ASAM | Association for standardisation of automation and measuring systems

Release Presentation

ASAM AE XIL 2.0.1

Generic Simulator Interface

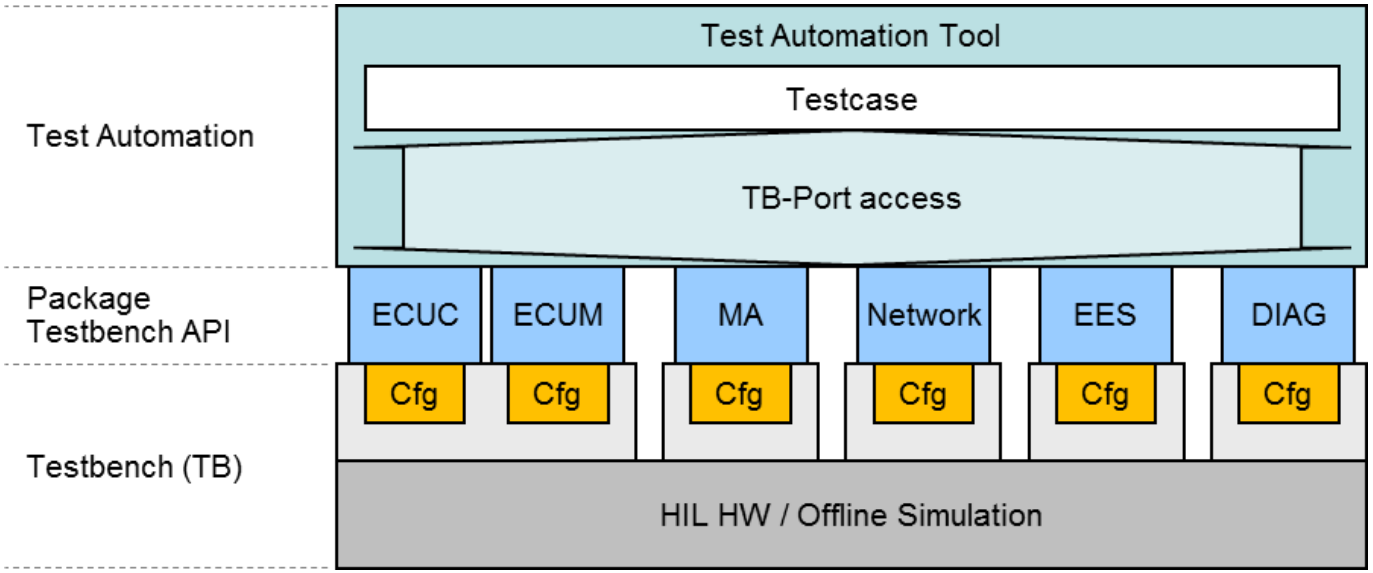
2014 / 09 / 30

Agenda

- Introduction and General Concepts
- What's New?
- Deliverables
- Compatibility
- Changes in Maintenance XIL 2.0.1

General Concepts (1)

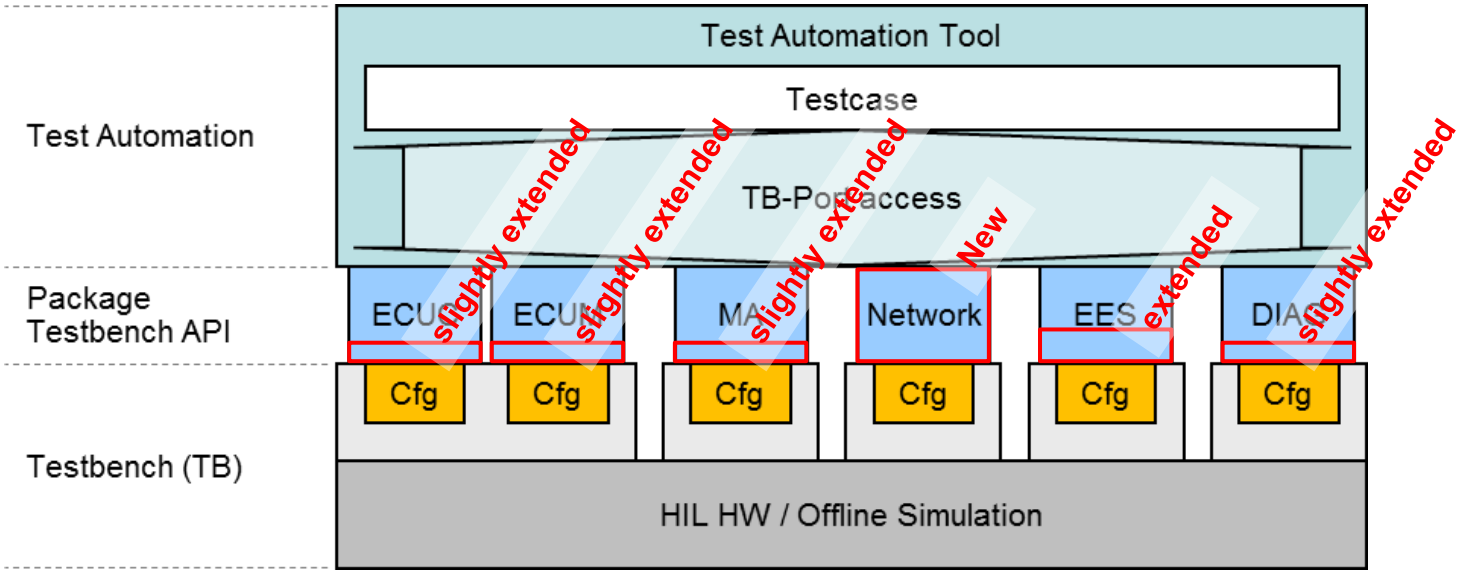
Testbench-based Access (as in HIL 1.0.2)



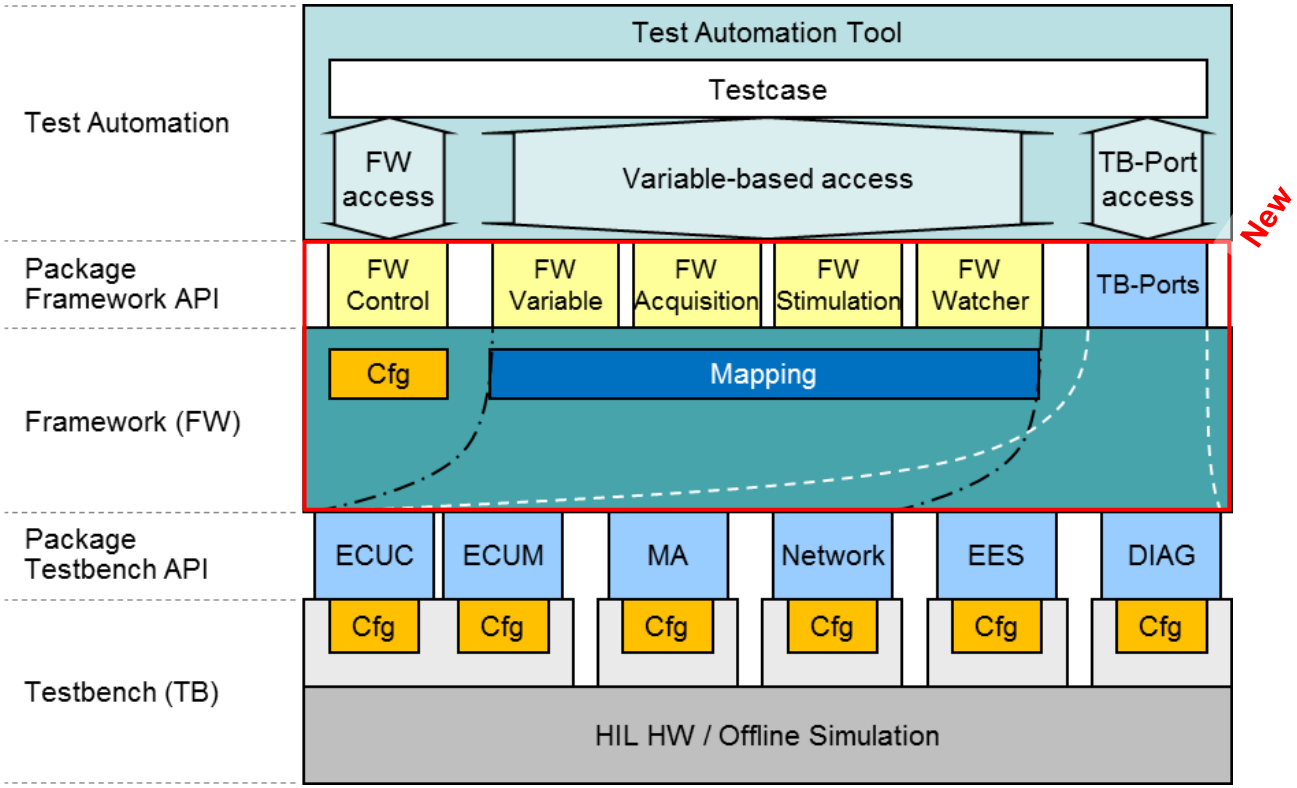
- Drawbacks of HIL 1.0.2:
  - Testcase has to implement start and shutdown of ports
  - Dealing with vendor-specific start and shutdown methods (not standardizedin 1.0.2)
  - Port-specific variable identifiers and data types
  - Missing overall concepts for measuring and stimulating accross different ports

General Concepts (2)

Testbench Extension



Framework-based Access (with XIL 2.0.0)



- Major Benefits:
  - Port independence of testcases by using an object-oriented access to variables
  - Framework starts and shuts down ports in a configured order
  - Test Developer can use both: Testbench Port access and Variable-based access
  - FW Variables provide access to the underlying Testbench Port

What's New?

- Maintenance Issues (motivated by market and crosstests)
- Extensions on existing ports (motivated by market)
- Extensions on existing ports (motivated by XIL 2.0.0 Framework)
- Network Port (new)
- New Features of XIL 2.0.0 Framework
  - Framework Variables
  - Mapping
  - Measuring
  - Stimulation
  - Configuration
- Software
  - XIL Support Library
  - Example Framework
  - Prototype and Test environment

What's New?

Testbench

- Maintenance Issues (motivated by market and crosstests)
- Extensions on existing ports (motivated by market)
- Extensions on existing ports (motivated by XIL 2.0.0 Framework)
- Network Port (new)

Framework

- New Features of XIL 2.0.0 Framework
  - Framework Variables
  - Mapping
  - Measuring
  - Stimulation
  - Configuration

Software Support

- Software
  - XIL Support Library
  - Example Framework
  - Prototype and Test environment

What's New?

- Maintenance Issues (motivated by market and crosstests)
- Extensions on existing ports (motivated by market)
- Extensions on existing ports (motivated by XIL 2.0.0 Framework)
- Network Port (new)
- New Features of XIL 2.0.0 Framework
  - Framework Variables
  - Mapping
  - Measuring
  - Stimulation
  - Configuration
- Software
  - XIL Support Library
  - Example Framework
  - Prototype and Test environment

Maintenance Topics within XIL 2.0.0

based on HIL 1.0.2

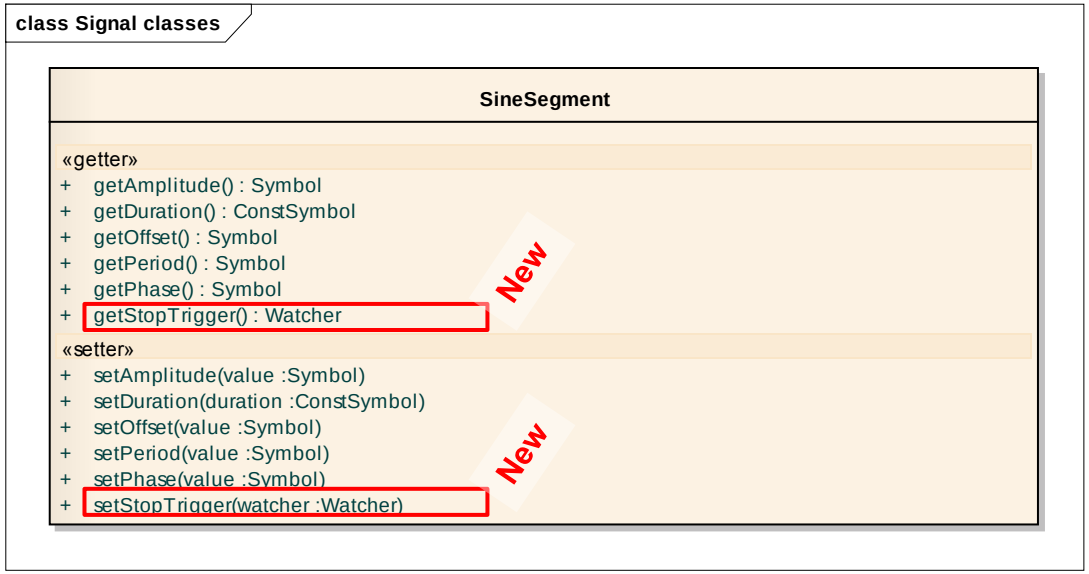
- Small differences between UML model and interfaces have been removed
- Capturing and Watcher of testbench have been documented more precisely (due to experiences gathered in cross tests and market)
  E. g. Task list names for capturing and some data types of simulation model variables are vendor specific
- Detailed definition of triggering (pre-trigger, timeouts of watchers)
- Segment Definitions of Signal Description Sets do not support Frequency modulation
- Providing some more detailed functionality, such as GetVariableNames on capture result objects, ZIP archives for signal descriptions (.sdz) in addition to pure XML and MAT (.sti and .mat)

What's New?

