Integrated Environment for Embedded Control Systems Design

Roman Bartosinski[1], Zdeněk. Hanzálek[2], Petr Stružka[3], and Libor Waszniowski[2]

[1]Czech Academy of Sci.,
Pod vodarenskou vezi 4,
182 08 Praha 8, CR
bartosr@utia.cas.cz

[2]Czech Technical University
Karlovo nám. 13
121 35 Prague 2, CR
{hanzalek, xwasznio}@fel.cvut.cz

[3]UNIS, Ltd.
Jundrovská 33
624 00 Brno, CR
pstruzka@unis.cz

Abstract

The motivation of our work is to make a design tool for distributed embedded systems compliant with HIS and AUTOSAR. The tool is based on Processor Expert, a component oriented development environment supporting several hundreds of microcontrollers, and Matlab Simulink which is the de-facto standard in the rapid prototyping of the control applications but it does not have an adequate HW support. The objective is to provide an integrated development environment for embedded controllers having distributed nature and real-time requirements. Therefore we discuss the advantages of using an automatically generated code in the development cycle of the control embedded software. We present a developed block set and Processor Expert Real-Time Target for Matlab Real-Time Workshop Embedded Coder. The case study shows a development cycle for a servo control design.

1. Introduction

Since the Matlab development tool chain has become a standard in the control applications development, we focus on its facilities for a code generation. As the Matlab main weakness is identified a poor support for handling hardware devices of a target microcontroller. Since Processor Expert (PE), a tool for the microcontrollers' hardware resources management and design at high level exists; we bring an improvement of the Matlab facilities for handling the controller hardware by integrating Processor Expert to the Matlab Simulink environment.

The design of control systems is often treated separately from the design of its software and hardware implementation. The increasing use of electronic control units in automotive applications has caused an increasing need for the simultaneous consideration of the control system and its implementation platform during the development. There is a need for supporting tools that assist designers in the modeling, the simulation and the analysis while capturing relationships among various requirements such as the control performance (e.g. rise time, overshoot, and stability), the response time, resources used (memory footprint, peripheral devices), the energy consumption, the robustness, the cost, and the design parameters related to the control system and platform design.

The product demands in terms of the competition and the legislation will moreover cause a need for the system design optimization. This is particularly relevant for a large series production where the goal is to make the hardware cost proportion as small as possible [5], [11].

The digital control theory normally assumes equidistant sampling intervals and a negligible or constant control delay from the sampling to the actuation. However, this can seldom be achieved in practice in a networked embedded system [4]. For control systems this is of particular concern. Timing variations in sampling periods and latencies degrade the control performance and may in extreme cases lead to the instability. One solution is to simulate such a behavior while using e.g. TrueTime [2], a Matlab/Simulink toolbox, which requires the precise representation of the control algorithm structure, the worst case execution time of operations and other parameters. The second solution, represented by Targetlink [3] or the approach shown in this article, is based on an automatic code generation and the processor-in-the-loop (PIL) or hardware-in-the-loop (HIL) testing.

Tools supporting the co-design of control systems and their real-time implementation [2],[3],[6],[8],[9],[10] have various objectives ranging from the simulation to the formal verification and the code generation. The current status and the future directions are surveyed in [1][5] and more extensive description of the surveyed tools is given in [7].

The model based design and the automatic generation of the production code is increasingly employed in the development of automotive applications. To support this trend, we have developed an embedded real-time target integrating the tool Processor Expert (PE) to the environment of Matlab Simulink.

PE is a tool generating a production quality C code that provides an hardware abstraction layer allowing to access peripherals (ADC, PWM, Timer,…) of many supported microcontrollers (MCU), covering the Freescale production line and many National Semiconductors and Fujitsu MCU, via an unified application interface that can be compliant with common standards (e.g. HIS or AUTOSAR [1]). A developer does not need to study all details relating to control registers of MCU peripherals. He only specifies the fundamental parameters (e.g. the resolution of ADC, the input pin, the conversion time, the mode of operation) and selects high level methods and events to access the peripheral (e.g. Measure, GetValue). Moreover, the selected parameters are verified by PE.

