Proceedings of the 5th WSEAS/IASME Int. Conf. on SYSTEMS THEORY and SCIENTIFIC COMPUTATION, Malta, September 15-17, 2005 (pp283-288)

Development Tool for DSP Architectures

ALEXANDRU ONEA[1] VASILE HORGA[2] MARCEL RĂȚOI[2]
[1]Department of Automatic Control and Industrial Informatics
[2]Department of Electric Drives
"Gh. Asachi" Technical University of Iaşi
Bd. Mangeron. 53A, 6600 Iaşi
ROMANIA

Abstract: - This paper presents Windows-based application, designed in order to assist a control algorithms developer in research and/or development activities. The application solves the main requirements needed in such a work. A new approach in software development was followed in order to obtain flexible and reusable DSP digital control applications. It was used a standard promoted by Texas Instruments that stimulates software modularity strategy. Experimental results show an implementation example of a modular control application using MSK243 DSP Motion Starter Kit. The software product can be used in educational process and in research as well.

Key-Words: - graphic user interface, DSP, motion control, framework application, software module, debugging

1 Introduction

Last years more and more applications were developed on DSP. A field of interest for DSP applications is motion control. A regulation concept able to incorporate the AC machines as the drive machine can only be achieved in an economically manner on a digital basis. The use of microelectronics in power control circuits enables the implementation of complex control concepts and allows the production of environmentally acceptable, self-optimized motor drives with applications ranging from computer peripherals, robotics, and machine tools to railway drive, ship propulsion, and rolling mills. Although with a mature technology and the basic problems already solved, controlled electric drives are still in the intense development phase [1].

Powerful and flexible, the DSP-based drive controllers create the potential for significant performance increase through the application of advanced control concepts. In most cases the embedded systems are endowed with minimum peripherals for user interface (LCD, keypad), thus ensuring fully autonomy to control systems. These peripherals narrow the possibilities of acquired data analysis and especially that of results presentation. The trend is to ensure a high flexibility for embedded systems, designing open architectures that allow remote control [2]. Thus autonomous digital control system concept is gradually replaced with an easily adaptable evolutionary structural ensemble. This way appears a migration from the producer-defined functionality to the user-defined functionality.

A new high-end a.c. drive provides closed loop control of motor speed, which combined with Windows-based configuration and hardware modularity plug-in industrial networking, gives the user the highest functionality. This solution provides intuitive applications programming, using a template approach with a library of function blocks that can be parameterized and interconnected easily using point-and-click operations. Moreover, many drives come with a suite of pre-programmed applications software for such applications that can be controlled via a rugged human-machine interface [3].

This background led to the idea to design and implement a software tool for behavior analysis of DSP motion control systems. With an induction motor electric drive stand and the Technosoft MSK243 DSP Motion Starter Kit, the authors widened the functionality of this ensemble developing a high-level graphical user interface. This application comprises both Windows interface powerful facilities and essential features of a software for motion control algorithms analyzing, evaluation, developing, and debugging.

2 Design Principles and Software Requirements

Digital Signal Processor (DSP) technology makes digital control more practical and also offers a high

level of performance. Traditionally, cost has been a potential disadvantage of the DSP solution, but this aspect has diminished with the continuing decline of DSP's costs. Floating point DSP's are very attractive for motion control system because their computational power and accuracy higher than the fixed point DSPs. For many applications in industry, fixed-point solutions are still preferred.

Texas Instruments (TI) developed a new controller concept with the new powerful single-chip motion controller family TMS320C24xx, which integrates a 16-bit fixed-point DSP core with intelligent peripherals to achieve a single chip solution. The DSP controller architecture is optimized for processing control signals.