- Maintenance Issues (motivated by market and crosstests)
- Extensions on existing ports (motivated by market)
- Extensions on existing ports (motivated by XIL 2.0.0 Framework)
- Network Port (new)
- New Features of XIL 2.0.0 Framework
  - Framework Variables
  - Mapping
  - Measuring
  - Stimulation
  - Configuration

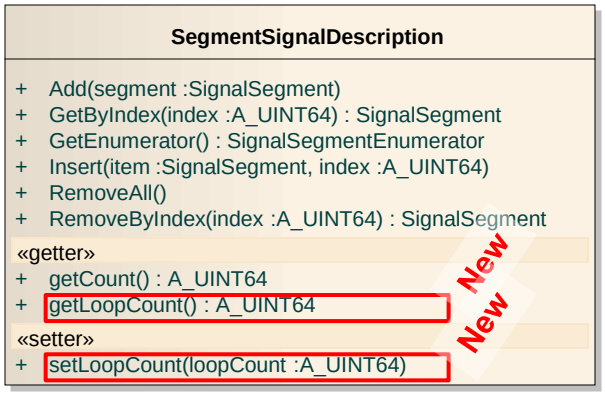
- Software
  - XIL Support Library
  - Example Framework
  - Prototype and Test environment

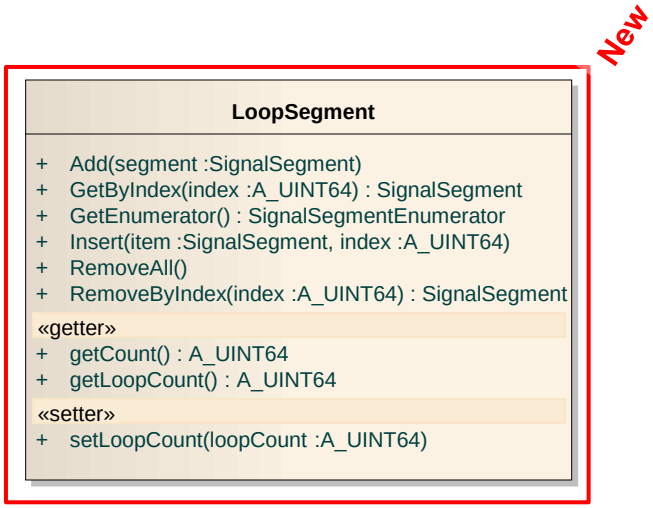
Stop Condition on Signal Segments



- Using existing Watcher objects to define reactive stop conditions

Loops within Signal Descriptions



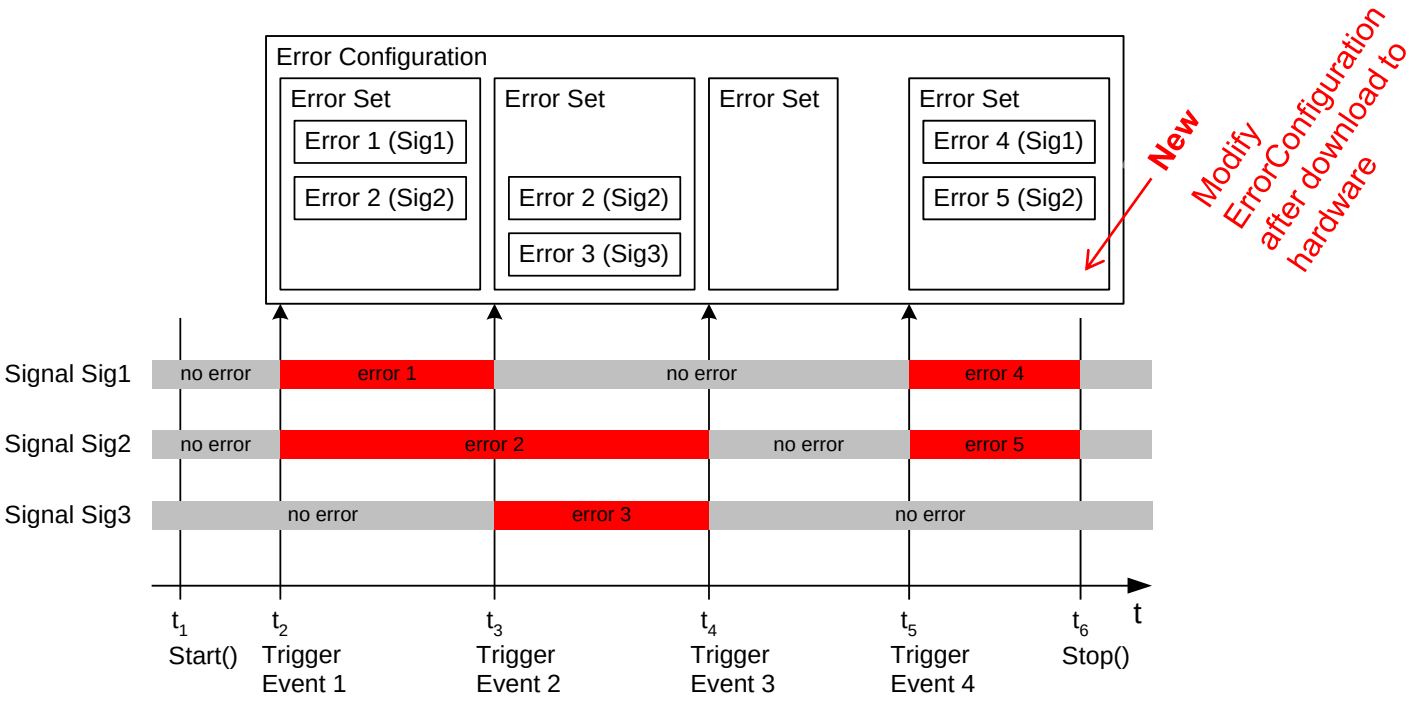


- Loop on Signals to repeat the entire signal
- New LoopSegment as a container to repeat an added sequence of segments

New Data File Segment

- For data replay based on MDF files with
  with options for selection of channel, TimeVector etc.

EESPort: Dynamic ErrorConfiguration



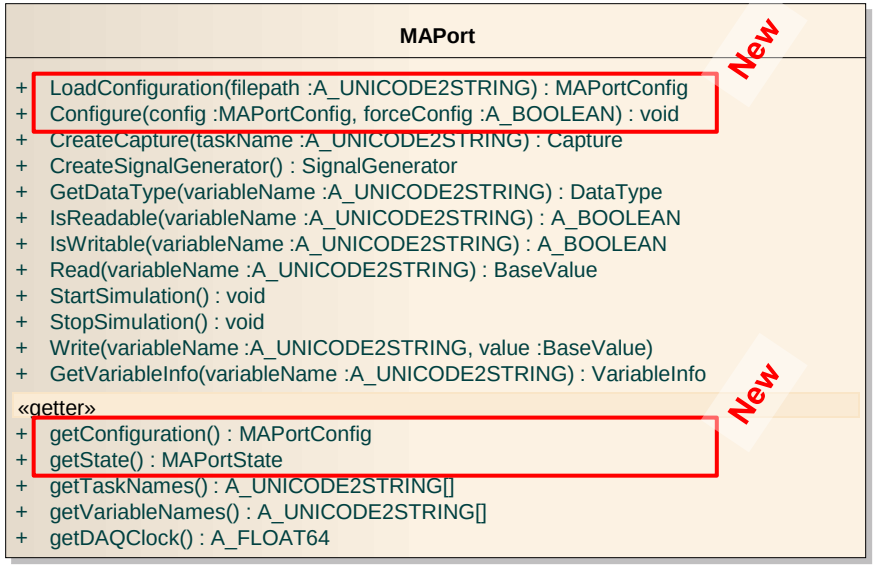
- New ErrorSets can be dynamically added to the ErrorConfiguration after it is downloaded to the hardware

What's New?

- Maintenance Issues (motivated by market and crosstests)
- Extensions on existing ports (motivated by market)
- Extensions on existing ports (motivated by XIL 2.0.0 Framework)
- Network Port (new)
- New Features of XIL 2.0.0 Framework
  - Framework Variables
  - Mapping
  - Measuring
  - Stimulation
  - Configuration

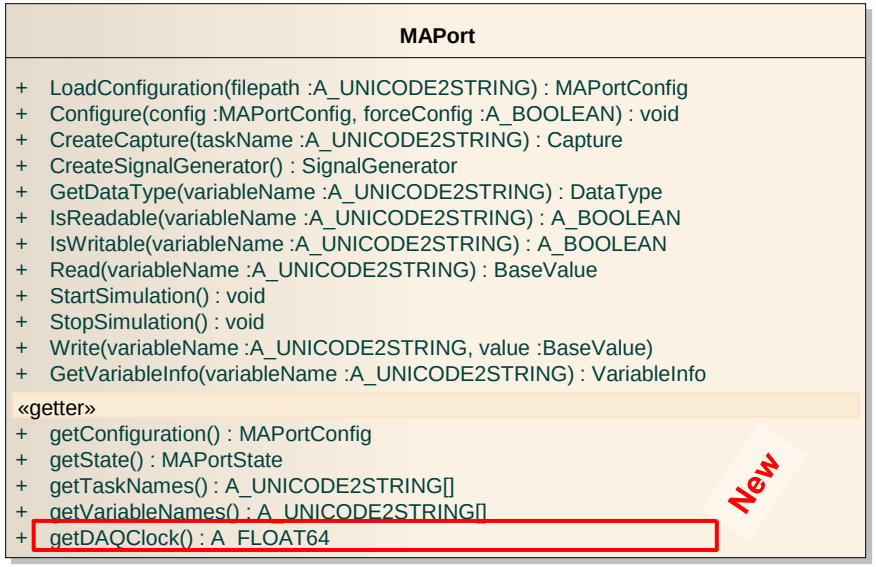
- Software
  - XIL Support Library
  - Example Framework
  - Prototype and Test environment

Port Configuration and Port States



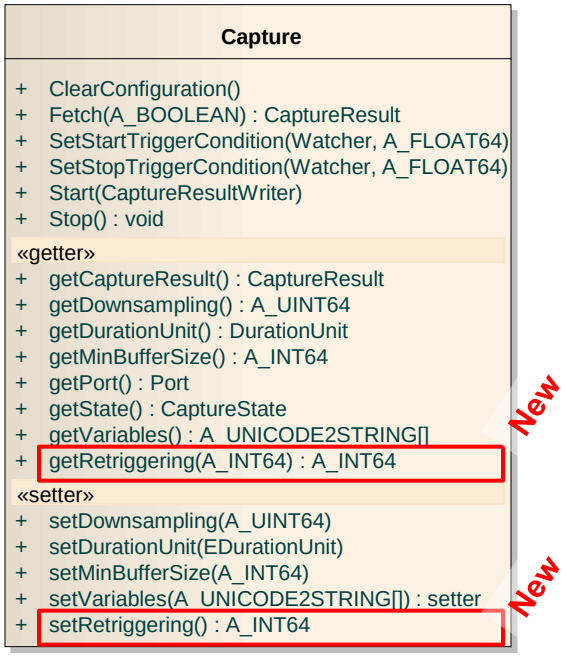
- All ports have configuration methods in order to set proper pre-conditions for testing
- All ports have access methods to get the port‘s current state

Access to hardware timers



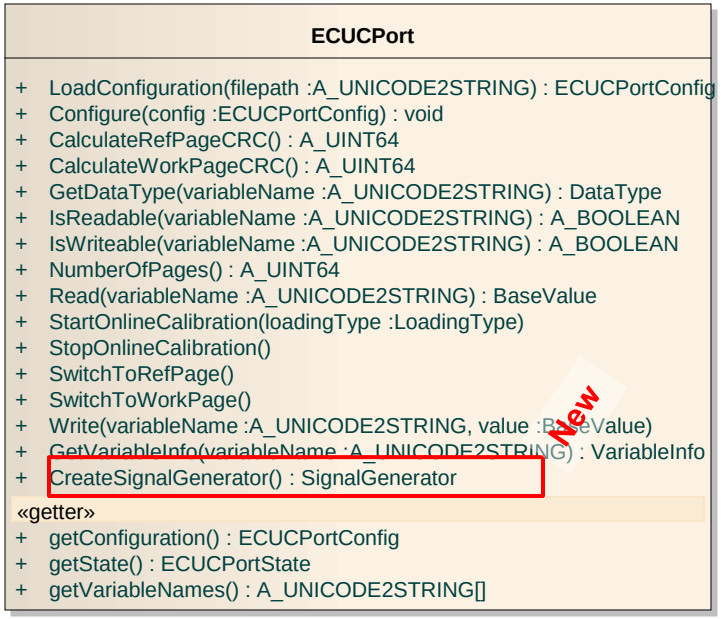
- All ports that allow data capturing (MAPort, ECUMPort, NetworkPort) have access to the hardware timers,
  i. e. for synchronised data acquisition

Capturing with retriggering



- Allow repeated start and stop triggers

More ports with SignalGenerator



- Allow SignalGeneration on ECU- and NetworkPort

What's New?

