MbedTarget - A Simulink Target for Cortex-M Microcontrollers

Olaf Hagendorf

HS Wismar, Research Group CEA; *olaf.hagendorf@hs-wismar.de*

SNE 28(2), 2018, 75 - 80, DOI: 10.11128/sne.28.sw.10417
Received: April 15, 2018 (Selected ASIM GMMS/STS 2018 Postconf. Publ.), Accepted: May 15, 2018
SNE - Simulation Notes Europe, ARGESIM Publisher Vienna, ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

Abstract. The MATLAB/Simulink add-ons for C and C++ code generation, MATLAB Coder, Simulink Coder and Embedded Coder, are a widely used and well established technologies for rapid prototyping, model based design, real-time simulation and similar technologies. Mathworks provides the prerequisites for code generation for many Simulink blocks and MATLAB functions. Missing is the direct support of peripheral functions like digital and analog input/outputs, communication functionalities and other microcontroller features when the goal is to execute a Simulink model within a small embedded system. To extend the above coders to support these functions, several add-on toolboxes exists, delivered by Mathworks itself or by third-party suppliers. In the following, the programming of microcontrollers is shortly introduced and their resource utilization is compared, starting with assembler up to code generation by a Mathworks coder products. After introducing and comparing different Cortex-M coder toolboxes, the MbedTarget toolbox as an open source alternative is presented. With examples and code snippets, the principle of the code generation process, the work flow of MbedTarget together with the principle to define a Simulink block is shown.

Introduction

MATLAB/Simulink is widely used in the engineering education, among others in control theories. Other topics of the student education are microcontroller (MCU) programming and the integration of these into physical processes. Main tools for implementing the algorithms are C and C++ compiler.

The interaction with physical systems makes knowledge about control theory often necessary. But the media break between Simulink at one side and C and C++ programming at the other side can make this complicated.

With MATLAB, Simulink and Embedded Coder, toolboxes to create C and C++ code out of Simulink models or MATLAB programs are provided.

To combine both topics, the direct support of peripheral functions like digital and analog input/outputs and other MCU features to interact with a physical system within a Simulink model is necessary. For this, additional packages to extend the coder toolboxes exist, as MATLAB or third-party Add-Ons.

After a short introduction of MCU programming variants in Chapter 1 and their resource comparison, the next chapter describes a few coder toolboxes. In chapter 3, the new MbedTarget is introduced as an alternative coder toolbox.

1 Cortex Microcontroller

Cortex MCUs are 32- and 64bit microprocessors, divided into three subfamilies Cortex-M, -R and -A. They are licensed by ARM Holding. The licensees are producing MCUs with the Cortex microprocessor core but company specific peripheral components.

1.1 Cortex-M family overview

Members of the Cortex-M family are 32bit MCUs with a broad range of processing power, memory sizes, peripheral components, etc. Since the introduction of the first variant, the Cortex-M3 core was released in 2005 and first silicon products were sold in 2006 [1], other variants with more or less powerful cores, e.g. Cortex-M7 and -M0, were introduced. The Cortex-M MCUs have been become a widely used technology, a huge amount of manufacturers are delivering variants. The distributor mouser [2] has ca. 6800 variants of Cortex-M MCUs in its catalog. At all, it lists ca. 40000 MCUs.

1.2 Programming

The following subchapters are shortly introducing and comparing typical embedded programming principles.

Assembler

Since the invention of microprocessors, assembly language has been used [3]. To have the full control over them and their periphery, it is still common to use assembly language. The importance of it decreased over the time in comparison to other languages, but it is still under the 10 most used [4].

