Development Environment for Controller Design using HIL Simulation

Petru DOBRA
Technical University of Cluj
St. C. |Daicoviciu 15
400020 Cluj
ROMANIA
Petru.Dobra@aut.utcluj.ro

Florin HURGOI
Technical University of Cluj
St. C. |Daicoviciu 15
400020 Cluj
ROMANIA
Florin.Hurgoi@cs.utcluj.ro

Mihai DUMITREAN
Technical University of Cluj
St. C. |Daicoviciu 15
400020 Cluj
ROMANIA
Dumitrean@aut.utcluj.ro

Daniel MOGA
Technical University of Cluj
St. C. |Daicoviciu 15
400020 Cluj
ROMANIA
Daniel.Moga@aut.utcluj.ro

Mirela TRUSCA
Technical University of Cluj
St. C. |Daicoviciu 15
400020 Cluj
ROMANIA
Mirela.Trusca@aut.utcluj.ro

Abstract – **With the great diversity of applications, a development environment must be flexible and provide exactly the functionality necessary for e.cient problem-solving. Throughout the whole development process, it must be working with the same MATLAB/Simulink/Stateflow environment and with the same graphical user interfaces, test scripts and parameter sets. In all the development steps, there are o.-the-shelf components that make the development task more convenient than ever before. Model-based control design is a highly e.cient and widely established approach enabling control engineers to work with a single, visualized system model in an integrated software environment. The advantages are obvious: There are no friction losses between teams concerned with di.erent design steps, because everybody is working with the same model; This reduces the risk of mistakes to a minimum and shortens the development cycles. Developers and technical managers gain more transparency and control over the development projects. It is easier to observe the work status, to preserve knowledge, and to understand what other team members are currently doing.**

I. INTRODUCTION

Developing controllers for applications (electrical drive systems) means large expenditure, when performed with usual development methods. The workload comprises development of a mathematical model as well as algorithm design and implementation, off-line simulation, and optimization. The whole process has to be restarted on occurring errors or divergences, which makes the development process timeconsuming and costly.

Rapid Control Prototyping is a way out of this situation, especially if the control algorithm is complex and a lot of iteration steps are necessary [1]. Intelligent software and hardware tools relieve the control engineer from cumbersome hand coding. The need to make use of these tools grows with the complexity of the control system to develop.

The Rapid Control Prototyping tool presented in this paper is based on MATLAB/Simulink, a widely used control development software. It allows to design the controller graphically in the Simulink block diagram environment. Using the Real-Time Interface to Simulink the control algorithms are downloaded to a real-time prototyping system, which replaces the handmade prototype and executes the algorithm. This way, changes to the designed controller are much easier to perform and iteration step times are reduced to a minimum.

Hardware for Rapid Control Prototyping has to be much more powerful than the target controller, especially when complex controller functions shall be implemented. Moreover, I/O of the prototyping system ideally corresponds to the I/O of the target controller. The hardware introduced in this paper is tailored for applications in the fields of electrical drives and provides both, high computational power and a wide range of powerful I/O.

A. *dSPACE Simulator for Hardware-in-the-Loop Simulation*

With dSPACE Simulator, it can be throughly tested the finished controller. dSPACE Simulator replaces the real environment, and the tests can be executed in any conceivable test scenario, systematically and reproducibly thanks to the comprehensive test automation. This way, complicated test runs in the real environment are reduced to a minimum. dSPACE Simulator can be used immediately, together with the development PC.

No matter what the requirements I/O functionality and signal generation or extremely high real-time calculation power the modularity of dSPACE Simulator offers a full range of options.

II. HARDWARE ARCHITECTURE

A. *Processor Section and Host Interface*

The DS1103 DSP Controller Board is equipped with a TI TMS320C31 DSP for fast floating-point calculation at 60 MHz (33.3 ns cycle time, see Figure 1). This high-performance superscalar microprocessor has three integer execution units, one floating-point arithmetic unit, and a separate load/store unit for fast memory access. The on-chip 128 K x 32-bit RAM, cache size 2 K x 32-bit for instruction and data each. The processor's ability to execute instructions out-of-order leads to a performance improvement of about factor 2 for typical simulation models compared to strictly serial instruction flow.

B *I/O Section*

The board can be adapted to a wide range of closed-loop applications due to its large number of I/O devices. High-resolution A/D converters (16-bit and 12-bit) with a sampling time of 4 µs and 1.25 µs, respectively, are available, as well as D/A output channels with a resolution of 12- bit and a 4 µs settling time. In addition, the following sub-modules are provided.

