<table>
<thead>
<tr>
<th>Deliverable No.</th>
<th>HIFI-ELEMENTS D3.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deliverable Title</td>
<td>Simulation Framework as xMOD simulation extension</td>
</tr>
<tr>
<td>Deliverable Date</td>
<td>2019-06-30</td>
</tr>
<tr>
<td>Deliverable Type</td>
<td>REPORT</td>
</tr>
<tr>
<td>Dissemination level</td>
<td>Public (PU)</td>
</tr>
<tr>
<td>Written By</td>
<td>Dr. Hassen HADJ-AMOR, FEV STS SA</td>
</tr>
<tr>
<td></td>
<td>2019-07-17</td>
</tr>
<tr>
<td>Checked by</td>
<td>Jens Ewald, FEV</td>
</tr>
<tr>
<td></td>
<td>2019-08-14</td>
</tr>
<tr>
<td>Approved by</td>
<td>Steering Committee</td>
</tr>
<tr>
<td></td>
<td>2019-09-06</td>
</tr>
<tr>
<td>Status</td>
<td>Final</td>
</tr>
<tr>
<td></td>
<td>2019-08-17</td>
</tr>
</tbody>
</table>
H2020-GV7-2017 – 769935 – High Fidelity Electric Modeling and Testing

Acknowledgement

The author(s) would like to thank the partners involved with the project for their valuable comments on previous drafts and for performing the review.

Project partners
1 – FEV – FEV Europe GmbH – DE
2 – RWTH – Rheinisch-Westfälische Technische Hochschule Aachen – DE
3 – RIC – Ricardo UK Limited – UK
4 – TEC – Fundacion TECNALIA Research & Innovation – ES
5 – CID – Fundacion CIDETEC – ES
6 – IVE – IVECO S.p.A. – IT
7 – DEN – Denso Automotive Deutschland GmbH – DE
8 – TOF – TOFAS Turk Otomobil Fabrikasi A. S. – TR
9 – IDI – IDIADA Automotive Technology SA – ES
10 – TNO – Nederlandse Organisatie voor Toegepast-natuurwetenschappelijk Onderzoek – NL
11 – MGEP – Mondragon Goi Eskola Politeknikoa J.M.A. S.Coop – ES
13 – VUB – Vrije Universiteit Brussel – BE
14 – MAG – MAGNA Engineering Center Steyr GmbH & Co. KG – AT
15 – YASA – YASA Motors Limited – UK
16 – FORD – Ford Werke GmbH – DE

Disclaimer:
This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 769935.
## Document information

### Additional author(s) and contributing partners

<table>
<thead>
<tr>
<th>Name</th>
<th>Organisation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hassen HADJ-AMOR</td>
<td>FEV STS SA</td>
</tr>
</tbody>
</table>

## Document Change Log

<table>
<thead>
<tr>
<th>Name</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>V0.1</td>
<td>2019-07-17</td>
<td>Initial draft</td>
</tr>
<tr>
<td>V1.0</td>
<td>2019-07-31</td>
<td>Version for Review</td>
</tr>
</tbody>
</table>

## Document Distribution Log

<table>
<thead>
<tr>
<th>Name</th>
<th>Organisation</th>
<th>Distributed to</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Verification and approval

<table>
<thead>
<tr>
<th>Name</th>
<th>Organisation</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Executive summary
This deliverable describes the required extensions to the xMOD simulation Framework to be integrated in the AMDF toolchain. A dedicated format to define a complete system consisting of one or more FMUs and xMOD models including the simulation configuration is introduced to support automatic creation of xMOD simulations. To remotely control the xMOD simulations from external tools (SYNECT, Test Framework), the support of XIL API standard is implemented. Test cases are also executed externally by means of XIL API functions.

