



# ASAM

Association for Standardization of  
Automation and Measuring Systems

---

## **ASAM XIL**

Generic Simulator Interface

Part 1 of 2

### **Specification**

Version 3.0.0

Date: 2024-09-30

### **Base Standard**

---

© by ASAM e.V., 2024



*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](http://www.asam.net/license)

# Table of Contents

|   |    |
|---|----|
| Foreword .....  | 12 |
| Introduction .....  | 13 |
| Overview .....  | 13 |
| Conventions and notations .....                           | 14 |
| Modal verbs .....   | 14 |
| Normative and informative content .....                   | 15 |
| Language-neutral notation .....                           | 15 |
| Deliverables .....  | 16 |
| 1 Scope .....   | 19 |
| 2 Normative references .....                              | 20 |
| 3 Terms and definitions .....                             | 21 |
| 4 Abbreviations .....                                     | 22 |
| 5 Backward compatibility .....                            | 25 |
| 6 General concepts .....                                  | 27 |
| 6.1 Versioning .....                                      | 27 |
| 6.2 XIL test system architecture .....                    | 27 |
| 6.3 Framework overview .....                              | 30 |
| 6.3.1 General .....                                       | 30 |
| 6.3.2 Initialization .....                                | 30 |
| 6.3.3 Framework variables .....                           | 31 |
| 6.3.4 Acquisition and stimulation .....                   | 31 |
| 6.4 Testbench overview .....                              | 33 |
| 6.4.1 General .....                                       | 33 |
| 6.4.2 Ports of the XIL Testbench API .....                | 34 |
| 6.5 Instance creation .....                               | 35 |
| 6.5.1 Implementation Manifest file .....                  | 35 |
| 6.5.1.1 File content .....                                | 36 |
| 6.5.1.2 File naming convention and storage location ..... | 40 |
| 6.5.2 Framework and Testbench factories .....             | 40 |
| 6.6 Assembly registration .....                           | 42 |
| 7 Framework .....   | 43 |
| 7.1 Configuration .....                                   | 43 |
| 7.1.1 Framework configuration file .....                  | 43 |

---

|   |    |
|---|----|
| 7.1.2 Configuring the Framework .....                           | 47 |
| 7.1.3 Shutting down the Framework .....                         | 50 |
| 7.1.4 Customize automatic port management of the Framework..... | 51 |
| 7.1.5 Object model of the Framework configuration .....         | 53 |
| 7.2 Mapping .....   | 54 |
| 7.2.1 Overview .....  | 54 |
| 7.2.2 The Mapping XML Schema .....                              | 55 |
| 7.2.2.1 Identifier lists .....                                  | 56 |
| 7.2.2.2 Framework label group .....                             | 59 |
| 7.2.2.3 Identifier mapping .....                                | 60 |
| 7.2.2.4 String mapping .....                                    | 61 |
| 7.2.2.5 Raster mapping .....                                    | 62 |
| 7.2.2.6 Conversion tables .....                                 | 63 |
| 7.2.2.7 Units .....   | 63 |
| 7.2.2.8 Computation tables .....                                | 64 |
| 7.2.2.9 Versioning .....  | 65 |
| 7.2.3 How the Mapping is used by test cases .....               | 66 |
| 7.2.3.1 The Mapping Info API .....                              | 66 |
| 7.2.3.2 Identifier mapping .....                                | 68 |
| 7.2.3.3 String mapping .....                                    | 68 |
| 7.2.3.4 Raster mapping .....                                    | 68 |
| 7.2.4 Tasks of the Framework.....                               | 68 |
| 7.2.4.1 Reading mapping XML files.....                          | 68 |
| 7.2.4.2 Consolidating mapping object trees .....                | 68 |
| 7.2.4.3 Implementing the Mapping Info API .....                 | 69 |
| 7.2.4.4 Resolve the mapping during variable creation .....      | 71 |
| 7.2.4.5 Perform automatic string mappings.....                  | 71 |
| 7.2.4.6 Error handling .....                                    | 71 |
| 7.2.5 Consistency rules .....                                   | 71 |
| 7.3 Framework variables .....                                   | 73 |
| 7.3.1 What is a Framework variable .....                        | 73 |
| 7.3.2 Framework variable classes .....                          | 75 |
| 7.3.2.1 Scalar variables .....                                  | 78 |
| 7.3.2.2 Vector variables .....                                  | 79 |
| 7.3.2.3 Matrix variables .....                                  | 80 |

