Project Title

Intelligent management and control of electrical loads using microcontroller-based embedded systems

Zigkirkas Grigorios

May 2015

Thesis supervisor: Dr. John Kalomiros

Dissertation submitted in partial fulfilment for the degree of

Master of Science in *Communication & Information Systems*

Department of Informatics & Communications

TEI of Central Macedonia

Abstract

The design and implementation of an intelligent 3-phase, programmable soft-start controller, for low voltage induction motors is presented. The controller is implemented with a low-cost microcontroller and is based on a timer-driven, fuzzy-logic closed loop. A TRIAC-based switching circuit in series with the line voltage is used for current flow control during start-up. A timer-based embedded application adjusts the TRIAC triggering angle by shifting the TRIAC ignition pulse within the voltage period at a constant rate. Multitasking logic controls all three phases simultaneously. As a result, the voltage is gradually increased in the three-phase system, until the motor reaches full speed. An external user-defined setting can adjust the time frame from start to full speed. Certain improvements are introduced with regard to former similar designs, the most important being a closed-loop, voltage sensing, three-rule fuzzy system that effectively balances the inductive behavior and restores a predictable time-ramp at start-up. Because of the inductive character of three-phase asynchronous motors, the residual current after switch-off leads to higher nominal voltage across the motor coils, depending on the actual load. Therefore, it is found that an open-loop timer-driven switching logic is not adequate for a reliable soft start controller, since it can result in start-up time being unpredictably shorter than the one corresponding to user settings.

In order to adjust torque to load behavior and produce a constant ramp depending only on user-defined time settings, a closed-loop fuzzy-logic system is introduced. A voltage sensor acquires the residual voltage in one of the system's phase lines at adequate sampling rate and computes an input to a three-rule fuzzy system, in the form of an error function. The output of the system infers the time interval to the next ignition pulse. Both the timer logic and the fuzzy system logic are embedded in a low-cost PIC18F252 microcontroller. The behavior of the soft-start controller is smooth and predictable. The voltage ramp from zero to maximum voltage is very close to that corresponding to an equivalent ohmic load.

Attestation

I understand the nature of plagiarism, and I am aware of the University's policy on this.

I certify that this dissertation reports original work by me.

Signature **Date**

Acknowledgements

I would like to thank my supervisor Dr. John Kalomiros for his assistance during the elaboration of this thesis. I would also like to thank my wife and my children for their patience and support.

Table of Contents

List of Figures

1 Introduction

The subject of this thesis is the design an intelligent management and control embedded system for electrical loads, using a microcontroller. The project focuses mainly, in the development and construction of a low-cost soft starter for three-phase induction motors.

1.1 Background and Context

The project aims to design and implement a soft starter, especially for low voltage, asynchronous three-phase induction motors. It is known that most industrial electric drive applications depend on induction motors. In fact, induction motors are used at a rate of over 95%. Despite of their advantages (construction, maintenance, etc.) they also have serious disadvantages: during the direct start-up process, they absorb high current from the electric network. This, in turn, puts a strain on the motor itself, but also on the electric grid. The problem is addressed by gradually adjusting the voltage to the motor windings, until it takes its nominal value. This process results in a proportional change of the current in the motor. So, by inserting a soft-starter into an electric drive system, a smooth start takes place, according to the above elementary process.

The soft starter consists of power electronic devices (mainly triacs or thyristors) which aim to control the supply voltage. It also includes a control unit, which generates the appropriate ignition pulses of semiconductor elements and receives signals that are considered necessary for the proper operation of the device. In essence, this is a comprehensive real-time system, which understands the situation of the input-output voltage soft-starter and based on a "typical starting ramp", triggers the firing pulses to the appropriate angles of sinusoidal supply voltage.

1.2 Scope and Objectives

This thesis discusses the implementation of a low-voltage soft-starter product. The main purpose of a soft starter is to mitigate the high starting current absorbed by the motor during the startup process. To control the starting current a voltage regulation system is used based on TRIACs, which is inserted between the lines of motor and power supply. So by controlling the firing angle of the TRIACs, the output voltage of the soft starter is adjusted according to the time-frames set by user. However, because of the inductive character of 3 ~ asynchronous motors, the control of damping of TRIACs is lost from the system (because the current still flows in the circuit after the zero-cross voltage). As a result, the output voltage of the starter gets its nominal value faster than the start time given by the user. To address this phenomenon a fuzzy logic system is adopted, the purpose of which is to determine the next ignition pulse of

the TRIACs. The value of the residual voltage is input information for the fuzzy system and depends on the inductance of the motor (constructional characteristics, motor load, etc.).

1.3 Achievements

In this work, the design and implementation of an intelligent 3-phase, programmable soft-start controller for low voltage induction motors has been achieved. The design is implemented with a low-cost PIC18F252 microcontroller and is based on a timer-driven, fuzzy-logic closed loop. A TRIAC-based switching circuit in series with the line voltage is used for current flow control during start-up. A timer-based embedded application adjusts the TRIAC ignition angle by shifting the TRIAC triggering pulse within the voltage period at a constant rate with the use only a timer of the MCU. Multitasking logic controls all three phases simultaneously. As a result, the voltage is gradually increased in the three-phase system, until the motor reaches full speed. An external user-defined setting can adjust the time frame from start to full speed. Certain improvements are introduced with regard to former similar designs, the most important being a closed-loop, voltage sensing, three-rule fuzzy system that effectively balances the inductive behavior and restores a smooth and predictable time-ramp at start-up. The corrected voltage ramp from zero to maximum voltage achieved under inductive load is very close to that corresponding to ohmic load. Only three TRIAC semiconductors are used, simplifying the computational process for the production of ignition pulses and reducing both the computational as well as the raw hardware cost of the proposed system.

1.4 Overview of Dissertation

In Section 3, 4 and 5 a theoretical introduction, concerning 3 ~ induction motors, power electronics and soft starter is presented. In Section 6 the MCU PIC 18F252 is which is used to implement the soft starter is described. Section 7 illustrates the principle of operation of the soft starter. Also the multitasking techniques are presented. The operation of the starter under inductive load and the need to adopt the fuzzy logic in the system is illustrated. In Section 8 the soft starter circuit is presented and the functioning of individual circuits that constitute it is analysed. Finally, Section 9 presents the conclusions derived from the measurements performed in the soft starter.

2 State-of-The-Art

AC induction motors are widely used in a large range of home and industrial applications, like conveyors, pumps, fans, compressors, etc. The proliferation of AC induction motors is often limited by the difficulty to regulate their speed efficiently and accurately. It is especially required to control the transients produced because of the voltage dips in the supply and the high power absorbed at start-up.

This is usually achieved by appropriate adjustment of the motor terminal voltage. The same principle can be applied to vary the motor speed or to efficiently control the stop process. As a result, the electrical and mechanical stress on cables and motor shaft is reduced and the life span of the equipment is increased.

Soft-start methods are mainly divided into electromechanical starters and electronic starters.

2.1 Electromechanical starters

Electromechanical starters employ direct-online starting (DOL), star-delta switches, starting relays, resistors, inductors, transformers, etc. They are designed to reduce the high starting current [1]-[3], [14]. In the following paragraphs the main conventional starting methods are presented.

a) Start with star - delta switch

It is the most common way of starting 3 ~ induction motors. The stator winding should be made to work in delta (D) connection. When starting the motor, winding is connected in star (Y). After the engine has started and the speed reached a limit, it is coupled with a delta (D) connection via a switch. In this way, at first, the phase voltage at the coil is 230V (Y connection) and then becomes 400V (D connection) [1]-[3], [10], [14].

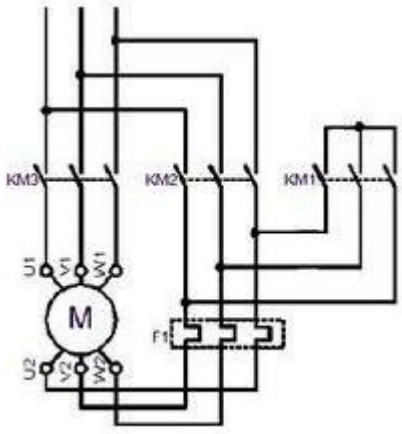


Figure 2.1 Connection of a star-delta switch

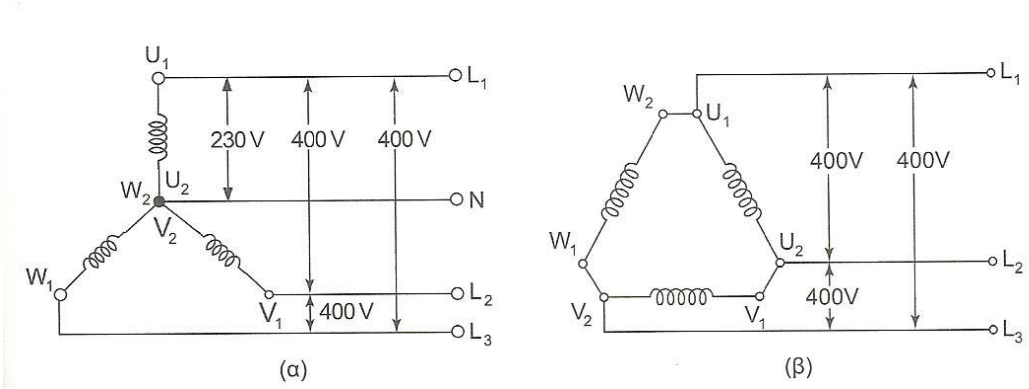


Figure 2.2 (α) Star connection, (β) Delta connection

b) Start with resistors in the stator winding

This method uses three resistors interposed in the supply circuit. The engine at startup starts with reduced voltage and thus with reduced current. Progressively as speed increases, resistance is removed, so that in normal operation all the resistors are cut out.

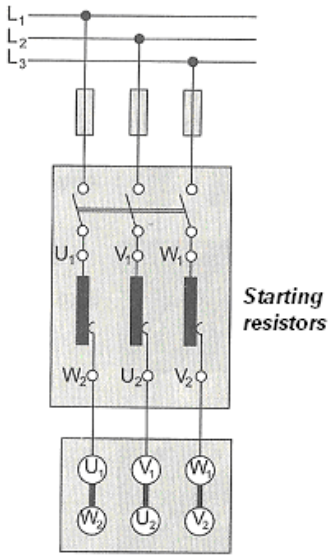


Figure 2.3 Connection starter resistors

c) Start with autotransformer

To start large 3~ induction motors three-phase autotransformers are commonly used. Initially the engine starts to 1/3 of the nominal voltage (depending on the number of downloads bearing the autotransformer) then applies ½ and successively all those values to the rated voltage. As in the above methods, the reduction of the supply voltage causes the reduction of the starting current. Additionally, the starting current in this case is limited due to the transfer function of autotransformer [1]-[3], [14].

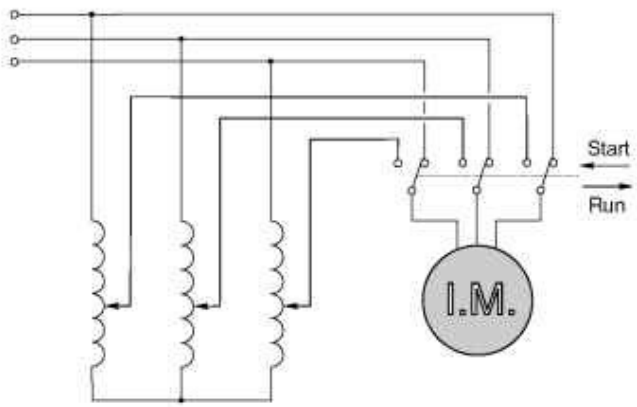


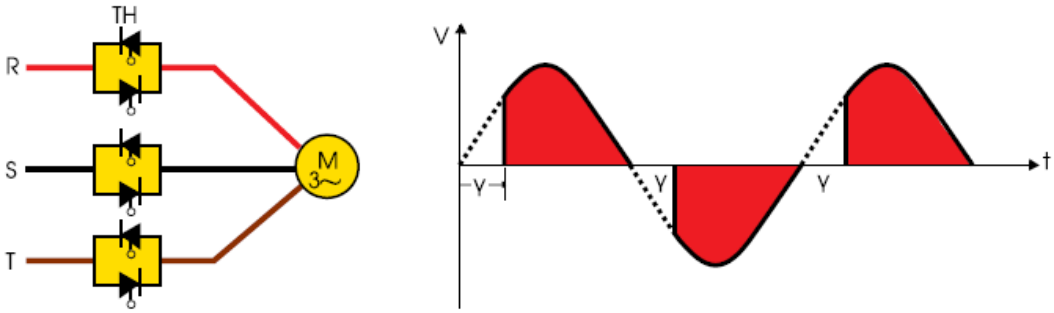
Figure 2.4 Starting with autotransformer

2.2 Electronic starters

The problem of the voltage dips, due to the high starting current is gradually eliminated by the design of new voltage regulators and soft-start controllers, appropriate for AC motor drives [4], [5], [13] including AC voltage controllers, Voltage / Frequency inverters or PWM converters [6]. The main purpose of a soft starter is to smoothly vary motor torque and in this way reduce the high current surge during the start-up process. Electronic AC voltage controllers employ solid state devices, like back-to-back connected thyristors (SCRs) or insulated gate bipolar transistors (IGBTs), etc. in order to implement the current controlling switching mechanism.

a) Starting with back-to-back connected thyristors (SCRs)

From the above-mentioned conventional methods, the basic principle for the start of a 3~ induction motor is to reduce the supply voltage. The same principle applies to modern electronic starters. The control modules are no longer switches, resistors, etc. but have been replaced by TRIACS and THYRISTORS. Their operation is very simple, because it is based on the ON - OFF control of elements with suitable pulses, resulting in a ratio of the input voltage appearing in the output [4], [5].



(a) (b)

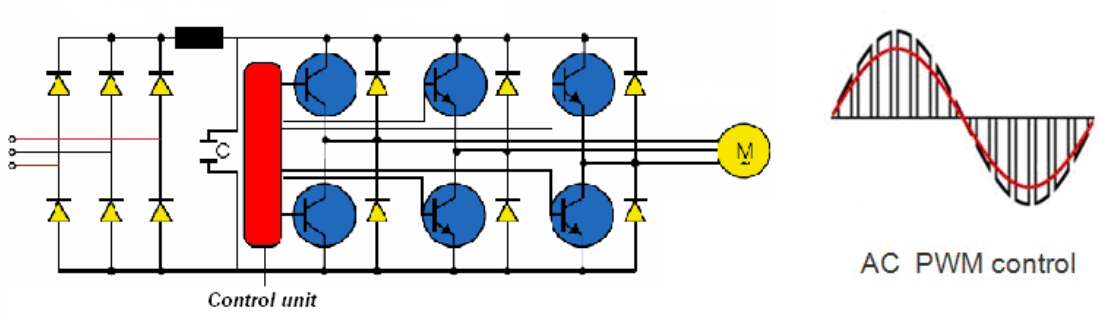
Figure 2.5 (a) Wiring connection with Thyristors. (b) The output of the starter

b) Starting with transistors

To control the starting of induction motors through transistors several techniques have been developed with converers, inverters, chopers etc. of which, the most important is the method of conversion of the voltage to frequency (V/f) and the other is the control of voltage output with pulse width modulation.

By keeping the ratio (V/f) constant, the starter acts as inverter, varying the voltage and frequency at the output of the system [13]. The result of this technique is gradual increase of the motor speed while adjusting the absorbed current (Fig. 2.6.a).

In another controller, same as in triac control, the voltage applied to the load is varied from zero to maximum value in a small span of time during start, with the use of a pulse width modulation technique (PWM) [13] as shown in Fig. 2.6 (b). The system directly modulates the mains a.c. voltage. The driver controls the voltage output via MOSFET, IGBTs or other clusters of transistos and the load is in series with a bridge rectifier.



(a) (b)

Figure 2.6 (a) Wiring connection of a starter with frequincy control (inverter)
(b) The output of the starter with ac PWM control (choper)

Today there is a trend in the literature towards intelligent, microcontroller-based soft-starters [7]-[11]. Many researchers employ artificial neural networks or fuzzy logic schemes, in order to generate the ignition angle of switching devices. Intelligent methods improve system stability and accuracy and correct for irregularities which are present in conventional methods.

3 Introduction to 3~ asynchronous motors

3.1 The electric three-phase AC motors

The electric motors are transducers that convert electrical energy into the mechanical and electromechanical energy used in various applications. The AC motors are divided into two categories:

- Synchronous electric motors and

- The asynchronous electric motors

The operation of both is based on the principle of the rotating magnetic field, but they have significant differences in their structure and in their function [14].

3.2 The principle of the rotating magnetic field

As mentioned above, the principle of operation of the synchronous and the asynchronous motor is based on the rotating magnetic field [14]. When powering the winding of the machinery it is observed that a rotating magnetic field relative to the axis of the machine is created. To explain the creation of the rotating magnetic field take two machines an alternator and the stator of a motor, at the most simplest form. From Fig. 3.1 we can see the time when the induced voltage takes its maximumvalue into the winding of a phase. For example, Phase 1-4 becomes maximum, the intensity of the current in this winding becomes maximum and thus the maximum magnetic field strength is induced in the winding. As illustrated, a bipolar magnetic field is created, which would create a fantastic natural magnet as it is designed into the motor stator. At the same time while the intensity of phase 1-4 is maximum, in the other two phases it has a value equal to ½ of the maximum in the opposite direction. The field is generated and amplifies the main magnetic field of the phase 1-4 without changing direction.

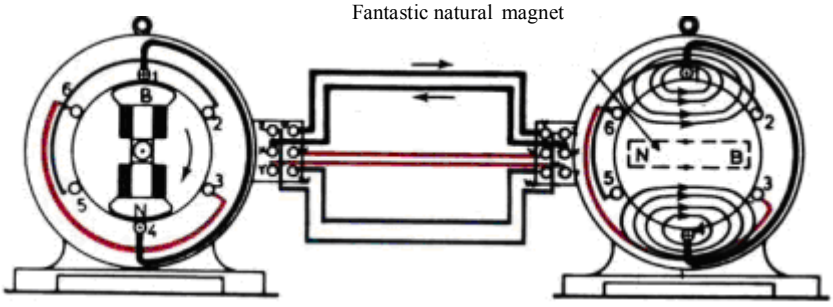


Figure 3.1 Creation a rotating magnetic field in pair alternator (generator) – Motor.

The same goes for phases 3-6 and 2-5, they are maximum at times 2 and 3 respectively. So a complete rotation of the rotor of the alternator generates a full rotation of the magnetic field within the motor stator. This rotation is done with period 0.02 sec (so the frequency is 50 Hz) as shown in Fig. 3.2, while the maximum or minimum values between the phases is with a difference 120 degrees. Also the results do not change if the machines have a different number of poles.

From the above principles results the relationship between the frequency (f), the pole machine pair number (2p) and the speed of rotation of the magnetic field (n_d):

$n_d = f / 2p$ in rotations per sec or
$n_d = f \times 60 / 2p$ in rotations per minute (rpm)

This speed is called the synchronous speed of the magnetic field and depends on the frequency of the current supplied to a machine and the number of poles of the winding. Finally synchronous speed is important because it determines the speed of AC motors.

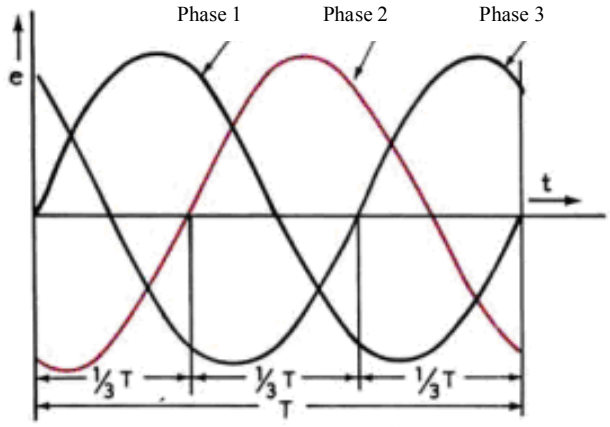


Figure 3.2 Three-phase system voltages applied in 3 ~ motor.

3.3 Three-phase asynchronous motors

The engines which are more prevalent today and are used in most applications are the Asynchronous 3~ motors. The asynchronous induction motor is the simplest motor in construction. The main elements of the motor are the stator and rotor [14]. The stator and rotor consist of circular plates which are toothed on the outer and inner periphery respectively. The stator and the rotor is formed by pressing the sheets so that the sheets to form the channels that will be placed winding. Stator carries three phase distributed winding where the voltage is applied thereby causing the rotating magnetic field. The stator winding consists of copper wire coils, within the stator channels. The rotor winding is compact. Within each channel there are

conductive bars which are shorted at their ends by short-circuit rings. In this way the current in the rotor generates a second magnetic field.

The motor operating principle is simple. The rotating magnetic field generated from the 3~ stator winding, induces an electromotive force in the rotor winding, which in turn generates a second field of in the shorted winding. The combination of the two fields causes rotary motion of the motor shaft [14]. The motor speed differs from the synchronous speed of the magnetic field. The shaft speed is reduced by a small number of turns, because the induced field cursor follows the stator field. That is, the shaft rotates asynchronously.

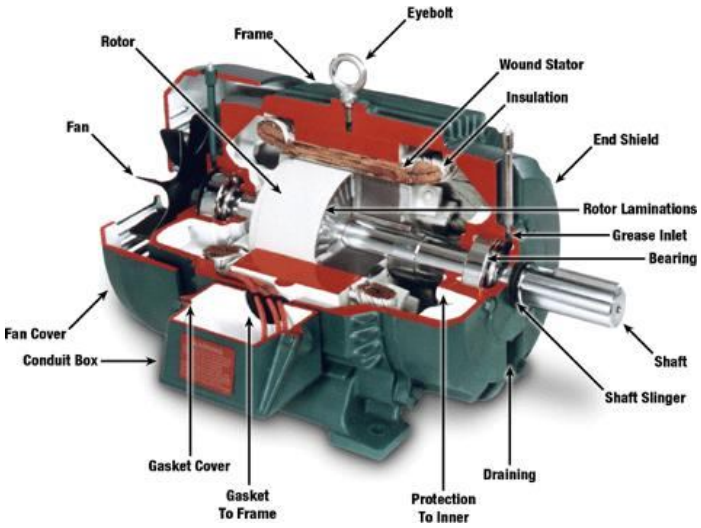


Figure 3.3 Section 3~ asynchronous induction motor.

3.4 Equivalent circuit asynchronous motor

In accordance with the electrical characteristics of the windings, the asynchronous 3~ motor presents ohmic resistances, inductances, etc. Combining all these elements together results in the equivalent electric circuit of the motor [14].

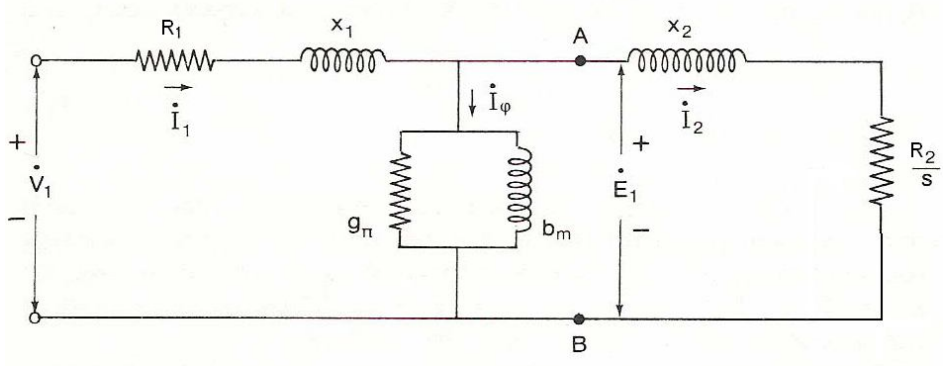


Figure 3.4 Equivalent circuit of asynchronous 3~ motor.

Since a rotating magnetic field is generated by the stator windings, it continues to rotate around the stationary first rotor. The current of the rotor, which causes the induced voltage, form together with the rotating field of the stator a cumulative torque.

$$M \sim \Phi_{stator} \text{ x } I_{rotor}$$

When this torque is too large, it begins to rotate the rotor in the same direction of the rotating magnetic field. Since rotor current is created by the effect of induction, the asynchronous motor is often called induction motor.

3.5 Typical torque – speed relationship

The general equation for the relationship of the induced torque on the speed of rotation in a 3~ motor is given by the following relationship. It is derived from the Thevenin equivalent electric circuit of the motor:

$$M = \frac{3V_{TH}^2 \cdot R_2/s}{\omega_{sync} \cdot [(R_{TH} + R_2/s)^2 + (X_{TH} + X_2)^2]}$$

While the graphical speed torque performance is as follows:

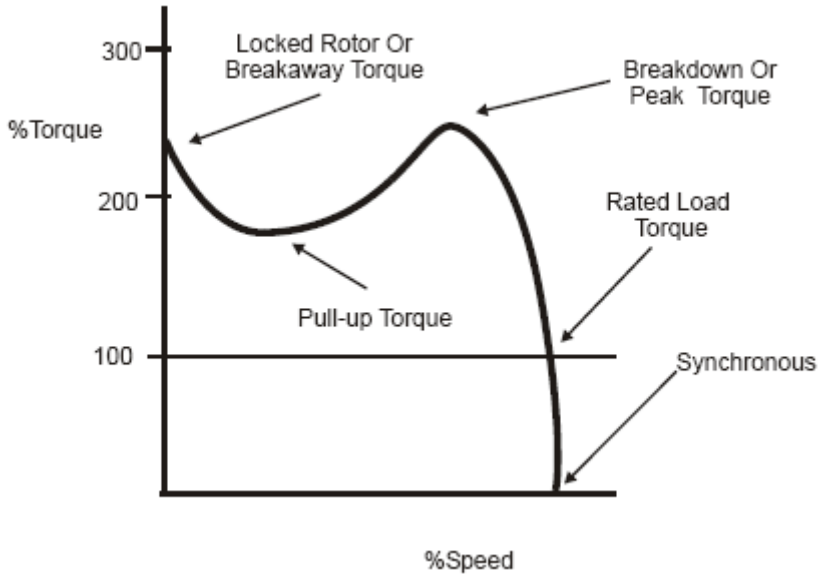


Figure 3.5 Graphical representation of torque-speed asynchronous 3~ motor.

The torque-speed curve of the induction motor shown in Fig. 3.5 and 3.6 gives some important information for the operation of induction motors.

- The induced torque of the asynchronous motor is equal to 0 if somehow the engine is approaching the synchronous speed of the rotating magnetic field.

- Between the operating points without load and full load, the torque-speed curve is almost linear. In this region the rotor ohmic resistance is much larger than the inductive reactance, and

thus the rotor current, the magnetic field and the induced torque increase linearly with increasing slip.

- There is a maximum value of torque. If it is overcome, the engine stops abruptly. This torque is called the peak torque and is two to three times greater than the nominal engine torque.

- The starting torque is slightly larger than the nominal torque and depends on the construction characteristics of the engine. So the asynchronous motor can be started under load.

- The torque varies with the square of the supply voltage. This fact is very useful in order to control the speed, by adjusting the voltage in the stator winding.

- If the rotor of the asynchronous motor is rotated faster than synchronous speed, the rotational direction of the induced torque is reversed so that the machine operates as a generator.

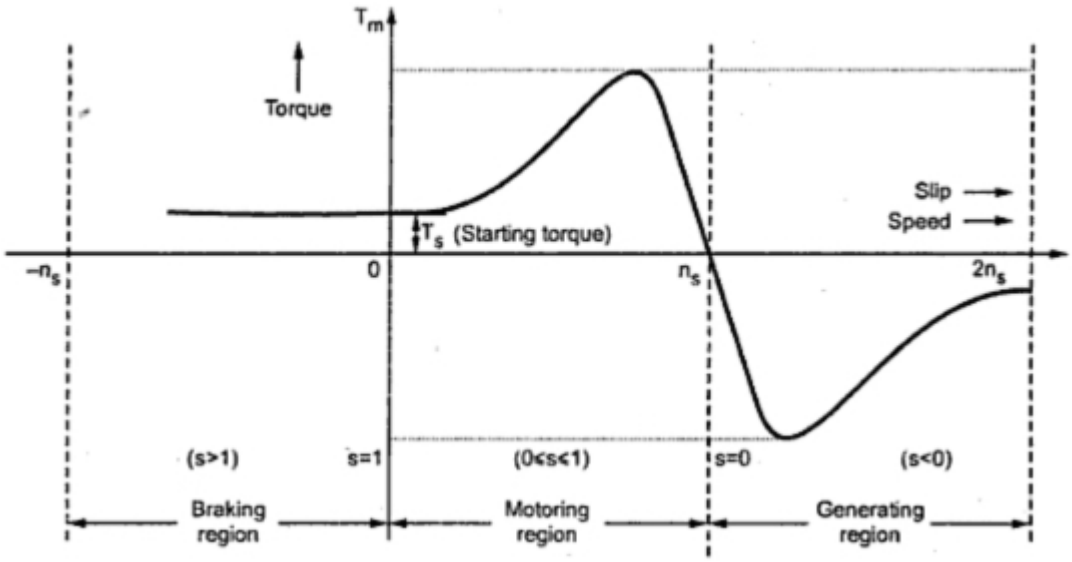


Figure 3.6 Graphical representation of torque-speed asynchronous 3~ motor in 3 operating ranges (braking area, motor and generator).

