



ST52x301 FOR STEPPER MOTORS FUZZY CONTROL

Authors: G. Grasso, M. Lo Presti

1. INTRODUCTION

This application note is intended to explain how to control a stepper motor by using ST52x301 Fuzzy Microcontroller.

Stepper Motors are well known in position controls field applications and are particularly used in Robotics, Computer peripherals, Industrial servo quality drivers and so on. One of the main advantages of stepper motors is the strong relation between electrical pulses and rotation discrete angle steps. This allows to exactly know the shaft position of the motor without using a position sensor.

Depending on the building characteristics of the motor and its driving techniques, it is possible to have a wide range of stepper motors. Generally speaking, permanent magnetic cores are referred to as Stepper Motor while soft iron cores are referred to as Variable Reluctance Motors.

2. FUNCTIONAL DESCRIPTION

The Stepper Motor's operating principle used in the current implementation, is described in fig. 1. A 12-pole stator is energized by means of a 3-phase winding. Each coil is assembled in order to generate, in the space, the same number of magnetic poles. The rotor consists in a permanent magnetic core shaded in order to produce 8 magnetic poles.

The poles produced by the stator current force the rotor to move in order to be aligned to the rotor magnetic field. This implies an alignment of the rotor core's nearest pole to the stator energized poles.

When the phase $\Phi 1$ is energized, the rotor pole P1 is aligned to the field. Then, if the phase $\Phi 1$ is de-energized and $\Phi 2$ is energized, a new set of magnetic poles will be created in the stator. This new magnetic field will force the rotor to move in counterclockwise direction to align its poles to the field. This action yields a shaft rotation of a step angle q .

Generally, for a q -phase motor with N_r rotor teeth, the following equation is given:

$$Step_angle = \frac{360^\circ}{qN_r}$$

Then, in our application:

$$15^\circ = \frac{360^\circ}{3 \times 8}$$

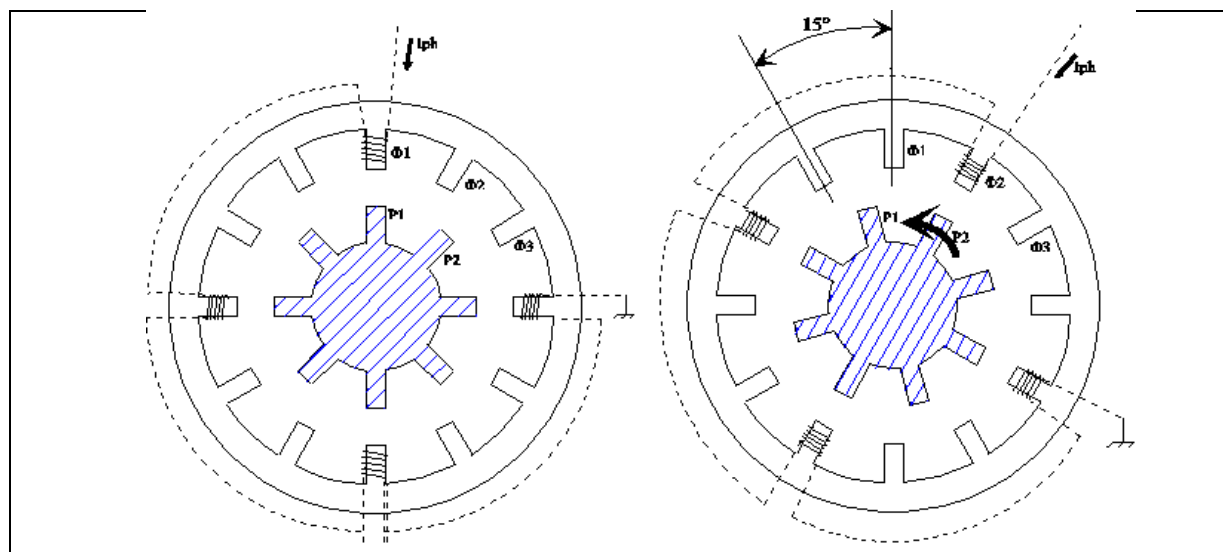
The spin direction can be controlled by a phase excitation sequence. Excitation sequence $\Phi 1$, $\Phi 3$ and $\Phi 2$ will move the rotor clockwise.

From energy considerations, it can be shown that:

$$T = f\left(i, \frac{dL}{d\theta}\right)$$

This general relation is true for any kind of stepper motor. For instance, the strong relation between T and $dL/d\theta$ allows the Variable Reluctance motor to move.

