Document Title	AUTOSAR BSW & RTE Conformance Test Specification Part 4: Execution Constraints
Document Owner	AUTOSAR GbR
Document Responsibility	AUTOSAR GbR
Document Identification No	284
Document Classification	Auxiliary

Document Version	1.0.0
Document Status	Final
Part of Release	3.0
Revision	0001

Document Change History			
Date	**Version**	**Changed by**	**Change Description**
14.11.2007	1.0.0	AUTOSAR Administration	Initial Release

Page left intentionally blank

Disclaimer

Advice to users of AUTOSAR Specification Documents:

AUTOSAR Specification Documents 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 Specification Documents for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such Specification Documents, 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 Overview of the document

This document is part of the process specifications for AUTOSAR BSW conformance testing. It describes the execution environment for the AUTOSAR BSW Conformance Test Specifications (CTSpecs) and its constraints. The objective is to provide the technical foundation and all information that allows the development and realization of a test infrastructure that is suitable for executing the AUTOSAR CTSpecs.
The overall CTSpec creation process is described in [4] and the methodology for its realization in [3].
The first part of this document (Chapters 2 and 3) describes the generic TTCN-3 test system architecture and its application to AUTOSAR. It further classifies the two possible test setups for AUTOSAR BSW conformance testing.
The second part (Chapters 4 and 5) contains the design overview for the Test Adapters and in particular the functional specification of the “target adapter”. The latter one makes the BSW module under test accessible for conformance testing. Within this functional specification, generic API functions to be provided by the target adapter are defined as well.

1.1 Purpose, Scope and Target Audience

1.1.1 Purpose and Scope

The main focus of this document is to mention all aspects that are related to enable the execution of the AUTOSAR conformance tests

- In a simulation environment (“Class A”) or
- Against a real embedded system running the BSW module implementation (“Class B”)

For a detailed definition of these classes of test setups, see Chapter 3.2.
It is not within the scope of this document to describe the process of properly executing the AUTOSAR conformance tests with the objective of officially certifying conformance of the BSW module under test. This certification process requires definitions of work steps and artifacts (such as the detailed recording of the parts of the CTSystem) that go beyond the definitions in this document.

1.1.2 Target Audience

This document set is intended to be used by any and all companies, groups and individuals engaged in the implementation of the execution environment for the AUTOSAR conformance tests. This document is the basic specification of this implementation. The content is applicable to both, the validation execution environment and the execution environment for conformance testing real BSW module implementations.
In order to completely understand this document, the documents [3] and [4] should have been read first.

1.2 Acronyms and Abbreviations

Abbreviation	Description
API	Application Program Interface
BSW	Basic Software
CC	Conformance Class
CD	Coder/Decoder (TTCN-3 – see Part 4)
CH	Component Handling (TTCN-3 – see Part 4)
CTA	Conformance Test Agency
CTSpec	Conformance Test Specification
CTS	Conformance Test Suite
CTSystem	Conformance Test System
ECU	Electronic Control Unit
FCC	Functional Conformance Class
ICC	Implementation Cluster Conformance Class
ICS	Implementation Conformance Statement
IP	Intellectual Property
PA	Platform Adapter (TTCN-3 – see Part 4)
PS	Product Supplier
RTE	Run Time Environment
SA	System Adapter (TTCN-3 – see Part 4)
SUT	System Under Test
SW-C	Software Component
SWS	Software Specification
TE	TTCN-3 Executable (TTCN-3 – see Part 4)
TM	Test Manager (TTCN-3 – see Part 4)
TRI	TTCN-3 Runtime Interface (TTCN-3 – see Part 4)
TTCN-3	Testing and Test Control Notation, version 3

1.3 Referenced Documents

[1] TTCN-3 specifications:
http://www.ttcn-3.org/StandardSuite.htm (accessed 20th of June, 2007)

[2] AUTOSAR BSW & RTE Conformance Test Specification Part 1: Background
https:/svn2.autosar.org/repos2/22_Releases
AUTOSAR_CTSpec_Background.pdf

[3] AUTOSAR BSW & RTE Conformance Test Specification Part 2: Process Overview
https:/svn2.autosar.org/repos2/22_Releases
AUTOSAR_CTSpec_Process_Overview

[4] AUTOSAR BSW & RTE Conformance Test Specification Part 3: Creation & Validation
https:/svn2.autosar.org/repos2/22_Releases

AUTOSAR_CTSpec_Creation_Validation

2 Generic TTCN-3 Test System Architecture

Figure 1 shows the generic architecture of a TTCN-3 based test system as defined in the TTCN-3 standard [1]. A series of TTCN-3 test cases is commonly referred to as a *TTCN-3 test suite*.

