Enhanced Control of an Alternating Current Motor Using Fuzzy Logic and a TMS320 Digital Signal Processor

Dr. Stefan Beierke
Texas Instruments

Ralph Königbauer, Bernhard Krause, Constantin von Altrock
Inform Software Corporation

SPRA057
March 1996

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 applications 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.

Abstract

Fuzzy logic technology enables the use of engineering experience and experimental results in designing an embedded system. In many applications, this circumvents the use of rigorous mathematical modeling to derive a control solution. The benefits of using fuzzy logic technologies in control system design have already been shown in a wide variety of applications and products. In the past, however, the computational resources needed to implement the fuzzy logic algorithm required special hardware such as dedicated fuzzy logic coprocessors or slow microcontrollers (MCUs) which limited the bandwidth of the system. A standard microcontroller using fuzzy logic does not provide the necessary performance for controlling an ac (induction) motor, and the cost of a fuzzy coprocessor is prohibitive for high volume applications. This report shows how the use of a digital signal processor (DSP) with a specialized fuzzy logic software kernel provides the required computing performance while maintaining a low cost. This report presents the following:

- A fuzzy logic design that enhances the system's ability to handle the abrupt momentum changes of an ac motor controller.
- The software technology used to implement the fuzzy logic design.

Test results obtained on a real-world motor control application show the performance boost that can be gained by using fuzzy logic on a DSP.

Introduction

Various market analyses show that 90% of all industrial motor applications use induction type motors. The many reasons for this include high robustness, reliability, low price, and high efficiency (up to 80% higher). However, the use of induction motors has several disadvantages. These include difficult controllability due to the complex mathematical model of the motors, their nonlinear behavior during saturation, and oscillations of the electrical parameters due to the physical influence of temperature. For example, the rotor time constant of an induction motor can change up to 70% over a motor's temperature range.

This paper presents a case study based on a well-known mathematical model for an induction motor. The fuzzy logic control approach was chosen to achieve a very short implementation time, high robustness towards the effects of temperature and saturation, and good lead and disturbance behavior.

The rest of this report deals primarily with the rapid implementation of fuzzy logic control in an induction motor. It shows the design of a fuzzy logic controller as well as the design process. The paper also compares the performance of a fuzzy logic enhanced controller to the standard field-oriented control approach.

Induction Motor Control Concepts

The conventional control schemes for induction motors were developed for use in a variety of situations, including control quality, performance, and controller design complexity. The common methods are:

- Stator voltage to stator frequency (V_S/f1) method
- Stator currents and open-loop flux control (I_S/f2) method
- Field-oriented control method

A comparison of the three methods is given in [1]. The field-oriented control method achieves the best dynamic behavior, but requires shorter control cycle times for good lead and disturbance behavior [2]. The field-oriented control method is the de facto standard for controlling an induction motor in adjustable speed drive applications. This is true for quickly changing loads as well as for reference speeds.

Table 1. Motor Parameter Values

PARAMETER	SYMBOL	VALUE
Nominal power	P_n	0.41 kW
Nominal velocity	v	3000 rpm
Nominal voltage	U	190 V
Nominal current	I	2.4 A
Stator resistance	R_S	1.68 Ω
Stator inductance	L_S	0.125 H
Rotor time constant	T_R	40 ms
Total leakage	σ	0.084
Inertia	J	2.8 kgcm^2

The Field-Oriented Control Approach

The principle of field-oriented control [3,4] is shown in Figure 1. With this control method, an induction cage machine can be controlled like a separately excited dc machine in which the direct (d–) path represents the flux component and the quadrate (q–) path sets the electrical torque.

The best dynamic behavior is obtained when the magnetizing current, i_{mR}, is kept constant and is in direct proportion to the rotor flux, Ψ_R. This assumes that the main inductance, L_h, is constant.

$$U_{Sd} = R_S \cdot i_{Sd} + \sigma \cdot L_S \cdot \frac{d}{dt} i_{Sd} - \sigma \cdot L_S \cdot \omega_{mR} \cdot i_{Sq} + (1 - \sigma) \cdot L_S \cdot \frac{d}{dt} i_{mR} \quad (1)$$

$$U_{Sq} = R_S \cdot i_{Sq} + \sigma \cdot L_S \cdot \frac{d}{dt} i_{Sq} + \sigma \cdot L_S \cdot \omega_{mR} \cdot i_{Sd} + (1 - \sigma) \cdot L_S \cdot \omega_{mR} \cdot i_{mR} \quad (2)$$

$$i_{Sd} = i_{mR} + T_R \cdot \frac{d}{dt} i_{mR} \quad (3)$$

$$i_{Sq} = (\omega_{mR} - \omega) \cdot T_R \cdot i_{mR} \quad (4)$$

$$m_d = \frac{2}{3} \cdot z_p \cdot (1 - \sigma) \cdot L_S \cdot i_{mR} \cdot i_{Sq} \quad (5)$$

$$\frac{J}{z_p} \cdot \frac{d}{dt}\omega = m_d - m_l \quad (6)$$

Based on the mathematical model of the induction cage motor in the field coordinates given in equations (1) through (6) above, a cascaded position controller was developed for this motor, and a flux model was derived. The model calculated the angle between stator and field system to transform the stator variables into field-oriented coordinates. The flux model used two of the three stator currents and the mechanical rotor position, ε, as input variables

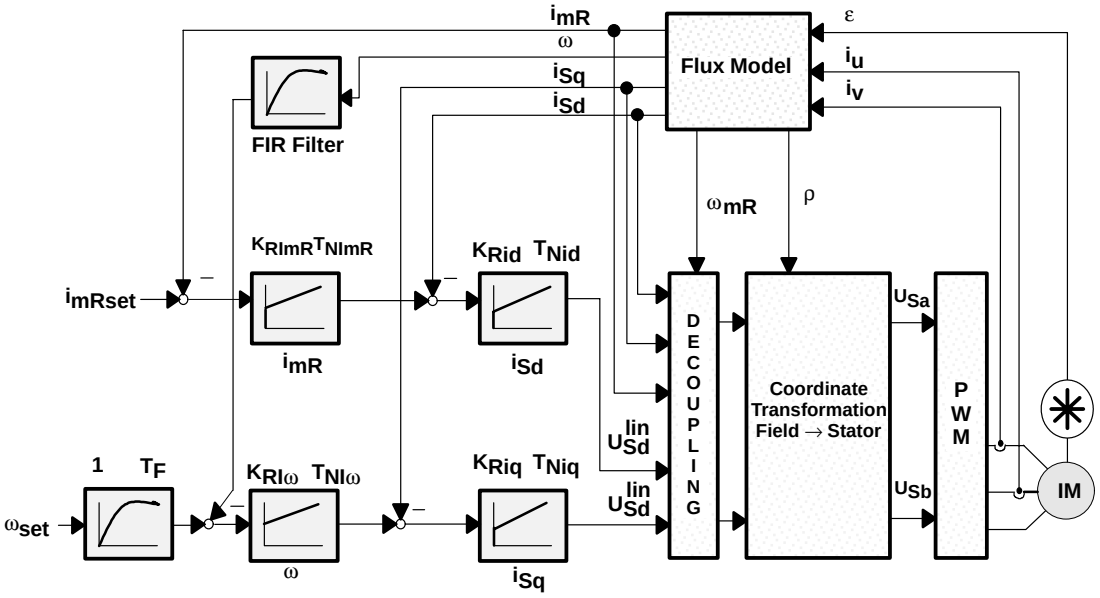

Figure 1. Field-Oriented Control Method

The left side of Figure 1 shows the two independent cascaded controllers developed for this application. The upper one consisted of an outer rotor flux controller (i_{mR}) with an inner current controller (i_{Sd}) . The lower cascaded control structure consisted of a superposed position controller, a velocity controller, and an inner current controller (i_{Sq}). To achieve better lead behavior, the velocity control unit shown in the diagram had a first order delay lag in the velocity reference value, and implemented a finite impulse response (FIR) filter to reduce actual velocity ripple.

The primary purpose of this controller design was to allow no overshoot of the desired velocity or position, and to provide good disturbance behavior. Overshoot in this design was reduced to zero with a first order delay lag. The design also showed good disturbance behavior on both inner proportional integral (PI) current controllers by using a short control cycle time. The current controller design used the optimized amplitude adaptation method. The velocity controller design used the symmetrical optimum method.

Both current controllers calculated output values for the linear part of the mathematical model of the ac machine. Decoupling nonlinear terms were added, as shown in the decoupling block in Figure 1. After transforming the field coordinates to stator coordinates, the output voltage was applied to the machine using the symmetric subharmonic pulse width modulation (PWM) method, which computed the pulse pattern on-line.

Fuzzy Logic Control

Some advantages of using fuzzy logic system control include short development times, easy transfer to different motor sizes, and a strong tolerance for electrical motor parameter oscillations. For this report, a fuzzy logic control system for an ac motor was implemented using the following methods, performed in consecutive steps:

1. A fuzzy system was developed that described a nonlinear function between slip frequency and stator current. To integrate the nonlinear function into a fuzzy system, the fuzzy system was adapted to the function characteristic using a neurofuzzy approach, which is described below.
2. The fuzzy logic controller calculated the slip frequency by the taking the difference between the stator and rotor frequencies.

The first step was based on the structure of the stator currents and the open-loop flux control (Is/f2) characteristic, which is shown in Figure 2. The fuzzy logic controlled system consisted of two control loops: an inner stator current control loop and an outer speed loop. The inner current loop controlled the three stator phase currents with normal PI controllers. The outer speed loop calculated the slip frequency, n_2, also using a standard PI controller.

The slip frequency was the input variable for the fuzzy logic block shown in Figure 2. The output value was the stator reference current. The fuzzy block provided a constant magnetizing current in all operation modes. This magnetizing current was a nonlinear function of the slip frequency, the rotor time constant, the rotor leakage factor, and a nonconstant offset current.

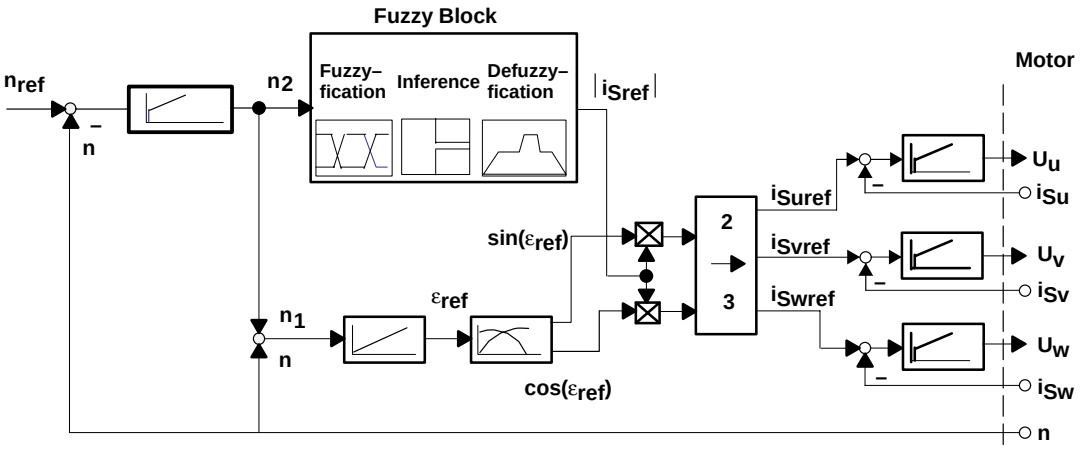

Figure 2. Fuzzy Logic Basic Control Method

Figure 2 shows how the stator frequency, n_1, was the sum of the measured rotor frequency, n, and the slip frequency, n_2. The actual reference position, ref, was the result of the integration of the stator frequency. ε_{ref} was modulated with a sine and a cosine function, and the result was multiplied by the i_{Sref} value. The two-phase system was split back into a three-phase system, resulting in a stator current reference value for each phase.

Based on the shape of the magnetizing current and the range of the motor parameters, a database was generated. The shape of the magnetizing current was symmetrical to the slip frequency when n_2 equalled 0. The database included only positive slip frequencies, which achieved a higher resolution.

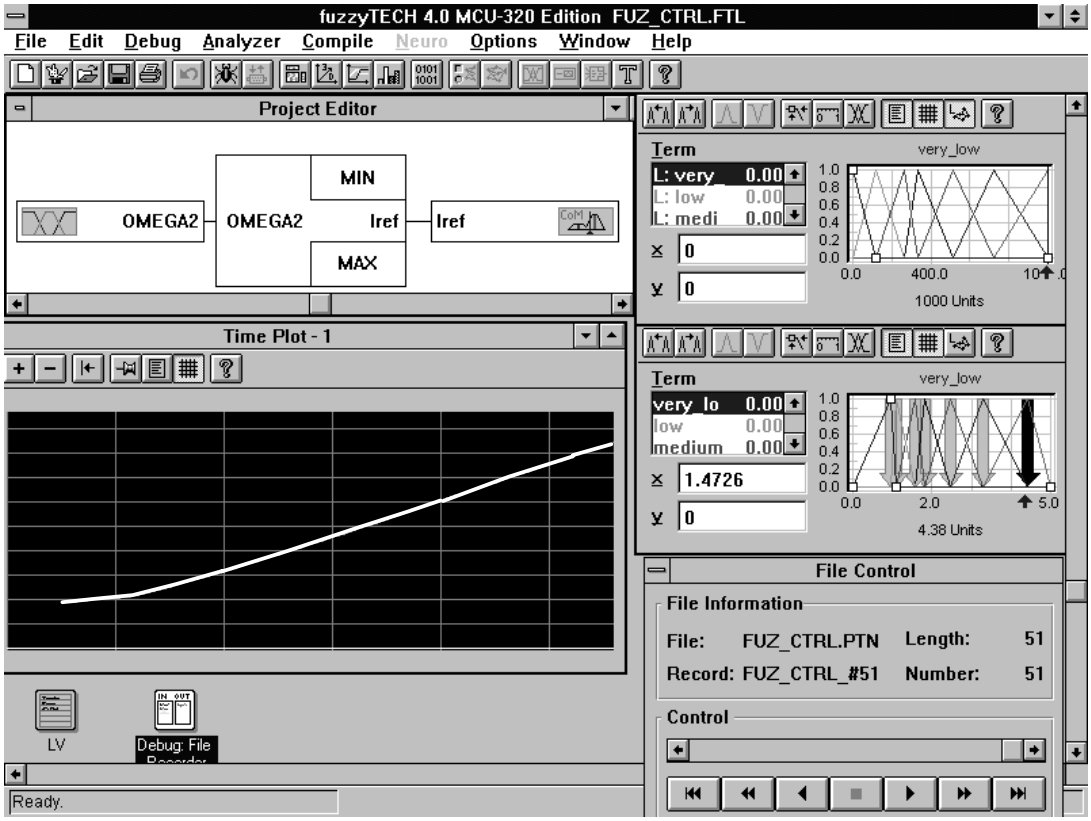

Figure 3. Neurofuzzy Design of the Nonlinear Magnetizing Characteristic Using the fuzzyTECH™ NeuroFuzzy Module

Neurofuzzy techniques allow a fuzzy logic system to adapt the contents of a database representing the desired system behavior. Adaptation refers to setting up the variables for fuzzification and defuzzification during implementation of a fuzzy logic system. For this procedure, neurofuzzy training, or database adaptation through neural logic, used fuzzyTECH's supervisor mode to adapt both fuzzy rules and language variables in the fuzzification and the defuzzification procedure. Implementation of this system took four days, including debugging and testing. The results are shown in the Results and Comparison Section. The results of training are shown in Figure 3. The upper right side of the figure shows the seven input terms and below them, the seven trained output terms. The time plot window of Figure 3 shows the expected behavior of the stator reference current versus the slip frequency.

Enhanced Fuzzy Controller

Next , an enhanced fuzzy logic controller was developed which included two fuzzy logic blocks: one that calculated the magnetizing current, and the other the slip frequency, n_2. The structure of the enhanced fuzzy logic controller is shown in Figure 4.

