Digital Motion Control Development Shell

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

Abstract: The paper presents a software product developed in University comparable with other industrial products. The digital motion control development shell solves the hard real-time constraints imposed by the AC drive control applications. It was developed because of lack of integrated software tools for TMS320C3Cx family. The main advantages of the shell proposed are: high level programming, integration in a natural way of the software tools delivered by Texas Instruments Inc., user-friendly programming environment, powerful library functions for peripherals, portability and readability of application programs. The software product can be used in research and in educational process as well.

Key-Words: DSP, digital motion control, shell, programming environment, communication, macroinstructions.

1 Introduction

During the last years a lot of research has been carried out in the field of electrical motor drives. Most complex power electronic systems and variable speed drives now incorporate microprocessors controls. Some common characteristics can be found related to high complexity of control algorithms, fast sample rate required, need of interfacing to different kind of sensors, power amplifiers and motors. Several manufacturers make the regulators for drives motor in digital form and integrate it into the positioning control system. Some of them include in the software supplementary functions as self-commissioning, flux observers or identification functions. 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 high signal processing effort for regulation, transformation and flow model computation preclude an analog system.

Different rotor-phase configurations are available, requiring corresponding power inverter technology. Control measurements (stator voltages and currents, rotor velocity, etc.) are made by means of different transducers, each of them with different interfacing and conditioning requirements. The controller output commands can be analog or PWM form.

The common characteristic for the most motor controller is the non-linearity. Robustness of the solution adopted is very important, since some parameters can change as a function of time or may be unknown. Several modern non-linear control methods in the control are reported in the literature as feedback linearisation techniques or robust techniques based on variable structure system [1].

An analysis of several control algorithms in the field showed a variety of computational and interfacing problems. Feedback linearisation and flux observer algorithms are the most demanding from the computational point of view. Some authors reveal that the sample rates as low as hundred of μs are often required to ensure dynamic performances. Therefore a powerful processing unit is required.

Regarding the interfaces, two different classes of problems can be considered: process interface and user interface. Process interface is the most demanding in terms of hardware resources, while the user interface is strongly software dependent. Most of the industrial solution for digital drives control are closed systems, and do not allow to the user to modify the control algorithm or to embed new supplementary functions. These systems are not useful for research. On the other hand, papers reporting university research reveal a poor experimental activity or only simulation results are presented [1].

Taking into account all these issues, the main constraints imposed to a development system for AC drives control can be summarized as follows:

- sample frequency in the range 20÷30 KHz,
- signal and coefficient unitisations/computational delay may degrade the performances,
- algorithms can be particularly sensitive to structural noise superposed to input signals,
- efficient man/machine interfaces for simple start-up short repair time in the event of malfunctions,
need of powerful interfaces to serve high-order control levels,
- access to internal system variables for debugging, statistical and optimisation purposes,
- an easy adaptation of the drive to the motor used,
- ability to be integrated in a wide-application spectrum, from single-axis stand-alone drive to multi-axis synchronous control systems.

Several years ago the parallel processing was considered a possible solution for this kind of applications. In the last years almost all solutions reported use a Digital Signal Processor (DSP). The use of a fixed point DSP or a floating point DSP is still an open problem.

In 1998 Texas Instruments introduced on the market the processor TMS320C240. This is a low cost fixed-point processor especially dedicated to electrical drives control [2]. It incorporates a PWM unit, 2 encoder entries, 4 timers, 2 D/A converters, 32 I/O bytes and 2 serial interfaces. The use of the fixed-point arithmetic implies complex scaling procedures in order to maintain a high precision. For research purpose, that implies complex computation for real-time flux and speed estimation, rotor resistance estimation the best solution remains a floating point DSP as TMS320C31.

The software tools provided by the manufacturer consist in standard tools for development (assembler, compiler, linker etc). Users eager to run their applications on a new DSP system are often disappointed by the lack of an immediate solution to classical problems like:
- how to download a program;
- how to check if a program works properly;
- how to debug a program if it does not work as expected;
- upload for visualization of the results,
- inspect modify DSP program and data memory.

In the particular case of motion control applications the necessity to provide a monitor is even stronger. Finally, a project management system, that provide an effective way of quickly visualizing, accessing and manipulating all project files and their dependencies, seems to be necessary to promote a more efficient development process.

2 Hardware description

A picture of the hardware structure used is presented in Fig.1. It consists of a C31MCDB 2.0 development board with special feature for motion control and a personal computer standard platform. The PC platform was selected since an open system has to be designed and among actual standard systems the PC is widely available, simple and cost-effective.