To use of the available computation power, the designer is required to apply advanced level skills for the design of the necessary software, that include know-how in electric motors and drives, digital control theory, real-time control implementation etc. For a fixed point DSP, all quantities (data, constants) are represented by integers in the [-32768,32767] range. This means that each quantity must be converted to fit within this range. Special techniques, such as Q15 representation, are used to meet these requirements. The software development process is longer and difficult. Furthermore an important part of the solution must be written in assembly language to perform Q15 operations and increased execution speed. Thus the high complexity of Digital Motion Control (DMC) application requires high trained and experimented specialists. This makes the development process a non-trivial one and its complexity is high.

While in traditionally approach the software application is fully developed at programmer location, in modem approach of components-based type, only the software components are developed at headquarters; the components integration in an application framework is done in the field by system integrator. The software component is defined as a composition unity with well-defined interface by an agreement and this has explicitly presented all the context dependencies. Software components can be independently installed each other and may be aggregated with third party [5]. To obtain a software component three steps must cover: component specification, component production, and component integration into a framework application.

The component specification is considered as industrial standard. Only this stage involves certain coordination of efforts in order to introduce and to impose standards for software components on the market. Based on these public specifications each independent producer of software components can develops his own component. The integration of components in applications frameworks is the last stage of this succession. This stage means even final goal of components existence - application development in industrial manner. The software integrator must identify needed components and then purchases them from the market. The integration of components into an application is realized by means of application development framework, which represents a prefab assembling of components.

TI recognizes that such methodologies need also be applied to high performance DSP systems [6]. TI is promoting a new Algorithm standard called XDAIS (eXpress DSP Algorithm Interface Standard), to be used as backbone. XDAIS focuses on a set of general rules and guidelines that can be applied to DSP algorithms, and, if followed, allow system integrators to quickly assemble production-quality systems from one or more algorithms.

Many modern DSP systems architectures can be partitioned along the line depicted in Fig.l [6].

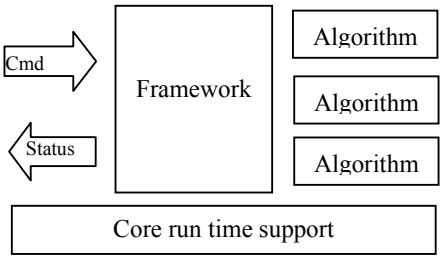

Fig.1. DSP software architecture

Algorithms are "pure" data transducers; i.e., they simply take input data buffers and produce some number of output data buffer. The framework is the "glue" that integrates the algorithms with the real-time data sources and sinks using the core run-time support, to create a complete DSP subsystem. An algorithm is compliant with XDAIS if the algorithm writer follows a set of general rules, some of them presented below:

- all algorithms must follow the run-time convention TI implementation of the C programming language (C - callable). This ensures that the system integrator is free to use the C language to "bind" various algorithms together, and to control the flow of data between algorithms.
- all algorithms must be reentrant within a preemptive system.
- all algorithms data references must be fully relocable. That is, there must be no "hard coded" data memory locations.
- all algorithms code must be no "hard coded" program memory locations.

- algorithms must never directly access any peripheral device. This is the responsibility of the client or framework.

All standard-compliant algorithm implementations must be modules, which manage instance objects. Objects simple encapsulate the persistent state that is manipulated by the other functions or methods provided by the module. A module manages only one type of object. Thus, a module that manages objects roughly corresponds to a C++ class that follows a standard convention for its configuration parameters, interface header, and all external identifiers. This is a way to get the benefits associated with object-oriented and component-based programming. Since all modules have a very clear interface and are self-contained, it is then simple to connect modules together and make them interact in a system. Thus systems typically consist in a "top level" framework with many function calls to various modules.

Embedded DSP systems are sensitive to small memory spaces and critical execution times. A combination of full "C" and C-callable assembly (CcA) functions is recommended. At the system level or framework, full "C" ensures maximum clarity, flexibility, and maintainability. At the module level, CcA functions allow memory and cycle efficient code to be developed with all the desired attributes expected from a "C" function, i.e. to be reentrant, preemptive, relocable, instanced multiple times etc.

