

© 2021-2023, Modelica Association and contributors.



This work is licensed under a <u>CC BY-SA 4.0 license</u>.

Modelica[®] is a registered trademark of the Modelica Association. eFMI[®] is a registered trademark of the Modelica Association. FMI[®] is a registered trademark of the Modelica Association. SSP[®] is a registered trademark of the Modelica Association. DCP[®] is a registered trademark of the Modelica Association.

Third party marks and brands are the property of their respective holders.



https://pixabay.com/illustrations/education-online-school-elearning-5307517/

© June 17, 2020 by ArtsyBee

I create these images with love and like to share them with you. My passion is to provide vintage designs to honor those artists that created something great and timeless. You are most welcome to use it for commercial projects, no need to ask for permission. I only ask that you not resell my images AS IS or claim them as your own creation. As always, a BIG thank you for the coffee donations I received, every dollar is a blessing for my family.

Education Online School royalty-free stock illustration. Free for use & download.

Content License Summary

Welcome to Pixabay! Pixabay is a vibrant community of authors, artists and creators sharing royalty-free images, video, audio and other media. We refer to this collectively as "Content". By accessing and using Content, or by contributing Content, you agree to comply with our Content License.

At Pixabay, we like to keep things as simple as possible. For this reason, we have created this short summary of our Content License which is available in full here. Please keep in mind that only the full Content License is legally binding.

What are you allowed to do with Content?

- Subject to the Prohibited Uses (see below), the Content License allows users to:
- Use Content for free
- Use Content without having to attribute the author (although giving credit is always appreciated by our community!)
- Modify or adapt Content into new works

What are you not allowed to do with Content?

We refer to these as Prohibited Uses which include:

- You cannot sell or distribute Content (either in digital or physical form) on a Standalone basis. Standalone means where no creative effort has been applied to the Content and it remains in substantially the same form as it exists on our website.
- If Content contains any recognisable trademarks, logos or brands, you cannot use that Content for commercial purposes in relation to goods and services. In particular, you cannot print that Content on merchandise or other physical products for sale.
- You cannot use Content in any immoral or illegal way, especially Content which features recognisable people.
- You cannot use Content in a misleading or deceptive way.
- Please be aware that certain Content may be subject to additional intellectual property rights (such as copyrights, trademarks, design rights), moral rights, proprietary rights, property rights or similar. It is your responsibility to check whether you require the consent of a third party or a license to use Content.





eFMI® Tutorial – Agenda

Part 1: eFMI® motivation and overview (40 min)

Part 2: Running use-case introduction (10 min)

Part 3: Hands-on demonstration in Dymola and

Software Production Engineering (former name CATIA ESP) (25 min)

Coffee break (30 min)

Part 3: Hands-on demonstration in Dymola and Software Production Engineering (former name CATIA ESP) (35 min)

Part 4: Live demonstration in TargetLink (30 min)

Part 5: Short presentation of further tooling (5 min)

Part 6: Conclusion (5 min)



Tutorial leader: Christoff Bürger



Presenter: Oliver Lenord BOSCH Invented for life



Presenter: Jörg Niere

dSPACE





Part 3: Hands-on demonstration in Dymola and Software Production Enginnering

eFMI® Tutorial – 15th International Modelica Conference – 9th of October 2023



Christoff Bürger Dassault Systèmes Christoff.Buerger@3ds.com

© 2023 Modelica Association | www.modelica.org



Tutorial requirements:

□ Own computer with Windows 10 or 11, 64-Bit, x86

You – i.e., every tutorial participant – should have gotten a software bundle with:

- □ This documentation (eFMI-Tutorial-Part-3.pdf in root directory)
- Preinstalled Dymola 2024x Beta 4 (/Dymola)
- □ Preinstalled Software Production Engineering (fomer name CATIA ESP) prototype (included in Dymola)
- □ Workdirectory where eFMUs will be generated and simulation artefacts stored (/work-directory)
- □ Modelica models we actually want to develop; for your reference if something goes wrong (/reference-models)
- □ eFMUs we actually want to build; for your reference if something goes wrong (/reference-eFMUs)
- □ Portable Microsoft Visual C++ and Microsoft Windows SDK required by Dymola (/portable-MSVC)
- Portable Java required by Software Production Engineering (/portable-Java)
- Portable Cppcheck (/portable-Cppcheck) and Python (/portable-Python) required for MISRA C:2012 compliance checks of production code
- □ Licenses of provided software (/licenses)





DISCLAIMER

The Microsoft Visual C++ and Microsoft Windows SDK provided in the /portable-MSVC directory are subject to licensing of Microsoft. The Java Development Kit (OpenJDK) provided in the /portable-Java directory is subject to licensing of the Free Software Foundation, Inc.

The Python provided in the /portable-Python directory is subject to licensing of the Python Software Foundation.

The Cppcheck provided in the /portable-Cppcheck directory is subject to licensing of Cppcheck Solutions AB.

The Dymola and Software Production Engineering provided in the /Dymola directory are subject to licensing of Dassault Systèmes.

The Python libraries and scripts pip, get-pip.py, argparse and Pygments are subject to their respective licensing.

BEFORE USING ANY OF ABOVE SOFTWARE, USERS MUST ACCEPT AND AGREE TO THEIR LICENSING (all licenses can be found in the /licenses directory).

THE DISTRIBUTED SOFTWARE IS PROVIDED ONLY FOR USAGE IN THE SCOPE OF THE "*eFMI® Tutorial*" OF THE "*15th International Modelica Conference*, 9-11 October 2023"; AND IT IS FOR PARTICIPANTS OF THE TUTORIAL ONLY.





