Int. J. Engng Ed. Vol. 21, No. 4, pp. 596–605, 2005
Printed in Great Britain.

0949-149X/91 $3.00+0.00

Rapid Control Prototyping using MATLAB/Simulink and a DSP-based Motor Controller*

DARKO HERCOG and KAREL JEZERNIK
University of Maribor, Faculty of Electrical Engineering and Computer Science, Smetanova 17, SI-2000 Maribor, Slovenia. E-mail: darko.hercog@uni-mb.si

A rapid control prototyping (RCP) system, based on commercially available software and custom in-house developed hardware is presented. An RCP system successfully combines the well-known simulation program MATLAB/Simulink and the custom DSP-based floating point motor controller. An RCP system provides smooth and fast transition from off-line simulation in Simulink to real-time operation on the embedded motor controller. On-the-fly parameter tuning and data visualization are provided in addition to rapid code generation. The presented solution represents a powerful, versatile and portable RCP system especially suitable for educational processes as well as motor control research. This article is supplemented by an example of cascade DC motor control.

INTRODUCTION

NOWADAYS, development tools play an important role in delivering new products onto the market. Rapid control prototyping (RCP) is one of the most important technologies for speeding up the product development time. RCP provides an easy transition from the model-based control design to target implementation. RCP is used in the design stage to quickly verify designed control strategy against real-world dynamics. The key element of the RCP is an automatic code generation, which eliminates tedious and error-prone hand coding procedures and thus making it possible for engineers to focus on control system design, implementation and evaluation, rather than on time-consuming low level programming. Several companies provide RCP software and hardware solutions. Software tools like VisSim (Visual Solutions Inc., www.vissim.com), MATRIXx (National Instruments, www.ni.com), RIDE (Hyperception Inc., www.hyperception.com) and MATLAB/Simulink (The MathWorks Inc., www.mathworks.com) enable control system design using a block-diagram programming paradigm. Among them, MATLAB/Simulink is probably the best known and widely used simulation programme. MATLAB is a high-level technical computing language for algorithm development, data visualization and data analysis, while Simulink is an interactive tool for modelling, simulating and analyzing dynamic systems. Simulink's add-on product Real-Time Workshop (RTW) provides an automatic ANSI-C or ADA code generation from the Simulink block diagram. RTW does not target specific hardware, therefore, generated code can be deployed on a variety of different targets including personal computers, digital signal processors or even microcontrollers.

The open architecture of MATLAB/Simulink and RTW motivated us to accommodate this applicable RCP software to the custom developed hardware i.e. a DSP-based embedded motor controller (DSP-2 controller). The successful combination of the commercially available software and the custom-developed motor controller, described in this article, represent a powerful and versatile RCP system, suitable for motor control research as well as hands-on experiments.

Several rapid control prototyping solutions have been proposed using MATLAB/Simulink/RTW and custom or commercially available hardware, based on digital signal processors or microcontrollers. Rebeschiess [1] presented a microcontroller-based real-time control system toolbox for Simulink (MIRCOS). MIRCOS enables graphical programming and real time operation of the standard 16-bit 80C166 microcontroller using Simulink. Hong *et al.* [2], described an implementation of digital signal processing algorithms using MATLAB and Texas Instruments TMS320C30 evaluation module (EVM). Lee *et al.* [3], proposed a 'target-identical' control prototyping platform for engine control that is based on an MPC555 controller. The 'target-identical' RCP term is used to address RCP hardware that is designed on the basis of a production electronic control unit.

The MathWorks Company Inc. released a few embedded targets for well-known, industry-proven microcontrollers and DSPs such as Motorola MPC555, Infineon C166 and Texas Instruments C2000 and C6000. From among all the described solutions, only the C2000 family of TI DSP

* Accepted 2 April 2005.

contains all the necessary peripheral for AC and DC motor control. The Embedded Target for TI C2000 DSP provides the ability for implementing and validating real-time control and signal processing designs directly on TMS320F2812 and TMS320F2407 eZdsp development boards.

Lee [3] addressed three key elements that conventional RCP systems should have:

1. A powerful floating-point processor, several times faster than the target processor.
2. Different types of flexible I/O.
3. A large memory.

If the RCP system does not satisfy the given criteria, developers spend more time dealing with RCP hardware constraints than control algorithm design. When considering this, and the fact that the TI C2000 DSP family is based on fixed-point arithmetic, eZdsp boards seem to be more appropriate for 'target-identical' RCP than conventional RCP. Hanselmann [4] from the dSPACE GmbH (www.dspace.de) presents 'Total Development Environment' (TDE) for rapid control prototyping. TDE includes MATLAB, Simulink, RTW, powerful hardware based on DSPs, and an additional set of software tools for online data visualization (COCKPIT, TRACE). Controller boards like DS1104 and DS1103 are appropriate for motion control and are fully programmable from the Simulink environment. Such large-scale RCP systems are very powerful and suitable for applications where functionality has precedence over price, such as in the research area.

There are, however, also many applications where such state-of-the-art solutions are sometimes unnecessary. In the education process, for example, less efficient, cost-effective and portable RCP solutions are welcome. RCP systems for educational purposes must also be as simple to use as possible. If so, students can focus on control system design and verification instead of learning how to handle an RCP system. Such RCP systems are hard to find on the market, therefore, institutions sometimes decide to develop customized in-house solutions, like the solution presented in this article.

