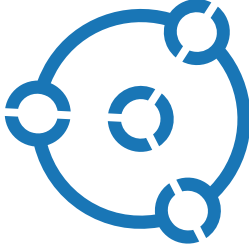


<b>D3.6</b> (M36)	<b>Efficient, traceable, and flexibly adaptable C/C++ code generation for embedded targets from OpenModelica (M36 prototype)</b>
Access <sup>1</sup> :	PU
Type <sup>2</sup> :	Prototype
Version:	1.0
Due Dates <sup>3</sup> :	M24, M36
 <b>openCPS</b> <i>Open Cyber-Physical System Model-Driven Certified Development</i>	
<b>Executive summary<sup>4</sup>:</b>	
<p>This report accompanies the first incremental development prototype of an efficient, traceable, and flexibly adaptable C code generator for embedded targets from OpenModelica. The prototype is part of the OpenModelica v1.12.0 distribution released on October 31, 2017, which can be downloaded from the OpenModelica website (<a href="https://www.openmodelica.org/">https://www.openmodelica.org/</a>). The embedded code generation target for OpenModelica is in an experimental state and can be activated by passing the option <code>--simCodeTarget=ExperimentalEmbeddedC</code> to the OpenModelica Compiler (OMC). This report presents the state of the M36 prototype.</p>	

<sup>1</sup> Access classification as per definitions in PCA; PU = Public, CO = Confidential. Access classification per deliverable stated in FPP.

<sup>2</sup> Deliverable type according to FPP, note that all non-report deliverables must be accompanied by a deliverable report.

<sup>3</sup> Due month(s) according to FPP.

<sup>4</sup> It is mandatory to provide an executive summary for each deliverable.

**Deliverable Contributors:**

	Name	Organisation	Primary role in project	Main Author(s) <sup>5</sup>
Deliverable Leader <sup>6</sup>	Lena Buffoni	SICSEast	WP3 Leader	
Contributing Author(s) <sup>7</sup>	Bernhard Thiele	SICSEast	Main Author	X
	Martin Sjölund	LIU	WP4 Leader	
Internal Reviewer(s) <sup>8</sup>	Per Sahlin	EQUA	T3.3 Leader	

**Document History:**

Version	Date	Reason for change	Status <sup>9</sup>
0.1	07/11/2017	First draft	Draft
0.2	07/11/2017	Internal review version completed	In Review
1.0	13/11/2017	Integrated comments by reviewer	Released
1.1	30/11/2018	Minor update	Released

<sup>5</sup>Indicate Main Author(s) with an "X" in this column.

<sup>6</sup>Deliverable leader according to FPP, role definition in PCA.

<sup>7</sup>Person(s) from contributing partners for the deliverable, expected contributing partners stated in FPP.

<sup>8</sup>Typically person(s) with appropriate expertise to assess deliverable structure and quality.

<sup>9</sup>Status = "Draft", "In Review", "Released".

## Contents

<b>Acronyms</b>	<b>4</b>
<b>1 Introduction</b>	<b>5</b>
<b>2 Features available in the prototype</b>	<b>5</b>
<b>3 Implementation</b>	<b>6</b>
<b>4 Usage</b>	<b>7</b>
<b>5 Conclusion</b>	<b>7</b>
<b>References</b>	<b>9</b>

## Acronyms

FMI Functional Mock-up Interface.

MDD Modelica\_DeviceDrivers.

MSL Modelica Standard Library.

OMC OpenModelica Compiler.

## 1 Introduction

This report accompanies the first incremental development prototype of an efficient, traceable, and flexibly adaptable C code generator for embedded targets from OpenModelica. The prototype is based on previous conceptual work within WP3 [TB16; TS16]. First experiments in using the embedded code generator for targeting severely restricted targets (in terms of computation power and memory) have been reported in [Thi+17]. A paper related to the flexible adaption of the code generation to different target systems has been published as a work-in-progress paper [BST17].

