

Computer Aided Design of Pneumatic Control Circuits

**M.Sc. Thesis
in
Mechanical Engineering
University of Gaziantep**

**Supervisor
Prof. Dr. Sedat BAYSEÇ**

**by
M. Sinan GÜZELBEY
September 2005**

ABSTRACT

COMPUTER AIDED DESIGN OF PNEUMATIC CONTROL CIRCUITS

GÜZELBEY, M. Sinan

M.Sc. in Mechanical Eng.

Supervisor: Prof. Dr. Sedat BAYSEÇ

September 2005, 90 pages

Automation problems concerning the sequencing of successive events can most simply be obtained by appropriately designed pneumatic control circuits. These pneumatic systems are called Cascade systems. Although the design of these systems is well known and documented, doing it free hand may be quite difficult and laborious especially for complicated sequences. Contrary to computer or PLC control, solid logic is more reliable, easier to implement and maintain, and is cheaper. In this study, the design procedure, which is always given in verbal form, is converted into an algorithm first and a computer program written in Borland Delphi 7.0. Output of the computer program is in form of a circuit diagram comprising of ISO standard pneumatic symbols and color coded pipelines. The program is user friendly, easy to use and fast.

Key Words: Pneumatic circuits, Pneumatic control, Cascade systems.

ÖZET

PNEUMATİK KONTROL DEVRELERİNİN BİLGİSAYAR DESTEKLİ TASARIMI

GÜZELBEY, M. Sinan

M.Sc. Makina Mühendisliği

Danışman: Prof. Dr. Sedat BAYSEÇ

Eylül 2005, 90 sayfa

Birbirini takip eden olay zincirlerinin otomasyon problemi en basitçe, uygun şekilde tasarlanmış pnömatik kontrol devreleri kullanarak çözülür. Bu şekilde çalışan pnömatik sistemlere “Kaskad Sistemler” denmektedir. Bu tür sistemlerin tasarımı yaygın bir şekilde bilinmekte ve doküman haline getirilmiş olmakla birlikte özellikle karmaşık sıralandırma problemleri için uygulanması oldukça zor ve zaman alıcı olabilmektedir. Bilgisayar ve PLC kontrolüne göre bu katı mantık uygulaması daha güvenilir, uygulaması ve bakımı kolay ve daha ucuz olmaktadır. Bu çalışmada daima sözel olarak anlatılan tasarım süreci önce bir algoritmaya dönüştürülmüş, daha sonrada Borland Delphi 7.0 dilinde yazılmış bir program haline getirilmiştir. Bilgisayar programının çıktısı farklı renklerle kodlanmış hava hatları ile birbirine bağlanmış ISO standart pnömatik semboller ile gösterilmiş bir devre resmi halindedir. Program kullanıcıların kolay kullanabildiği, basit ve hızlı bir programdır.

Anahtar kelimeler: Pneumatik Devreler, Pneumatik Kontrol, Kaskad Sistemler.

ACKNOWLEDGMENTS

I would like to thank my supervisor, Prof. Dr. Sedat Bayseç for his help, valuable comments, and supervision.

My very special thanks are due to my family for their motivation, support and patience during the preparation of this thesis.

M. Sinan GÜZELBEY
September 2005
Gaziantep

TABLE OF CONTENTS

ABSTRACT.....	i
ÖZET	ii
ACKNOWLEDGMENTS	iii
TABLE OF CONTENTS	iv
LIST OF FIGURES	vi
LIST OF TABLES	vii
CHAPTER 1: INTRODUCTION	1
1.1. Structure of the Thesis	4
1.2. Literature Survey	6
1.3. Brief History of Pneumatic Systems	8
1.4. Principles of Pneumatic Systems	11
1.4.1. Pascal’s Law	11
1.4.2. Boyle’s Law	13
1.4.3. Charles’ Law	14
1.4.4. Characteristic Equation of a Gas.....	15
1.5. Compression and Expansion of Air	15
1.6. Solid Logic	19
CHAPTER 2: DEFINITION OF PNEUMATIC CIRCUITRY.....	20
2.1. Introduction	20
2.2. Pneumatic Components	21
2.2.1. Double-Acting Cylinders	21
2.2.2. Five Way Two Position Valves	23
2.2.3. Three Way Two Position Valves	25
2.3. Solution Techniques of Pneumatic Circuits Problems.....	26
2.3.1. Single Circuits	27
2.3.2. More Intricate Circuits	28
CHAPTER 3: PROGRAMMING “PNEUDESIGN”	34
3.1. Pneumatic Components of the “Pneudesign”	35

3.2.	Definition of the Program Application	38
3.2.1.	Control of the Correctness of the Sequence	40
3.2.2.	Grouping a Sequence	42
3.2.3.	Layout of the Pneumatic Circuit	45
3.2.4.	Method of Design of Pipeline Connection	46
CHAPTER 4:	EXAMPLES OF PNEUMATIC CIRCUITS	55
4.1.	Examples Containing Two Cylinders	58
4.2.	Examples Containing Three Cylinders	60
4.3.	An Example Containing Four Cylinders	62
4.1.	An Example Containing Five Cylinders	63
CHAPTER 5:	DISCUSSION AND CONCLUSIONS.....	64
5.1.	Discussion of Achievements	64
5.2.	Recommendations for Future Work ...	65
REFERENCES		67
APPENDIX A.	ILLUSTRATED PHOTOS OF PNEUMATIC CIRCUITS	70
APPENDIX B.	SOURCE CODES OF FORM1	74
APPENDIX C.	SOURCE CODES OF FORM2	78

LIST OF FIGURES

Figure 1.1. Piston being extended by a continuous supply of compressed air . . .	16
Figure 1.2. P-V diagram of a compression process. Expansion follows the same route in opposite sense.	17
Figure 2.1. Double-acting cylinder is operated by a 5/2 control valve	22
Figure 2.2. Double-acting cylinder	23
Figure 2.3. A spool valve with inlet and outlet ports	24
Figure 2.4. 5/2-way double control valve, pneumatically actuated, by boost chambers on both sides of the spool	24
Figure 2.5. 5/2 Pneumatic valve per ISO 5599-1.	24
Figure 2.6 Placement of 3/2 pilot valves with respect to the actuator.	25
Figure 2.7 3/2 valve with roller lever control	26
Figure 2.8 Cascade circuit for the sequence A+ B+ B- A-	33
Figure 3.1 The piston will be extended by pressure air.	35
Figure 3.2 Connection between the control valve and cylinder for an A movement.	36
Figure 3.3 Five port valves used to give a five-group cascade circuit	37
Figure 3.4 Flow-chart of Pnuedesign.exe.	39
Figure 3.5 The first form at the Pnuedesign	40
Figure 3.6 The stages of operations in the designing of a sequencing circuit. . . .	41
Figure 3.7 Layout of the pneumatic components on the work chart.	47
Figure 3.8 Pneumatic circuit of the sequence [A+B+B-C+A-C-]	54
Figure 4.1 Circuit of the sequence [A+A-B+B-]	58
Figure 4.2 Circuit of the sequence [A+B-B+A-].	59
Figure 4.3 Circuit of the sequence [A+B-C+A-C-B+].	60
Figure 4.4 Circuit of the sequence [A-B+C-B-A+C+].	61
Figure 4.5 Circuit of the sequence [A+C+D-C-D+A-B+B-]	62
Figure 4.6 Circuit of the sequence [A+C-B+C+E+A-D+D-E-B-]	63

LIST OF TABLES

Table 3.1. Codes of control valves at the program.	36
Table 3.2. Codes of Pilot valves at the program	38
Table 3.3 Codes of signal ports of cascade valves	51

CHAPTER 1

INTRODUCTION

In industry, we often require to transmit power from one location where it is generated to another location where it is consumed. Power transmission can be in various forms, as electric power, in form of an electric current flowing from a higher potential to lower via electrical conductors, mechanical power, in form of a rotating shaft delivering torque or a translating ram applying a force of actuation or in fluidic form via a pressurised fluid flowing through a pipe or hose. Each of these forms of power and power transmission has their own advantages and disadvantages over each other. Electric power for example can be transmitted to faraway distances with electric wiring whereas mechanical power can be transmitted to short distances with rigid or flexible shafts. Fluidic power can be transmitted to longer distances than that of a shaft length but not as much as an interconnected electric wiring network.

Fluidic power is loaded onto a bulk of fluid by increasing its internal energy. The energised fluid then is transmitted to the required location via pipes or flexible hoses, where power is extracted. Internal energy of a fluid bulk has two important components; one is the dynamic component arising from the speed of the fluid: any moving body will have a kinetic energy. Second one is the static component due to the pressure of the fluid. Power can be loaded onto a fluid bulk hence by either increasing its speed or increasing its pressure. The two techniques differ from each other considerably and hence do the technology involved. Machines which charge or discharge energy by imposing velocity variations are called “rotodynamic machines” and the ones which do energy exchange by changing pressures are called the “hydrostatic machines”.

Fluid can be of any type, technically and economically feasible. For example at a place where water is abundant, fluid best to use may be water. This is a large area of

application by which greater portion of our power requirements are met and is called the “water power systems”. Water is corrosive and is not a good lubricant. Mineral oils can serve as better power transmitting liquids. Systems using oil as power transmitting medium are called “Oil Hydraulic systems”. Although the theories beneath water and oil power systems are common, they embed completely different technologies and hence are regarded as different fields of specialisation. Air is perhaps the cheapest fluid to use. It has a very small density and so loading energy onto a bulk of air by dynamic means is generally not very feasible. Instead, air can be energised or de-energised effectively by changing its pressure. Systems using pressurised air as power transmission medium are called “Pneumatic systems”.

Pneumatic systems are important parts of the modern industrial world. They are widely used in applications where compressed air can readily be obtained and where horsepower requirements are not large. Pneumatic systems are very effective in situations where the motion should be limited by a mechanical stop, which is often called a “bang-bang” type of actuation. The design and development of all fluid power systems are very complex due to the nonlinear nature of the physical phenomena incorporated. Actually no direct design methods exist and each application is a separate problem in itself, requiring special theoretical and experimental work. Pneumatic systems are even more complicated due to the fact that the fluid used is extremely compressible. In hydraulic systems the medium is somewhat compressible as attested to by the “bulk modulus”, however, the magnitudes of compressibility encountered are much less than that of air. In addition, when the air is compressed the density changes significantly which even further complicates the analytical considerations. Due to the complexity of the models which realistically describe fluid power components and systems, designers have preferred to use only steady-state calculations, although in most problems which occur with pneumatic applications, become apparent in the dynamic state and not in the steady-state condition [1].

Energy is loaded onto air by compressing it. Compressors are mechanical devices incorporating variable volume closed chambers inside. Volume of the chambers

increase and decrease successively by the operating mechanisms. Variable volume chambers can be in form of pistons reciprocating inside cylinders, can be the gap between two successive threads of a pair of Lysholm screws or the space entrapped between an eccentrically rotating rotor and its blades. Similarly energy is extracted by allowing the air to act over a surface, hence generate a hydrostatic force and then allowing this surface to move, either in form of a rotation or a translation. These devices are called “pneumatic motors” or “actuators”. Generally the same device can be used as a compressor or as a motor. Between the two, there are hoses or pipes through which pressurised air will flow. Control of power transmitted can be done by either controlling the flowrate or the pressure, as fluidic power is the product of the two. The device used to control the power is a valve, which, in general, is in form of a variable size orifice, which acts as an obstacle to the flow of fluid. When orifice area is made zero, no air passes and hence no power is transmitted. When orifice area is made the largest, air can flow very freely and carry all the power it bears. The nature of the flow through an orifice is that not only the flowrate but also the pressures change while going through an orifice. So the pressure and flowrate of air against the control variable of the valve is a complicated and non-linear function [2].

Any machine can be actuated by pneumatic actuators. In general, a machine is a combination of mechanisms, each of which conduct a motion, which is a part of the complicated useful work the machine does. In pneumatic actuation, each mechanism is equipped with an actuator of its own, which in turn is activated by a control valve of its own. So, the operation of the machine and the coordination of the motions, individual pneumatic actuators must be controlled externally. External controller can be a computer or a PLC. The sequencing of the processes can also be done by using “Pneumatic logic”. Operation is in a sequence, each actuator activating the next operation at the end of its stroke. Apart from this simple sequencing, logic operations can be enriched by AND and OR gates, inverters and flip-flops. Pneumatic logic is more sturdy and robust against dust, dirt, vibration and electromagnetic emissions of running motors in a workshop. They are also much cheaper than a computer and its peripherals, or a PLC. Therefore, in many pneumatic applications, sequencing and coordination of the motions in a system or machine is done by Pneumatic Logic.

1.1. Structure of the Thesis:

Solving a sequencing problem is not an easy task in general. Processes occurring within the same duration must be actuated in parallel. Processes which are done at a later time must be put into an order or cascade. Design of the circuit for a cascaded operation is a complicated task, although well known and well understood. If the size of the problem becomes bigger, the circuit becomes more complicated. So, the help of a computer would be very beneficial. A program segment which will draw the circuit diagram after finishing the design in standard symbols will also be very helpful. The subjectmatter of this thesis is the definition of a cascaded process and generation of the circuit diagram accordingly by using a computer.

Chapter 1 starts with a literature survey on papers and books written on pneumatic control, extending from simple general knowledge articles to those containing complicated theories. The nature of air flow is examined, and reasoning of why most pneumatic problems are assumed static fluid problems. Principles governing the state of static fluids namely Pascal's, Boyle's and Charles' laws. A formulation for the work of compression or expansion is given. These actually do not contribute to the preparation of the resulting computer program, but are essential if the static fluid problem is to be converted into a dynamic form. Some well known advantages and disadvantages of the pneumatic logic and some interesting captions about the historical events in the development of pneumatics science are also amended to this chapter.

Chapter 2 provides the theory of designing cascade circuits. Firstly the equipment required in such circuits, namely double-acting cylinders, 5 way 2 position valves and 3 way 2 position valves are described. Then simple sequential tasks solvable by "Single Circuits" are explained. Then more intricate circuits which can not be solved by Single Circuits are defined. These tasks can only be done by "Cascade Circuits". Method of solution of solving a cascaded circuit problem is presented which can be found in many texts, some of which are given in the reference list. This method which can be found in papers as well as books is in verbal form and as it is difficult

to explain, many of them have preferred working on an example problems, while giving a definition of each step taken. In this chapter, the same technique is followed on the $[A+ B+ B- A-]$ sequence. This sequence is the simplest cascade circuit problem and hence extensively used by almost every document written on this subject. A computer printout of the solution circuit drawn by the computer program “*Pneudesign*” is presented at the end.

Chapter 3 describes how the technique, verbally given in the previous chapter is converted into a computer program, which is the subjectmatter of this thesis. The computer program, named as “*Pneudesign*” by its designer is an interactive, user friendly and simple to use program written in Borland Delphi 7.0. It has a data input and data output sections which provide the interface between the user and the computer. An algorithm is given how the problem is solved systematically. After the solution is complete, the resulting circuit is given out in form of a circuit diagram composed of the ISO standard drawings of all the components used and the necessary pipeline connections between them.

Chapter 4 comprises examples to demonstrate the capabilities of the “*Pneudesign*”. Totally six different examples are given at various levels of complexity. The sequences worked on are $[A+ A- B+ B-]$, $[A+ B- B+ A-]$, $[A+ B- C+ A- C- B+]$, $[A- B+ C- B- A+ C+]$, $[A+ C+ B- C- D+ A- B+ B-]$ and $[A+ C- B+ C+ E+ A- D+ D- E- B-]$. First two are of two actuators, next three are of three four and five actuators respectively. Program codes of each example sequence are also presented with the circuit diagram.

Chapter 5 gives a final discussion on the thesis, presents the proposals for the future work, some of which are presumably quite difficult to do.

The dissertation contains 5 chapters in total. As amendment, Appendices A,B and C are given. The circuit diagrams of cascaded circuits are in general complex and once a circuit comes up, it is almost impossible to visualise the pressure variations in the circuit and flow of events occurring just by following the pipelines. The examples

given in chapter 4 are quite difficult to follow indeed. Normally an experimentation is required to verify whether a solution circuit is operational or not. As part of this thesis, the circuits given in chapter 4 are patched on a Festo Pneumatic Designer Set and were all observed to operate properly. The witness photographs of the experimental set up are given in **Appendix A**.

Whether a computer program is to be amended to a dissertation or not is a common issue of argument among people doing computational work and has not been settled yet. Some programmers argue that the program is a tool for solving a certain, supposedly big problem and hence has a commercial value and do not present their programs explicitly. They present only algorithms and solution examples. Some even bug the programs and amended programs are of no use at all then. The programmer of *Pneudesign* at the writer of this thesis, believing the necessity of helping anyone who seeks his help, has put the complete program in **Appendix B**.

Pneudesign brings a solution to a sequencing problem in a fast manner. The term “Fastness” in this sense is relative. No work is done on measuring computation time, neither this is found necessary. The program does not contain any optimisation, trial and error type iterations or arithmetical operations like integration, differentiation, filtering etc. So, time required for a specific problem to solve is practically nil. At the end of the solution, the resulting circuit is presented in form of a technical drawing showing the ISO standard symbols of each component used and the piping between them in colour coding. Preparation of the drawing program had been laborious and required special attention. The program segment which does the circuit drawing is presented in **Appendix C**.

1.2. Literature Survey

In designing a pneumatic circuit, a real challenge is to enable designers to visualize the circuit during the design stage and answer “what if?” questions. The objective of this preliminary study was to develop an interface design alternative (icon, pull-down, and screen menus) for pneumatic circuit design. It was also an attempt to

determine the possibility of using it as a tool to help designers associate the menu systems with commands, and visualize circuit design improvements.

At The Department of Industrial Technology in University of Northern Iowa, Professor Ahmed Elsayy and Veekit O'Charoen have done a study that provides three main benefits. Firstly, a pictorial man-machine interface, commanded by the appropriate menus helps designers to associate these pictures with commands and to visualize circuit design improvement. Secondly it proved to be a tool enabling to preview pneumatic circuits and minimize rejected designs. Lastly it came up with the ability to create more prototypes in less time and at lower cost. In that study, users can expand and shape AutoCAD according to their need. It was assumed that the menu system is developed for the group of designers who have some knowledge of AutoCAD's (R13) user interface and pneumatic circuit design. That study was limited to two-dimensional pneumatic schematics. Furthermore, the symbols emphasize the function and methods of operation of components. The AutoCAD's user interface (icon, screen, and pull down menus) are preconfigured by Autodesk. To develop and analyze the user interface for pneumatic circuit designs, type-of-hierarchy diagrams are used to illustrate the task and subtasks of the menu systems [3,4].

A study entitled as "Modelling and simulation of pneumatic cylinders" for a physiotherapy robot by R.Richardson and A.R.Plummer in School of Mechanical Engineering, University of Leeds, UK was found helpful in pneumatics technology. That study examines the possibility of reducing seal friction in actuators. Low friction pneumatic cylinders are now being considered in applications for which only electric motors or hydraulics were previously considered suitable. One potential application of low friction pneumatics is robotic physiotherapy where the high power to weight ratio and low cost could be exploited. As part of an ongoing project to develop a pneumatic physiotherapy robot, that study presents an analysis of pneumatic cylinder characteristics that simplifies controller design. Using mathematical modelling and simulation, non-linearities of modern pneumatic systems have been investigated. The derived models give an excellent representation of the system, despite the inclusion of a simplified friction model [5].

Another important work has been prepared by Zafer Tekiner and Ihsan Korkut from Gazi University. They have designed an interactive computer program for computer-aided pneumatic circuit design. The pneumatic circuit elements are selected during the design by the in accordance with the aim the user is going to specify. A database was established by forming IGES -les for pneumatic circuit elements. In addition to this database, lists displaying the connection nodes of each element were prepared. The coordinated nodes are connected during the circuit drawing by referring to these lists. The developed program uses Delphi 3.0. Its main disadvantage is that it can produce design circuits with only up to three cylinders. And circuits are pre stored, and are chosen by the user from the computer database. So it can not be regarded as a sequencing circuit designer, but a draughting program with data bases containing some circuit formats [6].

Xiong Wei, Bao Gang, Wang Zuwen from Harbin Institute of Technology, Harbin, P.R.China have come up with another CAD program. Entitled as “Study of CAD Software for Pneumatic Driving Systems”, their work considers pneumatic systems by a new modeling method based on pneumatic dual-channel pseudo-bond graph and a complete bond graph model of pneumatic driving system is presented. A CAD software, PneuCAD, which has five main functions including pneumatic circuit drawing, design for pneumatic power circuit, design for pneumatic control circuit, simulation and model selection, is also coupled to their program [7].

1.3. Brief History of Pneumatic Systems

From the earliest recorded times, man has used the air around him to drive his sailing ships and turn the blades of his windmills. Sailors and shipbuilders knew through countless years of trial and error how to cut and set a sail for maximum efficiency. Why the pressure and flow of air made one shape better than another and the theory on the interaction of the moving sail and wind was not known until the recent times. Early Egyptian writings tell of priests using simple bellows to compress air. The air was then made to "magically open and close" temple doors to the wonderments of the assembled rulers and common people.

First known application of compressed air have been not for power transmission, but for supporting charcoal fire for smelting metals. Air bellows, pumping air onto the fire caused speeding up of combustion and resulting high temperatures enabled the smelting of copper, and later bronze. Bellows used were primitive compressors which compressed air and caused it to move violently over the fire. These bellows were most probably made of animal skin stretched over a wooden frame. Air must have been entering through holes other than the holes through which it leaves out. Either holes or sets of holes were closed by thin reeds acting as a check valve, allowing flow only in one direction [8,9].

Power transmission first required a sort of compression, to increase the pressure of air. Transfer of pressurised air has always been a great problem and clay pipes used for the transmission of water have been unsatisfactory for compressed air. First notable technical development is perhaps the vacuum and air pumps made by Otto Von Guericke, Magdeburg born German physicist and engineer. With his experimental machines, he investigated on the role of air in combustion and respiration. He made his first suction pump in 1647 and air pump in 1650. With his suction pump, he performed the famous experiment of “Magdeburger Hemispheres” before Emperor Ferdinand III at Ragensburg and demonstrated the greatness of atmospheric pressure forces. In 1672 he constructed a cylinder with a close fitting piston and used it to generate mechanical forces by vacuum. This work comprises an important example on the art of manufacturing which was so primitive by then [10].

Pneumatics have been a clean and cheap way of transmitting power from one location to other and have found greatest field of application in construction machinery. One of the earliest applications of Pneumatics science in construction has perhaps been the Mount Cenis tunnel in French Alps. It is a 8.5 mile tunnel joining Modane in France to Bardonerchia in Italy, via a railroad. During this construction, which took 14 years starting from 1857, pneumatic rock drills invented by Germain

Someiller, French engineer from the University of Turin were used extensively [11]. Another notable example, this time from the New World, has been the Hoosac

tunnel, which is a railway tunnel beneath the Berkshire Mountains in Western Massachusetts. A marvel of its age, the 4.82 mile tunnel was first proposed in 1819 and was completed in 1875. Its construction embedded some new techniques of tunnel making when existing methods of gunpowder explosions came up useless. Uses of Nitro in holes cut into the rock by hand drills as a new technique have increased the speed of tunnel making to 60 feet-per-month. Later in 1866, the Burleigh Drill, a pneumatic equipment invented by Charles Burleigh of Fitchburg, Massachusetts, and a specially designed 4-cylinder compressor to power them were first taken into service, nearly doubled the construction speed [12].

Another very notable example of pneumatic power application is the famous Paris City Sewers. Piping extending all over the city brought pneumatic power from a 2000 HP compressor battery in the middle of Paris. This central compressor station has been instrumental in the construction of all roads, bridges and other constructions of art, which has made Paris a glamorous city. The sewers are tunnels beneath each street and are 2100 kilometers in total and within the main arteries, are total of 751 kilometers of compressed air pipes. Later, after the advent of diesel engines, this monumental pressurised air network became obsolete and is used by touristic purposes now. In Paris, trams using compressed air were in use in 1900's on the route TJ from Passy to Paris, Hotel de Ville. Between the trams is the compressed air, water, and coke charging point. Each tram had 8 reservoir cylinders under the floor, in which a charge of compressed air at 80 atmospheres was stored. The air was then supplied via a regulator to two driving cylinders, at about 8-10 atmospheres. Mounted on the front platform near the regulator, was a cylindrical tank known as a "Bouillotte", where a small amount of steam was produced and mixed with the air, to prevent the icing when air expands. French engineers have always boasted to use the pneumatic power at the least expected scenes. For example, the Eiffel Tower, the most famous Parisien monument, is composed of 12 000 metal parts put together by 2,500,000 rivets, all hammered by pneumatically driven riveters [13].

Up to late 1980's, pneumatic power could not be controlled very effectively due to the compressibility of air. Control was "bang-bang", actuators stopping only when

they come against mechanical stops and usually having great velocities at the instant of impact. With this kind of a control, intricate jobs like guidance of a rocket or the manipulation of a dynamic load by a robot or controlling a machine tool to follow a cam profile could not be achieved. From 1980 onwards, the Festo Company of Germany succeeded in making servo valves controlling air, functioning in comparable performance to that of a hydraulic servo valve. This company, within their Esslingen plants, are pioneering the horizons of the science of pneumatics today.

Applications of pressurised air is so vast now, ranging from modern compressed air motors to gas turbines, from air guns to vehicle suspension, from power braking to steering systems. Development of servo valve technology by the German company Festo opened new horizons in the last decade of the twentieth century, to cover innumerable servo actuation applications [14].

1.4. Principles of Pneumatic Systems

Atmospheric air is comprised of a mixture of different gases in volumetric proportion of nitrogen and oxygen, 78 per cent and 21 per cent by volume respectively [15], together with trace elements of argon, helium, hydrogen and neon.

Although air is a mixture of different gases, for practical purposes we can treat it as though it is comprised of a single gas, since it conforms to the General Gas Laws. In the application of pneumatics to the transmitting and control of power we are not concerned with the development of pneumatic devices and components with rather with the use of components already in existence thus there is no need for us to go into the theoretical implications of the Gas Laws in any great depth [16].

1.4.1. Pascal's Law

As indicated in the previous section, pneumatic systems are circuits in which air containing various levels of energy runs. At an energy injection locality, pressures

and/or flowrates will be high. At an energy deduction locality, pressures and/or flowrates will be low. At any condition, the medium which carries energy is air and motion of air particles are governed by Newton's second law. If there is a net force acting on a fluid bulk, it will accelerate in proportion to and in direction of the force. Accelerations through time will generate velocity variations. Forces active in a fluidic environment are mainly pressure forces, viscous forces, gravity forces due to variation in elevation, gravity forces, all inertia forces due to effects like centrifugal accelerations, capillarity, surface tension and electromagnetic effects, if any. Resulting formulas are very complicated in nature [2]. They are in form of partial differential equations and as they are very stiff, numerical integration becomes difficult. Main contribution to a mathematical stiffness is that the mass of the moving air particles is very small, that is, a small force generates great accelerations and this causes great velocity and displacement variations.

Mass of a moving air bulk is often assumed zero and Newton's first law is converted into a unique and singular form as "Zero equals to zero". Then problem becomes decisive: net force and net moment acting on a fluid particle is zero. Application of Newton's first law on fluid bodies was first done by the French scientist Blaise Pascal and his findings is now known as "*Pascal's Law*". Pascal's law is summarised in the following statement: "Pressure on a confined fluid is transmitted unchanged through the fluid", with its simplest form. The important aspects to be referred to separately will then be the followings [17]:

A static fluid must be confined in a closed volume, otherwise it flows away. The pressure on a static fluid will be the same throughout the fluid. This statement neglects the effect of gravitation. Normally the particles at the bottom of a container are effected by not only the fluid pressure, but by the weight of the fluid above it, but as the mass of air particles is negligibly small, this statement is justified in itself. The static pressure acts equally in all directions at the same time, that is when pressure is effected by some means, increased by a pump or reduced by a motor, variation in the rest of the whole circuit is instantaneous, without any delay. Actually pressure propagates in a wave form and it takes some finite time to reach to a particular

faraway point. As the size of a pneumatic circuit is not too big, propagation of the pressure wave can be regarded at infinite speed.

State of a fluid particle or bulk is characterised by its position in space, conveniently by its x, y, and z coordinates, its pressure, temperature, density, viscosity and time. As pneumatic air is regarded static, its position will not change. So, first three characteristics become just constant numbers and corresponding motion equations diminish automatically. Viscosity is assumed zero and time in a static problem is immaterial. Remaining three, that is, the pressure, temperature and density arise as the variables describing the state of pneumatic air. Energy is loaded onto an air bulk by increasing its pressure, but a variation in pressure causes corresponding variations in volume and temperature. The relationships between these three variables have been found during an era called the “*Scientific Revolution*” which came just before the “*Industrial Revolution*” and have facilitated it. Gas Laws are so popular and important that references on them are numerous, including freshman textbooks [18], more scientific sophomore textbooks [19] and technology books [16].

