ATEE - 2004

FIXED POINT DSP ANALYSIS AND CODE GENERATOR MATLAB FOR MOTOR CONTROL APPLICATIONS

Liviu KREINDLER, Augustin IGNAT, Daniel POPA and Aurelian SARCA

Politehnica University of Bucharest, Romania
Spl. Independentei 313, Bucharest
Tel. +40 1 413 7657, E-mail: l_kreindler@technosoft.ch

***ABSTRACT**: This paper propose a state-of-the-art development approach of digital motor control applications, which implements the developers' old dream: start with the complete system model, design the control blocks and analyse its expected behaviour by simulation, then automatically generate executable code for the target control system and perform the tests on the real system. Such an approach not only significantly reduces the development time, but also lets you focus on the application functionality and performance, thus shortening the road from design and laboratory phase up to the industrial application level.*

I. INTRODUCTION

Embedding the Matlab code generator feature in a DSP-based motor control structure offers many significant advantages, as compared with classical approaches: automatic C code generation - eliminates the need to handwrite C and Assembly code; visual modelling and simulation - selection of control structure; optimisation of control parameters for specific application aspects; analysis on the DSP system; validation of the solution on the real control environment; "plug-and-play" approach (you get a ready-to-run platform).
Using a step-by-step approach, you can change components in the system one by one – either at H/W level (motor, power converter, sensors), or at S/W level (system model, controllers, etc.). Each change can be individually tested and validated, as soon as it has been made. If any problem occurs, it will be easy to locate the error source, as normally this error will be related to the last performed change in the system structure. This will significantly reduce the time required to migrate from the initial kit to your final application, as well as the volume and level of support you might need for this process.

II. MOTOR CONTROL STRATEGY

From the general equations of the synchronous drive, we obtain the relations (1) that represent the Matlab/Simulink mathematical model used in this paper:

$$\begin{aligned} u_d &= R\, i_d + \frac{d\psi_d}{dt} - \psi_q \frac{d\theta}{dt} \\ u_q &= R\, i_q + \psi_d \frac{d\theta}{dt} + \frac{d\psi_q}{dt} \\ m &= \frac{3}{2} p \cdot (\psi_d \cdot i_q - \psi_q \cdot i_d) \\ J \frac{d\Omega}{dt} &= m - mr - F \cdot \Omega \end{aligned} \qquad (1)$$

The implementation of a digital motor control scheme requires a microprocessor able to implement all required computations, including coordinates transformations and specific control loops (as current, speed and/or position). Specific I/O interfaces are also needed, as

PWM outputs, analogue inputs for current/voltage measurement and speed/position measurement specific interfaces.

As quite significant computational power is needed, the DSP controllers offer a good device covering these features.

In this paper, the PMSM digital control scheme is implemented using a DSP controller produced by Texas Instruments. This DSP is a component of the TMS320 family, specifically designed for motion control and having a 150 MIPS computation speed.

The PMSM digital motion control scheme has the following components and features:

- the motor currents are measured using current transducers placed on inverter legs; If the three measurements and PWM signals are synchronized, then the measured motor phase currents can be used in the fast control loop.
- the motor speed is estimated from motor position using an incremental encoder. In order to keep the model simple, the speed information is obtained from the motor position difference at successive time samplings;
- the model uses two control loops:
  - current control fast loop - 100 μs ;
  - position/speed control slow loops - 1 ms;
- the mathematical computations are performed in IQMATH format;
- the fast loop control uses discrete PI controllers, and the slow one uses PID ones

The motion control strategy is presented in Fig. 1. and has the following modules:

- communication: transmit/receive the motion commands from a computer/industrial application to reference generator;
- reference generator: generate the speed/position reference of the motor;
- motor control: contains the speed/position controllers;
- current control: contains coordinates transformations, and flux/current controllers;
- PWM generator: based on voltage reference, issues the correct PWM commands towards the inverter;

The software is organized on different priority levels such as:

- command interpreter;
- reference generator;
- I/O Data Exchange - used for debugging;
- interrupts: serial communication, A/D conversions, fast/slow loops;
- I/O Interface: serial, A/D converter, PWM generator as command for inventor bloc, timers, encoder;

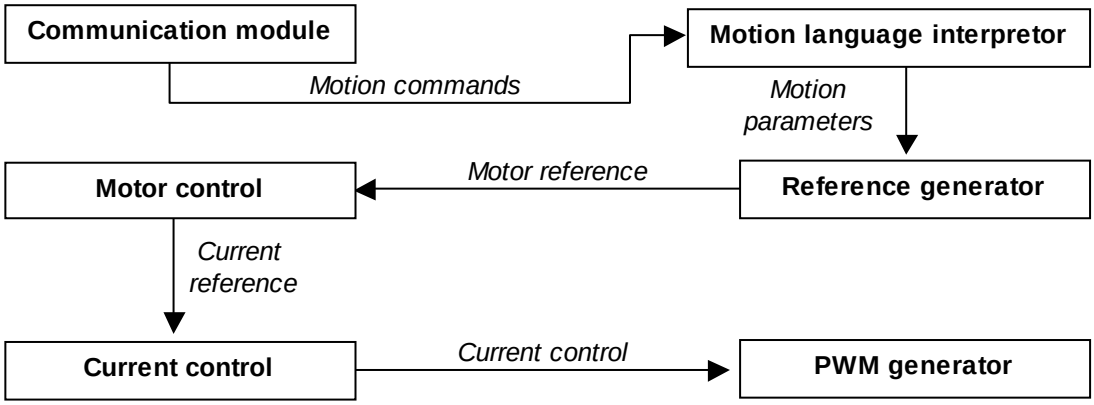


Fig. 1. Motion control program structure

III. MODELLING OF THE POSITION VECTOR CONTROL SCHEME

The general diagram of a drive system that controls the position of an electrical motor is presented in Fig. 2. To model an electrical drive system as the one presented in Fig. 2, you need to develop specific components as presented in this chapter, based on the already existing blocks in the Matlab-Simulink simulation environment.

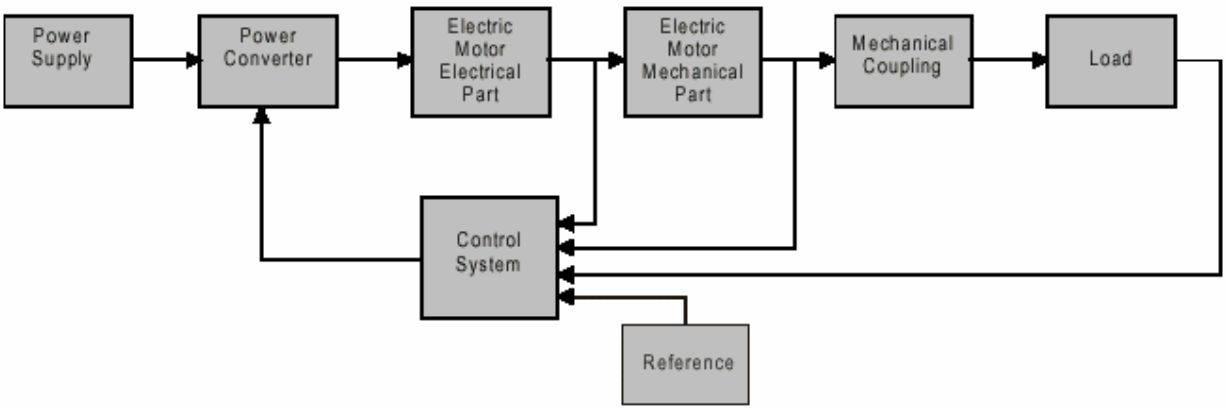


Fig. 2. Block diagram of an electrical drive system

The equivalent scheme, as implemented in the MATLAB-Simulink environment, is presented in Fig 3. This includes the analogue parts (motor, power converter, sensors), as well as digital ones (digital controllers, A/D, D/A conversions, and digital implementation aspects related to fixed point representation and computations, as used in the DSP controller).