Additional features are also developed to support real-time testing.
Contents

1 Purpose of the Document ........................................................................................................ 7
  1.1 Document Structure ........................................................................................................... 7
  1.2 Deviations from original Description in the Grant Agreement Annex 1 Part A ............... 7
    1.2.1 Description of work related to deliverable in GA Annex 1 – Part A ......................... 7
    1.2.2 Time deviations from original planning in GA Annex 1 – Part A ......................... 7
    1.2.3 Content deviations from original plan in GA Annex 1 – Part A ............................ 7
  2 Introduction .......................................................................................................................... 8
    2.1 Overall Target of the HIFI-ELEMENTS project ......................................................... 8
    2.2 Target of Work Package 3 (WP3) ................................................................................. 8
  3 Architecture ......................................................................................................................... 10
    3.1 Extending xMOD (ZXMIPS) ......................................................................................... 11
      3.1.1 XMIPS schema file ................................................................................................. 11
      3.1.2 Importing a ZXMIPS ............................................................................................. 12
    3.2 XIL API MA support ................................................................................................... 13
    3.3 FMU for Real-Time ...................................................................................................... 16
      3.3.1 Export of RTX based FMUs .................................................................................... 16
    3.4 Supporting EtherCAT protocol for HiL applications ................................................... 18
      3.4.1 EtherCAT driver .................................................................................................... 19
    3.5 Supporting Simulink Protected models ....................................................................... 20
  4 Summary ............................................................................................................................... 22
  5 Risk Register ....................................................................................................................... 23
    5.1 References ................................................................................................................... 23
  6 Quality Assurance ............................................................................................................... 24
  7 Acknowledgment .................................................................................................................. 25

Figure 1-The AMDF toolchain architecture ............................................................................... 10
Figure 2-Usage of the XIL API Model Access Port .............................................................. 11
Figure 3- XMIPS Schema graphical view ................................................................................... 12
Figure 4- Importing ZXMIPS from the xMOD Editor .......................................................... 13
Figure 5 Model access (MAPort) standard ............................................................................... 13
Figure 6 Code snippets: Initialization of the FEV XIL Implementation ............................... 14
Figure 7 Code snippets: MAPort creation and configuration .................................................. 14
Figure 8 Code snippets: Reading a value .............................................................................. 15
Figure 9 Code snippets: Writing a value ................................................................................ 15
Figure 10 Code snippets: Generating Stimulus Signals ....................................................... 16
Figure 11 xMODFMU Generator overview ............................................................................ 17
Figure 12 xMODFMU Generator main window ..................................................................... 18
Figure 13 xMOD FMUGenerator toolchain verification ...................................................... 18
Figure 14 EtherCAT solution in xMOD ................................................................................. 19
Figure 15 EtherCAT bus configuration between xMOD and KPA Studio .............................. 19
Figure 16 EtherCAT Native block in xMOD library ............................................................. 20
Figure 17 Configuration block to prepare EtherCAT Receive and Send blocks in the xMOD MIPS ......................................................................................................................... 20
Tables
Table 1: D3.4 contribution to HIFI objectives ................................................................. 8
Table 2: Main elements of the XMIPS schema ................................................................. 11
Table 3: xMOD/FMI compatibility ................................................................................. 17
1 Purpose of the Document

This document is the forth deliverable of WP3. It presents the xMOD extension to support automatic testing and calibration in a global toolchain composed by third party software; SYNECT from dSPACE for Data Management and a test framework (Tool for Electrical Vehicle Testing) and simulation tool. The test framework as well as the Data Management tools are described in greater detail in respective deliverables. The focus of this document is to describe the extensions for the Simulation Framework xMOD to be able to complete the full tool chain.

1.1 Document Structure

The chapters of the document have the following focus.

