

**COMPUTER AIDED
ANTENNA CONTROL UNIT DESIGN
USING STEPPER MOTORS**

97374

77374

by

Ebru (AYVAZ)TAŞCI

**September, 1998
İZMİR**

COMPUTER AIDED ANTENNA CONTROL UNIT DESIGN USING STEPPER MOTORS

**A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of
Dokuz Eylül University
In Partial Fulfillment of the Requirements for
the Degree of Master of Science in Electrical-Electronics
Engineering, Electronics-Telecommunication Program**

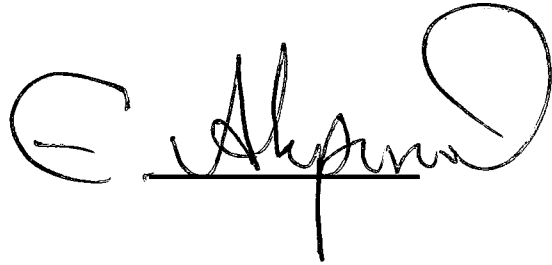
**By
Ebru (AYVAZ) TAŞCI**

September, 1998

İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We certify that we have read this thesis and that in our opinion it is fully adequate,
in scope and in quality, as a thesis for the degree of Master of Science.



Assoc.Prof.Dr.Eyüp Akpınar

(Advisor)



Prof.Dr.E.Uyan

(Committee Member)



Dr. E. UNGAN

(Committee Member)

Approved by the
Graduate School of Natural and Applied
Sciences



Prof.Dr. Cahit Helvacı

ACKNOWLEDGEMENTS

I would like to express my gratitude to Assoc.Prof.Dr. Eyüp Akpınar for his kind supervision, supports and valuable comments.

I wish to thank Türk Telekom Satellite Communication Centre Türksat Satellites Ground Control Station for the facilities provided that made this work possible.

I wish to thank my dear husband Alper Taşcı for his never-lasting encouragement and guidance at all stages of this work and to my parents for their supports.

I wish to thank my dear friend and colleague Sinan Özcan for his help in the design of the system software and my dear colleague Y.Göksel Eren for his valuable comments.



ABSTRACT

The object of this thesis is to design and implement computer aided satellite antenna control unit system for open loop control of its motion in two axis.

In order to control the direction of satellite antenna in this system, stepper motors are used. The motions of the stepper motors are controlled according to the commands directed from a computer via parallel port.

In this design, software has a great importance on the system performance. In order to control the motion of stepper motors and to make the necessary calculations and the visual design, the system software has been prepared. The programming language Borland C 5.0++ for Windows95 is used. The system software consists of three sections.

In first section, the coordinates (longitude and latitude) of any selected point in Turkey map are calculated. In second section, azimuth and elevation angles which antenna will rotate in the same point are calculated by using longitude and latitude values. In last section, two stepper motors are rotated according to the calculated azimuth and elevation angles. The antenna must have been set to a reference position before and after every installation for making the proper rotation.

ÖZET

Bu tezde, uydu anteninin iki boyuttaki hareketinin açık döngü denetimi için bilgisayar destekli uydu anteni kontrol ünitesinin tasarlanması ve gerçekleştirilmesi amaçlanmıştır.

Bu sistemde, uydu antenini hareket ettirebilmek için step motorlar kullanılmıştır. Step motorların hareketi, paralel port aracılığı ile bilgisayardan gönderilen komutlar tarafından kontrol edilmektedir.

Bu tasarımda yazılım, sistem performansı üzerinde büyük bir önem taşır. Step motorların hareketini kontrol etmek, gerekli hesaplamaları ve yazılımın görsel tasarımını yapmak amacıyla sistem yazılımı hazırlanmıştır. Sistem yazılımı için, Windows95 ortamında Borland C 5.0++ programlama dili kullanılmıştır. Sistem yazılımı üç bölümden oluşmaktadır. Birinci bölümde, Türkiye haritasında seçilen herhangi bir noktanın koordinatları (enlem ve boylam) hesaplanır. İkinci bölümde, seçilen aynı noktada enlem ve boylam değerleri kullanılarak antenin döneceği azimuth ve elevation açıları hesaplanır. Son bölümde, step motorlar hesaplanan azimuth ve elevation açılarına döndürülür. Antenin doğru pozisyona dönebilmesi için, sistem her kurulma aşamasının öncesinde ve sonrasında resetlenir.

CONTENTS

	Page
Acknowledgements.....	IV
Abstract.....	V
Özet.....	VI
Contents.....	VII
List of Tables.....	X
List of Figures.....	XI
Abbreviations.....	XIII

Chapter One INTRODUCTION

1. INTRODUCTION.....	1
1.1. STEPPER MOTOR ADVANTAGES AND DISADVANTAGES.....	2
1.2. STEPPER MOTOR TYPES.....	4
1.2.1. Variable Reluctance Stepper Motors.....	4
1.2.2. Permanent Magnet Stepper Motor.....	5
1.2.3. Hybrid Stepper Motor.....	5
1.2.4. Comparison of Stepper Motor Types.....	6
1.3. BASICS OF STEPPER MOTORS.....	7
1.3.1. The Rotating Magnetic Field.....	7
1.3.2. Torque Generation.....	8
1.3.3. Phases, Poles and Step Angles.....	9
1.3.4. Stepping Modes.....	10
1.3.4.1. Wave Drive.....	11
1.3.4.2. Full Step Drive.....	11
1.3.4.3. Half Step Drive.....	11
1.3.4.4. Microstepping Drive.....	12

1.4. STEPPER MOTOR PHYSICS.....	12
1.4.1. Statics.....	12
1.4.1.1. Torque vs. Angle Characteristics.....	12
1.4.1.2. Step Angle Accuracy.....	14
1.4.1.3. Mechanical Parameters, Load, Friction, Inertia.....	14
1.4.2. Dynamics.....	14
1.4.2.1. Torque vs. Speed Characteristics.....	14
1.4.2.2. Single Step Response.....	16
1.4.2.3. Resonance.....	17
1.5. STEPPER MOTOR DRIVING.....	18
1.5.1. Winding Resistance and Inductance.....	18
1.6. STEPPER MOTOR DRIVE CIRCUIT SCHEMES.....	20
1.6.1. Flux Direction Control.....	21
1.6.1.1. Unipolar Drive.....	21
1.6.1.2. Bipolar Drive.....	22
1.6.2. Current Control.....	24
1.6.2.1. Resistance Limitation of the Current.....	24
1.6.2.2. The Bilevel L/R Drive.....	25
1.6.2.3. Chopper Control.....	26
1.7. OUTLINE OF THESIS.....	28

Chapter Two

THE DESIGN OF ANTENNA CONTROL UNIT

2. THE DESIGN OF ANTENNA CONTROL UNIT	29
2.1. HARDWARE DESIGN	30
2.1.1. Electromechanical Design	30
2.1.1.1. Stepper Motors.....	30
2.1.2. Electronic Design.....	33
2.1.2.1. Stepper Motor Driver Circuit.....	33
2.1.2.2. Power Supplies.....	40
2.1.2.3. Interfacing to the Computer.....	41

2.1.3. Mechanical Design.....	43
2.2. SOFTWARE DESIGN.....	44
2.2.1. Main Section.....	46
2.2.1.1. Antenna.cpp File.....	46
2.2.1.2. Antenna.rc File.....	46
2.2.1.3. Antenna.h File.....	47
2.2.1.4. Dtostring.cpp File.....	47
2.2.2. Functional Sections.....	48
2.2.2.1. File I / O.....	48
2.2.2.2. Hardware I / O.....	48
2.2.2.3. Map Reading & Location Finding.....	50
2.2.2.4. Azimuth & Elevation Calculation.....	50
2.3. FLOWCHARTS OF SOFTWARE PROGRAMME.....	57

Chapter Three

MOUNTING AND OPERATING THE ANTENNA CONTROL UNIT

3. MOUNTING AND OPERATING THE ANTENNA CONTROL UNIT	64
3.1. MOUNTING.....	64
3.2. OPERATING THE SYSTEM.....	66
3.3. HOW TO USE THE SYSTEM SOFTWARE.....	67

Chapter Four

CONCLUSIONS

4. CONCLUSIONS.....	70
REFERENCES.....	72
APPENDIX A.....	73
APPENDIX B.....	

LIST OF TABLES

	Page
Table 1.1. Excitation sequences for different drive modes.....	12
Table 2.1. Characteristics of Kollmorgen stepper motors.....	31
Table 3.1. Satellite Spacing.....	64



LIST OF FIGURES

	Page
Figure1.1. A typical Stepper Motor.....	1
Figure1.2. Cross section through a variable reluctance stepper motor....	4
Figure1.3. Cross section through a permanent magnet stepper motor....	5
Figure1.4. A cross section of hybrid stepper motor.....	6
Figure 1.5. Exploded drawing illustrating the tooth pitch off-set.....	6
Figure 1.6. Magnetic flux path through a two-pole stepper motor.....	8
Figure 1.7. Unipolar and bipolar wound stepper motors.....	10
Figure 1.8. Torque versus angular position.....	13
Figure 1.9. A typical speed –torque curve of a stepper motor.....	16
Figure 1.10. Single step response versus time.....	17
Figure 1.11. Current waveform in an inductive-resistive circuit.....	19
Figure 1.12. Current waveform in an inductive-resistive circuit.....	20
Figure 1.13. Bipolar and unipolar drive schemes to control the current and the flux direction in the phase winding.....	21
Figure 1.14. Basic unipolar drive.....	21
Figure 1.15. Unipolar drive using 8-lead motor.....	22
Figure 1.16. Simple bipolar drive.....	22
Figure 1.17. Bipolar bridge.....	23
Figure 1.18. Different winding configurations for bipolar drive	23
Figure 1.19. Resistance limitation of the current.....	24
Figure 1.20. The bilevel drive.....	25
Figure 1.21. Chopper Control.....	27
Figure 2.1. Block Diagram of ACU System and Subsystems Design.....	30
Figure 2.2. Performance curves of Kollmorgen stepper motors.....	31
Figure 2.3. Wave drive excitation chart.....	32
Figure 2.4. Half step drive excitation chart.....	32
Figure 2.5. Unipolar wiring diagram for Kollmorgen stepper motors.....	32

Figure 2.6. The functional diagram of Kollmorgen driver 7026M.....	34
Figure 2.7. The pin configuration.....	34
Figure 2.8. PWM output current waveform.....	35
Figure 2.9. PWM control(Run mode).....	35
Figure 2.10. The motor driver circuit diagram of ACU.....	38
Figure 2.11. The configuration of power supply box.....	41
Figure 2.12. 25-way Female D-Type Connector.....	42
Figure 2.13. The Data Configuration of Parallel Port.....	42
Figure 2.14. A General Block Diagram of The System.....	45
Figure 2.15. The Functional Diagram of The System Software.....	45
Figure 2.16. The Data Configurations of Azimuth Positioning.....	49
Figure 2.17. The Data Configurations of Elevation Positioning.....	49
Figure 2.18. Azimuth and elevation.....	51
Figure 2.19. The Azimuth angles with respect to subsatellite point.....	52
Figure 2.20. Triangle TSO.....	53
Figure 2.21. Triangle to calculate elevation.....	53
Figure 2.22. Comparison of elevation angles.....	55
Figure 2.23. Comparison of azimuth angles.....	56
Figure 3.1. AZ / EL Mount.....	65
Figure 3.2. AZ / EL Mount Geometry.....	66
Figure 3.3. The Photograph of Antenna Control Unit.....	66
Figure 3.4. The Visual Design of The System Software.....	69

ABBREVIATIONS

VR :	Variable Reluctance
PM :	Permanent Magnet
HB :	Hybrid
V :	Volt
L :	Inductance
A :	Amper
k Ω :	Kilo Ohm
pF :	Pico Farad
PWM :	Pulse Width Modulation
EIRP :	Effective Isotropic Radiated Power
RC :	Resistance and Capacitance
Tx :	Transmit
Rx :	Receive
G :	Ground
A, AZ :	Azimuth
E, EL :	Elevation
I / O :	Input / Output
CW :	Clock Wise
CCW :	Counter Clock Wise



CHAPTER ONE

INTRODUCTION

Stepper motors are electromechanical devices which convert digital pulses into discrete mechanical movements. A stepper motor essentially a digital input-discrete motion output device, particularly well suited to the application where control signals appear as pulse trains rather than analog voltages. The shaft or spindle of a stepper motor rotates in discrete step increments when electrical command pulses are applied to it in the proper sequence. The motor's rotation has the following several direct relationships to these applied input pulses:

- a) The sequence of the applied pulses is directly related to the direction of motor shafts rotation,
- b) The motor's rotation speed is directly related to the frequency of the input pulses,
- c) The length of rotation is directly related to the number of input pulses applied (Ericsson).

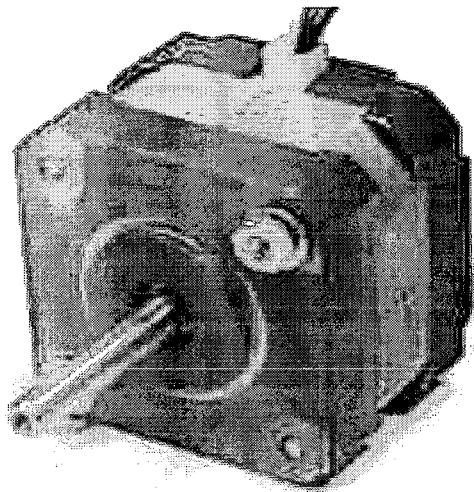


Figure 1.1. A typical Stepper Motor

The first known stepper motors were used in the British Navy in the early 1930's in a remote positioning system for transmitting shaft rotations. The system was later adapted by the U.S. Navy and used widely in World War II. During the same period, rotary solenoids were used for steering torpedoes, using a serial pulse train. Stepper motors are inherently low-efficiency electromagnetic energy conversion devices when compared with conventional AC and DC motors. However, the advances made in the recent years in the field of digital technology have made stepper motors very desirable devices in many control applications and their performance has been considerably improved in the past years. Today stepper motors are found in many control systems used in industry. Relatively large quantities are being used in the most types of computer peripheral equipment, such as printers, plotters, hard disk drives and memory access mechanisms. Stepper motors are also used in numerical control systems, machine tool controls, process control systems, and in medical and office equipments, automotive and aerospace industry and many more (Özdamar, 1976).

The object of this thesis is to design computer aided satellite antenna control system capable of pointing the satellite antenna to the desired satellite (particularly TÜRKSAT satellites) in the selected location of TURKEY map. In this way, satellite antenna will be controlled in two axis using stepper motors, stepper motors will rotate according to the commands directed from a computer via parallel port.

Before going into the details of the designed satellite antenna control unit system, a brief summary of the stepper motors and the drive circuits of the stepper motors will be given.

1.1. STEPPER MOTOR ADVANTAGES AND DISADVANTAGES

One of the most significant advantages of a stepper motor is its ability to be accurately controlled in an open loop system. In open loop control, no feedback information about position is needed. This type of control eliminates the need for

expensive sensing and feedback devices such as optical encoders. The position is known simply by keeping track of the input step pulses. A stepper motor can be good choice whenever controlled movement is required. It can be used to have advantage in applications where we need to control rotation angle, speed, position and synchronism. For that reason, in many applications simpler stepper motor and driver systems are taking place of the sophisticated servo systems. But as a result of extensive study on these motors many disadvantages may be altered in the near future. The advantages and disadvantages of stepper motor are listed below.

Stepper motor have the following advantages:

- 1.The rotation angle of the motor is proportional to the input pulse,
- 2.The motor has full torque at standstill (if the windings are energized)
- 3.Precise positioning and repeatability of movement since good stepper motors have an accuracy of 3-5%of a step and this error is non cumulative from one step to the next.
- 4.Excellent response to starting/stopping/reversing.
- 5.Excellent torque at low speeds.
- 6.Very reliable since there are no contact brushes in the motor.Therefore the life of the motor is simply dependant on the life of the bearing.
- 7.The motors response to digital input pulses provides open loop control, making the motor simpler and less costly to control.
- 8.It is possible to achive very low speed synchronous rotation with a load that is directly coupled to the shaft.
- 9.The speed of rotation is proportional to the frequency of the input pulses.
- 10.Stepper motor is mechanically simple, requires little or no maintenance.

Stepper motor have the following disadvantages:

- 1.Resonances can occur if not properly controlled.
- 2.Not easy to operate at extremely high speed (Ericsson).

1.2. STEPPER MOTOR TYPES

Although various types of stepper motor have been developed, they all fall into three basic categories:

1. Variable Reluctance (VR)
2. Permanent Magnet(PM or tin can)
3. Hybrid (HB)

1.2.1.Variable Reluctance Stepper Motor

This type of stepper motor has been used for a long time. Figure 1.2 shows a cross section of a typical V.R. stepper motor. This type of motor consists of a soft iron multi-toothed rotor and a wound stator. When the stator windings are energized with DC current the poles become magnetized. Rotation occurs when the rotor teeth are attracted to the energized stator poles. As the rotor does not have a permanent magnet it rotates freely i.e. it has no detent torque.

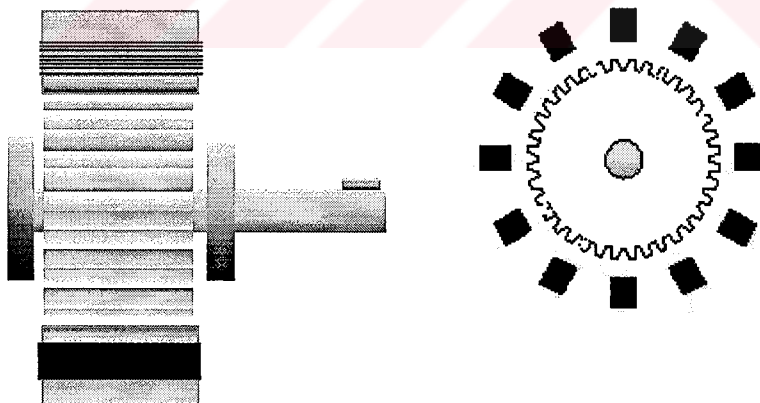


Figure 1.2. Cross section through a variable reluctance stepper motor

1.2.2. Permanent Magnet (PM or tin can) Stepper Motor

Often referred to as a “tin can” or “canstock” motor the permanent magnet stepper motor is a low cost and low resolution type motor with typical step angles of 7.5° to 15° . Figure 1.3 shows a cross section of a typical PM stepper motor. PM motors have permanent magnets added to the motor structure. The rotor no longer has teeth as with the VR motor. Instead the rotor is magnetized with alternating north and south poles situated in a straight line parallel to the rotor shaft. These magnetized rotor poles provide increased magnetic flux intensity and because of this the PM motor exhibits improved torque characteristics when compared with the VR type.

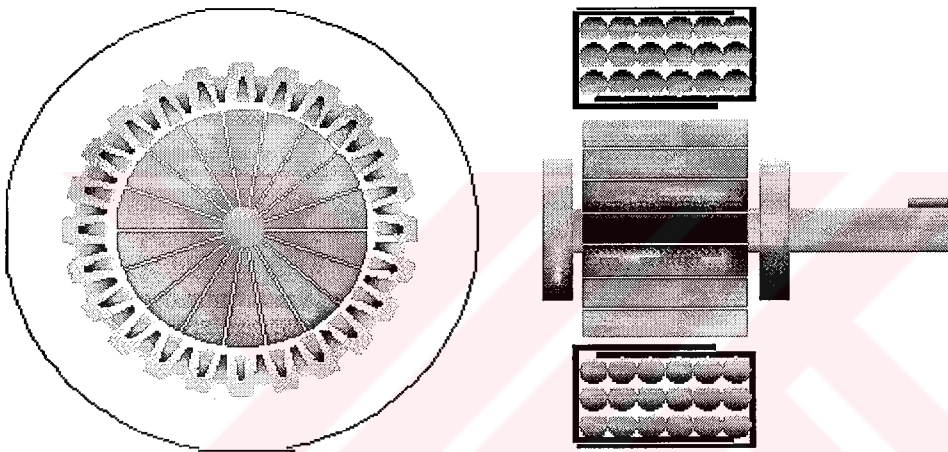


Figure 1.3 Cross section through a permanent magnet stepper motor

1.2.3. Hybrid (HB) Stepper Motor

Another type of stepper motors that has a permanent-magnet rotor is called the hybrid stepper motor. The term hybrid comes from the fact that the motor torque is produced both due to the permanent magnet and variable reluctance action. Figure 1.4 shows a cross section of a typical HB stepper motor. The hybrid stepper motor combines the best features of both the PM and VR type stepper motors. The rotor is multi-toothed like the VR motor and contains an axially magnetized concentric magnet around its shaft, with the opposing teeth off-set by half of one tooth pitch (Figure 1.5) to enable a high resolution of steps. The teeth on the rotor provide an even better path, which helps guide the magnetic flux to preferred locations in the air

gap. This further increases the detent, holding and dynamic torque characteristics of the motor when compared with VR and PM types. Hybrid stepper motors have high detent torque and excellent holding and dynamic torques, and they can operate high stepping speeds (Jennings, 1996).

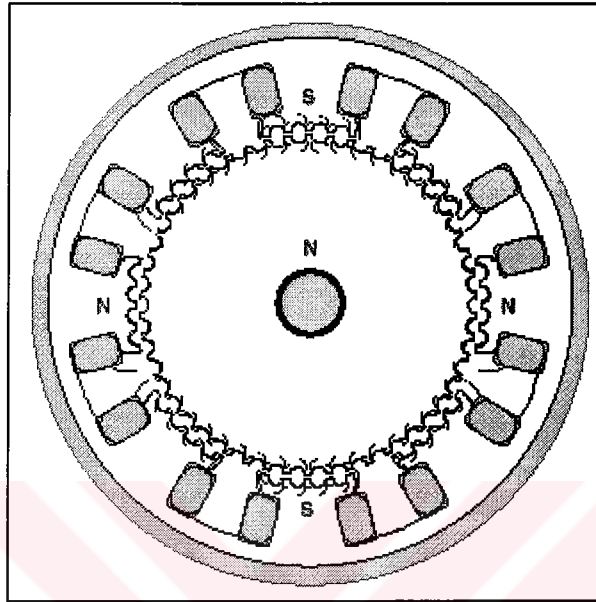


Figure 1.4. A cross section of hybrid stepper motor.

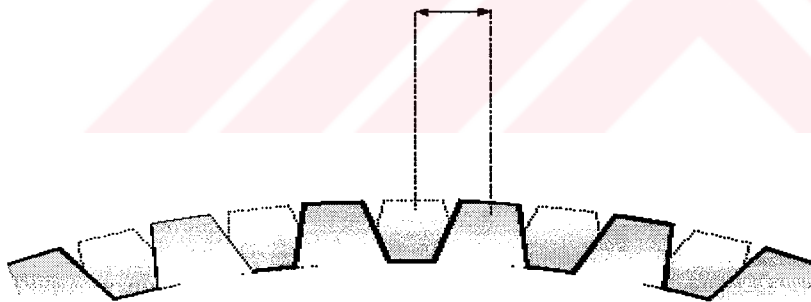


Figure 1.5. Exploded drawing illustrating the tooth pitch off-set

1.2.4. Comparison of Stepper Motor Types

The hybrid stepper motor is probably the most widely used of all stepper motors. It is more expensive than the other types of stepper motors but provides better performance with respect to step resolution, torque and speed. Hybrid stepper motors have a small step length (typically in the range from 3.6° to 0.9°), which can be great advantage when high resolution angular positioning is required. The torque producing capability is greater in the hybrid than in the variable reluctance and

permanent-magnet motors, so the hybrid stepper motor is the best choice of applications requiring a small step length and high torque.

In PM and HB stepper motors, due to the permanent magnet there is a “detent” torque developed in the motor even when stator windings are not excited. Although the detent torque is less than the energised torque, it can be a useful feature in applications where the rotor position must be preserved during a power failure. In VR stepper motor, the rotor teeth have a little residual magnetism, as found in PM and Hybrid motors, so there is no detent torque on the rotor when the stator is not energised.

VR stepper motors have two important advantages when the load must be moved a considerable distance. Firstly, typical step lengths (15 degrees) are longer than in the hybrid type so less steps are required to move a given distance. A reduction in the number of steps implies less excitation changes. The speed of excitation changes can ultimately limit the time taken to move the required distance. A further advantage is that VR stepper motor has a lower motor mechanical inertia than the hybrid and PM stepper motors, because there is no permanent magnet on its rotor. Therefore, this significant reduction in total inertial load on the motor permits faster acceleration (Acarnley, 1984).

1.3. BASICS OF STEPPER MOTORS

1.3.1 The Rotating Magnetic Field

When a phase winding of a stepper motor is energised with current a magnetic flux is developed in the stator. The direction of this flux is determined by the “Right Hand Rule” which states:

“ If the coil is grasped in the right hand with the fingers pointing in the direction of the current in the winding (the thumb is extended at a 90° angle to the fingers), then the thumb will point in the direction of the magnetic field.”

Figure 1.6. shows the magnetic flux path developed when phase B is energised with winding current in the direction shown .The rotor then aligns itself so that the flux opposition is minimised. In this figure, the motor would rotate clockwise so that its south pole aligns with the north pole of the stator B at position 2 and its north pole aligns with the south pole of stator B at position 6. To get the motor to rotate we can see that we must provide a sequence of energising the stator windings in such a way that provides a rotating magnetic field, which the rotor follows due to magnetic attraction.

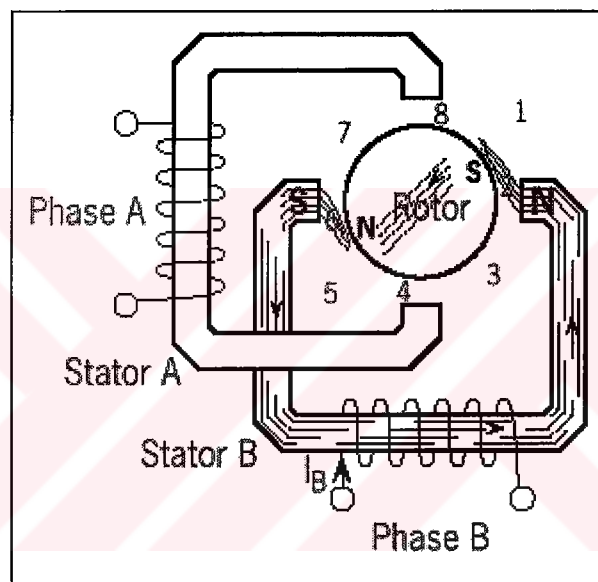


Figure1.6. Magnetic flux path through a two-pole stepper motor with a lag between the rotor and stator.

1.3.2.Torque Generation

The torque produced by a stepper motor depends on the following factors:

- The step rate
- The drive current in the windings
- The type of driver

In a stepper motor a torque is developed when the magnetic fluxes of the rotor and stator are displaced from each other. The stator is made up of a high permeability

magnetic material. The presence of this high permeability material causes the magnetic flux to be confined for the most part to the paths defined by the stator structure. This serves to concentrate the flux at the stator poles. The torque output produced by the motor is proportional to the intensity of the magnetic flux generated when the winding is energised.

The basic relationship that defines the intensity of the magnetic flux is defined by:

$$H = (N \times i) / l$$

Where:

N = The number of winding turns

i = Current

H = Magnetic field intensity

l = Magnetic flux path length

This relationship shows that the magnetic flux intensity and consequently the torque is proportional to the number of winding turns and the current and inversely proportional to the length of the magnetic flux path.

1.3.3. Phases, Poles and Step Angles

Usually stepper motors have two phases, but three- and five- phase motors also exist. A bipolar motor with two phases has one winding per phase and a unipolar motor has one winding, with a centre tap per phase. Sometimes the unipolar stepper motor is referred to as a “ four-phase motor”, even though it only has two phases.

A **pole** can be defined as one of the regions in a magnetised body where the magnetic flux density is concentrated. Both the rotor and the stator of a stepper motor have poles. Figure 1.6. shows a picture of a two-phase stepper motor having 2 poles for each phase on the stator, and 2 poles on the rotor. In reality several more poles are added to both the rotor and stator structure in order to increase the number of steps per revolution of the motor, or in other words to provide a smaller stepping angles.

It is the relationship between the number of rotor poles and the equivalent stator poles, and the number of phases that determines the full-step angle of a stepper motor.

$$\text{Step Angle} = 360 / (N_{Ph} \times Ph) = 360 / N$$

N_{Ph} = Number of equivalent poles per phase = number of rotor poles

Ph = Number of phases

N = Total number of poles for all phases together.

1.3.4. Stepping Modes

There are four commonly used excitation modes. These excitation sequences for the following drive modes are summarised in Table 1.1.

- Wave Drive
- Full Step Drive
- Half Step Drive
- Microstepping

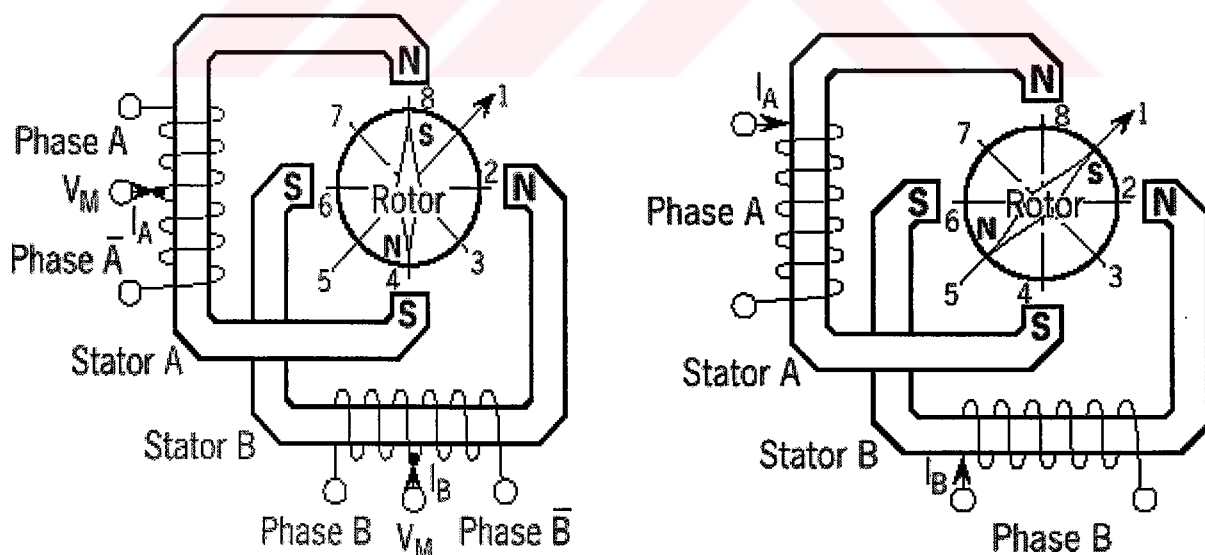


Figure 1.7. Unipolar and bipolar wound stepper motors.

1.3.4.1. Wave Drive:

In wave drive, only one winding is energised at any given time. The stator is energised according to the sequence $A \rightarrow B \rightarrow A' \rightarrow B'$ and the rotor steps from position $8 \rightarrow 2 \rightarrow 4 \rightarrow 6$ referring to Figure 1.7. For unipolar and bipolar wound motors with the same winding parameters this excitation mode would result in the same mechanical position. The disadvantage of this drive mode is that in the unipolar wound motor the energy used is 25% and in the bipolar motor the energy used is 50% of the total motor winding at any given time. This means that the maximum torque output from the motor is not got. Therefore, This mode is only used where torque and speed performances are not important, i.e. where the motor is operated at a fixed speed and load conditions are well defined. Problems with resonance can preclude operation at some speeds.

1.3.4.2. Full Step Drive:

Two phases are energised at any given time. The stator is energised according to the sequence $AB \rightarrow A'B \rightarrow A'B' \rightarrow AB'$ and the rotor steps from position $1 \rightarrow 3 \rightarrow 5 \rightarrow 7$ referring to Figure 1.7. Full step mode results in the same angular movement as wave drive but one half of a full step offsets mechanical position. The torque output of the unipolar wound motor is lower than the bipolar motor since the unipolar motor uses only 50% of the available winding while the bipolar motor uses the entire winding. Therefore, this mode provides good torque and speed performance with a minimum resonance problems.

1.3.4.3. Half Step Drive:

Half step drive combines both wave and full step drive modes. The stator is energised according to the sequence $AB \rightarrow B \rightarrow A'B \rightarrow A' \rightarrow A'B' \rightarrow B' \rightarrow AB' \rightarrow A$ and the rotor steps from position $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8$. This results in angular movements that are half of normal step size. Half-stepping can reduce resonance, which can be experienced in wave and full step drive modes.

