Disclaimer: This document was part of the First European DSP Education and Research Conference. It may have been written by someone whose native language is not English. TI assumes no liability for the quality of writing and/or the accuracy of the information contained herein.

Implementing Field-Oriented Control of AC Motors with the TMS320C25 DSP

Authors: D. Fodor, Jozsef Vass, Z. Katona

ESIEE, Paris
September 1996
SPRA326

TEXAS INSTRUMENTS

IMPORTANT NOTICE

Texas Instruments (TI™) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain application using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

Copyright © 1997, Texas Instruments Incorporated

TRADEMARKS

TI is a trademark of Texas Instruments Incorporated.

Other brands and names are the property of their respective owners.

CONTACT INFORMATION

US TMS320 HOTLINE	(281) 274-2320
US TMS320 FAX	(281) 274-2324
US TMS320 BBS	(281) 274-2323
US TMS320 email	dsph@ti.com

Contents

Figures

Tables

Implementing Field-Oriented Control of AC Motors with the TMS320C25 DSP

Abstract

The field-oriented theory is the base of a special control method for induction motor drives. With this control method, induction motors can successfully replace expensive DC motors. Induction motors require complex control algorithms, because there is no linear relationship between the stator current and either the torque or the flux. This means that it is difficult to control the speed or the torque, because of the transients until the motor reaches its new stationary state. The problem can be solved by controlling the rotor flux since it cannot be measured, only computed. Because of the complexity and nonlinearity of control equations it is useful to implement a part of the control system by a fuzzy logic controller. By using linguistic variables in place of numerical variables, that approach represents a substantive departure from the conventional quantitative techniques of system analysis and control. In the present paper we use fuzzy logic based speed control for field-oriented AC motor.

For implementing the control algorithm we developed a digital signal processor (DSP) based single board controller based on the Texas Instruments (TI™) TMS320C25 DSP. The board contains a configurable digital interface, which can be used to extend the functionality of the board by connecting analog and digital I/O peripherals. It has a flexible memory subsystem with EPROM, SRAM and nonvolatile SRAM support; an RS-232C serial port; a keypad with 6 keys; and four 7 segments LED display. In this paper we present the state-of-the-art of field-oriented control of AC motors, and fuzzy logic controllers. We describe our control system from both software and hardware point of view. The experimental results have been compared by in the case of traditional P1 type speed control, and shows good dynamical behavior.

This document was part of the first European DSP Education and Research Conference that took place September 26 and 27, 1996 in Paris. For information on how TI encourages students from around the world to find innovative ways to use DSPs, see TI's World Wide Web site at www.ti.com.

Product Support on the World Wide Web

Our World Wide Web site at www.ti.com contains the most up to date product information, revisions, and additions. Users registering with TI&ME can build custom information pages and receive new product updates automatically via email.

Introduction

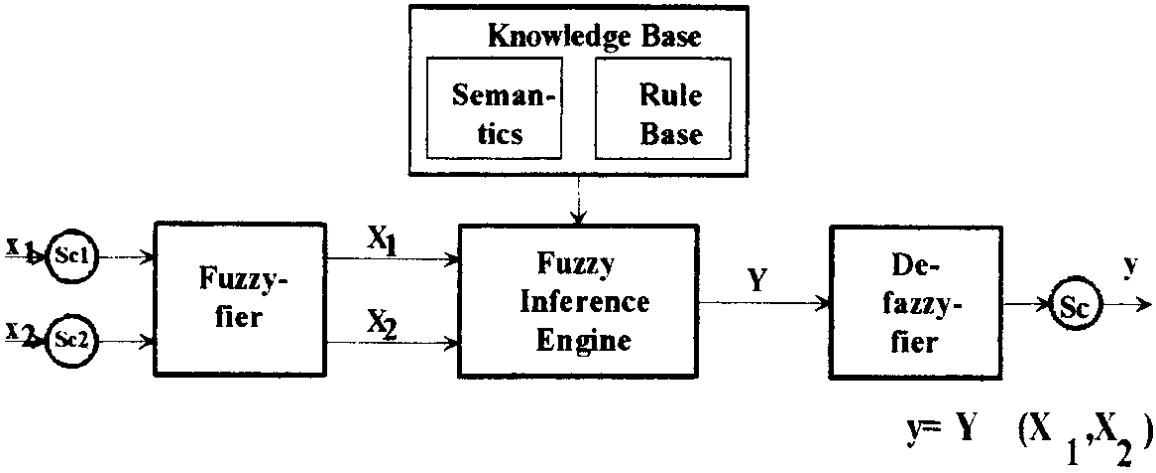
By using linguistic variables in place of numerical variables, that approach represents a substantive departure from the conventional quantitative techniques of system analysis and control. The fuzzy set theory gives the tools to represent and manipulate the linguistic variables. This approach provides an effective mean for describing systems which are too complex or too ill-defined to admit a precise mathematical model. Fuzzy set theory fundamentals have been investigated in detail, and applications have been proposed in economics, artificial intelligence, information retrieval, diagnostics and so on.

Fuzzy (Logic) Control is one of the most interesting fields to which the fuzzy theory can effectively applied. Recently some applications of FLCs to motor drives have been also reported. In our cases an asynchronous motor has been chosen as a bench to explore the design of a FLC drive system, and to investigate by simulation and experiment its performance. The drive is preliminary simulated with conventional digital P1 speed regulator in order to establish a term of comparison. The current control is performed according to an instantaneous voltage equation of d-q model of the asynchronous motor. The implemented drive is fully digitised, the control algorithm is supported by TMS320C25 fixed-point digital signal processor. The program considers the dead time caused by the current filters and measurement and the program running time. In the P1 controllers we gain a program running time about 1000 us. In the case of fuzzy logic control the inference and composition block together with the fuzzification and defuzzification blocks, have been implemented by means of a look-up table. The content of the table is off-line computed and thus it is greatly reduced the demand to the digital system of on line computation. This allows the control cycle to be as short as required by field-oriented AC motor drive application. A disadvantage of the look-up table technique is the low resolution in the input and output variables, unless a large and memory consuming table is used.

Fuzzy Logic Controller

The basic configuration of a FLC with three linguistic variables (two inputs and one output) is shown in Figure 1. The implementation of FLC needs a digital hardware, in our case a TMS320C25 fixed-point DSP. From the Figure 1, three main blocks can be distinguish whose function is hereafter briefly explained. Some further details will be given in the next Section.

Figure 1. Fuzzy Logic Controller



Pre-Processing and Fuzzification

This block receives the reference and feedback signals and determines the variables x_1 and x_2 which are chosen, in linguistic form, like antecedents by the control rules. Same linear and non-linear scaling factors (amplification) may be also used. Since the universes of the fuzzy sets, representing the linguistic values of the input variables, are finite, limitation is usually performed. In addition, if the universes are discrete, a quantification concludes the pre-processing, delivering the input numerical values.

The choice of the shape of the fuzzy sets is another important aspect in designing FLC. Triangular fuzzy sets, are often utilised, but trapezoidal exponential or monotone forms have been experimented. A general criterion in choosing the form of fuzzy set related to the input variables is that they must cover the whole universe X, with some overlapping between adjacent fuzzy sets in order to avoid high discontinuities of the control action as a consequence of a small change in the input variables. This is of particular importance in Flocs for motor drives because they usually exhibit small time constant.

In this work, the compositional rule of inference used to relate $\mu_{C'*}$ to $\mu_{A'*}$, $\mu_{B'*}$, and $\mu_{R* \text{ is}}$ is defined by (min max composition).

$\mu_{C'*}(z) = \max \min (\mu_{A'*}(x), \mu_{B'*}(y), \mu_{R*}(x,y,z))\ x,y$

However, there are also different definitions. For instance, those based on the max-product model use the algebraic product in place of the min operator for the logical and operational and/or the implication and/or the inference, with the advantage in some cases to produce a smoother and better controlled output. The fuzzification relates the input values to the fuzzy sets X_1, X_2, which will be composed with the fuzzy algorithm, according to compositional rules of inference shown in Figure 2.

All the designers agree that the numerical values may be well represented by fuzzy singletons, the membership functions of which are equal to zero everywhere except at the measured values x_1 and x_2 and where they equal one. This choice greatly simplifies the expression of the composition rule of inference.