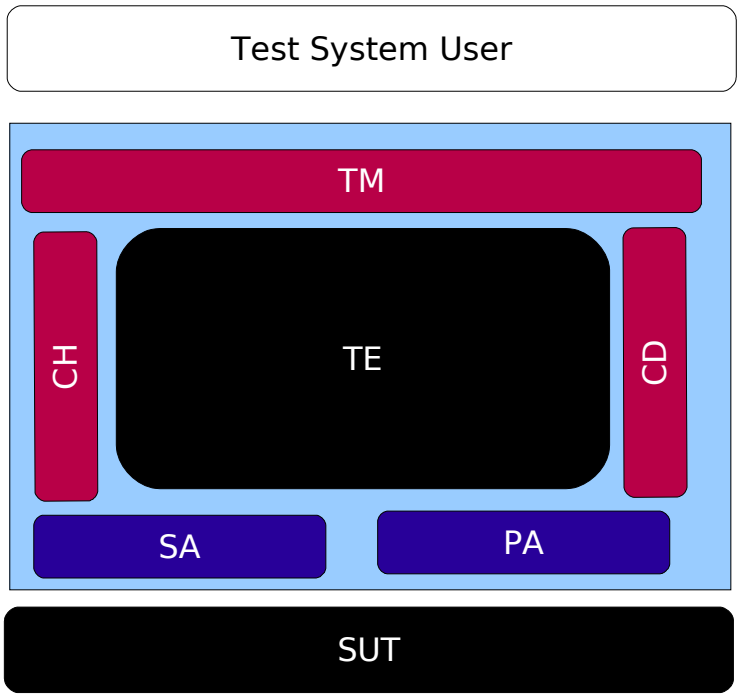

Figure 1 - TTCN-3 test system overview

An executable TTCN-3 test suite is thus made up of the following parts[1] (see Figure 1 and [1] for more details):

- **SUT** (System Under Test): The system that is tested.
- **TE** (TTCN-3 Executable): Contains the compiled TTCN-3 test cases and handles the execution of the test cases' statements.
- **SA** (SUT Adapter): Implements the interface between the TE and the SUT. It handles the sending and reception of all messages or procedure/function invocations between the TE and the SUT.
- **PA** (Platform Adapter): Realizes platform dependent functions, such as timers and TTCN-3 external functions.
- **CD** (Coder/Decoder): Transforms between TTCN-3 abstract data types used in the TE and concrete data types of the SUT.
- **TM** (Test Management): Controls the execution order of test cases in a test suite, logs all events from the TE, and implements the user interface of the test system.
- **CH** (Component Handling): Handles the creation, distribution and termination of TTCN-3 test components used in a test case.

[1] The abbreviations used for these parts are the same as in the TTCN-3 standardization documents.

The parts **TM** and **CH** are in general implemented and provided by the TTCN-3 tooling. When designing a TTCN-3 test system for a specific test job (such as AUTOSAR BSW conformance testing), the parts **SA**, **PA** and **CD** need to be specified. For AUTOSAR BSW conformance testing, the **TE** part is automatically generated by compiling the TTCN-3 test cases from the CTSpecs, and the **SUT** part is identical to the BSW module under test.

3 Specification of the Test System Architecture

3.1 The TTCN-3 Test Architecture for AUTOSAR Conformance Tests

3.1.1 Test Management

When executing conformance tests against an AUTOSAR BSW module under test, the user interacts with the TM as it implements the user interface. For execution of the CTSpecs, the user requires TM functionality on selecting and loading TTCN-3 files, and on selecting and executing the “control part” of TTCN-3 modules.
The logging and reporting functionality is of course also a crucial part of the TM as it delivers the results of the test execution and provides further information that is required when errors have occurred and need to be found.
However, TM functionality is generally implemented by the various TTCN-3 tools. Thus, the actual way of user interaction with the TM is tool specific.

3.1.2 Component Handling

A TTCN-3 test case contains at least one *test component* which resembles a self-contained thread that executes test steps. Complex test cases usually employ multiple test components that can potentially run on different software environments or hardware devices.
For AUTOSAR BSW conformance testing, test cases make use of multiple test components but they all run on the same software and hardware environment: the Tester PC.
Component Handling is a functionality provided by the TTCN-3 tool. Since for AUTOSAR BSW conformance testing, test components do not need to be distributed onto several software and hardware environments, the basic CH functionality provided by all TTNC-3 tools is sufficient. Since this basic CH functionality is usually hidden from the user, no further specification on CH is required.

3.1.3 TTCN-3 Executable

The TE is the compiled and ready-to-be-executed form of TTCN-3 test cases. Building the TE from the TTCN-3 test cases is usually a core functionality of TTCN-3 test systems. The handling of this build process is tool specific.
Since the execution behavior of the CTSpec is highly dependent on the configuration parameters, the TE has to be specifically generated for a set of configuration parameters. That is, for a new set of configuration parameters, the *TTCN-3 Config Module* has to be generated (see [3]) and added to the CTSpec which is then recompiled to generate a new TE.

3.1.4 System Under Test

For AUTOSAR BSW conformance testing, the SUT refers to the BSW module under test. Since the SUT is also highly configured by configuration parameters, a generation/compilation process is usually required to obtain the SUT for a specific configuration set.
The SUT is executed within an environment provided by the target platform. Prior to execution of the SUT, the BSW module has to be integrated into the software environment of the target platform. This includes in particular the

- integration of the SUT with the “target adapter” (see Chapter 4.1)
- correct linking with called functions as well as callback functions and
- provision of the scheduling scheme as specified in the specification for the BSW module (e.g. periodic calling of the main function of the BSW module).

Please note that the scheduling scheme can be overridden by the test case (see 5.2.3) if required.

3.1.5 Platform Adapter

The PA provides platform specific services to the TE with regard to timers and external functions.
For conformance testing of AUTOSAR BSW modules with no real-time behavior (e.g. the four modules dealt with in the CT pilot), the accuracy of the standard real-time clock of the Tester PC can be regarded as sufficient. Thus, the standard timer functionality of the PA commonly provided by TTCN-3 tools is used.
TTCN-3 external functions are functions that can be invoked from the test cases but are implemented in a programming language other than TTCN-3 (e.g. Java, C). The PA realizes the invocation and return of *TTCN-3 external functions*. They can be helpful in case when special functionality is required by a test case (e.g. fast calculation of a CRC). If an AUTOSAR BSW CTSpec makes use of *TTCN-3 external functions*, the external function has to be carefully specified and should be easily portable.

