SCAFBOT - A SERVO-CONTROLLED SCAFFOLDING DEVICE

by

ERIC HEATZIG

SUBMITTED TO THE DEPARTMENT OF
MECHANICAL ENGINEERING
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE
DEGREE OF

BACHELOR OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1987

Signature of Author_____
Department of Mechanical Engineering
May 15, 1987

Certified by_____
Professor Alexander H. Slocum
Thesis Supervisor

Accepted by_____
Prof. Peter Griffith
Chairman, Department Committee

SCAFBOT - A SERVO CONTROLLED SCAFFOLDING DEVICE

by

ERIC HEATZIG

Submitted to the Department of Mechanical Engineering
on May 15, 1987 in partial fulfillment of the
requirements for the Degree of Bachelor of Science in
Mechanical Engineering

ABSTRACT

Mobile scaffolding units are used extensively for both building construction and maintainance. Work on a scaffolding unit is both dangerous and time-consuming and given one of the highest risk ratings in the construction industry and any alternative method for working on the outside of buildings would be welcomed.

Most buildings built today have tracks which allow platforms to be suspended from the roof and to move up and down accurately. However, horizontal motion involves dismantling the unit and reassembling it on the next set of tracks. For this reason, we have decided to design a system whereby a platform could be moved simultaneously in both the horizontal and vertical directions. It would have to be able to position itself anywhere on the building facade with a high degree of accuracy. In this manner we hope that it will be possible to replace the human workers on the platform with robotic devices..

In order to ensure high accuracy and to prevent swaying,Professor Alex Slocum developed a new cable configuration suspending the platform which was predicted to give the platform extremely high lateral stiffness. To test the feasibility of the concept and the cable dynamics, we built a working 1/20th scale model of a complete scaffolding system with integrated computer control.

The system is fully functional and preliminary tests have shown that the cable configuration prevents the platform from swaying and allows it to be positioned accurately. Although simple control software has been written, more complex work is under way and until it is fully developed, the exact accuracy of the system cannot be determined. However, it is expected that we will be able to position the platform to within five thousandths of an inch of the desired position.

This thesis includes a detailed description of the existing model and the control problems encountered as well as recommendations for a possible prototype design.

Thesis Supervisor: Professor Alex Slocum

Title: Assistant Professor of Civil Engineering

Acknowledgements:

I would like to thank my Brothers at Nu Delta for all of their help and support. Special thanks go to Carsten Hochmuth for working beyond the call of duty. Many thanks to the makers of Medaglia D'oro coffee for keeping me going in the early hours of the morning. I would like to dedicate this work to my parents whose love and support kept me going the whole time.

Table of Contents

List of Figures

1. Introduction

A new and untested cable configuration (see figure 1) was tested using a scale model of a servo-controlled scaffolding unit for operation on large buildings. The buildings for which this unit is designed are aproximately 100 ft wide and 200 ft high. A 1/20th scale model of such a building measures 5 ft by 10 ft. and this represents a feasible scaling for our purposes.

Since the primary purpose of this project is to test the cable configuration, the exact scaling of the motors and other components is unimportant. However, the cable characteristics must be scaled down. The use of 1/16 in aircraft cable is adequate; this gives a full-scale cable diameter of 1.25 in. The feasibility of a surveying Electronic Distance Meter (EDM) for a full-scale model was also tested.

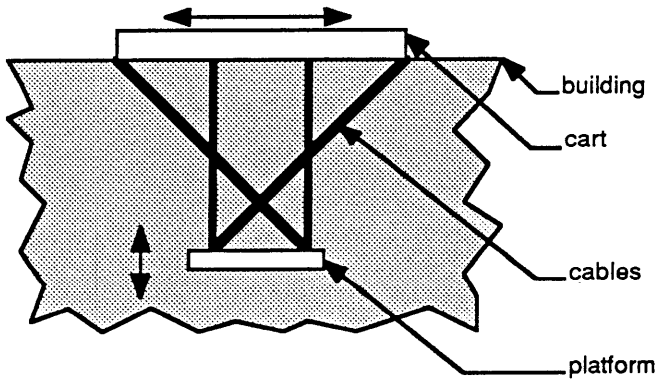


Figure 1: new cable configuration.

It was clear that a cart of some sort would have to be able to move horizontally and motors and drums mounted on the cart would coil up the cables as shown in figure 2.