Figure 1

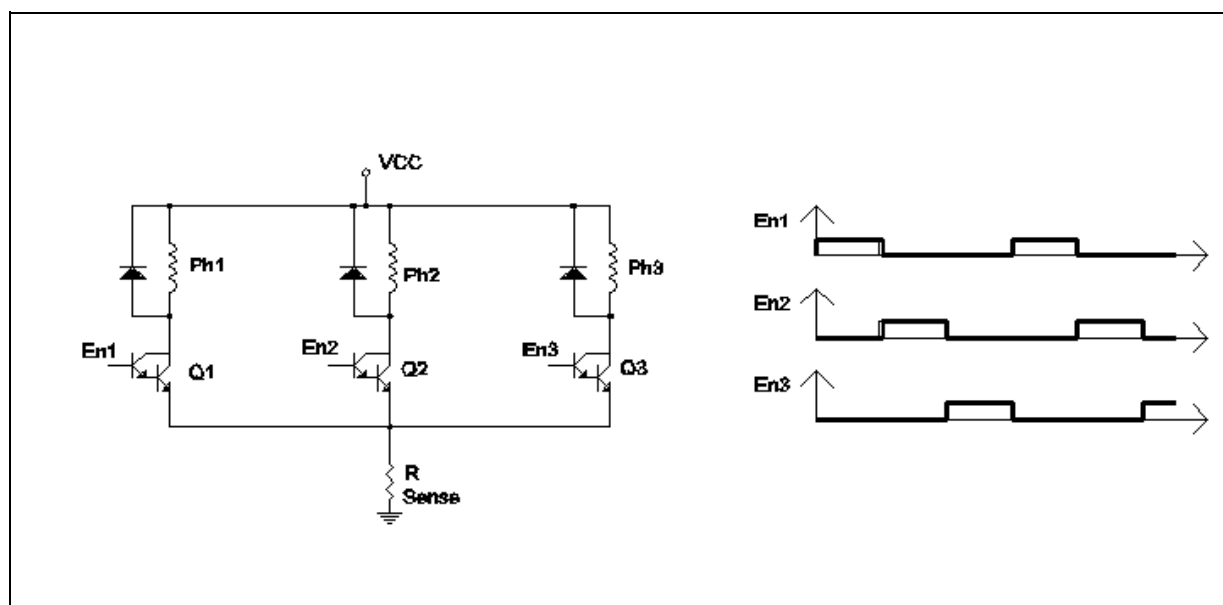


3. DRIVING TOPOLOGY

A variety of configurations is possible to energize the winding phase in a better way, nevertheless, the simplest driving topology will be investigated here.

Due to the internal connections of our motor, a unipolar driver is used. Fig.2 shows the basic topology used to energize the three phases of the motor.

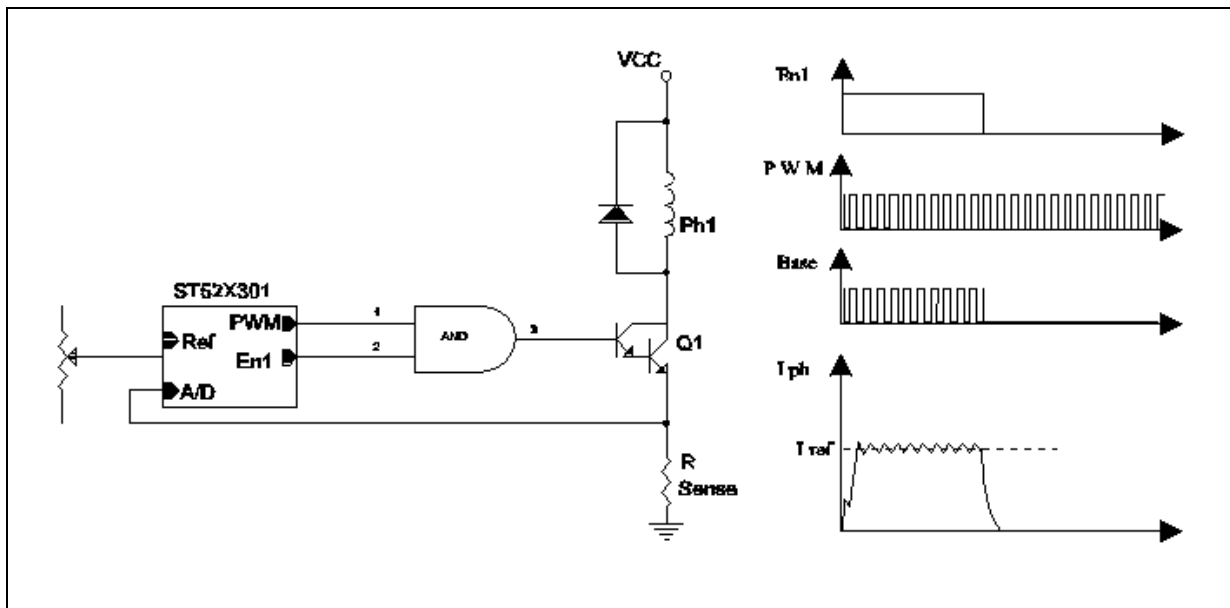
Figure 2



Motor specifications impose a maximum voltage of 24V for each phase and a phase current of 0.5A then, there are no problems to use Darlington BJT as switches.

