PEERT™- BLOCKSET FOR PROCESSOR EXPERT™ AND MATLAB®/SIMULINK® INTEGRATION

R.Bartosinski[1], P.Stružka[2], L.Waszniowski[3]

[1]Institute of Information Theory and Automation, Czech Academy of Sciences

[2]UNIS, spol. s r.o., Brno

[3]Department of Control Engineering, Faculty of Electrical Engineering, CTU

Abstrakt:

This paper describes source code generation for embedded systems from MATLAB®/Simulink® model by Real-Time Workshop® and the PEERT™ library. The PEERT™ is briefly presented and generation of source code is shown on two simple examples.

1 Introduction

An application can be created by hand or by tools which generate source code (It can be created by combination of both ways, of course). Source code generators have many advantages. The main advantage of source code generators for personal computer is rapid development of such application. Other advantages of generated code for embedded application are better properties of code (smaller, faster or with lower memory demands), transparency, errorless, safety. Important advantage is compliance to required standards and easier and faster portability of application between different platforms.

Programming with such tools is focused on visual design of an application and writing of program functions which mostly use other libraries. Next simplifications and speed up of an application programming are bringing with higher languages and other developing tools. One of these tools can be MATLAB®/Simulink® with Real-Time Workshop® (RTW). An application can be generated from a Simulink model with these tools. Simulink model determines functions in the application (blocks) and data flow in the system (blocks interconnections), hence source code of full application can be generated.

A RTW Embedded Coder is add-on product for use with Simulink and RTW. It generates code that can be optimized for speed, memory usage and simplicity [5]. General diagram of embedded software model is shown on figure 1. In most cases tools generate source code of application but hardware abstraction layer (HAL) must be written by hand for each platform. Some tools generate HAL, too. Unfortunately these tools generate HAL only for one target platform or for small limited group of similar platforms (e.g. Embedded Target for Motorola HC12 or Embedded Target for TI C6000 DSP). Another disadvantage of existing Simulink add-ons for source code generation is worst support for simulation.

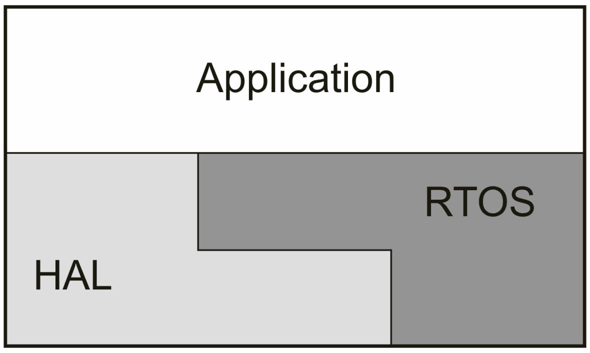
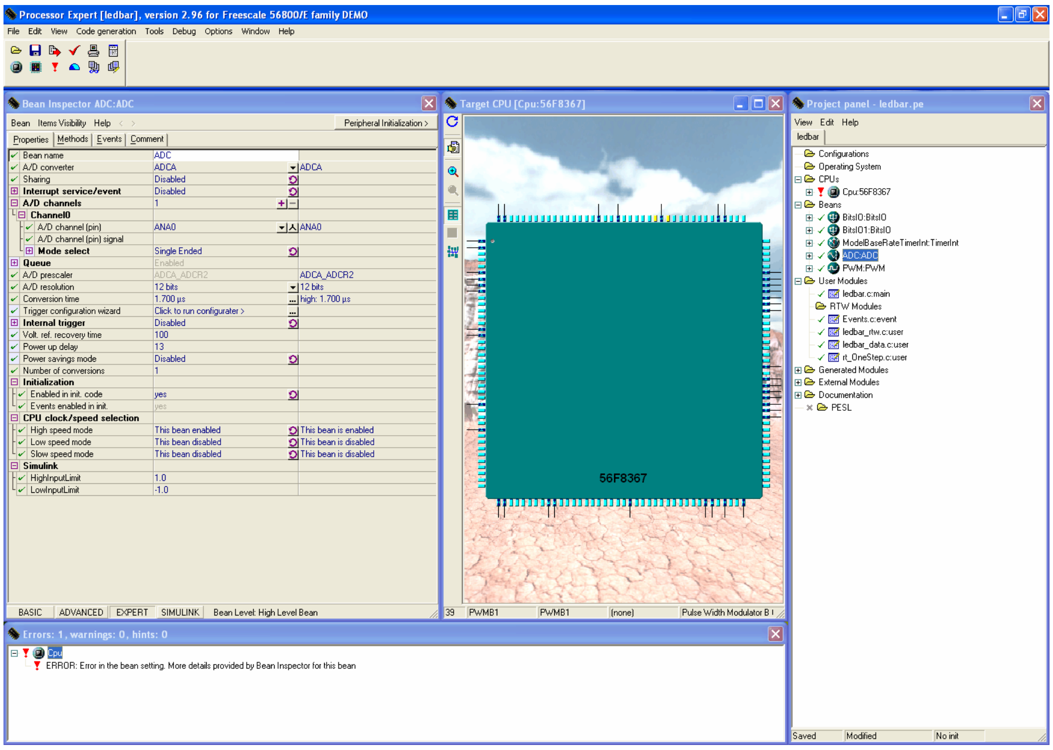