1.3.4.4. Microstepping Drive:

In Microstepping drive the currents in the windings are continuously varying to be able to break up one full step into many smaller discrete steps. For example, a standard 1.8° motor has 200 steps/revolution. If the motor is micro-stepped with a 'divide-by-10', then each micro-step would move the motor 0.18° and there would be 2000 steps/revolution. Typically, micro-step modes range from divide-by-10 to divide-by-256 (51200steps/rev for 1.8° motor). The micro-steps are produced by proportioning the current in the two windings according to sine and cosine functions. This mode is only used where smoother motion or more resolution is required (Ericsson).

Table 1.1. Excitation sequences for different drive modes

	Wave Drive				Normal full step				Half-step drive							
Phase	1	2	3	4	1	2	3	4	1	2	3	4	5	6	7	8
A
B						
\overline{A}					
\overline{B}			

1.4. Stepper Motor Physics

1.4.1. Statics

1.4.1.1. Torque versus Angle Characteristics

The torque versus angle characteristics of a stepper motor are the relationship between the displacement of the rotor and the torque which applies to the rotor shaft when the stepper motor is energised at its rated voltage. An ideal stepper motor has a sinusoidal torque versus rotor angular position as shown Figure 1.8. The actual shape of this curve depends on the pole geometry of both rotor and stator, and neither this curve nor the geometry information is given in the motor data sheets

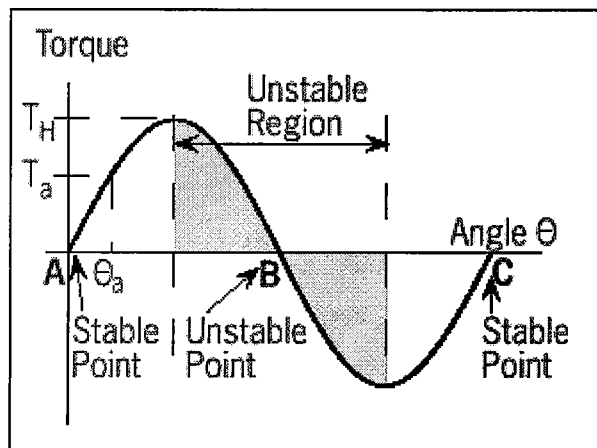


Figure 1.8. Torque versus angular position

Positions A and C represent stable equilibrium points when no external force or load is applied to the rotor shaft. When we apply an external force T_a to the motor shaft we in essence create an angular displacement, θ_a . This angular displacement, θ_a , is referred to as lead or lag angle depending on whether the motor is actively accelerating or decelerating. When the rotor stops with an applied load it will come to rest at the position defined by this displacement angle. The motor develops a torque, T_a , in opposition to the applied external force in order to balance the load. As the load is increased the displacement angle also increases until it reaches the maximum holding torque, T_H , of the motor. Once T_H is exceeded the motor enters an unstable region. In this region, a torque in the opposite direction is created and the rotor jumps over the unstable point to the next stable point. The displacement angle is determined by the following relationship:

$$X = (Z \div 2\pi) \times \sin (T_a \div T_H) \quad \text{where:}$$

Z = rotor tooth pitch

T_a = load torque

T_H = motors rated holding torque

X = displacement angle

1.4.1.2. Step Angle Accuracy

One reason why the stepper motor has achieved such popularity as a positioning device is its accuracy and repeatability. Typically stepper motors will have a step angle accuracy of 3 – 5 % of one step. This error is also non-cumulative from one step to step. The accuracy of the stepper motor is mainly a function of the mechanical precision of its parts and assembly.

1.4.1.3 Mechanical Parameters, Load, Friction, Inertia

The performance of a stepper motor system (motor and driver) is also highly dependent on the mechanical parameters of the load. The load is defined as what the motor drives. It is typically frictional, inertial or a combination of the two.

Friction is the resistance to motion due to the unevenness of surfaces, which rub together. Friction is constant with velocity. A minimum torque level is required throughout the step in order to overcome this friction (at least equal to the friction). Increasing a frictional load lowers the top speed, lowers the acceleration and increases the positional error. The converse is true if the frictional load is lowered.

Inertia is the resistance to changes in speed. A high inertial load requires a high inertial starting torque and the same would apply for braking. Increasing an inertial load will increase speed stability, increase the amount of time it takes to reach a desired speed and decrease the maximum self start pulse rate. The converse is true if the inertia is decreased.

1.4.2.Dynamics

1.4.2.1.Torque versus Speed Characteristics

Under full dynamic conditions, the performance of the motor is described by the torque-speed curve as shown in the Figure 1.9. To get better understanding of this curve it will be useful to define the different aspect of this curve.

Holding Torque:

The maximum torque produced by the motor at standstill.

Detent Torque:

The maximum torque that can be applied to the shaft of an unexcited motor without causing continuous rotation. The detent torque appears only in motors having a permanent magnet.

Pull-in Curve:

The pull-in curve defines an area referred to as the start stop region. This is the maximum frequency at which the motor can start/stop instantaneously, with a load applied, without loss of synchronism.

Maximum Pull-In Rate (Start Rate):

The maximum starting step frequency with no load applied.

Pull-Out Curve:

The pull out curve defines an area referred to as the slew region. It defines the maximum frequency at which the motor can operate without losing synchronism.

Maximum Pull-Out Rate (Slew Rate):

The maximum operating frequency of the motor with no load applied.

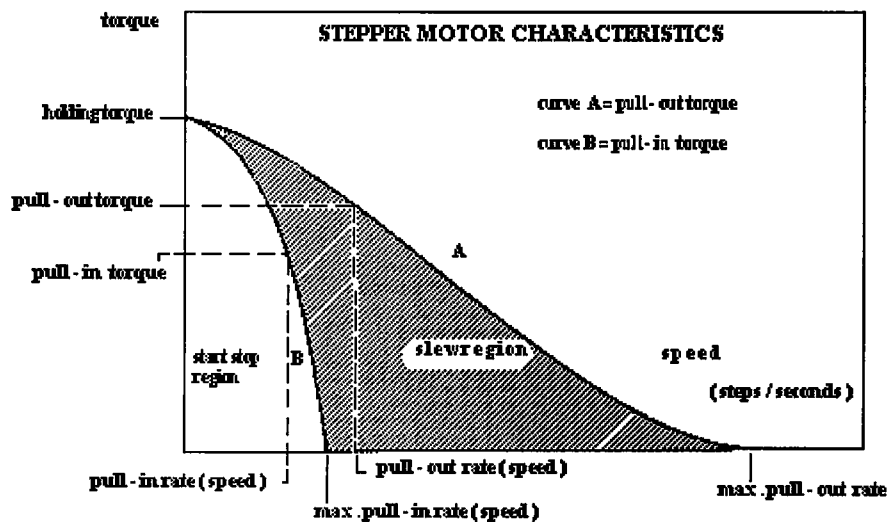


Figure 1.9. A typical speed – torque curve of a stepper motor

As shown in the Figure 1.9, there are two operating ranges, the start /stop (or pull in) region and the slew (pull out) region. Within the start/stop region, the motor can be started or stopped by applying pulses at constant frequency. At speeds within this range, the motor has sufficient torque to accelerate its own inertia. Clearly, if an inertial load is added, this speed range is reduced. So the start/stop speed range depends on the load inertia.

It can be seen from the shape of the curve that step rate affects the torque output capability of stepper motor. The torque output decreases as the speed increases. To operate the motor at faster speeds, it is necessary to start at a speed within the start/stop region and then accelerate the motor into the slew region. Similarly, when stopping the motor, it must be decelerated back into the start/stop range. Using acceleration and deceleration “ramping” allows much higher speeds to be achieved (Parker Automation).

1.4.2.2. Single Step Response

The single step response characteristics of a stepper motor is shown in Figure 1.10. When one step pulse is applied to a stepper motor, the rotor behaves in a

manner as defined by the curve in the figure. The step time t is the time it takes the motor shaft to rotate one step angle once the first step pulse is applied.

Since the torque is a function of the displacement it follows that the acceleration will also be. Therefore, when moving in large steps a high torque is developed and consequently a high acceleration. This can cause overshoots and ringing as shown. The settling time T is the time it takes these oscillations or ringing to cease. In applications, this phenomena can be undesirable. It is possible to reduce or eliminate this behaviour by microstepping.

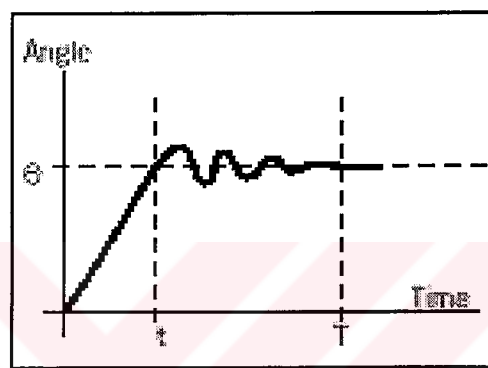


Figure 1.10. Single Step Response versus time

1.4.2.3. Resonance

Stepper motors can often exhibit a phenomena referred to as resonance at certain step rates. This can be seen as a sudden loss or drop in torque at certain speeds which can result in missed steps or loss of synchronism. It occurs when the input step pulse rate coincides with the natural oscillation frequency of the rotor. Often there is a resonance area around the 100-200 pps region and also one in the high step pulse rate region. The resonance phenomenon of a stepper motor comes from its basic construction and therefore it is not possible to eliminate it completely. It is also dependent on the load conditions. It can be reduced by driving the motor in half or micro step driving.

1.5. STEPPER MOTOR DRIVING

In this section, basic electrical characteristics of a stepper motor winding and the various types of stepper motor drivers are discussed.

1.5.1. Winding Resistance and Inductance

Resistance and inductance are two inherent physical properties of the winding of a stepper motor. These two basic factors limit the possible performance of the motor. The resistance of the windings is responsible for the major share of the power loss and heat up of the motor. The power loss is given by:

$$P_R = R \cdot I_M^2$$

Inductance makes the motor winding oppose current changes, and therefore limits high-speed operation. Figure 1.11. shows the electrical characteristic of an inductive-resistive circuit. When a voltage is connected to the winding the current rises according to the equation:

$$I(t) = (V/R) \cdot (1 - e^{-t \cdot R/L})$$

Initially the current increases at a rate of

$$\delta I / \delta t (0) = V/L$$

The rise rate decreases as the current approaches the final level:

$$I_{MAX} = V/R$$

The value of $\tau_e = L/R$ is defined as the electrical time constant of the circuit. τ_e is the time until the current reaches 63% ($1 - 1/e$) of its final value. When the inductive-resistive circuit is disconnected at the instant $t = t_1$, the current starts to decrease:

$$I(t) = (V/R) \cdot e^{-(t-t_1) \cdot R/L}$$

at an initial rate of $I(t) = -V/L$

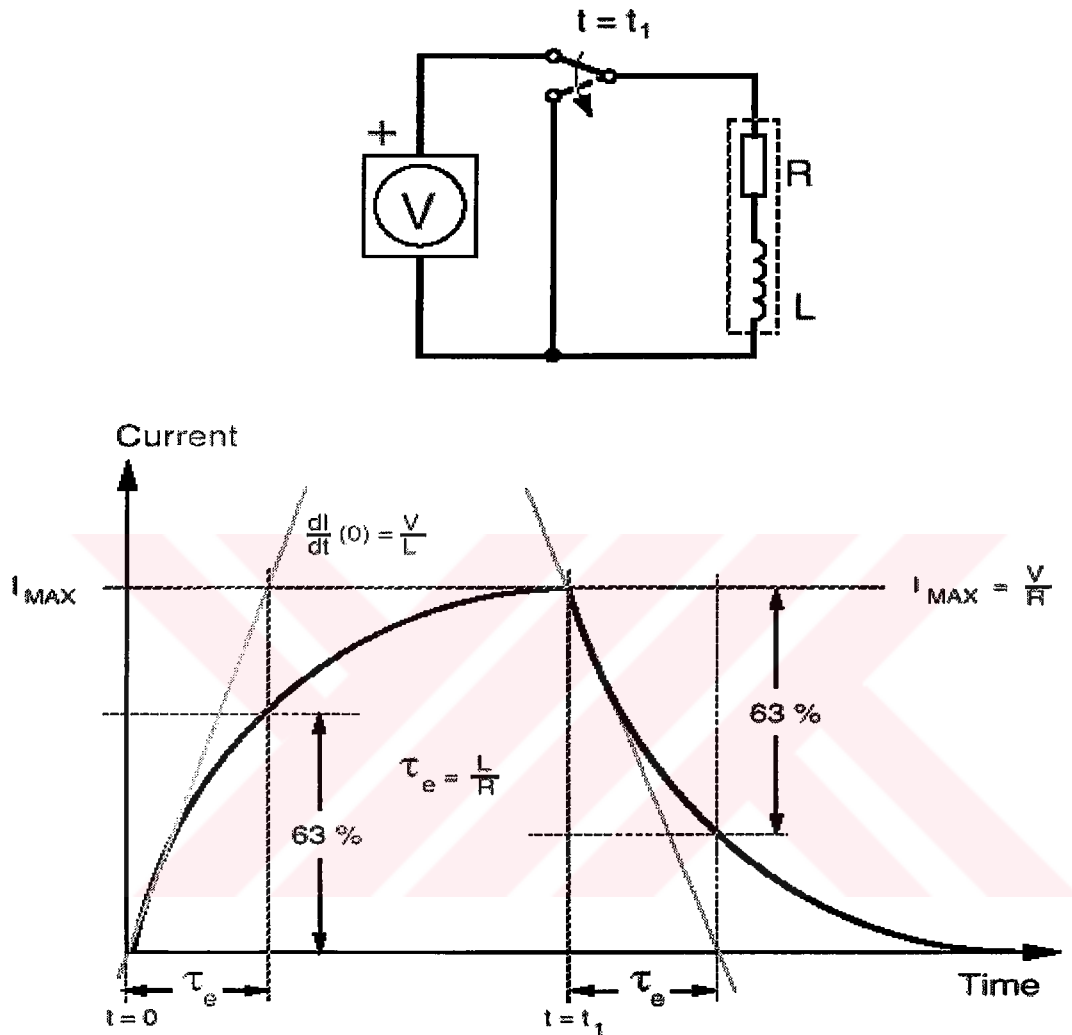


Figure 1.11 Current waveform in an inductive- resistive circuit

When a square wave voltage is applied to the winding, which is the case when full stepping a stepper motor, the current waveform will be smoothed. Figure 1.12 shows the current at a certain frequency that the current reaches its maximum value.

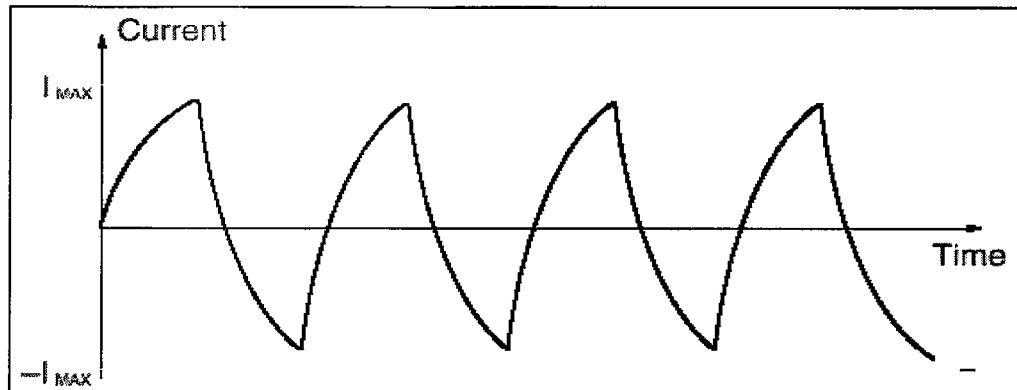


Figure 1.12. Current wave form in an inductive-resistive circuit.

As the torque of the motor is approximately proportional to the current, the maximum torque is reduced as the stepping frequency increases. To overcome the inductance and gain high-speed performance of the motor two possibilities exist:

- Increase the current rise rate (V / L)
- Decrease the time constant

If the resistance is increased, the power loss increases. It is preferably the ratio V / L that should be increased to gain high-speed performance. To drive current through the winding, we should:

- use as high voltage as possible
- keep the inductance low.

So, a low inductance/resistance motor has a higher current rating. As the maximum current is limited by the driver, we find that high performance is highly dependent on the choice of driver. The limiting factor of the motor is the power dissipation, and not the current itself. To use the motor efficiently, power dissipation should be at the maximum allowed level (Ericsson).

1.6. STEPPER MOTOR DRIVE CIRCUIT SCHEMES

The stepper motor driver circuit has two major tasks:

- To change the current and flux direction in the phase windings
- To drive a controllable amount of current through the windings

1.6.1. Flux Direction Control

Stepping of the stepper motor requires a change of the flux direction, independently in each phase. The direction change is done by changing the current direction, and may be done in two different ways, using a bipolar or a unipolar drive. Figure 1.13 shows the two schemes.

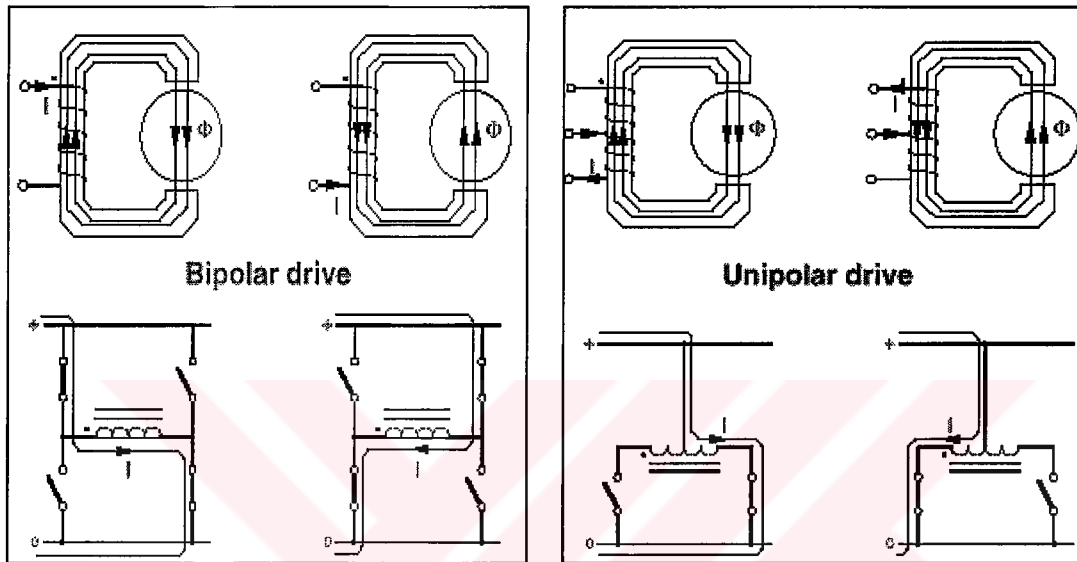


Figure 1.13 Bipolar and unipolar drive schemes to control the current and the flux direction in the phase winding.

1.6.1.1. Unipolar Drive

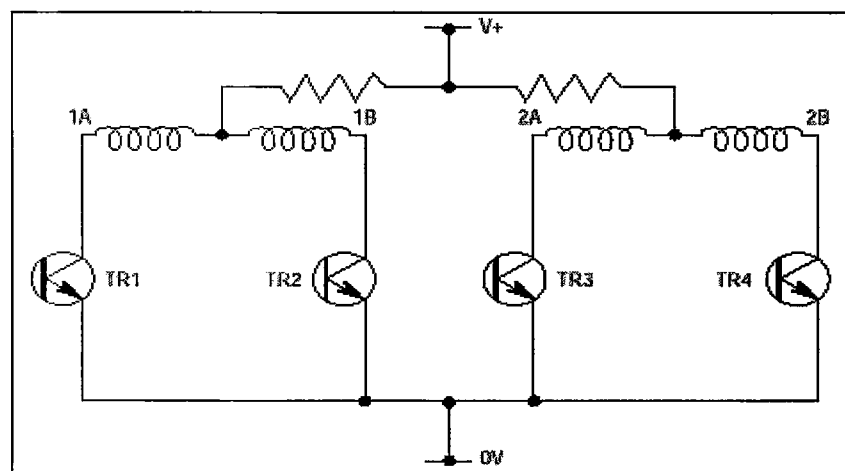


Figure 1.14 Basic Unipolar Drive

The unipolar drive principle requires a winding with a centre-tab, or two separate windings per phase. (See Figure 1.7.) Flux direction is reversed by moving the current from one half of the winding to the other half. A drawback of the unipolar drive is its inability to utilize all the coils on the motor. At any time, there will only be current flowing in one half of each winding. If we could utilize both sections at the same time, we could get a 40% increase in torque for the same power dissipation in the motor. The unipolar motor has three leads per phase. A motor having two separate windings per phase is usually referred to as 8-lead motor (See Figure 1.15) (Parker Automation)

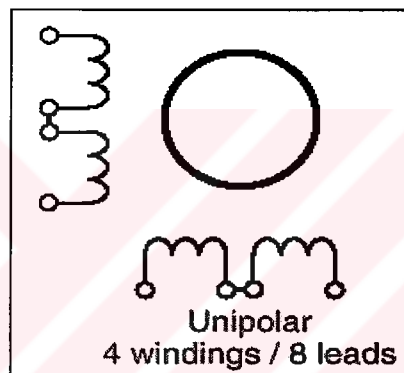


Figure1.15 Unipolar drive using an 8-lead motor

1.6.1.2. Bipolar Drive

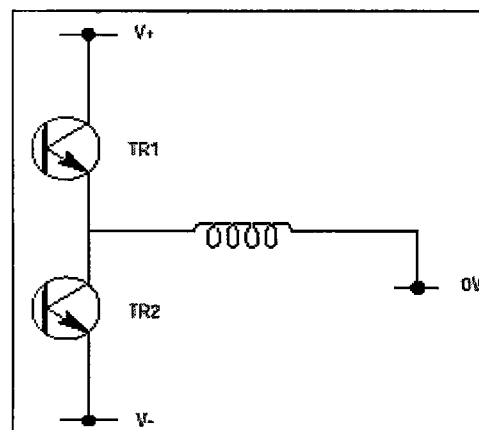


Figure 1.16 Simple Bipolar Driver

In bipolar drive circuit, shown in the Figure 1.16, two power supplies are used together with a pair of switching transistors. Bipolar drive refers to the principle where the current direction in one winding is changed by shifting the voltage polarity across the winding terminals. Current can be made to flow in either direction through the winding by turning on one transistor or the other. However, there are distinct drawbacks to this scheme. First, two power supplies are needed. When all the current is coming from one supply the other is doing nothing at all, so the power supply utilization is poor. Second, the transistors must be rated at double the voltage that can be applied across the motor, requiring the use of costly components. The standard arrangement used in bipolar motor drives is the bridge system shown in Figure 1.17. Although this uses an extra pair of switching transistors, the problems associated with the previous configuration are overcome and only one power supply is needed (Parker Automation).

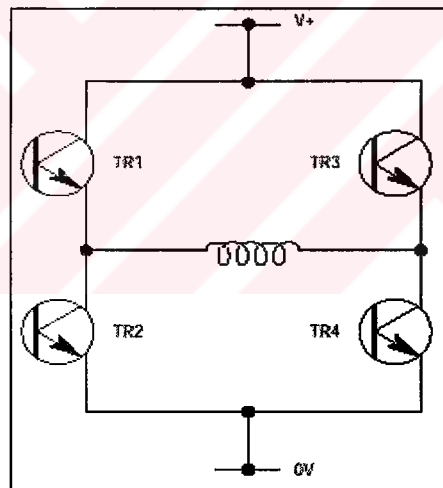


Figure 1.17 Bipolar Bridge

The bipolar drive method requires one winding per phase. A two-phase motor will have two windings and accordingly four connecting leads. (See Figure 1.18)

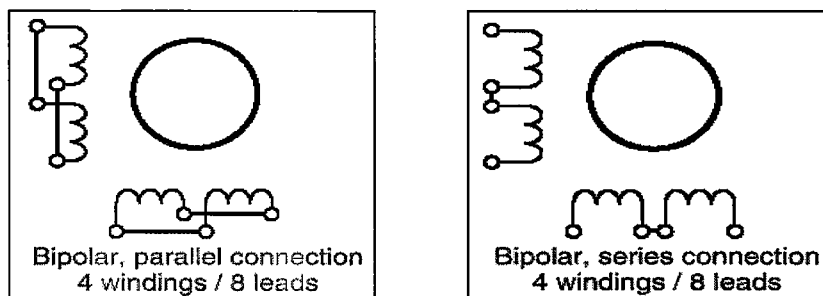


Figure 1.18 Different configurations for bipolar drive using an 8-lead motor.

1.6.2. Current Control

To control the torque as well as to limit the power dissipation in the winding resistance, the current must be controlled or limited. Two principles to limit the current are described here, the resistance limited drive and the chopper drive. Any of the methods may be realised as a unipolar or bipolar driver.

1.6.2.1. Resistance limitation of the current (L / R drive)

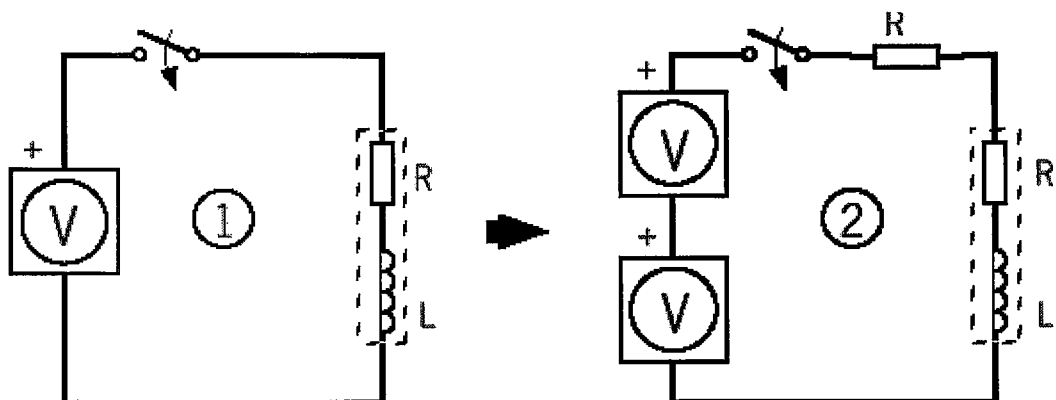
In this basic method the current is limited by supply voltage and the resistance of the winding, and an additional external resistance (dropping resistor):

$$I_M = V_{\text{supply}} / (R + R_{\text{ext}})$$

For a given motor, increasing the supply voltage increases high-speed performance. An increased supply voltage in the resistance limited drive must be accompanied by an additional resistor (R_{ext}) in series with the winding to limit the current to the previous level. The time constant:

$$\tau_e = L / (R + R_{\text{ext}})$$

decreases, which shortens the current rise time (See Fig.1.19). The drawback of using this method is the power loss in the additional external resistors.



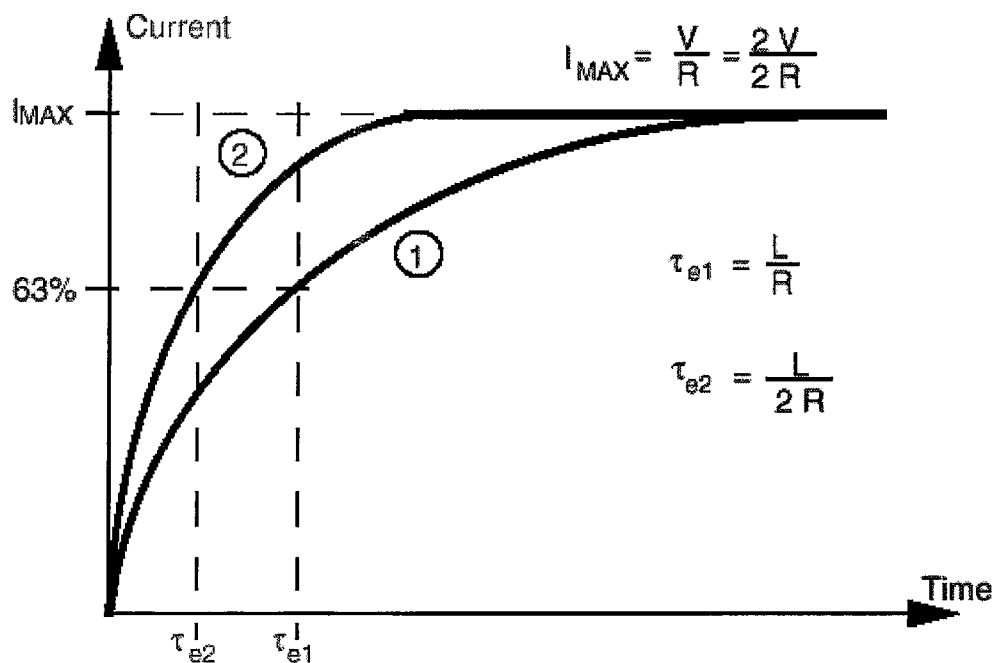
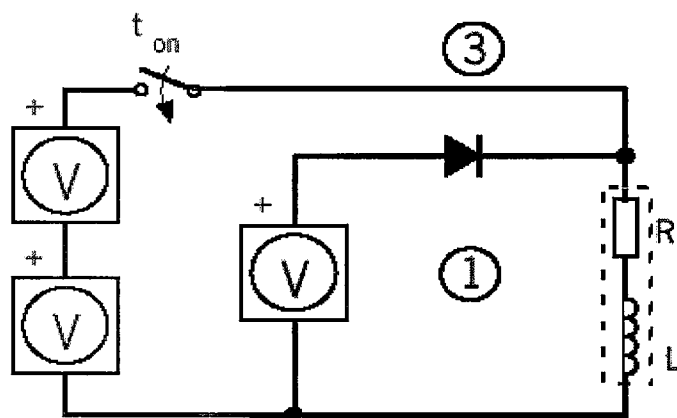


Figure 1.19. Resistance limitation of the current

1.6.2.2. The Bilevel L / R – Drive

The bilevel L/R drive provides a solution to the power waste using dropping resistors. In the beginning of the current build-up period, the winding is connected to a secondary high voltage supply. When the current has reached its nominal level, the second level supply is disconnected. (See Figure 1.20.)

The disadvantage of bilevel drive is the need of a second level power supply. So this method costs very much.



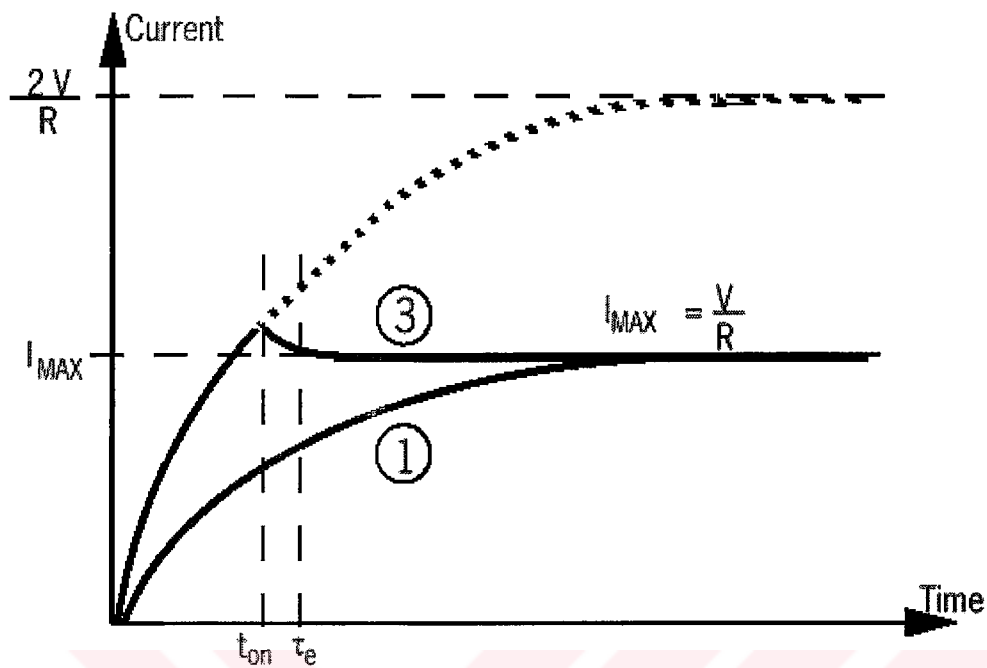


Figure 1.20. The Bilevel Drive

1.6.2.3. Chopper Control

Another method of current control used in most stepper drives is the chopper control. (Figure 1.21.) Chopper driver provides an optimal solution to current control. This approach consists of the four-transistor bridge, recirculation diodes and a sense resistor. The resistor is of low value (typically 0.1Ω) and provides a feedback voltage proportional to the current in the motor.