An IBM PC AT is used to control Scafbot and this presented several challenges early on in the project. The choice of feedback systems and interfacing devices was important since we are trying to model the cable dynamics under various cart operations. We decided to use of a special Digital Input/Outpt PC plug-in card and a Digital/Analog converter board.

2 Technical Discusion

2.1 Mechanical system

2.1.1 Cart construction

Figure 2 shows the complete mechanical system as built. The top view of the cart is shown in figure 3. Figures 4 through 8 show the part diagrams used during construction. A large plate of aluminium is used as the base of the cart The motors, drums, encoders, and brakes are all mounted on the cart, as are three 1.25 inch diameter grooved 'wheels' which ride on a set of tracks (see figure 9). These tracks are screwed into a 4x4 inch x8 ft aluminum box section.

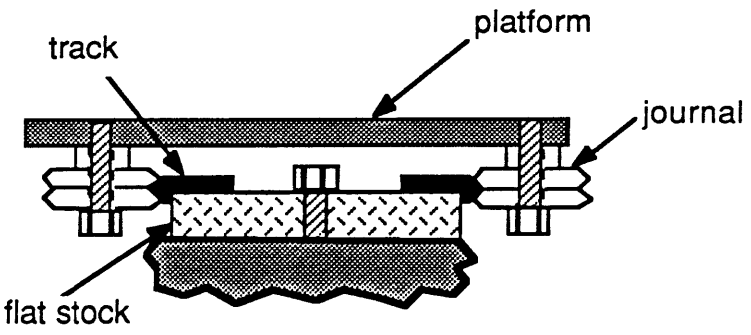


Figure 9 : track system

The wheels have very low rolling friction but high static friction coefficients with the track. This enables us to use one of them as a drive motor, moving the cart horizontally. (see figure 6 for details on the drive motor). Two idler journals and one fixed journal are required to support the cart. The drive motor was placed on one of the top two journals such that the weight of the cart and platform would increase the contact force between the journal and track. thus increasing the drive traction. The third journal was mounted using an ecentric mount so that the force between the journals and track could be varied.

The platform is raised and lowered by winding the cables up or down on two 2 inch diameter grooved drums driven by two motors (with gear reductions of 52.9:1). The weight of the platform suspended by the cables is about 10 lbs. This results in an approximate tension of 2.5 lbs and a torque on the drums of 5 inlbs. This is sufficent to slowly back-drive the motors. In order to avoid this and avoiding the necessitity of reverse-torquing the motors with controlled applied voltages, small friction power-off brakes are mounted on the drum shafts. This provides a type of fail-safe so that, in the event of an electrical power failure, the platform will not unwind the cables. It also locks the system up when it is not in use.The motors are connected to the drum shafts using HeliCal flexible couplings which eliminate the need for absolute accuracy when lining up the motors and drum shafts.

Three small optical encoders are mounted on the cart near the drive motor and two drum motors. They are connected to the shafts via 1 inch timing pulleys and belts. There is no mechanical gain between the encoder shaft and the drum shaft . Figure 3 shows the cart layout. The encoders are mounted using L-shaped brakets with slots in them. These slots allow the tension in the timing belt to be varried. The encoder and mount are shown in figure 7.

The motors operate on +-24 vdc and draw a current of approximately 0.5 amps. They produce a torque of 834 ozin and the maximum speed (at 24 v) under loaded

conditions is about 2 rev per second. This produces a horizontal velocity of 6 in per second, and a vertical velocity of 12 in per second. However, the vertical velocity was limited to 5-6 in per second in order to make it possible to keep the platform level.. The motors and drums are all mounted using mounts and 1/4 inch ID, 1/8 in thick bearings. Figure 8 shows the drum and mount. The drums were lightly threaded at 14 threads per inch with a customised tool in order to facilitate the ordered coiling of the cables. Had the cables been left to coil randomly on their own, one rotation of the drum would not necessarily coil up the same amount of cable and the platform would not always be level. The drums are secured to the shaft with recessed set screws. The cable is attached through a small hole at an angle to the drum by a set screw. The power-off brakes are mounted on the drum mounting brackets and are secured using two set screws.