- Maintenance Issues (motivated by market and crosstests)
- Extensions on existing ports (motivated by market)
- Extensions on existing ports (motivated by XIL 2.0.0 Framework)
- Network Port (new)
- New Features of XIL 2.0.0 Framework
  - Framework Variables
  - Mapping
  - Measuring
  - Stimulation
  - Configuration

- Software
  - XIL Support Library
  - Example Framework
  - Prototype and Test environment

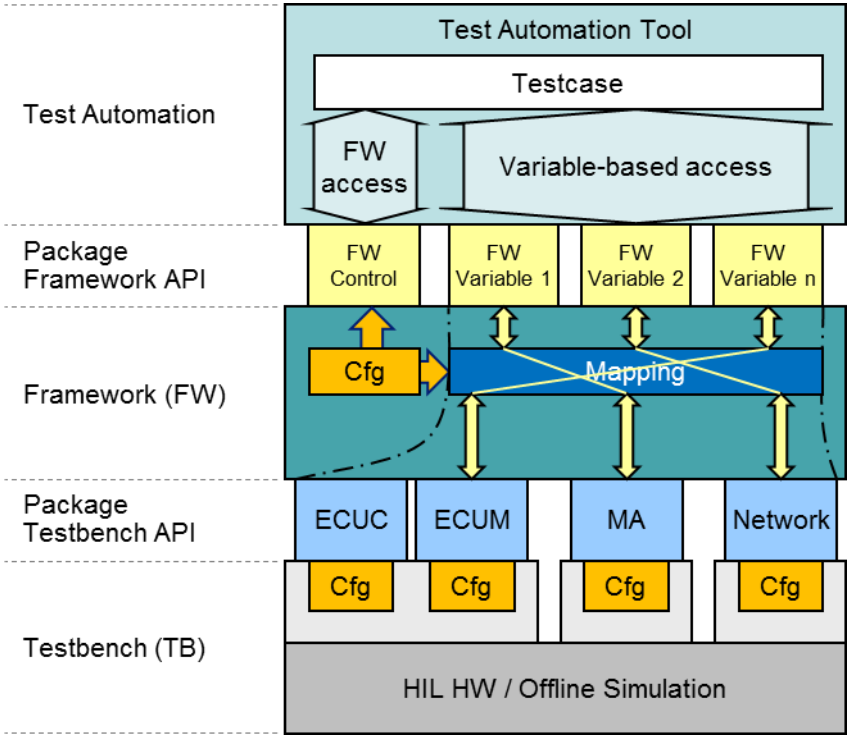
Network Access port

- The Network Access port provides access to field bus systems such as CAN (XIL API 2.0.0), LIN or FlexRay.
- The Port is designed mainly to operate in parallel to a HIL system, ECU or any other system.
- It allows measurement (monitoring) and transmission (single transmit or replay) of bus data.
- The mostly bus system independent object model will be created automatically based on the communication matrix information.
- The Network Port organizes its object classes in a tree-like hierarchy, by defining Cluster-, Channel-, Frame- ,PDU- and Signal- objects.
- Intentionally not covered by the Port: Bus-stress (too specific for a standard) or electrical fault insertion (covered by EES Port)

What's New?

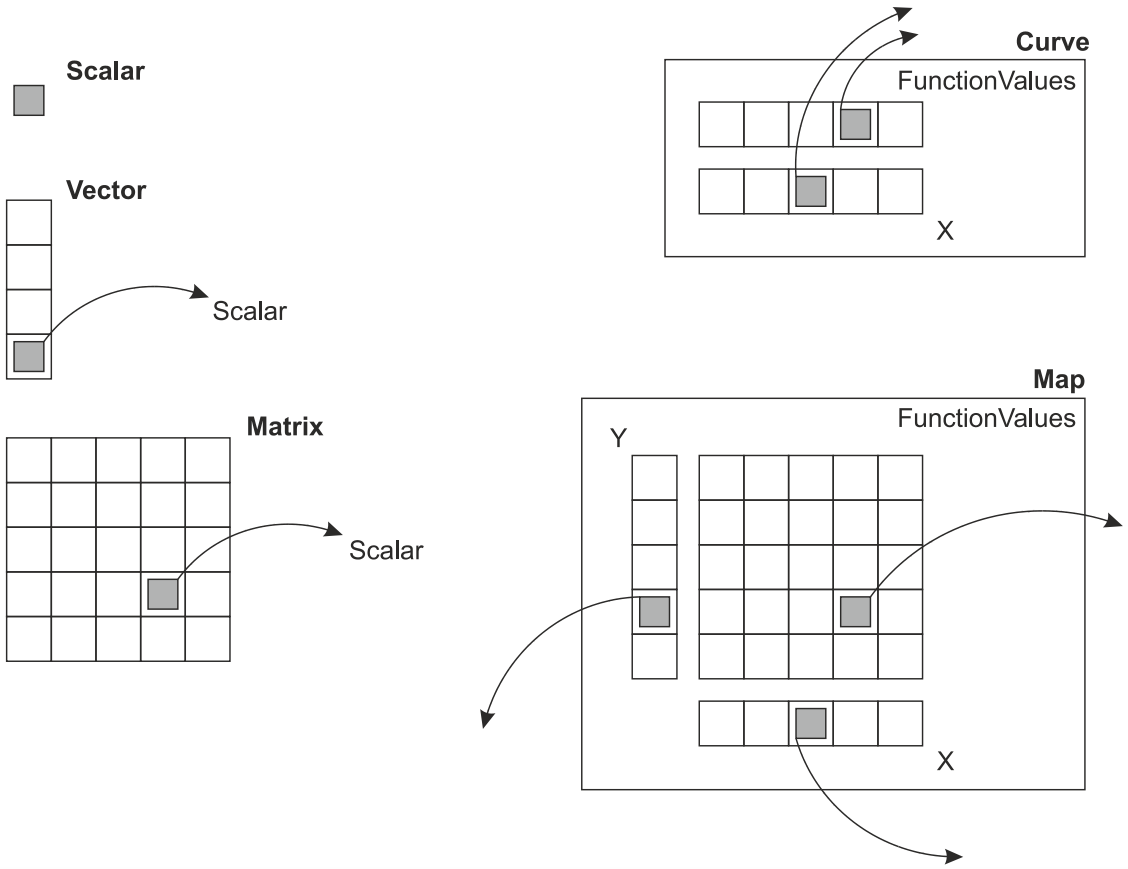
- Maintenance Issues (motivated by market and crosstests)
- Extensions on existing ports (motivated by market)
- Extensions on existing ports (motivated by XIL 2.0.0 Framework)
- Network Port (new)
- New Features of XIL 2.0.0 Framework
  - Framework Variables
  - Mapping
  - Measuring
  - Stimulation
  - Configuration
- Software
  - XIL Support Library
  - Example Framework
  - Prototype and Test environment

Framework Variable

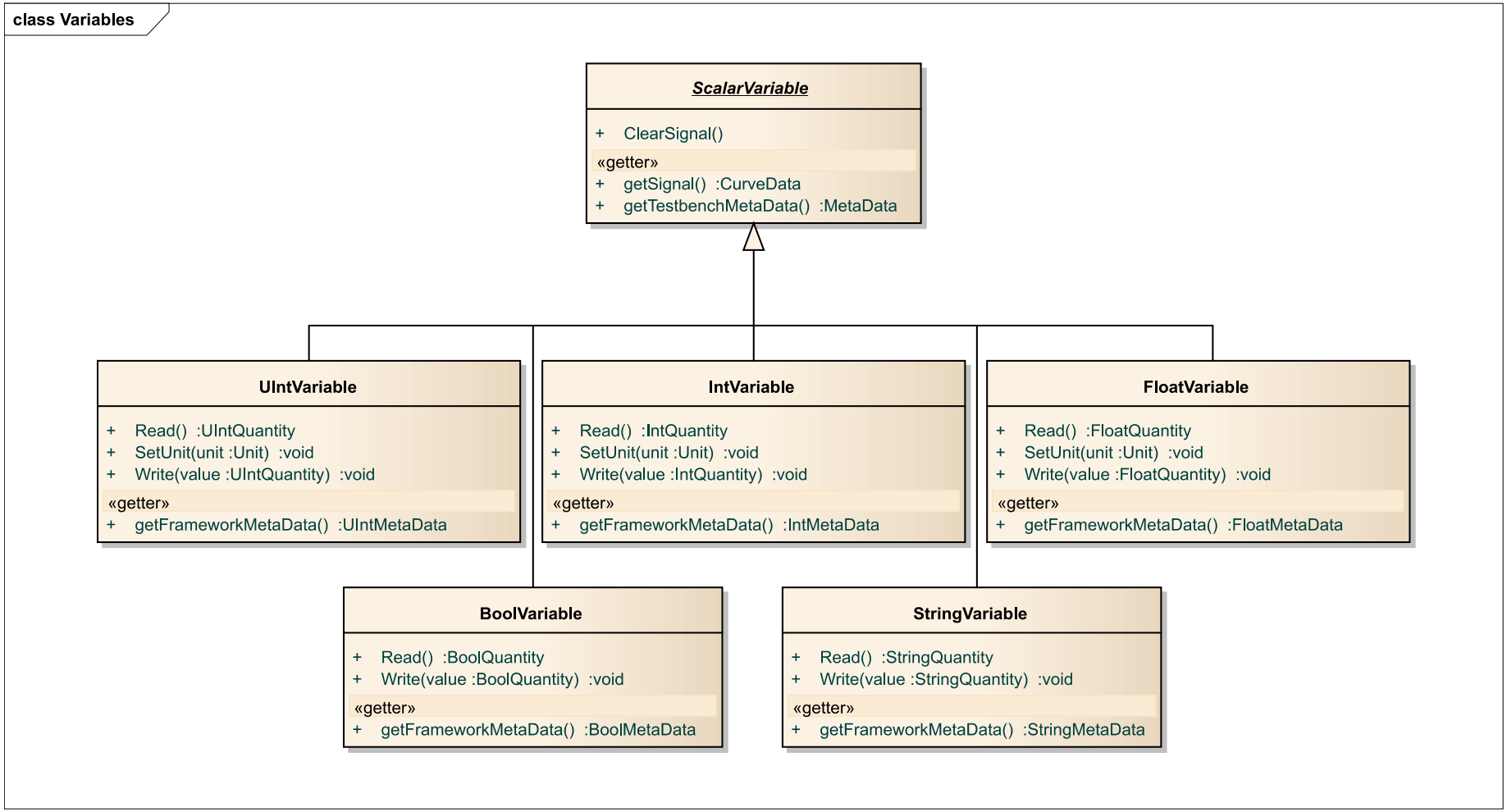


- Abstract Identifiers on test case side (e. g. “n_mot”).
- Object oriented access
- Guarantees that test cases are independent of the underlying test system
- **Thus, independent of vendor and process stage ➔ XIL**