---

|  |     |
|--|-----|
| 7.3.2.4 Curve variables .....                              | 81  |
| 7.3.2.5 Map variables .....                                | 83  |
| 7.3.3 Quantities .....                                     | 87  |
| 7.3.3.1 Scalar quantity classes .....                      | 87  |
| 7.3.3.2 Complex quantity data classes .....                | 90  |
| 7.3.3.3 Usage of quantities .....                          | 92  |
| 7.3.4 Unit and physical dimensions .....                   | 96  |
| 7.3.4.1 Unit class .....                                   | 98  |
| 7.3.4.2 PhysicalDimension class .....                      | 98  |
| 7.3.5 MetaData .....                                       | 98  |
| 7.3.6 Calculation with quantity objects .....              | 99  |
| 7.3.6.1 Overview .....                                     | 99  |
| 7.3.6.2 Addition and subtraction .....                     | 100 |
| 7.3.6.3 Multiplication and division .....                  | 100 |
| 7.3.6.4 Computation table .....                            | 101 |
| 7.3.6.5 Comparison .....                                   | 104 |
| 7.3.7 Data type conversion .....                           | 105 |
| 7.3.8 Unit conversion .....                                | 105 |
| 7.3.9 Usage of mathematical operations .....               | 106 |
| 7.3.9.1 Absolute and relative unit conversion .....        | 106 |
| 7.3.9.2 Neutral unit .....                                 | 106 |
| 7.3.9.3 Mathematical and physical constants .....          | 107 |
| 7.3.9.4 Example .....                                      | 107 |
| 7.4 Measuring .....  | 108 |
| 7.4.1 Motivation .....                                     | 108 |
| 7.4.2 Setting up an Acquisition .....                      | 108 |
| 7.4.2.1 Introduction .....                                 | 108 |
| 7.4.2.2 Using the Acquisition interface .....              | 109 |
| 7.4.2.3 Acquisition states .....                           | 110 |
| 7.4.2.4 Using the AcquisitionConfiguration interface ..... | 110 |
| 7.4.2.5 Principle of triggered acquisition .....           | 110 |
| 7.4.2.6 Synchronized data acquisition .....                | 114 |
| 7.4.3 Setting up a recording .....                         | 117 |
| 7.4.3.1 Introduction .....                                 | 117 |
| 7.4.3.2 Configuring an AcquisitionRecorder .....           | 118 |

---

|         |   |     |
|---------|---|-----|
| 7.4.3.3 | Configuring a Recorder .....                        | 118 |
| 7.4.3.4 | Using RecorderResult readers and writers .....      | 119 |
| 7.4.3.5 | Working with the recorded results .....             | 121 |
| 7.4.3.6 | Recorder states .....                               | 123 |
| 7.4.4   | Sequence of Acquisition and Recorder commands ..... | 123 |
| 7.5     | Stimulation .....                                   | 124 |
| 7.5.1   | Motivation .....                                    | 124 |
| 7.5.2   | Setting up a Stimulation .....                      | 125 |
| 7.5.2.1 | Introduction .....                                  | 125 |
| 7.5.2.2 | Using the Stimulation interface .....               | 125 |
| 7.5.2.3 | Stimulation states .....                            | 126 |
| 7.5.3   | Setting up a Player .....                           | 126 |
| 7.5.3.1 | Introduction .....                                  | 126 |
| 7.5.3.2 | Configuring a player .....                          | 127 |
| 7.5.3.3 | Using a player .....                                | 128 |
| 7.5.3.4 | Player states .....                                 | 129 |
| 8       | Testbench .....                                     | 131 |
| 8.1     | Common functionalities .....                        | 131 |
| 8.1.1   | ValueContainer .....                                | 132 |
| 8.1.1.1 | Overview .....                                      | 132 |
| 8.1.1.2 | General ValueContainer classes .....                | 133 |
| 8.1.1.3 | Specific ValueContainer classes .....               | 135 |
| 8.1.2   | Document handling .....                             | 136 |
| 8.1.3   | Script .....  | 137 |
| 8.1.3.1 | Overview .....                                      | 137 |
| 8.1.3.2 | States of Script .....                              | 138 |
| 8.1.3.3 | Script parameters .....                             | 139 |
| 8.1.4   | TargetScript .....                                  | 140 |
| 8.1.4.1 | Overview .....                                      | 140 |
| 8.1.4.2 | Parameters .....                                    | 141 |
| 8.1.4.3 | Custom properties .....                             | 141 |
| 8.1.4.4 | Usage of TargetScript .....                         | 142 |
| 8.1.5   | Signal descriptions .....                           | 143 |
| 8.1.5.1 | General remarks about segment-based signals .....   | 145 |
| 8.1.5.2 | Signal segments .....                               | 148 |