Figure 1: Typical model of embedded control unit software layers.
(HAL-Hardware abstraction layer, RTOS-Real-time operation system)

MATLAB/Simulink PEERT library, which is created in our SESA project, is based on mentioned model of embedded software. The code of application layer is generated from the Simulink model by the PEERT and the code of HAL is generated in Processor Expert™ (PE) tool. PE is described in the next chapter. The PEERT library is described in chapter 3 and chapter 4 presents library usage for simulation and code generation on simple example.

2 Processor Expert™

PE is advanced, component oriented, open Rapid Application Development (RAD) environment for embedded systems. The main task of PE is to manage CPU and other hardware resources and to allow virtual prototyping and design. PE contains knowledge base of information about CPUs and its on-chip peripherals. They are divided to separate units called Embedded Bean™ (bean). Beans encapsulate functionality of basic elements of embedded systems like CPU core, CPU on-chip peripherals, standalone peripherals, virtual devices, and pure software algorithms and change these facilities to properties, methods, and events (like objects in OOP). Beans are well tested software and that is why they can save months of work of the expert programmer [7].


Processor Expert [ledbar], version 2.96 for Freescale 56800/E family DEMO
File Edit View Code generation Tools Debug Options Window Help
Bean Inspector ADC:ADC
Bean Items Visibility Help
Peripheral Initialization >
Properties Methods Events Comment
Target CPU [Cpu:56F8367]
56F8367
Project panel - ledbar.pe
View Edit Help
ledbar
BASIC ADVANCED EXPERT SIMULINK Bean Level: High Level Bean
39 PWMB1 PWMB1 (none) Pulse Width Modulator B I
Saved Modified No init