Figure 2. Compositional Rules of Inference

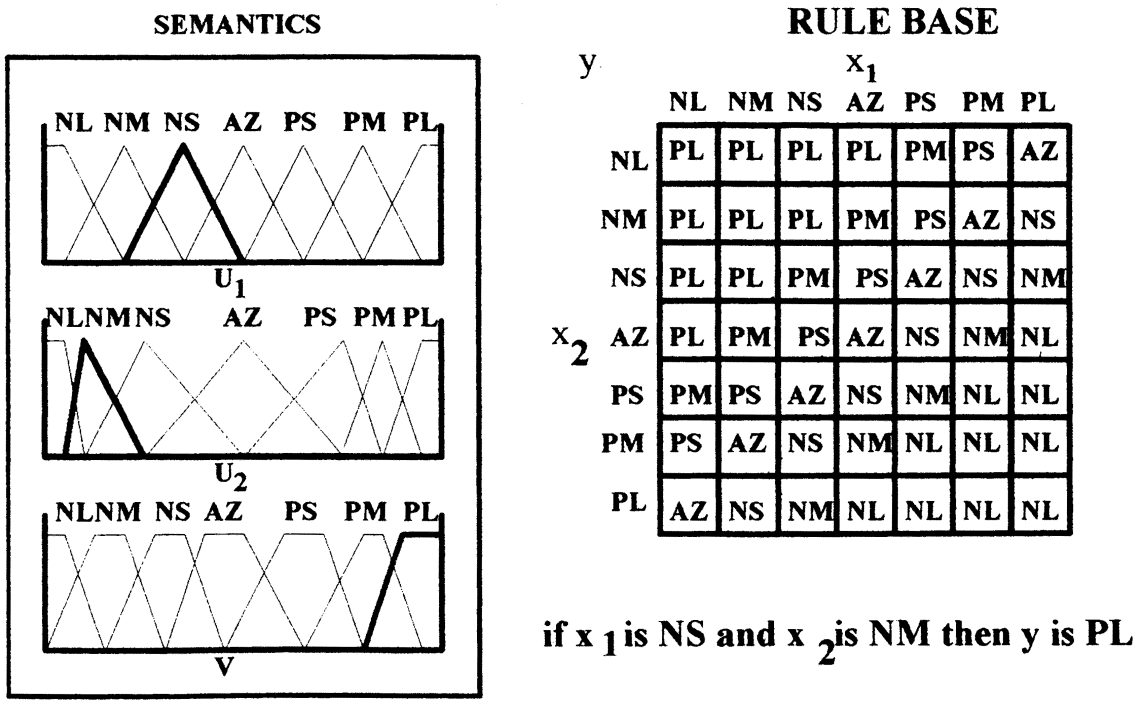
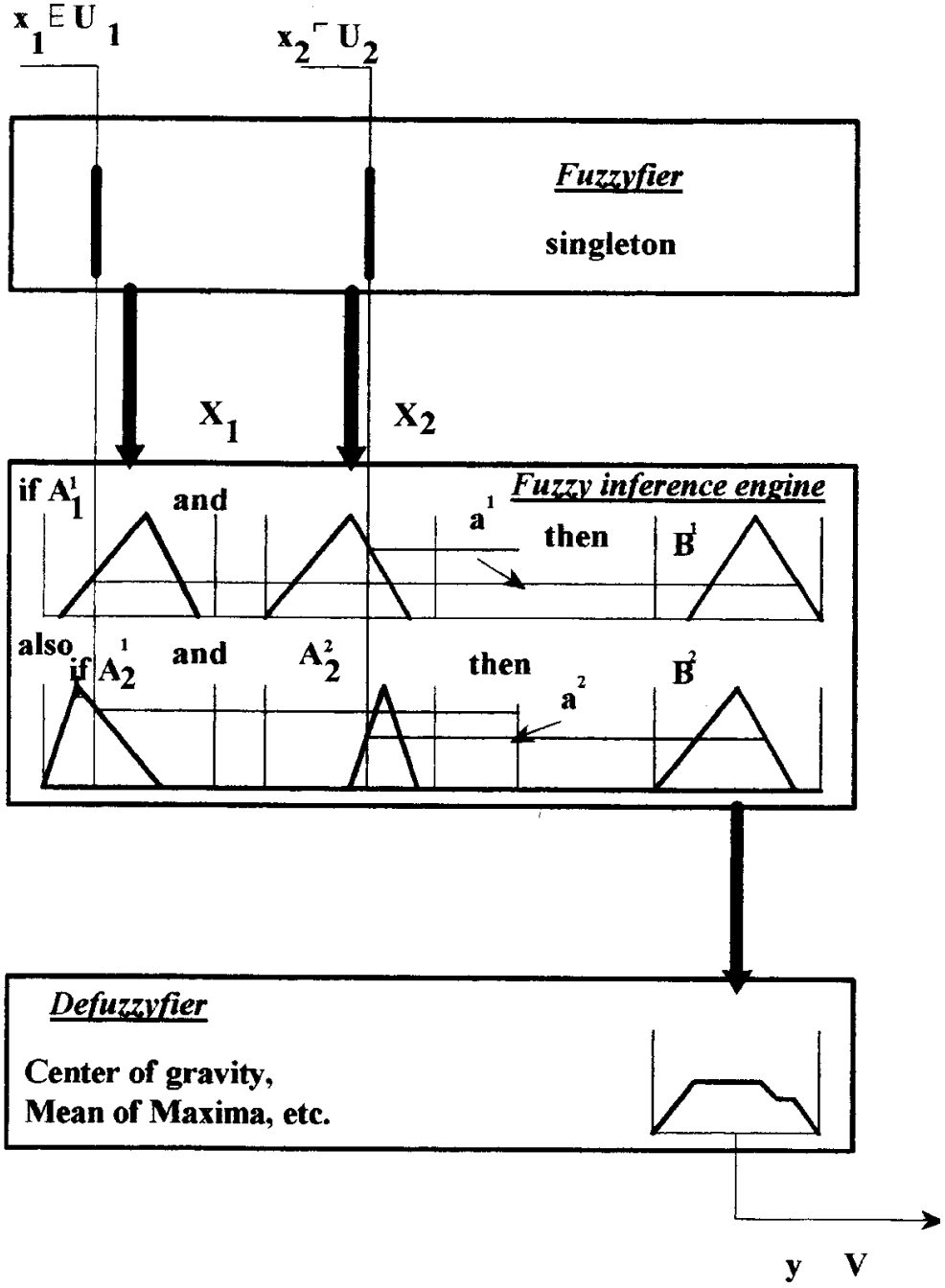


Figure 3. MAMDANI Controller



Inference

The second block in Figure 3 constitutes the fuzzy algorithm. According to the logical *or* which joins together the rules the fuzzy output set C'* is established by the union of all the fuzzy sets inferred by each rule $C'^* = C_1'^* + C_2'^* + C_3'^* + \ldots$

Defuzzification and Post-Processing

The input of the control system is a single numerical value and therefore the defuzzification block extracts from C'* the numerical value y which may be considered the best representative element of the fuzzy output set.

Alternatively, the entire form of C'*, which would mean the contribution of all rules, is taken into account. This is obtained by the center of gravity method which consists of averaging all the elements of C'* weighted with their own membership grade. A scaling factor is applied to the numerical value *y*, in the post processing part of the block, to obtain the output variable *y.* Finally, this is manipulated to generate the control command to the plant.

Application to an Induction Motor Drive

We make different measurements with PI speed controller and FLCs. For example, a signal response of the control system under the same working condition to a step change in the reference speed from -3000 min^{-1} to 3000 min^{-1} in the case of the PI speed controller is about 160 ms, and in the case of FLC is about 80 ms.

This means a better dynamic behaviour in the case of fuzzy control, but because a small look-up table was used a better reference speed accuracy is achieved by PI type control. The speed overshoot has disappeared and the current swings have been significantly reduced as well as the effect on the speed of the load torque disturbance. The robustness of the FLC is recognised.

Asynchronous Motor Modeling

The induction machine is a non-linear high-order system and for this reason complicated models must be used to control it. The dynamic behaviour of the induction motors can be described by a set of differential equations in a rotating frame of reference [14] [16] with the angular velocity ω_λ. For the purposes of this work, it is advantageous to express these equations in a rotating frame of reference (d, q) with the same angular velocity ω_λ. This means that the general equation of the torque is transformed to an advantageous form. If the fluxes expressed by means of currents and the terms i_r and ψ_s are eliminated, moreover it is assumed that a stationary reference frame is fixed to the stator ($\omega_\lambda = 0$), we obtain the general two-phase model in a matrix form [18] (Figure 4). This model is used in the motor simulation.