3 A Design Example for Motion Control - Direct Torque Control Method

High efficiency control and estimation techniques for asynchronous motors found more application fields with Blaschke's well-known Field-Oriented Control (FOC) created in 1971. It was an intensive work to improve the dynamic response and reduce the complexity of FOC methods. One method is the Direct Torque Control (DTC) method developed by Takahashi in 1984 and has get increased attention due to the improved dynamic performance and simplified control strategy that it offers. More detailed information can be found in [7],[8].

The DTC method involves direct control of the appropriate/optimum switching modes and keeps the flux and torque errors within a prefixed band limit. The errors are defined as differences between the reference and measured/estimated values of flux and torque. DTC techniques require the use of hysterezis band controllers instead of flux and torque controllers. To replace the coordinate transformations of FOC, DTC use look-up table to carry out the switching procedure based on the inverter states. However, its extreme simplicity is counterbalanced by some inconveniences: torque ripple is typically higher and the inverter switching frequency is variable as a function of load torque and rotor speed.

To ensure a proper voltage selection by the DTC controller, the estimation of stator flux must be accurate. The calculation of the electromagnetic torque depends on the stator flux estimation accuracy too. Stator flux and torque estimations are based on the asynchronous machine dynamic equations in the stationary stator reference frame.

Fig.2 shows the speed DTC system. The speed controller in the diagram is a PI controller. The development of the sector selector and the switching table is based on Takahashi's study.

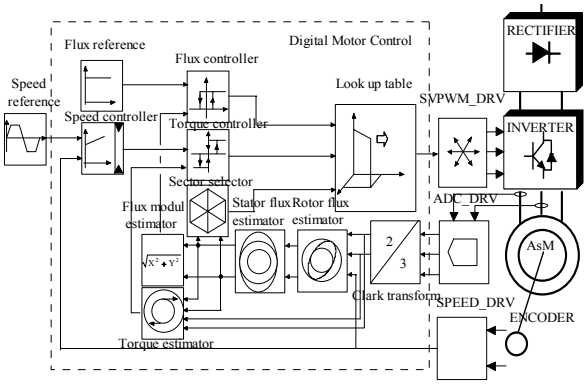

Fig.2. Speed sensored DTC system

3.1. Speed DTC Implementation using eXpress DSP Standard Concepts

In order to digitally implement the control structure shown in Fig.2 the authors have used the DSP - Motion Starter Kit -MSK243 [9]. The digital signal processing algorithms represent implementations of mathematical functions, which process "pure" data. This matter requires that the interaction with input/output information do not be directly achieved by means of hardware peripherals of DSP digital system but by means of associated software drivers. On the other hand, peripherals drivers are software components specific to available hardware resources and, therefore, together with services offered by framework make up the minimal core for any digital control structure of electric drive system.

As application structure, the configuration can be presented like a hierarchical multilayer functional structure (Fig.3):

- physical interface layer;
- logical interface layer;
- digital signal processing algorithms layer.

At the physical interface layer level the application use the embedded hardware resources of DSP controller:

- serial communication interface - SCI;
- analog to digital converter - ADC;
- quadrature encoder pulse interface - QEP;
- PWM generator unit;
- general purpose timer/counter circuits - GPTx:
- hardware interrupts system.

The next layer, of logical interface, implements the data exchange with external environment. This exchange is made by means of some data transfer functions and memory buffers for reading/writing obtained data.

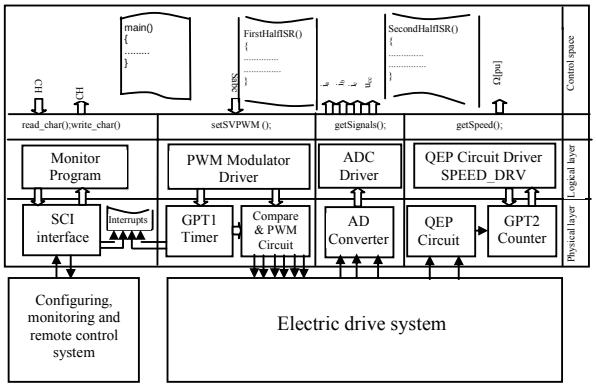

Fig.3. Hierarchical multilayer functional structure of DMC applications

The last hierarchical layer, of digitally processing mathematical functions, performs proper processing of information and supplying adequate command in order to reach established control objectives.