Before getting started, please make sure you can use the provided Dymola:

File Graphics Documentation Text Simulation Image: Animation HTML Graph Image: Animation Image: Animation Proje Dymola License Setup Proje Dymola License Setup Checked out license options Setup	
Mode > > 2. License Setup General Details Borrow Setup Checked out license options A. Code export is checked ("Dymola Source")	ıd
Standard OK Enable code export options Image: Code Concertation License Code Concertation License Code Concertation License	e
Flexnet: "f875a4e7ac9c f875a4e7ac9c f8e4e33ffed5 f8e4e33ffed5 f8e4e33ffed9 00155d534cf6 00155de0490c 00155d5ff1e0" (a) The host id may change if the computer is connected to a docking station. In such a case, please copy all host ids. Copy to Clipboard OK Cancel Cancel	(-up



Before getting started, please make sure you can use the provided Dymola:

🗋 😭 🔚 ካ ሮ 💽 🔇	3 ₹ Dymola - Dynamic Modeling Laborator	у
File Graphics Du Image: Common strict in the strine strine strict in the strine strict in the strict in	ocumentation Text Simulation Tools inands Translate Simulate Stop 1 stop: stop: 1 1 1	 Check compilers are available for simulation: 1. Simulation ribbon 2. Setup button 3. Compiler tab 4. Verify Compiler button You can also pick any of the default Microsoft Visual Studio versions if you have a local installation. If not, please use the provided portable, on custom path << location of software bundle >> /portable-MSVC/
	Embedded server (Requires Visual Studio as C compiler)	Ŧ
	© None ○ DDE server	(
۲ filter variablec	Export DLL Export model as DLL with API	Etmi Interface for embedded systems





Ok, lets get started!







The user interface for eFMI support in Dymola is provided by means of a Modelica library: DymolaEmbedded

Load DymolaEmbedded via the *eFMI* button in the *Tools* ribbon \rightarrow *Load Libraries...* \rightarrow *OK*:

			Dymola - I	Dynamic Mo	odeling Lal	ooratory -	[Diagr	am]			—		\times
File Graphics Docu	imentation Text	Simulation	Tools								🗟 Win	dows 🔻	፹ — ፬ ×
Clipboard Image Anim	ation HTML Gra	oh Library Managem	License ent Setup	Options	₽ ₽ Update	å ⊕ Publish	追 Log	Status Refresh More ▼	? Help Documents	Dymola Website	About Dymola	eFMI	-G(s)+ Linear Analysis ▼
vraries	5 ×	Unnamed	× +										•
odel name	← → ▼ ↓□ ×	<u> </u>											
Dymola Commands	.DymolaEmbed	ded menu.load l	braries						×				
Favorites													
Modelica Reference	load_libraries												
Modelica	Description								-				
	f Load all e eFMI, eFM	FMI related Mode II_TestCases and	elica libraries s eFMI_TestCase	hipped with s_Embedde	Dymola, i dConfigur	.e., the Dy ations libr	molaEi aries.	mbedded,					
		ОК	Inf	o Co	opy Call	Execu	te	Close					
· · · · · · · · · · · · · · · · · · ·													
ibraries Projects													
		m Filter Variab	o Drouwoor - Ct	andy State	C		~						

Other menu entries permit to build or delete eFMUs for whole package hierarchies and load their co-simulation stubs (this convenience use-cases will become clear throughout the tutorial).



Functional Mock-up Interface for

embedded systems





The following libraries are loaded:

🗋 🎓 🔚 🔊 🍘 💽 🐼 🖛 🛛 Dymo	laEmbedded - DymolaEmbedded (Read-Only) - [Documentation]	- 🗆 X											
File Graphics Documentation	File Graphics Documentation Text Simulation Tools												
👫 🕶 🚺 Formatted 🗮	Cut 🗙 🏘 Normal 🗸 B x, A 🔻 🗉 🗐	達 E の Table マ											
Info Editor 😯 🍽	Copy $\stackrel{\text{\tiny black}}{=}$ (Default) \checkmark $I \times^2$ $\stackrel{\text{\tiny black}}{=}$	律 듣 🕰											
Info Source	🖹 Paste 🛝 🔪 (Auto) 🗸 🗉 😑												
Model Layer Language Clipboard Find Font Paragraph Tools													
Libraries 🗗 🗙 🗟 DymolaEmbedded 🗙 +													
Model name Modelica Reterence Modelica R													
Modelica Information													
Imodelica													
eFMI_TestCases A detailed documentation is given in the <u>user's guide</u> .													
DymolaEmbedded	DymolaEmbedded Important: Please note the library requirements.												
> 🚺 User's guide (overview, requireme.	Copyright												
> 🕨 Examples													
> 🔛 EmbeddedConfiguration													
> 🔀 BuildUtilities		11 T											
> 🚺 BuiltinFunctions	DHSSHU												
> f SupportFunctions	SYSTEM	ES											
> 🔀 SupportModels													
> i lcons	Copyright © 2016-2023, Dassault Systèmes												
EMI_TestCases_EmbeddedConfigu	This software component is a part of Dymola and subject to the Dy is regulated by the licensing conditions of Dymola; it comprises the	/ <u>mola Legal Notices;</u> its use e following parts:											
Libraries Projects	1. eFMI related software artefacts generation and embedded so facilities of <i>Dymola</i> (called <i>Dymola Embedded Code Generati</i>	urce code generation ion Facilities in the following)											
Libraries Projects Components Files [iagram Filter Variable Browser Steady State Sweep Parameters	» 🗕 🗧 🖬 🖻 🖆											

eFMI:

- Support library to ease adaptation of existing Modelica models for eFMI (mostly about MSL → eFMI table adapters)
- Public domain, © MA, MAP eFMI

eFMI_TestCases:

- eFMI application examples used for official cross-checks of eFMI tooling; Modelica tooling agnostic
- Public domain, © MA, MAP eFMI
- Contains our running use-case, M04