Figure 4. Matrix Form of Two Phased Model

$$\frac{d}{dt}\begin{bmatrix} i_{sd} \\ i_{sq} \\ \psi_{rd} \\ \psi_{rq} \end{bmatrix} = \begin{bmatrix} -\frac{\bar{R}}{\sigma L_\lambda} & 0 & \frac{L_m R_r}{\sigma L_s L^2_r} & \frac{\omega L_m}{\sigma L_s L_r} \\ 0 & -\frac{\bar{R}}{\sigma L_\lambda} & -\frac{\omega L_m}{\sigma L_s L_r} & \frac{L_m R_r}{\sigma L_s L^2_r} \\ \frac{L_m R_r}{L_r} & 0 & -\frac{L_r}{R_r} & -\omega \\ 0 & \frac{L_m R_r}{L_r} & \omega & -\frac{L_r}{R_r} \end{bmatrix} \begin{bmatrix} i_{sd} \\ i_{sq} \\ \psi_{rd} \\ \psi_{rq} \end{bmatrix} + \begin{bmatrix} \frac{1}{\sigma L_s} & 0 \\ 0 & \frac{1}{\sigma L_s} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_{sd} \\ u_{sq} \end{bmatrix}$$

The program calculates the voltages corresponding to the necessary current of the motor instead of using a current control loop. Because the algorithm running time is less than the rotor time constant the flux can be considered as a constant. Consequently, we can choose an equivalent circuit where the rotor leakage impedance is included into the stator transient impedance (L')

$$\underline{U}_s = I_s R_s + \frac{d}{dt}\underline{I}_s L + \underline{j}\omega_s L_m I_{sd}$$

After reducing the motor's physical parameters and transforming the equations mathematically, we obtained the stator voltage vector equation[15] .

Vector Control of the Asynchronous Motor

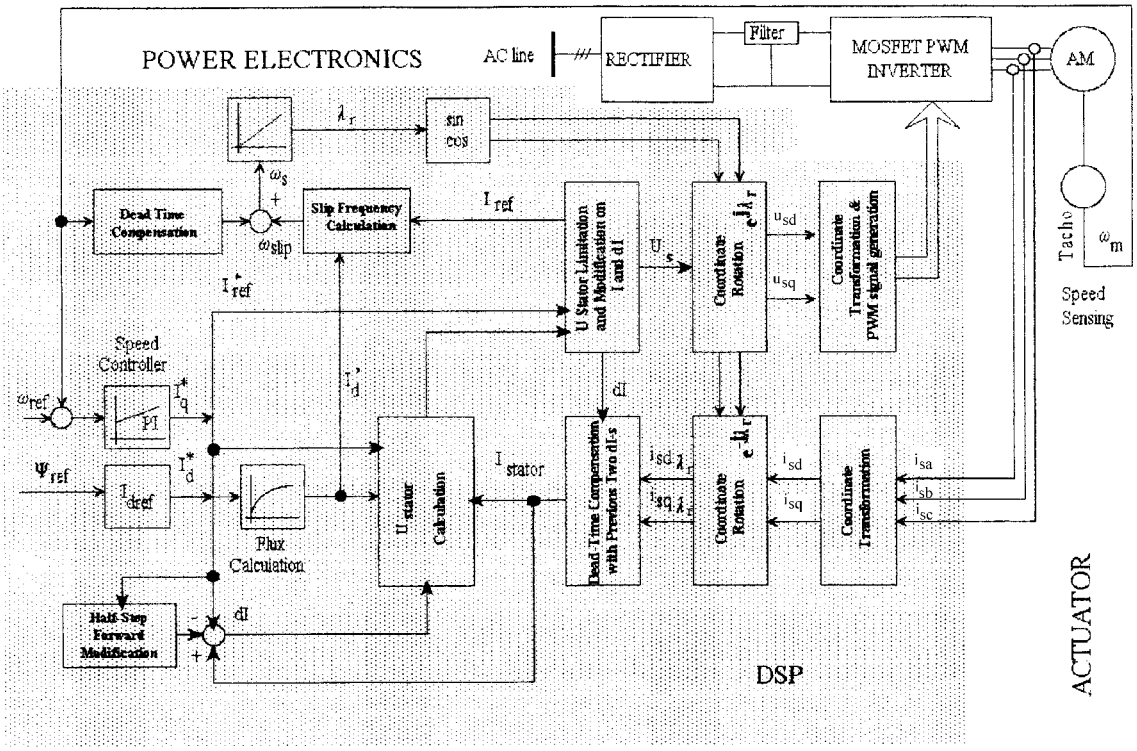
The field-oriented theory [2] ,[5] ,[6] is the base of a special control method for induction motor drives. With this control method, induction motors can successfully replace the expensive DC motors. Nowadays this method has become general in induction motor drives of high precision. The main advantages of induction motors are their simplicity and price, as well as greater reliability especially in harsh industrial environments. Induction motors require very complex control algorithms, because there is no linear relationship between the stator current and either the torque or the flux. This means that it is difficult to control the speed or the torque, because of the transients until the motor reaches its new stationary state.

The problem can be solved by controlling the rotor flux since it cannot be measured, only computed. The purpose of he controller is to keep the amplitude of the rotor flux at a constant value so the only its direction is changed. The field-oriented theory offers a suitable method for optimally control of the inuduction motors. The complexity of this method is compensated by its advantages. The most often used method is the one with the rotor flux orientation because of the simple structure of the control loops and command variable calculation. The space phasor of the stator current is split into two components, which become control variables. Vector rotation techniques are used to transform three phase axes into rotating two-phase d-q axes. This two-phase rotation technique greatly simplifies the analysis making it equivalent to analyzing separately excited DC motors because in this case there are two independently controllable currents: the field current and the armature current.

Description of Experimental System

Figure 5 shows a block diagram of the control system for a vector controlled induction motor in a velocity loop using a fuzzy controller.

Figure 5. Control System Block Diagram



The program calculates the voltages corresponding to the necessary current of the motor. All calculations in the block diagram - the co-ordinate transformation and rotation, fuzzy controller implementation, control calculation, and so on - are done by software. The TMS320C25 16-bit, fixed point Digital Signal Processor used in the control algorithm implementation gives a relatively long running time. Consequently, the program takes into account the dead times caused by the current filters and program running.

The algorithm implements indirect field-oriented control because the rotor flux is not computed exactly. The flux is considered a constant by the algorithm in the rotating d-q co-ordinate system and the currents are controlled according to this condition would be true. Knowing the maximum output stator voltage limited by the inverter the program modifies the active current (I_q) if necessary. The passive current (I_d) remains invariable, so the flux remains invariable too. Form the measured currents (i_{sa}, i_{sb}, i_{sc}) after co-ordinate transformation and rotation the control program estimates the current vector (I_{stator}) at the beginning of the control cycle taking into account the current changes in the previous two cycles (dead-time compensation). This estimation is essential because of the significant difference between the measured currents and the real currents at the beginning of the next cycle. The first reason of it is the delay caused by the measurement of stator currents. Approximately a 150 ms delays is caused by the filters and the A/D conversion takes some time too. The second reason is that the stator currents are changing during the running time of the program.

The active reference current (I^*_q) is generated from the angular velocity error signal by a PI (Proportional-Integral) controller. The passive reference current (I^*_d) can be controlled by the velocity (in the field weakening region) taking into account the time constant of the motor. The difference between the estimated stator current and the reference current gives the current vector *dI*.

As the real current vector is behind the reference current because the current reaches its reference value at the end of the cycle, so its average value differs from the required one and the program modifies (turns forward) the reference current vector (half-step forward modification) to generate the corresponding value. The stator voltage (U_{stator}) is calculated so that the current reaches its reference value in one step.