MOTIVATIONS

A few years ago it was very hard to find a motor controller based on a digital signal processor (DSP), with the desired peripheral, performance and floating point arithmetic. Therefore, at the Faculty of Electrical Engineering and Computer Science (FERI), University of Maribor, a decision was made to develop a custom DSP-based motor controller that could be used for research in motor control. The so-called 'DSP-2 controller' (Fig. 1) [5] was developed. The key components of the DSP-2 controller are the TI TMS320C32 floating point processor which is used for control algorithm execution, and the Xilinx FPGA of the Spartan family, which implements the pulse width modulator (PWM), and the peripheral interfaces (Fig. 1). In addition, the DSP-2 controller contains all the necessary peripheral, for AC and DC motor control i.e. A/D and D/A converters, 3-phase pulse width modulator (PWM), an optically isolated digital I/O, interface for incremental encoder, RAM, FLASH ROM and CAN controller (Fig. 1). Technical details of the DSP-2 controller are summarized in Table 1. Although the DSP-2 controller was initially developed for the torque, speed and position control of the AC and DC motors, it can also be used for general purpose

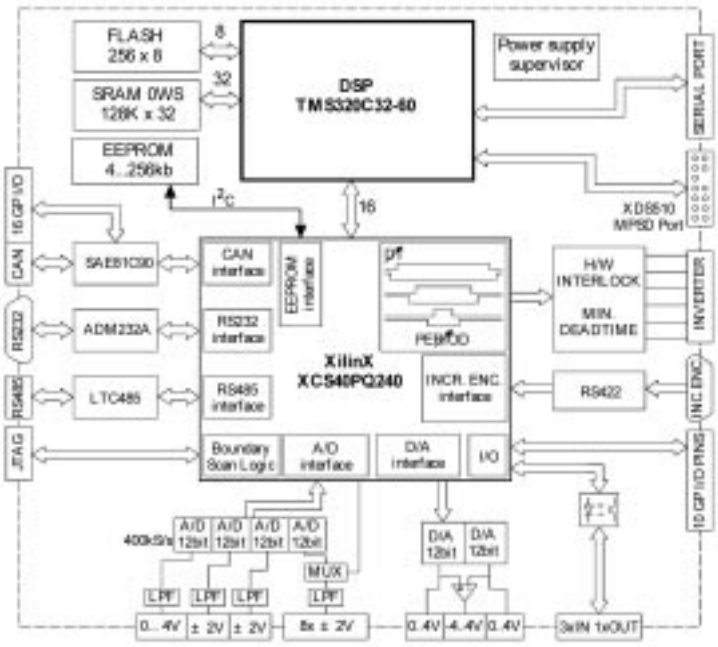

Fig. 1. DSP-2 controller block diagram.

Table 1. DSP-2 controller technical details

DSP	Texas Instruments TMS320C32; 60 MHz;
FPGA	Xilinx XCS40PQ240
SRAM	128Kx32
FLASH	256Kx8; 70ns
Analog inputs	4 × 12 bit simultaneous A/D converters: • conversion and transfer to FPGA registers takes 2.6 μs for all A/D channels • 1 × A/D with a unipolar input range from 0 to 4.095 V • 2 × A/D with bipolar input range from –2.048 to 2.047 V • 1 × A/D with an input 8/1 multiplexer and bipolar input range
Analog outputs	2 × 12 bit D/A converter with unipolar output 0 to 4.095 V
Digital inputs	3 × optically isolated
Digital outputs	1 × optically isolated
PWM	3 × synchronous symmetrical PWM with 66 ns time resolution
Encoder	1 × incremental encoder. Speed measurement with improved MT method
Communication	RS232, RS485, CAN

applications. Currently, DSP-2 controllers are used in the research of motor control, teaching, and also a few of them in industry.

At the outset, programming of the DSP-2 controller was only possible in C and assembler programming languages and, therefore, a lot of effort was needed to implement control algorithms. Almost all the control algorithms had been previously simulated and verified in the MATLAB/Simulink. After successful simulation, a tedious, time-consuming and error-prone hand coding procedure was necessary to implement the designed control algorithm on the DSP-2 controller. On the other hand, the Simulink add-on product 'Real-Time Workshop' generates architecture-independent and optimized C code from the Simulink block diagram. In order to avoid the stated problems with text-based programming, the decision was made to take advantage of MATLAB, Simulink and RTW, and apply this well-know RCP software to the DSP-2 controller. The result of this challenging task would enable automatic binary code generation from the Simulink block diagram and automatic deployment of the generated code on the DSP-2 controller. If so, the RCP system would reduce implementation time of control or signal processing algorithms on the DSP-2 controller. RCP system would also be very suitable for educational purposes because a deep knowledge of DSP programming would be unnecessary.

In general, Real-Time Workshop [6, 7] generates two types of C codes: (1) generic C code and (2) embedded C code or production code. The latter is much more optimized in performance and space usage and, consequently, it is more useful for embedded targets, such as digital signal processors and microcontrollers. In spite of some restrictions with the embedded C code (only the discrete Simulink blocks can be used in the Simulink model), and some additional work that must be done to develop the custom embedded target (for each custom Simulink block a corresponding TLC file must be written [8]), a decision was made in favour of this type of C code generation.

Figure 2 shows the code generation process from the Simulink model [6]. The Real-Time Workshop Build procedure converts the Simulink model into the model description file (model.rtw). In order to create a target-specific application, Real-Time Workshop requires a template makefile (system.tmf) that specifies the appropriate code generation tools (compiler, assembler, linker) and options for the build-process. During code generation, Real-Time Workshop transforms the template makefile into a target makefile (model.mk) by performing token expansion, specific to a given model. Afterwards, the Target Language Compiler (TLC) [7] generates C code based on target files from the model description file. Target files instruct the Target Language Compiler, how to generate the code for each Simulink block. At the end, make utility is invoked into the code generation process. The make utility automatically determines which pieces of a generated C code need to be recompiled, and issues commands to recompile them. Finally, the generated executable code is downloaded to the target processor (microcontroller, digital signal processor, PC or any other processor).

