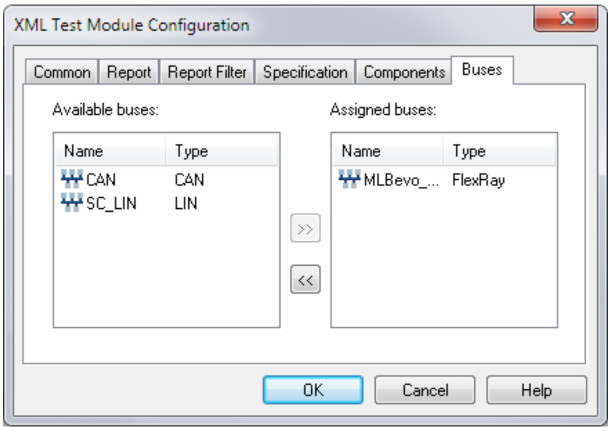


XML Test Module Configuration

Rtm (Type: XML Test Module):

Name	Netw...	Type	Path
☑ Rtm_Canoe.can	(All)	CAPL Library	D:\...

Buses Select your appropriate bus and move (**>>**) it to the **Assigned buses.** Confirm the window with **[OK]**.



Available buses:

Name	Type
CAN	CAN
SC_LIN	LIN

Assigned buses:

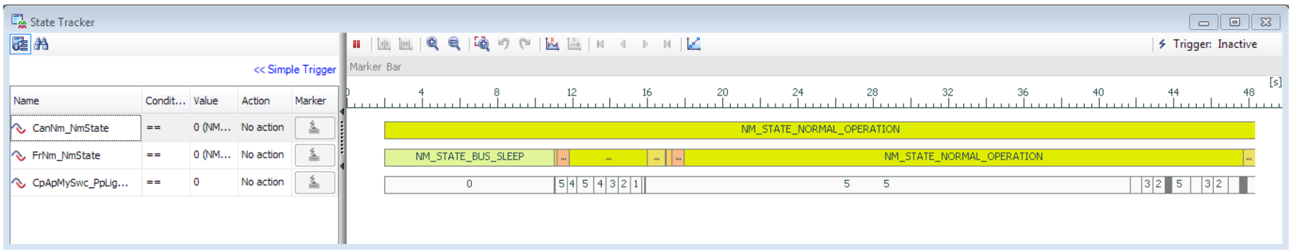
Name	Type
MLBevo_...	FlexRay

Open the window Now open the just configured window via **Test|Test Module|Rtm**.

4.7 STEP Measurement and Result

4.7.1 Measurement and Result of DBG

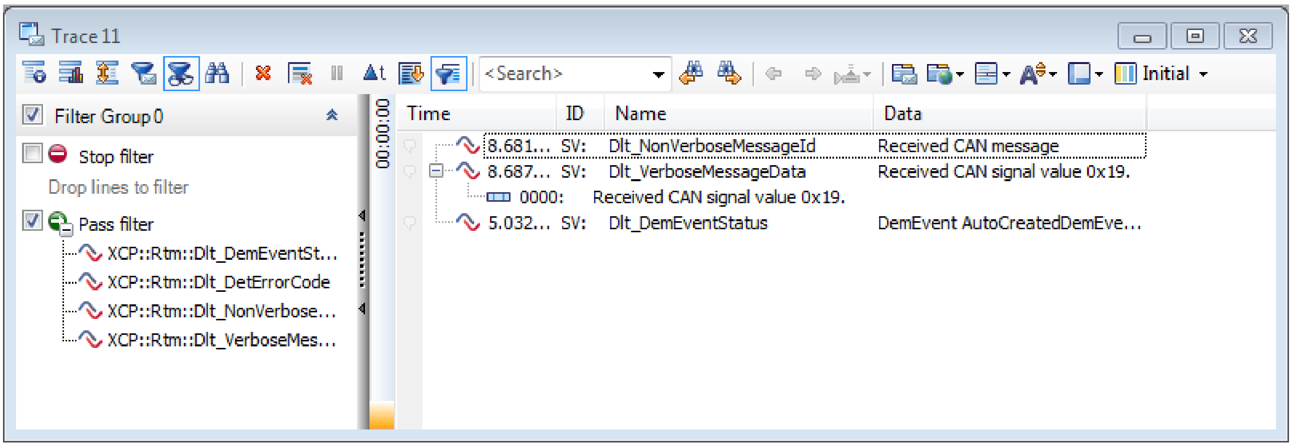
Watch the **State Tracker** window to observe the BSW variables you have entered before.