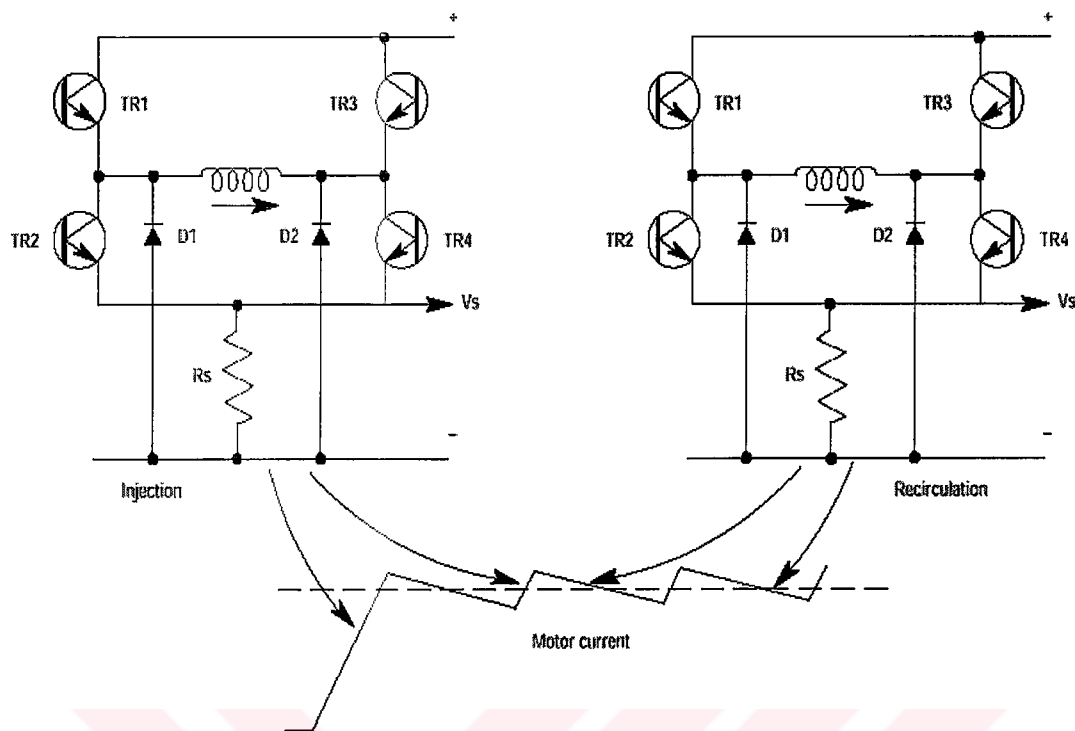


Figure 1. 21. Chopper Control

Current is injected into the winding by turning on one top switch and one bottom switch, and this applies the full supply voltage across the motor. Current will rise in an almost linear and this current can be monitored by looking across the sense resistor. When the required current level has been reached, the top switch is turned off and the stored energy in the coil keeps the current circulating via the bottom switch and the diode. Losses in the system cause this current to slow decay, and when pre-set lower threshold is reached, the top switch is turned back on and the cycle repeats. The current is therefore maintained at the correct average value by switching or “chopping” the supply to the motor.

This method of current control is very efficient because very little power is dissipated in the switching transistors other than during the transient switching state. A variant of this current is the regenerative chopper. In this drive, the supply voltage is applied across the motor winding in alternating directions, causing the current to ramp up and down at approximately equal rates. This technique tends to require

fewer components and is consequently lower in cost; however, the associated ripple current in the motor is usually greater and increases motor heating.

The chopper control is very efficient since it does not waste power by dropping voltage through a resistor. However, good current control in the motor is essential to deliver optimum rotor power. Pulse width modulation (PWM) and threshold modulation are two types of chopper control techniques. PWM controls the average of the motor current and is very good for precise current control. Threshold modulation controls current to a peak level. Threshold modulation can be applied to a wider range of motors, but it does suffer greater loss of performance than PWM. Both chopper control techniques improves the power dissipation in the motor and drive and overall system efficiency. As system performance increases, the complexity and cost of the drive increases. (Parker Automation)

1.7. OUTLINE OF THESIS

In this chapter a brief introduction to stepper motors is given. The type of stepper motors, its advantages and disadvantages, its static and dynamic characteristics are investigated. Furthermore, drive circuit basics, drive circuit schemes are studied in some detail.

In Chapter 2, the design criteria for the Antenna Control Unit system will be given and its basic structure will be introduced.

Chapter 3 is concerned with the mounting and operating of the system and software. Chapter 4 is also concerned with the general conclusions and discussions on the overall control system.

The circuit diagrams of power supplies, wiring diagram of all units, some technical drawings and lists of system software are presented in the appendices.

CHAPTER TWO

THE DESIGN OF ANTENNA CONTROL UNIT

Today, almost all TV and radio broadcastings and data communications in the world are provided by satellites. These satellites describe a circular orbit on the equatorial plane at an altitude of 35786 km. This distance is very important, because the period of revolution of these satellites around the centre of the Earth results in 24-hour in this altitude. They are thus synchronous with the Earth's rotation and appear relatively motionless in relation to a reference point on the Earth's surface. This characteristic enables the satellite to provide permanent coverage area for a given region. They are consequently called Geostationary Orbit Satellites (Tri, 1990).

These satellites receive the signals which are directed from the Earth stations, amplify, convert its frequency and send the signals to the fixed coverage areas. In order to receive the signals on the Earth, the diameter of satellite dish antenna, which will be used, shall be calculated according to the downlink EIRP parameter of the satellites. For example, In Turkey, to receive satellite broadcastings will require 60cm. diameter for Turksat satellites and 90-120 cm. diameter antenna for Eutelsat and Arabsat satellites. In the same way, in Sweden, one shall use 2.5m. diameter antenna to receive the broadcastings of Turksat satellites.

Whatever diameter of antenna is, in order to receive signals of good quality, our satellite antenna on the Earth must have been absolutely focused to the satellite antenna. If we want to receive broadcastings from more than one satellite, we must rotate our antenna to focus to the desired satellite. This manual adjustment process wastes time and effort. Therefore, the antenna control unit has been designed to control the position of satellite dish antenna. This designed system is controlled via parallel port of the computer and is focused to the desired satellite automatically with stepper motors.

In previous chapter, a brief summary on stepper motors and detailed explanation of stepper motor driver circuits are given. In this chapter, the main features and structure of the designed system will be explained. The sub-systems of designed system are shown in the following Figure 2.1.

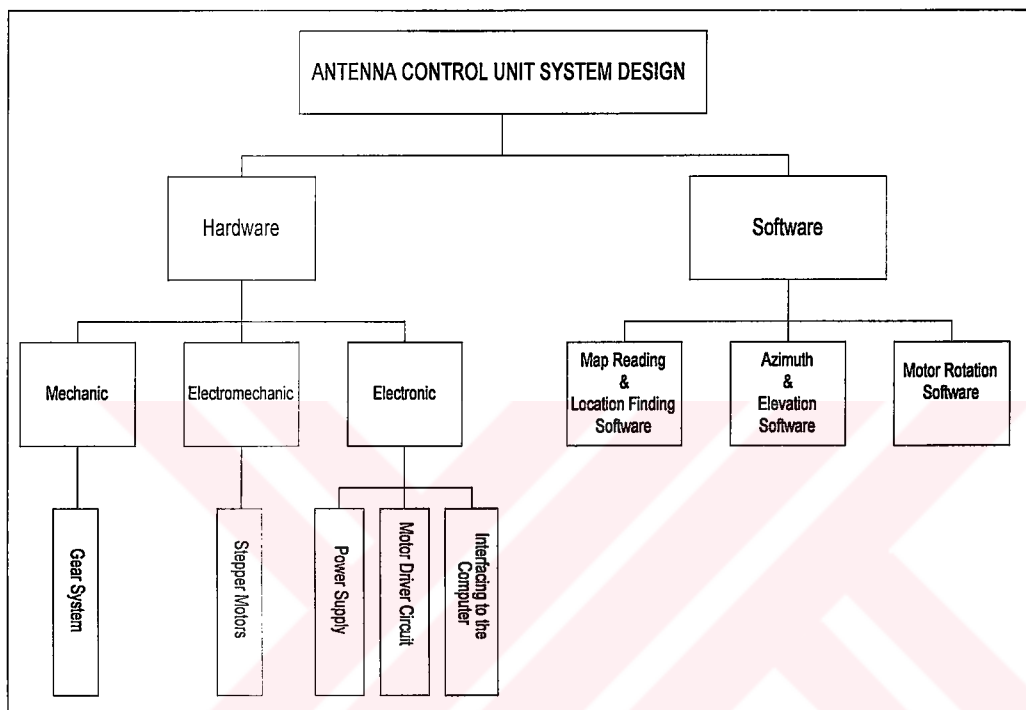


Figure 2.1. Block Diagram of Antenna Control Unit System and Subsystems Design

2.1.HARDWARE DESIGN

2.1.1. Electromechanical Design

2.1.1.1.Stepper Motors

In this project, two hybrid stepper motors are chosen and used. While one of these motors is used for positioning the system in azimuth angle, the other one is used for the positioning the system in elevation angle. The reasons of the selection of hybrid type stepper motors have been discussed in Chapter 1.2.3. These selected motors are Kollmorgen PJT55-A1W and PJT80-A1W. Their characteristics and performance

curves are given in Table 2.1. and Figure 2.2, respectively. Technical drawings of the stepper motors are also presented in Appendix A. This selection has been done by the support of Kollmorgen Motion Technologies Group to operate safely with the defined inertia and friction, and also to use these motors for other purposes.

Table 2.1. Characteristics of Kollmorgen Stepper motors

Models	PJT55-A1W	PJT80-A1W
Step Angles (°)	1.8	1.8
Step Angle Accuracy (%)	± 5	± 5
Number of leads	8	8
Steps per Revolution	200	200
Winding Type	unipolar or bipolar	unipolar or bipolar
DC operating Voltage (V)	3.4	3.36
Operating Current (A/Ø)	2	2
Winding Resistance (Ω/Ø)	1.7	1.68
Winding Inductance (mH/Ø)	1.7	2.2
Holding Torque (oz-in)	83.3	111.1
Rotor Inertia (oz-in ²)	70.3x10 ⁻²	132.6x10 ⁻²
Ambient Temp Range, Operating (C°)	-10 ~+60	-10 ~+60
Temperature Rise (C°)	70	70
Weight (oz)	18.3	29.3

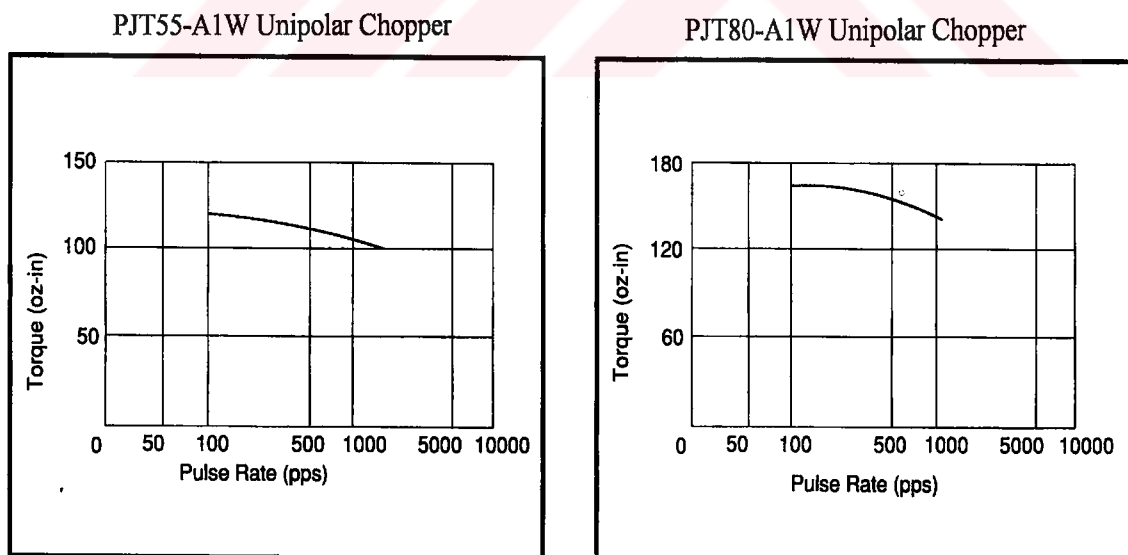
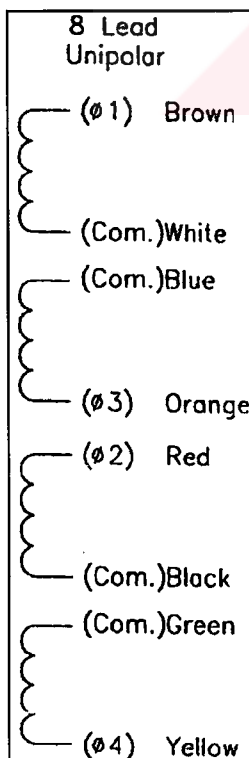


Figure 2.2. Performance Curves of Kollmorgen Stepper Motors

As shown in the Figure 2.2., the stepper motors will not be operated with a rotational speed larger than 1000 steps/second. This graphic shows the start/stop speed range of the motors. (Start/stop speed rate of a stepper motor are the speed values where the motor can start and stop directly without losing any step.) Therefore, any speed value can easily be obtained with staying in start/ stop range of the used stepper motors.

As shown in Table2.1, the stepper motors have 8 leads. This feature allows greater flexibility. The motors can be run as six-lead motor with unipolar drives. With bipolar drives, the windings can then be connected in either series or parallel. The stepper motors used in this project are unipolar wound for unipolar chopper driving. The unipolar connection diagram is shown in Figure2.5. In addition to this, “ Wave Drive “ for Azimuth motor and “ Half Step Drive” for Elevation motor are selected and used. The usage of “ Half Step Drive” has increased the sensitivity of motion in elevation angle and given smoother movement to the Elevation motor. Wave Drive and Half Step Drive excitation charts are shown in the Figures 2.3 and 2.4.(Kollmorgen).



8 Leads Color	Ø 1 A Brown	Ø 2 B Red	Ø 3 \bar{A} Orange	Ø 4 \bar{B} Yellow	Common	Direction of Rotation
Step 1	On	Off	Off	Off	White	CCW
2	Off	On	Off	Off	Blue	
3	Off	Off	On	Off	Black	
4	Off	Off	Off	On	Green	
1	On	Off	Off	Off		CW

Figure 2.3. Wave Drive Excitation Chart

8 Leads Colour	Ø 1 A Brown	Ø 2 B Red	Ø 3 \bar{A} Orange	Ø 4 \bar{B} Yellow
Step 1	On	Off	Off	Off
2	On	On	Off	Off
3	Off	On	Off	Off
4	Off	On	On	Off
5	Off	Off	On	Off
6	Off	Off	On	On
7	Off	Off	Off	On
8	On	Off	Off	On
1	On	Off	Off	Off

Figure 2.4. Half Step Drive Excitation Chart

Figure 2.5. Unipolar Wiring Diagram for Kollmorgen Stepper Motors

The stepper motor winding may be considered as an inductance in series with a resistance. The time constant (τ) of the winding is L / R . It is typically of the order of 10 milliseconds. For the motors used, the time constants are 1 millisecond and 1.3 milliseconds, respectively. Therefore, if a voltage source is applied across the winding, 95% of full current is reached in 3 time constants. To make sure that the winding current reaches 95% of full current at each step, the pulse rate for the used motors must not exceed the values:

$$\frac{1}{3 \times 2 \times 10^{-3}} = 166.6 \text{ steps/second} \quad \frac{1}{3 \times 2 \times 1.13 \times 10^{-3}} = 147.5 \text{ steps/second}$$

In most cases, as in this system, such a severe limitation on motor speed is not acceptable. So it is necessary to build up the winding current more rapidly. (See Chapter 1.5.1. for more details) (Özdeşlik, 1982).

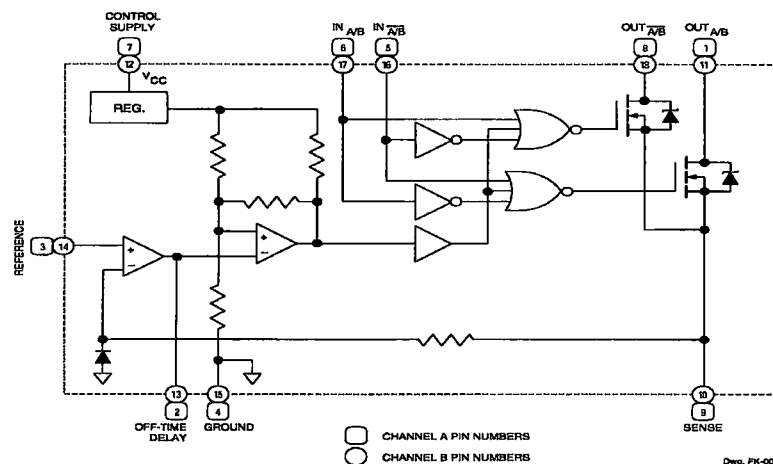
2.1.2. Electronic Design

2.1.2.1. Stepper Motor Driver Circuit

When designing a stepper motor system, it is important to select the right motor and driver type to get the best performance. In the design of this project, at first, the hybrid stepper motors were selected. After that, due to that the motor current is 2A, Kollmorgen 7026M unipolar chopper driver was selected. Because it can drive maximum output current of 3A. Two drivers are used in the system. One driver is used for each motor. The specifications and technical drawings of the drivers are presented in Appendix A. The motor driver circuit diagram used in this system is also given in the Figure 2.10.

The functional diagram of driver 7026M is given in the Figure 2.6. The pin configuration is shown in the Figure 2.7. Drivers are rated for an absolute maximum limit 46V (V_{cc}) and utilise NMOS FETs for the high-current, high-voltage driver outputs. These FETs provide excellent ON resistance, improved body diodes, and very-fast switching. PWM current is regulated by appropriately choosing current-sensing resistors, a voltage reference, a voltage divider, and RC timing networks.

The RC components limit the OFF interval and control current decay. Inputs are compatible with 5V logic. Complete application information is given in the following page (Kollmorgen, Allegro MicroSystems).



Note that channels A and B are electrically isolated.

Figure2.6. The functional diagram of Kollmorgen driver 7026M

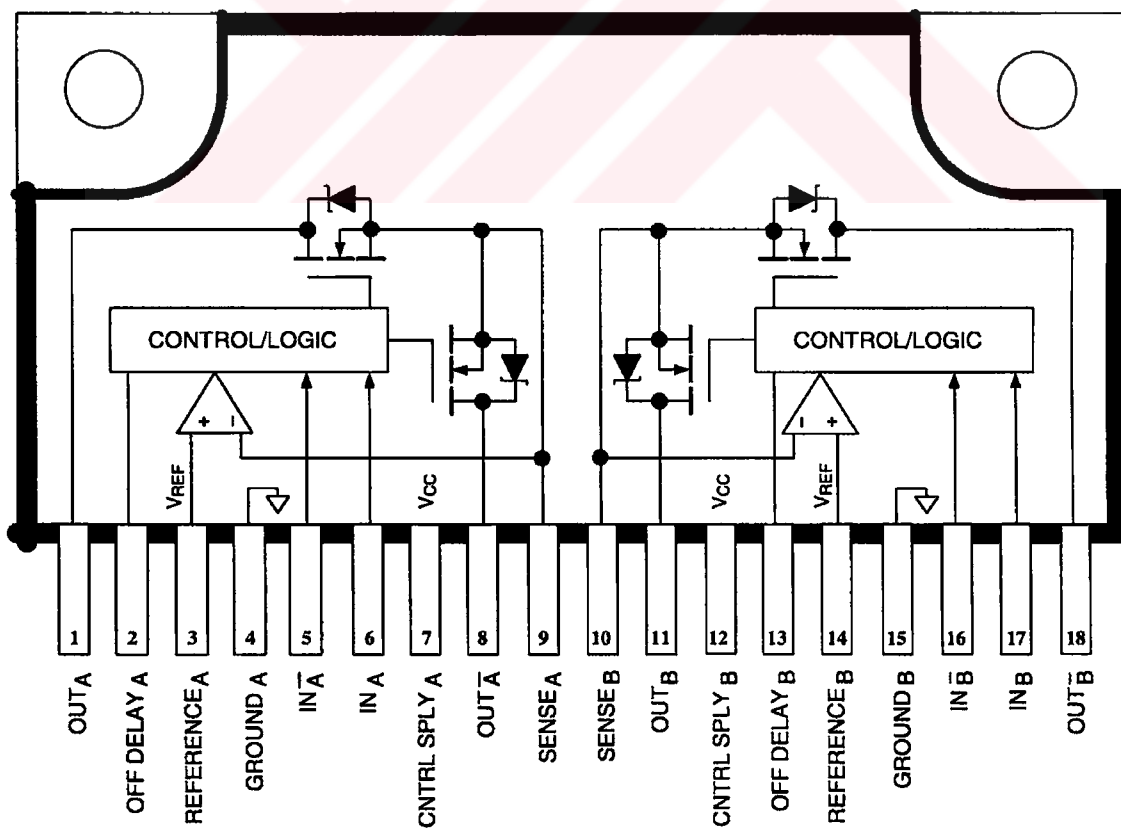


Figure2.7. The Pin Configuration

The regulated PWM output current I_{OUT} (motor coil current) waveform is illustrated in Figure 2.8. Setting the PWM current trip point requires various external components. Using Figure 2.8, the following variables will be selected.

V_b = Reference supply (typically 5V)

R_1, R_2 = Voltage-divider resistors in the reference supply circuit

R_s = Current sensing resistor(s)

$$V_r = V_b \times R_2 / (R_1 + R_2) = 5 \times 100 / 610 = 0.82$$

Note that the maximum allowable V_r input voltage is 2.0V.

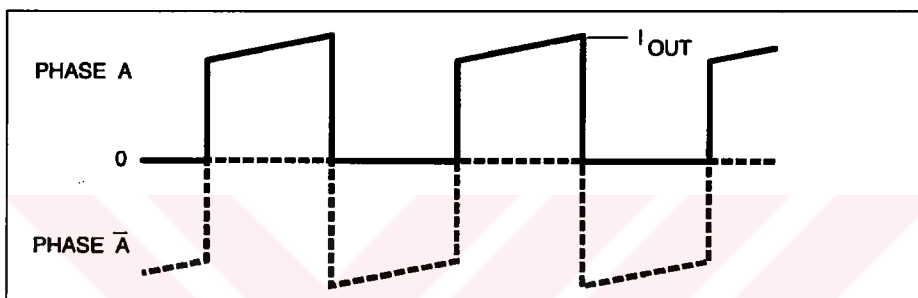


Figure 2.8. PWM output current waveform

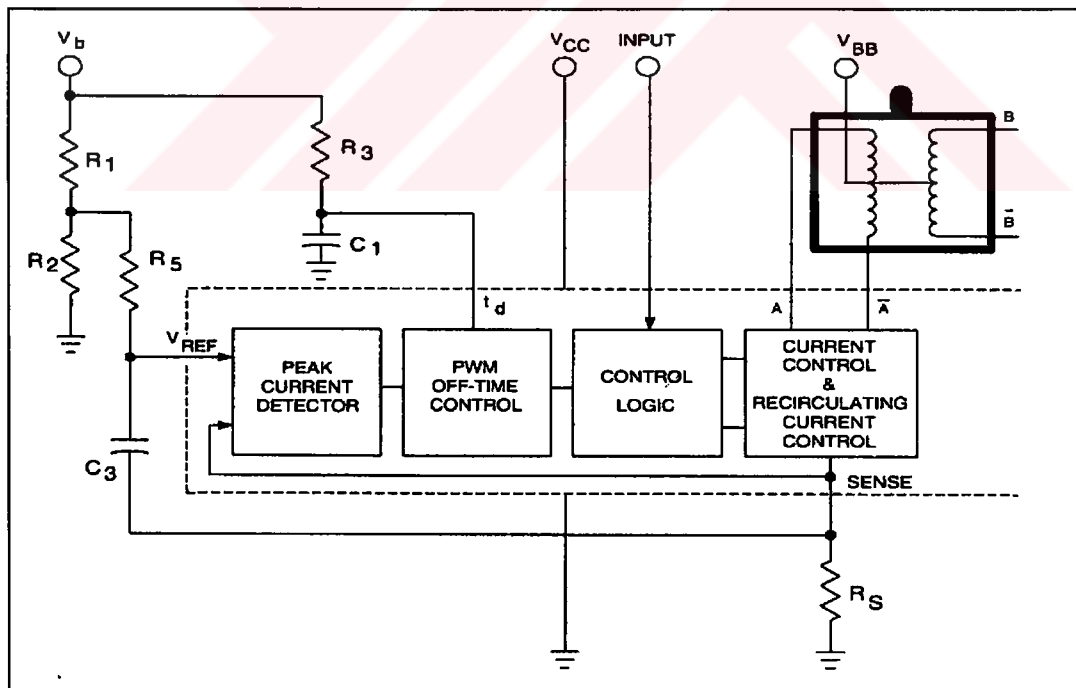


Figure 2.9. PWM control (Run mode)

In normal PWM (Full-current/Running) mode, I_{out} is set to meet the specified running current for the motor (Figure 2.9.) and is determined by:

$$I_{out} = V_{ref} / R_s = 0.82 / 0.4 = 2.05 \text{ A}$$

The Motor PWM Frequency is determined according to the following equation. PWM OFF time fixed by $R3$ and $C1$ at input Td . The OFF time can be calculated as:

$$T_{off} \approx -R3 \times C1 \times \log(1 - 2 / V_b)$$

In the designed system, circuit constants and T_{off} are:

$$V_b = 5V$$

$$R3 = 47k\Omega$$

$$C1 = 470pF$$

$$T_{off} = 12\mu s$$

When designing this system, the heatsink is not needed for the drivers. But, it could be use for preventive. The basic constituents of conduction losses (internal power dissipation) include:

- a- FET output power dissipation ($I_{out}^2 \times R_{ds(on)}$ or $I_{out} \times V_{ds(on)}$),
- b- FET body diode power dissipation ($V_{sd} \times I_{out}$),
- c- Control circuit power dissipation ($V_{cc} \times I_{cc}$).

Device conduction losses are calculated based on the operating mode:

$$\text{Wave Drive} = 0.5 (I_{out}^2 \times R_{ds(on)}) + 0.5 (V_{sd} \times I_{out}) + (V_{cc} \times I_{cc})$$

$$\text{Half-step Drive} = 0.75 (I_{out}^2 \times R_{ds(on)}) + 0.75 (V_{sd} \times I_{out}) + (V_{cc} \times I_{cc})$$

In the system designed, wave drive operation mode is used for the azimuth motor. Half-step drive is used for the elevation motor. The following necessary values are electrical characteristics of the 7026M driver that are shown in appendix A. In addition to this, the current and voltage graphics of the motor phases A and B are shown in Appendix A.

$$I_{out} = 2.05 \text{ A}, I_{cc} = 10\text{mA (typical)}$$

$$R_{ds(on)} = 285 \text{ m}\Omega$$

$$V_{sd} = 0.9 \text{ V (typical value)}$$

$$V_{cc} = 34.4 \text{ V (Control Supply Voltage.)}$$

The power dissipation in Azimuth motor driver:

$$P = 0.5 (2.05^2 \times 285 \text{ m}\Omega) + 0.5 (0.9 \times 2.05) + (34.4 \times 10\text{mA})$$

$$= \mathbf{1.9 \text{ W}},$$

and the power dissipation in Elevation motor driver:

$$P = 0.75 (2.05^2 \times 285 \text{ m}\Omega) + 0.75 (0.9 \times 2.05) + (34 \times 10\text{mA})$$

$$= \mathbf{2.6 \text{ W}}.$$

Thus, it can be seen that half-step operation mode wastes more power than the wave drive operation mode in the driver.



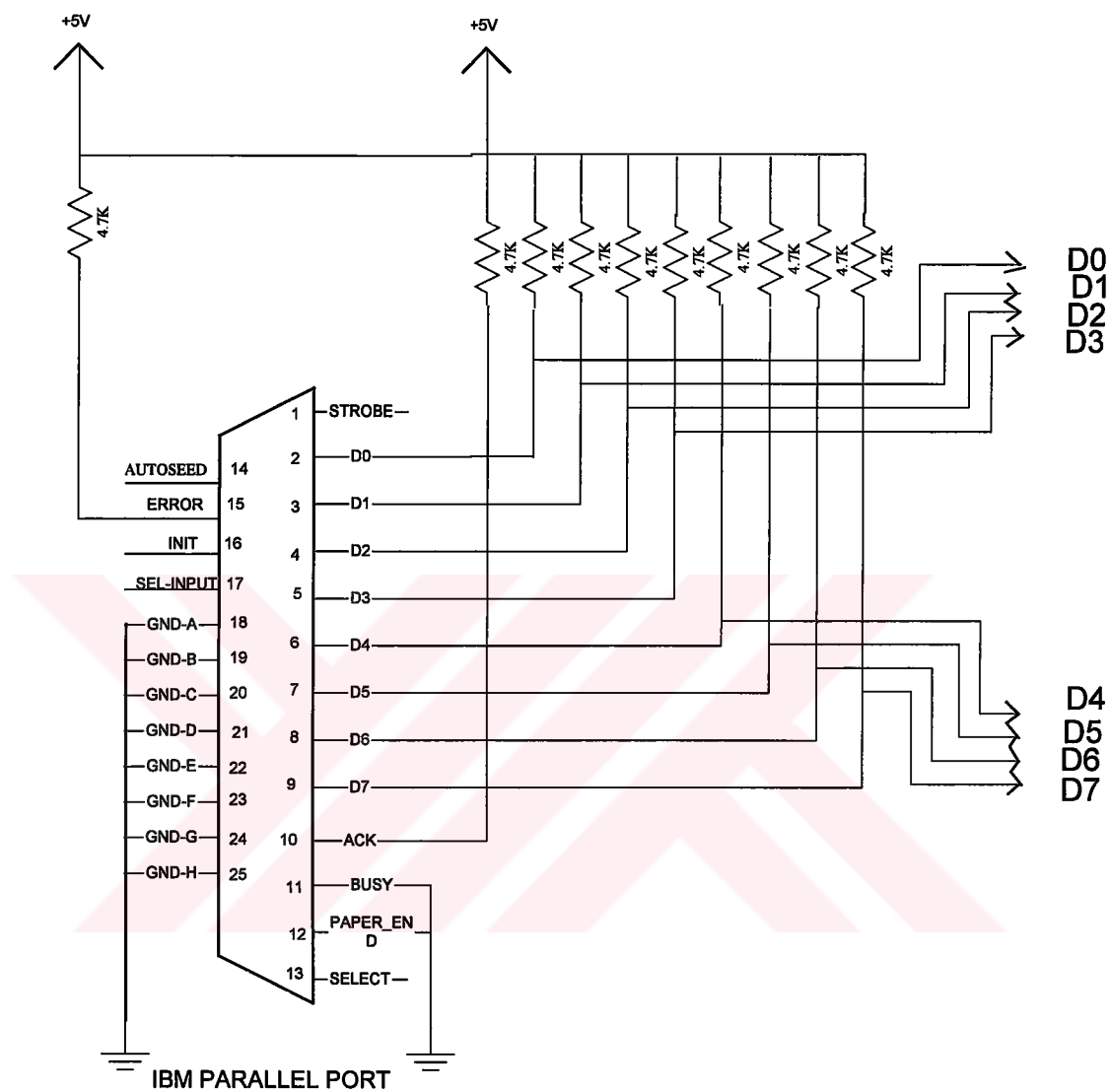
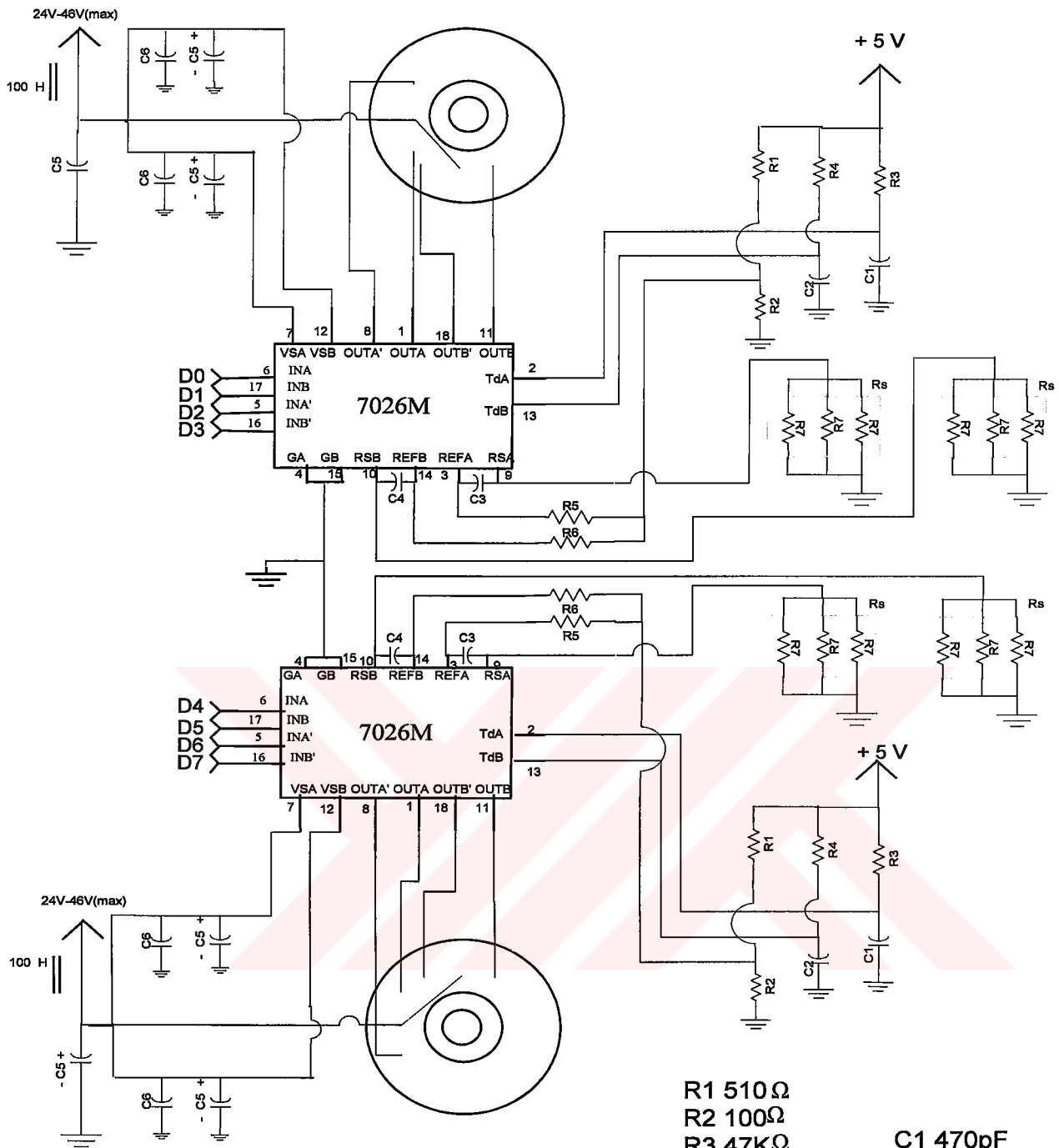


Figure 2.10. The motor driver circuit diagram of Antenna Control Unit System



R1 510 Ω
 R2 100 Ω
 R3 47K Ω
 R4 47K Ω
 R5 2.4K Ω
 R6 2.4K Ω
 R7 1.2 Ω
 Rs 0.4 Ω

C1 470pF
 C2 470pF
 C3 2200 pF
 C4 2200 pF
 C5 10 F
 C6 10 F

2.1.2.2. Power Supplies

For all chopper drivers, regulated power supplies are normally required. Because, the over-all system efficiency decreases due to the losses in the power supply. This increases transformer cost and heating problems. If unregulated supplies are used, the supply voltage affects holding and pullout torque. So, in this project, two regulated power supplies are selected and used.