DSP-2 LIBRARY FOR SIMULINK

Figure 3 presents the result of the set task, the so-called 'DSP-2 Library for Simulink'. This library integrates MATLAB/Simulink and RTW with the DSP-2 Controller. The DSP-2 Library for Simulink is a Simulink add-on library that provides blocks specific to the DSP-2 controller. The library is composed of a few subsystems, the most important being the DSP-2 device driver blockset (Fig. 4). This subsystem contains input and output blocks (DSP-2 blocks), specific to the DSP-2 controller that enables programming of the DSP-2 controller using the Simulink. The DSP-2 device driver blockset (Fig. 4) includes blocks for all available I/O ports of the DSP-2 controller; including blocks for analog I/O, digital I/O, 3-phase pulse width modulation (PWM), incremental encoder, memory read/write, serial and CAN communication, and few transformation blocks.

DSP-2 blocks have been created using the system functions (S-functions) [9]. S-functions are

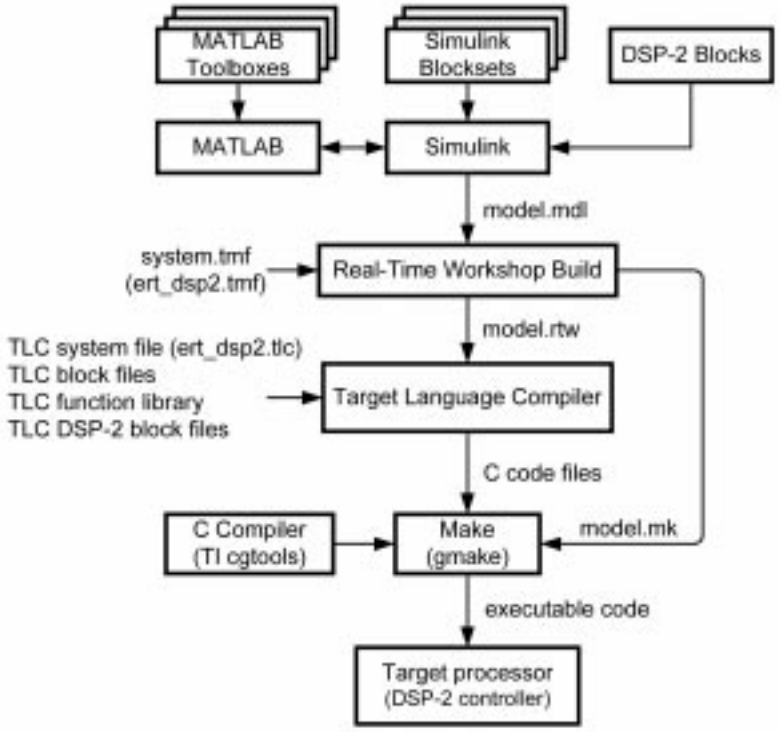

Fig. 2. Real-Time Workshop code generation process.

powerful mechanism for creating custom Simulink blocks. Each of the developed DSP-2 blocks has its own mask window where parameters can be set, specific to each individual block. The user's manual [10] explains the meanings of each DSP-2 block in greater detail.

In order to accommodate the Real-Time Workshop code generation process (Fig. 2) to the DSP-2 controller, a system target file (ert_dsp2.tlc), template make file (ert_dsp2.tmf) and target files (TLC files) for all DSP-2 blocks had to be developed (Fig. 2). The DSP-2 controller is based on TI DSP, therefore, TI code generation tools (compiler, assembler and linker) for the C3x4x family of the digital signal processors are invoked by the make utility during executable code generation (Fig. 2).

Fig. 3. DSP-2 Library for Simulink.

DSP-2 blocks are meaningless during simulation execution. The main reason for this kind of operation is that Simulink is incapable of accessing the DSP-2 controller peripheral (DSP-2 controller and the PC are connected via the RS-232 serial connection) while performing simulation. The essential applicability of DSP-2 blocks comes to be expressed in a code generation process. When an executable code is generated from the Simulink model and deployed on the DSP-2 controller, the DSP processor placed on the DSP-2 controller actually performs reading from and writing to the DSP-2 controller peripherals, depending on the DSP-2 blocks used in the Simulink model.

Code generation and deployment processes are fully automatic and transparent to the user. After executable code download, an algorithm is executing in an interrupt service routine (ISR), where the period of ISR depends on the *Fixed Step* parameter used in the Simulink model. Lower priority tasks are performed outside the ISR, such as communication between the DSP-2 controller and PC. In addition to automatic code generation, the DSP Terminal [11] running on the host PC provides data visualization and parameter tuning. The DSP Terminal (Fig. 5) is a stand-alone programme used for binary code download, data visualization, online parameter tuning, and data logging. A unique feature of this programme is an automatic front-end creation capability. The appearance of the DSP Terminal front-end