Matlab Simulink, on the other hand, allows engineers to develop a control application algorithm in the high level graphical language of data-flow and state-flow diagrams. The C code can be automatically generated from the model. However, the hardware (HW) of peripheral devices is not supported well. Only few MCUs are supported, portability is limited since blocks representing peripherals are different for different MCUs; they do not usually allow adjusting all HW parameters and no verification of this error prone process is done.

It is clear from the mentioned brief description of PE and Matlab Simulink that they complement one another perfectly. To allow designers to use the best features of each of these tools at the rapid application development cycle, we have developed a peripheral devices block set and a code generator target integrating PE to Simulink. PE generates the code of a HW abstraction layer and Simulink use it in the application code.

The integration of PE to Simulink allows control engineers familiar with the graphical environment of Simulink to adjust the HW peripherals that are an inseparable part of each control application on the high level, without the detail knowledge of the specific MCU.

Contrary to the other existing targets for the code generation from Simulink, PE block set allows to use all features of the HW – there are no predefined adjustments that can not be changed by the user. Since many peripherals generate interrupts and they are all supported by the corresponding blocks in the PE block set, the control application can consist of both, event driven and time driven tasks.

From the strategic point of view, it is important that due to the HW abstraction layer provided by PE, the PE block set and the target automatically support all MCUs supported by PE - that are the most important families of MCU produced by Freescale. Also new MCUs coming to the market will be supported by the PE producer.

The model with the PE blocks can be moreover extremely simply ported to another MCU by selecting another CPU bean in the PE project window. The application design in Simulink therefore becomes HW independent.

The motivation for using an automatic code generation in the development cycle of embedded control applications is discussed in section 2. Matlab facilities for the code generation and weaknesses of the existing code generation targets are described in section 3. A brief description of PE follows in section 4. The integration of PE to the Matlab tool chain, the main contribution of this paper, is described in section 5. A support of the processor in the loop simulation is described in section 6. A short case study demonstrating the using of this technology is presented in section 7. Finally concluding remarks and future work directions are indicated in section 8.

2. Motivation to Automatic Code Generation

Problems related to the manually coded software arise from its huge complexity on one side and the requirement of its high reliability and short time to the market on the other side. The powertrain control unit software, for example, consists of 50000 lines of a code, its development effort takes 40 man-years and the average productivity of the coding process is 6 lines per day. The time to the market is only 24 months, the validation takes 5 months and the changing rate is 3 years [14].

Regardless of the quality and the efficiency problems with the manually coded SW, there is also a problem originating in the classical development process of the control application. The control strategy is formulated as a control algorithm that is continually improved and refined by the simulation on a model. Once the algorithm design is finished the implementation is done manually. With the exception of simple projects, the tasks of the control algorithm design and its implementation are done by different specialists, or even teams. Realize however, that the points of view of these specialists, their qualifications and the used tools are different. While the control engineer sees the controlled object and the requirements on the control quality, the software engineer focuses on the implementation and the architecture of the real-time system and probably does not know the details of the controlled system dynamic and the motivations for the decisions done by the control engineer.

However, the implementation is not only a simple translation of the algorithm specified in details by the simulation model to the target language. Also many decisions affecting the behavior of the control algorithm (e.g. the hardware/software deployment, the scheduling policy) must be done.

Once the controller is manually implemented, its code is handled by software development tools (compilers, debuggers, profilers etc) that do not support the control theory point of view. Simultaneously, due to the manual implementation, the tool used for the control algorithm design and simulation looses the link between the model and the executable application. A validation and tuning of the implemented controller is therefore hard.

An automatically generated code allows a seamless development process where the only one control engineer, or a team, designs and implements the entire system. The designer therefore focuses on the controlled object from the beginning to the end of the development and the implementation issues as MCU HW, the programming language, the scheduling policy and the other non-functional aspects remain in the background [13]. All the implementation issues are covered by the code generator target developed by the real-time and MCU specialists as a support for the control engineers work.

The quality of the generated code is comparable to the hand-written code, it is readable, the development time is shorter and possible error sources are reduced.

