DESIGNING A POSITION REGULATOR FOR AN ACTUATOR POWERED BY A CONTINUOUS CURRENT MOTOR USING THE PIC16F73 MICROCONTROLLER

Monica-Anca CHITA[1], Paul SCHIOPU[2]

The paper presents the design of a position regulator for an actuator powered by a continuous current motor using the PIC16F73 microcontroller. The system contains the electric drive motor, transducer (an incremental encoder), an impulse counter, the processor, the digital-to-analog converter, and the power amplifier.

The novelty is represented by the usage of the PIC16F73 microcontroller to design the position controller for an actuator powered by a DC motor, and the usage of the HS oscillator when designing.

Computer interfacing was performed in a Java development environment.

Keywords: the PIC16F73 microcontroller; the HS oscillator; the PWM (Pulse Width Modulation) module, position controller for a DC motor

1. Introduction

Advanced automation and miniaturization, available nowadays in all areas of engineering, require the continued development of a variety of safe and compact drives within the structure of modern systems.

Actuators are controllable execution elements that transform the input energy (electrical, magnetic, thermal, optical or chemical) into mechanical work. The conversion of the input energy into the useful output energy is achieved by means of magnetic fields, due to physical phenomena: the piezoelectric phenomenon, the magneto-strictive phenomenon, the memory storage phenomenon due to the expansion of the bodies to the temperature increase, the phase changes, the electro-rheological, electro-hydrodynamic, diamagnetic effect.

The actuator mechanism transforms, amplifies and transmits the movement by agreeing with parameters specific to the technological purpose.

Generally, actuator means a subassembly that converts a form of energy (electrical, pneumatic, thermal, chemical, etc.) into mechanical energy. Its structure can no longer be broken down into sub-structures except at the risk of

[1] Assoc Prof., Dept. of Electronics, Computers Science and Electrical Engineering, University of Pitesti, Romania, e-mail: chita_monica@yahoo.com

[2] Prof., Dept. of Dept.of Electronics Technology and Reliability, University POLITEHNICA of Bucharest, Romania, e-mail: paul.schiopu@yahoo.com

losing its capacity to generate motion.

The overall structure of an actuator is shown in Fig. 1.

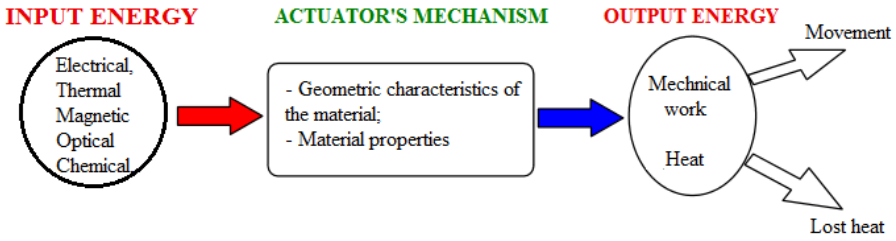

Fig. 1 Structura generală a unui actuator

2. Block diagram of the system

The block diagram of the system is shown in Fig. 2.

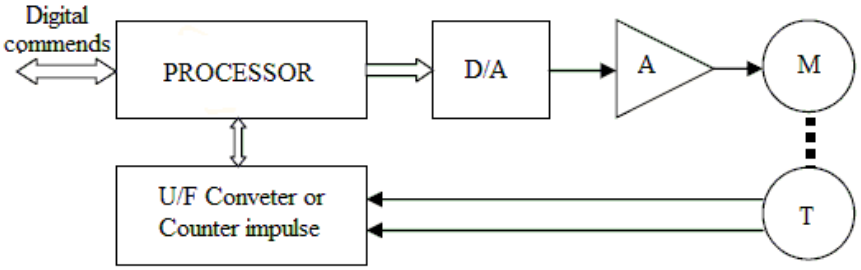

Fig.2 Block diagram of modern digital system: D / A - digital-analog converter; A - Power Amplifier; T - transducer; M - DC motor

The system contains the electric drive motor, the transducer (an incremental encoder), the impulse counter, the processor, the digital-to-analog converter, and the power amplifier.

