OPEN LOOP COMPUTER CONTROL SYSTEM-
INTERFACING A SMALL, USER GUIDABLE PROGRAM OPERATED, TRAM TO A PERSONAL COMPUTER.

A Project Report
submitted in partial fulfillment of
the requirements for the award of

BACHELOR OF ENGINEERING.
(Computer Science and Engineering)

Presented By
V.RAMA ARAVIND

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
Dr.Navalar Nedunchezhiyan College of Engineering
Tholudur-606303
2000-2003

This Small Piece of work is
to Adorn the feet of
My ***LORD***
Shri. SHIRIDI SAIBABA
for granting me all that I wished.

ACKNOWLEDGEMENTS

"Every man is capable of doing anything he thinks. Only thing that he should have is self confidence". This has been cultivated in me by A Divine Power. I want to thank my father who spurred me whenever I am sluggish and for inspiring me at every moment. I would like to convey my gratitude to all who helped me in completing this project.

I would like to thank my honourabl;e principal Mr. K. Achutan,B.E(Hons.),Msc(Engg.) who had helped a lot through words. I am very thankful to our respectable Head of the Department Mr. P. Balaji B.E, for helping in each and every aspect.

Besides these celebrities there are many other people who stood with me for support in achieving this small piece of work. I would like to thank Mr.Madhu, M/s. Brilliant systems, Chandralok complex, secunderabad for offering me the hardware equipment in gratis. I would like to convey them my wholehearted veneration and may lord give us power to achieve all that we wish in our life.

V.Rama Aravind.

HYD, Dec 2nd, 2002.

SIZZLERS INSIDE

1. *INTRODUCTION*
2. *OPEN LOOP CONTROL SYSTEMS*
3. *COMPUTERS TO CONTROL SYSTEMS*
4. *TRAM CONSTRUCTION*
5. *PARALLEL PORT & INTERFACING TECHNIQUE*
6. *RELIABILITY OF THE SYSTEM*
7. *HOW TO USE PROGRAM*
8. *SALIENT FEATURES OF THE PROGRAM*
9. *PROGRAM*
10. *FINAL WORDS*
11. *BIBLIOGRAPHY*

INTRODUCTION

At present a myriad of electronic gadgets and computer games woo people to be more technological as never before and with them the daily life of an ordinary man has been filled with ample gay and it would not be a metaphor to say that some kids, today, might even bring down the whole World Wide Web! as the technological perception has grown to very high standards in every one's mind. Now, as our part we should endeavour to add our drop in that technology ocean.

This is the project work, which will be on a whole aimed at interfacing a small, user guidable program operated, tram to a personal computer and is controlled through a program designed in C.

It (tram) can be genuinely designed, so that, it can be marketed aiming industrial, domestic and entertainment fields. Although emphasis is on designing an interface circuitry to study "control systems", by proper fostering, it can be made to woo the market as it stands in its way distinctly. Because, for example, children till now may have had played computer games by having only the images on the computer screen. Now it might be fascinating for them to play with a robot like device that commutes as they control it through keyboard.

Like wise, industrial people can have its advantage by programming it to perform continuous commuting of goods or any such commodities, which will save, labour cost and maximizes profits.

Similarly, domestically it can be used for various types of work like "bring & go" which may assist home maids and others in need.

Considering this project from technical point of view, keeping aside the general considerations, it describes the technique of interfacing any general-purpose device to the Personal Computer.

You might have known that a goal can be achieved in many ways and in This too, the interfacing technique can be achieved in many ways. It depends on the way we choose, for minimizing the cost and achieving what we wish.

Here in this project I chose the technique which is wholly hand made that costs least and achieves the desired objective.

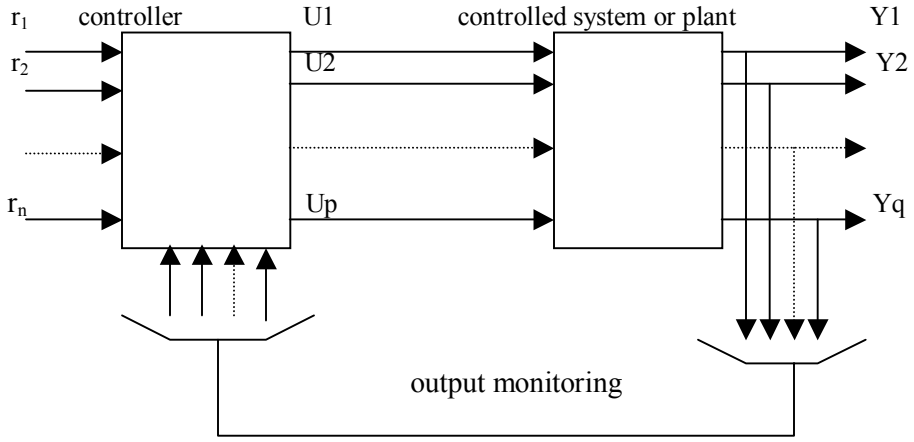
The sections that follow depicts and describes both interfacing techniques and the explanation for constructing the device as well as provide a better information on how to operate the device.

I hope this report helps you in finding at least a little bit of information, which might assist you in designing more complex systems and if it assists you in any of your works, it is a success to me.

OPEN LOOP CONTROL SYSTEMS

Over the past two decades, Modern Control System theory has been gaining great importance, for being potentially applicable to an increasing number of widely different disciplines of human activity. Basically, the Modern Control System theory has involved the study of analysis and control of any dynamical system – whether engineering, economic, managerial, medical, social, or even political. This theory first gained considerable maturity in the discipline of engineering, particularly receiving great impetus from aerospace engineering.

In fact, the development of physical models (in this case the tram) of physical systems (actual control of a real world object) requires knowledge of each specific field. We here consider the physical models of real physical systems as systems themselves. And therefore, in our terminology, *a physical system is a real object or collection of such objects in the real world; and a 'system' is a physical model of the real- world system resembling it in certain salient features.*



General structure of an open loop control system.

In principle, it is possible to change the outputs of a system in any prescribed fashion (at least with in reasonable limits) by means of intelligent manipulation of its inputs. This then, in a general case, constitutes a *controlled system*.

It is always useful to determine various structural solutions of carrying out the system requirements and objectives at the initial stage itself. The more solutions initially considered, the greater is the probability of success in the final system. The optimum solution is selected on the basis of the functional and hardware requirements of the system.

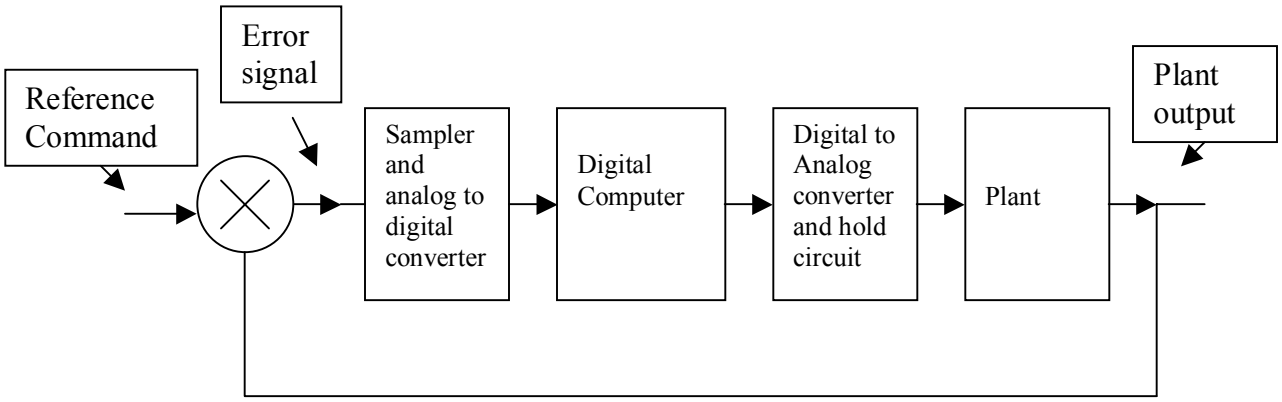
The general structure of a *multiple-input/multiple-output* (multivariable) control system is shown in previous page. The *plant* is that part of the system which is to be controlled. It is generally a fixed component of the system whose physical characteristics are beyond control. The output of the plant is measured by *q* variables Y1(t), Y2(t), …,Yq(t) whose values give an indication of plant performance. Direct control of the plant is exerted by means of P control forces U1(t),…,Up(t). These forces are applied by some controlling device, called *controller* which determines proper control action based upon the reference *commands* R1(t), …,Rq(t) and information obtained via output sensors concerning the actual output. The feed back of output information results in a closed-loop signal flow and the term *closed-loop control* is often used for such a control action. In *OPEN LOOP CONTROL SYSTEMS,* the controller operates on some pre-set pattern without taking account of the outputs.

The plant accepts continuous-time signals as inputs and gives out continuous-time signals as outputs. If the controller elements are such that the controller produces continuous-time control signals from continuous-time input signals (analog controller), then the overall control system is a *continuous-time system*, where in the signal at every point is a continuous function of time.

COMPUTERS TO CONTROL SYSTEMS

The complexity of the controller needed to implement a control law is a function of the plant and the stringency of the control requirements. The cost of an analog controller rises steeply with increasing control function complexity. In fact, implementing a complex control function may even become technically infeasible if one is restricted to use only analog elements. A *digital controller,* in which either a special purpose or a general-purpose computer forms the heart is usually an ideal choice for complex control systems. A general-purpose computer, if used, lends itself to time-shared use for other control functions in the plant or process. A digital controller also has versatility in the sense that its control function can be easily modified by changing a few program instructions.

Digital controllers have the inherent characteristic of accepting the data in the form of short duration pulses and producing a similar kind of output as control signal. Figure below shows simple control scheme employing a digital controller for a single-input/single-output system.



