AN1112
APPLICATION NOTE

Three-Phase Motor Control by using ST52x301

Authors: M. Di Guardo, G. Grasso, M. Lo Presti

Introduction

Induction motors with squirrel-cage are widely used in industrial environments because of their low cost and rotors rugged construction.

The induction motor is a simple and robust machine, but its control migth be a complex task. When managed directly from the line voltage, the motor operates at nearly a constant speed. To obtain speed and torque variations, it is necessary to modify both the voltage and the frequency, by using an electronic converter must be used to perform this operation.

The best way to run the motor is by using a PWM sine-wave modulation, but in many applications this can result very expensive for the complex implementability.

The six-step modulation (square wave) is a low-cost solution that allows to run the motor at various speeds.

The aim of this application is to describe how ST52x301 can easily work to obtain frequency and voltage variations in the Inverter Driver and how to perform a closed loop control.

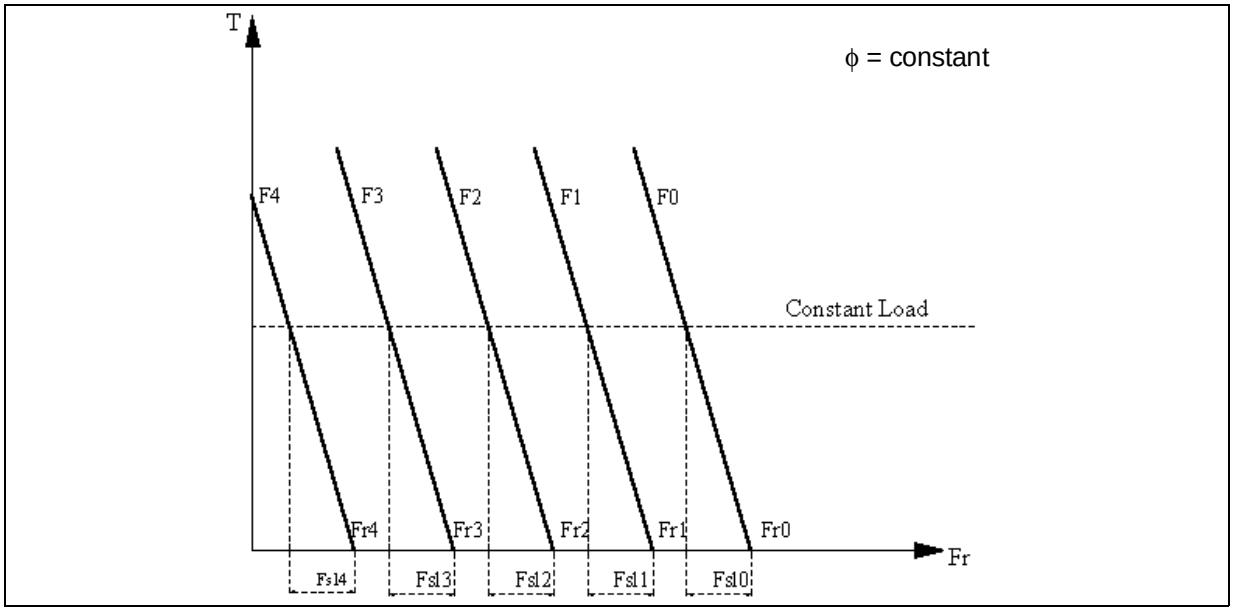
Speed control by varying stator frequency and voltage

The rotor speed can be controlled by varying the frequency of the stator voltage F. This is possible by varying Vs in a linear proportion to F.

Varying the stator frequency and voltage is the preferred technique in most variable-speed induction motor drive applications. This technique is known as V/F=constant.

As displayed in figure 1, for a small value of slip frequency (Fsl= Fs-Fr) and for fixed flux values, a linear relationship between Torque T and Fsl (slip) takes place at any frequency F.

Fig . 1 Torque Speed Characteristics

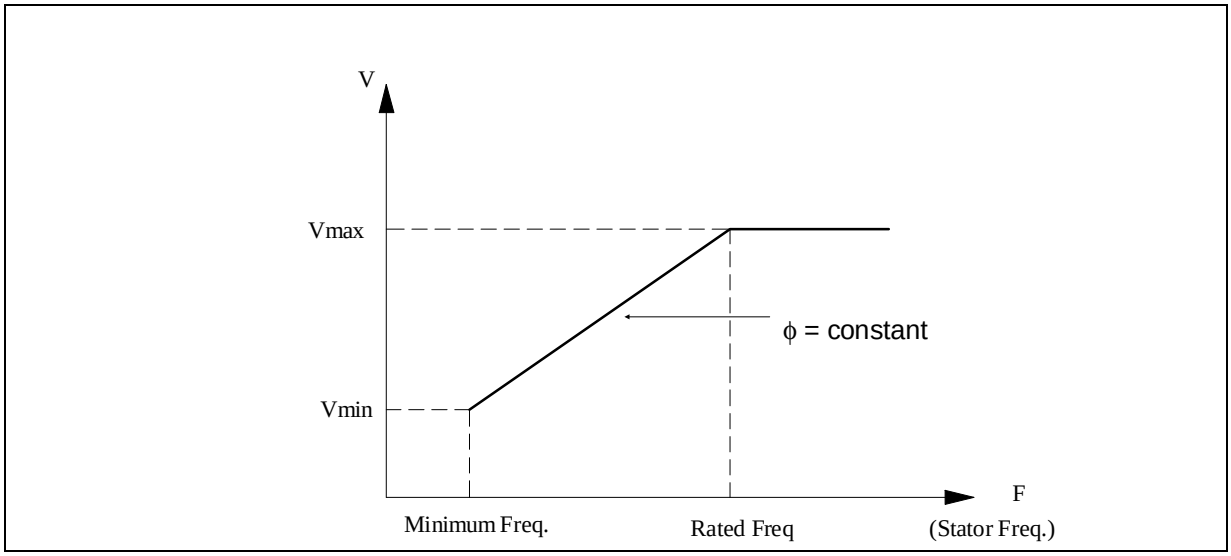


In order to eliminate one control variable, it is possible to fix the V/F ratio as follows:

$$\frac{V}{F}=const=K$$

this is equal to fix the magnetic flux in the motor.

Fig . 2 - Voltage vs Frequency Relation



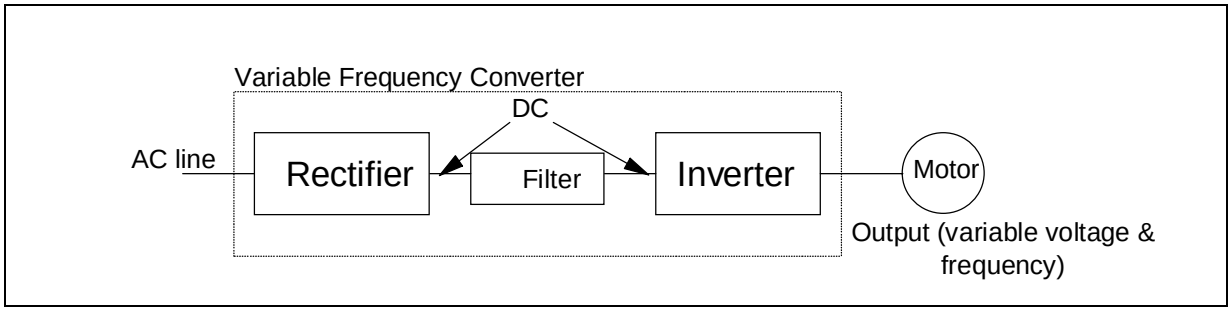
V/F=K control method or V/F fuzzy variation can be successfully applied for a large class of applications.

A way to choose this value could be, to determinate the max torque applied (Load) and the corresponding speed; that means to evaluate the maximum requested magnetic flux.