When the phase Ph1 is energized (Q1 on), the voltage V_{CC} is applied to the phase coil; then the current raises with a rise-time related to the inductance value L and to the back-EMF of the coil. Roughly, the maximum value for the current is limited by a winding resistance unless the controller avoids the current to go above a fixed level. To perform this current limitation it is possible to “sense” the current level in the phase and reduce the applied voltage to the phase coil. A Pulse Width Modulation is the simplest way to control the mean value of the applied voltage. Fig. 3 shows the basic principle of modulation.

Figure 3



During the powering-time of each phase, ST52x301 can modulate T_{on} and T_{off} of the PWM signal in order to increase or decrease the mean applied voltage.

As we will see later, the AND gates can be replaced by a software “AND” inside the main program of ST52x301.

4. CURRENT CONTROL BY USING ST52X301 TRIAC-PERIPHERAL

ST52x301 provides up to 4 Analog inputs in the range between 0 and 2,5V with a conversion time of $33\mu s$ each.

Converted values are stored in 8-bit internal registers until the following conversion. Due to the input range, ADC resolution is 10 mV. Choosing the appropriate value for R_{sense} it is possible to read all the working current of each phase winding.

Moreover, ST52x301 provides a Triac-Driver peripheral to manage directly a Triac in several working modes. For example, In PWM mode, the user only needs to fix the PWM frequency and the duty-cycle of the wave. For instance, we can fix a constant PWM frequency according to the maximum switching frequency of the Darlington in order to maximize switches efficiency. After that, we can control the duty-cycle of the PWM wave to achieve the desired voltage on the phase coil.

Fig. 4 - 5 display the settings for the AD Converter and the Triac peripheral in the FUZZYSTUDIO™ 3.0 environment.