DymolaEmbedded:

- Interface for Dymola's eFMI facilities
- Provides means to configure eFMU generation & generate various eFMI containers

eFMI_TestCases_EmbeddedConfigurations:

- **eFMU generation configurations for** eFMI_TestCases
- Already contains a configuration for M04 (we will develop from scratch in the following)







Create a new eFMU generation configuration for the M04 controller:



Create package extending EmbeddedConfiguration:

- 1. *File* → *New* → *Package*, Name: MyM04eFMU
- 2. New package visible in *Package Browser & Projects* (not *Libraries*)
- 3. Double click MyM04eFMU; switch to Text ribbon
- 4. Add extends .DymolaEmbedded
 .EmbeddedConfiguration;
- 5. Switch to Graphics ribbon

Dymola and Software Production Engineering eFMU code generation can be configured from the diagram layer of MyM04eFMU.; it is an eFMU generation configuration.







Create a new eFMU generation configuration for the M04 controller:

= 😒 🔾 🖓 🖉 🔚 🔄	MyM04eFMU - MyM04eFM	IU - [Diagram] — 🗆 🛛
File Graphics Documenta	model_configuration in MyM04eFMU	
Recent ▼ Icon I ● Back □ Diagram ● Forward 100% ∨	General Add modifiers Attributes	s Icon
Model Layer	Name model_configuration	
Projects	Comment on. Must be configured b	y concrete embedded-configurations (sub-packages).
Model name 🔶 🔿 🏹 Unnamed	Model	Select Model ×
🗸 🔀 MyM04eFMU	Path DymolaEmbedded.BuildUtil	It Search:
> 🚺 extends DymolaEmbedde	Parameters	Packages
	not_yet_supported •••••	 PlantModels Controllers ControllerInterface SimpleNoFeedback Controller TestSetups
	model_name ""	Tests an eFMU.
Ibraries Projects	only_clocked false V	Configures, whether the given model is either, (1) a system model with clocked sub-partitions for which only code will be generated or (2) if code is generated for the whole model (i.e., not only its clocked sub-partitions). If 'true', clock independent model parts are ignored and will not be computed by the generated eFMU. Using this flag, whole system models can be

Configure Dymola's GALEC code generation:

- 1. Double click model configuration
 - \rightarrow model_name
 - \rightarrow *Edit* (package tree icon)
 - → **Select** eFMI_TestCases .M04_DrivetrainTorqueControl .Controllers.Controller
 - $\rightarrow OK$
 - $\rightarrow OK$
- 2. Double click code configuration → obfuscate: None
 - $\rightarrow OK$
- 3. Double click integrator configuration
 - → sample_period: 5e-4
 - \rightarrow *solver_method*: Explicit Euler
 - $\rightarrow OK$







Create a new eFMU generation configuration for the M04 controller:



Software Production Engineering is already default configured:

- 32-Bit and 64-Bit floating-point precision production codes
- 32-Bit and 64-Bit x86 ISA binary codes (self-contained static linked libraries)
- \Rightarrow 2 Production Code & 4 Binary Code containers







Investigate the eFMU generation configuration MyM04eFMU for the M04 controller:



All eFMU build activities are inherited from DymolaEmbedded.EmbeddedConfiguration:

- Available via the *extends* entry in the *Package Browser* & *Libraries* / *Projects* view (depending if configuration is write protected or not)
- Preconfigured with eFMU generation configuration
- Activities grouped according to eFMI container type:
 - Algorithm Code: Generate GALEC code
 - **Behavioral Model:** Derive experiment packages to configure test scenarios & tolerances; use experiment packages to generate respective Behavioral Models
 - Production Code: Generate & MISRA C:2012 check Software Production Engineering code
 - **Binary Code:** Generate Software Production Engineering binaries & Modelica proxies for cosimulating such; export FMU







Generate the eFMU configured in MyM04eFMU for the M04 controller:



Build the eFMU with Algorithm Code, 2x Production Code and 4x Binary Code containers:

- 1. Right click MyM04eFMU.build in the Package Browser / Projects view
 - \rightarrow Call Function...
 - $\rightarrow OK$
- 2. You can check the build log in the *Commands* window

Browse the generated eFMU:

- 1. Right click MyM04eFMU.browse_code in the Package Browser / Projects view
 - \rightarrow Call Function...
 - $\rightarrow OK$







Investigate the generated eFMU (MyM04eFMU/eFMU):



Contained containers:

Algorithm Code with GALEC code x64, 64-Bit floating-point precision Binary Code x86, 64-Bit floating-point precision Binary Code x64, 32-Bit floating-point precision Binary Code x86, 32-Bit floating-point precision Binary Code 64-Bit floating-point precision Production Code 32-Bit floating-point precision Production Code Content manifest listing all containers

Take some time to investigate the eFMU, e.g.:

- How cross references between manifests work
- Quality of generated GALEC code (self-contained / inlined, error handling of symbolic optimized linear equation systems, local vs. global variables etc)





Check the eFMU and its production codes:



Check MISRA C:2012 compliance of all production codes via Cppcheck:

1. Right click MyM04eFMU.ProductionCode
 .check_code in Package Browser / Projects view
 → Call Function...

 $\rightarrow OK$

2. Analyses reports for each production code are provided in your webbrowser (note, that block.c, the actual production code, satisfies MISRA)

Check eFMU with *eFMI Container Manager* and *eFMI Compliance Checker* (MAP eFMI released tools):

- 1. Right click MyM04eFMU.check_eFMU in the Package Browser / Projects view
 - \rightarrow Call Function...
 - $\rightarrow OK$







Congratulations, you are halfway through!



eFMU generation done.

Let's go on to Behavioral Models & software-in-the-loop (SiL) simulation.





For which target did we just generate binaries? How do I pick my embedded target?