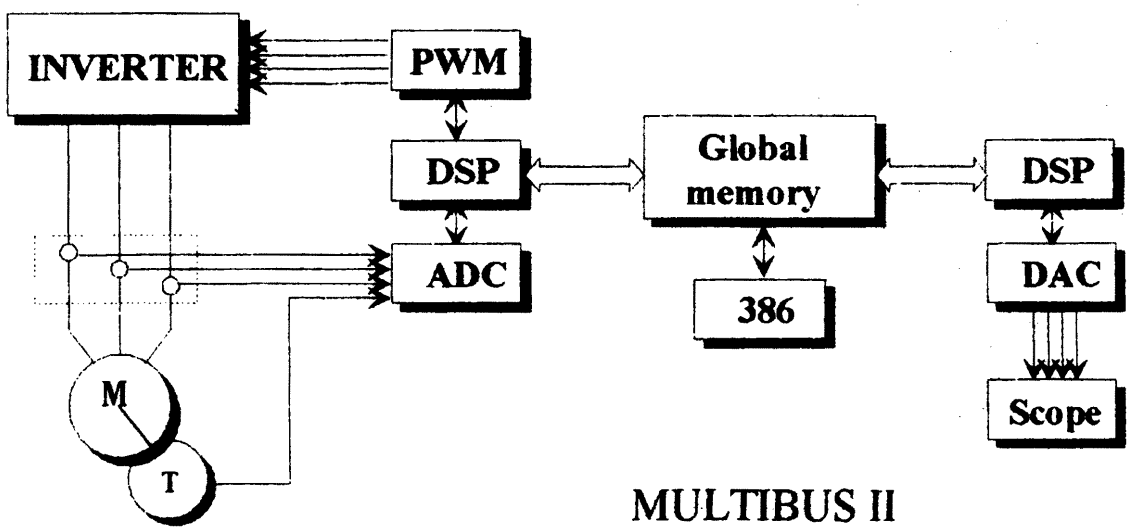
The calculation is based on a simple physical single-phased equivalent model of the induction motor. This model can be used because the running time of the program is far less than the rotor's time constant so the rotor flux can be considered a constant. When the calculated stator voltage is greater than the output voltage of the inverter, the corresponding current vector will not be as expected; consequently, the flux will not be invariable. Therefore, the program limits the stator voltage. Only the q axis component of the stator voltage is modified to keep the flux generating constant current. Of course, if the stator voltage changes, the reference current vector must be recalculated with this new value. The resulting stator voltage vector is rotated back and transformed and used to generate the PWM values.

The program uses 16-bit integer arithmetic and was written primarily in C language. Lookup tables are used for limiting the stator voltage, for storing the trigonometric values to co-ordinate rotation. Lookup table was used too to storing the control information for the fuzzy controller. In the realized solution of the algorithm, the control system works on a small power motor, using a sampling period of 1000 μs. Although the time constant of the motor is small, the slow sampling rate gives a satisfactory result. The dynamic behavior of the algorithm (speed reversal, torque changing) is good despite some torque fluctuation during the speed reversal.

Realized Hardware Setup in the Development Stage

The field-oriented control drive consists of an electronic control, inverter, current sensors, and speed sensor (Figure 6). An inverter was developed with power MOSFETs, which is suitable for field-oriented control of the motor. The motor used in our experiments is a simple three-phase squirrel cage induction machine (380/220 V, 370 W). The PWM switching frequency is 10 kHz. Galvanic separation and stator current detection are realised through Hall-type sensors. Stator currents are filtered by passive LC low-pass filters and sampled and held at every sampling instant, then converted with 12-bit accuracy.

Figure 6. Hardware Block Scheme



The experimental system was built in a MULTIBUS II crate. In the crate there were two DSP boards based on TMS320C25 processor to control and compute the algorithm, and a development system based on an Intel 80386 processor. The first DSP generates the PWM pulses and measures and pre-processes the stator currents, the speed, and the line voltage. The second DSP computes the field-oriented algorithm and handles a 4-channel D/A converter to visualise the selected variables during development. The reference signals and the process parameters of the control algorithm are interactively controlled from the development system.

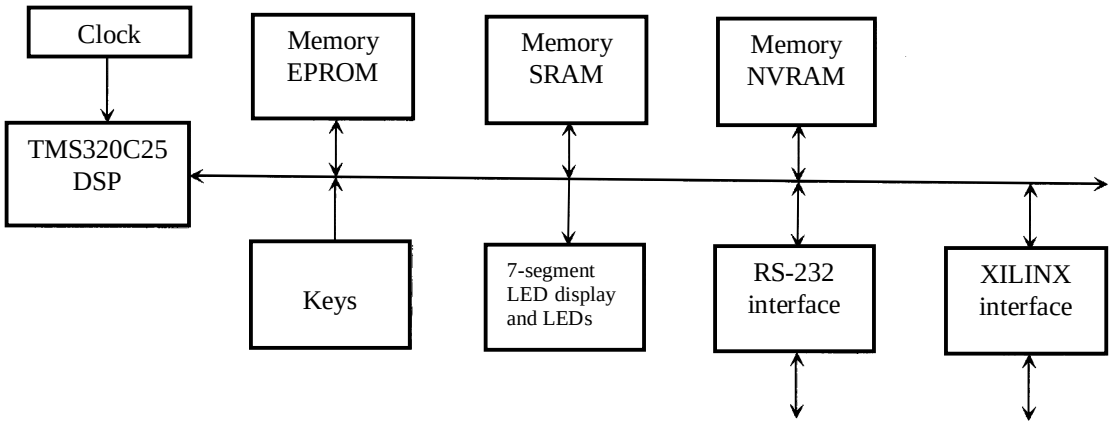
Because the development system presented above is too expensive for a real industrial application we developed and executed a single board TMS320C25 DSP based computer system.

The DSP Single Board Computer

General Description

The DSPSBC board based on Texas Instrument's TMS320C25 Digital Signal Processor (DSP). According to the configurable interface different kind of measurement and control I/O instruments can be connected to the board. In this way it is suitable for an internal controller task, where the high computing capacity of DSP is required. The block diagram of the board are shown in Figure 7.

Figure 7. Block Diagram of the TI TMS320C25 DSP



Features of the Board:

- Texas Instruments TMS320C25 Digital Signal Processor, 40 MHz clock-signal.
- Memory configuration is based upon the necessities:

  8, or 32 kword EPROM (3 wait state),

  8, or 32 kword static RAM (0, or 1 wait state).

  2 Kbytes nonvolative memory (1 wait state).
- RS-232C serial port.
- Keyboard consisting 6 tastes, which can be controlled by burned-in software.
- 4 pieces of 7 segments LED display and 4 LED's, controlled by the same burned-in internal software.
- Configurable XILINX FPGA interface circuit for different I/O devices, dedicated mostly for control and measurement device connectivity.

The Processor

The Texas Instruments TMS320C25 16-bits, fixed point Digital Signal Processor (DSP) with 40 MHz clock rate give rise for a 100 ns instruction (cycle) execution time. An internal timer and external interrupt connection are available. Considering the internal structure of the processor it was built with Harvard architecture, which means the program- and data memory phisically and logically are separated. In this way they can be accessed parallely in the same time allowing a quick instruction execution time. In the case of external memory the logical separation exists, but not the phisically not, only one address and data-bus exist and parallel memory access is not possible.

Memory

On the board there are three kinds of memory, EPROM, SRAM and NVRAM. The EPROM memory consists of 2 pieces of 8-Kbytes, or 2 pieces of 32 Kbytes chips. This memory contain the program code and the initialised data. The RAM memory which can be used as program-, or data memory is consists of 2 pieces of 8 Kbytes, or 2 pieces of 32 Kbytes chips. The NVRAM, which memory preserve the date after the turn-off, is consisted by one pieces of 2 Kbytes chip.

