Document Title	Requirements on Debugging in AUTOSAR
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	332
Document Classification	Auxiliary
Document Status	Final
Part of AUTOSAR Release	4.2.2

Document Change History		
Release	**Changed by**	**Change Description**
4.2.2	AUTOSAR Release Management	• Marked the document as obsolete
4.2.1	AUTOSAR Release Management	• Editorial changes
4.1.2	AUTOSAR Release Management	• Updated reference to RS feature document • Editorial changes
4.1.1	AUTOSAR Administration	• Updated the format of requirements according to TPS_StandardizationTemplate • Updated the chapters 2 and 5 • Replaced Complex Device Driver by Complex Driver
3.1.5	AUTOSAR Administration	• Initial release

Disclaimer

Advice for users

AUTOSAR specifications may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).

Any such exemplary items are contained in the specifications for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such specifications, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

Table of Contents

1 Scope of Document

This document describes the requirements for a debugging module in AUTOSAR, including the

- API
- configuration
- communication
- interactions to, and requirements on surrounding BSW modules

This specification is obsolete and will be removed from the standard in an upcoming release.

2 Conventions to be used

- The representation of requirements in AUTOSAR documents follows the table specified in [8].
- In requirements, the following specific semantics shall be used (based on the Internet Engineering Task Force IETF).

  The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as:

  - SHALL: This word means that the definition is an absolute requirement of the specification.
  - SHALL NOT: This phrase means that the definition is an absolute prohibition of the specification.
  - MUST: This word means that the definition is an absolute requirement of the specification due to legal issues.
  - MUST NOT: This phrase means that the definition is an absolute prohibition of the specification due to legal constraints.
  - SHOULD: This word, or the adjective "RECOMMENDED", mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
  - SHOULD NOT: This phrase, or the phrase "NOT RECOMMENDED" mean that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
  - MAY: This word, or the adjective „OPTIONAL“, means that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation, which does not include a particular option, MUST be prepared to interoperate with another implementation, which does include the option, though perhaps with reduced functionality. In the same vein an implementation, which does include a particular option, MUST be prepared to interoperate with another implementation, which does not include the option (except, of course, for the feature the option provides.)

3 Functional Overview

The goal of the debugging module is to offer as much information as possible about the runtime behavior of the systems, making it easier to spot the source of a problem when the integrated software does not behave as expected.

Main tasks of the debugging module are to

- Collect and store data for tracing purpose
- Collect and immediately transmit data to host
- Modify data in target memory on request from host
- Transmit stored data to host
- Accept commands to change the behavior of the debugging module

The debugging module shall

- be able to interface to ECU internal modules and to an external host system via communication
- be able to transmit data to the host system
- provide an internal buffer to decouple data collection from data transmission
- offer the possibility to analyze data post mortem

The debugging module collects raw data. All interpretation shall take place on the host side.

4 Requirements Specification

4.1 Functional Requirements

4.1.1 Configuration

4.1.1.1 [SRS_Dbg_00001] Description of semantics of data

[

Type:	valid
Description:	The debugging SWS shall define an XML structure to be used by the BSW modules to describe the semantics of data. The functionality shall be oriented on the OSEK ORTI description.
Rationale:	Give more information about the semantic of variables to support symbolic debugging.
Use Case:	Separation of data into bit fields and assignment of meanings to values is not part of a standard linker output (e.g. signal in an IPDU with multiple possible values). The debugging host needs this information to support the user adequately during a debugging session.
Dependencies:	WP General Methodology and Configuration
Supporting Material:	[2], [3]

] (RS_BRF_02240)

4.1.1.2 [SRS_Dbg_00002] Inclusion of BSW header files

[

Type:	valid
Description:	The debugging module needs to know the size of data items to be collected. To do so, the debugging module shall #include the needed header files of the BSW modules.
Rationale:	Only the address is always available in the Linker output file, not the size.
Use Case:	Calculate size of data item automatically with the C language ‘sizeof()’ operator.
Dependencies:	--
Supporting Material:	--