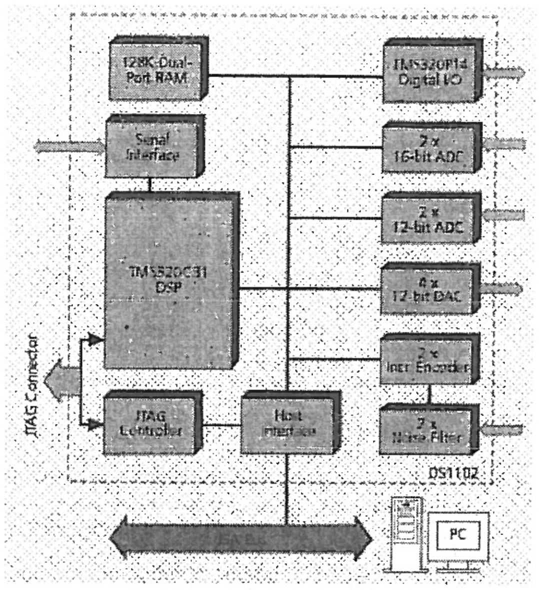


Fig. 1 DS1102 DSP Controller Board block diagram

Incremental encoder subsystem. Six digital incremental encoder inputs are supported for applications (e.g. in robotics). Digital noise pulse filtering is available. An additional analog encoder input serves as a high-resolution interface to sinusoidal incremental position sensors.

Programmable digital I/O subsystem. Based on TI's 25 MHz TMS320P14 DSP: 16 digital I/O lines (bit-selectable), Capture/compare unit with 8 channels (2 in, 4 out, 2 in/out), PWM generation on up to 6 channels (40 ns resolution) and User interrupt.

III. CONTROLLER IMPLEMENTATION FROM BLOCK DIAGRAMS

A. *Real-Time Interface to Simulink*

Using MATLAB and Simulink for modeling, analysis, design and offline simulation has become a de-facto standard for control system development. The Real-Time Interface enhances the Simulink block library with additional blocks, which provide the link between Simulink and the real-time hardware (Figure 2). These blocks cover the I/O functionality of the prototyping hardware.

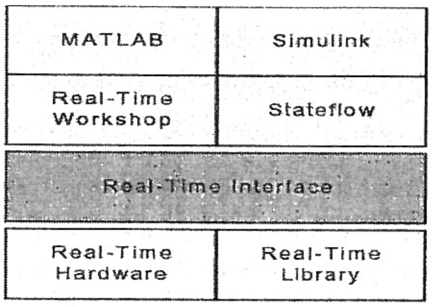


Fig. 2 The Real-Time Interface in the MATLAB/Simulink environment

To graphically specify an I/O channel the corresponding block icon has to be picked up from the I/O block library and attached to the Simulink controller model. I/O parameters, such as voltage ranges or resolutions, can be set in appropriate dialog boxes. Thus, even complex I/O devices such as incremental encoders or a CAN interface can be configured in the block diagram. For multitasking applications, a pre-emptive scheduler guarantees real-time behavior with response times of a few microseconds. Tasks and priorities are also defined graphically within the Simulink block diagram.

The Simulink model then is transferred into real-time code, using the Real-Time Workshop and the Real-Time Interface. Code generation includes the I/O channel specification and the multitasking setup, which are translated into appropriate function calls of the Real-Time Library. This library is a C function library providing a high-level programming interface to the hardware. The Real-Time Library includes access functions for the slave DSP and the CAN microcontroller, which make the interprocessor communication completely transparent to the user.

B. *Simulink Block Library for DS1102*

The block library for the DS1102 DSP Controller Board is shown in Figure 3 comprises all I/O units that are directly served by the TMS320C31. Block icons for the standard I/O channels such as A/D, D/A converters, and digital I/O are included, as well as the more complex incremental encoder blocks. An overall encoder setup block provides a comfortable means for defining the parameters of the encoder subsystem, such as counter ranges and signal types. A further block is available for obtaining encoder outputs for position measurement and delta position values for speed measurement. The remaining blocks are used for defining encoder zero positions and for index search. Apart from the master setup block, all block icons can be duplicated to handle multiple encoder channels.

C. *Multitasking Implementations*

Multitasking is supported by Simulink's capability to define the sample rate of timer-driven blocks and to set up triggered and enabled subsystems. A special hardware interrupt block is used to select one of the interrupt sources available on the board as the trigger signal for a subsystem. Figure 4 shows an example block diagram consisting of three subsystems.