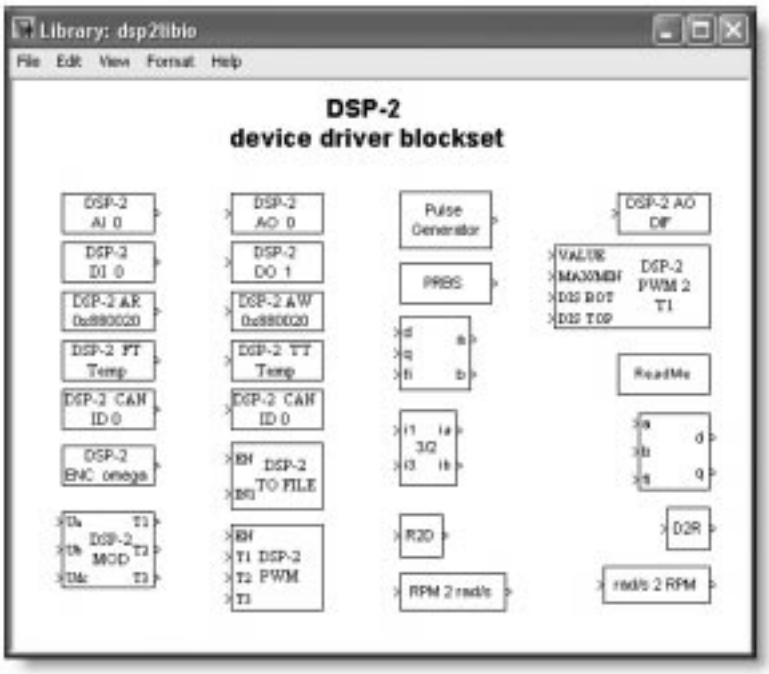

Fig. 4. DSP-2 device driver blockset.

depends on the DSP-2 global signals defined in the Simulink model. Those signals are exchangeable between the DSP-2 controller and the host PC and have to be defined using special blocks provided in the DSP-2 device driver blockset.

When the binary code starts executing on the DSP-2 controller, the DSP Terminal retrieves the DSP-2 global signals definitions from the application running on the embedded controller. For each of the DSP-2 global signals, a numerical control or

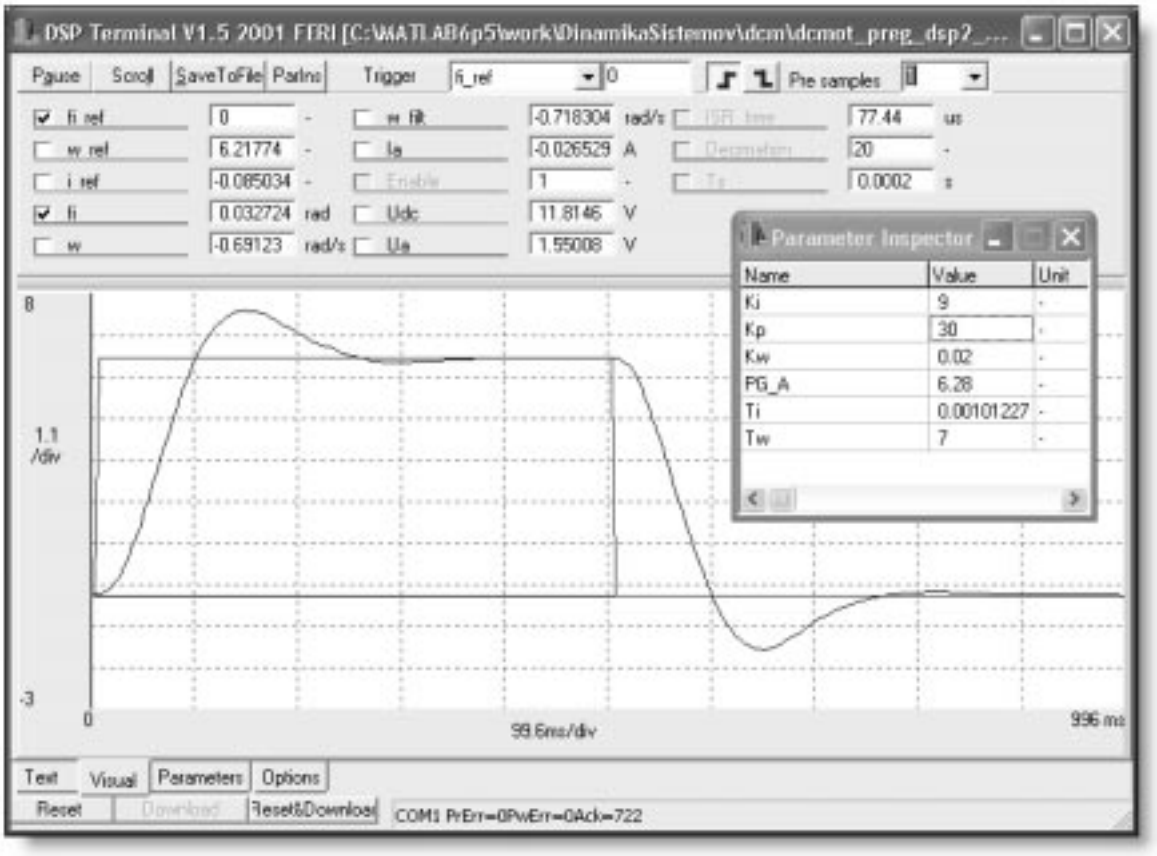

Fig. 5. DSP Terminal with the *Parameter Inspector* window.

numerical indicator is automatically created on the DSP Terminal's front-end. After Terminal GUI creation, the communication link between the terminal front-end controls and the DSP-2 global input signals are automatically established, as well as the connection between the terminal front-end indicators and the DSP-2 output global signals. Whenever the front-end controls are changed, the DSP Terminal automatically downloads them to the DSP-2 controller. Vice versa, the DSP-2 global output signals, retrieved from the DSP-2 controller, are displayed in the terminal front-end indicators. In addition to the described textual mode, the DSP Terminal provides scope capabilities. In the scope mode, a small portion of code running on the DSP-2 controller handles data acquisition and storing management. The selected DSP-2 global signals are, firstly, captured and then stored in the temporary controller memory. After that, the captured data is transferred to the PC and graphically presented in a single graph placed at the bottom side of the Terminal (Fig. 5). The DSP Terminal front-end enables selecting signals to be captured, number of samples, decimation and trigger settings that include defining the trigger signal, trigger level, slope, and the number of pre-samples (Fig. 5).