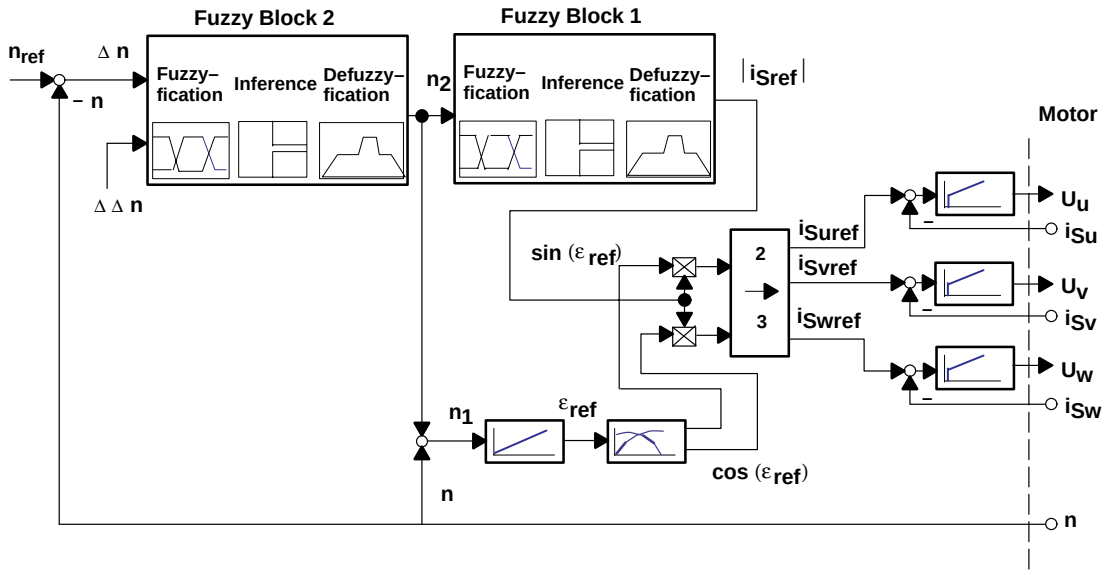

Figure 4. Fuzzy Logic Enhanced Control Method

Fuzzy logic block 1 in Figure 4, was responsible for providing a constant magnetizing current during implementation. The block is identical to the design shown in Figure 3. Fuzzy logic block 2 had two input variables: the deviation of the speed, Δn, and its gradient, ΔΔn (acceleration). The second fuzzy logic block replaced the speed PI controller of the first implemented fuzzy system. This replacement provided the reason for the influence of speed measurement noises during low speed operations (the special region at slip frequency, $n_2 = 0$), and gave an improvement in the control of overshoot behavior versus the classic PI controller.

Fuzzy logic block 2 is shown implemented in Figure 5. The procedure used the following variables:

- The input variable d_omega, which was equal to Δn. d_omega was divided into seven input terms. Three of these were close in value to d_omega equals 0. This increased the sensitivity of the fuzzy controller at d_omega near zero.
- The input variable dd_omega, which was equal to ΔΔn. dd_omega was divided into three terms and had less influence than d_omega.
- The output variable d_slip, which was divided into five terms. Three of these output terms were near in value to d_slip equals zero. This achieved a high sensitivity over the range, and reduced overshoots.

Implementation of the fuzzy slip frequency estimator took one day, and its optimization took two additional days. The entire enhanced fuzzy logic controller was implemented in seven days.

Further fuzzy logic controller optimizations could include an air-gap flux observer based on the output voltages, the measured speed, and the measured currents. Knowing the air gap, it is possible to implement

a closed loop controller for the magnetizing current. This air-gap controller would further improve the system's dynamic behavior.

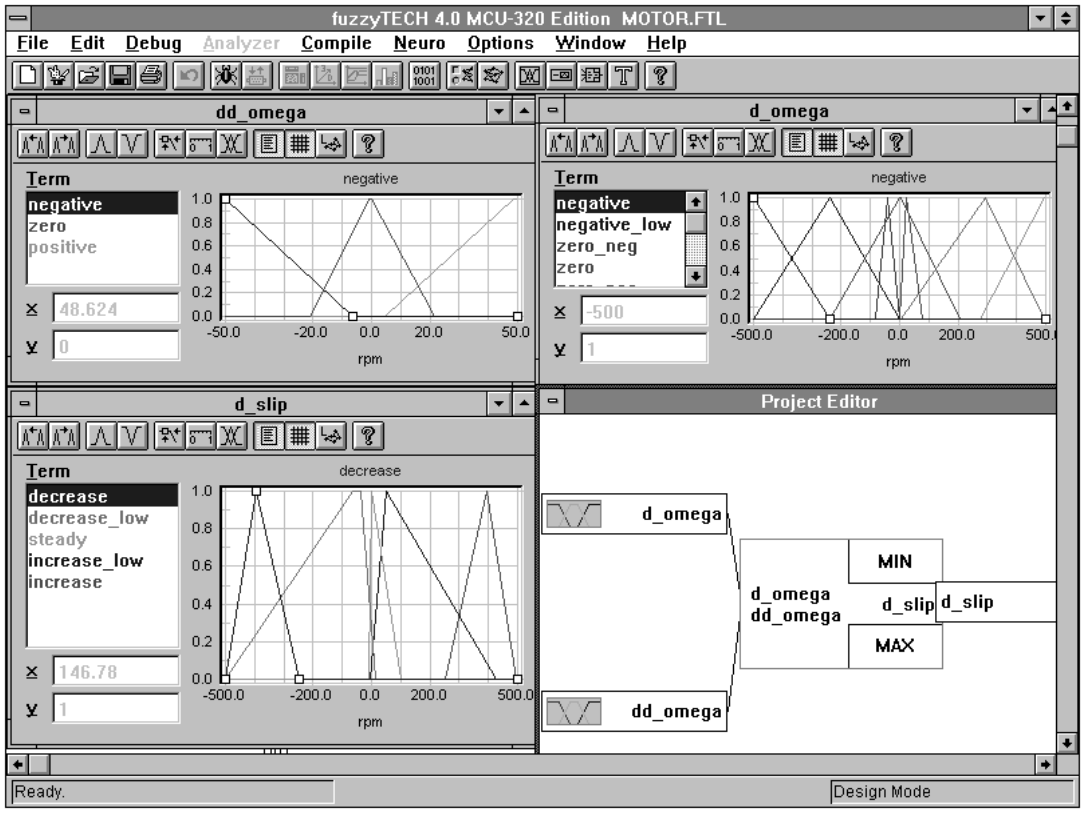

Figure 5. fuzzyTECH Fuzzy Logic Controller for the Design of a Fuzzy Slip Frequency Estimator

Simulation of Control Concepts Using fuzzyTECH and Matlab™ Simulink™

During development of the fuzzy logic controllers, prototypes were created and tested using simulations. The fuzzy logic workbench, fuzzyTECH, enabled the easy design and redesign of these prototypes for testing purposes. fuzzyTECH contains an M code generator that allows fuzzy logic systems to execute in a simulation environment. In addition, a fuzzy logic system can be generated in C code, which may be compiled and integrated in Matlab Simulink as a DLL or MEX file, depending on the C compiler used.

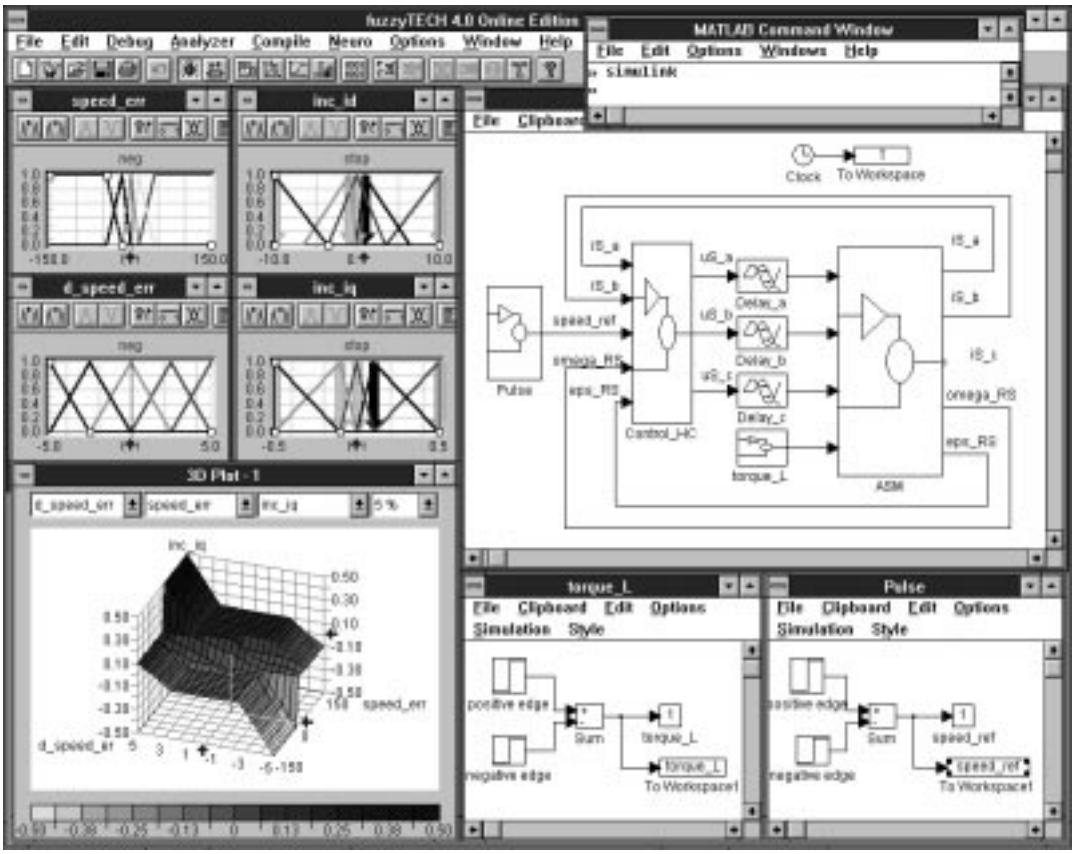

Figure 6. Integration of a fuzzyTECH Controller Into an ac Motor Model Created in Matlab Simulink

Hardware and Software Environment

Figure 7 shows a diagram of the target hardware. The control software operated on a system using a dSPACE DS1102 processor board, a commercial PC board using a TMS320C31 floating-point DSP [5], and a TMS320C14 as a pulse-pattern generator. Additional DS1102 peripherals [9] used in the system setup are shown in Figure 7. These include analog-to-digital (A/D) analog-to-digita) converters and incremental encoder interfaces.

A standard PC served as host for the software development environment. It used fuzzyTech for generation of the fuzzy logic controllers and a Texas Instruments floating-point DSP C compiler, along with dSPACE's COCKPIT31 software. It also used dSPACE's TRACE31, which is a tool that visualizes all control software variables in realtime [10, 11]. COCKPIT31's instrument panel is a graphic user interface for experiment control. The user can configure the control panel in the Cockpit software's edit mode to display all desired system information. In Cockpit's animation mode, all display elements are updated periodically according to the current DSP variable value, and the input elements transfer new parameters to the DSP's memory.

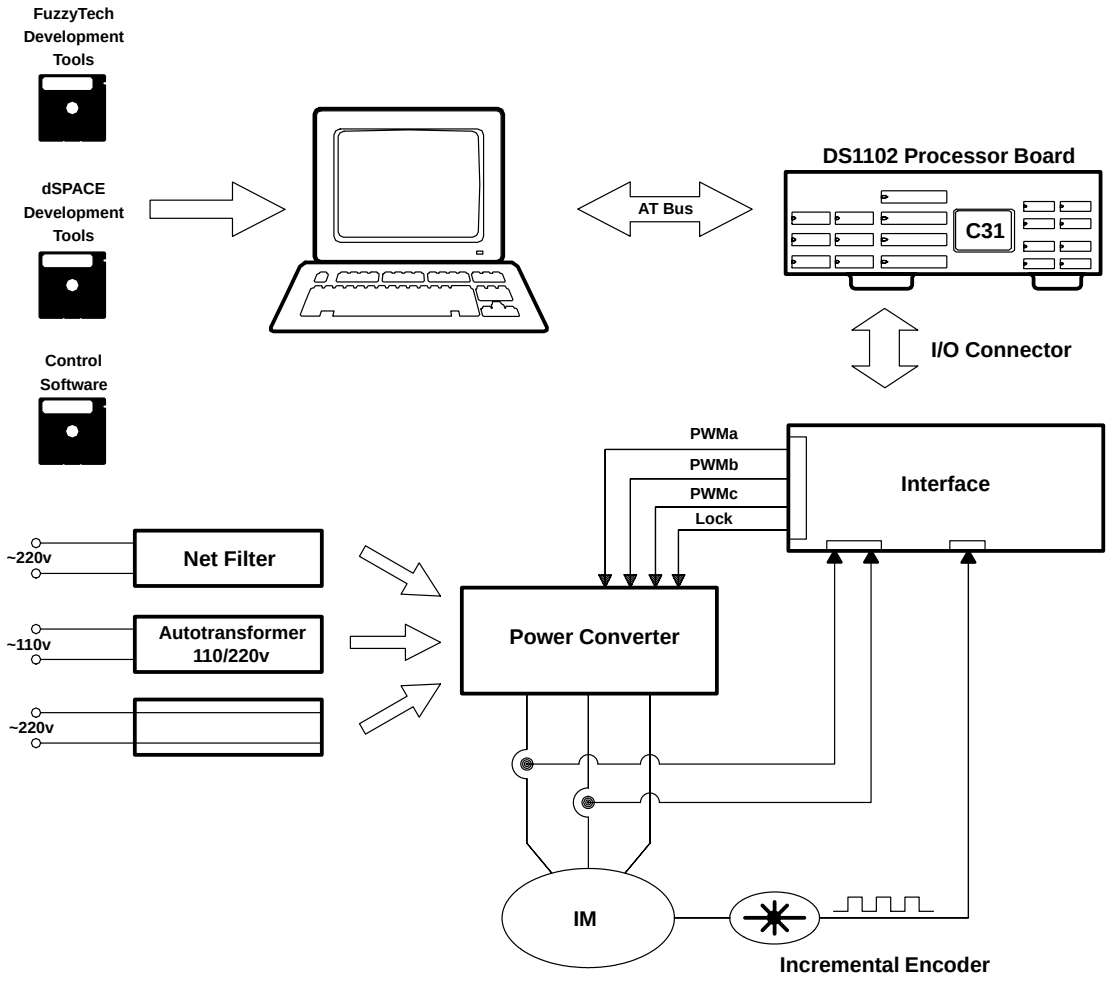

Figure 7. Target Hardware System Setup

The interface block decoupled the digital electronic part of the signal from the power electronic signal. The three output channels, PWMa, PWMb, and PWMc were connected via fiber optic links to the power converter. The power converter used MOSFETs as power semiconductors. The position was measured with an incremental encoder that has a resolution of 14400 increments per revolution.

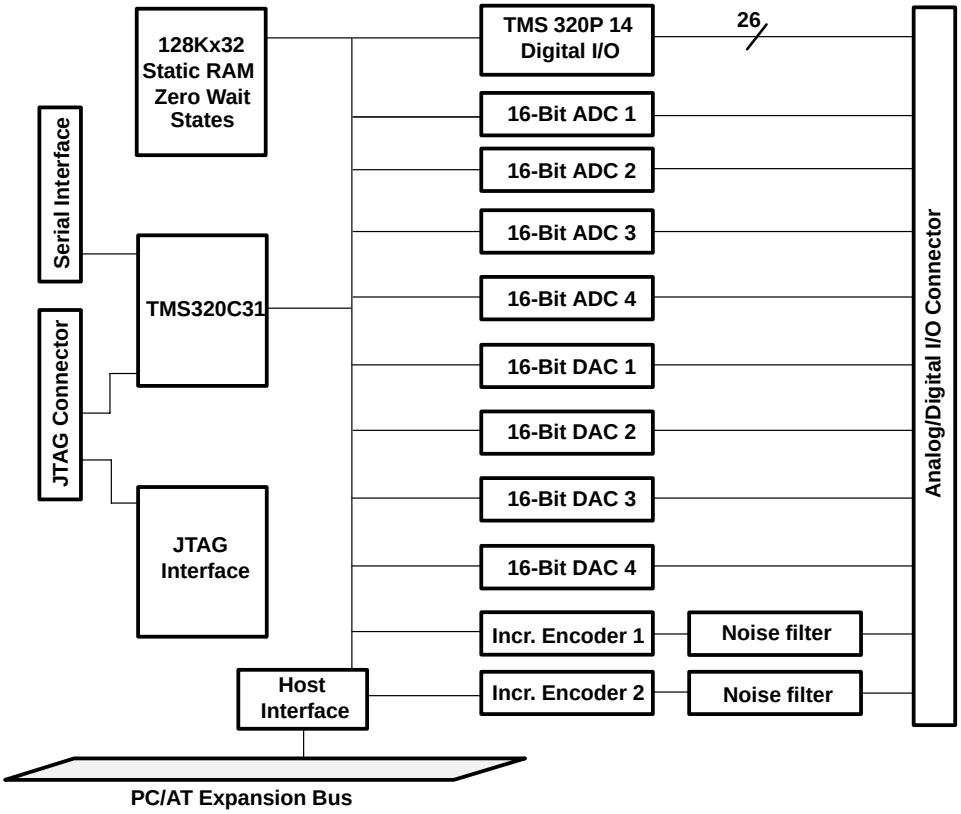

Figure 8. Hardware Design of the DSP PC Board Used in the Target System

The TMS320C31 is a low cost 32-bit DSP that offers the advantages of a floating point processor as well as ease of use. The 40 MHz version of the TMS320C31 used in this test operates with a 50-ns instruction cycle time and offers speed up to 40 million floating-point operations per second MFLOPS.