Which kind of limitations on Modelica models exist? What is supported (signal buses, discrete, events, state machines, ...)?









Which kind of Modelica models / equation systems do not work? What about very stiff systems of equations?







What are the differences between Dymola & Software Production Engineering code configuration, Dymola C code generation, eFMI code generation and eFMU bundle configuration?









What is the *.alg file in the ACode_Dymola container of the eFMU?











Congratulations, you are halfway through!

See you in the second half of the hands-on after the coffee break!



eFMU generation done.

Let's go on to Behavioral Models & software-in-the-loop (SiL) simulation.







Congratulations, you are halfway through!

Welcome back to the second half of the hands-on!



eFMU generation done.

Let's go on to Behavioral Models & software-in-the-loop (SiL) simulation.







Generate eFMU co-simulation stub:

L) 🖏 💾 🔊	ج 😒 🕒 🗠	-	MyM	04eFMI	U - MyM04eFMU - [D	ocumentation]			\times
File Grap	ohics Docu	umentation	Text Sim	nulatior	n Tools			🔁 Windows	▼ ↑ -	- @ ×
<table-of-contents> Recent 🔹</table-of-contents>	i Format	MyM04e	MU.BinaryCo	de.buil	ld_binary_stub					
🗣 Back	📑 Info Ed									
Forward	🔛 Info So	build_bir	hary_stub		build_binary_stub	Tree data				
Model	Lay		production_co	des	Description					
Package Browse	er	> 🌇 b	oinary_codes		Generate and	l/or load a eFN	/U co-simulation	stub wrapping	the eFML	J
Model name					ordinary Mod	odes generate delica compon	d by CATIA ESP : ient.	for SiL simulatio	n as	
MyM04e	FMU									
✓ ★ extends	5 DymolaEml				Inputs					
> 🚫 Clocke	ed				build_binary_stub	true 🗠 🕨	Configures, if	the eFMU co-sir	nulation s	stub is <u>c</u>
💢 check	_configuratic									
🛃 build					load binary stub	true × •	Configures, if the eFMU is ch	the eFMU co-sir beckd for any ex	nulation s	tub is le
🧌 clean					loud_bindry_stab	dde	'build_result.su	ucceeded = false	i'.	b Which
🔭 brows	se_code									
风 check	_eFMU				Outputs					
> 📄 Algori	ithmCode				build_result			Þ	Build suc	cess rej
> 🔀 Behav	ioralModel									
> 🖹 Produ	ictionCode									
🗸 🚺 Binary	Code									
💢 chec	k_configurat							ОК		Info
🛃 build	d L		= =	le				OK		
🔛 build	d_binary_stub	~	Dymol	laEmbe	edded_menu.load_	libraries();			~
			CO							
Libraries Pack	age Browser	Component	Browser File	es Dia	gram Filter Variable E	Browser Stea	dy State	» 🖶) [[8

1. Right click
MyM04eFMU.BinaryCode.build_binary_stub
in Package Browser / Projects view

→ Call Function...
→ OK

A new package 'MyM04eFMU.eFMU_SiL_Support' is generated. Its BinaryStub model is a Modelica proxy to the static linked libraries – and therefore production codes – generated by Software Production Engineering.







Investigate generated eFMU co-simulation stub:

🗋 🎲 🔚 🔊 🍘 🕟 🐼 🗧 🛛 TuningBus - 'MyM04eFMU.eFMU_SiL_Support'.TuningBus - [Documentation] — 🗌 🛛 🗡													\times					
File Grap	phics Documer	ntatio	n 1	Text Simulation Tools						🔁 Windows 🔻 📧 -								
Recent 🔻	i Formatted	rev			ж X	Abr	Normal	\sim	в	X ₂	A	•			Ξ	ග	Table	-
Back	Info Editor		M+			ab ⊯ac	(Default)	\sim	I	×2	۲	•	E		1223	-		
Forward	Info Source	==				Ð	(Auto)	~	U			11				e=		
Model	Laver		Langu	Jage	Clipboard	Find		Font					Para	agraph			Tools	
Projects		5	×	ου Τι	uningBus	×	+							5				-
Model name	$\leftrightarrow \Rightarrow \gamma$	r Ļo	* F	Typar	ndable con	nector	type for prov	isioni	ina r	untir	me v	alue		t for n	ecalih	oratio	n	*
Unname	d			-Apai			type for prot		g			aiuo			oouni	natio		- 1
			1	nfor	mation													- 1
Y MyM04e	PFMU																	- 1
> 🔀 extends	s DymolaEmbed		_															- 1
~ 🔲 'MyM04e	eFMU.eFMU_SiL		E	Exten	ds from <u>.Dy</u>	/molaEr	mbedded.lcon	s.Tuni	ng <u>Bu</u>	<u>s</u> (Ic	on f	or tur	ing bu	ises ei	nablin	g reca	alibration	, for
example the tuning buses provided by eFMU co-simulation stubs.), <u>Tuning</u> (Support record to compiling all tuneable parameters, each typed with a fitting builtin Modelica type.)											record ty	/be						
Rinan/S	tub		0	compi	iling all tune	eable pa	arameters, ead	:h type	ed wi	h a t	fittin	g bui	tin Mo	delica	type.)			
BinaryS	itub		c	compi	iling all tune	eable pa	arameters, eac	h type	ed wi	th a f	fittin	g bui	tin Mo	delica	type.)).	,	100
Binarys	itub Configuration		c (compi Conte	iling all tune ents	eable pa	arameters, eac	h type	ed wi	th a f	fittin	g bui	tin Mo	delica	type.)).		,po
BinaryS 👾 Tuning(itub Configuration Bus		c (compi	iling all tune ents Type	eable pa	Name	h type	ed wi	th a f	fittin	g bui	tin Mo Desc	delica riptior	type.) 1).		
BinaryS	itub Configuration Bus BusSampler		c (iling all tune ents Type	eable pa	Name	h type Mor	nent	th a f	fittin ertia	g buil	Desc n2]	delica riptior	type.) ו). 		
BinaryS	itub Configuration Bus BusSampler			compi Conte Inertia Real	iling all tune ents Type a	eable pa	Name J_M Ni_PI	Mor Ni*1	nent	of in me c	fittin ertia	g buil I [kg.i	Desc n2] of anti-	delica riptior windu	type.) n p com	ipensa	ation	
BinaryS Constant Constan	itub Configuration Bus BusSampler tionCodes		(Conte Conte Inertia Real Real Real	iling all tune ents Type a	eable pa	Name J_M Ni_PI Ti_PI C res	h type Mor Ni*1 Tim	ment Fi is ti e cor	of in me o istan	fittin ertia cons nt of	g buil (kg.i tant o Integ	Desc n2] of anti- rator b	riptior windu lock	type.) n p com	ipensa	ation	
BinaryS Constraints Tuningt Constraints Froduct Fresolve	itub Configuration Bus BusSampler tionCodes _code_configurati.		(Conte Conte Inertia Real Real Real	iling all tune ents Type	eable pa	Name J_M Ni_PI Ti_PI c_res	h type Mor Ni*1 Tim Res ~1/(ment Fi is ti culting	of in me o istan g stiff mu)	ertia cons nt of fnes +1/c	g buil (kg.i tant o Integ for ro	Desc n2] of anti- rator b pad an	riptior windu lock d pow	type.) n p com ertrair	ipensa	ation	
BinaryS Tuningt Tuningt Tuningt Product resolve	itub Configuration Bus BusSampler tionCodes _code_configurati. . interface			compi Conte Inertia Real Real Real Real	iling all tune ents Type a ionalDampi	eable pa	Name J_M Ni_PI Ti_PI c_res	h type Mor Ni*1 Tim Res ~1/(Dan	ment Fi is ti culting (1/c_ nping	of in me o istan g stiff mu) con	fittin ertia cons nt of fnes +1/c	g buil (kg.i tant o Integ for ro) nt [N.i	Desc n2] of anti- rator b oad an n.s/rac	riptior windu lock d pow	type.) n p com ertrair	ipensa	ation	
BinaryS Tuningt Tuningt Tuningt Product resolve resolve	itub Configuration Bus BusSampler tionCodes _code_configurati. _interface			Conte Conte Real Real Real Real Real Real	iling all tune ents Type a ionalDampi	ingCons	Name J_M Ni_PI Ti_PI c_res stant d_res f_cut	h type Mor Ni*1 Tim Res ~1/(Dan Cut	ment Ti is ti cor culting (1/c_ nping -off fr	of in me o nstan g stiff mu) i con eque	fittin ertia cons nt of fnes +1/c star ency	g buil a [kg.i stant o Integ for ro) nt [N.i	Desc n2] of anti- rator b oad an n.s/rac	riptior windu lock d pow	type.) n p com ertrair	ipensa	ation	
BinaryS Tuningt Tuningt Tuningt Product resolve resolve	itub Configuration BusSampler tionCodes _code_configurati. _interface			Conte Inertia Real Real Real Real Real Real Real	iling all tune ents Type a ionalDampi	ingCons	Arameters, eac J_M J_M Ni_PI Ti_PI c_res stant d_res f_cut gearRatio	Mor Ni*1 Tim Res ~1/(Dan Cut o Gea	ment Fi is ti culting (1/c_ nping -off fr ar rati	of in me c istan y stiff mu) i con eque	fittin ertia cons ht of fnes +1/c astar ency	g buil a [kg.i atant o Integ for ro) at [N.i	Desc n2] of anti- rator b oad an n.s/rac	riptior windu lock d pow	type.) p com ertrair	ipensa	ation	
BinaryS Tuningt Tuningt Tuningt Product resolve resolve	itub Configuration BusSampler tionCodes _code_configurati. _interface			Conte Inertia Real Real Real Real Real Real Real	iling all tune ents Type a ionalDampi	ingCons	Arameters, eac J_M J_M Ni_PI Ti_PI c_res stant d_res f_cut gearRatio k_PI	Mor Ni*1 Tim Res ~1/(Dan Cut 0 Gea	ment Fi is ti cor sulting (1/c_ nping -off fr ar ration	of in me (astan y stiff mu); (con eque o contre	fittin ertia cons nt of fnes +1/c astar ency oller	g buil (kg.i tant o Integ for ro) nt [N.i	Desc n2] of anti- rator b bad an n.s/rac	delica riptior windu lock d pow	type.) p com ertrair	ipensa	ation	
BinaryS Tuning(Tuning(Tuning(Product resolve resolve	itub Configuration BusSampler tionCodes _code_configurati. _interface			Conte Inertia Real Real Real Real Real Real Real Rea	iling all tune ents Type a ionalDampi	ingCons	Name J_M Ni_PI Ti_PI c_res f_cut gearRatio k_PI k_accCo	h type Mor Ni*1 Tim Res ~1/(Dan Cut- Dan Cut- Dan Gain r Gain	ment i is ti e cor ulting ((1/c_ nping -off fr nr rati n of c n vali	of in me c istan g stiff mu)- i con eque o contru ue m	fittin ertia cons nt of fnes +1/c astar ency oller nultip	g buil [kg.1 itant (Integ for ro) it [N.1 itage ()	tin Mo Desc n2] of anti- rator b oad an n.s/rac	riptior windu lock d pow 1]	n p com ertrair	ipensa	ation	
BinaryS Tuning(Tuning(Tuning(Product resolve, resolve,	itub Configuration BusSampler tionCodes _code_configurati. _interface			Compi Conte Real Real Real Real Real Real Real Rea	iling all tune ents Type a ionalDampi	ingCons	Arameters, eac J_M J_M Ni_PI Ti_PI c_res f_cut gearRatio k_PI k_accCoo tauM_ma	h type Mor Ni*1 Tim Res ~1/(Dan Cut 0 Gea Gain r Gain x Max	ment Ti is ti e corr sulting ((1/c_ nping -off fr n r rati n of c n vali c mot	of in me c stan g stiff mu) i con eque o o ontri ue m or to	fittin ertia cons nt of fnes +1/c istar ency oller nultip orque	g buil I [kg.i tant (Integ for r() nt [N.i lied v	Desc n2] of anti- rator b oad an n.s/rac	riptior windu lock d pow d]	n p com ertrair nal	ipensa	ation	
BinaryS Tuningt Tuningt Tuningt Product resolve resolve	itub Configuration Bus BusSampler tionCodes _code_configurati. _interface			Compi Conto Inertize Real Real Real Real Real Real Real Rea	ionalDampi	ingCons	Name J_M Ni_PI Ti_PI c_res stant f_cut gearRation k_PI k_accCool tauM_ma	Mor Ni*T Tim Res ~1/(Dan Cut. Dan Cut. Gai Gai Gai	ment is ti e cor sulting ((1/c_ nping -off fr ar rati n of c n vali	of in me c astan g stiff mu)- i con eque o contru- ue m or to	fittin ertia cons ht of fnes +1/c istar ency oller nultip prque	g buil [kg.1] ttant (Integ for ru) nt [N.1]	Desc n2] of anti- rator b bad an n.s/rac	delica riptior windu lock d pow 네]	n p com ertrair). Ipensa	ation	
BinaryS	itub Configuration Bus BusSampler tionCodes _code_configurati. _interface			Compi Conto Inertia Real Real Real Real Real Real Real Rea	iling all tune ents Type a ionalDampi :: TuningBu 'MyM04eFt	ingCons s MU.eFN	Arameters, eac J_M J_M Ni_PI Ti_PI c_res f_cut gearRatic k_PI k_accCo tauM_ma	Mor Ni*T Tim Res ~1/(Dan Cut- Cut- Cut- Gai Gai Cut- Cut- Cut- Cut- Cut- Cut- Cut- Cut-	ment Fi is ti e cor ulting (1/c_ nping -off fr n of c n valu c mot ningE	of in me c istan g stiff mu) contro c	fittin ertia cons nt of fnes +1/c star ency oller nultip	g buil (kg.i ttant (Integ for ro) nt [N.i	Desc n2] of anti- rator b bad an n.s/rac	riptior windu lock d pow 1]	n p com ertrair	npensa	ation	
BinaryS	itub Configuration BusSampler tionCodes _code_configurati. _interface			Inertia Real Real Real Real Real Real Real Rea	ionalDampi ents ionalDampi :: TuningBu 'MyM04eFt	ingCons s MU.eFM	Name J_M Ni_PI Ti_PI c_res f_cut gearRatio k_PI k_accCoi tauM_ma	Mor Ni*T Tim Ress ~1/(Dan Cut- Dan Cut- Dan Cut- S Gain Cut- S Gain X Max	ment is ti e cor ulting (1/c_ nping -off fr ar rati n of c mot	of in me c istan g stiff mu); i con eque o contru ue m or to Bus	fittin ertia cons nt of fines +1/c nstar ency oller nultip rque	g buil I [kg.1 ttant (Integ for ro) it [N.1 	Desc n2] of anti- ator b vad an n.s/rad	ription windu, lock d pow J] 	n p com ertrair	ipensa 1	ation	