Chapter 2: Introduces the goals of the HIFI project and WP3. Clarifies the scope of this deliverable.
Chapter 3: Presents the concepts of the architecture used to integrate the test and Data Management tool.
Chapter 4: Explains the support of Real-Time communication protocol for HIL applications
Chapter 5: Summary

1.2 Deviations from original Description in the Grant Agreement Annex 1 Part A

1.2.1 Description of work related to deliverable in GA Annex 1 – Part A

Within WP3, the component models (including parameter handling), the model integration process, as well as the test processes will be specified and implemented for offline as well as real-time simulation based on a central management system. Furthermore, the software framework to enable real-time co-simulation (including automated generation of system models) will be refined and extended by a test framework for automatic execution of functional and nonfunctional test cases on component and system level. Finally, the model-based calibration will be performed.

1.2.2 Time deviations from original planning in GA Annex 1 – Part A

Due to holiday season, the review process took longer than expected which results in a slight delay of the report completion and submission.

1.2.3 Content deviations from original plan in GA Annex 1 – Part A

There are no deviations from the Annex 1 – Part A with respect to the content.
2 Introduction

2.1 Overall Target of the HIFI-ELEMENTS project

The HiFi-Elements project aims to reduce the design and validation effort for e-drivetrains by defining interface standards and workflows to combat insufficient model reuse and to ensure model interoperability and scalability. With the development of a structured toolchain it is possible to realize different software (virtual validation software, model data management software) to let them work in symbiosis by following a standardized automated path. Within the developed workflow, automatic parameter identification functions and automatic test case generation will also be implemented. The proposed workflow and interface standards will provide the possibility of earlier system validation and, as a result, it aims to:

- Increase the efficiency of the development process
- Decrease the development, testing and system integration effort to reduce time-to-market
- Increase safety and reliability of EVs

To test the improvement of the developed workflow and toolchain, four different use cases will be used to record the efforts expended for the different tasks in each of them. The recordings will be compared with the assessment of documented efforts from the past for similar tasks utilizing the current practice.

2.2 Target of Work Package 3 (WP3)

Within WP3, the component models (including parameter handling), the model integration process, as well as the test processes will be specified and implemented for offline as well as real-time simulation based on a central management system. Furthermore, the software framework to enable real-time co-simulation (including automated generation of system models) will be refined and extended by a test framework for automatic execution of functional and nonfunctional test cases on component and system level.

This deliverable summarizes the outcome of task 3.4 Simulation framework as xMOD simulation extensions.

This task covers the development of simulation framework based on xMOD software including coupling algorithm to perform seamless execution of real-time co-simulation and frontloading of model based calibration activities.

Contribution by partners:
FEV will develop a software framework (based on xMOD) for automated real-time and non-real-time co-simulation. This framework will support different types of model formats (FMI standard, SimulinkCoder...etc.). Necessary Plug-ins will be developed to make the framework compatible with data management and test framework system. The software framework will support also an advanced real-time communication protocol. FEV will deploy models for MiL and HiL testing. MGEP and DSPA will contribute to define the interfaces to data management system and test framework.

Table 1 lists the HIFI-Elements objectives which are supported by Task 3.4:

<table>
<thead>
<tr>
<th>HIFI Objective</th>
<th>Description</th>
<th>Deliverable D3.4 Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>O2a</td>
<td>Exchange models (within a component class) in a co-simulation tool chain</td>
<td>By supporting FMI standard and by providing its own format (xmodel), xMOD is able to handle models independent of the specific implementation.</td>
</tr>
<tr>
<td>O2b</td>
<td>Calibrate one model to another or calibrate a model to a dataset of physical test results, targeting a 50% reduction in effort.</td>
<td>Support of parameters calibration via XIL API standard</td>
</tr>
<tr>
<td>O2d</td>
<td>Create system co-simulation models that can be used in real-time HiL-test setups, targeting a 75% reduction in effort.</td>
<td>FMU support for Real-Time</td>
</tr>
<tr>
<td>O2e</td>
<td>Plan and execute a test program that meets a given set of test objectives, targeting a 50% reduction in effort.</td>
<td>Support of external test via XIL API standard</td>
</tr>
<tr>
<td>-----</td>
<td>-------------------------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>O4</td>
<td>Increase in validation test coverage up to 10-fold</td>
<td>Supporting automatic execution of test cases (in combination with automatic test code generation) allow to reach this goal.</td>
</tr>
</tbody>
</table>
3 Architecture

The toolchain which is developed in HIFI-Elements is composed of several software products. An overview of the AMDF toolchain architecture is shown in Figure 1.