1.4.2. Boyle’s Law

The inverse proportionality between the pressure and volume of a constant mass of gas was first discovered by English Scientist Robert Boyle in 1662. Robert Boyle, while at the Oxford University, was acquainted with the air and vacuum pumps of Otto Von Guericke, referred to at some earlier part of this dissertation and with the help of Robert Hooke, prepared a similar machine which they called the “*Machina Boyleana*”, meaning “Pneumatical Engine” in Latin. With this machine he could change the pressure of air. This lead to the statement: *under conditions of constant temperature and quantity, there is an inverse relationship between the volume and pressure for an ideal gas*, which can be mathematically stated as:

$$P \propto 1/V \quad (1.1)$$

or $PV = Constant. \quad (1.2)$

Equation 1.2 can be written as:

$$P_1V_1 = P_2V_2 \quad (1.3)$$

Where, indices 1 and 2 stand for two different states of the same gas. This law assumes a perfect elasticity for the medium, which is hence called the “ideal gas”. For a normal gas, because of intermolecular friction, a pressure greater than that specified in equation 1.2 is required. Equation for a realistic gas is specified as:

$$PV^n = \text{Constant}. \quad (1.4)$$

n is a characteristic to the particular gas, called the “Polytropic index” [16,18,19].

1.4.3. Charles’ Law

The next important advance in the study of gases came in the early 1800's in France. Hot air balloons were extremely popular at that time and scientists were eager to improve the performance of their balloons. Two of the fore-running French scientists, Jacques Charles and Joseph-Louis Gay-Lussac, made detailed measurements on how the volume of a gas was affected by its temperature. Charles brought a certain mass of gas entrapped in a glass flask to a temperature he set with a thermostatic bath. He kept the pressure constant by the weight of a mercury column acting on the heated gas. As the gas changed its volume, location of the mercury column changed. With this experimentation, he found out that Volume and Temperature of the gas are in direct proportionality, that is, when temperature increases, volume increases or vice versa. This statement can be represented mathematically as :

$$V \propto T \quad (1.5)$$

or

$$V/T = \text{Constant} \quad (1.6)$$

Equation 1.6 can be written as:

$$V_1 / T_1 = V_2 / T_2 \quad (1.7)$$

Where, indices *1* and *2* stand for two different states of the same gas again. *T* is the absolute temperature of the gas [16,18,19].

1.4.4. Characteristic Equation of a Gas

Boyle's and Charles' laws can be combined into a single equation as:

$$PV = mRT \quad (1.8)$$

Where *m* is the mass of the gas and *R* the *Characteristic Gas Constant* for the particular gas. Its value for air is 286.9 J/kg °K [16,18,19].

1.5. Compression and Expansion of Air

Assuming that gas is an elastic medium, it stores potential energy when compressed and delivers this energy back when allowed to expand. Of course, speeding up the gas is a second alternative of loading energy in kinetic form. Technology for that is completely different and is out of the context of this thesis. The change in volume is accompanied by a change in pressure or temperature, or both. The way in which the pressure, temperature and volume change is dependent upon the rate of flow of energy into or out of the gas. Provided there is no change in the mass of the gas, which means in practise that gas does not leak away, it will obey the characteristic gas equation.

In pneumatic machinery, generally the volume is the independent variable, controllable by a variable volume device like the cylinder-piston arrangement seen in figure 1.1. High pressure active inside the compartment behind the piston generates a force *F*, delivered from the piston rod as:

$$F=PA \quad (1.9)$$



Figure 1.1 Piston being extended by a continuous supply of compressed air

Where A is the area of the piston over which pressure is acting. On the other side of the piston, there is atmospheric pressure, which is at zero gauge. Operation of the cylinder is reversible. If force F is applied externally, compression is done and if gas in the system is allowed to expand, it will apply the force F and move the piston with a speed v , where, from continuity of the medium,

$$v = Q/A \quad (1.10)$$

Q being the volume flowrate of air into the actuator in liters/minute. So, a study of compression or expansion of air is the basis in all power calculations.

Volume is the independent variable in any such process. Any compressor or motor mechanisms control the position of the piston for example, completely arbitrarily. In compression, some of the input mechanical power is converted into heat, which increases the temperature of the gas. This causes a thermal expansion of the gas and the force required to drive the piston increases. This is a cause of inefficiency and must be reduced as much as possible. For that, the generated heat must be driven out of the gas and the temperature of the gas must be kept constant. This is a time consuming process and so is not easy to achieve during the compression. A realistic compression process is never isothermal. In practise, while in operation, a compressor is always at a higher temperature than its environment and so continuously transferring heat to outside. Therefore a realistic compression is neither adiabatic. The law governing the compression or expansion of air is:

$$PV^n = \text{Constant.} \quad (1.11)$$

Here, n is the isentropic index. Its numerical value is 1 for an isothermal process, and is equal to C_p/C_v for an adiabatic process. *Specific Heat at constant Pressure* C_p for air is 1.005 KJ/kg-°K and *Specific Heat at Constant Volume* C_v is 0.718 kJ/kg-°K for the range where pneumatic systems operate [19]. n for an adiabatic process is hence 1.4, without any unit. For any compression, n is anything between 1 and 1.4, smaller if compressor is running slowly and greater if compressor is running fast [16].

A compression process is seen on a P - V diagram in Figure (1.2). Area under the P - V curve is the Work of Compression. As process is cyclic, V can be replaced by Q , and then, area under the curve becomes power. Every object on the surface of the earth is under a thick blanket of air and atmospheric air is not at zero pressure, but at 1 bar absolute at sea level. So, air coming into a compressor cylinder already contains energy of P_1V_1 . After compression to state 2 as shown in the figure, pressure reaches

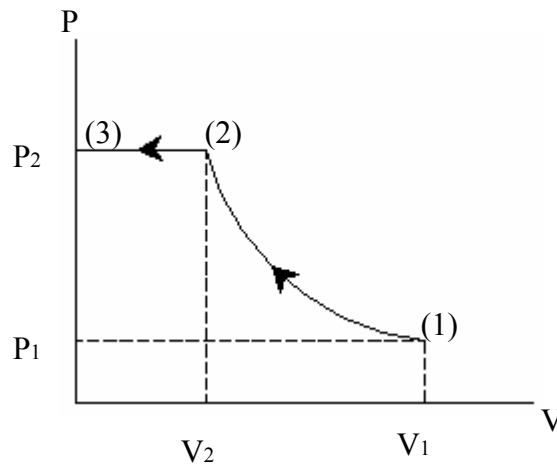


Figure 1.2 P-V diagram of a compression process.
Expansion follows the same route in opposite sense

to P_2 , the utility pressure, which is kept at a constant value by the relief valves. After point 2, piston starts to drive the air out into the system, and requires an additional work of P_2V_2 , hence, energy of compression or expansion is:

$$\text{Work of Compression} = \int_{V_2}^{V_1} P dV + P_2V_2 - P_1V_1 \quad (1.12)$$

$$\text{But as } PV^n = C \quad \text{and} \quad P = CV^{-n} \quad (1.13)$$

$$\text{Work of Compression} = \int_{V_2}^{V_1} CV^{-n} dV + P_2V_2 - P_1V_1 \quad (1.14)$$

$$= \frac{CV^{1-n}}{1-n} \Bigg|_{V_2}^{V_1} + P_2V_2 - P_1V_1 \quad (1.15)$$

$$= \frac{CV_1^{1-n} - CV_2^{1-n}}{1-n} + P_2V_2 - P_1V_1 \quad (1.16)$$

Negating the numerator and denominator of the fraction we get:

$$= \frac{CV_2^{1-n} - CV_1^{1-n}}{n-1} + P_2V_2 - P_1V_1 \quad (1.17)$$

$$= \frac{CV_2^{-n} V_2 - CV_1^{-n} V_1}{n-1} + P_2V_2 - P_1V_1 \quad (1.18)$$

But CV_2^{-n} is P_2 and CV_1^{-n} is P_1 , Substituting we get:

$$\text{Work} = \frac{P_2V_2 - P_1V_1}{n-1} + \frac{(n-1)(P_2V_2)}{(n-1)} - \frac{(n-1)(P_1V_1)}{(n-1)} \quad (1.19)$$

Taking into the common denominator we get:

$$\text{Work of Compression} = \frac{n}{n-1} (P_2V_2 - P_1V_1) \quad (1.20)$$

The gas laws and the compression work equation given above are not directly related with the computer program prepared in this thesis but are essential if a dynamic simulator is to be made perhaps as a second stage amendment for this work

1.6. Solid Logic

Automation problems concerning the sequencing of successive events can most simply be obtained by appropriately designing pneumatic control circuits. Contrary to computer control, these circuits are more reliable, easier to implement and maintain, and are cheaper. Due to these reasons, automations with pneumatic logic circuits have become quite common in clean machinery handling food, drugs and cosmetics and packaging machinery. There are many unique features of pneumatic control compared to other types of control. These are fundamental and account for the wide use of pneumatic control. Some of advantages of solid logic are the followings:

- Cheap components
- Easy to understand and apply
- Easy & cheap to maintain
- Cheap to make
- Reliable
- No electric components

Although pneumatic controls offer many distinct advantages, several disadvantages tend to limit their use. Major disadvantages are the followings:

- Only valid for simple problems of automation
- Can not adapt itself to external disturbances
- Requires pneumatic source which is not readily available

CHAPTER 2

DEFINITION OF PNEUMATIC CIRCUITRY

2.1. Introduction

Machines perform a useful task in a series of operations, one carried out after the former. Each operation is done by a specific mechanism. These mechanisms require actuation in a serial manner. So, some means of sequential and intermittent drive is necessary. This can be obtained by a computer control. More rugged form of electronic controllers suitable for operation on shop floor are *Programmable Logic Controller's*. A more reliable, yet cheaper means is the use of *Fluid Logic*. Pneumatic sequencing circuits contain more than one actuator, but it is necessary to know how a single circuit is commanded before commencing to more complicated circuits.

In pneumatic logic, each successive event is started by the forward action of an actuator and terminated by its reverse action. Each such actuator is controlled by a 2 position 5-way spool valve, whose spool position is controllable by the pressure variations within the boost chambers. Position of each actuator is monitored by two valves, one triggered at the end of extension and other triggered at the end of retraction. These valves are of 2-position and 3-way type. Design of a circuit for simple problems is somehow easy, but circuits for job descriptions concerning irregular timing and sequencing may become extremely difficult. A systemacy based on grouping a complicated task into simpler sub problems, called cascaded circuits is available to design such a circuit.

In this study, a computer program is built for designing complex pneumatic circuits. The method described can be used on a design table, implemented by pen and paper as can be put in form of a computer program. As referred to above, the program

breaks a complicated sequence of tasks into simpler tasks and puts them into an order of operation by cascading. After the circuit comes up, it is brought to the computer screen in standard symbols and also can be printed on paper.

2.2. Pneumatic Components

In studying either pneumatic or hydraulic fluid power circuits, it is recommended that the less complicated circuits be studied before advancing to the more complicated circuits. Design process obeys the same philosophy, "*Simpler the design the better it is*". A useful pneumatic circuit has adequate enough pneumatic components. The pneumatic design program generated in this study uses the fewest pneumatic components. The pneumatic components used are the following:

1. Double-acting cylinders
2. 5 way 2 position valves (5/2 valves)
3. 3 way 2 position valves (3/2 valves)

Only these three components are enough to design a useful cascade pneumatic circuit. In this study 5-way 3-position valve is not used. Centre position of 5/3 valve is neutral and is active when not effected. In automation the process is in continuous operation and neutral position is not required. The Computer program presented here also shows pipe connections of the pressurised air between the valves and cylinders.

2.2.1. Double-Acting Cylinders

In a general sense, the term "*Actuator*" can be defined as a device that converts fluidic power into mechanical, in form of a translating ram, exerting a force. A wide variety of actuators are available. Their common characteristic is that they all provide variable volume enclosed compartments and any mechanism which provides this facility can be used as an actuator. Actuators may involve both linear and/or rotary motion. Some provide a continuous rotation and hence are called "*Motors*". Mechanisms like sliding, rotating or swinging vanes, external or internal gear pairs,

and abutments provide various designs. Many incorporate a battery of cylinders. Actuating cylinders, one or more in number, contain pistons and piston rods with the necessary seals. In a double-acting cylinder, fluid under pressure can be applied to either side of the piston to provide movement in the corresponding direction [20].

To control the extension or retraction strokes of a double-acting cylinder, the appropriate cylinder ports have to be pressurised and exhausted. The supply of air to and from the cylinder can be done by using a row of four 2-way 2-position valves, or two 3-way 2-position valves, one controlling each port of the cylinder, or by one 5-way 2-position valve or by one 5-way 3-position valve [21]. Among these, the first and second possibilities are not practically feasible as they utilise greater number of components. Similarly a 3 position valve, the fourth possibility, with a neutral position, generates a non-definitive state of pressures there and hence not preferred in automatically sequencing circuits, the subjectmatter of this thesis. In this study, the use of 5-way 2-position valve to control a cylinder is adapted, as shown in Figure (2.1).

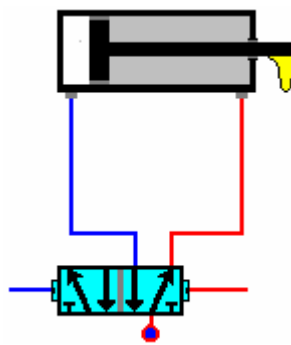


Figure 2.1 Double-acting cylinder , operated by a 5/2 control valve

The portion of a circuit given in Figure (2.1) shows the “*Power*” portion, where great powers of actuation are exchanged. This portion is common to all actuators. Actuators can be single or double action. In *Single Action* cylinders, there is only one control port through which pressurised air can be sent. So, actuator is controllable only in one direction. Reverse action is not fluidic and can be performed by gravity or a return spring. In most applications, actuator is required to apply forces in both directions and hence a *Double Action* cylinder is more preferable. Due to the

difference of area between the full bore side and the annulus side of the piston, the extending force will be greater than the retracting force at the same supply pressure. On the contrary, the retraction speed will be faster than the speed of extension given the same supply conditions.

End-position cushioning can be achieved whenever necessary by redeeming or switching off the incoming pneumatic power by effecting the control valve using control cams or a permanent magnet mounted on the cylinder piston, whose magnetic field can trigger a proximity switch [14].



Figure 2.2 Double-acting cylinder
(Festo, Actuator coding starting with “D” standing for “Double Acting”)

2.2.2. Five Way Two Position Valves

Two 3-way valves, assembled together to be fed from the same supply line are also available with five external ports. One of the ports is the pressure port, two actuator ports, and two exhaust ports. Such valves provide the same basic control of flow paths as the four-ported version, but have individual exhaust ports. In four port designs, two exhaust lines are coupled together and are generally used in hydraulic systems. There, exhausted oil is not discarded but returned into the oil tank. In the fluid power field this is referred to as a *"Five-port, two-way valve."* This type of valve brings all flow paths to individual external ports. The pressure port is connected to system feed after a regulator which keeps the pressure coming to the valve at the required levels by dynamic means. Actuator ports are connected to inlet and outlet ports of a cylinder or motor. Each exhaust port serves an actuator port [22].

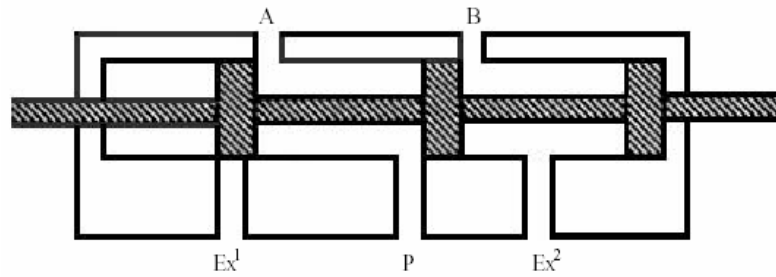


Figure 2.3 A spool valve with inlet and outlet ports
(Control port A is pressurised and B exhausted.)

These valves are the most commonly used as power valves, used for controlling both cylinders and group change lines. These valves are controlled with pneumatic action. A typical 5-port 2-position valve made by Festo Company [14] is shown in Figure (2.4).



Figure 2.4 5/2-way double control valve, pneumatically actuated, by boost chambers on each side of the spool. (Festo, J type valves)

The pneumatic double pilot valve is reversed by pneumatic signals sent to boost chambers on alternate sides of the spool. The circuit state is retained after removal of the signal until the next counteracting signal comes.

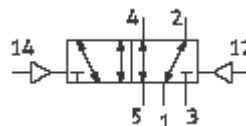


Figure 2.5 Standard symbol of a 5/2 Pneumatic valve per ISO 5599-1

2.2.3. Three Way Two Position Valves

These are usually simple, poppet type valves and are in “*Normally Closed*” (NC) form in their de-energised condition. Poppet configuration is adopted due to simplicity in design and great flowrates through relatively small body sizes. Typical applications of this type of valve include the control of “*Spring-Loaded*” or “*Gravity-Return*” cylinders and the indirect control of large cylinders by providing pilot control switching for the main control valves [23]. Two 3/2 valves can be used for each actuator, one is at the end of extension and the other is at the end of retraction, as shown in Figure (2.6).

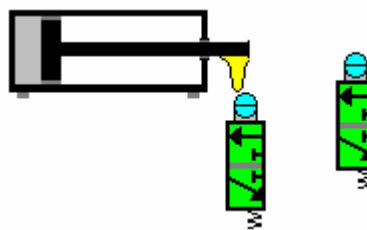


Figure 2.6 Placement of 3/2 pilot valves with respect to the actuator.

In such cycles, instead of relying on the physical manipulation of any valves to set each movement in train, the respective relay valves controlling the movements are changed over when required by air pressure from “pilot” valves. Pressure from these blows the spools or pistons of the relay valves over, causing the relay valve themselves to redirect the flow of air, which is connected to their inlet ports [16].

Valves used to drive pneumatic actuators have to be correctly sized to handle the required air-flow rate. When large valves are used the pipework connecting the valves and actuators must be of a correspondingly large bore. The valves should be positioned as near as possible to the actuator so as to reduce the length of pipework between the valve and actuator to a minimum; this will result in a reduced air consumption per cycle. The valve may have to be operated remotely, and one method of doing so, in standard symbols, energizing ports 12 or 14, the inlet ports to boost chambers. In Figure (2.7), a 3/2 valve, mechanically controlled by a roller-lever is

shown in standard symbols. This valve can be used to provide a pilot air signal to operate the main valve spool [14].



Figure 2.7 3/2 valve with roller lever control.
(Festo AR type valves, R standing for “Roller”)

2.3. Solution Techniques of Pneumatic Circuit Problems

Circuits designed to do simple tasks will accordingly be simple and hence can be prepared without using any systematic method, but more difficult tasks certainly require some more complicated circuitry, neither possible to design free hand nor to visualise how it operates. So, it is important to approach any circuit about to be designed with the mind completely free from any preconceived ideas of how any part of the circuit will actually be designed. Once it has been decided to use a pneumatic control circuit, as against any possible alternatives, the designing of the circuit should be carried out in a definite, logical series of steps, no step being taken for granted, assumed or skipped [16].

A sequence of operations may be;

- a) Event based
- b) Time based
- c) A combination of event and time based

In an event based sequence the completion of one operation causes the next operation to start. Time-based sequences have events occurring at pre-set time intervals and are usually controlled by either mechanical or electronic programmers [23]. In this study only event based sequences are considered. Time requirements of individual events can be met by controlling the air flowrate by needle valves.

Standard technique of defining a sequencing problem adopted by all the books and other documentation is that each cylinder will be given a reference letter in consecutive order, starting from “A”. The cylinder state, that is, whether extended or retracted, is denoted by “+” or “-”, thus A+ means that cylinder A is extended and A- means that cylinder A is retracted. For example, the sequence given as A+ B+ A- B- means:

- Start of cycle
- Then cylinder A extends (A+)
- Next cylinder B extends (B+)
- Then cylinder A retracts (A-)
- Finally cylinder B retracts (B-)
- End of cycle.

2.3.1. Single Circuits

A “*Single Circuit*” is a simple circuit where successive actuators trigger the operation of the next actuator by sending pilot signals to their power valves in succession. Operation and design is easy without the need for group changing and hence extra valves to do it which will be referred to as the “*Cascade Valves*”. Simplest such problem is the sequence A+ B+ A- B-. One further step problem is the sequence A+ B+ C+ A- B- C-. These sequences are event-based single-cycle operations. At the end of cycle the cylinders all come to the start position. In this case all the cylinders must be fully retracted at the end of the cycle. In any complete sequence there must be the same number of extending operations (+ signs) as retract operations (- signs) for each cylinder.

2.3.2. More Intricate Circuits

Tasks which contain criss-crossing events and not looking symmetrical in definition and not orderly can not be done by “Single Circuits” These circuits are complicated, difficult to design and require more number of components and are called “*Cascade Circuits*”. As the operation of proper simple circuits are commanded by cascade valves. The Cascade principle has been developed for pneumatic sequences where trapped signals may occur.

Design of cascade circuits require the application of a systematic solution technique, which can be found in moderately advanced design or technology books. Even though the solution procedure is straightforward, solution can be very laborious and time consuming for more difficult problems, containing greater number of actuators and cascading. At the end of the solution the circuit coming up may still be inoperative, usually signal inversions being required here and there. So, an experimental verification is always necessary. A good circuit simulator program can also be utilised instead of physical experimentation.

The design procedure can be found in numerous references at various levels [16,24] The explanation about the selection of the right number of components in Pinches and Callear book [23] which is used as textbook in related courses at the university where this thesis is prepared is quite understandable. In the following section the procedure of the method will be repeated from this book. None of the references examined by the author of this thesis unfortunately contains any systemacy to do the pipeline connections between the valves. So, after the components are identified by using the abovementioned reference, pipeline connections are done according to a specially designed systemacy and its computer implementation in the next chapter. All are then explained by examples.

In the design procedure, the sequence is divided into a number of sections or groups, which do not contain a trapped signal. The pilot air is switched so that only one group is supplied with pilot air at any one time. To design a cascade circuit a series

of steps are followed, as illustrates in the following example to do the sequence [A+ B+ B- A-], using double pilot-operated direction control valves.

1. The sequence is divided into groups so that no letter repeats in any group. So the sequence under consideration is to be separated into two groups as follows:

A+ B+ / B- A-
Group I / Group II

2. Each cylinder will be powered by a 5/2, pilot operated valve for switching the operation on or off, a stop/run valve should be added.
3. Put pilot air changeover valves (cascade valves) at the far end of the circuit where group lines are located. If 5-port valves are used, number of valves required will be one less than the number of groups. In this example only one cascade valve is required as there are only two groups.
4. Group lines I and II are to be prepared next to the cascade valves. The group lines are sometimes referred to as '*Bus bars*'. Colour coding may greatly help distinguishing the pipelines from each other.
5. The sequence starts with A+, which is the first step in group I. So, to initiate the sequence, a signal from group I 'bus bar' will be taken and fed to V+ actuation, preferably through a start/stop valve. V in this symbol represents "Valve".

In general, if the sequence does not start with a new group, for example;

A+ / A- B+ / B-
I / II / I

take a signal in the appropriate group from the start/stop valve to initiate the first step. Work through the sequence and use the last operation to supply a signal to the start/stop valve which will start the sequence over again.

6. At this step cylinder A extends (A+). This is in group I, so trip valve 'P_A+' signal must be supplied with group I pilot air. The output will be taken to valve V_B to start the next step B+.
7. At this step cylinder B will extend. This operation is still in group I, so group I pilot air is supplied to the trip valve as 'P_B+'. The output signal from 'P_B+' is used to switch the cascade valve which exhausts group I lines and supplies air to the group II lines.
8. The first step in group II is initiated by feeding group II pilot air directly to the ' - ' side of valve 'V_B-' causing cylinder B to retract (B- move).
9. When cylinder B retracts, it operates 'P_B-' , which is supplied with group II air. The output from P_B- trip is fed to the ' - ' side of valve 'V_A-' to give A-movement
10. Cylinder A retracting operates trip valve 'P_A-' which again is supplied with group II air. The output from 'P_A-' is used to switch the group change valve back to group I.
11. The cylinders have now been through a complete cycle. The last operation in the cycle is used to initiate the next, in this case the change to group I, with group I pilot air being fed to the start/stop valve.

The cycle is now complete, as illustrated in Figure (2.7). When splitting a sequence into groups for a cascade circuit ensures that the minimum number of groups is determined. Split the sequence into groups starting at the beginning or end of the sequence. These are forward and reverse splitting. Forward splitting is used in this

study. Consider the following sequence shown split in both forward and reverse directions:

Forward splitting: **A+ B+ / A- C+ D+ / D- C- B-**
 Group I / Group II / Group III

Reverse splitting: **A+ / B+ A- C+ D+ / D- C- B-**
 Group I / Group II / Group I

Method of splitting may effect the total number of cascade valves used. Rules for the design of a cascade circuit free hand are the followings [23].

1. Divide sequence into groups so that no trapped signals can occur, that is, no letter repeats in a group.
2. Draw the circuit components, cylinders, direction control, trip and stop/run valves. Mains air can be shown supplied through the direction control valves to set the cylinders into their start conditions.
3. Draw the cascade valves and group pilot air lines.
4. All the pilot valves in a group are supplied with air from that group. The signal from the last step in a group is used to switch the appropriate cascade valve and thus step the pilot air onto the next group.

These four steps listed above are taken directly from the book of Pinches and Callear, entitled “Power Pneumatics”. Similar recopies can be found in many other books, but as seen, there is no direct explanation of the pipeline connections between the valves and in general authors explain the method on an example.

The first operation in a group is initiated by pilot air directly from the group air line. With other methods of overcoming trapped signals, such as signal breakers, switch valves or pulse valves, it is necessary to identify exactly where the trapped signal will occur. Cascade circuits avoid the problem because ‘ – ’ providing the above

procedure and rules have been obeyed ‘ – ’ air will be supplied to a particular section of the control circuit only when it is needed. Although trip valves triggered earlier in the sequence may remain in the operation condition, no signal is transmitted because their air supply has been removed.

Cascade circuits offer a relatively foolproof system for the control of many sequences. This does not mean that they necessarily present the best solution in every situation. They can, however, provide a sound basis for developing a more practical or economical circuit [2]. In Figure (2.8), the circuit diagram to perform the sequence A+ B+ B- A- is shown as taken out from the computer program prepared in this thesis. This program is named as “*Pneudesign*” by its programmer. As seen, the output comes in form of an ISO standard chart, colour coded for distinguishing the pipelines from each other and for following the flow of events in a simpler manner. There are port names of control valves, names of pilot valves too.

This sequence in the example is a widely used one. For example, a machine which prepares cardboard boxes first shapes the properly cut cardboard over a die with the action of actuator A as A+. While the cardboard is in position, stapling can be done by actuator B as B+. Then staplers are retracted by B- and the finished box in position is released by the first actuator by the A- movement.