Using this minimal kernel to implement control structures, the system integrator can aggregate in control space different software modules, which are standardized as interface and independently functional. The system framework is implemented in C programming language.

The implementation of elementary algorithms, typical for blocks presented in Fig.2, was realized with modules, defined according to XDAIS. The general structure of a module is presented in Fig.4.

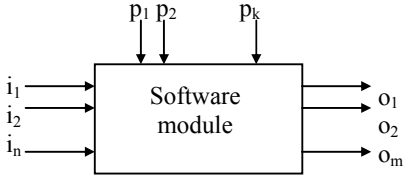

Fig.4. The general structure of a software module

Using own initializing function for settling $p_1,p_2 \ldots\ldots p_k$ parameters, each module can be adapted/configured for suitable requirements of the used electric drive system. The $o_1,o_2,...o_m$ output values are calculated on the basis of input quantities, transferred by application framework, and of specific parameters. For the reason of speed execution and space memory requirements optimization the output evaluation functions were implemented using DSP assembly language. Moreover, to obtain maximum accuracy of information representation both input/output quantities and parameters have adequate representation format, in Q8-Q15 range.

3.2 Graphical User Interface Design Subsection

A well-designed graphical user interface is developed on the basis of some principles and development process that take into account the user and his/her actions. A well-designed graphical user interface ensures a consistent and intelligible development environment and can influence the user devotion for that product.

In order that a graphical user interface to be user-friendly any programmer has to comply with a lot of principles that confer versatile features to it. The requirements to which a graphical user interface must respond are [10],[11]:

- *Application control by the user.* The first and very important requirement is that the interface have to be realized in such a way that the user feel the actions of the application are controlled by himself and not vice versa. The user must have also the possibility to personalize some interface aspects varying with his/her preferences.
- *Consistency.* The consistency allows user to transfer their knowledge to new processes, and to rapidly learn them. Any new application must respect the standard that the Windows operating system requires.
- *Association of "retorts" for application actions.* It is recommended that the interface transmit a "retort" for each user action. It impose that the application confirms, visual or audio, that it assumed user request and it is executing.
- *Aesthetics.* Another important part of interface is visual elements design. Visual attributes furnish valuable data and communicate important information relatively to interaction behavior of some particular objects.
- *Simplicity.* An interface must be simple, easily to learn and easily to manipulate. It must allow the access at all functionalities offered by an application. Maximum functionality of an

application and creation of a simple interface are two conflicting goals but they can be balanced through a good design. A good interface will present the information in a hierarchical manner. When menus are designed, there are not recommended too many submenu levels because the user is constrained to memorize many commands to access a certain option. Usually, for frequent commands buttons are associated to access the commands as simple as possible.

3.3. DSPGraphs - a Tool for Behaviour Analysis of DSP Motion Control Systems

The Visual C++ rapid application development tool was used to implement this graphical environment.

DSPGraphs - a sovereign application - does not implement all the functions of a standard integrated developing environment. When a COFF object file is obtained DSP standard tools are used. When binaries are obtained, the interface can be used to transfer them into target system memory, to remote start/stop the control application, to analyze different control structure quantities etc. Thus all specific variables of current project (path, COFF file name, analyzed quantities, interface configuration etc) are saved in a project file with *gph* extension; these files are used whenever workspace must be recovered. The *File* menu comprises all standard options of a Windows based application *(New, Open, Save, Save as, Exit etc).*

Once the control application included in project (the file with *out* extension) the user has the possibility to configure the client window with four displays, each of them with maximum four studied variables. Fig.5 presents a graphical analyzing example of variables tracks belonging to a DTC structure presented in Fig.2.