The rapid application development approach does not bring only the automatic code generation. It is a model based development method supported by a tool chain covering entire “V” model development chain. The validation of each development phase is done by the simulation in the Matlab Simulink. First “Model in the Loop” validates the model of the controller. After the code generation, the “Processor in the Loop” simulation can be used to validate the real-time execution of the controller on the MCU in the loop with the plant model in Simulink. Then the “Hardware in the Loop” simulation can be used to validate the entire control unit. All these phases can be supported by Simulink and the corresponding code generator target. The results of each experiment are used to continuous improvement of the Simulink model that remains still the actual documentation. Contrary to the hand-written code, there is no gap between the model and the implementation.

3. Code Generation in Matlab

The C code for a rapid prototyping is generated by the tool Real-Time Workshop (RTW) [19]. The add-on RTW Embedded Coder [16] is used for the highly optimized production quality code. The tool StateFlow Coder is used for the code generation from StateFlow charts.

Besides these tools, the platform dependent target is needed [20]. The platform means a specific MCU, an operating system (or none for a bare board) and development tools (compiler, linker etc.). The target, except other, defines the language (C/C++), details about the MCU (8/16/32bit, little/big endian ....), and it calls the development tools. The target intended for the real-time execution of the model defines the infrastructure deploying the generated code to bare board interrupts or operating system tasks.

An inseparable part of each target is a block set – a library of Simulink blocks representing the functional components of the targeted platform. A block set usually contains blocks interfacing the HW peripherals, the operating system and the communication services etc. Each of these blocks, implemented as an s-function [21], defines its simulation behavior and provides a user interface for the parameters setting. The code generated by RTW for each of these blocks is defined by a Target Language Compiler (TLC) script in a tlc file [22].

During the code generation, a code is generated for each block in the model according to the corresponding tlc file. These codes are combined according to the data flow in the model. Finally a make file is generated from the predefined template; the code is build and the executable application is downloaded to the development board. There are several points in this process, where user defined hooks can be called. This mechanism is used for cooperation with the target specific development tools.

3.1. Weakness of Existing Code Generation Targets

There are several weaknesses that motivate us to develop a new target based on PE.

- Only few targets exist and therefore far from all MCU families and derivates are supported.
- Each MCU target has its own block set. This fact prevents the reusability and the portability of the model using these HW specific blocks.
- The way in which the peripheral HW is handled by the generated code is predefined by the target developers and it can not be changed by the user.
- Validation of the HW settings in the time and the resource domain is missing. Each parameter changes are therefore an error prone process.
- The simulation behavior of blocks representing peripherals is trivial (pass-through)
- Block sets are based on the low level API which is not very comfortable for users.

4. Processor Expert Overview

PE [23] is a component oriented tool for the rapid development of embedded applications. Its main task is to manage the HW resources of the MCU and to allow the design at the high level. PE contains information about supported MCUs and their on-chip peripherals. The functionality of the basic elements of the embedded systems like the MCU core, the MCU on-chip peripherals etc. are encapsulated in Embedded Beans. An interface to a bean is provided via properties, methods, and events.

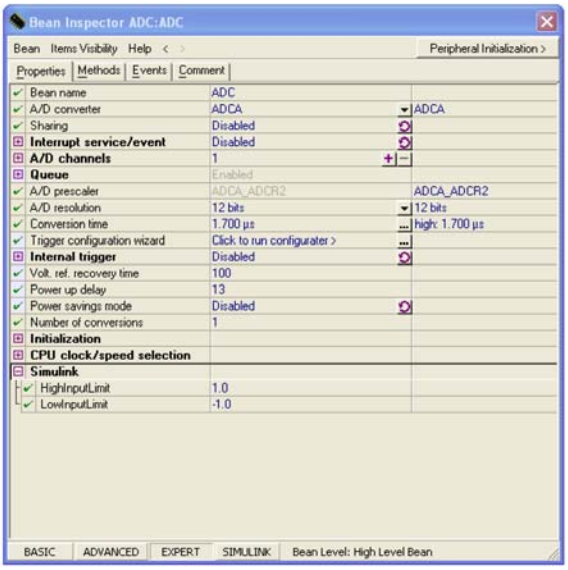
Bean properties are used to specify the HW setting at the design-time. Since it is done via well arranged dialogs of the Bean Inspector menu (Fig. 4.1), it is not necessary to study the HW details and the registers values. Some design parameters, such as settings of common prescalers or useable resources for the needed functionality are calculated by the expert system. Verification of user decisions is provided. All the on-chip peripherals are supported and all the HW features are accessible to the user; there are no predefined settings that can not be changed.