Figure 4

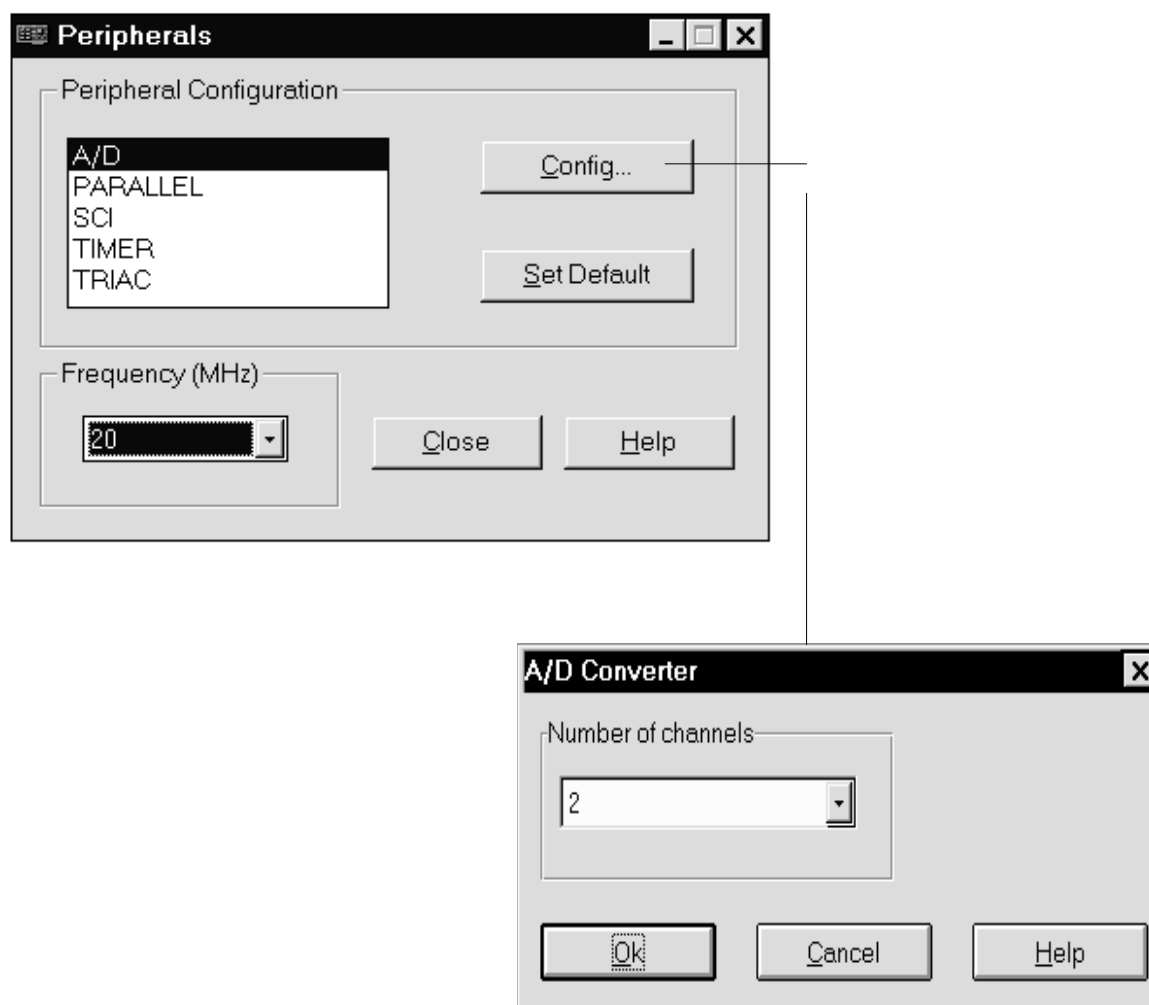


Figure 5

The image shows a software window titled "Triac Driver" with a standard Windows-style title bar (minimize, maximize, close buttons). The window contains several configuration sections:

- Frequency:** A text box showing "20" and the unit "MHz".
- Prescaler Setting:** Two text boxes. The first is labeled "Value" and contains "22". The second is labeled "Control Period (μs)" and contains "294.4".
- Output Polarity:** Two radio buttons. The first is labeled "Positive" and is selected (indicated by a filled circle). The second is labeled "Negative" and is unselected (indicated by an empty circle). Waveform icons are shown next to each label.
- Mode:** Three radio buttons. "PWM" is selected. "Burst" and "Phase Part." are unselected.
- Interrupt Source:** Two checked checkboxes. "Counter Out Rising Edge" and "Counter Out Falling Edge" are both selected.
- Counter loading from:** Three radio buttons. "Register" is selected. "Fuzzy Output 0" and "Fuzzy Output 1" are unselected.
- Clock Source:** Three radio buttons. "Internal" is selected. "External from MAIN1 pin" and "External from Power Line" are unselected.
- MAIN2 Pin Setting:** Two radio buttons. "Input/Tristate" is selected. "Output" is unselected.

At the bottom of the window are three buttons: "Ok", "Cancel", and "Help".

5. SPEED SEQUENCE BY USING ST52X301 TIMER-PERIPHERAL

The speed of the motor shaft can be varied by controlling the excitation time of the phases. To do this, it is possible to use TIMER-peripheral of ST52x301.

Timer settings are shown in fig. 6. The 16-bit Prescaler of the Timer is configured to give an internal clock pulse of 1ms. Then, the Timer period can vary between 1ms and 1ms x 256 according to the Timer Counter value.

Figure 6

The image shows a 'Timer Configuration' dialog box with the following settings:

- Frequency:** 20 MHz
- Start/Stop:** Internal (selected), Level (selected)
- Prescaler Out Period:** 19999 Value, 1000 μ s
- Counter loading from:** Fuzzy Output 0 (selected), Fuzzy Output 1, Register
- Output Polarity:** High (selected), Low
- Clock Source:** Internal (selected), External
- TIMEROUT Wave Form:** Two waveforms are shown, with the first one selected (indicated by a radio button).
- Interrupt Signal On:** Counter Stop (unchecked), TIMEROUT Rising Edge (checked), TIMEROUT Falling Edge (checked)

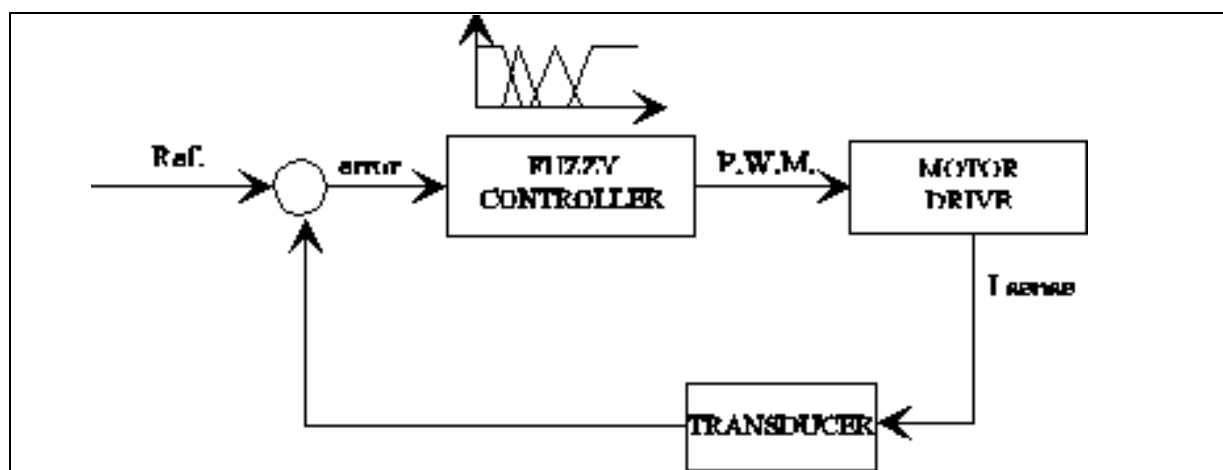
Buttons at the bottom: Ok, Cancel, Help.