In addition to data visualization, DSP Terminal provides online parameter tuning. The selected parameters of the Simulink blocks appear in the *Parameter Inspector* window of the DSP Terminal (Fig. 5). These parameters are changeable on the fly, thus, fine parameter tuning of the designed controller can be achieved.

RAPID CONTROL PROTOTYPING IN AN EDUCATIONAL PROCESS

Initially, the DSP-2 controller was mainly used for researching motor control. Thanks to RCP software support it has also become appropriate for educational purposes. The DSP-2 learning module (Fig. 7) has been developed from a desire to offer students a powerful and universal learning system. The learning module is composed of the DSP-2 controller and an additional board, where the power supply and expansion connector take place, for important DSP-2 I/O signals. The DSP-2 learning module is versatile, light and small, handy, and an easy to use learning system. In combination with a laptop computer it represents a mobile rapid control prototyping system appropriate for hands-on experiments or in-class demonstrations. The developed learning module is plant flexible because a variety of in-house developed plants or plants from different manufactures can be connected to the module through an expansion connector. Several DSP-2 learning modules have been developed so far, the majority of them are used at the Faculty of Electrical Engineering and Computer Science, University of Maribor, whilst

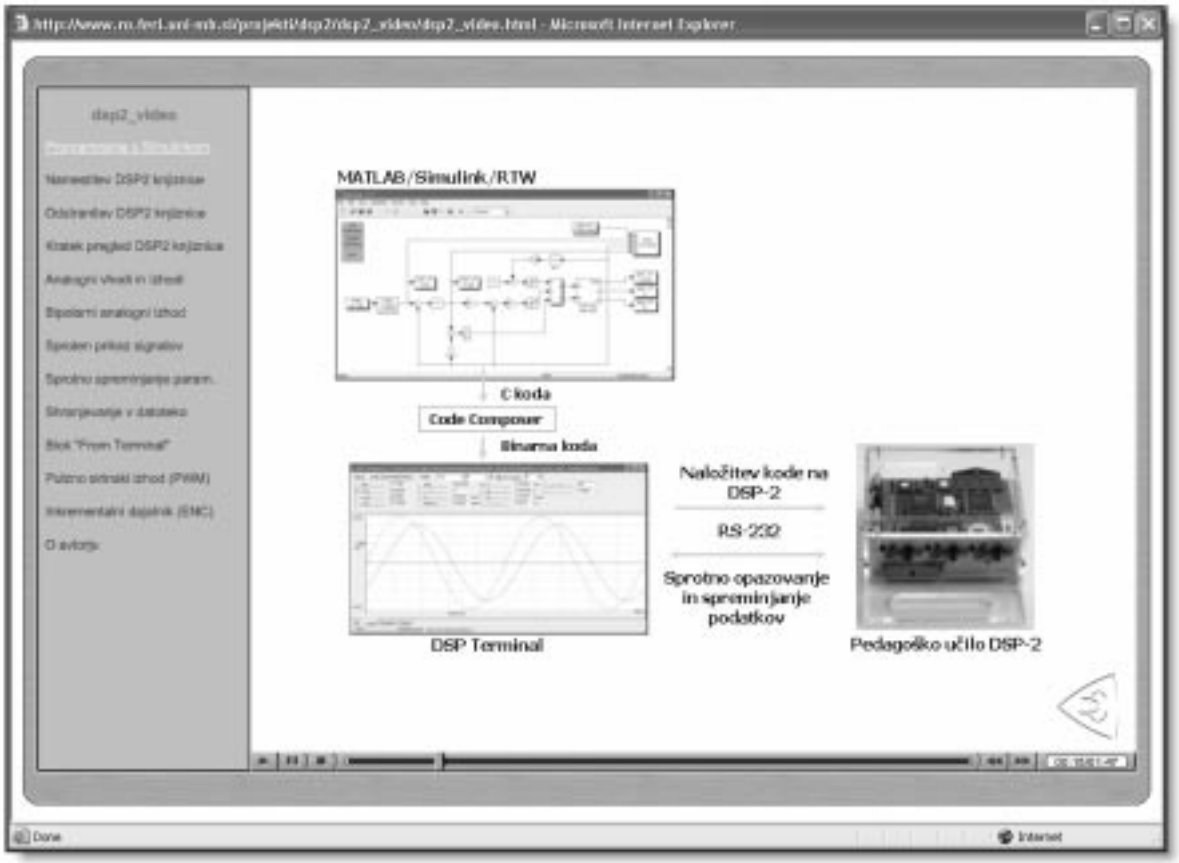

Fig. 6. Multimedia material of the DSP-2 Library for Simulink.

the remainders are employed at other universities around the world.

This mobile RCP system has been used for over two years on two different control courses. In an introductory control course students become familiar with the basic control design concepts and intuitive Simulink block-diagram programming. During the course, students work on control stability, bode plot, root locus and state space controller design [12]. In a second control course named 'Servo Systems' students are introduced to the basic concepts of DC and AC servo systems.