] (RS_BRF_02240)

4.1.1.3 [SRS_Dbg_00003] Static configuration of data items to be debugged

[

Type:	valid
Description:	The debugging module shall be configured statically. All data items or clusters of data items, which will be collected during a debugging session, shall be selected during configuration.
Rationale:	Simplify debugging module (no dynamic management of data items) and reduce bus load (possibility to assign simple static identifiers to clusters of data elements).
Use Case:	--
Dependencies:	--
Supporting Material:	--

] (RS_BRF_02256)

4.1.1.4 [SRS_Dbg_00039] Symbolic and physical configuration of data items to be debugged

⌈

Type:	valid
Description:	The configuration of the debugging module shall allow to symbolically configure global data items (calculate the address by a linker symbol, size by C sizeof() statement). For all data items, it shall also be supported to directly offer the address and size of the variable.
Rationale:	Support debugging of local data items, e.g. if a module has been compiled with compiler debugging information.
Use Case:	--
Dependencies:	--
Supporting Material:	--

⌋ (RS_BRF_02256)

4.1.1.5 [SRS_Dbg_00004] Static configuration of behavior of the debugging module

⌈

Type:	valid
Description:	It shall be possible to statically configure all behavior of the debugging module, which can be modified during runtime, to a default behavior.
Rationale:	When debugging the startup phase no communication to a host exists and debugging needs to start without any previous interaction. All configuration items need to have a sensible default value.
Use Case:	Switching on timestamps for debugging of startup phase
Dependencies:	[SRS_Dbg_00013], [SRS_Dbg_00023], [SRS_Dbg_00025]
Supporting Material:	--

⌋ (RS_BRF_02272)

4.1.1.6 [SRS_Dbg_00005] Behavior on internal buffer overflow

⌈

Type:	valid
Description:	It shall be possible to statically configure the behavior of the internal buffer in the case of an overflow. Two different strategies shall be supported: • Overwrite oldest • Discard newest
Rationale:	Support different use cases as described below.
Use Case:	In the case of ongoing debugging “overwrite oldest” is preferred. In the case of a snapshot, the stored data must stay intact for analysis.
Dependencies:	--
Supporting Material:	--

⌋ (RS_BRF_02272)

4.1.1.7 [SRS_Dbg_00038] Support for post build configuration

⌈

Type:	valid
Description:	All parameters which define addresses and sizes of items to be debugged shall be post built configurable
Rationale:	Changing the selection of accessible debugging data shall be made possible without re-linking the complete system
Use Case:	This allows to reconfigure the debugging session via re-flashing, as long as the rest of the system does not change, in case there is the need for other debugging data.
Dependencies:	--

Supporting Material:	--

] (RS_BRF_02256)

4.1.2 Initialization

4.1.2.1 [SRS_Dbg_00006] Debugging during system startup

[

Type:	valid
Description:	To be able to debug in the earliest possible state, the debugging module shall allow collection of data directly after the C initialization.
Rationale:	Support debugging at system startup time
Use Case:	Find errors in the startup code
Dependencies:	All functionality which assumes AUTOSAR OS to be up and running will only be available at a later stage.
Supporting Material:	--

] (RS_BRF_02256)

4.1.3 Normal Operation

4.1.3.1 [SRS_Dbg_00007] Collect data on a running ECU

[

Type:	valid
Description:	The debugging module should not intervene in the scheduling of the ECU and shall not halt it. It shall only collect data on a running ECU.
Rationale:	The impact of the debugging module on the runtime behavior of other BSW modules shall be as small as possible.
Use Case:	Being able to collect debug information while the normalsystem is up and running.
Dependencies:	--
Supporting Material:	--

] (RS_BRF_02256)