Single-input/Single-output system with a digital Computer

The sampler and analog to digital converter are needed at the computer input; the sampler converts the continuous-time error signal into a sequence of pulses, which are then expressed in a numerical code. Numerically coded output of the *digital computer* are

decoded into a continuous-time signal by digital to analog converter and hold circuit. This continuous-time signal then controls the plant. The overall system is *hybrid*, in which the signal is in a sampled forming the digital controller and in a continuous form in the rest of the system. A system of this kind is referred to as a sampled-data control system.

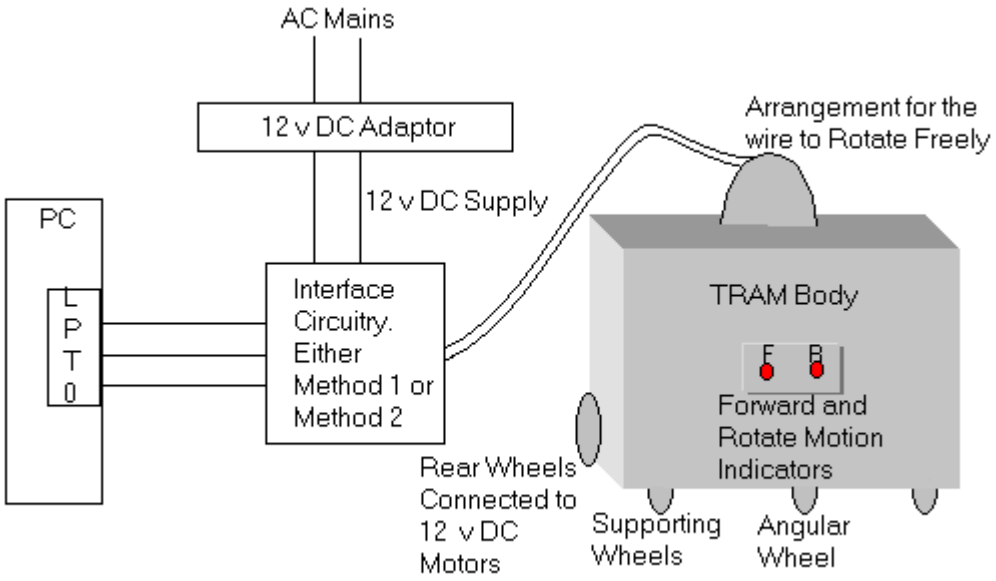
Although we are not using much of the facts explained here, this notes provides a good insight into the field of Modern Control Systems both manual and automatic. This explanation till now might have snobbed you but it animates many the facts about the Control Systems.

TRAM CONSTRUCTION

CONSTRUCTION OF DEVICE

CONSTRUCTION OF TRAM:

For constructing the body of the TRAM (device), polypropylene sheet of plastic is used because of its softness and malleability. The device is constructed in the form of a simple cube able to move on wheels as directed by the computer program. The device shown in the figure below depicts the structural details.

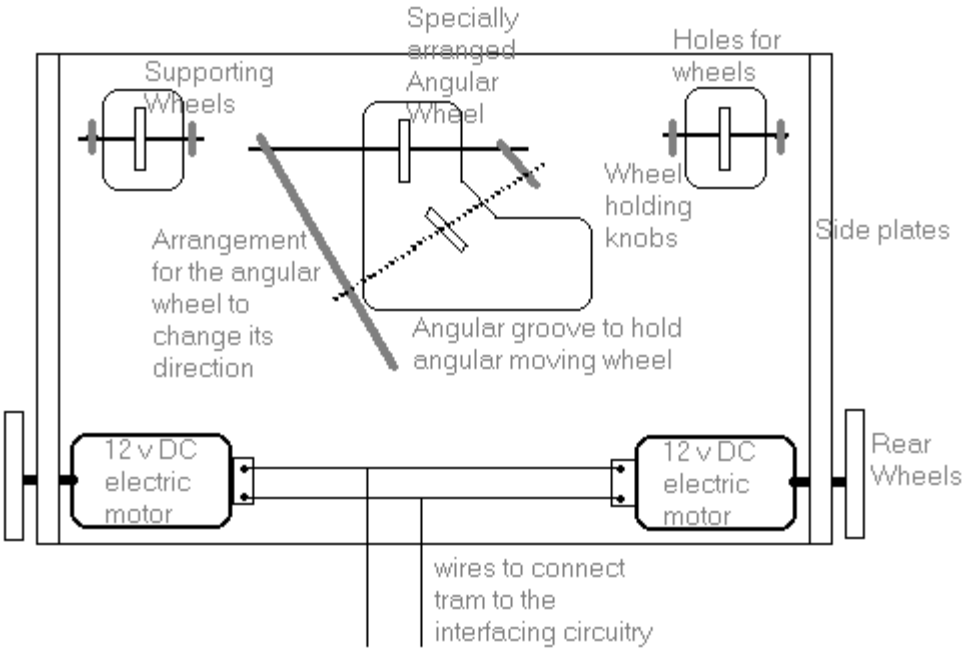


Block diagram of the overall view of the system.

The salient feature lies in the construction of the bottom plate of the device in which, an angular wheel able to change its position in order to change the direction of the device, is grooved at the top center. This wheel, when the device is translating forward, rests straight for the device to translate forward and when the device is translating backwards, this wheel changes its angle by moving only one of its axles while the other rests in its position, making the device to *Rotate* backwards, which is an important consideration in changing direction of the body.

In addition to this angular wheel the bottom plate also includes two other body-supporting wheels at the top two corners. These are intended to support the body from being balanced out. These two wheels help in holding the body parallel to the ground and also support the device to sustain against weights that it carries on.

The bottom plate also includes two low current 12v dc electric *ccw* motors that are used to rotate the wheels attached to the shafts of those motors. These two motors are connected in a way that both can rotate forward or backward simultaneously. Here 12 v motors are chosen as to sustain with some weights to carry on the device, and these are low current motors that accelerate linearly. The bottom plate including the angular wheel and motors is shown below and it depicts the above description.



Top view of the bottom plate of tramcar interfaced to personal computer.

On top of the device a special arrangement to carry any form of goods is set up. A synthetic paper glued to the four poles erected on top of the device makes this arrangement. Poles erected are made by cutting thin Galvanized Iron rods as these rods offer more strength with less weight.

This is the overall description of the construction details of the tramcar, which is used as an example to show the techniques of interfacing any general-purpose device to a personal computer.

PARALLEL PORT & INTERFACING TECHNIQUE

PARALLEL PORT

The PC supports both the parallel interface and serial interface. The parallel interface compatible to the standard centronics interface, which can be used for interfacing any parallel device. The serial interface is compatible to the standard RS-232C specifications. It is used for interfacing a serial interface printer, a mouse, a local terminal, a remote terminal through a modem or another computer.

The hardware merely serves as a multiple ports adapter. It provides a two way communication path between the software and the printer interface signals. All the protocol sequences are implemented by system software. While troubleshooting a problem in the printer controller, it is essential to know the software sequence followed for print operations. Without this it is impossible to trace some types of faults.

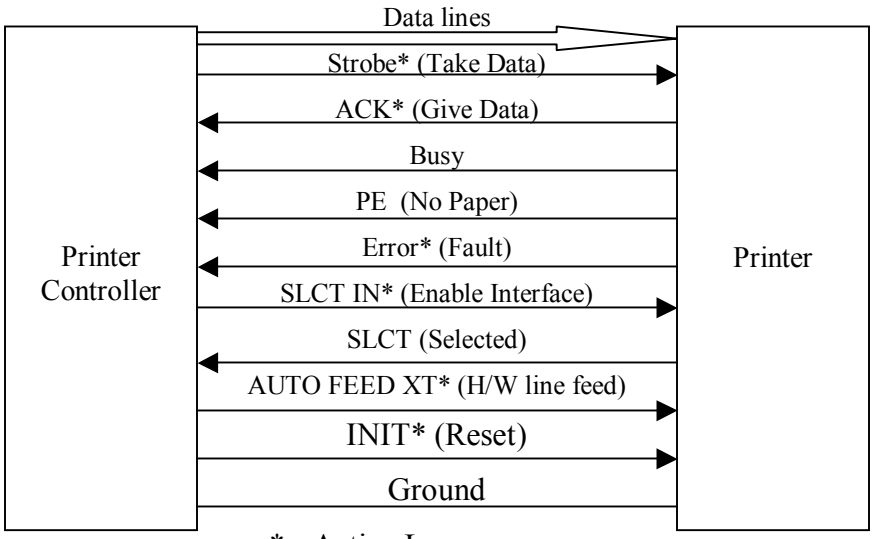
The parallel interface is called printer adapter or ***PARALLEL PORT.*** It may be physically present as a separate printer adapter board or as a part of MDA board, multi-I/O board, mother board, CGP, MGP, etc.

CENTRONICS INTERFACE:

The centronics interface provides a handshake protocol between a computer and any device and supports a maximum data transfer speed of 100 kb/s. the printer side of the interface is a 36 pin connector and the PC side is 25 pin D type connector. The PC uses 36 pin flat cable in which every alternative wire is for the ground. Most of the signals should have twisted pair wiring in the cable. The signals are the TTL level signals and the twisted pair return ground wire for each signal is connected to the signal ground level. To prevent noise effects the twisted pair wires are shielded and the shield is connected to the chassis ground in the system box.

Figure below depicts signals in the centronics interface.

Signals in Centronics Interface.



* - Active Low.

Since a PC uses 25 pin connector, there is a shortage of pins for four twisted pair signals. Usually, four different signals don't have twisted pair wires. The pin configurations for both printer side and PC side are listed in the table below.

PRINTER CABLE SIGNALS LIST.