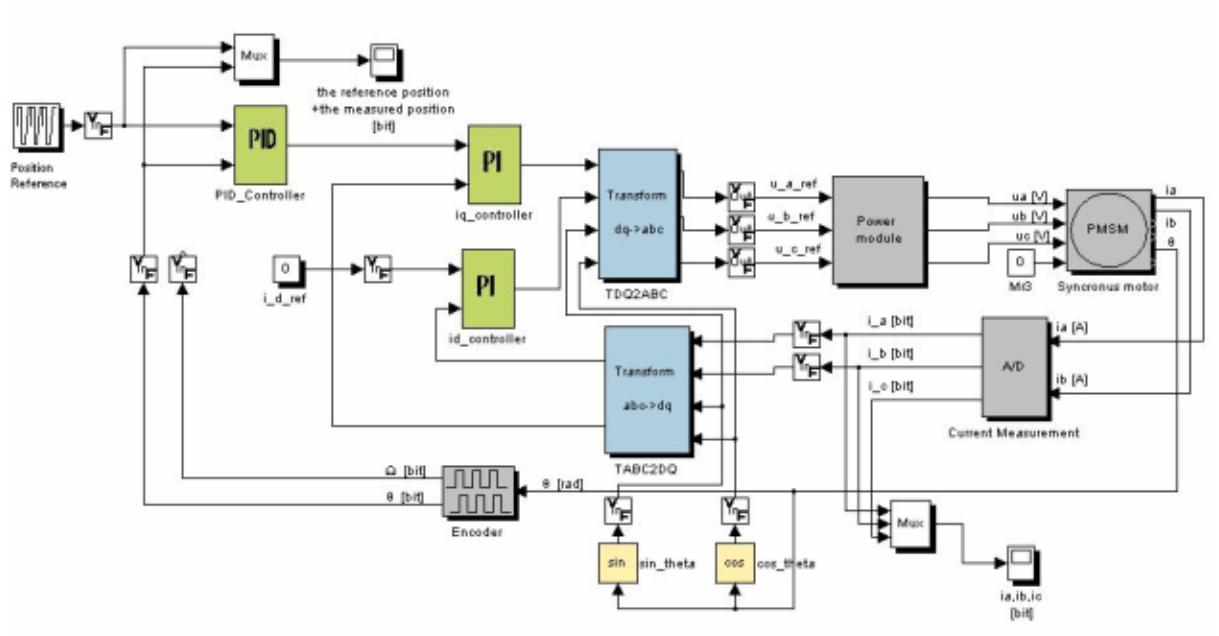


Fig. 3. Motion control scheme model in Simulink

IV CODE GENERATION

Once the system has been simulated, and you are satisfied with its expected behaviour, you can proceed to the next level: generating C/C++ code for the control blocks of the system, in order to implement and test it on the 2812 DSP controller. To do this, you can use the Real

Time Workshop tool of the MATLAB environment. It allows you to generate the complete C/C++ code associated to the blocks of the system's Simulink model.
The C code generated from MATLAB is finally included in a basic real-time interrupt application, which can be executed on a TMS320F2812 DSP controller based module. Using the Digital Motor Control Developer Pro (DMCD-Pro) IDE platform, you will be able to download and run the application on the real digital control environment.
Code generation is done in four steps if the **compile** option is selected; otherwise, there are only two steps.

1. The Real-Time Workshop analyses the **Simulink** model, and then it generates the *model.rtw* file through a hierarchical representation.
2. The Target Language Compiler analyses the *model.rtw* file and, depending on the options found in the "ti_c28xx.tlc" file, generates the corresponding C code. If you choose to uncheck the "*compile*" option, the code generation process stops right here; otherwise, the next steps follow.
3. The Target Language Compiler (TLC) compiles and builds the 'make' file, taking into account the model defined in the "ti320c28xx.tmf" file.
4. The compiling utility reads the 'make' file and compiles the C files, links the object and library files, then creates the .exe file.