Bean name	ADC	
A/D converter	ADCA	ADCA
Sharing	Disabled	
Interrupt service/event	Disabled	
A/D channels	1	
Channel0		
A/D channel (pin)	ANA0	ANA0
A/D channel (pin) signal		
Mode select	Single Ended	
Queue	Enabled	
A/D prescaler	ADCA_ADCR2	ADCA_ADCR2
A/D resolution	12 bits	12 bits
Conversion time	1.700 µs	high: 1.700 µs
Trigger configuration wizard	Click to run configurater >	
Internal trigger	Disabled	
Volt. ref. recovery time	100	
Power up delay	13	
Power savings mode	Disabled	
Number of conversions	1	
Initialization		
Enabled in init. code	yes	
Events enabled in init.	yes	
CPU clock/speed selection		
High speed mode	This bean enabled	This bean is enabled
Low speed mode	This bean disabled	This bean is disabled
Slow speed mode	This bean disabled	This bean is disabled
Simulink		
HighInputLimit	1.0	
LowInputLimit	-1.0	

```
Errors: 1, warnings: 0, hints: 0
Cpu
  ERROR: Error in the bean setting. More details provided by Bean Inspector for this bean
```

```
Configurations
Operating System
CPUs
  Cpu:56F8367
Beans
  BitsIO:BitsIO
  BitsIO1:BitsIO
  ModelBaseRateTimerInt:TimerInt
  ADC:ADC
  PWM:PWM
User Modules
  ledbar.c:main
  RTW Modules
  Events.c:event
  ledbar_rtw.c:user
  ledbar_data.c:user
  rt_OneStep.c:user
Generated Modules
External Modules
Documentation
  PESL
```

Figure 2: The main window of Processor Expert®.

Unlike common libraries, beans are implemented for all possible peripherals, with optimal code. Methods are interfacing bean functionality to user's code. All enabled methods are generated into appropriate bean modules during code generation process. Some beans allow handling the hardware or software events related to the bean. User can specify the name on function invoked in the case of event occurrence.

PE knows exactly the relation between allocated peripherals and selected beans. When the user chooses a peripheral in the bean properties, PE proposes all the possible candidates but signals which peripherals are allocated already and also signalizes peripherals that are not compatible with current bean settings. In the case of an unrealizable allocation, an error is generated.

New beans can be created and edited by Bean Wizard. It provides a powerful interface for the composition of new beans, and generates the bean files.

These tools can generate optimal code which can be compliant with common standards (e.g. HIS or AUTOSAR standards for automotive software). The generated code from PE has a uniform interface for all supported microcontrollers, and therefore the target platform can be changed without changing model or tool.

3 PEERT library

The Processor Expert Embedded Real-time Target (PEERT™) library integrates MATLAB®/Simulink® with Processor Expert™. The library is divided into three parts. The first part provides interconnection between Simulink and PE. It synchronizes an model and used blocks with the corresponding PE project and beans. The next part of the library is a blockset with blocks, which are corresponding with selected basic beans (Figure 3). The last part is RTW target based on Embedded Real-Time Workshop target (ERT) for source code generation from model.

Interconnection between a Simulink model and a PE project is based on inter-process communication through Microsoft Component Object Model (COM) [8]. PE contains COM server which provides functions. They can be called from the library. The PE COM server allows registering PE callback functions, as well. These callback functions are invoking in PE by events (e.g. save project, remove bean, property changing) and the library can respond with the corresponding actions in the model (e.g. save model, remove block, property changing). This mutual communication guarantees synchronization between Simulink model and the corresponding PE project.

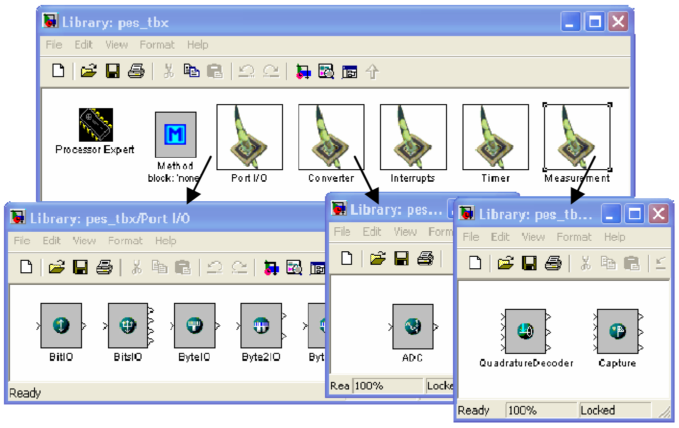


Figure 3: The PEERT™ basic blockset for Simulink®.