4.1.3.2 [SRS_Dbg_00008] Collect and store data for tracing purpose

[

Type:	valid
Description:	The debugging module shall be able to collect and store data, which can later be transmitted to a host for tracing purpose. An internal buffer shall be provided to decouple data collection from later data transmission.
Rationale:	Basic functionality of debugging; the debugging module shall be able to decouple collection and transmission of data. This allows to debug short data bursts, that otherwise could not be transmitted on the bus without data loss.
Use Case:	E.g. collect identifiers of running tasks, collect data before the communication bus is up, buffer data bursts
Dependencies:	--
Supporting Material:	--

] (RS_BRF_02272)

4.1.3.3 [SRS_Dbg_00009] Transmit stored data to host

[

Type:	valid
Description:	The debugging module shall on request of the host transmit the temporary stored data to the host.
Rationale:	Basic functionality of debugging

Use Case:	Transmit data, which was stored, while communication was unavailable
Dependencies:	--
Supporting Material:	--

] (RS_BRF_02272,RS_BRF_02240)

4.1.3.4 [SRS_Dbg_00010] Collect and immediately transmit data to host

[

Type:	valid
Description:	The debugging module shall be able to collect and immediately transmit data to a host for tracing purpose
Rationale:	Basic functionality of debugging
Use Case:	Same as [SRS_Dbg_00008] in the case no buffering in the ECU is possible
Dependencies:	Because it cannot be assumed that bus transmission will immediately take place, internal buffering of one item needs to be performed.
Supporting Material:	--

] (RS_BRF_02240)

4.1.3.5 [SRS_Dbg_00011] Enabling/disabling of data buffering

[

Type:	valid
Description:	The debugging module shall accept requests to dynamically switch on/off data buffering for each data item.
Rationale:	Limit bus load, memory consumption and run time overhead of the debugging module; make immediate transmission possible.
Use Case:	For high priority transmission buffering is bypassed.
Dependencies:	--
Supporting Material:	--

] (RS_BRF_02232)

4.1.3.6 [SRS_Dbg_00012] Collect data with automatic timestamp

[

Type:	valid
Description:	The debugging module shall on request collect a time stamp with a data item, and transmit both together.
Rationale:	Identify timing relations between different data items or events.
Use Case:	Measure time between function entry and exit, repetition rates, etc..
Dependencies:	--
Supporting Material:	--

] (RS_BRF_02224)

4.1.3.7 [SRS_Dbg_00013] Enabling/disabling of time stamping

[

Type:	valid
Description:	The debugging module shall accept requests to dynamically switch on/off time stamping for each data item.
Rationale:	Limit bus load, memory consumption and run time overhead of the debugging module
Use Case:	The user is not always interested in timestamps.
Dependencies:	--
Supporting Material:	--

] (RS_BRF_02240)

4.1.3.8 [SRS_Dbg_00015] Accept commands to change the behavior of the debugging module

⌈

Type:	valid
Description:	The debugging module shall offer a command interface to the host for controlling parameters of the debugging session. The exact commands shall be defined in the SWS, but shall at least cover: • Start of communication with the host (start of session) • Stop of communication with the host (stop of session)
Rationale:	Allows the user to control the debugging session
Use Case:	Switch between “data collection on” and “data collection off”
Dependencies:	--
Supporting Material:	--

⌋ (RS_BRF_02248)

4.1.3.9 [SRS_Dbg_00016] Selectable behavior on start of communication

⌈

Type:	valid
Description:	The command to start the communication shall support 2 options: • Discard all buffer contents and cancel pending communication requests of the debugging module (prepare for a new session) • Keep all buffered data and continue with communication request (continue last active session)
Rationale:	Allows the user to handle both situations described below.
Use Case:	Option 1 is standard use case. Option 2 is used to retrieve data collected during system startup, or during a short bus off.
Dependencies:	--
Supporting Material:	--

⌋ (RS_BRF_02248)