The power requirements of the designed system are +5 V DC. as reference voltage and +34,4 V DC. as control supply voltage for stepper motors and motor driver circuits. The circuit diagram of +5 V DC. power supply is given in appendix A. The control supply voltage should not exceed +46 V DC. maximum (It is the maximum rating of the IC 7026M). The value of the control supply voltage only affects the pullout torque and the speed of rotation. At the power supply unit, two power supplies have been built in one box. 0 – 17 V DC. adjustable and regulated supply (max. 1A. output current) has been used for +5 V DC. reference voltage. It has been adjusted the output voltage to +5 V DC. A 120 VA transformer is connected to this power supply unit. The output of 12 V AC. of the secondary part of the transformer is connected to the AC. inputs of the supply unit. So the maximum DC. output value equals to $12\sqrt{2}=16,92$ V DC.

For the control supply voltage, another supply of 34.4 V DC. output voltage has been used. The power supply has been tested due to different load resistors. The test results are presented in appendix. The configuration of power supply box is as follows in the Figure 2.11.

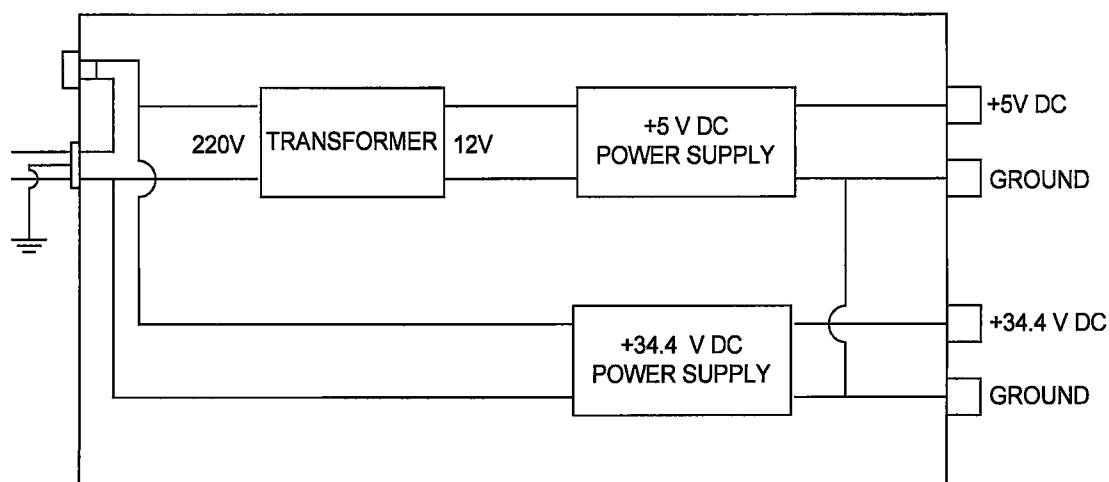


Figure 2.11. The Configuration of Power Supply Box

2.1.2.3. Interfacing to the computer

Serial and parallel communications are methods of transferring data from a computer to a remote unit such as Antenna Control Unit. In this case, the data consists of parameters such as azimuth or elevation angle. Both communication techniques are generally bi-directional allowing the computer to both transmit and receive information from a remote unit.

Serial communication transmits data one bit at a time on a single data line. Single data bits are grouped together into a byte and transmitted at a predetermined baud rate. Serial communication links can be as simple as 3-line connection; transmit (Tx), receive (Rx) and ground (G). This is an advantage from a cost viewpoint, but usually results in slower communications than parallel communications. Common serial interfaces include RS-232, RS-422, RS-485, and RS-423.

Parallel communication transmits data one byte (8 bits) at a time. The advantage of communicating in parallel versus serial is faster communications. However, since parallel communications more communication lines, the cost can be higher than serial communications. Parallel bus structures include IEEE-488, IBM PC, VME, and STD.

The original IBM-PC's Parallel Printer Port has a total of 12 digital outputs and 5 digital inputs accessed via 3 consecutive 8-bit ports in the processor's I/O space. The pinout configuration is shown in Figure 2.12 (Parker Automation).

- 8 output pins accessed via the DATA Port
- 5 input pins (one inverted) accessed via the STATUS Port
- 4 output pins (three inverted) accessed via the CONTROL Port
- The remaining 8 pins are grounded

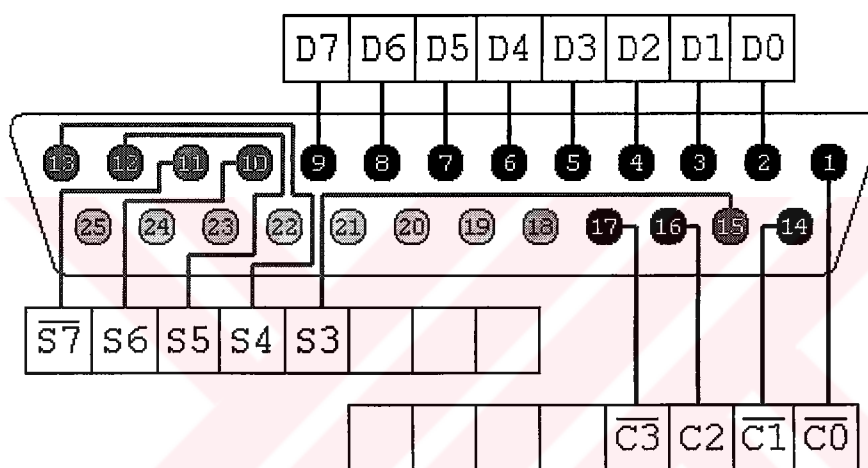


Figure2.12. 25-way Female D-Type Connector

In this designed system, data is transferred from computer to motor driver circuit via parallel port. The first four bit have been used for transferring data to Azimuth motor and the next four bit have been used for transferring data to Elevation motor. (Figure2.13.)

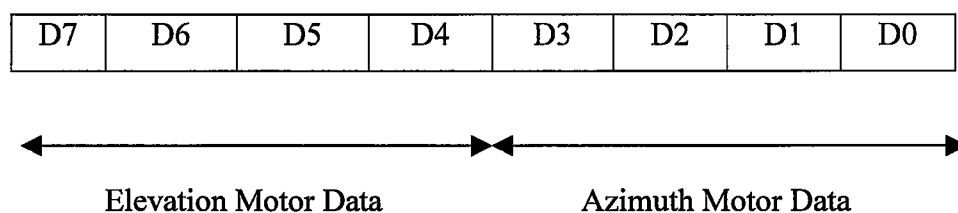


Figure 2.13. The data configuration of parallel port

In the system, at first Azimuth motor is rotated, after that elevation motor is rotated. This means that two data is not sent to the parallel port at the same time. Therefore, while azimuth data is sent to port, elevation data is equal to "0" and "0" is sent to port. In the same way, while elevation data is sent to port, azimuth data is "0" and "0" is sent to port.

2.1.3.Mechanical Design

Mechanical design of the parts which form the antenna control unit is very important. Better performance can only be achieved with a good mechanical design. To work with electromechanical parts (i.e. stepper motors) in a harmony, the design of mechanical parts and connections should be done carefully. Mechanical design of the system consists of iron and aluminium plates which are used to fix stepper motors and antenna to the system, bearings, gear drive, some screws and antenna.

Geosynchronous satellites share a limited space within the arc of geostationary orbit. An antenna on the earth must be directed to satellite exactly. Because, 0.01° deviation in direction of antenna causes a decrease in signal strength and broadcast quality. In other words, the pointing accuracy or sensitivity is an important criterion for an antenna. The characteristics of the antennae which are existing in Türksat Ground Control Station are given in Appendix A.

In order to provide sensitive system, gear drive and hybrid stepper motors which have small step angles are used. In the system designed, gear drive, which is originally a scrap, provides to reduce the step angle of azimuth motor and to amplify the output torque. The usage of gear system provides that the step angle of azimuth motor is divided by 100. In the system, when the azimuth motor is rotated one step (1.8°), the output of gear system and antenna rotates 0.018° . The sensitivity in azimuth direction can be calculated.

In order to rotate the antenna 90° , the rotor of motor must rotate $90 / 1.8 = 50$ steps. and gear system must move $90 / 0.018 = 5000$ steps.

$$\text{Sensitivity} = 50 / 5000 = 0.01^\circ$$

This sensitivity value is excellent result, because, pointing accuracy values of 11m. diameter Full Motion Antenna and 9m. diameter Limited Motion Antenna which are existing in Türksat Ground Control Station are 0.01° .

In the system designed, the output of elevation motor provides the rotation of antenna directly. In order to increase sensitivity in elevation direction, elevation motor is excited in “Half-step drive” mode. So, the step angle of elevation motor is 0.9° . In this case, the sensitivity is 0.9° and this value is greater than the characteristics of antennae in Türksat Ground Control Station.

2.2.SOFTWARE DESIGN

In the design and construction of antenna control unit system, system software has a great importance on the overall system performance. The software has also brought flexibility to the system functions and performance. The system software has been prepared mainly for controlling the motion of two stepper motors in elevation and azimuth directions, making the necessary calculations and visual design. The programming language Borland C ++5.0 for Windows 95 has been used for this system software.

In the system software, at first, the desired satellite or one of Türksat satellites location is entered to the computer. The desired location in Turkey Map is selected by the help of mouse. After that, The Calculate button in Functions Menu is pushed to calculate azimuth and elevation angles which are the destination points of antenna. When the Rotate button is pushed, the system software calculates the number of steps for two stepper motors and using parallel port sends the required electrical pulses to the drivers of motors. According to these pulses, motors make the proper rotations in azimuth and elevation directions, respectively.

No feedback is normally required in this system. This feature is one of the advantages of stepper motors. Today, stepper motors find their applications in speed

and position controls without expensive feedback loops. This driving method is referred to as the open-loop drive. A general block diagram is shown in Figure 2.14. (Takashi,1984)

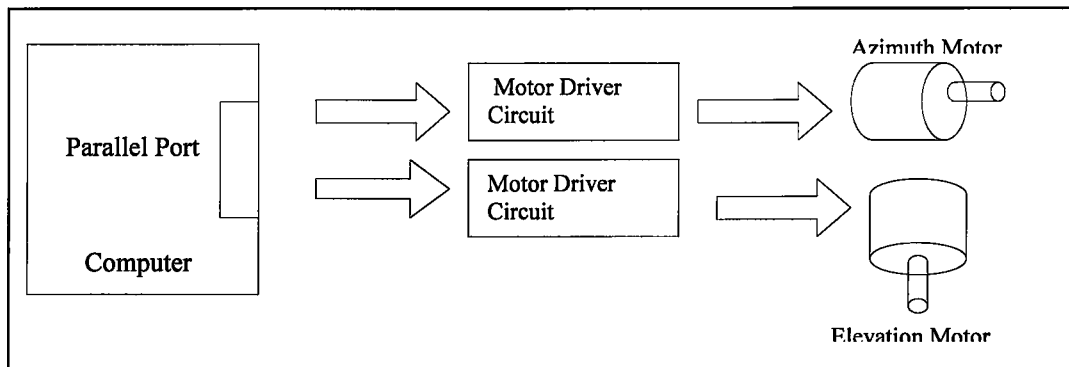


Figure 2.14. A General Block Diagram of the System

In this chapter, system software and its main functions will be explained. All of the system software is given in Appendix A. System software design consists of one main and four functional sections as shown in the Figure 2.15.

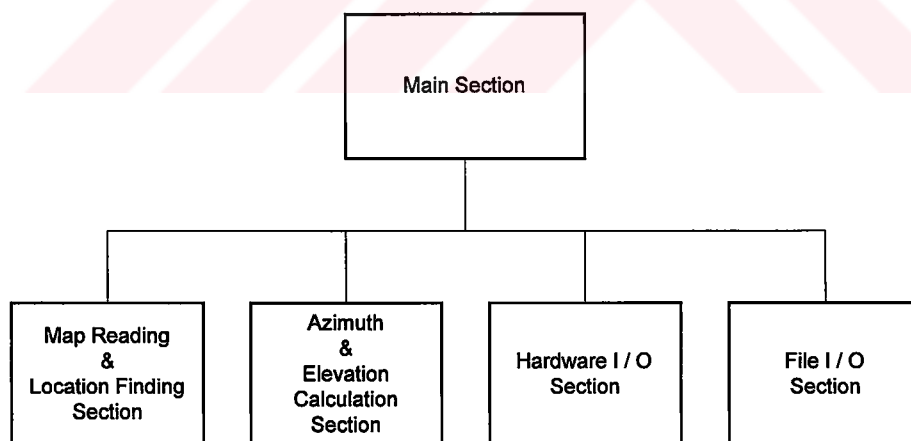


Figure 2.15. The Functional Diagram of the System Software

2.2.1. MAIN SECTION

Main section of the system software includes four program files. These are:

- Antenna .cpp file
- Antenna .rc file
- Antenna .h file
- Dtostring .cpp file

Main section of the system software controls the functional sections. In the following chapters, the program files of main section will be explained.

2.2.1.1. ANTENNA . CPP FILE

This file is the main and most important file of main section of the system software and has the following responsibilities:

- Verifying presence of Windows 95 operating system. On the contrary, system software doesn't work.
- Loading all the resources (bitmaps, dialog boxes, icons and buttons) which are necessary for the visual design of the system software.
- Deleting the resources before exiting the software and freeing system memory.
- Controlling the message loops for the system specific procedures with WinMain and OpenDlgProc functions.
- Calling the necessary functional sections, which are shown in the Figure 2.15.

2.2.1.2. ANTENNA .RC FILE

A resource is binary data that is linked to an application executable (.EXE). Usually a resource defines one or more of the following user interface components:

- A dialog box, which is a pop-up window that uses labels, text boxes, buttons, check boxes, scroll bars and other controls to give information to and receive information from a user.

- A menu, which is a list of commands from which a user can choose.
- A bitmap, which is a graphic image, such as picture, logo, or other drawing.
- A cursor, which is a graphic image that shows the position of the mouse on the screen and indicates the types of actions a user can perform.
- An icon, which is a graphic image that represents a minimized window.

As shown in the Figure 3.4, the visual design of the system software depends on the above-explained resources. These resources have been created using Resource Workshop in Borland C ++ Compiler Packet.

ANTENNA.RC file is the resource file of our software and supports the visual design of the system software. At the design time, the visual design of the system software has been seen in a text manner with a text editor opening the ANTENNA.RC file. The printout of this file is added the software in Appendix A. The text in this file has a resource language specific type and notation. The extension of this file is “.RC” . When this file is compiled, it becomes “.res” file that is linked into our application.

2.2.1.3. ANTENNA .H FILE

Necessary variables and their types (extern, int, double, char, bool, struct), necessary definitions (ID numbers of all objects of the system software which appear on the monitor.), necessary included header files and prototypes of functions are defined in this header file.

2.2.1.4. DTOSTRING.CPP FILE

All calculations in the system software are done with decimal numbers. In order to show these numbers on the screen, DTOSTRING.CPP file converts these numbers to characters.

2.2.2. FUNCTIONAL SECTIONS

2.2.2.1. FILE I / O

ANTENF.CPP has been written for file input / output processing. This file realizes all operations of ANTENNA.TXT file, which saves the last coordinates (azimuth and elevation angles) of the system and the last azimuth and elevation data, which sent to the motor windings.

When the system software is run first time, it searches ANTENNA.TXT file, if it hasn't been created, it opens a new ANTENNA.TXT file. On the contrary of this situation it reads the initial values of angles and data from precreated ANTENNA.TXT file via ANTENF.CPP FILE. The initial values of the content of ANTENNA.TXT file are as follows:

Azimuth Angle = 0°

Elevation Angle = 0°

Azimuth Data = 1

Elevation Data = 1

After every rotation of the motors, ANTENF.CPP file takes the new azimuth, elevation angles and data from ANTENNA.CPP file and writes these values to ANTENNA.TXT file.

2.2.2.2. HARDWARE I / O

ROTATES. CPP has been written for communicating with parallel port of the computer. This file provides to make proper rotation of the motors according to the number of steps and datas which calculated in ANTENNA.CPP file. ROTATES.CPP file sends the necessary pulses to parallel output port. This file consists of 6 different sections.

- Sagadon function

This section has been written for azimuth positioning in CW direction. The binary data is sent to parallel port according to the Figure 2.16.

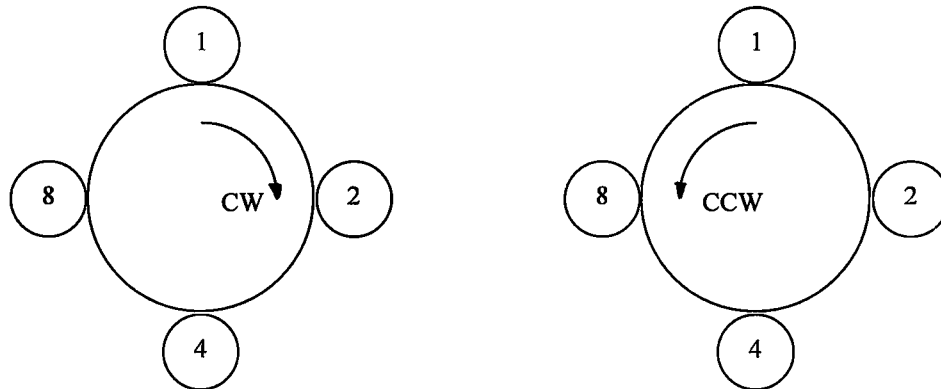


Figure 2.16. The Data Configurations of Azimuth Positioning

- Soladon Function

This function provides azimuth positioning in CCW (Counter Clock Wise) direction. The binary data is sent to parallel port according to the Figure 2.16.

- Sagadon1 Function

This function provides elevation positioning in CW direction. The binary data is sent to according to the following Figure 2.17.

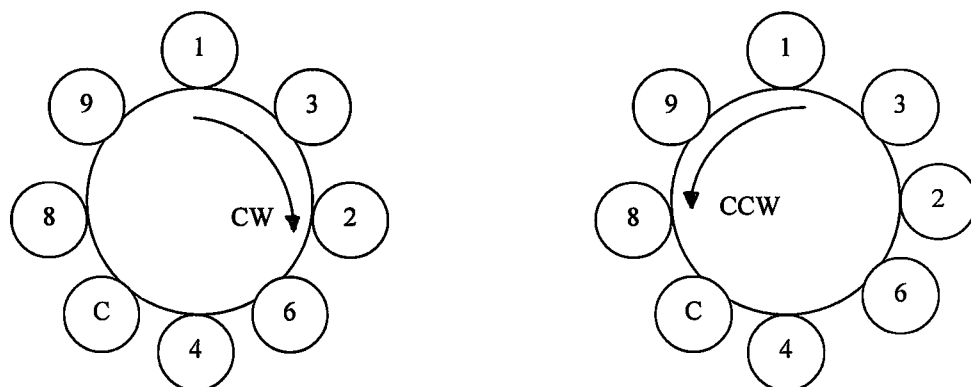


Figure 2.17. The Data Configurations of Elevation Positioning

- Soladon1 function

This function provides elevation positioning in CCW direction. The binary data is sent to parallel port according to the Figure 2.17.

- Gecikme Function

This function provides the necessary delay time between two azimuth data pulses. The delay time is 10 milliseconds between azimuth data pulses.

- Gecikme 1 Function

This function provides the necessary delay time between two elevation data pulses. The delay time is 100 milliseconds between elevation data pulses.

2.2.2.3. Map Reading & Location Finding

MAPS.CPP is written for calculating the coordinates (latitude and longitude values) of the selected point on the map by the mouse pointer. MAPS.CPP provides that the mouse works only in Turkey Map area for calculating the desired coordinates properly.

When the user click the mouse on one point in the map, ANTENNA.CPP file calls MAPS.CPP file. MAPS.CPP file calculates the latitude and longitude values of that point and sends these coordinates to ANTENNA.CPP file. After that, ANTENNA.CPP file provides to show these coordinates on the screen.

2.2.2.4. Azimuth & Elevation Calculation

AZIMELEV.CPP file is written for calculating the azimuth and elevation angles of any selected point on the map by the help of mouse pointer.

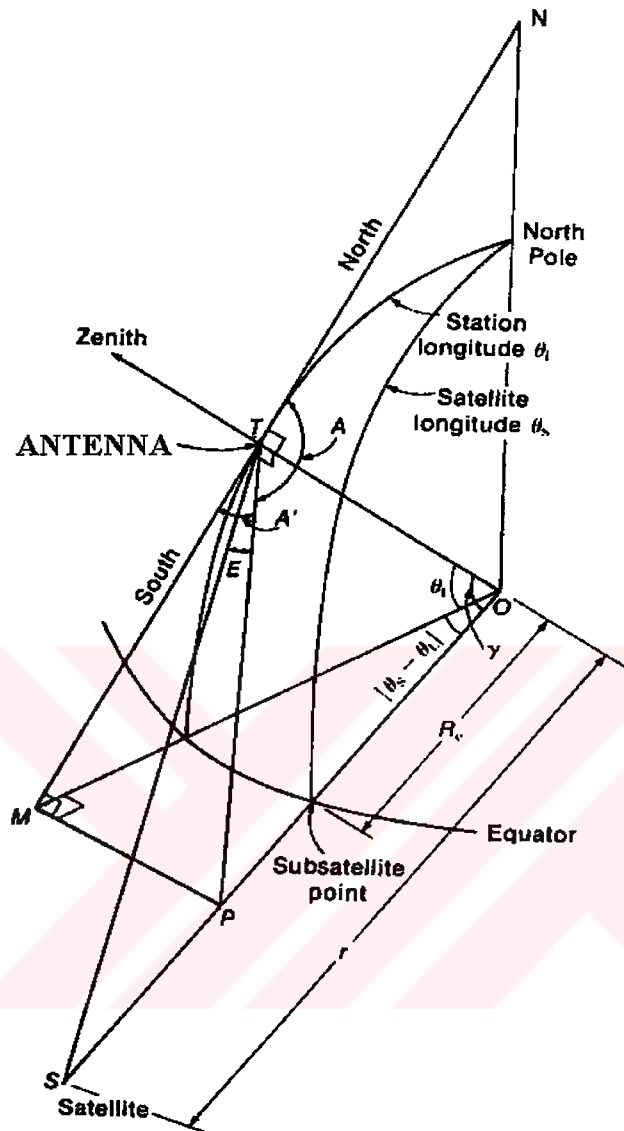


Figure 2.18 Azimuth and elevation

The azimuth angle A and the elevation angle E can be calculated using knowledge of antenna latitude θ_1 and longitude θ_L and the satellite longitude θ_s as shown in the Figure 2.18. The azimuth angle is defined as the angle measured clockwise from the true north to the intersection of the local horizontal plane TMP and the plane TSO (passing through the antenna, the satellite and the earth's center). The azimuth angle A is 0 and 360°. Depending on the location of antenna with respect to the subsatellite point (See Figure 2.19.), the azimuth angle A is given by :

1- Northern Hemisphere

Antenna west of satellite : $A = 180^\circ - A'$

Antenna east of satellite : $A = 180^\circ + A'$

2- Southern Hemisphere

Antenna west of satellite : $A = A'$

Antenna east of satellite : $A = 360^\circ - A'$

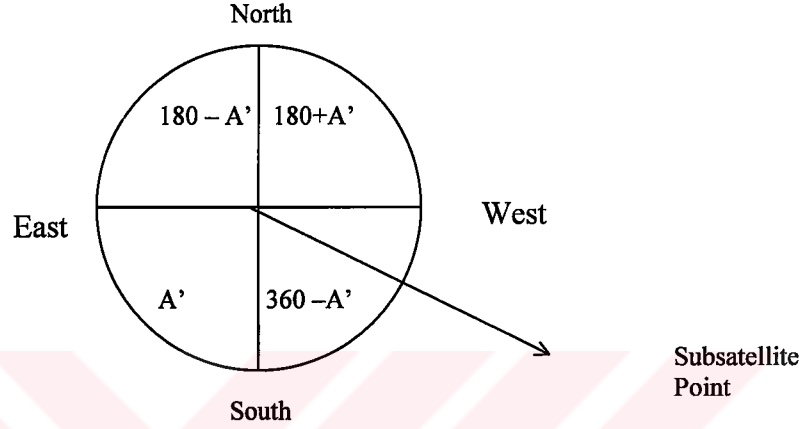


Figure 2.19. The Azimuth angles with respect to subsatellite point

A' is the positive angle defined in Figure 2.18. In the system software, azimuth angle has been calculated according to $A=180+A'$ equation. The elevation angle E is defined as the angle produced by the intersection of the local horizontal plane TMP and the plane TSO with the line of sight between the antenna and the satellite. In the following equations, we assume that the earth is a perfect sphere with radius R_e . From Figure 2.18, we have

$$A' = \tan^{-1} \left(\frac{MP}{MT} \right)$$

$$A' = \tan^{-1} \left(\frac{MO \tan |\theta_s - \theta_L|}{R_e \tan \theta_l} \right)$$

$$A' = \tan^{-1} \left[\frac{\left(\frac{R_e}{\cos \theta_l} \right) \tan |\theta_s - \theta_L|}{R_e \tan \theta_l} \right]$$

$$A' = \tan^{-1} \left(\frac{\tan |\theta_s - \theta_L|}{\sin \theta_l} \right)$$

To calculate the elevation angle E , let us consider the triangle TSO shown in Figure 2.18. and redrawn in Figure 2.20.

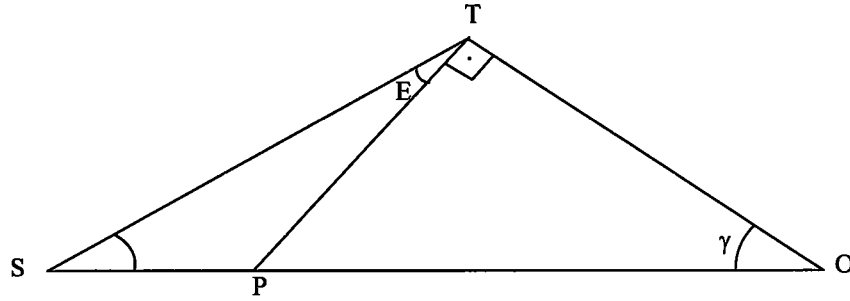


Figure 2.20. Triangle TSO

The same triangle can be drawn as the following Figure 2.21.

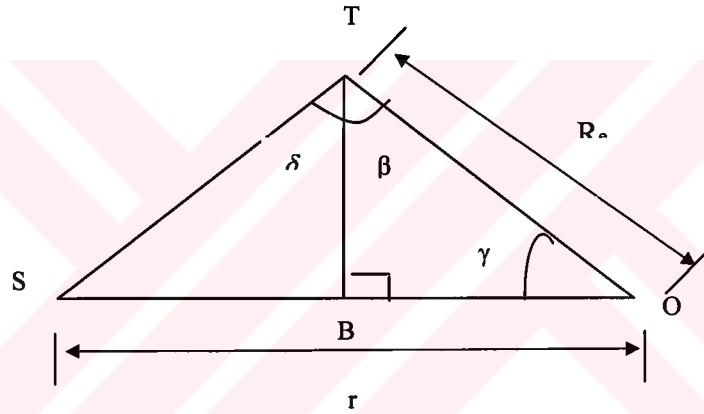


Figure 2.21. Triangle to calculate elevation

According to the Figure 2.21,

$$E = \beta + \delta - 90^\circ$$

$$E = (90^\circ - \gamma) + \delta - 90^\circ$$

$$E = \delta - \gamma$$

The angle γ can be evaluated from the triangle TPO as follows:

$$\gamma = \cos^{-1}\left(\frac{R_e}{OP}\right)$$

Since,

$$OP = \frac{MO}{\cos|\theta_S - \theta_L|} = \frac{R_e}{\cos\theta_l \cos|\theta_S - \theta_L|}$$

as seen from the triangles MPO and TMO, we have

$$\gamma = \cos^{-1}(\cos \theta_I \cos |\theta_S - \theta_L|)$$

To evaluate the angle β in Figure 2.21., we note that

$$\beta = \tan^{-1}\left(\frac{SB}{TB}\right)$$

$$\beta = \tan^{-1}\left(\frac{r - R_e \cos \gamma}{R_e \sin \gamma}\right)$$

$$\beta = \tan^{-1}\left(\frac{r - R_e \cos \theta_I \cos |\theta_S - \theta_L|}{R_e \sin[\cos^{-1}(\cos \theta_I \cos |\theta_S - \theta_L|)]}\right)$$

Thus the elevation angle E can be expressed by

$$E = \tan^{-1}\left(\frac{r - R_e \cos \theta_I \cos |\theta_S - \theta_L|}{R_e \sin[\cos^{-1}(\cos \theta_I \cos |\theta_S - \theta_L|)]}\right) - \cos^{-1}(\cos \theta_I \cos |\theta_S - \theta_L|)$$

ANTENNA.CPP file sends the latitude and longitudes value of the selected point on the map to AZIMELEV.CPP file. After that, AZIMELEV.CPP file calculates the new azimuth and elevation angles using the above equations, which are shown, in box and sends these new angles to ANTENNA.CPP file.

ANTENNA.CPP file reads the old azimuth and elevation angles which show the last coordinates of the system from ANTENNA.TXT file and then compares the new elevation and azimuth angles with the old values of them.