Once all of the components had been built, their positions were marked on a large piece of 3/8 in aluminium plate using figure 10 as a template. Mounting holes were drilled and then tapped and the components mounted. One side of a piece of 1 in square box section was was partially removed and a piece of 1 in wide aluminium plate machined with grooves such that it can slide over the box section as a cover, see figure 11. This was mounted on the cart , and all of the cables were routed through

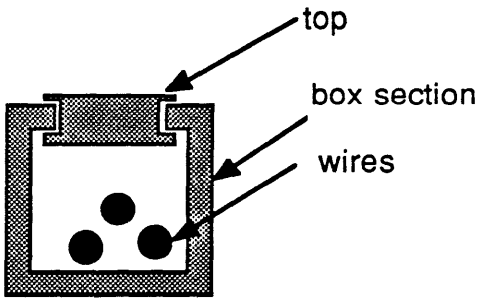


Figure 11: wiring housing

All of the wiring leads from the wiring housing to a set of 9-pin connectors mounted on the connector panel on the side of the cart (see figure 12).

2.1.2 Platform construction

The platform which is suspended by the cables is shown in figure 13, and the pulley system is shown in figure 4 . Originally the cable coming from the drum and the one attached to the extension arm were not in the same vertical plane. For this reason, the pulley system rotates about the base. However, the cables were later made to be in the same plane and the pulley system does not need to rotate. Rather than removing the press-fit bearing, the large bolt arrangement was tightened and the pulley system is no longer able to rotate.

The original cable configuration specified that the cables have to be firmly fixed to the platform when the platform was not moving. In a full-scale operation this would involve magnetic particle brakes around the cables. However, the size brake that our model needed do not exist. It was decided that we could simulate the effects of such a brake by mounting a small electrically powered friction brake, similar to the ones used to brake the drums on the cart, to brake the pulley around which the cables run. Power-off brakes were selected because of their fail-safe operation. A braking torque of 3 inlb was deemed to be sufficient. Figure 4 shows the mounting of the brake. The power to the brakes comes from a long ribbon cable reaching from the platform to the connector panel on the cart.

Three large bicycle reflectors were mounted horizontally on the platform. They are used to reflect the infra-red signal from the EDM back to a large mirror mounted on the cart at 45 degrees to the vertical plane and then back to the EDM.

2.2 Electrical system - hardware

Figure 14 shows a schematic of the interfacing of Scafbot with the IBM PC.

2.2.1 Interfacing the encoders

The encoders produce two pulses 1000 times per shaft revolution. One of the pulses leads the other one by 90 degrees, thus indicating direction of rotation.

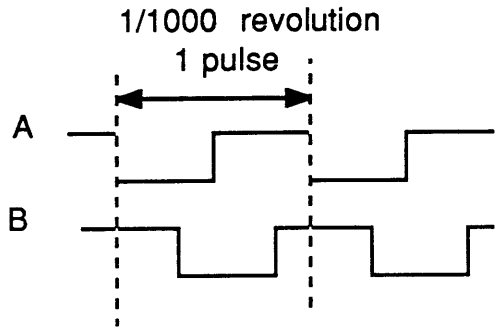


Figure 14a : encoder output

The A and B signals from the encoders are connected to the inputs of three 'A QUAD B' boards which interpret the data and count either up or down depending on the direction of the encoder shaft rotation. In this way, the A QUAD B board produces a 22 - bit number representing the absolute position of the encoder. The board analyses both the A and B encoder signals and produces four counts per ' pulse' of the encoder and thus increases the accuracy of the encoder by a factor of 4. Thus one revolution of the encoder produces 4,000 counts, which for the drive motor results in 4.71 inches of travel horizontally. This resolution results in one A QUAD B board count representing 0.0012 inches of motion.

The A QUAD B board requires several signals to operate. They are as follows;

CLR - a signal telling the board to reset itself to 0 - active low

CLK - a 2 MHz square wave clock signal

READ/LATCH - normally low, but set high when a reading is desired

Only the lowest 16 bits of the 22 bits produced are read, resulting in 40,000 divisions in the 4ft horizontal track (in 0.0012 inch steps) - perfectly adequate for our purposes.

2.2.2 Interfacing the EDM

The EDM has an RS232 adapter and can be plugged directly into one of the computer's serial ports. In its tracking mode, the EDM takes measurements every 0.5 seconds, and in its more accurate mode takes measurements every 4 seconds.

2.2.3 Powering the motors

The motors run on +- 24 vdc at 0.5 amps peak but the Data Translation Digital to Analog (DT2815) board only produces +-5 vdc at 20 mA. It is therefore necessary to amplify the the DT2815 signal using a high-power amplifier. The simplest type of amplifier is the non-inverting op-amp circuit. Three 3573am op-amps are used to power the three motors. Figure 15 shows the circuit diagram for the amplifiers. By using op-amps, the voltage to the motors can be varied between -24 v and +24 v in infinetly small steps. However, the DT2815 can only vary its output with 12-bit resolution. The output of the amplifiers cannot have a higher resolution than its input, thus the voltage to the motors can be varied in steps of 0.0117 v.