Pulse Width Modulation (PWM) output, i.e. pulse duration modulation, of the microcontroller may be a signal that carries both the PWM information and the steering signal, or a single signal containing only the PWM information. In the first case, a PWM signal fill factor, that is equal to 50%, will produce a zero voltage at the output (the voltage at the motor terminals will be zero), a 0% fill factor will provide a maximum negative voltage at the output and for a 100% fill factor, and a maximum positive voltage will be obtained. In the second case, a PWM signal fill factor equal to 50% will produce a U/2 voltage at the output, a 0% fill factor will provide a zero voltage at the output, and for a 100% fill factor, a positive maximum voltage (U) will be obtained.

The encoder used (fig. 3) is directly integrated into the motor, it is of the incremental type in quadrature (it presents the signal A and B, B being 90 degrees ahead or back depending on the directifigureon of rotation) with 282 pulses per revolution. If it is used for triggering both signal A and signal B obtain a resolution of 564 pulses per revolution, i.e. 360/564 = 0.6383 degrees.

Fig.3 Example of encoder used

The controller works only in automatic mode, the adjustment function is of the PI type with the band of proportionality, integration time, and sampling time established by MatLAB simulations, with small adjustments on the real model.

3. The designing of the digital part of the circuit and the low power part

3.1 Presentation of the PIC16F73 microcotroller

The chosen microcontroller has the following features that have led to its choice: CPU RISC (Reduced Instruction Set Computing), i.e. it has a small set of instructions that allows it to work faster than an extended set controller; 35 single-word instructions; all instructions are executed in a single cycle, except those that cause leaps; operating frequency: 20MHz, 200ns / instruction; 8K x 14-bit FLASH memory RAM memory 368 x 8 bits; EEPROM memory 256 x 8 bits; maximum 14 sources of masked interruptions; 8-level hardware stack; direct, indirect and relative addressing modes; supply voltage from 2.0V to 5.5V; current debited / absorbed by 25mA / pin; low power consumption below 0.6mW at 3V and 4MHz; two timers, of which 8-bit TIMER0, 16-bit TIMER1, 8-bit TIMER2, each with a special function, two PWM outputs; A 10-bit A / D converter built-in RS232 serial port; DIP28 (Dual Inline Plastic, 40 pins) capsule.

3.2 Microcontroller power supply

Generally, correct power supply is of utmost importance for the correct operation of the microcontroller system. For proper operation of any microcontroller, it is necessary to provide a stable power supply, a safe reset when you turn it on and an oscillator.

3.3 Choosing the type of oscillator

The PIC16F73 can work with four different oscillator configurations. Because the crystal oscillator and resistor-capacitor (RC) configurations are the ones most commonly used, it is only them to be mentioned here.

The HS oscillator

The operating frequency of the 20MHz crystal oscillator (maximum catalogue frequency) was chosen. To design the oscillator, a ceramic capacitor of 22pF is required with the other end to the ground to be connected to each pin, as in fig. 4.

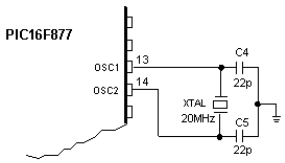

Fig. 4 Designing the HS Oscillator

3.4 The Reset

The reset is used to set the microcontroller in a 'known' condition. This basically means that the microcontroller may behave incorrectly under some undesirable conditions. To continue working properly, it must be reset, meaning all registers will be set in a start-up status. To prevent a logic zero from reaching the MCLR pin accidentally (the above line means that the reset is enabled by a logic zero), the MCLR must be connected via a resistor to the positive pole of the power supply. The resistor should be between 5 and 10k (Fig. 5). The PIC16F73 microcontroller has several reset sources: reset to power, POR (Power-On Reset); reset during normal work by bringing a logical zero to the MCLR pin of the microcontroller; reset during SLEEP mode; reset to the watchdog timer (WDT) exceeding; reset during WDT exceeding during SLEEP mode.