C/C++ with HAL library

Many MCU producing companies not only provide the silicon chip but also deliver software libraries. Experiences show that unlike with previous 8 and 16bit processors, the programming of current 32bit MCUs without these libraries considerably increases the overhead. As an example, STM32Cube by STMicroelectronics (STM) [5] is introduced. The software consists of two parts:

- **STM32CubeMX** A tool for C code generation for initialization of hardware functions, adding and configuring of middleware libraries like TCP/IP stack, RTOS, USB, file systems etc. and finally the generation of project files for several integrated development environments (IDEs).
- **STM32 MCU** packages These packages implement a hardware abstraction layer (HAL) to provide standardized API calls for all STM MCU families. An additional part of the packages are middleware libraries like TCP/IP stack, RTOS, USB, file systems and graphical libraries.

Mbed library

The Mbed project was started in 2005 by two ARM employers, Simon Ford and Chris Styles, who are helping out in undergraduate and after-school projects. Both were not satisfied with the current situation and developed the idea to ease the MCU development [6]. Among others, main ideas of this are [7]:

- open source MCU debugging and programming
- hardware
- online toolchain object oriented HAL with adaption layer to different company specific HAL libraries

The object oriented HAL is a C/C++ MCU software platform containing object oriented peripheral and library APIs, C adaption layers and startup code for 232 MCUs [7].

Additionally, several Python tools are part of Mbed, providing functions for compiling, testing library management, exporting project files to several IDEs etc.

Mbed significantly reduces the effort of getting started with ARM based MCUs. A drawback is the amount of additional software layers which leads to an increase of binary sizes. The possibility to compile the same code for different MCUs is advantage and a disadvantage at the same time.

Data flow oriented programming

This programming paradigm, i.e. visual programming, has some advantages in comparison to the above mentioned imperative, text based languages [8, 9]:

- the depiction of a real world object succeeds particularly well
- challenging concepts of imperative languages like variables and dynamic data structures are not relevant
- it provides parallel execution without the need of explicit instruction
- the clarity eases prototypical implementation

Conclusion

As a comparison of the different programming principles, the hello world of MCU programming, a LED blinking application, is compared in its resource usage (sizes of read-only and read/write memory) [10]. The effort to program the application, can only be qualitatively estimated. The effort decreases from the usage of assembler language, over a general C and HAL programming and Mbed. The data flow oriented programming will produce the lowest cost in terms of time to create the application. But it also has the highest resource utilization as can be seen in Table 1.

2 MATLAB/Simulink Targets for Cortex-M Controller

Simulink is a widely used data flow oriented development tool. To program MCUs with it, beside the MbedTarget, introduced in this paper, several other targets exist. They are provided by the Add-Ons manager of MATLAB, or are available as third-party tools. Three of these: Embedded Coder Support Package for STMicroelectronics Discovery Boards [11], Simulink Coder Support Package for STMicroelectronics Nucleo Boards [12] and STM32-MAT/TARGET [13] are introduced in the following.

Programming principle	Flash size (byte)	RAM size (byte)
assembler	88	0
C without library	716	1632
MCU specific HAL	1392[1] 2852[2]	1032[1] 1032[2]
Mbed HAL	22576[3] 37716[4]	1432[3] 8484[4]
Data flow oriented[5]	5893[2]	8060

[1] low layer library used [5]
[2] high layer library used [5]
[3] without RTOS
[4] with RTOS (default config.)
[5] MbedTarget v1

Table 1. Resource comparison of programming principles using the blinking example [10].

2.1 Embedded Coder Support Package for STMicroelectronics Discovery Boards

STM32 Discovery kits are low cost solutions for the evaluation of STM32 Cortex-M MCUs by STM. Beside the MCU and the necessary infrastructure to use it, the kits contains additional peripheral items like displays and sensors. Mouser [2] lists 20 kits in its catalog, the support package can handle three of them: STM32F746G-DISCO, STM32F769I-DISCO and STM32F4-Discovery.

For the STM32F4-Discovery the old, no longer supported standard peripheral library is used, the other two boards are using the Mbed library.

The only supported compiler is GNU GCC.

For the STM32F4-Discovery board, Simulink blocks for analog in, digital in/out, audio in/out and Interrupt handling are available. All MCU pins are usable.