3.6 Voltage change in the motor

If in a given asynchronous motor the voltage changes from the U-value to the value U' then the magnetic flux Φ of the induced field varies [14]. Hence the induced voltage in the rotor in the starting operation changes its value from U_{20} to U'_{20}. The engine torque is given by:

$$M = C \cdot U_{20}^2 \frac{s \cdot R_2}{R_2^2 + (s \cdot X_2)^2}$$

(U_{20} represents the induced voltage generated by the magnetic field of the stator).

Therefore, one can write approximately that the variation in torque of an asynchronous three-phase motor is proportional to the square of the variation of motor voltage.

$$M' \approx M\left(\frac{U'}{U}\right)^2$$

This gives us very important information for the operation of the engine when the supply voltage adjustment methods are applied and the engine is under load.

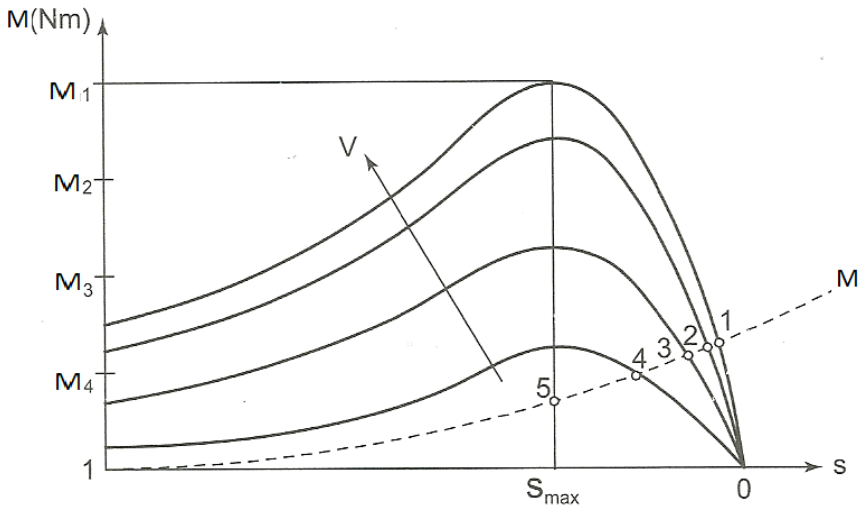


Figure 3.7 Graphical torque performances asynchronous 3~ motor at various values of the supply voltage.

3.7 Starting current

When starting an asynchronous 3~ motor, the current can take values from 3 to 7 times the nominal value [14]. With increasing speed, the current intensity is reduced, and especially the region of the roll slip (point where the torque is maximum) Fig. 3.8. The maximum starting current corresponds to the short-circuit current that occurs when the engine is stationary and is determined by the value of the motor resistance. The starting current does not depend on the load on the shaft, but the start-up time and the flowing duration of that large current is dependent on this. Direct start is only under conditions determined by the administrator of the electricity network because of the large volume of starting current. Therefore, in most industrial electric drive applications a starting system is required that mitigates the value of starting current.

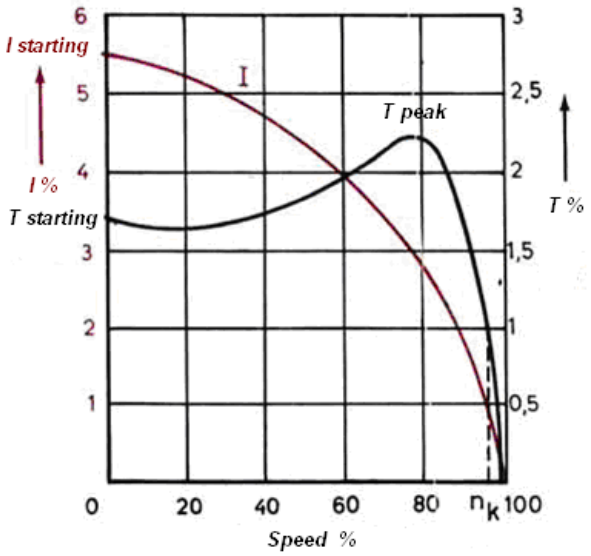


Figure 3.8 Graph of Torque vs. speed and current vs. speed in asynchronous 3~ motors.

4 Introduction to power electronics

Since the late 1950s, when silicon controlled rectifier (SCR) appeared dynamically, the development of the scientific field of Power Electronics has been rapid. New semiconductor manufacturing techniques improved all the features of such elements and today they control most industrial applications. Following the evolution of diodes and Thyristors, other semiconductor switch elements were developed, with improved characteristics for industrial applications, such as Triacs, GTO, BJT, MOSFET power, IGBT etc. Anywhere in the industry where it is required to regulate voltage, current or frequency, power electronic devices find application [13].

4.1 The Diode

The active material of the power diode is silicon (Si), which is a semiconductor and its conductivity is located between conductors and insulators. The diode is composed of two semiconductor elements P-type and N-type [13]. In the N-type semiconductors, majority carriers of current are electrons and minority carriers are holes. The opposite happens in the P-type semiconductors. The diode shown in Fig. 4.1 is formed by contacting the P-type material and N within a unit crystal. On contact, the free electrons of the N material and the free holes of P material are combined leaving the N side with positive charges and the P side with negative charges. Thus there is a potential barrier along the contact, with value of about 0.65V.

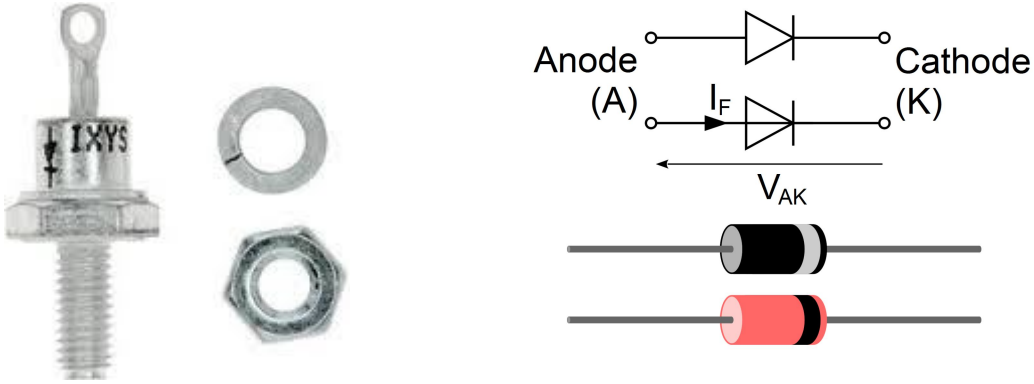


Figure 4.1 Schematic and true portrayal of a diode.

The characteristic curve of the diode is shown in Fig. 4.2. A positive voltage between P (anode) and N (cathode) will cause current to flow as soon it overcomes the barrier potential of 0.65V, giving an overall correct voltage drop of about 0.7 V at nominal current. The application of a reverse voltage will prevent current flow [13].

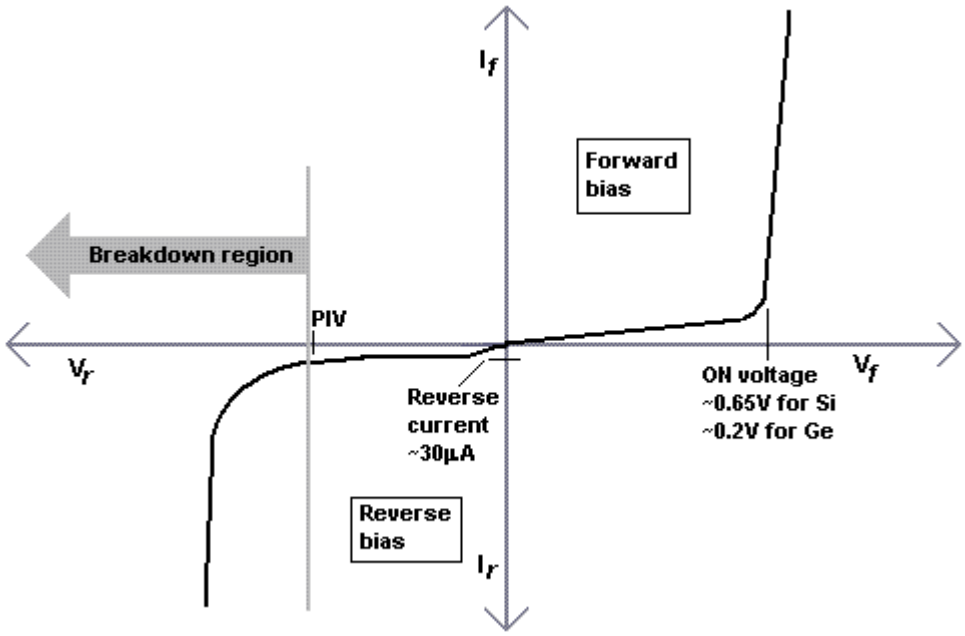


Figure 4.2 Typical current - voltage characteristic of a diode.

4.2 The Thyristor

The thyristor is a four-layer element P-N-P-N with a third gate terminal [G], as shown in Fig. 4.3. The two anode-cathode terminals are the power circuit of the thyristor, while the ignition control is through the gate electrode. As opposed to the diode, when the gate of the thyristor is not given ignition pulse even it is correctly polarized, there is no-current flow [13]. It is a controlled electronic switch which conducts when it is given a pulse at the gate.

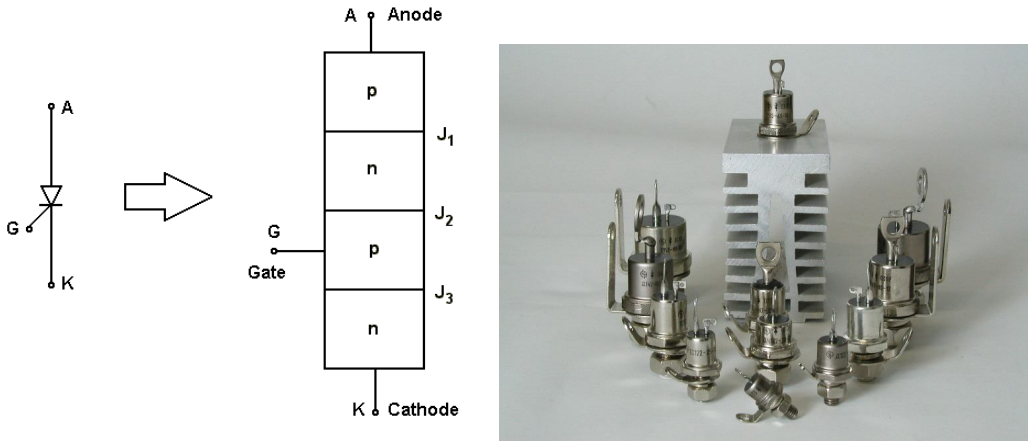


Figure 4.3 Schematic and true portrayal of a thyristor.

The thyristor when correctly polarized (positive anode) can be driven into the conducting state by introducing current to the gate terminal from the positive gate voltage - cathode as shown in Fig. 4.4. The action of the gate current is to introduce holes in the inner P slice, which

together with electrons from the cathode layer N, cleave the central control contact, resulting in the thyristor conduction state. Since the anodic current exceeds the critical current level of treatment, the gate current can be reduced to zero and the thyristor remains in the conducting state regardless of the gate current.

To extinguish the thyristor, anode current must be reduced below the level of holding current and a relatively large time must pass in order to enable the control contact of the thyristor to regain the cutt-off ability, before a forward voltage can be applied again while the thyristor remains in the switch off state. Typically, to extinguish the thyristor, the anodic current is reversed by means of the external circuit. When a a reverse current passes for a very short time, the charge is allowed to run through the P-N layers and leads the two external contacts to cleave any reverse current after the recovery of the space charge. The space charges is due to the presence of current carriers in the contact area. However, the central control contact will not acquire the ability to reactivate a forward voltage before certain a time period.

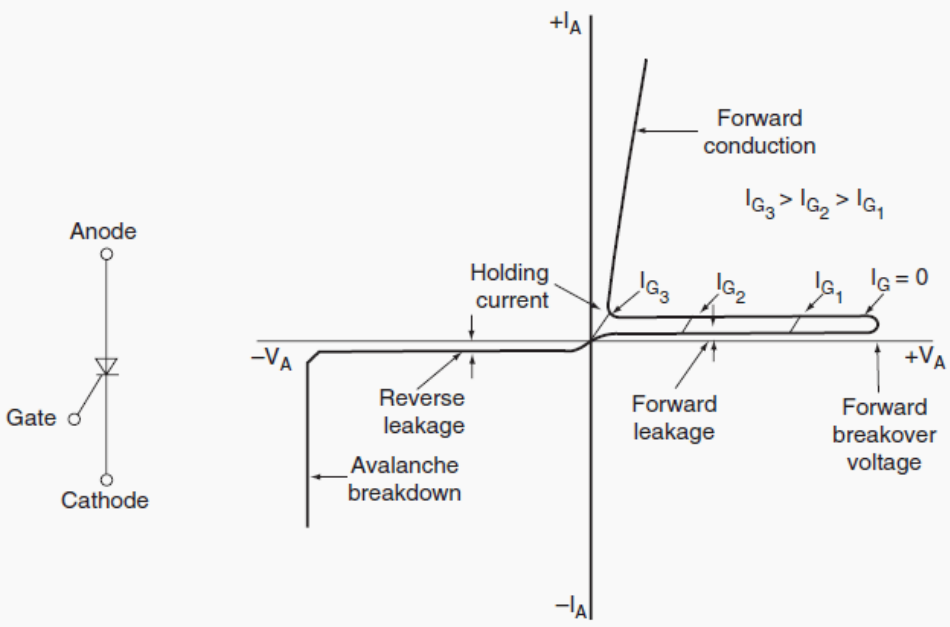


Figure 4.4 Typical curve current - voltage of thyristor.

From the above conduct and non-conduct operation of thyristors we observe that the semiconductor presents a rudimentary form of memory. When applying current to the gate, it conducts as long as it is correctly polarized and the upstream current has not been decreased or reset. (Fig. 4.5).

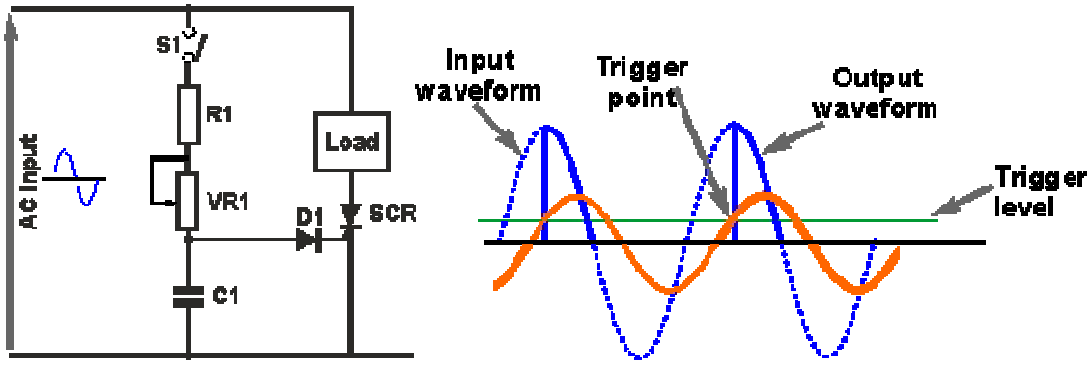


Figure 4.5 Connecting a thyristor for the control of 1~ ac load.

In addition to the conventional thyristor which can be switched off only by reducing or zeroing the anodic current there is an element known as a "quenching gate thyristor", in which quenching is possible by reverse gate current. The ignition requirements are similar to those of the conventional thyristor, but the cost of additional gate circuits, the restrictions of the nominal values and the losses have delayed its introduction to many applications.

4.3 The Triac

Triac is an element with five (5) layers as shown in Fig. 4.6 which shows a structure of P-N-P-N in any direction of the terminals A1 and A2 and thus can conduct in either direction as indicated by the symbol. Electrical the TRIAC includes in an element that would need two thyristors (Fig. 4.6).

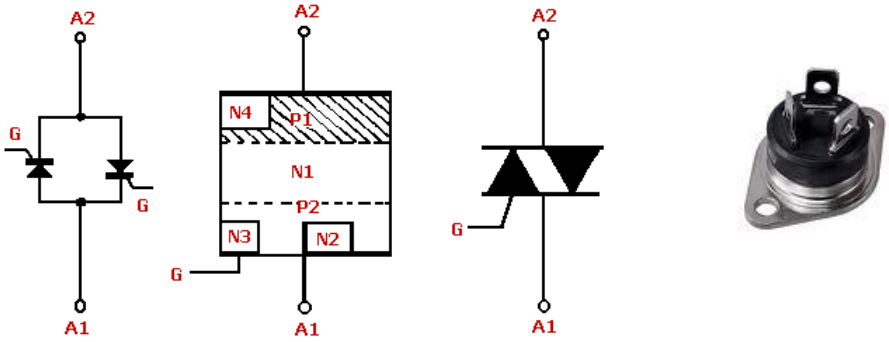


Figure 4.6 Schematic and true portrayal of a triac.

The TRIAC can be driven in the conducting state with positive or negative gate current, but is more sensitive if a positive current is introduced when the A2 is positive or negative current when the A1 is negative.

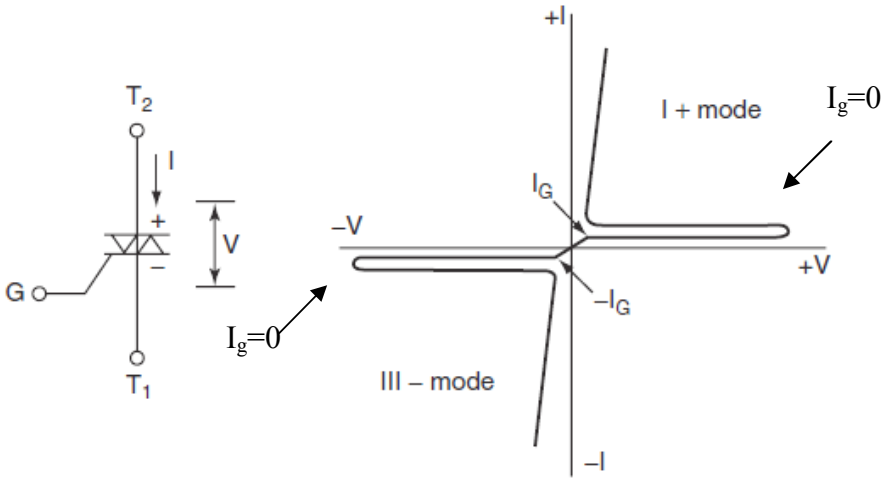


Figure 4.7 Typical current - voltage characteristic of a triac.

However, in practice always negative gate current is used, as shown in the characteristic of Fig. 4.7. The maximum nominal values are constant and transitional situation is similar to that of a thyristor [13]. Obviously the triac finds applications in AC/AC converters, as a bidirectional switch. The voltage drop across the TRIAC (or thyristor), will result in the production of heat and for this reason the device needs cooling.

The transient voltages that may occur at the junction require repression with a RC filter along the static switch. The presence of this network means that, with an open circuit on the load, in which case the supply voltage can drop on the load, which is dangerous.

The advantages of the semiconductor static switch are the speed of response, accuracy of operation, the absence of regular maintenance and lack of noise.

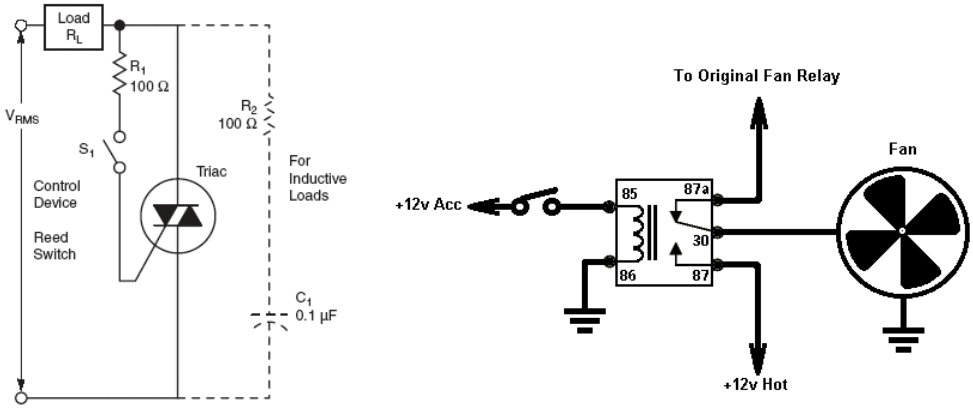


Figure 4.8 The operation of the TRIAC as a static switch.

4.4 Inductive loads and power electronics

The thyristor and TRIAC can be used in a number of devices for controlling the voltage of an AC load circuits in both single phase and 3-phase. Such circuits are used to control voltage loads that are ohmic or contain some induction.

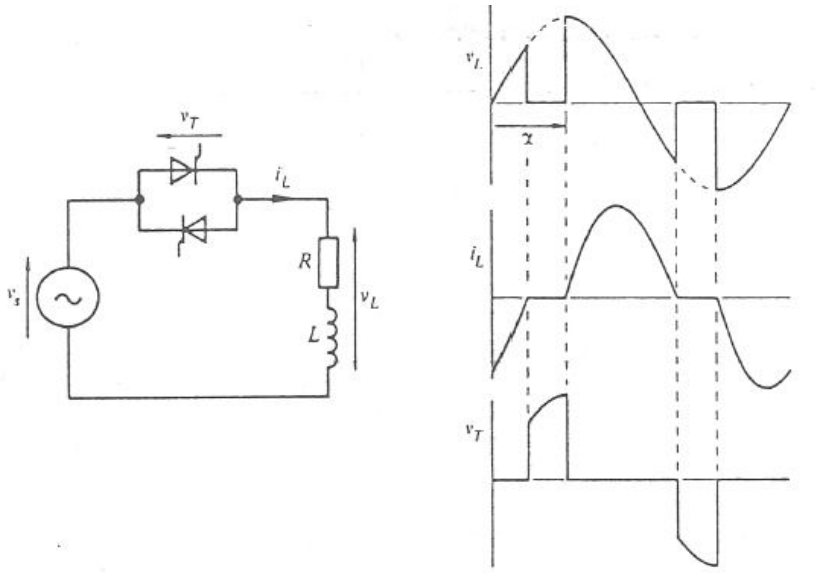


Figure 4.9 Waveforms of voltage and current in the inductive load controlled with antiparallel thyristor (TRIAC).

Figure 4.9 shows the waveforms obtained when the phase angle delay at inductive loads. The current continues beyond the zero of the supply voltage due to induction, and thus gives the waveform of the voltage V_L shown [13]. Rapid voltage increase in antiparallel thyristors during turn off, might lead to a large dv/dt. The high derivative dv/dt makes the problem bigger than the TRIAC in thyristor.

In inductive loads, continued connection beyond the zero voltage means losing control under a certain firing angle. The ignition control of the semiconductor is recovered after zeroing the current.

5 The Soft Starters

The soft starters are self-starting systems installed mainly in electric motion systems and in places where there are many 3~ motors, large horsepower motors or working with continuous starts. With the start command, the soft starter connects the live cables of the electric network with the motor and controls the value of the supply voltage with power electronic components, so that to reach the nominal operation within fixed (adjustable) time. At the end of the start process, the electricity is restored through a bypass relay directly connecting the motor windings in the electric network [10], [11].

Figure 5.1 Series Soft Starters from trade.

5.1 The principle of operation

The principle of soft starter operation is the controlled gradual acceleration of the asynchronous motor from inaction (stop) to the nominal operation of the motor supply voltage. The process has as a parameter the start time. The whole circuit is similar to an AC/AC converter. The method succeeds in the limitation of current during startup, due to the reduced value of the applied voltage, which increases progressively with time. The start time ranges from a few seconds up to several minutes, depending on the motor and the load on the shaft. The soft-starter sets the voltage by controlling the firing angle of the thyristors or triacs. In this way it controls the effective value of the voltage across the motor during startup. The ignition angle is determined by a microcontroller which monitors, controls and analyzes all motor startup information. After the soft start takes place, the microcontroller signals the By-Pass relay to take the entire electric motor load relieving the semiconductors from electrical strain while they cut off the gate signal. The microcontroller continues to monitor the motor after the end of the starting process by executing various diagnostic tests. In normal operation, the

microcontroller keeps the bypass relay stimulated and controls the electrical measurements for the nominal operation of the motor.

5.2 Settings and parameters

The most common parameter set by the user in order the soft starter to operate efficiently is the starting time or ramp rise time. In essence, this is the time it takes the motor to recover "smoothly" its full operation. Besides the time ramp, other functions can also be parameterized in a soft starter, such as ramp down time, the initial voltage at startup, the permissible limit starting current, etc. In the following, we review the main parameters of a starter.

a) Ramp rise time

The time in which the soft starter gets full voltage at its output is called rise time. It starts from an initial value. The rise time is determined by the load carrying on the shaft of the motor and its structural features. Selecting very long startup time may cause the overheating phenomenon, with the possible failure of the thermal protection switch. If the motor has no initial load, then the period may be less than it would have been if the motor had a load. In fact with the setting of the timing of the rise ramp, the slope of the line ramp is as shown in Fig. 5.2.

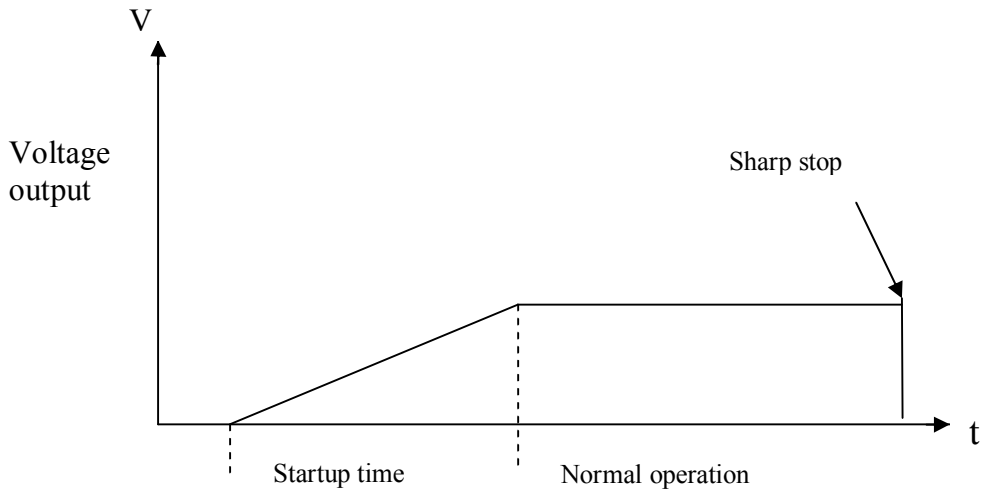


Figure 5.2 Graphical representation of the rise time ramp.

b) Ramp down time

Except from the start up process, a soft starter has the possibility to gradualy decelerate the motor, allowing smooth reduction of speed until stop. This feature has a special utility mainly when the motor has a particular load in its shaft. For example, if the shaft has a pump, avoiding in this way the water shock generated during heavy stop of the pair motor - pump or when it comes to powertrain wear belts and conveyor belt drivers. In smooth stop, voltage in starter

output is gradually reduced until it is zeroing according to the predetermined time. The current of the motor depends on the load torque on the shaft and the desired deceleration time. By setting the time of the descent ramp we adjust the slope of the descent line. If the fall time is selected equal to 0, then the slope is infinite, and this would correspond to a sharp stop (Figure 5.2 and 5.3).