The PEERT blockset contains two special blocks and other blocks which correspond to selected PE beans (e.g. ADC, PWM, PortIO, Quadrature Decoder). The most important block in the PEERT is Processor Expert block. It provides interconnection between the model and the PE project described formerly. When a PE block is inserted to the model a new PE project is created. If the model contains a PE block and it is saved, the PE project is saved to the same directory with the same name. The PE block allows CPU selection and other settings in PE tool, as well.

Blocks in the blockset are divided to similar groups as in PE according to their function. These blocks implement basic behavior and provide properties of represented beans – mostly hardware on-chip peripherals which separate processor and its environment. Therefore the part of the model for generation can be encapsulated to a subsystem. This concept allows simulation and generation from one common model.

Each block from the PEERT blockset behaves in a model as selected method of the corresponding PE bean. For example: ADC block makes measuring on input and value translating to output in required format. All input and output ports of PEERT blocks have data type set to unsigned integer implicitly. Majority of implemented blocks in the PEERT contain events (e.g. an event 'end of measuring' in the ADC bean). They are represented as sources of function-call signals in PEERT blocks. These signals can be used for event-driven systems and with the Stateflow tool.

Properties of blocks needed for a simulation and code generation are read from and written to PE beans through COM connection, because PE beans use the same properties for optimal bean source code generation. It allows to use PE Bean Inspector window for property settings, as well. This concept takes advantage of PE knowledge base about entire created system and its dependency on the other used beans/blocks.

Each block in the PEERT blockset provides only one selected function, but corresponding beans provides more functions. Therefore the special block Method is in the PEERT library. This block allows to use all other functions of PE beans in Simulink model. It allows to use beans which are not in the PEERT blockset, as well. This block cannot be simulated, i.e. block is ignored in simulation, and it can be disadvantage in some cases.

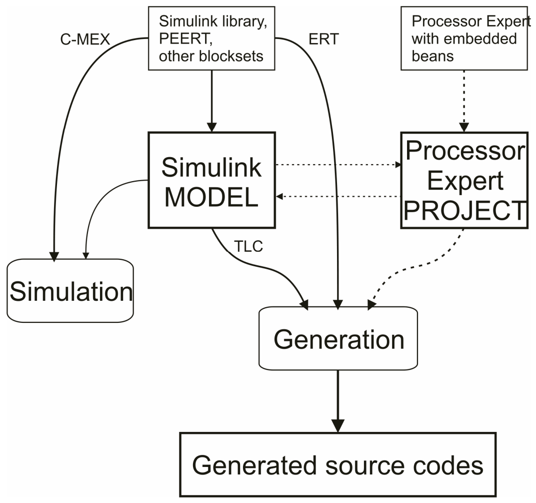


Figure 4: Source code generation and simulation flow. Dotted lines mean automatic process.

Source code generation from a Simulink model and the corresponding PE project is shown on figure 4. Before invoking generation a CPU bean must be added into the PE project and all beans must have correctly set their properties. When RTW build process is invoking on the model, RTW Embedded Coder creates RTW file with model description and then it uses Target Language Compiler to creation source code of model blocks from description in TLC files, which must be provided with each user block.

After invoking generation, the first step is creation of RTW file which describes model. The next step is generation of block code from TLC files. These block source codes are combined according to data flow in the model. Source codes of PEERT blocks are mainly calling of the corresponding PE bean selected function. There is invoking generation of the used PE beans source codes in the PE as the next step. The last step of the generation is automatic correction of generated code – the generated PE init and loop functions are called from the RTW generated functions. If there wasn't any error, generated code can be compiled and linked to target application.

4 Examples of system with PEERT™ blocks

The first example shows how to use Processor Expert blockset to create a simple application. The application is a simple LED bar. The used CPU has 12bit ADC and six bits with LEDs as output. Behavior of application is described as follows. When the ADC value equals to zero, no LED will be switched on. When the ADC value will increase, the LEDs will switch on in sequence. And if the ADC value will be on maximum, all LEDs will be switched on.