The xMOD software [1] is the central runtime to execute different vehicle model architectures. These models are developed with different tools and have different file formats. The focus of this deliverable is to describe the extensions developed in xMOD to support:
- loading a description of connected models
- executing the simulation remotely by supporting the ASAM XIL API standard [2]
- Supporting external tests loaded from SYNECT [3] and created in the test framework tool (TEVET) developed by Mondragon. [4]
- Supporting the execution of Real-Time FMU models
- Supporting for EtherCAT standard for real-time communication with third party hardware

![The AMDF toolchain architecture](image)

*Figure 1-The AMDF toolchain architecture*

For details on the Test Framework and the Data management there are separate deliverables [4, 5].
3.1 Extending xMOD (ZXMIPS)

To create xMOD simulations automatically from a third party software (by SYNECT in this case) a specific model integration format called “ZXMIPS” is introduced. A ZXMIPS file is a ZIP archive containing all component model files (FMUs, XMODELs) and an XML file (XMIPS) containing all required meta data such as the input & output connections of the components.

3.1.1 XMIPS schema file

All static information related to an ZXMIPS is stored in the text file Simulation_Name.xmips in XML format. Especially, the models Inputs and outputs connections are stored in this file. The structure of this XML file is defined with the schema file “xmips.xsd” provided to WP3 partners to allow them to generate ZXMIPS simulation files.

The first line of an XML file, such as Simulation_Name.xmips, must contain the encoding scheme of the XML file. It is required that the encoding scheme is always UTF-8:

```xml
<?xml version="1.0" encoding="utf-8"?>
```

The first node in the XMIPS file defines the xMIPSDef root element. Information regarding the simulation configuration such as the execution mode and the stop time are defined.

On the top level, the schema consists of the following elements:

| Models | Lists the models used in the Simulation. 2 types of models can be used: FMU models and XMODEL models (native format of xMOD). A FMU model is described in a sub-element FMUBloc. The XMODEL model is defined in a sub-element ModelBloc. |
| Links | Lists all the links between the models independently from their types |
| Quantities | List all the variables to be used in the Simulation. The variables are scalars. This due to the fact that XIL API MA is not supporting vectors. Moreover, FMU models are not supporting vectors too. |

**Table 2: Main elements of the XMIPS schema**

This is the root-level schema file and contains the following definition (the figure below contains main elements in the schema file).
The schema of the XMIPS was checked continuously with WP3 members so that all required features are supported.

### 3.1.2 Importing a ZXMIPS

Importing a ZXMIPS file is possible with 2 possibilities:

- **xMOD Editor (manually)**
  
  The xMOD editor should allow a user to import a ZXMIPS file and to play it after the import he import feature should be accessible from the Simulation toolbar menu.
Figure 4 - Importing ZXMIPS from the xMOD Editor

From the simulation tool bar, the user can select the Import button. Then, a drop-down menu will give the possibility to import .xmod files (this is a native option in xMOD). The second line in the drop-down menu allows to import ZXMIPS files.

When the user clicks on "Import ZXMIPS" a dialog box is then open and asks the user to select a ZXMIPS file. An extension filter .zxmips should be applied.

A set of verification and check of errors is made automatically at each import. A corrupted ZXMIPS file will be rejected and not imported.

- **Command line**

  It was mandatory to provide a way to import ZXMIPS files automatically. The command line to import a ZXMIPS file is the following:

  \[ \text{MIPSLoader.exe \text{-f ImportZmxips(Filename,SimulationName)}} \]

  Filename: represents the ZXMIPS filename
  SimulationName: represents the simulation name to be created in xMOD based on the ZXMIPS filename.