In both courses students start experimental work by building mathematical models of the real plant. After mathematical model derivation, they work on theoretical control algorithm design and perform closed-loop simulation in Simulink. When the simulation results satisfy the given criteria, students must also verify the designed controller, on the real system. In such a way students become aware of the strengths and limitations of the simulation tools by comparing the simulation results with those obtained from the real system. Students also become acquainted with the nonlinearities in the system, like saturation, which appear in the real world and are usually unconsidered in simulations. Using the RCP system, they can focus on control system design, simulation, implementation and evaluation of the designed controller, and not on those tasks that are not a requisite of the control course (like low level controller programming). It was noticed, that students learn faster and show more interest in control development design when they can immediately observe the results of their work.

Nowadays, multimedia plays an important role in delivering topics to the end user. Sometimes, using multimedia, difficult to explain content can be easily presented in a very informative and illustrative way. Following these new learning trends, screen capture movies (Fig. 6) with a total length of approximately 45 min's have been created in order to give students an easy transition from simulation in Simulink to the real-time operation on the DSP-2 controller. The basic principles of operation are explained, together with the contents of the majority of Simulink DSP-2 blocks. Movies are equipped with sound in the Slovene language and are accessible on the DSP-2 web page [13]. They are in Flash format, therefore, only a standard web browser with a Flash plug-in player is needed for viewing them. After examination of the movies, students gain enough information to successfully start using the described RCP system.

MOTOR CONTROL EXPERIMENT

This subsection presents the realization of the DC motor cascade control by using the described RCP system. The presented experiment is one of many that a student carries out during the *Servo System* control course. The experimental system (Fig. 7) is composed of the DSP-2 learning module, H Bridge for the DC motor (attached to the DSP-2 learning module expansion connector), and a commercially available DC motor equipped with an incremental encoder.

Figure 8 presents the Simulink model of the DC motor cascade control. This model contains a mathematical model of the DC motor (*DC motor SIM model* subsystem), current, speed and position control loops, and the position reference generator. The *DC motor SIM model* subsystem (Fig. 9), which is realized by using Simulink continuous blocks, has two inputs i.e. armature voltage (Ua) and load torque (TL), and three outputs i.e. motor

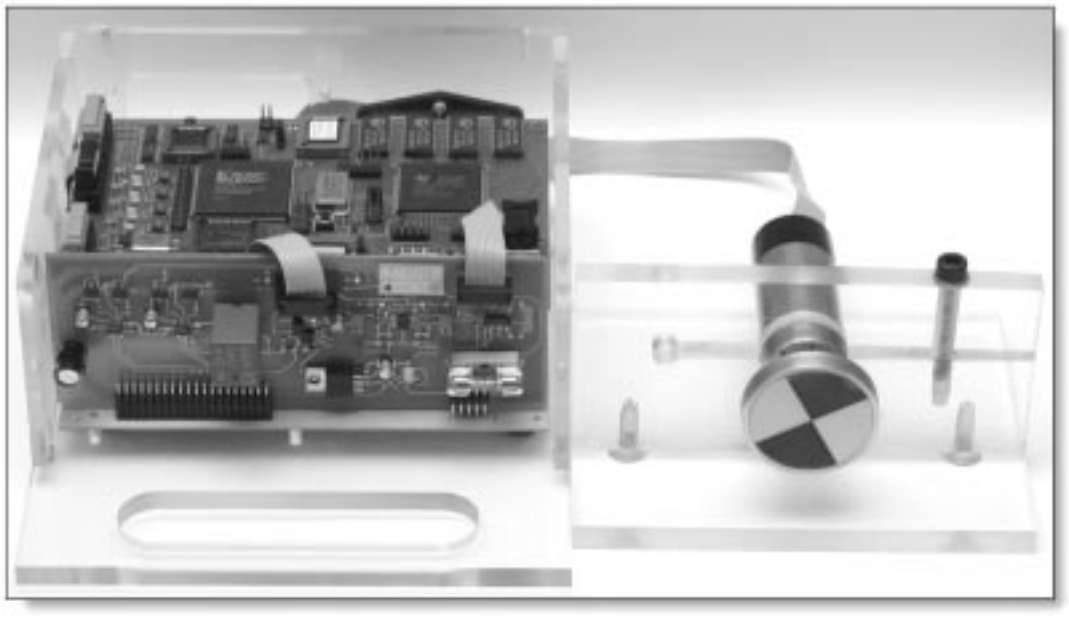

Fig. 7. DSP-2 learning module with the H Bridge and DC motor.

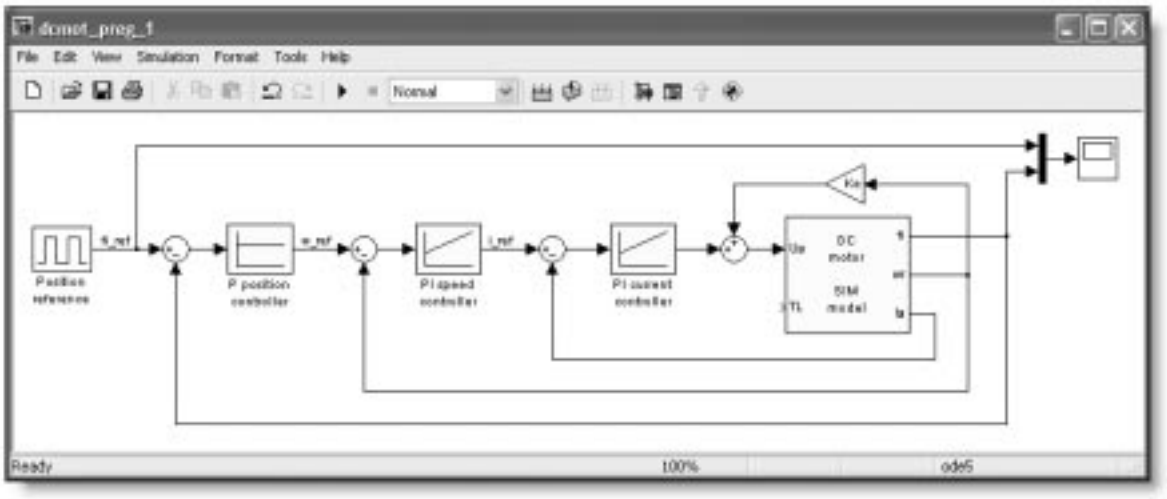

Fig. 8. Simulink model of current, speed and position control of the DC motor.

