



ASAM

Association for Standardisation of
Automation and Measuring Systems

ASAM AE XIL

Generic Simulator Interface

Part 1 of 4

Programmers Guide

Version 2.1.0

Date: 2017-06-19

Base Standard

© by ASAM e.V., 2017

Disclaimer

This document is the copyrighted property of ASAM e.V.
Any use is limited to the scope described in the license terms. The license terms can be viewed at www.asam.net/license

Table of Contents

Foreword	9
1 Introduction	11
1.1 Overview	11
1.2 Motivation	11
1.3 What is Hardware in the Loop Simulation	12
1.4 Technical Approach	13
1.5 Technology Independence	13
2 Relations to Other Standards	15
2.1 Backward Compatibility to Earlier Releases	15
2.2 References to Other Standards	15
2.3 Versioning.....	15
3 General Concepts	16
3.1 XIL Test System Architecture.....	16
3.2 Overview Framework	18
3.2.1 General.....	18
3.2.2 Initialization	18
3.2.3 Framework Variables	18
3.2.4 Acquisition and Stimulation	19
3.3 Overview Testbench.....	22
3.3.1 General.....	22
3.3.2 Ports of the XIL Testbench API	22
3.4 ASAM Data Types.....	23
3.5 Instance Creation	23
3.5.1 Implementation Manifest File	24
3.5.1.1 File content.....	24
3.5.1.2 File Naming convention and storage location	28
3.5.2 Framework and Testbench Factory.....	29
4 Framework	30
4.1 Configuration.....	30
4.1.1 Framework Configuration File	30
4.1.2 Configuring the Framework	33
4.1.3 Shutting down the Framework.....	35
4.1.4 Customize Automatic port management of the framework	36
4.1.5 Object model of the framework configuration	38
4.2 Mapping	39
4.2.1 Overview.....	39
4.2.2 The Mapping XML Schema.....	40

4.2.2.1	Identifier Lists	42
4.2.2.2	Identifier Mapping	45
4.2.2.3	String Mapping	47
4.2.2.4	Raster Mapping	48
4.2.2.5	Conversion Tables	48
4.2.2.6	Units	49
4.2.2.7	Computation Tables	51
4.2.2.8	Versioning	52
4.2.3	How a Mapping is used by Test Cases	52
4.2.3.1	The Mapping Info API.....	52
4.2.3.2	Identifier Mapping	54
4.2.3.3	String Mapping	54
4.2.3.4	Raster Mapping	54
4.2.4	Tasks of the Framework.....	54
4.2.4.1	Reading mapping XML files	54
4.2.4.2	Consolidating mapping object trees	54
4.2.4.3	Implementing the Mapping Info API	55
4.2.4.4	Resolve the Mapping during Variable Creation.....	57
4.2.4.5	Perform automatic string mappings	57
4.2.4.6	Error handling.....	57
4.2.5	Consistency Rules	57
4.3	Framework Variables	59
4.3.1	What is a Framework Variable	59
4.3.2	Framework variable Classes	60
4.3.2.1	Scalar Variables	62
4.3.2.2	Vector Variables	63
4.3.2.3	Matrix Variables.....	64
4.3.2.4	Curve Variables.....	65
4.3.2.5	Map Variables	67
4.3.3	Quantities.....	69
4.3.3.1	Scalar Quantity Classes	70
4.3.3.2	Complex Quantity Data Classes	73
4.3.3.3	Usage of Quantities	75
4.3.4	Unit and Physical Dimensions	78
4.3.4.1	Unit Class	80
4.3.4.2	Physical Dimension Class	80
4.3.5	MetaData	80
4.3.6	Calculation with quantity objects	81
4.3.6.1	Overview.....	81
4.3.6.2	Addition and Subtraction	82
4.3.6.3	Multiplication and Division	82
4.3.6.4	Computation Table	83
4.3.6.5	Comparison	86
4.3.7	Data Type Conversion	86
4.3.8	Unit Conversion	86
4.3.9	Usage of Mathematical Operations	87
4.3.9.1	ABSOLUTE and RELATIVE.....	87
4.3.9.2	NEUTRAL UNIT	87
4.3.9.3	Mathematical and Physical Constants	88
4.3.9.4	Example.....	88
4.4	Measuring	89
4.4.1	Motivation	89
4.4.2	Setting up an Acquisition.....	90
4.4.2.1	Introduction.....	90
4.4.2.2	Using the Acquisition Interface.....	90
4.4.2.3	Acquisition States	92