### 3.2 XIL API MA support

xMOD provides a built-in API to control a simulation. It is called MAPI. This API is not suitable to fulfill all the requirements of the HIFI toolchain. It was decided commonly in WP3 to support the ASAM XIL API MA standard in xMOD. Consequently, SYNECT and TEVET can interact with xMOD based on a standard way. ASAM XIL-MA is an extract from ASAM XIL and contains the model access port specification as well as some common functionality from this standard. The main goal of this standard is to separate test automation tools from simulation tools and provide access to a simulation model. Furthermore, the API enables data exchange between different tools, which allows that test cases can run with different simulators without major migration efforts.

For details on the ASAM XIL API standard, visit [http://www.asam.net](http://www.asam.net)

The xMOD XIL MA .NET implementation is based on the ASAM XIL-MA standard version 2.1.0. It covers a large part of the Model Access Port (MAPort) for access to xMOD simulations in any mode (ASAP, Windows, RTX HIL). Several prototype versions of this implementation were exchanged with WP3 partners. A first official implementation of the ASAM XIL MA standard is available in xMOD 2019 Release.

![Figure 5 Model access (MAPort) standard](image-url)
To support XIL API in xMOD two main components were identified:
- The XIL API implementation
- The extension of the existing service (MAPI) to control xMOD remotely. Indeed, the XIL API standard proposes many functions and features to control completely a XIL server application. These functions are not implemented in the previous version of the MAPI xMOD service.

xMOD XIL MA .NET implementation supports a first level of the MAPort specifications.

Creating FEV XIL API TestbenchFactory and Testbench

xMOD XIL API .NET uses the factory pattern to create Testbench specific objects. A TestbenchFactory object should be created to prepare the vendor specific Testbench

```c#
ITestbenchFactory MyTestbenchFactory = new TestbenchFactory();
ITestbench MyTestbench = MyTestbenchFactory.CreateVendorSpecificTestbench("FEV SA", "xMOD", "2019");
```

```python
## Create TestBench
MyTestbenchFactory = TestbenchFactory();
MyTestbench = MyTestbenchFactory.CreateVendorSpecificTestbench("FEV SA", "xMOD", "2019");
```

Figure 6 Code snippets: Initialization of the FEV XIL Implementation

Creating and configuring MAPorts

Create and configure a MAPort object

```c#
IMAPortFactory MyMAPortFactory = MAPortFactory; IMAPort DemoMAPort = MyMAPortFactory.CreateMAPort("DemoMAPort"); string MAPortConfigFile = @"D:\\temp\\MAPortConfig.xMOD.xml";
IMAPortConfig MAPortConfig = DemoMAPort.LoadConfiguration(MAPortConfigFile);
DemoMAPort.Configure(MAPortConfig, false);
```

```python
## Create MAPort and load the configuration
DemoMAPort = MAPortFactory.CreateMAPort("Training MAP");
DemoMAPortConfig = DemoMAPort.LoadConfiguration(MAPortConfigFile);
DemoMAPort.Configure(MAPortConfig, 0);
```

Figure 7 Code snippets: MAPort creation and configuration

Below an example of FEV XIL API MAPort configuration file:

```xml
<?xml version="1.0" encoding="utf-8"?>
<MAPortConfig>
<ModelFile>C:\xMOD\Simulations\Test_XIL\Test_XIL.xsim</ModelFile>
<PlatformName>xMOD</PlatformName>
<Host>FRM0758</Host>
<Port>0</Port>
</MAPortConfig>
```

With this configuration file, you can access the Simulation named "Test_XIL" running on the PC named "FRM0758".
Once the XIL API MAPort is ready, then various operations can be done by the client: reading, writing, recording...etc.