2.2.4 Powering the brakes

The brakes operate on +24 vdc . The larger cart brakes require 0.37 amps and the smaller platform ones need 0.18 amps and when power is aplied to them, they release. Since they are controlled by the same DT2815 board as the motors, they need amplification of the signals to operate. However, since they either need 0 v to be on or 24 v to release, the use of an op-amp is not required - simple relays can be used. The relays must be able to handle up to 0.4 A yet be able to switch with less 20 mA across the coil. Ideally, solid-state relays should be used for this operation, however, miniture mechanical ones seem to work. The switching time of a mechanical relay, though slower than that of a solid-state one , is small enough when used with a component such as a brake.

2.2.5 Power requirements of the system

Assuming that all three motors and all brakes are powered at the same time, the current requirements of the system would be;

$$I = 3*(0.5) + (2*0.37) + (2*0.18)$$

$$I = 2.6 \text{ A}$$

The power supplies used are only rated at 2.4 A, however, the controlling software limits the number of motors on at one time to two and so the power requirements are satisfied.

2.3 Interfacing of the IBM PC

2.3.1 Using the Data Translation DT2815 board

The DT2815 is a general purpose Digital to Analog (D/A) converter which plugs into one of the expansion slots of an IBM PC. It can be used as either a current or voltage source, or a mixture of both. It can be used in a bipolar (-5 to +5) or unipolar (0 to 5) mode. Its resolution is 12-bit (decimal 4,096) and its current output is limited to 20 mA. Figure 16 shows the pin connections between the DT2815 and the outside world. The motors require 3 voltage outputs and the brakes 4 voltage outputs. The DT2815 has 8 channels of voltage output organised in two sets of 4 (J1 and J2).

Before the board can be used for anything it must be configured to the users needs. Since the motors expect -24 to +24 vdc from the amplifiers and they expect -5 to +5 vdc, the DT2815 was configured to operate in its bipolar mode (although the brakes need only +24 vdc, their channels were set for bipolar operation for simplicity).

When using any extra plug in cards in the IBM PC, the base address of the board must be set so as not to coincide with othe existing boards. The base address is set by changing various 'jumpers' around the board. The DT2815 was set up for an address of $224 (hex224). The mode was set to bipolar by changing another jumper.

2.3.2 Using the Scientific Solutions BASEBOARD

The BASEBOARD is a 96-line digital input/output controller. It has 4 ports of 24 lines each. Each port is subdivided into 3 bytes (8 lines each). Figure 17 shows the connection to one of the ports. The BASEBOARD is used as an interface between the A QUAD B boards and the IBM PC. The 3 bytes can be configured in any one of 9 combinations of input and output. Since the A QUAD B board produces 16 bits of useful information and requires 2 bits of control (CLR and READ/LATCH, CLK comes from another circuit), two of the bytes have to be input and the other one output. The base adress of the BASEBOARD is set at 1040.

2.3.3 The Electronics board

This board was built as a simple connection board between the DT2815 , the BASEBOARD and Scafbot. Figure 18 is a diagram of the set up of the board. The power supplies, op-amps and circuits, relays and A QUAD B boards are all mounted on this board. The left hand side of the board is output and the right is input (apart from the A QUAD B board-BASEBOARD connections).

2.4 Electrical - software

2.4.1. Software control of the DT2815

The software control of the DT2815 is fairly straightforward. The base address of the board is $224. The board has to be reset before it can be used as a voltage source. In order to do this, address $225 has to be checked that it contains the value $4. If it does not, then $0 has to be sent to it.

Once the DT2815 has been reset, the process for obtaining a voltage at one of the channels begins. First of all, the number , between 0 and 4096, to be sent to the channel

must be separated into its high order byte and low order nibble (half a byte). The low order bits are then shifted by 16 and the channel address and mode select added to it. The mode select is 1 for voltage output. The channel address represents the channel (0-7) to which the voltage is to be sent. Figure 19 shows the data format,