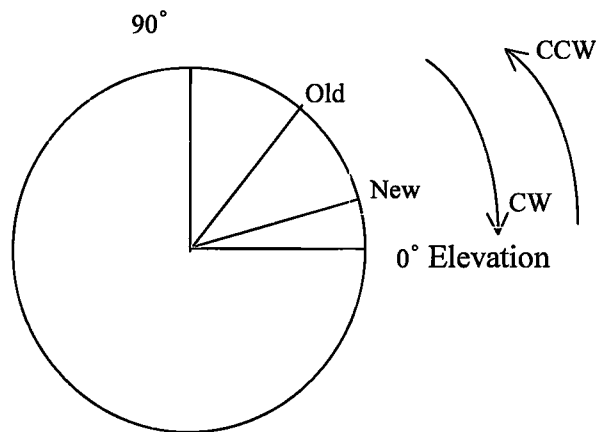


Figure 2.22 Comparison of elevation angles

ANTENNA.CPP file compares elevation angles at first.(See Figure 2.22.) If old elevation angle is greater than the new one, ANTENNA.CPP file calculates the number of steps in elevation angle, which the rotor of motor will rotate according to the following equation:

$$N = \frac{oldele - newele}{0.9}$$

N = Number of Steps

0.9 = Step angle of Elevation motor

oldele = old elevation angle

newele=new elevation angle

After calculation of number of steps, ANTENNA.CPP file calls Sagadon1 function, which provides elevation positioning in CW direction. If old elevation angle is smaller than the new one, ANTENNA.CPP file calculates the number of steps according to the following equation:

$$N = \frac{newele - oldele}{0.9}$$

and calls soladon1 function, which provides elevation positioning in CCW direction.

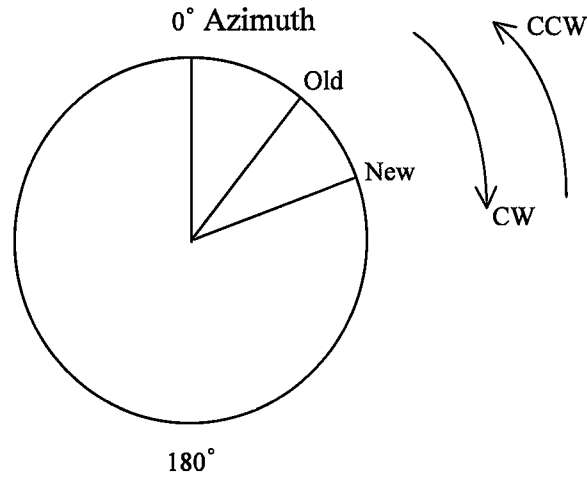


Figure 2.23 Comparison of Azimuth Angles

After these all processing, ANTENNA.CPP file compares the new azimuth angle with the old one. (See Figure 2.23) If the new azimuth angle is greater than the old one, ANTENNA.CPP file calculates the number of steps according to the following equation:

$$N = \frac{newazi - oldazi}{0.018}$$

newazi = new azimuth angle

oldazi = old azimuth angle

0.018 = step angle of output of the gear drive which is connected to rotor of azimuth motor.

After this calculation, ANTENNA.CPP calls sagadon function, which provides azimuth positioning in CW direction. If the new azimuth angle is smaller than the old one, the number of steps are calculated by the following equation:

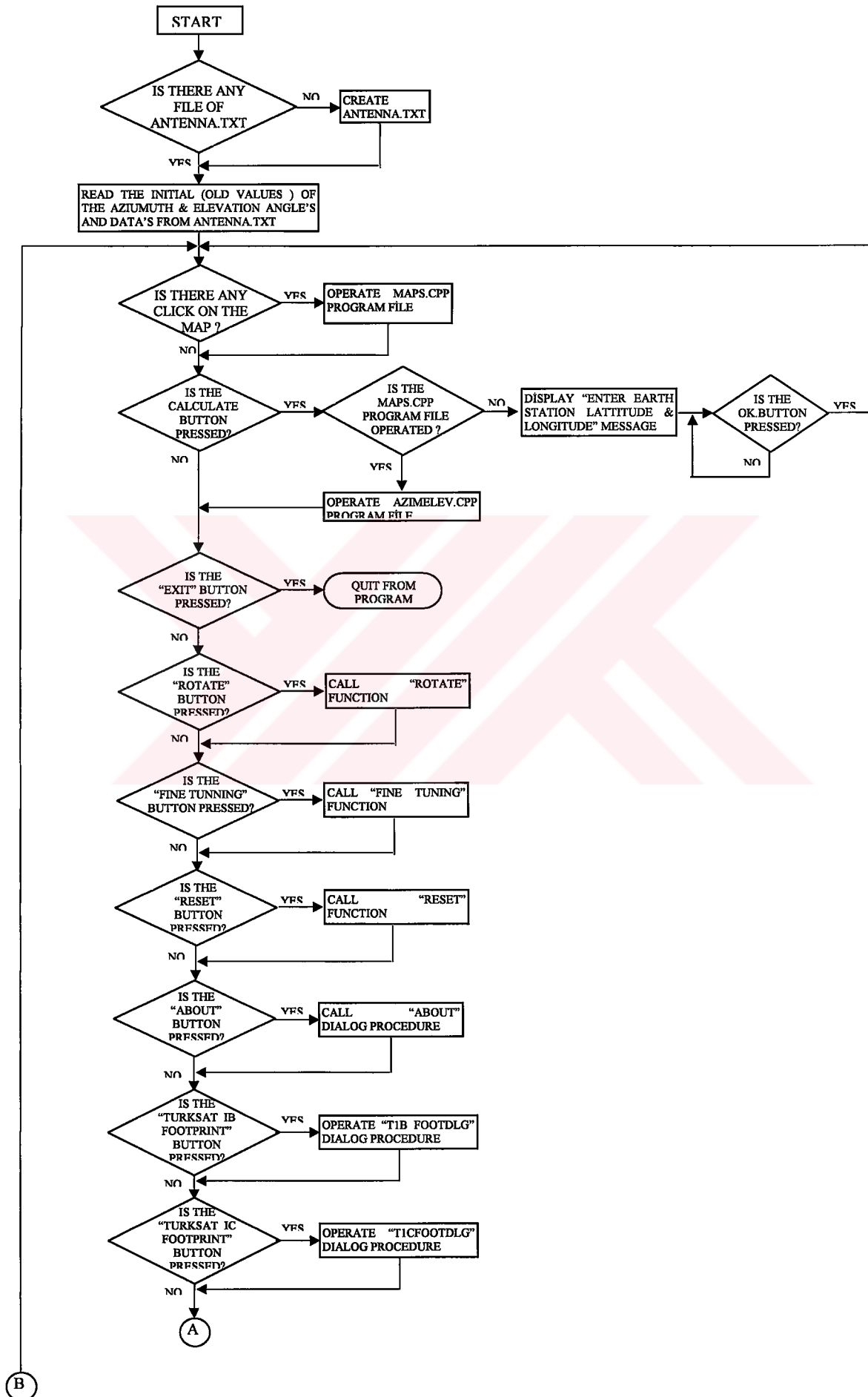
$$N = \frac{oldazi - newazi}{0.018}$$

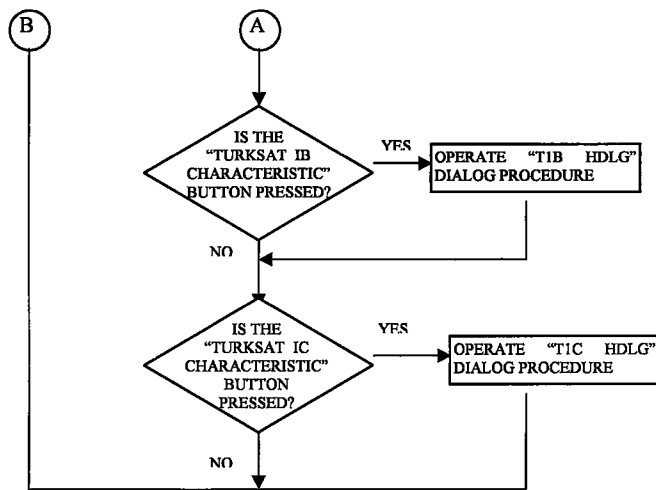
According to result of this calculation, ANTENNA.CPP file calls soladon function which provides azimuth positioning in CCW direction.

2.3. THE FLOWCHARTS OF SOFTWARE PROGRAMME

The flowcharts of main program, fine tuning, reset and rotate functions have been prepared during software design. These flowcharts are given in the following pages.

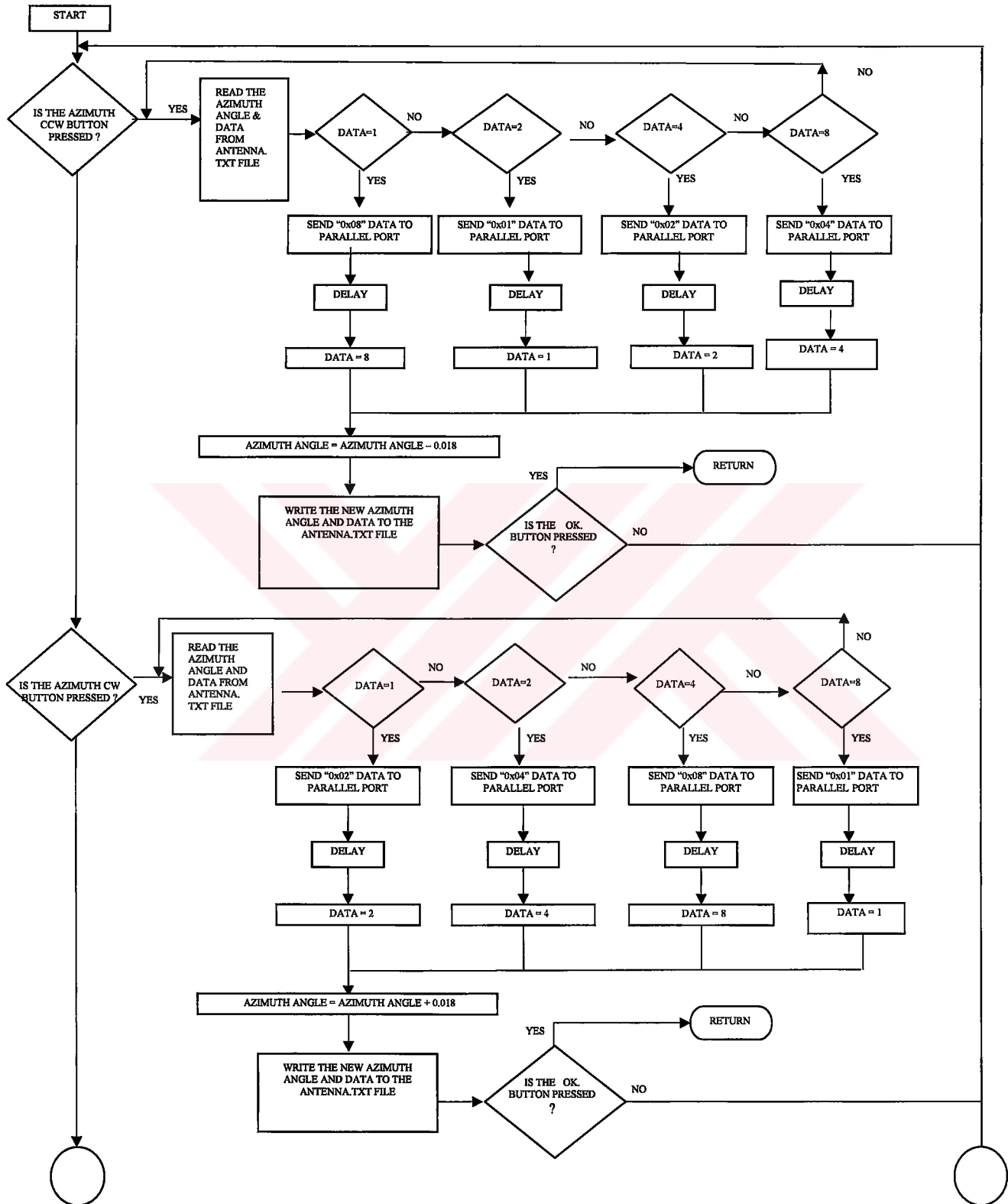


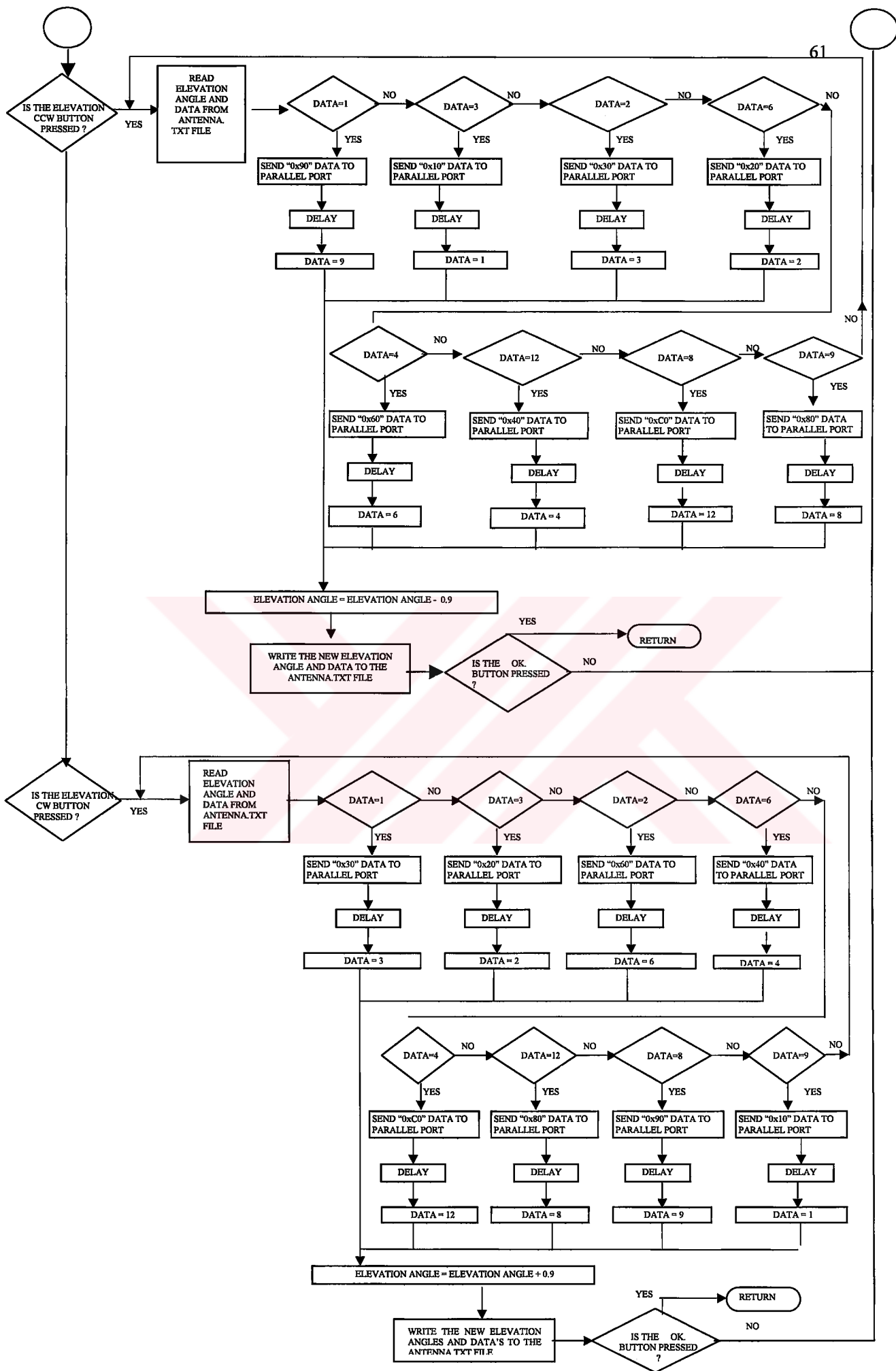




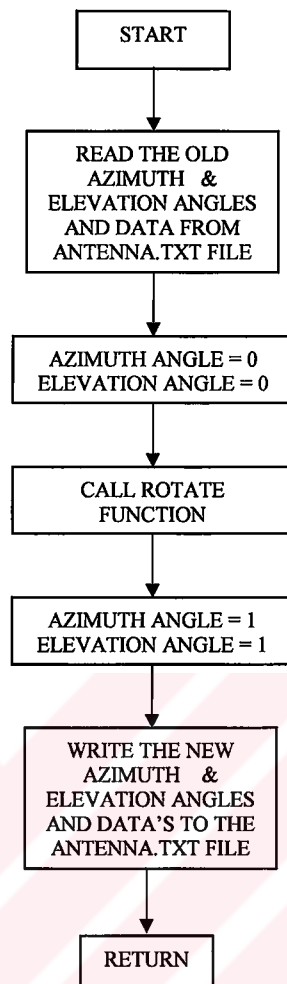
THE FLOWCHART OF MAIN PROGRAM

THE FLOWCHART OF FINE TUNING FUNCTION

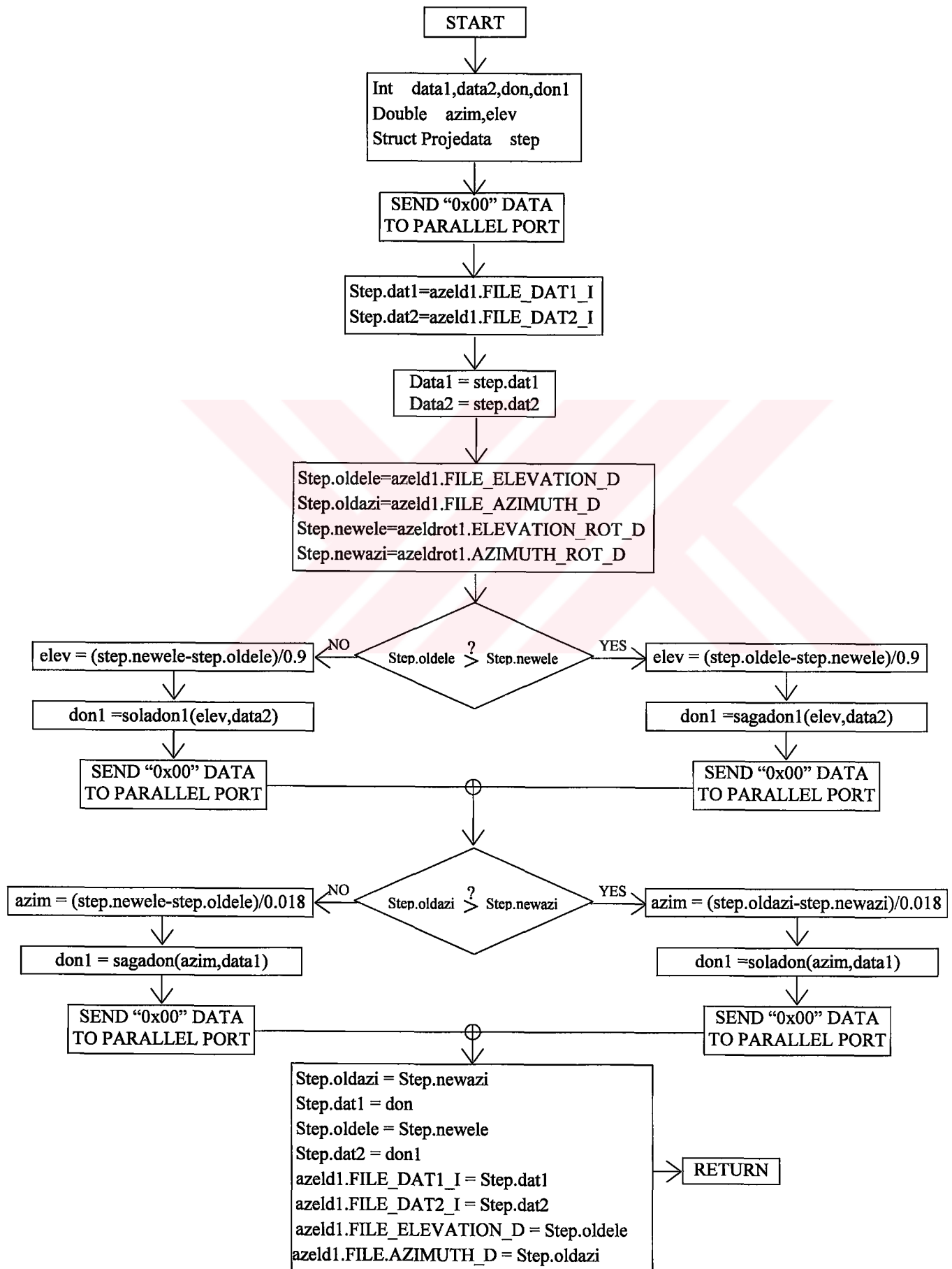




THE FLOWCHART OF RESET FUNCTION



THE FLOWCHART OF ROTATE FUNCTION



CHAPTER THREE

MOUNTING AND OPERATING THE ANTENNA CONTROL UNIT

3.1. MOUNTING

The purpose of mounting is to aim antenna of the system towards a desired satellite accurately and securely. Using a mount that provides stability and pointing accuracy is critical in designing and installing a reception system, especially for Ku-band antennae that have narrow beamwidths and thus target a very small portion of the sky. Because, 0.01° deviation in direction of antenna causes the centre of antenna vision to scan 75km. at the geosynchronous arc leading to a decrease in signal strength and broadcast quality. (See Table 3.1.)

Table 3.1. Satellite Spacing

Angular Spacing	Separation between Satellites
0.5	369 km.
1.0	739 km.
2.0	1477 km.
3.0	2216 km.
4.0	2955 km.

There are three principal classes of mounts: X-Y mount, AZ / EL mount and polar mount. In this system, AZ / EL mount is used on every installation.

In AZ / EL mount, the location of a point on earth can be described by using the azimuth over elevation coordination system. Azimuth is defined as an angle produced by rotation about an axis, which is perpendicular to the local horizontal plane. Elevation axis rotate in the local horizontal plane as azimuth angle rotates. A

change in the elevation angle will cause a rotation of antenna in the vertical plane (Intelsat, 1995)

Installation of AZ / EL mount is simple and not very critical. Any communication satellite is pointed by first moving the mount to the correct azimuth angle, which is along a plane parallel to the surface of the earth, and then by rotating up to the required elevation angle. (See Figures 3.1. and 3.2.) There is no tracking error so each satellite can be perfectly targeted. This is an advantage of AZ / El mount (Baylin, 1997).

AZ / EL mount is used in the construction of this system. On every installation, the user must be sure about azimuth and elevation angles of the antenna are adjusted correctly according to reference angles which are mentioned at Figure 3.2.

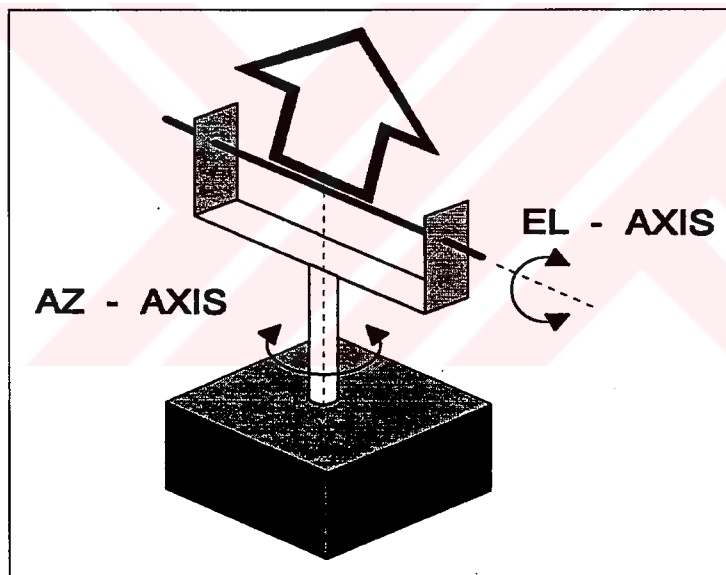


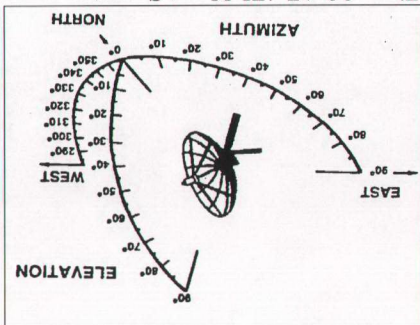
Figure 3.1 AZ /EL Mount



Figure 3.3 The Photograph of Antenna Control Unit

3.2. OPERATING THE SYSTEM

Figure 3.2. AZ / EL Mount Geometry



As it is seen from Figure 3.3. (Photograph), the elements of the system are as follows:

+34.4 V DC Power Supply

+5 V DC Power Supply

Two Stepper Motors

Motor Driver Circuit

Computer (including Windows 95 operating system)

Parallel Port Cable

Stepper motor windings have the terminals named with brown, red, orange, and yellow. These windings are connected to the driver circuit through these terminals. The windings named white/blue common and black/green common are connected 34.4 V DC power supply. + 5V and +34.4VDC supplies are used to energize the driver circuit and the motor windings, respectively. Computer is necessary for operating the system software and controlling the motion of stepper motors. In order to run the system software, computer must include Windows 95 operating system. The stepper motor driver IC has pulse input terminals. These are connected to the corresponding bits of the computer's parallel output port. For this purpose, parallel port cable that has 2.5.m. length.

3.3. HOW TO USE THE SYSTEM SOFTWARE

In this section the usage of the system software is explained. The procedure that should be followed after the connections of the system elements is given below. The visual design is also shown in Figure 3.4.

Step 1. Click one point by the left button of mouse on the map.

“Antenna Map Location” display will indicate the latitude and longitude coordinates of that point on the map.

Step 2. Select one of Türksat satellites on “Satellite Selection” menu or enter the location of the desired satellite on “Satellite Location” menu.

Step 3. Press “Calculate” button on “Functions” Menu.

Calculated values display will indicate the calculated azimuth and elevation values.

Step 4. Press “Rotate” button on “Functions” Menu

Step 4.1. When the message about azimuth angle is appeared, press “OK” button.

Azimuth motor starts to rotate after this command.

Step 4.2. When the message about elevation angle is appeared, press “OK” button.

Elevation motor starts to rotate after this command.

Step 5. Press “Fine Tuning” button on “Functions” menu.

The message menu “Antenna Manual Adjustment” appears after this command.

Step 5.1. Select the direction (CW or CCW) of azimuth motor on this menu

Step 5.2. Press “OK” button.

After this command, azimuth motor rotates one step (0.018°) in the selected direction.

Step 5.3. Select the direction (CW or CCW) of elevation motor on” Antenna Manual Adjustment” menu.

Step 5.4. Press “OK” button.

After this command, elevation motor rotates one step (0.9°) in the selected direction.

Step 6. Press “Reset” button before exit the system software.

Step 6.1. When the message about azimuth angle is appeared, press “OK” button.

Azimuth motor starts to rotate towards the reference direction.

Step 6.2. When the message about elevation angle is appeared, press “OK” button.

Elevation motor starts to rotate towards the reference direction.

Step 7. Select one of Türksat satellites on “Foot Prints” menu, in order to see footprints of the selected satellite.

Step 8. Select one of Türksat satellites on “Characteristics” menu, in order to see characteristics and performances of the selected satellite.

Step9. Press “Exit” button for terminating the system software.

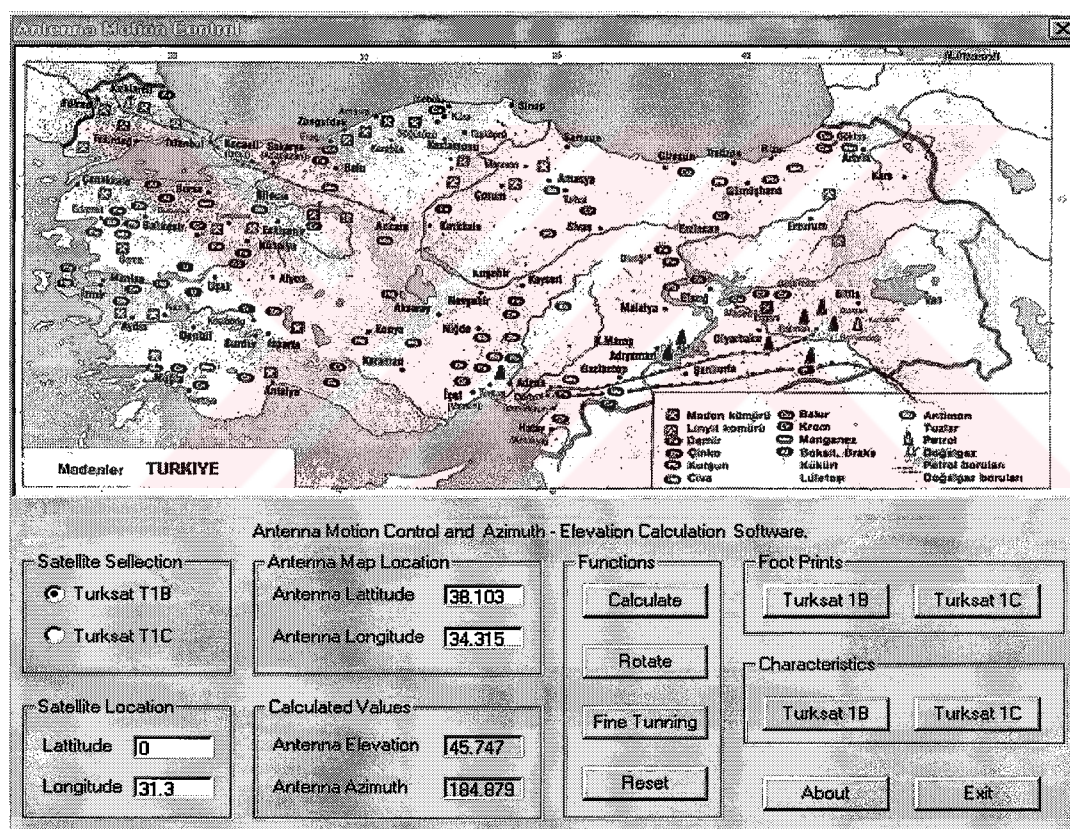


Figure 3.4. The Visual Design of the System Software

In order to maintain and preserve the antenna's reference azimuth and elevation angles which are adjusted at the first installation, for the later operation and proper usage of the system, the user must reset the system before every exit from software.

CHAPTER FOUR

CONCLUSIONS

In this thesis, Computer Aided Satellite Antenna Control Unit System has been designed and implemented using stepper motors. The system is aimed to be capable of controlling the antenna in two axis (azimuth and elevation) using stepper motors and pointing the antenna to the desired satellite.

To achieve this aim, stepper motors and their types, stepper motor drive and excitation methods have been investigated. The system includes hardware and software designs. In the hardware designed, the right hybrid stepper motor and motor driver ICs have been selected according to the load conditions of the system. Stepper motor driver circuit and power supplies have been designed. Mechanical design is important for getting the best performance of the system. In mechanical design, gear drive is used for reducing the step angle of azimuth motor and amplifying the output torque. In order to interface to the computer, parallel port is selected between the computer and stepper motor driver circuit.

The system software has been written to provide the following main functions:

- Reading map and finding the locations of the selected point on the map,
- Calculating azimuth and elevation angles according to the locations of selected point,
- Rotating the stepper motors to calculated azimuth and elevation angles, resetting of the antenna, fine tuning .

A satellite in a geostationary orbit appears to be stationary with respect to a point on the earth. Therefore, if an antenna is within the coverage of the satellite, it can receive the broadcasts of the satellite by simply pointing toward satellite. If the user of antenna want to receive broadcasts from more than one satellite, the antenna must be rotated to point the desired satellite. Manual adjustment of this process is difficult and wastes time and effort. Therefore, the antenna control unit system has been

designed to control the position of antenna in this thesis. The designed system is used for antennas which have small diameter. Even if the stepper motors are sufficient to rotate the small diameter antennas, different technologies must be used to control the antennae which have 11m. or 30m. diameters on the Earth Stations. The system software is also a prototype of the softwares which control the mentioned big antennae.