Main characteristics of eFMU co-simulation stubs:

- Support multiple instantiation (each is atomic)
- All production codes available (32-Bit & 64-Bit floating-point precision simulation)
- Support modification, input-dependent initialization, recalibration & reinitialization
- Provide & assert eFMI error signals
- Preserve original model interface (dimensionalities, diagramatic layout of in- & output connectors etc)
- Provide sampling with period of generated eFMU
- "Just" a production code proxy (no additional equations; no solver required; "simply" implement GALEC block live-cycle)





Derive experiment package to define test scenarios & generate Behavioral Model container:

- 😒 🔾 🔊 🖉 🔜 =	build_tests - MyM04eFMU.BehavioralModel.build_tests (Read-Only) - [Documentation] —	
File Graphics Docun	MyM04eFMU.BehavioralModel.build_tests ×	x - 0
Recent Formattee Back Forward	build_tests Description	Table 🔻
Model Layer	Derive from an existing model an experiment-package defining the test scenarios of a Dymola-administered Behavioral Model container. Each eBlock	pols
Model name Image: Comparison of the second sec	defines one test scenario of the Behavioral Model container. The generated package can be used to add the respective Behavioral Model container to the eFMU (i.e., generate reference results and manifest) and to software-in-the-loop (SiL) test the CATIA ESP production codes of the eFMU.	f a ∋nario of pective ∋st) and
 extends DymolaEmbec Clocked check_configuration 	Inputs The model Select Model X source_experiment Image: Select Model Image: Select Model X replace false v Configures, the experiment Search: Image: Select Model replace false v Image: Select Model Image: Select Model	
♥ build │ clean ■ browse_code	Outputs Build success reprint information. Build success reprint information. Controllers Controlers	lodelica
	OK Info experi > experi > ReferenceTests >	e
	Outputs Controller_ExplEuler Type Name BuildResult build_result Build tests OK	
Image: Second	Path: DymolaEmbedded.EmbeddedConfiguration.comanoramioaer.coma_tests Filename: C:/Users/CBR5.DSONE/Desktop/eFMI- omponent Browser Files Diagram Filter Variable Browser Steady State *	1 2