LEDs are connected to two output ports controlled by two bytes. These bytes can be simply obtained from one common byte. Therefore the application behavior can be described by the following universal formula (all variables must have integer type).

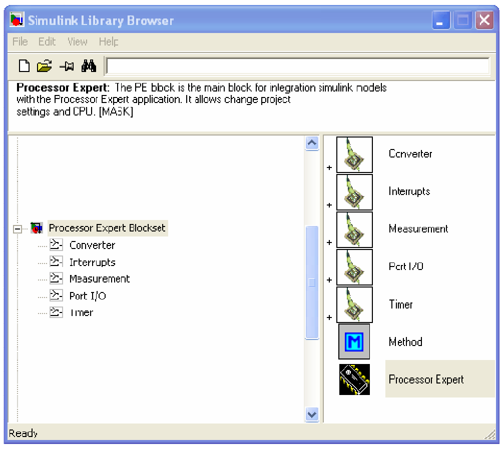
$$y = \left[1 << \left(\frac{x(m+1)}{2^n} \right) \right] - 1 \tag{1}$$

Where x is input value from ADC, y is output value, m is number of output's bits (number of LEDs) and n is number of input's bits (ADC resolution).

The following list is simplified step-by-step guide with commentary:

- Create a new Simulink model.
- Add a Processor Expert block from the PE blockset to model. – It creates a new PE project automatically.

- Add all other blocks from the PE blockset (They all have common specific design, as shown on figure 4). - When some block from the PE blockset is added to model, the corresponding PE bean is automatically added to the PE project.



Processor Expert: The PE block is the main block for integration simulink models with the Processor Expert application. It allows change project settings and CPU. [MASK]

Figure 5: Processor Expert Blockset with selected Processor Expert block.

- Add other blocks from Simulink libraries as needed to model the application function.
- Add additional blocks to simulate and verify model. (After this step the model looks as shown on figure 6.)

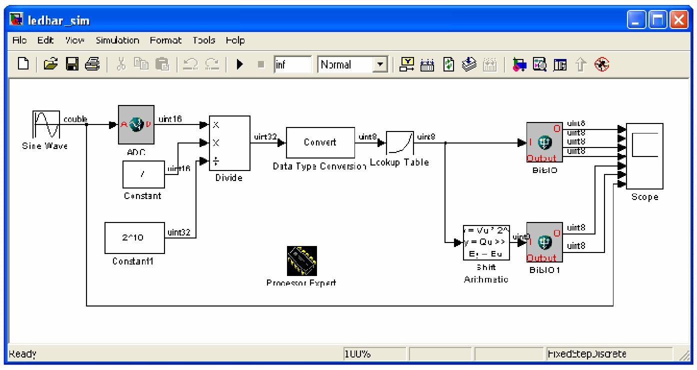
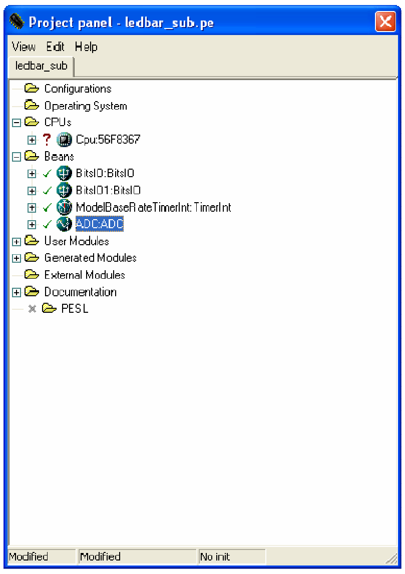
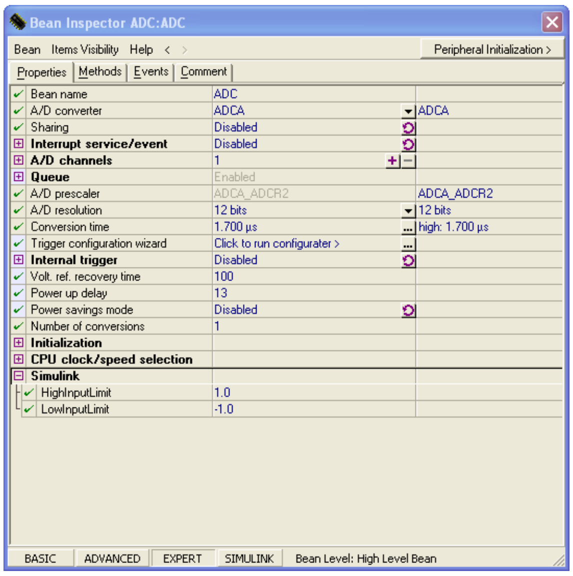