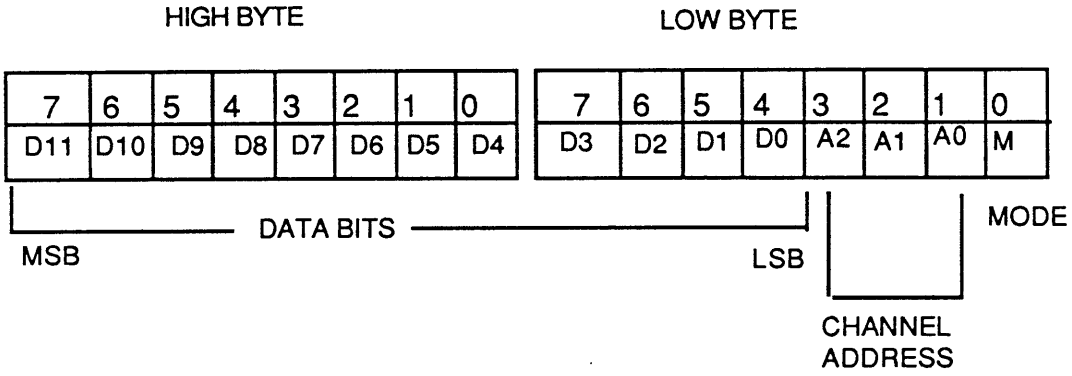


Figure 19: data register format for DT2815

Before sending the low byte, the address must be checked to see if the board is ready to accept the low byte. This is done by waiting until address $225 = 0, at which point the low byte is sent. The board must then be checked whether it is ready to accepet the high byte, if it is then send it. If the board does not want to accept any bytes then a timeout error occurs and the program is halted.

2.4.2 Software control of the BASEBOARD

The base address of the BASEBOARD is set at 1040. Each of the 4 24-line ports is subdivided into 4 bytes : CONTROL WORD, PORT C DATA, PORT B DATA, PORT C DATA. The control word is used to configure ports A,B and C for any combination of input/output . Data is then sent to or received from the ports using the PASCAL command

PORT[..] := xx to output onto the port, or,

xx := PORT[..] to input from the port.

2.4.3 Software control of the EDM

The EDM continually sends serial data into the RS232 serial port of the IBM PC. Information about the pressure, temperature and humdity settings as input by the user are sent as well as the distance recorder. Since we are only interested in the actual distance measured, the rest of the information is screened out by realising that the distance begins with a '+' and ends with an 'm'. For example, the data may appear as follows;

xxxxx xx +2307m x xxx xx

The distance is 2307 mm. A short software routine is used to only record the characters between the '+' and 'm'.

2.5 PASCAL control of Scafbot

A PASCAL program was written in a modualr form to implement PD-seriec control of Scafbot. Figure 20 is the flowchart of the main program. Figure 21 is the flowchart of the PD implementation. Appendix A contains the entire PASCAL program.

3 Conclusion

Scafbot has been completely built and is controllable from the IBM PC. It is currently possible to position the cart to within 0.005 inch of a desired location. The optimal parameters of the PD control system have not yet been found and a PID control system is currently being developed.

The EDM's accuracy of 1 mm renders it rather useless since the encoders are providing accuracy of 0.001 inch. However, it would be necessary in a full scale model where an acuracy of 1mm over a range of 100 m is required. Although qualitative tests of the new cable configuration have not yet been carried out, the system appears to be very promising.

A full scale model would have to use magnetic particle brakes on the cables rather than friction brakes on the pulleys since the friction between the pulley and cable is the limiting factor in determining the lateral stiffness of the cable configuration.