The TMS32014 is a first generation DSP, and is designed for control system. It operates as a coprocessor on the DS1102 board, and has useful on-chip peripherals such as the event manager. The event manager consists of four capture inputs and six compare outputs, a bit-selectable I/O port, two general purpose timers, and the PWM unit. The PWM supports six independent channels with time resolution of up to 40 ns. The PWM channels control the power converter.

DSP Implementation of Fuzzy Logic Systems

Control of an ac motor requires very fast loop times, on the order of a few milliseconds or less. On a standard 8-bit MCU such as the 8051, the processing of the fuzzy logic system alone requires more than a millisecond. The applications described above require faster processing of the fuzzy logic algorithm than can be supported by these 8-bit microcontrollers.

To expedite fuzzy logic computation, different approaches exist. One approach calls for a dedicated hardware unit, frequently called a fuzzy processor or a fuzzy chip. Another approach uses a DSP in combination with fuzzy logic software emulation, frequently called a fuzzy software kernel.

Using Dedicated Fuzzy Chips

When using fuzzy logic hardware, the sequence of computational steps can be represented with a layer diagram, as shown in Figure 9. The link to the host MCU is shown in Figure 10. The inner layer of Figure 9 is called fuzzy hardware, and provides fuzzy logic computation routines such as fuzzication, rule inference, and defuzzification. The next layer, called the FL system, contains the membership functions and fuzzy rules for this fuzzy logic control strategy.

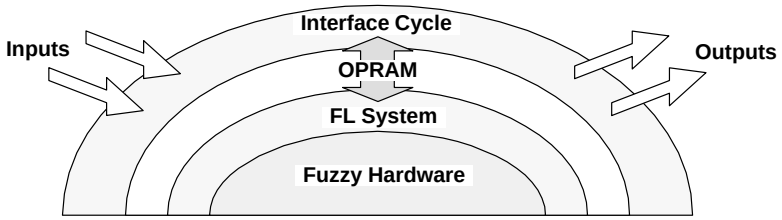

Figure 9. Computational Sequence Using Fuzzy Logic Hardware

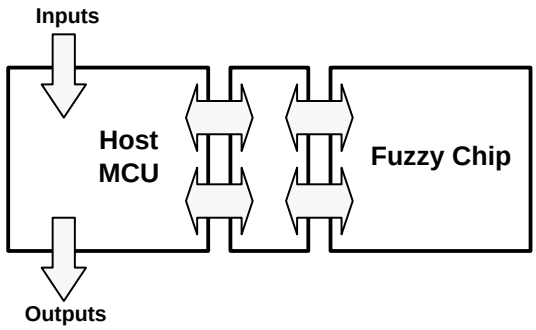

Figure 10. A Fuzzy Chip Linked to the Host MCU by a Dual-Ported RAM for High-Speed Throughput

Presently, fuzzy logic chips are capable of processing only fuzzy logic. Since every application requires preprocessing of input data and postprocessing of output data when linked to the periphery, these computations, called the interface code, must be handled by a host MCU. Thus, two processors and a communication link must be added when using a fuzzy chip. This significantly adds to the cost of the solution. In addition, the two processors must be synchronized and the communication overhead between the two processors slows down the speed gain of the fuzzy hardware over conventional control systems. A state of the art fuzzy logic processor such as VLSI's VY86C500, operating at 20 MHz, performs only 0.87 million rules per second, excluding fuzzification, defuzzification, and communication with the host MCU.

Using Fuzzy Kernels on a DSP

An alternative approach is to use only one processor to emulate fuzzy logic functionality in a software kernel. Such a fuzzy software kernel has been jointly developed by Texas Instruments and Inform Software Corporation, and is available for most members of the TMS320 DSP family. The fuzzy software kernel works in combination with *fuzzy*TECH, the fuzzy logic graphic development environment of Inform Software Corporation [6].

Figure 11 shows the sequence of computational steps in the fuzzy software kernel approach, represented as a layer diagram. Figure 12 shows the one-chip alternative. The *fuzzy*TECH kernel provides the same fuzzy logic functionality as the fuzzy hardware layer of Figure 9. It duplicates the function of fuzzy hardware, using only the software in the DSP. The interface code interfaces directly with the fuzzy logic computation using a simple function call. This means that no second processor and no additional communication hardware or software are required

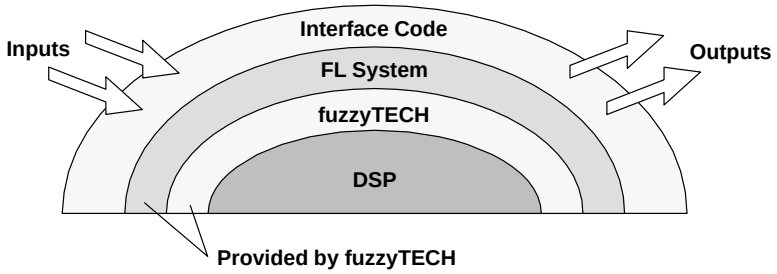

Figure 11. fuzzyTECH Kernel for TI DSP's

The single-chip solution is much less expensive than a multiprocessor solution using fuzzy logic hardware, and is much faster, as well. For example, fuzzyTECH kernel implementation on a standard TMS320C52 DSP with 25 ns instruction time delivers a fuzzy logic computation performance of three million rules per second including fuzzification, rule inference, defuzzification, and communication.

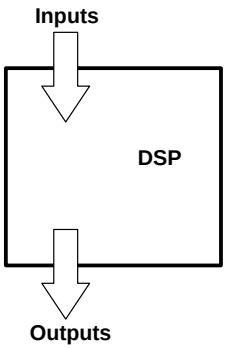

Figure 12. One-Chip Solution Fuzzy Logic Controller and Conventional Control Modules on the Same DSP

Benchmark Comparisons

Figure 13 shows a standard set of fuzzy logic benchmarks [7] that compare the fuzzy logic computation performance of different hardware platforms. The four benchmarks range from a simple positioning controller to an intervehicle dynamics controller with eight inputs, four outputs, and 500 rules.

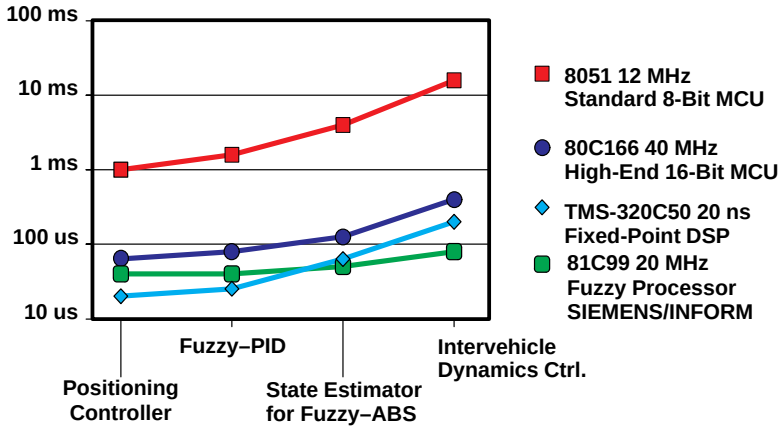

Figure 13. Performance Comparisons: DSP Versus MCUs with Dedicated Fuzzy Processors

These benchmarks compare the 8051/12, as an example of a typical 8-bit MCU, the 80C166/40, as an example of a typical high-end 16-bit MCU, a fuzzy processor (81C99/20), and a fuzzy software kernel implemented on a DSP. The 81C99 fuzzy logic processor is the fastest fuzzy chip commercially available as of this report. The benchmarks show that the DSP with *fuzzy*TECH software outperforms high-end 16-bit MCUs significantly, and even outperforms the fastest fuzzy logic chip available today for small fuzzy logic systems. Thus, fuzzy logic on a DSP is as fast as the advanced fuzzy logic chips and as flexible as MCUs because of its pure software implementation.

Results and Comparison

The following table shows a comparison of the minimum implementation time for each of the three control approaches mentioned in this document. The times include include debugging and testing of the control systems:

Table 2. Implementation Times

CONTROL ALGORITHM	CYCLE TIME	IMPLEMENTATION TIME
Basic Fuzzy Logic Control	150 µs	4 days
Enhanced Fuzzy Control	200 µs	7 days
Field-Oriented Control	100 µs	3 months

To compare the implemented control approaches, speed reversion was used as hard test condition. The reference speed was switched between ± 1000 rpm, generating a rectangular waveform. The results were measured using the same reference values for speed.

The performance requirement of the DSP is important for achieving minimal control cycle time. The differing control cycle times of the DSPs used in the various systems that were benchmarked account for the different periods of the reference speed waveforms, because the reference generator uses control cycle times as a time basis.

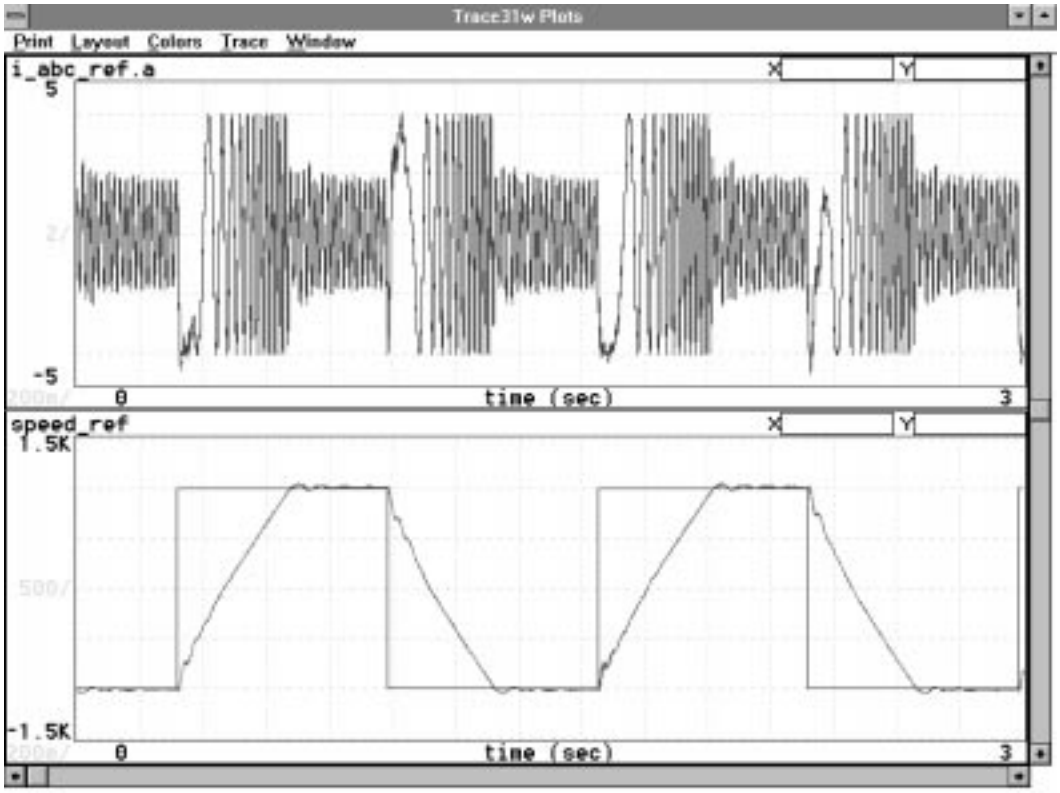

Figure 14. Dynamic Behavior of the Basic Fuzzy Logic Controlled System

Figure 14 shows the results of testing of the first fuzzy logic controlled system. The upper part of the diagram shows the reference current i_abc_ref.a and the measured current i_abc.a of the corresponding stator phase. The curves are nearly identical. This shows that the inner current controller operated correctly.

The lower part of the diagram presents a comparison of the reference speed and the actual measured speed. The speed-reversion process went from –1000 to +1000 rpm, and the reversion time hovered in the range of 400 ms. The speed response also showed a ripple effect in the measured speed.

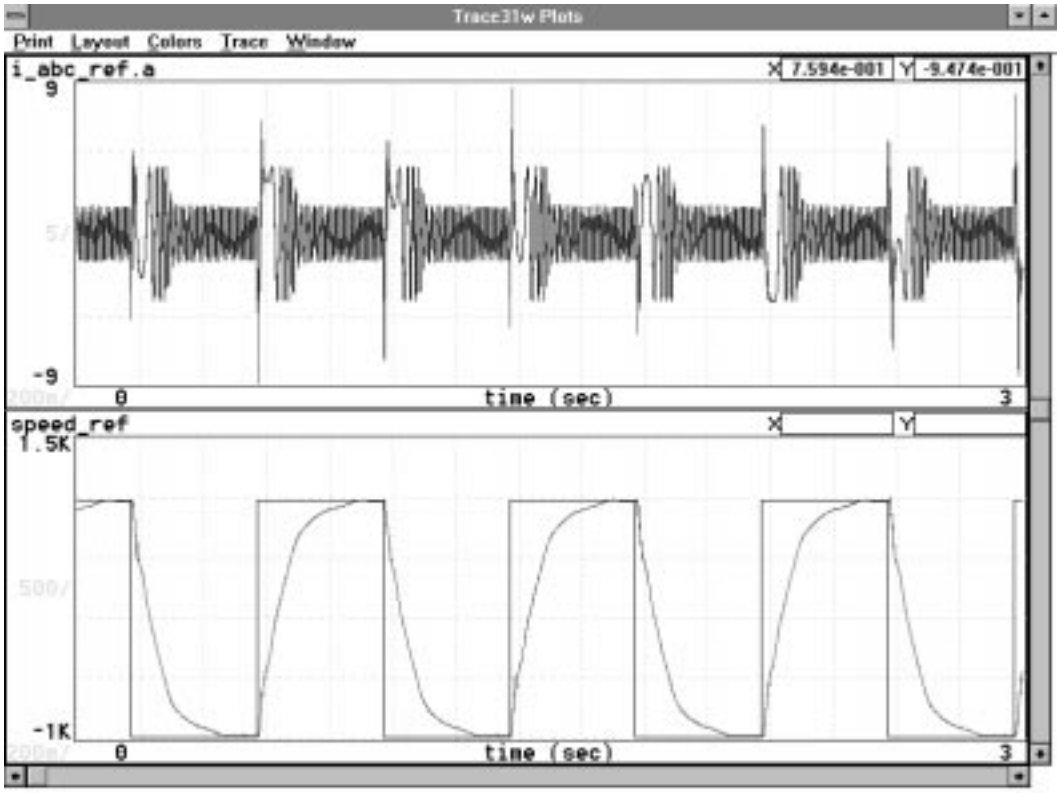

Figure 15. Dynamic Behavior of the Enhanced Fuzzy Logic Controlled System

Figure 15 shows the results achieved by testing the enhanced fuzzy logic controlled system. The upper part of the diagram shows the reference current i_abc_ref.a and the measured current i_abc.a of the corresponding stator phase. The comparison demonstrates that the inner current controller operated correctly in this system, as well.

The lower diagram presents a comparison of the reference speed and the actual measured one. The speed reversion process again went from –1000 rpm to +1000, and the reversion time hovered in the range of 300 ms. This performance was better than the first fuzzy logic controlled system. The speed response showed no overshoot and closely approximated the reference value.