3.1.6 Coder/Decoder

The CD is responsible for converting data between their abstract representation in the test cases and their concrete representation in the SUT.
In AUTOSAR BSW, basic data types (e.g. signed und unsigned integers of various bit width), enumeration types, pointers and structures based on the aforementioned types are used.
Conversion of the basic data types is simple and might involve a conversion between little-endian and big-endian representation of data.
Within the C source of BSW modules, enumeration values are mapped to unsigned integer values. For TTCN-3 conformance test cases, AUTOSAR enumeration types are also mapped to unsigned integer. This means, that the CD can handle enumeration types as unsigned integer resulting in the simple conversion described above.

Pointers are in general handled transparently by the test cases. That means, the whole pointer value (i.e. the memory address), regardless of its bit width, is handled by a conformance test cases when invoking or receiving APIs of a BSW module containing pointer parameters or pointer return values. However, the test case cannot directly evaluate these pointer values. It has to use specific target adapter functions to process and evaluate pointer values when necessary as described in Section 5.3.

3.1.7 SUT Adapter

The SA connects the TE with the SUT. It transforms the communication operations (i.e. stimulation and observation activities) from within a TTCN-3 test case into appropriate operations towards the SUT.

3.2 Possible Test Setups

For conformance testing of AUTOSAR BSW modules, two different basic test setups are possible which have been identified and described in Chapter 5 of [2] and are referred to as “Class A” test setup and “Class B” test setup. The CTSpec is implemented and applicable to both test setups.

3.2.1 Class A Test Setup

With the Class A test setup, the BSW module under test (i.e. the SUT) is executed within a simulation environment on a PC which is (for the sake of simplicity) in general the same PC that executes the conformance test cases. With this type of test setup, hardware related functionality can often not be tested unless the simulation environment also provides a simulation of the hardware that is interfacing with the SUT.
However, providing a simulation of a hardware component and correctly interfacing it with the SUT usually involves high efforts and might often be abandoned in favor for the Class B test setup with real hardware.
Therefore, Class A test setups are primarily expected to be used for BSW modules that have interfaces to other software modules only and do not interface with hardware components. For those types of BSW modules, Class A tests provide the possibility to execute conformance tests without the hassle of setting up and handling an embedded target system with its required hardware components.

3.2.2 Class B Test Setup

Class B test setups require the execution of the BSW module under test on the embedded target system. In general, this type of test setup promises more reliable results on conformance of a BSW module since the BSW module is executed under conditions that are potentially very close to those of the later production use.

Figure 2 shows the Class B test setup consisting of the test system made up of its different parts, the target platform with the target adapter and the communication means between test system and target platform.
However, the separation between test system and target platform (i.e. an embedded system) introduces additional complexity:

- A communication means and scheme between test system and target platform is required.
- A “SUT adapter” (or “upper test adapter”) on the Test PC side is required. It is used to give the test case execution access to the communication means.
- A “target adapter” (or “lower test adapter”) that provides this communication functionality on the target system side and interacts with the BSW module is required.
- Both “SUT adapter” and “target adapter” need to be properly configured with respect to the properties of the communication means and the SUT.

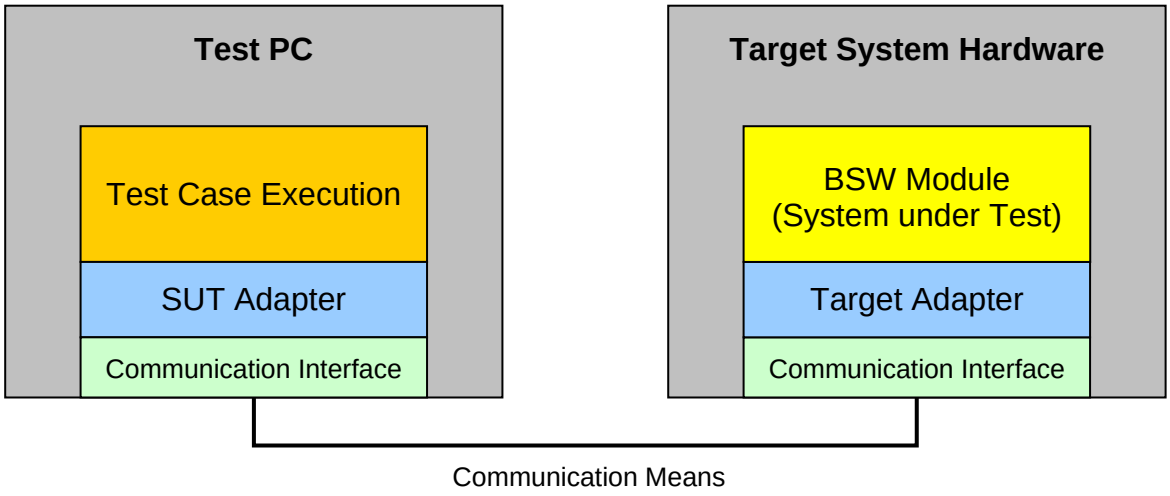

Figure 2 - Class B test setup

3.2.3 Class A vs. Class B from the view point of the CTSpec

The conformance test cases should be as much as possible independent from the type of test setup that is chosen for execution of the tests. In this way, the CTSpecs can be used at a wide range of the development process (e.g. Class A test setup during an early development stage when the module is developed within a PC based simulation environment).
Meeting this objective is very well supported by the TTCN-3 concept of “test adapters” which can be regarded as an “intermediate layer” between the test case execution and the SUT. The purpose of the test adapter is to provide the test cases with an “abstract” test interface. This concept is intended to be used for abstraction issues as, in this case, to abstract as much as possible from the type of test setup being used (Class A or Class B).