Figures

All dimensions given are in inches

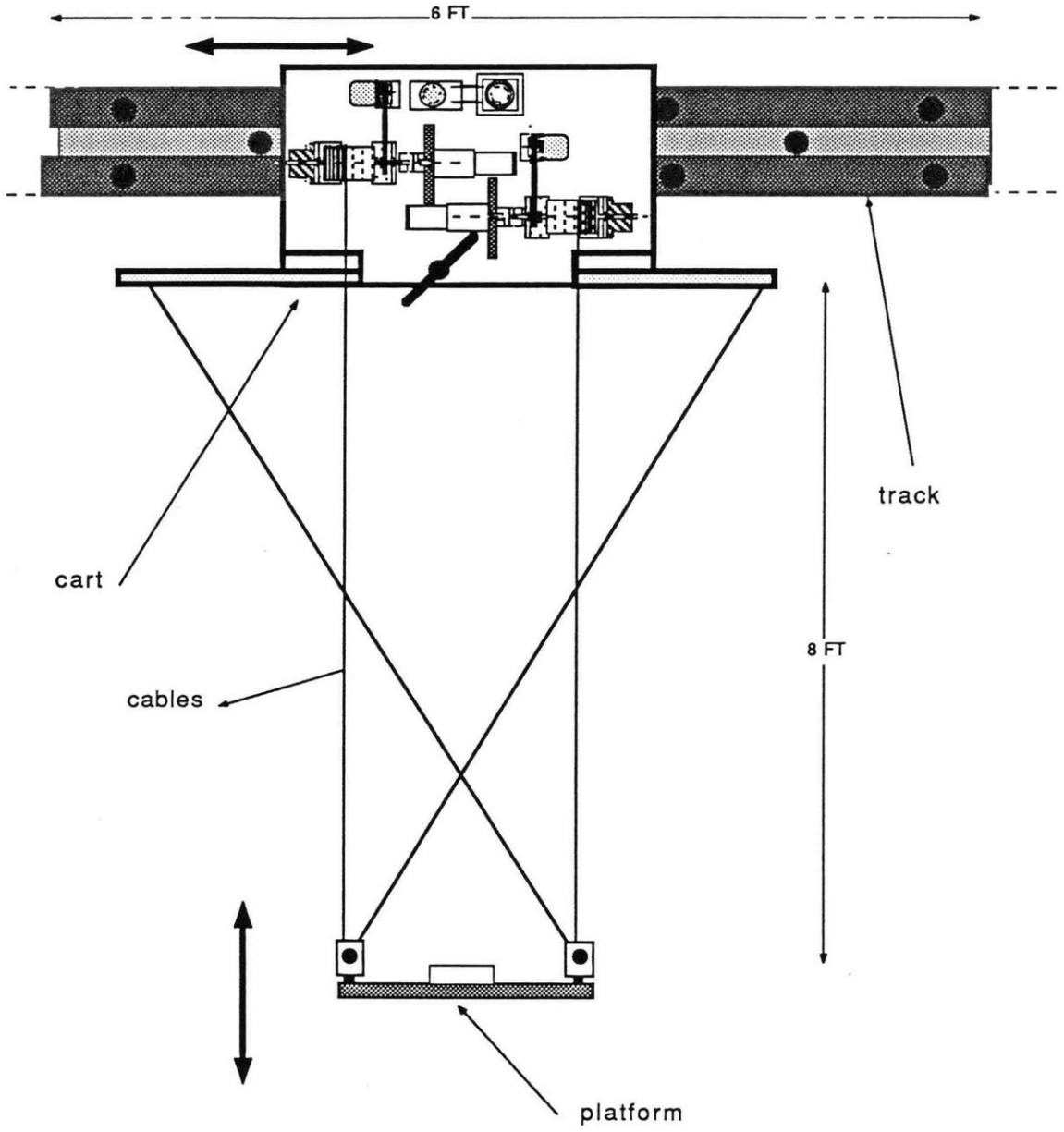


Figure 2: complete Scafbot system