To switch among the three phases, we can use the Timer Interrupts provided by the peripheral each time the Timer has completed a count.

6. FUZZY CONTROL

The goal of our Fuzzy Control is to maintain the desired Torque regardless to the applied load to the shaft. As observed in the equation (3), Torque is related to the windings current, then we can use current information on Rsense to manage the provided Torque.

Figure 7

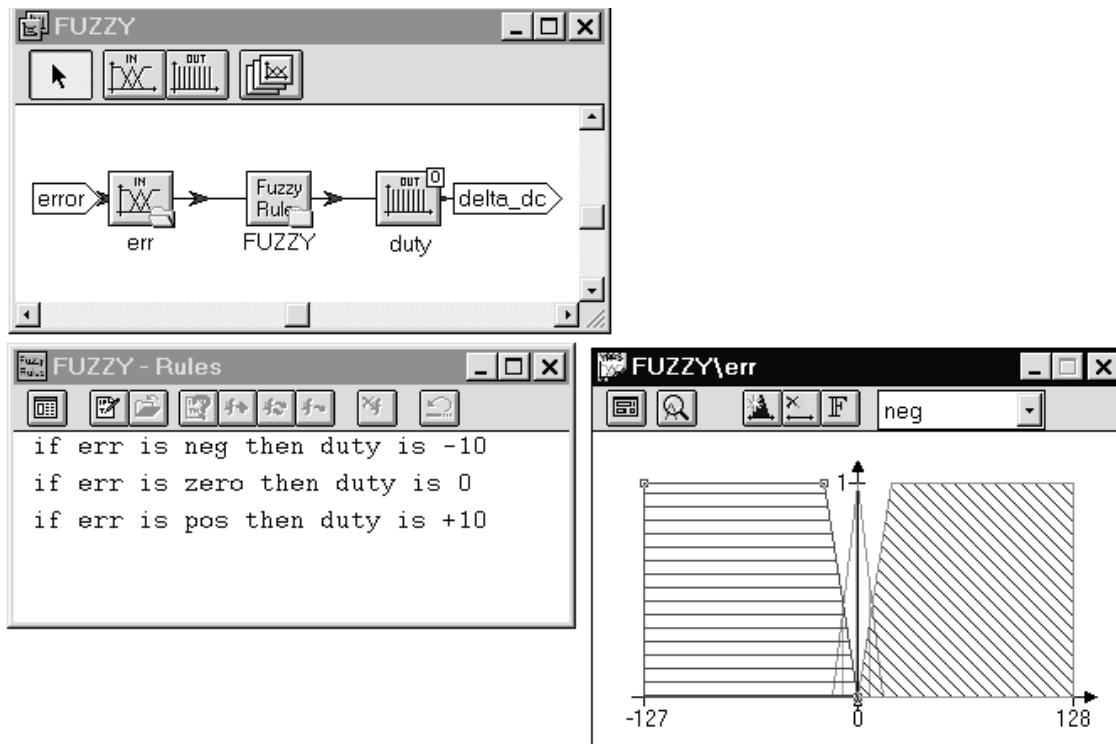


To carry out this control ST52x301 reads Torque “Ref” value from AD Channel0 and the instantaneous Torque supplied to the motor by means of Rsense and AD Channel 2. A software task performs the “error” calculation

$$error = VREF - Vtorque$$

“error” variable forms also the Fuzzy input for the Fuzzy controller block. We chose to cover the “error” Universe of Discourse by means of three Membership Functions. The Fuzzy algorithm uses three rules to compute the fuzzy out achieving an incremental variable for PWM duty-cycle. This incremental value, added to actual duty-cycle value, will be sent to Triac counter to change the Ton and Toff of the wave.

Figure 8



By observing the rules, it is possible to understand the PID strategy. If error is negative (i.e. Torque \ll sense), shaft is forcing with more torque than requested. This implies a useless current into the windings. The action to perform, is to reduce the applied voltage to the winding; then Ton on PWM wave must decreased. The first rule produces a negative Fuzzyout in case "error is neg". This negative Fuzzyout value will be added to the duty-cycle variable before sending it to the Triac Counter (duty-cycle=duty-cycle + Fuzzyout). The opposite action is performed by the third rule.

The second rule will be activated when torque \cong sense and will produce no variation on PWM duty-cycle.