The Fig. 4 shows how the code is generated by taking into account the user options:

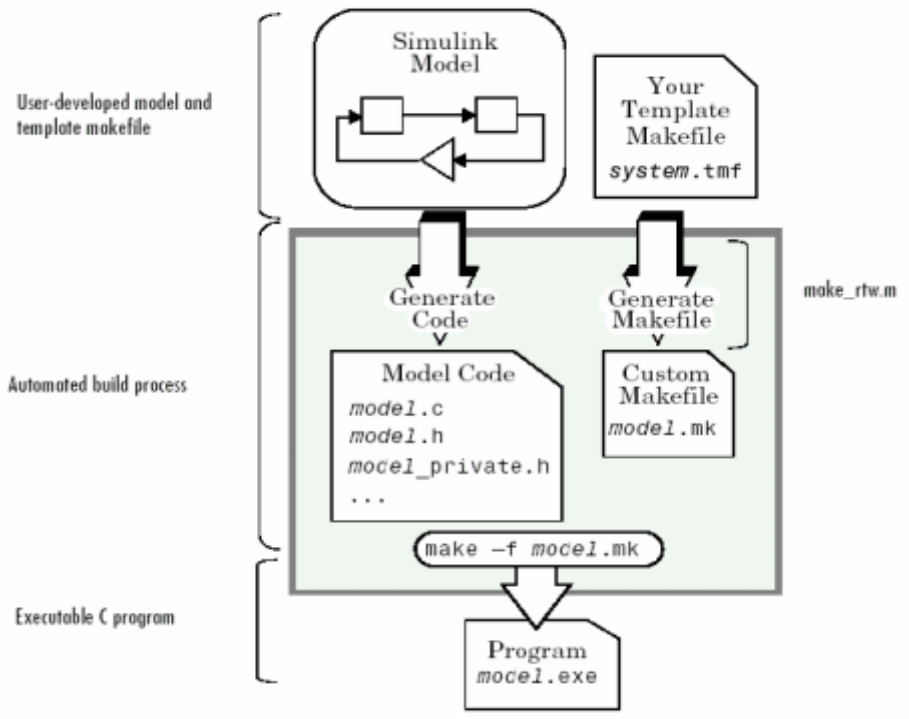


Fig. 4. Code generation steps

In the second part of the code generation process, the TLC generates the C++ code. The TLC is an interpreter created to convert the model description found in the *model.rtw* into C++ code. The Target Language Compiler executes a TLC program that contains script files. As the TLC is an interpreter, the scripts specify how the code is generated, having as input the *model.rtw* file.
Due to certain differences between TI's C compiler used by DMCD-Pro, and the MATLAB compiler, the generated C++ code is not compiled at MATLAB level. The C++ code is

embedded in the DMCD-Pro projects, compiled in these projects using TI tools and then downloaded and tested on the real-time environment of the DSP controller.