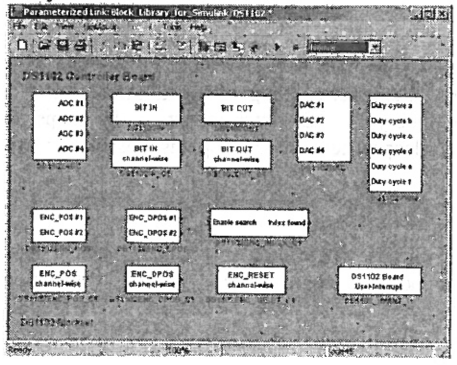


Fig. 1 Simulink block library for DS1102

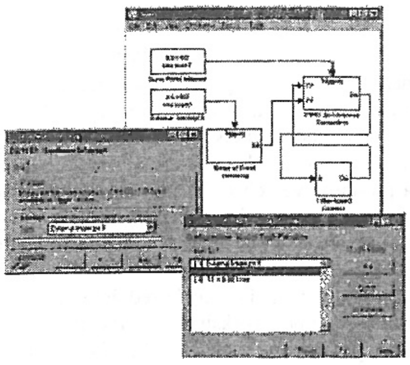


Fig. 2 Specification of a multitasking control system in Simulink

The time-based part of the control algorithm is driven at regular sample intervals with a definable step size. No external trigger input is needed for this subsystem. Another subsystem combines all parts of the controller that must be synchronized to PWM signal generation at the slave DSP. For that we use a hardware interrupt, which is generated by the slave DSP at a predefined time during the PWM signal period. A third subsystem handles external events that occur asynchronously to the normal operation of the controller. An integrated statechart specification with Stateflow might be used to describe the behavior of this subsystem. Double clicking on an interrupt block opens the dialog for selecting the hardware interrupt source for that specific trigger signal.

An important feature of the Real-Time Interface is the ability to use I/O block icons in any subsystem and, for timer-driven subsystems, with any sample rate. Clearly, the same I/O channel can be used only once. All interrupt-triggered subsystems and model parts with different sample rates are handled as different tasks. The priority setting dialog shown in Figure 4 displays all tasks of the model for assigning priorities. Timer-driven tasks always follow the rate-monotonic scheme: tasks with higher sample rates get higher priorities.

IV. APPLICATION EXAMPLE: CONTROL OF A DC MOTOR

As a typical application, the control of an a DC motor is presented for verification of the proposed techniques. The squirrel cage DC motor comprises the motor, a digital encoder for position and velocity measurement, and an DC/DC electronic converter. The DC/DC electronic converter is driven by pulse width modulated (PWM) signals from the DS1102. The whole test setup is shown in Figure 5. It is described in greater detail in [2] and [3]. The control theory of DC motor is out of the scope in this paper. A deeper view into this topic is presented in [4].

A. *System description and Signal Domains*

Main purpose of the control scheme presented here is to run the motor at a given velocity. ADC conversion and digital encoder counting are performed by the I/O devices on-board. The PWM signal generation requires high time resolution.

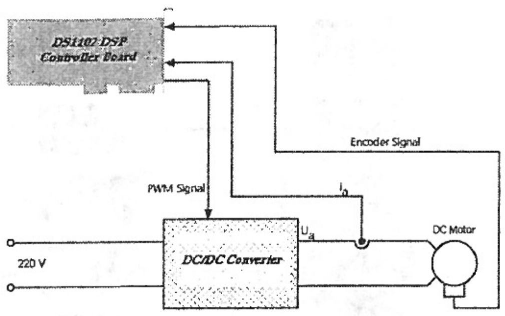


Fig. 3 DC motor control setup with DS1102

B. *Modeling with Simulink*

The system can easily be set up by exploiting some of the new features available with the latest version of the Real-Time Interface (RTI). The execution of the outmost control layer is triggered by this hardware interrupt generated at the beginning of every PWM\ period. Receiving the hardware interrupt is utilized by a standard unit block supplied with the Real-Time Interface.

Strictly speaking the application is then no longer run at equidistant times but upon arbitrary occurrences of interrupts. These however are launched periodically. The main controller block triggered by external hardware interrupt from encoder is shown in Figure 6. As no time base is available any more, the different sample rates are implemented by means of a software interrupt. A trigger signal is raised by a counter upon reaching the sample rate ratio. Then the counter is reset and the blocks sampled at lower frequencies are executed.

The output variable is write into an I/O block controlling PWM actions. Input to the main controller block shown in Figure 6 are the desired velocity speed selected by the user, the actual velocity speed $\left(\omega \right)$ and the rotor current $\left(i_{a}\right)$. All values are read in by standard I/O blocks for reading incremental encoder position and analog signal.