In order to change the stator voltage frequency it is necessary to use a variable frequency converter. The variable frequency converter, which acts as an interface between the utility power system and the induction motor, must satisfy the following basic requirements:

- Ability to adjust the frequency according to the desired output speed
- Ability to adjust the output voltage so as to maintain a constant air gap flux in the constant torque region.
- Ability to supply a rated current on a continuous basis at any frequency.

Fig . 3 - Variable Frequency Converter



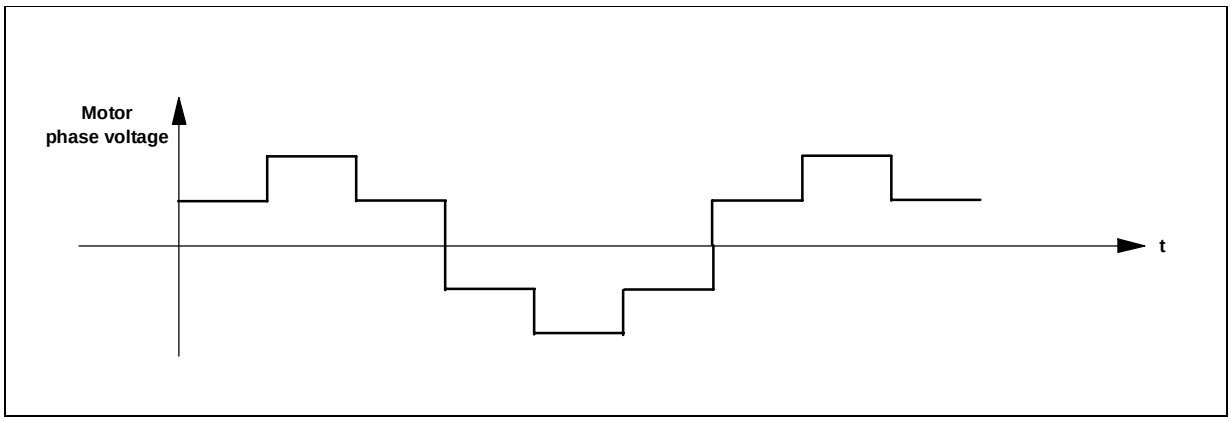
The variable frequency converter can be implemented by using several techniques:

- Pulse Width Modulated sinusoidal voltage source inverter (PWM VSI)
- Square wave voltage source inverter (square wave VSI or six step modulation)

Square Wave Inverter (Six step modulation)

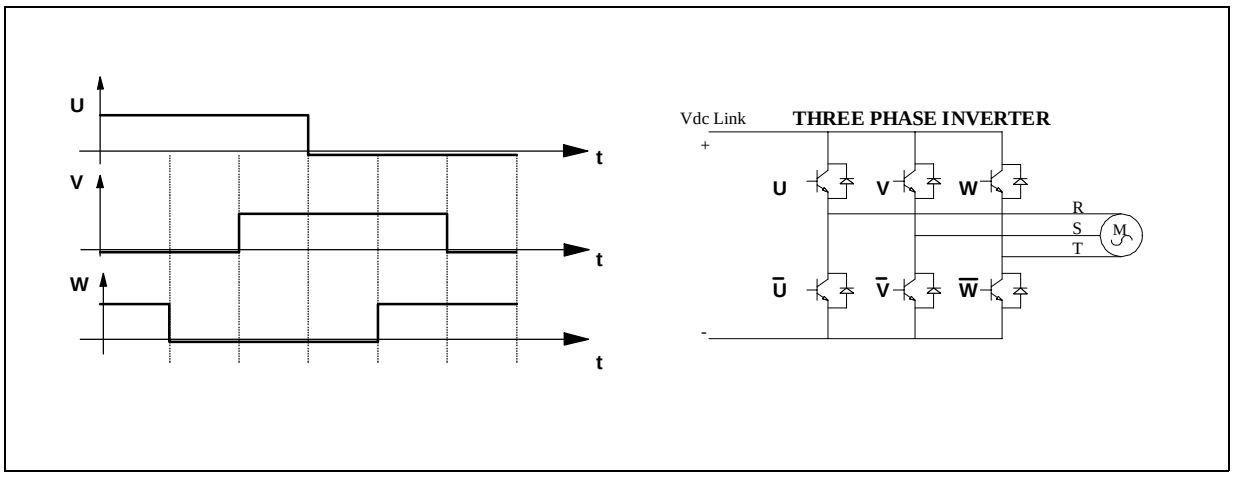
With square wave inverter operations, each inverter switch is on for 180° and a total of three switches are on at any instant of time. The resulting voltage is shown in figure 4.

Fig . 4 - One Phase Voltage



In order to implement the six-step modulation, the Inverter must be driven by using the following signals:

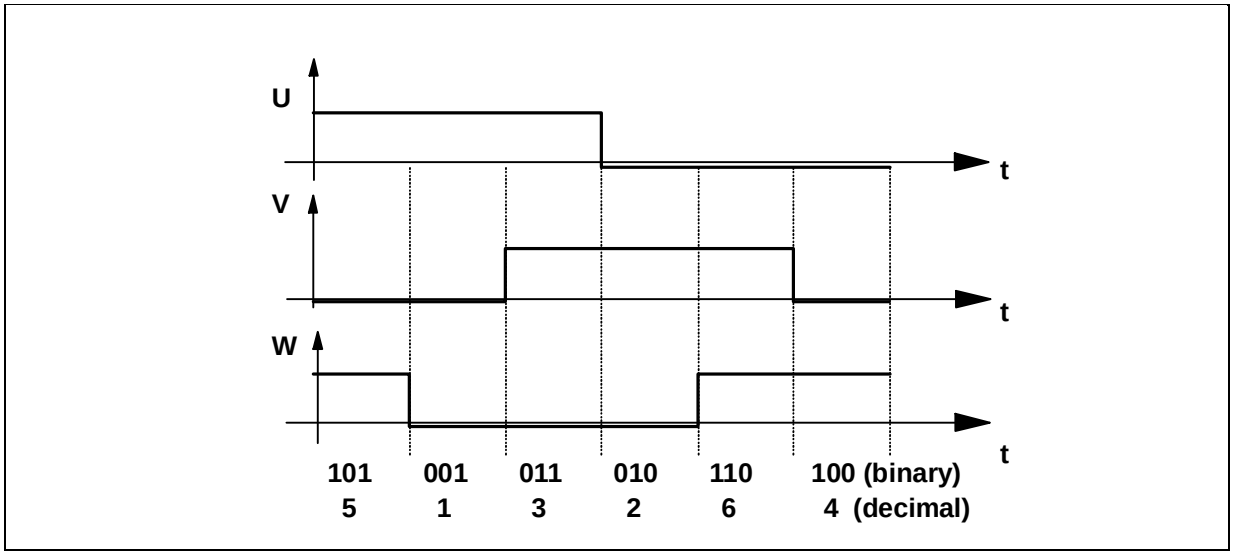
Fig . 5 - Bridge Control Signals



Since the inverter is operating in a square wave mode, the magnitude of the motor voltage is controlled by Vdc link, i.e. the DC bridge supply voltage.