Figure 6: The initial model of LED bar application.

- Set blocks properties in windows which are invoked by double-click on blocks. (Properties of blocks from PEERT blockset are set in Bean Inspector window in PE.)




Bean Inspector ADC:ADC
Bean Items Visibility Help
Peripheral Initialization >
Properties Methods Events Comment
BASIC ADVANCED EXPERT SIMULINK Bean Level: High Level Bean

Bean name	ADC	
A/D converter	ADCA	ADCA
Sharing	Disabled	
Interrupt service/event	Disabled	
A/D channels	1	
Queue	Enabled	
A/D prescaler	ADCA_ADCR2	ADCA_ADCR2
A/D resolution	12 bits	12 bits
Conversion time	1.700 µs	high: 1.700 µs
Trigger configuration wizard	Click to run configurater >	
Internal trigger	Disabled	
Volt. ref. recovery time	100	
Power up delay	13	
Power savings mode	Disabled	
Number of conversions	1	
Initialization		
CPU clock/speed selection		
Simulink		
HighInputLimit	1.0	
LowInputLimit	-1.0	

Figure 7: Processor Expert project panel and bean inspector of ADC bean.

- Set model configuration – type to ‘fixed-step’, solver to ‘discrete’ and fixed-step size to 0.2s (Fixed-step size is used as main loop sample rate in generated code.) and RTW target to PEERT target. (Now the model can be simulated.)
- For source code generation from the model with blocks for simulation, blocks for generation must be grouped into a subsystem.
- Double-click on the Processor Expert block invoke the PE project, where a CPU bean must be added and set.
- If all beans are correctly set all properties building of the subsystem can be started. – Set of output source codes is under model directory.
- Compile and link these source codes to obtain the application for target platform.
- Run the application on the target platform.

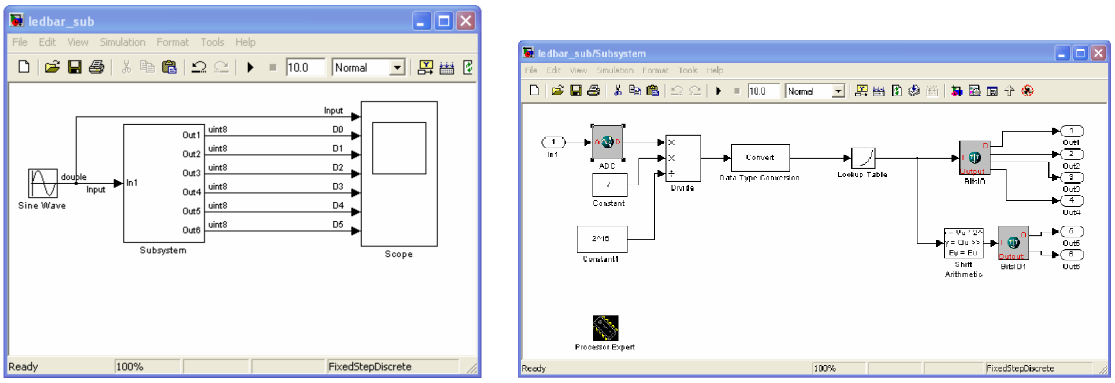


Figure 8: Model of the example system and its subsystem in detail.