Bean Methods provide a unified interface to the user application code. The same methods on different MCUs are compatible from the application point of view. As PE covers the Freescale MCU product line, it supports application code portability. Methods code is well tested, highly optimized and scaled to the selected MCU. The generated methods code can be compliant with common standards (e.g. HIS or AUTOSAR standards for automotive software).

Bean events can be used by the user to handle interrupts.

5. Processor Expert Integrated in Simulink

It is clear from the mentioned brief description of the Processor Expert and Matlab Simulink features for code generations that they complement one another perfectly. To allow designers to use the best features of each of these tools at the rapid application development cycle, we have developed a peripheral devices block set and an embedded target for Real Time Workshop Embedded Coder, the Processor Expert Real-Time Target (PEERT) integrating Processor Expert to the Simulink environment. The development cycle with PEERT is outlined in Fig. 6.1



✓ 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	

Fig. 4.1 Bean Inspector Window

PEERT consists of three main parts - the PE block set, the PES_COM communication library and the RTW Embedded Coder target.

The PE block set contains blocks representing general peripherals such as Timers, ADC, PWM, PortIO, Quadrature Decoder etc. Each block in the Simulink model corresponds to a bean in the PE project. Each PE block is implemented as an s-function that reads properties of the corresponding bean and simulate the behavior of the corresponding peripheral.

The synchronization of the Simulink model with the PE project and the communication of both these tools through the Microsoft Component Object Model (COM) interface [24][25] is provided by the PES_COM library. The PE COM server allows the full integration of the PE functionality with Simulink. User changes in the model (PE block insertion, erasure, rename etc.) are propagated to the PE project and opposite. PE block properties are set via the PE bean inspector menu (see Fig. 4.1) that is open by a double-click on the PE block and they are therefore immediately verified by the PE knowledge base.

The PE block set supports the single model approach to the development. The model consists of two interconnected subsystems – a controller and a plant in the closed loop. The code is of course generated for the controller subsystem only. During the simulation, the PE blocks remain in the model since they have inputs/outputs for signals from/to the plant model. The advantage of the single model approach is that it is not necessary to create one model for the simulation (without peripherals blocks) and the second (without plant) for the code generation.

During the simulation, the PE blocks do not simply pass the data from/to the plant to/from the controller through, but reflects the main HW properties. For example, the ADC block representing the 12 bits AD converter on the MCU chip really provides the controller model with values with the 12 bits resolution, even though the data type of the input signal from the plant model is double and the data type of the output signal to the controller model is uint16.

The majority of the PE blocks represents beans with events corresponding to the hardware interrupt (e.g. “end of conversion” in the case of ADC). The events are represented as function-call ports in the PE blocks. They can be used for the event-driven triggering of a subsystem block execution or an asynchronous change of a Stateflow chart state.

The RTW Embedded Coder target has been developed for the C code generation. It defines the code generated for each block in the PE block set (via tlc files) and the real-time execution infrastructure. Only the uniform API of beans is used in tlc files. They are therefore MCU independent.

Periodic parts of the model code are executed non-preemptively in a timer interrupt. Function-call subsystems that are executed asynchronously are executed within interrupt service routines of triggering events. The initialization is done in the main function. There can also be executed a manually written background task.

Moreover, the target manages the code generation process. peert_make_rtw_hook.m file implements hook methods called by RTW in the defined points of the code generation process. It is used for the automatic configuration of beans (it for example enables the code generation for methods used in the corresponding tlc file). The peert_make_rtw_hook.m file moreover starts the generation of beans code by PE and integrates the code generated by RTW to the code generated by PE. Then the PE project is complete and can be used for building the application.