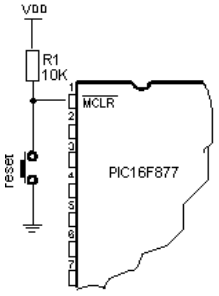

Fig .5 Example of reset to PIC 16F877 microcontroller

The most important reset resources are a) and b). The first occurs every time the microcontroller is powered and serves to bring all of the registers to the initial start position. The second is to bring a logical zero to the MCLR pin during the normal operation of the microcontroller. It is often used in program development.

3.5 Serial communication

SCI is an abbreviation for the Serial Communication Interface, and as a special subsystem, it is available for the majority of microcontrollers. At the PIC16F877 microcontroller this is hardware made.

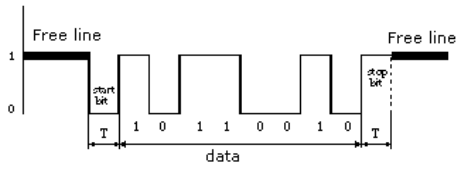

Fig. 6 Example of SCI designing at the PIC16F877 microcontroller

In the case of hardware communication, we use the standard NRZ (Non Return to Zero) format known as 8 (9) -N-1 or 8 or 9 bits of data, without parity and with a stop bit. The free line is defined as the one logical status. Beginning of the transmission - The Start Bit has a logical zero status. After the data bits following the start bit (the first bit is the least significant bit) there follows a Stop Bit that has logical one status. The length of the stop bit, "T", depends on the transmission speed and it is adjusted according to the transmission needs. For a transmission speed of 9600 baud, T is 104us.

To connect a microcontroller to a serial port on a PC, we need to adjust the signal level for the communication to take place. The signal level at a PC is -10V for logic zero and +10V for logic one. Because the signal level at a microcontroller is + 5V for logic one and 0V for logic zero, we need an intermediate stage to convert the levels. A specially designed integration for this task is MAX232. The diagram of the interface is detailed in fig. 7.

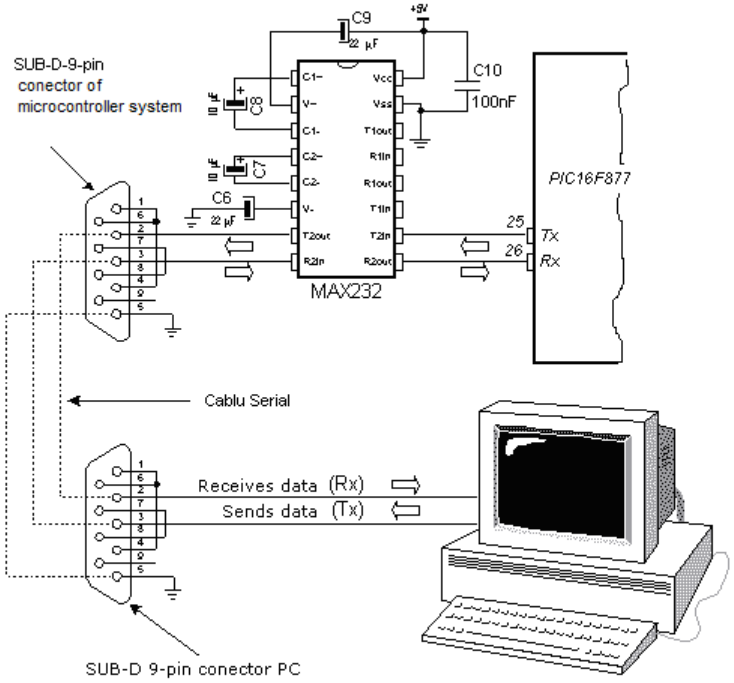

Fig. 7 Interfacing the microcontroller with the computer

3.6 The PWM mode

3.6.1 Calculating the minimum frequency of the PWM signal

The choice of the PWM signal frequency is made taking into account the electric time constant of the motor $\tau = \frac{L}{R}$. The calculation of the minimum frequency of the PWM signal is done at a filling factor of 0.5, as it follows in Fig. 8:

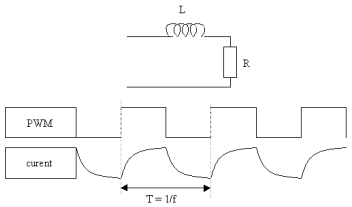

Fig. 8 The minimum frequency of the PWM signal is calculated at a fill factor of 0.5