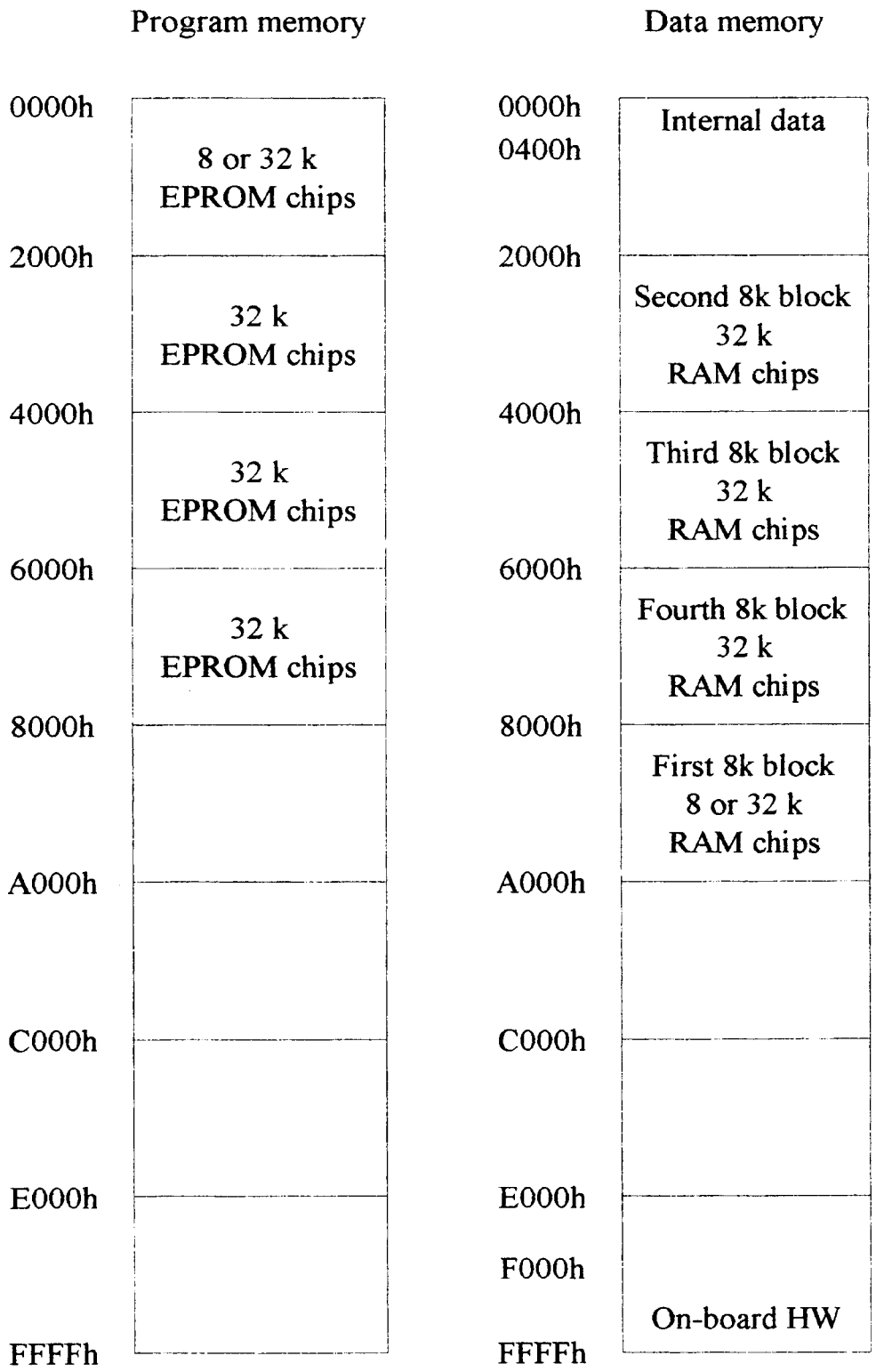
Keyboard

There are 6 pieces of Hall-generator type tastemounted on the board. The status of these keys can be read by the processor via a dedicated register.

Displays

One can find 4 pieces of 7-segments LED’s display and 4 traditional LED's on the board. The processor accesses the displays via two registers and displays hexadecimal numbers. In the first one must be written the values of the desired numbers while in the second controls the LED's operation for fraction dots, and the validation of the numbers displayed by indicators.

Figure 8. Memory Configuration after Reset



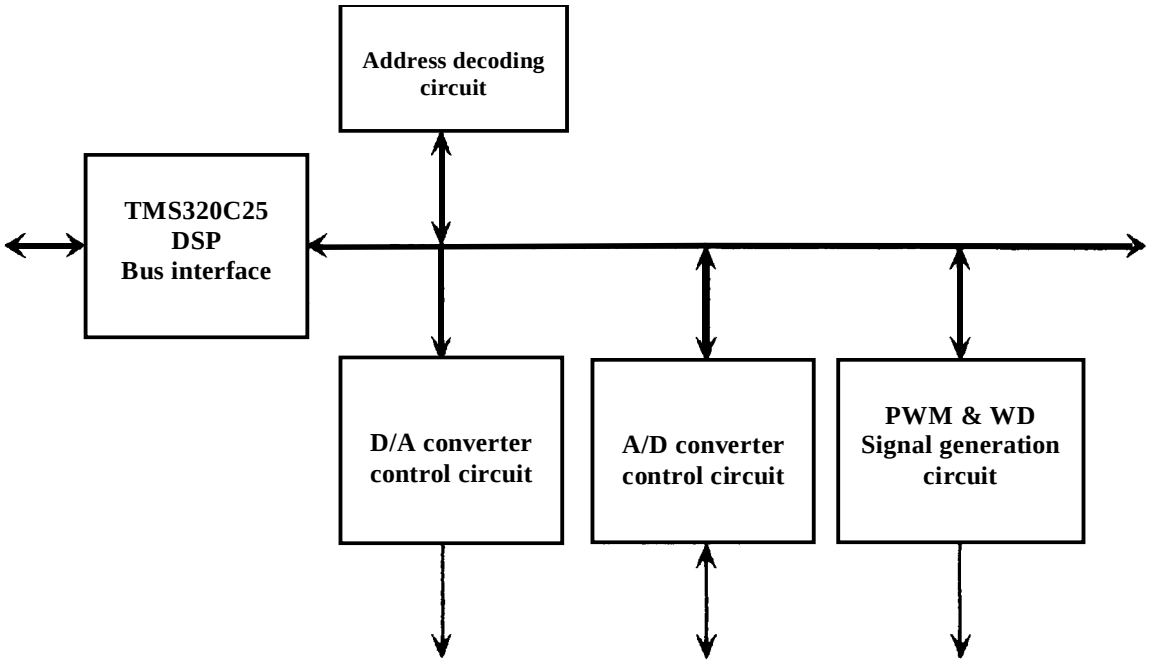
Serial Ports

The board has two serial ports: one of them is the DSP own serial port and the other one is a standard RS-232C serial interface.

XILINX I/O Interface

This interface is dedicated for connection to different types of I/O devices, mainly measurement and control devices. With the reconfiguration of the XILINX FPGA circuit, the board can be easily adapted for a new task. In the present configuration, the board is set-up for the control of a vector controlled AC motor. That means that the XILINX FPGA chip is configured to control an ERT 4.05.01.0 type 8-channel analogue-digital converter board and a 4-channel digital-analogue ERT 4.10.01.0 converter board. More over the XILINX chip is responsible for the generation of the PWM (Pulse Width Modulation) and watchdog signals.

Figure 9. Internal Architecture of the XILINX I/O Interface



The internal architecture of the XILINX chip is sown in Figure 9. The DSP bus interface circuit receive the control, data and address signals from the DSP and together with the address decoding circuit doing the write and read tasks of the chip internal registers. The internal registers of the XILINX circuit are enumerated in Table 1.

Table 1. XILINX Circuit Internal Register Values

NAME	ADDRESS	ACCESS MOD
PWMA	F800h	RW
PWMB	F801h	RW
PWMC	F802h	RW
SETWD	F803h	WR
DACCTRL	F804h	RW
DADATA	F805h	WR
ADDATA	F805h	RD
ADCSTART	F806h	WR

The PWMA, PWMB and PWMC registers contain three data defining the actual PWM values. The dates from the registers are written in the same cycle in the internal readable registers.

The data written into the SETWD register define the status of the watchdog signal. The DACCTRL register deserve for setting of desired channel as a chanel which can be write, and for setting the functionality of the 4-channel D/A converter. In the here selected channel (or the selected ones) the data is written to the DADATA register.

By writing in the ADCSTART register the circuit measures the 6 channels of the A/D converter, and, with the help of an interrupt signal, inform the DSP processor of the channel's measurement termination. The data's stored in the internal FIFO can be read from the ADDATA register.

Memory Configuration

The memory map of the processor can be viewed in Figure 10. The memory state after the reset can be viewed in Figure 7 while the EPROM chips state after power-off can be viewed in the Figure 10. Because of the low speed of the EPROM memories is recommended to copy the contents into the RAM memory, running the programs from here. The board allow this operation through the turn-off possibility of EPROM's, when the RAM memory serve both as program, and data memory. It was previously mentioned before in the processor description, that the external memories can not be accessed parallely in the same time. For this reason after removing power off from the EPROM's chips the physical separation of program and data memory can not take place. This mean that with the same logical address in the programme and data memory physically the same memory is addressed. This memory configuration problem must be taken into account in the software linking.

The I/O devices on the board can be accessed via data memory address. The F000h - FFFFh address region map is shown in Figure 11. The I/O devices address can be seen in Table 2.