Changing the frequency of the three signals it is possible to change the frequency of the stator voltage. These three square wave signals can be easily obtained by using 3 digital I/0s and the Timer of ST52x301. In particular, these signals are obtained by writing sequentially on the 3 selected bits of the digital I/O port as shown in figure 6:

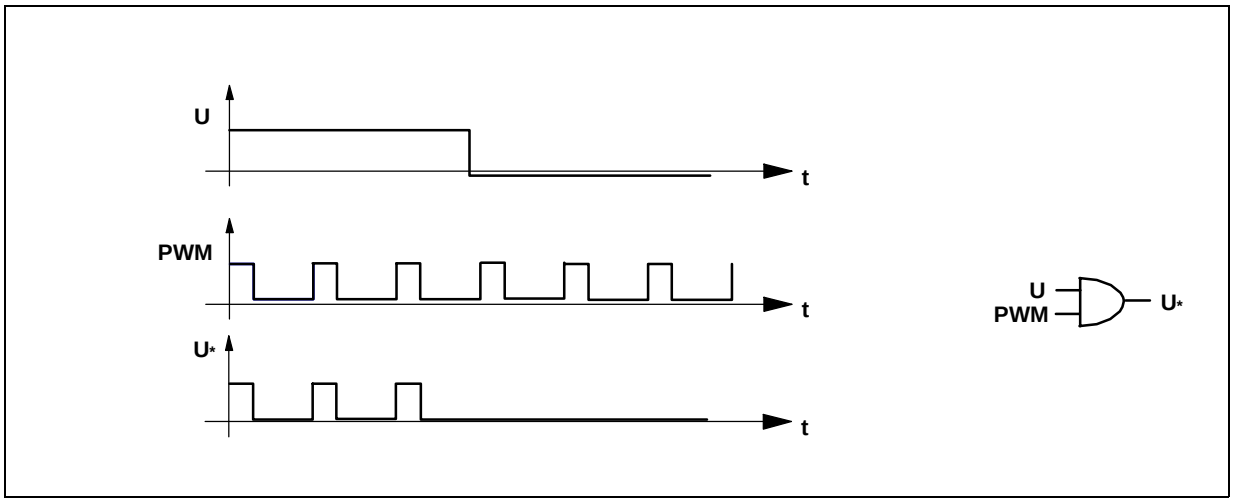
Fig. 6 - Digit sequency on ST52x301 parallel port



The Timer counter is used to change the speed in writing sequence, in this way the frequency of the 3 square waves is modified. Instead, the magnitude of the stator voltage is modified by using the ST52x301 PWM.

In order to change the value of the stator voltage, the 3 square waves are AND'ed with a high frequency PWM whose duty cycle is managed by the Triac counter register of the ST52x301 pin (24). The high frequency PWM is then the switching frequency of the inverter transistors (Fig. 7).

Fig. 7 - Bridge PWM control signal



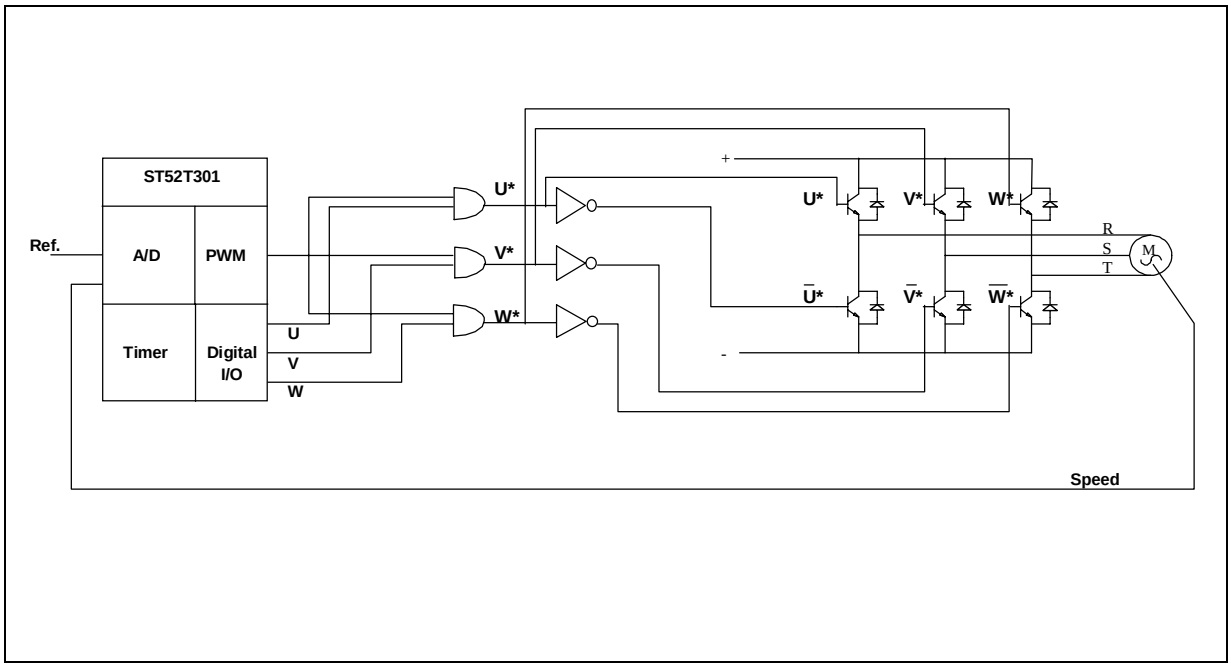
By using this method the line to line r. m. s. voltage is about:

$$V(rms) = 0.78*Vd*d$$

where d is the PWM duty cycle

In the following figure is shown the schematic block of the ST52x301 board that allows to reproduce a six step modulation. The main parts of this system are the microcontroller, the circuit utilized to provide the bridge control signals (**U*, V*, W*** and **U***, **V***, **W***) and the three-phase inverter driver.

Fig . 8 - System Schematic block



In order to avoid cross conduction problems a delay, "dead time", must be added between **U*, V*, W*** and **U***, **V***, **W***. In figure 9 is shown the circuit used to create the "dead time".
The RC value and the negative schmitt trigger threshold voltage, V_N, determine the delay t_{dt} as follows:

$$t_{dt}=-RC*\ln(V_N/5)$$

Fig.9 Dead Time Circuit

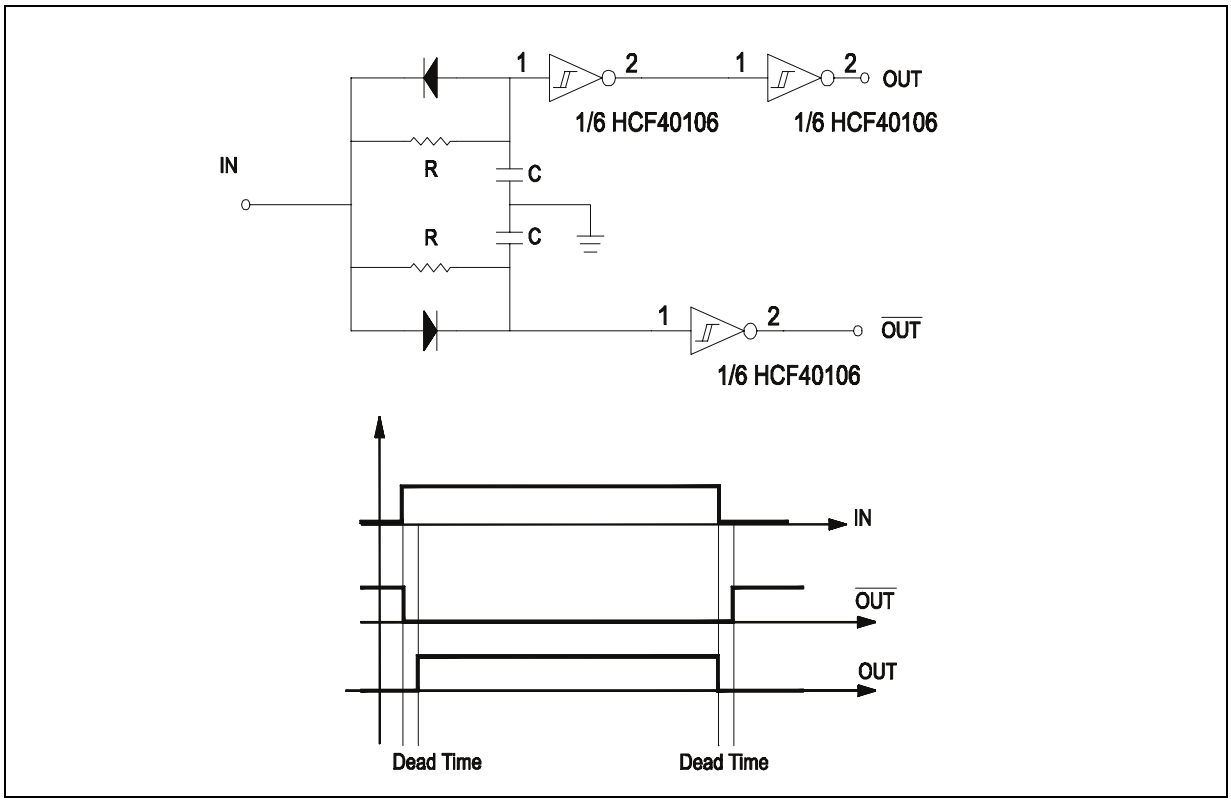
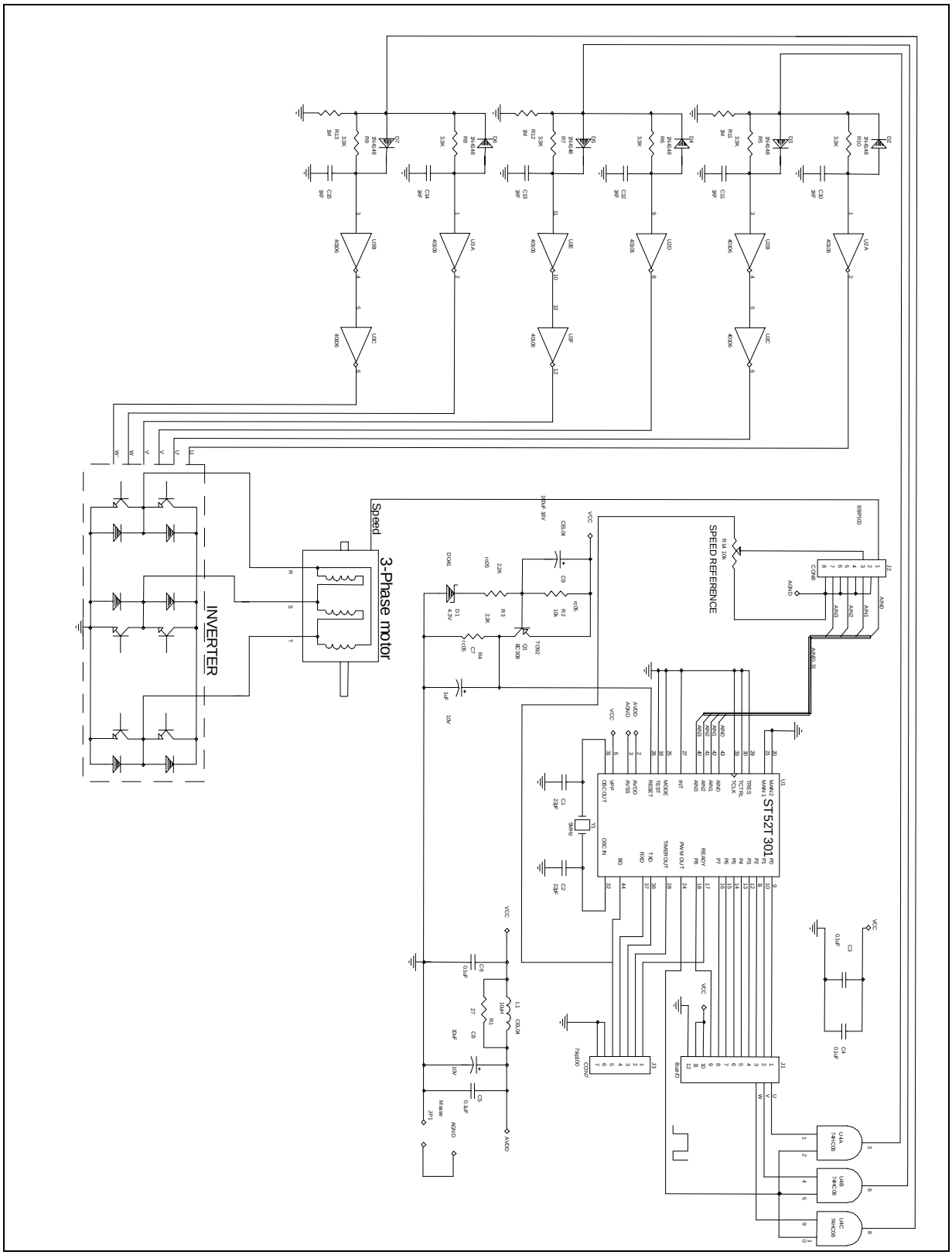


Fig.10 Six Step modulation schematic for HW implement.



Six step modulation S/W implementation by ST52x301

The software implementation is shared in a main program and an interrupt subroutine designed with FUZZYSTUDIO™ 3.0.