---

|           |   |     |
|-----------|---|-----|
| 8.1.5.3   | Converting SignalDescriptions to sample based representations             | 157 |
| 8.1.5.4   | Using SignalDescriptions  | 158 |
| 8.1.5.5   | Signal description file   | 163 |
| 8.1.5.6   | Usage of parameterized SignalDescriptions                                 | 164 |
| 8.1.6     | SignalGenerator   | 168 |
| 8.1.6.1   | Interpretation of SignalDescriptions by the SignalGenerator               | 168 |
| 8.1.6.2   | Parameters  | 173 |
| 8.1.6.3   | Custom properties   | 173 |
| 8.1.6.4   | Usage of SignalGenerator (Stimulating model variables)                    | 173 |
| 8.1.7     | Document handling for SignalGenerator and SignalDescriptionSet            | 176 |
| 8.1.8     | Watcher   | 177 |
| 8.1.8.1   | General   | 177 |
| 8.1.9     | Duration  | 179 |
| 8.1.10    | MetalInfo package   | 180 |
| 8.1.10.1  | Metadata on variables   | 180 |
| 8.1.10.2  | Metadata on conversion methods  | 182 |
| 8.1.10.3  | Metadata on acquisition rasters / processor tasks                         | 184 |
| 8.1.11    | VariableRef and value representation mode                                 | 185 |
| 8.1.12    | Data capturing  | 186 |
| 8.1.12.1  | Introduction  | 186 |
| 8.1.12.2  | Capture configuration and control   | 187 |
| 8.1.12.3  | Untriggered capturing   | 188 |
| 8.1.12.4  | Triggered capturing   | 189 |
| 8.1.12.5  | Re-triggered capturing  | 191 |
| 8.1.12.6  | Obtain CaptureResult from a Capture                                       | 193 |
| 8.1.12.7  | CaptureResult   | 195 |
| 8.1.12.8  | Relation between triggers, capture states, capture events and data frames | 198 |
| 8.1.12.9  | Reader and writer for CaptureResult                                       | 207 |
| 8.1.12.10 | Usage examples for Capture and CaptureResult                              | 208 |
| 8.1.13    | Monitoring  | 216 |
| 8.1.13.1  | Concept   | 216 |
| 8.1.13.2  | Monitor types   | 220 |
| 8.2       | Model Access Port   | 222 |
| 8.2.1     | User concept  | 222 |
| 8.2.1.1   | General   | 222 |

---

---

|          |  |     |
|----------|--|-----|
| 8.2.1.2  | MAPort interface .....   | 222 |
| 8.2.1.3  | States of the MAPort .....   | 223 |
| 8.2.2    | Usage of MAPort .....  | 224 |
| 8.2.2.1  | Creation and configuration .....   | 225 |
| 8.2.2.2  | Obtaining information on available model variables, tasks and their properties ..... | 226 |
| 8.2.2.3  | Reading and writing model variables .....  | 229 |
| 8.2.2.4  | State dependency of writability and readability of variables .....                   | 233 |
| 8.2.2.5  | Relation between MAPort and Capturing / SignalGenerator .....                        | 234 |
| 8.2.2.6  | Pausing and stepwise execution of the simulation .....                               | 235 |
| 8.2.2.7  | EventMonitor .....   | 239 |
| 8.3      | Diagnostic Port .....  | 239 |
| 8.3.1    | Overview .....   | 239 |
| 8.3.2    | API .....  | 240 |
| 8.3.2.1  | Communication modes .....  | 241 |
| 8.3.2.2  | ECU .....  | 241 |
| 8.3.2.3  | Functional groups .....  | 242 |
| 8.3.3    | States of the DiagPort .....   | 243 |
| 8.3.4    | Usage of DiagPort .....  | 243 |
| 8.3.4.1  | Creation and configuration .....   | 244 |
| 8.3.4.2  | Getting the ECU object .....   | 244 |
| 8.3.4.3  | Reading and clearing the fault memory .....  | 245 |
| 8.3.4.4  | Reading the variant coding data .....  | 246 |
| 8.3.4.5  | Reading identification data .....  | 246 |
| 8.3.4.6  | Reading and writing values from and to the EEPROM by alias names .....               | 247 |
| 8.3.4.7  | Reading from the EEPROM .....  | 248 |
| 8.3.4.8  | Writing to the EEPROM .....  | 248 |
| 8.3.4.9  | Implicit and explicit communication .....  | 249 |
| 8.3.4.10 | Sending HEX services with explicit communication .....                               | 249 |
| 8.3.4.11 | Executing jobs .....   | 250 |
| 8.3.4.12 | Reading data from a functional group .....   | 250 |
| 8.3.4.13 | Using the BaseController .....   | 251 |
| 8.3.5    | Special hints .....  | 252 |
| 8.3.5.1  | Structure of returned collections .....  | 252 |
| 8.3.5.2  | States in the diagnostic tool .....  | 252 |
| 8.4      | ECUPort .....  | 252 |