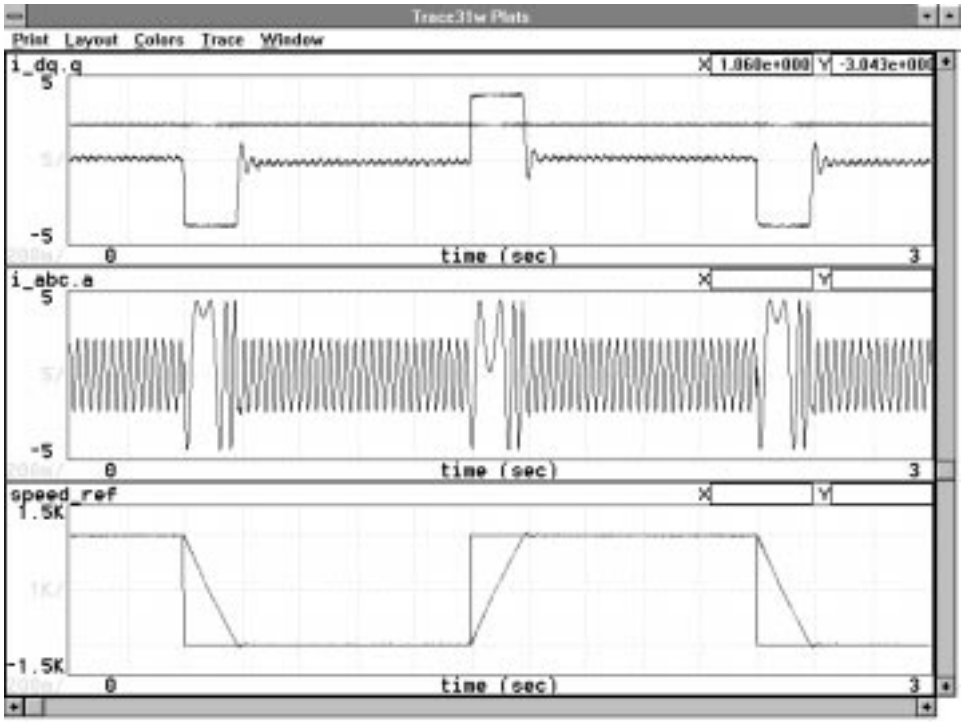

Figure 16. Dynamic Behavior of the Field-Oriented Controlled System

Figure 16 shows three diagrams that were obtained in testing the field-oriented controlled system. The reference value of the magnetizing current was two amperes, and remained constant during speed reversion. The other component of the upper diagram represented the i_{Sq} component, which was proportional to the torque. The limitations of the i_{Sq} shape came from the limitations of the power converter. The middle diagram shows one of the three stator phase currents. Using this diagram it is possible to compare the results of the field-oriented controller with the systems controlled by fuzzy logic. The lower diagram presents a comparison of the reference speed and the actual measured speed. The speed reversion process went from –1000 to +1000 rpm and the reversion time remained in the range of 200 ms. The speed response showed no overshoot. This was the best reversion time of all three control approaches that were compared.

Conclusion

The above comparison of the lead behavior shows that the field-oriented control method achieves the fastest speed reversion time. However, since both fuzzy approaches were developed in only a few days, further improvements of the fuzzy systems are very likely. This gives the designer several options for designing a control system.

The development process for a field-oriented control approach is based on the mathematical model of the physical behavior of the motor. Therefore, the design of the controller requires a complete mathematical understanding of an induction motor. For the fuzzy logic approach, this mathematical model is replaced by a verbal description of the motor behavior.

Future tests to be executed by Texas Instruments and Inform Software Corporation should confirm the expected higher tolerance of parameter oscillations of fuzzy logic controlled systems. The field-oriented

control method shows the strong influence of the rotor time constant [2]. Based on the calculations referenced in this document, the fuzzy logic controlled system appears to show more tolerant behavior.

Other DSP Fuzzy Logic Applications

Other applications of fuzzy logic on DSP hardware include:

- Sensor signal analysis
  - Intelligent processing of complex sensor signals by human expertise
  - Radar and sonar signal interpretation
  - Multisensor arrays
  - Acoustic quality control
  - Ultrasound image processing
- Servo motor control
  - Compensation for load changes in brushless motors
  - Switched reluctance (SR–) motors
  - Induction cage and other motors
- Speech recognition
  - Classification of sub-phonemes for recognition and ultra-dense speech compression
- Image identification and moving video compression
  - Differentiation among parts of pictures
  - Classification of objects
- Digital communication
  - Analysis of signals for cell switching
  - Voice mail systems
- Automotive applications
  - Road-adapting antilock braking systems
  - Electronic power steering systems
  - Intervehicle dynamics
  - Engine control
  - Suspension control
- Active noise cancellation
  - Realtime performance analysis of cancellation quality and adaptation of filters

References

1. G. Heinemann, “Comparison of Several Control Schemes for AC Induction Motors under Steady State and Dynamic Conditions”, EPE, Aachen, 1989.
2. D. Naunin, S. Beierke, P. Heidrich, “Transputers Control Asynchronous Servodrives, EPE, Florenz”, 1991.
3. F. Blaschke, “The Principle of Field Orientation as Applied to the New TRANSVECTOR Closed Loop Control System for Rotating Field Machines”, Siemens Rev., 1972, pg. 217.
4. W. Leonard, *Control of Electrical Devices*, Springer, Berlin, Heidelberg, 1985.
5. *TMS320C3x User’s Guide*, (literature number SPRU031D), Texas Instruments, 1994.
6. C. von Altrock, *Fuzzy Logic and NeuroFuzzy Applications Explained*, Prentice Hall, 1995.
7. C. von Altrock, “Toward Fuzzy Logic Standardization”, EUFIT Conference, Aachen, 1995.
8. dSpace, *Induction Motor Control with SIMULINK – Application Note*, dSPACE GmbH, 1994.
9. *DS1102 User’s Guide Version 2.0*, dSPACE GmbH.
10. *TRACE for MS Windows 3.1 User’s Guide, version 1.2*, dSPACE GmbH.
11. *COCKPIT for MS Windows 3.1 User’s Guide, version 1.0*, dSPACE GmbH.

Appendix A

This appendix provides the DSP assembly language code developed for the enhanced fuzzy logic controller. These source files were generated by the FuzzyTECH 4.0 MCU-320 edition code generation tool of Inform Software Corporation. These files are available from Texas Instruments TMS320 Bulletin Board System (BBS) at (713) 274–2323. Internet users can access the BBS via anonymous ftp at ti.com.

A complete Microsoft Windows-based simulation of a DSP-based fuzzy logic hard disk control system is contained in [6]. Refer to the References section for further reading on fuzzy logic applications and design methodology.

AC-MOT.DAT

This file includes the training data used to create the fuzzy logic magnetizing current estimator, using the neurofuzzy approach.

```
AC motor characteristic print
     OMEGA2           Iref
-------------------------------------------
AC–MOT#0000:       0        1
AC–MOT#0001:       5  1.00026
AC–MOT#0002:      10  1.00102
AC–MOT#0003:      15   1.0023
AC–MOT#0004:      20  1.00408
AC–MOT#0005:      25  1.00637
AC–MOT#0006:      30  1.00916
AC–MOT#0007:      35  1.01245
AC–MOT#0008:      40  1.01622
AC–MOT#0009:      45  1.02049
AC–MOT#0010:      50  1.02524
AC–MOT#0011:      55  1.03045
AC–MOT#0012:      60  1.03614
AC–MOT#0013:      65  1.04228
AC–MOT#0014:      70  1.04888
AC–MOT#0015:      75  1.05591
AC–MOT#0016:      80  1.06338
AC–MOT#0017:      85  1.07127
AC–MOT#0018:      90  1.07957
AC–MOT#0019:      95  1.08828
AC–MOT#0020:     100  1.09738
AC–MOT#0021:     105  1.10687
AC–MOT#0022:     110  1.11673
AC–MOT#0023:     115  1.12695
```

```
AC-MOT#0024:     120  1.13752
AC-MOT#0025:     125  1.14844
AC-MOT#0026:     130  1.15969
AC-MOT#0027:     135  1.17127
AC-MOT#0028:     140  1.18316
AC-MOT#0029:     145  1.19535
AC-MOT#0030:     150  1.20783
AC-MOT#0031:     155   1.2206
AC-MOT#0032:     160  1.23365
AC-MOT#0033:     165  1.24696
AC-MOT#0034:     170  1.26052
AC-MOT#0035:     175  1.27434
AC-MOT#0036:     180   1.2884
AC-MOT#0037:     185  1.30268
AC-MOT#0038:     190  1.31719
AC-MOT#0039:     195  1.33192
AC-MOT#0040:     200  1.34685
AC-MOT#0041:     205  1.36199
AC-MOT#0042:     210  1.37732
AC-MOT#0043:     215  1.39284
AC-MOT#0044:     220  1.40853
AC-MOT#0045:     225   1.4244
AC-MOT#0046:     230  1.44044
AC-MOT#0047:     235  1.45663
AC-MOT#0048:     240  1.47299
AC-MOT#0049:     245  1.48949
AC-MOT#0050:     250  1.50614
AC-MOT#0051:     255  1.52292
AC-MOT#0052:     260  1.53984
AC-MOT#0053:     265  1.55689
AC-MOT#0054:     270  1.57406
AC-MOT#0055:     275  1.59135
AC-MOT#0056:     280  1.60875
AC-MOT#0057:     285  1.62627
AC-MOT#0058:     290  1.64389
AC-MOT#0059:     295  1.66162
AC-MOT#0060:     300  1.67944
AC-MOT#0061:     305  1.69736
AC-MOT#0062:     310  1.71537
```

```
AC-MOT#0063:     315  1.73346
AC-MOT#0064:     320  1.75164
AC-MOT#0065:     325  1.76991
AC-MOT#0066:     330  1.78825
AC-MOT#0067:     335  1.80666
AC-MOT#0068:     340  1.82515
AC-MOT#0069:     345   1.8437
AC-MOT#0070:     350  1.86233
AC-MOT#0071:     355  1.88101
AC-MOT#0072:     360  1.89976
AC-MOT#0073:     365  1.91857
AC-MOT#0074:     370  1.93744
AC-MOT#0075:     375  1.95635
AC-MOT#0076:     380  1.97532
AC-MOT#0077:     385  1.99435
AC-MOT#0078:     390  2.01341
AC-MOT#0079:     395  2.03253
AC-MOT#0080:     400  2.05169
AC-MOT#0081:     405  2.07089
AC-MOT#0082:     410  2.09013
AC-MOT#0083:     415  2.10941
AC-MOT#0084:     420  2.12873
AC-MOT#0085:     425  2.14808
AC-MOT#0086:     430  2.16746
AC-MOT#0087:     435  2.18688
AC-MOT#0088:     440  2.20633
AC-MOT#0089:     445   2.2258
AC-MOT#0090:     450  2.24531
AC-MOT#0091:     455  2.26484
AC-MOT#0092:     460   2.2844
AC-MOT#0093:     465  2.30398
AC-MOT#0094:     470  2.32358
AC-MOT#0095:     475   2.3432
AC-MOT#0096:     480  2.36285
AC-MOT#0097:     485  2.38251
AC-MOT#0098:     490  2.40219
AC-MOT#0099:     495  2.42189
AC-MOT#0100:     500   2.4416
AC-MOT#0101:     505  2.46133
```

```
AC-MOT#0102:     510  2.48107
AC-MOT#0103:     515  2.50082
AC-MOT#0104:     520  2.52059
AC-MOT#0105:     525  2.54036
AC-MOT#0106:     530  2.56015
AC-MOT#0107:     535  2.57994
AC-MOT#0108:     540  2.59975
AC-MOT#0109:     545  2.61956
AC-MOT#0110:     550  2.63937
AC-MOT#0111:     555   2.6592
AC-MOT#0112:     560  2.67903
AC-MOT#0113:     565  2.69886
AC-MOT#0114:     570  2.71869
AC-MOT#0115:     575  2.73853
AC-MOT#0116:     580  2.75837
AC-MOT#0117:     585  2.77821
AC-MOT#0118:     590  2.79806
AC-MOT#0119:     595   2.8179
AC-MOT#0120:     600  2.83774
AC-MOT#0121:     605  2.85758
AC-MOT#0122:     610  2.87742
AC-MOT#0123:     615  2.89726
AC-MOT#0124:     620  2.91709
AC-MOT#0125:     625  2.93692
AC-MOT#0126:     630  2.95675
AC-MOT#0127:     635  2.97657
AC-MOT#0128:     640  2.99639
AC-MOT#0129:     645   3.0162
AC-MOT#0130:     650  3.03601
AC-MOT#0131:     655   3.0558
AC-MOT#0132:     660   3.0756
AC-MOT#0133:     665  3.09538
AC-MOT#0134:     670  3.11515
AC-MOT#0135:     675  3.13492
AC-MOT#0136:     680  3.15468
AC-MOT#0137:     685  3.17443
AC-MOT#0138:     690  3.19417
AC-MOT#0139:     695   3.2139
AC-MOT#0140:     700  3.23361
```

AC-MOT#0141: 705 3.25332
AC-MOT#0142: 710 3.27302
AC-MOT#0143: 715   3.2927
AC-MOT#0144: 720 3.31237
AC-MOT#0145: 725 3.33203
AC-MOT#0146: 730 3.35168
AC-MOT#0147: 735 3.37131
AC-MOT#0148: 740 3.39093
AC-MOT#0149: 745 3.41053
AC-MOT#0150: 750 3.43012
AC-MOT#0151: 755   3.4497
AC-MOT#0152: 760 3.46926
AC-MOT#0153: 765   3.4888
AC-MOT#0154: 770 3.50833
AC-MOT#0155: 775 3.52785
AC-MOT#0156: 780 3.54734
AC-MOT#0157: 785 3.56683
AC-MOT#0158: 790 3.58629
AC-MOT#0159: 795 3.60574
AC-MOT#0160: 800 3.62517
AC-MOT#0161: 805 3.64458
AC-MOT#0162: 810 3.66397
AC-MOT#0163: 815 3.68334
AC-MOT#0164: 820   3.7027
AC-MOT#0165: 825 3.72204
AC-MOT#0166: 830 3.74136
AC-MOT#0167: 835 3.76066
AC-MOT#0168: 840 3.77994
AC-MOT#0169: 845   3.7992
AC-MOT#0170: 850 3.81844
AC-MOT#0171: 855 3.83766
AC-MOT#0172: 860 3.85686
AC-MOT#0173: 865 3.87603
AC-MOT#0174: 870 3.89519
AC-MOT#0175: 875 3.91433
AC-MOT#0176: 880 3.93344
AC-MOT#0177: 885 3.95253
AC-MOT#0178: 890 3.97161
AC-MOT#0179: 895 3.99065

```
AC-MOT#0180:     900  4.00968
AC-MOT#0181:     905  4.02869
AC-MOT#0182:     910  4.04767
AC-MOT#0183:     915  4.06663
AC-MOT#0184:     920  4.08556
AC-MOT#0185:     925  4.10447
AC-MOT#0186:     930  4.12336
AC-MOT#0187:     935  4.14223
AC-MOT#0188:     940  4.16107
AC-MOT#0189:     945  4.17989
AC-MOT#0190:     950  4.19868
AC-MOT#0191:     955  4.21745
AC-MOT#0192:     960   4.2362
AC-MOT#0193:     965  4.25492
AC-MOT#0194:     970  4.27361
AC-MOT#0195:     975  4.29228
AC-MOT#0196:     980  4.31093
AC-MOT#0197:     985  4.32955
AC-MOT#0198:     990  4.34814
AC-MOT#0199:     995  4.36671
AC-MOT#0200:    1000  4.38526
```

FUZ_CTRL.CFG

This file provides the configuration data for the FUZ_CTRL project.

```
[DIRECTORIES]
PROJECT=c:\dsp_cit\fuzzy400
COMPILETO=c:\dsp_cit\fuzzy400
SIMULATION=c:\dsp_cit\fuzzy400
COMMAND=c:\dsp_cit\fuzzy400
[WINDOWPOSITION]
FT           1     1     1     1
PROJECT     -1    -1    -1    -1         0         0
LV           -1    -1    -1    -1
LVEDIT      276    -2   507   276  0  Iref
LVEDIT      276   274   508   219  0  OMEGA2
RB           13    -3   266   263  SS  RB1
```