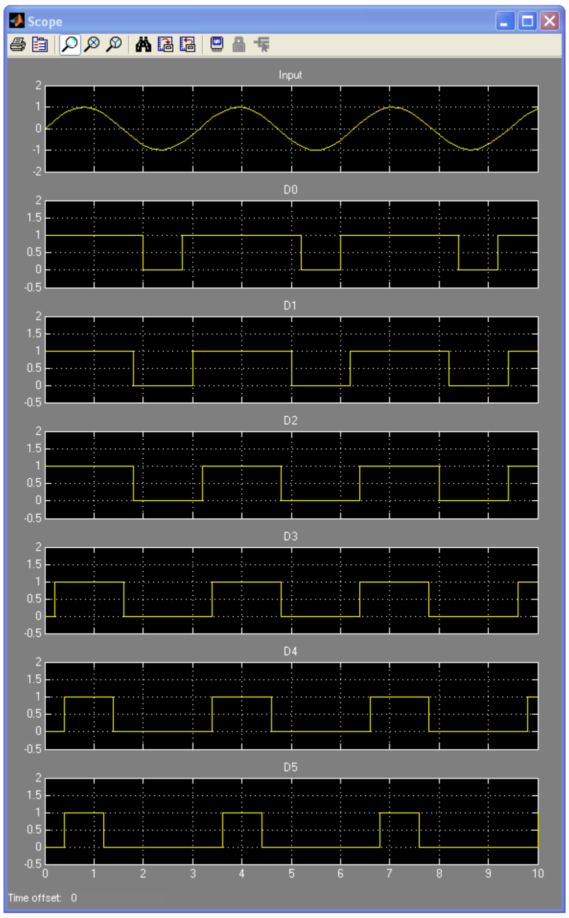
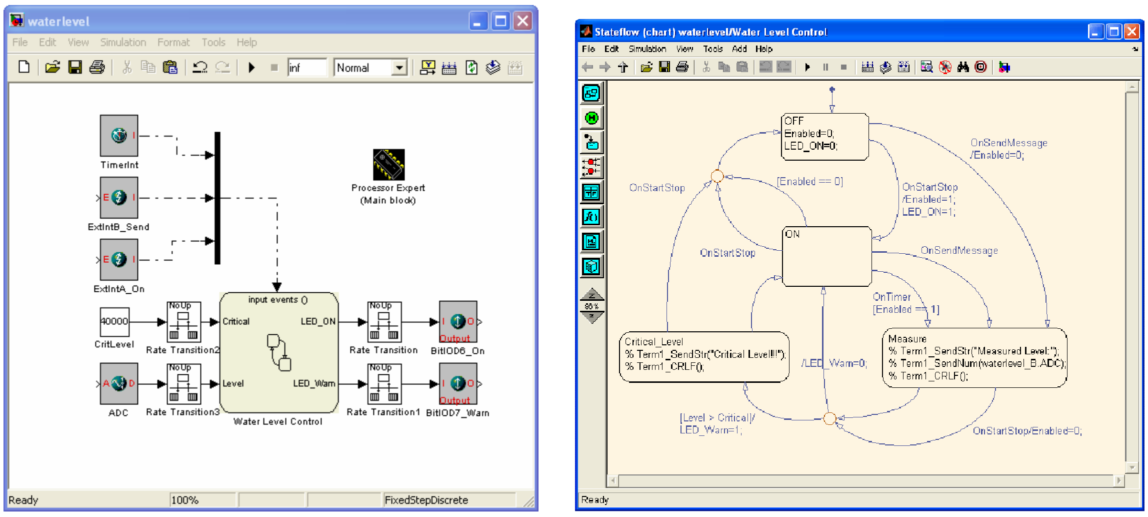


Figure 9: Scope with signals of the simulated example system.