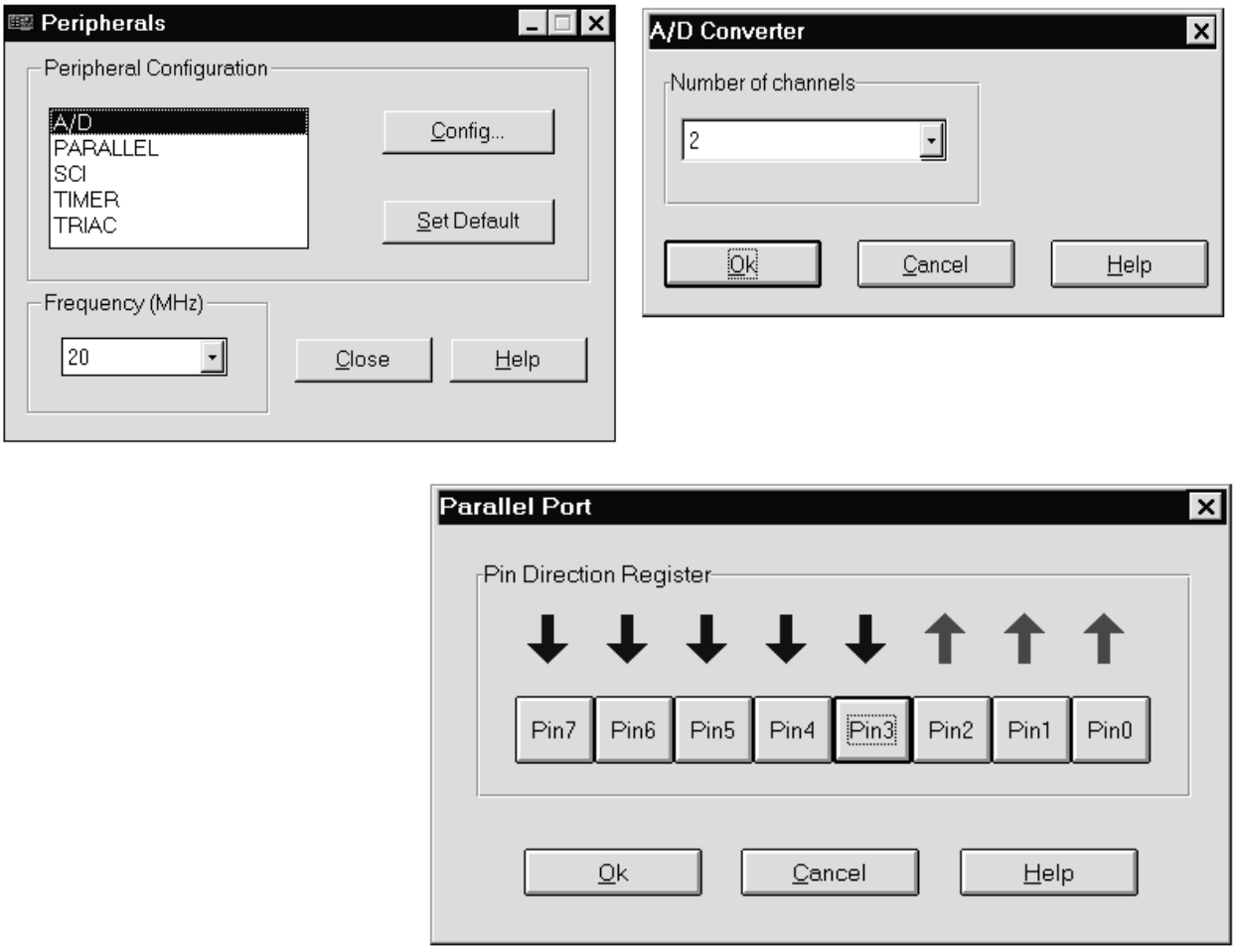
Peripherals Configuration

The first step of the program development is the peripherals configuration.

The following figures show the peripheral configurations:

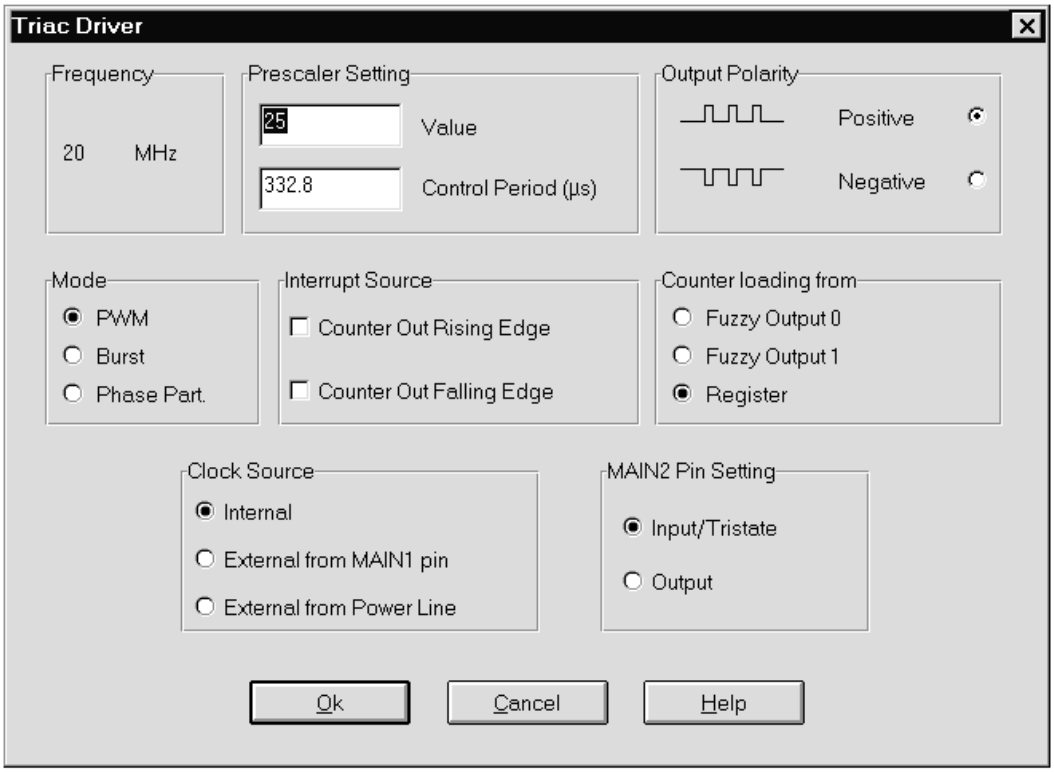
- the parallel port is configured with 3 output lines and 5 input lines; Pin0 = U; Pin1=v; Pin2 = W
- the Analog to Digital converts: two channels, CHAN0 = Speed feedback, CHAN1 = Reference.

Fig.11 Peripherals Configuration



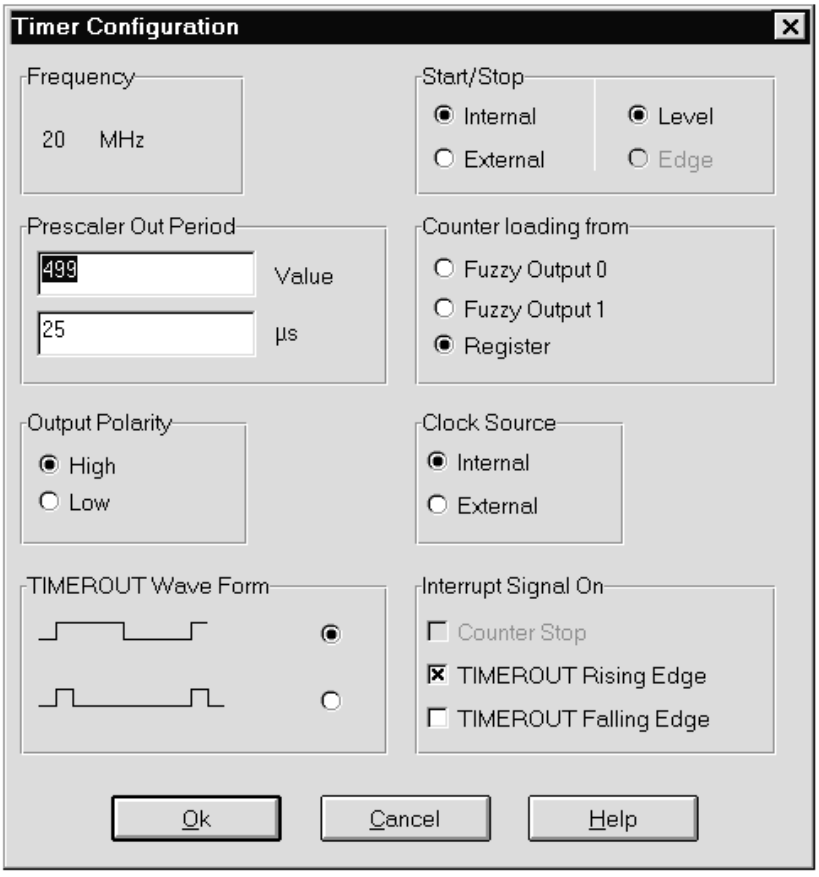
- The peripheral named Triac PWM Driver, generates a PWM signal with a fixed period. This period, imposed by the Prescaler value, 25, is 332 µs (3 kHz). The duty cycle of this signal is utilized to modulate the voltage amplitude.

Fig . 12 - Peripherals Configuration



- The Timer configuration allows to change the motor speed range. Six step signal period changes, according to the Timer counter register value, from 400µs x 1 x 6 (416 Hz) up to 400µs x 255 x 6 (17Hz), if the prescaler is 7999.

Fig . 13 - Peripherals Configuration



Main Program

The following figure shows the Main Program Window. The first two blocks ("*Timer_Int_Setting*" and "*Timer_Int_Priority*") allow the interrupts mask configuration. In this case, only the Timer is enabled to supply an interrupt, each time the Timer reaches the counter register value. This interrupt is utilized to synchronize the phase switching.

The block "*Variables_Initialization*" assigns a default value to the global variables while the "*Digital_Port_bit_set*" block sets the parallel port U-V-W pins. The following blocks, "*Voltage_level_setting*" and "*Start_PWM*", are used respectively to set and start the PWM signal on ST52x301 Triacout pin.

The "*Start_AD*" block enables analog to digital converter to work in continuous mode.