FUZ_CTRL.FTL

This file includes the complete design software for the magnetizing current estimator. You can load this file directly from the FuzzyTECH 4.0 MCU-320 edition software tool. The corresponding C code is named FUZ_CTRL.C and its header file FUZ_CTRL.H (see Appendix B, FUZ_CTRL.C and FUZ_CTRL.H).

```
PROJECT {
  NAME = FUZ_CTRL.FTL;
  DATEFORMAT = DD.MM.YYYY;
  LASTCHANGE = 07.12.1995;
  CREATED    = 07.06.1995;
  SHELL = MCU_320;
  SHELLOPTIONS {
    ONLINE_REFRESHTIME = 55;
    ONLINE_TIMEOUTCOUNT = 0;
    ONLINE_CODE = OFF;
    COMMENTS    = ON;
    TRACE_BUFFER = (OFF, PAR(100));
    PUBLIC_IO = OFF;
    FAST_CMBF = ON;
    FAST_COA  = OFF;
    FILE_CODE = OFF;
    BTYPE = 16_BIT;
    C_TYPE = ANSI;
    WINDLL = OFF;
  } /* SHELLOPTIONS */
  MODEL {
    VARIABLE_SECTION {
      LVAR {
        NAME    = Iref;
        BASEVAR = Units;
        LVRANGE = MIN(0.000000), MAX(5.000000),
                  MINDEF(0), MAXDEF(65535),
                  DEFAULT_OUTPUT(2.500000);
        RESOLUTION = XGRID(0.000100), YGRID(1.000000),
                     SHOWGRID (ON), SNAPTOGRID(ON);
        COLOR = RED (0), GREEN (255), BLUE (0);
        TERM {
          TERMNAME = very_low;
          POINTS = (0.000000, 0.000000),
                   (0.967600, 1.000000),
                   (1.097300, 0.000000),
                   (5.000000, 0.000000);
          SHAPE = LINEAR;
          COLOR = RED (255), GREEN (0), BLUE (0);
```

```
}
TERM {
  TERMNAME = low;
  POINTS = (0.000000, 0.000000),
           (0.967600, 0.000000),
           (1.097300, 1.000000),
           (1.572700, 0.000000),
           (5.000000, 0.000000);
  SHAPE = LINEAR;
  COLOR = RED (0), GREEN (255), BLUE (0);
}
TERM {
  TERMNAME = medium_low;
  POINTS = (0.000000, 0.000000),
           (1.097300, 0.000000),
           (1.572700, 1.000000),
           (1.822300, 0.000000),
           (5.000000, 0.000000);
  SHAPE = LINEAR;
  COLOR = RED (0), GREEN (0), BLUE (255);
}
TERM {
  TERMNAME = medium;
  POINTS = (0.000000, 0.000000),
           (1.572700, 0.000000),
           (1.822300, 1.000000),
           (2.466300, 0.000000),
           (5.000000, 0.000000);
  SHAPE = LINEAR;
  COLOR = RED (128), GREEN (0), BLUE (0);
}
TERM {
  TERMNAME = medium_high;
  POINTS = (0.000000, 0.000000),
           (1.822300, 0.000000),
           (2.466300, 1.000000),
           (3.285600, 0.000000),
           (5.000000, 0.000000);
  SHAPE = LINEAR;
```

```
    COLOR = RED (0), GREEN (128), BLUE (0);
  }
  TERM {
    TERMNAME = high;
    POINTS = (0.000000, 0.000000),
             (2.466300, 0.000000),
             (3.285600, 1.000000),
             (4.401500, 0.000000),
             (5.000000, 0.000000);
    SHAPE = LINEAR;
    COLOR = RED (0), GREEN (0), BLUE (128);
  }
  TERM {
    TERMNAME = very_high;
    POINTS = (0.000000, 0.000000),
             (3.285600, 0.000000),
             (4.401500, 1.000000),
             (5.000000, 0.000000);
    SHAPE = LINEAR;
    COLOR = RED (255), GREEN (0), BLUE (128);
  }
}  /* LVAR */
LVAR {
  NAME    = OMEGA2;
  BASEVAR = Units;
  LVRANGE = MIN(0.000000), MAX(1000.000000),
            MINDEF(0), MAXDEF(65535),
            DEFAULT_OUTPUT(500.000000);
  RESOLUTION = XGRID(0.020000), YGRID(1.000000),
               SHOWGRID (ON), SNAPTOGRID(ON);
  COLOR = RED (255), GREEN (0), BLUE (0);
  TERM {
    TERMNAME = very_low;
    POINTS = (0.000000, 1.000000),
             (1.640000, 1.000000),
             (114.700000, 0.000000),
             (1000.000000, 0.000000) : OPEN (0.000000, 1000.000000);
    SHAPE = LINEAR;
    COLOR = RED (255), GREEN (0), BLUE (0);
```

```
}
TERM {
  TERMNAME = low;
  POINTS = (0.000000, 0.000000),
           (1.640000, 0.000000),
           (114.700000, 1.000000),
           (266.740000, 0.000000),
           (1000.000000, 0.000000) : OPEN (0.000000, 1000.000000);
  SHAPE = LINEAR;
  COLOR = RED (0), GREEN (255), BLUE (0);
}
TERM {
  TERMNAME = medium_low;
  POINTS = (0.000000, 0.000000),
           (114.700000, 0.000000),
           (266.740000, 1.000000),
           (332.680000, 0.000000),
           (1000.000000, 0.000000) : OPEN (0.000000, 1000.000000);
  SHAPE = LINEAR;
  COLOR = RED (0), GREEN (0), BLUE (255);
}
TERM {
  TERMNAME = medium;
  POINTS = (0.000000, 0.000000),
           (266.740000, 0.000000),
           (332.680000, 1.000000),
           (513.620000, 0.000000),
           (1000.000000, 0.000000) : OPEN (0.000000, 1000.000000);
  SHAPE = LINEAR;
  COLOR = RED (128), GREEN (0), BLUE (0);
}
TERM {
  TERMNAME = medium_high;
  POINTS = (0.000000, 0.000000),
           (332.680000, 0.000000),
           (513.620000, 1.000000),
           (720.420000, 0.000000),
           (1000.000000, 0.000000) : OPEN (0.000000, 1000.000000);
  SHAPE = LINEAR;
```

```
      COLOR = RED (0), GREEN (128), BLUE (0);
    }
    TERM {
      TERMNAME = high;
      POINTS = (0.000000, 0.000000),
               (513.620000, 0.000000),
               (720.420000, 1.000000),
               (998.760000, 0.000000),
               (1000.000000, 0.000000) : OPEN (0.000000, 1000.000000);
      SHAPE = LINEAR;
      COLOR = RED (0), GREEN (0), BLUE (128);
    }
    TERM {
      TERMNAME = very_high;
      POINTS = (0.000000, 0.000000),
               (720.420000, 0.000000),
               (998.760000, 1.000000),
               (1000.000000, 1.000000) : OPEN (0.000000, 1000.000000);
      SHAPE = LINEAR;
      COLOR = RED (255), GREEN (0), BLUE (128);
    }
  }  /* LVAR */
}  /* VARIABLE_SECTION */
OBJECT_SECTION {
  INTERFACE {
    INPUT = (OMEGA2, FCMBF);
    POS = –191, –39;
    RANGECHECK = ON;
  }
  INTERFACE {
    OUTPUT = (Iref, COM);
    POS = 196, –39;
    RANGECHECK = ON;
  }
  RULEBLOCK {
    NAME = RB1;
    INPUT = OMEGA2;
    OUTPUT = Iref;
    AGGREGATION = (MIN_MAX, PAR (0.000000));
```

```
        COMPOSITION = (GAMMA, PAR (0.000000));
        RESULT_AGGR = MAX;
        POS = 0, –73;
        RULES {
          IF    OMEGA2 = very_low
          THEN  Iref = very_low   WITH 1.000;
          IF    OMEGA2 = low
          THEN  Iref = low   WITH 1.000;
          IF    OMEGA2 = medium_low
          THEN  Iref = medium_low   WITH 1.000;
          IF    OMEGA2 = medium
          THEN  Iref = medium   WITH 1.000;
          IF    OMEGA2 = medium_high
          THEN  Iref = medium_high   WITH 1.000;
          IF    OMEGA2 = high
          THEN  Iref = high   WITH 1.000;
          IF    OMEGA2 = very_high
          THEN  Iref = very_high   WITH 1.000;
        }  /* RULES */
      }
    }  /* OBJECT_SECTION */
  }  /* MODEL */
}  /* PROJECT */
TERMINAL {
    BAUDRATE     = 19200;
    STOPBITS     = 1;
    PROTOCOL     = NO;
    CONNECTION   = NOPORT;
    INPUTBUFFER  = 1024;
    OUTPUTBUFFER = 1024;
}  /* TERMINAL */
NEUROFUZZY {
  LEARNRULE     =RandomMethod;
  STEPWIDTHDOS  = 0.101563;
  STEPWIDTHTERM = 0.100000;
  MAXDEVIATION  = (0.590490, 0.100000, 0.900000);
  AVGDEVIATION  = 0.100000;
  MAXSTEPS      = 100;
  NEURONS       = 1;
```

```
  DATASEQUENCE  = RANDOM;
  UPDATEDBGWIN  = OFF;
}  /* NEUROFUZZY */
```

SPEED.CFG

This file provides the configuration data for the nonlinear speed controller project.

```
[DIRECTORIES]
PROJECT=c:\dsp_cit\fuzzy400
COMPILETO=c:\dsp_cit\fuzzy400
SIMULATION=c:\dsp_cit\fuzzy400
COMMAND=c:\dsp_cit\fuzzy400
[WINDOWPOSITION]
FT           1     1     1     1
PROJECT     -1    -1    -1    -1         0         0
LV          -1    -1    -1    -1
LVEDIT       1     0   400   231  1  d_omega
LVEDIT     398   318   385   192  0  d_slip
LVEDIT       0   231   400   261  1  dd_omega
RB         401   -15   400   334  SS  RB1
```

SPEED.FTL

This file includes the complete design software for the nonlinear speed controller. You can load this file directly using the FuzzyTECH 4.0 MCU-320 edition software tool. The corresponding C code is named SLIP_FU.C and its header file SLIP_FU.H (see Appendix B, SLIP_FU.C and SLIP_FU.H).

```
PROJECT {
  NAME = SPEED.FTL;
  DATEFORMAT = DD.MM.YYYY;
  LASTCHANGE = 07.12.1995;
  CREATED    = 06.06.1995;
  SHELL = MCU_320;
  SHELLOPTIONS {
    ONLINE_REFRESHTIME = 55;
    ONLINE_TIMEOUTCOUNT = 1100;
    ONLINE_CODE = OFF;
    COMMENTS    = ON;
    TRACE_BUFFER = (ON, PAR(100));
    PUBLIC_IO = ON;
    FAST_CMBF = ON;
```

```
  FAST_COA  = OFF;
  FILE_CODE = OFF;
  BTYPE = 16_BIT;
  C_TYPE = ANSI;
  WINDLL = OFF;
} /* SHELLOPTIONS */
MODEL {
  VARIABLE_SECTION {
    LVAR {
      NAME    = d_omega;
      BASEVAR = rpm;
      LVRANGE = MIN(–500.000000), MAX(500.000000),
                MINDEF(0), MAXDEF(50000),
                DEFAULT_OUTPUT(0.000000);
      RESOLUTION = XGRID(0.020000), YGRID(1.000000),
                   SHOWGRID (ON), SNAPTOGRID(ON);
      COLOR = RED (255), GREEN (0), BLUE (0);
      TERM {
        TERMNAME = negative;
        POINTS = (–500.000000, 1.000000),
                 (–237.860000, 0.000000),
                 (500.000000, 0.000000);
        SHAPE = LINEAR;
        COLOR = RED (255), GREEN (0), BLUE (0);
      }
      TERM {
        TERMNAME = negative_low;
        POINTS = (–500.000000, 0.000000),
                 (–237.860000, 1.000000),
                 (0.440000, 0.000000),
                 (500.000000, 0.000000);
        SHAPE = LINEAR;
        COLOR = RED (128), GREEN (0), BLUE (0);
      }
      TERM {
        TERMNAME = zero_neg;
        POINTS = (–500.000000, 0.000000),
                 (–83.820000, 0.000000),
                 (–41.540000, 1.000000),
```

```
            (–0.580000, 0.000000),
            (500.000000, 0.000000);
  SHAPE = LINEAR;
  COLOR = RED (0), GREEN (0), BLUE (128);
}
TERM {
  TERMNAME = zero;
  POINTS = (–500.000000, 0.000000),
            (–236.420000, 0.000000),
            (1.880000, 1.000000),
            (3.200000, 1.000000),
            (221.660000, 0.000000),
            (500.000000, 0.000000);
  SHAPE = LINEAR;
  COLOR = RED (0), GREEN (0), BLUE (255);
}
TERM {
  TERMNAME = zero_pos;
  POINTS = (–500.000000, 0.000000),
            (0.000000, 0.000000),
            (24.280000, 1.000000),
            (24.340000, 1.000000),
            (80.120000, 0.000000),
            (500.000000, 0.000000);
  SHAPE = LINEAR;
  COLOR = RED (255), GREEN (0), BLUE (128);
}
TERM {
  TERMNAME = positive_low;
  POINTS = (–500.000000, 0.000000),
            (3.200000, 0.000000),
            (294.460000, 1.000000),
            (496.100000, 0.000000),
            (500.000000, 0.000000);
  SHAPE = LINEAR;
  COLOR = RED (0), GREEN (128), BLUE (0);
}
TERM {
  TERMNAME = positive;
```

```
    POINTS = (–500.000000, 0.000000),
             (222.040000, 0.000000),
             (484.180000, 1.000000),
             (500.000000, 1.000000);
    SHAPE = LINEAR;
    COLOR = RED (0), GREEN (255), BLUE (0);
  }
}  /* LVAR */
LVAR {
  NAME    = d_slip;
  BASEVAR = rpm;
  LVRANGE = MIN(–500.000000), MAX(500.000000),
            MINDEF(0), MAXDEF(50000),
            DEFAULT_OUTPUT(0.000000);
  RESOLUTION = XGRID(0.020000), YGRID(1.000000),
               SHOWGRID (ON), SNAPTOGRID(ON);
  COLOR = RED (0), GREEN (0), BLUE (255);
  TERM {
    TERMNAME = decrease;
    POINTS = (–500.000000, 0.000000),
             (–395.140000, 1.000000),
             (–394.500000, 1.000000),
             (–248.860000, 0.000000),
             (500.000000, 0.000000);
    SHAPE = LINEAR;
    COLOR = RED (255), GREEN (0), BLUE (0);
  }
  TERM {
    TERMNAME = decrease_low;
    POINTS = (–500.000000, 0.000000),
             (–63.100000, 1.000000),
             (–35.140000, 1.000000),
             (11.680000, 0.000000),
             (500.000000, 0.000000);
    SHAPE = LINEAR;
    COLOR = RED (0), GREEN (128), BLUE (0);
  }
  TERM {
    TERMNAME = steady;
```

```
    POINTS = (–500.000000, 0.000000),
             (–7.980000, 0.000000),
             (0.020000, 1.000000),
             (100.840000, 0.000000),
             (500.000000, 0.000000);
    SHAPE = LINEAR;
    COLOR = RED (0), GREEN (255), BLUE (0);
  }
  TERM {
    TERMNAME = increase_low;
    POINTS = (–500.000000, 0.000000),
             (–6.820000, 0.000000),
             (50.160000, 1.000000),
             (424.640000, 0.000000),
             (500.000000, 0.000000);
    SHAPE = LINEAR;
    COLOR = RED (128), GREEN (0), BLUE (0);
  }
  TERM {
    TERMNAME = increase;
    POINTS = (–500.000000, 0.000000),
             (250.180000, 0.000000),
             (395.820000, 1.000000),
             (396.160000, 1.000000),
             (496.980000, 0.000000),
             (500.000000, 0.000000);
    SHAPE = LINEAR;
    COLOR = RED (0), GREEN (0), BLUE (255);
  }
}  /* LVAR */
LVAR {
  NAME    = dd_omega;
  BASEVAR = rpm;
  LVRANGE = MIN(–50.000000), MAX(50.000000),
            MINDEF(0), MAXDEF(50000),
            DEFAULT_OUTPUT(0.000000);
  RESOLUTION = XGRID(0.002000), YGRID(1.000000),
               SHOWGRID (ON), SNAPTOGRID(ON);
  COLOR = RED (0), GREEN (255), BLUE (0);
```

```
    TERM {
      TERMNAME = negative;
      POINTS = (–50.000000, 1.000000),
               (–6.310000, 0.000000),
               (50.000000, 0.000000);
      SHAPE = LINEAR;
      COLOR = RED (255), GREEN (0), BLUE (0);
    }
    TERM {
      TERMNAME = zero;
      POINTS = (–50.000000, 0.000000),
               (–21.100000, 0.000000),
               (–0.936000, 1.000000),
               (–0.160000, 1.000000),
               (21.686000, 0.000000),
               (50.000000, 0.000000);
      SHAPE = LINEAR;
      COLOR = RED (0), GREEN (0), BLUE (255);
    }
    TERM {
      TERMNAME = positive;
      POINTS = (–50.000000, 0.000000),
               (4.792000, 0.000000),
               (48.482000, 1.000000),
               (50.000000, 1.000000);
      SHAPE = LINEAR;
      COLOR = RED (0), GREEN (255), BLUE (0);
    }
  }  /* LVAR */
}  /* VARIABLE_SECTION */
OBJECT_SECTION {
  INTERFACE {
    INPUT = (d_omega, FCMBF);
    POS = –218, –156;
    RANGECHECK = ON;
  }
  INTERFACE {
    INPUT = (dd_omega, FCMBF);
    POS = –218, –8;
```

```
  RANGECHECK = ON;
}
INTERFACE {
  OUTPUT = (d_slip, COM);
  POS = 76, –90;
  RANGECHECK = ON;
}
RULEBLOCK {
  NAME = RB1;
  INPUT = d_omega, dd_omega;
  OUTPUT = d_slip;
  AGGREGATION = (MIN_MAX, PAR (0.000000));
  COMPOSITION = (GAMMA, PAR (0.000000));
  RESULT_AGGR = MAX;
  POS = –70, –122;
  RULES {
    IF    d_omega = negative
    THEN  d_slip = decrease   WITH 1.000;
    IF    d_omega = negative_low
      AND dd_omega = negative
    THEN  d_slip = decrease   WITH 1.000;
    IF    d_omega = zero
      AND dd_omega = negative
    THEN  d_slip = decrease_low   WITH 1.000;
    IF    d_omega = zero
      AND dd_omega = zero
    THEN  d_slip = steady   WITH 1.000;
    IF    d_omega = zero
      AND dd_omega = positive
    THEN  d_slip = increase_low   WITH 1.000;
    IF    d_omega = positive_low
      AND dd_omega = positive
    THEN  d_slip = increase   WITH 1.000;
    IF    d_omega = positive
    THEN  d_slip = increase   WITH 1.000;
    IF    d_omega = negative_low
      AND dd_omega = zero
    THEN  d_slip = decrease_low   WITH 1.000;
    IF    d_omega = positive_low
```

```
            AND dd_omega = zero
          THEN  d_slip = increase_low   WITH 1.000;
          IF    d_omega = zero_neg
          THEN  d_slip = decrease_low   WITH 1.000;
          IF    d_omega = zero_pos
          THEN  d_slip = increase_low   WITH 1.000;
        }  /* RULES */
      }
    }  /* OBJECT_SECTION */
  }  /* MODEL */
}  /* PROJECT */
TERMINAL {
    BAUDRATE     = 19200;
    STOPBITS     = 1;
    PROTOCOL     = NO;
    CONNECTION   = NOPORT;
    INPUTBUFFER  = 1024;
    OUTPUTBUFFER = 1024;
}  /* TERMINAL */
NEUROFUZZY {
  LEARNRULE     =RandomMethod;
  STEPWIDTHDOS  = 0.101563;
  STEPWIDTHTERM = 1.000000;
  MAXDEVIATION  = (50.000000, 1.000000, 0.750000);
  AVGDEVIATION  = 0.100000;
  MAXSTEPS      = 100;
  NEURONS       = 1;
  DATASEQUENCE  = SEQUENTIAL;
  UPDATEDBGWIN  = OFF;
}  /* NEUROFUZZY */
```