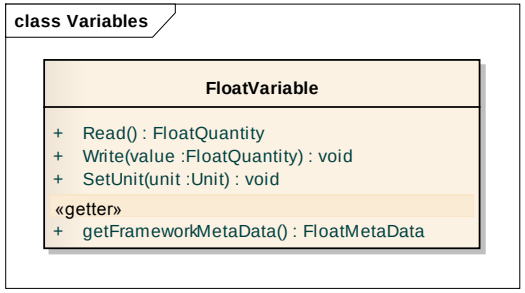
Framework Variable Types



Framework Variable Scalar Types

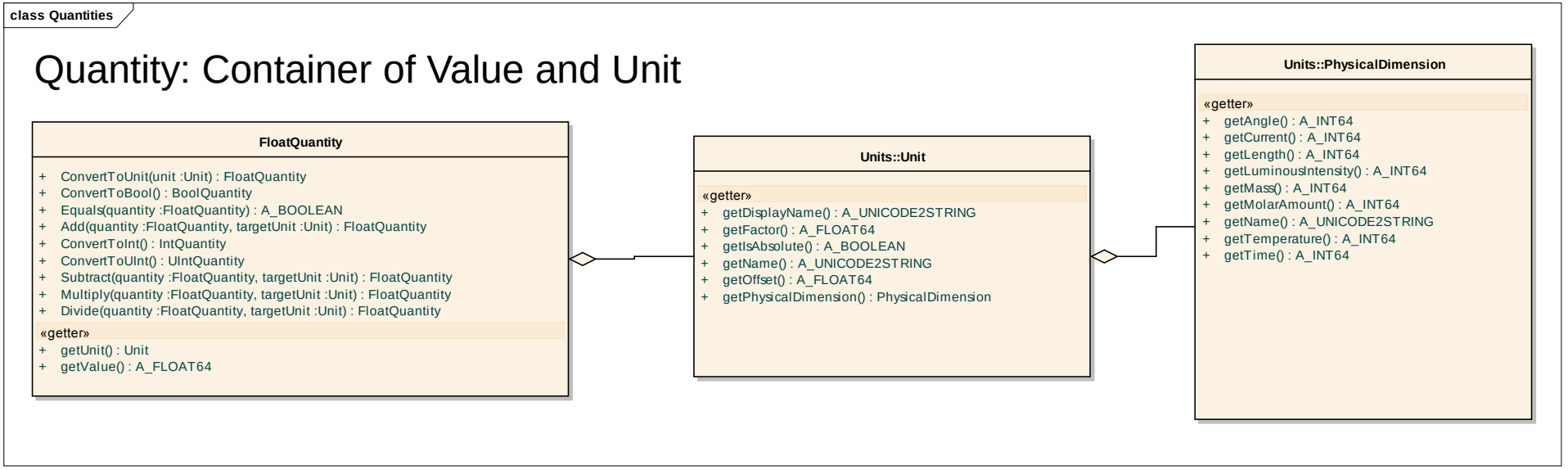


Reading a value – an Example



Framework Variable: Reads and Writes Quantities

```
IFloatVariable A = (IFloatVariable)FrameworkControl.CreateVariable("engine speed");
FloatQuantity aValue = A.Read();
double nativeValue =  aValue.Value;
```



Type Conversion of Quantities

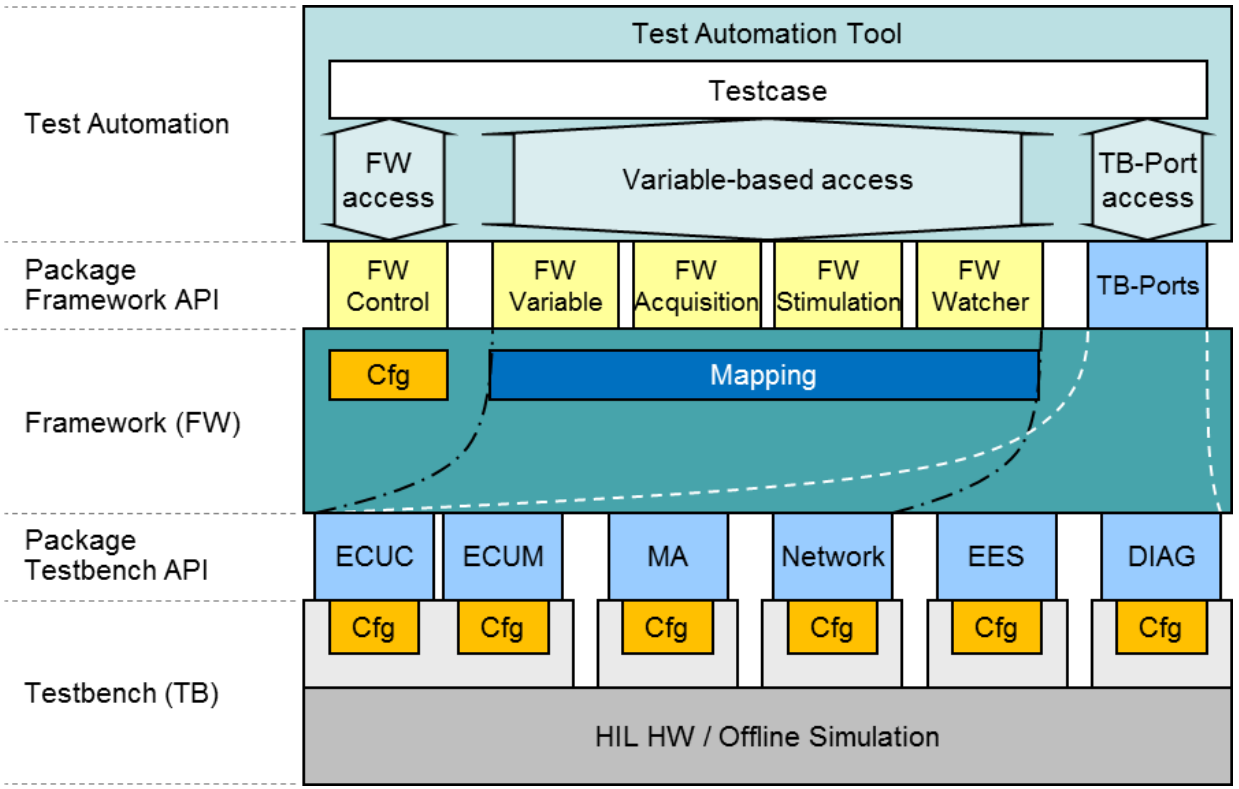
Quantity	Conversion Method	Rule to apply	New Quantity	Unit of the new Quantity
BoolQuantity	ConvertToFloat()	True ---> 1 False --> 0	FloatQuantity	Empty Unit[1]
	ConvertToInt()	True ---> 1 False --> 0	IntQuantity	Empty Unit[1]
	ConvertToUInt()	True ---> 1 False --> 0	UIntQuantity	Empty Unit[1]
UIntQuantity	ConvertToFloat()	[2]	FloatQuantity	Same Unit as original Quantity
	ConvertToInt()	MaxUint / 2 --> MaxInt MaxUint --> -1	IntQuantity	Same Unit as original Quantity
	ConvertToBool()	!0 --> True otherwise --> False	BoolQuantity	No Unit
IntQuantity	ConvertToFloat()	[2]	FloatQuantity	Same Unit as original Quantity
	ConvertToUInt()	MaxInt --> MaxUint / 2 -1 --> MaxUint	UIntQuantity	Same Unit as original Quantity
	ConvertToBool()	!0 --> True otherwise --> False	BoolQuantity	No Unit
FloatQuantity	ConvertToInt()	float --> round --> int	IntQuantity	Same Unit as original Quantity
	ConvertToUInt()	float --> round --> uint	UIntQuantity	Same Unit as original Quantity
	ConvertToBool()	!0.0 --> True otherwise --> False	BoolQuantity	No Unit

What's New?

- Maintenance Issues (motivated by market and crosstests)
- Extensions on existing ports (motivated by market)
- Extensions on existing ports (motivated by XIL 2.0.0 Framework)
- Network Port (new)
- New Features of XIL 2.0.0 Framework
  - Framework Variables
  - Mapping
  - Measuring
  - Stimulation
  - Configuration
- Software
  - XIL Support Library
  - Example Framework
  - Prototype and Test environment

Motivation for Mapping

- Exchanging the testbench without changing the tests for Variable-based access with respect to the variable‘s
  - Port connection
  - Unit
  - Type
  - Identifier



Mapping

- **Identifier Mapping**
  Maps a framework label to a testbench label.
  E. g. maps an abstract identifier of a test case variable to a concrete identifier within the simulation model
- **String Mapping**
  Maps framework strings to test bench strings.
  For example this can be used to map abstract filenames on the framework side to concrete filenames on the test bench side
- **Raster mapping**
  Maps abstract raster names on the framework side to concrete raster names on the testbench side.
- **Based on XML Schema**

Mapping

- Identifier Mapping:

  Mapping of Identifiers, Units, Types on both sides: test case and test bench

- Advantages
  - ✓ Check for compatibility possible (a_1==a_2? ..., g_1==g_2?)
  - ✓ Compatible quantities can be converted (offset, scale)

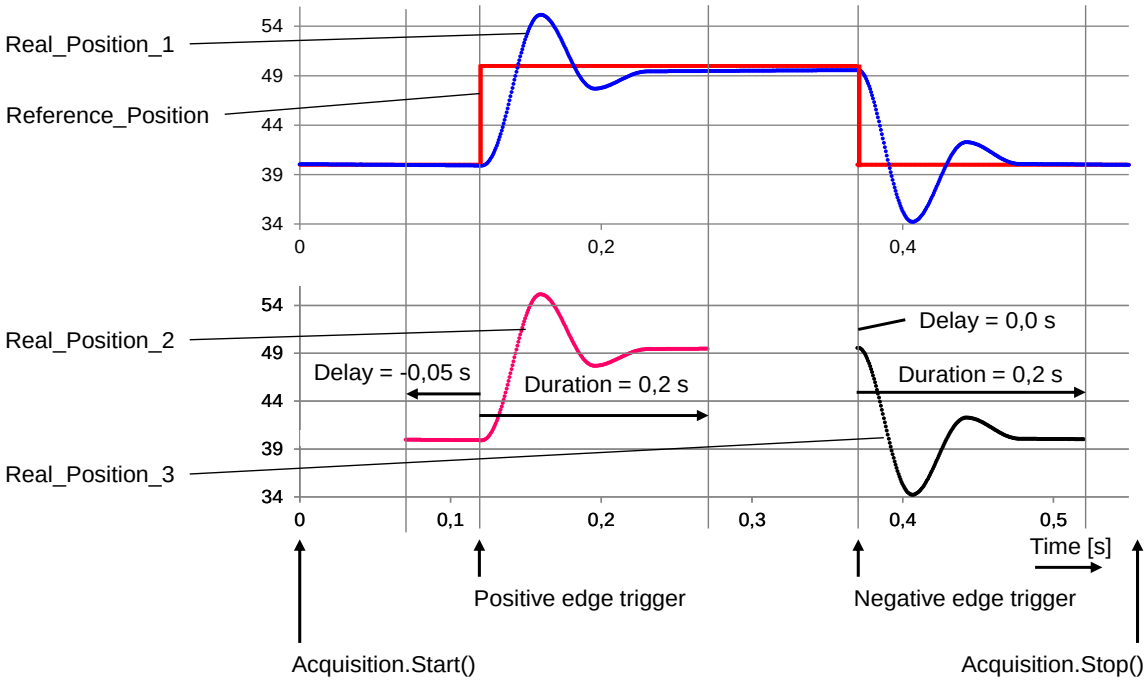
- Examples:
  - *[km/h] 0,0 + [3,6 * m^1 * s^(-1)]*
  - *[miles/h] 0,0 + [2,369 * m^1 * s^(-1)]*
  - *[°C] -273,15 + [1,0 * K^1 * 1]*
  - *[°F] -459,67 + [1,8 * K^1 * 1]*
  - *[kN] 0,0 + [1000,0 * m^1 * kg^1 * s^(-2)]*

What's New?