Sl.No.	Signal Name	PC Side	Printer Side
1.	STROBE*	1	1
2.	TWPR-STROBE*	19	19
3.	Data 0	2	2
4.	TWPR-Data 0	-	20
5.	Data 1	3	3
6.	TWPR-Data 1	20	21
7.	Data 2	4	4
8.	TWPR-Data 2	-	22
9.	Data 3	5	5
10.	TWPR-Data 3	21	23
11.	Data 4	6	6
12.	TWPR-Data 4	-	24
13.	Data 5	7	7
14.	TWPR-Data 5	22	25
15.	Data 6	8	8
16.	TWPR-Data 6	-	26
17.	Data 7	9	9

18.	TWPR-Data 7	23	27
19.	ACK*	10	10
20.	TWPR-ACK*	-	28
21.	BUSY	11	11
22.	TWPR-BUSY	24	29
23.	PE	12	12
24.	TWPR-PE	25	30
25.	SLCT	13	13
26.	AUTO FEED XT*	14	14
27.	ERROR*	15	32
28.	INIT*	16	31
29.	SLCT IN*	17	36
30.	NC (No Connection)	-	15
31.	Logic Ground	-	16
32.	Chassis Ground	-	17
33.	NC	-	18
34.	Ground	18	33
35.	NC (No Connection)	-	34
36.	-	-	35

Note: The entries against Sl. No. 36 are just listed to show the assignment of pin Nos. on the 36 pin connector. The pin No. 35 on the printer side is pulled to +5V dc through 4.7k ohms resistance in the printer. * - Active Low.

I/O PORT ADDRESSES:

The port addresses in PC are 16 bits as per 8088's I/O mapped I/O scheme. Hence, theoretically the PC can address 64 kilo input ports. The CPU addresses the I/O ports through input (IN) or output (OUT) instruction.

The I/O addresses hexa 000 to 0FF are reserved for the motherboard. The addresses hexa 100 to 3FF are available for use in daughter boards. Table Overleaf provides an I/O map for PC.

Careful observation of the port addresses assigned in PC for various I/O ports in the system reveal the following points:

1. All motherboard I/O port addresses have 0's from A8 to A11.
2. In all daughterboards, I/O port addresses have either A8 or A9 as 1 or both A8 and A9 as 1's.

I/O MAP IN IBM PC.

Port addresses	**Allocation**

Hexa-Range	
000-01F	DMA Controller
020-021	Interrupt Controller
040-043	Timer, 8253-5
060-063	PPI, 8255a-5
080-083	DMA Page Registers, 74LS670
0AX	NMI Mask Register
200-20F	Game Controller
210-217	Expansion Unit
2F8-2FF	Serial Port 2 (Asynch)
300-31F	Prototype Card
320-32F	Hard Disk Controller
378-37F	**Parallel (interface) Printer Port**
380-38F	SDLC Communications
3B0-3BF	Monochrome Display and Printer Adapter
3D0-3DF	Colour/Graphics Monitor Adapter
3F0-3F7	Floppy Disk Controller
3F8-3FF	Serial Port 1 (Asynch)

PARALLEL PORT ADDRESS SUMMARY:

The port address used by the printer adapter in different boards are summarized in the table overleaf. The port addresses under the additional coloumn are used when two printer controllers, other than the MDA board, are present.

PORT ADDRESS SUMMARY:

Port Name	Port Address (MDA board)	Port Address (Printer Adapter-Main)	Port Address (Printer Adapter-Additional)
Data Out	3BC	378	278
Command out	3BE	37A	27A
Status In	3BD	379	279
Data In	3BC	378	278
Command in	3BE	37A	27A

PARALLEL INTERFACE:

The parallel interface between printer controller and the printer is known as CENTRONICS PARALLEL INTERFACE. The printer data and command signals sent by the software is simply passed on to the printer by the printer controller. Similarly the status signals sent by the printer are simply made available to the software on an input port. Perhaps because the printer controller does not have much intelligence and acts only as a coupler between the CPU and the printer, IBM has termed it a printer adapter.

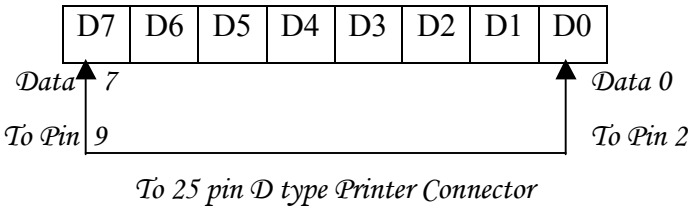
The printer controller supports data transfer in two different modes:

1. Programmed mode
2. Interrupt mode.

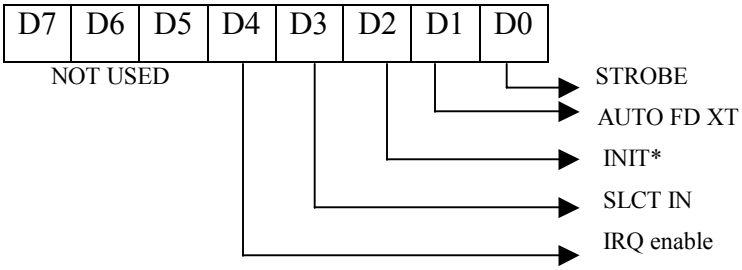
Different software routines operate the printer controller in one of these two modes. The software chooses the mode of data transfer and informs the printer controller of this by an appropriate command. Three different printer controllers are available in the IBM PC. These are named LPT1, LPT2 and LPT3 by software.

I/O PORTS – CONFIGURATION:

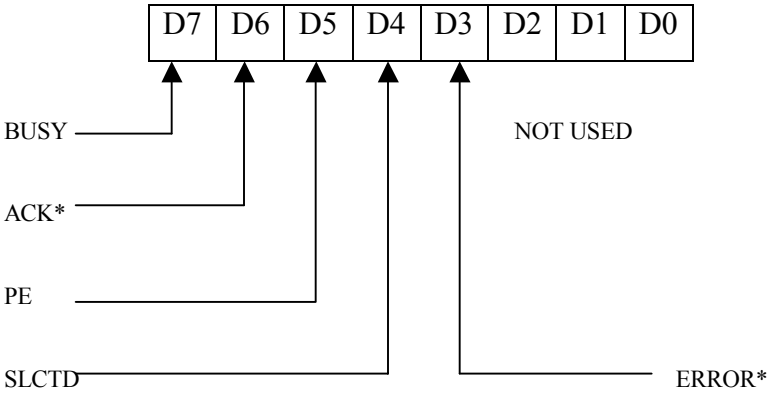
The printer controller appears to the software as two output ports and three input ports. Figures below depicts the bit assignments for these five ports.



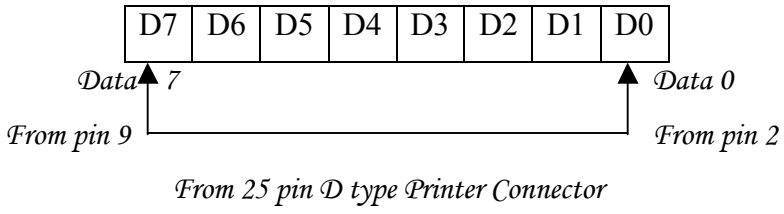
DATA OUT PORT CONFIGURATION



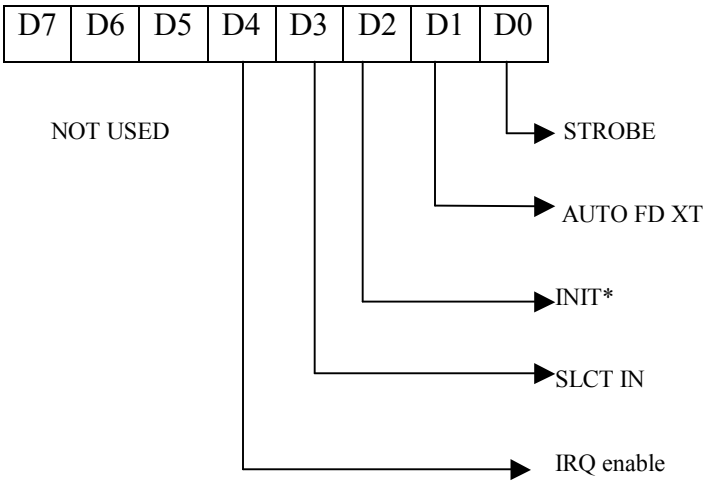
COMMAND OUT PORT CONFIGURATION



STATUS IN PORT CONFIGURATION



DATA IN PORT CONFIGURATION



COMMAND IN PORT CONFIGURATION

INTERFACING TECHNIQUE

INTERFACE: An interface is similar to a mediator between two people of different languages. Technically an interface is a device, which is used to, transform one form of signals into another form or make signals perceivable to another device or to synchronize any two devices.

The interface circuitry in this system is used to switch the 12v Regulated Power Supply to the two 12 v dc electric motors in the tram in two different polarities, only one at a time, depending upon the signals generated by the computer program in the parallel port LPT 0.

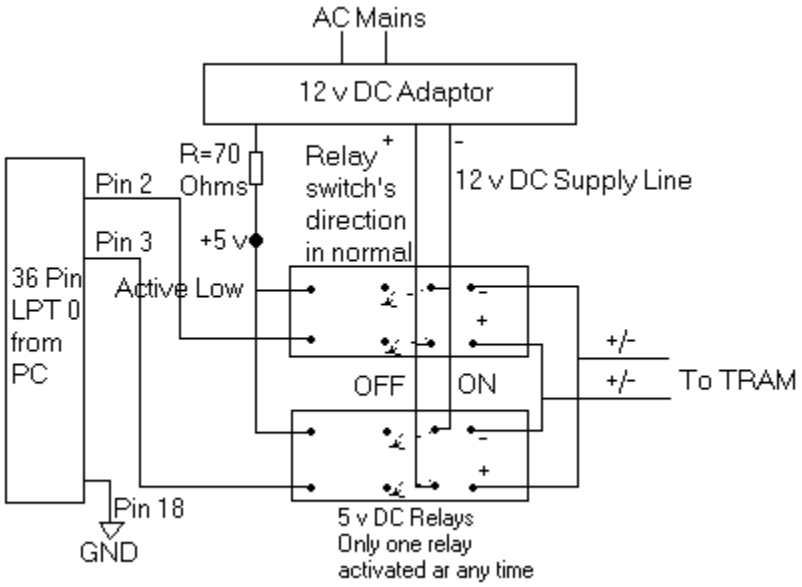
There are two possible methods of constructing interface circuitry, which are explained as under and depicted in the figures.