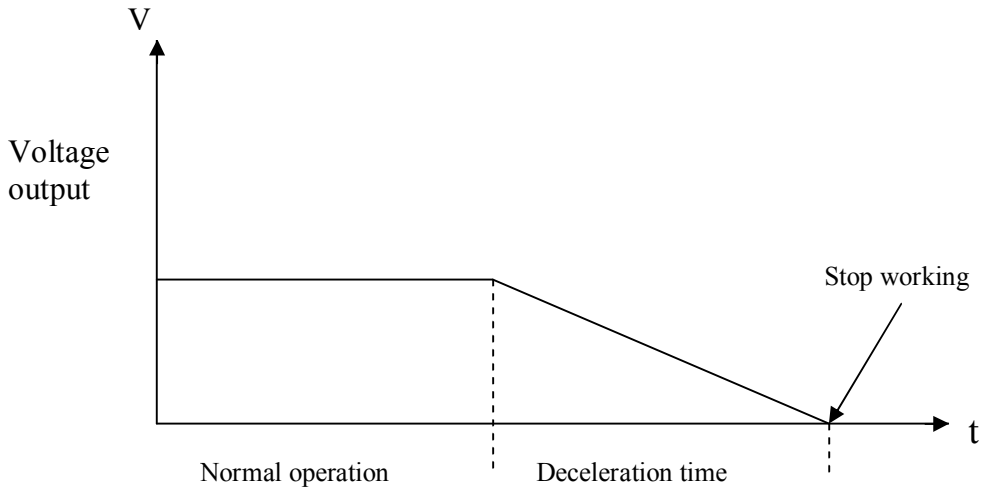


Figure 5.3 Graphical representation of the deceleration time ramp.

c) Initial voltage

The initial value applied to the output of the starter is called initial voltage. This follows the activation of startup process, or the reduced value during the deceleration process. Usually, the voltage on the two intervals of rise and down ramp time is mostly the same as the opposite slope (Fig. 5.4).

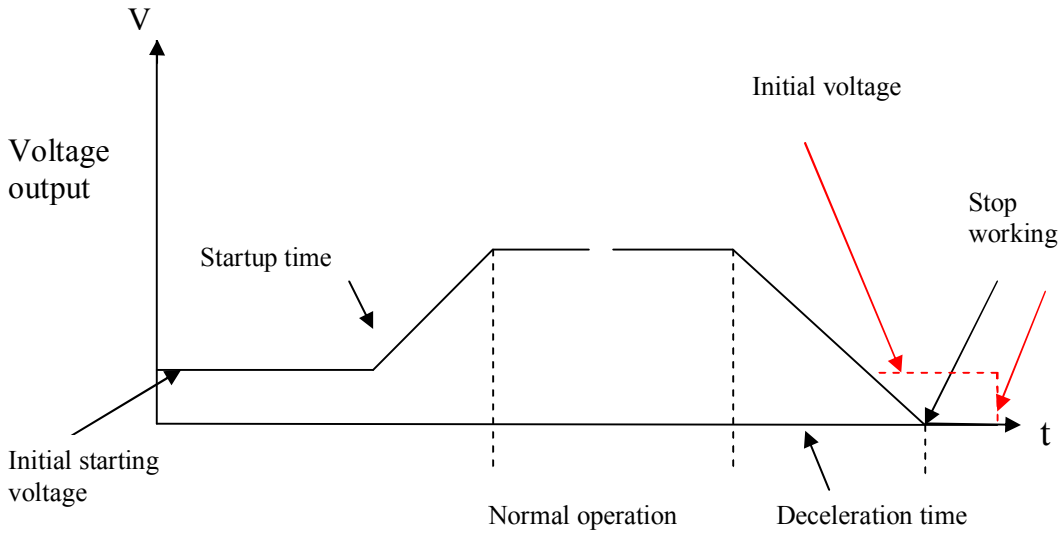


Figure 5.4 Graphical representation of the rise time and deceleration time ramp with initial voltage.

The engine torque decreases with the square of the voltage so this means that if the initial voltage set too low, eg for 20% of the rated voltage, the starting torque will become 0.2^2 = 0.04 or 4%. In other words, the motor will not be able to start very early. For these reasons it is important to seek out a satisfactory level of initial voltage so the motor can be started smoothly without increased heat stress.

d) Permissible limit of elecrtic current

Another important characteristic function, which could intervenes anyone and makes corrective adjustments, is the limit of current. This limit can be used in applications where a limited starting current is required, or in the case of a hard startup which can not be achieved only by regulating the initial voltage and ramp time.

5.3 Advantages of Soft Starters

Using the soft starters many mechanical and electrical problems are treated. Soft-starters ensure maximum performance, economy and life of motors. All conventional starters for induction motors present problems such as:

- Shock startup currents affecting the electric network.
- The impulse startup torque stress on mechanical and hydraulic loads.
- The inability to adapt to the particularities of loads.
- The need for maintenance.
- Failure to smoothly brake the motor.
- Increased thermal losses.
- The motor wear.
- The inability of many starts per hour.
- The need to use 6 wires (Start with Star-Delta switch).

The soft start starters, provide a solution to all these problems. Modern soft starters offer a choice of ready programmed applications, fully integrated electronic protection, display, ability to control voltage and torque, analog input and output, built-in contactor by-pass, communicate using various communication protocols and connectivity in line or inside delta motor connection.

6 The PIC 18F252 microcontroller

6.1 PIC 18F series Microcontrollers

The PIC 18F series are integrated circuits belonging to the class of microcontrollers [17]. They took their name from the initials of Peripheral Interface Controller. They contain all the necessary components such as central processing unit, input-output ports for communication with peripheral systems and memory in order to compose a digital programmable system [12], [15] .The 18F Series devices advance the core, while keeping peripherals reasonably constant. This feature, allow the PIC microcontroller to move into a bigger field – feature which enhance real time operation, ease the use of high-level languages and allow interaction with much larger memories[15]. Some microcontrollers also include A/D and D/A converters, PWM modules, formal communication channels, such as UART or I2C and other special peripherals. So, have the role of a central control unit of a system. External looks like simply digitally integrated circuit, but inside hide a microcomputer, so often mentioned as computer on a programmable chip [12], [15]. The PIC microcontroller is a small autonomous computer system programmed to perform a specific logical sequence of commands that have been entered in the permanent memory. Whene the microcontroller is restarted performs the same logic. Receives the data, processes them and based on the results, controls its environment. It is about for a special purpose system, dedicated to the control and automation of a particular service [12]. The characteristics that make PIC18F series more powerful and more attractive in applications than the other sereis of the same class are [17]:

• The number of instructions more than doubled, with 16-bit instruction word

• Enhanced Status register

• Hardware 8x8 multiply

• More external interrupts

• Two prioritised interrupt vectors

• Radically different approach to memory structures, with increased memory size

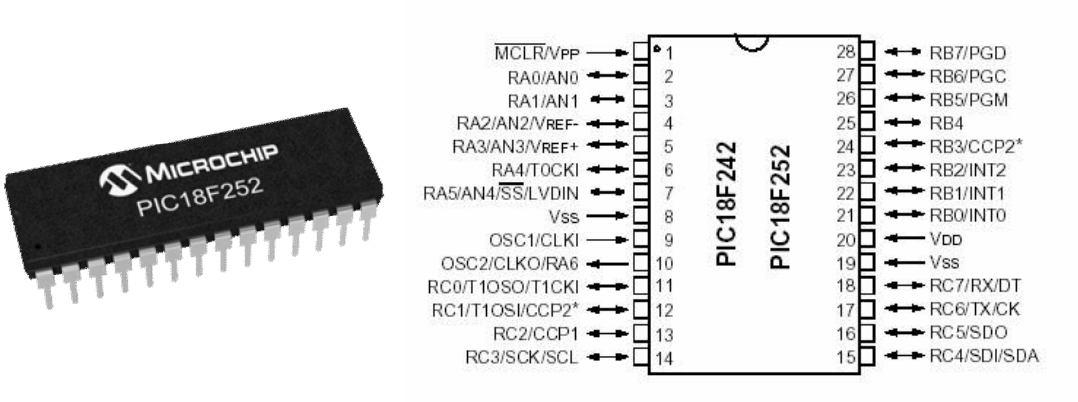
• Enhanced address generation for program and data memory

• Bigger Stack, with some user access and control

• Phase-locked loop (PLL) clock generator.

6.2 The PIC 18F252

The PIC18F252 is a powerful and easy-to-program CMOS FLASH-based 8-bit microcontroller of Microchip Company [17], suitable for advanced embedded projects. It is based on architecture RISC and consists of 28 pins as illustrated in Fig 6.1.



(a) (b)

Figure 6.1 (a) The MCU PIC 18F252, (b) The pins of PIC 18F252.

One feature that makes the PIC18F252, very attractive and suitable for applications is the ability of program memory be erased and reprogrammed. Like all microcontrollers PIC, model 18F252 integrates Harvard architecture, whose main feature is the different commands and data path [12], [15], [17]. Thus there is difference in program memory and RAM. The command bus has 16 bits width. Then each command consists of a word of 16 bits and therefore runs on a single execution cycle, which is a basic feature of the RISC architecture. To generate clock pulses, the microcontroller usually bring to its terminals an external crystal with frequency that can be from 32 KHz to 40 MHz [17].

The PIC18F252 microcontroller has 1536 bytes RAM memory where the user can define variables or to store temporary data. Each register in the data memory has a 12-bit address, allowing up to 4096 bytes of data memory. The data memory contains Special Function Registers (SFR) and General Purpose Registers (GPR) and is divided into as many as 16 banks that contain 256 bytes each. The lower 4 bits of the Bank Select Register (BSR) select which bank will be accessed. The SFRs start at the last location of Bank 15 (0xFFF) and extend downwards. Any remaining space beyond the SFRs in the Bank may be implemented as GPRs. GPRs start at the first location of Bank 0 and grow upwards. To ensure that commonly used registers (SFRs and select GPRs) can be accessed in a single cycle, regardless of the current BSR values, an Access Bank is implemented. A segment of Bank 0 and a segment of Bank 15 comprise the Access RAM [17].

Address	Name	Address	Name	Address	Name	Address	Name
FFFh	TOSU	FDFh	INDF2(3)	FBFh	CCPR1H	F9Fh	IPR1
FFEh	TOSH	FDEh	POSTINC2(3)	FBEh	CCPR1L	F9Eh	PIR1
FFDh	TOSL	FDDh	POSTDEC2(3)	FBDh	CCP1CON	F9Dh	PIE1
FFCh	STKPTR	FDCh	PREINC2(3)	FBCh	CCPR2H	F9Ch	—
FFBh	PCLATU	FDBh	PLUSW2(3)	FBBh	CCPR2L	F9Bh	—
FFAh	PCLATH	FDAh	FSR2H	FBAh	CCP2CON	F9Ah	—
FF9h	PCL	FD9h	FSR2L	FB9h	—	F99h	—
FF8h	TBLPTRU	FD8h	STATUS	FB8h	—	F98h	—
FF7h	TBLPTRH	FD7h	TMR0H	FB7h	—	F97h	—
FF6h	TBLPTRL	FD6h	TMR0L	FB6h	—	F96h	TRISE(2)
FF5h	TABLAT	FD5h	T0CON	FB5h	—	F95h	TRISD(2)
FF4h	PRODH	FD4h	—	FB4h	—	F94h	TRISC
FF3h	PRODL	FD3h	OSCCON	FB3h	TMR3H	F93h	TRISB
FF2h	INTCON	FD2h	LVDCON	FB2h	TMR3L	F92h	TRISA
FF1h	INTCON2	FD1h	WDTCON	FB1h	T3CON	F91h	—
FF0h	INTCON3	FD0h	RCON	FB0h	—	F90h	—
FEFh	INDF0(3)	FCFh	TMR1H	FAFh	SPBRG	F8Fh	—
FEEh	POSTINC0(3)	FCEh	TMR1L	FAEh	RCREG	F8Eh	—
FEDh	POSTDEC0(3)	FCDh	T1CON	FADh	TXREG	F8Dh	LATE(2)
FECh	PREINC0(3)	FCCh	TMR2	FACh	TXSTA	F8Ch	LATD(2)
FEBh	PLUSW0(3)	FCBh	PR2	FABh	RCSTA	F8Bh	LATC
FEAh	FSR0H	FCAh	T2CON	FAAh	—	F8Ah	LATB
FE9h	FSR0L	FC9h	SSPBUF	FA9h	EEADR	F89h	LATA
FE8h	WREG	FC8h	SSPADD	FA8h	EEDATA	F88h	—
FE7h	INDF1(3)	FC7h	SSPSTAT	FA7h	EECON2	F87h	—
FE6h	POSTINC1(3)	FC6h	SSPCON1	FA6h	EECON1	F86h	—
FE5h	POSTDEC1(3)	FC5h	SSPCON2	FA5h	—	F85h	—
FE4h	PREINC1(3)	FC4h	ADRESH	FA4h	—	F84h	PORTE(2)
FE3h	PLUSW1(3)	FC3h	ADRESL	FA3h	—	F83h	PORTD(2)
FE2h	FSR1H	FC2h	ADCON0	FA2h	IPR2	F82h	PORTC
FE1h	FSR1L	FC1h	ADCON1	FA1h	PIR2	F81h	PORTB
FE0h	BSR	FC0h	—	FA0h	PIE2	F80h	PORTA

Note 1: Unimplemented registers are read as '0'.
2: This register is not available on PIC18F2X2 devices.
3: This is not a physical register.

Figure 6.2 Special Function Register map of PIC 18F252.

One of the major advantages of PIC 18F252 is the structure of the ports. The device has twenty three input/output pins, divided into three ports called PORTA (7 pins), PORTB (8 pins) and PORTC (8 pins) respectively. Each port has three registers for its operation. The TRIS register (data direction register), the PORT register (reads the levels on the pins of the device) and the LAT register (output latch). The data latch (LAT register) is useful for read-modify-write operations on the value that the I/O pins are driving.The LAT register associated with an I/O pin eliminates the problems that could occur with read-modify-write instructions. A read of the LAT register returns the values held in the port output latches, instead of the values on the I/O pins. A read-modify-write operation on the LAT register, associated with an I/O port, avoids the possibility of writing the input pin values into the port latches. A write to the LAT register has the same effect as a write to the PORT register.

Another important feature of PIC18F252 devices is the multiple interrupt sources and the interrupt priority that allows each interrupt source to be assigned a high priority level or a low

priority level. The high priority interrupt vector is at 000008h and the low priority interrupt vector is at 000018h. High priority interrupt events will override any low priority interrupts that may be in progress.

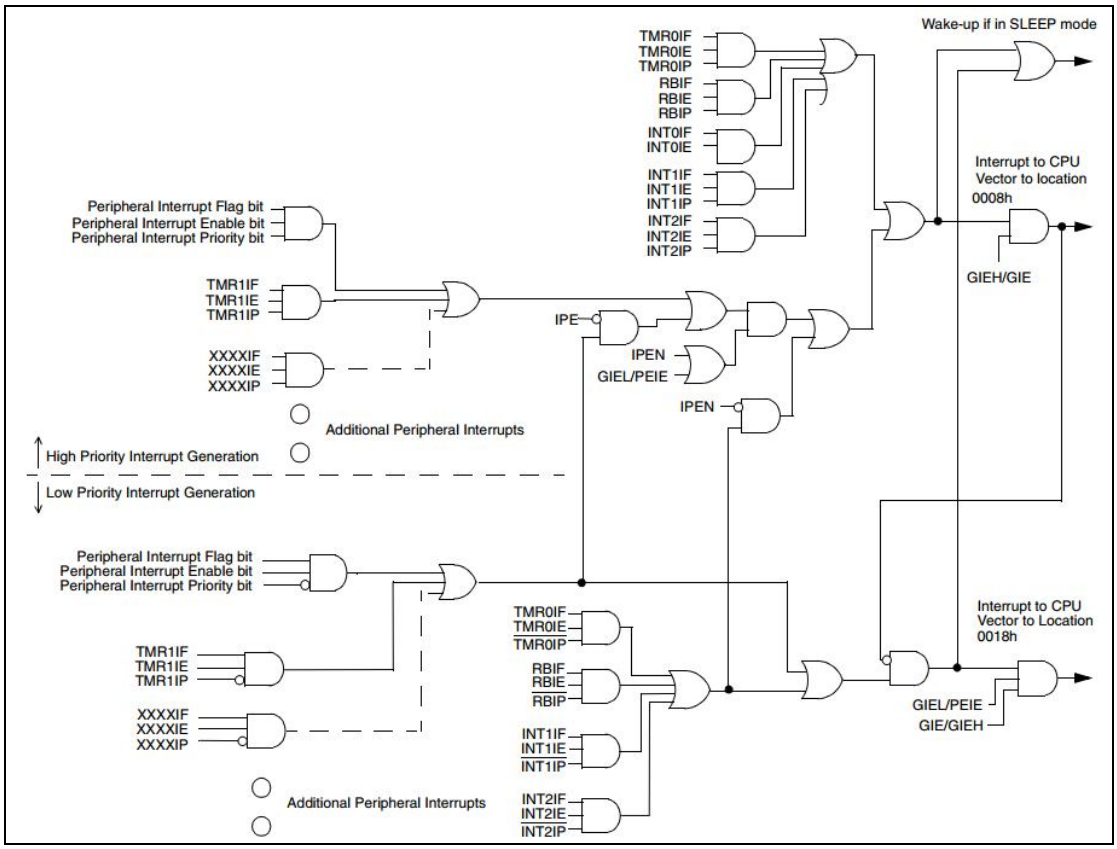


Figure 6.3 Logic diagram of interrupts in PIC 18F252.

The microcontroller has four timers that operate as counters. The TMR0 is an 8-bit or 16-bit timer/counter where the most low bits is named TMR0L and the most high bits is named TMR0H . The TMR1 is a 16-bit timer/counter with two 8-bit registers, TMR1H and TMR1L. The TMR2 is an 8-bit timer and can be used as the PWM time-base for the PWM mode of the CCP module. Finally the TMR3 is a 16-bit timer/counter with two 8-bit registers, the TMR3H and TMR3L. Inside the PIC 18F252 also has an oscillating timer which supervises and resets the microcontroller, in case is out of control. This timer is called WDT or Watchdog Timer. The microcontroller's program memory has a capacity of 32K bytes. In addition, the PIC 18F252 has other 256 bytes EEPROM memory, intended for permanent data storage. Furthermore, the microcontroller is provided by a series of remarkable peripheral devices such as two capture/compare/PWM modules, two serial communication modules and five, 10-bit resolution, ADC channels.

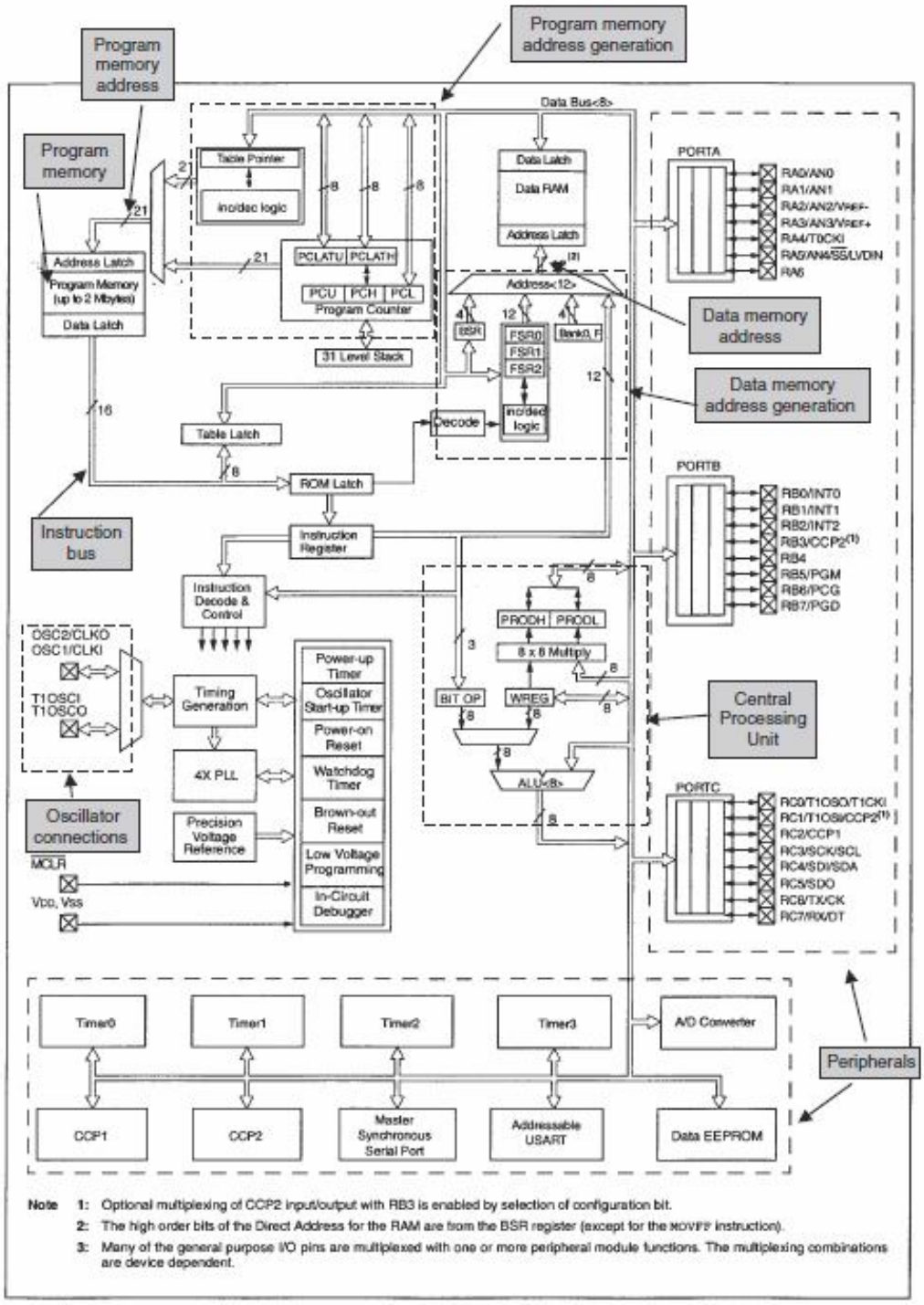


Figure 6.4 The PIC 18F252 block diagram.

7 Designing the Soft Starter

The subject of this thesis is the design and construction of a simple soft starter for 3~ induction motors, using power electronics controlled by the microcontroller PIC18F252.

7.1 The principle of operation – basic assumptions

The basic function to be performed by the soft starter is the smooth start of an asynchronous 3~ squirrel cage motor by gradually increasing the voltage at its output, from the stop to the nominal operation.

Control of motor voltage is achieved by controlled semiconductors and specifically for this system, with three Triacs, one in each phase. As is known, by controlling the firing angle pulse to the gate of Triac, we can control the time will conduct. In this way only a percentage of the rms input voltage will goes to the output of the starter, according to the appearance of the pulse.

Due to the inductive character of motor the control of Triacs, lost from the system (because of the current still flows in the circuit, after the zeroing of the voltage). This result the output voltage of the starter gets its nominal value faster than the defined start time given by the user. For dealing this phenomenon, adopts a fuzzy logic that its purpose is to determine the next pulse ignition of Triacs and consequently the next step of weighted starter output voltage. The above procedure is achieved, taking for granted that the voltage appearing at the output of the starter due to the non-quench of Triacs. The value of the residual voltage is information about the system and is dependent on the inductance of the motor.

The triggering of the Triac is performed in time with consecutive steps, until the output voltage becomes equal to the input voltage. When this happens, then activates a bypass relay and the handling of the load is transferred to it, in order not stressed electrically the Triacs. Apart from the above operation, the soft starter should autonomously adjust the display time of the firing pulse, in order to achieve a gradual increase of the output voltage to a fixed time putting as information the user. In other words, the start time will be determined and taken as a parameter for the implementation of a Soft Starting.

Summarizing the above, the soft starter must:

1. Understands the zero - crossing in each phase.
2. Determine the firing angle of Triacs with triggering pulses in each phase taking into account the nature of the load and taking as parameter the desired time of the start ramp.
3. To implement the By – Pass condition.

These processes are controlled by the microcontroller PIC 18F252 having sole responsibility to handle all signals of Soft Starter.

7.2 The trigger pulses

As is known, the frequency of the electrical network is 50Hz, this means that one full alternation of the sinusoidal voltage takes T = 1/50 = 0,02 sec. The MCU, have to perceive those points that the alternating voltage is zeroed. This will be implemented by an analog circuit detection of “transit from zero” based on opto-coupler logic. Every time, when the alternating voltage is zeroed, it will produce a short width 5V pulse (logical 1).

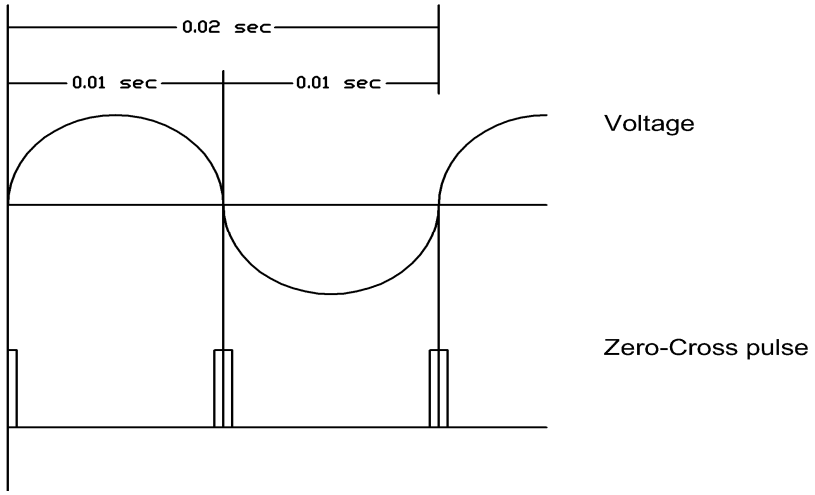


Figure 7.1 Creation of zero-cross pulses.

The zero cross pulses of the three phases are applied as inputs to the MCU which now is in a position to know, when the alternating voltage is zeroed in each phase, with a small tolerance. The main process of the MCU is to produce a pulse width modulated in time capable to trigger the Triac control. The evolution of the modulation of the pulses is determined by the steps of soft starter’s rise ramp.

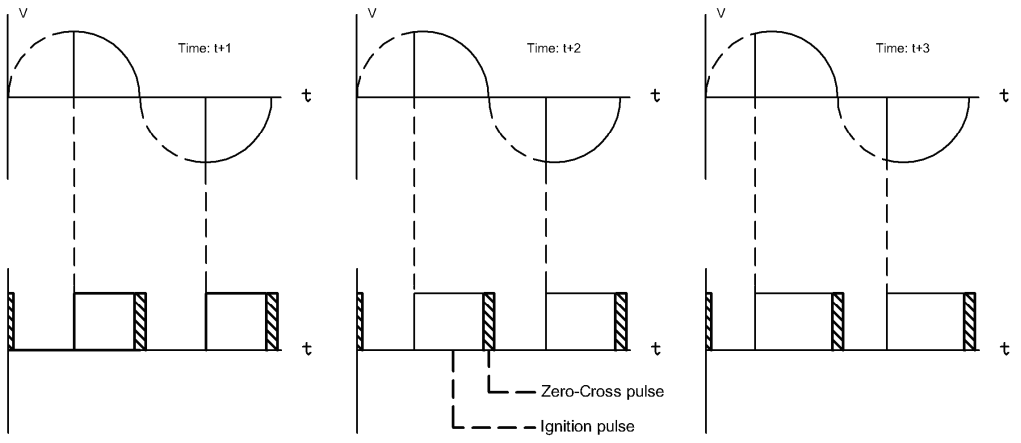


Figure 7.2 Output of the system at various ignition pulses.

Focusing on one half period of a phase, we selected a number of 36 steps that will make up the rise ramp. Namely, 36 levels of starter output to the nominal voltage are defined (full lifting of the ramp).

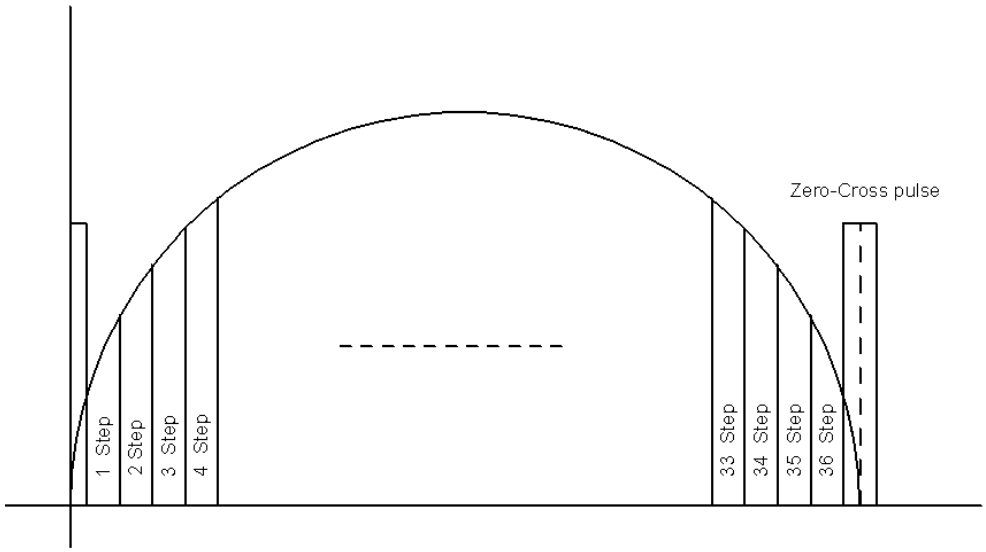


Figure 7.3 The steps of the ramp during a half-periode.