Appendix B

This appendix provides the real time source software and setup files for the dSPACE environment software COCKPIT31 and TRACE31. These files are available from Texas Instruments TMS320 Bulletin Board System (BBS) at (713) 274–2323. Internet users can access the BBS via anonymous ftp at ti.com.

FUZ_CTRL.C

This file provides the C source function, which is generated by the FUZ_CTRL project of the FuzzyTECH 4.0 MCU-320 edition software.

```
/*---------------- ------------------------------------------------------*/
/*----------- fuzzyTECH 4.0 MCU-320 Edition --- C-Precompiler -------------*/
/*(c) 1995 by Inform Software Corp., 2001 Midwest Rd., Oak Brook, IL 60251 */
/*-------- (c) 1995 by INFORM GmbH, Pascalstr. 23, D-52076 Aachen ---------*/
/*-------------------------------------------------------------------------*/
/*------------ code generation date: Wed Jun 07 18:43:01 1995 -------------*/
/*-------------------------- project: FUZ_CTRL ----------------------------*/
/*-------------------------------------------------------------------------*/

#define FTLIBC16
#include "ftlibc.h"
#define FUZZYDEFINED
#define FLAGSDEFINED
#include "fuz_ctrl.h"

static FUZZY crispio[1+1];

static FUZZY fuzvals[7+7+0];

static const FUZZY tpts[28] = {
   0x0000, 0x0000, 0x006C, 0x0012,
   0x006C, 0x0012, 0x1D5C, 0x000D,
   0x1D5C, 0x000D, 0x4449, 0x001E,
   0x4449, 0x001E, 0x552A, 0x000B,
   0x552A, 0x000B, 0x837D, 0x000A,
   0x837D, 0x000A, 0xB86D, 0x0007,
   0xB86D, 0x0007, 0xFFFF, 0x0000};

static const FUZZY xcom[7] = {
   0x318A, 0x382E, 0x5086, 0x5D4E, 0x7E45, 0xA838, 0xE15A};

static const BYTE rt0[23] = {
   7, 0,
   0, 1,   7,
   0, 1,   8,
   0, 1,   9,
   0, 1,  10,
   0, 1,  11,
   0, 1,  12,
   0, 1,  13};

static const FRAT frat0[8] = {
   0x2, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3, 0x3};

FLAGS fuz_ctrl(FUZZY lv0_OMEGA2, FUZZY *lv1_Iref) {
crispio[0] = lv0_OMEGA2;

fuzptr = (FUZZY *) fuzvals;
tpptr  = (FUZZY *) tpts;

crisp = crispio[0];
itcnt = 7;
flmss();
```

```
rtptr   = (BYTE *) rt0;
pfuzvals = (FUZZY *) fuzvals;
fuzptr   = (FUZZY *) &fuzvals[0];
fratptr  = (FRAT *)  frat0;
noit     = 7;
Min(); /* min aggregation */

invalidflags = 0;
fuzptr       = &fuzvals[7];
xcomptr      = (FUZZY *) xcom;

crispio[1] = 0x8000;
otcnt  = 7;
defuzz = &crispio[1];
com();

*lv1_Iref = crispio[1];

return invalidflags;
}

void initfuz_ctrl(void) {
for (fuzptr = &fuzvals[7];
     fuzptr < &fuzvals[14];
     *fuzptr++ = 0);
}

/*
data size knowledge base (bytes):
RAM:        32   00020H
ROM:       109   0006DH
TOTAL:     141   0008DH
*/
```

FUZ_CTRL.H

This file provides the header data for the FUZ_CTRL.C function, which is generated by the FUZ_CTRL project of the FuzzyTECH 4.0 MCU-320 edition software.

```
/*-------------------------------------------------------------------------*/
/*----------- fuzzyTECH 4.0 MCU-320 Edition --- C-Precompiler -------------*/
/*(c) 1995 by Inform Software Corp., 2001 Midwest Rd., Oak Brook, IL 60251 */
/*-------- (c) 1995 by INFORM GmbH, Pascalstr. 23, D-52076 Aachen ---------*/
/*-------------------------------------------------------------------------*/
/*------------ code generation date: Wed Jun 07 18:43:01 1995 -------------*/
/*--------------------------- project: FUZ_CTRL ---------------------------*/
/*-------------------------------------------------------------------------*/


/*-------------------------------------------------------------------------*/
/*----------------- export interface of project FUZ_CTRL ------------------*/
/*-------------------------------------------------------------------------*/

/*-------------------------------------------------------------------------*/
/*-------------------------------- typedefs -------------------------------*/
/*-------------------------------------------------------------------------*/
#ifndef FUZZYDEFINED
/*------- datatype for all computations in the fuzzy logic system ---------*/
typedef unsigned short FUZZY;
#define FUZZYDEFINED
#endif

#ifndef FLAGSDEFINED
/*--------------- datatype of return value of fuzzy controller
```

```
--------------*/
typedef unsigned int FLAGS;
#define FLAGSDEFINED
#endif
/*---------------------------------------------------------------------------*/
/*------------------------- function prototypes -----------------------------*/
/*---------------------------------------------------------------------------*/
/*----- for starting up the generated fuzzy logic system, call once -------*/
void initfuz_ctrl(void);
/*------------- for calling the generated fuzzy logic system --------------*/
FLAGS fuz_ctrl(FUZZY lv0_OMEGA2, FUZZY *lv1_Iref);
```

SLIP_FU.C

This file provides the C source function, which is generated by the SPEED project of the FuzzyTECH 4.0 MCU-320 edition software.

```
/*---------------------------------------------------------------------------*/
/*-- -------- fuzzyTECH 4.0 MCU-320 Edition --- C-Precompiler -------------*/
/*-(c) 1995 by Inform Software Corp., 2001 Midwest Rd., Oak Brook, IL 60251*/
/*-------- (c) 1995 by INFORM GmbH, Pascalstr. 23, D-52076 Aachen ---------*/
/*---------------------------------------------------------------------------*/
/*------------ code generation date: Thu Jun 08 15:11:30 1995 -------------*/
/*----------------------------- project: MOTOR ------------------------------*/
/*---------------------------------------------------------------------------*/

#define FTLIBC16
#include "ftlibc.h"
#define FUZZYDEFINED
#define FLAGSDEFINED
#include "motor.h"
static FUZZY crispio[2+1];
static FUZZY fuzvals[10+5+0];
FUZZY * const pcvmotor = crispio;
static const FUZZY tpts[40] = {
   0x0000, 0x0000, 0x0000, 0x000A,
   0x0000, 0x000A, 0x3333, 0x000B,
   0x5149, 0x003E, 0x598B, 0x0040,
   0x337B, 0x000B, 0x6248, 0x000D,
   0x61A8, 0x006C, 0x6669, 0x002F,
   0x6248, 0x0009, 0x9B2B, 0x000D,
   0x978B, 0x000C, 0xFFFF, 0x0000,
   0x0000, 0x0000, 0x0000, 0x0006,
```

```
   0x3872, 0x000D, 0x6158, 0x000C,
   0x6B04, 0x0006, 0xFFFF, 0x0000};
static const FUZZY xcom[5] = {
   0x148B, 0x5810, 0x61A9, 0x6B74, 0xAEFF};

static const BYTE rt0[42] = {
   11, 0,
   0, 1,  10,
   1, 1,   7,  10,
   1, 1,   8,  11,
   0, 1,  11,
   1, 1,   7,  11,
   1, 1,   8,  12,
   1, 1,   9,  13,
   0, 1,  13,
   1, 1,   9,  14,
   1, 1,   8,  13,
   0, 1,  14};
static const FRAT frat0[8] = {
   0x2, 0x3, 0x8, 0x3, 0xC, 0x3, 0x8, 0x3};
FLAGS motor(void) {
fuzptr = (FUZZY *) fuzvals;
tpptr  = (FUZZY *) tpts;
crisp = crispio[0];
itcnt = 7;
flmss();
crisp = crispio[1];
itcnt = 3;
flmss();
rtptr   = (BYTE *) rt0;
pfuzvals = (FUZZY *) fuzvals;
fuzptr   = (FUZZY *) &fuzvals[0];
fratptr  = (FRAT *)  frat0;
noit     = 7;
Min(); /* min aggregation */
invalidflags = 0;
fuzptr       = &fuzvals[10];
xcomptr      = (FUZZY *) xcom;
crispio[2] = 0x61A8;
```

```
otcnt  = 5;
defuzz = &crispio[2];
com();
return invalidflags;
}
void initmotor(void) {
for (fuzptr = &fuzvals[10];
     fuzptr < &fuzvals[15];
     *fuzptr++ = 0);
}
/*
data size knowledge base (bytes):
RAM:        36   00024H
ROM:       150   00096H
TOTAL:     186   000BAH
*/
```

SLIP_FU.H

This file provides the header data for the SLIP_FU.C function, which is generated by the SPEED project of the FuzzyTECH 4.0 MCU-320 edition software.

```
/*-------------------------------------------------------------------------*/
/*----------- fuzzyTECH 4.0 MCU-320 Edition --- C-Precompiler -------------*/
/*(c) 1995 by Inform Software Corp., 2001 Midwest Rd., Oak Brook, IL 60251 */
/*-------- (c) 1995 by INFORM GmbH, Pascalstr. 23, D-52076 Aachen ---------*/
/*-------------------------------------------------------------------------*/
/*------------ code generation date: Thu Jun 08 15:11:30 1995 -------------*/
/*----------------------------- project: MOTOR ----------------------------*/
/*-------------------------------------------------------------------------*/


/*-------------------------------------------------------------------------*/
/*------------------ export interface of project MOTOR --------------------*/
/*-------------------------------------------------------------------------*/

/*-------------------------------------------------------------------------*/
/*------------------------------- typedefs --------------------------------*/
/*-------------------------------------------------------------------------*/
#ifndef FUZZYDEFINED
/*------- datatype for all computations in the fuzzy logic system ---------*/
typedef unsigned short FUZZY;
#define FUZZYDEFINED
#endif

#ifndef FLAGSDEFINED
/*------------- datatype of return value of fuzzy controller --------------*/
typedef unsigned int FLAGS;
#define FLAGSDEFINED
#endif

/*---------------------- data only used by fuzzyTECH ----------------------*/
extern FUZZY * const pcvmotor;
```

```
/*-------------------------------------------------------------------------*/
/* use the following #defines to write the inputs to the fuzzy controller -*/
/*-------------------------------------------------------------------------*/
#define d_omega_motor                 (*(pcvmotor+  0)) /* 0000H .. C350H */
#define dd_omega_motor                (*(pcvmotor+  1)) /* 0000H .. C350H */

/*-------------------------------------------------------------------------*/
/*use the following #defines to read the outputs from the fuzzy controller */
/*-------------------------------------------------------------------------*/
#define d_slip_motor                  (*(pcvmotor+  2)) /* 0000H .. C350H */

/*-------------------------------------------------------------------------*/
/*------------------------- function prototypes ---------------------------*/
/*-------------------------------------------------------------------------*/

/*----- for starting up the generated fuzzy logic system, call once -------*/
void initmotor(void);

/*------------- for calling the generated fuzzy logic system --------------*/
FLAGS motor(void);
```