6. Processor in the Loop simulation support

The control system development cycle is not finished by the implementation, but the validation and tuning phase must follow. The validation and tuning process can proceed on different levels of abstraction. At the earliest phases of the control algorithm development, before its implementation, its model has been validated by the simulation with the model of the controlled plant in the closed loop. This most abstract phase is called model in the loop (MIL) simulation. Since the model of the control algorithm does not contain, at this time unknown, real time properties (execution times, jitters caused by scheduling and interrupts,...), a more detail validation must be done by executing the implemented code on the targeted processor.

It is usual that the software is developed simultaneously with the hardware or even simultaneously with the controlled plant. Before the hardware and the plant are finished, the code of the implemented control algorithm can be validated by so called processor in the loop (PIL) simulation. The implemented code of the control algorithm is executed on a universal development board, the model of the controlled plant is simulated by a simulator and the input and output data are interchanged by a communication line.

The PIL simulation is provided in the real time. It shows the execution times of the implemented controller

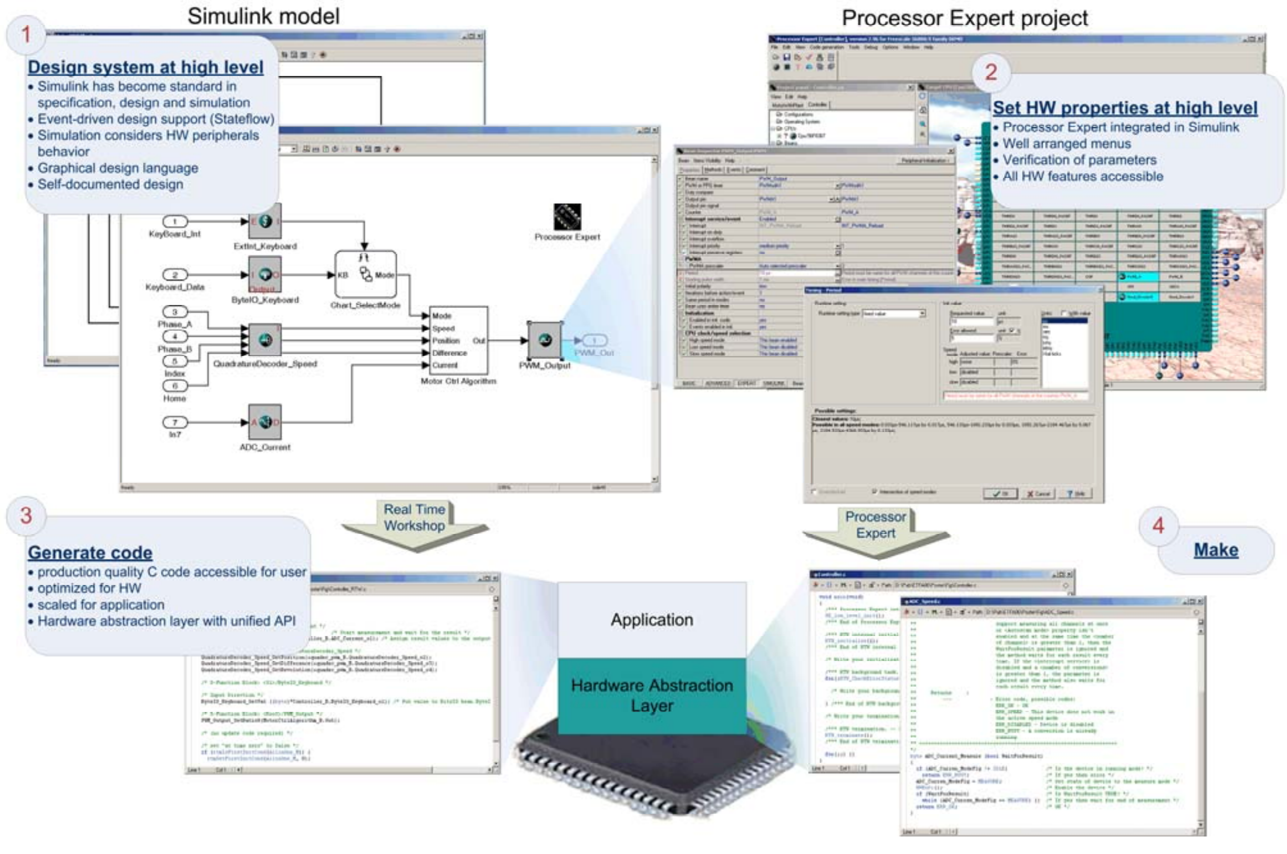


Fig. 6.1 Processor Expert integrated in Simulink

code, interrupts response times, sampling jitters, memory and stack requirements etc. It does not only provide the profiling information, but it shows the impact of the runtime parameters on the control quality. Even though the results are not absolutely precise (due to the abstract simulation model of the plant, the impact of the communication times and the minor changes in the code required for the input and output data redirection from peripherals to the communication line), they are the best can be obtained at this phase of the development. The PIL simulation can, for example, answer the question whether the computation power of the processor is sufficient and whether the scheduling parameters are chosen properly.

More precise results can be obtained by the simulation of the complete hardware of the control unite in the loop with a simulator of the plant (so called hardware in the loop simulation - HIL) or with the real plant. These approaches are applicable in final phases of the development and the final version of the code is used. On the contrary, a special version of the code is used in the PIL simulation. The inputs are not measured by the hardware peripherals but their values are obtained via the communication line, similarly the outputs are not written to the hardware peripherals but to the communication line and some interrupt service routines are not invoked by the peripherals but the communication interrupt service routine when a corresponding event is indicated by the received packet. Therefore, a support for PIL simulation is required in the code generation target. Our Processor Expert Real-Time Target supports PIL simulation. The concept of the system for the PIL simulation is outlined in Fig. 6.2.