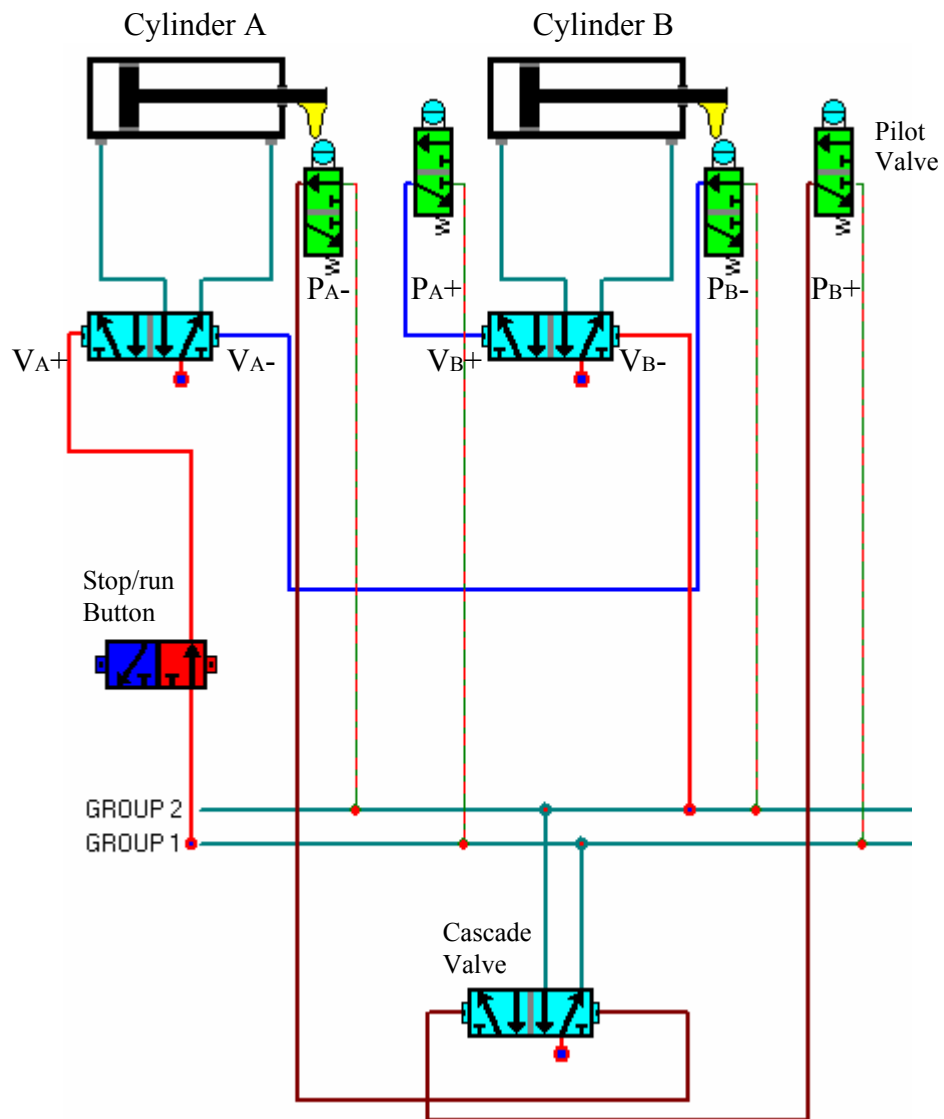


Figure 2.8 Cascade circuit for the sequence A+ B+ B- A-
 In the form output by the computer program "Pneudesign"
 This circuit is patched on the pneumatic trainer bench of
 Festo Didactic and proved to be properly working.

CHAPTER 3

THE PROGRAM “PNEUDSIGN”

A computer program is prepared in this thesis for designing a cascade pneumatic circuit. This Computer program is called, “ PNEUDSIGN ”. This is an interactive program which, upon definition of the sequence, outputs to the user:

- A pneumatic circuit drawn in ISO standards with all the required pneumatic components
- Pipeline connections between the pneumatic components in colour coding.

The program consists of two user interfacing forms. First form is prepared for the description of the problem by the user. In this form, the user chooses two parameters. These are:

1. Selection of the number of cylinders.
This is an integer number selectable from a menu.
2. Definition of the sequence in the standard format, that is, naming by letters starting from A and movements as ‘+’ for extension and ‘-’ for retraction.
For example: (A+ B+ B- C+ A- C-) and so.

The program does a simple check and gives out a diagnostic warning if sequence has an error and repeats the problem definition session.

The second form shows the user, the resulting pneumatic circuit. Use of this computer program is very simple and handy. The programming language is “*Borland Delphi Enterprise Version 7*” [25]. This computer program is prepared at

Intel Pentium II 350MHz CPU, 128 Mbytes Ram and 8 Mbytes VGA. Program works on 1024 x 768 screen resolution.

3.1. Pneumatic Components of the “Pneudesign”

The pneumatic circuits are designed with only 3 different pneumatic components. The figures of the components are designed at the “*Borland Image Editor Version-3*”. These are:

- Pneumatic actuator (Double-acting cylinder).
- 5-way 2-position pilot operated pneumatic valve.
- 3-way 2-position roller actuated, spring return pilot valve.

An actuator is designated with two separate figures; a cylinder and a piston with a rod.

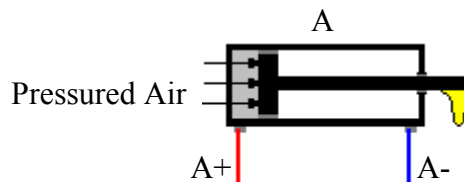


Figure 3.1 The piston will be extended by pressurised air

As shown in Figure (3.1) the cylinder “A” has two ports. The air entry port “A+” has pressured air, indicated in red. So the port connected to the compartment behind the piston will cause the extension of it and is denoted as “A+” For A+ to extend the actuator, compartment in front of the piston should be exhausted, and hence is shown in the output chart in blue. When A- port is energised, cylinder will retract. The connection of pipeline between the cylinders and control valves are the same as shown in Figure (3.2). Control valves are 5-way 2 position pneumatic components, properly piloted for the required movement. According to ISO standard drawings, which can be found in any pneumatic component catalogue or brochure, pilot on the

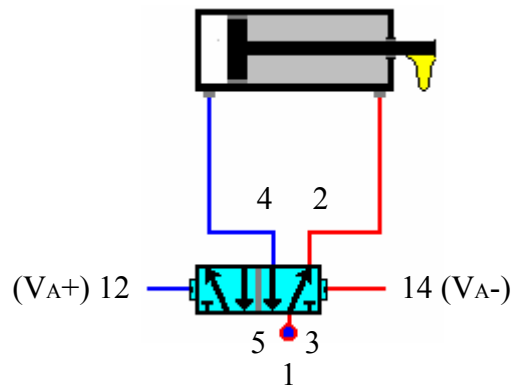


Figure 3.2 Connection between the control valve and cylinder for an A- movement.

left is denoted as port no 12 and pilot on the right as port no 14. When the pipeline that connection at the right side of the control valve is pressurised (Port 14), the cylinder retracts. When the pipeline that connection at the left side of the control valve (Port 12) is pressurised then the cylinder extends. In the computer program, ports of control valves that is, port 12 and port 14 have codes, as is shown in Table (3.1). These codes are required in designation of pipelines. Table 3.1 shows the codes up to 5 actuators, which can practically be extended without any limit The data related the actuator port codes are stored in an array named “z”.

Table 3.1 Codes of control valves at the program

Port letters of control valves	V _{A+}	V _{B+}	V _{C+}	V _{D+}	V _{E+}	V _{A-}	V _{B-}	V _{C-}	V _{D-}	V _{E-}
Codes in Pnuedesign (Array z)	1	2	3	4	5	6	7	8	9	10

All the other 5-way 2-position valves used or referred to in the circuit are “Cascade Valves”.

Figure (3.3) shows how group valves are arranged in a systemacy. For a 5 group sequence for example, 4 group valves are required. When group I is activated; all the other groups are exhausted and so, none of them are operational. This condition

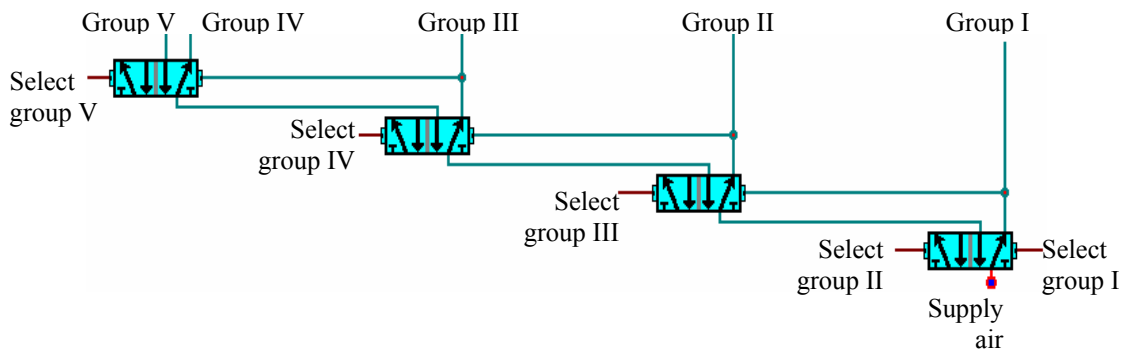


Figure 3.3 Five port valves used to give a five-group cascade circuit.

allows only group I to function. The signal coming from the last trip valve in the final group, which just finished its operation is used to switch the bottom cascade valve to re-start the new cycle. After the first group of operations are completed, it puts air into cascade valve of group II, and at the same time exhausting group I. Group II air is used to reset the third cascade valve so it is in the correct position ready for the next group change. The signal from the last trip in group II is used to switch the second cascade valve which puts air into group III, exhausting group II. The signal from the last trip in group III is used to switch the third cascade valve which puts air into group IV, exhausting group III. The last signal from the group IV is used to switch the top valve to group V and the last signal from group V resets the bottom valve, putting air back into group I. Air in the group I line resets the second valve ensuring that it is ready for the next change. At any particular time there is only one group line pressurised all the other lines are exhausting.

In the cascade system shown in Figure (3.3) the pilot air to the final group, that is group V has to flow through all four cascade valves before going to the trip valves. This can give rise to considerable pressure drops and will result in slow responses and even malfunction of the circuit. A method of overcoming this difficulty is to provide one cascade valve per group [23].

Three way two position valves in the circuit are used for two functions, as pilot valves and start/stop. There are two pilot valves for each cylinder and usually only

one start/stop button, normally not operational throughout the cycle as seen in Figure (2.7). In the systemacy of Pnuedesign, position of each pilot valve is defined by an array named m. It has elements in number twice the number of cylinders. Elements of array m are consecutive numbers, representing the ‘+’ signals available from each actuator in succession and then the ‘–’ signals available from the actuators in succession. For example, P_A⁺ signal is available when actuator A is extended and ‘P_A[–]’ signal is available when actuator A is retracted and so on. Program codes of pilot valves are shown in Table (3.2) for 5 actuators A,B,C,D,E. Actuator number can be extended without any limit.

Table 3.2 Codes of Pilot valves at the program

Letters of Pilot valves	P _A ⁺	P _B ⁺	P _C ⁺	P _D ⁺	P _E ⁺	P _A [–]	P _B [–]	P _C [–]	P _D [–]	P _E [–]
Codes in Pnuedesign (m)	1	2	3	4	5	6	7	8	9	10

3.2. Definition of the Program Application

The computer program of design of pneumatic sequencing circuits is an ergonomic, easy-to-use interactive computer program written in Delphi 7.0. It has two user interfacing forms. First form is named **“Sequences”** and the other form is named **“Draw”** by the writer of this thesis. In figure (3.4), a flowchart is shown how the **“Pnuedesign”** operates when started. First step is the input of data to describe the sequencing task. This is done interactively on a presentable page of the form called sequence as shown in figure (3.5). The user will choose the number of cylinders first. Then combo boxes of the sequence letters will be shown on the form. When the user finishes the selections of letters, he clicks the ‘OK’ button. The program then carries out a task control procedure. All letters representing the cylinders must be used in the task. No idle actuators are allowed. Also each actuator is operated only once in a cycle. It can extend first and then retract or retract first and then extend. There should

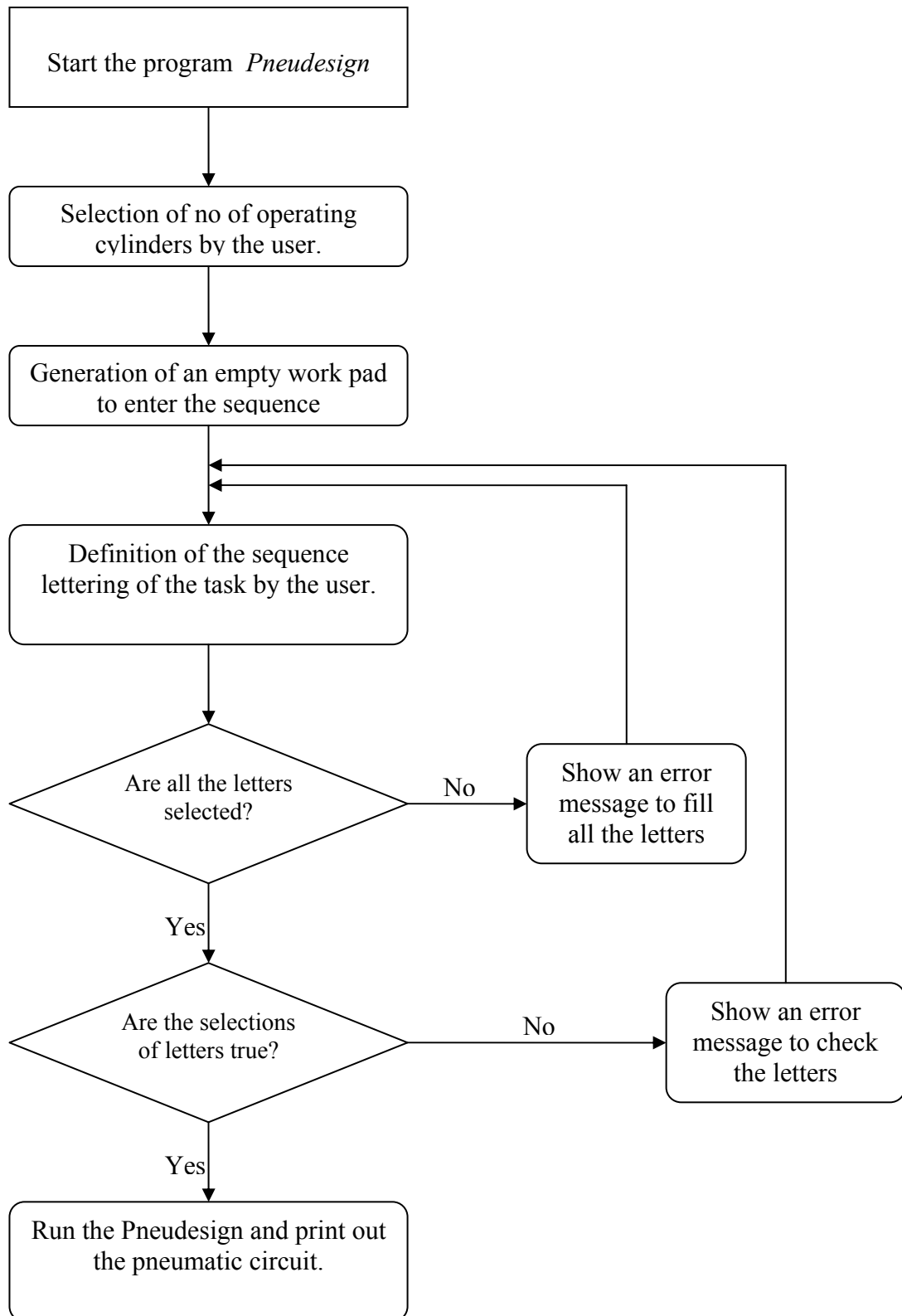


Figure 3.4 Flow-chart of Pnuedesign.exe. This section does the user interfacing and printout of the resulting circuit. It also directs input data to Pnuedesign.

be no uncompleted operation. Every actuator must be commanded to move twice, one extension and one retraction. If this definition of the task is not fulfilled, the program prints out an error message and goes back to the data input session. If the letters are selected correctly then the problem solving algorithm starts running. The algorithm is shown in a flowchart in Figure (3.6).

The screenshot shows a Windows-style window titled "Form1". Inside, there's a section "Choose Cylinder" with four radio button options: "2 Cylinder (A, B)", "3 Cylinder (A, B, C)", "4 Cylinder (A, B, C, D)", and "5 Cylinder (A, B, C, D, E)". Below this is a "Motion Sequence" section. It features a row of six numbered boxes (1 to 6), each containing a dropdown menu. The current selections are A+, B+, B-, C+, A-, and C-. Below the sequence, the text "A+B+B-C+A-C-" is displayed. An "OK" button is located below the sequence. At the bottom right, there is a "NEXT" button. A dropdown menu for the 6th step is currently open, showing a list of options: A+, A-, B+, B-, C+, and C-.

Figure 3.5 The first form at the Pnuedesign used for data input for problem definition. This form is called “sequences” by the programmer.

3.2.1. Control for Correctness of the Sequence

When the user chooses a cylinder number, program will note the cylinder number operational and the no of steps in the cycle, which is twice that number. *Step-number* means number of letters in the sequence. The program creates combo boxes as many

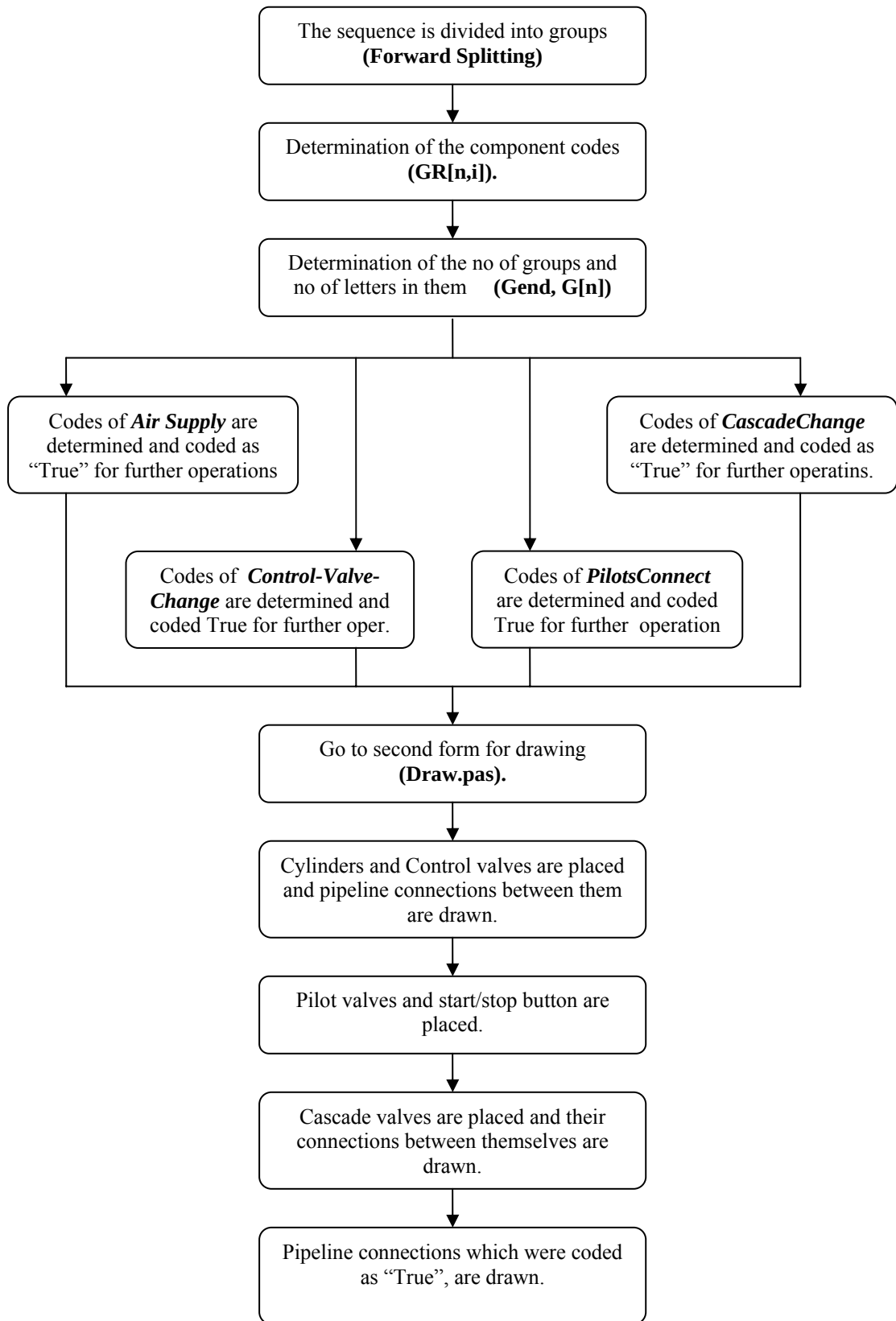


Figure 3.6 The stages of operations in the designing of a sequencing circuit.

as the stepnumber. Selection items of combo-boxes create letters as sequence of the chosen cylinder number. There are four *radio-button*'s for cylinder number. One of them is chosen accordingly by the user from :

- 2 Cylinders (A, B)
- 3 Cylinders (A, B, C)
- 4 Cylinders (A, B, C, D)
- 5 Cylinders (A, B, C, D, E)

If two cylinder radio-button is chosen, then step-number is 4. A sequence can be any combination of letters A+, A-, B+ and B- . In a sequence, a letter can be used only once. Note that “A+” is a letter; and “A-” is another letter. All combo-boxes must be filled with the selected letters.

3.2.2. Grouping a Sequence

Where the letters do not present the pattern of direct repetition, the letters in the sequence are split into groups. A group is obtained such that no letter appears twice there. In splitting the letters into groups they should always be split starting from the front. It must be remembered also that a cycle repeats and that the last part could be grouped with the first if the conditions specified above are fulfilled [4]. This is very important for cascade circuits because if the final group can be superimposed onto the beginning of the first group, total number of groups is reduced by one and correspondingly, number of cascade valves used is reduced by one. This is hence necessary for the optimization of the circuit towards minimising the no of components to use.

In the program, grouping is created by two algorithms. The first algorithm divides the sequence into groups. The second algorithm checks the last group; if the last group could be grouped with the first, then setting of the problem could be changed. For example, the sequence [A+ B+ B- C+ A- C-] can be grouped as follows:

Grouping by the first algorithm:

A+ B+ / B- C+ A- / C-
Group I / Group II / Group III

The contents of the final group is not included within the first and so it can be put at the beginning of the first group as :

Grouping by the second algorithm:

C- A+ B+ / B- C+ A-
Group I / Group II

Here the no of groups and hence the number of cascade valves used has been reduced by one.

As a second example, the sequence [A- B+ C+ A+ D- C- D+ B-] is grouped as :

Grouping at first algorithm:

A- B+ C+ / A+ D- C- / D+ B-
Group I / Group II / Group III

Grouping at second algorithm:

A- B+ C+ / A+ D- C- / D+ B-
Group I / Group II / Group III

In the analysis done by the second algorithm, components of the last group, 'D B', cannot be grouped with the first, 'A B C' as there would then be two B's in Group I. Second algorithm therefore has not changed the sequence definition. Thus, there are definitely three groups in this example. This technique used by "*forward splitting*". The total number of groups is important for the number of cascade valve. Remember that the number of Cascade valve is equals one less than the total number of groups. For that reason the program tries to split twice. At following, the first algorithm is explained as the program below:

When the user selects the letters from combo-boxes correctly and clicks the 'OK' button, group dividing starts. Selection of the combo-box will be saved to memory in a matrix form. For that, some parameters will be determined. As terminology, each letter in the sequence will be referred to as a "*Component*". These parameters are:

Memory code of count of components in that group: $G [n]$

Memory code of components: $GR [n , i]$

n : Group Number,

i : Placement of that particular component in that group.

For example: $A^+ B^+ / B^- C^+ A^- / C^-$

First component is 'A+'. It appears in group I, so it is the first component of group I;

$GR [n , i]$ will be: $GR [1 , 1] = 'A^+'$.

Second component is B^+ . It appears in group I, so it is the second component in group I;

$GR [n , i]$ will be: $GR [1 , 2] = 'B^+'$.

Count of components in group I is 2;

$G [n]$ will be: $G [1] = 2$.

'B-' is included within group II and it is the first component in that group, thus;

$GR [n , i]$ will be: $GR [2 , 1] = 'B^-'$

'C+' is included within group II and it is the second component in that group, thus

$GR [n , i]$ will be: $GR [2 , 2] = 'C^+'$

'A-' is included within group II and it is the third component in that group, thus;

$GR [n , i]$ will be: $GR [2 , 3] = 'A^-'$.

Count of components in group II is 3;

$G [n]$ will be: $G [2] = 3$.

'C-' is included within group III and it is the first component in that group, thus,

$GR [n , i]$ will be: $GR [3 , 1] = 'C^-'$

$G [n]$ will be: $G [3] = 1$.

If the last group completely or partly could be superimposed onto the components of group I as a whole or as part, then group changing will take action. The second

algorithm will change the groups accordingly. For example, the previous sequence is processed by the second algorithm as follows:

Contents of GR [3, 1] becomes nil. And G [3] will be zero
 $G [3] = 0$

So codes of group I are changed as:

GR [1 , 1] will be: GR [1 , 1] = 'C-'

GR [1 , 2] will be: GR [1 , 2] = 'A+'

GR [1 , 3] will be: GR [1 , 3] = 'B+'

The number of components of group I will be 3;

$G [n]$ will be $G [1] = 3$.

The number of groups is taken from the program memory with the variable name '**GEND**' at the computer program. This code is important for design the circuit. In the example above; **GEND** = 2.

3.2.3. Layout of the Pneumatic Circuit

After dividing the sequence into groups, pneumatic components to use are already determined. These are actuators in number as defined by the user, 3/2 pilot valves twice the number of actuators, 5/2 control valves equal in number to actuators, one will be allocated to power each actuator, one start/stop valve, in 3/2 format and cascade valves one less in number than the number of groups. These are placed onto the work chart of the "**Second Form**" of "**Pneudesign**" by the computer. Work chart is the complete screen. Actuators are drawn above the chart conventionally and group lines and cascade valves at the bottom. Start/stop button is placed on the middle left extreme. Control valves powering the actuators are drawn right below them. Two pilot valves which a particular actuator nudges when extended or retracted are placed in front of the piston rod. The 5/2 valves are made blue and 3/2

valves green for distinction. In the preceding example, the following parameters should be selected by the user:

Cylinder number: 3

Sequence: [A+ B+ B- C+ A- C-]

Then the program splits the sequence into groups as explained above. There are 2 groups in that example, hence, **GEND**= 2.

So, the numbers of pneumatic components are determined as the following:

The number of cylinder: 3 (A, B, C),

The number of pilot valves: 6 (P_{A+}, P_{A-}, P_{B+}, P_{B-}, P_{C+}, P_{C-}),

The number of control valves: 3 (V_A, V_B, V_C),

The number of cascade valve: 1 (**GEND**-1)

The appearance of the work sheet for this example is shown in Figure (3.6). This figure contains the pipelines which power the actuators but not the ones used for sequence control. This part of the problem is done by a program section named as “*PilotsConnect*”, explained in the next section.

3.2.4. Method of Design of the Pipeline Connections

How to do the pipeline connections have not been explained explicitly in the references examined by the writer of this thesis. So, generation of a working systemacy has been an important part of this study. This method consists of five different algorithms to do the following tasks:

1. Air supply connections from the group lines to pilot valves.
2. Pipeline connections from the pilot valves to control valves.
3. Pipeline connection from the pilot valves to cascade valves.
4. Pipeline connection from the group lines to control valves.
5. Pipeline connection between the cascade valves

Supply air of control valves are taken from the main compressor air. Control valves drive the pneumatic actuators. Supply air of the first cascade valve is taken from the

main compressor air as well. Succeeding cascade valves are actuated from other lines as shown in Figure (3.3). Supply air of pilot valves are taken from the group lines.

The method depends upon the groups of the sequence. Consider that the preceding example [A+ B+ B- C+ A- C-]. This sequence includes two groups, so there are two group lines at the pneumatic circuit. The first algorithm that is activated is for the pipeline connection for the inlet port of the pilot valves. This program segment is named '*Airsupply*'.

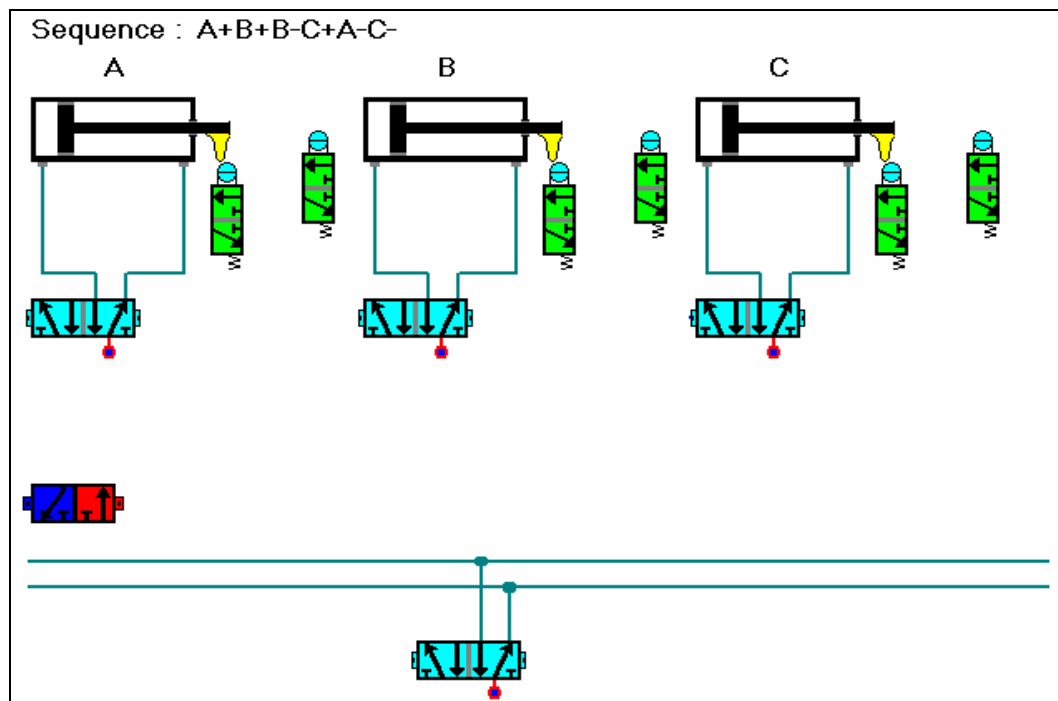


Figure 3.7 Layout of the pneumatic components on the work chart.
At this stage of the program execution, pneumatic components to use are determined but pipeline connections have not been made yet.