So every step of the rise ramp is identified with a corresponding firing angle of the Triac. The zero-cross pulses are generated by the zero-crossing detection circuit and they will have duration sufficient so that the MCU responds in the time frames of the control loop. The duration specified in 1.2msec (0.6msec on each side of a half-periode). Knowing that the duration of a half-periode is 0.01sec or 10msec, the actual field for implementing the trigger pulse in Triacs is 10-1.2 = 8.8msec. So, the duration of each step will be:

$$\frac{8.8m\sec}{36} = \frac{8800\mu\sec}{36} = 244.44\mu\sec$$

Therefore, if the MCU modulate the pulse trigger in Triacs from 36th step to the 1st step, the value of the starter output voltage will be changed.

The modulation of the trigger pulse depends on consecutive overflows of TMR0, of PIC 18F252, which are carried out every 244.44 μsec. The number of overflows will be measured and then compared to the current step of the starter (Fig. 7.3).

Initially the number of steps in the software of the MCU will be 36. When the MCU understand that has passed from the condition zero-cross, namely from logic "1", the zero-cross pulse is made logical "0" (value 0 Volts), the upcoming overflow will be realized from TMR0, will increases the content of the overflow counter and then compare it with the number of steps. If the number of overflows and the number of steps differ, then the TMR0 reloaded for another overflow. However, if the number of steps is the same as the number of overflows, then the

MCU extract a trigger pulse to ignite the Triac. The trigger pulse has duration until the upcoming pulse crossing from zero. We observe that, TMR0 is one component in the MCU and, by extension, in the software that is intended to produce a continuous overflow signal in time with constant frequency f = 1/244.44μsec = 4.09KHz from consecutives overflows and reloads. This component taken as a parameter and defines the ignition pulses of the Triacs in the three phases. Namely, to implement the ignition pulses in all three phases, only the operation of TMR0 is needed.

The overflow of TMR0 as mentioned should be done every 244.44μsec. To achieve this overflow time we first have to determine the initial contents of the TMR0. The content of the timer for the specific delay time is calculated from the relationship:

$$delay = \frac{4 \times \text{Prescaler} \times (256 - TMR0)}{f_{ocs}}$$

Since the timer TMR0 adjusted to 8-bit mode and the frequency divider is set to 1:16, the content of TMR0 obtained is:

$$244.44 \times 10^{-6} = \frac{4 \times 16 \times (256 - TMR0)}{20 \times 10^{6}} \Rightarrow TMR0 = 179.61$$

Such value with decimal place is impossible be entrusted to an 8-bit register, so become one round and the value assigned to TMR0 is the number $\underline{\mathbf{179_{10}}}$.

So depending on the result of comparing the number of overflows with the number of steps, the MCU will extract an amplitude modulated trigger signal evolving in time in which the width depends on the value of the current step of the soft starter. In this way the pulse ignition of Triacs is formed from 0% to 100% at startup time limits given by the user.

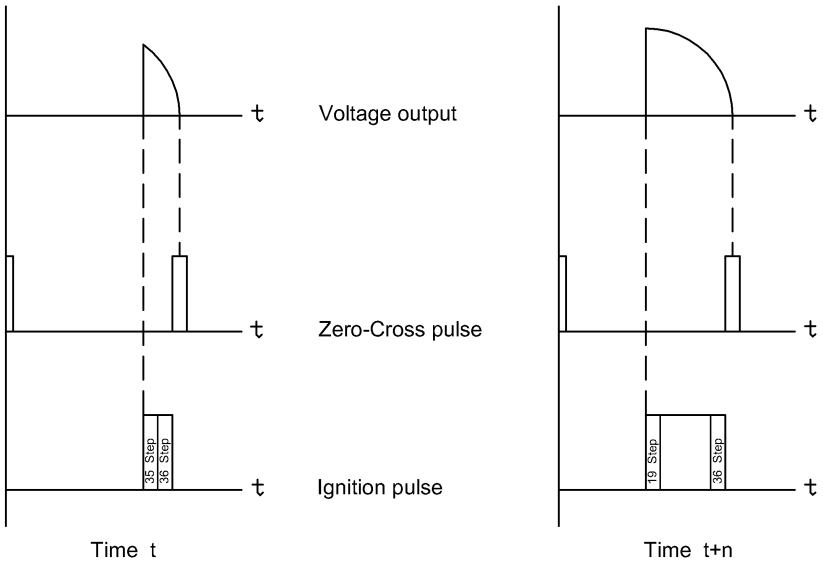


Figure 7.4 Output of the starter depending on location of the step, during one half-periode.

The result of the above process, as shown in Fig. 7.4 is the stepwise increment of the starter output voltage. Summarizing, the TMR0 performs continuous overflows, where the MCU PIC 18F252 perceives and counting, in all 3 phases. Then, the MCU compare the number of overflows to the number of steps creating the trigger signals of Triacs in 3 phases of soft starter.

7.3 The start time and rise ramp steps

The time for complete lifting of the soft starter ramp depends on the zero-cross pulses and the desired start time delay which is given by the user in the form of 4-bit information. The technique chosen in this study to determine the rising ramp time of soft starter is the count of zero-cross pulses. As mentioned above, the zero-cross pulses appear from zero crossing detection circuit, every 10msec (Fig. 7.1). So for example, if you want a delay 1sec should count 100 zero-cross pulses [0.01sec x 100 = 1sec]. Taking account of the foregoing that the trigger pulses correlate with the value of the current step of starter, the transition from one step to the next will be after some specific zero-cross pulses. The number of zero-cross pulses defines a Hyper-period (N) of the system, at which changes the starter step. Therefore if the user set value for a Hyper-period that are required 30 zero-cross pulses, then the total time for the complete lifting of the start ramp will be:

Total Time Interval = steps x value of Hyper-period x period of the zero-cross pulse =

= 36 x 30 x 0.01 sec = 10.8 sec

According to the information received by the MCU on starting time, require a large number of zero-cross pulses for slow step by step transition and consequently slow startup, or a small number of zero-cross pulses for quick step by step transition and consequently faster startup. Thus, the MCU perceives and counts the zero-cross pulses by increasing the content of a z-c pulse counter. The counter content is then compared to the value of Hyper-period (N), which defined by the user as information time delay. If the number of zero-cross counter pulses is smaller than the value of Hyper-period (N), no action is taken on the content of starter steps. Otherwise, the MCU reduces the step of the starter by one, resets the counter zero-cross and the above process is repeated until zeroing steps.

Summarizing, the ignition pulses of Triacs as we saw, is dependent on the outcome of the comparison between the number of overflows with the value of the current step. The evolution of starter steps depends on the outcome of the comparison of the counter z-c pulses with the system reference input, which is the determination of Hyper-period (N) value. This value indicates the zero-cross pulses that required for the next step transition.

7.4 The behavior of the soft starter in inductive load

During operation of the starter with inductive load is observed the conducting state of Triac beyond changing the polarity of the alternating supply voltage as discussed in section 4.4. In the hatched area of Fig. 7.5 is shown the non-quenching of semiconductor due to the inductive current.

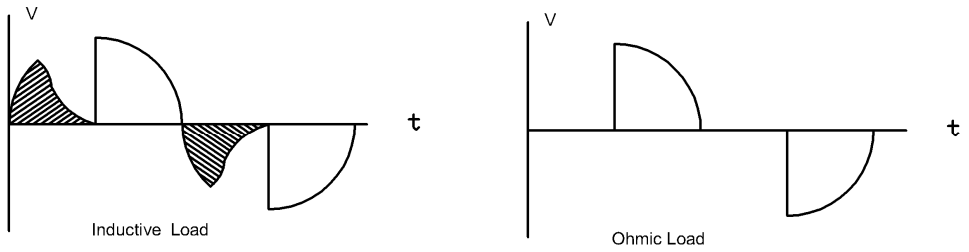


Figure 7.5 The output of the starter in inductive and ohmic load.

The result of this phenomenon is the output voltage recovers faster its nominal value, consistently the start process take place faster. In other words, the start of the load takes place shorter than the time frames set by the user. In the case when the starter has at its ends a purely ohmic load the relative number of step - output voltage is linear. Namely for 36/2 = 18 step the output voltage gets the value 230/2 = 115 V. Because of non-quenching of the semiconductor, in the inductive load, the output voltage shows a higher value than it would had the starter with an ohmic load. The voltage at the starter output corresponds to a smaller step with inductive load. The starter runs ahead the starting process so that the output voltage takes its nominal value before running the start time witch setted by the user. The phenomenon of early starting is the main field examined by this thesis.

7.5 Tackling the phenomenon of early starting

The way it is suggested to deal with the phenomenon of the early starting, is the appropriate redefinition of the firing angle, that the start time and the output voltage of the system have a more linear relationship.The effect of conducting state of the semiconductor after changing the polarity of the alternating voltage, still remains, but the change of step is dependent not only by the value of Hyper-period (N). The basic idea is to inform the MCU from the voltage appearing on the non-quenching of the semiconductor within the limits of the zero-cross pulse, and the upcoming firing pulse measuring the value of the hatched area of Figure 7.6 with the help of an ADC.

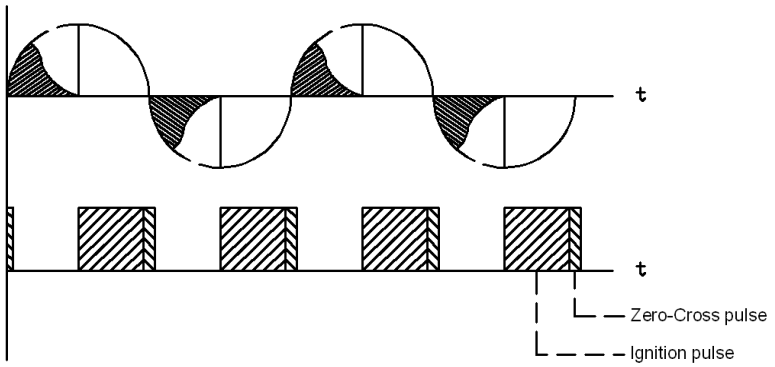


Figure 7.6 The output due to non-quenching of the semiconductor.

In its memory, the MCU will has a Look Up Table (LUT), which assigns the value of the voltage measured in number step (Appentix 1). If for example the value of the measured voltage corresponds to 8 steps and the ignition of Triac performed at the 18th step, the starter output voltage corresponds to 18-8 = 10th step. So the next ignition according to the voltage appearing at the output of starter should be made in 10-1=9^{th} step and not at 18-1 = 17^{th} step. In this way, a transition delay is achieved to the next step, which is equal to the difference between the value of the current step and the step corresponding to the measured voltage (m) times the value of the hyper-period (N).

Next Step Delay=m x N

On the other hand, abrupt flight of the 17th step in the 9th may cause high current so the starter to lose his character. Based on what has been mentioned above, it is proposed to incorporate a fuzzy logic system so that the setting of the next step will be done in a intelligent and efficient way.

7.6 The operation of the starter with the fuzzy Logic

The basic idea is based on comparing the current step starter with "fuzzy step" set by fuzzy logic. Initially, the starter will operate and calculates the step of ignoring the form of the voltage at its output (like it had at the ends an ohmic load). If during the startup, the soft starter realizes not switching of semiconductor, activates fuzzy logic to determine the "fuzzy step". The system then proceeds to compare "fuzzy" and current step and estimates the outcome. If the "fuzzy step" is less than current step then the ignition angle of Triacs not changed. In the background the starter change its current step according to the value of hyper-period (N), but its output remains unchanged. When two quantities are equal, the ignition angle changes according to definetion of the "fuzzy step". This has as result to vary the voltage output of the starter. In this

way is succeeding in increasing the output voltage in the time frames set by the user so as not to have abruptly outbursts current absorbed by motor. With regard the weighted steps of the voltage determined from the semiconductor ignition angle are not chanched linear and depends on the inductive character of the motor.

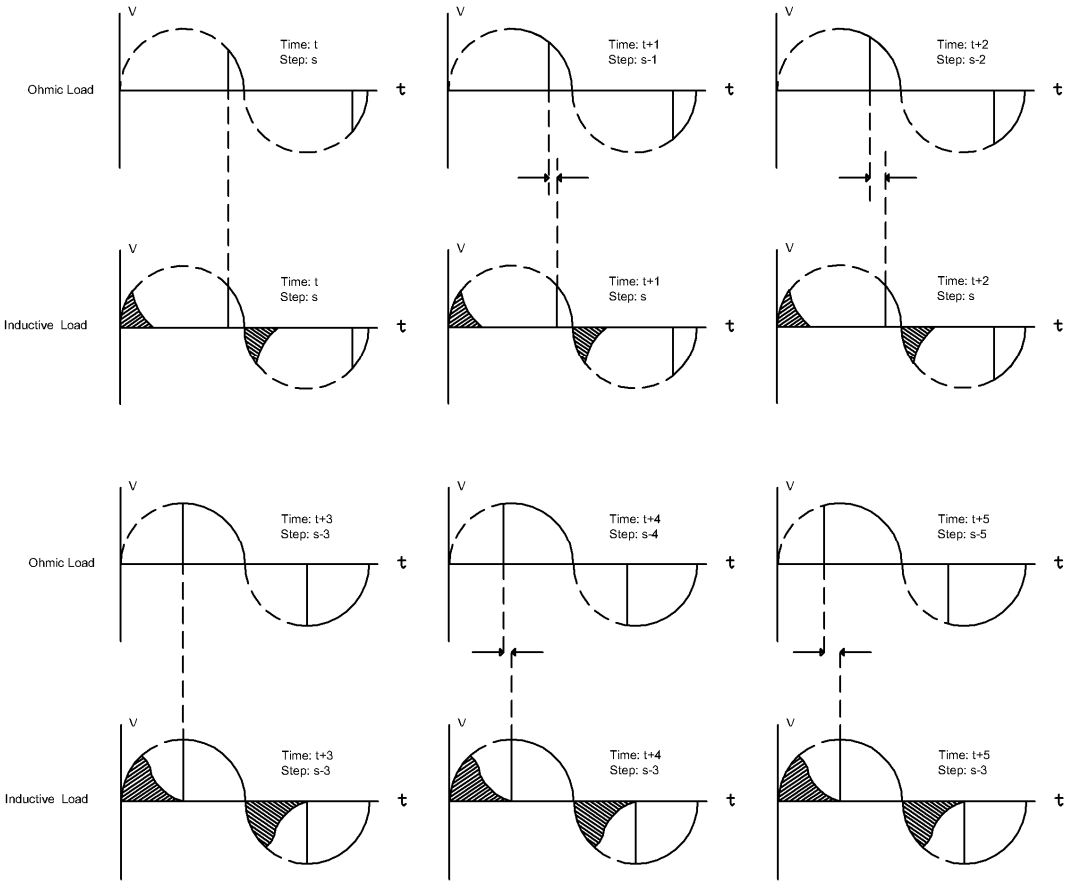


Figure 7.7 The proposed evolution of steps.

As reported the system must know the non-quenching of semiconductor. It also needs to know the value of the voltage appearing on the non-quench area, (hatched area in Fig. 7.7) which is caused by the inductive load behavior.

The determination of the voltage value is done by using a channel of A/D from PIC18F252 MCU. After the zero-cross pulse, A/D is activated and performs consecutive conversions each time overflows timer TMR0 (Fig. 7.8). The result of conversion is collected until the upcoming semiconductor ignition pulse. Because the 3~ induction motor is a 3~ uniformly distributed load, the above process is sufficient to only one of the three phases.

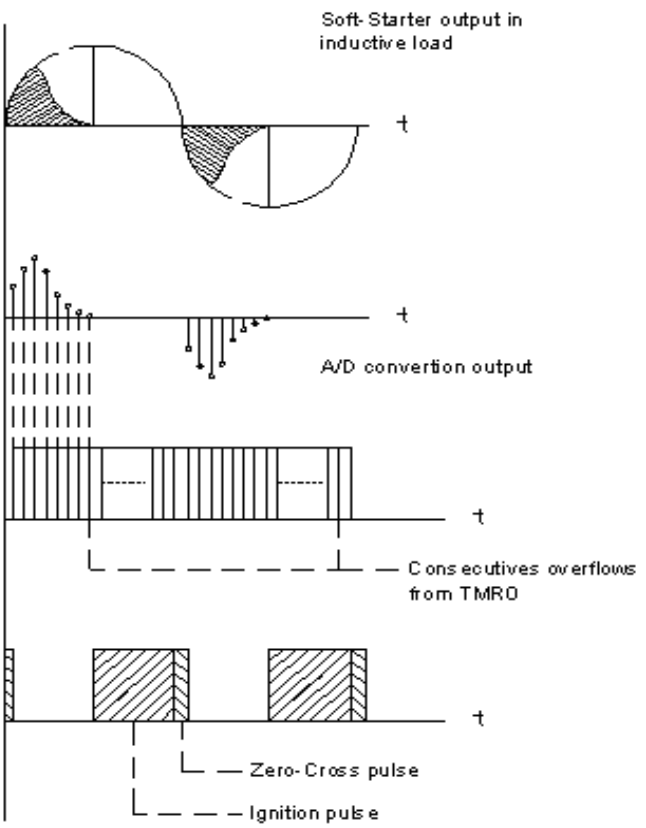


Figure 7.8 The determination of fantastic step from residual voltage using ADC.

The estimated value of the conversion (which contains all the conversions of the ADC) is matched to a number of steps with the use of a Look Up Table (LUT) which is located in the memory of the MCU. The number of the steps that come from the LUT "fantastic step" is input to the fuzzy system and activates, if the value is less than the current step of the starter.

7.7 The realization of fuzzy system

The purpose of the fuzzy system is to determine the next step of weighted output, taking as inputs the current step of the starter and the "fantastic step" that came from the LUT.

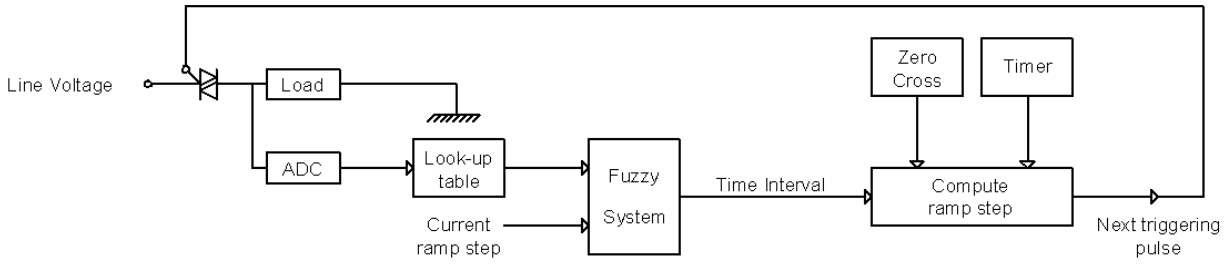


Figure 7.9 Block diagramm of the soft starter.

The fuzzy system will operates with three rules and membership functions, (in inputs and output) will have a triangular form [16]. The fuzzy system takes two inputs. The real or current step of starter and the "fantastic step" determined by the voltage at the output due to non-quenching of the Triac. The output of the fuzzy system will determine the next transition step, which will take place firing the semiconductor.

The inputs because they have a clear value will be entered into the system as a fuzzy singleton in the interval [0,36] (i.e. in space all steps).

The rules of the fuzzy system are listed below and use the inference mechanism (operator) max-min of Mamdani [16].

RULE 1

IF the real step has the value X **AND** the "fantastic step" is big **THEN** the ignition angle will change after a few steps.

RULE 2

IF the real step has the value X **AND** the "fantastic step" is medium **THEN** the ignition angle will change after several steps.

RULE 3

IF the real step has the value X **AND** the "fantastic step" is small **THEN** the ignition angle will change after many steps.

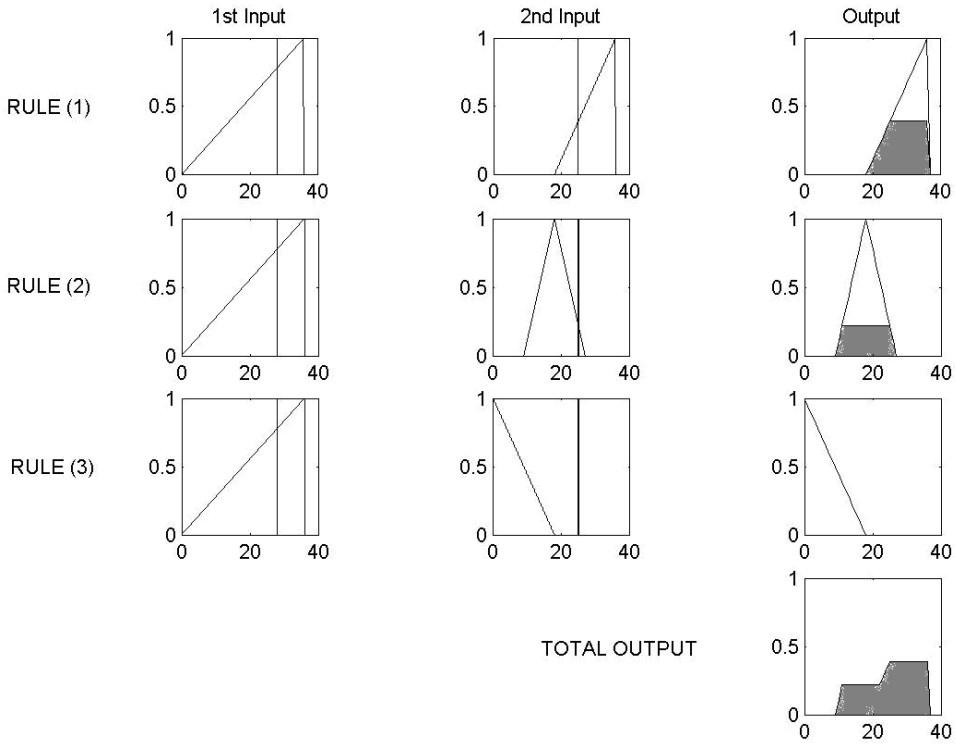


Figure 7.10 Graphical representation of the fuzzy system.

The output generated by the activation of the rules has fuzzy value. By using defuzzyfier Center Of Area (COA) [16] output has a crisp value and the system extract the step which will change the semiconductor ignition angle. The starter in the background counts and reduces the steps in accordance with the value of hyper-period (N), but its output is not changed. Continuing the above syllogism firing angle in Tiacs will change when equalize the current step with the step defined by the fuzzy system "fuzzy step". Result of this process is to vary the weighted output voltage step of starter, without changing suddenly starting current. Finally, an additional technique is introduced to ensure the smooth progress of the weighted steps of the output voltage and starting current without additional rules contained in the system is the following relationship:

$$"fuzzy\_step" = "fuzzy\_step" + \frac{current\_step - "fuzzy\_step"}{2}$$

So the definition of "fuzzy step" with the above relationship does not burden the MCU with additional computational load as to require additional RAM memory for storing membership functions.

7.8 The transition to the By-Pass situation

Addition to soft starting which will perform starter, common tactic is after the completion of the startup phase, to activate a relay which is connected in parallel with Triacs that the total load to be controlled by it and not by the provisions of power electronics (Fig. 7.11).

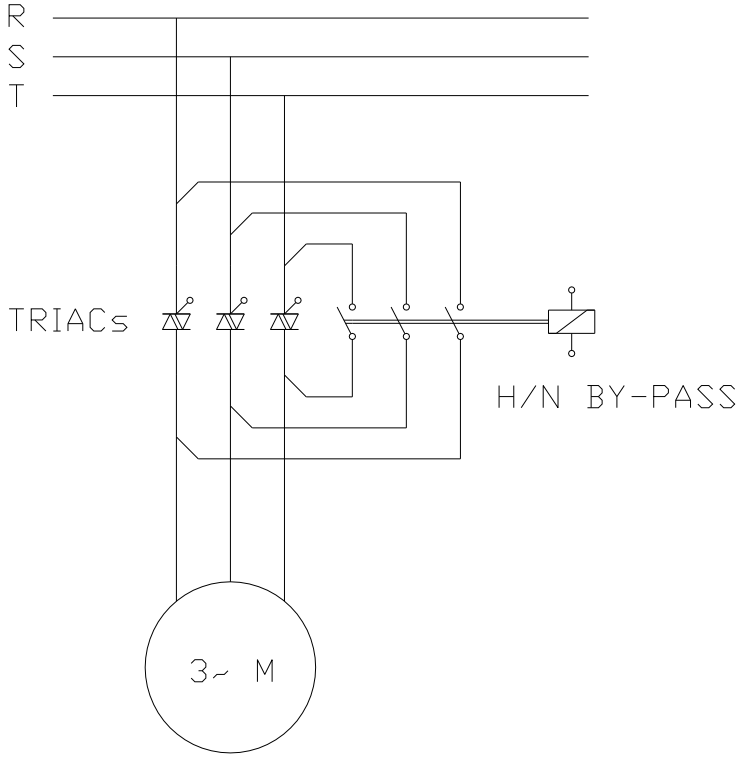


Figure 7.11 Power circuit of the soft starter connected with external bypass relay.

The purpose of this tactic is to avoid electrical stress of semiconductors due to high currents to reduce losses occurring in the form of heat, increasing their life span and ensure the operation of the motor independently of the gate signal to each Triac. The transition from start-up to normal operation of the motor takes to make the microcontroller PIC 18F252. When the MCU realized that the current step is set to zero then it will give a signal arming the relay By-Pass. The relay arm and for a few time Triacs remains in conduct.

After this delay, disabled the trigger signals from Triacs so do not conduct and appear as off. This deactivation delay of semiconductors is made for restoration purposes the of response relay.

7.9 The flow diagram of the main program

The program of the MCU PIC 18F252 reflects the main control loop as described in the preceding paragraphs, which runs continuously and composed of a set of subroutines, each of which performs a specific job. The control loop is the basic routine of MCU. From this begins all control processes by subroutines available. Its purpose is to perform the start-up process and then pass the system in the state By-Pass. Additionally manages all external siglals concerning the operation of the soft starter. The following steps describe the basic processes that make up the algorithm:

Level 1

At first performed initialization of the ports and all those units of MCU will be activated for the start of the scheme doing simultaneously the required diagnostic tests concerning the operation of the MCU. Additionally the system is preparing for the start of its operation, for zeroing and setting all the involved program variables.

Level 2

Then the system enters to a standby mode until the button START is activated by the user to operate the soft starter. The purpose, besides the above, is to perceive the zero-crossing at a specific phase in order to have a conventional sequence that activates the timer TMR0.

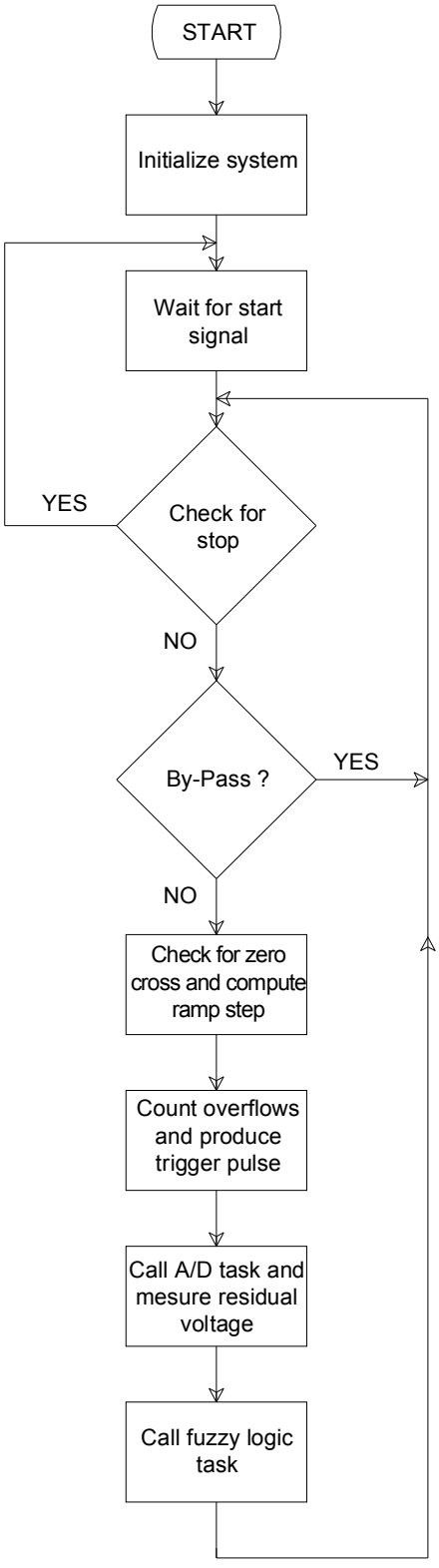


Figure 7.12 The flow diagram of the main program.