The host PC runs Simulink where the models of the plant and the controller are developed and simulated by the MIL approach.

For the PIL simulation, the process of the code generation and simulation is automated by our PEERT_PIL target.

The PEERT_PIL generates the code of the controller subsystem by RTW Embedded Coder, compiles it and downloads it to the development board. The code generated for the peripheral blocks does not handle the peripherals hardware, but read/write the data from/to the communication buffer.

The PEERT_PIL then substitute the controller subsystem by a communication block providing a code that composes outcoming communication packets from the signals from the plant subsystem and parses incoming packets to the signals for the plant subsystem. The code for the model of the plant with the communication block is then generated by RTW for the xPC target and started on the simulator PC.

Both, the plant and the controller codes are executed in the real-time on the simulator PC and the development board respectively and they exchange the simulation data at the end of each simulation step (control period). The communication between the simulator PC and the development board is provided by RS232 asynchronous serial line. Event though the communication over RS232 is very slow, the main advantage of this interface is that it is present on any development board (an advantage over faster USB), and simultaneously, it is usually unused in the application (an advantage over CAN or SPI).

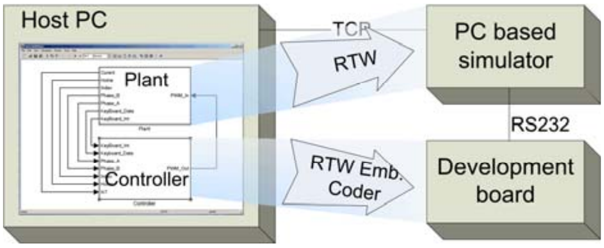


Fig. 6.2 The concept of the system for the PIL simulation

Any chosen data can by visualized on the host PC due to the fast interconnection with the simulator via Ethernet with the TCP protocol.

7. Case Study

A simple control application developed by the presented technology is briefly described in this section. The considered application is a speed control of a mechanically commutated DC motor. The motor is actuated by a power transistor switched by a pulse width modulated (PWM) signal from the MCU. The feedback is provided by an incremental rotating encoder (IRC) generating the quadrature modulated signal (100 periods of two phase shifted pulse signals A and B per rotation and one index pulse per rotation). These signals are handled by the MCU counters. A few button keyboard is used to set the speed set-point and switch between the manual and the automatic control mode. The MCU is 16-bit Hybrid Controllers (DSP and MCU functionality) MC56F8367.

The software of the application is developed as a model in Simulink. The model consists of the plant subsystem and the controller subsystem (see Fig. 7.1). This model is validated by the MIL simulation. Since the controller subsystem (see Fig. 7.2) already contains PE blocks simulating the behavior of the peripherals, the modeled input and output data has the same resolution as the hardware peripherals. It is also important to specify data types of all the signals and the parameters in the controller model. The default data type used in Simulink is double. This type is, however, not appropriate for the implementation in the 16-bit microcontroller without the floating point unite. Simulink allows choosing and validating an appropriate fix-point representation of real numbers in the controller model.