For the other boards, blocks for analog in, digital in/out, serial communication, timer, TCP, UDP and audio in/out are available. The MCU pin usage is restricted to 22 pins.

The supported MCU hardware functionality for all three variants is very basic, the pin usage is partly restricted.

To generate, compile and link the C Code of a Simulink model a single step – Deploy to Hardware – is necessary.

The package requires the Embedded Coder toolbox.

Figure 1 shows an example, an alternating digital pin.

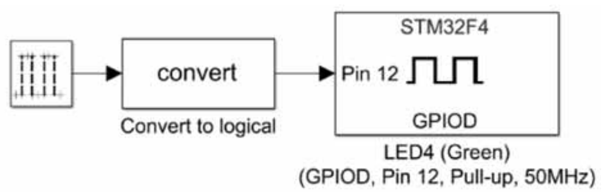


Figure 1. Example with Embedded Coder Support Package for STM Discovery Boards.

2.2 Simulink Coder Support Package for STMicroelectronics Nucleo Boards

STM32 Nucleo kits are the lowest cost solutions for the evaluation of STM32 Cortex-M MCUs by STM. Beside the MCU and the necessary infrastructure to use it, the kits does not contain additional peripheral items except a few kits with Ethernet connectivity.

Mouser [2] lists 38 kits in its catalog, the support package supports 9 of them: Nucleo-F401RE, Nucleo-F103RB, Nucleo-F302R8, Nucleo-F031K6, Nucleo-L476RG, Nucleo-L053R8, Nucleo-F746ZG, Nucleo-F411RE and Nucleo-F767ZI.

For all boards the Mbed library is used.

The only supported compiler is GNU GCC.

For all boards, Simulink blocks for analog in, digital in/out, serial communication and timer are available. The MCU pin usage is restricted to 22 pins. The supported MCU hardware functionality is very basic.

To generate, compile and link the C Code of a Simulink model a single step – Deploy to Hardware – is necessary.

The package requires the Simulink Coder toolbox. Figure 2 shows an example, an alternating digital pin.

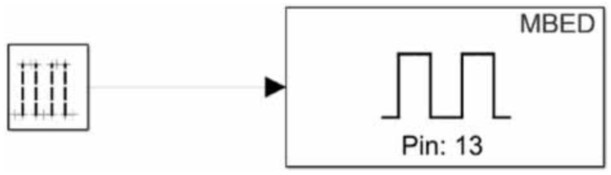


Figure 2. Example with Simulink Coder Support Package for STM Nucleo Boards.

2.3 STM32-MAT/TARGET

This packages is provided by STM. It is based on-STM32Cube. The Simulink package supports allSTM32 MCUs.

All compiler supported by STM32CubeMX can beused: EWARM, Keil MDK V4 and V5, TrueSTUDIO,SW4STM32 and GNU GCC.

Blocks for analog in/out, digital in/out, serial communication, timers, watchdogs are provided. The block set uses the code generation utility STM32CubeMX. Because of this, the supported MCU hardware functionality is considerably large.

To generate, compile and link the C Code of a Simulink model, two steps are necessary: executing the Simulink function: Deploy to Hardware and project building in the chosen IDE.

The package requires the Embedded Coder toolbox. Figure 3 shows an example, an alternating digital pin.

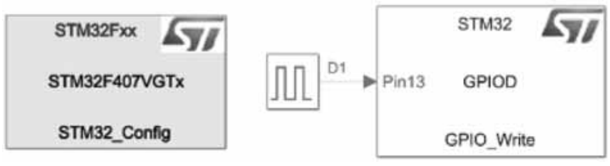


Figure 3. Example with STM32-MAT/TARGET.

2.4 Conclusion

Whereas the provided functionality of the support packages introduced in Chapter 2.1 and Chapter 2.2 is very basic, the other package offers nearly as much functions as the MCU hardware provides.