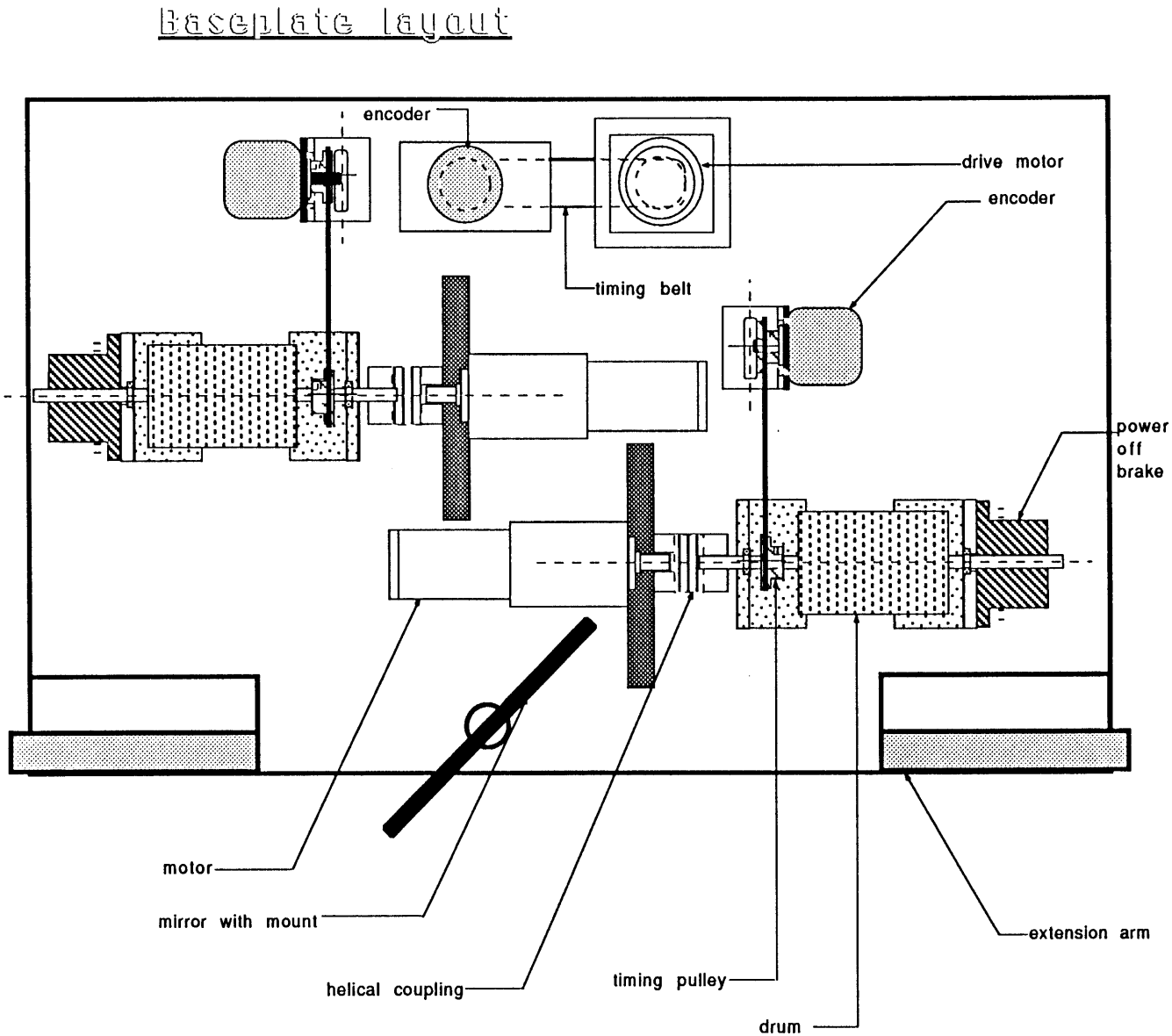


Figure 3: top view of cart

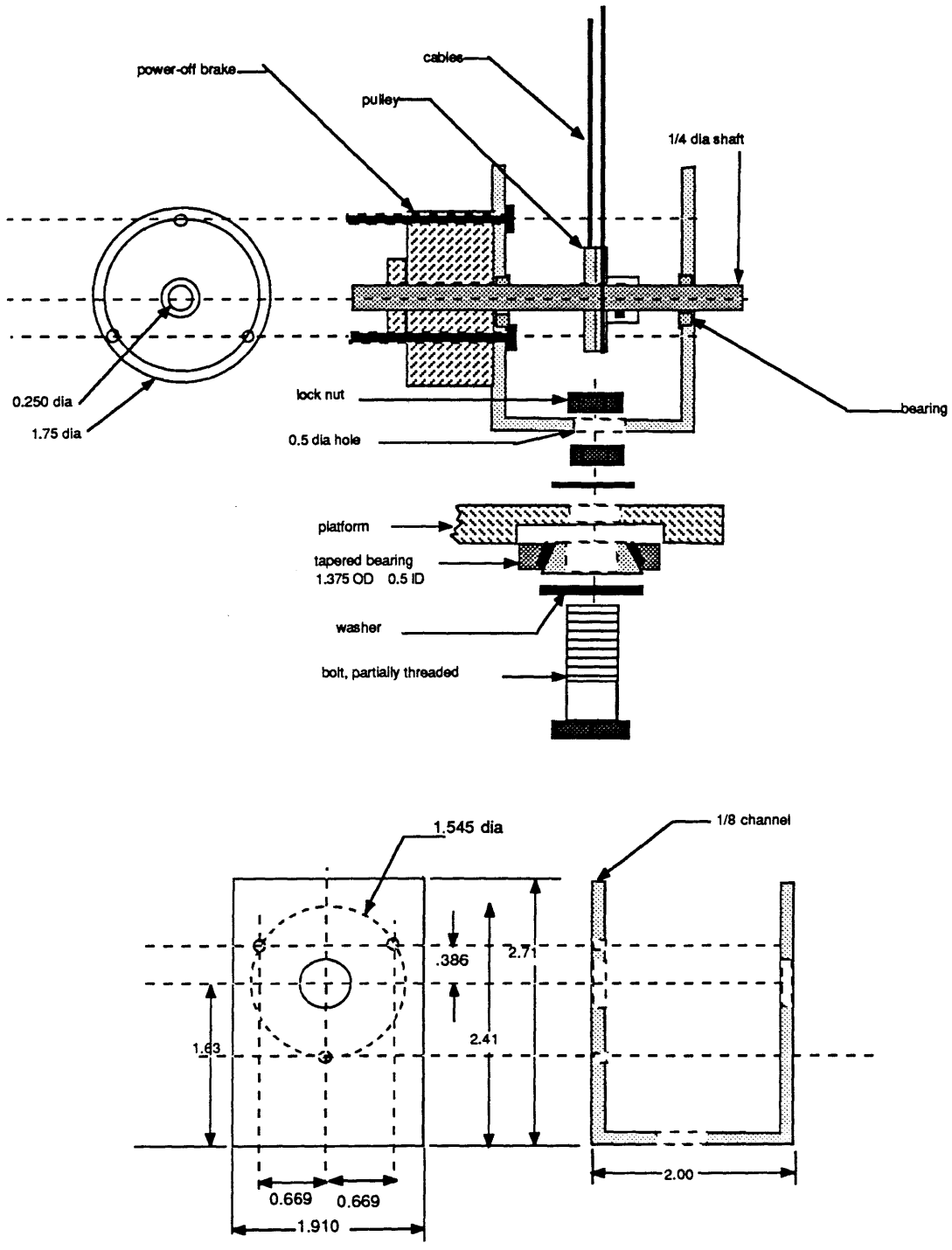