The sequence is: [A+ B+ B- C+ A- C-]

Group I is : 'C- A+ B+'

Group II is : 'B- C+ A-'

At the algorithm '*Airsupply*' character "P" is put before the each letters of the sequence representing 'Pilot'. For example, 'P_A+' means pilot valve which is to be actuated by actuator A, when extended and 'P_A-' is the second valve of actuator A

which it actuates when retracted. All the pilot valves in a group are supplied with air from that group line.

Group I is : 'P_C- P_A+ P_B+'

Group II is : 'P_B- P_C+ P_A-'

So 'P_C-', 'P_A+' and 'P_B+' pilot valves are supplied with air from group I line. Pipelines should be connected between inlet ports of these pilot valves and group I line. 'P_B-', 'P_C+' and 'P_A-' pilot valves are supplied with air from group II line. Pipelines should be connected between inlet ports of these pilot valves and group II line. At the computer program these jobs are represented as some numeric codes stored in an array called '**Air**'. Colour coding of the pipelines are also considered. They are represented as green and red dashed lines. In pneumatic circuitry, dashed lines are used for pilot action. Contrary to main power carrying pipelines which are represented by solid lines, pilots do not carry substantial power in them. Figure (3.7) shows the finalised form of the circuit which does the sequence in the given example. Structure of the array '**Air**' is as:

- Air [m , n]

where

m: Code of pilot valves

n: Particular group-no

'P_C-' is supplied with air from group I. This is created at the program by Air [8,1]. The numeral 8 is code of 'P_C-' as shown in Table (3.2). The second numeral 1 is the code of group I.

The second algorithm is for the pipeline connections from the pilot valves to control valves. Position of each actuator is monitored by two pilot valves. This algorithm and related program segment are called '**Pilotsconnect**' at the computer program. At the algorithm of '**Pilotsconnect**' character "P" is put before the first letter of the group and character "V" is put before the second letter of the group

Operation of '**Pilotsconnect**' begins with the first component of that group and repeats until $(G[n] - 1)$ 'th component.

1. Step: $GR[1, 1] = 'C-'$ so; character "P" is put as. 'Pc-' meaning connection from outlet port of the pilot valve
2. Step: $GR[1, 2] = 'A+'$ so; character "V" is put as 'VA+' meaning connection to port name of control valve

Connections between the pilot and control valves are governed by the information stored in the array as:

- $Pil[m, z]$

where

m: Code of pilot valve for connection from.

z: Code of control valve for connection to.

The code in array $Pil[8,1]$. The numeral 8 is code of 'Pc-' as shown in Table (3.2). The second numeral 1 is code of control valve as shown in Table (3.1). These two steps are for one pipeline connection only. Then procedure is repeated for the next component of Group I as:

1. Step $GR[1, 2] = 'A+'$ so; character "P" is put as. 'PA+' meaning connection from outlet port of the pilot valve.
2. Step $GR[1, 3] = 'B+'$ so; character "V" is put as. 'VB+' meaning connection to port name of control valve

Array Pil for this second component hence becomes: $Pil[1,2]$.

This completes the operation of '**Pilotsconnect**' for group I. The same method is repeated for the other groups. Pilot connections are made in blue as seen in Figure (3.7)

The third algorithm carries out pipeline connection from the pilot valves to cascade valves. A pilot valve may actuate a control valve or cascade valve but not both. This algorithm is called '**Cascadechange**' at the computer program. At this algorithm character "P" is put before the last letter of the particular group. This means, pilot valve belongs to the last component in a group. Then this character is added onto the last component of the other groups. Codes of cascade valves are variable dependent upon the number of groups as shown in Table (3.3). In the computer program an array named '**Cas**' is defined to store data for connections. As seen in Figure (3.7) these connections are shown in brown solid lines.

The last component of that group is: $GR [n , G [n]]$.

Where n is the number of that particular.

Connections between the pilot and cascade valves are governed by the information stored in the array as:

- $Cas [m , n]$.

Where,

m: Code of pilot valve belonging to the last component at that group

n: Particular group-no

For all groups excepting the last group are in the form: $Cas [Code (GR[n, G[n]]), n]$

This algorithm repeats operation until the last group (group-no: "Gend-1"). The last group is connected to the cascade valve differently. The last component of the last group is connected to the right side of the first cascade valve, the bottom-most valve on the work chart. The output signals from the other acting pilot valve is connected to the left side of the cascade valve of that group Array '**Cas**' for the final connection is as:

- $Cas [m , 11]$

where

m: pilot valve of the last component in the last group, that is,

Cas [Code (GR[Gend, G[Geng])], 11] This code is only for Group no= Gend.

Table 3.3 Codes of signal ports of cascade valves

Gend:	2 groups		3 groups		4 groups		5 groups	
Cascade direction:	Left	Right	Left	Right	Left	Right	Left	Right
Cascade 1 <i>Topmost</i>	1	11	2		3		4	
Cascade 2			1	11	2		3	
Cascade 3					1	11	2	
Cascade 4 <i>Bottommost</i>							1	11

In the preceding example, ‘Gend’ (number of groups) is 2.

GR [1 , G[1]] is the last letter in group I.

G [1] = 3,

GR [1, 3] = B+ so; with the prefixing of character “P”, ‘PB+’ becomes connection from outlet port of pilot valve actuated by actuator B, when extended.

Cas [code of ‘PB+’, group no] becomes

Cas [2, 1], the code of algorithm at computer program.

GR [2 , G[2]] is the last letter in group II.

G [2] = 3,

GR [2, 3] = ‘A-’ so; with the prefixing of character “P”, ‘PA-’ becomes connection from outlet port of pilot valve actuated by actuator A, when retracted.

Cas [code of ‘PA-’, cascade code of the last group (11)] becomes:

Cas [6, 11] , the code of algorithm at computer program.

The fourth algorithm is for the pipeline connections from the group lines (bus-bars) to control valves. This algorithm is called ‘**ControlValveChange**’ at the computer

program. At this algorithm, character 'V', standing for (Control) Valve, is put before the first letter of that group. Control valve belonging to the first component of a particular group will be actuated by a connection from the particular group line, coded with the appropriate 'V' description. Then the character 'V' is again added before the first component of the other groups. Which are similarly coming directly from their group lines. Codes of signal ports of control valves are shown in Table (3.1). When group line is pressurised with air, the signal port of the control valve directly connected is hence pressurised with air. So direction of the control valve is changed. In cascade circuits, always one group line is connected directly to one port of a control valve and actuates it. The computer program determines these connections in conjunction with some codes it generates. In the computer program codes are stored in a two dimensional array '**Cont**'. In Figure (3.7) red coloured solid lines are generated by this algorithm. It is used as:

- $\text{Cont} [z, n]$.

Where,

- z: code of control valve that belongs to the first component at that group,
- n: Particular group number.

In the preceding example, 'Gend' (number of groups) is 2.

GR [1, 1] is to be the first letter in Group I and hence,

GR [1, 1] = 'C-'. Then character 'V' is prefixed to yield. 'Vc-'. This is the signal which causes actuator C to retract. It comes directly from the group-line I and on the chart, is connected to the right-hand-side boost chamber of the control valve, powering actuator C. Accordingly, the 'Cont' codes are found as:

Cont [code of 'Vc-', group-no] becomes

Cont [8, 1] is code of algorithm at computer program.

This means that signal port of control valve (Vc-) is connected to group-line I.

For the second group in the example,

GR [1, 2] is to be the first letter in Group II and hence,

GR [1, 2] = 'B-'. Then character 'V' is prefixed to yield 'VB-'. This is the signal which causes actuator B to retract. It comes directly from the group-line II and on the chart, is connected to the right-hand-side boost chamber of the control valve powering actuator B. Accordingly the 'Cont' codes are found as:

Cont [code of 'VB-', group no] becomes

Cont [7, 2] is code of algorithm at computer program.

This means that signal port of control valve (VB-) is connected to group-line II.

Pipeline connections between cascade valves are as shown in Figure (3.3). Pipeline connections are complicated indeed. Resulting pneumatic circuit is shown in Figure (3.7). All codes generated for this example are the followings:

Sequence: [A+B+B-C+A-C-]

Air [8, 1], Air [1, 1], Air [2, 1], Air [7, 2], Air [3, 2], Air [6, 2]

Cas [2, 1], Cas [6, 11]

Cont [8, 1], Cont [7, 2]

Pil [8, 1], Pil [1, 2], Pil [7, 3], Pil [3, 6]

These codes are of 'Boolean' type in Delphi programming language. When the codes are correctly generated, they are allocated the 'true' character. The second form, after laying out the pneumatic circuit elements on an empty work chart, draws each pipeline connection as soon as it is defined 'True'. Program '*Pneudesign*' is designed for 5 actuators in its present form, but can be extended to any number of actuators. Increasing the actuator number puts forth a scaling problem, hence to be able to fit onto the computer screen and work chart, components must be drawn smaller. All probable pipeline connections between the components have been drawn at the program memory. Boolean types of all probabilities of pipelines are 'false' at the beginning of the execution. When codes of pipeline connection in Boolean type have been defined 'True' then the program draw them on the work chart. Bellow, is a program segment showing the drawing of the pipeline defined by Pil[8,1] lines determined for code of 'Pil [8, 1]'

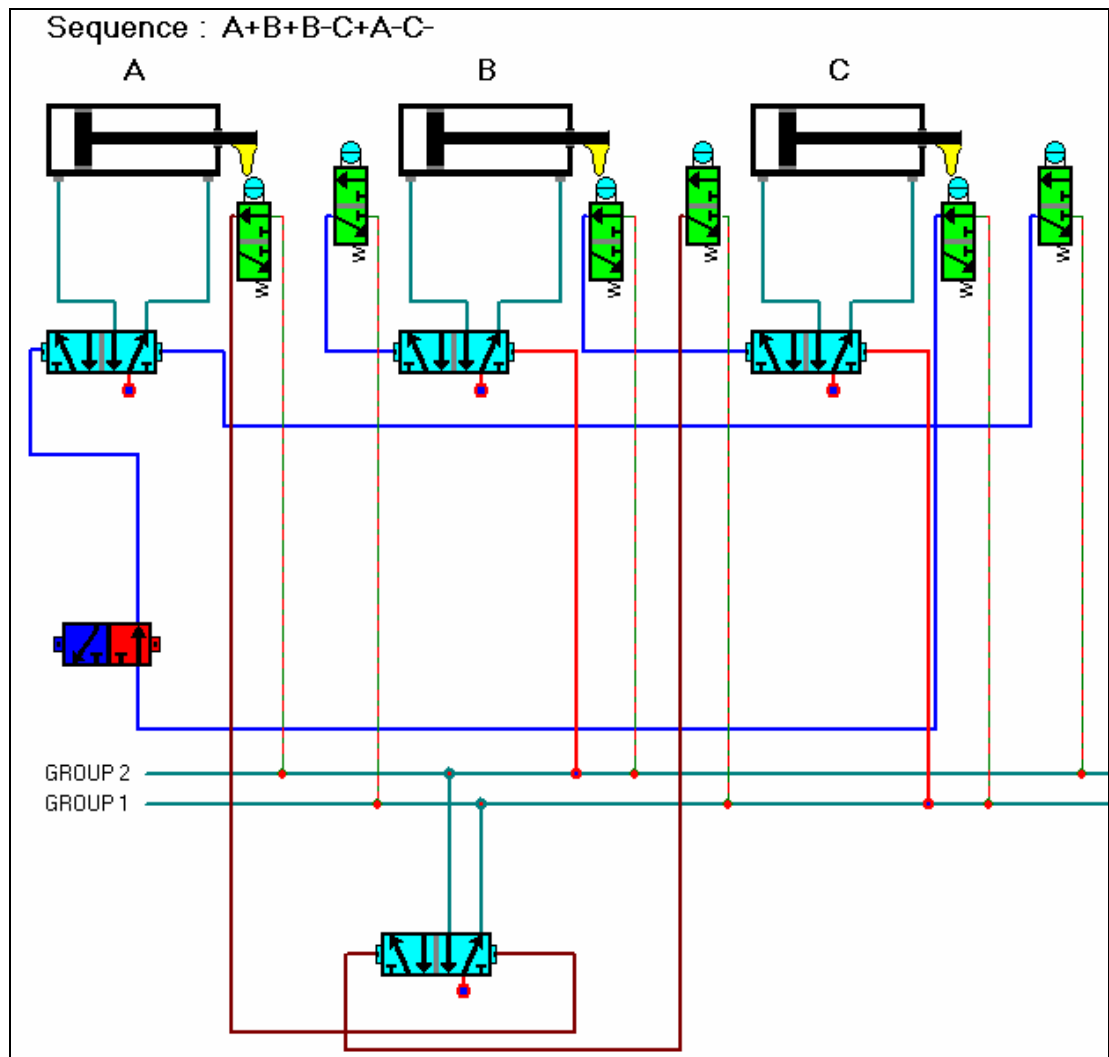


Figure 3.8 Pneumatic circuit of the sequence [A+B+B-C+A-C-]

```

{
    if Pil[8,1]=true then    //if Pil[8,1]=true then draw:
        Canvas.Polyline([point(528,132),Point(521,132),Point(521,420),Point(68,420
        ),Point(68,380)]);
}

```

Numbers in parantheses represent point coordinates in pixels up to which lines are drawn.

CHAPTER 4

EXAMPLES OF PNEUMATIC CIRCUITS

To complete a useful job, generally a machine has to do more than one steps of operations and these operations are seldom done in a sequential manner. For example, in a soft drink filling machine, “filling” operation can not start before a clean bottle comes and similarly a bottle can not be capped before it is filled. So, recycling empty bottles have to be placed on presumably a conveyor, washed, rinsed and dried before going into the filling station. After filled and capped it can go through some more stages where for example, a paper label is stuck and perhaps a date for final usage may be printed. All production activity is composed of a sequence of operations, one carried out after the other. Coordination of these operations must be controlled by a ‘Sequencer’. Sequencers can range from a simple cam to a complicated digital computer in different complexity and flexibility. As complexity and flexibility of a sequencing device increases its cost in the first installation and maintenance increases. Unfortunately devices of higher complexity can malfunction very easily and hence special care is required in their usage. Pneumatic sequencing circuits can be placed somewhere in between the two extremes, can provide a reliable, easy to use and maintain, cheap solution to a sequencing problem. This dissertation has focussed onto the generation of a pneumatic circuit, which powers several actuators in succession, to follow any required order. This is the main task of pneumatic logic. Degree of complexity can be increased by the use of further logic elements as “AND” and “OR” gates, inverters making “NAND” and “NOR” gates, flip flops for data storage. Fluidic, mechanical or electronic time relays can be used to add time gaps between each actuator movement whenever required. Manufacturers of pneumatic elements have added solenoids to actuate their valves, have made electrical switches to replace 3/2 pneumatic valves whenever required such that an operation essentially pneumatic

can be coupled to a computer or a PLC for more flexible and intelligent operations and for operation together with some other electrical systems.

Each mechanical operation is facilitated by an actuator. There are more than one actuators and hence they are distinguished from each other by their names. Actuator names are capital letters starting from A. Actuators are in “bang-bang” operation and hence they either extend or retract. Extension is denoted by a ‘ + ’ sign indicating an increase in length and a retraction with a ‘ – ’ sign, similarly representing a reduction in length.

Designing of cascade circuits is a laborious task and all the text and technology books teaching this subject prefers working on examples rather than putting forth step by step procedures. So they all contain several examples, often in application. For example in the book entitled “Practical Pneumatics” by E.B Paterson [16], three examples are given in chapter 7. Just to give an example, the second problem on page 8 is talking about a machine designed to cut off short lengths of a material with a high speed saw. The feed mechanism is operated by the extension of an air cylinder. When the feed cylinder is fully extended, another cylinder extends, operating a clamp which holds the material firmly. The saw then cuts, being pushed through the cutting stroke by the extension of a cylinder. The cut completed, the saw cylinder retracts, the clamp is released and another cylinder extends to eject the workpiece. The ejector cylinder has to retract before the cycle can be repeated. Analysing the movements, it is found that the main non-productive movement is the retraction of the feed cylinder. In setting up the sequence of movement, this has to be taken into account and this has to be allowed to happen while some other movement is taking place. It will be necessary however, to make sure it has retracted by the end of the cycle so that in feeding in the material in the new cycle it will travel its full stroke. The other non-productive movements can not be absorbed in this way. For example the clamp can not release until the cut is complete and the saw back out of the way. Similarly the ejector retraction must take place before another feed stroke otherwise there would be a mechanical foul-up. Speed of feed and saw forward strokes will need to be controlled. Thus, the sequence to do the job comes up as [A+

B+ C+ C- A- B- D+ D-]. Here A is the feed actuator, B actuates the clamp, C provides the saw feed and D operates the work piece ejector. Similar example applications can be found in the paper of Özcan, Işıl and Kırıcı etitled “Pnömatik Devre Tasarımı” [26] and Power Pneumatics of Pinches and Callear [23] and many others.

In this chapter, various examples are set and given to the program “*Pneudesign*” for solution. Example problems, six in total do not represent a certain practical problem. They are set to demonstrate the ability of the program and are given in ascending level of complexity. The problem sequence, the codes generated for solution in the program and resulting pneumatic circuit diagrams are presented. The circuits had to be checked whether they are operating properly or not. For that, a Festo Didactic Pneumatic Trainer Set is used. Every circuit is patched on the trainer set in the Dynamic Systems Laboratory of the University of Gaziantep/Turkey, and was observed to operate properly without any malfunction. The witness photographs are taken and presented in Appendix A.

Section 4.1 below comprises of two examples with two actuators each. First problem worked on is the sequence [A+ A- B+ B-], two cylinders A and B extending and retracting in succession. This is can be considered a simple example as it contains only two actuators. Second problem is the sequence [A+ B- B+ A-] another possibility of operation with two cylinders. Section 4.2 contains two examples, each containing three actuators and hence are more intricate than those in section 4.1. Third example is the sequence [A+ B- C+ A- C- B+]. This is a more difficult sequence with three actuators and two groups, hence requiring only one cascade valve. Next example in section 4.2 is the sequence [A-B+ C- B- A+ C+], again having a single cascade valve. Section 4.3 presents a single example of the sequence [A+ C+ D- C- D+ A- B+ B-]. Contrary to its complicated appearance this sequence contains only two groups and requires only one cascade valve. Section 4.4 presents the final example containing five actuators, sequence being [A+ C- B+ C+ E+ A- D+ D- E- B-]. This sequence contains three groups and hence requires two cascade valves.

4.1. Examples Containing Two Cylinders

Example 1:

Sequence is “A+A-B+B-”

Program codes are:

Air [7, 1], Air [1, 1], Air [6, 2], Air [2, 2]

Cas [1, 1], Cas [2, 11]

Cont [7, 1], Cont [6, 2]

Pil [7, 1], Pil [6, 2]

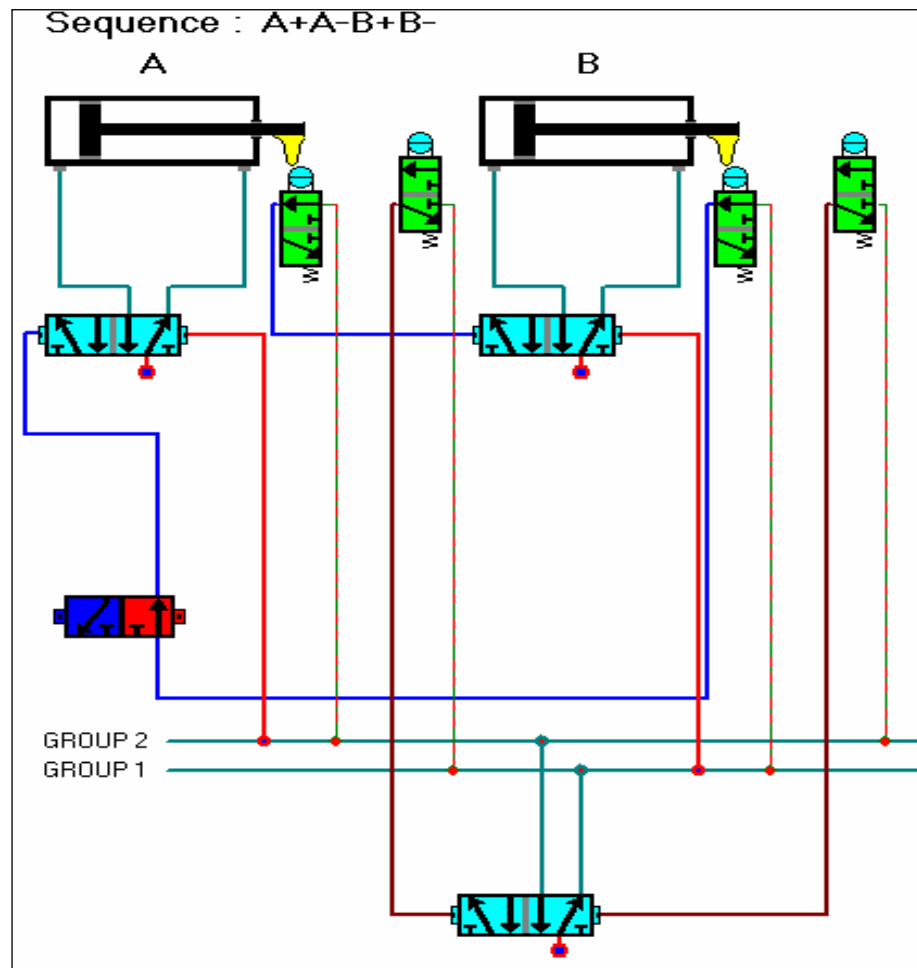


Figure 4.1 Circuit of the sequence [A+A-B+B-] as output from the worksheet of the program “Pneudesign”

Example 2:

Sequence is [A+B-B+A-]

Program codes are:

Air [1, 1], Air [7, 1], Air [2, 2], Air [6, 2] (red-green dashed)

Cas [7, 1], Cas [6, 11] (coloured maroon)

Cont [1, 1], Cont [2, 2] (coloured red)

Pil [1, 7], Pil [2, 6] (coloured blue)

This example is illustrated as shown at Appendix A.

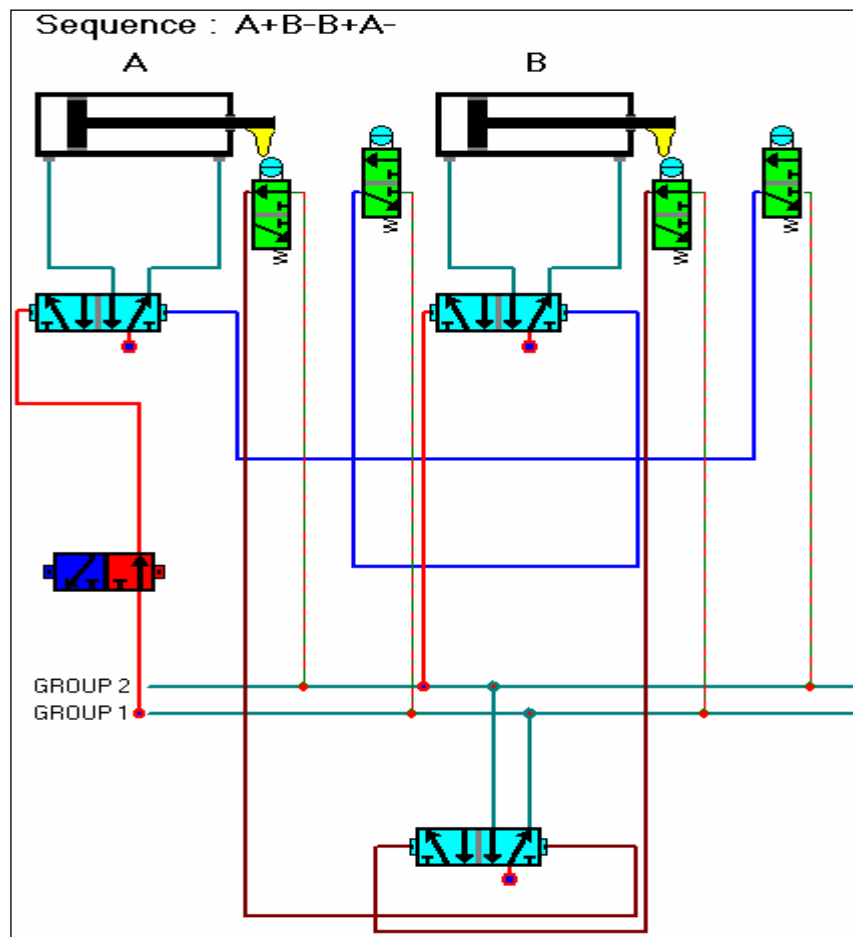


Figure 4.2 Circuit of the sequence [A+B-B+A-] as output from the worksheet of the program “Pneudesign”

4.2. Examples Containing Three Cylinders

Example 3:

Sequence is [A+B-C+A-C-B+]

Program codes are:

Air [1, 1], Air [7, 1], Air [3, 1], Air [6, 2], Air [8, 2], Air [2, 2]

Cas [3, 1], Cas [2, 11]

Cont [1, 1], Cont [6, 2]

Pil [1, 7], Pil [7, 3], Pil [6, 8], Pil [8, 2]

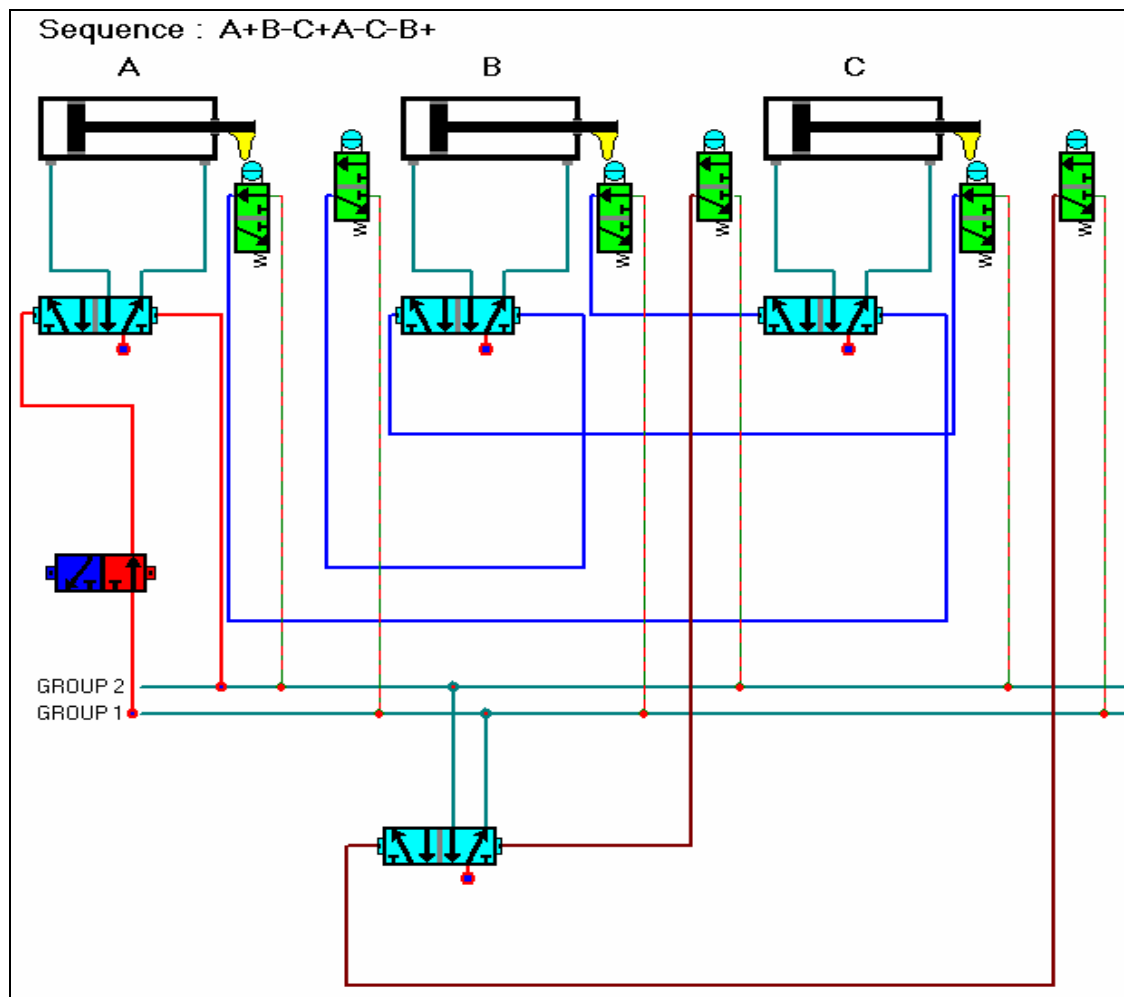


Figure 4.3 Circuit of the sequence [A+B-C+A- C- B+] as output from the worksheet of the program “Pneudesign”

Example 4 :

Sequence is “A-B+C-B-A+C+”

Program codes are:

Air [6, 1], Air [2, 1], Air [8, 1], Air [7, 2], Air [1, 2], Air [3, 2]

Cas [8, 1], Cas [3, 11]

Cont [6, 1], Cont [7, 2]

Pil [6, 2], Pil [2, 8], Pil [7, 1], Pil [1, 3]

This example is illustrated as shown at Appendix A.