All packages are free of charge. But the main disadvantage of all is: they are closed source. Own extensions are difficult to realize or even not possible. The supported MCU depends on the manufacturer, mainly STM32 products are supported. Mathworks and third parties offers some more targets supporting e.g. NXP KL25Z and K64F, Infineon XMC, Nordic Semiconductor NRF51 and BBC micro:bit. Although the number of Cortex-M MCUs is huge, the selection of Simulink targets and supported MCUs is restricted.

3 MbedTarget

MbedTarget is completely open from both sides: all code and configuration files i.e. MATLAB code, templates, configurations etc. as well as all MCU libraries are available free of charge and as source codes. It is based on MbedOS 5, therefore all MbedOS 5 compatible microcontrollers can be used to run Simulink models. MbedTarget uses internally GNU GCC to compile the generated source code. Additionally, project files for all Mbed supported IDEs can be generated. With these project files, the generated code can be manually compiled and/or debugged when the Simulink model does not run or own blocks are developed.

The Simulink block library contains two groups of blocks:

- for Mbed functionality
- for sensors, actors, .. based on additional libraries

The first group contains blocks for analog in/out, digital in/out, user LEDs, user buttons, serial communication, timers, Ethernet, RTOS etc.

The second group contains blocks for external analog to digital converter, digital to analog converter, external digital in/out chips, several sensors and actors like temperature, pressure, motion, magnetometer, range, motor control, display, etc. Additionally, MCU specific blocks for encoder, input capture, random number generator and counter are provided.

MbedTarget supports both available coder toolboxes with Simulink model code generation capabilities, Simulink Coder and Embedded Coder.

To generate, compile, link and flash the C Code of a Simulink model a single step – Deploy to Hardware – is necessary.

Figure 4 shows an example, a blinking LED.

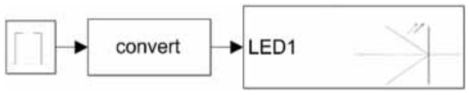


Figure 4. Example with MbedTarget.

3.1 Principle of MbedTarget code generation

The principle workflow when processing a Simulink model to an executable binary is shown in Figure 5.

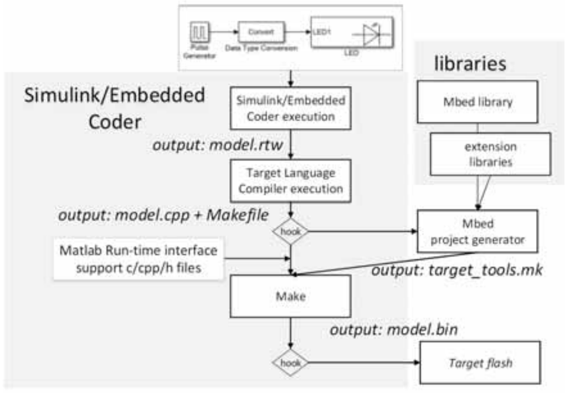


Figure 5. Principle of the MbedTarget code generation.

Simulink or Embedded Coder are generating an rtw file from the model. That is a specific textual representation of the model. Together with tlc files (target language compiler files) which are part of MATLAB and of the specific Simulink target, the target language compiler (TLC) generates several c/cpp/h files and a Makefile meanwhile the process can be influenced by hook calls.

MbedTarget uses the code generation principle based on tlc files: mbed.tlc for the Embedded Coder or mbed_grt.tlc for Simulink Coder as the starting points.

The TLC offers several hooks to customize the code generation process i.e. the transformation process from rtw to c/cpp files. Mainly two hooks are used by MbedTarget: *before_calling_make* for code handling processes and *after_calling_make* for the compiling and flashing process.