C31MCDB board puts together the power of TMS320C31 floating point DSP with a large set of I/O devices. Due to an RS-232 interface the board can be easily connected with the PC. Version 2.0 comes with a monitor program that is automatically loaded from EPROM memory and executed, after reset.

A Texas Instruments TMS320C31 floating point digital signal processor running at 32 MHz with 60 ns cycle time that offers up to 32Mflops and 16 MIPS in performance ensures the required computational power [2].

The board is equipped with 128Kx32 static RAM. The RAM chips have access time below 25 μs, and the processor can run with no wait state during all external memory access.

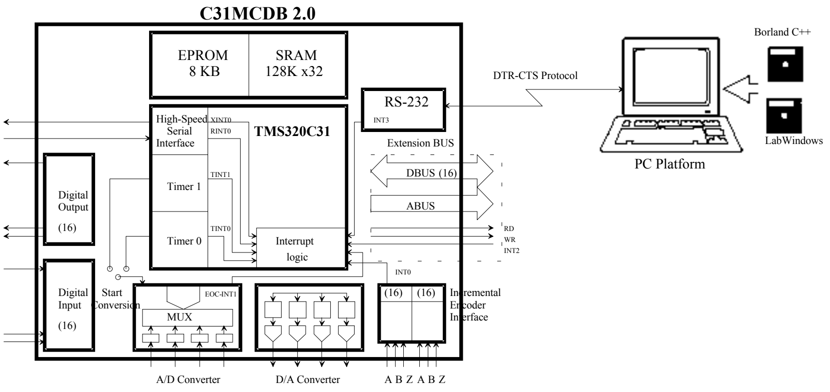


Fig.1. Hardware structure of the C31MCDB board

The board is equipped with 128Kx32 static RAM. The RAM chips have access time below 25 μs, and the processor can run with no wait state during all external memory access.

The C31MCDB is a product of Technosoft S.A. Switzerland.

The processor has an internal set of peripherals as external buses, a direct memory access channel, a serial port and two timers. The board is also provided with several interfaces, all of them useful in motion control applications: 4 bipolar analog inputs with simultaneous sampling, 4 bipolar analog output with independent or synchronous update, dual 16 bit interface for quadrature incremental encoder, 16 TTL digital inputs and 16 HCMOS digital outputs, serial interface PC compatible supporting up to 38.4 Kbauds, extension bus for future development.

The set of internal and external peripherals are sufficient to carry out all kind of application in the field of AC motion control.

The D/A converter has double-buffered latches, with 2's complement inputs. The output range is ±5V and the channels may be independently or simultaneously updated with 4μs settling time. The D/A converter of C31MCDB board is the Analog Devices chip AD75004.

The 12-bit A/D converter has four channels, all ±10V range sampled simultaneously, thus preserving the relative phase information. C31MCDB board uses an Analog Devices AD7874 chip, with track and hold amplifiers on each channel. Maximum sample rate for all four channels is 29 KHz. The A/D converter has a single read/write address. The read address is duplicated. That happens because TMS320C31executes external read cycles using static accesses (read strobe signal remains active during consecutive read cycles). This facilitates faster accesses to external memories but creates problems with most I/O devices that request low to high or high to low transitions for the read strobe to begin or end a read cycle. As effect, consecutive read accesses from the A/D converter, encoder interfaces, ports connected through extension bus and external serial interface are not allowed. Due to that, all I/O devices have duplicated read addresses. The second read addresses are equal to the first I/O device read addresses plus a 400000H offset.

The incremental encoder interface has two channels. Each channel has differential inputs and digital noise filter. The two encoder interfaces are completely independent except the interrupt request signal linked to TMS320C31external pin INT0.

The external serial interface is a Universal Asynchronous Receiver Transmitter (UART) interface implemented with a Philips SCC 2691 chip, to communicate with a PC.

3 Digital motion control development platform

3.1 Existing software

The board comes with a monitor program that is automatically loaded from EPROM. The monitor program works as a communication interface between the board and a PC. Through the serial link it accepts the following commands from PC:

- A (followed by 3 bytes) – set program memory/data address,
- L (followed by 1 byte) – set data block length for the next read/write command,
- W (followed by data) – write data to board memory, starting from the address fixed by the previous command A,
- R read data from the board memory, starting from current address fixed by the previous command A. The length is established by the previous command L.