The following sections discuss the most relevant aspects of the CTSpec with regard to their handling and realization by a Class A or Class B test setup

3.2.3.1 Software APIs

Within the conformance test cases, the APIs of the BSW modules that interface with other software components are handled in the same way for Class A and Class B test setups.
For Class A test setups, it is the responsibility of the test adapter integrated with the simulated execution environment for the BSW module to correctly forward API calls, callbacks and their parameters to the BSW module.
For Class B test setups, the test adapter together with the target adapter has to realize the forwarding of API calls, callbacks and their parameters. This functional requirement on the target adapter is described in more detail in Chapter 5.2.

3.2.3.2 Hardware interfaces

With a Class B test setup, the BSW module interfaces with hardware (mostly hardware driver modules), for example, by directly accessing specific memory registers and with the help of interrupt routines. These mechanisms are hardware specific and internal to the BSW implementation.
When the same BSW module implementation is to be executed in a simulation environment for Class A tests, the same mechanisms for hardware interaction need to be provided by the simulation environment.
Since currently, hardware interfaces of BSW modules are not directly relevant for conformance testing, the issues associated with handling these interfaces for Class A and Class B test setups are not further analyzed within this document.

3.2.3.3 Configuration of the SUT

The configuration parameters for the BSW module under test as specified by AUTOSAR are in general valid for both Class A and Class B tests. For both kinds of tests, the configuration parameters have to be used in the same way for parameterizing the BSW module during generation and for parameterizing the conformance test cases.
Some AUTOSAR configuration parameters may relate to hardware interfaces (see Chapter 3.2.3.2). The handling of these parameters with respect to Class A and Class B tests needs to be analyzed within the issue of interfacing with hardware and is out of scope of this document.

3.2.3.4 Invocation of the SUTs Main Function

For both Class A and Class B test setups, the invocation of the SUTs main function generally needs to be controlled by the conformance test cases. Additionally, an “automatic” way of invoking the SUTs main function is specified (see Chapter 5.2.3) in order to simplify test case design.

Thus, both the test adapter for Class A test setup and the target adapter for Class B test setup are required to have control over the invocation of the SUTs main function.

3.2.3.5 Test Adapters

For both Class A and Class B test setups, the test adapters need to be implemented specifically for the components of the test environment. In Chapter 4, functional requirements on the test adapters are given. However, it is not in scope of this document to present a detailed design or an implementation specification for these adapters.
For Class A test setups, the design and implementation of the test adapters mainly depends on the interfaces and properties of the simulation environment.

- For Class B test setups, the design and implementation of the SUT adapter and the target adapter mainly depend on the communication means

4 Design Overview of the Test Adapter

As already described in Chapter 3.2 and depicted in Figure 2, the test adapters consist of both the SUT adapter and the target adapter.

4.1 Integration with Execution Environment

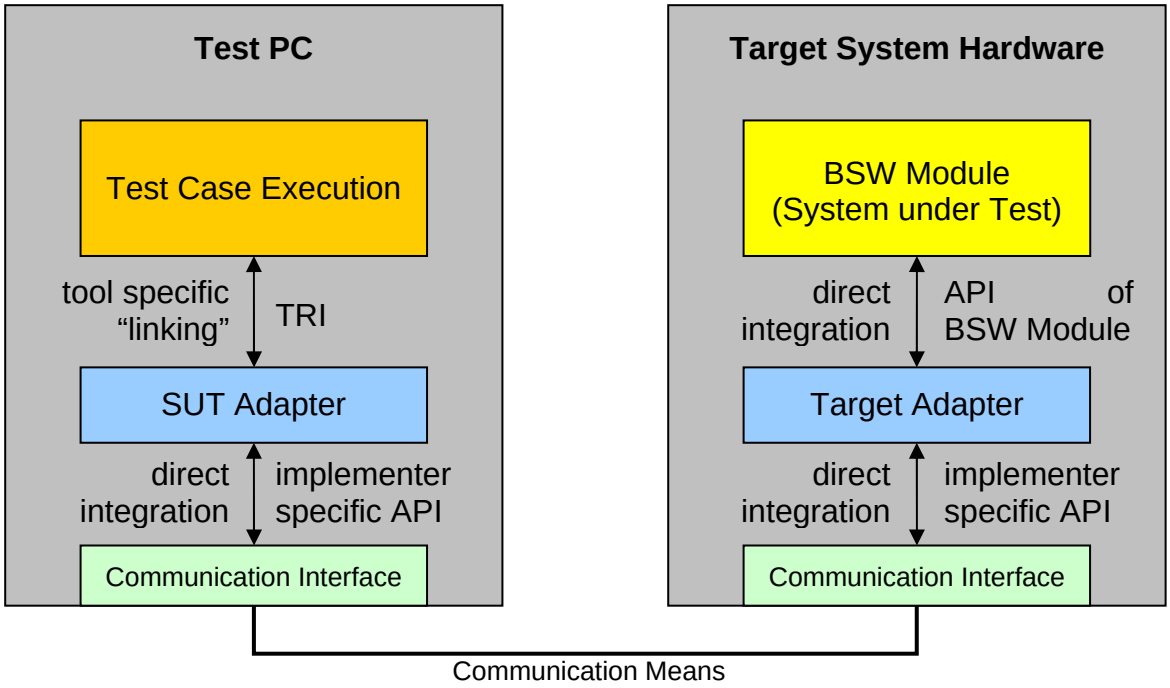

Figure 3 - Integration interfaces for SUT adapter and target adapter

As shown in Figure 3, the SUT adapter and target adapter need both to be integrated with their execution environments:

- SUT adapter:
  - The integration with the test execution is realized by implementing the "TTCN-3 Runtime Interface" (TRI). The TRI is specified in [1]. Implementing the TRI involves implementing TRI functions to be used by the test execution and making use of TRI functions that are provided by the test execution. How the SUT adapter and the test execution are actually "linked" together is tool specific.
  - The integration with the communication interface needs to be done in a direct way (i.e. compiled and linked together). In general, the SUT adapter implements directly the communication functions (e.g. socket functions for TCP/IP communication). Alternatively, an implementer specific communication API can be used here.

  Target adapter:
  - The integration with the BSW module needs to be done directly using the API specified for the BSW module. That means, the software struc-

ture of the target adapter must implement direct calls to API functions of the BSW module and must provide function stubs to be called as API call-backs by the BSW module.

- The integration with the communication interface is the same as for the SUT adapter.

4.2 Communication Means

The communication means consists of both the hardware realizing the data transfer (i.e. Ethernet adapters and cabling) and the software/protocols that provides the communication services (e.g. Ethernet driver, TCP/IP protocol).
With respect to AUTOSAR BSW conformance testing, the main purpose of the communication means that interconnects the SUT adapter with the target adapter is to efficiently transmit

- function calls and their parameters (if defined),
- function returns with return value (if defined)

in both directions.
The following “quality of service” for this communication means is required:

- Communication events (i.e. function calls and function returns) must not get lost during transmission.
- The content of communication events must not be altered.
- Reordering of communication events must not happen.
- Delay and jitter for transmission of communication events must be magnitudes of order smaller than the smallest time-out value in the BSW module or in the CTSpec.

How the communication means is realized is implementer specific and out of scope of AUTOSAR standards.

4.3 SUT Adapter

The main functionality of the SUT adapter is to forward communication events (i.e. function calls and function returns) between test execution and target adapter by making use of the communication means. In order to realize an efficient communication, this forwarding functionality usually involves encoding and decoding of the function names, their parameters and the possible return value. This main functionality is illustrated in Figure 4.

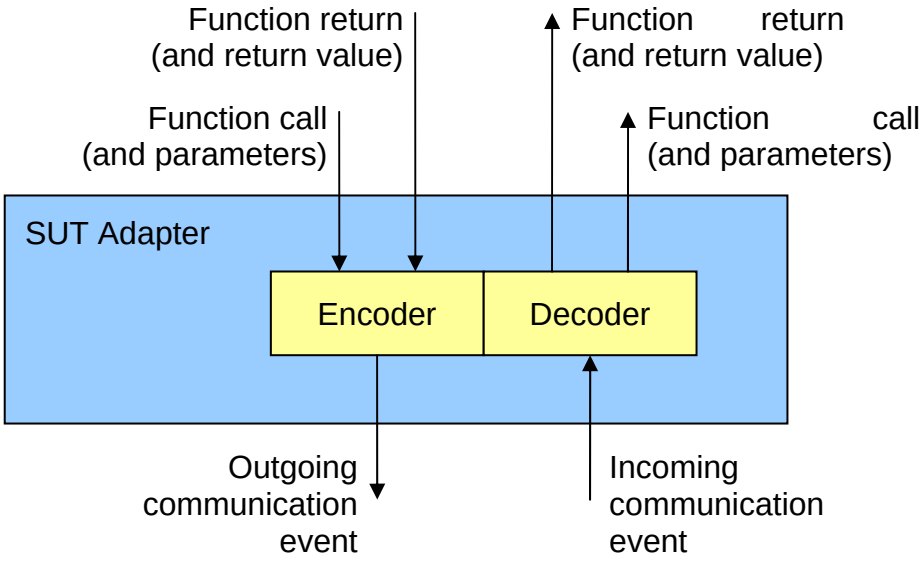

Figure 4 - Overview on the main functionality of the SUT adapter

The encoding/decoding strategy and scheme is implementer specific and out of scope of AUTOSAR standards.

4.4 Target Adapter

The target adapter must be able to apply the encoding/decoding scheme that the SUT adapter uses to correctly convert communication events received from the SUT adapter into appropriate function calls or function returns towards the BSW module and vice versa.
In addition to that, the target adapter must implement the following active functionality:

- Custom test functions (“target adapter functions”) that the test cases might use (e.g. for interacting directly with the memory of the target system).
- Automatic cyclic invocation of the main function of the BSW module (can be activated/deactivated by target adapter functions)

Figure 5 gives an overview on the main functionality of the target adapter. The target adapter functionality is specified in more detail in the following chapter.