REFERENCES

- Acarnley, P.P. (1984). Stepping Motor: A Guide to Modern Theory and Practice.
London; Peter Pregrinus Ltd.
- Allegro MicroSystems, Inc. High-Current Unipolar Stepper Motor Controller / Drivers.
- Baylin, Frank.(1997) Digital Satellite TV (5 th Ed.). Boulder, Colorado; Baylin Publications.
- Ericsson. Industrial Circuits Databook and Stepper Motor Control Handbook Application Notes.
- INTELSAT (1995) Eart Station Technology (4th Ed.)
- Jennings,S. (1996). Basics of Stepping Motors.
- Kollmorgen. Step Motors and Controls. Radford, VA..USA
- Kollmorgen , NP-7026M, 7024M, 2918M Unipolar & Bipolar Chopper Drive Chips Databook. Radford, VA.USA.
- Özdamar, F. (1976) . A Two Dimensional Digital Position Control System. METU.
- Özdeşlik, M.(1982). Programmable Step Motor Drive. METU
- Parker Automation. Compumotor Step Motor and Servo Motor Systems and Controls Catalog. USA.
- Takashi, Kenjo.(1984). Stepping motors and their microprocessor controls. Kanagawa, Japan. Clarandon Press, Oxford.
- Tri, T. Ha. (1990). Digital Satellite Communications (2nd Edition). McGraw – Hill Publishing Company

APPENDIX A



Stepper Motor Drive IC NP-7026M Specifications

1. General Specifications

-1 Product	Hybrid IC
-2 Construction	Transfer-mold
-3 Main application	Stepper motor drive
-4 Excitation	2 phase or 1-2 phase
-5 Applicable motor	4 phase bi-filar wound stepper motors (6- leads)

2. Max. Absolute Ratings (Ta = 25°C)

Item	Symbol	Standard
Supply Voltage	YCC	46 V max
FET Output Standing	YDC	100 V max
Control Supply Voltage	YS	46 V max
TTL Input Voltage	YIN	7 V max
Reference Voltage	YREF	2 V max
Output Current	IO	3 A max
Allowable Loss	PD	4.5 W max without radiating heat fan
Allowable Loss	PD	35 W max (Tc = 25°C)
Junction Temp.	TJ	150°C max
Storage Temp.	TSTG	-40 to +150°C

3. Electrical Characteristics

-1 DC Characteristics

Item	Symbol	Condition	Ratings			Unit
			Min.	Typ.	Max.	
Control Supply Current	IS	YS=44V		10	15	mA
Control Supply Voltage	YS		10	24	44	V
FET ON Voltage	YDS	ID=3A, YS=10		0.8		V
FET Drain-leak Current	IDSS	YDDS=100V, YS=44V			4	mA
TTL Input Current	IiH	YiH=2.4V, YS=44V			40	μA
TTL Input Current	IiL	YiL=0.4V, YS=44V			-0.8	mA
TTL Input Voltage(Active 'H')	YiH	ID=3A	2.0			V
TTL Input Voltage(Active 'H')	YiL	IDSS=100V			0.8	V
TTL Input Voltage(Active 'L')	YiH	IDSS=100V	2.0			V
TTL Input Voltage(Active 'L')	YiL	ID=3A			0.8	V
FET Diode Order Voltage	YSD	ISD=3A			2.3	V

-2 AC Characteristics

Item	Symbol	Condition	Ratings			Unit
			Min.	Typ.	Max.	
Switching Time	TR	YS=24V, ID=3A		0.5		μS
Switching Time	TSTG	YS=24V, ID=3A		0.7		μS
Switching Time	TF	YS=24V, ID=3A		0.1		μS

4. Notes

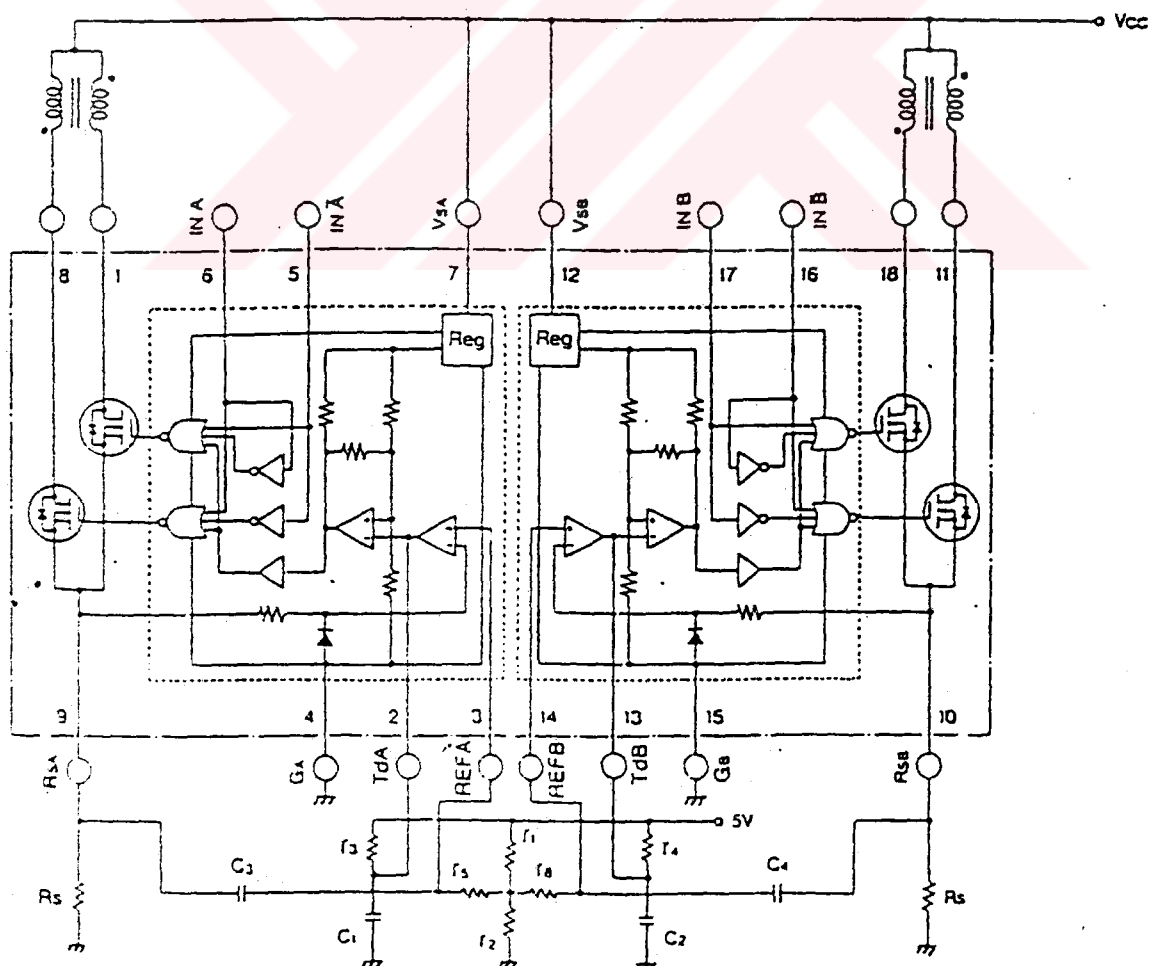
The excitation input signal of NP-7026M can be applied by either Active 'H' or Active 'L'. However, there is the difference on wiring hook-up.

Pin #	1	8	11	18
Input signal Active 'H'	OUT A	OUT \bar{A}	OUT B	OUT \bar{B}
Input signal Active 'L'	OUT \bar{A}	OUT A	OUT \bar{B}	OUT B

5. Circuit Diagram

Output Pin Assignment (Excitation Input Signal)

Pin #	1	8	11	18
Input signal Active "H"	OUT A	OUT \bar{A}	OUT B	OUT \bar{B}
Input signal Active "L"	OUT \bar{A}	OUT A	OUT \bar{B}	OUT B



6. Standard Circuit

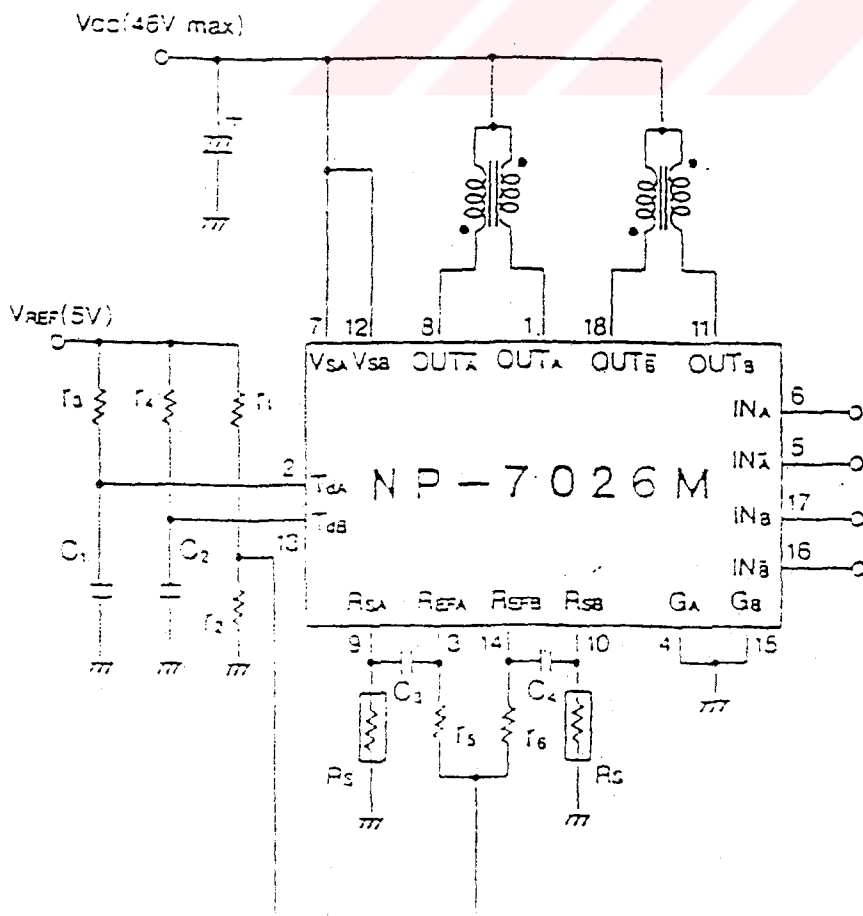
- 1 Excitation Timing Chart (Active 'H')

2 Phase excitation

Clock	0	1	2	3	0	1
INA	H	L	L	H	H	L
INA ⁻	L	H	H	L	L	H
INB	H	H	L	L	H	H
INB ⁻	L	L	H	H	L	L

1-2 Phase excitation

Clock	0	1	2	3	4	5	6	7	0	1	2	3
INA	H	H	L	L	L	L	L	H	H	H	L	L
INA ⁻	L	L	L	H	H	H	L	L	L	L	L	H
INB	L	H	H	H	L	L	L	L	L	H	H	H
INB ⁻	L	L	L	L	L	H	H	H	L	L	L	L



- R_1 510 Ω
- R_2 100 Ω (VR)
- R_3 47k Ω
- R_4 47k Ω
- R_5 2.4k Ω
- R_6 2.4k Ω
- C_1 470pF
- C_2 470pF
- C_3 2200pF
- C_4 2200pF
- R_s 0.68 Ω (typ) 1-2W

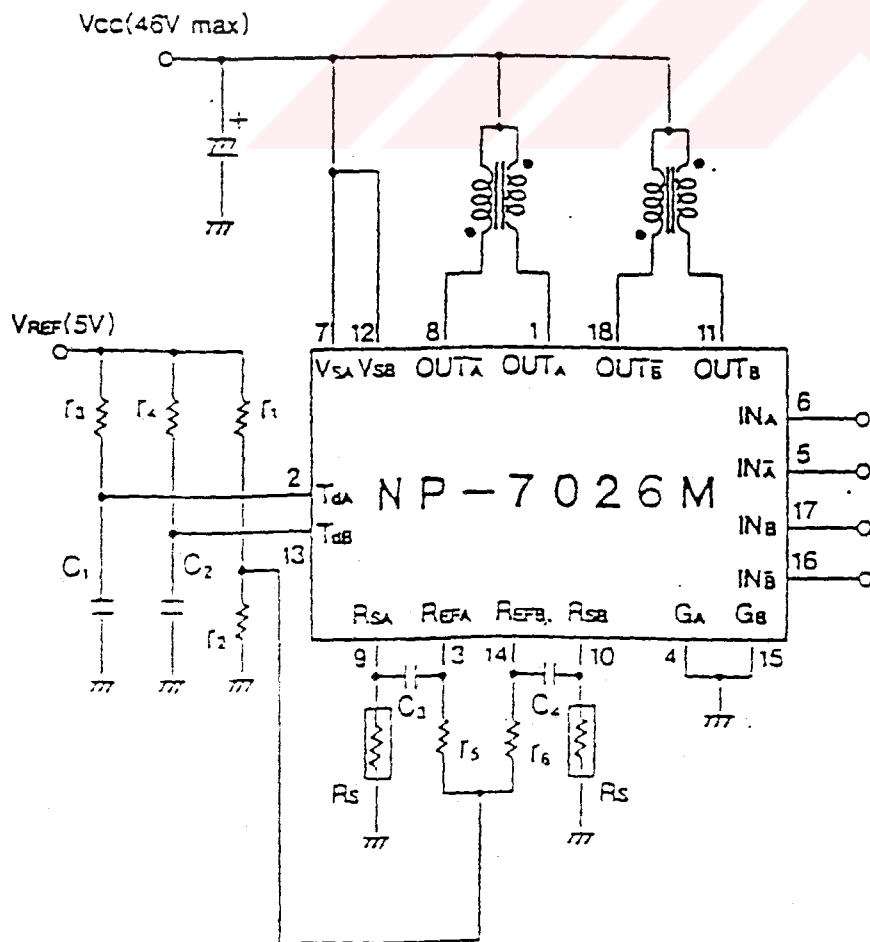
-2 Excitation Timing Chart (Active 'L')

2 Phase excitation

Clock	0	1	2	3	0	1
INA	L	H	H	L	L	H
INA ⁻	H	L	L	H	H	L
INB	L	L	H	H	L	L
INB ⁻	H	H	L	L	H	H

1-2 Phase excitation

Clock	0	1	2	3	4	5	6	7	0	1	2	3
INA	L	L	H	H	H	H	H	L	L	L	H	H
INA ⁻	H	H	H	L	L	L	H	H	H	H	H	L
INB	H	L	L	L	H	H	H	H	H	L	L	L
INB ⁻	H	H	H	H	H	L	L	L	H	H	H	H



- R_1 510 Ω
- R_2 100 Ω (VR)
- R_3 47k Ω
- R_4 47k Ω
- R_5 2.4k Ω
- R_6 2.4k Ω
- C_1 470pF
- C_2 470pF
- C_3 2200pF
- C_4 2200pF
- R_s 0.68 Ω (typ)1-2W

1. Dimensions

1 External Appearance

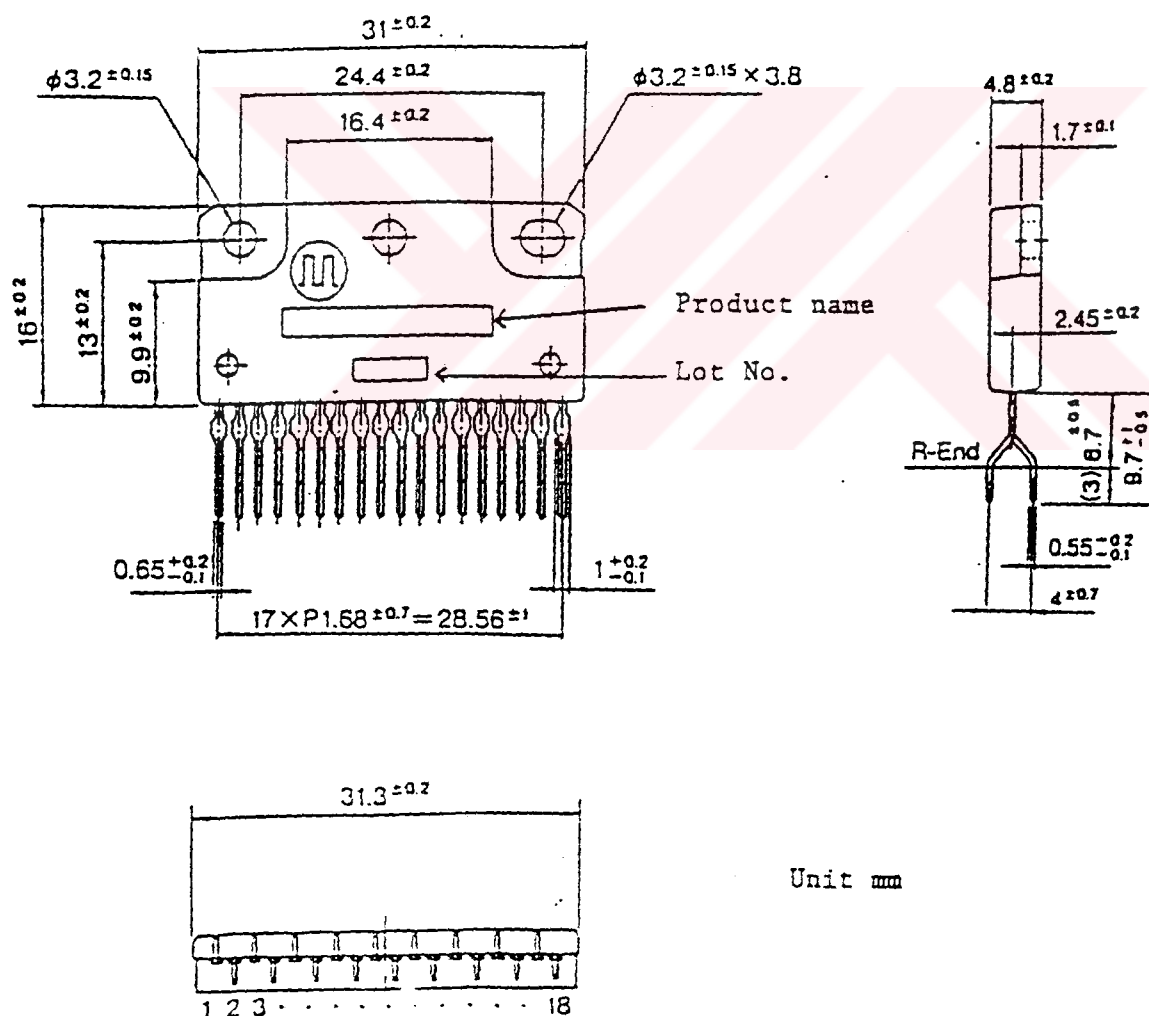
External appearance should be free of dirt, flaw or crack.

2 Dimensions

See below

3 Marking

Company symbol, product name & lot No. should be marked with white ink.



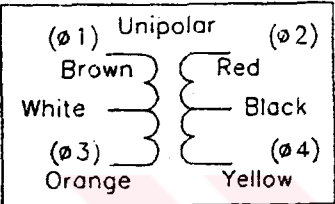
SPECIFICATIONS ARE SUBJECT TO CHANGE WITHOUT PRIOR NOTICE FOR IMPROVEMENT.

and PQ Series, Unipolar

	ø 1	ø 2	ø 3	ø 4	Common		Direction of Rotation <div><div>↓</div><div>↑</div><div>CW</div><div>CCW</div></div>
ds	Brown	Red	Orange	Yellow			
ds	Brown	Red	Orange	Yellow			
1	On	On	Off	Off	<div><div>White</div><div>Blue</div><div>Black</div><div>Green</div></div>	White Black	
2	Off	On	On	Off			
3	Off	Off	On	On			
4	On	Off	Off	On			
1	On	On	Off	Off			

- 8 Lead Unipolar
- \bar{A}
- ($\phi 1$) Brown
 - (Com.) White
 - (Com.) Blue
- \bar{A}
- ($\phi 3$) Orange
 - ($\phi 2$) Red
- \bar{B}
- (Com.) Black
 - (Com.) Green
- \bar{B}
- ($\phi 4$) Yellow

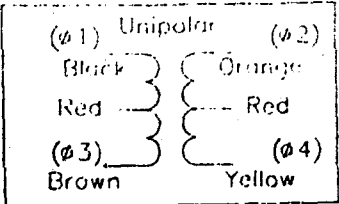
Wiring Diagram



eries, Unipolar

	$\phi 1$	$\phi 2$	$\phi 3$	$\phi 4$	Common	Direction of Rotation ↓ CCW ↑ CW
ds	Black	Orange	Brown	Yellow		
	On	On	Off	Off	Red Red	
	Off	On	On	Off		
	Off	Off	On	On		
	On	Off	Off	On		
	On	On	Off	Off		

Wiring Diagram



CLLMORGEN

tion Technologies Group

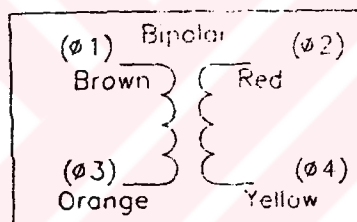
201 Rock Road
Radford, VA, 24141
Ph: 540-633-4175
Fax: 540-731-451

ur Series

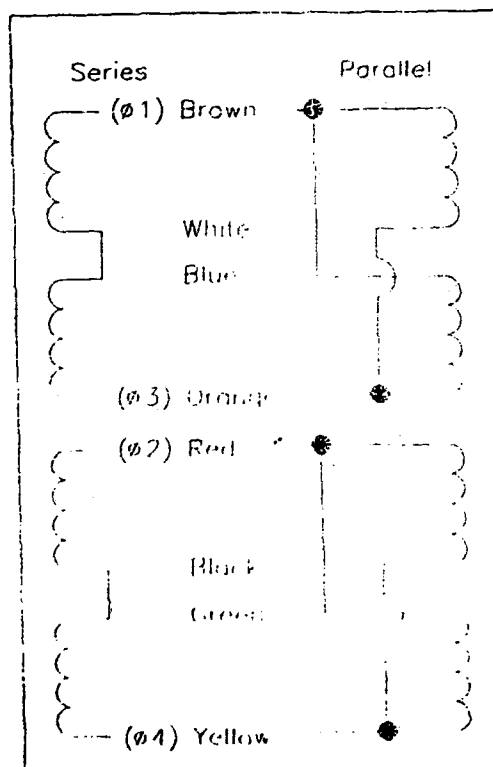
Color	$\phi 1$	$\phi 2$	$\phi 3$	$\phi 4$	Direction of Rotation
Leads					
ind PQ 4 Leads	Brown	Red	Orange	Yellow	
ind PQ 8 Leads					
Step 1	On	Off	On	Off	
2	On	Off	Off	On	
3	Off	On	Off	On	
4	Off	On	On	Off	
1	On	Off	On	Off	

Wiring Diagram

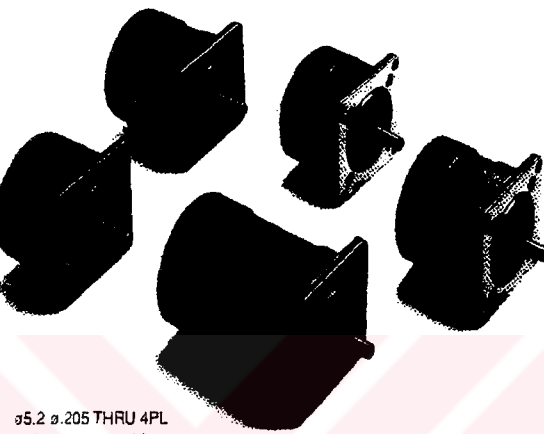
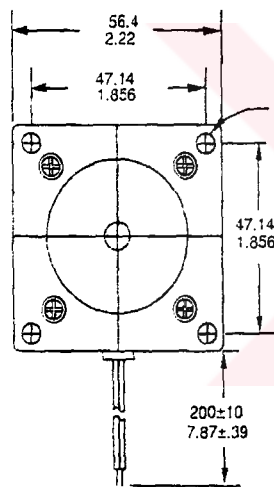
4 Leads Bipolar



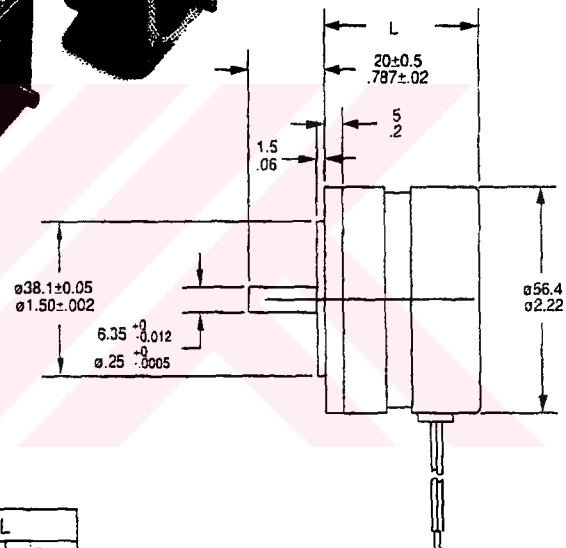
8 Leads Bipolar



Unit: mm
in.

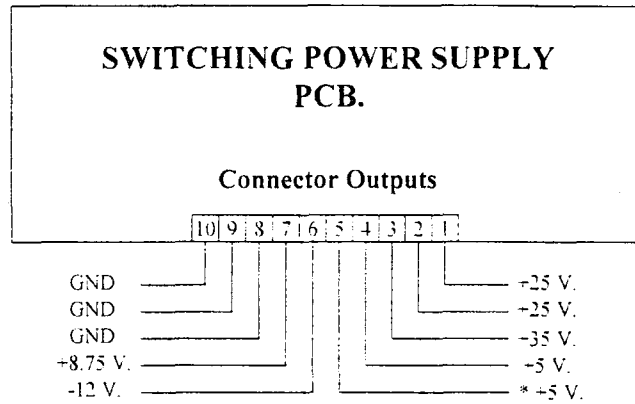


ø5.2 ø.205 THRU 4PL
B.C.D. 66.68 2.625



	L	
	mm	in
PJT39	39	1.54
PJT42	42	1.65
PJT50	50	1.97
PJT55	55	2.17
PJT70	70	2.75
PJT80	80	3.15

TEST RESULTS FOR 35 V DC. POWER SUPPLY

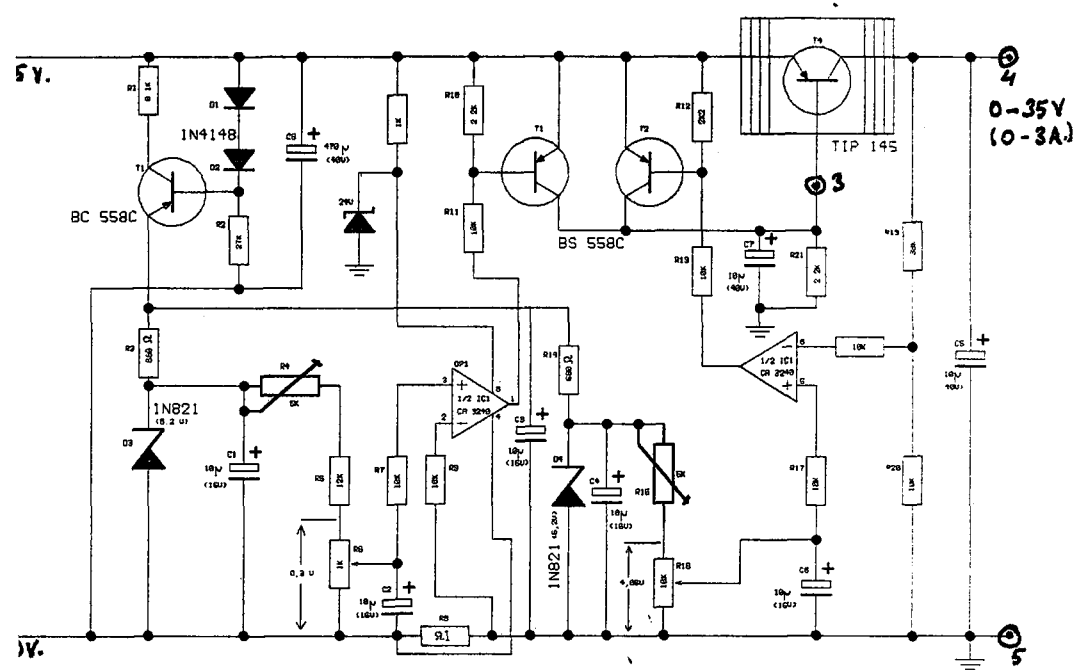


V_{out} (no load)	R_L (Load resistor)	V_{out} (Output when load resistor connected)	I_L (Load current)	COMMENTS
+24.5 V _{dc}	220 Ω	+24.4 V _{dc}	102 mA	OK.
+34.5 V _{dc}	336 Ω	+34.3 V _{dc}	101 mA	OK.
+5.03 V _{dc}	52.1 Ω	+5.02 V _{dc}	93.8 mA	OK.
* +5.02 V _{dc}	52.1 Ω	+2.29 V _{dc}	47 mA	Low Current. (pin 5)
-12.10 V _{dc}	152.8 Ω	-12.08 V _{dc}	78 mA	OK.
+8.75 V _{dc}	88.7 Ω	+7.76 V _{dc}	85 mA	OK.

Note 1-) The measurements have been done under all connector outputs loaded with relevant R_L .

V_{out} (no load)	I (Driven current)	V_{out} (Output when load resistor connected)	V_{ripple} (Ripple voltage occured at the output)
+24.5 V _{dc}	1 A.	+24.2 V _{dc}	200 mV _{p-p}
+34.5 V _{dc}	1 A.	+33.4 V _{dc}	900 mV _{p-p}
	0.5 A.	+33.9 V _{dc}	550 mV _{p-p}
+5.03 V _{dc}	1 A.	+4.92 V _{dc}	110 mV _{p-p}
* +5.02 V _{dc}	15.5 mA.	+4.91 V _{dc}	60 mV _{p-p}
	25 mA.	+4.79 V _{dc}	70 mV _{p-p}
	30 mA.	+4.66 V _{dc}	80 mV _{p-p}
	35 mA.	+4.3 V _{dc}	500 mV _{p-p}
	37 mA.	+4.0 V _{dc}	720 mV _{p-p}
-12.10 V _{dc}	150 mA.	-12.01 V _{dc}	Not loaded so much since heat sink does not mounted !
+8.75 V _{dc}	25 mA.	+7.93 V _{dc}	225 mV _{p-p} 275 mV _{p-p}
	50 mA.	+7.83 V _{dc}	
	75 mA.	+7.78 V _{dc}	
	150 mA.	+7.68 V _{dc}	

Note 2-) All outputs are measured one by one. During the measurement of an output, all other outputs are loaded with the relevant load resistors as mentioned above tables.