E executes a program loaded on the memory board from the current address. NULL command is used to restore the communication protocol after a communication error.

3.2 Development requirements

Because of the increasing complexity of DSP applications the designer must kept track of an great number of different project files, such as C, ASM, library and *.cmd files. The shell proposed in this paper provides an easy and efficient integration of the Texas Instruments DSP tools. It was developed following the natural flow of files processing until to the direct executable files that can be loaded in the target DSP system.

Another requirement is to increase the portability and the readability of the C programs. On that purpose, it was developed a library with macroinstructions and functions for peripherals. This library is in fact a collection of drivers for peripherals and allows the increase of the programming level. A major advantage of this library is that allows working with peripherals without precise knowledge about the hardware.

In AC drives control, especially when adaptive or optimal control algorithms are designed, the user needs to use matriceal calculus. Thus a matrix function library was developed. This library contains functions for elementary matrix computations.

An easy and robust data exchange between the DSP board and PC is another requirement. After the reception of a group of data, the DSP monitor sends an acknowledge character and a checksum to confirm the correctness of the data. This protocol detects communication errors and guaranties the integrity of the exchanged data. The communication functions implemented on the PC platform must treat the response of the DSP board in order to guarantee the transferred data integrity.

The C3x floating-point format is not compatible with the IEEE standard 754 format [3]. A problem is floating-point data transfer. Two routines for format conversions are compulsory. These routines must be embedded in the communication functions. They are useful when are transmitted and received floating-point data.

The shell proposed in this paper solves all these requirements. A graphic representation of the software development flow in conjunction with the communication and downloaded facilities is given in Fig.2.

The graphic user interface was designed using the LabWindows package. The graphic user interface was designed using the LabWindows package. The integrated menu is user-friendly and covers all common steps followed to run an application program on the C3MCDB. Several additional features are developed as: *configure, communication test, configure link.cmd and configure sim.cmd*. The options of the graphic user interface are presented in Fig.3.

The option *configure lnk.cmd* uses the directives SECTION and MEMORY of the linker and specifies the structure of he DSP memory and peripherals. The option *configure sim.cmd* ensures the compatibility between the hardware resources of the DSP board and the virtual resources simulated in the PC. This option is very useful in the debugging phase of the program. The user can detect the addressing errors that may appear.

3.3 Serial communication

A set of functions for communications was developed to cover the communication needs between the PC and the DSP board. This is a set of high-level functions different from the elementary commands of the monitor. Two classes of functions were developed. The first one covers the transfer from/to the PC and the second one covers the transfer to/from the DSP board. The functions from the second class are included in the application program. When the application run the bi-directional communication is ensured by these functions.
The functions, included in rs232pc.h, are:

void Init_rs232(void);
void communic_test(void);
void send_byte_pc(char byte);
int receiv_byte_pc(void);
void send_float_pc(float number);
    function call: send_byte_pc(char byte)
float receiv_float_pc(void);
    function call:receiv_byte_pc(void);
*void send(char *buffer_mem, int no_bytes);*
*void receiv(char *buffer_mem, int no_bytes);*

The functions, included in rs232c31.h, are:

void Init_rs232(void);
void send_byte_dsp(int byte);
int receiv_byte_dsp(void);
void send_float_dsp(float number);