Level 3

After that, the program flow goes to the stopping system control where puts it off. Its purpose is to perceive those stimuli that require the turn-off of the starter operation. This handling enforced by the user, allows shutdown signals via the button STOP, both during startup and during normal operation.

Level 4

If stop signal is not given, algorithm passes to control phase transitions in the By-Pass state. Now the program assumes responsibility for the transition of the system from the process of soft starter in continuous operation. So after exhausting all steps of raise ramp, the MCU activates a bypass relay disables the trigger signals of the Triacs. As result the load is transferred to the power relay without participating in the circuit the semiconductors.

Level 5

The algorithm passes to the next stage when not given a signal to the By-Pass relay, listing the zero-cross pulses in the three phases of the network, so that the system knows when performed zero crossing of the AC voltage in each phase. The aim now is to inform the appropriate variables, to be controlled efficiently the trigger signals of Triacs and managing the overflows. Also at this point determined the step of lifting ramp of starter taking account the value of "fuzzy step".

Level 6

The next stage in the evolution of the algorithm is to control the timer overflows to update overflows counters for each phase. Depending on the number of overflows which have been made, the MCU outputs trigger pulse to Triacs, targeting the appropriate firing angle after the comparison with the step in which the starter is located. At this point the system performs a number of processes (multi-tasking), which aims to enable or interrupt the ignition signals in Triacs according to the counter overflows each phase, the current step and the existence of zero-cross pulses.

Level 7

Thereafter to determine the value of the feedback signal from residual voltage activates the A/D device. From consecutive conversions are carried out until the pulse ignition, estimated the "fantastic step", determined by the voltage at the output for non-quenching of Triac with the help of a LUT carrying in the memory of the MCU.

Level 8

Finally, the value of "fantastic step" with the current step entering as inputs to the fuzzy system incorporated in the program of MCU. The "fuzzy step" set by the fuzzy logic is an important parameter that determines the duration of the system's start ramp.

Until the start-up process completes the algorithm runs continuous from Level 3 to Level 8. After the completion of starting the soft starter keeps stimulated the bypass relay and expects flag state to stop the operation.

When it recognize the flag state from the button STOP, the flow of algorismou passes at Level 2 and expects START signal.

7.10 The program of microcontroller PIC 18F252

The application program that enrolled in the PIC 16F84 with necessary comments presented in Appendix 2.

8 Analysis and presentation of the soft starter circuit

The soft starter is an ac electronic converter (AC/AC), the process of conversion is automatically adjusted and it takes some time. The most crucial element, as shown in the drawing of soft starter, located in Appendix 3, is the microcontroller PIC18F252 that receives process and gives all the information needed to control Triacs in each phase. As a unit, soft starter is composed of different circuits each performing its own work. The purpose of all the circuits is the cooperation and their contribution to the proper functioning of the starter.

8.1 The power supply circuit

The soft starter consists of a power supply circuit, which reduces the AC main voltage and converts it to a continuous voltage. The power supply circuit outputs the stabilized voltage 5 V DC to supply the MCU and other control and display circuit (Fig. 8.1).

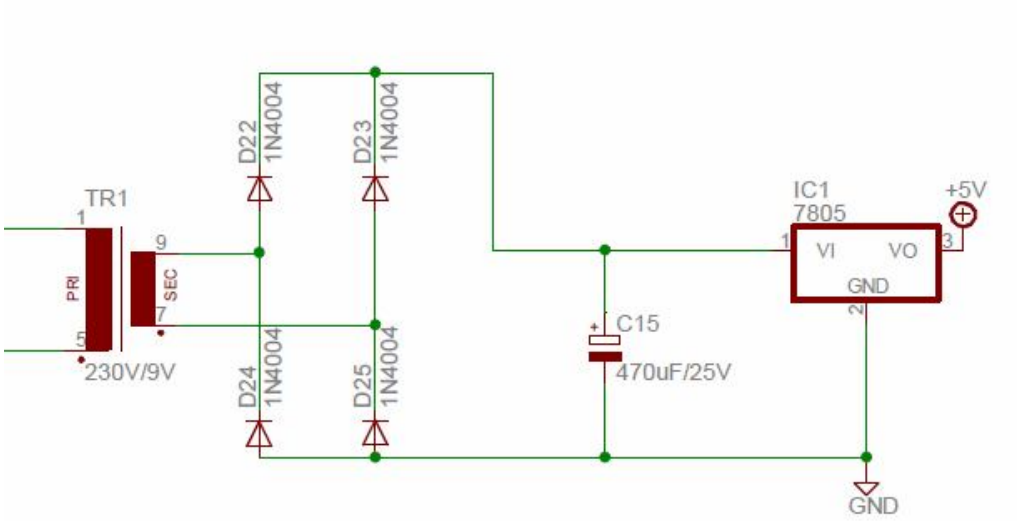


Figure 8.1 Power supply circuit of the soft starter.

8.2 The zero-crossing detection circuit

As mentioned in the previous chapter to synchronize triggering signals of Triac's required to know the MCU those points, that the alternating voltage is zeroed. The circuit in Fig. 8.2 shown below, is the transit detection unit from zero in each phase.

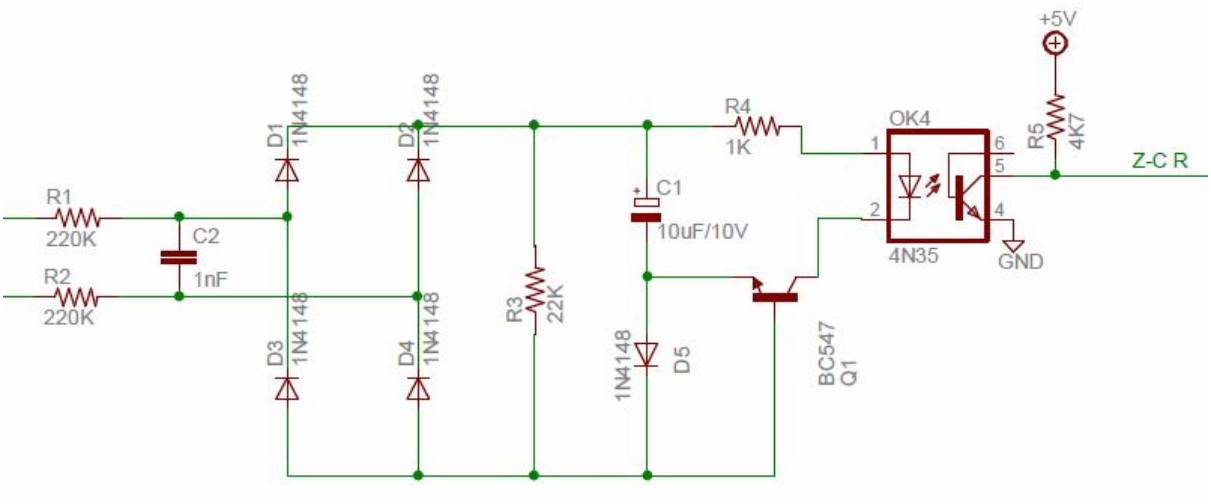


Figure 8.2 The zero – cross detection circuit.

The creation of zero-cross pulses is performed by optical coupling to avoid galvanic connection of the three line voltage with MCU. Also, in this way, the zero-cross signals are not affected by the noise which may enter from the supply lines in the system.

8.3 The circuit of the microcontroller PIC 18F252

The circuit of the MCU PIC 18F252 illustrated in Fig. 8.3 is the most important part of the soft starter. This circuit is responsible for the handling of the synchronization signals. After the necessary processes, output signals are produced for the control of the system.

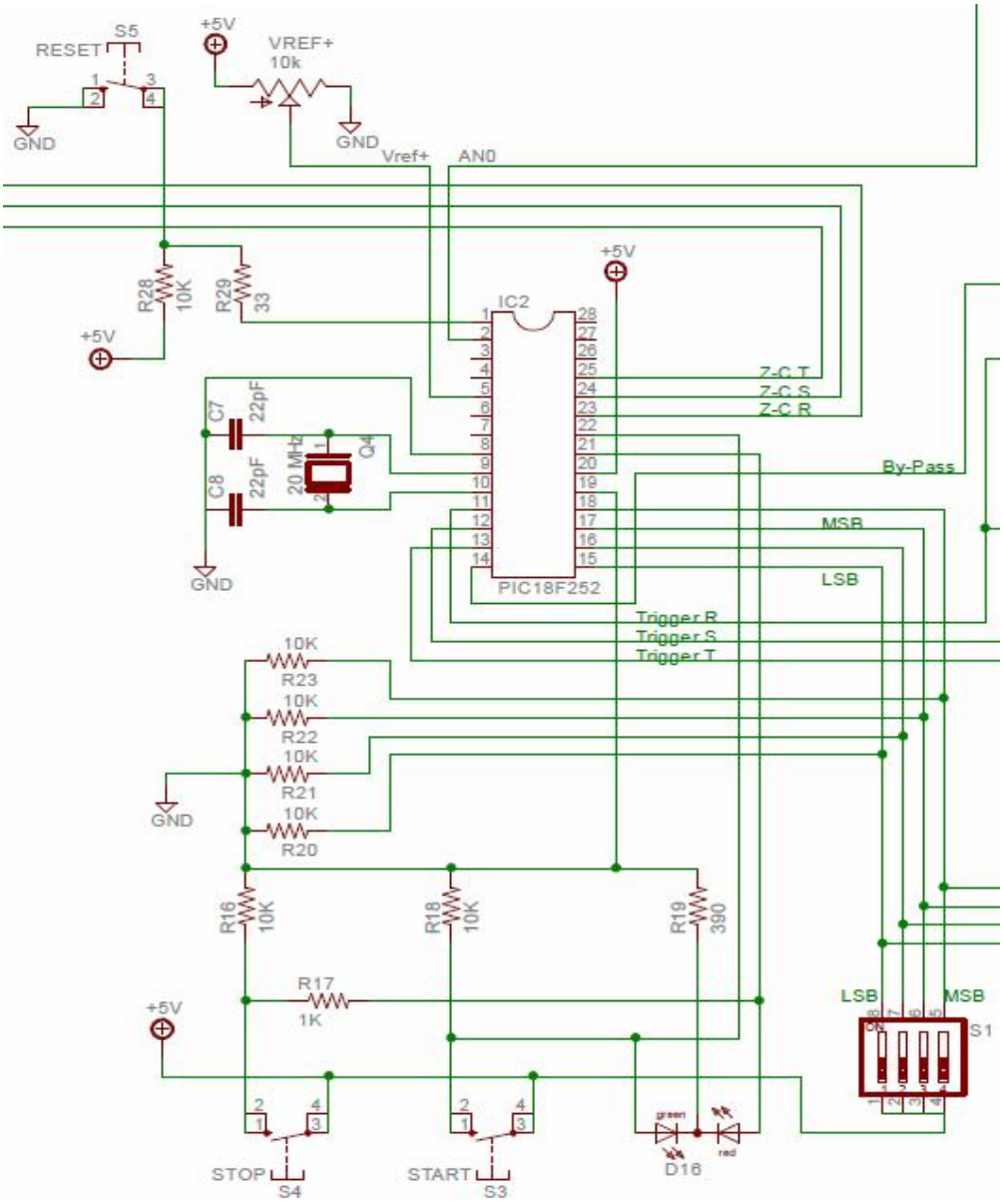


Figure 8.3 The PIC18F252 circuit and the pin connection terminal.

The transmission of all signals is performed by the 5-bit port A and 8-bit port B and C of the MCU. Apart from the zero-cross pulses of the three phases needed for synchronizing the system, there is a 4-bit signal information via the DIP switch, which indicates the rise time of the start ramp. Finally starting and stopping of the soft starter is made by signals given from the buttons START and STOP respectively.

Also for the operation of the A/D the MCU receives a signal from the output voltage of the motor with the help of a voltage sensor and voltage reference. The output signals managed by the PIC 18F252 is the trigger pulses on Triacs in three phases and after completion of the startup process the bypass signal on By-Pass relay is activated. Except from above the MCU circuit includes the crystal clock which clocks the PIC 18F252 at 20MHz and the reset circuit of the microcontroller.

8.4 The signals driving circuit

To supply the electronic elements during their activation requires enough power. For this reason a driver circuit is needed.

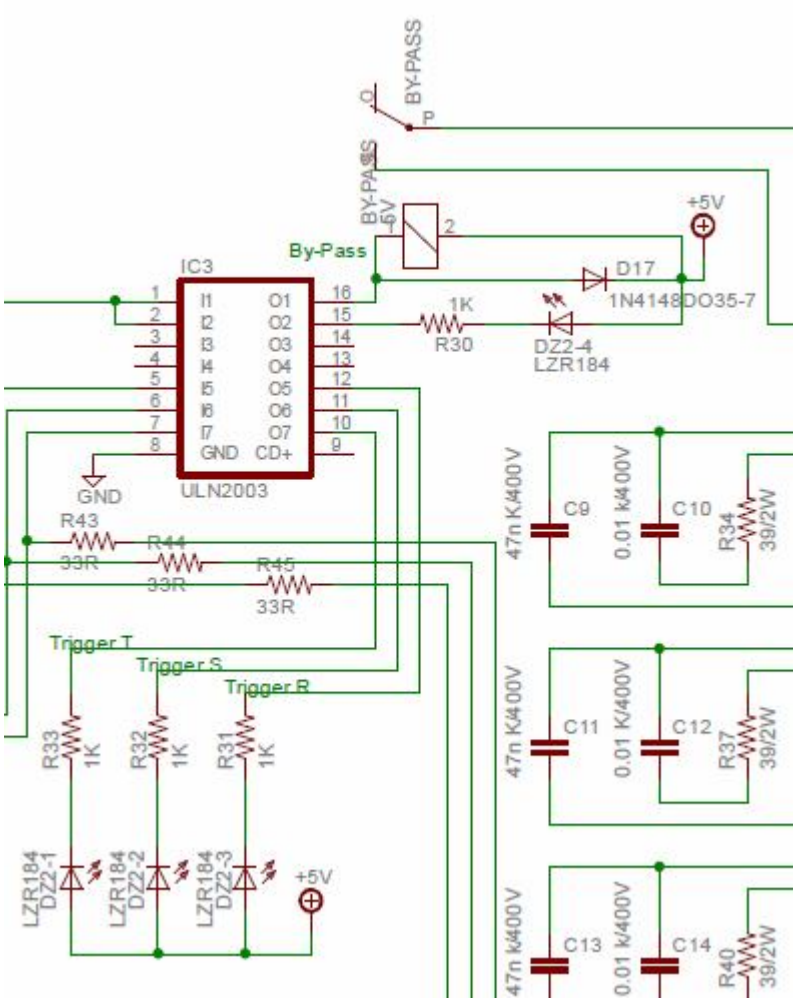


Figure 8.4 Connection of signal driver ULN2003.

In the circuit of Fig. 8.4 the IC3 is a ULN2003 Driver consisting of Darlington wiring Transistors. At the outputs of ULN2003 the relay By-Pass is connected as well as Leds that indicate the trigger signals. With the the driver overloading of the MCU output channels is avoided. Also possible damage due to high or shock currents is avoided. The driver gives the possibility to indicate the status of a signal without affecting the logic state from the indicative Leds, leaving the original signal unaffected.

8.5 The Power circuit

The power circuit of the soft starter consists of the Triacs in each phase, aimed at control the motor voltage and the bypass circuit that is activated after the soft start.

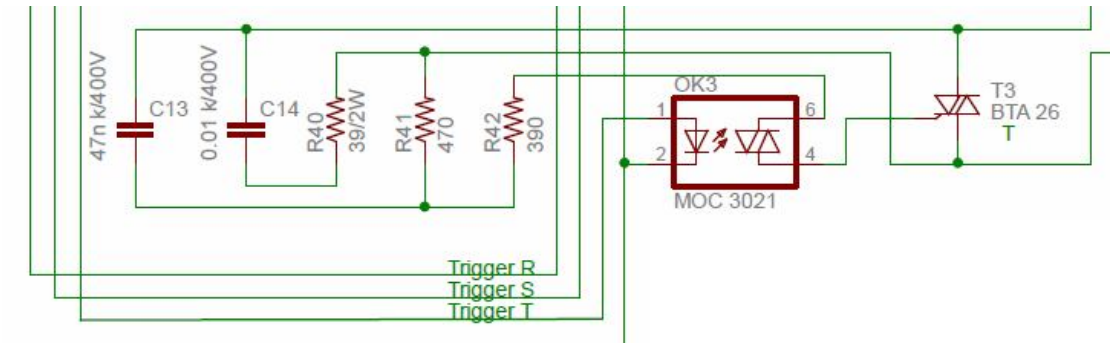


Figure 8.5 The power circuit in one phase Triac.

The circuit shown in Fig. 8.5 represents the power circuit of the Triac (in one phase). The coupling of the gate signal to the Triac becomes optically through the photo-Diac MOC3021. This also gives the galvanic isolation of the power circuit 230V / ac and the signal circuit of 5V / dc by possible step voltages and contact voltages. Additionally, the power circuit is consisted of a RC filter that provides protection to the Triac from overvoltages of network.

Switching from Soft-Starting to normal motor operation is achieved with a bypass relay. Transduction is given by micro-relay bearing the starter on its board, this in turn activates an external 3-pol power relay assumes the load of the motor (Fig. 8.6). At the same time in the micro-relay circuit connected a free flow diode designed to remove the current generated when tends to reverse the voltage of the coil.

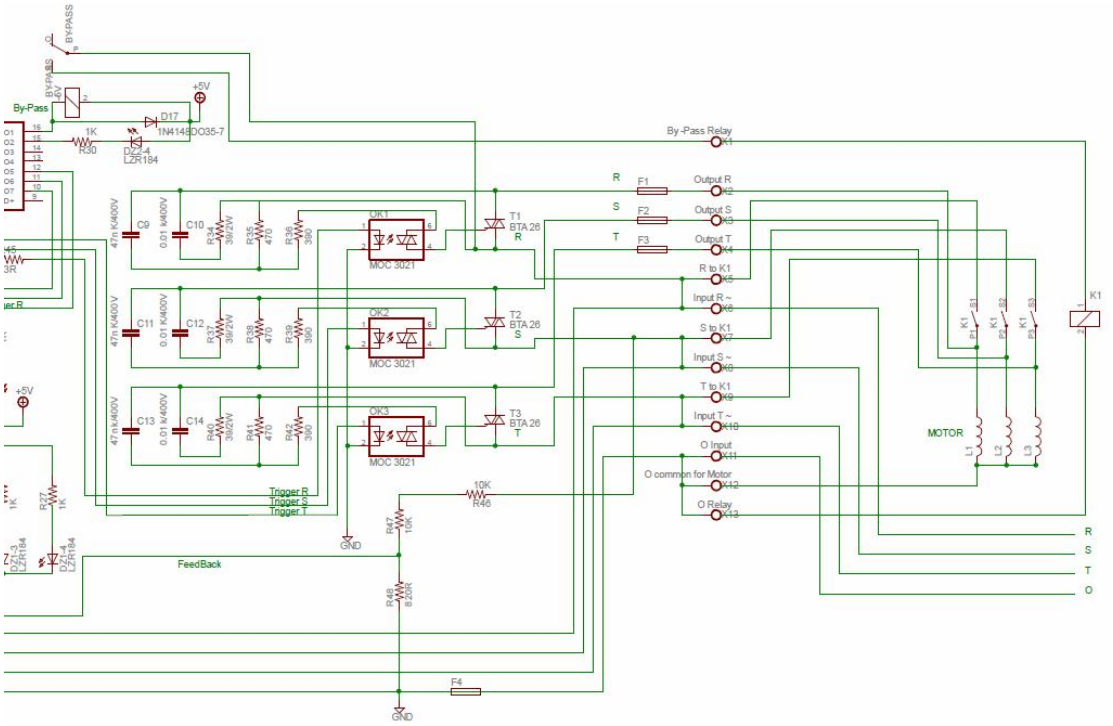


Figure 8.6 The power circuit of soft starter.

8.6 The voltage sensor

The voltage sensor performs the closed loop of soft starter feedback. The sensor receives the output voltage and transmits at the input of ADC of the PIC 18F252. It consists of a voltage divider (Fig. 8.6) which reduces the voltage to be inserted in a circuit by Operational Amplifiers. The regraded signal enters a inverting and one non inverting OpAmp. Their outputs are then composing the final feedback signal by means of two diodes. For reasons of adjustment, before the feedback signal is passed to the MCU first enters to a following voltage OpAmp. By this way there is no effected the microcontroller by the online signal because of the high impedance presents

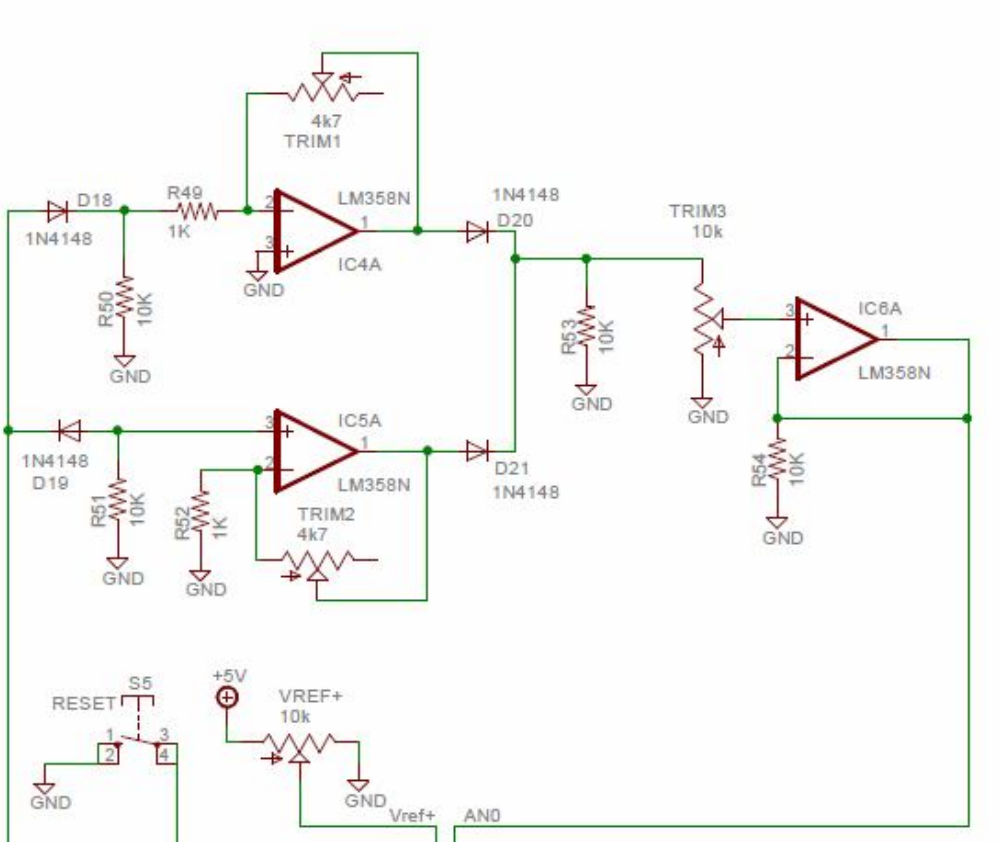


Figure 8.7 The circuit of voltage sensor.

8.7 Handling and indications

The handling of soft starter is done with the help of the button START and STOP for starting and stopping, respectively, but the duration of the soft start is determined by the binary information provided by switch S1. The user must first determine the parameter of the start time and then activate the button START. The stopping can be done both during the startup process and at the time of normal operation (Fig. 8.8).

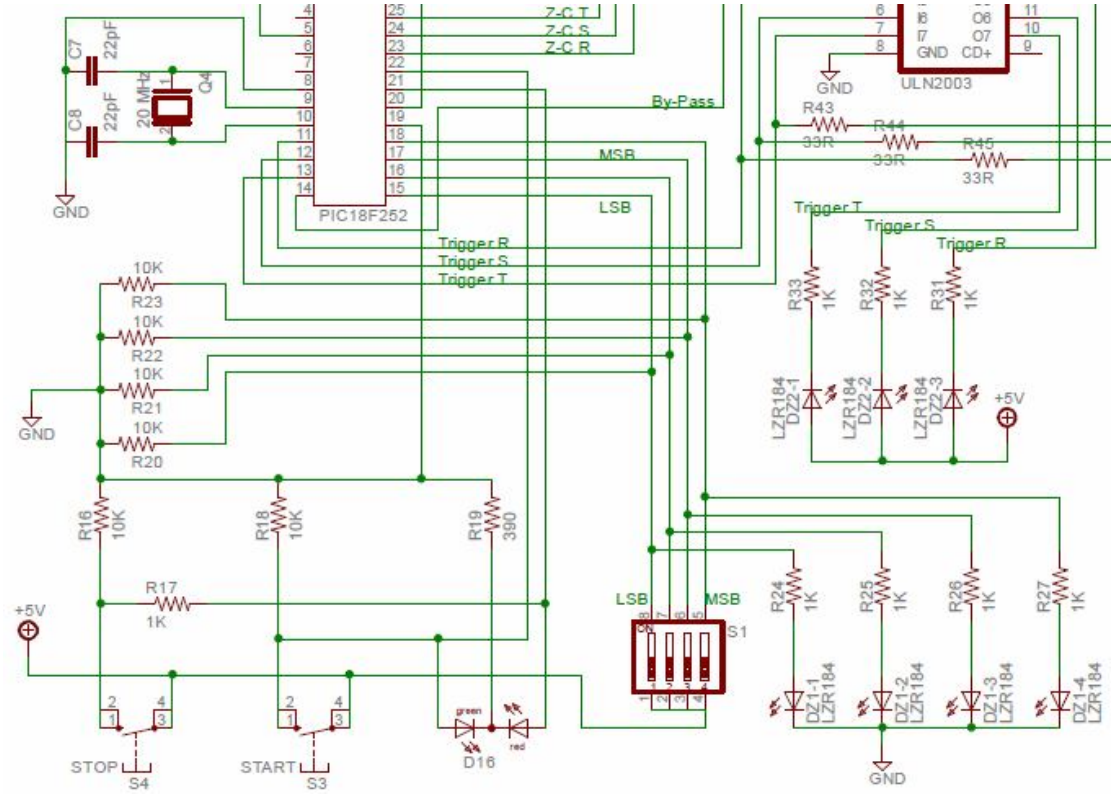


Figure 8.8 Handling and display circuits.

The function indications of the soft starter provided by Leds carrying on circuit board. The binary information of start time is illustrated in four Leds, connected to the switch S1 formed of 4-bit binary number of 0000_2 (3 sec) to 1111_2 (18 sec). Also the width modulated pulses trigger the Triacs illustrated in leds through the Driver ULN 2003. Finally bicolor Led D16 is activated when pressed one of the buttons START or STOP with green or red color respectively.

9 Measurements - Results - Conclusions

In this chapter shows the waveforms at the output of the soft starter and tested the operation and its behavior in various loads. With the help of oscilloscope also examined the input signals and output signals managed by the microcontroller PIC 18F252.

9.1 Zero-Cross pulses

As mentioned in paragraph 7.2, the zero-cross pulses appear at those points where the AC voltage zeroed. Below is shown the zero-cross pulses wave-associated with ac supply voltage.

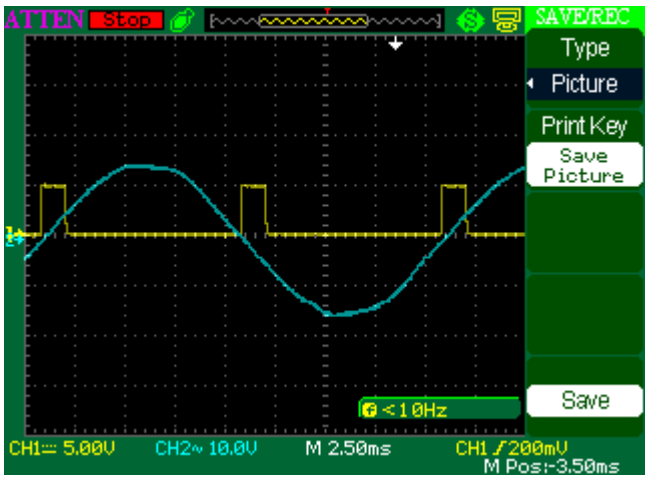


Figure 9.1 Zero-cross pulses relative to the full wave.