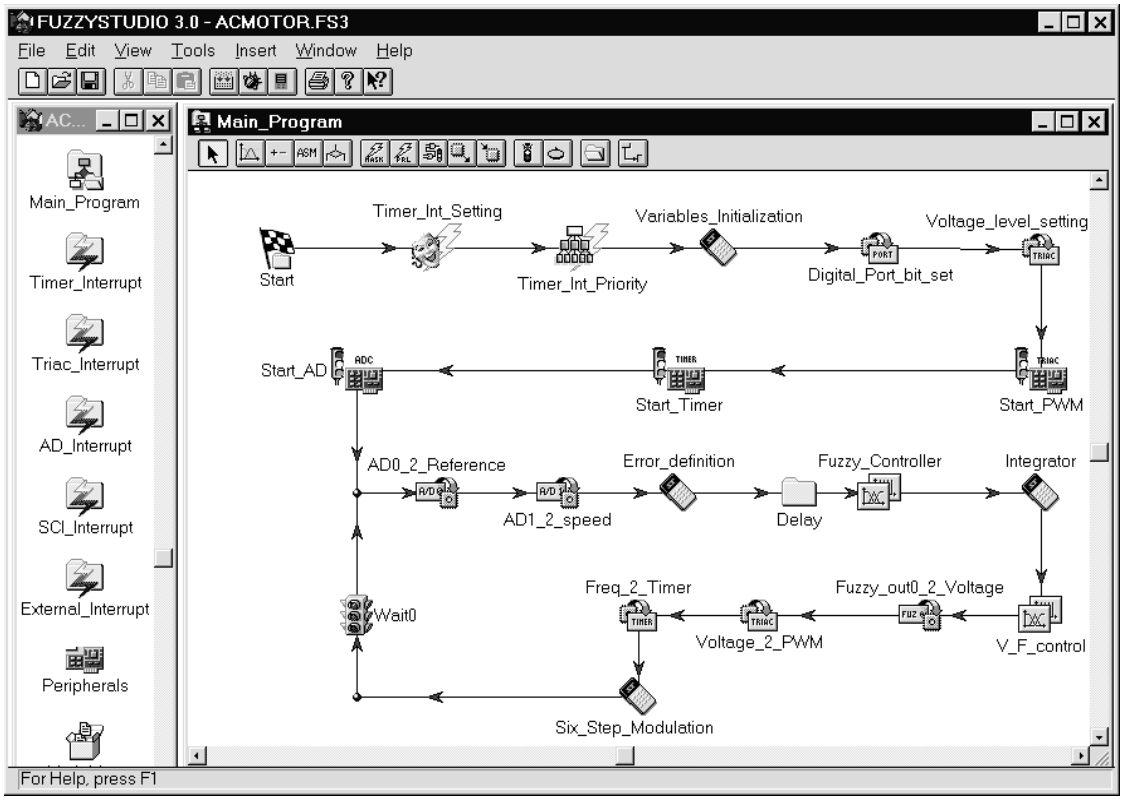
By means of the following two blocks, the converted values of measured speed and speed reference are read and stored into two global variables. The "*Error_definition*" block performs the error calculation as folllows:

error=Reference - speed

After a time delay the resulting 'error' value is sent to the "*Fuzzy_controller*" Block. This fuzzy block implements a fuzzy algorithm which produces the frequency increment or decrement Δf to be added to the actual stator frequency, in order to obtain the desidered speed.

Details on this algorithm will be given later on.

Fig . 14 - Program's Main view



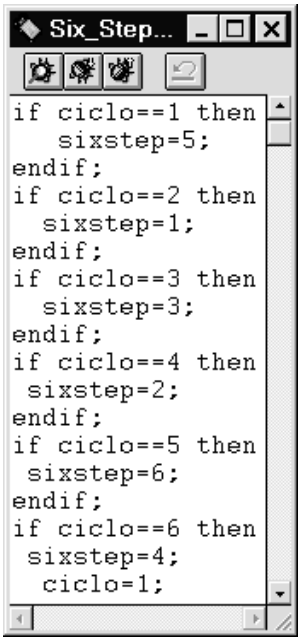
The block named "*integrator*" updates the current speed and checks for possible under/overflow in the algebric sum. The "*V_F_control*" Fuzzy Block is used as fuzzy model of the Voltage-Frequency relationship, whose diagram is shown in fig.2. This fuzzy block outputs a voltage value for each frequency input value. "*Voltage_2_PWM*" and "*Freq_2_Timer*" blocks updates Triac counter register and Timer counter register, in order to change the PWM mean value and rotor speed.

The figure below shows the content of the "*Six_Step_Modulation*" block. It is an arithmetic block and is used to sequentially repeat the 6 logic states of ST52x301 (P0 .P1. P2) pins. The same decimal digits are reported in fig. 6.

The variable named "ciclo" is incremented at each turn of the software loop with a timing imposed by the timer interrupts with "*Wait0*" block.

The 'sixstep' variable is sent to the port by the "*Square_Wave_to_IOline*" block inside Timer interrupt routine.

Fig. 15 - Arithmetic block



Fuzzy Controller

In figure 16 is reported the content of the "*Fuzzy_controller*" block described in the main program. The Global Variable 'Error" initializes the local fuzzy input. The global variable range (0 : 255) is mapped into the (-10:+10) Universe of Discourse.

Five MemBership Functions cover the fuzzy input range in order to share the Universe of Discourse in 5 fuzzy subsets. Each subset is named in order to give semantic meaning to the variable values.

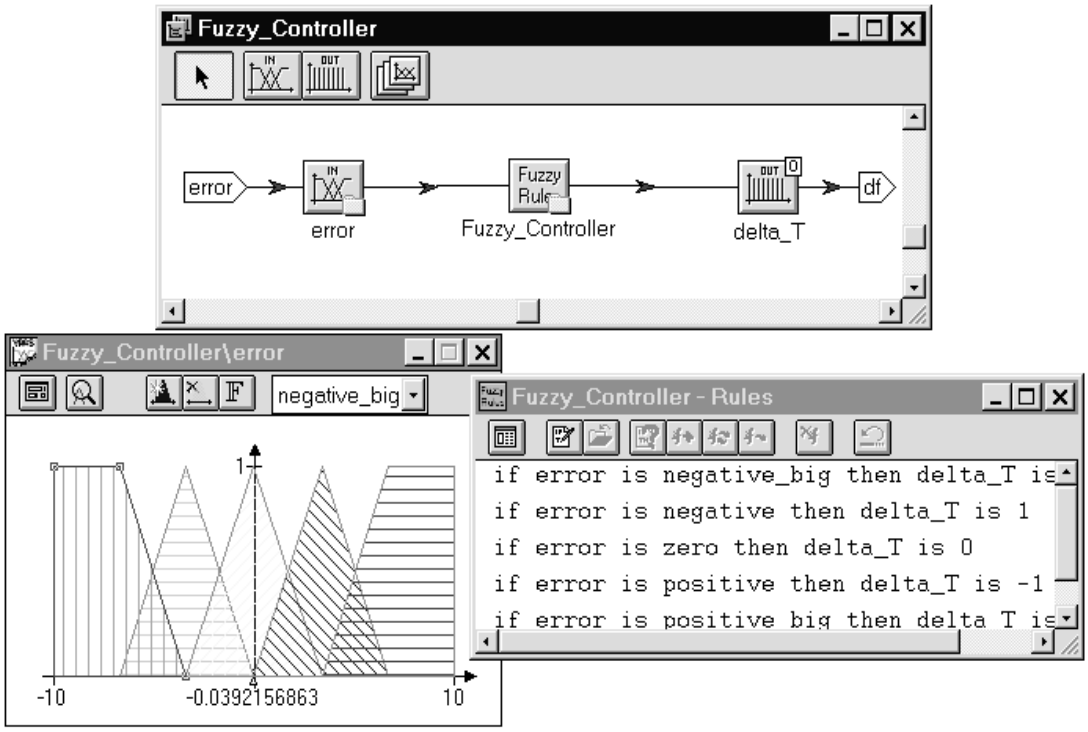
The first rule produces a strong increment of the stator frequency. In fact, 'error' is "negative_big" means that "speed >> Reference", then the controller must decelerate the motor shaft rotation. The way to slow down the speed is to decrease the stator frequency. In term of wave period, this leads to an increment of the timer counter register value (wave period increment).