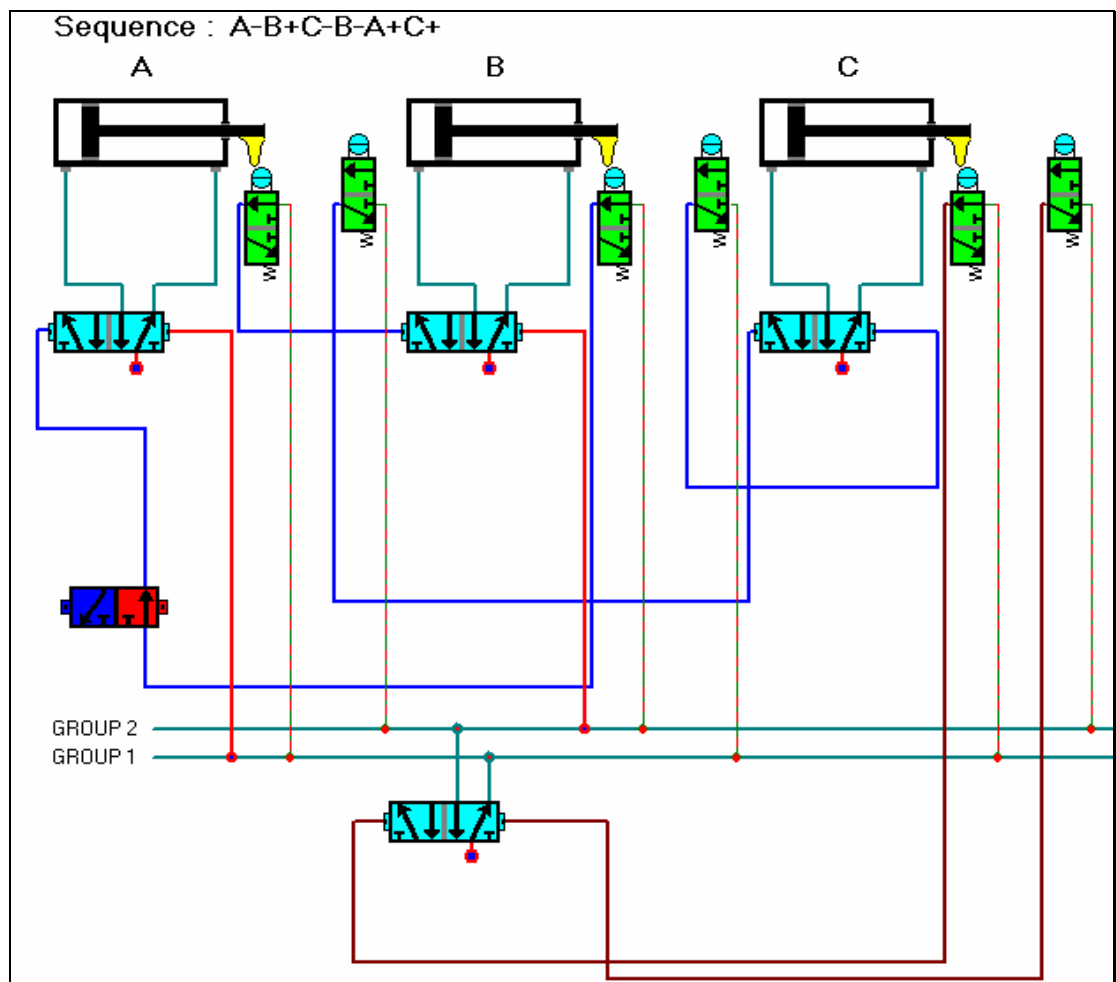


Figure 4. 4 Circuit of the sequence [A- B+ C- B- A+ C+] as output from the worksheet of the program “Pneudesign”

4.3. An Example Containing Four Cylinders

Example 5 :

Sequence is “A+C+D-C-D+A-B+B-”

Program codes are:

Air [7, 1], Air [1, 1], Air [3, 1], Air [9, 1], Air [8, 2], Air [4, 2], Air [6, 2], Air [2, 2]

Cas [9, 1], Cas [2, 11]

Cont [7, 1], Cont [8, 2]

Pil [7, 1], Pil [1, 3], Pil [3, 9], Pil [8, 4], Pil [4, 6], Pil [6, 2]

This example is illustrated as shown at Appendix A.

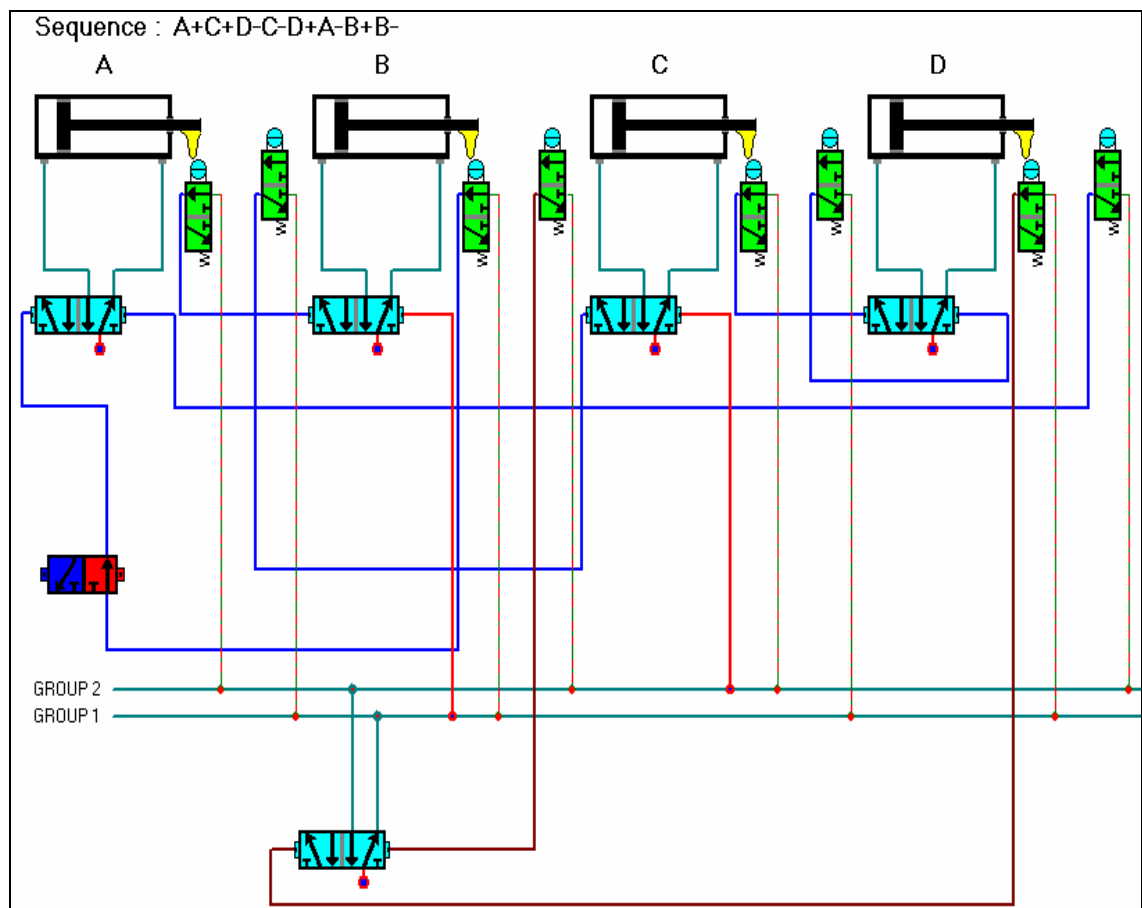


Figure 4.5 Circuit of the sequence [A+C+ D- C- D+ A- B+ B-]
as output from the worksheet of the program “Pneudesign”

4.4. An Example Containing Five Cylinders

Example 5 :

Sequence is “A+C-B+C+E+A-D+D-E-B-”

Program codes are:

Air [1, 1], Air [8, 1], Air [2, 1], Air [3, 2], Air [5, 2], Air [6, 2], Air [4, 2], Air [9, 3]

Air [10, 3], Air [7, 3]

Cas [2, 1], Cas [4, 2], Cas [7, 11]

Cont [1, 1], Cont [3, 2], Cont [9, 3]

Pil [1, 8], Pil [8, 2], Pil [3, 5], Pil [5, 6], Pil [6, 4], Pil [9, 10], Pil [10, 7]

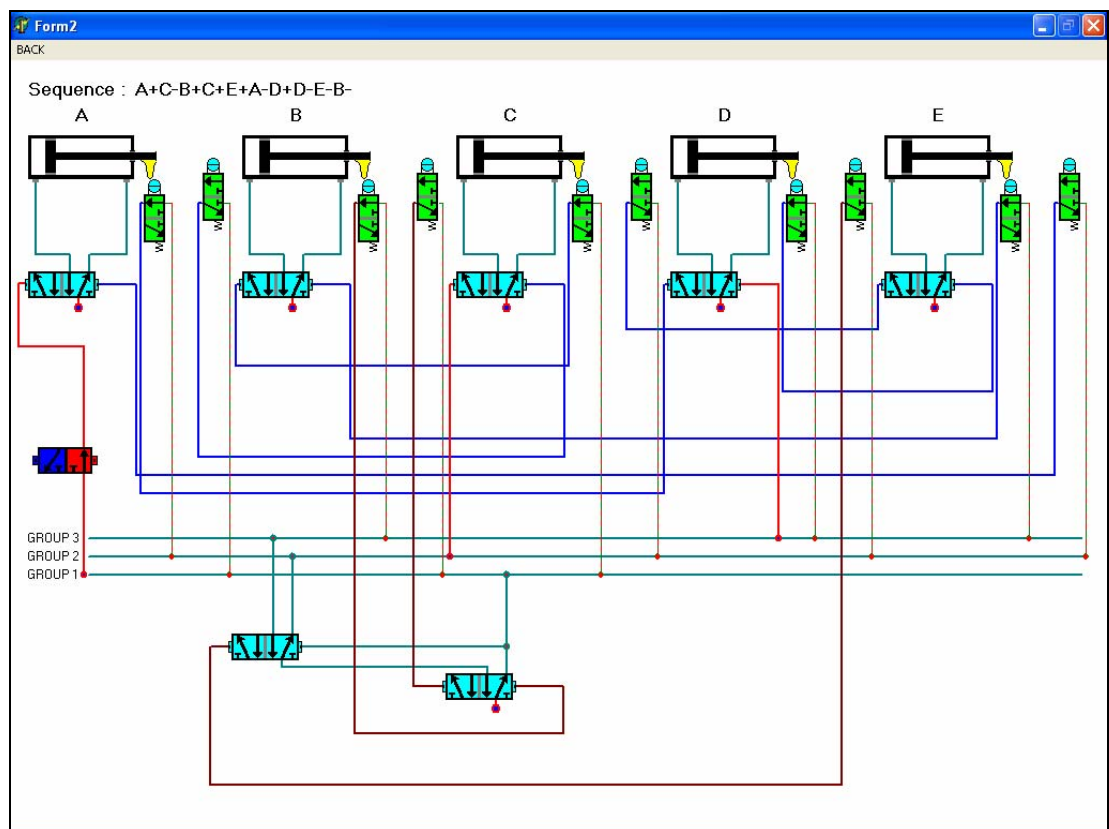


Figure 4. 5 Circuit of the sequence [A+ C- B+ C+ E+ A- D+ D- E- B-] as output from the worksheet of the program “Pneudesign” . Figure shows how the computer screen appears during usage.

CHAPTER 5

DISCUSSION AND CONCLUSIONS

5.1. Discussion of Achievements:

The computer program named “Pneudesign” was tested for correctness many times, by introducing difficult sequencing problems. In all of them it proved to be working correctly. Checking of a resulting circuit by freehand scribble over the circuit is never reliable. One may very easily get lost within the circuit and can not be sure whether the circuit is going to operate correctly or not. So, an experimentation is always necessary. The correction tests of the circuits obtained from test runs of “Pneudesign” have all been tested at the laboratory. Until a circuit not functioning is detected, it must be assumed that all the program portions and all the logic flowing throughout this program is correct. The program is simple to use as explained in the previous chapters. Output program is in form of an ISO standard technical drawing. User friendly interface is an important achievement.

As referred to in earlier chapters, obtaining a sequence of operations in machine automation is simpler and cheaper if done by pneumatic logic, but unfortunately, the methods of designing such circuits are not widely used and not well documented. This is partly due to the widespread use of computer and PLC control. Electronic controllers can perform “IF” operations, by which branching in control sequence can be obtained. Any improper state can be detected and precaution performed. Another very important feature of electronic controllers is the “greater than” operation, by which, it can be detected whether one signal is greater than another in magnitude. Thirdly time can be kept easier in electronic controllers. Another reason of pneumatic logic being not so widespread is that methods of designing a circuit is not known by practising engineers. The use of pneumatic circuits is increasing day by day, especially due to the possibility of obtaining servo motion. A designer may go to electronic control as long as this is justified from technology and cost points of

view. It is never recommended to use electronic control if the same control problem can be solved by using pneumatic sequencing. This program provides a valuable tool to standard designer who does not know how to prepare cascade circuits. Nevertheless the program generated is a distinct addition to the field of Pneumatics as such a design package is not in commercial availability.

One distinct difficulty the writer of this thesis faced has been the programming language. Borland Delphi 7.0 is a high level language unlike standard Fortran or Basic. It is designed over the general purpose languages with the necessity of enabling users not very professional in programming to prepare extensive programs. Languages of this sort are not standard teaching material in engineering schools. Although they provide vast horizons in programming, they must be learned in fluency first.

5.2. Recommendations for Future Work:

Apart from experimental work, validation of a circuit could also be done by using a computer simulation of the experiment rather than physically carrying it out. Such a simulation would be a valuable tool in the verification of the designwork. Development of a simulation package is a big problem in itself and inclusion of it into the work presented here would increase its volume unnecessarily. So this problem was made the subject matter of another MSc thesis carried out at Mustafa Kemal University, Department of Mechanical Engineering [27]. This work is also in form of a computer program to which the user describes a pneumatic circuit by selecting components from menus and doing necessary piping in between. The program then runs the circuit as a real experimental set up and stops if an error arises. One development related with the work presented here could be the inclusion of the simulator program, which will facilitate a test run of the designed circuit, without actually carrying out a physical experiment at laboratory. Secondly, the circuit, and hence the components used can be enriched by further components. “AND” and “OR” gates are available as pneumatic components. Inversion is possible and hence “NAND” and “NOR” gates can be made essentially pneumatic. Data storage is

possible by pneumatic Flip-Flops. Whether the magnitude of a signal is greater than the other or not can be sensed by Differential Pressure Transducers, but as pneumatic circuits have been operating in “bang-bang” manner up to now, what a differential pressure transducer would feel would be a zero or the supply pressure. So use of them similar to a “Greater than” operation has never been done. But, in pneumatic servo circuits, pressure levels are analog now, at any value between zero and the supply pressure, so, practically a “Greater than” operation is available. All such fancy tools can be included in circuits and made them to bear a certain level of intelligence to do decision making and adapting itself to a different operating condition. Such tools can also be included in programs like the one presented in this dissertation, to obtain better design tools.

A complete design page should help the designer in selection of the paper size components. If the components are not selected in matching size, components there will be bottlenecks in timing and amount of power flow through the circuit. The computer program presented in this work can be improved as to give a suggestion in the selection of proper sizes. It could have a list of available component types and sizes and could do a proper selection from it. Many of the companies producing pneumatic components provide software aid programs. These programs are commercially available but they are merely computerised catalogues of their own products and forces the designer to use their products only. Such programs also provide a draughting page by which the user can have circuit diagrams put on paper in ISO standards. If the program described in this thesis should contain a large list of components to select from it would be more helpful.

As indicated in Chapter 1, fluid in motion is assumed of zero mass and in static condition. Although this is correct for all practical purposes, a pneumatic system is actually dynamic and the study of the dynamics of a compressible fluid is very complicated. As pneumatic servos are now available, many people will be working on the dynamic problem. A realistic dynamic simulation of a pneumatic circuit, incorporating as many different components as possible would be an interesting study.

REFERENCES

- [1] Tessmann, R. K. & Hong, T. (1996). Illinois international fluid power exposition and technical conference, Chicago: *The dynamic analysis of pneumatic systems using hypneu*.
- [2] Meritt, H. E. (1997). *Hydraulic control systems*, Library of Congress, Catalogue card no: 66-28759, John Wiley and Sons.
- [3] Grabowski, R. (1997). *The AutoCAD; Programming and customizing quick reference - RI"*, New York: Autodesk Press.
- [4] Elsayy, A., Charoen, V. (1998). *A development of an interface design alternative for pneumatic design circuits*. Department of Industrial Technology University of Northern Iowa.
- [5] Richardson, R., Plummer, A.R. & Brown, M. (2000). *Modelling and simulation of pneumatic cylinders for a physiotherapy robot*. School of Mechanical Engineering, University of Leeds, UK
- [6] Tekiner, Z. & Korkut, İ. (2001). *Bilgisayar destekli pnömatik devre tasarımı* Gazi University Teknik Eğitim Fakültesi Ankara. Tubitak: 25 (1999) , 31- 38.
- [7] Wei, X., Gang, B. & Zuwen, W. (2001). The fifth international conference on fluid power transmission and control, Harbin Institute of Technology, Hangzhou, China: *Study of cad software for pneumatic driving system*.
- [8] Pneumatic Device (2005). *Encyclopaedia Britannica Premium* Article no: 9060481.
- [9] Yeaple, F. D.(1966). *Hydraulic and pneumatic power and control: Design, performance and application*. New York: McGraw-Hill.

[10] Eugenii K. (2005). *The history of electrochemistry*, Electricity and Electronics, Department of Chemistry at the Hebrew University of Jarusalem, Israel, Available as a web site at <http://chem.ch.huji.ac.il>

[11] Cenis Tunnel (2005). *Encyclopaedia Britannica Premium*. Article no: 9054012.

[12] Hoosac Tunnel (2005). *Encyclopaedia Britannica Premium*. Article no: 9068667.

[13] Paris (1977). *Encyclopaedia Americana*. Vol.21, pp 300-317, ISBN 0-7172-0108-2

[14] Festo Company, *Pneumatic main catalogue*. 28th edition Esslingen/Germany
<http://www.festo.com>

[15] Johnson, O. A.(1975). *Fluid power pneumatics*, ISBN 0-82693-675-X. Chicago: American Technical Society.

[16] Paterson, E.B. (1991). *Practical Pneumatics an introduction to low cost automation*. ISBN: 0-07-008303-7. Mc Graw Hill Book Company New Zealand Limited, Auckland.

[17] Pippenger J. & Hicks T. (1980). *Industrial Hydraulics*. (3th ed.). ISBN: 0-070050140-8, Mc Graw Hill Inc.

[18] Halliday, D. & Resnick, R. (1974). *Fundamentals of Physics*. Library of congress catalogue Card Number: 70 102867, John Wiley and Sons Inc., USA.

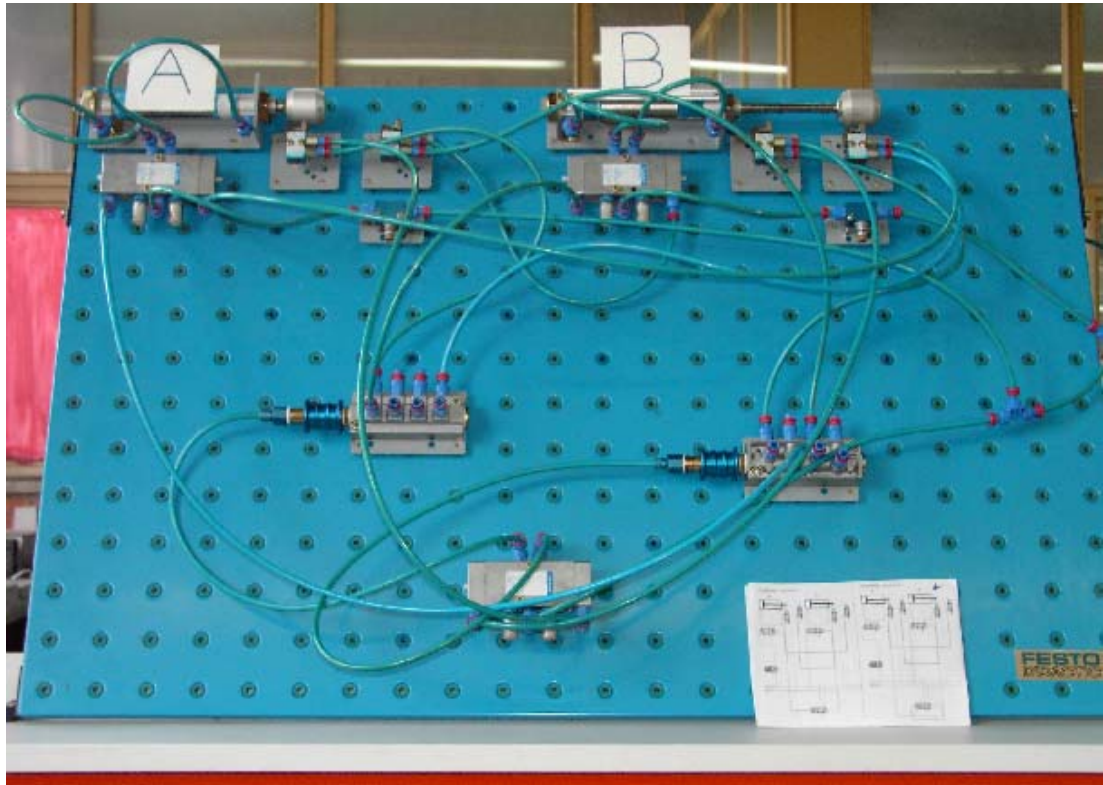
[19] Çengel, Y. A. & Boles, M. A. (1994). *Thermodynamics, an engineering approach*. ISBN: 0-07-113249-X. Mc Graw Hill Inc. NJ, USA.

- [20] Steawart H. L. (1970). *Hydraulic and Pneumatic Power for Production*, Industrial Pres Inc.
- [21] Khaimovich, E. M. (1965). *Hydraulic control of machine tools* (O. Tedder & K. Foster, Trans.). Oxford: Pergamon Press.
- [22] Prisciandaro, J. & Butchko, D *Introduction to Pneumatics and Pneumatic Circuit Problems for FPEF Trainer*. Fluid Power Educational Foundation Birmingham, Michigan. <http://www.fpef.org/>
- [23] Pinches, M. J. & Callear B. J. (1996). *Power Pneumatics*, ISBN: 0-13-48790-0 Prentice Hall Europe.
- [24] Akdoğan, A. & Turgay, R.E. *Otomasyon sistemleri elektropnömatik*, Birsen Yayınevi.
- [25] Karagülle, İ. (2003). *Delphi 7*, ISBN 975-6812-97-4. İstanbul: Türkmen Kitabevi,
- [26] Özcan, F., Işıl, Ş. & Kırıcı, A. (1986). *II. Ulusal Makine Teorisi Sempozyumu* ODTÜ Gaziantep: *Pnömatik devre tasarımı*, pp. 549-585
- [27] Güler, S. (2005). *Pnömatik devrelerin kinematik simülasyonu*, Mustafa Kemal University, MSc. in Mechanical Enginnering