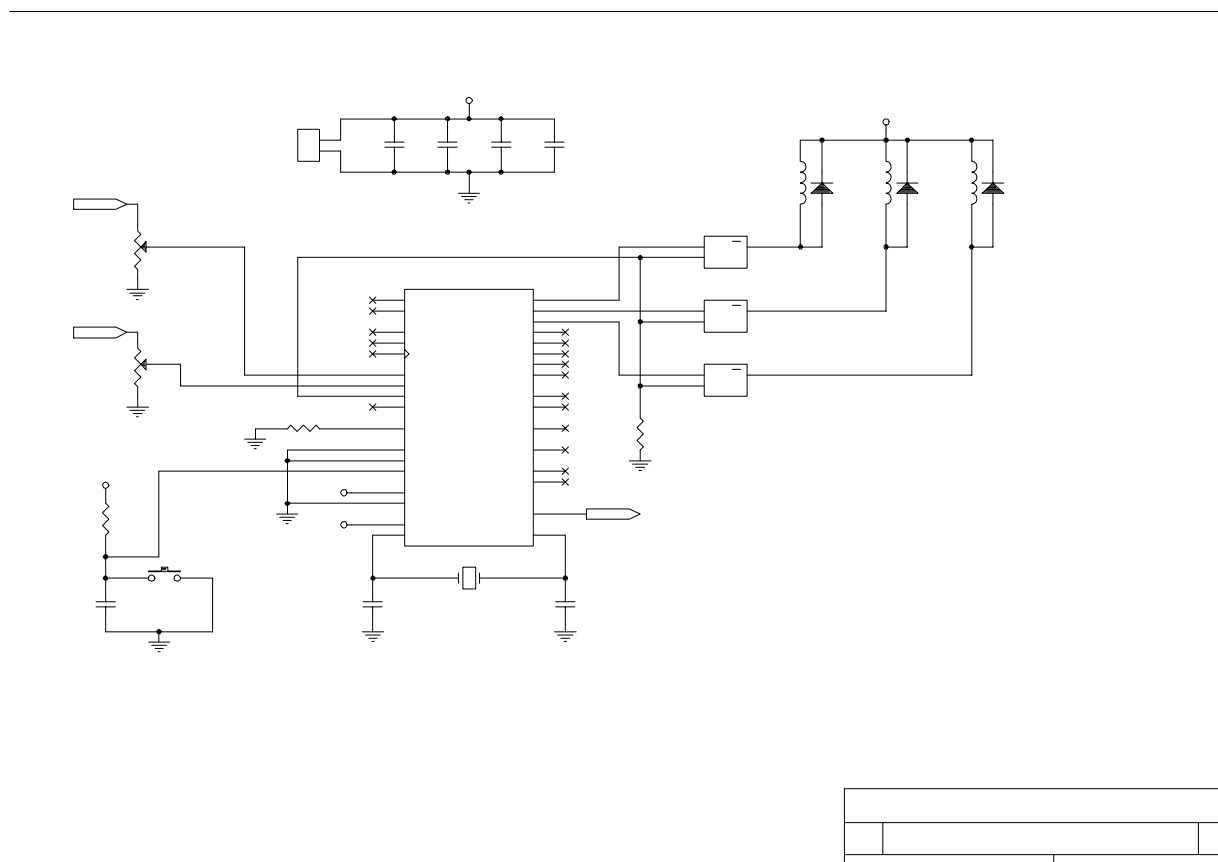
7. HARDWARE DESCRIPTION

The figure below shows the complete schematic used to implement the real system. ST52x301 and an integrated Darlington array are enough to drive the stepper motor. An internal software task is used instead of three external AND's to feed the PWM wave to the BJT.

A 20MHz oscillator provides the system clock whereas 4 capacitors provide a filtering of the power supply just near ST52x301 power supply pin's.

Linear integrate ULN2075 is a four Darlington array able to provide up to 1.5A for each transistor. This interface logic chip is designed to accepts TTL input levels.

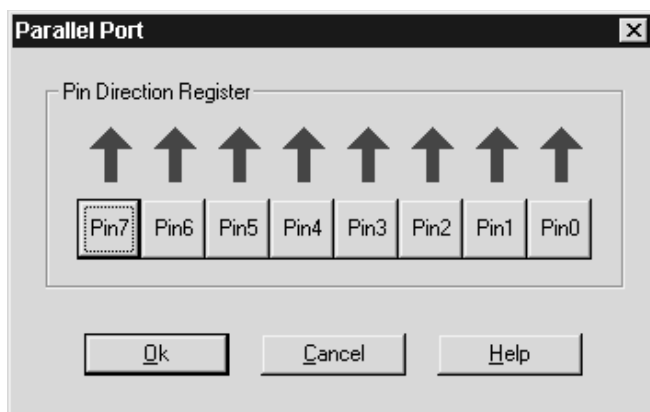
Figure 9



8. SOFTWARE DESCRIPTION

Let us describe the software tasks by means of pictures. First of all we need to initialize the ST52x301 peripherals and to define the variables. The figures below show the Parallel Port (OUT mode) and A/D Converter settings (2 input channels).

Figure 10



With this configurations, ST52x301 will convert sequentially 3 analog inputs in about 66 μ s and will give 8 output lines with 5 μ A maximum current.