- Maintenance Issues (motivated by market and crosstests)
- Extensions on existing ports (motivated by market)
- Extensions on existing ports (motivated by XIL 2.0.0 Framework)
- Network Port (new)
- New Features of XIL 2.0.0 Framework
  - Framework Variables
  - Mapping
  - Measuring
  - Stimulation
  - Configuration
- Software
  - XIL Support Library
  - Example Framework
  - Prototype and Test environment

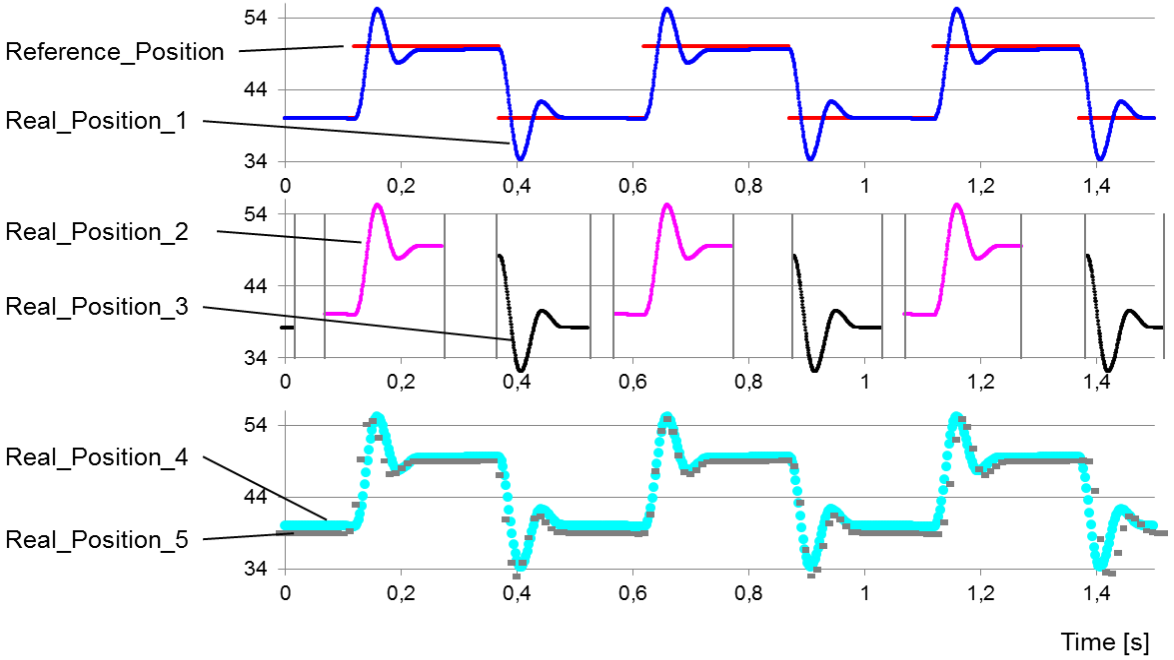
Measuring (1)

- collecting time traces of variables, e. g. for analyzing the effects of changes within simulation models and ECUs on different ports



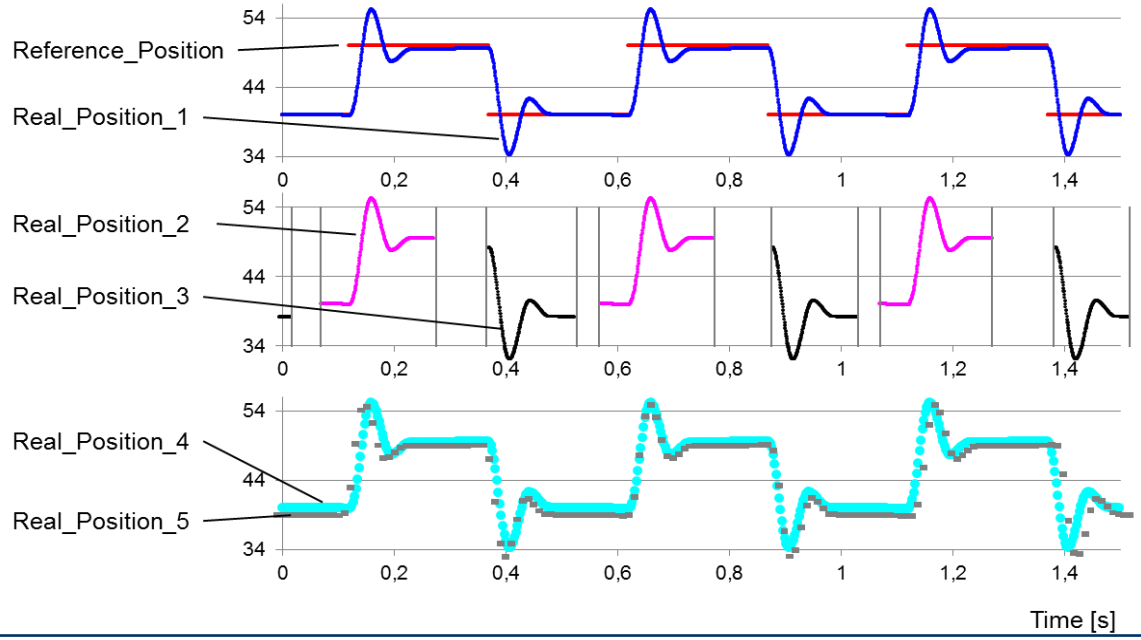
- Defining sophisticated trigger conditions
  - using signals across different ports
  - on a common framework time base

Measuring (2)

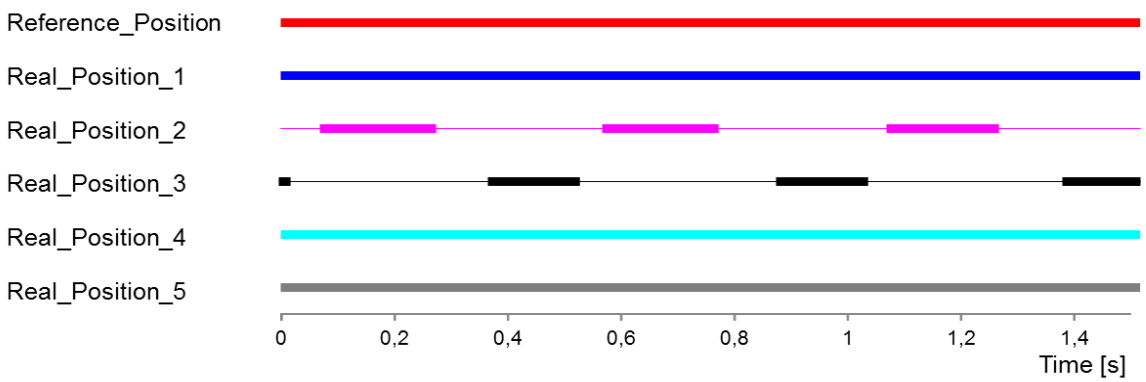


- Within one acquisition using
  - repeated start and stop triggering
  - different capture tasks and downsamplings
  - data sources from different ports

Measuring (2)

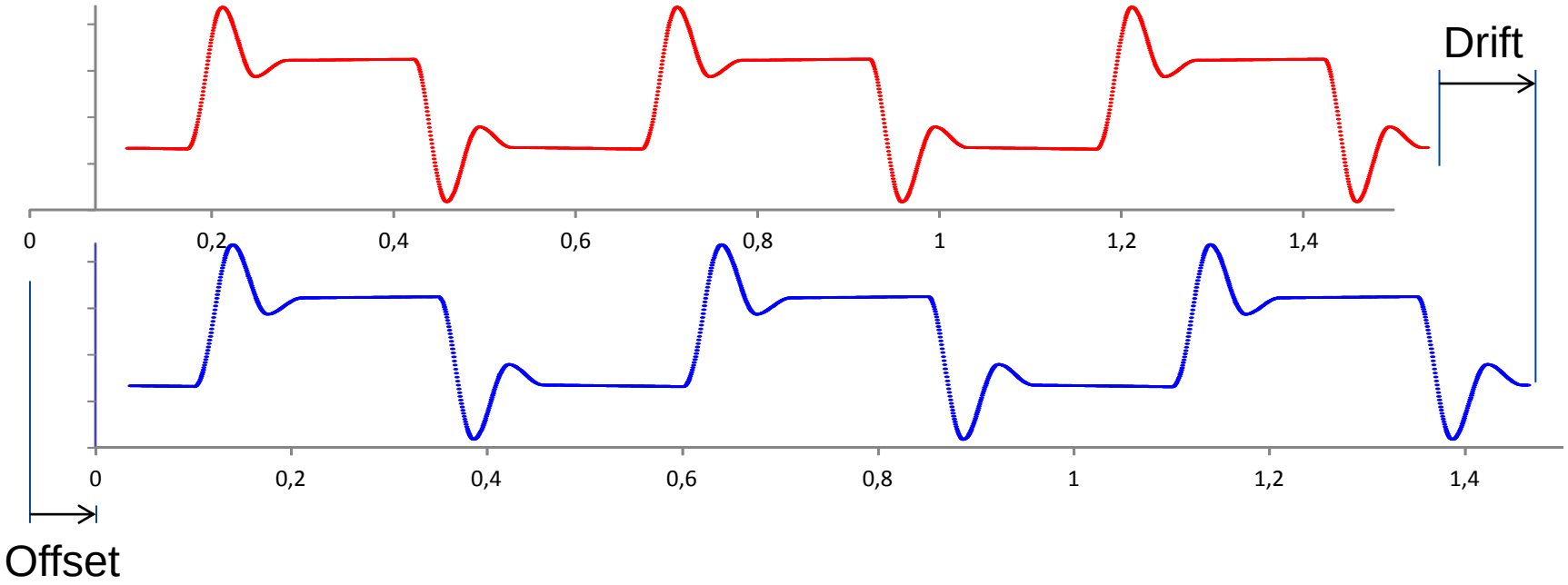


View 1:
Time traces as Plots



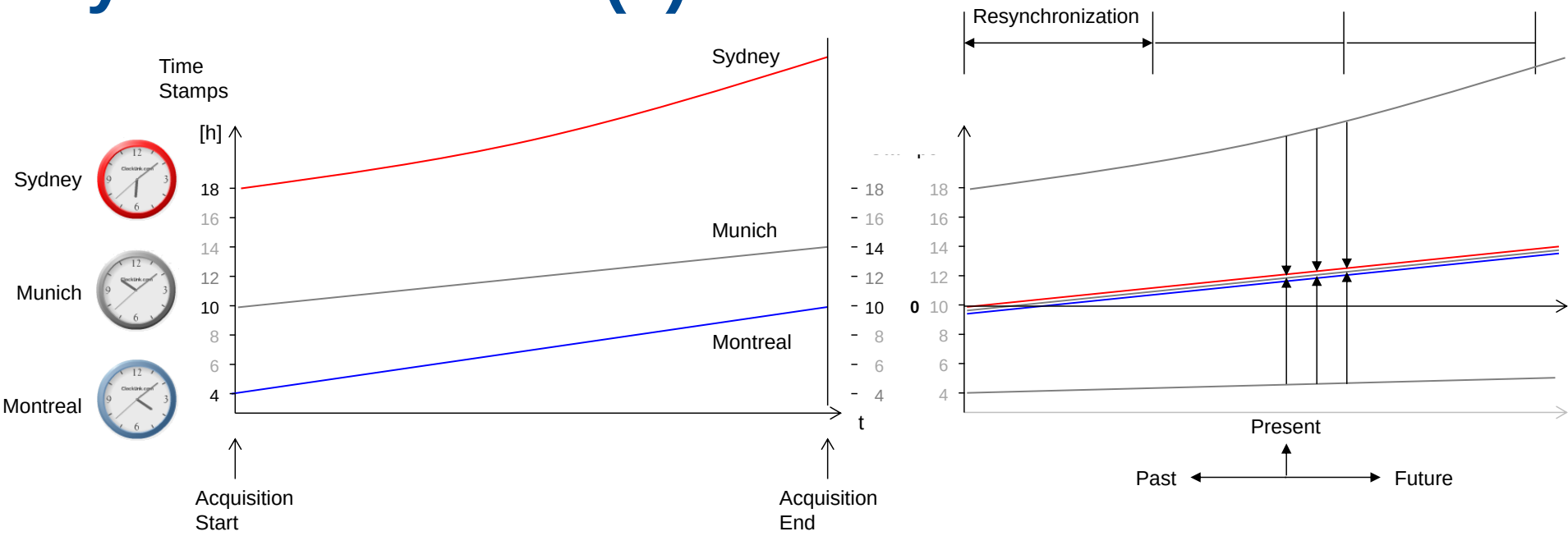
View 2:
Time traces as acquired data
Thick lines: data received
Thin lines: no data received due to invalid trigger condition