The example shows benefits of Simulink model and PE project integration. The resulted application was created in very short time with guaranteed verification of algorithm (using simulation in Simulink and drivers' code generated from PE). This approach means the application will get sooner to the market with less effort and less time spent with development. The application remains portable over several platforms (supported by PE).

The second example illustrates using of PEERT with the Stateflow tool (Figure 10). The example is a water level alarm. The system has two inputs for buttons (‘On/Off’ for switch on/switch off and ‘Send’ for immediate measuring) and one analog input for a water level sensor. The buttons are connected to interrupts. The system has two bit outputs (LED ‘On’ and LED ‘Warn’). When the system is switched on, the ‘On’ LED shines and the water level is periodically measured (The period of measuring is set in the TimerInt block/bean). When the water level is higher than limit, the ‘Warn’ LED is switched on. If the ‘Send’ button is pressed, the water level is immediately measured.



```
OFF
Enabled=0;
LED_ON=0;
```

```
Critical_Level
% Term1_SendStr("Critical Level!!!");
% Term1_CRLF();
```

```
Measure
% Term1_SendStr("Measured Level:");
% Term1_SendNum(waterlevel_B.ADC);
% Term1_CRLF();
```

Figure 10: Model of the model of water level alarm.

Connection PE blocks (PE maps hardware resources like peripheral interrupts as block events.) with Stateflow charts brings new potential to create complicated event-driven systems. Resulting model can still be simulated.

5 Conclusions and future work

The paper briefly presents PEERT™ blockset which integrates tools Processor Expert™ and MATLAB®/Simulink®/Real-Time Workshop®. The PEERT™ target is based on Embedded Real-Time target and it generates source code for embedded systems with Processor Expert™ supported CPU. The generated code is consisted of two parts – hardware abstraction layer which is produced by PE and application which is produced from a Simulink model with our blockset.

Integration of these tools is very useful, because entire source code of embedded application is generated and developer needn’t write any part of code by hand. Another reason is errorless of code and potential compliancy with the required standards. One of advantages over other source code generators is capability to simulate system before generation. The PEERT™ library is suitable to generate event-driven application, as well.

The future work on the PEERT™ library will be focused on expanding group of supported blocks and creation special library for AUTOSAR automotive standard.

Acknowledge

The SESA project is supported by Academy of Sciences of the Czech Republic under project No. 1ET400750406.

References

[1] *MATLAB - External Interfaces*. The MathWorks, Inc., www.mathworks.com, 2005.
[2] *Simulink - Writing S-functions.* The MathWorks, Inc., www.mathworks.com, 2005.
[3] *Real-Time Workshop User's Guide.* The MathWorks, Inc., www.mathworks.com, 2005.
[4] *Real-Time Workshop Target Language Compiler.* The MathWorks, Inc., www.mathworks.com, 2005.
[5] *Real-Time Workshop Embedded Coder User's Guide.* The MathWorks, Inc., www.mathworks.com, 2005.
[6] *Real-Time Workshop Embedded Coder Developing Embedded Targets.* The MathWorks, Inc., www.mathworks.com, 2005.
[7] *Processor Expert help.* UNIS, spol. s r.o., www.processorexpert.com, 2005.
[8] *Microsoft Component Object Model.* Microsoft Corporation, msdn.microsoft.com, 2005.

Roman Bartosinski
Department of Signal Processing,
Institute of Information Theory and Automation,
Czech Academy of Sciences,
Pod vodarenskou vezi 4, 18208 Praha 8, Czech Republic
Email: bartosr@utia.cz
www: www.utia.cz/ZS

Petr Struzka
UNIS, spol. s r. o.
Jundrovska 33, 62400 Brno, Czech Republic
www: www.unis.cz, www.processorexpert.com

Libor Waszniowski
Czech Technical University
Faculty of Electrical Engineering
Department of Control Engineering
Technicka 2, 166 27 Praha 6, Czech Republic
www: www.dce.felk.cvut.cz