METHOD 1:

Originally the interface circuitry is designed electronically using low current 6v dc relays and Printed Circuit Boards, which was very efficient and reliable. In this first method two relays are driven by the two data pins from the port LPT 0 as shown.

The two 6v relays are connected to switch the 12v dc power supply to the two12v dc electric motors simultaneously in two different polarities with only one polarity given at any time and both of the relays will be OFF in general.

The 25pin to 36pin printer data cable is used for connecting the system to the LPT 0 port of the personal computer. The 36 pin male connector of the cable is interfaced with 36pin dual side female connector that is extracted from the logic board of the printer. This 36pin dual side female connector offers us the data lines, which can be used to connect to the interfacing circuitry. This connector is soldered on a Printed Circuit Board and the cable is connected to that with the common ground (earth) wire of connector soldered to its steel body.



Interfacing tram to PC using Relays.

The pins 2 to 9 of the connector form the 8 pins for data byte and the pin 18 is the ground pin for all these 8 pins and only two of the data pins along with the ground are

used in the interfacing technique. These data pins are connected to the two relays. The voltages on these pins are controlled by the program. There may be a 0volt or a 5volt signal at any time on the line and a 5volt signal stimulates the relay and a 0volt signal makes relay *off*.

One of the relays when stimulated, switches the 12 v dc power supply to the two 12 v dc electric motors, in a polarity, making them to rotate in one direction (consider forward) and when the other relay is stimulated, it switches the 12 v dc power supply to the two 12 v dc electric motors, in the other polarity, making them to rotate in the other direction (consider rotating backwards) and only one relay is stimulated at any moment of time if necessary or else both relays will be in *off* state.

This was the originally conceived technique for interfacing. But it is not implemented due to lack of human resources. Although this technique is very efficient and reliable it is not implemented as it is not economical and cannot be implemented within the economy I had.

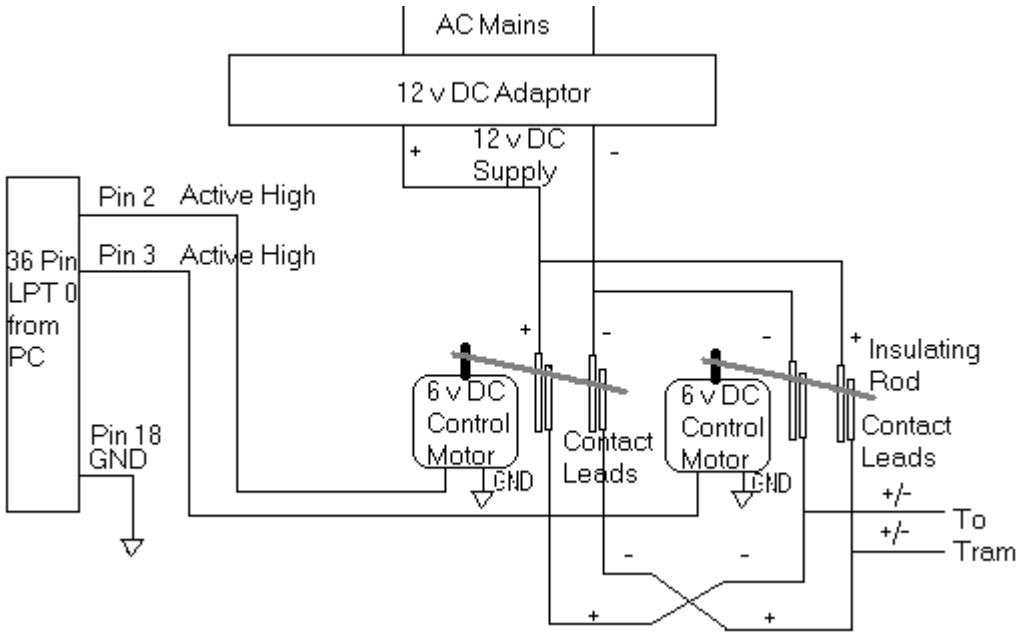
METHOD 2:

As an alternate economical method, two 6v dc motors are used to switch the 12 v Regulated Power Supply to the two 12v dc electric motors. The shafts of the two electric motors are joined with small insulating sticks and just beside each of the motors two pairs of wires are erected and slightly separated with a little tension in them.

Here too a 25pin to 36pin printer data cable is used for interfacing the system to the personal computer. The 36 pin male connector of the cable is interfaced with 36pin dual side female connector which is extracted from the logic board of the printer. This 36pin dual side female connector offers us the data lines, which can be used to connect to the interfacing circuitry. This connector is soldered on a Printed Circuit Board and the cable is connected to that with the common ground (earth) wire of connector soldered to its steel body.

The pins 2 to 9 of the connector form the 8 pins for data byte and the pin 18 is the ground pin for all these 8 pins and only two of the data pins along with the ground are used in the interfacing technique. These two data pins are connected to the two *ccw* 6v dc motors. The voltages on these pins are controlled by the program. There may be a 0volt or a 5volt signal at any time on the line and a 5volt signal stimulates the 6v dc motor to rotate and a 0volt signal makes the 6v dc motor to be idle. The whole interfacing circuitry is depicted in the figure below.

When a motor is activated by the data pin of the parallel port LPT 0, the insulating rod joined to the shaft of the 6v dc motor makes the two pairs of wires contact each other and with that action the 12 v dc power supply is passed on to the motors with one polarity making the two motors simultaneously to rotate in one direction. In the very similar way when the other 6 v dc motor is activated by the other data pin the 12v dc power supply is passed on to the two motors in the other polarity making them to rotate in the other direction.



Interfacing Tram to PC using 6v dc motors (economical).

This polarity reversal is achieved by a twisted pair connection. This twisted pair connection, which is generally used in connecting master and slave hard disks in one

computer is implemented here with the same task. This connection twists a pair of wires between the two pairs of wires erected beside the 6v dc motors which results in the polarity reversal and there arises no conflict as only one motor will be activated at any one time. This twisted pair connection is shown in the above figure.

With these two methods the tram can be interfaced to the personal computer and in this project the second method is chosen because of its simplicity and low cost. For the above two methods the 12 v dc power supply is produced by the 12 v-500ma dc adaptor.

RELIABILITY OF THE SYSTEM

The system with its parts: interfacing circuitry and cable connector, assembled in a enclosed container, are shatter proof and sustains to small amounts of stresses and is well operable at normal temperature. The system is reliable and results good motion if it is maintained properly protecting it from sudden shocks.

The interfacing circuitry is susceptible to the sudden shocks and jerks and if it malfunctions it can be serviced very easily. The sudden shock may increase the tiny gap between the two pairs of wires erected beside each of the 6v dc motors, which makes them not to contact when the 6v dc motor is activated by a 5v signal on one of the data pins coming out from the cable connector.

When this occurs just open the top flap of the interfacing circuitry box and adjust the wires so that they both contact each other when the motor is activated. This is the only malfunction that may arise due to sudden shocks and jerks.

Another malfunction that might arise due to ageing would be wire shorts in the twisted pair connection in the interfacing circuitry. This might become a big problem. If it occurs the wires should be replaced with the other wires and there is no other alternative.

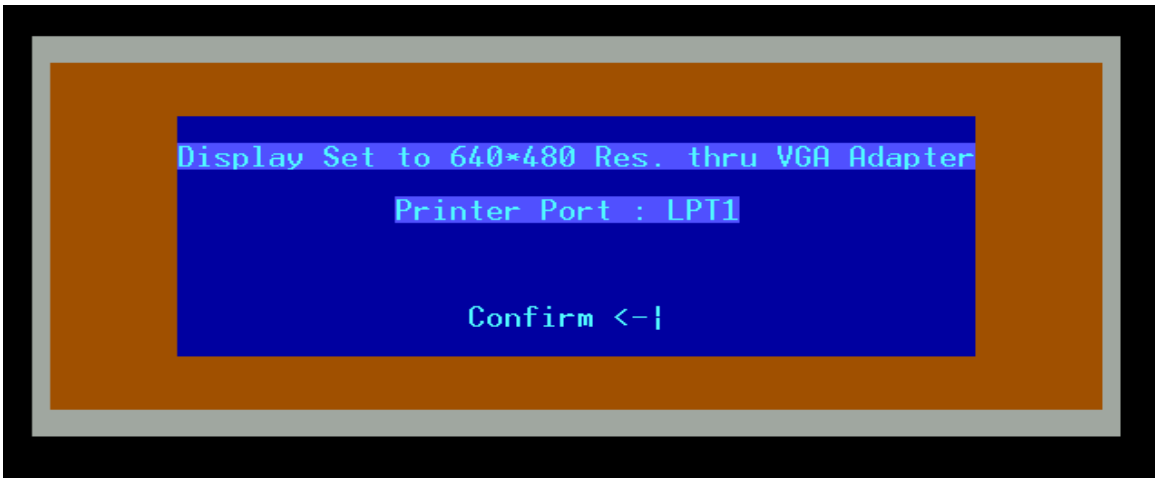
Another inconvenience that might arise is that the wheels at the bottom of the tram may become loose due to expansion of the wires with which they were tied. It is a simple problem and for that just unscrew the tram body and re-tie the wheels into the grooves with new insulating wire or with any other strong binding thread.

The program is so genuinely designed that it never malfunctions and there will not be any sort of bugs in the program as it is designed with much care and taking all aspects into consideration. If these problems are checked from time to time the system offers better operation in all conditions. Thus I say that the system designed is reliable which is achieved with the minimum cost possible. The section How to Use Program includes a detailed description of the application and how to use it.