The controller code is then automatically generated by the PEERT target from the controller subsystem only. The controller subsystem must contain the Processor Expert block that must be inserted to the model as the first block from the processor expert block set.

8. Conclusions and Future Work

The advantages of using the automatically generated code in the development cycle of the control embedded software have been presented in this paper. Since Matlab is probably the most often used tool for the simulation and the control algorithm design we focused on its capability for the embedded code generation.

As the main weakness of the current Matlab facilities for the code generation was identified a poor support for handling the HW devices of the target MCU. We have presented the improvement of this situation by integrating the tool PE to the Simulink environment. The developed Processor Expert Real-Time Target for Matlab Real-Time Workshop Embedded Coder has been described.

The contribution of the presented integration is that the developed code generation target is intended for the production quality code, while most of the other existing targets seem to be more appropriate for rapid prototyping. The main advantages of the developed target with respect to other existing targets are the MCU independency of the model, the unconstrained flexibility of the peripherals setting, the possibility of the hardware setting at high level with an immediate validation of designer decisions, and the higher precision of the simulation due to the consideration of the main hardware features.

There are two variants of the block sets. In the first variant the blocks represent the PE beans while in the second variant the blocks represent AUTOSAR peripherals. The blocks of both variants are the same from the functional point of view, but they differ in HW settings and the API of generated code.

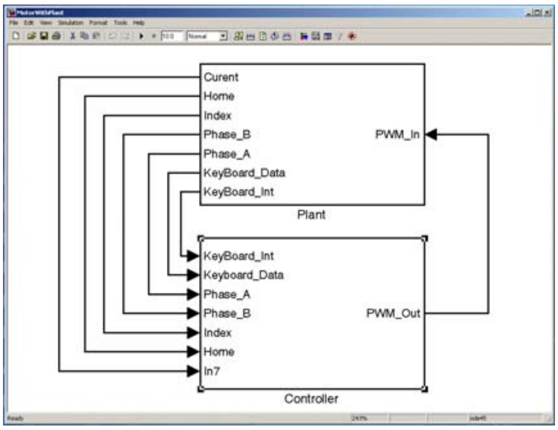


Fig. 7.1 Close loop model consisting of plant and controller subsystems

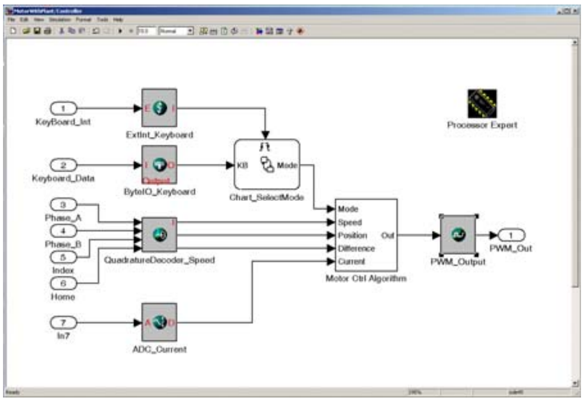


Fig. 7.2 Controller model

Currently we are working on the first industrial applications using the first variant of block set. The second variant currently contains basic blocks and we are working on its completion.

Concerning the support for the PIL simulation, we would like to develop a Linux target for the simulator. The disadvantages of the currently used xPC target are that it is closed and does not allow us to implement a support for new communications (e.g. SPI). Linux would also allow us to use a non PC hardware. Many embedded computers (e.g. based on power PC processors) offer communication and real-time facilities and sufficient computation power.

Acknowledgement

This work was supported by Academy of Science of the Czech Republic under Project 1ET400750406.

References

[1] Karl-Erik Ĺrzén, Anton Cervin: "Control and Embedded Computing: Survey of Research Directions". In Proc. 16th IFAC World Congress, Elsevier, July 2005.