4.4.2.4	Using the AcquisitionConfiguration Interface	92
4.4.2.5	Principle of Triggered Acquisition.....	94
4.4.2.6	Synchronized Data Acquisition	98
4.4.3	Setting up a Recording.....	101
4.4.3.1	Introduction.....	101
4.4.3.2	Configuring an AcquisitionRecorder.....	102
4.4.3.3	Configuring a Recorder	102
4.4.3.4	Using RecorderResult Readers and Writers	103
4.4.3.5	Working with the Recorded Results	105
4.4.3.6	Recorder States	107
4.4.4	Sequence of Acquisition and Recorder Commands	108
4.5	Stimulation.....	110
4.5.1	Motivation	110
4.5.2	Setting up a Stimulation	110
4.5.2.1	Introduction.....	110
4.5.2.2	Using the Stimulation Interface	111
4.5.2.3	Stimulation States	111
4.5.3	Setting up a Player.....	112
4.5.3.1	Introduction.....	112
4.5.3.2	Configuring a Player.....	113
4.5.3.3	Using a Player	115
4.5.3.4	Player States	116
5	Testbench	117
5.1	Common Functionalities.....	117
5.1.1	Valuecontainer	117
5.1.1.1	Overview.....	117
5.1.1.2	General Value Container Classes.....	118
5.1.1.3	Application Oriented Value Container Classes	119
5.1.1.4	Attributes	120
5.1.2	Document Handling.....	121
5.1.3	Script.....	122
5.1.3.1	Overview.....	122
5.1.3.2	States of Script	123
5.1.3.3	Script Parameters.....	126
5.1.4	TargetScript	126
5.1.4.1	Overview.....	127
5.1.4.2	Parameters.....	127
5.1.4.3	Custom Properties.....	128
5.1.4.4	Usage of TargetScript	128
5.1.5	Signal Descriptions	129
5.1.5.1	General Remarks about Segment-Based Signals	132
5.1.5.2	Signal Segments	135
5.1.5.3	Using Signal Descriptions	155
5.1.5.4	Signal Description File.....	161
5.1.5.5	Usage of Parameterized SignalDescriptions.....	162
5.1.6	SignalGenerator	164
5.1.6.1	Parameters.....	165
5.1.6.2	Custom Properties.....	165
5.1.6.3	Usage of SignalGenerator (Stimulating Model Variables)	165
5.1.7	Document Handling for SignalGenerator and SignalDescriptionSet ...	167
5.1.8	Watcher	170
5.1.8.1	General.....	170
5.1.8.2	Using the TimeOut	171
5.1.9	Duration	171
5.1.10	Meta Info.....	171

5.1.11	Data Capturing	172
5.1.11.1	Introduction	172
5.1.11.2	Capturing	172
5.1.11.3	Modes of Capturing	175
5.1.11.4	Untriggered Capturing	175
5.1.11.5	triggered Capturing	176
5.1.11.6	Re-triggered Capturing	178
5.1.11.7	Different ways to access the acquired data of a Capture object	179
5.1.11.8	Capture Result	180
5.1.11.9	Capture Results, trigger delays and capture events	181
5.1.11.10	Document Handling for Capture Data	190
5.1.11.11	Usage of Capturing	191
5.2	Model access Port	194
5.2.1	User Concept	194
5.2.1.1	General	194
5.2.1.2	Model Access Port class	194
5.2.1.3	States of the MAPort	195
5.2.2	Usage of this Port	197
5.2.2.1	Creation and Configuration	197
5.2.2.2	Reading & Writing Model Variables	199
5.2.2.3	Relation between MAPort and Capturing / SignalGenerator	202
5.2.2.4	Pausing and stepwise execution of the simulation	203
5.3	Diagnostic Port	205
5.3.1	Overview	205
5.3.2	API	206
5.3.2.1	Communication Modes	207
5.3.2.2	ECU	207
5.3.2.3	Functional Groups	208
5.3.3	States of the DiagPort	208
5.3.4	Usage of this Port	210
5.3.4.1	Creation and Configuration	210
5.3.4.2	Getting The Ecu Object	211
5.3.4.3	Reading And Clearing The Fault Memory	212
5.3.4.4	Reading The Variant Coding Data	213
5.3.4.5	Reading Identification Data	213
5.3.4.6	Reading and Writing Values from and to the Eeprom by Alias Names	214
5.3.4.7	Reading from the Eeprom	215
5.3.4.8	Writing to the Eeprom	215
5.3.4.9	Implicit and Explicit Communication	216
5.3.4.10	Sending Hex Services with Explicit Communication	216
5.3.4.11	Executing Jobs	217
5.3.4.12	Reading Data from a Functional Group	218
5.3.4.13	Using the Basecontroller	219
5.3.5	Special Hints	219
5.3.5.1	Structure of Returned Collections	219
5.3.5.2	States in the Diagnostic Tool	219
5.4	ECUMPort	220
5.4.1	Overview	220
5.4.2	States of the ECUMPort	221
5.4.3	Usage of ECUMPort	223
5.4.3.1	Creation and Configuration	223
5.4.3.2	Getting Lists of Variable and Task Names	224
5.4.3.3	Read a Scalar Variable Value and its Properties	225
5.4.3.4	Read an Array Variable Value and its Properties	225
5.4.3.5	Read a Matrix Variable Value and its Properties	226