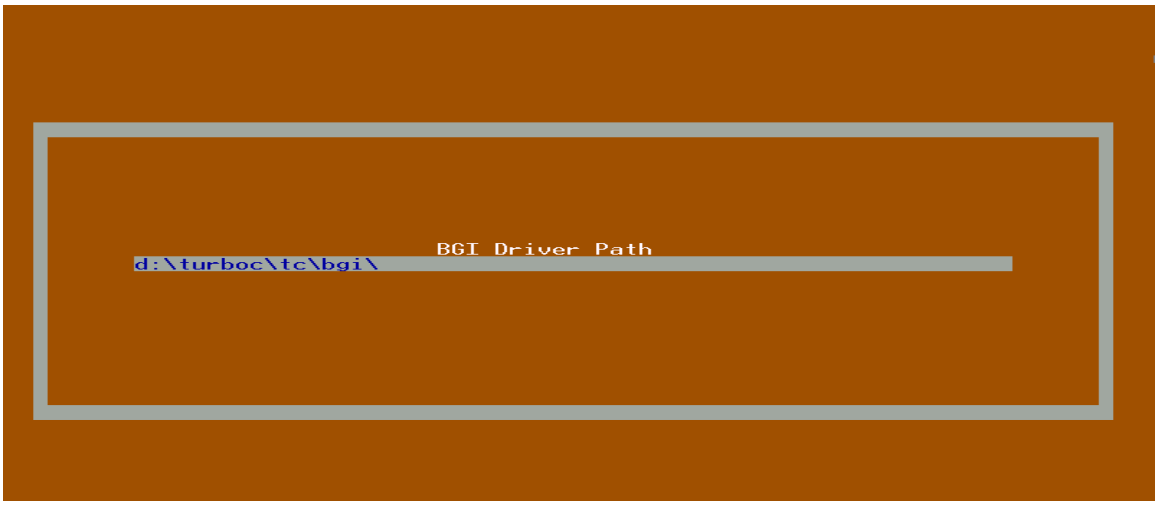
HOW TO USE PROGRAM

The program that is designed in C offers even a naïve user all the provisions for understanding it easily and operating it effectively.

The program when called upon for our work, initializes the parallel port LPT0 to read and send data with the display adapter set to 640*480 pixel resolution.



First initialization screen showing Graphics Mode and Printer Port

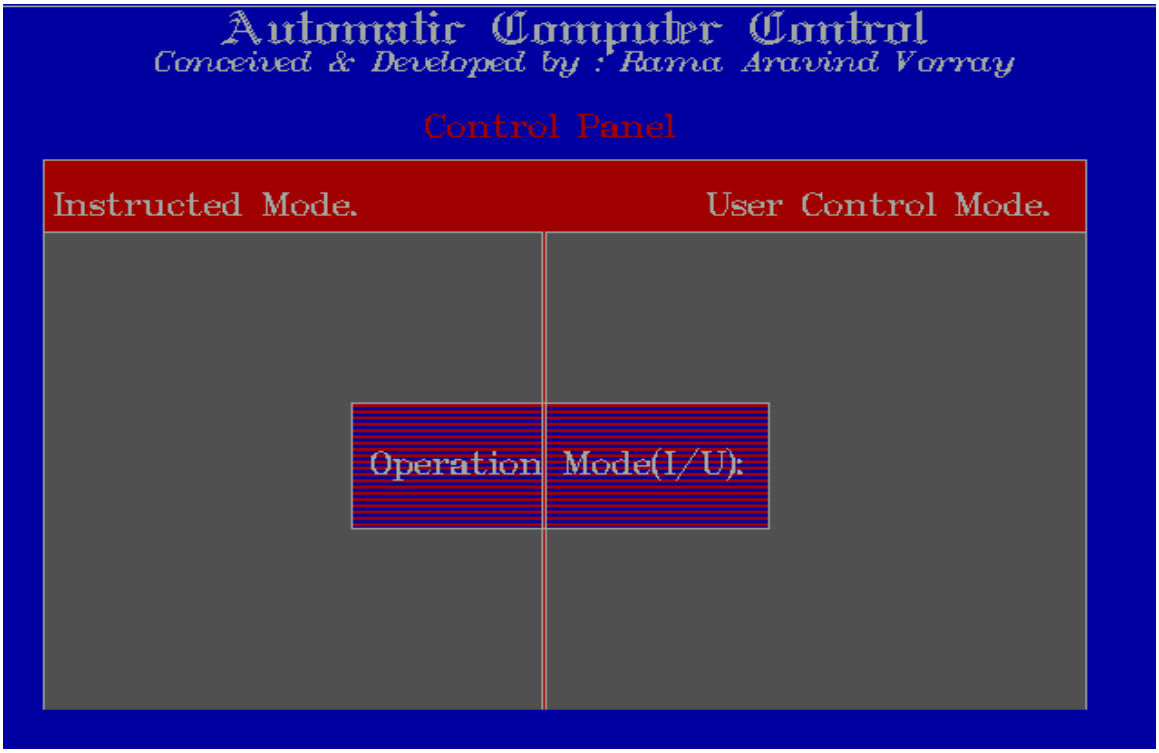


Second Screen prompting for path containing BGI driver files.

This program offers users the flexibility of storing the graphics driver files at their interesting locations as it prompts user every time it starts with a default graphics driver

path and if the files are at some other location then we can change the path in the *BGI driver path* screen. The program shifts to graphics mode screen if the path points to the BGI files.

The main task of the program starts from this graphics screen. This screen shows us the title and the two control modes in which the program can be operated and prompts us to select any one of the two *operation modes* by key depressal.



Main Screen showing the Title and two Modes of Operation.

The two operating modes of the program are:

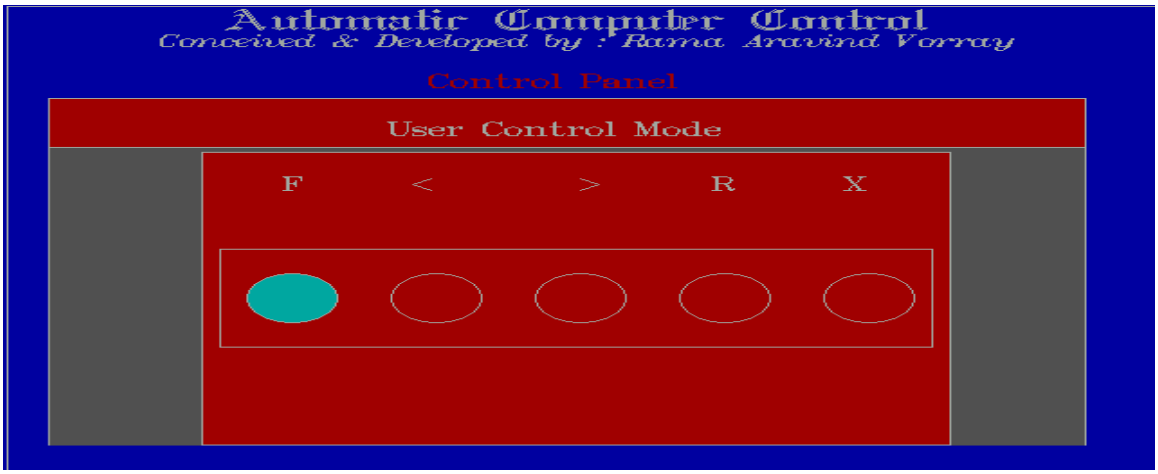
1. User Mode.
2. Instruction Mode.

Operation in *User Mode* can be chosen by keying in the U letter on the keyboard and the operation in *Instruction Mode* can be chosen by keying in the I letter on the keyboard. Description of the two modes follows.

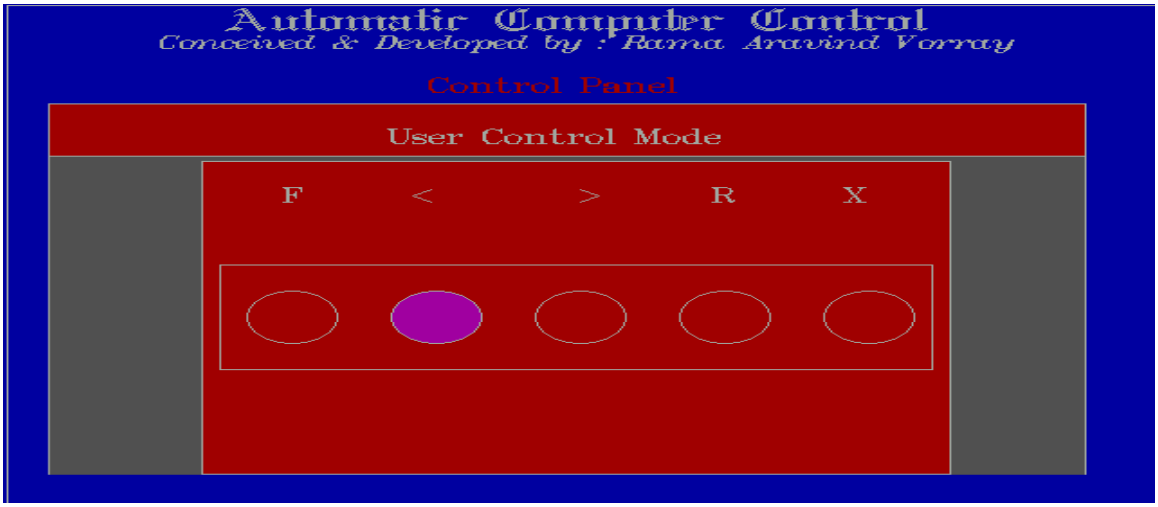
USER MODE:

In this mode of operation, user of the system controls the tram by the help of four *arrow keys*. The tram changes its position as predefined according to the order of the key depressals. The meanings assigned to each of the four arrow keys are as below.