The Modelica Language supports discrete-time and continuous-time models. While the continuous-time part allows modeling of the physical behavior via differential and algebraic equations, the discrete-time part allows modeling of digital control systems based on the synchronous data-flow paradigm, difference equations, state machines, and digital support logic. Since digital control applications play a major role in embedded systems, efforts have been started within WP3 to extend the OpenModelica Compiler (OMC) with support for generating low-footprint code for embedded targets directly from Modelica models. The basic approach consists of generating target-agnostic ANSI C code by the OMC and *injecting* target specific functionality by suitable Modelica libraries. The goal for the final prototype is supporting code generation from hybrid models in which the continuous-time part facilitates the efficient implementation of advanced control and diagnosis functions based on physical models.

## 2 Features available in the prototype

The embedded target for OpenModelica started as an experiment for understanding if it is possible to create a much simpler code generator which is able to generate code for very restricted platforms such as the Atmel AVR 8-bit microcontrollers.

The generated code is target-agnostic ANSI C code which provides an Functional Mock-up Interface (FMI) [FMI14] like structure as discussed in the preceding conceptual report D3.3 [TB16].

The prototype only supports a rather restricted Modelica language subset. It does not support arrays, strongly connected components, event-based behaviour or initialization. However, since the OMC compiler transforms many array equations into scalar equation it still works for many models which use arrays.

Instead of having a big runtime system that is linked in (as is the case for the regular code generator), the code generator will only generate C-functions corresponding to the Modelica function used. This results in the compiler for the embedded target not trying to compile functions that end up unused in the model.

Target platform specific code can be *injected* by carefully designed Modelica libraries<sup>10</sup>. This

---

<sup>10</sup>This can be achieved by using Modelica's external function interface which basically allows to inline C code by using Modelica's `include` annotation. Care needs to be taken that C code, in particular required initialization code, is executed in the correct sequence. This can be achieved by using Modelica's *external object* facilities and introducing suitable artificial "dummy" data-flow dependencies between external object constructor functions which ensure valid execution sequences. See [Thi+17; BST17] for more details on this technique.

approach allows users adding support to new target systems without needing to understand and modify the OMC compiler. This provides a simple mechanism for flexibly adapting code generation to different target systems. If the code generation needs to be changed in a more profound way it is possible to directly modify the Susan template [Fri+09] which is used for the embedded code generation. As a proof of concept the Modelica\_DeviceDrivers (MDD)<sup>11</sup> library was extended for providing support for the AVR ATmega16 and ATmega328P (=Arduino Uno) [Thi+17] and (very recently) STM32F4 boards [BST17]. Figure 1 shows One of the AVR examples included in MDD is the single board heating system (SBHS<sup>12</sup>), shown in Figure 1.



Figure 1: Single board heater system running a real-time control algorithm using firmware based on MDD code [Thi+17]).

The SBHS was developed by IIT Bombay and is used for teaching and learning control systems. The device has a heat assembly, a fan, and a temperature sensor. The aim is to control the fan so that the temperature settles at a set value. The example included in MDD realizes the control logic by Modelica code and provides drag-and-drop blocks for reading sensor values (analog-to-digital converter block) and writing to the actuator (pulse-width-modulation block).

### 3 Implementation

The regular C-code generator creates huge data structures and contains much debugging information while the run-time system contains many numerical solvers and is around 6MB in size (of which 0.5MB is textual strings for error messages). The regular code is compiled to simulation binaries targeted at desktop PCs where this code size imposes no problem. However, the code size is prohibitive huge for restricted embedded systems. It proved to be difficult to reach the necessary improvements in code size reduction by modifying and optimizing the existing code generator, therefore a new code generation target was started from scratch.

<sup>11</sup>MDD library, [https://github.com/modelica/Modelica\\_DeviceDrivers/](https://github.com/modelica/Modelica_DeviceDrivers/).

<sup>12</sup>SBHS, <http://sbhs.fossee.in/>.