Name	Condit...	Value	Action	Marker
CanNm_NmState	==	0 (NM...	No action	
FrNm_NmState	==	0 (NM...	No action	
CpApMySwc_PpLig...	==	0	No action	

4.7.2 Measurement and Result DLT

Observe the DLT trace window to get informed about any DLT messages.



☑ Filter Group 0

☐ Stop filter

Drop lines to filter

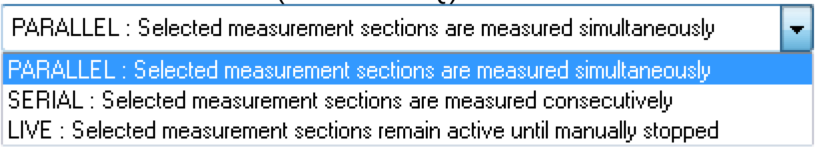
☑ Pass filter
- XCP::Rtm::Dlt_DemEventSt...
- XCP::Rtm::Dlt_DetErrorCode
- XCP::Rtm::Dlt_NonVerbose...
- XCP::Rtm::Dlt_VerboseMes...

Time	ID	Name	Data
8.681...	SV:	Dlt_NonVerboseMessageId	Received CAN message
8.687...	SV:	Dlt_VerboseMessageData	Received CAN signal value 0x19.
0000:		Received CAN signal value 0x19.	
5.032...	SV:	Dlt_DemEventStatus	DemEvent AutoCreatedDemEve...

4.7.3 Measurement and Result of RTM

Now select one of the following measurement method:

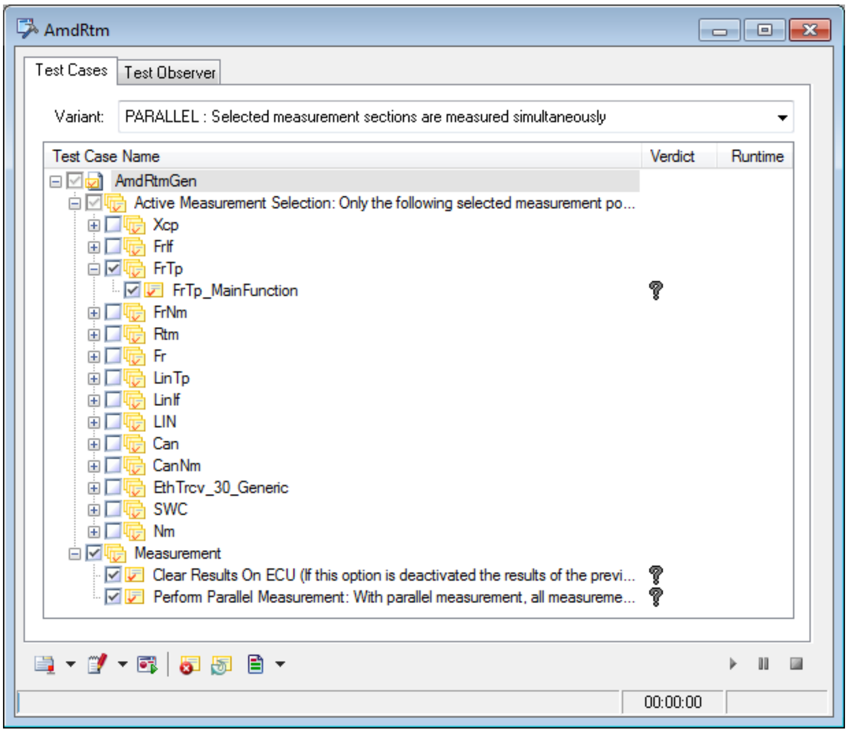
- Parallel
- Serial
- Live measurement (all set to DAQ)



PARALLEL : Selected measurement sections are measured simultaneously

PARALLEL : Selected measurement sections are measured simultaneously
SERIAL : Selected measurement sections are measured consecutively
LIVE : Selected measurement sections remain active until manually stopped