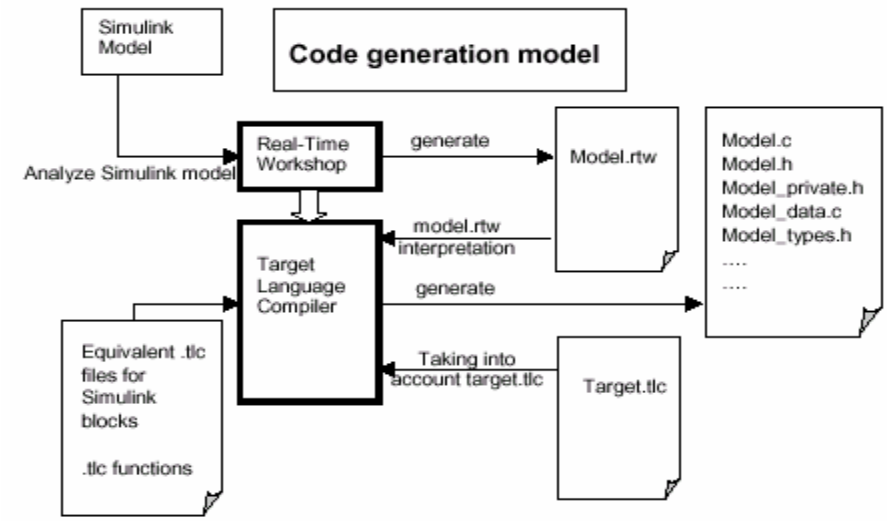


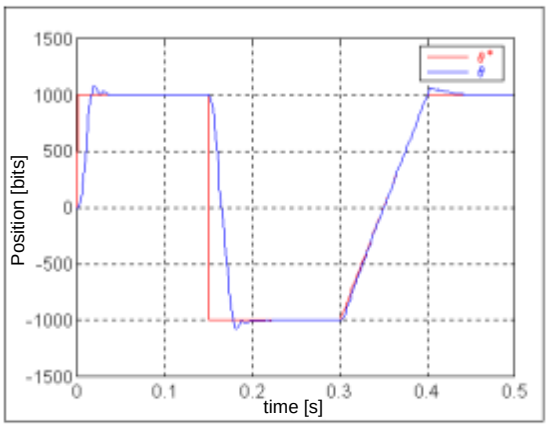
Fig. 5. Code generation model using the **Embedded Target for TI320 C28xx DSP** library

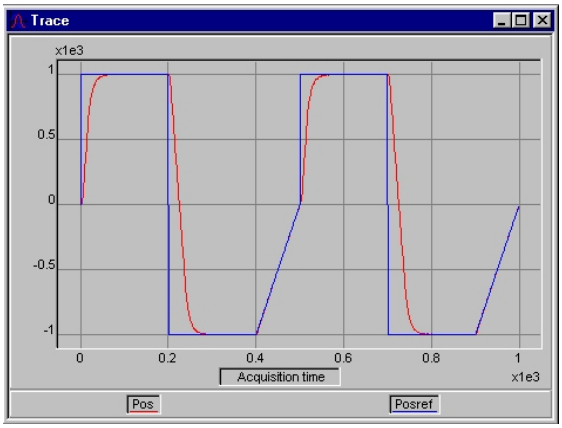
V. EXPERIMENTAL RESULTS AND CONCLUSIONS

The results were obtained using a ***MCK2812*** motor control kit and a ***Pittman 3441*** permanent synchronous motor with technical data presented below:

- phase resistance	5.25	Ohm
- phase inductance	0.46	mH
- torque constant	25	mNm/A
- rated voltage	19	V
- nomimal current	1.16	A
- peak torque	94	mNm
- rotor inertia	9.9	$kgm^2 10^{-7}$
- number of pole pairs	2	

On the first column are presented the Matlab/Simulink results and on the second column are show the experimental results obtained using DMCD-Pro.