Synchronization (1)



- Different Hardware Times of different signals sources usually have
  - Offeset
  - Drift
- XIL 2.0.0 provides interfaces to configure compensation of Offset and Drift on a common time base

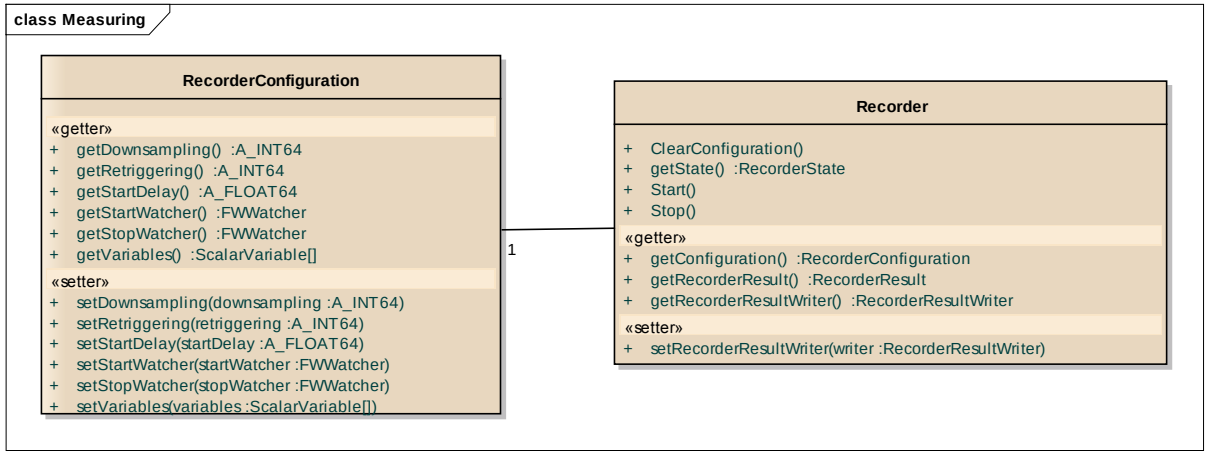
Synchronization (2)



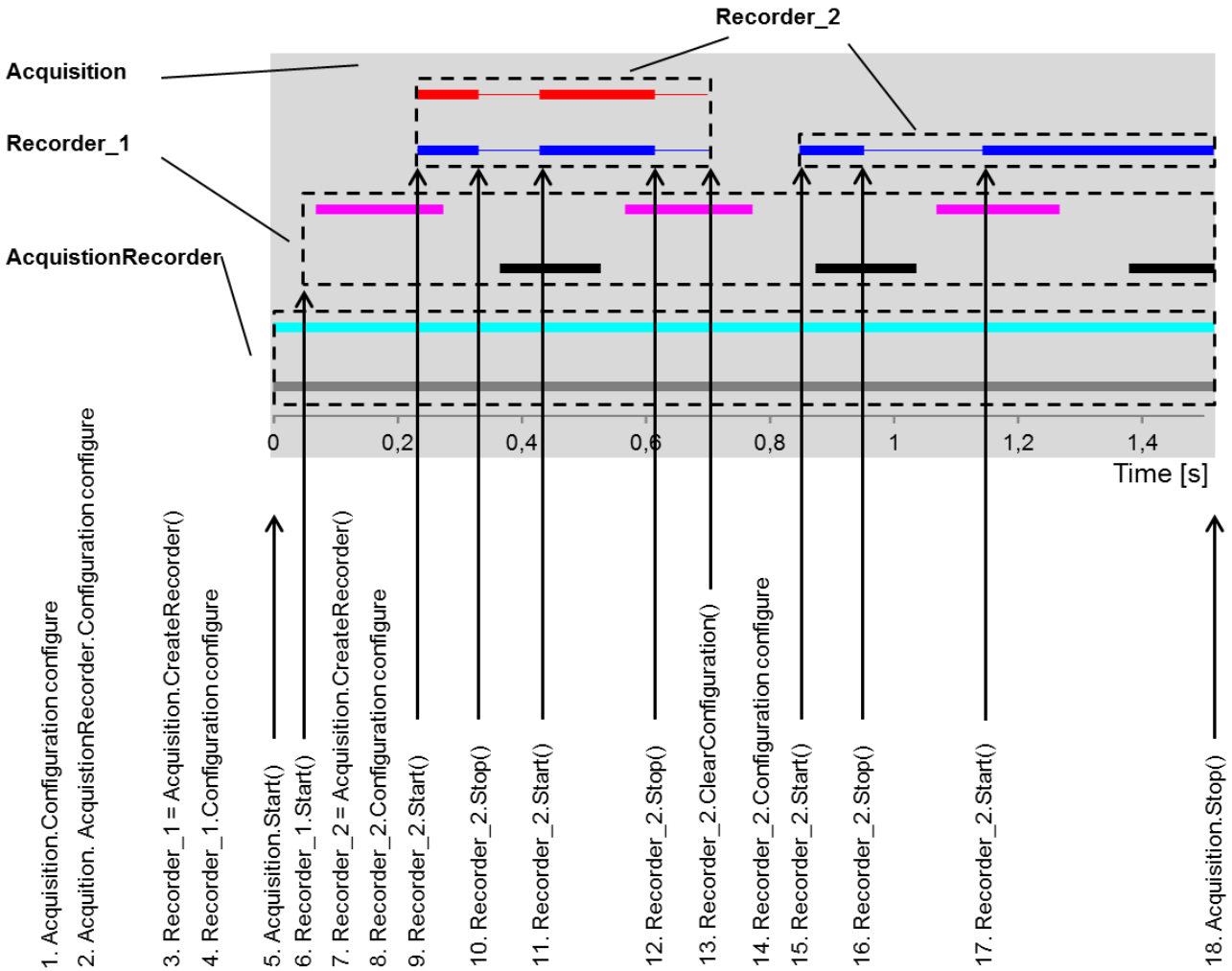
- Timers of different signal sources have offsets and drift apart due to different reset times and different frequencies which may depend on environmental factors such as changing temperatures.
- XIL 2.0.0 Framework allows to set an Resynchronization interface in order to re-calculate correction factors for offset and drift for future data acquisition based on measured data history (past).
- The synchronized data, that XIL 2.0.0 Framework has added to the acquistion, remains unchanged, which means there is no post precessing.
- The User can define recorders to store measured data either to memory or to file (MDF4 or higher)

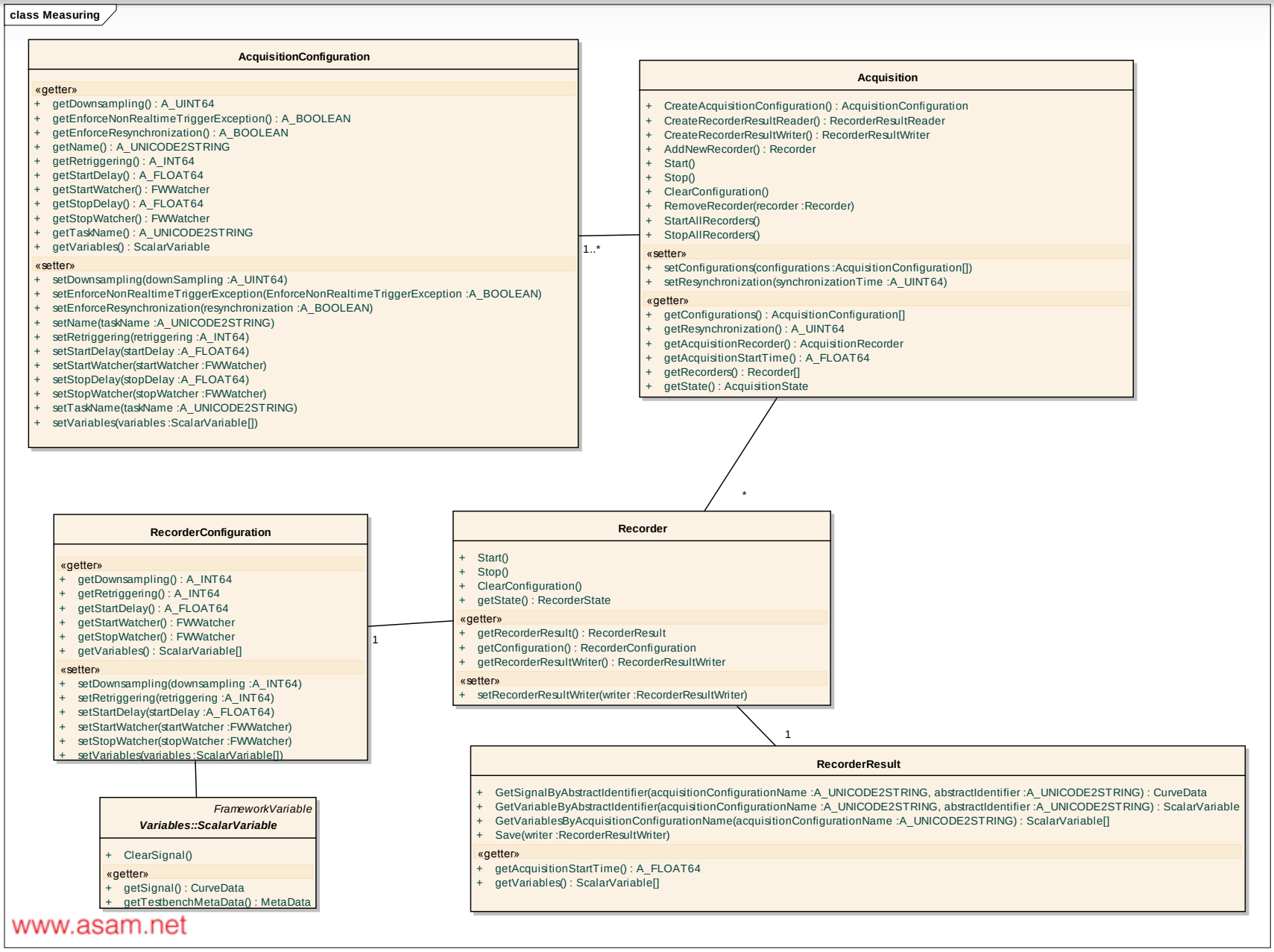
Recordings

- Make subsets of the acquisition data available to the user
- A recording result (i.e., a subset of the acquisition data) can exist either in memory as a RecorderResult object (similar to the CaptureResult known from HIL API 1.0.2) or as a file (e. g. in MDF4 format).
- Test developer can use these recording results for further analysis within the test case
- Configuration options can be used in order to focus on relevant parts of the acquired data.