AmdRtm measurement window



Variant: PARALLEL : Selected measurement sections are measured simultaneously

Test Case Name	Verdict	Runtime
⊟ ☑ AmdRtmGen		
⊟ ☑ Active Measurement Selection: Only the following selected measurement po...		
⊞ ☐ Xcp		
⊞ ☐ FrIf		
⊟ ☑ FrTp		
☑ FrTp_MainFunction	?	
⊞ ☐ FrNm		
⊞ ☐ Rtm		
⊞ ☐ Fr		
⊞ ☐ LinTp		
⊞ ☐ LinIf		
⊞ ☐ LIN		
⊞ ☐ Can		
⊞ ☐ CanNm		
⊞ ☐ EthTrcv_30_Generic		
⊞ ☐ SWC		
⊞ ☐ Nm		
⊟ ☑ Measurement		
☑ Clear Results On ECU (If this option is deactivated the results of the previ...	?	
☑ Perform Parallel Measurement: With parallel measurement, all measureme...	?	

Select all measurement points that shall be activated in the measurement.

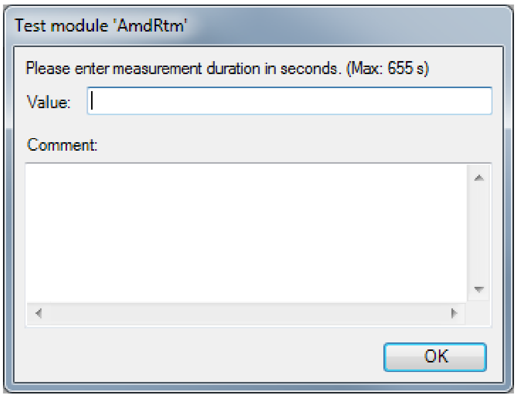
Start CANoe via **<F9>.**

Wait until the XCP Master (running on CANoe) has successfully connected to the XCP Slave (running on the ECU). Start the measurement via the red play button (see left) on the low right side of the window.

Parallel / Serial measurement

Parallel / Serial Measurement

For parallel and serial measurement the following window opens. Enter the duration of the measurement and click **[OK]** to start the measurement.



Please enter measurement duration in seconds. (Max: 655 s)

Value:

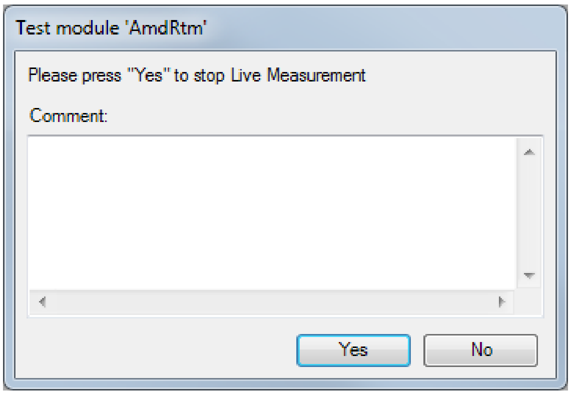
Comment:

Live measurement

Live Measurement

When live measurement is selected the measurement starts immediately with the click on the play button ▶ .

Click **[Yes]** to stop the measurement.



Test module 'AmdRtm'

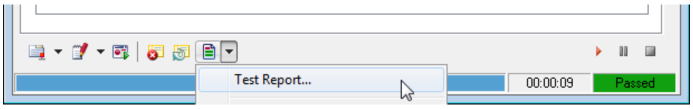
Please press "Yes" to stop Live Measurement

Comment:

Live observation during live measurement

Via context menu in a **Data** or **Graphics** window of CANoe you can add the values of your measurement (**Add Variables…**) and watch the values during the measurement.