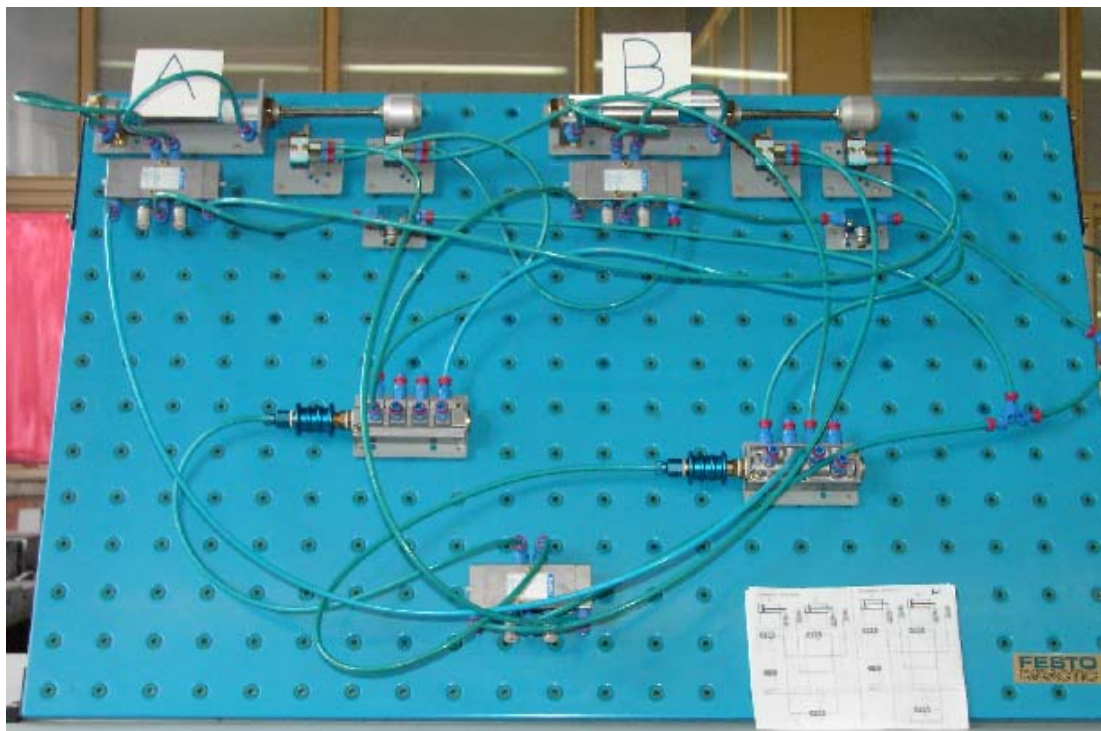
APPENDIX A

Sequence : A+B-B+A-

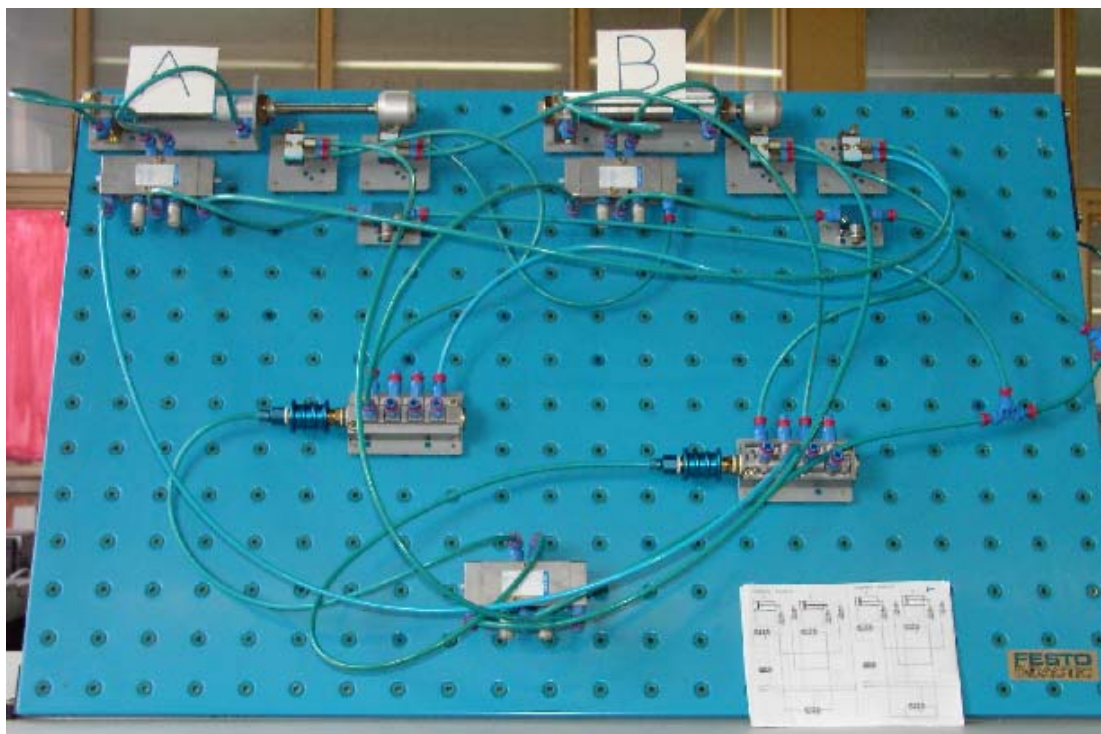
Before starting



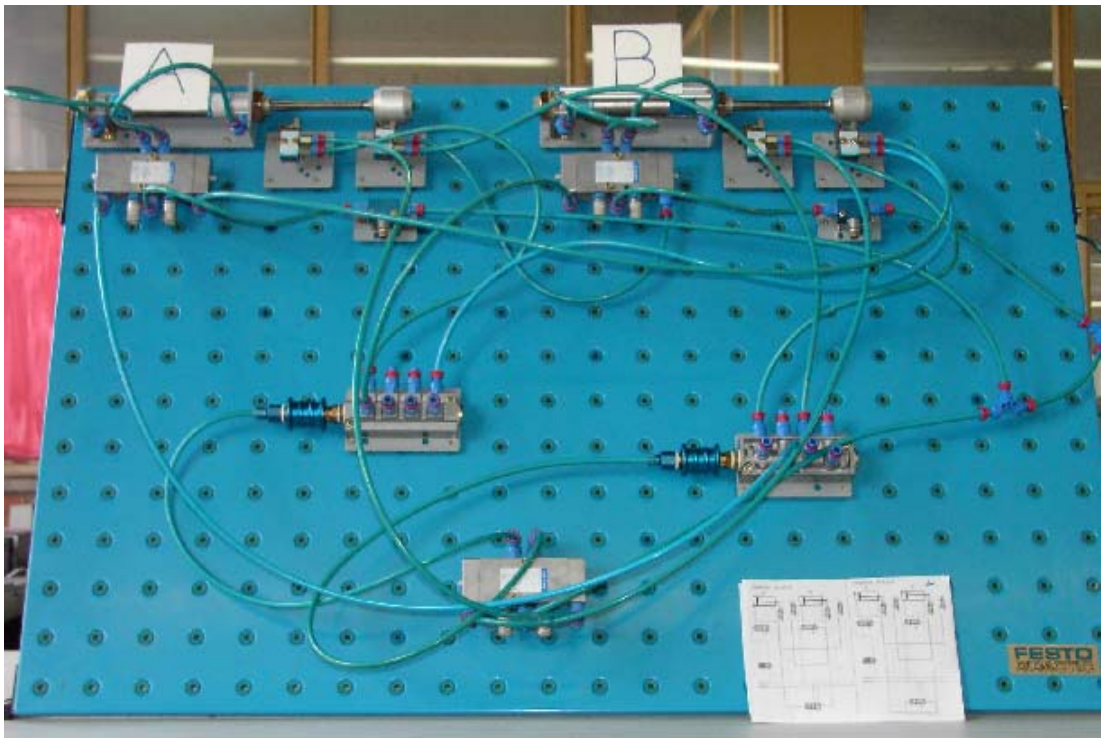
1. Step



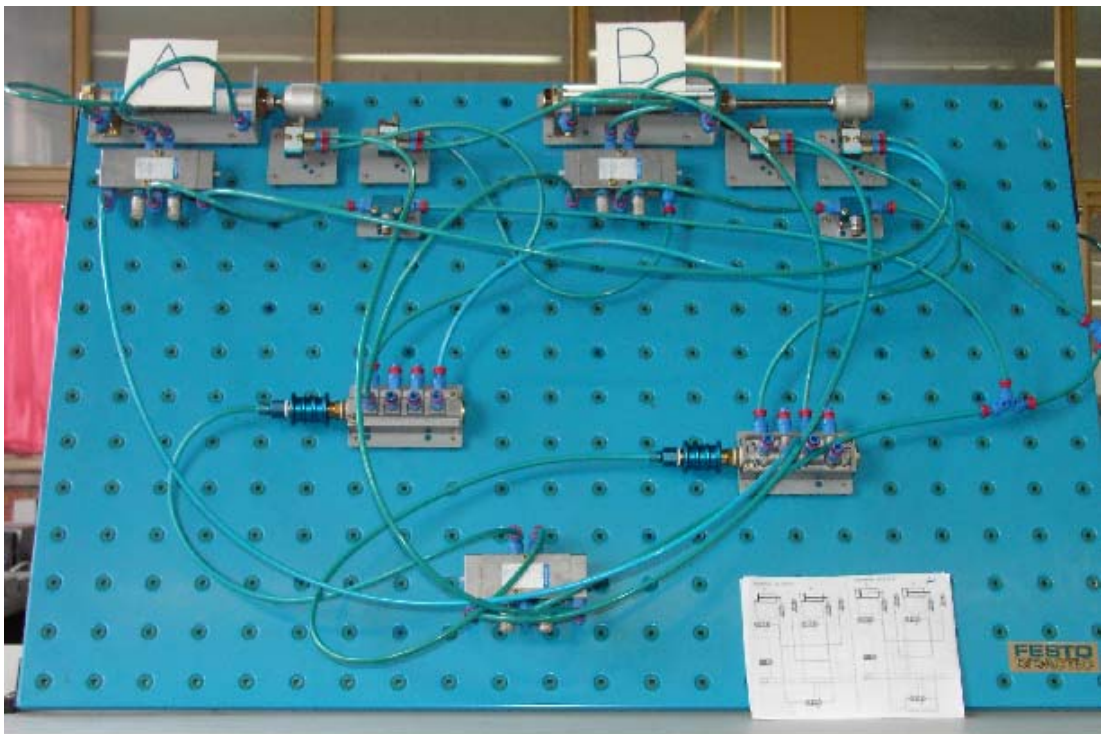
2. Step



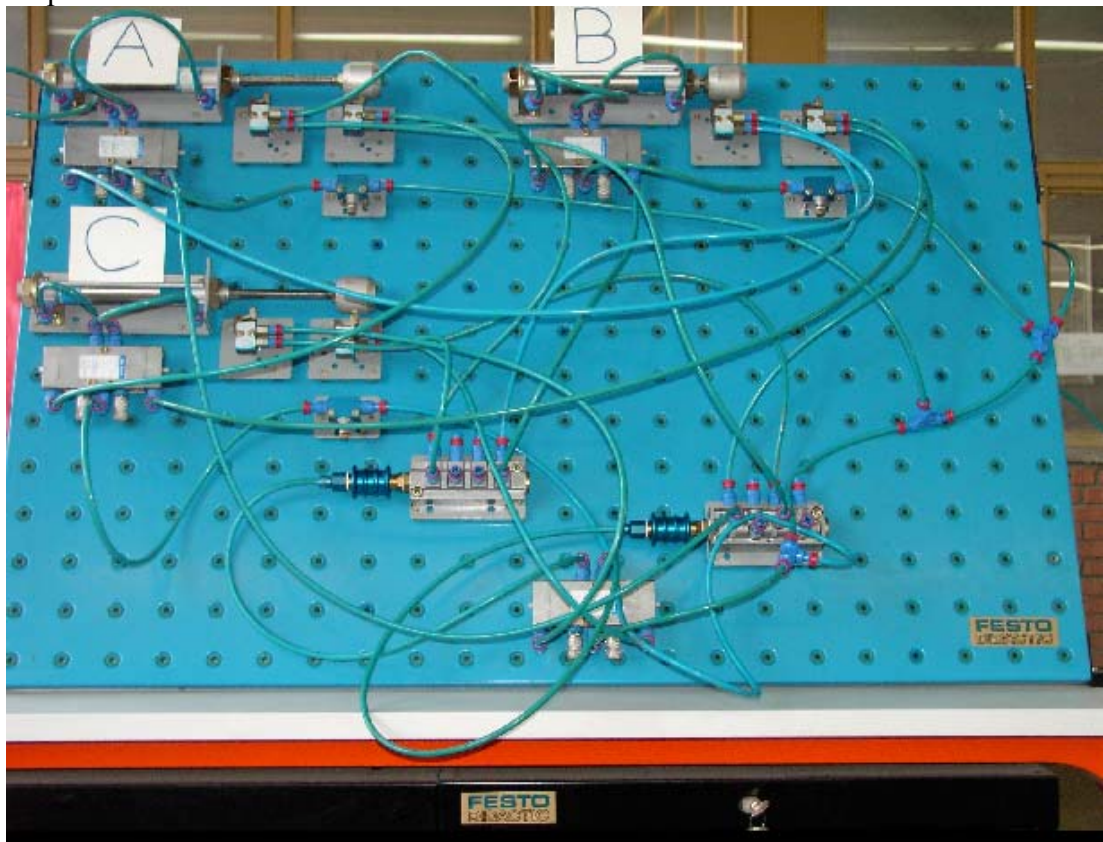
3. Step



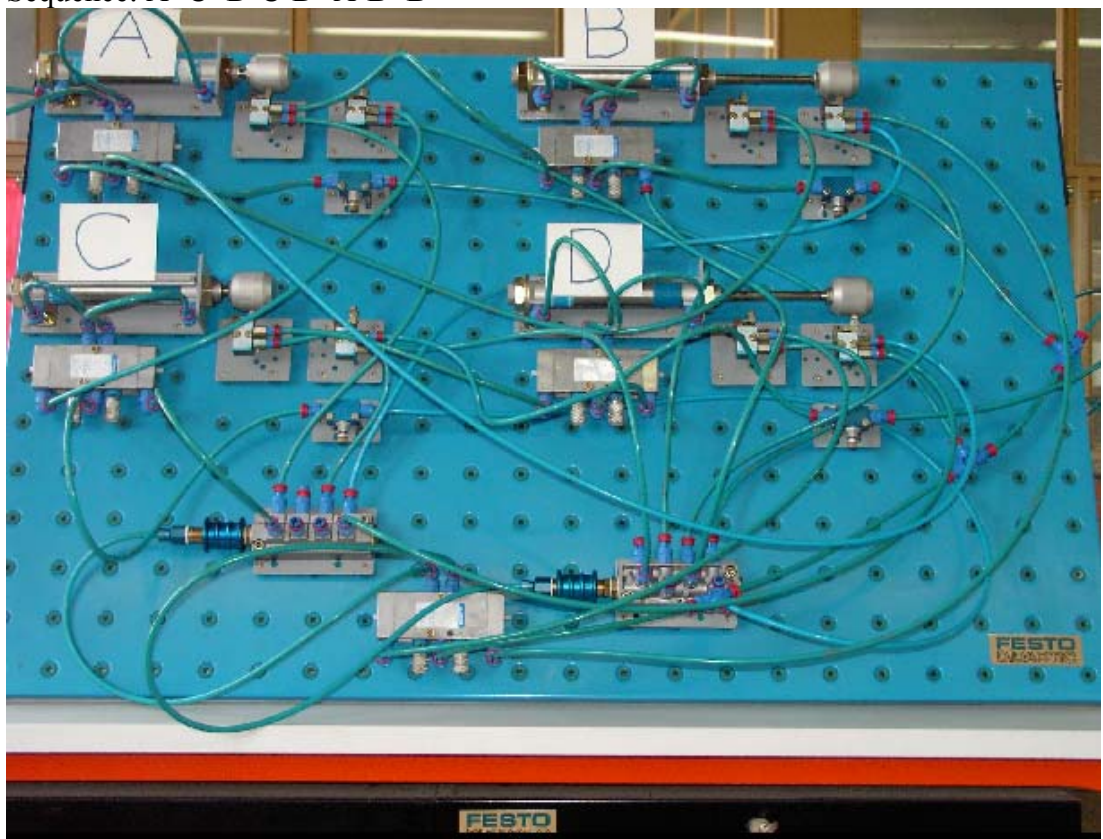
4. Step



Sequence : A-B+C-B-A+C+



Sequence: A+C+D-C-D+A-B+B-



APPENDIX B

```

procedure
FormCreate
begin
Height:=760;
Width:=1000;
Font.Size:=15;
Label1.Caption:='Motion Sequence :';
for i:=1 to 10 do with Step[i] do
begin
X:=(i-1)*67+250;
y:=250;
kombo:=TComboBox.Create(self);
kombo.Parent:=Form1;
kombo.Left:=X;
kombo.Top:=y;
kombo.Width:=70;
kombo.Height:=28;
kombo.Style:=csDropDownList;
kombo.DropDownCount:=10;
kombo.Visible:=False;
end;
end;
Procedure
letterlist
begin
CylinderNumber:=StepNumber div 2;
For i:=1 to StepNumber do with Step[i] do begin
For k:= 1 to CylinderNumber do
Begin
letter:=chr(64+k)+'+';
kombo.Items.Add(Step[i].letter);
letter:=chr(64+k)+'-';
kombo.Items.Add(Step[i].letter);
End;
kombo.Visible:=True;
Steplabel.Visible:=True;
End;
end;
procedure
RadioGroup1Click
begin
if RadioGroup1.ItemIndex=0 then
begin
StepNumber:=4;
For i:=1 to 10 do with Step[i] do begin
kombo.Visible:=False;
Steplabel.Visible:=False;
kombo.Items.Clear;
end;
letterlist(4,2);
end;
if RadioGroup1.ItemIndex=1 then
begin
StepNumber:=6;
For i:=1 to 10 do with Step[i] do begin
kombo.Visible:=False;
Steplabel.Visible:=False;
kombo.Items.Clear;
end;
letterlist(6,3);
end;
if RadioGroup1.ItemIndex=2 then
Begin
StepNumber:=8;
For i:=1 to 10 do with Step[i] do begin
kombo.Visible:=False;
Steplabel.Visible:=False;
kombo.Items.Clear;
end;
letterlist(8,4);
end;
if RadioGroup1.ItemIndex=3 then
begin
StepNumber:=10;
For i:=1 to 10 do with Step[i] do begin
kombo.Visible:=False;
Steplabel.Visible:=False;
kombo.Items.Clear;
end;
letterlist(10,5);
end;
end;
Procedure
Button1Click
Begin
for i:=1 to StepNumber do
begin
for k:=i+1 to StepNumber do
begin
if (Step[k].Kombo.ItemIndex > -1) then
begin
if Step[i].Kombo.Text = Step[k].Kombo.Text
then
begin
Showmessage('CHECK THE LETTERS'
'Do not use the same letter twice');
Step[k].kombo.SetFocus;
end
Else
begin
Seq.Caption:=Step[1].Kombo.Text+Step[2].
Kombo.Text+Step[3].Kombo.Text+
Step[4].Kombo.Text+Step[5].Kombo.Text
+Step[6].Kombo.Text+Step[7].Kombo.Text+
Step[8].Kombo.Text+Step[9].Kombo.Text+
Step[10].kombo.text;
Button2.Enabled:=True;
end;
end
Else
ShowMessage('Please fill all box');

```

```

End;
for i:=1 to 5 do begin
for k:=1 to 5 do
GR[i,k]:="";
end;
i:=0;
while i<StepNumber do begin
splitting[i]:=True;
i:=i+2;
end;
for s:=1 to 6 do G[s]:=0;
for i:=1 to StepNumber do
begin
indeks:=0;
while indeks<StepNumber do
begin
if (Step[i].Kombo.ItemIndex=indeks)
or (Step[i].Kombo.ItemIndex=(indeks+1)) then
if splitting[indeks]=True Then
begin
GR[1,i]:=Step[i].Kombo.Text;
splitting[indeks]:=False;
[1,i]);
G[1]:=i
end
else
begin
Grup2(i);
exit;
end;
indeks:=indeks+2;
end;
end;
end;

```

```

Procedure
Grup2(k:integer);
var s:integer;
begin
i:=0;
while i<StepNumber do begin
splitting[i]:=True;
i:=i+2;
end;
s:=1;
For i:=k to StepNumber do
begin
indeks:=0;
while indeks<StepNumber do
begin
if (Step[i].Kombo.ItemIndex=indeks) or
(Step[i].Kombo.ItemIndex=(indeks+1)) then
if splitting[indeks]=True Then
begin
GR[2,s]:=Step[i].Kombo.Text;
splitting[indeks]:=False;
begin

```

```

G[2]:=S;G[3]:=0; s:=s+1;
end
else
begin
Grup3(i);
exit;
end;
Indeks:=indeks+2;
end;
End;
End;

```

```

Procedure
Grup3(k:integer);
var s:integer;
begin
i:=0;
while i<StepNumber do begin
splitting[i]:=True;
i:=i+2;
end;
s:=1;
For i:=k to StepNumber do
begin
indeks:=0;
while indeks<StepNumber do
begin
if (Step[i].Kombo.ItemIndex=indeks) or
(Step[i].Kombo.ItemIndex=(indeks+1)) then
if splitting[indeks]=True Then
begin
GR[3,s]:=Step[i].Kombo.Text;
splitting[indeks]:=False;
G[3]:=S; G[4]:=0; s:=s+1;
end
else
begin
Grup4(i);
exit;
end;
indeks:=indeks+2;
end;
End;
End;
Procedure TForm1.Grup4(k:integer);
var s:integer;
begin
i:=0;
while i<StepNumber do begin
splitting[i]:=True;
i:=i+2;
end;
s:=1;
For i:=k to StepNumber do
begin
indeks:=0;
while indeks<StepNumber do

```

```

if (Step[i].Kombo.ItemIndex=indeks) or
(Step[i].Kombo.ItemIndex=(indeks+1)) then
if splitting[indeks]=True Then
begin
GR[4,s]:=Step[i].Kombo.Text;
splitting[indeks]:=False;
G[4]:=S; G[5]:=0; s:=s+1;
end
else
begin
Grup5(i);
exit;
end;
indeks:=indeks+2;
end;
End;
End;

```

```

Procedure
Grup5(k:integer);
var s:integer;
begin
i:=0;
while i<StepNumber do begin
splitting[i]:=True;
i:=i+2;
end;
s:=1;
For i:=k to StepNumber do
begin
indeks:=0;
while indeks<StepNumber do
begin
if (Step[i].Kombo.ItemIndex=indeks) or
(Step[i].Kombo.ItemIndex=(indeks+1)) then
if splitting[indeks]=True Then
begin
GR[5,s]:=Step[i].Kombo.Text;
splitting[indeks]:=False;
G[5]:=S; G[6]:=0; s:=s+1;
end
else
begin
GR[6,1]:='E-';
G[6]:=1;
exit;
end;
indeks:=indeks+2;
end;
End;
End;

```

```

Procedure
GrupChange(Sec:integer);
var x:integer;
begin
x:=0;
Form2.show;

```

```

For k:=1 to G[1] do begin
for i:=1 to Sec do begin
if (LeftStr(GR[1,k],1) <> LeftStr(GR[Gend,i],1))
then x:=x+1;
end;
end;
if x=G[1]*Sec then Begin
for x:=G[1] downto 0 do
GR[1,sec+x]:=GR[1,x];
for x:=1 to sec do begin
GR[1,x]:=GR[Gend,x];
GR[Gend,x]:"";
end;
G[1]:=G[1]+Sec;
Gend:=Gend-1;
end;
end;
end;

```

```

procedure
AirSupply;
Begin
for i:=1 to Gend do begin
for s:=1 to G[i] do begin
Air[Code[i,s],i]:=True;
end;
end;
end;

Procedure
CascadeChange;
begin
for i:=1 to Gend-1 do begin
Cas[Code[i,G[i]],i]:=True;
end;
Cas[Code[Gend,G[Gend]],11]:=True;
end;

```

```

Procedure
ControlValfChange;
begin
for i:=1 to Gend do begin
Cont[Code[i,1],i]:=True;
end;
end;
end;

```

```

Procedure
PilotsConnect;
begin;
for i:=1 to Gend do begin
for s:=1 to G[i]-1 do begin
Pil[Code[i,s],Code[i,s+1]]:=True;
end;
end;
end;
end;

```

```

procedure
Button2Click
begin

```

```

end;

GrupChange(G[Gend]);
For i:=1 to 5 do begin
  For s:= 1 to 5 do begin
    For k:= 1 to 5 do begin
      if GR[i,s]=(chr(64+k)+'+') then
        Code[i,s]:=k;
      //Codes: A+:1,B+:2,C+:3,D+:4,E+:5
    end;
  end;
end;
For i:=1 to 5 do begin
  For s:= 1 to 5 do begin
    For k:= 1 to 5 do begin
      if GR[i,s]=(chr(64+k)+'-') then
        Code[i,s]:=k+5; //A-:6,B-:7,C-:8,D-:9,E-:10
    end;
  end;
end;
AirSupply;
CascadeChange;
ControlValfChange;
PilotsConnect;
end;
End.

```

Appendix B is source codes of ‘Sequences’ form.

APPENDIX C

Form-2 ('Draw')

```

Procedure
FormShow
begin
  Cascade1.Top:=534;
  Cascade2.Top:=571;
  Cascade3.Top:=608;
  Cascade4.Top:=645;
  Cascade1.Left:=203;
  Cascade2.Left:=403;
  Cascade3.Left:=603;
  Cascade4.Left:=803;
  Start.Left:=20;
  Start.Top:=360;
  Label3.Top:=438;
  Label4.Top:=455;
  Label5.Top:=472;
  Label6.Top:=489;
  Label7.Top:=504;
  if gend=2 then Begin
    Label3.Caption:='GROUP 2';
    Label4.Caption:='GROUP 1';
  end;
  if gend=3 then Begin
    Label3.Caption:='GROUP 3';
    Label4.Caption:='GROUP 2';
    Label5.Caption:='GROUP 1';
  end;
  if gend=4 then Begin
    Label3.Caption:='GROUP 4';
    Label4.Caption:='GROUP 3';
    Label5.Caption:='GROUP 2';
    Label6.Caption:='GROUP 1';
  end;
  if gend=5 then Begin
    Label3.Caption:='GROUP 5';
    Label4.Caption:='GROUP 4';
    Label5.Caption:='GROUP 3';
    Label6.Caption:='GROUP 2';
    Label7.Caption:='GROUP 1';
  end;
  label1.Caption:=form1.Seq.Caption;
  Cylinder:=StepNumber div 2;
  for i:=1 to Cylinder do
    with Grup[i] do begin
      X:=(i-1)*200+10;
      Y:=-10;
      Visible:=true;
      etiket:=TLabel.Create(self);
      etiket.Parent:=Form2;
      etiket.Left:=X+50;
      etiket.Top:=Y+50;
      etiket.Visible:=Grup[i].Visible;
      etiket.Font.Size:=12;
      etiket.Font.Style:=[fsBold];
      etiket.Caption:=chr(64+i);
      with Silindir do begin
        X1:=X+6;
        Y1:=Y+78;
        hareket:=False;
        Yon:=True;
        resim:=TImage.Create(self);
        resim.Parent:=Form2;
        resim.Left:=X1;
        resim.Top:=Y1;
        resim.Picture.LoadFromFile('Silindir.bmp');
        resim.Transparent:=True;
        resim.Visible:=Grup[i].Visible;
      end;
      with Piston do begin
        X1:=X+21;
        Y1:=Y+78;
        hareket:=False;
        Yon:=True;
        resim:=TImage.Create(self);
        resim.Parent:=form2;
        resim.Left:=X1;
        resim.Top:=Y1;
        resim.Picture.LoadFromFile('Rod.bmp');
        resim.Transparent:=True;
        resim.Visible:=Grup[i].Visible;
      end;
      with Valf do begin
        X1:=X+3;
        Y1:=Y+206;
        hareket:=False;
        Yon:=True;
        resim:=TImage.Create(self);
        resim.Parent:=form2;
        resim.Left:=X1;
        resim.Top:=Y1;
        resim.Picture.LoadFromFile
          ('Valvemavi.bmp');
        resim.Transparent:=True;
        resim.Visible:=Grup[i].Visible;
      end;
      with Pilot1 do begin
        X1:=X+114;
        Y1:=Y+120;
        hareket:=False;
        Yon:=True;
        resim:=TImage.Create(self);
        resim.Parent:=form2;

```

```

resim.Left:=X1;
resim.Top:=Y1;
resim.Picture.LoadFromFile
('Pilot.bmp');
resim.Transparent:=True;
resim.Visible:=Grup[i].Visible;
end;
with Pilot2 do begin
X1:=X+169;
Y1:=Y+100;
hareket:=False;
Yon:=True;
resim:=TImage.Create(self);
resim.Parent:=form2;
resim.Left:=X1;
resim.Top:=Y1;
resim.Picture.LoadFromFile
('Pilot.bmp');
resim.Transparent:=True;
resim.Visible:=Grup[i].Visible;
end;
end;
procedure
FormPaint
var x,y:integer;
begin
Canvas.Pen.Width:=2;
Canvas.Pen.Style:=psSolid;
Canvas.Pen.Color:=clred;
Canvas.Brush.Color:=clblue;
x:=0;
for i:=1 to Cylinder do
begin
x:=(i-1)*200;
Canvas.Polyline
([Point(x+63,220),
Point(x+63,230)]);
Canvas.Ellipse(x+59,226,x+67,234);
MainLine[i]:=true;
MainLine[i+5]:=true;
end;
if gend=5 then begin
Canvas.Polyline([Point(853,664),Point(853,678)]
);
Canvas.Ellipse(849,674,857,682);
end;
if gend=4 then begin
Cascade4.Hide;
/**/ Cascade valf ana hava girişi
Canvas.Polyline([Point(653,627),Point(653,641)]
);
Canvas.Ellipse(649,637,657,645);
end;

if gend=3 then begin
Cascade4.Hide;
Cascade3.Hide;

```

```

Canvas.Polyline
([Point(453,590),Point(453,604)]);
Canvas.Ellipse(449,600,457,608);
end;
if gend=2 then begin
Cascade4.Hide;
Cascade3.Hide;
Cascade2.Hide;
Canvas.Polyline
([Point(253,553),Point(253,567)]);
Canvas.Ellipse(249,563,257,571);
end;

Canvas.Pen.Width:=2;
Canvas.Pen.Style:=psSolid;
Canvas.Pen.Color:=clTeal;
Canvas.Brush.Color:=clred;
// A valf silindir arası
if MainLine[1]=true then
Canvas.Polyline([Point(55,197),Point(55,180),Po
int(23,180),Point(23,108)]);
if MainLine[6]=true then
Canvas.Polyline([Point(73,197),Point(73,180),Po
int(108,180),Point(108,108)]);
// B valf silindir arası
if MainLine[2]=true then //B+
Canvas.Polyline([Point(255,197),Point(255,180),
Point(223,180),Point(223,108)]);
if MainLine[7]=true then // B-
Canvas.Polyline([Point(273,197),Point(273,180),
Point(308,180),Point(308,108)]);
// C valf silindir arası
if MainLine[3]=true then // C+
Canvas.Polyline([Point(455,197),Point(455,180),
Point(423,180),Point(423,108)]);
if MainLine[8]=true then //C-
Canvas.Polyline([Point(473,197),Point(473,180),
Point(508,180),Point(508,108)]);
// D valf silindir arası
if MainLine[4]=true then //D+
Canvas.Polyline([Point(655,197),Point(655,180),
Point(623,180),Point(623,108)]);
if MainLine[9]=true then //D-
Canvas.Polyline([Point(673,197),Point(673,180),
Point(708,180),Point(708,108)]);
// E valf silindir arası
if MainLine[5]=true then //E+
Canvas.Polyline([Point(855,197),Point(855,180),
Point(823,180),Point(823,108)]);
if MainLine[10]=true then //E-
Canvas.Polyline([Point(873,197),Point(873,180),
Point(908,180),Point(908,108)]);
//Cascadeler arası bağlantıları
//alttan 1. Gend:5ise
if Gend >= 5 then begin
Canvas.Polyline([Point(72,513),Point(1000,513)]
); //5. gurup çizgisi
Canvas.Polyline([Point(653,625),Point(653,639),