The whole procedure can be summarized by the following steps:

1. The TLC uses the current MATLAB working path to store all generated files in a folder with a name constructed using the Simulink model name and 'slprj', e.g. blinky_slprj. During this process Simulink tlc files where used for each standard Simulink block, for each non-Simulink block another tlc file is provided by MbedTarget. Beside the c/cpp/h files generated out of tlc files, also a makefile is created based on template makefiles mbed.tmf or mbed_grt.tmf. This makefile needs an additional make include file, generated within the hook function, described in step 2.
2. After step 1, during the hook call *before_calling_make*, a target specific folder is created. Into this folder, all generated files are copied and a Mbed specific make include file is generated. This include file is very similar to a standard gcc makefile used by Mbed.
3. After executing the hook *before_calling_make* make is called and a bin file is created.
4. During the hook call *after_calling_make,* the bin file is flashed to the target MCU.

3.2 MbedTarget Simulink block usage

This chapter describes shortly the components of an MbedTarget Simulink block. The Digital Output block is chosen because it is one of the simplest blocks. It has only a single input port to write a digital value to MCU pin and is depicted in Figure 6.

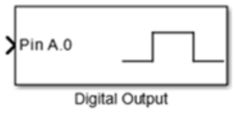
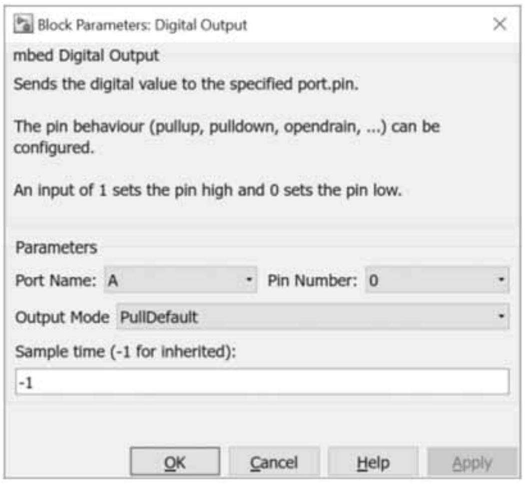


Figure 6. MbedTarget Digital Output block.

With a double click onto the block, a configuration dialog opens as shown in Figure 7. The block has 4 parameters which has to be configured in the dialog:


Block Parameters: Digital Output

mbed Digital Output

Sends the digital value to the specified port.pin.

The pin behaviour (pullup, pulldown, opendrain, ...) can be configured.

An input of 1 sets the pin high and 0 sets the pin low.

Parameters

Port Name: A Pin Number: 0

Output Mode PullDefault

Sample time (-1 for inherited):

-1

Figure 7. Parameter dialog box.

- Port Name and Pin Number to choose a digital port, e.g. PA0
- Output Mode for pull up, pull down, open drain, … - options are corresponding to Mbed options of the DigitalInOut C++ class for digital in/out pins.
- Sample time defines the time period, how often the digital value is written to the MCU pin. The value has to be a multiple of the global step size.

3.3 MbedTarget Simulink block creation

To implement custom Simulink blocks, the following elements are necessary:

- a block mask
- a S-function
- a tlc file consisting of a mixture of target language and C/C++ code

The *block* mask defines the outlook of the block as shown in the Simulink model editor, defines the input items available in the block mask dialog. It also defines block title, block help text and the help menu entry.

The block parameters connect the block with a *S-function*, i.e. a binary mexw64 file implemented in C, and a tlc file witch has the identical name. In MbedTarget, the S-function mainly describes the behavior of the block: number and type of input and output ports, checks parameters and prepares the transfer of these to the TLC.

A *tlc file* is a mixture of tlc code, a script like language, and C/C++ code in a form of snippets. The script code controls the usage of the C/C++ snippets and how and where they are put into a generated C/C++ file. The tlc file contains mainly three functions: *Setup*, *Start* and *Output*.

Setup controls the inclusion of additional headers and source files, where *Start* is executed once to generate initialization code and *Output* is called once in every simulation loop. Even if the name is output, the function has to handle also input values, send by Simulink to input ports.