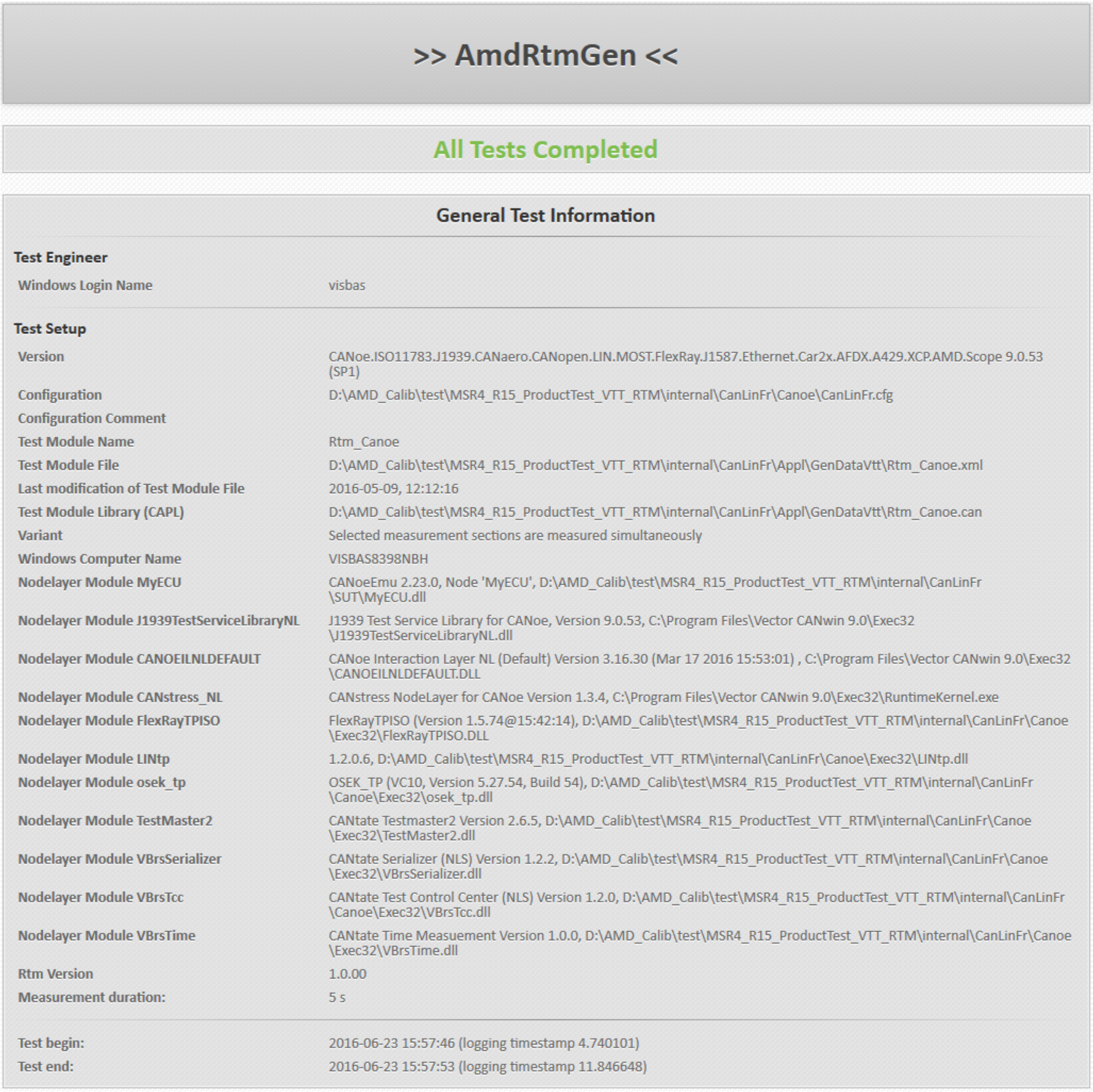
Show Report for Parallel or Serial Measurement



The report for serial and parallel measurement is automatically created. To view the report open the pull-down menu and select **Test Report**.