The embedded target for OpenModelica is implemented using OpenModelica's code generation template language Susan [Fri+09]. The generated code is target-agnostic ANSI C code which provides an FMI [FMI14] like structure as discussed in the preceding conceptual report D3.3 [TB16]. Traceability information is retained during the complete translation process in order to allow tracing back generated C code fragments to the Modelica model elements that caused their generation. Within the generated C code files the traceability information is included as source code comments next to the generated C code fragments.

Target specific code and logic can be provided by carefully designed Modelica libraries. The previously mentioned proof of concept examples (AVR ATmega16, ATmega328P (=Arduino Uno) and STM32F4) are included in the MDD library, where constants and constant evaluation of functions are used for giving hints to the compiler about possible optimizations. The result is a small, easy to compile, executable that can integrate continuous-time system dynamics (currently using the forward Euler method). The compiler itself (in this case OpenModelica) does not know anything about Atmel AVR or STM32F4, but generates C-code which the user can compile with the corresponding tool-chain and upload to an embedded target which makes the process flexible and extensible.

This can be contrasted to common approaches to embedded system targets in modeling tools, where the modeling tool knows about the intrinsics of the embedded target and it is only possible to compile for these supported targets.

## 4 Usage

The embedded code generation support is part of the OpenModelica v1.12.0 distribution, where it is available as an experimental feature. The embedded code generation target for OpenModelica can be activated by passing the option

```
--simCodeTarget=ExperimentalEmbeddedC
```

to the OMC.

As explained in Section 3 the generated code needs to be compiled with a tool-chain that corresponds to the embedded target. Since these tool-chains will be different for different targets, the instructions for the necessary procedure will differ. The MDD library includes documentation how the tool-chains need to be set up for the respectively supported targets (AVR ATmega16, ATmega328P (=Arduino Uno) and STM32F4). If a user wants to support additional targets, these examples can be used as a starting point for custom extensions.

## 5 Conclusion

The prototype scheduled for M36 has been developed and is part of OpenModelica since the v1.12.0 distribution released on October 31, 2017, which can be downloaded from the OpenModelica website (<https://www.openmodelica.org/>). Two papers related to the code generator prototype have been published, or have been accepted for publication [Thi+17; BST17]. An interesting detail regarding the second publication, which describes the STM32F4 target extension, is that the responsible main developer and first author is not affiliated to any organisations

working with the OPENCPS project — this is a good indicator that the goal of providing code generation support which can be flexibly adapted to different targets by third party developers has been successfully achieved.



## References

- [BST17] Lutz Berger, Martin Sjölund, and Bernhard Thiele. “Code Generation for STM32F4 Boards with Modelica Device Drivers [Work in Progress]”. In: *Proceedings of the 8th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools (EOLTL2017)*. Munich, Germany: ACM, 2017.
- [FMI14] FMI development group. *Functional Mock-up Interface for Model Exchange and Co-Simulation v2.0*. Modelica Association Project “FMI”. Standard Specification. Oct. 2014. URL: <https://www.fmi-standard.org/>.
- [Fri+09] Peter Fritzson et al. “Towards a Text Generation Template Language for Modelica”. In: 7<sup>th</sup> *Int. Modelica Conference*. Ed. by Francesco Casella. Como, Italy, Sept. 2009. DOI: [0.3384/ecp09430124](https://doi.org/10.3384/ecp09430124).
- [TB16] Bernhard Thiele and François Beaudé. *Concept for customizing code generation including flexible adaptation to different target systems*. Technical Note D3.3. ITEA3, Project 14018: OPENCPS project, Dec. 2016.
- [Thi+17] Bernhard Thiele et al. “Towards a Standard-Conform, Platform-Generic and Feature-Rich Modelica Device Drivers Library”. In: 12<sup>th</sup> *Int. Modelica Conference*. Ed. by Jiří Kofránek and Francesco Casella. Prague, Czech Republic, May 2017. DOI: [10.3384/ecp17132713](https://doi.org/10.3384/ecp17132713).
- [TS16] Bernhard Thiele and Per Sahlin. *Translation validation and traceability concept from acausal hybrid models to generated code*. Technical Note D3.2. ITEA3, Project 14018: OPENCPS project, Dec. 2016.