Figure 9.1 shows the output circuit producing zero-cross pulses duration 1.2 msec (yellow waveform) from a ac wave in first phase (blue waveform).

9.2 The trigger signal

The trigger signal is directly dependent on the zero-cross pulse. The MCU extracts the trigger signal when realized the end of the of transit through zero of the alternating voltage, considering the current step of rise ramp and the value of "fuzzy step" that extracts fuzzy system while interrupts the signal when perceive the next zero-cross pulse. Below are shown trigger pulses of TRIACs at different firing angles associated with zero-cross signal. It is obvious that the ignition angle of Triac's with the duration of the trigger signal is coincided and has a direct relationship. In Fig. 9.2 below is shown a trigger signal (yellow waveform) which zeroed when the MCU perceive the existence of zero-cross pulse (blue waveform). With the

help of an oscilloscope measured the width of the trigger signal and found equal to 700μsec. This value corresponds to a large semiconductor ignition angle.

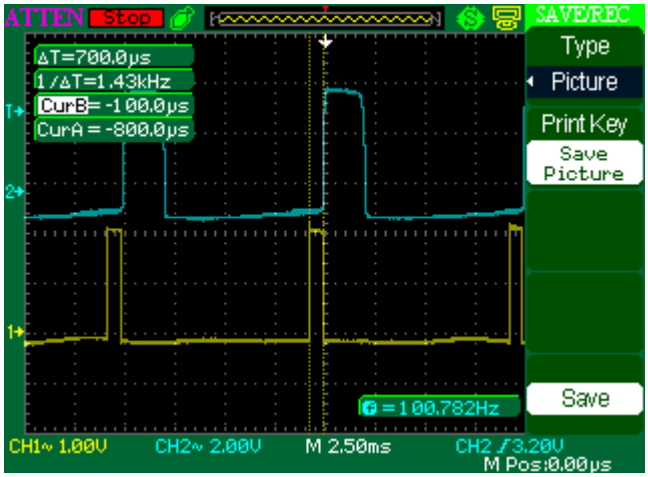


Figure 9.2 Trigger pulse width 700μsec.

Next in Fig. 9.3, is shown a range where the trigger signal is 6.70msec corresponding to a high ignition angle of the semiconductor.

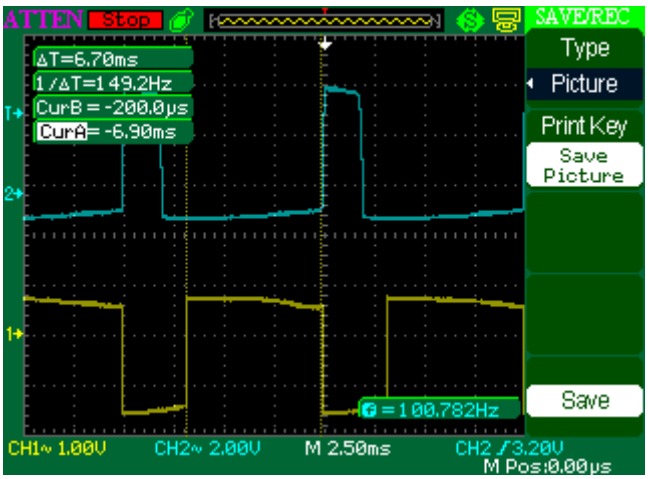


Figure 9.3 Trigger pulse width 6.70msec.

As we observe the triggering signal range varies according to the value of the step of soft starter. This is therefore a pulse width modulated (PWM), which range depends on the result of comparing the number of overflows with the value of the current step while the evolution depends on the result of current step comparison with "fuzzy step". So by changing the MCU the pulse width is changed the angle of triggering of the semiconductors in turn changing the output voltage the starter.

9.3 The feedback signal in A/D

As mentioned above, the voltage sensor transfers the system feedback signal to the MCU. In Figures 9.4 and 9.5 below shows the output voltage of the sensor at different times in the startup process, when the ends of the soft starter carries inductive load. The system feedback signal enters in A/D of MCU that with its assistance is estimated the "fantastic step". Because the A/D only accepts the dc voltage, feedback signal has a full rectified form with positive values only.

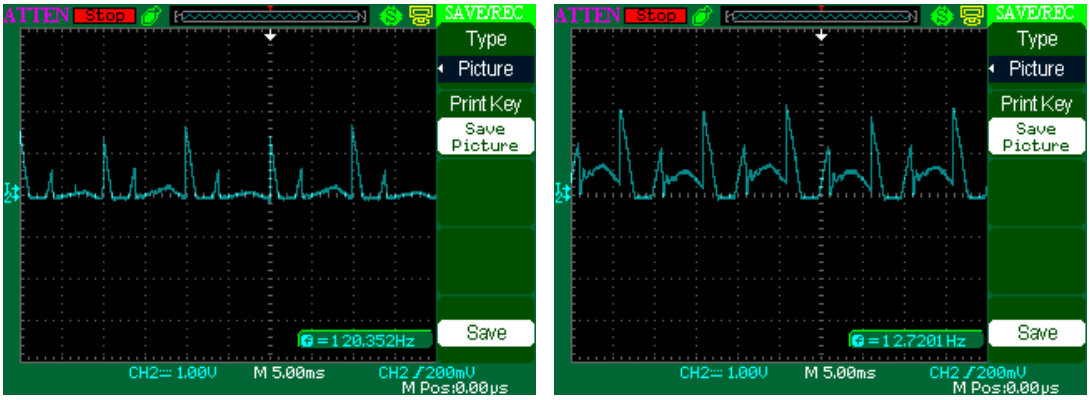


Figure 9.4 The feedback signal to A/D at the beginning of the startup process.

In figures clearly is discerned the area of non-quenching of the semiconductor from the beginning of each half-periode by the time ignition of the element. This range provides the necessary information for the starter adout inductive behavior of the load and it is an input in the fuzzy logic to determine the "fuzzy step". When run out all the steps of the starter, the output takes the form of a full sine, but due to limitations of the functions in the feedback system, the feedback signal is presented as a full rectified wave.

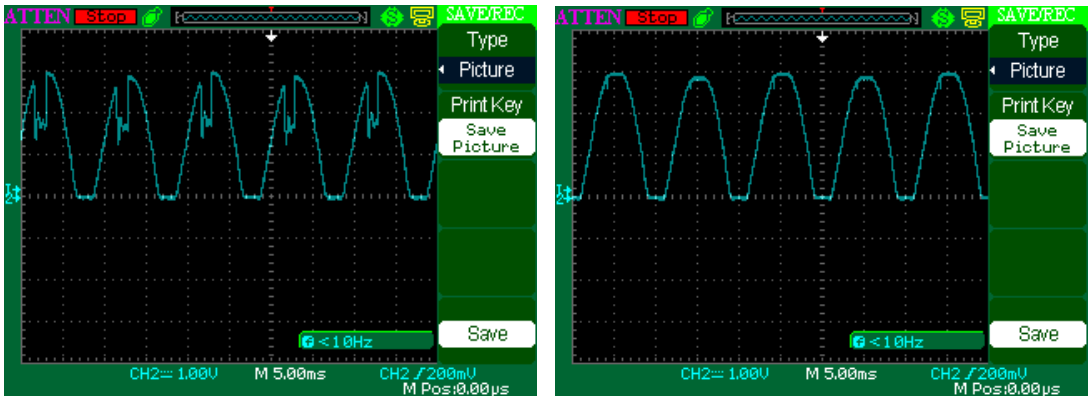


Figure 9.5 The feedback signal to A/D at the completion of the startup process.

9.4 The output of the soft starter

Below the response of the soft starter is examined in 3~ inductive and ohmic loads observing the operation with or without the effect of the fuzzy logic.

a) Ohmic load

In figure 9.6, the output of the system is shown (blue waveform) at the earliest stage in the startup process, in relation to trigger signal (yellow waveform) providing by MCU. The trigger signal has a small width when the semiconductor ignition performed at a high angle. Therefore the rms value of the voltage at the output of the starter is low.

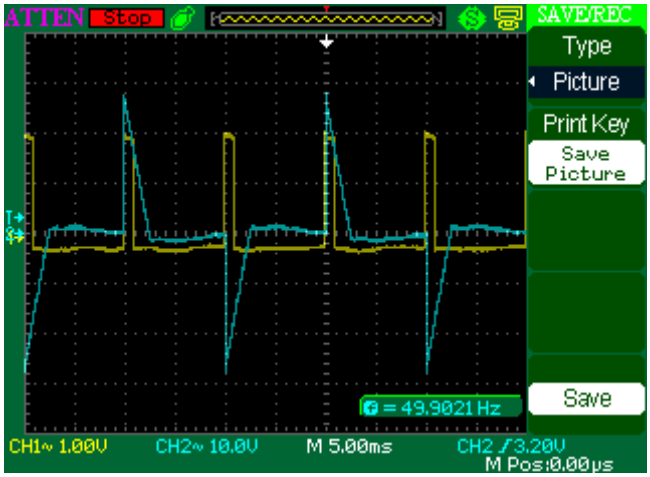


Figure 9.6 Output of the starter with a large firing angle in ohmic load.

In figure 9.7 below, the output of the system is presented (incorporating fuzzy logic), in relation to trigger signal of the MCU having a greater range. The semiconductor ignition takes place at a larger angle, and hence the rms value of the voltage on the starter output is at a higher level.

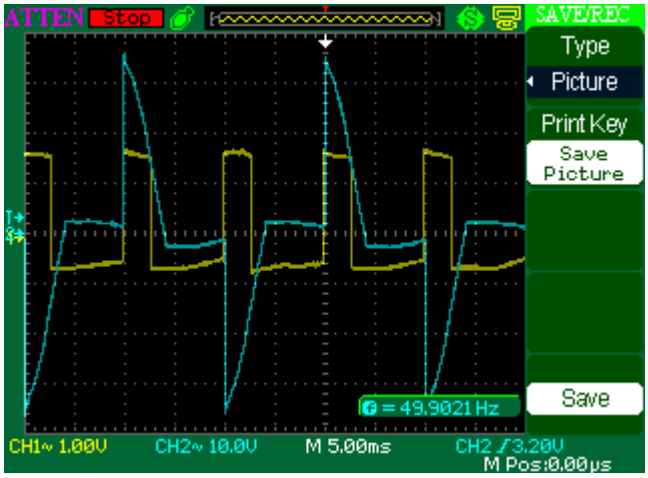


Figure 9.7 Output of the starter with a small firing angle in ohmic load.

On the above measurements screenshots which were presented, observed that the fuzzy logic is not effect the starter output when a purely ohmic load is on its end. The fuzzy logic does not act because of the non-appearance voltage caused by non- quenching of the semiconductor resulting in the operation of the starter to "bypass" the fuzzy system.

Moreover, discontinuation of the signal trigger becomes at those points that the alternating voltage tends to zero. Namely the appearance points of the zero-cross pulse. Measurements below relate the input voltage of the starter (blue waveform) with the output voltage (yellow waveform), in ohmic load. In figure 9.8 is observed that the system output voltage having a value that corresponds to a small proportion of the supply voltage due to the high firing angle a1.

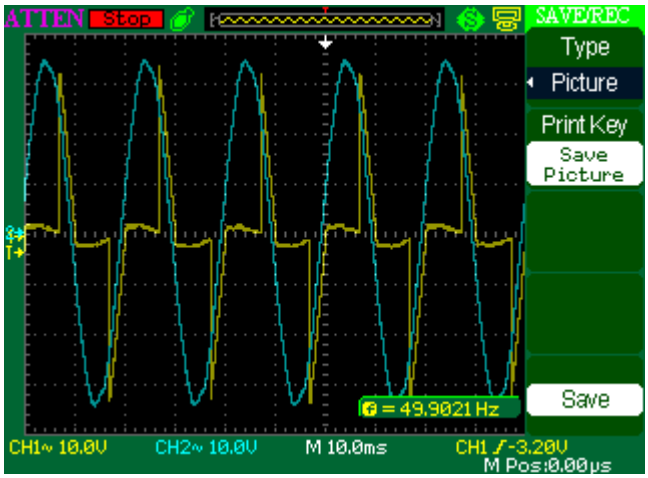


Figure 9.8 Output of the starter compared with the input voltage in an ingition angle a1 in ohmic load.

However, if the semiconductor turn-on at an angle a2 which is smaller than the angle a1 (ie. a1> a2), the system output voltage corresponds to a large percentage of the supply voltage, as shown in Fig. 9.9.In this way is achieved a gradual increase in the current flowing through the load in a linear fashion, according to the growth of the starter output voltage.

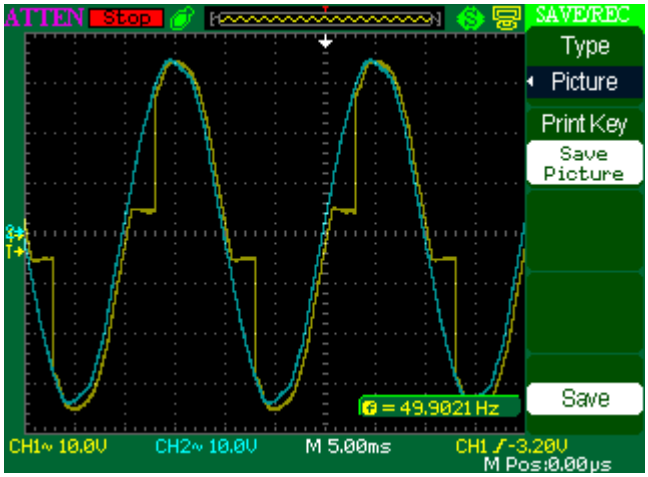


Figure 9.9 Output of the starter compared with the input voltage in an ingition angle a2 in ohmic load.

Conclusions:

The tests concluded that the output of the soft starter obeys the trigger signals that extracts by MCU with no exceptions to the output. The fuzzy logic does not affect the operation of the soft starter, because the ohmic nature of the load. The greater the width of trigger pulse, grow, the value of the output voltage having thereby reduced "smoothly" the firing angle of the semiconductor. Thus according to the above, this entire control logic makes the adjustment of the starting current to the load due to the increment of the applied voltage. The ignition of TRIACs is always donne in the positive trigger pulse forehead while quenching is done each time in zeroed point alternating input voltage.

b) Inductive load without incorporating fuzzy logic

In Figure 9.10, the output of the system is presented (blue waveform) at the earliest stage in the startup process, in relation to trigger signal (yellow waveform) providing by MCU. The trigger signal has a small width when the semiconductor ignition performed at a high angle. Unlike the corresponding event in ohmic load, the effective voltage starter output has a relatively high value. This is due to the inductive character of the load due to the delay in the phase angle between voltage and current. The current continues to flow beyond zeroing the voltage and this is the main reason that not turns off the semiconductor and the voltage across load is quite high.

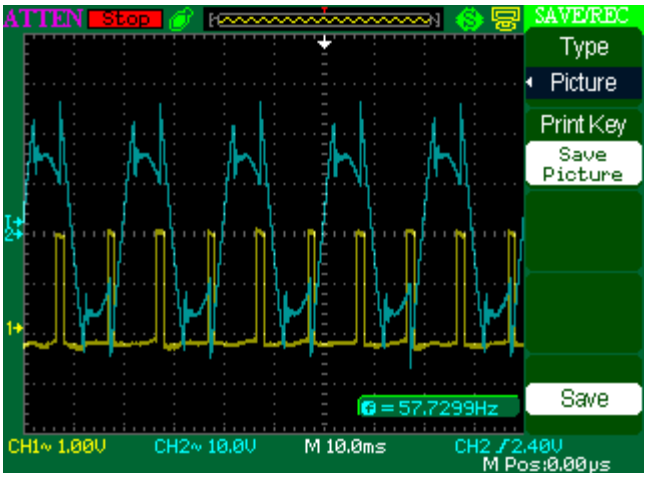


Figure 9.10 Output of the starter with a large firing angle in iductive load.

The voltage on the output of the starter due to induction growing very quickly before MCU exhausts all the steps of starting ramp. As shown in Figure 9.11 output has taken its nominal value without the trigger signal has the maximum width. The triggering of the semiconductor becomes at the point commands the trigger pulse, but because of the high inductance component in the system is lost the control of quench.

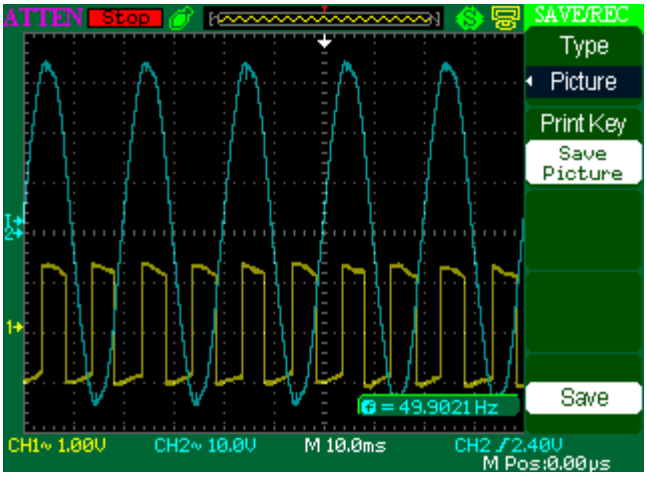


Figure 9.11 Output of the starter with a small firing angle in iductive load.

Regarding the above measurements which were presented, it observed that the quenching of the semiconductor is not made to those points where zeroed voltage. Also, the startup process is performed more quickly resulting in the current in the load, to recover its nominal value more rapidly, irrespective of the sequence of steps specified by the MCU and the system does not work efficiently. The phenomenon of early startup due to the loss of control of the quenching semiconductor is the most important problem in this case and is aggravated with increasing output voltage and turns on the motor shaft. Cause of these, as mentioned above is the inductive character of the load. The disconnection of the signal trigger becomes again at the occurrence of zero-cross pulse, but the semiconductor is still in contact.

Measurements below relate the input voltage of the starter (blue waveform) with the output voltage (yellow waveform) applied in inductive load. In Figure 9.12 is observed that the system output voltage having a value that corresponds to a large percentage of the supply voltage independently of the large firing angle a1.

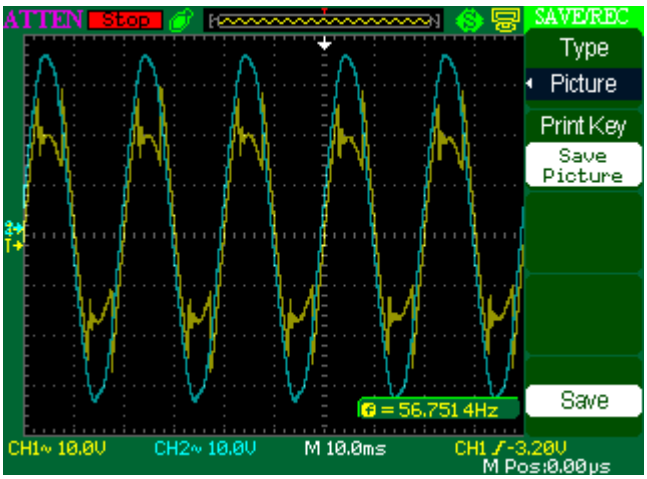


Figure 9.12 Output of the starter compared with the input voltage in an ingition angle a1 in inductive load.

However, if the semiconductor turn-on at an angle a2 which is smaller than the angle a1 (ie. a1> a2), the system output voltage corresponds almost in all of the percentage of the supply voltage, so that the output voltage takes its nominal value as shown in Fig. 9.13.

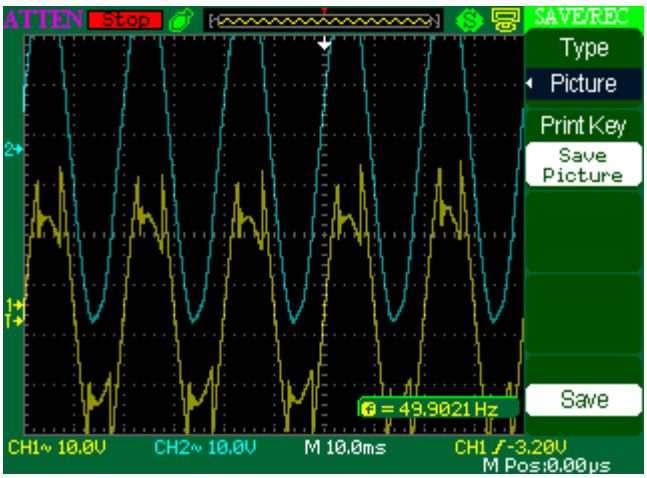


Figure 9.13 Output of the starter compared with the input voltage in an ingition angle a2 in inductive load.

The gradual increase of voltage and current flowing through the load becomes with exponential manner and very early the load recovers its nominal values before he exhausted startup time. In Figure 9.14 is observed the system output voltage in step 36-n (ie. at a sizable angle of ignition). Although the output voltage has a relatively low value the current in the load delayed to zero and that is why it presented a delay in quenching of the semiconductor.

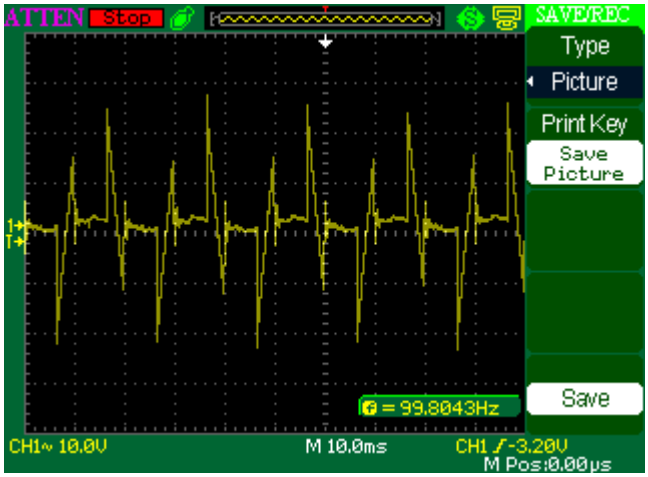


Figure 9.14 Output of starter on inductive load in step 36-n.

In the next step below (ie. at step 36- [n + 1]) the rms value of the voltage to the load is increased in an exponential manner as shown in Fig. 9.15. The semiconductor quenching no longer takes place due to induction and it is given the illusion that the soft starter depletes all steps faster than boot time originally set by the user.

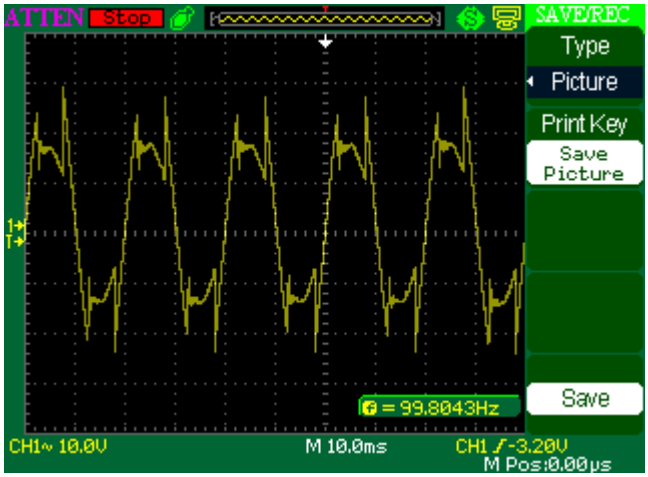


Figure 9.15 Output of starter on inductive load in step 36-[n+1].

Conclusions:

Measurements in the inductive load are quite different to those of the ohmic load. The narrowing firing angle, the control of quenching is losse in TRIACs. As mentioned the current flowing through the circuit continues beyond zero voltage due to the inductive character of the load. Is observed then, the rise ramp during startup acquires an exponential form unlike the linearity that was in ohmic load. Additional the restoration of the full sine wave voltage at the output is achieved faster than the rise time of the characteristic ramp set in microcontroller. This early "virtual" complement of steps is due to the continuing contacting of semiconductor elements, over the zeroing of the ac voltage.

c) Composite ohmic - inductive load without incorporating fuzzy logic

In Figure 9.16, the output of the system is shown (blue waveform) in the initial stages the startup process, relative to the trigger signal (waveform yellow) providing by the MCU. The trigger signal has a small width when the semiconductor ignition performed at high angle. The rms voltage starter output not so great value. This is due to the character of the load having an ohmic and an inductive component. The voltage occurs because the non-quenching semiconductor smoothed enough and has a milder form. The current continues to flow beyond zeroing voltage and this is the main reason that, not off the semiconductor and the voltage across the load is relatively little high.

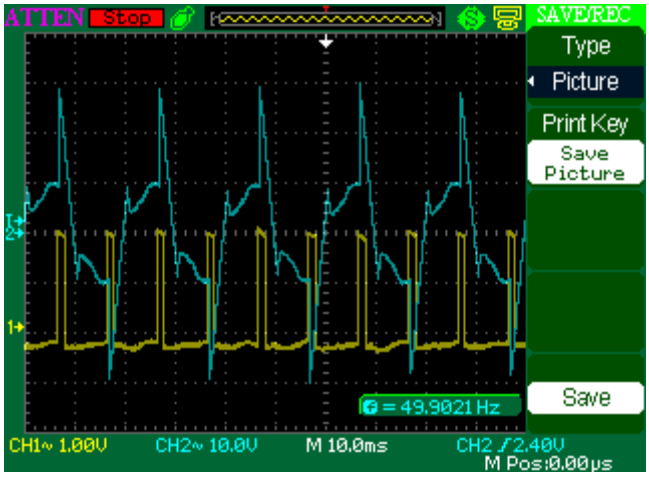


Figure 9.16 Output of the starter with a large firing angle in composite load.

The voltage on the output of the system, by reducing the value of the step of starting ramp rises fairly quickly and drags the current flowing through the load. The effect of the inductive component is grown. This has as a resulr to loose control of the quenching in the semiconductoras as shown in Fig. 9.17. Although the trigger signal has a higher width (namely greater firing angle) we observe that the output tends to be close enough to the sine wave form of input.

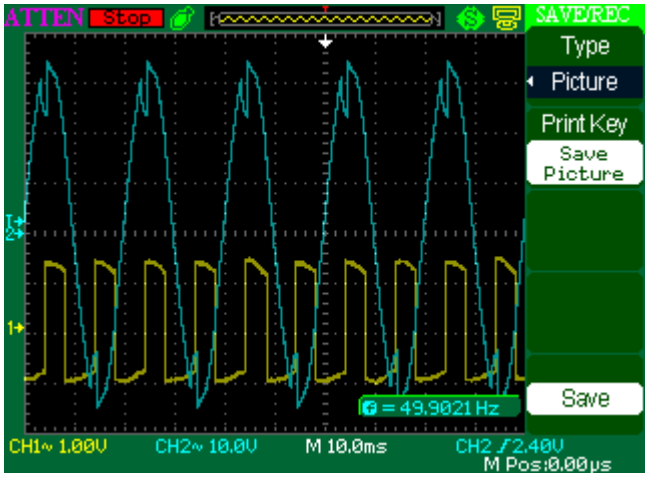


Figure 9.17 Output of the starter with a small firing angle in composite load.

In the above presented measurements, it is observed that the quenching of the semiconductor in the early stages the startup process is performed with a large delay. However, as higher the output voltage of the starter is increased and the effect of the inductive component of the composite load thus not performed quenching in those parts which zeroed voltage. The startup process occurs relatively quickly so the current in the load, recovers its nominal value before starter exhausted all steps. The trigger signal is switched-off again at the onset of zero-cross pulse, but the semiconductor is still in contact after the initial startup stages.

Measurements below relate the input voltage of the starter (blue waveform) with the output voltage (yellow waveform) applied to the composite load (ohmic - inductive). In Figure 9.18 is observed in the output voltage of the system has a value that corresponds to an average rate of of the supply voltage independently of the firing angle a1 witch is relatively large.