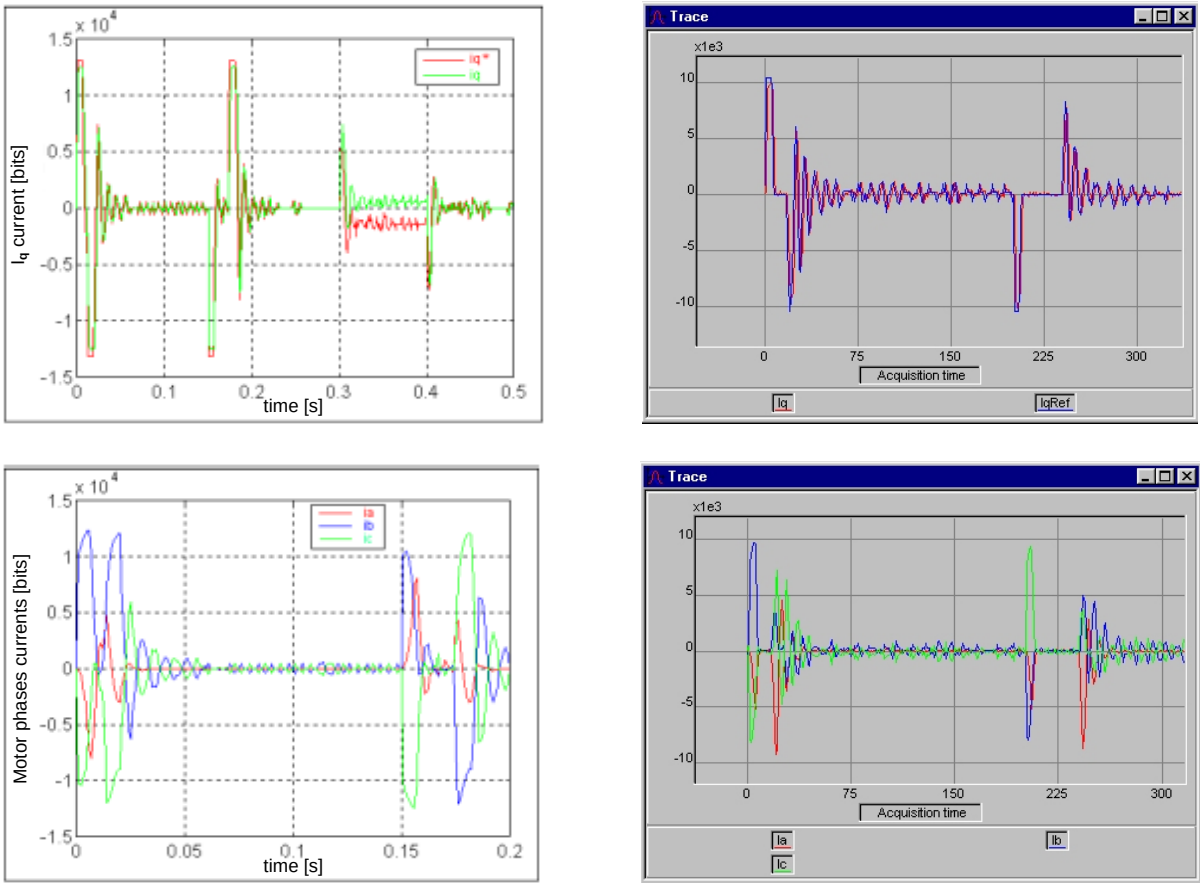


Fig. 6. Simulated (left column) and experimental results (right column)

As we can see from Fig. 6., there is a very good fit between the simulated and experimental results. This validates the proposed approach, and confirms the correctness of the solution. At this point, we can easily change specific control block in the Matlab-Simulik model, in order to eventually test other control strategies and schemes and, once simulation results seem satisfactory, test the new control schemes on the real DSP system.

The proposed solution represents an excellent platform for theoretical research and experimental validation of advanced control schemes for digital motor control configurations, and is specifically suitable for laboratory and advanced studies activities in this area.

VI. REFERENCES

[1] L.Kreindler, A.Sarca, R.Giuclea, L.Antognini, "High-level Tools for advanced Digital Motion Control Applications Implementation", SPS/IPC/Drives 2001 Conference, Oct. 2001, Nuremberg, Germany

[2] L.Kreindler, R.Giuclea, A.Sarca, V.Burtea, "High Level Graphical Tools for Fixed Point DSP Motor Control Implementation", IMCS Conference, San Jose, USA, July 1998

[3] P. Vas: "Future Trends and Developments of Electrical Machines and Drives", Power Conversion and Intelligent Motion PCIM'95 Conference, pp. 67-78, Nuremberg, Germany

[4] R. Isermann: "On the Design and Control of Mechatronic Systems — A Survey", Ind. Electronics, Feb. 1996, Vol. 43, No. 1, pp. 4-15

[5] MathWorks: "Targeting Real-Time Systems" – User manual MathWorks 2003

[6] Texas Instruments: "TMS320C24x DSP Controllers – Peripheral Library and Specific Devices",Texas Instruments 1997

[7] Texas Instruments: "TMS320F2810, TMS320F2812 Digital Signal Prossesors" – Product Overview,Texas Instruments 2000