Figure 11

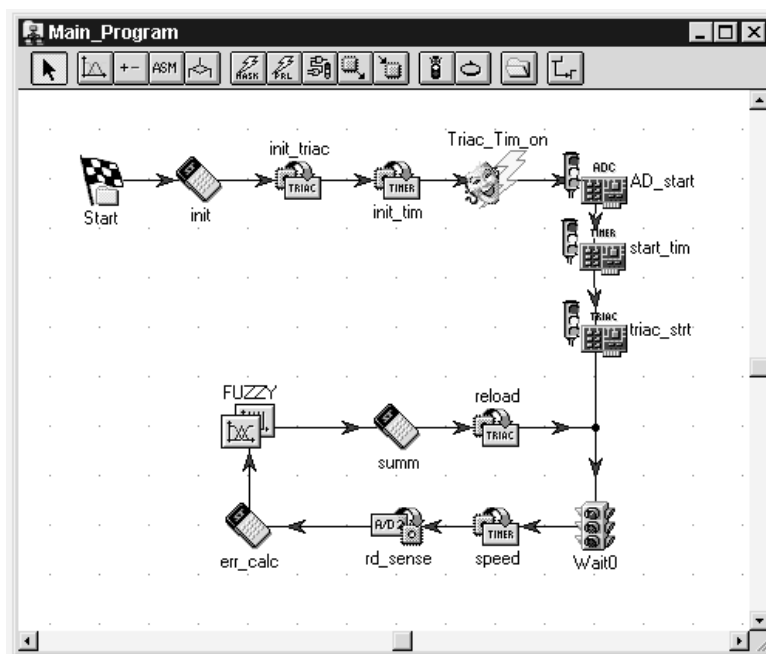
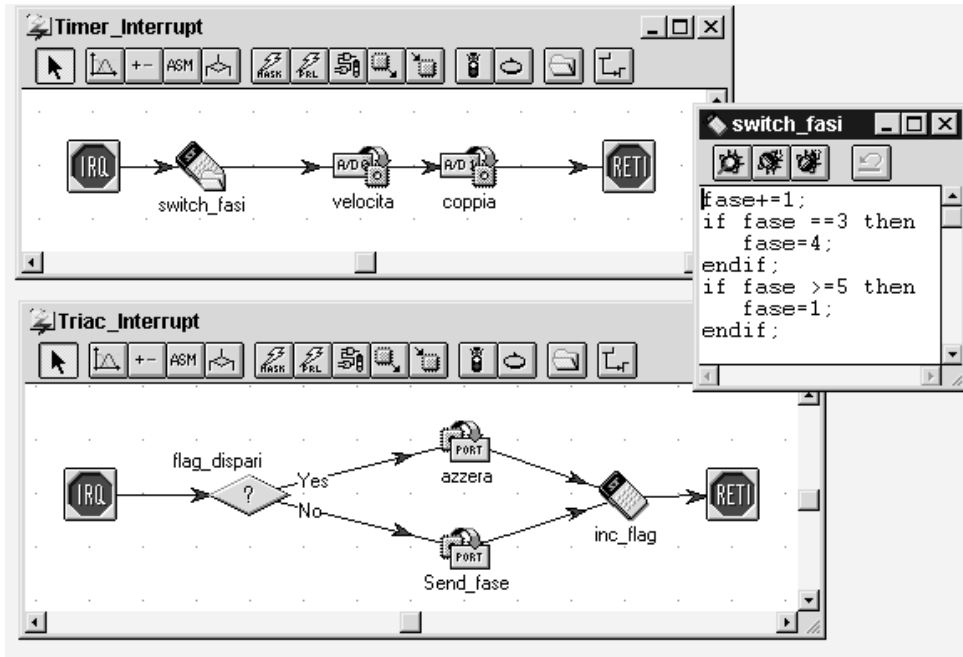


Fig. 11 shows the main program. After an initialization block for the variables, Triac and Timer counter values are setted.

A Mask block named "TriaC_Tim_on" enables these peripherals to generate Interrupts.

Three “start” blocks allow ADC, Timer and Triac peripheral to work. After that, a software loop form the main routine. In this loop, ST52x301 waits for an interrupt (Timer or Triac INT) and an INT coming from the program counter goes ahead to the “Speed” and “Torque” blocks.

Figure 12



In these blocks Timer and Triac counter will be refreshed with the new value.

The main point of the software program is the INT subroutines. The following figure shows the phase generation by means of interrupt routines.

In the Timer Interrupt sub-routine, the first block switches phases among three different variables' values. The variable “phase” changes among “1”, “2”, “4”. These values will be sent to the parallel port. With Timer Prescaler settings, a switch occurs about one hundred milliseconds. The Parallel port will produce three sequential square waves on P0, P1, P2 pins.

In the Triac Interrupt sub-routine, a flag switches on each coming INT, between set and reset of PWM out. The Triac Interrupts routine is executed about every 150 µs.