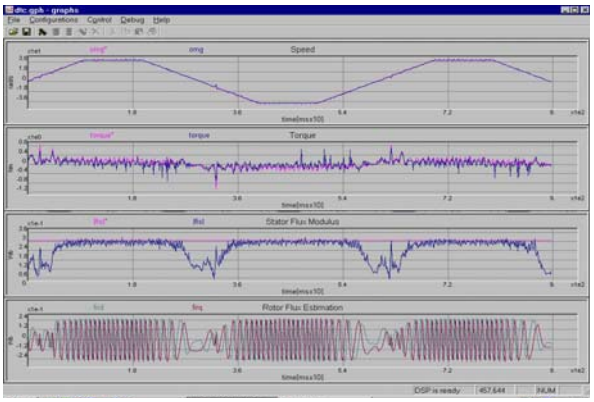

Fig.5. A graphical analysis example – DTC structure implemented on MSK243 DSP Motion Starter KIT

The user must select them opening a dialog box like that presented in Fig.6. For each selected variable the user must introduce the scale factor, which take into account Q_x representation format, gain sensor, and used measure units.

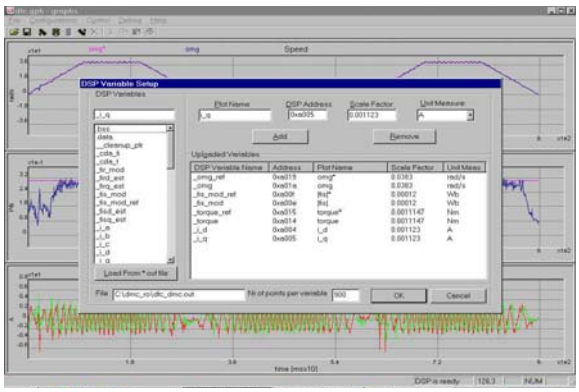

Fig.6.The dialog box for variable selection

In the next step the user can distribute selected variables into interface displays. All commands for variables selection, variable distribution, color, width, and style plots setting, displays arrangement etc are grouped in *Configurations* menu.

By means of *Control* menu options the user achieves the remote control of the DSP board. This way the user can transfer the control application in DSP data/program memory, to start the application with/without acquiring selected variables, to acquire the selected variables during the application running, to stop the application or to upload the acquired variables in order to be presented and analyzed.

If as a result of the electric drive system analysis and user evaluation concludes that the behavior is not the predicted/desired one, he/she can analyze more carefully the software structures that granulate the DSP application. The structures inspector from the *Debug* menu allows to get and/or to set the structures members values for structures used in control algorithm. The structures are based on reusable and flexible modules concept as in Fig.4.

The example presented in Fig.7 shows the evaluation modality of proportional coefficient ki belonging to PI speed controller structure.

For drawing experimental date *DSPGraphs* application enables for selected display the exports of EMF Windows image file format. Fig.8 presents a screen capture with the menu for export options. The EMF image file resulted from this operation is shown in Fig.9.

If somone wants other type of numerical analysis

DSPGraphs saves all the acquired variables in ASCII type file, compatible with MATLAB m-file format.