[2] Dan Henriksson, Anton Cervin, Martin Andersson, Karl-Erik Ĺrzén: "TrueTime: Simulation of Networked Computer Control Systems". In Proceedings of the 2nd IFAC Conference on Analysis and Design of Hybrid Systems, Alghero, Italy, June 2006.

[3] Targetlink – Production Code Generation Guide – for Targetlink 2.0. dSPACE GMBH.

[4] Hristu-Varsakelis, Dimitrios; Levine, William S. (Eds.) Handbook of Networked and Embedded Control Systems. 2005, ISBN: 0-8176-3239-5.

[5] Torngren, M., Arzen, K.-E., Henriksson, D., Cervin, A. - Hanzálek, Z.: Tool Supporting the Co-Design of Control Systems and Their Real-Time Implementation; Current Status and Future Directions. In IEEE Symposium on

Computer-Aided Control System Design 2006. Piscataway: IEEE, 2006, p. 53-55. ISBN 0-7803-9797-5.

[6] Šůcha, P. - Kutil, M. - Sojka, M. - Hanzálek, Z.: TORSCHE Scheuling Toolbox for Matlab. In IEEE Symposium on Computer-Aided Control System Design 2006. Piscataway: IEEE, 2006, p. 50-52. ISBN 0-7803-9797-5.

[7] Törngren M., Henriksson D., Redell O., El-Khoury J., Simon D., Hanzalek Z. and Årzén K.. Co-design of Control Systems and their real-time implementation - A Tool Survey. Technical report. Department of Machine Design. Royal Institute of Technology – KTH, Stockholm Sweden. August 2006.

[8] Redell, O., J. El-Khoury, and M. Törngren (2004): The AIDA tool-set for design and implementation analysis of distributed real-time control systems. Journal of Microprocessors and Microsystems, 28:4, pp. 163.182.

[9] Hylands, C., E. Lee, J. Liu, X. Liu, S. Neuendorffer, Y. Xiong,Y. Zhao, and H. Zheng (2003): Overview of the Ptolemy project.. Technical Report UCBERL M0325. Department of Electrical Engineering and Computer Science, University of California Berkeley, CA.

[10] Simon, D., B. Espiau, K. Kapellos, and R. Pissard-Gibollet (1997): Orccad: Software engineering for real-time robotics. A Technical Insight, Robotica, Special Issues on Languages and Software in Robotics, 15:1, pp. 111.116.

[11] Keutzer K. Newton A.R. Rabaey J.M. Sangiovanni-Vincentelli A. 2000. System-level design: orthogonalization of concerns and platform-based design. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. Vol 19. No 12. Dec.2000. p1523-1543.

[12] AUTOSAR, http://www.autosar.org, 2006.

[13] Ludes R. and T. Pfund: Code Generation for Manufacturing. In proceedings of the 7th LuK Symposium, LuK GmbH & Co., April 2002.

[14] Romeo F.: Embedded Systems: the Real Story. Design Automation Conference, Las Vegas, June 20th, 2001

[15] Alberto Sangiovanni-Vincentelli and Grant Martin: A Vision for Embedded Software. Proceedings of CASES 2001, Atlanta, Georgia, November, 2001.

[16] Real-Time Workshop Embedded Coder User's Guide. The MathWorks, Inc., www.mathworks.com, 2005.

[17] dSpace, www.dspaceinc.com, 2006.

[18] ASCET, http://en.etasgroup.com/products/ascet, 2006

[19] Real-Time Workshop User's Guide. The MathWorks, Inc., www.mathworks.com, 2005.

[20] Real-Time Workshop Embedded Coder Developing Embedded Targets. The MathWorks, Inc., www.mathworks.com, 2005.

[21] Simulink – Writing S-functions. The MathWorks, Inc., www.mathworks.com, 2005.

[22] Real-Time Workshop Target Language Compiler. The MathWorks, Inc., www.mathworks.com, 2005.

[23] Processor Expert help. UNIS, spol. s r.o., www.processorexpert.com, 2005.

[24] Microsoft Component Object Model. Microsoft Corporation, msdn.microsoft.com, 2005.

[25] MATLAB – External Interfaces. The MathWorks, Inc., www.mathworks.com, 2005.

[26] OSEK/VDX, http://www.osek-vdx.org, 2006