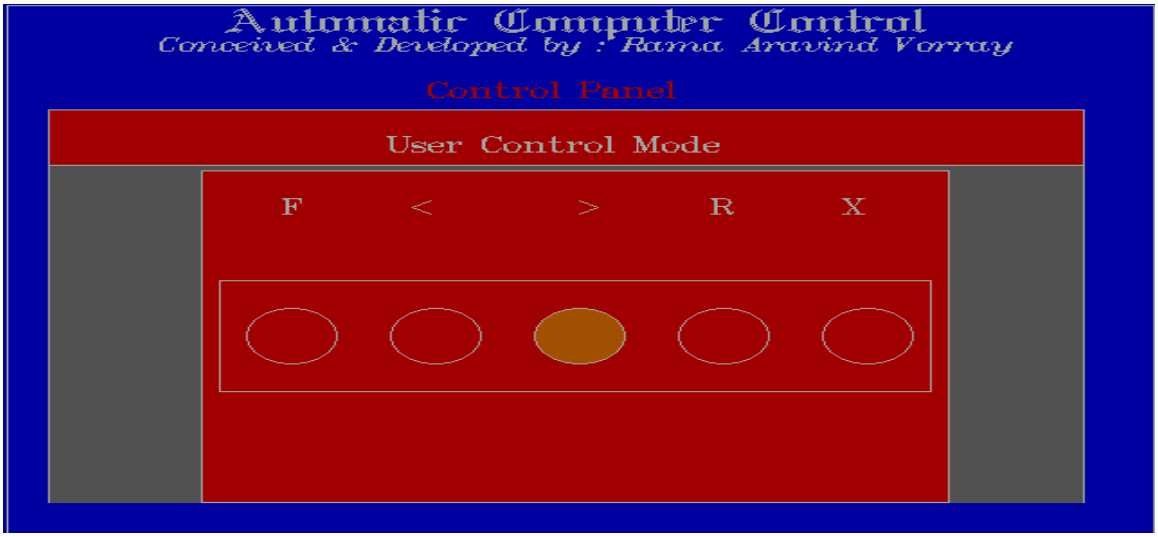
Main screen of the User Control mode of operation.



User Mode Screen indicating the UP arrow key depressal.

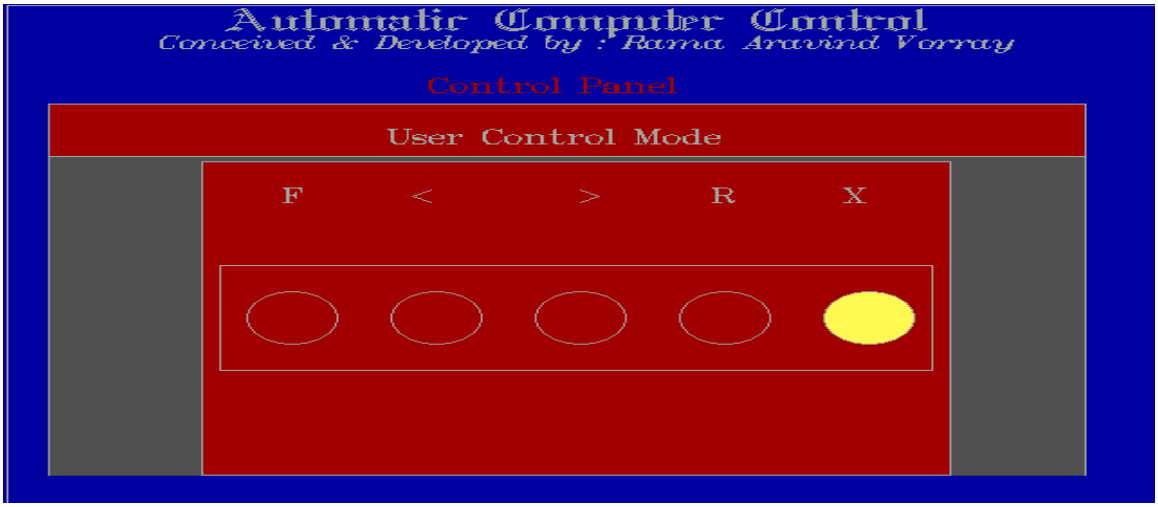


User mode screen indicating the LEFT arrow key depressal.



User mode screen indicating RIGHT arrow key depressal

User mode screen indicating Down arrow key depressal for rotating.



User Mode screen indicating ESC key depressal to exit.

Up arrow key: This key once depressed enables the device to move forwards for a distance of 50 cm relatively from its current position. The depressal of the key is shown on the screen by a cyan circle displayed for the time the tram moves. The speed with which the device moves will be fixed and can be varied by changing the speed control knob of any of the two 12v dc electric motors inside the body of tram.

Left arrow key: This key once depressed enables the device to change its direction so that it faces left side of its current position and it achieves it by rotating back to an angle of 270^{o} from its current position. This makes it to turn to its left side. The depressal of the key is shown on the screen by a magenta circle displayed for the time the tram changes its direction to left. This motion is achieved by the specially arranged angular moving wheel at the top center of the bottom plate of the tram.

Right arrow key: This key once depressed enables the device to change its direction to right side from its current position and it is achieved by rotating backwards to an angle of 90^{o} from its current position. This makes the device to turn to its right side. The depressal of the key is shown on the screen by a brown circle displayed for the time the tram changes its direction to right. This motion is achieved by the specially arranged angular moving wheel at the top center of the bottom plate of the tram.

Down arrow key: This key once depressed enables the device to rotate backwards for an angle of 360^{o} and is just an optional motion and is as much useful as other motions do in controlling the motion of the tram. The depressal of the key is shown on the screen by a light blue circle displayed for the time the tram rotates.

Esc key: This key is intended to escape from the *User Mode* of operation to the main screen and when depressed a yellow circle informs us that we are shifting to the main screen and then shifts to the main screen.

This is about the *User Mode* of operation and offers an efficient and flexible use of the program in controlling the motion of the tram. This mode of operation is mainly intended for short guided operations and for naïve users.

INSTRUCTION MODE:

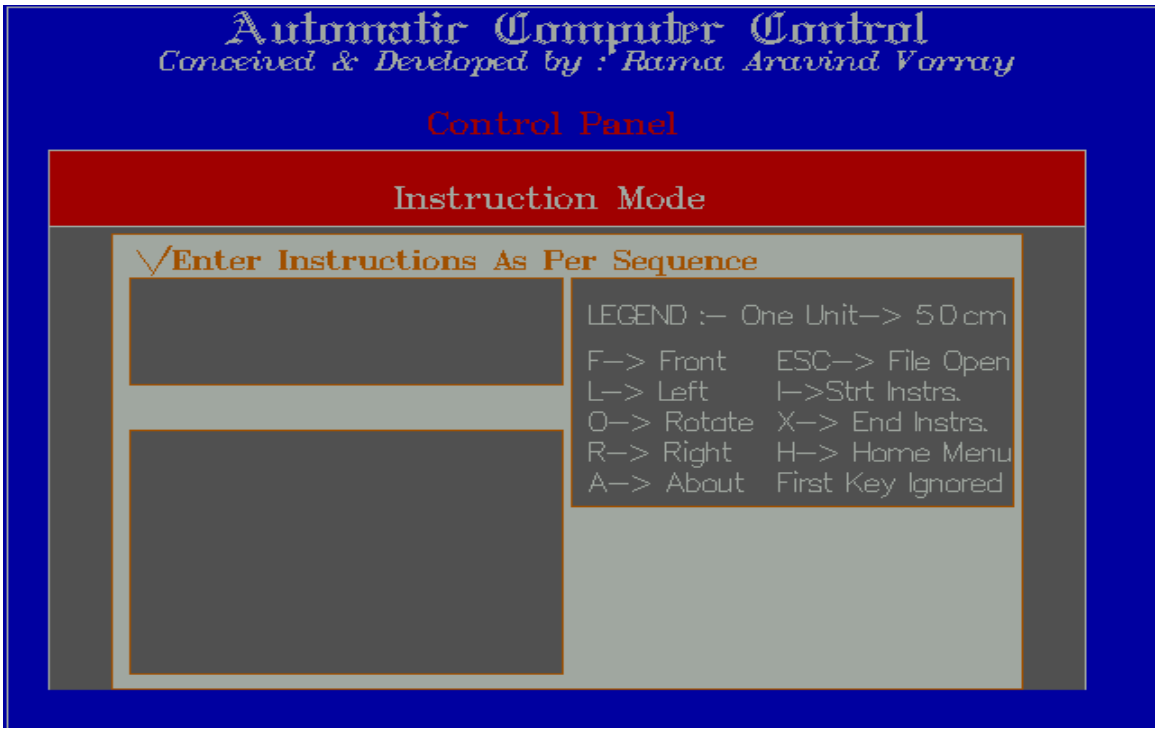
In this mode of operation, a group of instructions expected to be written or written already in a file issue signals to the interface circuitry and controls the tramcar according those instructions. In other words it is similar to a program file to control the motion of tramcar. This program file contains the four instructions in any sequence and the file may be with any extension.

The four instructions available in this mode of operation are:

1. F – forward motion for 50cm
2. L – left side motion (270^{o})
3. O – rotate backwards (360^{o})
4. R – right side motion (90^{o})

These four single letter instructions can be written in any sequence in the file and by reading each instruction one by one the program issues signals to the interface circuitry and the tram moves according to the instructions read and program quits to read and process instructions when the very first X appears in the sequence. If the instruction file contains any other letters other than those assigned they are just *ignored*. The program

offers the user to write a sequence of up to 40 single letter instructions. The instructions can be saved in a file with any extension.