IM_CTRL.C

This file provides the complete C source software including both fuzzy logic blocks SLIP_FU.C and FUZ_CTRL.C, the field-oriented control method, the general functions for PI controllers and FIR filters (GEN_MOD.C), and the complete real time environment.

```
/* IM_CTRL.C ************************************************************
*                                                                         *
* Fuzzy Control and Field Oriented Control of Induction Motor             *
*                                                                         *
*                                                                         *
* author : F. Schuette , R. Koenigbauer , S. Beierke                      *
* date   : Feb 20, 1995                                                   *
* dSPACE & UNI-GH PADERBORN/LEA $ Inform & TI                             *
*                                                                         *
*************************************************************************/
#include  "brtenv.h"          /* basic real-time environment */
#include  "ctrl.h"            /* general header of control structure */
#include  "siggen.c"
#include  <math.h>
#include  "slip_fu.h"         /* Enhanced Fuzzy Controlled System */
float     x,y;                /* auxiliar variables */
#define fuzzy 1
#define FUZZY_SPEED_CONTROL 0
/* error flag for CHKERR31 at last dual-port memory location */
int *error = (int *) (DP_MEM_BASE + DP_MEM_SIZE -1);
main()
```

```
{
  /* init DAC mode, calibrate ADCs */
  init();
  /* download standard communication and pwm–code to P14 */
  load_slave ();
  /* declare input/output pins */
  ds1102_p14_pin_io_init (LOCK_PWM);
  /* lock pwm inverter */
  ds1102_p14_pin_io_set (LOCK_PWM);
  /* clear incremental encoder */
  ds1102_inc_clear_counter(INC1);
  /* initialize control parameters */
  varinit();
  /* initialize signal generator */
  init_sgen(p_freq, p_duty, (T_S * SPEED_CTRL_R));
  /* initialize overload error flag to avoid termination from chkerr31 */
  *error = NO_ERROR;
  /* poll power–on switch */
  p14_ioport = NOT_READY;
  while (p14_ioport == NOT_READY)
    p14_ioport = ds1102_p14_pin_io_in () & NOT_READY;
  /* unlock pwm inverter */
  ds1102_p14_pin_io_clear (LOCK_PWM);
  /* compute offset for minimal motor current (rpm = 0) */
  {
    unsigned int tmp = 0;
    fuz_ctrl( tmp, &current_offset ); /* both uint */
  }
  test_db = (float) current_offset;
  test_db = 1/test_db ;
  start_isr_t0(T_S);          /* start interrupt service routine timer0 */
  while (*error == NO_ERROR)  /* background process */
  {
    /* start speed control */
    if ( flag == SET )
    {
      flag = !SET;

      /* signal generator */
```

```
   if (sig_flag == 1)
    speed_ref = sgen(p_freq,p_ampl,p_offs,p_duty,p_wave,T_S*SPEED_CTRL_R);
   /* speed controller */
   /* fol: first order lag  */
   fol(speed,speed_fol,fol_speed);
   if( !flag_pi_fuzzy )
   {
/* fol only used for step functions */
if (sig_flag == 1 && (p_wave == 1 || p_wave == 4))
{
 pictrl(speed_ref,speed_fol,i_ref_dq.q,pi_ctrl_speed);
}
else
{
 fol(speed_ref,speed_ref_fol,fol_speed_ref);
 pictrl(speed_ref_fol,speed_fol,i_ref_dq.q,pi_ctrl_speed);
}
/* flux controller */
pictrl(psi_rd_ref,psi_rd,i_ref_dq.d,pi_ctrl_flux);
   } /* end only for field oriented */
   if( flag_pi_fuzzy ) {
/*******************************************/
/* INPUTS  : speed_ref = OMEGA-ref / OMEGA-0 */
/*           speed     = OMEGA     / OMEGA-0 */
/*                                           */
/*           pi_ctrl_res_speed               */
/*******************************************/
/* 1st path: 'calculation of the slip speed using a PI Controller' *
 * for not enhanced Fuzzy control system                          */
   #if !FUZZY_SPEED_CONTROL
/* slip_Speed = OMEGA2-ref / OMEGA-0 */
/* inputs( speed_ref, speed ) in rpm (mechanic) */
pictrl( speed_ref, speed_fol, slip_Speed, pi_ctrl_res_speed); /* rpm */
/* output( slip_Speed ) in rpm (mechanic) */
/* bounds set to 500 rpm */
   #endif
   } /* end only for fuzzy */
 } /* every 5 T_S cycles */
 service_cockpit();
```

```
  }
   /* Reset PWM inverter */
  ds1102_p14_phase_voltages
    ( PVV_ADDRESS,
      (long) 0,
      (long) 0,
      (long) 0);

}
/* interrupt service routine timer0 */
isr_t0()
{
  begin_isr_t0(*error);
  count0 = count_timer(TMR0);         /* counter exec_time calculation */
/*********************************************************************/
  /* start ADCs */
  ds1102_ad_start();
  /* read incremental encoder counter */
  inc_k = read_inc(INC1);
  /* calculate mechanical angle */
  eps_m += (PI2 / (FOURFOLD * INC_LINES * SCALE_INC))\
      * (float)(inc_k – inc_k1);
  inc_k1 = inc_k;
  /* max. increase of eps_m in 1 sample T_S –> */
  /* 3000rpm  / 60sec * 100us * 2PI = 0,0314 degrees */
  angle_limit(eps_m);
  /* calculate angle between rotor and stator */
  eps_rs = Pp * eps_m;                  /* Pp –> 2 Polpaare */
  angle_limit(eps_rs);
  /* get phase currents */
  get_phase_currents(i_abc);
  if( flag_pi_fuzzy ) {
    /*****************************************/
    /* INPUTS  : speed_ref = OMEGA–ref / OMEGA–0 */
    /*           speed     = OMEGA     / OMEGA–0 */
    /*           i_abc     –> 3 phase currents   */
    /*                                           */
    /*           pi_ctrl_res_speed               */
    /*           pi_ctrl_current –> for 3 phases */
```

```
    /*******************************************/
 /* 1st path: 'calculating speed for the Enhanced Fuzzy Controlled System' */
  #if FUZZY_SPEED_CONTROL
    d_omega = speed_ref – speed_fol;
    dd_omega = d_omega – d_omega_old;
    d_omega_old = d_omega;
    /* limitation to +/– 500 rpm  */
    if( d_omega >  500.0 ) d_omega =  500.0;
    if( d_omega < –500.0 ) d_omega = –500.0;
                          /* –> uni + 1000 */
                          /* –> 0 .. +50000 */
    d_omega_motor  = (FUZZY) ((d_omega + 500.0) * 50.0);   /* +/– 500 */
    if( dd_omega >  50.0 ) dd_omega =  50.0;
    if( dd_omega < –50.0 ) dd_omega = –50.0;
                          /* –> uni + 100 */
                          /* –> 0 .. +50000 */
    dd_omega_motor = (FUZZY) ((dd_omega + 50.0) * 500.0); /* +/– 50 */
    slip_fu();                          /* generated with FuyyzTech 4.0  */
                    /* –> 0 .. +50000 */
    slip_Speed = (((double) d_slip_motor * 0.02) – 500.0); /* +/– 500 */
  #endif
    /* 1st path: 'calculating current' */
    /* max. sineoidal 4A */
    /* non linear fuzzy controller */
    /* i_ref = Is–ref / Is–0 */
/***** DEBUG ******/
#if 0
    /* slip_Speed (mechanic) */
    slip_Speed = slip_Speed_db;
#endif
/***** DEBUG ******/
    /* slip_Speed in range 0–500 */
    in = (unsigned int) fabs(slip_Speed*2.0) * 65.53;
    /* scaling corresponds to basevar range in fuzzyTECH */
    /* inputs(in) in modified rpm */
    fuz_ctrl( in, &out );
    /* output( out ) in Ampere */
    i_ref = (float) out;
    /* scaling corresponds to basevar range 0–4.5A in fuzzyTECH */
```

```
    i_ref *= test_db;   /* normalize 16 bit uint to 1.0 float */
    i_ref –= 1.0;       /* eleminate offset */
    i_ref += Is_offs;   /* add offset */
    i_ref *= Is_mul;    /* scale motordependant (cur. motor = 1A at 0 rpm) */
    /* reference current limitation for power converter */
      if( i_ref > +4.0 ) i_ref = +4.0;
    /* 2nd path: 'calculating angle' */
    /* new_speed = OMEGA1 / OMEGA0  (Pp tranforms to electrical system) */
    new_speed = (slip_Speed + speed) * Pp;   /* node: summarize (rpm) */
    /* integrates every T_S */
    /* block: integate (rotation per sample) */
    /* 0.01666 = 1 / 60  */
    sum_angle = sum_angle + (new_speed * 0.01666  * T_S);
#if 1
    if( sum_angle > 1.0 )          /* limit positive bound */
      sum_angle –= 1.0;
    if( sum_angle < 0.0 )          /* limit negative bound */
      sum_angle += 1.0;
    /* result always in range 0.0 to 1.0 (one complete rotation) */
#else
    sum_angle –= sum_angle > 1.0 ? 1.0 : 0.0;
#endif
    /* already done by limiting sum_angle in the range 0.0 to 1.0 */
    /* ref_vector = temp – (floor(temp / PI2) * PI2); */ /* modulo 2PI */
    ref_vector = sum_angle * PI2;          /*  */
    ref_vector = ref_vector – PI;          /* move to range –PI <–> +PI */
    /* angle_limit( ref_vector ); */       /* done in line before */
    /* transform vector to sin/cos components */
    sincos( ref_vector, ref_angles );
    /* compute vector with total of 'i_ref' and sin/cos components */
    i_ab.alpha = i_ref * ref_angles.cos;
    i_ab.beta  = i_ref * ref_angles.sin;
    /* transform vector to 3–phase system */
    fc2_3( i_ab, i_abc_ref );
    /* compute delta (ref – is) current */
    /* start current controller for each phase */
    pi_ctrl_current_b.kp = pi_ctrl_current_a.kp;
    pi_ctrl_current_c.kp = pi_ctrl_current_a.kp;
    pi_ctrl_current_b.ki = pi_ctrl_current_a.ki;
```

```
  pi_ctrl_current_c.ki = pi_ctrl_current_a.ki;
  pictrl( i_abc_ref.a, i_abc.a, u_out_abc.a, pi_ctrl_current_a );
  pictrl( i_abc_ref.b, i_abc.b, u_out_abc.b, pi_ctrl_current_b );
  pictrl( i_abc_ref.c, i_abc.c, u_out_abc.c, pi_ctrl_current_c );
  u_out_abc.a *= (UD/I_FUZ_MAX);
  u_out_abc.b *= (UD/I_FUZ_MAX);
  u_out_abc.c *= (UD/I_FUZ_MAX);
  /*****************************************/
  /* OUTPUTS : u_out_abc.a                 */
  /*           u_out_abc.b                 */
  /*           u_out_abc.c                 */
  /*****************************************/
} else {
  /* convert phase currents to stator current vector */
  fc3_2_spec(i_abc,i_ab);
  /*************************************************************************
  *        fluxmodel based on current model                                *
  *************************************************************************/
  /* calculate transformation angle */
  eps_fr += vel_fr * imodel_const.Ki;
  angle_limit(eps_fr);
  eps_fs = eps_fr + eps_rs;
  angle_limit(eps_fs);
  /* transform stator current vector to rotorflux frame */
  sincos (eps_fs,trig_fct);
  vr_neg(i_ab,trig_fct,i_dq);
  /* calculate rotorflux */
  fol(i_dq.d,psi_rd,fol_imodel);
  /* calculate slip frequency */
  if ((x = psi_rd * imodel_const.Ksl) < 0.01) vel_fr = 0.0;
  else vel_fr = i_dq.q / x;
  /*************************************************************************
  *        current control and decoupling circuit                          *
  *************************************************************************/
  /* calculate stator frequency */
  vel_fs = vel_fr + vel_rs;
  /* d-current controller */
  pictrl(i_ref_dq.d,i_dq.d,u_ctrl_dq.d,pi_ctrl_id);
  /* decouple d-voltage output */
```

```
    y = ctrl_decoup.Kl * vel_fs;                    /* auxiliar variable */
    u_decouple_dq.d = y * i_dq.q;
    u_out_dq.d = u_ctrl_dq.d – u_decouple_dq.d;
    /* limit d–voltage output */
    limit(u_out_dq.d,US_MAX);
                      /* d–priority !! */
    /* calculate limit q–voltage output */
    pi_ctrl_iq.ctrl_ub = u_q_max = sqrt(US_MAX * US_MAX – u_out_dq.d *
u_out_dq.d);
    pi_ctrl_iq.ctrl_lb = – u_q_max ;
    /* q–current controller */
    pictrl(i_ref_dq.q,i_dq.q,u_ctrl_dq.q,pi_ctrl_iq);
    /* decouple q–voltage output */
    u_decouple_dq.q = y * i_dq.d;
    u_emf = ctrl_decoup.K_emf * vel_rs * psi_rd;
    u_out_dq.q = u_ctrl_dq.q + u_decouple_dq.q + u_emf;
    /* limit q–voltage output */
    limit(u_out_dq.q,u_q_max);
    /* transform stator voltage vector from rotorflux to stator frame */
    vr_pos(u_out_dq,trig_fct,u_out_ab);
    /* transform stator voltages vector to phase voltages */
    fc2_3(u_out_ab,u_out_abc);
  }
#if 0
/* test pwm */
u_out_abc.a = –UD;       /* 0V */
u_out_abc.b = –UD;       /* 0V */
u_out_abc.c = –UD+UD/10; /* 5V(4V) */
#endif
#if 0
/* test strom */
u_out_abc.a = 0;     /* 0A */
u_out_abc.b = 0;     /* 0A */
u_out_abc.c = UD/30; /* 10V == xA */
#endif
  ds1102_p14_phase_voltages
    ( PVV_ADDRESS,
      (long) (u_out_abc.a / UD * 32767),
      (long) (u_out_abc.b / UD * 32767),
```

```
      (long) (u_out_abc.c / UD * 32767));
/*************************************************************************
*      realization of secound rate for speed controller                  *
*************************************************************************/
  count1++;
  if (count1 >= SPEED_CTRL_R )
  {
   flag = SET;
   count1 = 0;
   /* calculate speed */
   speed =  ( 60 / ( FOURFOLD * INC_LINES * SCALE_INC * T_S * SPEED_CTRL_R))\
        * (float)( inc_k – inc_k1_speed );
   /* calculate velocities */
   vel_m = ( PI2 / 60 ) * speed;
   vel_rs = Pp * vel_m;
   inc_k1_speed = inc_k;
 }
  exec_time = time_elapsed(TMR0, count0);
  service_trace();
  end_isr_t0();
}
```

IM_CTRL.H

This is the header file for file IM_CTRL.C.

```
/* IM_CONST.H ************************************************************
*                                                                         *
* constants for induction motor                                           *
*                                                                         *
* author : F. Schuette                                                    *
* date   : Jan 17, 1994                                                   *
* dSPACE & UNI–GH PADERBORN/LEA                                           *
*                                                                         *
**************************************************************************/

/* motor–constants  */
#define Ls              130.00e–3               /* stator inductivity */
#define Lh              120.00e–3               /*   main inductivity */
#define Lr              130.00e–3               /*  rotor inductivity */
#define Rr              3.0                     /*   rotor resistance */
#define Rs              1.86                    /*  stator resistance */
#define Pp              2.0                     /*    number of poles */
#define sigmaS         (Ls/Lh–1.0)              /* stator leakage factor */

/* control–constants */
             /* standard T_S 100us  */
#define T_S             150.0e–6               /* sampling periode */
#define SPEED_CTRL_R    5   /* ratio T_S(speed–ctrl)/T_S(current–ctrl) */
#define FLUX_CTRL_R                     5
```

```
#define US_MAX          169.0                   /*       voltage limit */
#define UD              300.0                   /*     DC link voltage */
#define I_FUZ_MAX         4.0                   /*                     */
#define ISD_MAX          10.0                   /*     d-current limit */
#define ISQ_MAX           4.0                   /*     q-current limit */
#define PSI_REF           0.26                  /* flux reference */
#define MAX_SPEED      3000.0                   /* limit speed (rpm) */

/* control-parameters */

#define K_ir            52.1                  /* PI current controller */
#define Tn_ir           4.3541e-3

#define K_nr            0.06                    /* PI speed controller */
#define Tn_nr           0.00698

#define K_fr                    600                     /* Flux controller */
#define Tn_fr             43e-3

#define Tn_filter       1.5e-3                          /* speed filter */
              /* Tn_filter = 3 * T_S * SPEED_CTRL_R */
```