---



---

|  |     |
|--|-----|
| 8.4.1 User concept .....   | 252 |
| 8.4.1.1 General .....  | 252 |
| 8.4.1.2 ECUPort interface .....  | 253 |
| 8.4.2 Usage of ECUPort .....   | 255 |
| 8.4.2.1 Creation and configuration .....   | 255 |
| 8.4.2.2 Accessing meta information of the variables of an ECU .....                | 256 |
| 8.4.2.3 Accessing meta information of the calibration memory pages of an ECU ..... | 257 |
| 8.4.2.4 Calibration functionality .....  | 258 |
| 8.4.2.5 Manage ECU memory pages .....  | 260 |
| 8.4.2.6 Capturing and recording of ECU variables .....                             | 261 |
| 8.5 EESPort .....  | 263 |
| 8.5.1 User concept .....   | 263 |
| 8.5.1.1 General .....  | 263 |
| 8.5.1.2 Configuration and execution of electrical errors .....                     | 265 |
| 8.5.1.3 Triggers in EES .....  | 268 |
| 8.5.1.4 Electrical errors .....  | 269 |
| 8.5.1.5 API .....  | 272 |
| 8.5.1.6 States of the EESPort .....  | 277 |
| 8.5.2 Usage of EESPort .....   | 279 |
| 8.5.2.1 Creation and configuration .....   | 279 |
| 8.5.2.2 Creating error configurations .....  | 280 |
| 8.5.2.3 Error stimulation .....  | 284 |
| 8.5.2.4 Extension of error stimulation .....                                       | 284 |
| 8.5.2.5 Creating error objects .....   | 285 |
| 8.5.2.6 Loading error configurations from file .....                               | 290 |
| 8.5.3 Special hints .....  | 291 |
| 8.5.3.1 EES hardware limitations and extensions .....                              | 291 |
| 8.6 NetworkPort .....  | 291 |
| 8.6.1 User concept .....   | 291 |
| 8.6.1.1 General .....  | 291 |
| 8.6.1.2 NetworkPort .....  | 292 |
| 8.6.1.3 States of the NetworkPort .....  | 293 |
| 8.6.2 Usage of the NetworkPort .....   | 294 |
| 8.6.2.1 Creation and configuration .....   | 294 |
| 8.6.2.2 Object model .....   | 295 |