4.1.3.10 [SRS_Dbg_00017] Offer a public API for BSW modules

⌈

Type:	valid
Description:	The debugging module shall offer an API to BSW modules to accept data generated by the BSW modules at certain events. The events shall be described in the SWS, but at least cover: • Function entry and exit • RTE debugging events • Handling of DET errors
Rationale:	Ability to collect data on event from the BSW modules
Use Case:	E.g. function tracing
Dependencies:	--
Supporting Material:	[1], [5]

⌋ (RS_BRF_02224)

4.1.3.11 [SRS_Dbg_00018] Communication between debugging module and host

⌈

Type:	valid
Description:	The debugging module shall be able to communicate with a host system. The communication media to be supported shall be defined in the SWS. At least CAN, FlexRay and Serial Communication shall be supported.
Rationale:	Basic functionality of debugging
Use Case:	Use the most common bidirectional communication interfaces on ECUs

Dependencies:	--
Supporting Material:	--

] (RS_BRF_02248)

4.1.3.12 [SRS_Dbg_00019] Communication to one host only at a time

[

Type:	valid
Description:	The connection of debugging module(s) to host is always n:1; there is always only one host, that optionally can handle several target debug modules.
Rationale:	Simplify the debugging module
Use Case:	If a 2nd session is started while the previous session has not ended correctly, there is always the possibility that there was a problem on the host side. The debugging module will accept the new session.
Dependencies:	--
Supporting Material:	--

] (RS_BRF_02248)

4.1.3.13 [SRS_Dbg_00020] Support of post mortem analysis

[

Type:	valid
Description:	The debugging module shall offer the possibility to analyze data post mortem. In order to be able to do this, the structure of all data of the debugging module, which is needed for post mortem analysis, shall be specified in the SWS (e.g. structure of the internal buffer).
Rationale:	Support of debugging in the case of system break down
Use Case:	Support of offline analysis after an erroneous system state
Dependencies:	--
Supporting Material:	--

] (RS_BRF_02256)

4.1.3.14 [SRS_Dbg_00021] Debugging support for development phase only

[

Type:	valid
Description:	The debugging module shall not be prepared to be present in production code. Especially no specific security measures shall be implemented.
Rationale:	Simplify the debugging module and the communication protocol
Use Case:	Debugging shall cover development phase problems.
Dependencies:	--
Supporting Material:	--

] (RS_BRF_02232)

4.1.3.15 [SRS_Dbg_00022] Tracing of global variables

[

Type:	valid
Description:	The debugging module shall be able to trace all global (public) data with known size. It shall not depend on specific debug information of the compiler.
Rationale:	Basic functionality of debugging. It has to be assumed, that BSW modules are delivered as object code and are not compiled with debug information.
Use Case:	Necessary to access data using names (symbols). The names can either be extracted from the source code (if available), the object code (if compiled with debugging information) or the XML description (always)
Dependencies:	[SRS_Dbg_00001]
Supporting Material:	--

] (RS_BRF_02272)

4.1.3.16 [SRS_Dbg_00023] Enabling/disabling tracing of variables

[

Type:	valid
Description:	The debugging module shall accept requests to dynamically switch on/off the tracing of individual variables.
Rationale:	Limit bus load, memory consumption and run time overhead of the debugging module
Use Case:	The user is not interested on all possible data items at the same time.
Dependencies:	--
Supporting Material:	--

] (RS_BRF_02272)

4.1.3.17 [SRS_Dbg_00024] Periodic tracing of variables

[

Type:	valid
Description:	The debugging module shall be configurable to trace variables automatically. Tracing shall be done periodically.
Rationale:	In the case of frequent changes of the value of a variable only the debugging module can collect the data in sufficiently short intervals.
Use Case:	E.g. debugging during startup phase.
Dependencies:	[SRS_Dbg_00022]
Supporting Material:	--

] (RS_BRF_02272)

4.1.3.18 [SRS_Dbg_00025] Modify tracing period

[

Type:	valid
Description:	The debugging module shall accept requests to dynamically modify the tracing period to collect data items
Rationale:	Adapt to different needs during a debugging session.
Use Case:	Shorten time period to collect data items more frequently to identify the function, which modifies the data.
Dependencies:	--
Supporting Material:	--