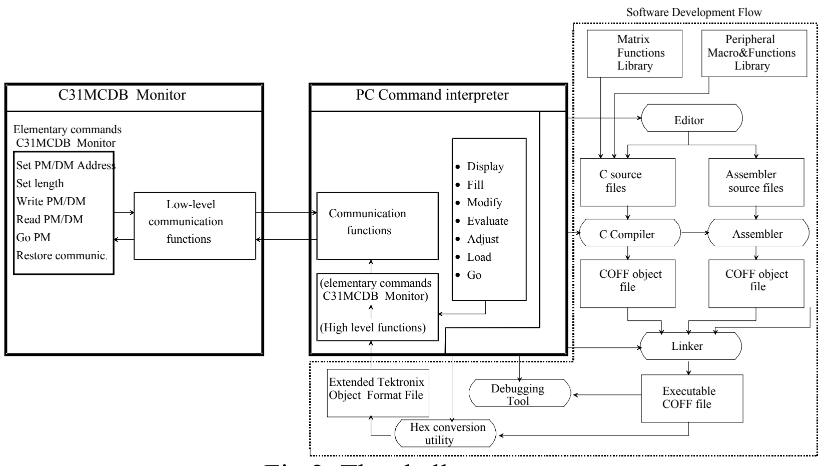
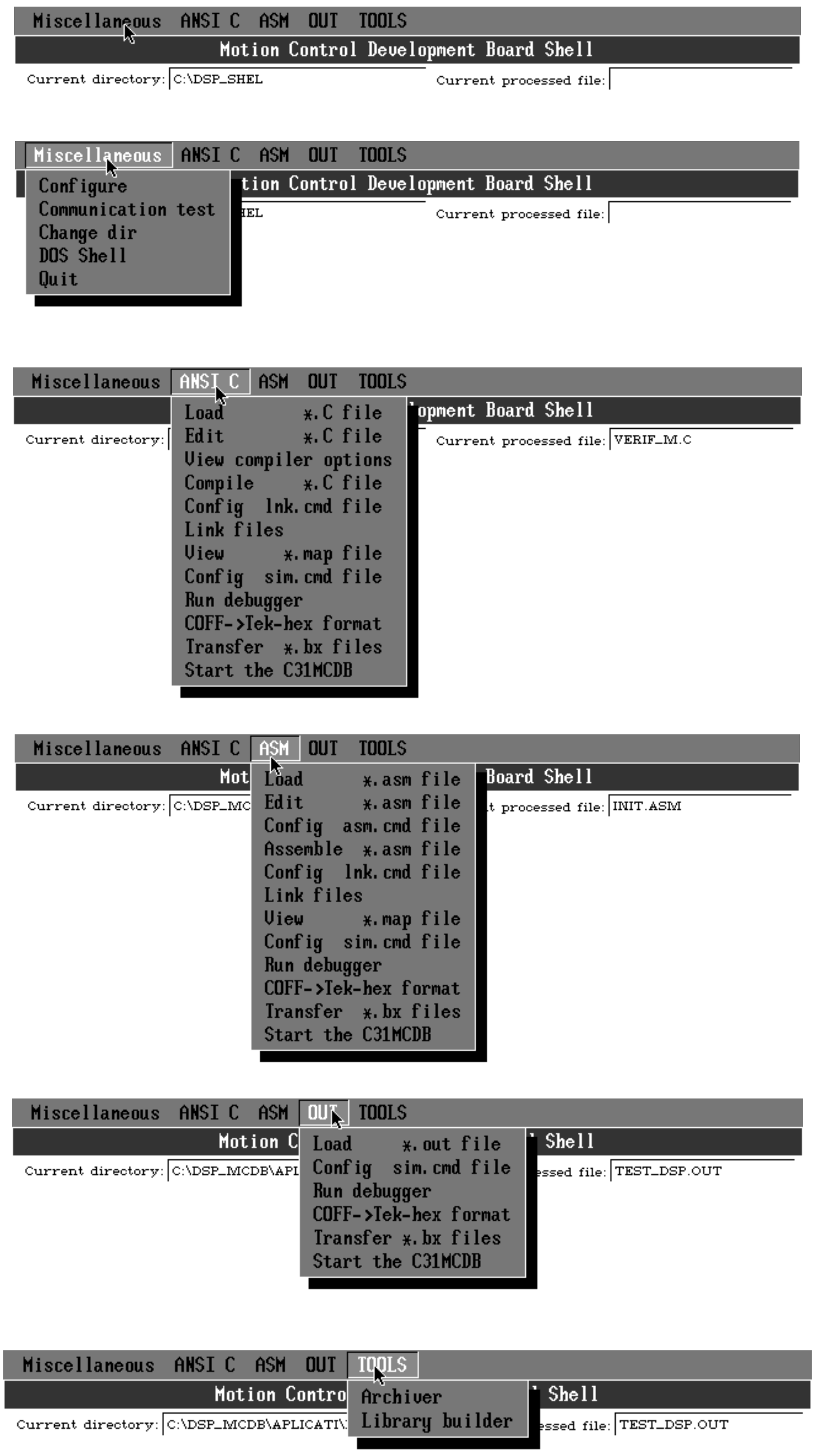


Fig.2. The shell structure



Current directory: C:\DSP_SHEL Current processed file:

Current processed file: VERIF_M.C

Current directory: C:\DSP_MC Current processed file: INIT.ASM

Current directory: C:\DSP_MCDB\APL Current processed file: TEST_DSP.OUT

Current directory: C:\DSP_MCDB\APLICATI\ Current processed file: TEST_DSP.OUT

Fig.3. The options of the graphic user interface

function call: int c312ieee(float number)
void send_byte_dsp(int byte);
float receiv_float_dsp(void);
function call: float ieee2c31(int number);
int receiv_byte_dsp(void);

3.4 Macroinstructions and functions for peripherals

The macroinstructions and functions are designed for A/D and D/A converters, incremental encoder interface, digital I/O interface and timers. These macroinstructions and functions are organized in libraries corresponding to each peripheral device.