	ANKARA TT&C STATION		ANKARA ODTU STATION
ANTENNA COORDINATES			
LATITUDE (°N)	39°38'27"		
LONGITUDE (°E)	32°49'25"		
HEIGHT	1045 m		
ANTENNA TYPE	11 m Full motion antenna	9 m Limited motion antenna	9 m Limited motion antenna
ANTENNA TRANSMIT GAIN (MIDBAND)	62.2 dBi	60.4 dBi	60.4 dBi
ANTENNA RECEIVE BAND (MIDBAND)	60.4 dBi	58.2 dBi	58.2 dBi
G/T (MIDBAND)	36.4 dB/K	35 dB/K	35 dB/K
EIRP	91 dBW	83 dBW	83 dBW
TRACKING MAX. SLEW RATE	MONOPULSE 0.5°/s	STEP-TRACK 0.08°/s	STEP-TRACK 0.08°/s
RMS POINTING ACCURACY	0.01° rms	0.01° rms	0.01° rms
RMS TRACKING ACCURACY	0.003° rms	0.015° rms	0.015° rms
MINIMUM ELEVATION	5°	5°	5°
POLARIZATION	Linear orientable Tracking linear/ circular	Linear orientable	Linear orientable
TRANSMIT FREQUENCY BAND RECEIVE FREQUENCY BAND	14-14.5 GHz 10.95-11.70 GHz	14-14.5 GHz 10.95-11.70 GHz	14-14.5 GHz 10.95-11.70 GHz
3 dB BEAMWIDTH (°)	Tx: 0.12° Rx: 0.15°	Tx: 0.15° Rx: 0.18°	Tx: 0.15° Rx: 0.18°

ANTENNA RECEIVE & TRANSMIT TYPICAL CHARACTERISTICS

APPENDIX B



Antenna.cpp File

```
#include "Antenna.h"
#include <windowsx.h>

bool  azimuth_elev_by_map=FALSE;
bool  testarea=FALSE;
MAPCOORD map_coor;
AZELD azeld1;
AZELDROT azeldrot1;
HBITMAP hMap_Turkey;
HBITMAP hMap_lcfoot;
HBITMAP hMap_lbfoot;
HBITMAP hMap_ebru;
HBITMAP hMap_chlb;
HBITMAP hMap_chlc;
HDC hDC;
HINSTANCE hInsAbout;
HINSTANCE hInsT1B;
HINSTANCE hInsT1C;
HINSTANCE hInsManual;
HINSTANCE hInsT1BH;
HINSTANCE hInsT1CH;
HWND hDlgrot;
int data,d;

void autorotate(void)
{
HOut(0x378,0x00);
int data1,data2,don,don1;
double azimuth,elev;
struct projedata step;

step.dat1 = azeld1.FILE_DAT1_I;
step.dat2 = azeld1.FILE_DAT2_I;
step.oldelev = azeld1.FILE_ELEVATION_D;
step.oldazim = azeld1.FILE_AZIMUTH_D;
data1=step.dat1;
data2=step.dat2;
step.newelev = azeldrot1.ELEVATION_ROT_D;
step.newazim = azeldrot1.AZIMUTH_ROT_D;

if(step.oldelev>step.newelev)
{
    MessageBox(hDlgrot,(LPSTR) "step.oldelev > step.newelev",
                (LPSTR) "Autorotate Function", MB_OK |
MB_ICONASTERISK);
    elev=(step.oldelev-step.newelev)/0.9;
    don1=sagadon1(elev,data2);
    HOut(0x378,0x00);
}
else
{
    MessageBox(hDlgrot,(LPSTR) "step.oldelev < step.newelev",
                (LPSTR) "Autorotate Function", MB_OK |
MB_ICONASTERISK);
    elev=(step.newelev-step.oldelev)/0.9;
    don1=soladon1(elev,data2);
    HOut(0x378,0x00);
}

if(step.oldazim>step.newazim)
{
    MessageBox(hDlgrot,(LPSTR) "step.oldazim > step.newazim",
                (LPSTR) "Autorotate Function", MB_OK |
MB_ICONASTERISK);
```

```

        azim=(step.oldazi-step.newazi)/0.018;
        don=soladon(azim,data1);
        HOut(0x378,0x00);
    }
else
    {
        MessageBox(hDlgrot,(LPSTR) "step.oldazi < step.newazi",
                    (LPSTR) "Autorotate Function", MB_OK |
MB_ICONASTERISK);
        azim=(step.newazi-step.oldazi)/0.018;
        don=sagadon(azim,data1);
        HOut(0x378,0x00);
    }

```

```

step.oldazi=step.newazi;
step.dat1=don;
step.oldele=step.newele;
step.dat2=don1;
azeld1.FILE_DAT1_I = step.dat1;
azeld1.FILE_DAT2_I = step.dat2;
azeld1.FILE_ELEVATION_D = step.oldele;
azeld1.FILE_AZIMUTH_D = step.oldazi;
}

```

```

BOOL CALLBACK AboutDlgProc(HWND hDlgAbout, UINT uMesg, WPARAM
wParama, LPARAM lParama)
{
    switch (uMesg) {
        case WM_INITDIALOG:
            return (TRUE);

        case WM_COMMAND:
            switch (GET_WM_COMMAND_ID(wParama, lParama))
            {
                case IDABOUTOK:
                {
                    EndDialog(hDlgAbout, TRUE);
                }
                return (TRUE);
            }
            break;

        default:
            return(FALSE);
    }
    return(TRUE);
}

```

```

BOOL CALLBACK T1BHDlgProc(HWND hDlgT1BH, UINT uMesgbh, WPARAM
wParama, LPARAM lParama)
{
    switch (uMesgbh) {
        case WM_INITDIALOG:
            return (TRUE);

        case WM_COMMAND:
            switch (GET_WM_COMMAND_ID(wParama, lParama))
            {
                case IDOK:
                {
                    EndDialog(hDlgT1BH, TRUE);
                }
                return (TRUE);
            }
            break;
    }
}

```

```

        default:
            return(FALSE);
        }
        return(TRUE);
    }

BOOL CALLBACK T1CHDlgProc(HWND hDlgT1CH, UINT uMsgch, WPARAM
wParama, LPARAM lParama)
{
    switch (uMsgch) {
        case WM_INITDIALOG:
            return (TRUE);

        case WM_COMMAND:
            switch (GET_WM_COMMAND_ID(wParama, lParama))
            {
                case IDOK:
                {
                    EndDialog(hDlgT1CH, TRUE);
                }
                return (TRUE);
            }
            break;

        default:
            return(FALSE);
    }
    return(TRUE);
}

BOOL CALLBACK ManualDlgProc(HWND hDlgManual, UINT uMesgm, WPARAM
wParama, LPARAM lParama)
{
    switch (uMesgm) {
        case WM_INITDIALOG:
            return (TRUE);

        case WM_COMMAND:
            switch (GET_WM_COMMAND_ID(wParama, lParama))
            {
                case IDOK:
                {
                    EndDialog(hDlgManual, TRUE);
                }
                return (TRUE);
            }
            case AZCCW:
            {
                ReadData(&azeld1.FILE_AZIMUTH_D,&azeld1.FILE_
ELEVATION_D,&azeld1.FILE_DAT1_I,&azeld1.FILE_DAT2_I);

                data = azeld1.FILE_DAT1_I;
                if(data==1)
                {
                    HOut(0x378,0x08);

                    gecikmel();

                    d=8;
                }
                if(data==2)
                {
                    HOut(0x378,0x01);

                    gecikmel();

                    d=1;
                }
                if(data==4)
                {

```

```

        HOut(0x378,0x02);
        gecikmel();
        d=2;
    }
    if(data==8)
    {
        HOut(0x378,0x04);
        gecikmel();
        d=4;
    }

    azeld1.FILE_AZIMUTH_D =
azeld1.FILE_AZIMUTH_D - 0.018;
    azeld1.FILE_DAT1_I = d;
    WriteData(azeld1.FILE_AZIMUTH_D,azeld1.FILE_ELEV
ATION_D,azeld1.FILE_DAT1_I,azeld1.FILE_DAT2_I);
    HOut(0x378,0x00);
    }
    return (TRUE);
    case AZCW:
    {
        ReadData(&azeld1.FILE_AZIMUTH_D,&azeld1.FILE_
ELEVATION_D,&azeld1.FILE_DAT1_I,&azeld1.FILE_DAT2_I);

        data = azeld1.FILE_DAT1_I;
        if(data==1)
        {
            HOut(0x378,0x02);
            gecikmel();
            d=2;
        }
        if(data==2)
        {
            HOut(0x378,0x04);
            gecikmel();
            d=4;
        }
        if(data==4)
        {
            HOut(0x378,0x08);
            gecikmel();
            d=8;
        }
    }
    if(data==8)
    {
        HOut(0x378,0x01);
        gecikmel();
        d=1;
    }

    azeld1.FILE_AZIMUTH_D =
azeld1.FILE_AZIMUTH_D + 0.018;
    azeld1.FILE_DAT1_I = d;
    WriteData(azeld1.FILE_AZIMUTH_D,azeld1.FILE_ELEV
ATION_D,azeld1.FILE_DAT1_I,azeld1.FILE_DAT2_I);
    HOut(0x378,0x00);
    }
    return (TRUE);
    case ELCCW:
    {
        ReadData(&azeld1.FILE_AZIMUTH_D,&azeld1.FILE_
ELEVATION_D,&azeld1.FILE_DAT1_I,&azeld1.FILE_DAT2_I);

        data = azeld1.FILE_DAT2_I;
        if(data==1)
        {
            HOut(0x378,0x90);

```

```

        gecikmel();
        d=9;
    }
    if(data==3)
    {
        HOut(0x378,0x10);
        gecikmel();
        d=1;
    }
    if(data==2)
    {
        HOut(0x378,0x30);
        gecikmel();
        d=3;
    }

    if(data==6)
    {
        HOut(0x378,0x20);
        gecikmel();
        d=2;
    }
    if(data==4)
    {
        HOut(0x378,0x60);
        gecikmel();
        d=6;
    }
    if(data==12)
    {
        gecikmel();
        d=4;
    }
    if(data==8)
    {
        HOut(0x378,0xC0);
        gecikmel();
        d=12;
    }
    if(data==9)
    {
        HOut(0x378,0x80);
        gecikmel();
        d=8;
    }

    azeld1.FILE_ELEVATION_D =
azeld1.FILE_ELEVATION_D - 0.9;
    azeld1.FILE_DAT2_I = d;
    WriteData(azeld1.FILE_AZIMUTH_D,azeld1.FILE_ELEV
ATION_D,azeld1.FILE_DAT1_I,azeld1.FILE_DAT2_I);
    HOut(0x378,0x00);
    }
    return (TRUE);
case ELCW:
    {
        ReadData(&azeld1.FILE_AZIMUTH_D,&azeld1.FILE_
ELEVATION_D,&azeld1.FILE_DAT1_I,&azeld1.FILE_DAT2_I);

        data = azeld1.FILE_DAT2_I;
        if(data==1)
        {
            HOut(0x378,0x30);
            gecikmel();
            d=3;

```



```

    }

    if(data==3)
    {
        HOut(0x378,0x20);
        gecikmel();
        d=2;
    }
    if(data==2)
    {
        HOut(0x378,0x60);
        gecikmel();
        d=6;
    }
    if(data==6)
    {
        HOut(0x378,0x40);
        gecikmel();
        d=4;
    }
    if(data==4)
    {
        HOut(0x378,0xC0);
        gecikmel();
        d=12;
    }
    if(data==12)
    {
        HOut(0x378,0x80);
        gecikmel();
        d=8;
    }
    if(data==8)
    {
        HOut(0x378,0x90);
        gecikmel();
        d=9;
    }
    if(data==9)
    {
        HOut(0x378,0x10);
        gecikmel();
        d=1;
    }
    azeld1.FILE_ELEVATION_D =
    azeld1.FILE_ELEVATION_D + 0.9;
    azeld1.FILE_DAT2_I = d;
    WriteData(azeld1.FILE_AZIMUTH_D,azeld1.FILE_ELEVATION_D,azeld1.FILE_DAT1_I,azeld1.FILE_DAT2_I);
    HOut(0x378,0x00);
    }
    return (TRUE);
}
break;
default:
return(FALSE);
}
return(TRUE);
}

BOOL CALLBACK T1BFootDlg(HWND hDlgT1B, UINT uMesgb, WPARAM wParama,
LPARAM lParama)
{
    switch (uMesgb) {
        case WM_INITDIALOG:

```

```

        return (TRUE);

    case WM_COMMAND:
        switch (GET_WM_COMMAND_ID(wParam, lParam))
        {
            case IDOK:
            {
                EndDialog(hDlgT1B, TRUE);
            }
            return (TRUE);
        }
        break;

        default:
        return(FALSE);
    }
    return(TRUE);
}

BOOL CALLBACK T1CFootDlg(HWND hDlgT1C, UINT uMsg, WPARAM wParam,
LPARAM lParam)
{
    switch (uMsg) {
        case WM_INITDIALOG:
            return (TRUE);

        case WM_COMMAND:
            switch (GET_WM_COMMAND_ID(wParam, lParam))
            {
                case IDOK:
                {
                    EndDialog(hDlgT1C, TRUE);
                }
                return (TRUE);
            }
            break;

            default:
            return(FALSE);
        }
        return(TRUE);
    }
}

BOOL CALLBACK OpenDlgProc(HWND hDlg, UINT uMsg, WPARAM wParam, LPARAM
lParam)
{
    char E_S_LAT_CHAR[20], E_S_LONG_CHAR[20];
    char SAT_LAT_CHAR[20], SAT_LONG_CHAR[20];
    char ELEVATION_CHAR[20], AZIMUTH_CHAR[20];
    hDlgrot = hDlg;

    switch (uMsg) {
        case WM_INITDIALOG:

            ELEVATION_CHAR[0] = NULL;
            AZIMUTH_CHAR[0] = NULL;
            CheckRadioButton(hDlg, IC_AZ_EL_SATELLITE_T1B,
                IC_AZ_EL_SATELLITE_T1C,
IC_AZ_EL_SATELLITE_T1B );
            strcpy(SAT_LAT_CHAR, "0");
            SetDlgItemText(hDlg, SAT_LATITUDE, SAT_LAT_CHAR);

            strcpy(SAT_LONG_CHAR, "31.3");
            SetDlgItemText(hDlg, SAT_LONGITUDE, SAT_LONG_CHAR);
            ReadData(&azeld1.FILE_AZIMUTH_D, &azeld1.FILE_ELEVAT

```

```

ION_D,&azeld1.FILE_DAT1_I,&azeld1.FILE_DAT2_I);
    return (TRUE);

// Sol Butona Basildiginda cursor koordinati alacak
    case WM_LBUTTONDOWN:
        testarea = Draw_Map(hDlg, hDC, lParam, hMap_Turkey);
        if (testarea == TRUE)
        {
            if (azim_elev_by_map == TRUE )
            {
                map_coor=Get_Map_Coor();
                strcpy(E_S_LAT_CHAR,map_coor.latitude);
                SetDlgItemText(hDlg,E_STATION_LAT,E_S_L
AT_CHAR);
                strcpy(E_S_LONG_CHAR,map_coor.longitude
);
                SetDlgItemText(hDlg,E_STATION_LONG,E_S_LONG_C
HAR);
            }
        }
        break;

    case WM_DESTROY:
        PostQuitMessage(0);
        break;

case WM_COMMAND:
    switch (GET_WM_COMMAND_ID(wParam, lParam))
    {
        case IC_AZ_EL_SATELLITE_T1B :
            strcpy(SAT_LAT_CHAR,"0");
            SetDlgItemText(hDlg,SAT_LATITUDE,SAT_LAT_CHAR);

            strcpy(SAT_LONG_CHAR,"31.3");
            SetDlgItemText(hDlg,SAT_LONGITUDE,SAT_LONG_CH
AR);
            break;

        case IC_AZ_EL_SATELLITE_T1C :
            strcpy(SAT_LAT_CHAR,"0");
            SetDlgItemText(hDlg,SAT_LATITUDE,SAT_LAT_CHAR);

            strcpy(SAT_LONG_CHAR,"42.0");
            SetDlgItemText(hDlg,SAT_LONGITUDE,SAT_LONG_CH
AR);
            break;

        case IDOK:
            {
                E_S_LAT_CHAR[0]=NULL;

                GetDlgItemText(hDlg,E_STATION_LAT,E_S_LAT_CHAR,20);
                if (E_S_LAT_CHAR[0]==NULL)
                    MessageBox(hDlg, (LPSTR) "ENTER EARTH
STATION LATITUDE",
                    (LPSTR) "ENTER THE VALUE",MB_OK |
MB_ICONASTERISK);

                E_S_LONG_CHAR[0]=NULL;
                GetDlgItemText(hDlg,E_STATION_LONG,E_S_LONG_CHAR,20
);
                if (E_S_LONG_CHAR[0]==NULL)
                    MessageBox(hDlg, (LPSTR) "ENTER EARTH
STATION LONGITUDE",
                    (LPSTR) "ENTER THE VALUE",MB_OK |
MB_ICONASTERISK);
            }
    }
}

```

```

        SAT_LAT_CHAR[0]=NULL;
        GetDlgItemText(hDlg,SAT_LATITUDE,SAT_LAT_CHAR,20);
        if (SAT_LAT_CHAR[0]==NULL)
            MessageBox(hDlg,(LPSTR) "ENTER
SATELLITE LATITUDE",
                    (LPSTR) "ENTER THE VALUE", MB_OK |
MB_ICONASTERISK );

        SAT_LONG_CHAR[0]=NULL;
        GetDlgItemText(hDlg,SAT_LONGITUDE,SAT_LONG_CHAR,20)
;
        if (SAT_LONG_CHAR[0]==NULL)
            MessageBox(hDlg,(LPSTR) "ENTER
SATELLITE LONGITUDE",
                    (LPSTR) "ENTER THE VALUE", MB_OK |
MB_ICONASTERISK);

        strcpy(ELEVATION_CHAR,calculate_elevation(E_S_LAT_C
HAR,E_S_LONG_CHAR,SAT_LAT_CHAR,SAT_LONG_CHAR));
        strcpy(AZIMUTH_CHAR,calculate_azimuth(E_S_LAT_CHAR,
E_S_LONG_CHAR,SAT_LAT_CHAR,SAT_LONG_CHAR));

        if ((E_S_LAT_CHAR[0]!=NULL) &&
(E_S_LONG_CHAR[0]!=NULL) &&
(SAT_LAT_CHAR[0]!=NULL) &&
(SAT_LONG_CHAR[0]!=NULL))
        {
            SetDlgItemText(hDlg,ANTENNA_ELEVATION,ELEVATI
ON_CHAR);
            SetDlgItemText(hDlg,ANTENNA_AZIMUTH,AZIMUTH_C
HAR);
        }

    }
    return (TRUE);

    case IDCANCEL:
        EndDialog(hDlg, TRUE);
        break;
    case IDMANUAL:
        DialogBox(hInsManual, MAKEINTRESOURCE(MANUALBOX),
hDlg, ManualDlgProc);
        break;
    case IDRESET:
        ReadData(&azeld1.FILE_AZIMUTH_D,&azeld1.FILE_ELEVAT
ION_D,&azeld1.FILE_DAT1_I,&azeld1.FILE_DAT2_I);
        azeldrot1 = Get_AZELDROT();
        azeldrot1.ELEVATION_ROT_D = 0;
        azeldrot1.AZIMUTH_ROT_D = 0;
        autorotate();
        azeld1.FILE_DAT1_I = 1;
        azeld1.FILE_DAT2_I = 1;
        WriteData(azeld1.FILE_AZIMUTH_D,azeld1.FILE_ELEVATION_
D,azeld1.FILE_DAT1_I,azeld1.FILE_DAT2_I);

        break;
    case IDABOUT:
        DialogBox(hInsAbout, MAKEINTRESOURCE(ABOUTBOX),
hDlg, AboutDlgProc);
        break;
    case T1BHISTORY:
        DialogBox(hInsT1BH, MAKEINTRESOURCE(T1BHISTORYBOX),
hDlg, T1BHDlgProc);
        break;
    case T1CHISTORY:

```

```

        DialogBox(hInstT1CH, MAKEINTRESOURCE(T1CHISTORYBOX),
hDlg, T1CHDdlgProc);
        break;
        case ID_1B:
            DialogBox(hInstT1B, MAKEINTRESOURCE(T1BFOOTBOX),
hDlg, T1BFootDlg);
            break;
        case ID_1C:
            DialogBox(hInstT1C, MAKEINTRESOURCE(T1CFOOTBOX),
hDlg, T1CFootDlg);
            break;
        case IDROTATE:
            ReadData(&azeld1.FILE_AZIMUTH_D,&azeld1.FILE_ELEVAT
ION_D,&azeld1.FILE_DAT1_I,&azeld1.FILE_DAT2_I);
            azeldrot1 = Get_AZELDROT();
            autorotate();
            WriteData(azeld1.FILE_AZIMUTH_D,azeld1.FILE_ELEVATION_
D,azeld1.FILE_DAT1_I,azeld1.FILE_DAT2_I);

            break;
    }
    break;

    default:
        return(FALSE);
}

return(TRUE);
}

```

```

//-----
-----
#pragma argsused
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
LPSTR lpCmdLine, int nCmdShow)
{
    OSVERSIONINFO osVer; // for GetVersionEx()
    // Verify presence of Windows 95
    osVer.dwOSVersionInfoSize = sizeof(osVer);
    if (!GetVersionEx(&osVer))
        return (FALSE);

    if (osVer.dwPlatformId != VER_PLATFORM_WIN32_WINDOWS)
    {
        MessageBox(NULL, "This example requires Windows 95.",
            "Antenna Control Software", MB_OK );
        return (FALSE);
    }

    azim_elev_by_map=TRUE;
    hMap_Turkey=LoadBitmap(hInstance,"TURKIYE");
    hMap_lcfoot=LoadBitmap(hInstance,"1CFOOT");
    hMap_lbfoot=LoadBitmap(hInstance,"1BFOOT");
    hMap_ebru=LoadBitmap(hInstance,"EBRU");
    hMap_ch1b=LoadBitmap(hInstance,"CH1B");
    hMap_ch1c=LoadBitmap(hInstance,"CH1C");

    DialogBox(hInstance, MAKEINTRESOURCE(MYDLGBOX), NULL,
OpenDlgProc);
    DeleteObject(hMap_Turkey);
    DeleteObject(hMap_lbfoot);
    DeleteObject(hMap_lcfoot);
    DeleteObject(hMap_ebru);
    DeleteObject(hMap_ch1b);
    DeleteObject(hMap_ch1c);
}

```

```
}    return(FALSE);
```



```
/*  
*****  
*****
```

antenna.rc

produced by Borland Resource Workshop

```
*****  
*****/  

```

```
#define IDC_GROUPBOX1 101  
#define T1BHISTORYBOX 160  
#define T1CHISTORYBOX 161  
#define T1BHISTORY 162  
#define T1CHISTORY 163  
#define ELCW 156  
#define ELCCW 155  
#define AZCW 154  
#define AZCCW 153  
#define MANUALBOX 152  
#define IDMANUAL 151  
#define IDRESET 150  
#define T1CFOOTBOX 118  
#define T1BFOOTBOX 117  
#define ID_1C 108  
#define ID_1B 107  
#define IDROTATE 102  
#define IDABOUTOK 11  
#define ABOUTBOX 101  
#define IDABOUT 10  
#define ANTENNA_AZIMUTH 106  
#define ANTENNA_ELEVATION 105  
#define E_STATION_LONG 132  
#define E_STATION_LAT 131  
#define SAT_LONGITUDE 104  
#define SAT_LATITUDE 103  
#define IC_AZ_EL_SATELLITE_T1C 121  
#define IC_AZ_EL_SATELLITE_T1B 120  
#define MYDLGBOX 100
```

```
MYDLGBOX DIALOG 0, 0, 425, 334  
EXSTYLE WS_EX_DLGMODALFRAME  
STYLE DS_MODALFRAME | DS_CENTER | WS_POPUP | WS_VISIBLE | WS_CAPTION  
| WS_SYSMENU  
CAPTION "Antenna Motion Control"  
FONT 8, "MS Sans Serif"  
{  
    CONTROL "Exit", IDCANCEL, "BUTTON", BS_PUSHBUTTON | BS_CENTER |  
WS_CHILD | WS_VISIBLE | WS_TABSTOP, 360, 310, 50, 14  
    CONTROL "turkiye", -1, "static", SS_BITMAP | SS_REALSIZEIMAGE |  
WS_CHILD | WS_VISIBLE | WS_BORDER, 0, 0, 424, 205, WS_EX_CLIENTEDGE  
    CONTROL "Satellite Selection", 601, "button", BS_GROUPBOX |  
WS_CHILD | WS_VISIBLE | WS_GROUP, 4, 215, 84, 52  
    CONTROL "Satellite Location", 602, "button", BS_GROUPBOX | WS_CHILD  
| WS_VISIBLE | WS_GROUP, 4, 276, 84, 52  
    CONTROL "Antenna Map Location", 603, "button", BS_GROUPBOX |  
WS_CHILD | WS_VISIBLE | WS_GROUP, 96, 215, 116, 52  
    CONTROL "Calculated Values", 604, "button", BS_GROUPBOX | WS_CHILD |  
WS_VISIBLE | WS_GROUP, 96, 276, 116, 52  
    CONTROL "Turksat T1C", IC_AZ_EL_SATELLITE_T1C, "button",  
BS_AUTORADIOBUTTON | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 12, 246, 56,  
8  
    CONTROL "Turksat T1B", IC_AZ_EL_SATELLITE_T1B, "button",  
BS_AUTORADIOBUTTON | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 12, 228, 56,
```

9

```

CONTROL "", SAT_LATITUDE, "edit", ES_LEFT | WS_CHILD | WS_VISIBLE |
WS_BORDER | WS_TABSTOP, 48, 293, 32, 9
CONTROL "", SAT_LONGITUDE, "edit", ES_LEFT | WS_CHILD | WS_VISIBLE |
WS_BORDER | WS_TABSTOP, 48, 310, 32, 9
CONTROL "", E_STATION_LAT, "edit", ES_LEFT | WS_CHILD | WS_VISIBLE |
WS_BORDER | WS_TABSTOP, 172, 228, 32, 9
CONTROL "", E_STATION_LONG, "edit", ES_LEFT | WS_CHILD | WS_VISIBLE |
WS_BORDER | WS_TABSTOP, 172, 246, 32, 9
CONTROL "", ANTENNA_ELEVATION, "edit", ES_LEFT | ES_READONLY |
WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP, 172, 292, 32, 9
CONTROL "", ANTENNA_AZIMUTH, "edit", ES_LEFT | ES_READONLY |
WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP, 172, 310, 32, 9
CONTROL "Latitude", -1, "static", SS_LEFT | WS_CHILD | WS_VISIBLE,
12, 293, 32, 9
CONTROL "Longitude", -1, "static", SS_LEFT | WS_CHILD | WS_VISIBLE,
12, 310, 32, 9
CONTROL "Antenna Latitude", -1, "static", SS_LEFT | WS_CHILD |
WS_VISIBLE, 104, 228, 60, 9
CONTROL "Antenna Longitude", -1, "static", SS_LEFT | WS_CHILD |
WS_VISIBLE, 104, 246, 60, 8
CONTROL "Antenna Elevation", -1, "static", SS_LEFT | WS_CHILD |
WS_VISIBLE, 104, 292, 64, 9
CONTROL "Antenna Azimuth", -1, "static", SS_LEFT | WS_CHILD |
WS_VISIBLE, 104, 310, 60, 9
CONTROL "Calculate", IDOK, "button", BS_PUSHBUTTON | BS_CENTER |
WS_CHILD | WS_VISIBLE | WS_TABSTOP, 228, 228, 50, 14
CONTROL "About", IDABOUT, "button", BS_PUSHBUTTON | BS_CENTER |
WS_CHILD | WS_VISIBLE | WS_TABSTOP, 300, 310, 50, 14
CONTROL "Antenna Motion Control and", -1, "static", SS_LEFT |
WS_CHILD | WS_VISIBLE, 96, 202, 112, 8
CONTROL "Azimuth - Elevation Calculation", -1, "static", SS_LEFT |
WS_CHILD | WS_VISIBLE, 188, 202, 100, 8
CONTROL "Software.", -1, "static", SS_LEFT | WS_CHILD | WS_VISIBLE,
288, 202, 60, 8
CONTROL "Rotate", IDROTATE, "button", BS_PUSHBUTTON | BS_CENTER |
WS_CHILD | WS_VISIBLE | WS_TABSTOP, 228, 254, 50, 14
CONTROL "Foot Prints", 605, "button", BS_GROUPBOX | WS_CHILD |
WS_VISIBLE | WS_GROUP, 292, 215, 128, 35
CONTROL "Turksat 1B", ID1B, "button", BS_PUSHBUTTON | BS_CENTER |
WS_CHILD | WS_VISIBLE | WS_TABSTOP, 300, 228, 50, 14
CONTROL "Turksat 1C", ID1C, "button", BS_PUSHBUTTON | BS_CENTER |
WS_CHILD | WS_VISIBLE | WS_TABSTOP, 360, 228, 50, 14
CONTROL "Reset", IDRESET, "button", BS_PUSHBUTTON | BS_CENTER |
WS_CHILD | WS_VISIBLE | WS_TABSTOP, 228, 306, 50, 14
CONTROL "Fine Tuning", IDMANUAL, "button", BS_PUSHBUTTON |
BS_CENTER | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 228, 280, 50, 14
CONTROL "Turksat 1B", T1BHISTORY, "button", BS_PUSHBUTTON |
BS_CENTER | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 300, 276, 50, 14
CONTROL "Turksat 1C", T1CHISTORY, "button", BS_PUSHBUTTON |
BS_CENTER | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 360, 276, 50, 14
CONTROL "Characteristics", 606, "button", BS_GROUPBOX | WS_CHILD |
WS_VISIBLE | WS_GROUP, 292, 258, 128, 39
CONTROL "Functions", 607, "button", BS_GROUPBOX | WS_CHILD |
WS_VISIBLE | WS_GROUP, 220, 215, 64, 113
}

```

turkiye BITMAP "turkiye.bmp"

```

ABOUTBOX DIALOG 0, 0, 209, 107
STYLE DS_MODALFRAME | DS_3DLOOK | DS_CENTER | DS_CONTEXTHELP |
WS_POPUP | WS_VISIBLE | WS_CAPTION
CAPTION " About"
FONT 8, "MS Sans Serif"
{
CONTROL "OK", IDABOUTOK, "BUTTON", BS_PUSHBUTTON | BS_CENTER |

```



```

WS_CHILD | WS_VISIBLE | WS_TABSTOP, 120, 82, 54, 14
CONTROL "Frame1", -1, "static", SS_ETCHEDFRAME | WS_CHILD |
WS_VISIBLE, 4, 4, 200, 99
CONTROL "Dokuz Eylul Universitesi", -1, "static", SS_RIGHT | WS_CHILD
| WS_VISIBLE, 108, 26, 76, 9
CONTROL "Elektrik Elektronik Muhendisligi", -1, "static", SS_RIGHT
| WS_CHILD | WS_VISIBLE, 96, 39, 100, 9
CONTROL "Master Bitirme Projesi", -1, "static", SS_RIGHT | WS_CHILD
| WS_VISIBLE, 104, 52, 76, 8
CONTROL "Ebru Tasci", -1, "static", SS_LEFT | WS_CHILD | WS_VISIBLE,
128, 13, 40, 8
CONTROL "Turksat 1998", -1, "static", SS_LEFT | WS_CHILD |
WS_VISIBLE, 124, 65, 52, 8
CONTROL "ebru", -1, "static", SS_BITMAP | SS_REALSIZEIMAGE |
WS_CHILD | WS_VISIBLE | WS_BORDER, 8, 8, 118, 144, WS_EX_CLIENTEDGE
}

```

Antenico ICON "antenico.ico"

```

T1BFOOTBOX DIALOG 0, 0, 344, 145
STYLE DS_MODALFRAME | DS_3DLOOK | DS_CENTER | DS_CONTEXTHELP |
WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "Turksat 1B Foot Print"
FONT 8, "MS Sans Serif"
{
CONTROL "OK", IDOK, "BUTTON", BS_PUSHBUTTON | BS_CENTER | WS_CHILD |
WS_VISIBLE | WS_TABSTOP, 148, 125, 50, 14
CONTROL "1bfoot", -1, "static", SS_BITMAP | SS_REALSIZEIMAGE |
WS_CHILD | WS_VISIBLE | WS_BORDER, 4, 4, 500, 174, WS_EX_CLIENTEDGE
}

```

```

T1CFOOTBOX DIALOG 0, 0, 344, 145
STYLE DS_MODALFRAME | DS_3DLOOK | DS_CENTER | DS_CONTEXTHELP |
WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "Turksat 1C Foot Print"
FONT 8, "MS Sans Serif"
{
CONTROL "OK", IDOK, "BUTTON", BS_PUSHBUTTON | BS_CENTER | WS_CHILD |
WS_VISIBLE | WS_TABSTOP, 144, 125, 50, 14
CONTROL "1cfoot", -1, "static", SS_BITMAP | SS_REALSIZEIMAGE |
WS_CHILD | WS_VISIBLE | WS_BORDER, 4, 4, 500, 176, WS_EX_CLIENTEDGE
}

```

```

MANUALBOX DIALOG 0, 0, 287, 117
STYLE DS_MODALFRAME | DS_3DLOOK | DS_CENTER | DS_CONTEXTHELP |
WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "Antenna Manual Adjustment"
FONT 8, "MS Sans Serif"
{
CONTROL "OK", IDOK, "BUTTON", BS_PUSHBUTTON | BS_CENTER | WS_CHILD |
WS_VISIBLE | WS_TABSTOP, 120, 95, 50, 14
CONTROL "Azimuth", 701, "button", BS_GROUPBOX | WS_CHILD |
WS_VISIBLE | WS_GROUP, 12, 13, 128, 70
CONTROL "Elevation", 702, "button", BS_GROUPBOX | WS_CHILD |
WS_VISIBLE | WS_GROUP, 148, 13, 128, 70
CONTROL "CCW", AZCCW, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD
| WS_VISIBLE | WS_TABSTOP, 20, 52, 50, 14
CONTROL "CW", AZCW, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD |
WS_VISIBLE | WS_TABSTOP, 80, 52, 50, 14
CONTROL "CCW", ELCCW, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD
| WS_VISIBLE | WS_TABSTOP, 156, 52, 50, 14
CONTROL "CW", ELCW, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD |
WS_VISIBLE | WS_TABSTOP, 216, 52, 50, 14
CONTROL "0.018 degree for one step", -1, "static", SS_LEFT |
WS_CHILD | WS_VISIBLE, 20, 30, 96, 8
}