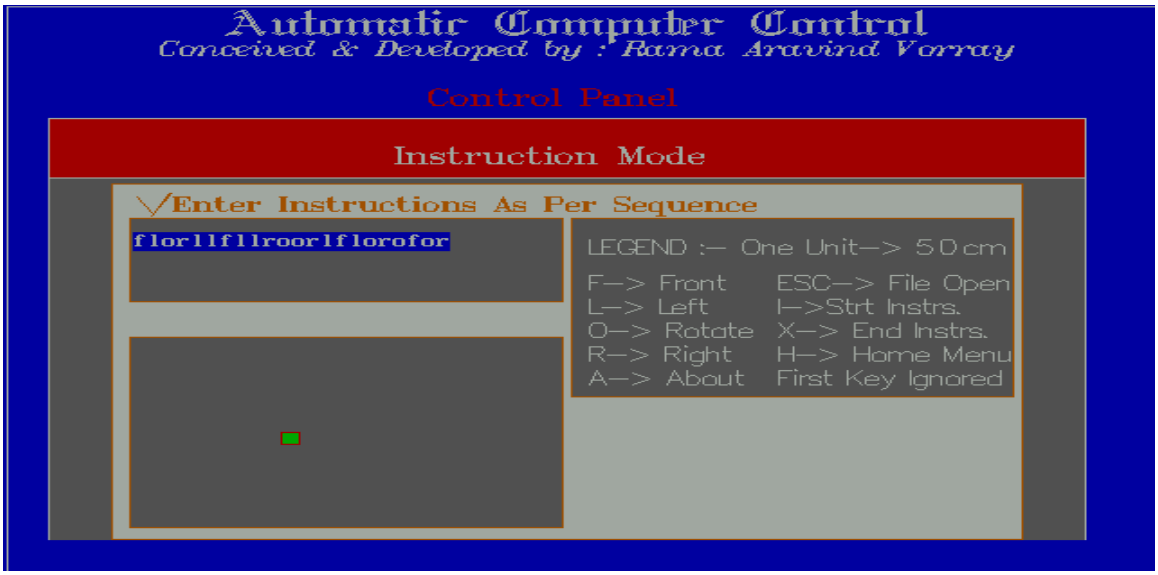
Main screen for Instruction Mode of operation.

***** This program offers the flexibility of using any text file with any content as an instruction file, which specifies that the instructions can be embedded in other text files which provides us very high degree of security. This is extremely useful if similar devices are used as security devices and if we want no one to know the sequence in which it may operate.

The first key depressal has certain meanings which are as below:

Esc – open an instruction file.
I – start entering instructions.
H – return to home (main screen).
X – end entering instructions.
Any other key entered at first is just ignored.

Instructions can be written at any instance after first key depressal and can be saved in a file.we can even open a previously saved file and let it to operate by keying in *Esc* key as the first key.



Instruction Mode screen executing a sequence of instructions with INDICATOR.

After we enter the instructions manually and key in X at the end the program asks us to *save in a file* and if any other key except N (no) is pressed the program prompts for a file name and we can save the instructions we entered in a file. After this program prompts us to *confirm instructions* and if we select N (no) the instructions are not processed and the program returns to the main screen. Otherwise the instructions are executed and the tram starts to move as instructed. The file may contain instructions in any sequence which may make the device even to rap by sizzling motions

Similar to the user mode, the program displays on the screen, the current motion of the tram by a tiny green square moving as the tram should move, which shows us the status of the program at any instance while execution.

After the execution of the instructions the program automatically returns to the main screen and if we want to halt the processing of instructions in between, it can be done just

by hitting any key while the program is processing instructions and the program aborts it after the current motion is completed.

This is the detailed description of the *Instruction Mode* of operation and thus as explained offers users the effective and efficient way to program the motion of the tramcar and in common any general purpose device attached to the computer.

SALIENT FEATURES OF THE PROGRAM

There are some important features of the program worth knowing. The program designed in C is intended to control the device attached to it through voltage variations on the data pins connected to the interface circuitry. These voltage variations should occur at certain time intervals for the device to perceive and move the assigned distance. The important feature of the program stands in calculating these time delays.

The program calculates these time intervals according to the speed of the processor, which may vary from system to system, implicitly through inbuilt function sleep() in dos.h header file.

The program offers user the flexibility of storing the graphics driver files at their interesting locations. The program in its *BGI driver path* screen prompts user with the default location of the BGI driver files and if they are stored at some other location the path can be keyed in.

The program offers good colourful character interactive screens that are easily operable and best illustrates the execution of the program at any instance of operation, which can be easily understood by all users.

The program is designed not keeping in view any particular user but it is designed for both industrial and domestic purposes. The program shows the screens in a format that are suitable for both industrial and domestic environments.

These are the salient features of the program designed to control any general purpose device through a personal computer.