The A/D conversion is done with a macroinstruction and a function. They are grouped in adcc31.h library. The macroinstruction START_CONV allows to start software the conversion an all four channels. The read address of the A/D converter is duplicated. The function *float adc(a_reg)* allows, by means of *a_reg*, to specify different addresses for consecutive reads: *a_adc* for the base address and *a_o_adc* for the complementary (offset) address.

The D/A conversion can be performed with a macroinstruction and two functions, included in the dacc31.h library. The function *void dac(int channel, float data)* starts a D/A conversion on a specified channel with a data value. The function *void load_dac(int channel, float data)* loads the first buffer of specified channel without starting the conversion. The macroinstruction UP_DAC_ALL starts the D/A conversion of all four channels.

In this manner were developed specific functions and macros for the other peripherals.

3.5 Matriceal library

A matriceal library was developed because several performant control algorithms use extensive matriceal computation (Kalman filter, Luenberger observer, recursive least squares estimators). This library contains elementary functions with matrix. These functions are:

- *float *Vector(int imax)* –Memory allocation for a vector with float components and length imax,
- *float **Matrix(int imax,int jmax)* - Memory allocation for a matrix with float components and size (imax,jmax),
- *int *IVector(int imax)* - Memory allocation for vector with integer components and length imax,
- *void MatInit(float **a,int n,int m)* - Initialization of a matrix with the size (n,m),
- *void MatUnit(float **a,int n,int m)*- Return a matrix with the size (n,m) and the elements a(i,i)=1,
- *void MatAdd(float **a,float **b, float **c,int n,int m)*-Compute the sum of two matrix a(i,j), b(i,j) and returns the sum matrix c(i,j),
- *void MatDiff(float **a,float **b, float **c,int n,int m)*- Compute the difference of two matrix a(i,j), b(i,j) and returns the difference matrix c(i,j),
- *void MatProd(float **a,float **b, float **c, int l, int m, int n)*- Compute the product of two matrix a(i,j), b(j,k) and returns the product matrix c(i,k),
- *void MatProdS(float **a,float b, float **c,int n,int m)* - Compute the product of a matrix a(i,j) with a scalar value b and returns the matrix c(i,j),
- *void MatTrans(float **a,float **b,int n,int m)* - Compute the transpose of a matrix a(i,j) and returns the matrix b(j,i),
- *int MatInv (float **a, float **b, int n)* - Compute the inverse of a matrix a(i,j) and returns the inverse matrix b(i,j) using LU factorization by the Crout method; it also test if the a is a singular matrix.

4 Conclusions

The electrical AC drives digital control is a real-time problem with severe time constraints. That implies the use of powerful processors. The DSP is a solution for these control problems, but the software tools are poor and the integrated software development platform is not available for all types of DSP.

The shell presented in this paper solves the integration problems, create an user-friendly programming environment, solves the communication aspects while an application run on DSP, and is an open system for further development.

The strong points of this software product are: high-level programming, powerful functions library for peripheral, portability and readability of application programs, and an easy adaptation, if necessary, to another DSP board with the same processor.

The key features of the digital motion control shell are comparable with products made by industrial manufacturers like DMC Developer and DMCD Pro, delivered by Technosoft S.A., Switzerland, for TMS320C2xx fixed-point processor family.

The software product is also useful in research applications in the field of AC drives control, as well as in educational process.

References

[1] Morici R., Rossi C., Tonielli A., Fast Prototyping of Nonlinear Controllers for Electric Motor Drives, *Proceedins of the 12th IFAC World Congress*, *Sydney, Australia,* Vol. 2, 1993, pp.445-450.

[2] Nekoogar F., Moriatry G., *Digital Control Using Digital Signal Processing*, Prentice Hall, Upper Saddle River, NJ, 1999.

[3] Madisetti V.K., *Digital Signal Processors,* IEEE Press and Butterworth-Heinmann, Newton, MA, 1995.

[4] x x x, *Development Tool Solutions* (on CD-ROM), Texas Instruments, Inc., 2000.

[5] x x x, *Software Development Systems - Customer Support Guide CD-ROM*, Texas Instruments, Inc., 2000.