The second rule performs the same action but with a lower action strength. In fact 'error' is "negative" involves a MemBership Function near to the zero error. This rule is activated when the shaft speed is close to the reference speed, so control action is more soft.

The third rule gives no correction to the stator frequency because it is activated when speed is equal to the reference. The same reasoning could be done for the other rules.

The output value of the "*Fuzzy_controller*" block is obtained with the contribution of all activated rules.

Fig . 16 - Fuzzy Controller Block



V/F Fuzzy Definition

This application implements a simple linear function between frequency and Voltage. By using fuzzy rules, it is possible to implement more complex non linear functions between the two variables. Furthermore, by using a fuzzy routine for the V/f definition, it is possible to avoid the use of a software division, that normally, in standard 8-bit microcontrollers, is time expensive and needs complex software routines.

The Global variable 'freq' initializes the fuzzy local variable 'Frequency'. Five MBFs have been chosen to cover the Universe of the Discourse (0 : 255). By using more MBFs, a very accurate model can be obtained if higher precision is required.

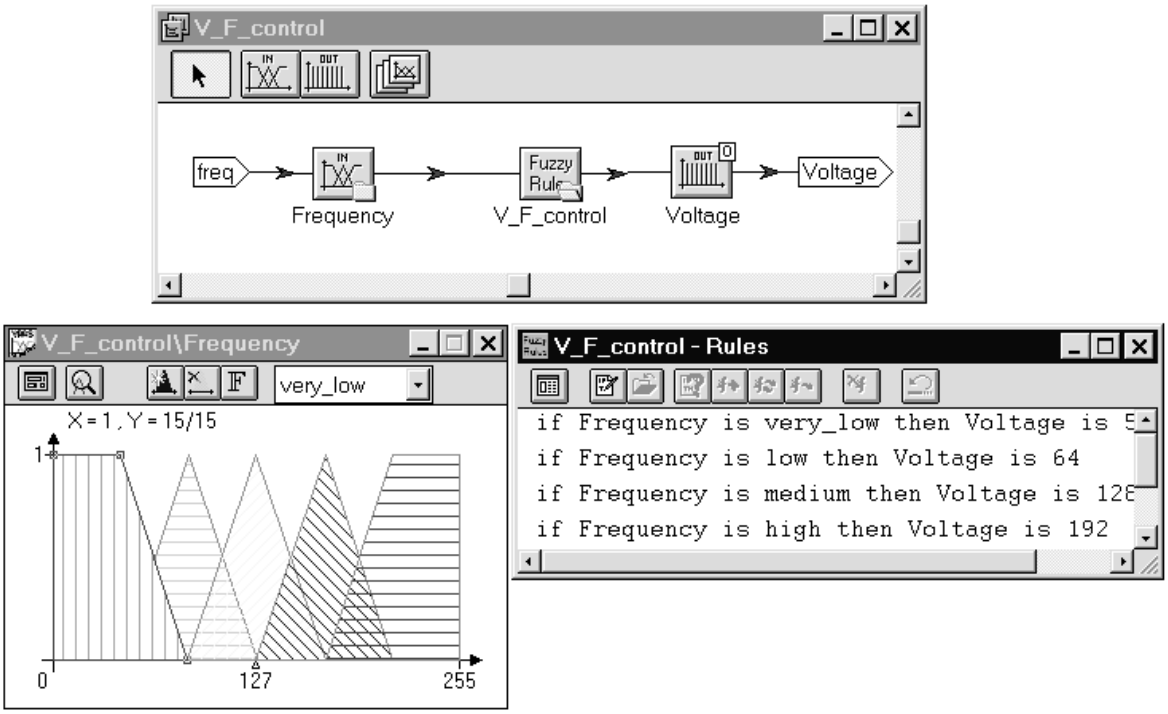
In order to obtain a linear variation of the ratio V/f, the rules have been defined by using experience and knowledge. Analogous results, or better, can be obtained by using fuzzy modeller software tools such as Adaptive Fuzzy Modeller (AFM) provided by STMicroelectronics.

Results and Conclusions

The AC 3-phase motor control described in this application note represents a good compromise between system costs and motor performances. The implemented six-step controller is good enough for a lot of consumer applications where the motor power and cost do not justify very complex control systems (Field Orientated Control and so on).

In this case, the graphical programming environment reduces the development time also for not expert programmers.

Fig . 17 - V/F model implementation



From figure18 it is possible to observe the system response. To evaluate acceleration characteristics and control goodness, some trials were made during the software development.

Fig.19 shows free acceleration characteristics starting from a quiet state of the shaft.

Although the implemented system is very simple the control performances do not degrade in comparison with other controls methodologies.

Fig. 18 Speed Reference following

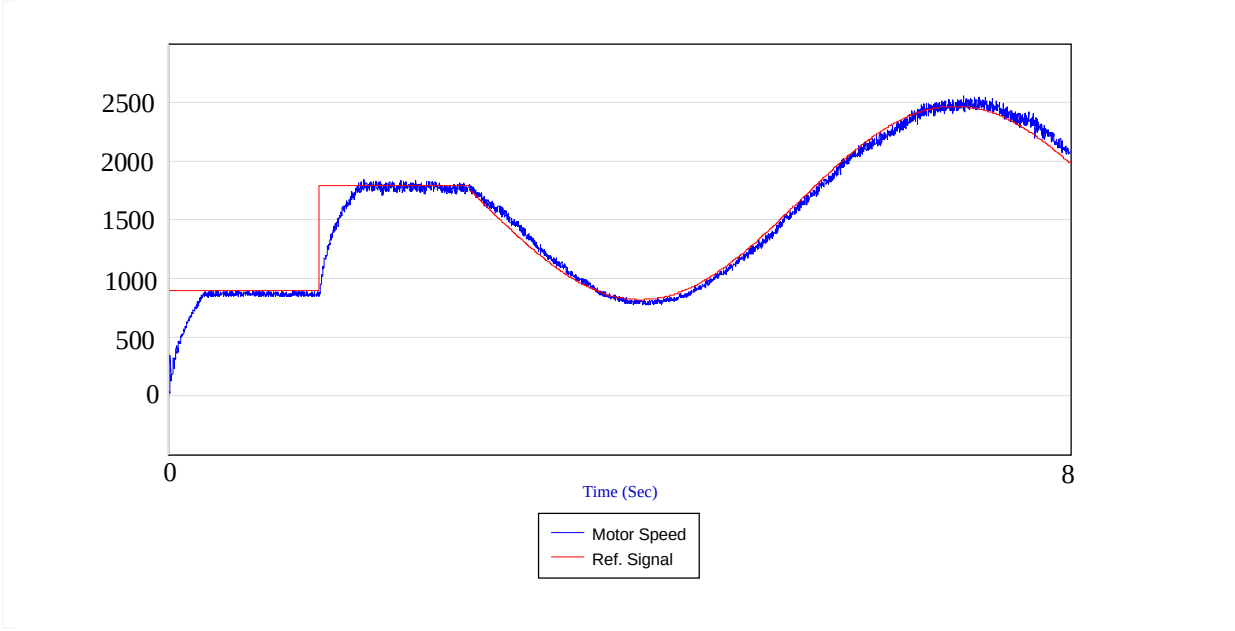
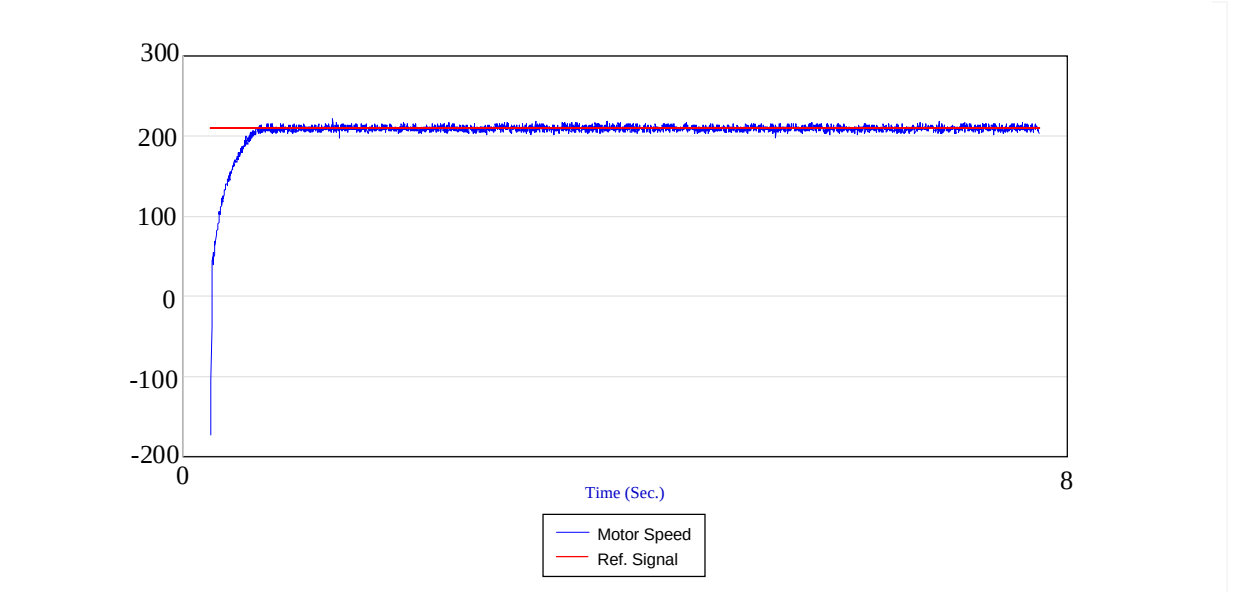