In the following, snippets of the *Start* function are described in detail as an example:

```
1. %assign nPort-
   Name=LibBlockParameterValue(PortName,0)
2. %assign nPin-
   Num=LibBlockParameterValue(PinNumber,0)
3. %assign
   pname="P"+FEVAL("char",nPortName+64)
4. %assign
   pname=pname+"_"+FEVAL("int2str",nPinNum-1)
```

Lines 1 and 2 fetches the variables entered in the block mask dialog. Lines 3 and 4 creates from these a pin name, e.g. PA_0 as used for STM32 MCUs.

```
5. %assign name = FEVAL("strrep",
   LibGetFormattedBlockPath(block),"/","_")
6. %assign name = FEVAL("strrep",name," ","_")
7. %assign name = FEVAL("strrep",name,"-","_")
```

Lines 5 to 8 create a unique name based on the complete block name, e.g. *blinky_Digital_Output* for a block *Digital Output* in a model *blinky*. The Simulink path contains characters, e.g. '/', spaces and '-', which are replaced by '_' to create a valid C identifier.

```
8.  %openfile declbuf
9.  DigitalInOut %<name>(%<pname>);
10. %closefile declbuf
11. %assign srcFile = LibGetModelDotCFile()
12. %<LibSetSourceFileSection(srcFile, "Decla-
    rations", declbuf)>
```

Finally, lines 8 to 12 create a single line in the declaration section of the generated source file, shown in line 13:

```
13. DigitalOut blinky_Digital_Output(PA_0);
```

The line 13 is the necessary Mbed code to create a digital output.

A few more lines in the tlc file, using the same principle, are creating the remaining, necessary C code.

3.4 MbedTarget main function

The MbedTarget supports the single task model of Simulink. To create the main function, the target contains template file: mbed_srmain.tlc and mbed_grt_main.cpp. The creation of multithreaded application by Simulink is not yet supported, but can be done manually with MbedTarget RTOS blocks.

4 Conclusion

MbedTarget was developed to improve and ease the MCU usage in several modules like robotics, sensor/ actor systems and embedded control systems of student education. A common characteristic of all these modules is the intensive usage of MATLAB/Simulink in the theoretical part. The practical implementation of the theoretical knowledge is time consuming and errorprone when using typical programming languages like C/C++. By using MbedTarget, the media break between Simulink and C programming could be removed. The disadvantage of an increased resource usage does not play a role in the prototypical implementations in this application field, enough MCU resources are available without problems.

MbedTarget is published at GitHub [14] and will be continuously developed there.

References

[1] Yiu J. Definitive Guide to ARM Cortex-M3 and Cortex-M4 Processors. Newnes, UK, 2013.

[2] https://www.mouser.de (accessed 01.2018)

[3] Kusswurm D. Modern X86 Assembly Language Programming. Springer, USA, 2014.

[4] https://www.tiobe.com/tiobe-index/ (accessed 12.2017)

[5] STM32Cube: http://www.st.com/en/embedded - software/stm32cube-mcu-packages.html (accessed 12.2017)

[6] https://os.mbed.com/handbook/Foundersinterview (accessed 12.2017)

[7] https://www.mbed.com/en/ (accessed 12.2017)

[8] Keltsch C. Ein visuelles Programmiersystem zur Modellierung deskriptiver Untersuchungen in der Datenanalyse. Diplomarbeit, Diplomica Verlag, 1998.

[9] Maydl W. Komponentenbasierte Softwareentwicklung für datenflußorientierte eingebettete Systeme. Dissertation, University Passau, Germany, 2005.

[10] https://github.com/ATM-HSW/ASIMFachtagung2018 (accessed 01.2018)

[11] https://de.mathworks.com/hardwaresupport/st-discovery-board.html?s_tid=AO_HS_info (accessed 01.2018)

[12] https://de.mathworks.com/hardwaresupport/st-nucleo.html?s_tid=AO_HS_info (accessed 01.2018)

[13] http://www.st.com/content/st_com/en/products/development-tools/software-developmenttools/stm32-software-developmenttools/stm32-utilities/stm32-mat-target.html (accessed 01.2018)

[14] https://github.com/ATM-HSW/mbed_target (accessed 01.2018) 130 Proc.