```

```

Point(845,639),Point(845,645)); //alttan 1.
Gend:5ise
Canvas.Polyline([Point(863,645),Point(863,513)]
);
Canvas.Ellipse(860,510,866,516);
Canvas.Polyline([Point(663,620),Point(863,620)]
);
Canvas.Ellipse(860,617,866,623); end;
//alttan 2. Gend:4ise
if Gend >= 4 then begin
Canvas.Polyline([Point(72,496),Point(1000,496)]
); //4. gurup çizgisi
Canvas.Polyline([Point(453,588),Point(453,602),
Point(645,602),Point(645,608)]); //alttan 2.
Gend:4ise
Canvas.Polyline([Point(663,608),Point(663,496)]
);
Canvas.Ellipse(660,493,666,499);
Canvas.Polyline([Point(463,583),Point(663,583)]
);
Canvas.Ellipse(660,580,666,586); end;
// alttan 3. Gend:3ise
if Gend >= 3 then begin
Canvas.Polyline([Point(72,479),Point(1000,479)]
); //3. gurup çizgisi
Canvas.Polyline([Point(253,551),Point(253,565),
Point(445,565),Point(445,571)]);
Canvas.Polyline([Point(463,571),Point(463,479)]
);
Canvas.Ellipse(460,476,466,482);
Canvas.Polyline([Point(263,546),Point(463,546)]
);
Canvas.Ellipse(460,543,466,549); end;
//alttan 4. Gend:2ise
if Gend >= 2 then begin
Canvas.Polyline([Point(72,462),Point(1000,462)]
); //2. gurup çizgisi
Canvas.Polyline([Point(72,445),Point(1000,445)]
); //1. gurup çizgisi
Canvas.Polyline([Point(263,534),Point(263,462)]
);
Canvas.Ellipse(260,459,266,465);
Canvas.Polyline([Point(245,534),Point(245,445)]
); //son grup'a geçiş
Canvas.Ellipse(242,442,248,448); end;

Canvas.Pen.Style:=psSolid;
Canvas.Pen.Width:=2;
Canvas.Pen.Color:=clBlue;
Canvas.Brush.Color:=clred;
Canvas.Polyline([Point(45,207),Point(7,207),Poi
nt(7,266),Point(68,266),Point(68,360)]);
//PAI
if Pil[6,7]=true then //if PAI2VBI=True then
Canvas.Polyline([point(123,132),Point(121,132),
Point(121,403),Point(317,403),Point(317,208),Po
int(280,208)]);

```

```

if Pil[6,8]=true then //if PAI2VCI=True then
Canvas.Polyline([point(123,132),Point(121,132),
Point(121,403),Point(517,403),Point(517,208),Po
int(480,208)]);
if Pil[6,9]=true then //if PAI2VDI=True then
Canvas.Polyline([point(123,132),Point(121,132),
Point(121,403),Point(717,403),Point(717,208),Po
int(680,208)]);
if Pil[6,10]=true then //if PAI2VEI=True then
Canvas.Polyline([point(123,132),Point(121,132),
Point(121,403),Point(917,403),Point(917,208),Po
int(880,208)]);
if Pil[6,3]=true then //if PAI2VCO=True then
Canvas.Polyline([point(123,132),Point(121,132),
Point(121,403),Point(410,403),Point(410,208),Po
int(445,208)]);
if Pil[6,4]=true then //if PAI2VDO=True then
Canvas.Polyline([point(123,132),Point(121,132),
Point
(121,403),Point(610,403),Point(610,208),Point(6
45,208)]);
if Pil[6,5]=true then //if PAI2VEO=True then
Canvas.Polyline([point(123,132),Point(121,132),
Point(121,403),Point(810,403),Point(810,208),Po
int(845,208)]);
//PBI
if Pil[7,6]=true then // if PBI2VAI=True then
Canvas.Polyline([point(328,132),Point(321,132),
Point(321,335),Point(117,335),Point(117,208),Po
int(80,208)]);
// PBI2VAO VAO KISMINDA DIR
if Pil[7,8]=true then //if PBI2VCI=True then
Canvas.Polyline([point(323,132),Point(321,132),
Point(321,335),Point(517,335),Point(517,208),Po
int(480,208)]);
if Pil[7,9]=true then //if PBI2VDI=True then
Canvas.Polyline([point(323,132),Point(321,132),
Point(321,335),Point(717,335),Point(717,208),Po
int(680,208)]);
if Pil[7,4]=true then //if PBI2VDO=True then
Canvas.Polyline([point(323,132),Point(321,132),
Point(321,335),Point(610,335),Point(610,208),Po
int(645,208)]);
if Pil[7,10]=true then //if PBI2VEI=True then
Canvas.Polyline([point(323,132),Point(321,132),
Point(321,335),Point(917,335),Point(917,208),Po
int(880,208)]);
if Pil[7,5]=true then //if PBI2VEO=True then
Canvas.Polyline([point(323,132),Point(321,132),
Point(321,335),Point(810,335),Point(810,208),Po
int(845,208)]);

//PCI
if Pil[8,6]=true then // if PCI2VAI=True then
Canvas.Polyline([point(523,132),Point(521,132),
Point(521,284),Point(117,284),Point(117,208),Po
int(80,208)]);

```

```
// PCI2VAO VAO KISMINDA DIR
if Pil[8,7]=true then //if PCI2VBI=True then
Canvas.Polyline([point(523,132),Point(521,132),
Point(521,284),Point(317,284),Point(317,208),Po
int(280,208)]);
if Pil[8,2]=true then //if PCI2VBO=True then
Canvas.Polyline([point(523,132),Point(521,132),
Point(521,284),Point(210,284),Point(210,208),Po
int(245,208)]);
if Pil[8,9]=true then //if PCI2VDI=True then
Canvas.Polyline([point(523,132),Point(521,132),
Point(521,284),Point(717,284),Point(717,208),Po
int(680,208)]);
if Pil[8,10]=true then //if PCI2VEI=True then
Canvas.Polyline([point(523,132),Point(521,132),
Point(521,284),Point(917,284),Point(917,208),Po
int(880,208)]);
if Pil[8,5]=true then //if PCI2VEO=True then
Canvas.Polyline([point(523,132),Point(521,132),
Point(521,284),Point(810,284),Point(810,208),Po
int(845,208)]);
```

//PDI

```
if Pil[9,6]=true then // if PDI2VAI=True then
Canvas.Polyline([point(723,132),Point(721,132),
Point(721,308),Point(117,308),Point(117,208),Po
int(80,208)]);
// PDI2VAO VAO KISMINDA DIR
if Pil[9,7]=true then //if PDI2VBI=True then
Canvas.Polyline([point(723,132),Point(721,132),
Point(721,308),Point(317,308),Point(317,208),Po
int(280,208)]);
if Pil[9,2]=true then //if PDI2VBO=True then
Canvas.Polyline([point(723,132),Point(721,132),
Point(721,308),Point(210,308),Point(210,208),Po
int(245,208)]);
if Pil[9,8]=true then //if PDI2VCI=True then
Canvas.Polyline([point(723,132),Point(721,132),
Point(721,308),Point(517,308),Point(517,208),Po
int(480,208)]);
if Pil[9,3]=true then //if PDI2VCO=True then
Canvas.Polyline([point(723,132),Point(721,132),
Point(721,308),Point(410,308),Point(410,208),Po
int(445,208)]);
if Pil[9,10]=true then //if PDI2VEI=True then
Canvas.Polyline([point(723,132),Point(721,132),
Point(721,308),Point(917,308),Point(917,208),Po
int(880,208)]);
```

//PEI

```
if Pil[10,6]=true then // if PEI2VAI=True then
Canvas.Polyline([point(923,132),Point(921,132),
Point(921,352),Point(117,352),Point(117,208),Po
int(80,208)]);
// PEI2VAO VAO KISMINDA DIR
if Pil[10,7]=true then //if PEI2VBI=True then
Canvas.Polyline([point(923,132),Point(921,132),
Point(921,352),Point(317,352),Point(317,208)
```

```
,Point(280,208)]);
if Pil[10,2]=true then //if PEI2VBO=True then
Canvas.Polyline([point(923,132),Point(921,132),
Point(921,352),Point(210,352),Point(210,208),Po
int(245,208)]);
if Pil[10,8]=true then //if PEI2VCI=True then
Canvas.Polyline([point(923,132),Point(921,132),
Point(921,352),Point(517,352),Point(517,208),Po
int(480,208)]);
if Pil[10,3]=true then //if PEI2VCO=True then
Canvas.Polyline([point(923,132),Point(921,132),
Point(921,352),Point(410,352),Point(410,208),Po
int(445,208)]);
if Pil[10,9]=true then //if PEI2VDI=True then
Canvas.Polyline([point(923,132),Point(921,132),
Point(921,352),Point(717,352),Point(717,208),Po
int(680,208)]);
if Pil[10,4]=true then //if PEI2VDO=True then
Canvas.Polyline([point(923,132),Point(921,132),
Point(921,352),Point(610,352),Point(610,208),Po
int(645,208)]);
```

//PAO

```
if Pil[1,7]=true then //if PAO2VBI=True then
Canvas.Polyline([Point(178,132),Point(175,132),
Point(175,369),Point(317,369),Point(317,208),Po
int(280,208)]);
if Pil[1,8]=true then //if PAO2VCI=True then
Canvas.Polyline([point(178,132),Point(175,132),
Point(175,369),Point(517,369),Point(517,208),Po
int(480,208)]);
if Pil[1,9]=true then //if PAO2VDI=True then
Canvas.Polyline([point(178,132),Point(175,132),
Point(175,369),Point(717,369),Point(717,208),Po
int(680,208)]);
if Pil[1,10]=true then //if PAO2VEI=True then
Canvas.Polyline([point(178,132),Point(175,132),
Point(175,369),Point(917,369),Point(917,208),Po
int(880,208)]);
if Pil[1,3]=true then //if PAO2VCO=True
then
Canvas.Polyline([point(178,132),Point(175,132),
Point(175,369),Point(410,369),Point(410,208),Po
int(445,208)]);
if Pil[1,4]=true then //if PAO2VDO=True
then
Canvas.Polyline([point(178,132),Point(175,132),
Point(175,369),Point(610,369),Point(610,208),Po
int(645,208)]);
if Pil[1,5]=true then //if PAO2VEO=True then
Canvas.Polyline([point(178,132),Point(175,132),
Point(175,369),Point(810,369),Point(810,208),Po
int(845,208)]);
```

//PBO

```
if Pil[2,6]=true then // if PBO2VAI=True then
Canvas.Polyline([point(378,132),Point(375,132),
```

```

Point(375,301),Point(117,301),Point(117,208),Po
int(80,208));
// PBO2VAO VAO KISMINDA DIR
if Pil[2,8]=true then //if PBO2VCI=True then
Canvas.Polyline([point(378,132),Point(375,132),
Point(375,301),Point(517,301),Point(517,208),Po
int(480,208)]);
if Pil[2,9]=true then //if PBO2VDI=True then
Canvas.Polyline([point(378,132),Point(375,132),
Point(375,301),Point(717,301),Point(717,208),Po
int(680,208)]);
if Pil[2,4]=true then //if PBO2VDO=True then
Canvas.Polyline([point(378,132),Point(375,132),
Point(375,301),Point(610,301),Point(610,208),Po
int(645,208)]);
if Pil[2,10]=true then //if PBO2VEI=True then
Canvas.Polyline([point(378,132),Point(375,132),
Point(375,301),Point(917,301),Point(917,208),Po
int(880,208)]);
if Pil[2,5]=true then //if PBO2VEO=True
then
Canvas.Polyline([point(378,132),Point(375,132),
Point(375,301),Point(810,301),Point(810,208),Po
int(845,208)]);

```

```

//PCO
if Pil[3,6]=true then // if PCO2VAI=True then
Canvas.Polyline([point(578,132),Point(575,132),
Point(575,250),Point(117,250),Point(117,208),Po
int(80,208)]);
// PCO2VAO VAO KISMINDA DIR
if Pil[3,7]=true then //if PCO2VBI=True then
Canvas.Polyline([point(578,132),Point(575,132),
Point(575,250),Point(317,250),Point(317,208),Po
int(280,208)]);
if Pil[3,2]=true then //if PCO2VBO=True then
Canvas.Polyline([point(578,132),Point(575,132),
Point(575,250),Point(210,250),Point(210,208),Po
int(245,208)]);
if Pil[3,9]=true then //if PCO2VDI=True then
Canvas.Polyline([point(578,132),Point(575,132),
Point(575,250),Point(717,250),Point(717,208),Po
int(680,208)]);
if Pil[3,10]=true then //if PCO2VEI=True then
Canvas.Polyline([point(578,132),Point(575,132),
Point(575,250),Point(917,250),Point(917,208),Po
int(880,208)]);
if Pil[3,5]=true then //if PCO2VEO=True
then
Canvas.Polyline([point(578,132),Point(575,132),
Point(575,250),Point(810,250),Point(810,208),Po
int(845,208)]);

```

```

//PDO
if Pil[4,6]=true then // if PDO2VAI=True then
Canvas.Polyline([point(778,132),Point(775,132),
Point(775,267),Point(117,267),Point(117,208),Po
int(80,208)]);

```

```

// PDO2VAO VAO KISMINDA DIR
if Pil[4,7]=true then //if PDO2VBI=True then
Canvas.Polyline([point(778,132),Point(775,132),
Point(775,267),Point(317,267),Point(317,208),Po
int(280,208)]);
if Pil[4,2]=true then //if PDO2VBO=True then
Canvas.Polyline([point(778,132),Point(775,132),
Point(775,267),Point(210,267),Point(210,208),Po
int(245,208)]);
if Pil[4,8]=true then //if PDO2VCI=True then
Canvas.Polyline([point(778,132),Point(775,132),
Point(775,267),Point(517,267),Point(517,208),Po
int(480,208)]);
if Pil[4,3]=true then //if PDO2VCO=True then
Canvas.Polyline([point(778,132),Point(775,132),
Point(775,267),Point(410,267),Point(410,208),Po
int(445,208)]);
if Pil[4,10]=true then //if PDO2VEI=True then
Canvas.Polyline([point(778,132),Point(775,132),
Point(775,267),Point(917,267),Point(917,208),Po
int(880,208)]);

```

```

//PEO
if Pil[5,6]=true then // if PEO2VAI=True then
Canvas.Polyline([point(978,132),Point(975,132),
Point(975,386),Point(117,386),Point(117,208),Po
int(80,208)]);
// PEO2VAO VAO KISMINDA DIR
if Pil[5,7]=true then //if PEO2VBI=True then
Canvas.Polyline([point(923,132),Point(921,132),
Point(921,386),Point(317,386),Point(317,208),Po
int(280,208)]);
if Pil[5,2]=true then //if PEO2VBO=True then
Canvas.Polyline([point(923,132),Point(921,132),
Point(921,386),Point(210,386),Point(210,208),Po
int(245,208)]);
if Pil[5,8]=true then //if PEO2VCI=True then
Canvas.Polyline([point(923,132),Point(921,132),
Point(921,386),Point(517,386),Point(517,208),Po
int(480,208)]);
if Pil[5,3]=true then //if PEO2VCO=True then
Canvas.Polyline([point(923,132),Point(921,132),
Point(921,386),Point(410,386),Point(410,208),Po
int(445,208)]);
if Pil[5,9]=true then //if PEO2VDI=True then
Canvas.Polyline([point(923,132),Point(921,132),
Point(921,386),Point(717,386),Point(717,208),Po
int(680,208)]);
if Pil[5,4]=true then //if PEO2VDO=True
then
Canvas.Polyline([point(923,132),Point(921,132),
Point(921,386),Point(610,386),Point(610,208),Po
int(645,208)]);
if Pil[6,2]=true then //if PAI2VBO=True then
Canvas.Polyline([point(123,132),Point(121,132),
Point(121,208),Point(245,208)]);
if Pil[1,2]=true then //if PAO2VBO=True then
Canvas.Polyline([point(178,132),Point(175,132),

```

```

Point(175,208),Point(245,208));
if Pil[7,3]=true then //if PBI2VCO=True then
Canvas.Polyline([point(323,132),Point(321,132),
Point(321,208),Point(445,208)]);
if Pil[2,3]=true then //if PBO2VCO=True then
Canvas.Polyline([point(378,132),Point(375,132),
Point(375,208),Point(445,208)]);
if Pil[8,4]=true then //if PCI2VDO=True then
Canvas.Polyline([point(523,132),Point(521,132),
Point(521,208),Point(645,208)]);
if Pil[3,4]=true then //if PCO2VDO=True then
Canvas.Polyline([point(578,132),Point(575,132),
Point(575,208),Point(645,208)]);
if Pil[9,5]=true then //if PDI2VEO=True then
Canvas.Polyline([point(723,132),Point(721,132),
Point(721,208),Point(845,208)]);
if Pil[4,5]=true then //if PDO2VEO=True then
Canvas.Polyline([point(778,132),Point(775,132),
Point(775,208),Point(845,208)]);

```

```

//VAO İLE BGLANTI y=440 dir
if Pil[7,1]=true then //if PBI2VAO=True then
Canvas.Polyline([point(328,132),Point(321,132),
Point(321,420),Point(68,420),Point(68,380)]);
if Pil[2,1]=true then //if PBO2VAO=True then
Canvas.Polyline([point(378,132),Point(375,132),
Point(375,420),Point(68,420),Point(68,380)]);
if Pil[8,1]=true then //if PCI2VAO=True then
Canvas.Polyline([point(528,132),Point(521,132),
Point(521,420),Point(68,420),Point(68,380)]);
if Pil[3,1]=true then //if PCO2VAO=True then
Canvas.Polyline([point(578,132),Point(575,132),
Point(575,420),Point(68,420),Point(68,380)]);
if Pil[9,1]=true then //if PDI2VAO=True then
Canvas.Polyline([point(728,132),Point(721,132),
Point(721,420),Point(68,420),Point(68,380)]);
if Pil[4,1]=true then //if PDO2VAO=True then
Canvas.Polyline([point(778,132),Point(775,132),
Point(775,420),Point(68,420),Point(68,380)]);
if Pil[10,1]=true then //if PEI2VAO=True then
Canvas.Polyline([point(928,132),Point(921,132),
Point(921,420),Point(68,420),Point(68,380)]);
if Pil[5,1]=true then //if PEO2VAO=True then
Canvas.Polyline([point(978,132),Point(975,132),
Point(975,420),Point(68,420),Point(68,380)]);

```

```

Canvas.Pen.Style:=psDashDot;
Canvas.Pen.Width:=1;
Canvas.Pen.Color:=clGreen;
Canvas.Brush.Color:=clred;
if Air[6,Gend-4]=True then begin //PAI1GR1
gend=5 >> GR:1
Canvas.Polyline([point(140,132),Point(150,132),
Point(150,513)]);
Canvas.Ellipse(147,510,153,516); end;
if Air[6,Gend-3]=True then begin //PAI1GR2
Gend=4>>GR:1 Gend=5>>GR:2

```

```

Canvas.Polyline([point(140,132),Point(150,132),
Point(150,496)]);
Canvas.Ellipse(147,493,153,499); end;
if Air[6,Gend-2]=True then begin //PAI1GR3
Gend=3>>GR:1 Gend=4>>GR:2
Gend=5>>GR:3
Canvas.Polyline([point(140,132),Point(150,132),
Point(150,479)]);
Canvas.Ellipse(147,476,153,482); end;
if Air[6,Gend-1]=True then begin //PAI1GR3
Gend=2>>GR:1 Gend=3>>GR:2
Gend=4>>GR:3 Gend=5>>GR:4
Canvas.Polyline([point(140,132),Point(150,132),
Point(150,462)]);
Canvas.Ellipse(147,459,153,465); end;
if Air[6,Gend]=True then begin //PAI1GR3
Gend=2>>GR:2 Gend=3>>GR:3
Gend=4>>GR:4 Gend=5>>GR:5
Canvas.Polyline([point(140,132),Point(150,132),
Point(150,445)]);
Canvas.Ellipse(147,442,153,448); end;
if Air[1,Gend-4]=True then begin //PAO1GR1
Canvas.Polyline([point(197,132),Point(204,132),
Point(204,513)]);
Canvas.Ellipse(201,510,207,516); end;
if Air[1,Gend-3]=True then begin //PAO1GR2
Canvas.Polyline([point(197,132),Point(204,132),
Point(204,496)]);
Canvas.Ellipse(201,493,207,499); end;
if Air[1,Gend-2]=True then begin //PAO1GR3
Canvas.Polyline([point(197,132),Point(204,132),
Point(204,479)]);
Canvas.Ellipse(201,476,207,482); end;
if Air[1,Gend-1]=True then begin //PAO1GR3
Canvas.Polyline([point(197,132),Point(204,132),
Point(204,462)]);
Canvas.Ellipse(201,459,207,465); end;
if Air[1,Gend]=True then begin //PAO1GR3
Canvas.Polyline([point(197,132),Point(204,132),
Point(204,445)]);
Canvas.Ellipse(201,442,207,448); end;

```

```

if Air[7,Gend-4]=True then begin //PBI1GR1
Canvas.Polyline([point(341,132),Point(350,132),
Point(350,513)]);
Canvas.Ellipse(347,510,353,516); end;
if Air[7,Gend-3]=True then begin //PBI1GR2
Canvas.Polyline([point(341,132),Point(350,132),
Point(350,496)]);
Canvas.Ellipse(347,493,353,499); end;
if Air[7,Gend-2]=True then begin //PBI1GR3
Canvas.Polyline([point(341,132),Point(350,132),
Point(350,479)]);
Canvas.Ellipse(347,476,353,482); end;
if Air[7,Gend-1]=True then begin //PBI1GR3
Canvas.Polyline([point(341,132),Point(350,132),
Point(350,462)]);

```

```

Canvas.Ellipse(347,459,353,465); end;
if Air[7,Gend]=True then begin //PBI1GR3
Canvas.Polyline([point(341,132),Point(350,132),
Point(350,445)]);
Canvas.Ellipse(347,442,353,448); end;

if Air[2,Gend-4]=True then begin //PBO1GR1
Canvas.Polyline([point(400,132),Point(403,132),
Point(403,513)]);
Canvas.Ellipse(400,510,406,516); end;
if Air[2,Gend-3]=True then begin //PBO1GR2
Canvas.Polyline([point(400,132),Point(403,132),
Point(403,496)]);
Canvas.Ellipse(400,493,406,499); end;
if Air[2,Gend-2]=True then begin //PBO1GR3
Canvas.Polyline([point(400,132),Point(403,132),
Point(403,479)]);
Canvas.Ellipse(400,476,406,482); end;
if Air[2,Gend-1]=True then begin //PBO1GR3
Canvas.Polyline([point(400,132),Point(403,132),
Point(403,462)]);
Canvas.Ellipse(400,459,406,465); end;
if Air[2,Gend]=True then begin //PBO1GR3
Canvas.Polyline([point(400,132),Point(403,132),
Point(403,445)]);
Canvas.Ellipse(400,442,406,448); end;
Canvas.Ellipse(548,510,554,516); end;
if Air[8,Gend-3]=True then begin //PCI1GR2
Canvas.Polyline([point(541,132),Point(551,132),
Point(551,496)]);
Canvas.Ellipse(548,493,554,499); end;
if Air[8,Gend-2]=True then begin //PCI1GR3
Canvas.Polyline([point(541,132),Point(551,132),
Point(551,479)]);
Canvas.Ellipse(548,476,554,482); end;
if Air[8,Gend-1]=True then begin //PCI1GR3
Canvas.Polyline([point(541,132),Point(551,132),
Point(551,462)]);
Canvas.Ellipse(548,459,554,465); end;
if Air[8,Gend]=True then begin //PCI1GR3
Canvas.Polyline([point(541,132),Point(551,132),
Point(551,445)]);
Canvas.Ellipse(548,442,554,448); end;

if Air[3,Gend-4]=True then begin //PCO1GR1
Canvas.Polyline([point(597,132),Point(604,132),
Point(604,513)]);
Canvas.Ellipse(601,510,607,516); end;
if Air[3,Gend-3]=True then begin //PCO1GR2
Canvas.Polyline([point(597,132),Point(604,132),
Point(604,496)]);
Canvas.Ellipse(601,493,607,499); end;
if Air[3,Gend-2]=True then begin //PCO1GR3
Canvas.Polyline([point(597,132),Point(604,132),
Point(604,479)]);
Canvas.Ellipse(601,476,607,482); end;

```

```

if Air[3,Gend-1]=True then begin //PCO1GR3
Canvas.Polyline([point(597,132),Point(604,132),
Point(604,462)]);
Canvas.Ellipse(601,459,607,465); end;
if Air[3,Gend]=True then begin //PCO1GR3
Canvas.Polyline([point(597,132),Point(604,132),
Point(604,445)]);
Canvas.Ellipse(601,442,607,448); end;
if Air[8,Gend-4]=True then begin //PCI1GR1
Canvas.Polyline([point(541,132),Point(551,132),
Point(551,513)]);

if Air[9,Gend-4]=True then begin //PDI1GR1
Canvas.Polyline([point(741,132),Point(751,132),
Point(751,513)]);
Canvas.Ellipse(748,510,754,516); end;
if Air[9,Gend-3]=True then begin //PDI1GR2
Canvas.Polyline([point(741,132),Point(751,132),
Point(751,496)]);
Canvas.Ellipse(748,493,754,499); end;
if Air[9,Gend-2]=True then begin //PDI1GR3
Canvas.Polyline([point(741,132),Point(751,132),
Point(751,479)]);
Canvas.Ellipse(748,476,754,482); end;
if Air[9,Gend-1]=True then begin //PDI1GR3
Canvas.Polyline([point(741,132),Point(751,132),
Point(751,462)]);
Canvas.Ellipse(748,459,754,465); end;
if Air[9,Gend]=True then begin //PDI1GR3
Canvas.Polyline([point(741,132),Point(751,132),
Point(751,445)]);
Canvas.Ellipse(748,442,754,448); end;

if Air[4,Gend-4]=True then begin //PDO1GR1
Canvas.Polyline([point(797,132),Point(804,132),
Point(804,513)]);
Canvas.Ellipse(801,510,807,516); end;
if Air[4,Gend-3]=True then begin //PDO1GR2
Canvas.Polyline([point(797,132),Point(804,132),
Point(804,496)]);
Canvas.Ellipse(801,493,807,499); end;
if Air[4,Gend-2]=True then begin //PDO1GR3
Canvas.Polyline([point(797,132),Point(804,132),
Point(804,479)]);
Canvas.Ellipse(801,476,807,482); end;
if Air[4,Gend-1]=True then begin //PDO1GR3
Canvas.Polyline([point(797,132),Point(804,132),
Point(804,462)]);
Canvas.Ellipse(801,459,807,465); end;
if Air[4,Gend]=True then begin //PDO1GR3
Canvas.Polyline([point(797,132),Point(804,132),
Point(804,445)]);
Canvas.Ellipse(801,442,807,448); end;
if Air[10,Gend-4]=True then begin //PEI1GR1
Canvas.Polyline([point(941,132),Point(951,132),
Point(951,513)]);
Canvas.Ellipse(948,510,954,516); end;
if Air[10,Gend-3]=True then begin //PEI1GR2

```

```

Canvas.Polyline([point(941,132),Point(951,132),
Point(951,496)]);
Canvas.Ellipse(948,493,954,499); end;
if Air[10,Gend-2]=True then begin // PEI1GR3
Canvas.Polyline([point(941,132),Point(951,132),
Point(951,479)]);
Canvas.Ellipse(948,476,954,482); end;
if Air[10,Gend-1]=True then begin // PEI1GR3
Canvas.Polyline([point(941,132),Point(951,132),
Point(951,462)]);
Canvas.Ellipse(948,459,954,465); end;
if Air[10,Gend]=True then begin // PEI1GR3
Canvas.Polyline([point(941,132),Point(951,132),
Point(951,445)]);
Canvas.Ellipse(948,442,954,448); end;
if Air[5,Gend-4]=True then begin // PEO1GR1
Canvas.Polyline([point(997,132),Point(1004,132),
Point(1004,513)]);
Canvas.Ellipse(1001,510,1007,516); end;
if Air[5,Gend-3]=True then begin // PEO1GR2
Canvas.Polyline([point(997,132),Point(1004,132),
Point(1004,496)]);
Canvas.Ellipse(1001,493,1007,499); end;
if Air[5,Gend-2]=True then begin // PEO1GR3
Canvas.Polyline([point(997,132),Point(1004,132),
Point(1004,479)]);
Canvas.Ellipse(1001,476,1007,482); end;
if Air[5,Gend-1]=True then begin // PEO1GR3
Canvas.Polyline([point(997,132),Point(1004,132),
Point(1004,462)]);
Canvas.Ellipse(1001,459,1007,465); end;
if Air[5,Gend]=True then begin // PEO1GR3
Canvas.Polyline([point(997,132),Point(1004,132),
Point(1004,445)]);
Canvas.Ellipse(1001,442,1007,448); end;