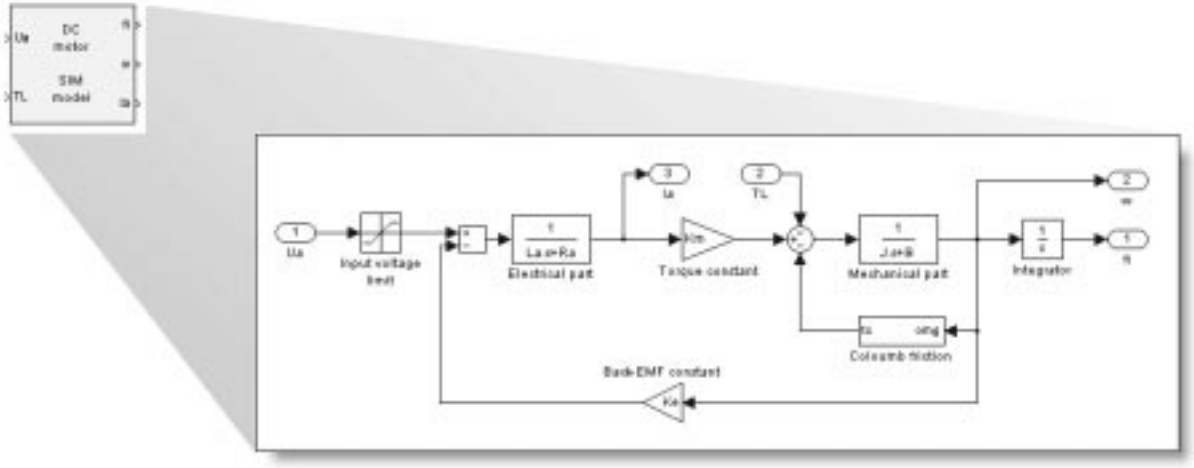

Fig. 9. Simulink *DC motor SIM model* subsystem.

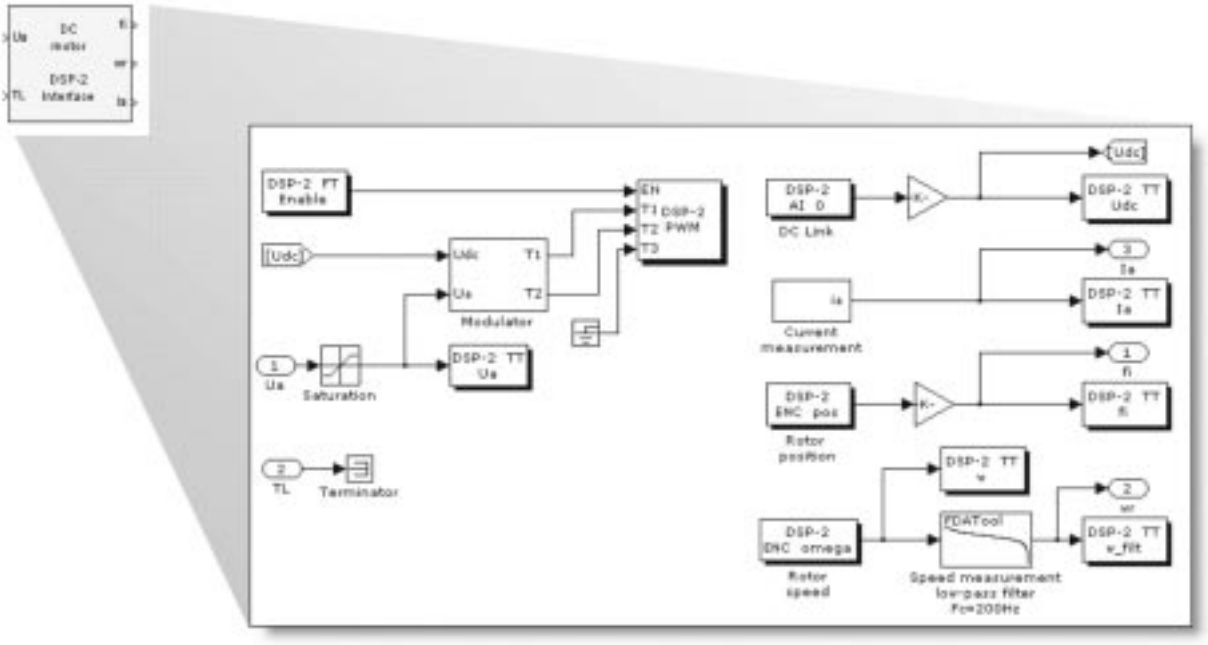

Fig. 10. Simulink *DC motor DSP-2 Interface* subsystem.

armature current (ia), speed (w) and the position (fi) of the motor shaft. During the course, students build the described simulation models and define controller parameters by using different control development methods. At each stage of the described cascade control (current, speed and position control loop), students simulate the designed controller in Simulink. After a successful step and disturbance response analysis, students also verify the designed controller on the real system (Fig. 7). In order to achieve this, only the *DC motor SIM model* subsystem (Fig. 9) needs to be replaced by the *DC motor DSP-2 Interface* subsystem (Fig. 10). This subsystem, which is pre-built, has the same input/output arrangements as the *DC motor SIM model* subsystem used during simulation. The *DC motor DSP-2 Interface* subsystem is realized by using DSP-2 blocks (these blocks are shaded in Fig. 10), and the Simulink built-in blocks. The subsystem (Fig. 10) contains an algorithm for armature current measurement, current offset compensation, the speed and position of the motor shaft calculation, speed measurement filter, PWM signal generation, and DC link voltage measurement.