Figure 10. Memory Configuration without EPROM

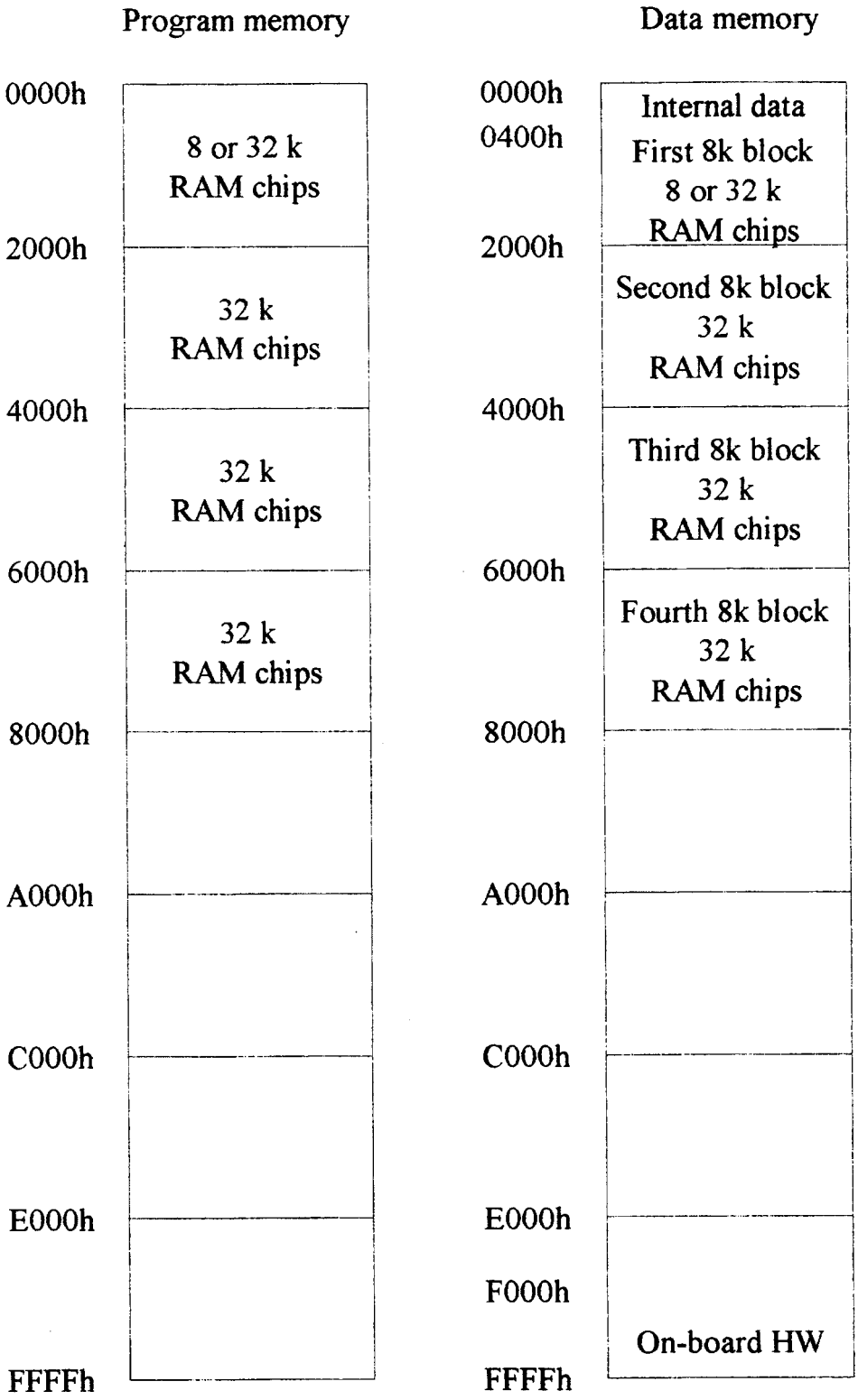


Figure 11. Input/Output Address Region Map

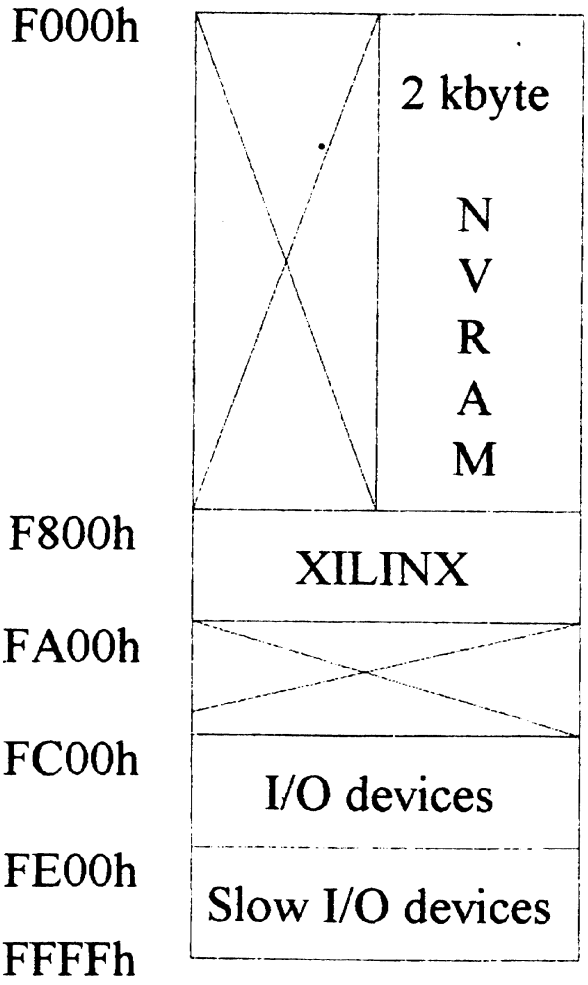


Table 2. Input/Output Device Addresses

fc00h	NVRAM non-volatile read or write
fc01h	KEYBOARD register read
fc02h	DISPLAY data register
fc03h	DISPLAY control register
fe00h	Serial chip

Burned In Firmware

The board burned-on firmware contain the basic programs. After Rreset the programme check the RAM memory on the board, copy the contain of EPROM into the RAM, without changing the programme memory address after turning off the EPROM's. After these steps, the monitor software is started that aids in the downloading and debugging of the software via the serial interface.

Physical Set-Up

The board size: l55 mm x 155 mm. The mechanical placement of connectors for different type of I/O devices are designed in the way that they can be placed bellow of the DSP board.

DSP Debugger

TI offers a wide range of development tools for the TMS320C25 DSP[1] including an evaluation module (EVM), a software development system (SWDS), and an emulator (XDS/22). At the beginning, we used an SWDS for software development. This is a useful tool if no hardware related instructions are used, but the handling of memory mapped I/O chips or circuitry is not possible because the I/O address range of the target system can be accessed through the emulator cable only. So we developed a program package to provide a debugging tool for these I/O devices (DSP Debugger). The DSP Debugger can be used to debug programs written in TMS320C25 assembly language in the target system. The Debugger consists of two parts: one of them runs on an IBM PC as a user interface while the other runs on the target system (burnt-in). The IBM PC can be used for software development (assembler, linker), also. The communications between the two parts is realized via a serial cable. Either the DSP's serial port can be used – in this case a special serial card is required for the PC - or, the RS-232C serial connection can be used, if a serial chip (UART) and line transceivers are mounted on the target board.

Main Features of the Debugger

1) Programs can be downloaded from the IBM PC into the target board's memory. After assembling the source code and linking the object codes, the program is converted to Intel hex file format. All of the required software development tools are available from Texas Instruments. If the target board contains no writeable program memory, the internal program memory (block 0) can be used. In this case, the downloaded program can be as long as 256 words.

a) Program starting, break point setting, and step-by-step execution of commands are available in the Debugger. Break point setting and step-by-step executing can be done only if the debugged program is loaded into a writeable program memory area.