At the time T = T/2 the current i_1 must not have a ripple greater than P% than at the time t = 0 (i_0). This results in the following condition:

$$i_{1.} = \left(1 - \frac{P}{100}\right) \cdot .i_0 \tag{3.1}$$

$$i = I \cdot e^{-\frac{t}{\tau}} = I \cdot e^{-\frac{tR}{L}} \tag{3.2}$$

so for the fill factor of 0.5 it results:

$$I \cdot e^{-\frac{tR}{L}} = (1 - \frac{P}{100}) \cdot Ie^{0} \Leftrightarrow e^{-\frac{tR}{L}} = 1 - \frac{P}{100} \Rightarrow T = -\frac{2L}{R} \cdot \ln(1 - \frac{P}{100}) \text{ or}$$

$$f = \frac{R}{-2l \cdot \ln\left(1 - \frac{P}{100}\right)} \tag{3.3}$$

For the parameters of the used motor (R = 1.8Ω and L = 8.5mH) and a 5% P ripple, the minimum frequency f = 6.89 kHz is obtained.

It was chosen the frequency of 19.53 kHz (maximum frequency of the PWM signag / 10-bit signal that can be obtained with the PIC16F877 microcontroller with the 20MHz clock) resulting in a 2% ripple at a fill factor of 0.5.

3.6.2 Programming the microcontroller for the given PWM frequency

In the Pulse Width Modulation - PWM - mode, the CCP1 pin produces a PWM output with a resolution of up to 10 bits.

The simplified scheme of this module is illustrated in the figure below.

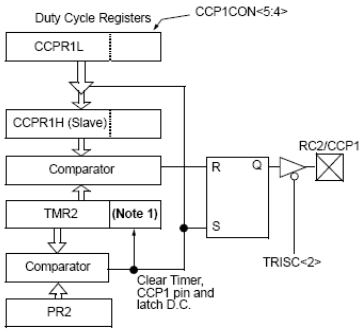

Fig. 9 The simplified scheme of the PWM module

The PWM output has a time base (period) and a fill factor, the period can be calculated with the relation:

$$PWM_{period} = \left[\left(PR2\right) + 1\right] \cdot 4 \cdot T_{OSC} \cdot \left(TMR2_{prescalervalue}\right) \tag{3.4}$$

The frequency F_{PWM} of 19.53 kHz was chosen, that is equal to the maximum frequency for this resolution, resulting in the following values: $TMR2_{prescaler}$ = 1 and PR2 register = 0xFFh;

To program the module, follow these steps: the PWM period is determined by entering the calculated value (0xFFh) in the PR2 register; the Duty Cycle PWM (filling factor) is set by writing the desired value in CCPR1L registers (bits 0:7 duty) and CCP1CON bits 4 and 5 (bits 8: 9 duty); et the CCP1 pin as output, resetting bit TRISC.2; set the $TMR2_{prescaler}$; the CCP1 module is configured in the PWM mode (write the bits 3:0 of the CCP1CON register with the value 11xx).

The wiring diagram resulting following the design is shown in Fig. 10.