Example: Acquisition and 3 Recorders

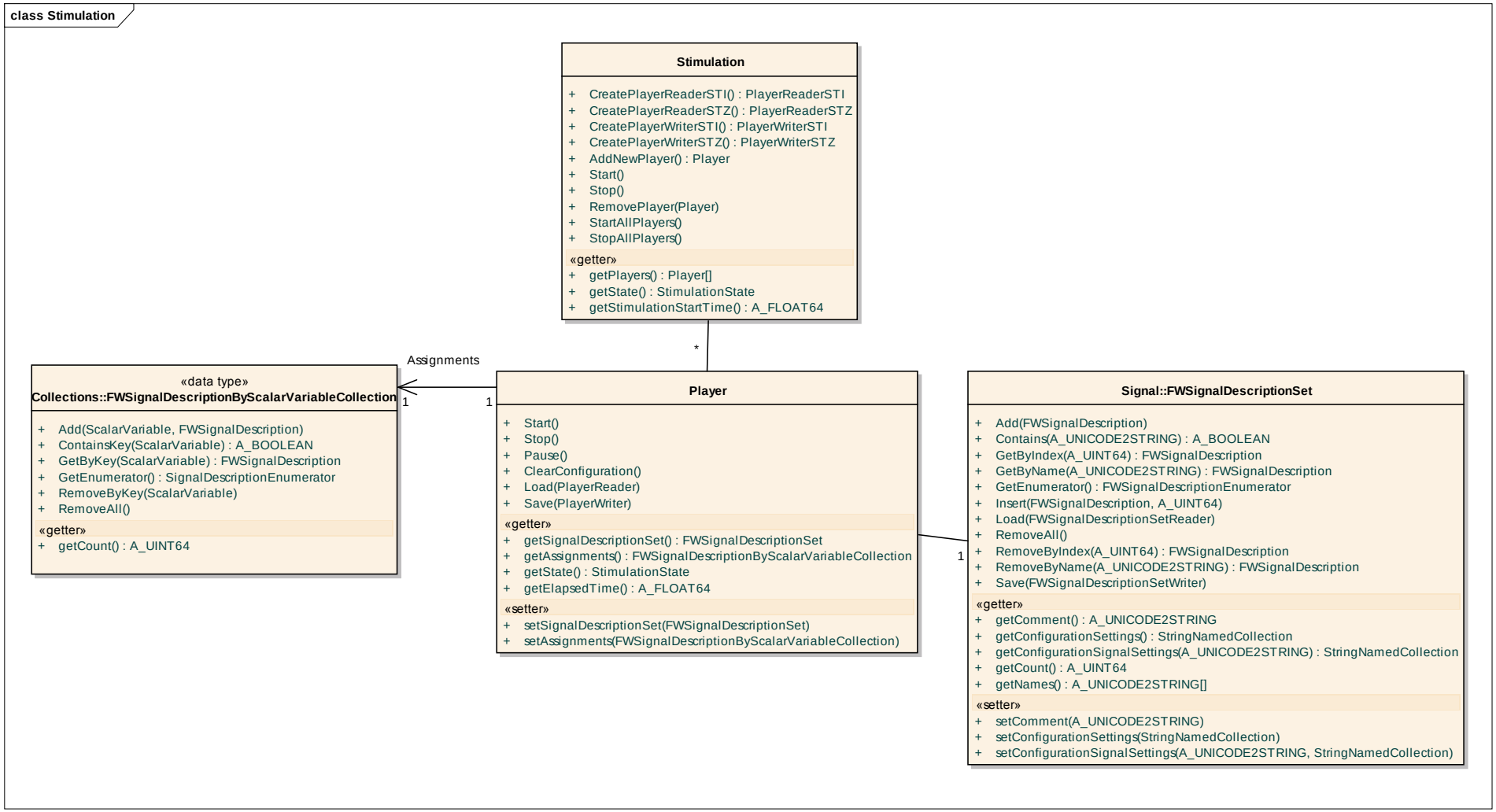



class Measuring
AcquisitionConfiguration
«getter»
+ getDownsampling() : A_UINT64
+ getEnforceNonRealtimeTriggerException() : A_BOOLEAN
+ getEnforceResynchronization() : A_BOOLEAN
+ getName() : A_UNICODE2STRING
+ getRetriggering() : A_INT64
+ getStartDelay() : A_FLOAT64
+ getStartWatcher() : FWWatcher
+ getStopDelay() : A_FLOAT64
+ getStopWatcher() : FWWatcher
+ getTaskName() : A_UNICODE2STRING
+ getVariables() : ScalarVariable
«setter»
+ setDownsampling(downSampling :A_UINT64)
+ setEnforceNonRealtimeTriggerException(EnforceNonRealtimeTriggerException :A_BOOLEAN)
+ setEnforceResynchronization(resynchronization :A_BOOLEAN)
+ setName(taskName :A_UNICODE2STRING)
+ setRetriggering(retriggering :A_INT64)
+ setStartDelay(startDelay :A_FLOAT64)
+ setStartWatcher(startWatcher :FWWatcher)
+ setStopDelay(stopDelay :A_FLOAT64)
+ setStopWatcher(stopWatcher :FWWatcher)
+ setTaskName(taskName :A_UNICODE2STRING)
+ setVariables(variables :ScalarVariable[])
1..*
Acquisition
+ CreateAcquisitionConfiguration() : AcquisitionConfiguration
+ CreateRecorderResultReader() : RecorderResultReader
+ CreateRecorderResultWriter() : RecorderResultWriter
+ AddNewRecorder() : Recorder
+ Start()
+ Stop()
+ ClearConfiguration()
+ RemoveRecorder(recorder :Recorder)
+ StartAllRecorders()
+ StopAllRecorders()
«setter»
+ setConfigurations(configurations :AcquisitionConfiguration[])
+ setResynchronization(synchronizationTime :A_UINT64)
«getter»
+ getConfigurations() : AcquisitionConfiguration[]
+ getResynchronization() : A_UINT64
+ getAcquisitionRecorder() : AcquisitionRecorder
+ getAcquisitionStartTime() : A_FLOAT64
+ getRecorders() : Recorder[]
+ getState() : AcquisitionState
*
RecorderConfiguration
«getter»
+ getDownsampling() : A_INT64
+ getRetriggering() : A_INT64
+ getStartDelay() : A_FLOAT64
+ getStartWatcher() : FWWatcher
+ getStopWatcher() : FWWatcher
+ getVariables() : ScalarVariable[]
«setter»
+ setDownsampling(downsampling :A_INT64)
+ setRetriggering(retriggering :A_INT64)
+ setStartDelay(startDelay :A_FLOAT64)
+ setStartWatcher(startWatcher :FWWatcher)
+ setStopWatcher(stopWatcher :FWWatcher)
+ setVariables(variables :ScalarVariable[])
1
Recorder
+ Start()
+ Stop()
+ ClearConfiguration()
+ getState() : RecorderState
«getter»
+ getRecorderResult() : RecorderResult
+ getConfiguration() : RecorderConfiguration
+ getRecorderResultWriter() : RecorderResultWriter
«setter»
+ setRecorderResultWriter(writer :RecorderResultWriter)
1
FrameworkVariable
Variables::ScalarVariable
+ ClearSignal()
«getter»
+ getSignal() : CurveData
+ getTestbenchMetaData() : MetaData
RecorderResult
+ GetSignalByAbstractIdentifier(acquisitionConfigurationName :A_UNICODE2STRING, abstractIdentifier :A_UNICODE2STRING) : CurveData
+ GetVariableByAbstractIdentifier(acquisitionConfigurationName :A_UNICODE2STRING, abstractIdentifier :A_UNICODE2STRING) : ScalarVariable
+ GetVariablesByAcquisitionConfigurationName(acquisitionConfigurationName :A_UNICODE2STRING) : ScalarVariable[]
+ Save(writer :RecorderResultWriter)
«getter»
+ getAcquisitionStartTime() : A_FLOAT64
+ getVariables() : ScalarVariable[]

What's New?

- Maintenance Issues (motivated by market and crosstests)
- Extensions on existing ports (motivated by market)
- Extensions on existing ports (motivated by XIL 2.0.0 Framework)
- Network Port (new)
- New Features of XIL 2.0.0 Framework
  - Framework Variables
  - Mapping
  - Measuring
  - Stimulation
  - Configuration
- Software
  - XIL Support Library
  - Example Framework
  - Prototype and Test environment

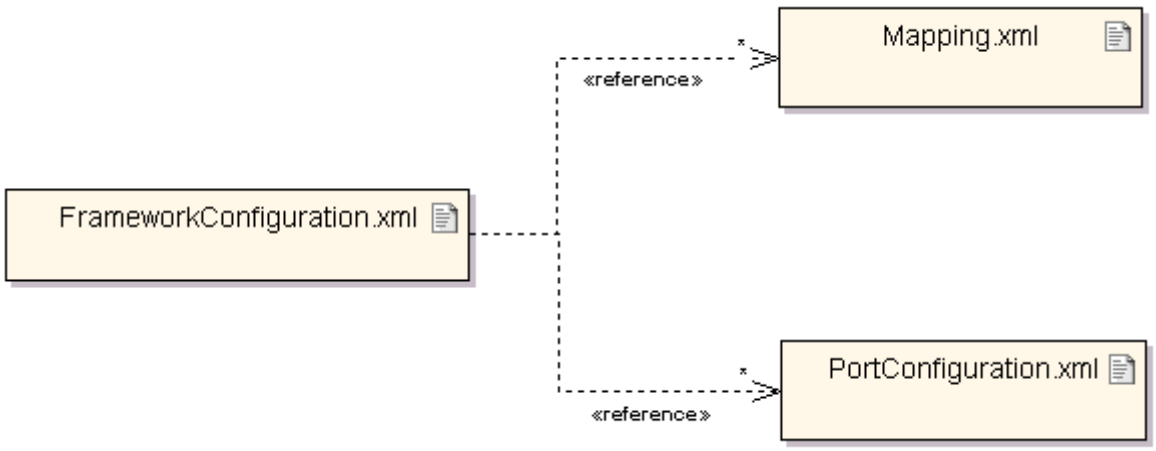
Stimulation



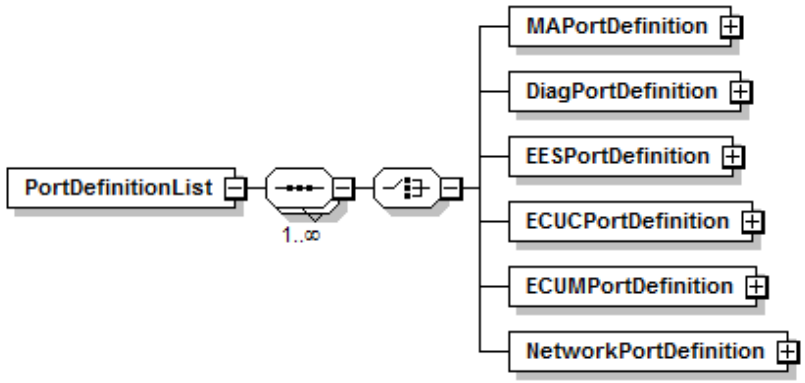
What's New?

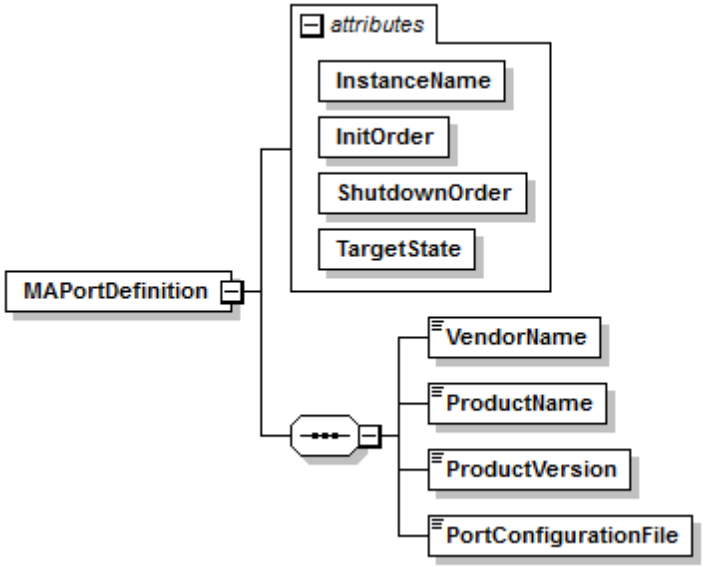
- Maintenance Issues (motivated by market and crosstests)
- Extensions on existing ports (motivated by market)
- Extensions on existing ports (motivated by XIL 2.0.0 Framework)
- Network Port (new)
- New Features of XIL 2.0.0 Framework
  - Framework Variables
  - Mapping
  - Measuring
  - Stimulation
  - Configuration
- Software
  - XIL Support Library
  - Example Framework
  - Prototype and Test environment