---

---

|   |     |
|---|-----|
| 8.6.2.3 Mapping to DBC and FIBEX .....  | 298 |
| 8.6.2.4 Navigating the object model .....   | 299 |
| 8.6.2.5 Send and receive frames .....   | 300 |
| 8.6.2.6 Send and receive signals .....  | 304 |
| 8.6.2.7 Capturing of bus signals .....  | 306 |
| 8.6.2.8 Capturing of bus frames .....   | 309 |
| 8.6.2.9 Replay of bus frames .....  | 312 |
| 8.6.2.10 Extending the CAN object model .....   | 313 |
| 8.7 SOCPort .....   | 316 |
| 8.7.1 General .....   | 316 |
| 8.7.2 SOCPort interface .....   | 316 |
| 8.7.3 SOCPort states .....  | 318 |
| 8.7.4 SOC Message Classes .....   | 319 |
| 8.7.5 Usage of SOCPort .....  | 320 |
| 8.7.5.1 Calling service methods as a client .....                                     | 320 |
| 8.7.5.2 Listening for events .....  | 325 |
| 8.7.5.3 Working with fields .....   | 331 |
| 8.7.5.4 Controlling simulation of the services .....                                  | 337 |
| 8.7.5.5 Capturing SOC communication .....   | 340 |
| 8.7.5.6 Monitoring of SOC messages .....  | 350 |
| 8.8 Global functionalities .....  | 355 |
| 8.8.1 Custom methods .....  | 355 |
| 8.8.2 Retrieval of log file locations .....   | 355 |
| Annex A (normative): Syntax of Watcher conditions .....                               | 356 |
| A.1 Other restrictions .....  | 359 |
| A.2 Syntax overview .....   | 359 |
| Annex B (normative): Syntax of ConstSymbol expressions .....                          | 362 |
| B.1 Other restrictions .....  | 365 |
| B.2 Syntax overview .....   | 365 |
| Annex C (normative): Syntax of references to AUTOSAR SOME/IP payload data items ..... | 367 |
| C.1 Definition of AUTOSAR SOME/IP payload reference syntax .....                      | 367 |
| C.2 Examples of ASAM SOC payload references .....                                     | 370 |
| Annex D (informative): Storage of data in MDF .....                                   | 371 |
| D.1 Framework measurement data .....  | 371 |
| D.1.1 Storage of identification data for the signals .....                            | 371 |

---

|   |     |
|---|-----|
| D.1.2 Storage of physical units for the signals .....                       | 372 |
| D.1.3 Retriggered measurements .....  | 374 |
| D.2 Testbench capture data .....  | 375 |
| D.2.1 Storage of client events .....  | 375 |
| Annex E (informative): Technology independence .....                        | 378 |
| E.1 General .....   | 378 |
| E.2 C# Coding and naming conventions .....                                  | 378 |
| E.2.1 Names for interfaces and classes .....                                | 378 |
| E.2.2 Properties .....  | 378 |
| E.2.3 Method signatures .....   | 379 |
| E.2.3.1 Return types .....  | 379 |
| E.2.3.2 Parameter types .....   | 379 |
| E.2.4 Enumerations .....  | 379 |
| E.3 Mapping of types .....  | 379 |
| E.3.1 ASAM data types relation .....  | 379 |
| E.3.2 Array relation .....  | 380 |
| E.3.3 Collection relation .....   | 380 |
| E.3.4 XIL Enumerators .....   | 381 |
| E.3.5 XIL data types .....  | 381 |
| E.4 Disposable .....  | 381 |
| E.5 Exception handling .....  | 381 |
| E.6 Linux support .....   | 381 |
| Annex F (informative): Breaking changes to the previous major release ..... | 382 |
| Bibliography .....  | 385 |
| List of figures .....   | 386 |
| List of tables .....  | 396 |

## Foreword

ASAM e.V. (Association for Standardization of Automation and Measuring Systems) is a non-profit organization that promotes standardization of tool chains in automotive development and testing. Our members are international car manufacturers, suppliers, tool vendors, engineering service providers, and research institutes. ASAM standards are developed by experts from our member companies and are based on real use cases. ASAM is the legal owner of these standards and is responsible for their distribution and marketing.

ASAM standards span a wide range of use cases in automotive development, test, and validation. They define file formats, data models, protocols, and interfaces. The standards enable easy exchange of data and tools within and across tool chains. They are applied worldwide.

ASAM XIL standardizes the communication between test automation software and X-in-the-Loop testbenches. This also applies to Open-loop use cases.

X-in-the-Loop stands for Model-in-the-Loop , Software-in-the-Loop , Hardware-in-the-Loop and others. The different testbenches cover different phases in development:

- MIL testbenches are used to test functional models.
- SIL testbenches are used to test the target software.
- HIL testbenches are used to test the target software in conjunction with the ECU hardware.

The application of the ASAM XIL standard allows users to combine test automation software and virtual or physical testbenches independent of the vendor, and also to enable the reuse of the existing programmed tests for subsequent development stages (for example, reuse of test cases for function models in software tests at a HIL simulator).

Overall, the use of the standard achieves independence from vendor software and hardware and a very high reusability of test cases. This results also in reduced costs for test programming and reduced training costs for testers.

Parts of the standard:

- Specification  
The Specification includes a conceptual description of the Testbench and Framework API.
- C# To Python Mapping Rules  
The mapping rules define the transformation rules of the technology dependent C# definitions into a Python specific representations.