```

/////CONT BAĞLANTILARI/////

```

Canvas.Pen.Style:=psSolid;
Canvas.Pen.Width:=2;
Canvas.Pen.Color:=clred;
Canvas.Brush.Color:=clblue;
// Start-Grup arası bağlantı
if Cont[1,Gend-4] then begin //VAOGR? alttan
1. grup çizgisi
Canvas.Polyline([Point(45,207),Point(7,207),Point(7,266),
Point(68,266),Point(68,360),Point(68,380),Point(68,513)]);
Canvas.Ellipse(65,510,71,516); end;
if Cont[1,Gend-3] then begin //VAOGR? alttan
2. grup çizgisi
Canvas.Polyline([Point(45,207),Point(7,207),Point(7,266),
Point(68,266),Point(68,360),Point(68,380),Point(68,496)]);
Canvas.Ellipse(65,493,71,499); end;
if Cont[1,Gend-2] then begin //VAOGR? alttan
3. grup çizgisi
Canvas.Polyline([Point(45,207),Point(7,207),Point(7,266),
Point(68,266),Point(68,360),Point(68,380),Point(68,479)]);
Canvas.Ellipse(65,476,71,482); end;
if Cont[1,Gend-1] then begin //VAOGR? alttan
4. grup çizgisi
Canvas.Polyline([Point(45,207),Point(7,207),Point(7,266),
Point(68,266),Point(68,360),Point(68,380),Point(68,462)]);
Canvas.Ellipse(65,459,71,465); end;
if Cont[1,Gend] then begin //VAGR? alttan
5. (en üstteki) grup çizgisi
Canvas.Polyline([Point(45,207),Point(7,207),Point(7,266),
Point(68,266),Point(68,360),Point(68,380),Point(68,445)]);
Canvas.Ellipse(65,442,71,448); end;
if Cont[2,Gend-4] then begin //VB+GR? alttan
1. grup çizgisi
Canvas.Polyline([Point(245,208),Point(210,208),Point(210,513)]);
Canvas.Ellipse(207,510,213,516); end;
if Cont[2,Gend-3] then begin //VB+GR? alttan
2. grup çizgisi
Canvas.Polyline([Point(245,208),Point(210,208),Point(210,496)]);
Canvas.Ellipse(207,493,213,499); end;
if Cont[2,Gend-2] then begin //VB+GR? alttan
3. grup çizgisi
Canvas.Polyline([Point(245,208),Point(210,208),Point(210,479)]);
Canvas.Ellipse(207,476,213,482); end;
if Cont[2,Gend-1] then begin //VB+GR? alttan
4. grup çizgisi
Canvas.Polyline([Point(245,208),Point(210,208),Point(210,462)]);
Canvas.Ellipse(207,459,213,465); end;
if Cont[2,Gend] then begin //VB+GR? alttan 5. (en üstteki) grup çizgisi
Canvas.Polyline([Point(245,208),Point(210,208),Point(210,445)]);
Canvas.Ellipse(207,442,213,448); end;
if Cont[3,Gend-4] then begin //VC+GR? alttan
1. grup çizgisi
Canvas.Polyline([Point(445,208),Point(410,208),Point(410,513)]);
Canvas.Ellipse(407,510,413,516); end;
if Cont[3,Gend-3] then begin //VC+GR? alttan
2. grup çizgisi
Canvas.Polyline([Point(445,208),Point(410,208),Point(410,496)]);
Canvas.Ellipse(407,493,413,499); end;
if Cont[3,Gend-2] then begin //VC+GR? alttan
3. grup çizgisi
Canvas.Polyline([Point(445,208),Point(410,208),Point(410,479)]);
Canvas.Ellipse(407,476,413,482); end;
if Cont[3,Gend-1] then begin //VC+GR? alttan
4. grup çizgisi
Canvas.Polyline([Point(445,208),Point(410,208),Point(410,462)]);
Canvas.Ellipse(407,459,413,465); end;
if Cont[3,Gend] then begin //VC+GR? alttan 5. (en üstteki) grup çizgisi
Canvas.Polyline([Point(445,208),Point(410,208),Point(410,445)]);
Canvas.Ellipse(407,442,413,448); end;

```

```

380),Point(68,479)]);
Canvas.Ellipse(65,476,71,482); end;
if Cont[1,Gend-1] then begin //VAOGR? alttan
4. grup çizgisi
Canvas.Polyline([Point(45,207),Point(7,207),Point(7,266),
Point(68,266),Point(68,360),Point(68,380),Point(68,462)]);
Canvas.Ellipse(65,459,71,465); end;
if Cont[1,Gend] then begin //VAGR? alttan
5. (en üstteki) grup çizgisi
Canvas.Polyline([Point(45,207),Point(7,207),Point(7,266),
Point(68,266),Point(68,360),Point(68,380),Point(68,445)]);
Canvas.Ellipse(65,442,71,448); end;
if Cont[2,Gend-4] then begin //VB+GR? alttan
1. grup çizgisi
Canvas.Polyline([Point(245,208),Point(210,208),Point(210,513)]);
Canvas.Ellipse(207,510,213,516); end;
if Cont[2,Gend-3] then begin //VB+GR? alttan
2. grup çizgisi
Canvas.Polyline([Point(245,208),Point(210,208),Point(210,496)]);
Canvas.Ellipse(207,493,213,499); end;
if Cont[2,Gend-2] then begin //VB+GR? alttan
3. grup çizgisi
Canvas.Polyline([Point(245,208),Point(210,208),Point(210,479)]);
Canvas.Ellipse(207,476,213,482); end;
if Cont[2,Gend-1] then begin //VB+GR? alttan
4. grup çizgisi
Canvas.Polyline([Point(245,208),Point(210,208),Point(210,462)]);
Canvas.Ellipse(207,459,213,465); end;
if Cont[2,Gend] then begin //VB+GR? alttan 5. (en üstteki) grup çizgisi
Canvas.Polyline([Point(245,208),Point(210,208),Point(210,445)]);
Canvas.Ellipse(207,442,213,448); end;
if Cont[3,Gend-4] then begin //VC+GR? alttan
1. grup çizgisi
Canvas.Polyline([Point(445,208),Point(410,208),Point(410,513)]);
Canvas.Ellipse(407,510,413,516); end;
if Cont[3,Gend-3] then begin //VC+GR? alttan
2. grup çizgisi
Canvas.Polyline([Point(445,208),Point(410,208),Point(410,496)]);
Canvas.Ellipse(407,493,413,499); end;
if Cont[3,Gend-2] then begin //VC+GR? alttan
3. grup çizgisi
Canvas.Polyline([Point(445,208),Point(410,208),Point(410,479)]);
Canvas.Ellipse(407,476,413,482); end;
if Cont[3,Gend-1] then begin //VC+GR? alttan
4. grup çizgisi
Canvas.Polyline([Point(445,208),Point(410,208),Point(410,462)]);
Canvas.Ellipse(407,459,413,465); end;
if Cont[3,Gend] then begin //VC+GR? alttan 5. (en üstteki) grup çizgisi
Canvas.Polyline([Point(445,208),Point(410,208),Point(410,445)]);
Canvas.Ellipse(407,442,413,448); end;

```

```

4. grup çizgisi
Canvas.Polyline([Point(445,208),Point(410,208),
Point(410,462)]);
Canvas.Ellipse(407,459,413,465); end;
if Cont[3,Gend] then begin //VC+GR? alttan 5.
(en üstteki) grup çizgisi
Canvas.Polyline([Point(445,208),Point(410,208),
Point(410,445)]);
Canvas.Ellipse(407,442,413,448); end;

if Cont[4,Gend-4] then begin //VD+GR? alttan
1. grup çizgisi
Canvas.Polyline([Point(645,208),Point(610,208),
Point(610,513)]);
Canvas.Ellipse(607,510,613,516); end;
if Cont[4,Gend-3] then begin //VD+GR? alttan
2. grup çizgisi
Canvas.Polyline([Point(645,208),Point(610,208),
Point(610,496)]);
Canvas.Ellipse(607,493,613,499); end;
if Cont[4,Gend-2] then begin //VD+GR? alttan
3. grup çizgisi
Canvas.Polyline([Point(645,208),Point(610,208),
Point(610,479)]);
Canvas.Ellipse(607,476,613,482); end;
if Cont[4,Gend-1] then begin //VD+GR? alttan
4. grup çizgisi
Canvas.Polyline([Point(645,208),Point(610,208),
Point(610,462)]);
Canvas.Ellipse(607,459,613,465); end;
if Cont[4,Gend] then begin //VD+GR? alttan 5.
(en üstteki) grup çizgisi
Canvas.Polyline([Point(645,208),Point(610,208),
Point(610,445)]);
Canvas.Ellipse(607,442,613,448); end;
if Cont[5,Gend-4] then begin //VE+GR? alttan
1. grup çizgisi
Canvas.Polyline([Point(845,208),Point(810,208),
Point(810,513)]);
Canvas.Ellipse(807,510,813,516); end;
if Cont[5,Gend-3] then begin //VE+GR? alttan
2. grup çizgisi
Canvas.Polyline([Point(845,208),Point(810,208),
Point(810,496)]);
Canvas.Ellipse(807,493,813,499); end;
if Cont[5,Gend-2] then begin //VE+GR? alttan
3. grup çizgisi
Canvas.Polyline([Point(845,208),Point(810,208),
Point(810,479)]);
Canvas.Ellipse(807,476,813,482); end;
if Cont[5,Gend-1] then begin //VE+GR? alttan
4. grup çizgisi
Canvas.Polyline([Point(845,208),Point(810,208),
Point(810,462)]);
Canvas.Ellipse(807,459,813,465); end;
if Cont[5,Gend] then begin //VE+GR? alttan 5.
(en üstteki) grup çizgisi
Canvas.Polyline([Point(845,208),Point(810,208),
Point(810,445)]);
Canvas.Ellipse(807,442,813,448); end;

if Cont[6,Gend-4] then begin //VA+GR? alttan
1. grup çizgisi
Canvas.Polyline([Point(80,208),Point(117,208),P
oint(117,513)]);
Canvas.Ellipse(114,510,120,516); end;
if Cont[6,Gend-3] then begin //VA+GR? alttan
2. grup çizgisi
Canvas.Polyline([Point(80,208),Point(117,208),P
oint(117,496)]);
Canvas.Ellipse(114,493,120,499); end;
if Cont[6,Gend-2] then begin //VA+GR? alttan
3. grup çizgisi
Canvas.Polyline([Point(80,208),Point(117,208),P
oint(117,479)]);
Canvas.Ellipse(114,476,120,482); end;
if Cont[6,Gend-1] then begin //VA+GR? alttan
4. grup çizgisi
Canvas.Polyline([Point(80,208),Point(117,208),P
oint(117,462)]);
Canvas.Ellipse(114,459,120,465); end;
if Cont[6,Gend] then begin //VA+GR? alttan 5.
(en üstteki) grup çizgisi
Canvas.Polyline([Point(80,208),Point(117,208),P
oint(117,445)]);
Canvas.Ellipse(114,442,120,448); end;
if Cont[7,Gend-4] then begin //VB+GR? alttan
1. grup çizgisi
Canvas.Polyline([Point(280,208),Point(317,208),
Point(317,513)]);
Canvas.Ellipse(314,510,320,516); end;
if Cont[7,Gend-3] then begin //VB+GR? alttan
2. grup çizgisi
Canvas.Polyline([Point(280,208),Point(317,208),
Point(317,496)]);
Canvas.Ellipse(314,493,320,499); end;
if Cont[7,Gend-2] then begin //VB+GR? alttan
3. grup çizgisi
Canvas.Polyline([Point(280,208),Point(317,208),
Point(317,479)]);
Canvas.Ellipse(314,476,320,482); end;
if Cont[7,Gend-1] then begin //VB+GR? alttan
4. grup çizgisi
Canvas.Polyline([Point(280,208),Point(317,208),
Point(317,462)]);
Canvas.Ellipse(314,459,320,465); end;
if Cont[7,Gend] then begin //VB+GR? alttan 5.
(en üstteki) grup çizgisi
Canvas.Polyline([Point(280,208),Point(317,208),
Point(317,445)]);
Canvas.Ellipse(314,442,320,448); end;
if Cont[8,Gend-4] then begin //VC+GR? alttan
1. grup çizgisi
Canvas.Polyline([Point(480,208),Point(517,208),
Point(517,513)]);
Canvas.Ellipse(514,510,520,516); end;

```

```

if Cont[8,Gend-3] then begin //VC-GR? alttan
2. grup çizgisi
Canvas.Polyline([Point(480,208),Point(517,208),
Point(517,496)]);
Canvas.Ellipse(514,493,520,499);end;
if Cont[8,Gend-2] then begin //VC-GR? alttan
3. grup çizgisi
Canvas.Polyline([Point(480,208),Point(517,208),
Point(517,479)]);
Canvas.Ellipse(514,476,520,482);end;
if Cont[8,Gend-1] then begin //VC-GR? alttan
4. grup çizgisi
Canvas.Polyline([Point(480,208),Point(517,208),
Point(517,462)]);
Canvas.Ellipse(514,459,520,465);end;
if Cont[8,Gend] then begin //VC-GR? alttan 5.
(en üstteki) grup çizgisi
Canvas.Polyline([Point(480,208),Point(517,208),
Point(517,445)]);
Canvas.Ellipse(514,442,520,448);end;

if Cont[9,Gend-4] then begin //VD-GR? alttan
1. grup çizgisi
Canvas.Polyline([Point(680,208),Point(717,208),
Point(717,513)]);
Canvas.Ellipse(714,510,720,516); end;
if Cont[9,Gend-3] then begin //VD-GR? alttan
2. grup çizgisi
Canvas.Polyline([Point(680,208),Point(717,208),
Point(717,496)]);
Canvas.Ellipse(714,493,720,499); end;
if Cont[9,Gend-2] then begin //VD-GR? alttan
3. grup çizgisi
Canvas.Polyline([Point(680,208),Point(717,208),
Point(717,479)]);
Canvas.Ellipse(714,476,720,482); end;
if Cont[9,Gend-1] then begin //VD-GR? alttan
4. grup çizgisi
Canvas.Polyline([Point(680,208),Point(717,208),
Point(717,462)]);
Canvas.Ellipse(714,459,720,465); end;
if Cont[9,Gend] then begin //VD-GR? alttan 5.
(en üstteki) grup çizgisi
Canvas.Polyline([Point(680,208),Point(717,208),
Point(717,445)]);
Canvas.Ellipse(714,442,720,448); end;

if Cont[10,Gend-4] then begin //VE-GR? alttan
1. grup çizgisi
Canvas.Polyline([Point(880,208),Point(917,208),
Point(917,513)]);
Canvas.Ellipse(914,510,920,516);end;
if Cont[10,Gend-3] then begin //VE-GR? alttan
2. grup çizgisi
Canvas.Polyline([Point(880,208),Point(917,208),
Point(917,496)]);
Canvas.Ellipse(914,493,920,499); end;
if Cont[10,Gend-2] then begin //VE-GR? alttan
3. grup çizgisi
Canvas.Polyline([Point(880,208),Point(917,208),
Point(917,479)]);
Canvas.Ellipse(914,476,920,482); end;
if Cont[10,Gend-1] then begin //VE-GR? alttan
4. grup çizgisi
Canvas.Polyline([Point(880,208),Point(917,208),
Point(917,462)]);
Canvas.Ellipse(914,459,920,465); end;
if Cont[10,Gend] then begin //VE-GR? alttan 5.
(en üstteki) grup çizgisi
Canvas.Polyline([Point(880,208),Point(917,208),
Point(917,445)]);
Canvas.Ellipse(914,442,920,448); end;

// CASCADE-PİLOT DEĞİŞİM
BAĞLANTILARI
Canvas.Pen.Style:=psSolid;
Canvas.Pen.Width:=2;
Canvas.Pen.Color:=clMaroon;
Canvas.Brush.Color:=clred;

if Cas[1,Gend-1] = true then
Canvas.Polyline([point(178,132),Point(176,132),
Point(176,546),Point(230,546)]);
if Cas[1,Gend-2] = true then
Canvas.Polyline([point(178,132),Point(176,132),
Point(176,583),Point(430,583)]);
if Cas[1,Gend-3] = true then
Canvas.Polyline([point(178,132),Point(176,132),
Point(176,620),Point(630,620)]);
if Cas[1,Gend-4] = true then
Canvas.Polyline([point(178,132),Point(176,132),
Point(176,657),Point(830,657)]);

if Cas[6,Gend-1] = true then
Canvas.Polyline([point(123,132),Point(121,132),
Point(121,546),Point(230,546)]);
if Cas[6,Gend-2] = true then
Canvas.Polyline([point(123,132),Point(121,132),
Point(121,583),Point(430,583)]);
if Cas[6,Gend-3] = true then
Canvas.Polyline([point(123,132),Point(121,132),
Point(121,620),Point(630,620)]);
if Cas[6,Gend-4] = true then
Canvas.Polyline([point(123,132),Point(121,132),
Point(121,657),Point(830,657)]);

if Cas[7,Gend-1] = true then
Canvas.Polyline([point(323,132),Point(321,132),
Point(321,600),Point(186,600),Point(186,546),Po
int(230,546)]);
if Cas[7,Gend-2] = true then
Canvas.Polyline([point(323,132),Point(321,132),
Point(321,583),Point(430,583)]);
if Cas[7,Gend-3] = true then
Canvas.Polyline([point(323,132),Point(321,132),
Point(321,620),Point(630,620)]);
if Cas[7,Gend-4] = true then

```

```

Canvas.Polyline([point(323,132),Point(321,132),
Point(321,657),Point(830,657)]);
if Cas[2,Gend-1] = true then
Canvas.Polyline([point(378,132),Point(376,132),
Point(376,600),Point(186,600),Point(186,546),Po
int(230,546)]);
if Cas[2,Gend-2] = true then
Canvas.Polyline([point(378,132),Point(376,132),
Point(376,583),Point(430,583)]);
if Cas[2,Gend-3] = true then
Canvas.Polyline([point(378,132),Point(376,132),
Point(376,620),Point(630,620)]);
if Cas[2,Gend-4] = true then
Canvas.Polyline([point(378,132),Point(376,132),
Point(376,657),Point(830,657)]);

if Cas[8,Gend-1] = true then
Canvas.Polyline([point(523,132),Point(521,132),
Point(521,630),Point(186,630),Point(186,546),Po
int(230,546)]);
if Cas[8,Gend-2] = true then
Canvas.Polyline([point(523,132),Point(521,132),
Point(521,630),Point(386,630),Point(386,583),Po
int(430,583)]);
if Cas[8,Gend-3] = true then
Canvas.Polyline([point(523,132),Point(521,132),
Point(521,620),Point(630,620)]);
if Cas[8,Gend-4] = true then
Canvas.Polyline([point(523,132),Point(521,132),
Point(521,657),Point(830,657)]);

if Cas[3,Gend-1] = true then
Canvas.Polyline([point(578,132),Point(576,132),
Point(576,635),Point(186,635),Point(186,546),Po
int(230,546)]);
if Cas[3,Gend-2] = true then
Canvas.Polyline([point(578,132),Point(576,132),
Point(576,635),Point(386,635),Point(386,583),Po
int(430,583)]);
if Cas[3,Gend-3] = true then
Canvas.Polyline([point(578,132),Point(576,132),
Point(576,620),Point(630,620)]);
if Cas[3,Gend-4] = true then
Canvas.Polyline([point(578,132),Point(576,132),
Point(576,657),Point(830,657)]);
if Cas[9,Gend-1] = true then
Canvas.Polyline([point(723,132),Point(721,132),
Point(721,670),Point(186,670),Point(186,546),Po
int(230,546)]);
if Cas[9,Gend-2] = true then
Canvas.Polyline([point(723,132),Point(721,132),
Point(721,670),Point(386,670),Point(386,583),Po
int(430,583)]);
if Cas[9,Gend-3] = true then
Canvas.Polyline([point(723,132),Point(721,132),
Point(721,670),Point(586,670),Point(586,620),Po
int(630,620)]);

if Cas[9,Gend-4] = true then
Canvas.Polyline([point(723,132),Point(721,132),
Point(721,657),Point(830,657)]);
if Cas[4,Gend-1] = true then
Canvas.Polyline([point(778,132),Point(776,132),
Point(776,675),Point(186,675),Point(186,546),Po
int(230,546)]);
if Cas[4,Gend-2] = true then
Canvas.Polyline([point(778,132),Point(776,132),
Point(776,675),Point(386,675),Point(386,583),Po
int(430,583)]);
if Cas[4,Gend-3] = true then
Canvas.Polyline([point(778,132),Point(776,132),
Point(776,675),Point(586,675),Point(586,620),Po
int(630,620)]);
if Cas[4,Gend-4] = true then
Canvas.Polyline([point(778,132),Point(776,132),
Point(776,657),Point(830,657)]);

if Cas[10,Gend-1] = true then
Canvas.Polyline([point(923,132),Point(921,132),
Point(921,685),Point(186,685),Point(186,546),Po
int(230,546)]);
if Cas[10,Gend-2] = true then
Canvas.Polyline([point(923,132),Point(921,132),
Point(921,685),Point(386,685),Point(386,583),Po
int(430,583)]);
if Cas[10,Gend-3] = true then
Canvas.Polyline([point(923,132),Point(921,132),
Point(921,685),Point(586,685),Point(586,620),Po
int(630,620)]);
if Cas[10,Gend-4] = true then
Canvas.Polyline([point(923,132),Point(921,132),
Point(921,685),Point(786,685),Point(786,657),Po
int(810,657)]);

if Cas[5,Gend-1] = true then
Canvas.Polyline([point(978,132),Point(976,132),
Point(976,690),Point(186,690),Point(186,546),Po
int(230,546)]);
if Cas[5,Gend-2] = true then
Canvas.Polyline([point(978,132),Point(976,132),
Point(976,690),Point(386,690),Point(386,583),Po
int(430,583)]);
if Cas[5,Gend-3] = true then
Canvas.Polyline([point(978,132),Point(976,132),
Point(976,690),Point(586,690),Point(586,620),Po
int(630,620)]);
if Cas[5,Gend-4] = true then
Canvas.Polyline([point(978,132),Point(976,132),
Point(976,690),Point(786,690),Point(786,657),Po
int(810,657)]);

if Cas[1,Gend+9] = true then
Canvas.Polyline([point(178,132),Point(176,132),
Point(176,590),Point(316,590),Point(316,546),Po
int(270,546)]);
if Cas[1,Gend+8] = true then

```

```

Canvas.Polyline([point(178,132),Point(176,132),
Point(176,627),Point(516,627),Point(516,583),Po
int(470,583)]);
if Cas[1,Gend+7] = true then
Canvas.Polyline([point(178,132),Point(176,132),
Point(176,664),Point(716,664),Point(716,620),Po
int(670,620)]);
if Cas[1,Gend+6] = true then
Canvas.Polyline([point(178,132),Point(176,132),
Point(176,695),Point(916,695),Point(916,657),Po
int(870,657)]);

if Cas[6,Gend+9] = true then
Canvas.Polyline([point(123,132),Point(121,132),
Point(121,590),Point(316,590),Point(316,546),Po
int(270,546)]);
if Cas[6,Gend+8] = true then
Canvas.Polyline([point(123,132),Point(121,132),
Point(121,627),Point(516,627),Point(516,583),Po
int(470,583)]);
if Cas[6,Gend+7] = true then
Canvas.Polyline([point(123,132),Point(121,132),
Point(121,664),Point(716,664),Point(716,620),Po
int(670,620)]);
if Cas[6,Gend+6] = true then
Canvas.Polyline([point(123,132),Point(121,132),
Point(121,695),Point(916,695),Point(916,657),Po
int(870,657)]);

if Cas[2,Gend+9] = true then
Canvas.Polyline([point(378,132),Point(376,132),
Point(376,546),Point(270,546)]);
if Cas[2,Gend+8] = true then
Canvas.Polyline([point(378,132),Point(376,132),
Point(376,627),Point(516,627),Point(516,583),Po
int(470,583)]);
if Cas[2,Gend+7] = true then
Canvas.Polyline([point(378,132),Point(376,132),
Point(376,664),Point(716,664),Point(716,620),Po
int(670,620)]);
if Cas[2,Gend+6] = true then
Canvas.Polyline([point(378,132),Point(376,132),
Point(376,695),Point(916,695),Point(916,657),Po
int(870,657)]);

if Cas[7,Gend+9] = true then
Canvas.Polyline([point(323,132),Point(321,132),
Point(321,546),Point(270,546)]);
if Cas[7,Gend+8] = true then
Canvas.Polyline([point(323,132),Point(321,132),
Point(321,627),Point(516,627),Point(516,583),Po
int(470,583)]);
if Cas[7,Gend+7] = true then
Canvas.Polyline([point(323,132),Point(321,132),
Point(321,664),Point(716,664),Point(716,620),Po
int(670,620)]);
if Cas[7,Gend+6] = true then
Canvas.Polyline([point(323,132),Point(321,132),
Point(321,695),Point(916,695),Point(916,657),Po
int(870,657)]);

if Cas[3,Gend+9] = true then
Canvas.Polyline([point(578,132),Point(576,132),
Point(576,640),Point(330,640),Point(330,546),Po
int(270,546)]);
if Cas[3,Gend+8] = true then
Canvas.Polyline([point(578,132),Point(576,132),
Point(576,583),Point(470,583)]);
if Cas[3,Gend+7] = true then
Canvas.Polyline([point(578,132),Point(576,132),
Point(576,664),Point(716,664),Point(716,620),Po
int(670,620)]);
if Cas[3,Gend+6] = true then
Canvas.Polyline([point(578,132),Point(576,132),
Point(576,695),Point(916,695),Point(916,657),Po
int(870,657)]);

if Cas[8,Gend+9] = true then
Canvas.Polyline([point(523,132),Point(521,132),
Point(521,640),Point(330,640),Point(330,546),Po
int(270,546)]);
if Cas[8,Gend+8] = true then
Canvas.Polyline([point(523,132),Point(521,132),
Point(521,583),Point(470,583)]);
if Cas[8,Gend+7] = true then
Canvas.Polyline([point(523,132),Point(521,132),
Point(521,664),Point(716,664),Point(716,620),Po
int(670,620)]);
if Cas[8,Gend+6] = true then
Canvas.Polyline([point(523,132),Point(521,132),
Point(521,695),Point(916,695),Point(916,657),Po
int(870,657)]);

if Cas[4,Gend+9] = true then
Canvas.Polyline([point(778,132),Point(776,132),
Point(776,546),Point(270,546)]);
if Cas[4,Gend+8] = true then
Canvas.Polyline([point(778,132),Point(776,132),
Point(776,583),Point(470,583)]);
if Cas[4,Gend+7] = true then
Canvas.Polyline([point(778,132),Point(776,132),
Point(776,620),Point(670,620)]);
if Cas[4,Gend+6] = true then
Canvas.Polyline([point(778,132),Point(776,132),
Point(776,695),Point(916,695),Point(916,657),Po
int(870,657)]);

if Cas[9,Gend+9] = true then
Canvas.Polyline([point(723,132),Point(721,132),
Point(721,546),Point(270,546)]);
if Cas[9,Gend+8] = true then
Canvas.Polyline([point(723,132),Point(721,132),
Point(721,583),Point(470,583)]);
if Cas[9,Gend+7] = true then
Canvas.Polyline([point(723,132),Point(721,132),
Point(721,620),Point(670,620)]);

```

```

if Cas[9,Gend+6] = true then
Canvas.Polyline([point(723,132),Point(721,132),
Point(721,695),Point(916,695),Point(916,657),Po
int(870,657)]);

if Cas[5,Gend+9] = true then
Canvas.Polyline([point(978,132),Point(976,132),
Point(976,546),Point(270,546)]);
if Cas[5,Gend+8] = true then
Canvas.Polyline([point(978,132),Point(976,132),
Point(976,583),Point(470,583)]);
if Cas[5,Gend+7] = true then
Canvas.Polyline([point(978,132),Point(976,132),
Point(976,620),Point(670,620)]);
if Cas[5,Gend+6] = true then
Canvas.Polyline([point(978,132),Point(976,132),
Point(976,657),Point(870,657)]);
if Cas[10,Gend+9] = true then
Canvas.Polyline([point(923,132),Point(921,132),
Point(921,546),Point(270,546)]);
if Cas[10,Gend+8] = true then
Canvas.Polyline([point(923,132),Point(921,132),
Point(921,583),Point(470,583)]);
if Cas[10,Gend+7] = true then
Canvas.Polyline([point(923,132),Point(921,132),
Point(921,620),Point(670,620)]);
if Cas[10,Gend+6] = true then
Canvas.Polyline([point(923,132),Point(921,132),
Point(921,657),Point(870,657)]);
end;

```