```

```
CONTROL "0.9 degree for one step", -1, "static", SS_LEFT | WS_CHILD  
| WS_VISIBLE, 160, 30, 96, 8  
}
```

```
lbfoot BITMAP "lbfoot.bmp"
```

```
lcfoot BITMAP "lcfoot.bmp"
```

```
ebru BITMAP "ebru.bmp"
```

```
T1CHISTORYBOX DIALOG 0, 1, 264, 178  
STYLE DS_MODALFRAME | DS_3DLOOK | DS_CENTER | DS_CONTEXTHELP |  
WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU  
CAPTION "Turksat 1C Characteristics And Performance"  
FONT 8, "MS Sans Serif"  
{  
CONTROL "OK", IDOK, "BUTTON", BS_PUSHBUTTON | BS_CENTER | WS_CHILD |  
WS_VISIBLE | WS_TABSTOP, 108, 159, 50, 14  
CONTROL "ch1c", -1, "static", SS_BITMAP | SS_REALSIZEIMAGE |  
WS_CHILD | WS_VISIBLE | WS_BORDER, 4, 4, 20, 20, WS_EX_CLIENTEDGE  
}
```

```
T1B HISTORYBOX DIALOG 0, 0, 264, 179  
STYLE DS_MODALFRAME | DS_3DLOOK | DS_CENTER | DS_CONTEXTHELP |  
WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU  
CAPTION "Turksat 1B Characteristics And Performance"  
FONT 8, "MS Sans Serif"  
{  
CONTROL "OK", IDOK, "BUTTON", BS_PUSHBUTTON | BS_CENTER | WS_CHILD |  
WS_VISIBLE | WS_TABSTOP, 104, 159, 50, 14  
CONTROL "ch1b", -1, "static", SS_BITMAP | SS_REALSIZEIMAGE |  
WS_CHILD | WS_VISIBLE | WS_BORDER, 4, 4, 18, 15, WS_EX_CLIENTEDGE  
}
```

```
ch1b BITMAP "ch1b.bmp"
```

```
ch1c BITMAP "ch1c.bmp"
```

Antenna.h File

```
#define STRICT
#include <windows.h>
#pragma hdrstop
#include <stdio.h>
#include <stdlib.h>
#include "iohw.h"

#include <conio.h>
#include <string.h>
#include <dos.h>

#define MYDLGBOX      100
#define ABOUTBOX      101
#define T1BFOOTBOX    117
#define T1CFOOTBOX    118
#define IDROTATE      102
#define IDOK           1
#define IDABOUT      10
#define IDRESET       150
#define IDABOUTOK    11
#define IC_AZ_EL_SATELLITE_T1B      120
#define IC_AZ_EL_SATELLITE_T1C      121
#define E_STATION_LAT   131
#define E_STATION_LONG  132
#define SAT_LATITUDE    103
#define SAT_LONGITUDE   104
#define ANTENNA_ELEVATION      105
#define ANTENNA_AZIMUTH       106
#define ID_1B      107
#define ID_1C      108
#define IDMANUAL    151
#define MANUALBOX   152
#define AZCCW       153
#define AZCW         154
#define ELCCW       155
#define ELCW        156
#define T1BHISTORYBOX      160
#define T1CHISTORYBOX     161
#define T1BHISTORY        162
#define T1CHISTORY        163

typedef struct tagAZELD{
    double FILE_ELEVATION_D;
    double FILE_AZIMUTH_D;
    int FILE_DAT1_I;
    int FILE_DAT2_I;
}AZELD;

typedef struct tagAZELDROT{
    double ELEVATION_ROT_D;
    double AZIMUTH_ROT_D;
}AZELDROT;

typedef struct tagMAPCOOR{
    char latitude[20];
    char longitude[20];
}MAPCOOR;

extern char *calculate_elevation(char *,char *,char *,char *);
extern char *calculate_azimuth(char *, char *,char *,char *);
extern bool Draw_Map(HWND , HDC , LPARAM , HBITMAP);
extern MAPCOOR Get_Map_Coor();
extern char *string_with_dec_point(double source);
```

```
int WriteData(double,double,int,int);
extern AZELDROT Get AZELDROT();
extern int ReadData(double *,double *,int *,int *);
void autorotate(void);
void gecikme (void);
void gecikme1(void);
struct projedata {
    int dat1;
    int dat2;
    double oldazi;
    double oldele;
    double newazi;
    double newele;
};

int soladon(int,int);
int soladon1(int,int);
int sagadon(int,int);
int sagadon1(int,int);
```



dtostring.cpp File

```
#include <string.h>
#include <stdlib.h>

char *string_with_dec_point(double source)
{
    char DESTINATION[20], DESTINATION1[20];
    int dec, sign, ndig=3, i;
    int k;
    strcpy(DESTINATION, fcvt(source, ndig, &dec, &sign));

    i=0;
    while (DESTINATION[i] != NULL)
        i++;

    if (source < 0)
    {
        strcpy(DESTINATION1, DESTINATION);
        for(k=0; k<=i; k++)
        {
            if(k<=3)
                DESTINATION[i-k+2]=DESTINATION1[i-k];

            if(k==3)
                DESTINATION[i-k+1]='.';

            if(k>3)
                DESTINATION[i-k+1]=DESTINATION1[i-k];

            if(k==i)
                DESTINATION[0]='-';
        }
    }
    else
    {
        DESTINATION[i+1]=NULL;
        DESTINATION[i]=DESTINATION[i-1];
        DESTINATION[i-1]=DESTINATION[i-2];
        DESTINATION[i-2]=DESTINATION[i-3];
        DESTINATION[i-3]='.';
    }

    return (DESTINATION);
}
```

Antenf.cpp File

```
#include "antenna.h"
```

```
FILE *fin;
```

```
// returns next token as string from fin file
```

```
int GetNextToken(char *token)
```

```
{
```

```
int idx = 0;
```

```
char ch;
```

```
while (!feof(fin))
```

```
{
```

```
    ch = fgetc(fin);
```

```
    if ((ch != 10) && (ch != 13) && (ch != ','))
```

```
    {
```

```
        token[idx] = ch;
```

```
        idx++;
```

```
    }
```

```
    else break;
```

```
}
```

```
token[idx] = 0; // null terminated string returns
```

```
return idx;
```

```
}
```

```
int ReadData(double *ffaz,double *ffel,int *ffdat1,int *ffdat2)
```

```
{
```

```
char mybuf[255];
```

```
    if ((fin = fopen("ANTEN.TXT", "rt")) != NULL)
```

```
    {
```

```
        if (GetNextToken(&mybuf) > 0)
```

```
            *ffaz=strtod(mybuf,NULL);
```

```
            else *ffaz = 0;
```

```
        if (GetNextToken(&mybuf) > 0)
```

```
            *ffel=strtod(mybuf,NULL);
```

```
            else *ffel = 0;
```

```
        if (GetNextToken(&mybuf) > 0)
```

```
            *ffdat1=atoi(&mybuf);
```

```
            else *ffdat1= 0;
```

```
        if (GetNextToken(&mybuf) > 0)
```

```
            *ffdat2=atoi(&mybuf);
```

```
            else *ffdat2= 0;
```

```
        fclose(fin);
```

```
    }
```

```
    else
```

```
    {
```

```
        WriteData(0,0,1,1);
```

```
    }
```

```
return 0;
```

```
}
```

```
int WriteData(double faz,double fel,int fd1,int fd2)
```

```
{
```

```
char wmybufa[20];
```

```
char wmybufe[20];
```

```
char wchdat1[4];
```

```
char wchdat2[4];
```

```
strcpy(wmybufa,string_with_dec_point(faz));
```

```
strcpy(wmybufe,string_with_dec_point(fel));
```

```
itoa(fd1,wchdat1,10);
```

```

itoa(fd2,wchdat2,10);

if ((fin = fopen("ANTEN.TXT", "w+t")) != NULL)
{
    int a=0;

    while (wmybufa[a] != NULL)
    {
        fputc(wmybufa[a], fin);
        a++;
    }
    fputc(',',fin);
    a = 0;
    while (wmybufe[a] != NULL)
    {
        fputc(wmybufe[a], fin);
        a++;
    }
    fputc(',',fin);
    a = 0;
    while (wchdat1[a] != NULL)
    {
        fputc(wchdat1[a], fin);
        a++;
    }
    fputc(',',fin);
    a = 0;
    while (wchdat2[a] != NULL)
    {
        fputc(wchdat2[a], fin);
        a++;
    }

    fputc(10,fin);
    fputc(13,fin);

    fclose(fin);
}
return 0;
}

```



```

while(i<puls){
    if(i<puls){
        HOut(0x378,0x08);
        gecikme();
        ++i;
        z=8;
    }
    if(i<puls){
        HOut(0x378,0x01);
        gecikme();
        ++i;
        z=1;
    }
    if(i<puls){
        HOut(0x378,0x02);
        gecikme();
        ++i;
        z=2;
    }
    if(i<puls){
        HOut(0x378,0x04);
        gecikme();
        ++i;
        z=4;
    }
}
}
if (sayi==8){
    while(i<puls){
        if(i<puls){
            HOut(0x378,0x01);
            gecikme();
            ++i;
            z=1;
        }
        if(i<puls){
            HOut(0x378,0x02);
            gecikme();
            ++i;
            z=2;
        }
        if(i<puls){
            HOut(0x378,0x04);
            gecikme();
            ++i;
            z=4;
        }
        if(i<puls){
            HOut(0x378,0x08);
            gecikme();
            ++i;
            z=8;
        }
    }
}
return(z);
}

int soladon(int puls,int sayi)
{
    int i=0;
    if (sayi==1){
        while(i<puls){
            if (i<puls){
                HOut(0x378,0x08);
                gecikme();
            }
        }
    }
}

```

```

        ++i;
        z=8;
    }
    if(i<puls){
        HOut(0x378,0x04);
        gecikme();
        ++i;
        z=4;
    }
    if(i<puls){
        HOut(0x378,0x02);
        gecikme();
        ++i;
        z=2;
    }
    if(i<puls){
        HOut(0x378,0x01);
        gecikme();
        ++i;
        z=1;
    }
}
if(sayi==2){
    while(i<puls){
        if(i<puls){
            HOut(0x378,0x01);
            gecikme();
            ++i;
            z=1;
        }
        if(i<puls){
            HOut(0x378,0x08);
            gecikme();
            ++i;
            z=8;
        }
        if(i<puls){
            HOut(0x378,0x04);
            gecikme();
            ++i;
            z=4;
        }
        if(i<puls){
            HOut(0x378,0x02);
            gecikme();
            ++i;
            z=2;
        }
    }
}
if(sayi==4){
    while(i<puls){
        if(i<puls){
            HOut(0x378,0x02);
            gecikme();
            ++i;
            z=2;
        }
        if(i<puls){
            HOut(0x378,0x01);
            gecikme();
            ++i;
            z=1;
        }
        if(i<puls){

```

```

        HOut(0x378,0x08);
        gecikme();
        ++i;
        z=8;
    }
    if(i<puls){
        HOut(0x378,0x04);
        gecikme();
        ++i;
        z=4;
    }
}
}
if(sayi==8){
    while(i<puls){
        if(i<puls){
            HOut(0x378,0x04);
            gecikme();
            ++i;
            z=4;
        }
        if(i<puls){
            HOut(0x378,0x02);
            gecikme();
            ++i;
            z=2;
        }
        if(i<puls){
            HOut(0x378,0x01);
            gecikme();
            ++i;
            z=1;
        }
        if(i<puls){
            HOut(0x378,0x08);
            gecikme();
            ++i;
            z=8;
        }
    }
}
return(z);
}

```

```

int soladon1 (int puls,int sayi)
{
    int i=0;
    if (sayi==1){
        while(i<puls){
            if(i<puls){
                HOut(0x378,0x90);
                gecikme1();
                ++i;
                z=9;
            }
            if(i<puls){
                HOut(0x378,0x80);
                gecikme1();
                ++i;
                z=8;
            }
            if(i<puls){
                HOut(0x378,0xC0);
                gecikme1();
                ++i;
                z=12;
            }
        }
    }
}

```

```

    }
    if(i<puls){
        HOut(0x378,0x40);
        gecikmel();
        ++i;
        z=4;
    }
    if(i<puls){
        HOut(0x378,0x60);
        gecikmel();
        ++i;
        z=6;
    }
    if(i<puls){
        HOut(0x378,0x20);
        gecikmel();
        ++i;
        z=2;
    }
    if(i<puls){
        HOut(0x378,0x30);
        gecikmel();
        ++i;
        z=3;
    }
    if(i<puls){
        HOut(0x378,0x10);
        gecikmel();
        ++i;
        z=1;
    }
}
if(sayi==9){
    while(i<puls){
        if(i<puls){
            HOut(0x378,0x80);
            gecikmel();
            ++i;
            z=8;
        }
        if(i<puls){
            HOut(0x378,0xC0);
            gecikmel();
            ++i;
            z=12;
        }
        if(i<puls){
            HOut(0x378,0x40);
            gecikmel();
            ++i;
            z=4;
        }
        if(i<puls){
            HOut(0x378,0x60);
            gecikmel();
            ++i;
            z=6;
        }
        if(i<puls){
            HOut(0x378,0x20);
            gecikmel();
            ++i;
            z=2;
        }
        if(i<puls){

```

```

        HOut(0x378,0x30);
        gecikmel();
        ++i;
        z=3;
    }
    if(i<puls){
        HOut(0x378,0x10);
        gecikmel();
        ++i;
        z=1;
    }
    if(i<puls){
        HOut(0x378,0x90);
        gecikmel();
        ++i;
        z=9;
    }
}
}
if(sayi==8){
    while(i<puls){
        if(i<puls){
            HOut(0x378,0xC0);
            gecikmel();
            ++i;
            z=12;
        }
        if(i<puls){
            HOut(0x378,0x40);
            gecikmel();
            ++i;
            z=4;
        }
        if(i<puls){
            HOut(0x378,0x60);
            gecikmel();
            ++i;
            z=6;
        }
        if(i<puls){
            HOut(0x378,0x20);
            gecikmel();
            ++i;
            z=2;
        }
        if(i<puls){
            HOut(0x378,0x30);
            gecikmel();
            ++i;
            z=3;
        }
        if(i<puls){
            HOut(0x378,0x10);
            gecikmel();
            ++i;
            z=1;
        }
        if(i<puls){
            HOut(0x378,0x90);
            gecikmel();
            ++i;
            z=9;
        }
        if(i<puls){
            HOut(0x378,0x80);
            gecikmel();

```

```

        ++i;
        z=8;
    }
}
if(sayi==12){
    while(i<puls){
        if(i<puls){
            HOut(0x378,0x40);
            gecikmel();
            ++i;
            z=4;
        }
        if(i<puls){
            HOut(0x378,0x60);
            gecikmel();
            ++i;
            z=6;
        }
        if(i<puls){
            HOut(0x378,0x20);
            gecikmel();
            ++i;
            z=2;
        }
        if(i<puls){
            HOut(0x378,0x30);
            gecikmel();
            ++i;
            z=3;
        }
        if(i<puls){
            HOut(0x378,0x10);
            gecikmel();
            ++i;
            z=1;
        }
        if(i<puls){
            HOut(0x378,0x90);
            gecikmel();
            ++i;
            z=9;
        }
        if(i<puls){
            HOut(0x378,0x80);
            gecikmel();
            ++i;
            z=8;
        }
        if(i<puls){
            HOut(0x378,0xC0);
            gecikmel();
            ++i;
            z=12;
        }
    }
}
if(sayi==4){
    while(i<puls){
        if(i<puls){
            HOut(0x378,0x60);
            gecikmel();
            ++i;
            z=6;
        }
        if(i<puls){

```

```

        HOut(0x378,0x20);
        gecikmel();
        ++i;
        z=2;
    }
    if(i<puls){
        HOut(0x378,0x30);
        gecikmel();
        ++i;
        z=3;
    }
    if(i<puls){
        HOut(0x378,0x10);
        gecikmel();
        ++i;
        z=1;
    }
    if(i<puls){
        HOut(0x378,0x90);
        gecikmel();
        ++i;
        z=9;
    }
    if(i<puls){
        HOut(0x378,0x80);
        gecikmel();
        ++i;
        z=8;
    }
    if(i<puls){
        HOut(0x378,0xC0);
        gecikmel();
        ++i;
        z=12;
    }
    if(i<puls){
        HOut(0x378,0x40);
        gecikmel();
        ++i;
        z=4;
    }
}

if(sayi==6){
    while(i<puls){
        if(i<puls){
            HOut(0x378,0x20);
            gecikmel();
            ++i;
            z=2;
        }
        if(i<puls){
            HOut(0x378,0x30);
            gecikmel();
            ++i;
            z=3;
        }
        if(i<puls){
            HOut(0x378,0x10);
            gecikmel();
            ++i;
            z=1;
        }
        if(i<puls){
            HOut(0x378,0x90);
            gecikmel();
        }
    }
}

```

```

        ++i;
        z=9;
    }
    if(i<puls){
        HOut(0x378,0x80);
        gecikme1();
        ++i;
        z=8;
    }
    if(i<puls){
        HOut(0x378,0xC0);
        ++i;
        z=12;
    }
    if(i<puls){
        HOut(0x378,0x40);
        ++i;
        z=4;
    }
    if(i<puls){
        HOut(0x378,0x60);
        ++i;
        z=6;
    }
}
}
if(sayi==2){
    while(i<puls){
        if(i<puls){
            HOut(0x378,0x30);
            ++i;
            z=3;
        }
        if(i<puls){
            HOut(0x378,0x10);
            ++i;
            z=1;
        }
        if(i<puls){
            HOut(0x378,0x90);
            ++i;
            z=9;
        }
        if(i<puls){
            HOut(0x378,0x80);
            ++i;
            z=8;
        }
        if(i<puls){
            HOut(0x378,0xC0);
            ++i;
            z=12;
        }
        if(i<puls){
            HOut(0x378,0x40);
            ++i;
            z=4;
        }
        if(i<puls){
            HOut(0x378,0x60);
            ++i;
            z=6;
        }
        if(i<puls){
            HOut(0x378,0x20);
            ++i;

```



```

        z=2;
    }
}
if(sayi==3){
    while(i<puls){
        if(i<puls){
            HOut(0x378,0x10);
            gecikmel();
            ++i;
            z=1;
        }
        if(i<puls){
            HOut(0x378,0x90);
            gecikmel();
            ++i;
            z=9;
        }
        if(i<puls){
            HOut(0x378,0x80);
            gecikmel();
            ++i;
            z=8;
        }
        if(i<puls){
            HOut(0x378,0xC0);
            gecikmel();
            ++i;
            z=12;
        }
        if(i<puls){
            HOut(0x378,0x40);
            gecikmel();
            ++i;
            z=4;
        }
        if(i<puls){
            HOut(0x378,0x60);
            gecikmel();
            ++i;
            z=6;
        }
        if(i<puls){
            HOut(0x378,0x20);
            gecikmel();
            ++i;
            z=2;
        }
        if(i<puls){
            HOut(0x378,0x30);
            ++i;
            z=3;
        }
    }
}
return(z);
}

```

```

int sagadon1 (int puls,int sayi)
{
    int i=0;
    if(sayi==1){
        while(i<puls){
            if(i<puls){
                HOut(0x378,0x30);
                gecikmel();
            }
        }
    }
}

```

```

        ++i;
        z=3;
    }
    if(i<puls){
        HOut(0x378,0x20);
        gecikmel();
        ++i;
        z=2;
    }
    if(i<puls){
        HOut(0x378,0x60);
        gecikmel();
        ++i;
        z=6;
    }
    if(i<puls){
        HOut(0x378,0x40);
        gecikmel();
        ++i;
        z=4;
    }
    if(i<puls){
        HOut(0x378,0xC0);
        gecikmel();
        ++i;
        z=12;
    }
    if(i<puls){
        HOut(0x378,0x80);
        gecikmel();
        ++i;
        z=8;
    }
    if(i<puls){
        HOut(0x378,0x90);
        gecikmel();
        ++i;
        z=9;
    }
    if(i<puls){
        HOut(0x378,0x10);
        gecikmel();
        ++i;
        z=1;
    }
}
}
if(sayi==3){
    while(i<puls){
        if(i<puls){
            HOut(0x378,0x20);
            gecikmel();
            ++i;
            z=2;
        }
        if(i<puls){
            HOut(0x378,0x60);
            gecikmel();
            ++i;
            z=6;
        }
        if(i<puls){
            HOut(0x378,0x40);
            gecikmel();
            ++i;
            z=4;
        }
    }
}

```

```

    }
    if(i<puls){
        HOut(0x378,0xC0);
        gecikmel();
        ++i;
        z=12;
    }
    if(i<puls){
        HOut(0x378,0x80);
        gecikmel();
        ++i;
        z=8;
    }
    if(i<puls){
        HOut(0x378,0x90);
        gecikmel();
        ++i;
        z=9;
    }
    if(i<puls){
        HOut(0x378,0x10);
        gecikmel();
        ++i;
        z=1;
    }
    if(i<puls){
        HOut(0x378,0x30);
        gecikmel();
        ++i;
        z=3;
    }
}
}
if(sayi==2){
    while(i<puls){
        if(i<puls){
            HOut(0x378,0x60);
            gecikmel();
            ++i;
            z=6;
        }
        if(i<puls){
            HOut(0x378,0x40);
            gecikmel();
            ++i;
            z=4;
        }
        if(i<puls){
            HOut(0x378,0xC0);
            gecikmel();
            ++i;
            z=12;
        }
        if(i<puls){
            HOut(0x378,0x80);
            gecikmel();
            ++i;
            z=8;
        }
        if(i<puls){
            HOut(0x378,0x90);
            gecikmel();
            ++i;
            z=9;
        }
        if(i<puls){

```

```

        HOut(0x378,0x10);
        gecikmel();
        ++i;
z=1;
    }
    if(i<puls){
        HOut(0x378,0x30);
        gecikmel();
        ++i;
        z=3;
    }
    if(i<puls){
        HOut(0x378,0x20);
        gecikmel();
        ++i;
        z=2;
    }
}
if(sayi==6){
    while(i<puls){
        if(i<puls){
            HOut(0x378,0x40);
            gecikmel();
            ++i;
            z=4;
        }
        if(i<puls){
            HOut(0x378,0xC0);
            gecikmel();
            ++i;
            z=12;
        }
        if(i<puls){
            HOut(0x378,0x80);
            gecikmel();
            ++i;
            z=8;
        }
        if(i<puls){
            HOut(0x378,0x90);
            gecikmel();
            ++i;
            z=9;
        }
        if(i<puls){
            HOut(0x378,0x10);
            gecikmel();
            ++i;
            z=1;
        }
        if(i<puls){
            HOut(0x378,0x30);
            gecikmel();
            ++i;
            z=3;
        }
        if(i<puls){
            HOut(0x378,0x20);
            gecikmel();
            ++i;
            z=2;
        }
        if(i<puls){
            HOut(0x378,0x60);
            gecikmel();

```

```

        ++i;
        z=6;
    }
}
if(sayi==4){
    while(i<puls){
        if(i<puls){
            HOut(0x378,0xC0);
            gecikmel();
            ++i;
            z=12;
        }
        if(i<puls){
            HOut(0x378,0x80);
            gecikmel();
            ++i;
            z=8;
        }
        if(i<puls){
            HOut(0x378,0x90);
            gecikmel();
            ++i;
            z=9;
        }
        if(i<puls){
            HOut(0x378,0x10);
            gecikmel();
            ++i;
            z=1;
        }
        if(i<puls){
            HOut(0x378,0x30);
            gecikmel();
            ++i;
            z=3;
        }
        if(i<puls){
            HOut(0x378,0x20);
            gecikmel();
            ++i;
            z=2;
        }
        if(i<puls){
            HOut(0x378,0x60);
            gecikmel();
            ++i;
            z=6;
        }
        if(i<puls){
            HOut(0x378,0x40);
            gecikmel();
            ++i;
            z=4;
        }
    }
}
if(sayi==12){
    while(i<puls){
        if(i<puls){
            HOut(0x378,0x80);
            gecikmel();
            ++i;
            z=8;
        }
        if(i<puls){

```

```

        HOut(0x378,0x90);
        gecikmel();
        ++i;
        z=9;
    }
    if(i<puls){
        HOut(0x378,0x10);
        gecikmel();
        ++i;
        z=1;
    }
    if(i<puls){
        HOut(0x378,0x30);
        gecikmel();
        ++i;
        z=3;
    }
    if(i<puls){
        HOut(0x378,0x20);
        gecikmel();
        ++i;
        z=2;
    }
    if(i<puls){
        HOut(0x378,0x60);
        gecikmel();
        ++i;
        z=6;
    }
    if(i<puls){
        HOut(0x378,0x40);
        gecikmel();
        ++i;
        z=4;
    }
    if(i<puls){
        HOut(0x378,0xC0);
        gecikmel();
        ++i;
        z=12;
    }
}

if(sayi==8){
    while(i<puls){
        if(i<puls){
            HOut(0x378,0x90);
            gecikmel();
            ++i;
            z=9;
        }
        if(i<puls){
            HOut(0x378,0x10);
            gecikmel();
            ++i;
            z=1;
        }
        if(i<puls){
            HOut(0x378,0x30);
            gecikmel();
            ++i;
            z=3;
        }
        if(i<puls){
            HOut(0x378,0x20);
            gecikmel();
        }
    }
}

```

```

        ++i;
        z=2;
    }
    if(i<puls){
        HOut(0x378,0x60);
        gecikmel();
        ++i;
        z=6;
    }
    if(i<puls){
        HOut(0x378,0x40);
        gecikmel();
        ++i;
        z=4;
    }
    if(i<puls){
        HOut(0x378,0xC0);
        gecikmel();
        ++i;
        z=12;
    }
    if(i<puls){
        HOut(0x378,0x80);
        gecikmel();
        ++i;
        z=8;
    }
}
if(sayi==9){
    while(i<puls){
        if(i<puls){
            HOut(0x378,0x10);
            gecikmel();
            ++i;
            z=1;
        }
        if(i<puls){
            HOut(0x378,0x30);
            gecikmel();
            ++i;
            z=3;
        }
        if(i<puls){
            HOut(0x378,0x20);
            gecikmel();
            ++i;
            z=2;
        }
        if(i<puls){
            HOut(0x378,0x60);
            gecikmel();
            ++i;
            z=6;
        }
        if(i<puls){
            HOut(0x378,0x40);
            gecikmel();
            ++i;
            z=4;
        }
        if(i<puls){
            HOut(0x378,0xC0);
            gecikmel();
            ++i;
            z=12;
        }
    }
}

```

```
        }
        if(i<puls){
            HOut(0x378,0x80);
            gecikme1();
            ++i;
            z=8;
        }
        if(i<puls){
            HOut(0x378,0x90);
            gecikme1();
            ++i;
            z=9;
        }
    }
    return(z);
}
```

```
void gecikme(void)
{
    Sleep(10);
}
```

```
void gecikme1(void)
{
    Sleep(100);
}
```


maps.cpp File

```
#include <windows.h>

float map_coor_x, map_coor_y;

typedef struct tagMAPCOOR{
    char latitude[20];
    char longitude[20];
}MAPCOOR;

extern char *string_with_dec_point(double source);

BITMAP bm;
POINT pt,ptsc;

bool Draw_Map(HWND hDlg1,HDC hDC, LPARAM lParam1, HBITMAP hMap)
{
    hDC=GetDC(hDlg1);
    GetObject (hMap, sizeof(BITMAP), (LPSTR) &bm);
    pt.x=bm.bmWidth;
    pt.y=bm.bmHeight;
    DPTOLP(hDC, &pt, 1);

    ptsc.x=LOWORD(lParam1);
    ptsc.y=HIWORD(lParam1);

    if ((ptsc.x <= pt.x) && (ptsc.y <= pt.y))
    {
        ReleaseDC(hDlg1, hDC);
        return(TRUE);
    }
    return(FALSE);
}

//MAPCOOR Get_Map_Coor(HWND hDlg2, LPARAM lParam2)
MAPCOOR Get_Map_Coor()
{
    MAPCOOR map_coor1;
    float map_upper_left_x, map_upper_left_y,
        map_lower_right_x, map_lower_right_y,
        x_map_aspect_ratio, y_map_aspect_ratio;

    map_upper_left_x=25.55;
    map_upper_left_y=42.086956;
    map_lower_right_x=45.11;
    map_lower_right_y=34.7826;
    //636
    //308

    x_map_aspect_ratio=(map_lower_right_x - map_upper_left_x) / (pt.x);
    y_map_aspect_ratio=(map_upper_left_y - map_lower_right_y) / (pt.y);

    map_coor_x=map_upper_left_x + ((ptsc.x) * x_map_aspect_ratio);
    map_coor_y=map_upper_left_y - ((ptsc.y) * y_map_aspect_ratio);

    strcpy(map_coor1.longitude, string_with_dec_point(map_coor_x));
    strcpy(map_coor1.latitude, string_with_dec_point(map_coor_y));
    return map_coor1;
}
```

azimelev.cpp

```
#include <stdlib.h>
#include <string.h>
#include <math.h>
```

```
double ED,AD;
```

```
typedef struct tagAZELDROT{
    double ELEVATION_ROT_D;
    double AZIMUTH_ROT_D;
}AZELDROT;
```

```
char *string_with_dec_point(double source)
{
    char DESTINATION[20];
    int dec,sign,ndig=3,i;

    strcpy(DESTINATION,fcvt(source,ndig,&dec,&sign));

    i=0;
    while (DESTINATION[i]!=NULL)
        i++;

    DESTINATION[i+1]=NULL;
    DESTINATION[i]=DESTINATION[i-1];
    DESTINATION[i-1]=DESTINATION[i-2];
    DESTINATION[i-2]=DESTINATION[i-3];
    DESTINATION[i-3]='.';
    return (DESTINATION);
}
```

```
AZELDROT azeldrot2;
```

```
char *calculate_elevation(char *E_S_LAT_CHAR1,char *E_S_LONG_CHAR1,
    char *SAT_LAT_CHAR1,char *SAT_LONG_CHAR1)
{
    char ELEVATION_CHAR1[20],*endptr1,*endptr2,*endptr3,*endptr4;
    double E_S_LAT_DOUBLE,E_S_LONG_DOUBLE,ELEVATION_DOUBLE,LONG_DIF;
    double SAT_LAT_DOUBLE,SAT_LONG_DOUBLE;
//    double SAT_LONG=31.232,SAT_LAT=0; /*for T1B*/
//    double SAT_LONG=41.994,SAT_LAT=0; /*for T1C*/
    double COS_MULT;

    E_S_LAT_DOUBLE=strtod(E_S_LAT_CHAR1,&endptr1);
    E_S_LONG_DOUBLE=strtod(E_S_LONG_CHAR1,&endptr2);
//    E_S_LAT_DOUBLE=39.639;
//    E_S_LONG_DOUBLE=32.80151;
    SAT_LAT_DOUBLE=strtod(SAT_LAT_CHAR1,&endptr3);
    SAT_LONG_DOUBLE=strtod(SAT_LONG_CHAR1,&endptr4);

    LONG_DIF=E_S_LONG_DOUBLE-SAT_LONG_DOUBLE;
    COS_MULT=cos(2.0 * M_PI * E_S_LAT_DOUBLE/360.0) *
cos(2.0*M_PI*LONG_DIF/360.0);

    ELEVATION_DOUBLE=360.0 * (atan((COS_MULT-0.151) / sqrt(1-
pow(COS_MULT,2))))/(2.0*M_PI);
    strcpy(ELEVATION_CHAR1,string_with_dec_point(ELEVATION_DOUBLE));
    ED = ELEVATION_DOUBLE;
    return (ELEVATION_CHAR1);
}
```

```
char *calculate_azimuth(char *E_S_LAT_CHAR2,char *E_S_LONG_CHAR2,
    char *SAT_LAT_CHAR2,char *SAT_LONG_CHAR2)
{
    char AZIMUTH_CHAR1[20],*endptr1,*endptr2,*endptr3,*endptr4;
    double E_S_LAT_DOUBLE,E_S_LONG_DOUBLE,AZIMUTH_DOUBLE,LONG_DIF;
```

```

    double SAT_LAT_DOUBLE, SAT_LONG_DOUBLE;
    //    double SAT_LONG=31.232, SAT_LAT=0; /*for T1B*/
    //    double SAT_LONG=41.994, SAT_LAT=0; /*for T1C*/

    E_S_LAT_DOUBLE=strtod(E_S_LAT_CHAR2, &endptr1);
    E_S_LONG_DOUBLE=strtod(E_S_LONG_CHAR2, &endptr2);
    //    E_S_LAT_DOUBLE=39.639;
    //    E_S_LONG_DOUBLE=32.80151;

    SAT_LAT_DOUBLE=strtod(SAT_LAT_CHAR2, &endptr3);
    SAT_LONG_DOUBLE=strtod(SAT_LONG_CHAR2, &endptr4);

    LONG_DIF=E_S_LONG_DOUBLE-SAT_LONG_DOUBLE;

    AZIMUTH_DOUBLE=180+(360.0 * (atan2(tan(2.0*M_PI*LONG_DIF/360),
sin(2*M_PI*E_S_LAT_DOUBLE/360))) / (2*M_PI));
    strcpy(AZIMUTH_CHAR1, string_with_dec_point(AZIMUTH_DOUBLE));
    AD = AZIMUTH_DOUBLE;
    return (AZIMUTH_CHAR1);
}

AZELDROT Get_AZELDROT()
{
    azeldrot2.ELEVATION_ROT_D = ED;
    azeldrot2.AZIMUTH_ROT_D = AD;
    return azeldrot2;
}

```