] (RS_BRF_02272)

4.1.3.19 [SRS_Dbg_00026] Event based tracing of variables

[

Type:	valid
Description:	The debugging module shall offer an API to other BSW modules to trace variables on request.
Rationale:	The debugging module cannot detect changes in variables. If a variable change needs to be stored in the debugging module for later evaluation, this can only be done by the BSW module responsible for the change.
Use Case:	Change of running task by the OS
Dependencies:	[SRS_Dbg_00022]
Supporting Material:	--

] (RS_BRF_02272)

4.1.3.20 [SRS_Dbg_00027] Tracing of functions

[

Type:	valid

Description:	The debugging module shall offer an API to other BSW modules to trace function entry and exit.
Rationale:	Necessary to identify the function call sequence (execution profiling).
Use Case:	If an incorrect value is written into a data item, the function trace will help to identify the responsible function. Also the correct execution flow of the BSW can be checked.
Dependencies:	[SRS_Dbg_00017]
Supporting Material:	--

] (RS_BRF_02272)

4.1.3.21 [SRS_Dbg_00028] Tracing of software components behavior

[

Type:	valid
Description:	The debugging module shall adapt the interfaces offered by the RTE to signal changes in the behavior of the software components and process the offered data. Interfaces to be adapted are at least the following RTE events: • signal transmission • signal reception • COM callback • start runnable • terminate runnable Details shall be defined in the SWS.
Rationale:	Necessary to identify the runnables, which call the basic software, as well as to track ECU local communication.
Use Case:	See rationale
Dependencies:	[SRS_Dbg_00017]
Supporting Material:	--

] (RS_BRF_02272)

4.1.3.22 [SRS_Dbg_00029] Tracing of development errors

[

Type:	valid
Description:	The debugging module shall offer an interface to the DET to forward the information passed to the DET to the debugging module.
Rationale:	Development errors shall not only be available to the DET, but also to the debugging host.
Use Case:	Erroneous parameters in a BSW module function call
Dependencies:	[SRS_Dbg_00017]
Supporting Material:	--

] (RS_BRF_02240)

4.1.3.23 [SRS_Dbg_00030] Support for transparent memory access

[

Type:	valid
Description:	The debugging module shall offer the possibility to read or write data of arbitrary size from an arbitrary address.
Rationale:	Access to data with no symbolic name assigned (e.g. stack)
Use Case:	See rationale
Dependencies:	--
Supporting Material:	--

] (RS_BRF_02240)

4.1.3.24 [SRS_Dbg_00031] Transmission of data items exceeding frame length

[

Type:	valid
Description:	The debugging module shall be able to transmit data items exceeding the frame length of a specific bus. To do so, a simple disassembly/reassembly protocol shall be defined in the SWS. Communication bandwidth and runtime consumption shall be minimized. No error detection or retransmission is needed. Only a simple recovery mechanism is needed to resynchronize to the next data item.
Rationale:	Data items will often exceed the size of e.g. a CAN message. The information has to be transmitted in pieces. The usage of e.g. CanTp would add dependencies to the debugging module. By avoiding dependencies, the integrator can make use of the debugging module early.
Use Case:	See rationale
Dependencies:	--
Supporting Material:	--

] (RS_BRF_01648)

4.1.4 Shutdown Operation

The debugging module needs no specific action during shut down.