XIL API port interface for model access

**Reading and Writing Values**

With the current xMOD XIL API .NET implementation it is possible to read and write values using MAPort instance. The following code snippet shows how to write a value in a specific variable in the simulation.

**C#**

```csharp
string Speed = "Test_XIL/Speed/Value";
IFloatValue readValue = (IFloatValue)DemoMAPort.Read(Speed);
```

**Python**

```python
#Read the value of "Test_XIL/Speed/Value"
Speed = "Test_XIL/Speed/Value";
readValue = DemoMAPort.Read(Speed);
```

**Figure 8 Code snippets: Reading a value**

**C#**

```csharp
string Speed = "Test_XIL/Speed/Value";
DemoMAPort.Write(Speed, testbench.ValueFactory.CreateFloatValue(1.0));
```

**Python**

```python
## Write a Value to the variable "MyValue"in xMOD
MyValueToHandle = "MyValue"
DemoMAPort.Write(MyValueToHandle, MyTestbench.ValueFactory.CreateFloatValue(1));
```

**Figure 9 Code snippets: Writing a value**
Measuring Variables
Capturing and Watcher are not supported for this version of xMOD XIL API .NET

Creating Stimulus Signals
xMOD XIL API .NET supports the creation of signals and symbols of the stimulus domain of the XIL API.

Generating Stimulus Signals
xMOD XIL API .NET supports generating Stimulus Signals from sti and stz files. The SignalGenerator object is used to generate and play the stimulus signals. Below a code snippet on handing signals:

```
C#
    ISignalGenerator DemoSignalGenerator = null;
    ISignalGeneratorFactory MySignalGeneratorFactory = MyTestbench.SignalGeneratorFactory;
    bool bSTI = STIFile.EndsWith(".sti", StringComparison.InvariantCultureIgnoreCase);
    if (bSTI)
    {
        Console.WriteLine("Creating SignalGenerator from STI file...");
        ISignalGeneratorSTReader DemoSTReader = MySignalGeneratorFactory.CreateSignalGeneratorSTReaderByFileName(STIFile);
        DemoSignalGenerator = DemoSTAPort.CreateSignalGenerator();
        DemoSignalGenerator.LoadDemoSTReader(DemoSTReader);
        Console.WriteLine("...done!");
    }
    else
    {
        Console.WriteLine("Creating SignalGenerator from STZ file...");
        ISignalGeneratorSTReaderDemoSTReader = MySignalGeneratorFactory.CreateSignalGeneratorSTReaderByFileName(STIFile);
        DemoSignalGenerator = DemoSTAPort.CreateSignalGenerator();
        DemoSignalGenerator.LoadDemoSTReader(DemoSTReader);
        Console.WriteLine("...done!");
    }
```

```
Python
    load sti file and play it
    DemoSignalGenerator = DemoSTAPort.CreateSignalGenerator();
    STIFile = "C:\temp\SignalDescriptionSet_sample.sti"
    DemoSTReader = MyTestbench.SignalGeneratorFactory.CreateSignalGeneratorSTReaderByFileName(STIFile)
    DemoSignalGenerator.Load(DemoSTReader)
    DemoSignalGenerator.Start()
```

Figure 10 Code snippets: Generating Stimulus Signals

3.3 FMU for Real-Time

FMUs (Functional Mockup Units) are simulation components fulfilling the FMI Standard [5]. Running a FMU under a Real-Time operating system must fulfill all Real-Time requirements.

For Task 3.3.2 it is requested to run FMUs in real-time for xMOD. xMOD for real-time is based on the Real-Time Operating System RTX provided by IntervalZero. [6]

For this:
- xMOD 2019 will be enabled to run RTX-based FMUs
- Tooling for xMOD 2019 will be extended to create RTX-based FMUs

By standard an FMU may contain at least one of following points:
- sources
- binaries (dynamic library / shared object)
- Windows or Linux
- 32bit or 64 bit

RTX-based FMUs are not supported by any vendor exporting FMUs. Together with vendors tools and processes will be developed for exporting RTX-based FMUs.