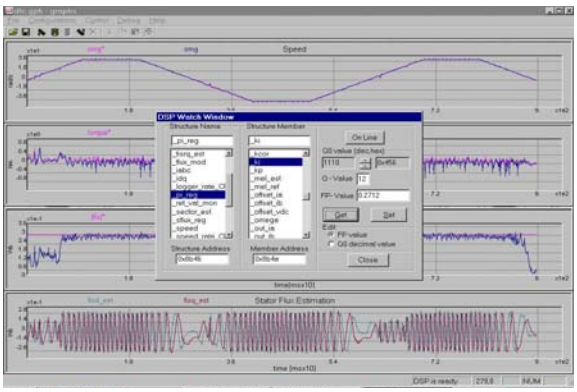

Fig.7. An example of members' structure evaluation

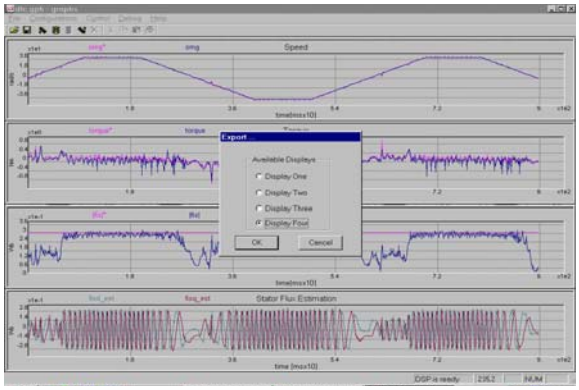

Fig.8. Export options of EMF Windows image file format

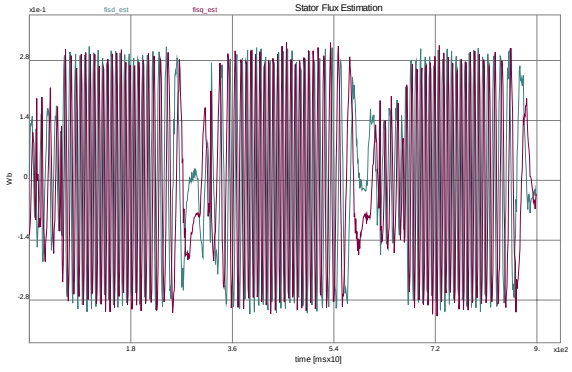

Fig.9. The EMF obtained from Fig.9, the 4th display

4 Conclusions

The DSP is a solution for hard real-time control problems, but the software tools are rather poor and an integrated software development platform is not available for most of DSP.

The software development environment presented in this paper solves the integration problem on a DSP platform, create an user-friendly programming environment on a PC platform which allows tracking of on-line variables, displays off- line and stores the program variables, solves the communication aspects while an application runs on DSP, and is an open system for further development.

An application framework for MSK243 Motion Starter Kit was presented. An example of a modular application was implemented with easy configurable and reusable software modules.

With the communication functions available in accompanying MSK243 Motion Starter Kit library and the debugging facilities offered by firmware monitor the authors designed and implemented a graphical tool for analyzing, evaluation and debugging of electric drives digital motion control applications. The graphical user interface offers features for a user concerned in DSP motion control. The developed software has the advantages of an open system that allows the new functionalities addition.

References:

[1] Vukosavic S., Controlled Electrical Drives - Status of Technology, *Zbornik XLII Konf. ETRAN-a,* Vrnjacka Banja, Sveska, 1998.

[2] Stefanescu C., Cupcea N., *Measurement and Control Intelligent Systems* (in Romanian), Editura Albastra, Cluj-Napoca, 2002.

[3] Kreindler L., *DSP Solutions for Digital Motion Control,* Journal of Electrical Engineering,Vol.2, 2002, pp. 13-22.

[4] De Carli A., Motion Control: An Emergent Technology,12*th World Congress IFAC,* Sydney, Australia, 1993, pp.423-426.

[5] Pârv B., Software Components. *PC Report* (in Romanian), 1998, No.69.

[6] ***, A Software Modularity Strategy for Digital Control Systems, TI, 2000.

[7] Vas P., Sensorless Vector and Direct Torque Control, *Oxford Univ. Press,* 1998.

[8] Răţoi M., Horga V., Albu M., Diaconescu M.P., A DSP Implementation of DTC Method for an Induction Motor, *3rd Int. Conf. on Electrical and Power Engineering,* Iaşi, Romania, 2004, pp.1077-1084.

[9] ***, MSK243 – DSP Motion Starter Kit, Technosoft, 2001.

[10] Cooper A., *About Interface. The Essentials of User Interface Design,* IDG Books, 1995.

[11] Suciu D.M., Graphic User Interface Design Principles, *PC Report* (in Romanian), 1996, No.51.