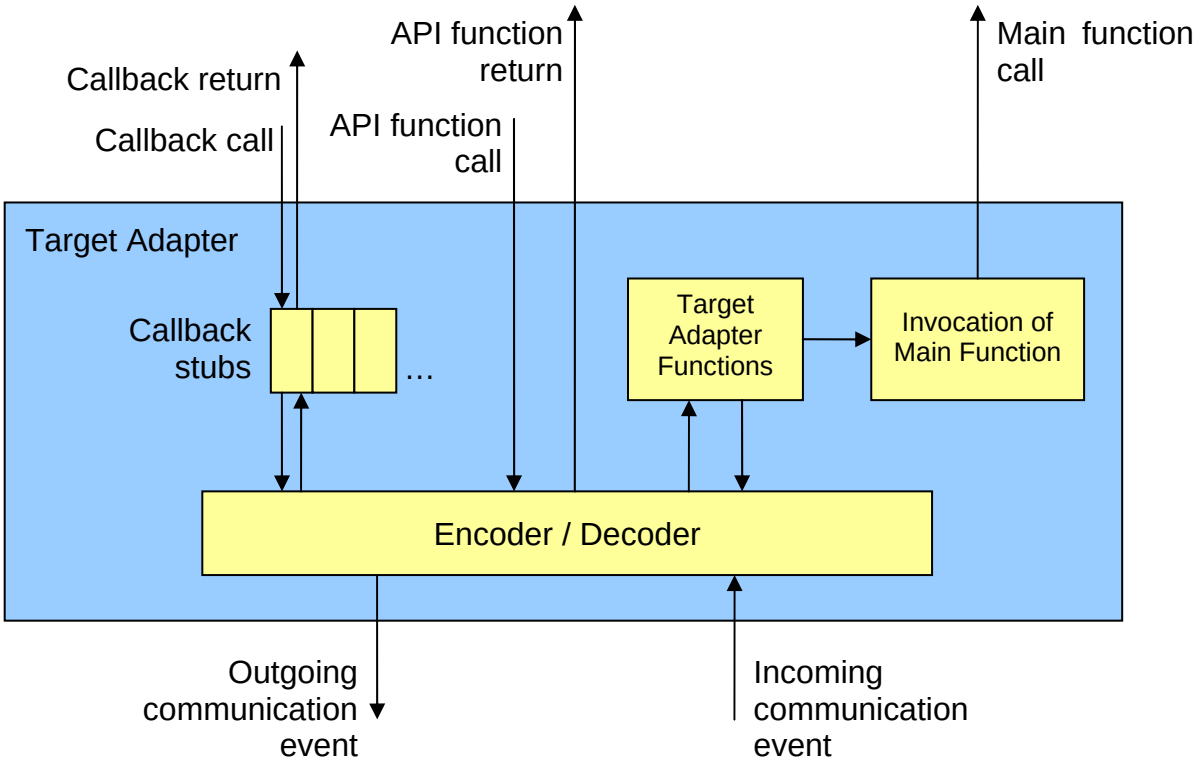

Figure 5 - Overview on the main functionality of the target adapter

5 Functional Specification of the Target Adapter

The target adapter (TA) refers to the lower part of the test adapter and runs on the target platform (Class B tests only). Therefore, it is usually specifically implemented for the target hardware.
It also may contain custom functions for testing certain BSW module functionality. The strategy for designing and adding these custom functions is described in Section 5.4.
Prior to the execution of the BSW conformance tests, the test personnel has to integrate the TA on the target platform. An overview on the integration issues has been given in Chapter 4.1.

5.1 Execution Model for the BSW Module Under Test

A fundamental specification issue is the execution model for the BSW module under test. This issue has a wide impact on AUTOSAR BSW conformance testing including the test case design and implementation, the target adapter design and implementation, the test integration and the overall potential for functional test coverage.
For all AUTOSAR BSW modules, a “main function” is specified together with a specification of its scheduling. For a large number of BSW modules this scheduling is a cyclic one with fixed or variable periodicity (e.g. NVRAM Manager, EEPROM Driver). There are also BSW modules that have a scheduling based on preconditions (e.g. EEPROM Abstraction). Furthermore, some BSW modules that offer synchronous functions only (e.g. Memory Abstraction Interface) do not require a “main” function and therefore have no specification on its scheduling.
Since the majority of BSW modules specify a main function with cyclic invocation, a pragmatic but sufficiently powerful approach is applied here:
The target adapter provides two modes for invoking the main function of the BSW module under test:

- Controlled Mode
- Automatic Cyclic Mode

The default mode is the Controlled Mode. During test case execution, the mode can be switched multiple times at arbitrary points in the execution flow.

5.1.1 Controlled Mode

Whenever the Controlled Mode is active, the target adapter does not invoke the main function by itself. It is rather the responsibility of the test case to invoke the main function at meaningful points in the execution flow of the test case.
Having the control over invocation of the main function enables testing of requirements that are tightly related to single invocations of the main function.

5.1.2 Automatic Cyclic Mode

When Automatic Cyclic Mode is switched on, the target adapter takes over the cyclic invocation of the main function. Once the target adapter has been put into Automatic Cyclic Mode, it periodically invokes the main function of the BSW module under test.

5.1.3 Example for code framework on main function invocation

The following code framework gives an example on how the controlled mode and the automatic cyclic mode can be implemented when only one execution thread is available for both target adapter and BSW module's functionality:

```
while (test case is running)
{
  if ("Automatic Cyclic mode is active" &&
      "time since last call to main function" >= periodicity)
  {
    Module_MainFunction();
  }
  // Handle incoming communication events from the tester.
  // Give other tasks a chance to be executed.
}
```

When the automatic cyclic mode is active, the main function is called with an approximately fixed periodicity. Between the calls to the main function, the target adapter has to handle the incoming communication events from the tester.
Additionally, it is necessary to allow for the execution of other processes (e.g. OS tasks) that might otherwise not be able to preempt the illustrated `while` loop.