PROGRAM

```
#include<stdio.h>
#include<process.h>
#include<conio.h>
#include<dos.h>
#include<graphics.h>
void instructions(void);
void displayf(void);
void displayl(void);
void displayr(void);
void displayo(void);
void usercontrol(void);
void menu(void);
char i;
void main(void)
{int a=9,b=2,ercode,prn1,aft_path_cont;
 char prn,grapath[30]={"d:\\turboc\\tc\\bgi\\"};     // path for bgi files
 textbackground(0);
 clrscr();
 window(10,5,69,19);
 textbackground(7);
 clrscr();
 window(11,6,68,18);
 textbackground(6);
 clrscr();
 window(18,8,61,16);
 textbackground(1);
 clrscr();
 textcolor(11+128);
 gotoxy(3,2);
 cprintf("Display to 640*480 Res. thru VGA Adapter");
```

```
asm{ mov ah,0x01
mov dx,0
int 0x17
mov prn,ah}
prn=prn&0x10;
if(prn==0x10)
{
outport(0x378,0xffff);
prn1=inport(0x378);
prn1=prn1&0x00ff;
if(prn1==0x00ff)
{
gotoxy(13,4);
cprintf("Printer Port : LPT1");
}
else
{gotoxy(1,4);
cprintf("Error Reading Data From LPT1");
gotoxy(1,6);
cprintf("Continue Anyway (y/n):");
i=getch();
if(i!='n'||i!='N')exit(0);
}}
else
{gotoxy(1,4);
 cprintf("Printer Port Not Initialized.");
 gotoxy(1,6);
 cprintf("Continue Anyway (y/n)");
 i=getch();
 if(i!='n'||i!='N')exit(0);
}
```

```
textcolor(11);gotoxy(15,8);
cprintf("Confirmed <-|");
getch();outport(0x378,0xff00);

window(1,1,80,25);textbackground(6);
clrscr();window(3,3,77,22);
textbackground(7);clrscr();
window(4,4,76,21);textbackground(6);
clrscr();textcolor(15);
gotoxy(28,8);cprintf("BGI Driver Path");
window(10,12,70,12);textbackground(7);
clrscr();textcolor(1);
cprintf("%s",grapath);
aft_path_cont=getch();
switch(aft_path_cont)
{case 13:break;
 default:getch();clrscr();
 gets(grapath);break;
 }
initgraph(&a,&b,grapath);
ercode=graphresult();
window(1,1,80,25);textbackground(0);
clrscr();window(10,11,70,14);
textbackground(1);clrscr();
textcolor(4);gotoxy(18,2);
switch(ercode)
{
  case -1:cprintf("Graphics(BGI) Not Installed");
  gotoxy(18,3);cprintf("X To Exit || ENTER To Resume");
  i=getch();if(i=='x'||i=='X')exit(0); main();break;
```

```
   case -2:cprintf("Graphics Hardware Not Found");gotoxy(18,3);
   cprintf("X To Exit || ENTER To Resume");
   i=getch();if(i=='x'||i=='X')exit(0); main();break;

   case -3:cprintf("Device Driver File Not Found");gotoxy(18,3);
   cprintf("X To Exit || ENTER To Resume");
   i=getch();if(i=='x'||i=='X')exit(0); main();break;

   case -4:cprintf("Invalid Device Driver File");gotoxy(18,3);
   cprintf("X To Exit || ENTER To Resume");
   i=getch();if(i=='x'||i=='X')exit(0); main();break;

   case -5:cprintf("Not Enough Memory To Load");gotoxy(18,3);
   cprintf("X To Exit || ENTER To Resume");
   i=getch();if(i=='x'||i=='X')exit(0); main();break;
  }
 setbkcolor(1);
 clearviewport();
 menu();
 getch();
}

void usercontrol(void)
{int x,y;
 int c;
 clearviewport();
 setcolor(7);rectangle(15,46,590,405);
 setfillstyle(1,4);
 floodfill(16,48,7);
 settextstyle(1,0,2);setcolor(4);
 outtextxy(226,25,"Control Panel");
```

```
setcolor(7);
outtextxy(203,74,"User Control Mode");
rectangle(15,96,590,405);setfillstyle(1,8);
floodfill(16,97,7);
setcolor(6);
outtextxy(120,200,"Use Arrow Keys to Control ROBOT");
getch();setcolor(7);
rectangle(100,101,515,400);setfillstyle(1,4);
floodfill(101,102,7);
outtextxy(105,130,"    F     <       >      R      X");
outport(0x378,0xff00);rectangle(110,200,505,300);
x=150;y=250;
while(x<500){circle(x,y,25);x+=80;}
while(1)
{c=getch();
switch(c)
{//case 32:setfillstyle(1,12);outport(0x378,0xff11);floodfill(450,250,7);
 //sleep(1);setfillstyle(1,4);floodfill(450,250,7);outport(0x378,0xff00);
 //continue;

 case 72:setfillstyle(1,3);outport(0x378,0xfff1);floodfill(150,250,7);
 sleep(2);setfillstyle(1,4);floodfill(150,250,7);outport(0x378,0xfff0);
 continue;

 case 75:setfillstyle(1,5);outport(0x378,0xfff2);floodfill(230,250,7);
 sleep(1);setfillstyle(1,4);floodfill(230,250,7);outport(0x378,0xfff0);
 continue;

 case 77:setfillstyle(1,6);outport(0x378,0xfff2);floodfill(310,250,7);
 sleep(2);setfillstyle(1,4);floodfill(310,250,7);outport(0x378,0xfff0);
 continue;
```

```
 case 80:setfillstyle(1,9);outport(0x378,0xfff2);floodfill(390,250,7);
 sleep(3);setfillstyle(1,4);floodfill(390,250,7);outport(0x378,0xfff0);
 continue;

 case 27:setfillstyle(1,14);floodfill(450,250,7);delay(500);menu();break;
}
}
//outport(0x378,0xff00);
//menu();
 }

void instructions(void)
{int i,j,l,ab,ab1,m,flag,tst;
 char instrs[40],*fname;
 FILE *f;
 ll:clearviewport();window(1,1,80,25);
 setcolor(7);rectangle(15,46,590,405);
 setfillstyle(1,4);
 floodfill(16,48,7);
 settextstyle(1,0,2);setcolor(4);
 outtextxy(226,25,"Control Panel");setcolor(7);
 outtextxy(207,74,"Instruction Mode");
 rectangle(15,96,590,405);setfillstyle(1,8);
 floodfill(16,97,7);setcolor(6);
 rectangle(50,101,555,400);setfillstyle(1,7);
 floodfill(51,102,6);
 settextstyle(1,0,1);
 outtextxy(55,115," \\/Enter Instructions As Per Sequence");
 setcolor(6);
 rectangle(60,130,300,200);setfillstyle(1,8);
```

```
floodfill(61,131,6);rectangle(60,230,300,390);
setfillstyle(1,8);floodfill(61,231,6);
rectangle(305,130,550,280);setfillstyle(1,8);
floodfill(306,131,6);
setcolor(7);settextstyle(6,0,1);
outtextxy(315,150,"LEGEND :- One Unit-> 50cm");
outtextxy(315,180,"F-> Front");outtextxy(315,220,"O-> Rotate");
outtextxy(315,200,"L-> Left");outtextxy(315,240,"R-> Right");
outtextxy(315,260,"A-> About");outtextxy(420,180,"ESC-> File Open");
outtextxy(420,260,"First Key Ignored");
outtextxy(420,220,"X-> End Instrs.");
outtextxy(420,240,"H-> Home Menu");
outtextxy(420,200,"I -> Strt Instrs.");
settextstyle(1,0,1);
flag=0;
tst=getch();
if(tst=='h'||tst=='H')menu();
if(tst==27){flag=1;goto xy;}
window(10,13,30,23);
gotoxy(1,1);
for(i=0;i<40;i++){
instrs[i]=getch();if(instrs[i]=='x'||instrs[i]=='X')break;
if(instrs[i]==13)continue;
printf("%c",instrs[i]);
ab=wherey();ab1=wherex();
if(ab1>22){ab++;ab1=1;}
gotoxy(ab1,ab);
}
m=i;
setcolor(4);
outtextxy(320,300,"Save In File (y/n):");
```

```
i=getch();if(i=='y'||i=='Y')
{setcolor(7);outtextxy(320,300,"Save In File (y/n):");
file:setcolor(4);
outtextxy(320,300,"File Name:");
window(55,23,70,23);clrscr();
scanf("%s",fname);
setcolor(7);outtextxy(320,300,"File Name:");
setcolor(4);
if(fopen(fname,"r")!=NULL){outtextxy(320,330,"File Exists Overwt.(y/n):");
i=getch();setcolor(7);outtextxy(320,330,"File Exists Overwt.(y/n):");
if(i=='n'||i=='N')goto file;}
f=fopen(fname,"w");for(i=0;i<m;i++)fputc(instrs[i],f);
fclose(f);}
else{setcolor(7);outtextxy(320,300,"Save In File (y/n):");
setcolor(4);}
outtextxy(320,330,"Confirm Instructions");
i=getch();
if(i=='n'||i=='N')goto ll;
setcolor(7);outtextxy(320,330,"Confirm Instructions");
setcolor(4);
xy:if(flag==1){flag=0;setcolor(4);
outtextxy(320,300,"Open File:");
window(55,23,70,23);clrscr();
scanf("%s",fname);setcolor(7);
outtextxy(320,300,"Open File:");
setcolor(4);f=fopen(fname,"r");
if(f==NULL){outtextxy(320,330,"Not A Valid File!!!");sound(1000);
delay(300);nosound();setcolor(7);outtextxy(320,330,"Not A Valid File!!!");
setcolor(4);}
i=0;
while(!feof(f)){instrs[i]=fgetc(f);i++;}
```

```
 m=i;}
 for(i=0;i<m;i++){
 if(kbhit()){getch();break;}
 switch(instrs[i])
 {case 'f':
   case 'F':outport(0x378,0xfff1);displayf();break;
   case 'o':
   case 'O':outport(0x378,0xfff2);displayo();break;
   case 'l':
   case 'L':outport(0x378,0xfff2);displayl();break;
   case 'r':
   case 'R':outport(0x378,0xfff2);displayr();break;
//case 'a':outport(0x378,0xff11);sleep(2);break;
   default:break;
 }}
 outport(0x378,0xff00);
 menu();
}

void menu(void)
{char c;   sound(1000);
 setviewport(0,0,639,479,1);clearviewport();
 setcolor(7);
 rectangle(0,0,639,479);
 settextjustify(0,1);settextstyle(4,0,4);
 outtextxy(106,9," Automatic Computer Control");
 settextstyle(7,0,1);
 outtextxy(86,33,"Conceived & Developed by : Rama Aravind Vorray");
 setviewport(10,50,600,450,1);
 setcolor(7);rectangle(15,50,590,405);
 setfillstyle(1,4);
```

```
floodfill(16,51,7);
settextstyle(1,0,2);setcolor(4);
outtextxy(226,25,"Control Panel");setcolor(7);
outtextxy(1,75,"  Instructed Mode.                User Control Mode.");
rectangle(15,96,290,405);setfillstyle(1,8);
floodfill(16,97,7);
rectangle(292,96,590,405);setfillstyle(1,8);
floodfill(301,97,7);
nosound();
while(1){setcolor(7);
rectangle(185,205,415,285);
setfillstyle(2,4);floodfill(186,206,7);
floodfill(400,206,7);outtextxy(186,240," Operation Mode(I/U): ");
outport(0x378,0xff00);
c=getch();
switch(c)
{ case 'I':
  case 'i':instructions();break;
  case 'U':
  case 'u':usercontrol();break;
  case 'X':
  case 'x':outport(0x378,0xff00);closegraph();exit(0);
  default :setcolor(0);outtextxy(186,240," Operation Mode(I/U): ");
    setfillstyle(1,0);floodfill(186,260,7);floodfill(400,260,7);
    floodfill(186,206,7);floodfill(400,206,7);setcolor(7);
    outtextxy(186,240,"     Select (I/U) ");sleep(1);setcolor(0);
    outtextxy(186,240,"     Select (I/U) ");continue;
}}}

void displayf(void)
{  int x=170,y=310,x1=180,y1=320;
```

```
setfillstyle(1,2);setcolor(4);
  for(;y>270;y--,y1--)
   {rectangle(x,y,x1,y1);floodfill(x+1,y+1,4);
    delay(80);setfillstyle(1,8);setcolor(8);
    rectangle(x,y,x1,y1);floodfill(x+1,y+1,8);
    setfillstyle(1,2);setcolor(4);}
}

void displayl(void)
{int x=170,y=310,x1=180,y1=320;
setfillstyle(1,2);setcolor(4);
  for(;x>140;x--,x1--)
   {rectangle(x,y,x1,y1);floodfill(x+1,y+1,4);
    delay(40);setfillstyle(1,8);
    setcolor(8);rectangle(x,y,x1,y1);
    floodfill(x+1,y+1,8);setfillstyle(1,2);
    setcolor(4);}
}

void displayr(void)
{int x=170,y=310,x1=180,y1=320;
 setfillstyle(1,2);setcolor(4);
 for(;x<200;x++,x1++)
 {rectangle(x,y,x1,y1);floodfill(x+1,y+1,4);
 delay(40);setfillstyle(1,8);
 setcolor(8);rectangle(x,y,x1,y1);
 floodfill(x+1,y+1,8);
 setfillstyle(1,2);setcolor(4);}
}
```

```
void displayo(void)
{int x=150,y=290,x1=200,y1=340;
 setfillstyle(1,2);setcolor(4);
 for(;y1>315;y++,y1--,x++,x1--)
 {rectangle(x,y,x1,y1);floodfill(x+1,y+1,4);
 delay(40);setfillstyle(1,8);
 setcolor(8);rectangle(x,y,x1,y1);
 floodfill(x+1,y+1,8);setfillstyle(1,2);
 setcolor(4);}
}
```

FINAL WORDS

Thus a software has been implemented to control any general purpose device with the help of a personal computer. This is rather simple device and if fostered properly much better closed loop control systems, for precise movements, can be achieved and the personal computers today offer all the capabilities that minicomputer can offer and if these are utilized in full then the whole world can be enlightened with everlasting sustainable technical products.

The main objectives that were laid down while conceiving the project work are described as under.

OBJECTIVES:

- To study and show various techniques of interfacing electronic equipment to the computer's parallel port LPT0.
- To study and show the techniques for controlling the motion of equipment by software instructions.
- To calculate real-time time delays between events depending upon the processor speed which may vary from computer to computer.
- To develop software intended to control any general purpose or ad hoc equipment using simple single letter commands.
- To develop encrypted instruction files to control the equipment which will be useful in security applications.

These are the objectives that were laid down while planning the project work and all the above are achieved as far as possible and this software addresses all these goals.

BIBLIOGRAPHY

1. *LET US C II e, by Yeshwant Kanetkar, BPB Publications.*

2. *MODERN CONTROL SYSTEM THEORY II e, by M.Gopal, New Age International (p) Limited Publishers.*

3. *IBM PC AND CLONES Hardware, Troubleshooting and Maintenance, by - Govindarajulu, TATA Mc Graw Hill Company.*

4. *IEEE SPECTRUM – Modular Robots, February 2002.*

5. *MASTERING TURBO C, by Kelly Bootle, BPB publications.*