Configuration



Configuration

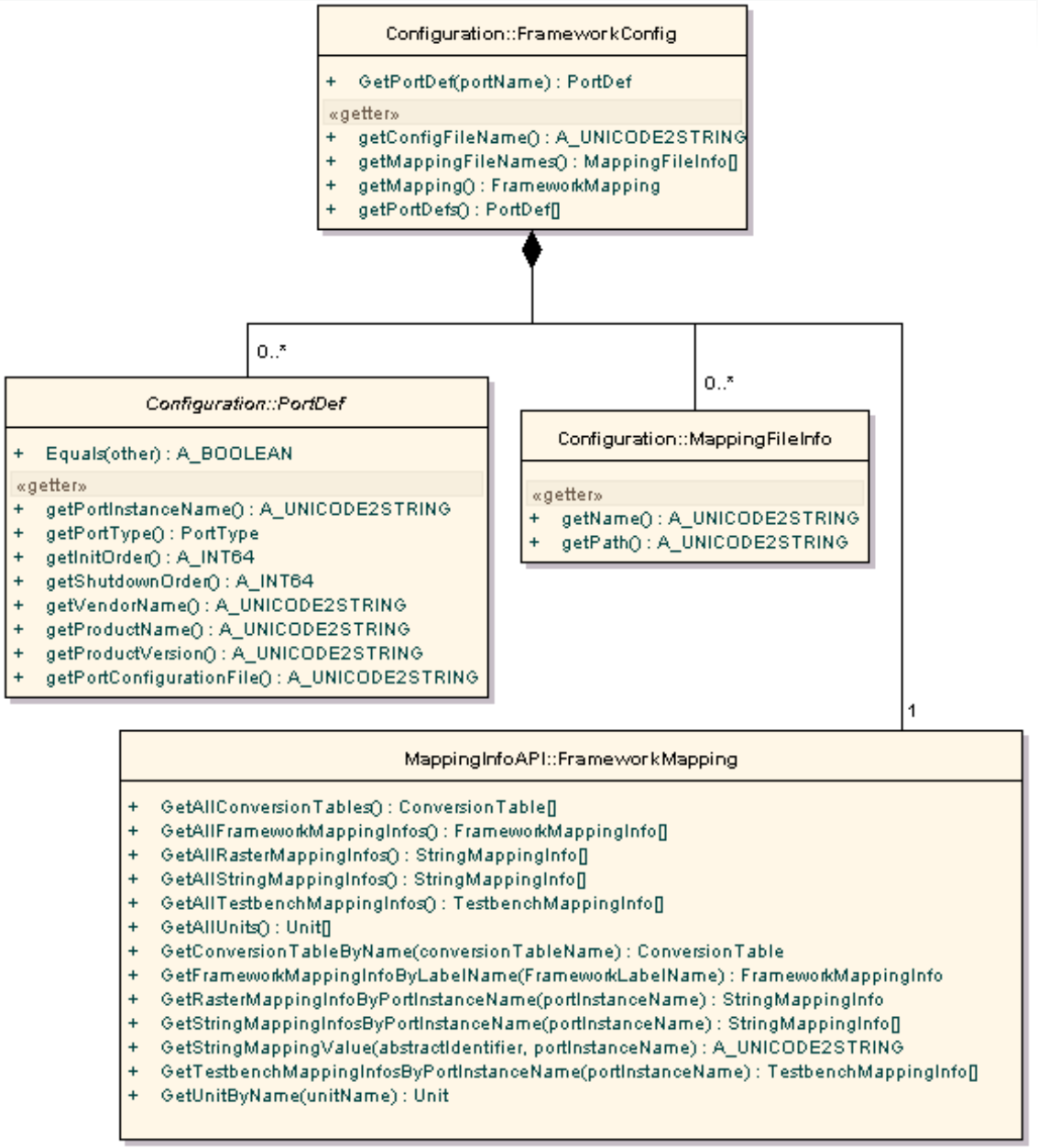




Configuration

PortDefinition type	TargetStates
MAPortDefinition	eSIMULATION_STOPPED
	eSIMULATION_RUNNING
DiagPortDefinition	eCONNECTED
EESPortDefinition	eCONNECTED
	eDOWNLOADED
	eACTIVATED
ECUCPortDefinition	eOFFLINE
	eONLINE
ECUMPortDefinition	eMEASUREMENT_STOPPED
	eMEASUREMENT_RUNNING
NetworkPortDefinition	eSTOPPED
	eRUNNING

Configuration

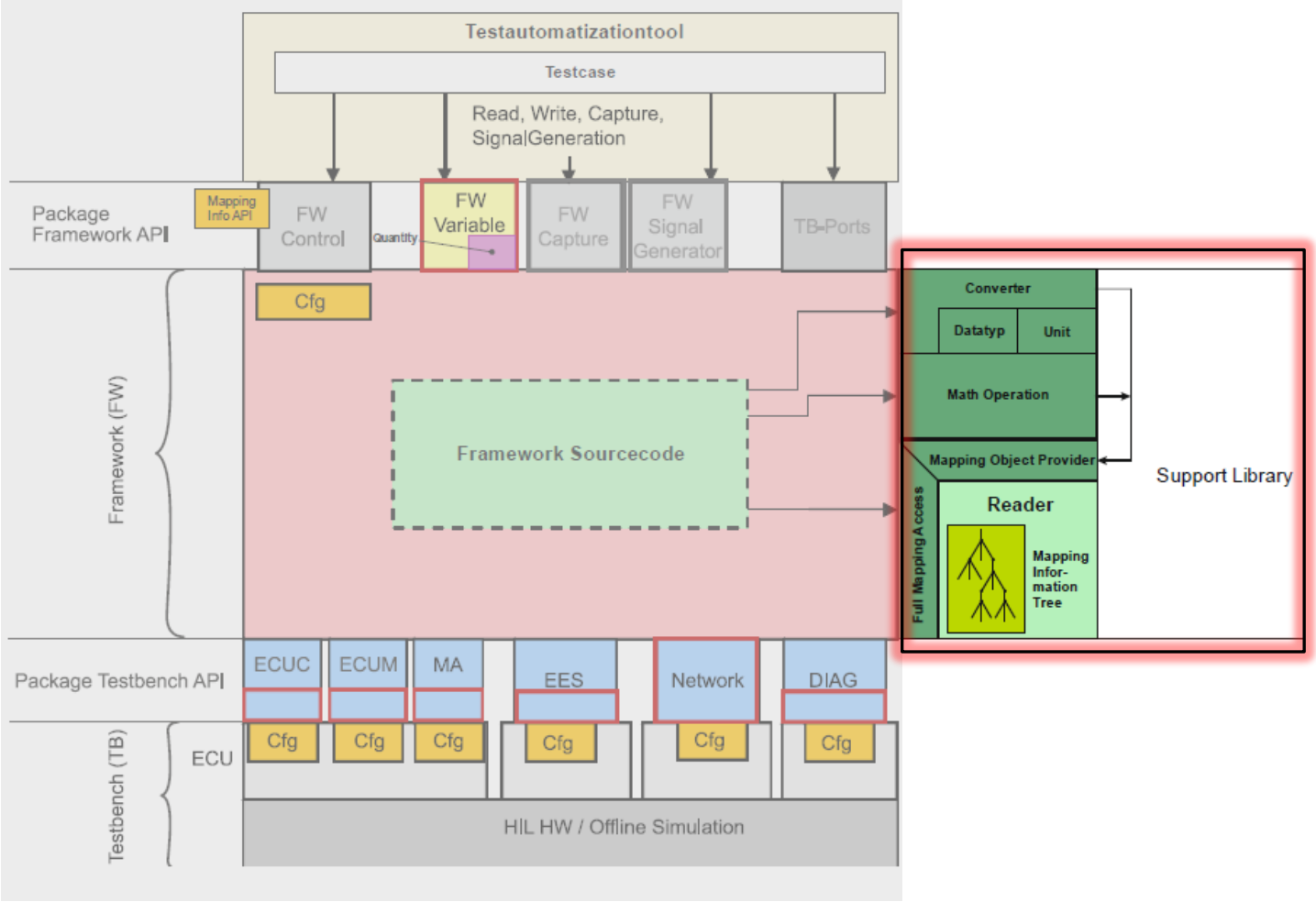


What's New?

- Maintenance Issues (motivated by market and crosstests)
- Extensions on existing ports (motivated by market)
- Extensions on existing ports (motivated by XIL 2.0.0 Framework)
- Network Port (new)
- New Features of XIL 2.0.0 Framework
  - Framework Variables
  - Mapping
  - Measuring
  - Stimulation
  - Configuration

- Software
  - XIL Support Library
  - Example Framework
  - Prototype and Test environment

XIL Support Library



Context

- XIL support library is open source software for:
  - Test case developers and framework vendors
  - The implementation of vendor independent common tasks like:
    - For both, framework and test case
      - Unit converter
      - Data type converter
      - Framework and test bench access
    - For framework vendors
      - 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 developers
      - Realization of framework mapping Info API
      - Mathematical operations

Content of XIL Support Library

- Mapping Reader
  - XML source construction (import a mapping file)
  - Full and Smart Access (for internal framework usage)
- Converter
  - unit converter
  - data type converter
- MathOperation (functionality for mathematical operation)
  - Calculation of quantities for scalar
- Framework variables
  - framework variables, MetaData, quantities and units, quantity-BaseValue transformer
- Framework mapping info API
- Framework and test bench factory

What's New?

- Maintenance Issues (motivated by market and crosstests)
- Extensions on existing ports (motivated by market)
- Extensions on existing ports (motivated by XIL 2.0.0 Framework)
- Network Port (new)
- New Features of XIL 2.0.0 Framework
  - Framework Variables
  - Mapping
  - Measuring
  - Stimulation
  - Configuration

- Software
  - XIL Support Library
  - Example Framework
  - Prototype and Test environment

Content of Example Framework

- Example Framework
  - Useage of datatype and quantity convertor and Mapping reader
  - Configuration of framework and test bench
  - Resolver (simulated MAPortConnector)
  - MA Port Dummy for value access at test bench
- XILSupportLibraryGoldFile (example file)
  - Example XML mapping gold file for the example framework
  - For test cases and unit tests
  - Contains framework variables and test bench values for all types
  - Can be used to show conversion of units and physical dimensions
- XIL user Example application
  - LoadConfiguration - Init
  - access via Framework Mapping Info API
  - Create variable and read and write
  - shutdown

What's New?

- Maintenance Issues (motivated by market and crosstests)
- Extensions on existing ports (motivated by market)
- Extensions on existing ports (motivated by XIL 2.0.0 Framework)
- Network Port (new)
- New Features of XIL 2.0.0 Framework
  - Framework Variables
  - Mapping
  - Measuring
  - Stimulation
  - Configuration

- Software
  - XIL Support Library
  - Example Framework
  - Prototype and Test environment

Content of Test environment

Test and unit test

- **150** test cases related to requirements defined
- **486** additional test cases (unit tests) written to maximize code coverage
- **Unit test** available for implementation of
  - Framework Variables
  - Quantities and Units
  - Data type Conversions
  - Unit Conversion
  - Mathematical operations
  - Mapping file reader
  - Mapping info API
  - Mapping object provider
- Unit tests are available as MS Visual Studio Test project and NUNIT tests
- Test and prototype environment with all unit tests and implementation available for ASAM

➔ Members are able to build their own framework from this point

Software Feedback

- Significantly improvement of standard quality
  - Proving of variable and mapping concepts
  - Checking of performance
  - Prototype realization
  - identified gaps in the standard
- Modification of XIL Standard for better handling
  - Introduction of **Framework Mapping Info API**
  - **Computation table** defined **with concept of user defined identifiers** and **introduced**
  - **Neutral unit** for using data type conversions according to boolean typed quantities and defining unit less constants of any type **added**
  - Mapping xsd tuning
  - Methods for **data type/unit conversions** and **mathematical operations added**
  - **Exception** implementation **added**
  - Permanently **unit switch** on framework variable introduced
  - Access about dimensions and read/write permissions added

Compatibility

- With XIL 2.0.0, all constructors have been replaced by factory methods
  Advantages:
  - All classes, represented by UML-based XIL API object model can be transformed into interfaces
  - Frees the test code from vendor-specific name space information, which would be necessary for calling constructors. Thus, code is free of vendor-specific information, what facilitates the exchange of tests among different testbenches
  - C# (this is the primary programming language provided by vendors in the market and covered by crosstests) does not provide constructors within its interface concept
- Re-organization within the UML package structure leads to changes within the namespaces
- Some changes in order to configure all ports in the same way
- Minor changes, since UML and Technology References (e. g. Interfaces) did not match always

Deliverables

- Package Standard
- Package XilSupportLibrary (framework software parts)
- Package Example Framework
- Package Prototype and test environment
- Package MSI Setup

Changes in Maintenance XIL 2.0.1

based on XIL 2.0.0

- Bugfixing of some workblockers, that have been detected during implementation phase, such as scalar was returned instead of a list of scalars;
  Signal generator factory now returns the correct ISignalGeneratorSTZWriter instead of ISignalGeneratorSTZReader;
  added missing value 'eDATAFILE' to SegmentTypes enum.
- Definition of Initial values to avoid invalid object creation of the new DataFileSegment class.
- Added some missing error codes and post conditions
- Added a new MAPort method GetTaskInfos to get information about existing tasks (eTimerDriven, eEventDriven, Sample Period)
- Added some functionality for simultanious read access of multiple clients and threads to the Testbench Manifest File (contains vendor-specific information about the Testbench)
- Capture now derives from Interface IDisposable to enable explicit instance destruction of Capture (to free system resources, e. g. real-time service code)
- Correction of errors in documentation (Guide and UML Model, e. g. SetStartTriggerCondition)
- Introduction of the correct short name in all documents:
  "XIL - Generic Simulator Interface" instead of “XIL - API for ECU Testing via XIL“