5.1.4 Notion of Time

A consequence of interaction between Tester PC and the BSW module under test is that the notion of time for the main function does not correspond with the real notion of time. More precisely, during the invocation of the main function (independent of the mode), communication with the test environment might occur that usually involves request/reply message exchanges with the Tester PC. This message exchange has no deterministic time limit and therefore the execution of the main function can potentially be halted for an indefinite amount of time.
For BSW modules that have no real-time dependency on their environment (e.g. NVRAM Manager), this approach is appropriate. For these modules, the "time stands still" when communication with the Tester PC takes place.
For other modules that rely on time-dependent functionality provided by other software or hardware components, this approach might not be suitable. In this case, a more suitable approach is the realization of a fixed scaling factor between "test time" and real time. That means, the SUT is executed in a simulated time environment which is, for example, 100 times slower than the real time.

5.2 Generic Functional Requirements

The target adapter has to provide the basic functionality described in the following sections.

5.2.1 Start-up

The target adapter is started after power-up of the embedded system. Since the default execution mode is the Controlled Mode, the main function of the BSW module under test is not called unless it has been explicitly invoked by the test case or the test case has switched the mode to Automatic Cyclic Mode.
After start-up of the SUT, the test case will in general call the initialization API as first test step.

5.2.2 Shutdown

As specified in the TTCN-3 standard, the target adapter does not get an explicit notification when the test case execution has terminated. Rather the test case itself is responsible to create the post-conditions at the SUT after the main test steps have been executed.
Since conformance testing usually requires the sequential execution of a large number of test cases, the post-conditions to be created at the end of a test case must result in an initialized SUT of a valid “idle” state which resembles the state after power-up and initialization of the SUT. In this way, the subsequent execution of other test cases is possible without hard-resetting of the SUT.

5.2.3 Automatic or Controlled Invocation of the Module’s Main Function

The target adapter can be set to `AutomaticCyclicMode` by invoking the target adapter function `TA_EnableCyclicMainFunction()`. In this mode, the target adapter has to periodically call the main function as described in Chapter 5.1.2.
The execution mode can be set back to Controlled Mode by invoking the target adapter function `TA_DisableCyclicMainFunction()`.

5.2.4 Invocation of API Calls at the SUT

During test case execution, the SUT is stimulated by invocation of its APIs.
The test executable sends an API invocation together with optional arguments to the target adapter which then calls the API with the optional arguments at the BSW module. After execution of the API call, it returns with an optional return value (depending on the specification of the API). When the API call returns, the target adapter sends the appropriate notification to the test executable together with the optional return value.

5.2.5 Reception of API Calls from the SUT

During test integration, the test integrator has to make sure that the invocation of APIs belonging to other modules and called by the SUT is redirected to the target adapter.
When the test case is executed, the SUT may invoke these APIs belonging to other modules but provided by the target adapter. Upon reception of an API invocation from the SUT, the target adapter forwards the API invocation together with optional arguments to the test executable. The test case receives this API invocation and reacts to it according to the implemented test steps. This reaction usually involves sending of a reply notification to the target adapter indicating the return of the API invocation with optional return values.

5.2.6 Invocation of Callbacks towards the SUT

The BSW module under test may provide callbacks to other modules, e.g. for notification purposes.
For conformance testing, these callbacks are issued by the test case. Since callbacks towards the SUT behave exactly like APIs provided by the SUT, the same mechanism as described in Section 5.2.4 applies for callbacks towards the SUT.

5.2.7 Reception of Callbacks from the SUT

The BSW module under test may invoke callbacks towards other modules.
Again, the callbacks provided by other modules behave exactly like APIs provided by other modules and invoked by the SUT. Therefore, the same mechanism as described in Section 5.2.5 applies for callbacks invoked by the SUT towards other modules.

5.2.8 Reception of Error Reports

During test case execution, development errors and/or diagnostic events may occur within the BSW module under test. These errors and events are received by the target adapter and forwarded to the test executable where they are logged and a proper action (e.g., the setting of test verdict “fail”) is concluded from them.
The mechanisms for error and event reporting by the SUT make use of the APIs `Det_ReportError()` and `Dem_ReportErrorStatus()` provided by the modules “Development Error Tracer (DET)” and “Diagnostic Event Manager (DEM)”, respectively.
For conformance testing, the target adapter provides these two APIs to the SUT instead of the real DET and DEM. When the SUT invokes one of these two APIs together with the arguments describing the error or event, the target adapter forwards the API invocation to the test executable. The test executable is then able to log the error or diagnostic event and to react appropriately on it.

5.3 Additional Test Functionality

In addition to the basic functionality described in the previous sections, the target adapter is required to provide test functionality that involves direct interaction with the embedded hardware platform.
Each additional test function can throw an exception of type `TA_ErrorType` in case of an error. `TA_ErrorType` is defined as follows:

```
type integer TA_ErrorType;
group g_TA_ErrorType
{
  const TA_ErrorType c_TA_ERROR_GENERAL := 1;
  const TA_ErrorType c_TA_ERROR_OUT_OF_RAM_BLOCKS := 2;
  // To be extended by additional error types.
}
```

5.3.1 Interaction with Memory

As described in [3], the concept of transparently using pointers within the TTCN-3 test cases requires operations available for test cases to interact with the memory (RAM) that is used by the BSW module under test. These operations work for both Class A and Class B test setups and need to be implemented as target adapter functions.
RAM blocks to be used during testing need to be defined within the source code of the target adapter due to the static memory management concepts of embedded software. A CTSpec can define an arbitrary number of RAM blocks to be used during testing. Each RAM block is specified by the following information:

- **Address:** Start address of the RAM block located on the target platform.
- **Size:** The size of the RAM block in bytes.