Report HTML of the parallel / serial measurement

HTML report of parallel and serial measurement

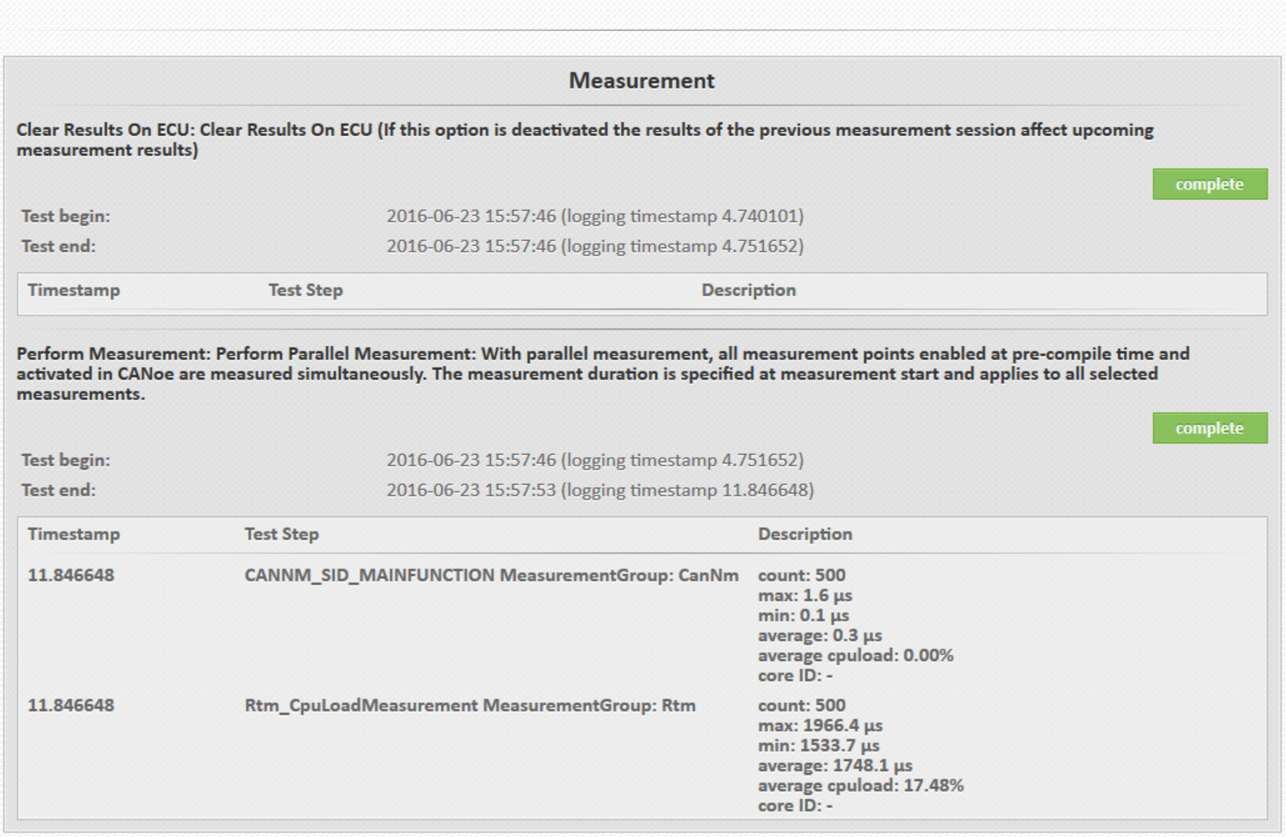


>> AmdRtmGen <<

All Tests Completed

General Test Information

Test Engineer	
Windows Login Name	visbas
Test Setup	
Version	CANoe.ISO11783.J1939.CANaero.CANopen.LIN.MOST.FlexRay.J1587.Ethernet.Car2x.AFDX.A429.XCP.AMD.Scope 9.0.53 (SP1)
Configuration	D:\AMD_Calib\test\MSR4_R15_ProductTest_VTT_RTM\internal\CanLinFr\Canoe\CanLinFr.cfg
Configuration Comment	
Test Module Name	Rtm_Canoe
Test Module File	D:\AMD_Calib\test\MSR4_R15_ProductTest_VTT_RTM\internal\CanLinFr\Appl\GenDataVtt\Rtm_Canoe.xml
Last modification of Test Module File	2016-05-09, 12:12:16
Test Module Library (CAPL)	D:\AMD_Calib\test\MSR4_R15_ProductTest_VTT_RTM\internal\CanLinFr\Appl\GenDataVtt\Rtm_Canoe.can
Variant	Selected measurement sections are measured simultaneously
Windows Computer Name	VISBAS8398NBH
Nodelayer Module MyECU	CANoeEmu 2.23.0, Node 'MyECU', D:\AMD_Calib\test\MSR4_R15_ProductTest_VTT_RTM\internal\CanLinFr\SUT\MyECU.dll
Nodelayer Module J1939TestServiceLibraryNL	J1939 Test Service Library for CANoe, Version 9.0.53, C:\Program Files\Vector CANwin 9.0\Exec32\J1939TestServiceLibraryNL.dll
Nodelayer Module CANOEILNLDEFAULT	CANoe Interaction Layer NL (Default) Version 3.16.30 (Mar 17 2016 15:53:01) , C:\Program Files\Vector CANwin 9.0\Exec32\CANOEILNLDEFAULT.DLL
Nodelayer Module CANstress_NL	CANstress NodeLayer for CANoe Version 1.3.4, C:\Program Files\Vector CANwin 9.0\Exec32\RuntimeKernel.exe
Nodelayer Module FlexRayTPISO	FlexRayTPISO (Version 1.5.74@15:42:14), D:\AMD_Calib\test\MSR4_R15_ProductTest_VTT_RTM\internal\CanLinFr\Canoe\Exec32\FlexRayTPISO.DLL
Nodelayer Module LINtp	1.2.0.6, D:\AMD_Calib\test\MSR4_R15_ProductTest_VTT_RTM\internal\CanLinFr\Canoe\Exec32\LINtp.dll
Nodelayer Module osek_tp	OSEK_TP (VC10, Version 5.27.54, Build 54), D:\AMD_Calib\test\MSR4_R15_ProductTest_VTT_RTM\internal\CanLinFr\Canoe\Exec32\osek_tp.dll
Nodelayer Module TestMaster2	CANtate Testmaster2 Version 2.6.5, D:\AMD_Calib\test\MSR4_R15_ProductTest_VTT_RTM\internal\CanLinFr\Canoe\Exec32\TestMaster2.dll