5.4.3.6	Capturing ECU Variables	227
5.5	ECUCPort.....	229
5.5.1	Overview.....	229
5.5.2	States of the ECUCPort	230
5.5.3	Usage of ECUCPort.....	232
5.5.3.1	Creation and Configuration	232
5.5.3.2	Accessing ECU Parameters.....	233
5.5.3.3	Getting the List of Variables of the ECUCPort	234
5.5.3.4	Manage ECU Memory Pages	235
5.6	EES Port.....	237
5.6.1	User Concept	237
5.6.1.1	General.....	237
5.6.1.2	Configuration and Execution of Electrical Errors	239
5.6.1.3	Triggers in EES	241
5.6.1.4	Electrical Errors	241
5.6.1.5	API.....	244
5.6.1.6	States of the EES Port	250
5.6.2	Usage of this Port	253
5.6.2.1	Port Creation and Configuration.....	253
5.6.2.2	Creating Error Configurations	254
5.6.2.3	Error Stimulation.....	257
5.6.2.4	Extension of Error Stimulation.....	257
5.6.2.5	Creating Error Objects.....	259
5.6.2.6	Loading Error Configurations from File	263
5.6.3	Special Hints	264
5.6.3.1	EES Hardware Limitations and Extensions.....	264
5.7	Network Port.....	264
5.7.1	User Concept	264
5.7.1.1	General.....	264
5.7.1.2	Network Port.....	265
5.7.1.3	States of the Network Port	266
5.7.2	Usage of the Network Port	268
5.7.2.1	Port Creation and Configuration.....	268
5.7.2.2	Object Model	269
5.7.2.3	Mapping to DBC and FIBEX	271
5.7.2.4	Navigating the Object Model	272
5.7.2.5	Send and Receive Frames.....	273
5.7.2.6	Send and Receive Signals	276
5.7.2.7	Capturing of Bus Signals.....	278
5.7.2.8	Capturing of Bus Frames	281
5.7.2.9	Replay of Bus Frames.....	284
5.7.2.10	Extending the CAN object model	285
6	Symbols and Abbreviated Terms	288
7	Bibliography	289
Appendix A.	Syntax of Watcher Conditions	290
A.1.	Other restrictions	292
A.2.	Syntax Overview.....	292
Appendix B.	Syntax of ConstSymbol Expressions	295
B.1.	Other restrictions	297

B.2. Syntax Overview.....	297
Appendix C. Key Value Pairs in CaptureResult MetaData	299
Appendix D. Storage of Data in MDF4	300
D.1. Framework Measurement Data.....	300
D.1.1. Retriggered Measurements.....	301
D.2. Testbench Capture Data	302
D.2.1. Storage of Client Events	302
Appendix E. Error Overview	304
Appendix F. Deprecated Elements	343
Figure Directory	344
Table Directory	349

Foreword

ASAM developed HIL API as a standard for the communication between test automation software and hardware-in-the-loop (HIL) testbenches. HIL API enables users to choose products freely according to their requirements, independent of the vendor. Several implementations of the latest version of the standard, HIL API 1.0.2, are available on the market now. This new version 2.0 of the API contains broadly extended functionality and enhanced applicability. It will support testbenches at all stages of the function software development process – MIL¹, SIL², HIL³, etc. As a result, the ASAM standardization workgroup has decided to change the name to XIL API – Generic Simulator Interface with release version 2.0. After all, XIL API allows engineers to reuse their existing tests and enables a better know-how transfer from one testbench to the other, resulting in reduced training costs for employees as well.

Some areas of the XIL API standard are not HIL-specific. The MAPort, for example, can also be used to adapt simulation tools. This allows engineers to develop test cases in very early stages and in different domains in order to reuse them in later stages at a real HIL Simulator using XIL API. The Functional Mock-up Interfaces (FMI) initiative has cooperated with the XIL API Project 2.0. As a result of the ITEA2-funded project Modelisar, standardized interfaces for model exchange and co-simulation of subsystems from different domains have been developed. These “functional mock-up interfaces” will support simulation system setup at all stages of function software development (MIL, SIL, HIL, etc.).

Thus, a subset of XIL API 2.0 mainly dealing with the MAPort and simulator control as “Functional Mock-up Interface for Applications” has been released separately. This means that tests written in those early simulation environments can be directly reused in real HIL environments at a later stage.

The standard consists of different parts:

- one part for the base standard specification (programmers guide)
- one part for the mapping rules of each technology reference and
- one part for the XIL support library documentation

The base standard contains the major parts

- *Framework*, which is completely new and contains broadly extended functionality such as variable measuring and mapping as well as managing of ports already known from HIL API 1.0.2. The *Framework* chapter deals with functionality, that is based on the
- *Testbench*, which comprises the ports, known from HIL API 1.0.2. The ports were extended slightly with respect to missing functionality in the previous standard version, such as configuration and initialization. In order to give test developers standardized access to CAN busses, the Network Port was completely new designed and added to the port family.

¹ Model-in-the-Loop

² Software-in-the-Loop

³ Hardware-in-the-Loop

The XIL support library contains open source software, which can be used by test case developers and framework vendors to realize vendor independent common tasks. Such tasks are

- for framework and test case
 - Unit Converter
 - Data type converter
- for Framework
 - Mapping reader (which generates a memory image from mapping file information with access possibility)
 - Creation of Framework variables (ready for usage) based on mapping information's
- for Test case
 - Realization of Framework mapping Info API
 - Mathematical operations

Factories are distributed for the generic instantiation of the framework and one or more testbenches from different vendors.