4.1.5 Fault Operation

4.1.5.1 [SRS_Dbg_00032] Handling of communication failure

[

Type:	valid
Description:	If a communication ‘send request’ of the debugging module fails, the request shall be repeated endlessly with a delay between the retries.
Rationale:	Simplify the debugging module. The host will detect a bus off situation on its side as well and restart the session if necessary.
Use Case:	Bus off
Dependencies:	--
Supporting Material:	

] (RS_BRF_02240)

4.2 Non-Functional Requirements

4.2.1 Timing Requirements

4.2.1.1 [SRS_Dbg_00033] Minimize runtime of the debugging module

[

Type:	valid
Description:	The debugging module on the ECU shall only perform functionality, which cannot be performed by the host. For functionality performed by the debugging module the SWS shall specify strategies, which minimize the runtime.
Rationale:	Minimize impact of debugging on the running ECU. The assumption is, that the host system has almost unlimited resources compared to the embedded target.
Use Case:	--
Dependencies:	--
Supporting Material:	--

] (RS_BRF_02256)

4.2.2 Resource Usage

4.2.2.1 [SRS_Dbg_00034] Minimize resource consumption of the debugging module

[

Type:	valid
Description:	The debugging module on the ECU shall only perform functionality, which cannot be performed by the host. For functionality performed by the debugging module the SWS shall specify strategies, which minimize the memory footprint and the communication bandwidth consumption.
Rationale:	Minimize impact of debugging on the running ECU and on the network.
Use Case:	--
Dependencies:	--
Supporting Material:	--

] (RS_BRF_02240)

4.2.3 Integration in AUTOSAR architecture

4.2.3.1 [SRS_Dbg_00035] Minimize dependency on other AUTOSAR BSW modules

[

Type:	valid
Description:	The debugging module shall use as few other AUTOSAR BSW modules as possible.
Rationale:	Debugging is intended to find problems in the AUTOSAR software stack. Therefore it has to assume, that these BSW modules are not fully functional, or even faulty.
Use Case:	Debugging of diagnostic stack: using the diagnostic stack in the debugging module would prevent this use case.
Dependencies:	--
Supporting Material:	[4]

] (RS_BRF_02240)

4.2.3.2 [SRS_Dbg_00036] Integration in AUTOSAR communication stack

[

Type:	valid
Description:	The debugging module shall interface to the PDU Router
Rationale:	Using the PDU Router gives access to all AUTOSAR supported communication media with a single interface.
Use Case:	--
Dependencies:	--
Supporting Material:	[4]

] (RS_BRF_01544)

4.2.3.3 [SRS_Dbg_00037] Separation between Main Debugging Module and communication part

[

Type:	valid
Description:	In case the debugging module needs to interface to a communication media not supported by the AUTOSAR architecture, the communication part needs to be replaced by a complex driver. To be able to do so, the debugging module shall be separated into a main part and a communication part. Clear

	interfaces shall be defined between these two parts.
Rationale:	Support of non AUTOSAR communication media.
Use Case:	µC specific interfaces like USB, serial interfaces (currently not supported), etc.
Dependencies:	--
Supporting Material:	[4]

] (RS_BRF_01544)

4.3 Not Applicable Requirements

4.3.1 [SRS_Dbg_99999] This requirement references all requirements which are not applicable for this SWS

[

Type:	valid
Description:	--
Rationale:	--
Use Case:	--
Dependencies:	--
Supporting Material:	--

](RS_BRF_02224,RS_BRF_02232,RS_BRF_02240,RS_BRF_02272, RS_BRF_02248,RS_BRF_02256)

5 References

[1] Specification of RTE Software,
AUTOSAR_SWS_RTE.pdf

[2] OSEK Run Time Interface (ORTI) Part A: Language Specification,
http://portal.osek-vdx.org/files/pdf/specs/orti-a-22.pdf

[3] OSEK Run Time Interface (ORTI) Part B: OSEK Objects and Attributes,
http://portal.osek-vdx.org/files/pdf/specs/orti-b-22.pdf

[4] Layered Software Architecture,
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

[5] Specification of Development Error Tracer,
AUTOSAR_SWS_DevelopmentErrorTracer.pdf

[6] General Requirements on Basic Software Modules,
AUTOSAR_SRS_BSWGeneral.pdf

[7] Requirements on AUTOSAR Features,
AUTOSAR_RS_Features.pdf

[8] Standardisation Template
AUTOSAR_TPS_StandardizationTemplate.pdf