Derive experiment package from existing closed loop experiment:

- 1. Right click MyM04eFMU.BehavioralModel
 - .build_tests in Package Browser / Projects view
 - \rightarrow Call Function...
 - \rightarrow source_experiment
 - \rightarrow *Edit* (package tree icon)
 - \rightarrow select eFMI_TestCases
 - .M04_DrivetrainTorqueControl
 - .ReferenceTests
 - .Controller_ExplEuler_ClosedLoop
 - $\rightarrow OK$ $\rightarrow OK$





Scenario 1: controller

Functional Mock-up Interface for

embedded systems

Investigate the derived experiment package:







Define tolerances for the test scenarios of the experiment package:



Define absolute and relative tolerances for all floatingpoint precisions and test scenarios (i.e., SiL tests). We can use a default for all scenarios (here only a single): 1. Double click tolerances_default (labeld default) in Diagram view of the experiment package → set tolerances for M_motor output a follows absolute_x32 (M_motor=1e-3) relative_x32 (M_motor=1e-4) absolute_x64 (M_motor=1e-6) relative_x64 (M_motor=1e-8) → OK





Generate Behavioral Model container form the experiment package:

🗋 😭 🔚 🄄 🍘 🖉 🕑 🗞 🗟 build - 'MyM04eFMU:eFMI_TestCases.M04_DrivetrainTorqueControl.ReferenceTests.Contr 🦳 🗌														×
File Grap	hics Docume	ntatior	n Text	Simulatio	on	Tools				ę	🗄 Win	dows	▼ 〒 -	- @ X
Recent 💌	i Formatted	rev	>	$\times \times$	٨bf	Normal 🗸 🗸	B ×	•	EI	÷	Ξ	ග	Table	-
🔶 Back	🗐 Info Editor	<>> rev	M+		ab ⊯ac	(Default)	I x	۵. 🔻	≣		1200	-		
Forward 🕨	🔛 Info Source	==		<u>لهم الم</u>	Ð	(Auto) 🗸	U		=			e=		
Model	Layer		Language	Clipboard	Find	Fon	t		Pa	ragrapł	n		Tools	
Projects 🗗 🗙 💌 build 🗙 +												•		
Model name 🔶 🔿 🖓 🕼 🛪 Build the Behavioral Model container administered by this experiment package.														
Unnamed	ł		Infor	mation										
∽ 🔛 MyM04e	FMU												٦	
> 🔀 extends	DymolaEmbed			'MyM04eF	MU:eF	MI_TestCases.M04_D	rivetrai	nTorqueC	Control.R	eferenc	eTes	\times		
> 🚺 'MyM04e	FMU.eFMU_SiL		E	build), or parts	
✓ MyM04€	FMU:eFMI_TestC		tr	Description										
	ration = M_VM04		c		41 D	- hand - Mandal - and								
Je build				pack	age.	enavioral Model con	ainer a	aminister	ed by tr	iis expe	riment			
		_	Е	Outputs										
Diowse			S											
✓ Vest_Ref	terenceExperiment	t	N	succeeded	ł	, w	/hether	the build	l succeed	ded or f	tailed.			
Test_SiL	_Scenario_1		P 'N	error_mes	sage	, FI	ror me	ssage in (case the	build fa	ailed.		er_ExplE	uler_C
			F										- ·	
			Ĭ	C	Ж	Info	Сору С	all	Execute		Close			
Libraries P	rojects		4											Þ
Libraries Proje	ects Component	s File	es Diagra	m Filter Va	riable	Browser Steady Sta	te Sw	eep Parar	meters	**	•			8

Build the Behavioral Model container with reference results taken from simulation of the reference experiment Test ReferenceExperiment:

 Right click build of experiment package in Package Browser / Projects view
 → Call Function...

 $\rightarrow OK$

Browse the generated Behavioral Model container:

 Right click browse_container of experiment package in the Package Browser / Projects view → Call Function...

 $\rightarrow OK$







Investigate the generated Behavioral Model container (BModel_Dymola_699250432):



Container content:

XML manifest with

- Test scenarios
- Links to Algorithm Code manifest for variable names and types (in-, output, tuneable parameter) & sample period
- Variables → CSV column name links (multi-dimensions are flattened to individual columns)
- Tolerances for various floating-point precisions
 Reference trajectories in comma separated values (CSV)
 files (one file per test scenario)

Take some time to investigate the manifest and CSV file.





Conduct SiL test of Software Production Engineering generated production codes:



- Double click Test_SiL_Scenario_1 of the experiment package in Package Browser / Projects view
- 2. Switch to Simulation ribbon
 - \rightarrow Click Simulate button
- 3. Right click 'M_motor|match' in diagram plot → Plot Variable
 - → select act (actual SiL simulation trajectory)
 - \rightarrow select *ref* (expected reference trajectory)
- 4. Zoom into *Plot* window to see there are differences

Note, that the test did not fail (see *Logs* window & dashboards). If you tighten tolerances – e.g., change the 32-Bit floating-point precision tolerances to the 64-Bit ones – it will fail.





Conduct SiL test of Software Production Engineering generated production codes:



- Double click Test_SiL_Scenario_1 of the experiment package in Package Browser / Projects view
- 2. Switch to Simulation ribbon
 - \rightarrow Click Simulate button
- 3. Right click 'M_motor|match' in diagram plot
 - \rightarrow Plot Variable
 - → select act (actual SiL simulation trajectory)
 - \rightarrow select *ref* (expected reference trajectory)
- 4. Zoom into *Plot* window to see there are differences

Note, that the test did not fail (see *Logs* window & dashboards). If you tighten tolerances – e.g., change the 32-Bit floating-point precision tolerances to the 64-Bit ones – it will fail.







Congratulations, you did it!