After subsystem replacement and code generation, the resulting binary code is downloaded to the DSP-2 learning module. In each ISR, the DSP controller executes the developed control algorithm (Fig. 8 and Fig. 10), while outside ISR communication is carried out between the DSP and host PC. In the described experiment, the ISR period was set at 200 μs and the control algorithm execution took approximately 80 μs. The experimental results for the DC motor closed loop position response are shown in Fig. 5. On-the-fly fine tuning of position response can be achieved by changing the controller's parameters, which appear in the *Parameter Inspector* window off DSP Terminal programme (Fig. 5).

CONCLUSION

A novel rapid control prototyping system, based on commercially available software and custom developed hardware has been presented. The RCP system is powerful, flexible, easy to use and, thus, suitable for an educational process, as well as motor control research. In contrast to traditional RCP systems that are based on personal computers and data acquisition boards, this RCP system is based on an embedded target, and thus, consecutively, only a standard PC with no additional hardware is necessary for control experiments realization.

Over two years, the RCP system has been successfully used in the educational process. Control courses are now integrated with demonstrations and hands-on experiments, with the purpose of minimizing the traditional gap between theory and practice. Experience has revealed that students quickly become familiar with the RCP system and Simulink intuitive model-based programming. Students can now concentrate on control system design, simulation and experimental control verification, rather than on low level programming. By comparing simulation results with those obtained from experiments, students also gain experience with non-ideal and nonlinear features present in a real world systems.

A LabVIEW virtual instrument (VI) for the DSP-2 controller is under development. In addition to DSP Terminal features, the LabVIEW VI will enable custom GUI development, online analysis and also, by *Remote Panels* technology the possibility for 'remote' operations. A DSP-2 add-on robotic board, which is currently at the test phase, will extend usage of the described RCP system to the robotic area.

REFERENCES

1. S. Rebeschiess, MIRCOS—microcontroller-based real time control system toolbox for use with Matlab/Simulink, *Proc. IEEE Int. Symp. Computer Aided Control System Design*, August 1999, pp. 267–272.
2. K. H. Hong, W. S. Gan, Y. K. Chong, K. K. Chew, C. M. Lee and T. Y. Koh, An integrated environment for rapid prototyping of DSP algorithms using and Texas Instruments' TMS320C30, *Microprocessors and Microsystems*, **24**(7) November 2000, pp. 349–363.
3. Lee Wootaik, Shin Minsuk, Sunwoo Myoungho, Target-identical rapid control prototyping platform for model-based engine control, *Proc. IMECH E Part D, J. Automobile Engineering*, **218**(7) July 2004, pp. 755–765.
4. H. Hanselmann, Automotive control: from concept to experiment to product, *Proc. IEEE Int. Symp. Computer-Aided Control System Design*, Dearborn, MI, September 1996, pp. 129–134.
5. Čarkovič Milan, *DSP-2 User's Manual, version t3*, March 2001, Institute of Robotics, FERI Maribor.
6. The MathWorks, Inc., *Real-Time Workshop User's Guide (rtw_ug.pdf), version 5*, July 2002.
7. The MathWorks, Inc., *Real Time Workshop Embedded Coder User's Guide (ecoder_ug.pdf), version 3*, July 2002.
8. The MathWorks, Inc., *Target Language Compiler Reference Guide (tlc_ref.pdf), version 1.2*, January 1999.
9. The MathWorks, Inc., *Writing S-Functions (sfunctions.pdf), version 3*, July 2002.
10. Darko Hercog, *DSP-2 Library for Simulink User's Manual*, May 2004, Institute of Robotics, FERI Maribor.
11. Evgen Urlep, *DSP Terminal User's Manual*, May 2004, Institute of Robotics, FERI Maribor.

12. Suzana Uran, Darko Hercog and Karel Jezernik, Experimental control learning based on DSP-2 learning module, *Proc. IEEE-ICIT*, 2004, December, Yasmine-Hammamet, Tunisia.
13. DSP-2 web page: www.ro.feri.uni-mb.si/projekti/dsp2.

Darko Hercog received his B.Sc. in 2001 from the Faculty of Electrical Engineering and Computer Science, University of Maribor, Slovenia. He is currently a Ph.D. candidate in electrical engineering. His research interests include real-time systems, digital control implementation, rapid control prototyping, remote control and virtual instrumentation.

Karel Jezernik received his B.Sc. (1968), M.Sc. (1974) and Dr.Eng. (1976) degrees in electrical engineering from the University of Ljubljana. In 1976 he joined the University of Maribor and in 1985 he became a Full Professor and Head of the Institute of Robotics. His research and teaching interests include automatic control, robotics, power electronics and electrical drives. Current projects in these areas are high precision tracking control in machine tools, DD robots and robust torque control in EVs. He consults on industrial servo control systems and other control and computer applications. Prof. Jezernik is an active member of the IEEE IES.