3.3.1 Export of RTX based FMUs

Sources of a FMU (if available) are used to create binaries to run under RTX RTOS. FMUs for Modelexchange are not planned to be supported yet, because xMOD solvers are not designed for real-time.
The xMODFMUGenerator is a functionality in xMOD Editor and a separate tool provided with an installation of xMOD (from xMOD 2019). It creates FMUs containing sources to FMUs containing binaries to be executed under RTX RTOS additionally.

![FMU with sources](FMU.png) ▶️ RTXFMU Generator ▶️ FMU with sources and binaries for RTX

**Figure 11 xMODFMU Generator overview**

### 3.3.1.1 FMU requirements

**FMI standard version**

The original FMU needs to be compliant with FMI Standard 2.0.

<table>
<thead>
<tr>
<th>FMI Standard</th>
<th>Ability to be run in xMOD in Real-Time(RTX) mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>No</td>
</tr>
<tr>
<td>2.0</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Table 3: xMOD/FMI compatibility**

**FMU sources and binaries**

A FMU is a compressed file containing a standardized folder structure. Optionally it contains a folder “sources”. This folder contains the C sources and optionally a static library together, representing the functionality of the FMU. Depending on the tool exporting the FMU the C sources are complete or refer to external sources (e.g. FMI Standard Headers).

The FMU must contain the sources folder.

Ideally, the FMU contains additionally a dll-File located inside the folder binaries/win64. This will allow the simulation (that runs the FMU) to run in all possible modes (ASAP, Real-Time (Windows), Real-Time(RTX)).

If the FMU does not contain the dll-File then the generated xMODFMU will limit the simulation to only run in RealTime(RTX) mode.

### 3.3.1.2 xMOD FMU Generator Workflow

The user can add multiple FMUs (to be converted to xMODFMUs) to the table area. The table will show which FMUs can be converted. The location of the generated FMU can be specified for each FMU separately.

Once the list is defined the generation of the xMOD FMUs is triggered by clicking “Generate” button. The output of the building process is shown in text window. If logging is enabled a file with “msbuild.log” is generated into the working directory. The new FMUs will be stored in defined location and the FMU name is appended with “_xmod”.

The xMODFMUs can be used to run a simulation in xMOD in Real-Time(RTX) – mode.
Setup Verification
The xMOD FMU Generator is based on a toolchain containing a compiler, source code, RTX RTOS installation etc. This makes the setup process of the toolchain not an easy task. For that, we extended the FMU Generator tool with a verification utility to validate the setup of the complete toolchain.

To check the environment setup, you can select Setup in the xMOD FMUGenerator menu.

You can check the following:
- Compiler Installation
- RTX environment
- MSBuild environment

3.4 Supporting EtherCAT protocol for HiL applications
To use xMOD as a HiL system, a standard protocol to enable the connection of real-time simulations in xMOD with external Hardware (ECU for example) is mandatory. To be compatible with a large number of Hardware, it was decided to support a standard communication protocol that can cover most of the use cases in this project.

For this case, EtherCAT is selected to be supported by xMOD for HiL use cases. EtherCAT is a real-time Industrial Ethernet technology originally developed by Beckhoff Automation [7]. The EtherCAT protocol which is disclosed in the IEC standard IEC61158 is suitable for hard and soft real-time requirements in automation technology, in test and measurement and many other applications. A dedicated module named EtherCAT Master is developed in xMOD.

The goal of the EtherCAT master driver in xMOD is to use the EtherCAT technology (Ethernet for Control Automation Technology).