Let's do some advanced SiL stuff, like recalibration and reinitialization.



© 2023 Modelica Association | www.modelica.org | CC BY-SA 4.0



Load prepared recalibration & reinitialization example for M04 controller:



1. Either, drag and drop model reference-models/Part-3/ RecalibrateAndReinitializeTest.mo in Package Browser / Projects view or load it via File → Open → Load...

The model has 4x M04 controller instances (eFMU cosimulation stub instances):

- 1. untuned: not modified, recalibrated nor reinitialized
- 2. parameterized: modified c_res & k_PI
 parameters, but not recalibrated nor reinitialized
- 3. tuned: unmodified, but via tuningBus runtime recalibrated c_res & k_PI parameters
- 4. tuned_and_reinitialized: like 3, but additionally at runtime reinitialized







All 4 controllers use the same production code for simulation (__defining_code modification set by the global record parameter in the upper left of the diagram).

The c_res & k_PI parameter changes are all switches from the default value to the same new value, just at different time points (as modification before simulation or as recalibration during simulation):

- c_res: 4710 → 2710 at t = 0s or 0.25s (step runtime value)
- k_PI: $-73 \rightarrow -10$ at t = 0s or 0.6s (step1 runtime value)

Reinitalization is done at t = 0.7005s (booleanTable runtime value).







Tuning is enabled by modifying co-simulation subs:

- ___enable_tuning = true
- selecting/activating the tuned parameters via _____tuning_configuration
- \Rightarrow The tuning bus connector (:::) is enabled.

New recalibration parameter values are provided as runtime values connected to the tuning bus. Only tuning-activated parameters have to be provisioned.

Tuning configuration & bus types are provided in the generated eFMU co-simulation stub (drag and drop).

In this model: Tuneable parameters are selected by the global __tuning_configuration record parameter in the upper left

of the diagram.







Reinitialization is enabled by modifying eFMI cosimulation subs:

- __enable_reinitialization = true
- \Rightarrow The "stop push button" (\bigcirc) is enabled.

New reinitialization requests are provided as runtime values connected to the "stop push button". Such are locked until the next sampling; it is sufficient to signal at any point inbetween two samplings that a reinitialization is requested – it is not necessary to ensure reinitialize == true exactly at the sampling.







- 1. Simulate RecalibrateAndReinitializeTest
- 2. Plot M_motor of all 4 co-simulation stubs
- 3. Plot recalibrated (true, iff recalibration done)
- 4. Zoom into the plot at $0.0 \le t \le 1.05$

When do parameterized and tuned plots align? When does untuned align? Is the controller fast adapting in case of errors that require a system restart?

Good to remember:

- All controllers use same production code
- c_res & k_PI parameters change consistently:
 - c_res at t = 0s or 0.25s (step)
 - k PI at t = 0s or 0.6s (step1)
- Reinitalization at t = 0.7005s (booleanTable)







- 1. Simulate RecalibrateAndReinitializeTest
- 2. Plot M_motor of all 4 co-simulation stubs
- 3. Plot _____recalibrated (true, iff recalibration done)
- 4. Zoom into the plot at $0.0 \le t \le 1.05$

When do parameterized and tuned plots align? When does untuned align? Is the controller fast adapting in case of errors that require a system restart?

Good to remember:

- All controllers use same production code
- c_res & k_PI parameters change consistently:
 - c_res at t = 0s or 0.25s (step)
 - k_PI at t = 0s or 0.6s (step1)
- Reinitalization at t = 0.7005s (booleanTable)





Final touch – export eFMU as FMU:



- 1. Right click MyM04eFMU.BinaryCode.build_FMU in Package Browser / Projects view
 - \rightarrow Call Function...
 - $\rightarrow OK$

The exported FMU has all conditional parameters of the eFMU co-simulation stub fixed to their defaults:

- Recalibration & reinitialization: disabled, i.e., _____enable_tuning = false, _____enable_reinitialization = false
- Error signals: asserted, i.e.,

_assert_error_signals = true

 Internal sampling: embedded & fixed, i.e., embedd clock = true







Congratulations, you did it like a PRO!









Assume my embedded target platform provides functionality I like to reuse. How do I link it to my GALEC / production code?

How can I interface existing C code / binaries in my controller?









What is the minimal setup I need, starting from Dymola? Which eFMU containers are optional? Which eFMI features are optional?







We used a lot of Modelica

libraries.

What are all the eFMI libraries loaded in Dymola good for?



ካ 🦳 🕞 🐼 ፡

Graphics
Documentation
Text
Simulation
Tools

Formatted
Image: Comparison of example of ex

 Support library to ease adaptation of existing Modelica models for eFMI (mostly about MSL → eFMI table adapters)

eFMI TestCases:

DymolaEmbedded - DymolaEmbedded (Read-Only) - [Documentation]

 eFMI application examples used for official cross-checks of eFMI tooling; Modelica tooling agnostic

DymolaEmbedded:

- Interface for Dymola's eFMI facilities
- eFMI_TestCases_EmbeddedConfigurations:
- **eFMU generation configurations for** eFMI_TestCases









Congratulations, you did it like a PRO!









eFMI® Tutorial – Agenda

Part 1: eFMI® motivation and overview (40 min)

Part 2: Running use-case introduction (10 min)

Part 3: Hands-on demonstration in Dymola and

Software Production Engineering (former name CATIA ESP) (25 min)

Coffee break (30 min)

Part 3: Hands-on demonstration in Dymola and Software Production Engineering (former name CATIA ESP) (35 min)

Part 4: Live demonstration in TargetLink (30 min)

Part 5: Short presentation of further tooling (5 min)

Part 6: Conclusion (5 min)



Tutorial leader: Christoff Bürger



Presenter: Oliver Lenord BOSCH Invented for life



Presenter: Jörg Niere

dSPACE