Nodelayer Module VBrsSerializer	CANtate Serializer (NLS) Version 1.2.2, D:\AMD_Calib\test\MSR4_R15_ProductTest_VTT_RTM\internal\CanLinFr\Canoe\Exec32\VBrsSerializer.dll
Nodelayer Module VBrsTcc	CANtate Test Control Center (NLS) Version 1.2.0, D:\AMD_Calib\test\MSR4_R15_ProductTest_VTT_RTM\internal\CanLinFr\Canoe\Exec32\VBrsTcc.dll
Nodelayer Module VBrsTime	CANtate Time Measuement Version 1.0.0, D:\AMD_Calib\test\MSR4_R15_ProductTest_VTT_RTM\internal\CanLinFr\Canoe\Exec32\VBrsTime.dll
Rtm Version	1.0.00
Measurement duration:	5 s
Test begin:	2016-06-23 15:57:46 (logging timestamp 4.740101)
Test end:	2016-06-23 15:57:53 (logging timestamp 11.846648)



Measurement

Clear Results On ECU: Clear Results On ECU (If this option is deactivated the results of the previous measurement session affect upcoming measurement results)

Test begin:	2016-06-23 15:57:46 (logging timestamp 4.740101)
Test end:	2016-06-23 15:57:46 (logging timestamp 4.751652)

Timestamp	Test Step	Description

Perform Measurement: Perform Parallel Measurement: With parallel measurement, all measurement points enabled at pre-compile time and activated in CANoe are measured simultaneously. The measurement duration is specified at measurement start and applies to all selected measurements.

Test begin:	2016-06-23 15:57:46 (logging timestamp 4.751652)
Test end:	2016-06-23 15:57:53 (logging timestamp 11.846648)

Timestamp	Test Step	Description
11.846648	CANNM_SID_MAINFUNCTION MeasurementGroup: CanNm	count: 500 max: 1.6 µs min: 0.1 µs average: 0.3 µs average cpuload: 0.00% core ID: -
11.846648	Rtm_CpuLoadMeasurement MeasurementGroup: Rtm	count: 500 max: 1966.4 µs min: 1533.7 µs average: 1748.1 µs average cpuload: 17.48% core ID: -

5 Appendix A: Addresses

Addresses on Vector homepage

Please find the contacts of Vector Informatik GmbH and all subsidiaries worldwide via:

http://www.vector.com/vi_addresses_en.html

6 Glossary

AMD	AUTOSAR Monitoring and Debugging – part of the Vector MICROSAR Product
DBG	Debugging Module
DLT	Diagnostic Log and Trace Module
RTM	Runtime Measurement Module

7 Index

More Information

- > News
- > Products
- > Demo Software
- > Support
- > Training Classes
- > Addresses

www.vector.com