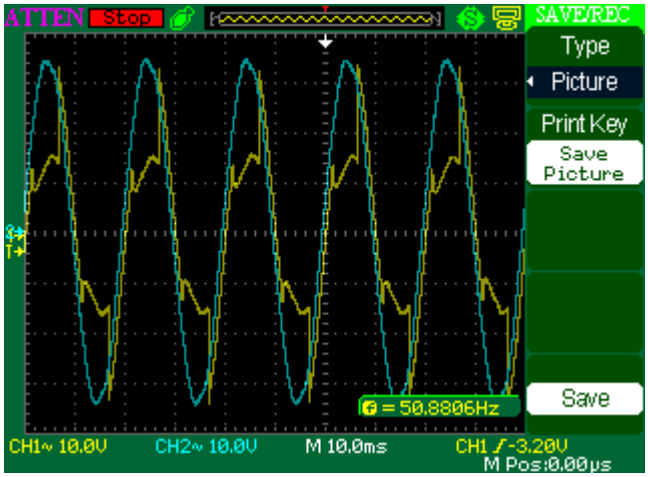


Figure 9.18 Output of the starter compared with the input voltage in an ingition angle a1 in composite load.

When the semiconductor is turned-on at an angle a2 which is smaller than the angle a1 (ie. a1> a2), the system output voltage corresponds to a fairly large percentage of the supply voltage, thus approaching the nominal value of the input voltage, as shown in Fig. 9.19.

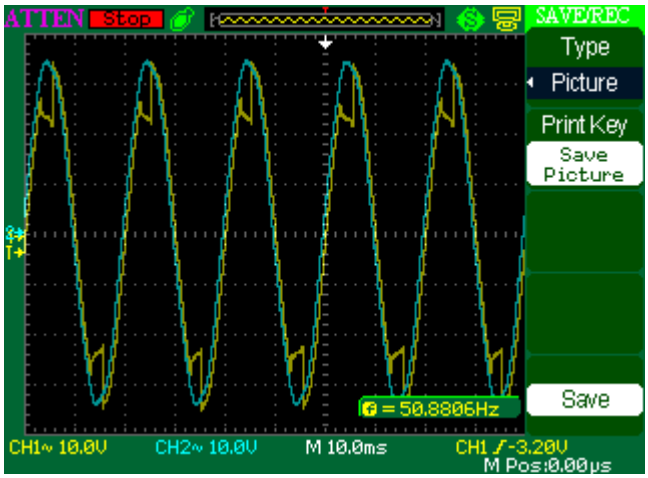


Figure 9.19 Output of the starter compared with the input voltage in an ingition angle a1 in composite load.

The gradual increase of the voltage was exponential manner in the purely inductive load, is smoothed enough in the composite load but still gives the illusion of filling the startup steps before they actually zeroed.

In following Fig. 9.20 is observed the system output voltage in step 36-n (ie. at a sufficiently large angle of ignition). Although the output voltage has a relatively low value, the semiconductor tends to be extinguished due to the relative small current load flowing.

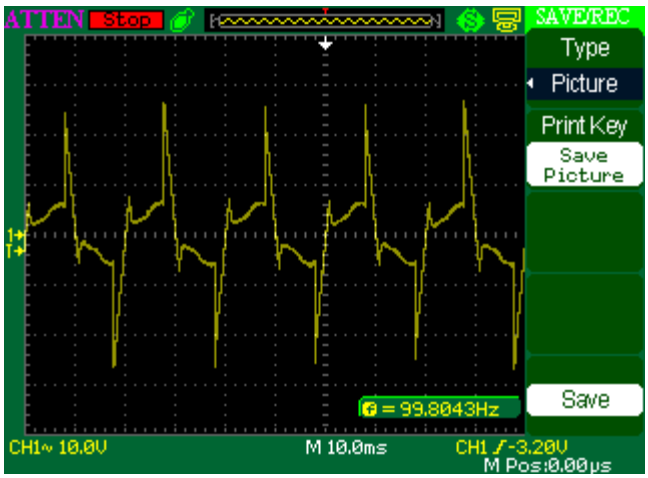


Figure 9.20 Output of starter on composite load in step 36-n.

After a few steps t that following (ie. at step 36- [n + t]) and after overtake the MCU the initial startup steps, the rms value of voltage and current to the load have an sharp rise as shown in Figure 9.21. The semiconductor quenching no longer occurs due to induction and from some point given the illusion that the access steps performed faster.

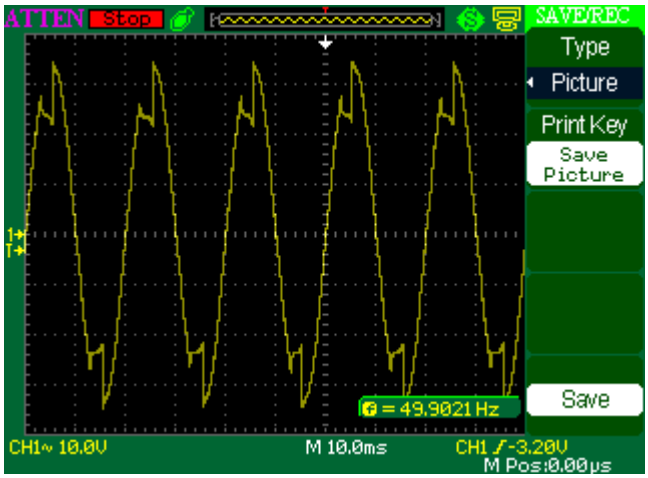


Figure 9.21 Output of starter on composite load in step 36-[n+1].

Conclusions:

In the composite load, system behavior is smoothed relative to the purely inductive because of the existence of the ohmic component as shown in the above waveforms. However, the response of the starter is not identical to that of the ohmic load. The effect of the condact of semiconductor elements over the zero voltage is present but is not as intense. Finally the exponential rise of starting ramp tends to get a more linear form by reducing the inductive character of the load. Summarizing the behavior of the starter is more improved, but this does

not mean that by adding ohmic component, solves the problem of loss of control of quenching in TRIACs.

9.5 The output of the soft starter incorporating fuzzy logic

The problem of the condact of semiconductors due to the inductive component of the load is what dictates the integration of fuzzy logic in the system. As mentioned above the exponential form of starting ramp causes the startup load faster than the timeframes set by the user. The fuzzy logic acts on the development of starter steps so that the startup process takes place within the specified limits. This is done by comparing the current with "fuzzy step" that extracts fuzzy system. Depending on the value of the residual voltage, the fuzzy logic determines the point at which It will change the angle of ignition semiconductor. The result of this technique is the occurrence of the starting steps they not have a linear progression.

For understanding and displaying the results of the fuzzy system used two identical soft starters which at their ends carry different loads. One of them has a inductive load and the other carries ohmic load. The waveform of the ohmic load (blue waveform) coincides and expresses starter exit at the current step because in this case the fuzzy system is not activated. The start time was set to be the same in both starters. In the Fig. 9.22 following output waveforms are shown the two systems. Contact beyond zero ac voltage is still present in the inductive load (yellow waveform), resulting in the evolution of the steps to be delayed. Observed early on that the starter ignition angle bearing ohmic load (blue waveform), is less because the starter ignition angle with the inductive load staying in the same step.

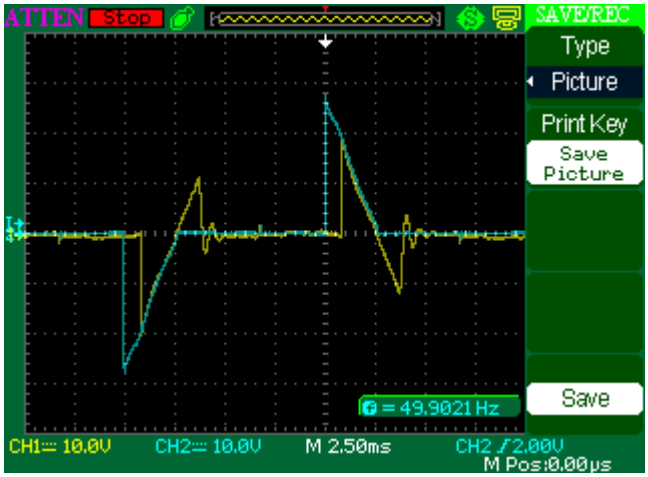


Figure 9.22 The antecedence "current step" compared with the "fuzzy step".

This is due to the non-equation current price to the "fuzzy step" according to what has been said in the previous chapters. The result of this behavior is the difference the ignition angles of both systems to increase.

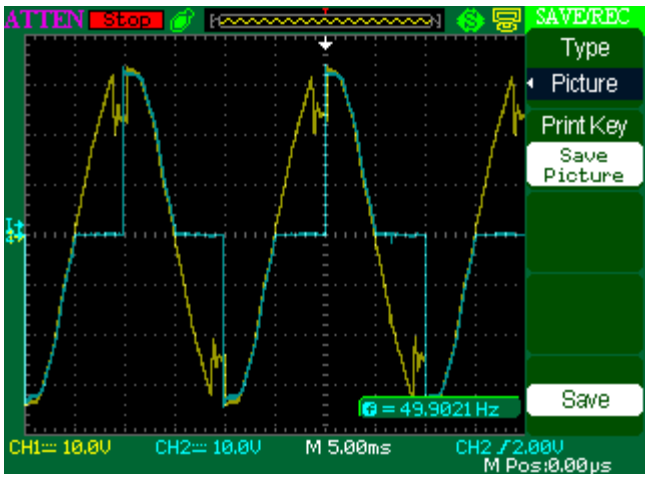


Figure 9.23 The equation "current step" with the "fuzzy step".

In Figure 9.23 we see the current step equation with "fuzzy step." At this point, change the weighted step of the output voltage. The change of the ignition angle in the inductive load carried out according to the output of the fuzzy system, that there are no sudden changes in output voltage. In this way the starter manages the current intelligently, keeping it in the permissible limits. After changing the ignition angle in the inductive load becomes the revaluation of "fuzzy step." The system remains firmly in the same firing angle until current step equation with "fuzzy step."

In the Figure 9.24 below shows the difference between the next firing angles in both starters. The startup process in ohmic load (blue waveform) is performed normally as opposed to startup inductive load (yellow waveform) that follows due to the stay of the firing angle to the same value.

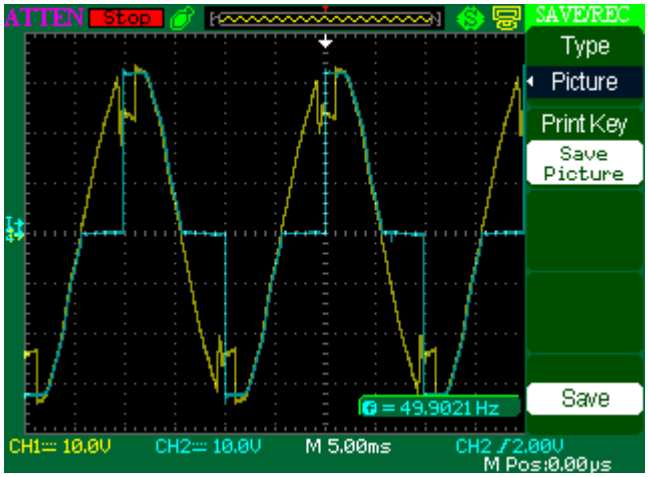


Figure 9.24 The difference of "current step" firing angle with "fuzzy step" firing angle.

The fuzzy logic gives the starter an intelligent behavior to handle the electrical current during the start of shipments. The system understands the character of each load and is adapted to the requirements of aiming a smooth startup. As shown above, when the starter has at its output ohmic load, fuzzy system remains inactive due to the non appearance residual voltage. Unlike, the fuzzy logic is activated and affects the system when realized inductive component to the load. Since a voltage value perceived by the starter decause of the non-quecching of semiconductor, system determines its step efficiently.

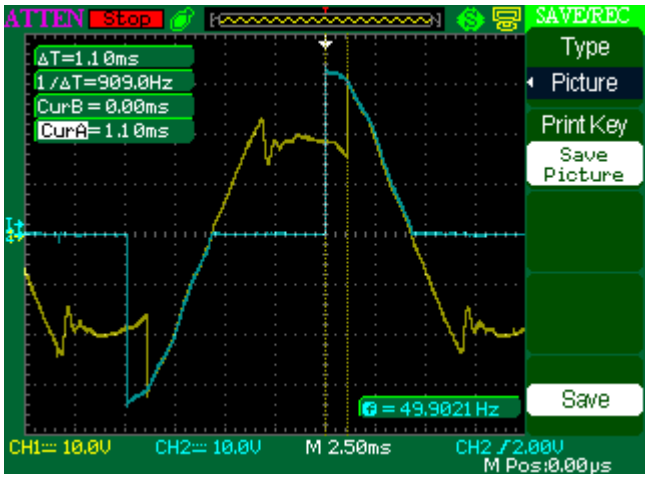


Figure 9.25 The effect of fuzzy logic in the development of starter steps.

Conclusions:

In the picture above, Fig. 9.25 is shown the effect of fuzzy logic in the development of starter steps to delay the transition to the next step that will take place ignition. Depending on the characteristics of the load, in an instance, there is a difference of the firing angle (ohmic-inductive) during the startup process. The dispute in the firing angle coincides with the difference current step with "fuzzy step". Thus each weighted change of the output voltage, in inductive load will not have the same relative rate to ohmic load. This is because the estimation of the "fuzzy step" relates from the value of voltage in the non-quecching of the semiconductor area. This fact characterizes the operation of the soft starter as an intelligent system, which aims to reduce the high starting current within the time limits originally set by the user, taking into account the behavior of the load is at its ends.

9.6 The graph of the rise ramp of the soft starter - Evaluation

In Figure 9.26 below shows the curves of the starter ramp resulting from the weighted value of the output voltage. The red curve represents the function the starter with inductive load without including fuzzy logic. The blue curve represents the function of the starter with

inductive load including fuzzy logic. Finally, the yellow curve represents the function of the starter to purely ohmic load.

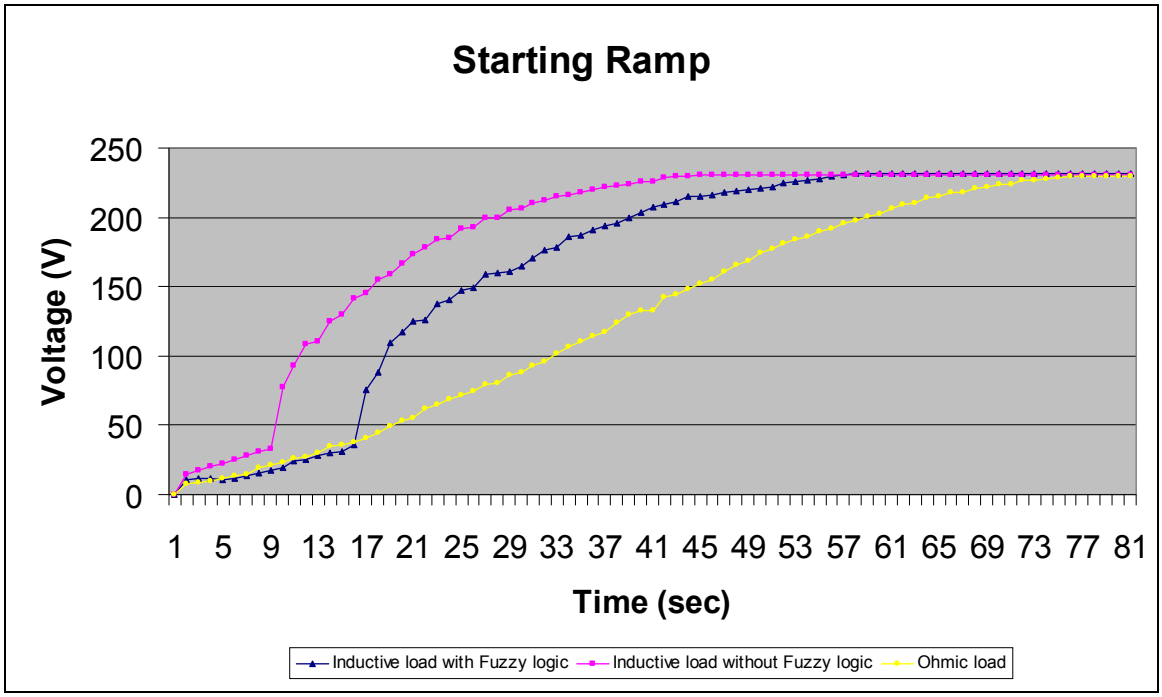


Figure 9.26 Soft starter ramps resulting from the output voltage in differents loads.

Commenting on the above diagram observed the linear behavior of the starter in ohmic load, accessing all the steps in the timeframe set by the user. Conversely when the starter carries inductive load, the ramp has exponential form. If not included fuzzy logic the phenomenon of intended startup is intense. The starting process is completed earlier without exhausting all startup steps because of inductance of the load. But when the system include fuzzy logic ramp is improved considerably. The curve does not lose its exponential form, but starting procedure is nearing completion in the time limit was given initially. Starting steps depleted largely providing a soft start of the load with more weighted levels of voltage.

Relating the above to the current flowing through the load follows the starting ramp in the graph shown in Fig. 9.27.

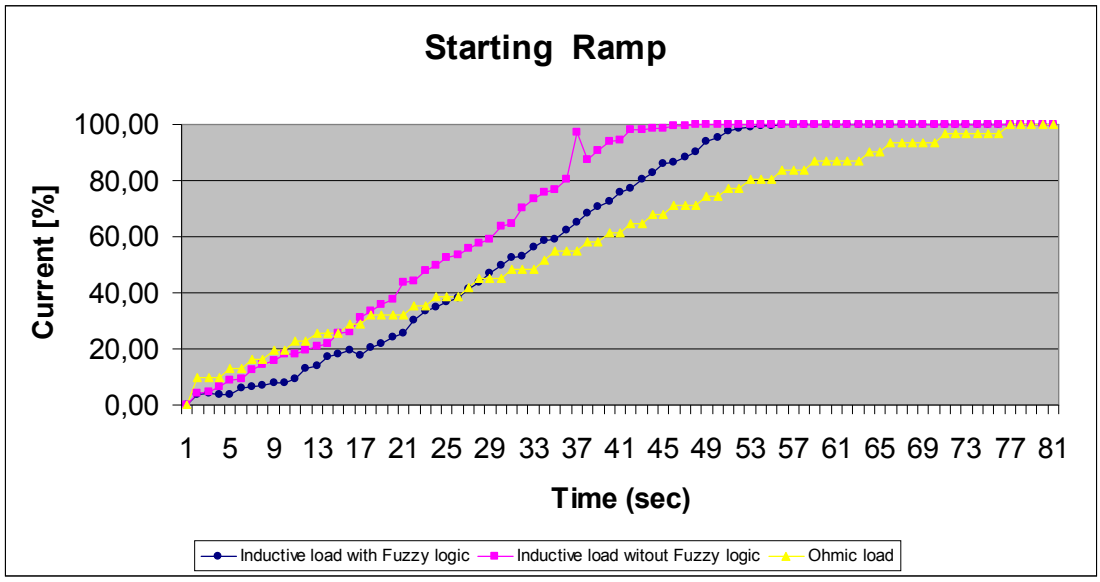


Figure 9.27 Soft starter ramps resulting from the current flowing in differents loads.

The yield of the graphs was reducing the current value in the unit in each case. Regardless of the type of load shape of the curves are linear. In the ohmic load the slope of the line (yellow curve) is low and similar to that of the above Fig. 9.26 due to the linear dependency voltage-current that exists in ohmic loads. In the inductive load (red curve), without the incorporation of fuzzy logic, the slope of the line increases with the result that the startup be performed faster due to the exponential increase of the voltage on the load. By incorporating fuzzy logic (blue curve) the slope of the ramp is smaller. The current approaches its nominal value close to the time limit was given. The improvement observed causes a smoother startup inductive load, while the benefit in energy savings is large enough.

Conclusions:

Adopting fuzzy logic improves significantly the operation and response of the starter in inductive loads with the results to be encouraging. With the participation of more weighted voltage steps the startup process is smoother and approaching the time limit set by the user, while the motor not stressed by high currents. But there is scope for improving the startup process, introducing more fuzzy rules in the system. However this presupposes memory efficiency and computing power.

Another advantage that presents soft starter is the energy savings offered when starting induction motors in electrical installations. As shown in Fig. 9.28 below, on the direct start up power absorbed by the motor gets values multiples of nominal current (In) thereby this causing waste of energy. The soft starter reduces this effect and the current does not present impactive values until the startup process completes.

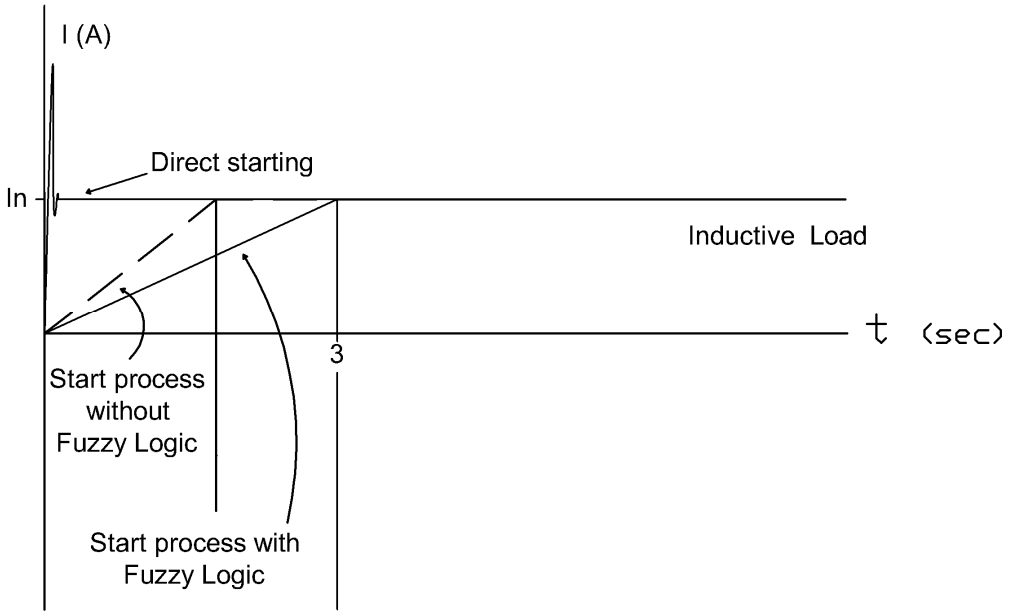


Figure 9.28 Current that absorbed by the motor in differents startup methods.

Summarizing the above conclusions, the fuzzy logic acts on the flattening of the starting ramp, as it has the inferences to further energy savings during startup. The system offers an intelligent management 3~ loads, focusing on their starting using a relatively small class microcontroller with the least possible resources it offers. Finally the implementation of the system does not require large financial cost.

9.7 Future Work

A proposal to further improve the response of the soft starter in iductive load is based on the normalization of the exponential rising ramp by appropriatly defining the ignition angle of semiconductors. Therefore it is proposed to incorporate more fuzzy rules that act primarily at that point where there is a sudden change in the weighted starter output voltage, as shown in Fig. 9.25 (knee of the curve). Finally another point that should be investigated is the increase of the system step in combination with the additional rules of fuzzy logic which can lead to even greater flattening of the rise ramp.

References

References

[1] A. J. Williams and M. S. Griffith, “Evaluating the effects of motor starting on industrial and commercial power systems,” IEEE Trans. Ind. Applicat., vol. IA-14, pp. 292–299, July/Aug. 1978.

[2] F. M. Bruce, R. J. Graefe, A. Lutz, and M. D. Panlener, “Reduced voltage starting of squirrel-cage induction motors,” IEEE Trans. Ind. Applicat., vol. IA-20, pp. 46–55, Jan./Feb. 1984.

[3] J. Nevelsteen and H. Aragon, “Starting of large motors-methods and economics,” IEEE Trans. Ind. Applicat., vol. 25, pp. 1012–1018, Nov./Dec. 1989.

[4] G. Zenginobuz, I. Cadirci, M. Ermis, and Guneyt Barlak, “Performance optimization of induction motors during voltage-controlled soft starting,” IEEE Trans. on Energy Conversion, vol. 19, no. 2. pp. 278-288, June 2004.

[5] H. A. Ashour and R. A. Ibrahim, "Comparison analysis of AC voltage controllers based on experimental and simulated application studies," in the Proceedings of the 2006 International Conference on Computer Engineering and Systems, Nov. 5-7, 2006, pp. 79-84.

[6] R. F. McElveen, and M. K. Toney, “Starting high-inertia loads,” IEEE Trans. Industry Applications, Vol. 37, No. 1, pp. 137-144, January/ February 2001.

[7] A. Gastli and M. M. Ahmed, “ANN-based soft starting of voltage-controlled-fed IM drive system,” IEEE Trans. Energy Conversion, pp. 1-7, 2005.

[8] K. Sundareswaran, P. Srinivasarao Nayak, “Ant colony based feedback controller design for soft-starter fed induction motor drive,” Applied Soft Computing 12, pp. 1566–1573, 2012.

[9] Xiaodong Liang, and Obinna Ilochonwu, “Induction motor starting in practical industrial applications,” IEEE Transactions on Industry Applications, Vol. 47, No. 1, pp. 271-280, 2011.

[10] Hamdy, A. Ashour and Rania A. Ibrahim, “Implementation and analysis of microcontroller-based soft-starters for three-phase induction motors”, in the Proceedings of the International Conference on Computer as a Tool (EUROCON’2007), Warsaw, Sept. 9-12, pp. 2193-2199, 2007.

[11] A. Rahman, N. Ansari, N. Ahmed, K. M. Rahman and Md. Z. Islam, "Development of a microcontroller-based AC voltage controller with soft start capability," in the Proceedings of the 8th International Conference on Electrical and Computer Engineering, 20-22 Dec. 2014, Dhaka, Bangladesh, pp. 611-614.

[12] John Kalomiros, Programming of Real Time embedded System, TEI of Serres, Dpt. of Informatics Engineering, Serres 2012.

[13] Kiranastasis George, Power electronics, TEI of Kavala, Dpt. of Electrician Engineering, Kavala 1989.

[14] Stephen J. Chapman, Electrical Machines DC-AC, editions McGraw-Hill / A.Tziola, 2001.

[15] Tim Wilmshurst, Designing Embedded Systems with PIC Microcontrollers Principles and applications, Newnes 2007.

[16] Paris Mastorokostas, Fuzzy Systems, Dpt. of Informatics Engineering, Serres 2014.

[17] Microchip data sheet for PIC 18F252 microcontroller. Found at: http://ww1.microchip.com/downloads/en/DeviceDoc/39564c.pdf. [access 2015].

Appendix 1

Lookup table (LUT) of the system.