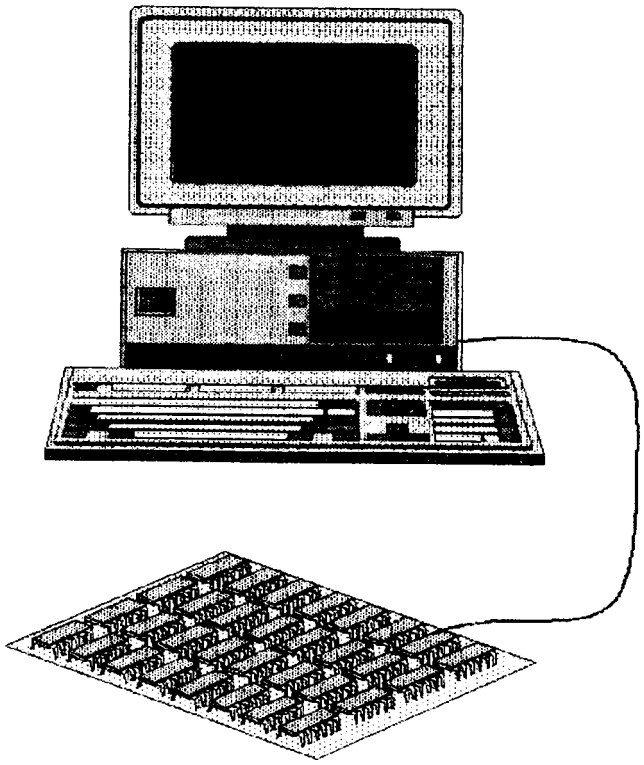
b) The contents of program and data memory, and the DSP's on-chip registers can be examined and modified by menu driven commands.

c) The Debugger can read and write the DSP's input-output ports.

d) There are some special functions and commands for testing the target board's hardware components.

e) The communication between the IBM PC and the DSP target board uses one character control commands and small packets based on Intel hex format. This simple communication method requires a small number of packet types only.

Figure 12. IBM PC – DSP Target Board Communication



Hardware Requirements for the Debugger

There are some restrictions for target boards intended to be used with the DSP Debugger.

- Memory requirements: the Debugger needs approximately 1 kword ROM program memory and 30 words of data memory. For that latter one the DSP's on-chip RAM Block 2 is reserved.
- Appropriate serial port (described previously).
- Writeable program memory for the user software, if the debugged routine is longer than 256 words.
- If step-by-step program execution is required, the Debugger uses the TRAP instruction.

Description of the Debugger

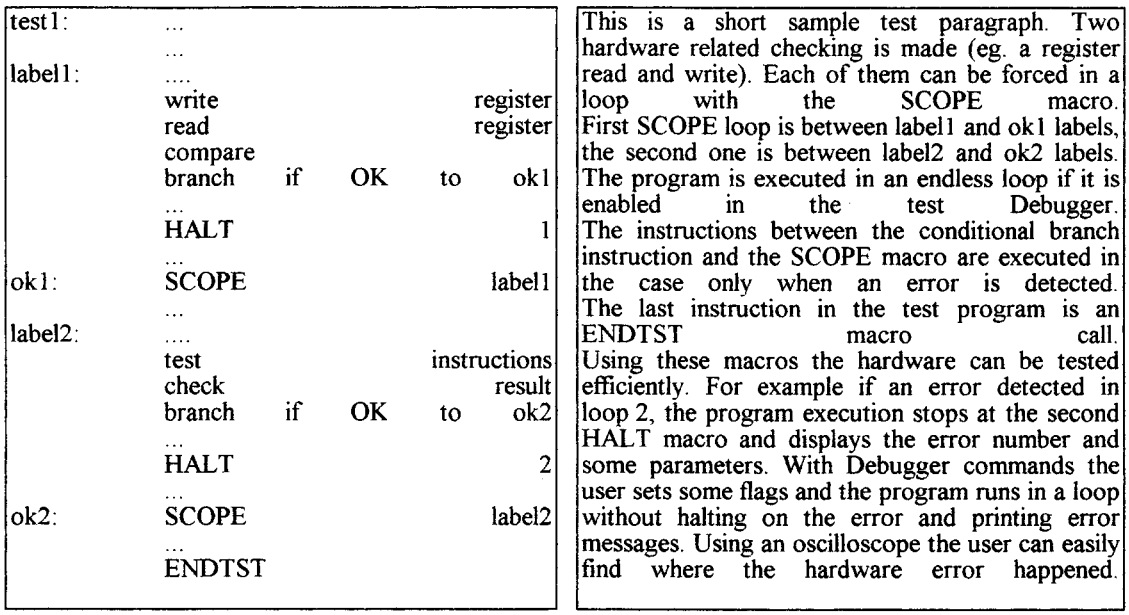
The DSP Debugger program has a simple, debugger-like user interface. The user is able to start and stop programs, display and modify the contents of the program, data memory, and the memory-mapped registers. There is a more powerful user interface that can be used during program development. The list of the debugged program is displayed on the screen. The breakpoints and the next instruction to be executed are shown with another display attribute. Instead of unassembling the program memory contents the Debugger uses a special list file that can be generated from the assembler list file. Using this user interface, program debugging will be more simple and comfortable.

Special Test Feature of the Debugger

There are some special functions and commands for testing the target board's hardware components. The testing method used in the Debugger serves two purposes: to explore the manufacturing defects (workshop test) and to examine the board's operational capability by the user (BIST). This testing theory is used for testing our non-intelligent VMEbus I/O boards, too[12] .

The test programs can run in two modes: in workshop test mode and in BIST mode. There are three built-in macros that can be executed from a test program:

Figure 13. Sample Test Program



```
test1:      ...
            ...
label1:     ....
            write                    register
            read                     register
            compare
            branch   if   OK   to    ok1
            ...
            HALT                     1
            ...
ok1:        SCOPE                    label1
            ...
label2:     ....
            test                     instructions
            check                    result
            branch   if   OK   to    ok2
            ...
            HALT                     2
            ...
ok2:        SCOPE                    label2
            ...
            ENDTST
```

This is a short sample test paragraph. Two hardware related checking is made (eg. a register read and write). Each of them can be forced in a loop with the SCOPE macro. First SCOPE loop is between label1 and ok1 labels, the second one is between label2 and ok2 labels. The program is executed in an endless loop if it is enabled in the test Debugger. The instructions between the conditional branch instruction and the SCOPE macro are executed in the case only when an error is detected. The last instruction in the test program is an ENDTST macro call. Using these macros the hardware can be tested efficiently. For example if an error detected in loop 2, the program execution stops at the second HALT macro and displays the error number and some parameters. With Debugger commands the user sets some flags and the program runs in a loop without halting on the error and printing error messages. Using an oscilloscope the user can easily find where the hardware error happened.

1) **HALT** macro:

   A test program calls this macro if an error is detected. In workshop test mode the Debugger prints an error message to the IBM PC display with a unique error identification number and some parameters. The test program is suspended and the Debugger takes over the control so the user can examine the contents of the memory, registers, etc. The printing of error messages and suspending the test program can be disabled by debugger commands.

   As no user intervention is expected in BIST mode, the testing is suspended and the error status is signaled via the BIST status.

2) **SCOPE** macro:

   In workshop test mode the SCOPE macro is used to force the test program to execute a test loop. For example, a register read-write-verify cycle can be executed in a loop, so the hardware can be checked with an oscilloscope. In BIST mode this macro is ignored.

3) **ENDTST** macro:

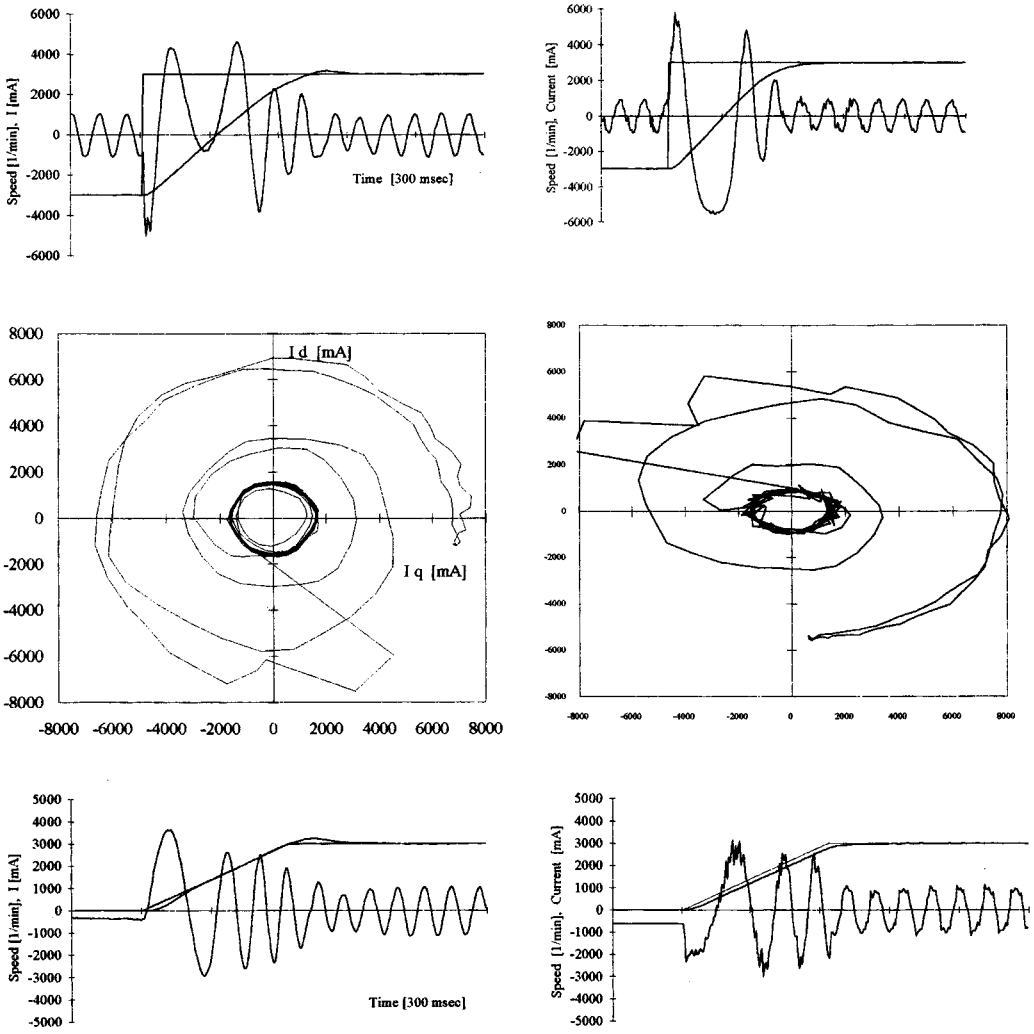
   This is the last instruction executed by the test program. In workshop test mode the Debugger takes over the control. In BIST mode the test handler starts the next test paragraph.