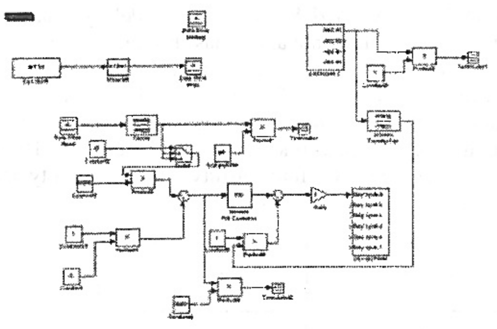
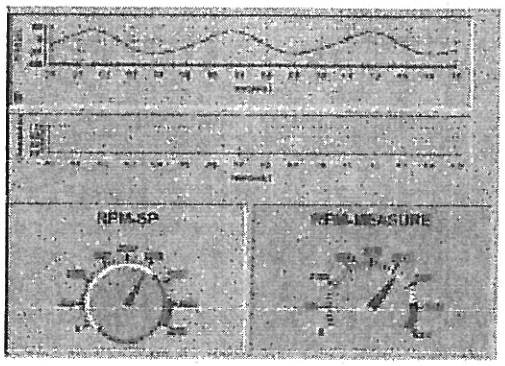


Fig. 4 Simulink block diagram of control system of DC Motor

C. *Instrumentation, Data Acquisition, and Simulation Results*

After automatic code generation for the control application, the executable is downloaded to the processor board. At this stage, all task assignments to processor are automatically made by the software. Once the code is loaded, the drive can be controlled by the real-time application. To select operation modes and adjust controller and user variables, the COCKPIT tool with an instrumentation panel as shown in Figure 7 is used.


a) Relay-based control of DC Motor

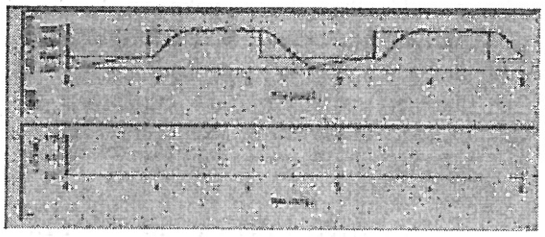

b) Step response with PID Controller

Fig. 5 Instrumentation control panel and data acquisition configuration

With this setup the motor can be run with several signal forms as well as with arbitrary static values. The motor velocity is observed among other properties, and main controller parameters are accessible for adjustment. For data acquisition, the TRACE tool is used as also shown in Figure 7. It allows for data captures of arbitrary model data in real-time utilizing trigger functions, and time plotting as well as trajectory plotting capabilities. In the given trace capture, a comparison of desired and measured drive velocity are shown.

V. CONCLUSION

Rapid Control Prototyping for fast drive systems requires a hardware with high-performance floating-point processor and an optimized range of I/O devices. The system presented in this paper provides both. The computing power of the DSP allows for the calculation of large-scale floating-point control algorithms in real-time. Incremental encoder interfaces and PWM outputs make the board a powerful tool for Rapid Control Prototyping especially for electrical drive applications.

New control strategies are designed in the MATLAB/Simulink environment. The Real-Time Interface provides additional Simulink blocks for the connection of I/O channels to the controller model. This allows to implement the new controller on the prototyping hardware. The real-time code for the complete system, including I/O functions, is automatically generated by means of the Real-Time Workshop and the Real-Time Interface. No hand-coding is required. Time for implementation and test of the new algorithm is minimized.

The paper presents a drives application, in which a PID controller (tuning with relay-method) for a DC motor is implemented and tested. The control algorithm runs on the DSP (TMS320C31) and contains parts, which have to be synchronized with the I/O signals. These I/O signals are processed by the slave DSP (TMS320C14), which in addition generates the necessary interrupts for synchronization.

VI. REFERENCES

[1] Hanselmann, H. DSP in Control: The Total Development Environment. International Conference on Signal Processing Applications and Technology, Boston, MA, 1995.

[2] Vater, J. The Need for and the Principle of High-Resolution Incremental Encoder Interfaces in Rapid Control Prototyping. In: Proceedings of the PCIM97, June 10-12, 1997

[3] N.N. Squirrel Cage Induction Motor Control with DS1102. Application Note, dSPACE GmbH, Paderborn, 1997.

[4] Trzynadlowski, A. M. The Field Orientation Principle in Control of Induction Motors. Kluwer Academic Publishers, Dordrecht, 1994.

[5] Aakolk, G. Regelung der Asynchronmaschine im Feldschwachbereich bei Orientierung am Stator- oder am Rotorfluβ mit dem Signalprozessor TMS320C31 und Vergleich des Regelverhaltens. Studienarbeit Katalog-Nr. 63, Fachgebiet Leistungselektronik und elektrische Antriebstechnik, Universitat Paderborn, 1996.

[6] ***Honeywell*** - Engineering Manual of Automatic Controls.