RESULTS AND CONCLUSION

By using ST52x301 Fuzzy Controller it is easy to implement motor driving by using few components. Presented stepper motor control is good compromise between system costs and motor performances. The graphical programming environment reduces the development time also for not expert programmers.

This application note was intended to show how easy is the use of ST52x301 as Controller in the servo-drives. A lot of improvement could be done in the presented control by changing the software and fuzzy algorithm.

APPENDIX

; Source file: P:\APPLIC\GRASSOG\SRM\STEPPWM.wcl
; Compile time: Fri Sep 25 15:54:31 1998
; Device type: ST52x301
; Compiler version: 01.00 (02.06.98)

data 0 0 20 147 0

data 0 1 15 127 15

data 0 2 0 107 20

stop

irq 3 Timer_Interrupt

irq 4 Triac_Interrupt

irq 1 AD_Interrupt

irq 2 SCI_Interrupt

irq 0 External_Interrupt

stop

@@WCLStart@@:

ldcf 0 255

ldcf 1 4

ldcf 2 10

ldcf 3 0

ldcf 4 31

ldcf 5 78

ldcf 6 40

ldcf 7 14

ldcf 8 22

ldcf 9 0

ldcf 10 4

ldcf 11 192

ldcf 12 64

ldcf 13 0

ldcf 14 0

ldcf 15 228

Start:

init:

ldrc 13 0

ldrc 14 0

init_triac:

mdgi

ldrc 0 128

ldpr 1 0

megi

```

init_tim:
    mdgi
    ldrc      0      200
    ldpr      0      0
    megi
Triac_Tim_on:
    ldcf      14     24
AD_start:
    ldcf      2      11
start_tim:
    ldcf      6      41
    ldcf      6      43
triac_strt:
    ldcf      11     194
    ldcf      10     7
Wait0:
    waiti
speed:
    ldpr      0      12
rd_sense:
    ldri      10     2
err_calc:
    mdgi
    ldrr      8      11
    subo      8      10
    megi
FUZZY:
    ldrr      0      8
    stop
    ldp       0      2
    ldp       0      2
    fzand
    con       117
    ldp       0      1
    ldp       0      1
    fzand
    con       127
    ldp       0      0
    ldp       0      0
    fzand
    con       137
    out
    stop

```

```
ldri          7          9
summ:
  mdgi
  ldrc         0         128
  add          9          7
  add          9          0
  megj
reload:
  ldpr         1          9
  jp           Wait0
External_Interrupt:
fine_corsa:
IRET4:
  reti
AD_Interrupt:
IRET1:
  reti
SCI_Interrupt:
IRET3:
  reti
Timer_Interrupt:
switch_phase:
  mdgi
  ldrc         0          1
  add         13          0
  megj
  mdgi
  ldrc         0          3
  sub         0          13
  megj
  jpnz         @@00000
@@00001:
  ldrc         13          4
@@00000:
@@00002:
  mdgi
  ldrc         0          5
  sub         0          13
  megj
  jpz          @@00004
  jpns         @@00003
@@00004:
  ldrc         13          1
```

```

@@00003:
@@00005:
speed_REF:
    ldri        12        0
torque_REF:
    ldri        11        1
IRET0:
    reti
Triac_Interrupt:
flag_even:
    mdgi
    ldrc        0        1
    and        0        15
    meg
    jpnz        @@00007
    jp          @@00006
@@00007:
    jp          reset
    jp          @@00008
@@00006:
    jp          Send_phase
@@00008:
Send_phase:
    ldpr        2        13
inc_flag:
    mdgi
    ldrc        0        1
    add        15        0
    meg
IRET2:
    reti
reset:
    mdgi
    ldrc        0        0
    ldpr        2        0
    meg
    jp          inc_flag
    stop

```

REFERENCES

- [1] STMicroelectronics - "Designer's Guide to Power Products" - Application manual
- [2] Mohan, Undeland, Robbins "Power Electronics: Converters, Applications and Design"
John WILEY & Sons
- [3] STMicroelectronics - FUZZYSTUDIO™ 3.0 - User Manual

Information furnished is believed to be accurate and reliable. However, STMicroelectronics assumes no responsibility for the consequences of use of such information nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of STMicroelectronics. Specification mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. STMicroelectronics products are not authorized for use as critical components in life support devices or systems without express written approval of STMicroelectronics.

The ST logo is a registered trademark of STMicroelectronics

© 1999 STMicroelectronics – Printed in Italy – All Rights Reserved

STMicroelectronics GROUP OF COMPANIES

Australia - Brazil - Canada - China - France - Germany - Italy - Japan - Korea - Malaysia - Malta - Mexico - Morocco - The Netherlands - Singapore - Spain - Sweden - Switzerland - Taiwan - Thailand - United Kingdom - U.S.A.

<http://www.st.com>