The role of the EtherCAT driver in xMOD is to allow the simulation designer to define the different EtherCAT input/output messages directly in the model. With this method, the model can receive directly EtherCAT message coming from the bus using an interface with the xMOD EtherCAT driver. The model can transmit directly EtherCAT message to the EtherCAT bus through the xMOD EtherCAT driver.
### 3.4.1 EtherCAT driver

A dedicated driver is developed to support the EtherCAT standard. This driver is based on a RTX master solution called KPA developed by koenig-pa. [8]

The global solution is described in the following figure:

**Figure 14 EtherCAT solution in xMOD**

The configuration and the setup of the EtherCAT bus is based on the KPA studio tool. Here all steps can be performed to configure all slave devices and setup the network parameters. A dedicated Plug-in is developed in KPA Studio to generate EtherCAT Network file (*.ecat) to be used in xMOD Master module.

**Figure 15 EtherCAT bus configuration between xMOD and KPA Studio**
To simplify the configuration of the EtherCAT network, a dedicated Plugin is developed. This Plugin allows to connect graphically xMOD models to the EtherCAT network inside the MIPS view in the Editor. A specific native block EtherCAT allows these operations by adding it to the MIPS by drag and drop.

![EtherCAT Native block in xMOD library](image)

Based on the user selection, one or two blocks will be displayed in the MIPS area. A first block will represent the EtherCAT Receive block (if some Inputs are selected). A second block will represent the EtherCAT Send Block (if some Outputs are selected).

The user can immediately connect these blocks to the Inputs and Outputs of other models available inside the MIPS.

![Configuration block to prepare EtherCAT Receive and Send blocks in the xMOD MIPS](image)

### 3.5 Supporting Simulink Protected models

Exchanging Simulink models from other partners (IVECO and TNO) was not possible. Indeed, exchanging the models without revealing its intellectual properties was a strong requirement. It was required from the xMOD tool chain for models compilation to support Simulink Protected models [9]. This was not planned at the beginning of the project. A protected model is a referenced model that does not support editing. The author of the protected model chooses whether to enable code generation of the model or not in this case.
The xMOD target is already supporting Model Reference. This is mandatory condition to support protected models. A dedicated xMOD target was developed to support the generation of XMODELS from protected models.
4 Summary

xMOD is the execution runtime for the AMDF toolchain. This means that xMOD have to answer any request coming from the Data management system (SYNECT) and from the Test Framework (TEVET). In this document, the xMOD extensions to build the Simulation Framework and to support the interactions with other tools are described.

The support of the ASAM Xil API Model Access is described in this document. Advanced functions are still not supported in xMOD (Watchers, records...etc.). A second version of xMOD will provide access to these advanced functions. For the current AMDF toolchain, we used the xMOD native functions to overcome this limitation.

The support of FMI for Real-Time is described in details. This function does not require the MATLAB/Simulink toolchain and is standalone. This gives the possibility to do real-time executions for the different vehicle architectures developed in this project.

The support of protected models was mandatory in this project to facilitate sharing Simulink models containing confidential information. The xMOD target is now capable to generate XMODELS from Simulink Protected models.

Finally, a dedicated protocol for communication in real-time was supported to provide the possibility to the WP partners to use xMOD in HiL context.
5 Risk Register

<table>
<thead>
<tr>
<th>Risk number</th>
<th>Description of Risk</th>
<th>Proposed Risk Mitigation Measure</th>
<th>Probability / effect¹</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

¹ Level of probability / effect: 1 = high, 2 = medium, 3 = low

5.1 References


6 Quality Assurance

The Steering Committee is the body for quality assurance. The procedure for review and approval of deliverables is described in the deliverable report D8.1 – “Project Handbook”. The quality will be ensured by checks and approvals of WP Leaders as part of the steering committee (see front pages of all deliverables).
7 Acknowledgment

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under Grant Agreement no. 769935.

Disclaimer
This document reflects the views of the author(s) and does not necessarily reflect the views or policy of the European Commission. Whilst efforts have been made to ensure the accuracy and completeness of this document, the HIFI-ELEMENTS consortium shall not be liable for any errors or omissions, however caused.