Fig.19 Step response



REFERENCES

[1] “Designer’s Guide to Power Products” - Application manual, STMicroelectronics
[2] Mohan, Undeland, Robbins “Power Electronics: Converters, Applications and Design” John WILEY & Sons
[3] Paul C. Krause "Analysis of Electric Machines", McGraw-Hill
[4] FUZZYSTUDIO™ 3.0 - User Manual, STMicroelectronics

Appendix: ST52x301 Assembler Code

```
;  Source file:          P:\APPLIC\ SIXSTEP\ACMOTOR.wcl
;  Compile time:         Tue Oct 06 11:09:14 1998
;  Device type:          ST52x301
;  Compiler version:     01.00 (02.06.98)

                data    0 0 42 213 0
                data    0 1 0 42 42
                data    0 2 44 171 42
                data    0 3 43 127 44
                data    0 4 42 84 43
                data    0 5 42 213 0
                data    0 6 0 42 42
                data    0 7 44 171 42
                data    0 8 43 127 44
                data    0 9 42 85 43
                stop
                irq     3       Timer_Interrupt
                stop
@WCLStart@@:
                ldcf    0       255
                ldcf    1       4
                ldcf    2       6
                ldcf    3       0
                ldcf    4       243
                ldcf5   1
                ldcf    6       40
                ldcf    7       12
                ldcf    8       25
                ldcf    9       0
                ldcf    10      4
                ldcf    11      0
                ldcf    12      64
                ldcf    13      0
                ldcf    14      0
                ldcf    15      228
Start:
Timer_Int_Setting:
                ldcf    14      8
Timer_Int_Priority:
                ldcf    15      210
```

```
Variables_Initialization:
                ldrc    15      0
                ldrc    13      0
                ldrc    14      1
                ldrc    9       0
Digital_Port_bit_set:
                ldpr    2       9
Voltage_level_setting:
                ldpr    1       15
Start_PWM:
                ldcf    11      2
                ldcf    10      7
Start_Timer:
                ldcf    6       41
                ldcf    6       43
Start_AD:
                ldcf    2       7
AD0_2_Reference:
                ldri    12      0
AD1_2_speed:
                ldri    10      1
Error_definition:
                mdgi
                ldrr    11      12
                subo    11      10
                megi
Delay:
Arth0:
                ldrc    7       0
Condition0:
                mdgi
                ldrc    0       255
                sub     0       7
                megi
                jpnz    @@00001
@00002:
                jp      Arth1
                jp      @@00003
@00001:
                jp      Return0
@00003:
Arth1:
                mdgi
```

```
                ldrc    0       1
                add     7       0
                megi
Return0:
@00000:
Fuzzy_Controller:
                ldrr    0       11
                stop
                ldp     0       1
                ldp     0       1
                fzand
                con     130
                ldp     0       4
                ldp     0       4
                fzand
                con     128
                ldp     0       3
                ldp     0       3
                fzand
                con     127
                ldp     0       2
                ldp     0       2
                fzand
                con     126
                ldp     0       0
                ldp     0       0
                fzand
                con     124
                out     0
                stop
                ldri    8       9
Integrator:
                mdgi
                ldrc    0       128
                add     13      8
                add     13      0
                megi
                mdgi
                ldrc    0       250
                sub     0       13
                megi
                jpz     @@00005
                jpns    @@00004
```

```
@00005:
                ldrc    13      250
@00004:
@00006:
V_F_control:
                ldrr    0       13
                stop
                ldp     0       6
                ldp     0       6
                fzand
                con     50
                ldp     0       9
                ldp     0       9
                fzand
                con     64
                ldp     0       8
                ldp     0       8
                fzand
                con     128
                ldp     0       7
                ldp     0       7
                fzand
                con     192
                ldp     0       5
                ldp     0       5
                fzand
                con     250
                out     0
                stop
                ldri    15      9
Fuzzy_out0_2_Voltage:
                ldri    15      9
Voltage_2_PWM:
                ldpr    1       15
Freq_2_Timer:
                ldpr    0       13
Six_Step_Modulation:
                mdgi
                ldrc    0       1
                sub     0       14
                megi
                jpnz    @@00007
@00008:
                ldrc    9       5
```

```
@00007:
@00009:
        mdgi
        ldrc    0       2
        sub     0       14
        megi
        jpnz    @@00010
@00011:
        ldrc    9       1
@00010:
@00012:
        mdgi
        ldrc    0       3
        sub     0       14
        megi
        jpnz    @@00013
@00014:
        ldrc    9       3
@00013:
@00015:
        mdgi
        ldrc    0       4
        sub     0       14
        megi
        jpnz    @@00016
@00017:
        ldrc    9       2
@00016:
@00018:
        mdgi
        ldrc    0       5
        sub     0       14
        megi
        jpnz    @@00019
@00020:
        ldrc    9       6
@00019:
@00021:
        mdgi
```

```
                ldrc    0       6
                sub     0       14
                megi
                jpnz    @@00022
@00023:
                ldrc    9       4
                ldrc    14      1
                jp      @@00024
@00022:
                mdgi
                ldrc    0       1
                add     14      0
                megi
@00024:
Wait0:
                waiti
                jp      AD0_2_Reference
Timer_Interrupt:
Square_Wave_to_IOline:
                ldpr    2       9
IRET0:
                reti
                stop
```