Experimental Results

Figure 14 shows some characteristic results of the control system for different types of controllers (PI, fuzzy) and different reference speeds. The upper two figures show the signal response of the control system to a step change in the reference speed from -3000 min^{-1} to 3000 min^{-1}. The speed, the reference speed, and one phase current are shown. On the left side, the results of PI type controllers are shown while on the right side the results of fuzzy type controllers. There is 50 ms between two ticks on the time axes.

The bottom two figures show the dynamic behaviour of the motor with linear reference from 0 to 3000 min^{-1}. The speed, the reference speed and one phase current are shown. The drive shows good dynamic behaviour.

Figure 14. Characteristic Control System Results



Conclusions

The availability of low cost microelectronics means that the field-oriented theory can be widely used in industrial applications. The software and hardware configurations are presented with experimental results in case of a conventional PI type controller and a fuzzy type controller. The FLC can be successfully used in the field of induction motor control due to the robustness of the controller.

1) This solution can be used in variable speed servo drives. A low cost DSP was used.

2) The algorithm considers the dead time caused by the current filters and measurements and the program running time.

3) No external hardware current control loop is required; the program calculates the stator voltages corresponding to the necessary currents of the motor.

4) If the calculated stator voltage is greater than the output voltage of the inverter the program limits only the q-axis component of the stator voltage if necessary.

5) The simulation and experimental results show good dynamic behaviour in both cases.

Symbols

$\underline{U}_s$, $\underline{U}_r$	stator, rotor voltage vector
i_s, i_r	stator, rotor current vector
$\underline{\Psi}_s$, $\underline{\Psi}_r$	stator, rotor flux vector
u_{sd}, u_{sq}	components of the stator voltage in the d-q frame of reference
$\underline{\Psi}_{rd}$, $\underline{\Psi}_{rq}$	the d-q frame of reference stator and rotor resistance
R_s, R_t	stator and rotor resistance
$\overline{R}$	equivalent resistance of the motor
L_s, L_r, L_m	stator, rotor, and magnetizing inductance respectively
σ	total leakage factor
J	moment of inertia
m_r	load torque
i_{sd}, i_{sq}	components of the stator current vector in the d-q frame of reference
i_α, i_β	components of the stator current vector in the stationary reference frame
$\underline{\Psi}_{ref}$	reference value of the flux
$d\lambda_r$, $q\lambda_r$	d-q axis of the reference system fixed to the rotor flux
$\omega_{\lambda r}$	rotor flux angular velocity
λ_r	position of the flux from the fixed slip
ε_s	position of the stator current vector from the fixed reference axis
ω_r	measured angular speed of the rotor
ω_{slip}	slip frequency
ω_s	stator current vector angular speed
ω_{ref}	reference value of the angular velocity

I^*_d, I^*_q reference values of i_{sd}, i_{sq}

I'_d the new reference value of the flux generating current

Relevant Literature

[1] TMS320C25 User's Guide (Texas Instruments)

[2] MULTIBUS II Bus Architecture Specification Handbook (Intel, Order No.146077)

[3] MULTIBUS II Transport Protocol Specification and Designer's Guide (Intel, Order No.453508)

[4] ANSI/IEEE Std1014 1987. IEEE Standard for A Versatile Bus: VMEbus

[6] MBRT 4.05.01.0 8-channel Sample & Hold Input Analog-to-Digital Converter Operating Manual (KFKI MSZKI, July 1992)

[7] MBRT 4.10.01.0. 8-channel Digital-to-Analog Converter Operating Manual **(**KFKI MSZKI, July 1993)

[8] ERT 2.01.01.0 Quad Scaler User's Manual (KEKI MSZKI, 1992)

[9] ERT 2.01.01.0 Quad Scaler Programmers Guide (KFKI MSZKI, 1992)

[10] ERT 2.23.01.0 Status Change Monitor User's Manual (KFKI MSZKI, 1992)

[11] TMS320 Family Development Support Reference Guide (Texas Instruments)

[12] M. Barkaszi, Z**.** Katona: *UBI Specific Test Software Environment,* XV International Symposium on Nuclear Electronics and International Seminar CAMAC-92, Warsaw, 29 September - 2 October 1992, pp.204 - 222

[13] IBM PC/AT VME Adaptor Model 403 (B1T3 Computer Corporation, pub. no.82003010)

[14] Ben-Brahim, L., Kawamura, A.*: A Fully Digitized Field-Oriented Controlled Induction Motor Drive Using Only Current Sensors*. IEEE Transactions on Industrial Electronics, Vol. 39, No.3, June 1992, pp. 241-249.

[15] Blaschke, F.: The Principle of Field Orientation as Applied to the New TRANSVECTOR Closed-Loop Control System for Rotating Field Machines. Siemens Rev, 1972, pp. 217-226.

[16] Hainemann, G., Leonard, W.: *Self-Tuning Field Oriented Control of an Induction Motor Drive*. International Power Electronics Conference, Tokyo 1990, pp. 465-472.

[14] Halasz, S.: Automatizalt Villamos Hajtasok. Tankonyvkiadó, Budapest, 1989.

[15] Kelemen, A., Imecs, M.: *Vector Control of AC Drives*. OMIKK Publisher, Budapest, 1991.

[16] Leonard, W.: Control of electrical drives. Springer Verlag, Berlin, 1985.

[18] Lessmeier, R., Schumacher, W., Leonard, W.: *Microprocessor-Controlled AC-Servo Drives with Synchronous or Induction Motors: Which is Preferable?* IEEE Transactions on Industry Applications, Vol. IA-22, No.5, Sept./Oct. 1986, pp.812-819.

[19] Rodriguez, J., Kastner, 0.: Non-linear current control of an inverter-fed induction machine. Etz Archive Bd.9, 1987 H.8, pp. 245-250.

[21] Vainio, 0., Ovasaka, S. P.: A Digital Signal Processing Approach to Real-Time AC Motor Modeling. IEEE Transactions on Industrial Electronics, Vol.39, No.1, February 1992, pp. 36-45.

[22] 0. Klir and Tina A. Folger: Fuzzy Sets, Uncertainty, and Information, Prentice-Hall, 1988.

[23] H.J. Zimmermann: Fuzzy Set Theory and its Applications, Kluwer Academic Publishers, Boston/Dordrecht/London 1990.

[24] D. Naunin, C. Karaali: Fuzzy Controller and Digital Cascaded State Controller for an Intelligent Synchronous Servodrive, Proceedings of IECON'94 Conference, Vol. II, pp 1298-1303.

[25] D. Fodor, Z. Katona, and E. Szesztay: Digitized Vector Control of Induction Motor with DSP, Proceedings of IECON '94 Conference, Vol III, pp 2057-2062.

[26] B. Kosko: Neural Networks and Fuzzy Systems, Prentice-Hall, 1992.

[27] D. Fodor, Z. Katona, E. Szesztay: Field-Oriented Control of Induction Motors Using DSP, Computing & Control Engineering Journal, Vol. 5. no 2, pp 61-65.