Step No	angle θ (degrees)	sin(θ)	Vref (volt)=5,0 Um (volt)=5,0	U=Um x sin(θ) (volt)	Vstep= Vref/1024	ADCout=U/Vstep 10-bit resolution	Left justified ADRESH	Left justified ADRESL	ADRESH (Dec)	LUT Data
36	22	0,374607	5	1,873	0,004882813	384	01100000	00	96	96
35	26	0,438371	5	2,192	0,004882813	449	01110000	01	112	208
34	30	0,5	5	2,500	0,004882813	512	10000000	00	128	336
33	34	0,559193	5	2,796	0,004882813	573	10001111	01	143	479
32	38	0,615661	5	3,078	0,004882813	630	10011101	10	157	636
31	42	0,669131	5	3,346	0,004882813	685	10101011	01	171	807
30	46	0,71934	5	3,597	0,004882813	737	10111000	01	184	991
29	50	0,766044	5	3,830	0,004882813	784	11000100	00	196	1187
28	54	0,809017	5	4,045	0,004882813	828	11001111	00	207	1394
27	58	0,848048	5	4,240	0,004882813	868	11011001	00	217	1611
26	62	0,882948	5	4,415	0,004882813	904	11100010	00	226	1837
25	66	0,913545	5	4,568	0,004882813	935	11101001	11	233	2070
24	70	0,939693	5	4,698	0,004882813	962	11110000	10	240	2310
23	74	0,961262	5	4,806	0,004882813	984	11110110	00	246	2556
22	78	0,978148	5	4,891	0,004882813	1002	11111010	10	250	2806
21	82	0,990268	5	4,951	0,004882813	1014	11111101	10	253	3059
20	86	0,997564	5	4,988	0,004882813	1022	11111110	00	254	3313
19	90	1	5	5,000	0,004882813	1024	11111111	11	255	3568
18	94	0,997564	5	4,988	0,004882813	1022	11111110	00	254	3822
17	98	0,990268	5	4,951	0,004882813	1014	11111101	10	253	4075
16	102	0,978148	5	4,891	0,004882813	1002	11111010	10	250	4325
15	106	0,961262	5	4,806	0,004882813	984	11110110	00	246	4571
14	110	0,939693	5	4,698	0,004882813	962	11110000	10	240	4811
13	114	0,913545	5	4,568	0,004882813	935	11101001	11	233	5044
12	118	0,882948	5	4,415	0,004882813	904	11100010	00	226	5270
11	122	0,848048	5	4,240	0,004882813	868	11011001	00	217	5487
10	126	0,809017	5	4,045	0,004882813	828	11001111	00	207	5694
9	130	0,766044	5	3,830	0,004882813	784	11000100	00	196	5890
8	134	0,71934	5	3,597	0,004882813	737	10111000	01	184	6074
7	138	0,669131	5	3,346	0,004882813	685	10101011	01	171	6245
6	142	0,615661	5	3,078	0,004882813	630	10011101	10	157	6402
5	146	0,559193	5	2,796	0,004882813	573	10001111	01	143	6545
4	150	0,5	5	2,500	0,004882813	512	10000000	00	128	6673
3	154	0,438371	5	2,192	0,004882813	449	01110000	01	112	6785
2	158	0,374607	5	1,873	0,004882813	384	01100000	00	96	6881
1	162	0,309017	5	1,545	0,004882813	316	01001111	00	79	6960

Appendix 2

The PIC 18F252 program for the implementation of soft starter

```
//fosc=20MHz
#include <p18f252.h>
#include <delays.h>
#include <stdio.h>
//------------------------------

// set configuration bits
#pragma config OSC = HS, OSCS = OFF    //oscillator type is HS, oscillator switch is off
#pragma config PWRT = ON, BOR = OFF  //power-up timer is on, brown-out detect is off
#pragma config WDT = OFF                    //watchdog timer is off
#pragma config STVR = ON, LVP = OFF   //Stack overflow reset enable is on,
                                                  //low voltage programming is off

//=============================================================================================================

//Define constants for Fuzzy control

#define N 3
#define M 36

//------------------------------

//Define constants for ADC


//------------------------------

//Define constants for Soft_Startig

//For fosc=20MHz, one instruction cycle is 0.2μec
```

```
// Delay10KTCYx(100)-> delay=10.000x100x0.2μsec=0.2sec
#define onesec Delay10KTCYx(500)        //delay 1 sec
#define m500sec Delay10KTCYx(250)       //delay 500 msec
#define m250sec Delay10KTCYx(125)       //delay 250 msec
#define m5sec Delay1KTCYx(25)            //delay 5 msec


//=============================================================================================================

//Define function for Fuzzy control
void Fuzzy(void);

//------------------------------

//Define functions for ADC
void ADCTask(void);
void INIT_ADC(void);
void EnableADC(void);
void CalcSum(void);
void ReadLUT(void);

//------------------------------


//Define functions for Soft_Starting
void INIT(void);
void Clear(void);
void ClearGate(void);
void Diagnostic(void);
void START(void);
```

```
void Alarm(void);
void ReadData(void);
void SetTimer(void);
void ControlCurrent(void);
void ControlStop(void);
void ControlTrigger(void);
void ControlTrigger_(void);
void ControlOverFlow(void);
void ControlGate(void);
void ControlStep(void);
void ControlByPass(void);
void test(void);

//=============================================================================================================

//Define arrays for fuzzy control

#pragma idata x1
 unsigned char x1[N][M]={       //field of value x1 for input 1
        0,7,14,21,28,35,42,49,56,63,70,77,84,91,98,105,112,119,126,133,140,147,154,161,168,175,183,191,199,207,215,223,231,239,247,255,
        0,7,14,21,28,35,42,49,56,63,70,77,84,91,98,105,112,119,126,133,140,147,154,161,168,175,183,191,199,207,215,223,231,239,247,255
        0,7,14,21,28,35,42,49,56,63,70,77,84,91,98,105,112,119,126,133,140,147,154,161,168,175,183,191,199,207,215,223,231,239,247,255};
                        //create Triangular membership functions for 1st inpuy. For each rule

#pragma idata x2
         unsigned char x2[N][M]={       //field of value x2 for input 2
                0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,14,28,42,56,70,84,98,112,126,140,154,168,182,196,210,225,239,255,
                0,0,0,0,0,0,0,0,0,28,56,84,112,140,168,197,225,255,225,197,168,140,112,84,56,28,0,0,0,0,0,0,0,0,0,0,
                255,239,225,210,196,182,168,154,140,126,112,98,84,70,56,42,28,14,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
                 //create Triangular membership functions for 2nd input. For each rule
```

```
#pragma idata y
        unsigned char   y[N][M]={               //field of value y for output
         0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,14,28,42,56,70,84,98,112,126,140,154,168,182,196,210,225,239,255,
         0,0,0,0,0,0,0,0,0,28,56,84,112,140,168,197,225,255,225,197,168,140,112,84,56,28,0,0,0,0,0,0,0,0,0,0,
         255,239,225,210,196,182,168,154,140,126,112,98,84,70,56,42,28,14,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
                        //create Triangular membership function for output. For each rule

#pragma idata A1_bar
 unsigned char    A1_bar[M]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

#pragma idata A2_bar
 unsigned char    A2_bar[M]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

#pragma idata B_bar_total
 unsigned int B_bar_total[M]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

#pragma udata B_bar
 unsigned char B_bar[N][M];

#pragma udata mini
 unsigned char mini[M];

//----------------------------

//Define array for ADC

//_____________________________________________LOOK UP TABLE_____________________________________________
#pragma idata StepADC
//Vmax= 5V
```

```
   unsigned int
StepADC[36]={96,208,336,479,636,807,991,1187,1394,1611,1837,2070,2310,2556,2806,3059,3313,3568,3822,4075,4325,4571,4811,5044,5270,
5487,5694,5890,6074,6245,6402,6545,6673,6785,6881,6960};


 //=============================================================================================

 // Define varables for Fuzzy control

  unsigned char n,m;           //index
  unsigned short long Num_y=0;
  unsigned int Denum_y=0;
  unsigned char w1,w2,w;
  unsigned char StepOn;         //Number of steps where Triac is ON after Zero-Cross
  unsigned char TotStepFB;      //Imagine number of steps where we have output
  unsigned char DeFuzzyStep=0;
 //TotStepFB is the number of total steps where we have output

 //-------------------------




 //Define variables for ADC

  unsigned char i=0;
  unsigned int sum=0;

 //---------------------------
```

```
//Define variables for Soft_Starting

typedef union
 {
 unsigned char data;

  struct
  {
    unsigned TRR:1;
    unsigned TRS:1;
    unsigned TRT:1;
    unsigned BP:1;
    unsigned SF:1;
    unsigned NCF:1;
    unsigned ZCO:1;
    unsigned UT1:1;
 };
 } MYUNION1;


typedef union
 {
 unsigned char data;

  struct
  {
    unsigned LOR:1;
    unsigned LOS:1;
    unsigned LOT:1;
```

```
    unsigned FSO:1;
    unsigned FSW:1;
    unsigned ADW:1;
    unsigned UO1:1;
    unsigned UO2:1;
  };
  } MYUNION2;

//-----------------------------



MYUNION1 TRSTATUS;
MYUNION2 OVSTATUS;
unsigned char InputData;
unsigned char StepLevel;
unsigned char PulseLevel;
unsigned char CounterZeroCross_R;
unsigned char CounterOverFlow_R;
unsigned char CounterOverFlow_S;
unsigned char CounterOverFlow_T;
unsigned char CurrentData;
unsigned char BackRoundStep;


//---------------------------------------------

void main (void)                        //main function starts here
{
```

```
        INIT();
                INIT_ADC();
                Diagnostic();
                Clear();
DataIn:

                START();



MAIN:
                ControlStop();
                if(TRSTATUS.BP==1)
                goto stop;
                //else
                ControlTrigger();
                ControlOverFlow();
                ControlStep();
                ControlGate();
                ControlByPass();
                if(OVSTATUS.ADW==1)
                ADCTask();
                if(OVSTATUS.FSW==1)
                Fuzzy();

stop:
                if(TRSTATUS.SF==0)
                goto MAIN;
                ClearGate();
                PORTCbits.RC3=0;     //cut off the signal for By-Pass Relay
```

```
                    Clear();
                    goto DataIn;


} //end main

//====================================FUNCTION_FOR_FUZZY_CONTROL==========================

void Fuzzy(void)
{

TotStepFB=StepLevel-StepOn;

//-------------------------------------

if (StepLevel<StepOn || StepOn<=3)
            {

                    //StepLevel=BackroundStep;
                    OVSTATUS.FSO=0;         //Declare that, we have not output from Fuzzy system
                    OVSTATUS.FSW=0;         //Turn off the Fuzzy system
            return;
            }

//-------------------------------------

A1_bar[StepLevel]=255;          //create fuzzy sigleton REAL STEP
A2_bar[TotStepFB]=255;          //create fuzzy sigleton FROM FEEDBACK

//-------------------------------------
```

```
startFuzzy:
                                ControlTrigger_();
                                ControlOverFlow();
                                ControlGate();
                                if(PORTBbits.RB3==0)      //Wait for new half-period in phase S
                                goto startFuzzy;
                                while(PORTBbits.RB3==1)       //Wait to pass the zero-cross in phase S
                                        {
                                        ControlTrigger_();
                                        ControlOverFlow();
                                        ControlGate();
                                        }

//--------------------------------------
   //Start working in the half-period of phase S

for(n=0;n<N;n++)
        {    //----------------------------------

                for(m=0;m<M;m++)
                        {
                                        ControlTrigger_();
                                        ControlOverFlow();
                                        ControlGate();

                                if(A1_bar[m]>x1[n][m])
                                        {
                                                mini[m]=x1[n][m];
                                        }
```

```
                        else
                                {
                                        mini[m]=A1_bar[m];
                                 }
                        }
        w1=mini[0];
        for(m=1;m<M;m++)
                        {
                                ControlTrigger_();
                                ControlOverFlow();
                                ControlGate();
                                if(mini[m]>w1)
                                  w1=mini[m];
                         }

//compute w1 max-min operator for each rule (input 1)

//-------------------------------------------

                for(m=0;m<M;m++)
                        {
                                ControlTrigger_();
                                ControlOverFlow();
                                ControlGate();
                        if(A2_bar[m]>x2[n][m])
                                {
                                        mini[m]=x2[n][m];
                                }
                        else
                                {
```

```
                                        mini[m]=A2_bar[m];
                                    }
                        }
        w2=mini[0];
        for(m=1;m<M;m++)
                        {
                                ControlTrigger_();
                                ControlOverFlow();
                                ControlGate();
                if(mini[m]>w2)
                w2=mini[m];
                        }

//compute w2 max-min operator for each rule (input 2)

//-----------------------------------------------

                if(w1>w2)
                        {
                        w=w2;
                         }
                else
                        {
                         w=w1;
                         }

//compute w min operator for each rule (from input1 and input 2)

//-----------------------------------------------
```

```
                for(m=0;m<M;m++)
                        {
                                ControlTrigger_();
                                ControlOverFlow();
                                ControlGate();
                        if(w>y[n][m])
                                        {
                                                B_bar[n][m]=y[n][m];
                                        }
                        else
                                        {
                                                B_bar[n][m]=w;
                                        }
                        }

//output for each rule

//-------------------------------------------------

        }//end for(n=0;n<N;n++)


//-------------------------------------------------

for(n=0;n<N;n++)
        {
      for(m=0;m<M;m++)
                {
                  ControlTrigger_();
                  ControlOverFlow();
```

```
                ControlGate();
                if(B_bar_total[m]<=B_bar[n][m])
                        B_bar_total[m]=B_bar[n][m];
                }
        }

//fuzzy oytput of the system

//---------------------------------------

//          Defuzzyfier COA

for(m=0;m<M;m++)
        {
                ControlTrigger_();
                ControlOverFlow();
                ControlGate();
                Num_y=Num_y+((m+1)*B_bar_total[m]);
                Denum_y=Denum_y+B_bar_total[m];
        }



DeFuzzyStep=(Num_y/Denum_y);
                ControlTrigger_();
                ControlOverFlow();
                ControlGate();

if (DeFuzzyStep>=StepLevel)
```

```
            {

                //StepLevel=BackroundStep;
                OVSTATUS.FSO=0;             //Declare that, we have not output from Fuzzy system
                OVSTATUS.FSW=0;             //Turn off the Fuzzy system
                return;
            }


DeFuzzyStep=DeFuzzyStep+((StepLevel-DeFuzzyStep)/2)+1;

            ControlTrigger_();
            ControlOverFlow();
            ControlGate();

//StepLevel=DeFuzzyStep;

      OVSTATUS.FSO=1;               //Declare that, we have output from Fuzzy system
      OVSTATUS.FSW=0;               //Turn off the Fuzzy system

}
//=====================================FUNCTIONS_FOR_ADC=========================================

void ADCTask(void)
{
startADC:
                sum=0;
                StepOn=0;
                ADRESH=0;
                ADRESL=0;
```

```
ADCON0bits.ADON=1;     //Turn on the ADC
ControlTrigger_();
ControlOverFlow();
ControlGate();
if(PORTBbits.RB3==0)      //Wait for new half-period in phase S
goto startADC;
while(PORTBbits.RB3==1)       //Wait to pass the zero-cross in phase S
        {
        ControlTrigger_();
        ControlOverFlow();
        ControlGate();
        }
while(PORTBbits.RB3==0)       //While is in the half-period in phase S DO
        {
                EnableADC();
                while(INTCONbits.TMR0IF==0)     //Wait overflow the TMR0
                        {
                                ControlTrigger_();
                                ControlGate();
                        }
                ControlOverFlow();
                CalcSum();                      //Calculate the total amound of ADC
                if(PORTCbits.RC1==1)            //If trere is signal to turn-on the Triac in phase S
                        break;                  //Exit and turn off the ADC
        }
ADCON0bits.ADON=0;
ReadLUT();                                      //Compiute the step from residual voltage
OVSTATUS.ADW=0;
OVSTATUS.FSW=1;
}
```

```
void INIT_ADC(void)
{
                        ADCON1bits.PCFG3=0;       //Only AN0 & AN1 bitS of PORTA are Analog inputs
                        ADCON1bits.PCFG2=1;       //Vref+=AN3  , Vref-=VSS=0V (ground)
                        ADCON1bits.PCFG1=0;
                        ADCON1bits.PCFG0=1;

//For fosc=20MHz -> A/D conversion clock is 16Tosc [101]
                        ADCON1bits.ADCS2=1;
                        ADCON0bits.ADCS1=0;
                        ADCON0bits.ADCS0=1;
                        ADCON0bits.CHS2=0;  //Select channel 0 (AN0) of ADC
                        ADCON0bits.CHS1=0;
                        ADCON0bits.CHS0=0;

                        ADCON1bits.ADFM=0;  //Left justification, ADRESH register has the Data
                        ADRESH=0;
                        ADRESL=0;
                        ADCON0bits.GO_DONE=0;
        }

//---------------------------------------------

void EnableADC(void)
{
                        if (ADCON0bits.GO_DONE==1)
                        return;
                        ADCON0bits.GO_DONE=1;
}
```

```
//---------------------------------------------

void CalcSum(void)
{
                        if(ADCON0bits.GO_DONE==0)
                        sum=sum+ADRESH;
                        return;
}

//---------------------------------------------

void ReadLUT(void)
{

                                for(i=0;i<36;i++)
                                {
                                  if(StepADC[i]<sum)
                                        {
                                                StepOn=i;
                                                ControlTrigger_();
                                                ControlOverFlow();
                                                ControlGate();
                                        }
                                    else
                                        break;
                                }

}
```

```
//=====================================FUNCTIONS_FOR_SOFT_STARTING=====================================

void INIT(void)
{
                        TRISA=0b11111111;  //All bits, Inputs
                        TRISB=0b00011111;
                        TRISC=0b11110000;
                        T0CON=0b11010011;    // 1/16 prescaler TMR0
                        Clear();
}

//-----------------------------------------

void Clear(void)
{

                        InputData=0;
                        CounterZeroCross_R=0;
                        CounterOverFlow_R=0;
                        CounterOverFlow_S=0;
                        CounterOverFlow_T=0;
                        TRSTATUS.data = 0b01000000;    //ZCO=1
                        OVSTATUS.data = 0x00;
                        PORTCbits.RC3=0;                    //CUT-OFF the signal for By-Pass Relay
                        ClearGate();
}

//-----------------------------------------------
```

```
void ClearGate(void)
{

                        _asm
                        movlw 0b11111000
                        andwf  PORTC,1,0
                        _endasm
}

//-------------------------------------------------

void Diagnostic(void)          //ON-OFF a led 2 times
{
                        TRISB=0b00011110;    //RB0 is output
                        PORTBbits.RB0=1;
                        m250sec;
                        PORTBbits.RB0=0;
                        m250sec;
                        PORTBbits.RB0=1;
                        m250sec;
                        PORTBbits.RB0=0;
                        m250sec;
                        TRISB=0b00011111;      //RB0 is input
}

//-------------------------------------------------
```

```
void START(void)
{

                    while(PORTBbits.RB1==0)   //Input start button ->RB1
                    {}                        //Wait until user press the button
                    m5sec;
                    while(PORTBbits.RB1==1)
                    {}                        //Wait release
                    m5sec;
                    BackRoundStep=36;          //36 is the total steps of soft starter
                    StepLevel=BackRoundStep;
                    ReadData();                //Read the data from user. It is the time for soft starting
                    while(PORTBbits.RB2==0)   //Is there a zero-cross pulse from R phase?
                    {}                        //No -> do nothing
                    TRSTATUS.TRR = 0;         //Yes-> clear the flag TRR, declare that zero-cross pulse in phase R
                    TRSTATUS.ZCO = 0;         //Recognise 1st time zero-cross
                    CounterZeroCross_R++;
                    SetTimer();

}

//-----------------------------------------------------

void SetTimer(void)
{

                    INTCONbits.TMR0IF=0;
                    TMR0L=179;

}
```

```
//-----------------------------------------------------

void ControlStop(void)
{


                    if(TRSTATUS.NCF==1)
                    goto SetFlag;
                    while(PORTBbits.RB0==0)   //input stop button ->RB0 ?
                        {
                            return;                     //No stop button pressed -> exit
                        }

SetFlag:
                    ClearGate();
                    PORTCbits.RC3=0;
                    TRSTATUS.SF=1;        //Declare that, the system must turn-off

}

//-----------------------------------------------------

void ControlTrigger(void)
{

                    if(PORTBbits.RB2==0)          //Control if there is zero-cross pulse from phase R
                    goto TriggR_Set;
                    if(TRSTATUS.ZCO==1)
                    CounterZeroCross_R++;
```

```
                TRSTATUS.ZCO=0;          //Recognise 1st time zero-cross
                PORTCbits.RC0=0;         //Clear the signal for the triac in phase R
                OVSTATUS.LOR=0;          //Clear the flag for the Last Over-flow in phase R
                TRSTATUS.TRR=0;          //There is zero-cross in phase R
                CounterOverFlow_R=0;
                goto loop1;
TriggR_Set:
                TRSTATUS.TRR=1;          //Passed the zero-cross signal in phase R
                TRSTATUS.ZCO=1;
loop1:
                if(PORTBbits.RB3==0)     //Control if there is zero-cross pulse from phase S
                goto TriggS_Set;
                PORTCbits.RC1=0;         //Clear the signal for the triac in phase S
                OVSTATUS.LOS=0;          //Clear the flag for the Last Over-flow in phase S
                TRSTATUS.TRS=0;          //There is zero-cross in phase S
                CounterOverFlow_S=0;
                goto loop2;
TriggS_Set:
                TRSTATUS.TRS=1;          //Passed the zero-cross signal in phase S
loop2:
                if(PORTBbits.RB4==0)     //Control if there is zero-cross pulse from phase T
                goto TriggT_Set;
                PORTCbits.RC2=0;         //Clear the signal for the triac in phase T
                OVSTATUS.LOT=0;          //Clear the flag for the Last Over-flow in phase T
                TRSTATUS.TRT=0;          //There is zero-cross in phase T
                CounterOverFlow_T=0;
                return;
TriggT_Set:
                TRSTATUS.TRT=1;          //Passed the zero-cross signal in phase T
}
```

```
//-----------------------------------------------------


void ControlTrigger_(void)                  //In this function it is not mesured the zero-cross pulse in phase R
{

                    if(PORTBbits.RB2==0)    //Control if there is zero-cross pulse from phase R
                    goto TriggR_Set_;
                    PORTCbits.RC0=0;        //Clear the signal for the triac in phase R
                    OVSTATUS.LOR=0;         //Clear the flag for the Last Over-flow in phase R
                    TRSTATUS.TRR=0;         //There is zero-cross in phase R
                    CounterOverFlow_R=0;
                    goto loop1_;
TriggR_Set_:
                    TRSTATUS.TRR=1;         //Passed the zero-cross signal in phase R
                    TRSTATUS.ZCO=1;
loop1_:
                    if(PORTBbits.RB3==0)    //Control if there is zero-cross pulse from phase S
                    goto TriggS_Set_;
                    PORTCbits.RC1=0;        //Clear the signal for the triac in phase S
                    OVSTATUS.LOS=0;         //Clear the flag for the Last Over-flow in phase S
                    TRSTATUS.TRS=0;         //There is zero-cross in phase S
                    CounterOverFlow_S=0;
                    goto loop2_;
TriggS_Set_:
                    TRSTATUS.TRS=1;         //Passed the zero-cross signal in phase S
loop2_:
                    if(PORTBbits.RB4==0)    //Control if there is zero-cross pulse from phase T
                    goto TriggT_Set_;
```

```
                    PORTCbits.RC2=0;        //Clear the signal for the triac in phase T
                    OVSTATUS.LOT=0;         //Clear the flag for the Last Over-flow in phase T
                    TRSTATUS.TRT=0;         //There is zero-cross in phase T
                    CounterOverFlow_T=0;
                    return;
TriggT_Set_:
                    TRSTATUS.TRT=1;         //Passed the zero-cross signal in phase T
}

//-----------------------------------------------------

void ControlOverFlow(void)
{
                    if(INTCONbits.T0IF==0)  //Is it happend overflow from TMR0?
                    return;                 //No -> exit
                    if(TRSTATUS.TRR==0)     //Is it happened  zero-cross in phase R?
                    goto loop3;             //No -> continue control for other phase
                    CounterOverFlow_R++;
                    if(OVSTATUS.LOR==0)     //Is it happend the Last Overflow befor Triac turn-on?
                    PORTCbits.RC0=0;        //No -> clear the gate R signal
loop3:
                    if(TRSTATUS.TRS==0)     //Is it happened  zero-cross in phase S?
                    goto loop4;             //No -> continue control for other phase
                    CounterOverFlow_S++;
                    if(OVSTATUS.LOS==0)     //Is is happend the Last Overflow befor Triac turn-on?
                    PORTCbits.RC1=0;        //No -> clear the gate S signal
loop4:
                    if(TRSTATUS.TRT==0)     //Is it happened  zero-cross in phase T?
                    goto SetTimerAndExit;   //No -> Set Timer for next overflow
                    CounterOverFlow_T++;
```

```
                    if(OVSTATUS.LOT==0)     //Is it happend the Last Overflow befor Triac turn-on?
                    PORTCbits.RC2=0;        //No -> clear the gate T signal
SetTimerAndExit:
                    SetTimer();

}

//-----------------------------------------------------

void ControlGate(void)
{

                    if(OVSTATUS.LOR==1)             //Is it happend the Last Overflow for phase R?
                    goto Gate_S;                    //Yes -> control the next phase
                    if(CounterOverFlow_R<StepLevel) //Is the number of overflows in phase R less than number of steps?
                    goto Gate_S;                    //Yes -> control the next phase
                    if(TRSTATUS.TRR==0)             //Run a zero-cross pulse in phase R?
                    goto Gate_S;                    //Yes -> control the next phase
                    PORTCbits.RC0=1;                //Set the signal to turn-on the Triac in phase R
                    OVSTATUS.LOR=1;                 //Declare that, the last overflow for pfase R is happened
Gate_S:
                    if(OVSTATUS.LOS==1)             //Is it happend the Last Overflow for phase S?
                    goto Gate_T;                    //Yes -> control the next phase
                    if(CounterOverFlow_S<StepLevel) //Is the number of overflows in phase S less than number of steps?
                    goto Gate_T;                    //Yes -> control the next phase
                    if(TRSTATUS.TRS==0)             //Run a zero-cross pulse in phase S?
                    goto Gate_T;                    //Yes -> control the next phase
                    PORTCbits.RC1=1;                //Set the signal to turn-on the Triac in phase S
                    OVSTATUS.LOS=1;                 //Declare that, the last overflow for pfase S is happened
Gate_T:
```

```
                    if(OVSTATUS.LOT==1)             //Is it happend the Last Overflow for phase T?
                    return;                         //Yes -> exit
                    if(CounterOverFlow_T<StepLevel) //Is the number of overflows in phase T less than number of steps?
                    return;                         //Yes -> exit
                    if(TRSTATUS.TRT==0)             //Run a zero-cross pulse in phase T?
                    return;                         //Yes -> exit
                    PORTCbits.RC2=1;                //Set the signal to turn-on the Triac in phase T
                    OVSTATUS.LOT=1;                 //Declare that, the last overflow for pfase T is happened

}

//-----------------------------------------------------

void ControlStep(void)
{

                    if(CounterZeroCross_R<PulseLevel)
                    return;
                    CounterZeroCross_R=0;
                    if(OVSTATUS.FSO==1 && BackRoundStep>DeFuzzyStep)        //DeFuzzyStep=y_output;

                    {
                            BackRoundStep--;
                            return;
                    }
                    OVSTATUS.FSO=0;
                    StepLevel=BackRoundStep;
                    BackRoundStep--;
                    OVSTATUS.ADW=1;
}
```

```
//-----------------------------------------------------

void ControlByPass(void)
{

                    if(StepLevel>0)
                    return;                         //PORTC=0b00001111;
                    PORTCbits.RC0=1;                //Turn-on the Triac for phase R
                    PORTCbits.RC1=1;                //Turn-on the Triac for phase S
                    PORTCbits.RC2=1;                //Turn-on the Triac for phase T
                    PORTCbits.RC3=1;                //Turn-on the relay By-Pass
                    TRSTATUS.BP=1;                  //Declare the By-Pass contition
                    onesec;
                    ClearGate();

}

//-----------------------------------------------------

void ReadData(void)
{
                    _asm
                    movf   PORTC,0,0                //PORTC->w
                    andlw  0b11110000
                    movwf InputData,1
                    _endasm
                    if(InputData==0b00000000)       // 3sec soft-starting->InputData=0000xxxx
                            {
                                    PulseLevel=12;
```

```
                goto exit1;
        }
if(InputData==0b00010000)       // 4sec soft-starting->InputData=0001xxxx
        {
                PulseLevel=16;
                goto exit1;
        }
if(InputData==0b00100000)       // 5sec soft-starting->InputData=0010xxxx
        {
                PulseLevel=20;
                goto exit1;
        }
if(InputData==0b00110000)       // 6sec soft-starting->InputData=0011xxxx
        {
                PulseLevel=24;
                goto exit1;
        }
if(InputData==0b01000000)       // 7sec soft-starting->InputData=0100xxxx
        {
                PulseLevel=28;
                goto exit1;
        }
if(InputData==0b01010000)       // 8sec soft-starting->InputData=0101xxxx
        {
                PulseLevel=32;
                goto exit1;
        }
if(InputData==0b01100000)       // 9sec soft-starting->InputData=0110xxxx
        {
                PulseLevel=36;
```

```
                goto exit1;
        }
if(InputData==0b01110000)       // 10sec soft-starting->InputData=0111xxxx
        {
                PulseLevel=40;
                goto exit1;
        }
if(InputData==0b10000000)       // 11sec soft-starting->InputData=1000xxxx
        {
                PulseLevel=44;
                goto exit1;
        }
if(InputData==0b10010000)       // 12sec soft-starting->InputData=1001xxxx
        {
                PulseLevel=48;
                goto exit1;
        }
if(InputData==0b10100000)       // 13sec soft-starting->InputData=1010xxxx
        {
                PulseLevel=52;
                goto exit1;
        }
if(InputData==0b10110000)       // 14sec soft-starting->InputData=1011xxxx
        {
                PulseLevel=56;
                goto exit1;
        }
if(InputData==0b11000000)       // 15sec soft-starting->InputData=1100xxxx
        {
                PulseLevel=60;
```

```
                goto exit1;
        }
    if(InputData==0b11010000)       // 16sec soft-starting->InputData=1101xxxx
        {
                PulseLevel=64;
                goto exit1;
        }
    if(InputData==0b11100000)       // 17sec soft-starting->InputData=1110xxxx
        {
                PulseLevel=68;
                goto exit1;
        }
    if(InputData==0b11110000)       // 18sec soft-starting->InputData=1111xxxx
        {
                PulseLevel=72;
                goto exit1;
        }
exit1:      m5sec;
}

//=============================================END=========================================
```

Appendix 3

Soft Starter electronic circuit.