The target adapter must provide sufficient RAM blocks as required by the CTSpec.

5.3.1.1 Obtaining a RAM Block Pointer

Since in general the addresses of the RAM blocks are only known after system integration and vary for different hardware platforms, they need to be provided to the test cases during run-time of the test cases.
During test case execution, a pointer to a RAM block of a certain minimum size can be requested in order to use it in subsequent calls towards the SUT (e.g. read or write operations).
To obtain the pointer for a RAM block, the following target adapter function is to be used:

```
signature TA_getNextRamBlockPtr(in uint16 minSize)
  return PtrType
  exception (TA_ErrorType);
```

This target adapter function returns a pointer to a RAM block located on the target platform with a minimum size as specified in the parameter `minSize`. Each invocation of `TA_getNextRamBlockPtr()` returns the next available pointer to a RAM

block of specified minimum size. If no RAM blocks are available anymore, the exception TA_ERROR_OUT_OF_RAM_BLOCKS is thrown.
In order to efficiently use the available RAM blocks, an implementation of this target adapter function shall return the address of an available RAM block that comes closest to the required minimum size.
The test case is responsible to store the returned pointer for later use since there is no way of re-requesting the address of an already requested RAM block.

5.3.1.2 Read from RAM/ROM

The target adapter must provide the functionality for reading a memory block of specified length at an arbitrary physical RAM or ROM address of the embedded system. The content of this memory block is sent to the test executable for further processing.
This functionality is provided to the test cases by the following signature definition:

```
signature TA_ReadMemBlock(in PtrType memPtr, in uint16 len)
return octetstring
exception (TA_ErrorType);
```

5.3.1.3 Write to RAM

The target adapter must provide the functionality to write a memory block of specified length to an arbitrary physical RAM address on the embedded system. The data to be written is provided by the test executable.
This functionality is provided to the test cases by the following signature definition:

```
signature TA_WriteMemBlock(in PtrType memPtr,
                           in octetstring memBlock)
exception (TA_ErrorType);
```

The number of bytes to be written is implicitly given with `memBlock` of type `octetstring`.

5.3.1.4 Compare Memory Blocks

The target adapter must provide the functionality to compare two memory blocks of specified length. If both memory blocks are located on the target system, the comparison is carried out on the target platform using the function `TA_MemBlocksEqual()`. In this case, no memory content is transferred to the test executable on the Tester PC.
If one memory block is located on the target system and the second one on the Tester PC (as data type `octetstring`), the content of the first memory block has to be transferred to the Tester PC. The comparison is then carried out on the Tester PC. The function `TA_CompareBlock()` is used here.

```
signature TA_MemBlocksEqual(in PtrType memPtr1,
                            in PtrType memPtr2,
                            in uint16 len)
return boolean
```

```
exception (TA_ErrorType);

  signature TA_CompareBlock(in PtrType memPtr
                            in octetstring data)
return boolean
exception (TA_ErrorType);
```

These functions returns TRUE if the relevant data in the two memory blocks (defined explicitly by the parameter `len` or implicitly by the length of the `octetstring`) have exactly the same content. Otherwise, this function returns FALSE.

5.3.2 Copy Memory Blocks

The target adapter must provide the functionality to copy the content of one memory block (located either in RAM or ROM) to another memory block (located in RAM). The copy operation is carried out on the target platform. Therefore, no memory content is transferred to the test executable on the Tester PC.

```
signature TA_CopyMemBlock(in PtrType SrcPtr, in PtrType DstPtr)
  exception (TA_ErrorType);
```

This functions returns when the copy operation has completed.

5.3.3 Power Control for Target Platform

For test cases that need to control powering of the target platform, two “TTCN-3 external” functions are specified.

```
signature EXT_SwitchOffTarget();
signature EXT_SwitchOnTarget();
```

`Ext_SwitchOffTarget()` powers down the target platform by switching off its power supply. `Ext_SwitchOnTarget()` powers up the target platform by switching on its power supply.
As default, the power for the target platform is supplied. This means, a test case makes use of these two functions only when it is required to switch off the power supply for test purposes.

5.4 Strategy for Functional Extensions

It is likely that BSW modules to be tested for conformance (especially those modules related directly to hardware) need additional test functionality at the target adapter.
Some possible additional functionalities are, for example:

- Direct interaction with hardware components (other than RAM/ROM)
- Real-time response to events at the SUT

The following guidelines should be followed when designing and implementing the functional extensions to the target adapter:

- When designing the functional extension, ensure that it does not affect the functionality already specified for and implemented in the target adapter. In other words, if a test case does not make use of the functional extension, it

must not make any difference whether the target adapter has the functional extension implemented or not.

- Define and implement the functional extension as a custom function with the name prefix TA_ (for target adapter functions) or EXT_ (for external functions) and appropriate parameters and/or return value (if required).
- Perform basic tests to verify whether the functional extension works correctly and does not affect the remaining target adapter functionality.