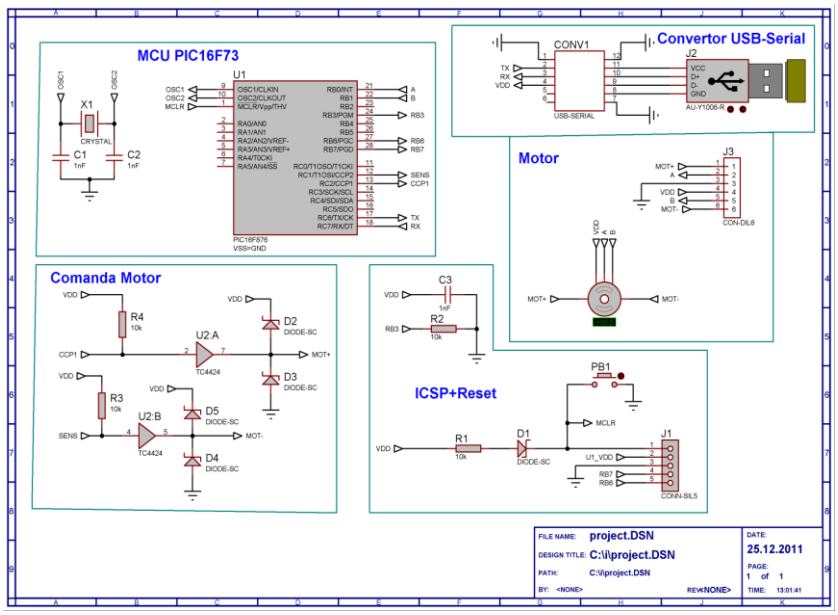

Fig. 10 The wiring diagram resulting following the design

3.7 Making the software

3.7.1 Designing the PID regulator

The PID continuous setting function is given by the 3.6 formula:

$$\begin{cases} G_{PID}(s) = K_P + \dfrac{K_1}{S} + K_D \cdot S \Rightarrow G_{PID}(z) = K_P + \dfrac{K_1 \cdot T_e}{2} \cdot \dfrac{z+1}{z-1} + K_D \cdot \dfrac{2}{T_e} \cdot \dfrac{z-1}{z+1} \\ \Rightarrow s = \dfrac{2}{T_e} \cdot \dfrac{z-1}{z+1} \end{cases} \tag{3.6}$$

for example, for $K_P = 100$, $K_I = 200$, $K_D = 10$, $T_e = 0.002$, the following unstable response of the system results, by processing in MatLAB:

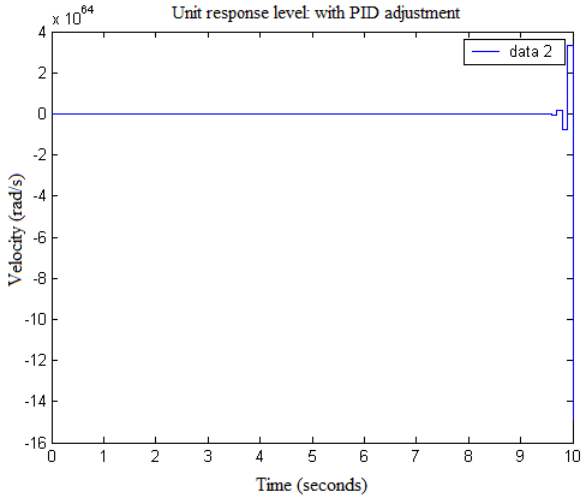

Fig. 11 Unit response level with PID adjustme

The program was written in C and compiled with compiler Boost C.

Computer interfacing was performed in a Java development environment, as it can be seen in Fig. 12:

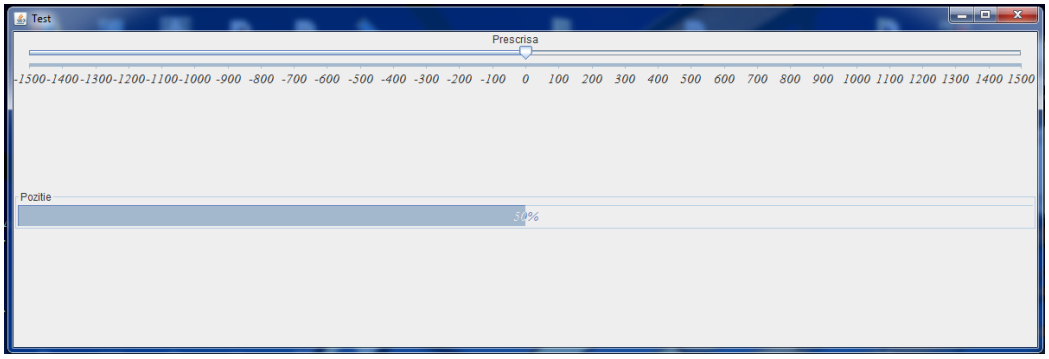

Fig. 12 Computer interfacing was performed in a Java development environment

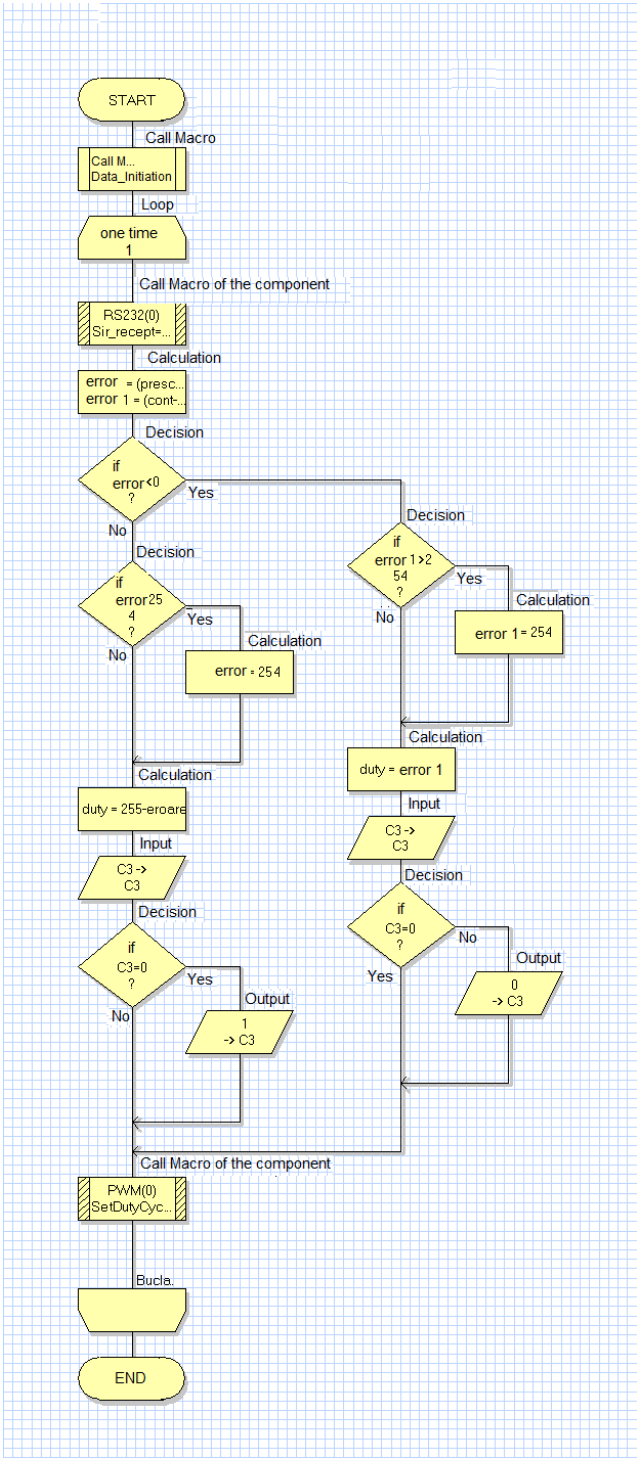

Fig.13 The main program

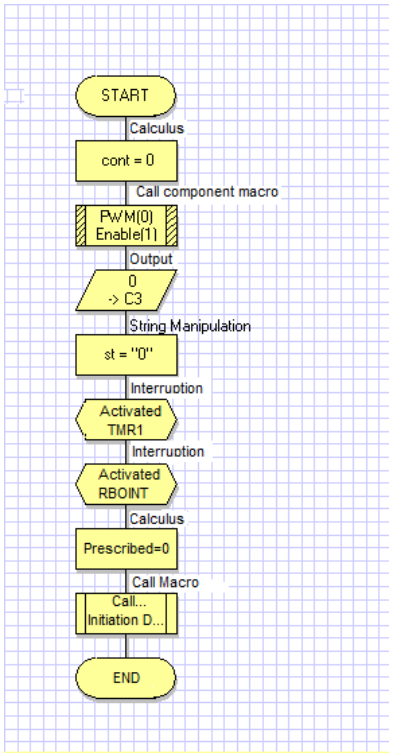

Fig. 14 Data initiation

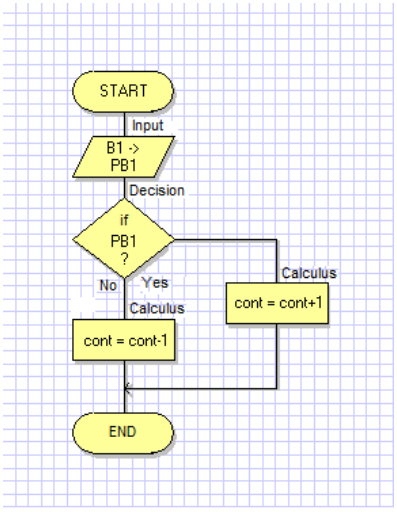

Fig. 15 Interruption routine INT0.

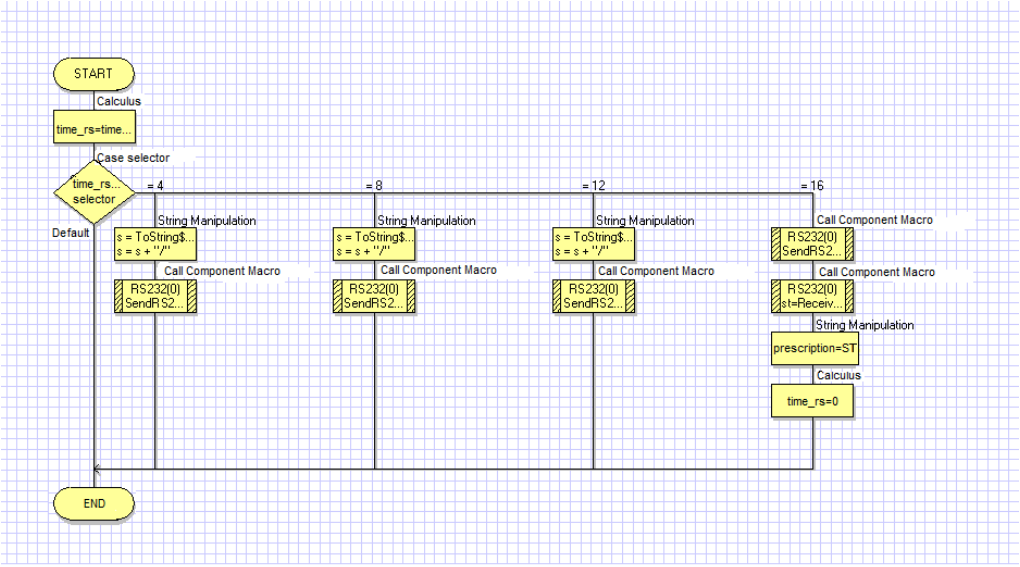

Fig. 16 Interruption routine TMR

4. Conclusions

The implementation of a position control system of a DC motor actuator with the help of microcontrollers allows for the adjustment of the position in a wide range of values, working at high frequencies, implementation of robust, stable positioning with an error of less than 1%.

The control system ensures high energy efficiency and a relatively low cost of the entire system, with practical implementation requiring a minimum of components.

Programming the PIC16F73 microcontroller by the user opens up numerous possibilities for adjusting the actuator position according to the inventiveness of each person implementing it.

R E F E R E N C E S

[1]. *M.A. Chita*, Senzori și actuatoare (Sensors and actuators), Editura MarixROM, București, 2017.

[2]. *Ioan Ciascai*, Aplicații industriale cu microcontrolere (Industrial applications with microcontrolers), Ability Thesis, Universitatea Tehnică din Cluj Napoca, 2015.

[3] *V. Bande,S. Pop, I. Ciascai,and D. Pitica* "Real-time sensor acquisition interfacing using MatLAB,, The 18th International Symposium for Design and Technology in Electronic Packaging (SIITME), October 2012, pg.189-192.

[4]. *S.K.Peddapelli*, Pulse Width Modulation:Analysis and Performance in Multilevel Inverter, Walter de Gruyter GmbH & Co KG Edition, 2017.

[5]. *E. Monmasson*, Power Electronic Converters: PWM Strategies and Current Control Technique, Willey Edition,2013.

[6]. *** Data Sheet PIC16F73 microcontroller, 2017.