VARS.H

This file provides variable declarations.

```
/* VARS.H ************************************************************
*                                                                    *
* declaration of variables                                           *
*                                                                    *
* author : F. Schuette, R. Koenigbauer, S. Beierke                   *
* date   : Jan 11, 1994                                              *
* dSPACE & UNI-GH PADERBORN/LEA & INFORM & TI                        *
*                                                                    *
**********************************************************************/

#include "typedef.h"

/*************************
* FLOAT and INT variables *
*************************/

/* voltages at control system */
dq_type         u_out_dq;
dq_type         u_ctrl_dq;
dq_type         u_decouple_dq;
ab_type         u_out_ab;
abc_type        u_out_abc;
float           u_emf;
float           u_q_max;                        /* max. absolute value */

/* currents */
dq_type         i_ref_dq;
dq_type         i_dq;
ab_type         i_ab;
abc_type        i_abc;

/* velocities */
float   vel_m;
float   vel_rs;
float   vel_fs;
float   vel_fr;

/* speed */
float   speed_ref;
```

```
float   speed_ref_fol;
float   speed;
float   speed_fol;
float   speed_fir;

/* angles at control system */
float   eps_fs;
float   eps_fr;
float   eps_rs;
float   eps_m;

/* space vector */
cossin_type     trig_fct;

/* rotorflux */
float   psi_rd;
float  psi_rd_ref;

/* execution time */
float   exec_time;

/* signal generator */
float   p_freq;
float   p_ampl;
float   p_offs;
float   p_duty;
int     p_wave;
int     sig_flag;

/* input variables */
long inc_k;
long inc_k1;
long inc_k1_speed;

/* counter variables */
unsigned long   count0;                             /* for exec_time */
unsigned long   count1;                           /* for speed control */

/* speed interrupt */
volatile int            flag;          /* protected against optimising */

/* slave processor variables */
long    p14_ioport;                             /* value of p14 ioport */

/* variables for fuzzy computation */
float slip_Speed;
float slip_Speed_db;
float new_speed;
float new_speed_fir;
float new_speed_db;
float sum_angle;
float i_ref;
float i_ref_fir;
float i_ref_db;
float ref_vector;
float d_omega;
float d_omega_old;
float dd_omega;

float test_db;

unsigned int current_offset;

int in,out,flag_pi_fuzzy;

float Is;
float Im;
float Us;
float omega1;
```

```
float temp;
float Is_offs;
float Is_mul;
ab_type     u_ab;

ab_type     i_ab;
abc_type    i_abc_ref;
abc_type    i_abc_fuz;
abc_type    i_abc_help;
cossin_type ref_angles;

/*************************
* FLOAT parameters       *
*************************/

/* struct PI_CTRL */
struct  pi_ctrl_type {
  float kp;
  float ki;
  float yi_k1;
  float e_k1;
  float ctrl_lb;                                        /* lower bound */
  float ctrl_ub;                                        /* upper bound */
} pi_ctrl_id, pi_ctrl_iq, pi_ctrl_speed, pi_ctrl_flux,
  pi_ctrl_res_speed, pi_ctrl_current_a, pi_ctrl_current_b, pi_ctrl_current_c;

/* struct foltr_1 */
struct  fol_tr_type {
  float K1;
  float K2;
  float u_k1;
} fol_imodel,fol_speed_ref,fol_speed;

/* struct flux–model */
struct  imodel_type {
  float Ki;
  float Ksl;
} imodel_const;

/* struct current control decoupling */
struct ctrl_decoup_type {
  float Kl;
  float K_emf;
} ctrl_decoup;

/* struct FIR */
struct FIR_type {
  float x0;
  float x1;
  float x2;
  float x3;
} FIR_speed, FIR_i_ref, FIR_new_speed;

/* struct FIR */
struct FIR2_type {
  float x0;
  float x1;
} FIR2_speed, FIR2_i_ref, FIR2_new_speed;
```

GEN_MOD.C

This file provides general functions, such as PI controllers and FIR filters.

```
/* GEN_MOD.C ************************************************************
*                                                                       *
* control and filter macros for induction motor control                 *
*                                                                       *
* author : F. Schuette                                                  *
* date   : Jan 12, 1994                                                 *
* dSPACE & UNI-GH PADERBORN/LEA                                         *
*                                                                       *
************************************************************************/

/************************************************************************
*                                                                       *
* first order lag                                                       *
*                                                                       *
* worst-case execution time: 0.43 us                                    *
*                                                                       *
* call : fol(u,y,struct_fol)                                            *
*    u : input                                                          *
*    y : output                                                         *
*                                                                       *
*   K1 : (2T1-T_S)/(2T1+T_S)          G(s)=kp/(1+s*T1)                  *
*   K2 : (kp*T_S)/(2T1+T_S)                                             *
*                                                                       *
* author : F. Schuette                                                  *
* date   : Jan 12, 1994                                                 *
* dSPACE & UNI-GH PADERBORN/LEA                                         *
*                                                                       *
************************************************************************/

#define fol(u,y,record) {\
 float x;                                        /* auxiliar variable */\
 x = record.K2 * u;\
 y = record.K1 * y + x + record.u_k1;\
 record.u_k1 = x ;\
}
/************************************************************************
*                                                                       *
* PI controller                                                         *
*                                                                       *
* worst-case execution time: 1.82 us                                    *
*                                                                       *
* call : pictrl(u1,u2,y,struct_pi)                                      *
*   u1 : reference input                                                *
*   u2 : actual input                                                   *
*    y : output                                                         *
*                                                                       *
*      kp : gain                      G(s)=kp*(1+s*Tn)/(s*Tn)           *
*      ki : T_S*Kp/(2*Tn)                                               *
*    e_k1 : error e(k-1)                                                *
*   yi_k1 : I output yi(k-1)                                            *
* ctrl_lb : lower bound  PI controller                                  *
* ctrl_ub : upper bound  PI controller                                  *
*                                                                       *
* author : F. Schuette                                                  *
* date   : Jan 12, 1994                                                 *
* dSPACE & UNI-GH PADERBORN/LEA                                         *
*                                                                       *
************************************************************************/

#define pictrl(u1,u2,y,record) {\
 float e,yi;                                    /* auxiliar variables */\
```

```
 e = u1 – u2;                                                    /* error */\
                 /* integration */\
 yi = (e + record.e_k1) * record.ki + record.yi_k1 ;       /* I output */\
 y = yi + record.kp * e;\
 if (((y < record.ctrl_lb) || ( y > record.ctrl_ub))\
    && (e * record.yi_k1 > 0.0 )) {}\
 else record.yi_k1 = yi;\
 if (y < record.ctrl_lb)   y = record.ctrl_lb;\
 if (y > record.ctrl_ub)   y = record.ctrl_ub;\
 record.e_k1 = e; \
}

/*************************************************************************
*                                                                        *
* angle limit                                                            *
*                                                                        *
* worst–case execution time: 0.56 us                                     *
*                                                                        *
* call : angle_limit(eps)                                                *
*  eps : angle                                                           *
*                                                                        *
* author : F. Schuette                                                   *
* date   : Jan 12, 1994                                                  *
* dSPACE & UNI–GH PADERBORN/LEA                                          *
*                                                                        *
*************************************************************************/

#define angle_limit(eps) {\
 if (eps >=  PI) eps –=PI2;\
 if (eps <= –PI) eps +=PI2;\
}

/*************************************************************************
*                                                                        *
* get_phase_currents                                                     *
*                                                                        *
* worst–case execution time:  2.78 us                                    *
*                                                                        *
*   call : get_phase_currents(i_abc)                                     *
*  i_abc : phase currents                                                *
*                                                                        *
* author : F. Schuette                                                   *
* date   : Jan 12, 1994                                                  *
* dSPACE & UNI–GH PADERBORN/LEA                                          *
*                                                                        *
*************************************************************************/

#define get_phase_currents(i_abc){\
 i_abc.a = SCALE_CURRENT * ds1102_ad(ADC3);\
 i_abc.b = SCALE_CURRENT * ds1102_ad(ADC4);\
 i_abc.c = – i_abc.a – i_abc.b;\
}
```

```
/*********************************************************************
* sincos                                                             *
*                                                                    *
* sine is approximated through polynom of 4.order                    *
* cosine is approximated through polynom of 5.order                  *
*                                                                    *
* worst-case execution time: 1.6 us                                  *
*                                                                    *
* call : sincos(eps,dsv);                                            *
* eps  : angle                                                       *
* dsv  : direction space vector with cosine-, sine-component         *
*                                                                    *
* author : F. Schuette                                               *
* date   : Jan 12, 1994                                              *
* dSPACE & UNI-GH PADERBORN/LEA                                      *
*                                                                    *
*********************************************************************/

#define sincos(eps,dsv) {\
  if ( eps >= 0.0 ) {\
    dsv.sin = (((0.0372*(eps)-0.2338)*(eps)+0.0544)*(eps)+0.9826)\
          *(eps)+0.0013; \
    dsv.cos = ((((-0.0076*(eps)+0.0595)*(eps)-0.0211)*(eps)-0.4879)\
          *(eps)-0.0028)*(eps)+1.0;\
  }\
  else {\
    dsv.sin = -((((0.0372*(eps)+0.2338)*(eps)+0.0544)*(eps)-0.9826)\
          *(eps)+0.0013);\
    dsv.cos = ((((0.0076*(eps)+0.0595)*(eps)+0.0211)*(eps)-0.4879)\
          *(eps)+0.0028)*(eps)+1.0;\
  }\
}

/*********************************************************************
* limit                                                              *
*                                                                    *
* worst-case execution time: 0.61 us                                 *
* call : limit(u,bound)                                              *
*    u : voltage input                                               *
*                                                                    *
* author : F. Schuette                                               *
* date   : Jan 12, 1994                                              *
* dSPACE & UNI-GH PADERBORN/LEA                                      *
*                                                                    *
*********************************************************************/

#define limit(u,bound){\
  if ( u > bound )  u = bound;\
  if ( u < (-bound) )  u = -bound;\
}

/*********************************************************************
*                                                                    *
* FIR 4. order                                                       *
*                                                                    *
* worst-case execution time: ?.?? us                                 *
*                                                                    *
* call : FIR4(u,y,record)                                            *
*    u : input                                                       *
*    y : output                                                      *
*                                                                    *
* author :                                                           *
* date   : Mar  9, 1995                                              *
*                                                                    *
*********************************************************************/
```

```
#define FIR4(u,y,record) {\
 record.x3 = record.x2; \
 record.x2 = record.x1; \
 record.x1 = record.x0; \
 record.x0 = u; \
 y = 0.25 * record.x3 + 0.25 * record.x2 + \
     0.25 * record.x1 + 0.25 * record.x0;  \
}
/*********************************************************************
*                                                                    *
* FIR 2. order                                                       *
*                                                                    *
* worst-case execution time: ?.?? us                                 *
*                                                                    *
* call : FIR2(u,y,record)                                            *
*    u : input                                                       *
*    y : output                                                      *
*                                                                    *
* author :                                                           *
* date   : April 2, 1995                                             *
*                                                                    *
*********************************************************************/
#define FIR2(u,y,record) {\
 record.x1 = record.x0; \
 record.x0 = u; \
 y = 0.5 * record.x1 + 0.5 * record.x0; \
}
```

IM_CTRL.LNK

This file provides the system memory map of the dSPACE board DS1102 and the TMS320C31 DSP.

```
/* IM_CTRL.LNK ***************************************************
*                                                                  *
* system memory map of the DS1102 TMS 320C31 processor board       *
*                                                                  *
* comment : used as linker command file                            *
*           .bss in on-chip memory                                 *
*           stack size 0x400                                       *
*                                                                  *
* author : F. Schuette                                             *
* date   : Jan 11, 1994                                            *
* dSPACE & UNI-GH PADERBORN/LEA                                    *
*                                                                  *
*******************************************************************/
-heap  0x0400
-stack 0x0400

MEMORY
{
  VECS: org = 0x000000  len = 0x0000c
  TRAP: org = 0x000020  len = 0x00020
  PMEM: org = 0x000840  len = 0x1f7c0   /* PRIMARY MEMORY             */
  IMEM: org = 0x809800  len = 0x00800   /* RAM BLOCK 0                */
}

/* SECTION ALLOCATION INTO MEMORY */

SECTIONS
{
  .vectors:             {} > VECS /* RESET VECTOR                     */
```

```
  .trap:                {} > TRAP /* TRAP VECTORS                      */
  .startup:             {} > PMEM /* STARTUP CODE                     */
  .text ALIGN (32):     {} > PMEM /* C-CODE                           */
  .cinit:               {} > PMEM /* INITIALIZATION TABLES            */
  .const:               {} > PMEM /* STRING LITERALS AND SWITCH TABLES */
  .stack:               {} > IMEM /* SYSTEM STACK                     */
  .bss: block(0x10000)  {} > PMEM /* GLOBAL & STATIC VARS (SEE NOTE 2) */
  .sysmem:              {} > PMEM /* DYNAMIC MEM - DELETE IF NOT USED  */
  .hostmem: {.+=8000h;} > PMEM, type=NOLOAD  /*  Host Memory          */
}

/* MODULES WHICH ARE ALWAYS LINKED */

-u startup
-l \dsp_cit\c31\ds1102.lib
-l \dsp_cit\c31\ft320_30.lib
-l \c30tools\rts30.lib
-l \dsp_cit\c31\slib31.lib
-l \dsp_cit\c31\flib31.lib
```

IM_CTRL.TRC

This file includes all the variables which you can use with dSPACE's COCKPIT31 and TRACE31 software.

```
-- IM_CTRL.TRC

-- variables available for TRACE and COCKPIT
-- induction motor
-- author : F.Schuette
-- date   : Jan 15, 1994
-- dSPACE & UNI-GH PADERBORN/LEA

sampling_period = 1.0e-4

exec_time       flt
-- rotorflux
psi_rd          flt
-- direction space vector
group "space vector"
trig_fct[0..1]  flt
trig_fct.cos  renames trig_fct[0]
trig_fct.sin  renames trig_fct[1]
endgroup
-- voltages at control system
group "voltages"
u_out_ab[0..1]  flt
u_out_ab.alpha  renames u_out_ab[0]
u_out_ab.beta   renames u_out_ab[1]

u_out_abc[0..2] flt
u_out_abc.a     renames u_out_abc[0]
u_out_abc.b     renames u_out_abc[1]
u_out_abc.c     renames u_out_abc[2]

u_out_dq[0..1]  flt
u_out_dq.d      renames u_out_dq[0]
u_out_dq.q      renames u_out_dq[1]

u_ctrl_dq[0..1] flt
u_ctrl_dq.d     renames u_ctrl_dq[0]
u_ctrl_dq.q     renames u_ctrl_dq[1]

u_decouple_dq[0..1]     flt
u_decouple_dq.d         renames u_decouple_dq[0]
```

```
u_decouple_dq.q         renames u_decouple_dq[1]
u_emf                   flt
u_q_max                 flt
endgroup

-- currents
group ”current”
group ”alpha/beta”
i_ab[0..1]      flt
i_ab.alpha      renames i_ab[0]
i_ab.beta       renames i_ab[1]
endgroup
group ”dq”
i_ref_dq[0..1]  flt
i_ref_dq.d      renames i_ref_dq[0]
i_ref_dq.q      renames i_ref_dq[1]
i_dq[0..1]      flt
i_dq.d          renames i_dq[0]
i_dq.q          renames i_dq[1]
endgroup
group ”abc”
i_abc[0..2]     flt
i_abc.a         renames i_abc[0]
i_abc.b         renames i_abc[1]
i_abc.c         renames i_abc[2]
i_abc_ref[0..2]     flt
i_abc_ref.a         renames i_abc_ref[0]
i_abc_ref.b         renames i_abc_ref[1]
i_abc_ref.c         renames i_abc_ref[2]
i_abc_help[0..2]     flt
i_abc_help.a         renames i_abc_help[0]
i_abc_help.b         renames i_abc_help[1]
i_abc_help.c         renames i_abc_help[2]
endgroup
endgroup
-- velocities
group ”velocity”
vel_m           flt
vel_rs          flt
vel_fs          flt
vel_fr          flt
endgroup
-- revolutions
group ”revolutions”
speed_ref       flt
speed_ref_fol   flt
speed           flt
speed_fol       flt
speed_fir       flt
endgroup
-- angles at control system
group ”angles”
eps_fs          flt
eps_fr          flt
eps_rs          flt
eps_m           flt
inc_k           int
endgroup
-- signal generator
group ”signal”
p_freq                  flt
p_ampl                  flt
p_offs                  flt
p_duty                  flt
```

```
p_wave                  int
sig_flag                int
flag_pi_fuzzy           int
endgroup
group ”PI-Parameter”
pi_ctrl_iq[0..6]   flt
pi_ctrl_iq.kp        renames pi_ctrl_iq[0]
pi_ctrl_iq.ki        renames pi_ctrl_iq[1]
pi_ctrl_speed[0..6]   flt
pi_ctrl_speed.kp        renames pi_ctrl_speed[0]
pi_ctrl_speed.ki        renames pi_ctrl_speed[1]
endgroup
-- fuzzy section
group ”fuzzyTECH”
slip_Speed              flt
slip_Speed_db           flt
new_speed               flt
new_speed_fir           flt
new_speed_db            flt
omega1       flt
i_ref                   flt
i_ref_fir               flt
i_ref_db                flt
current_offset          int
test_db                 flt
sum_angle               flt
ref_vector              flt
Is        flt
Is_offs      flt
Is_mul       flt
Im        flt
Us        flt
d_omega      flt
d_omega_old     flt
dd_omega     flt
in                      int
out                     int
ref_angles[0..1]        flt
ref_angles.cos  renames ref_angles[0]
ref_angles.sin  renames ref_angles[1]
pi_ctrl_res_speed[0..6]   flt
pi_ctrl_res_speed.kp        renames pi_ctrl_res_speed[0]
pi_ctrl_res_speed.ki        renames pi_ctrl_res_speed[1]
pi_ctrl_current_a[0..6]   flt
pi_ctrl_current_a.kp        renames pi_ctrl_current_a[0]
pi_ctrl_current_a.ki        renames pi_ctrl_current_a[1]
endgroup
```