Figure 4: pulley system

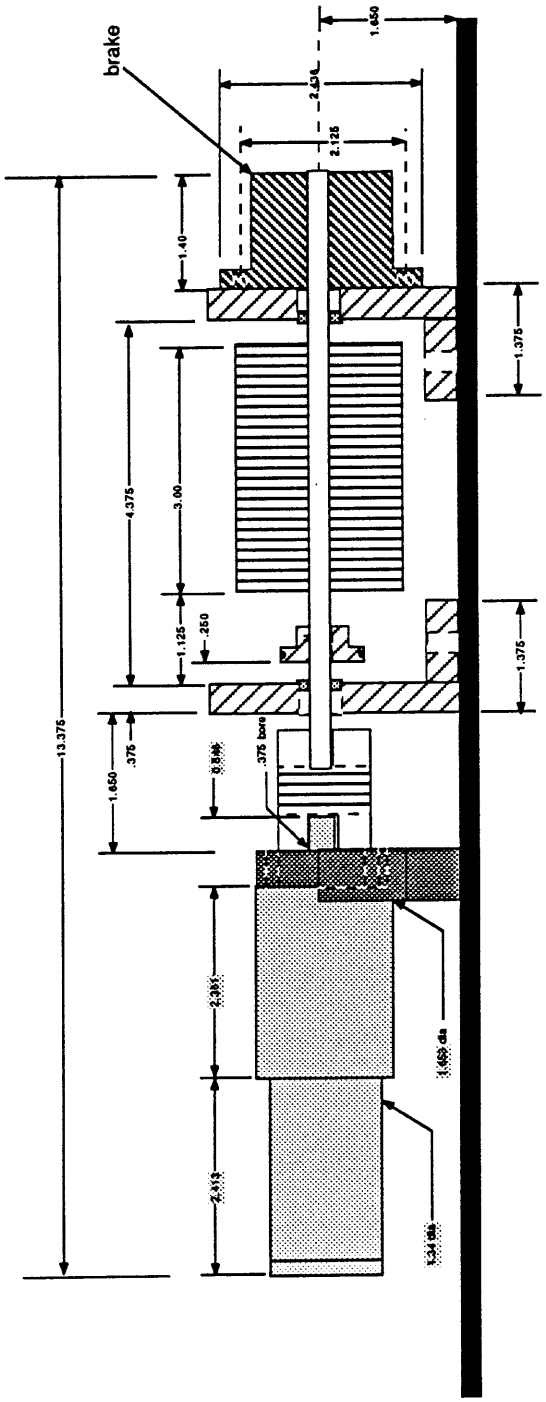


Figure 5: side view of motor and drum

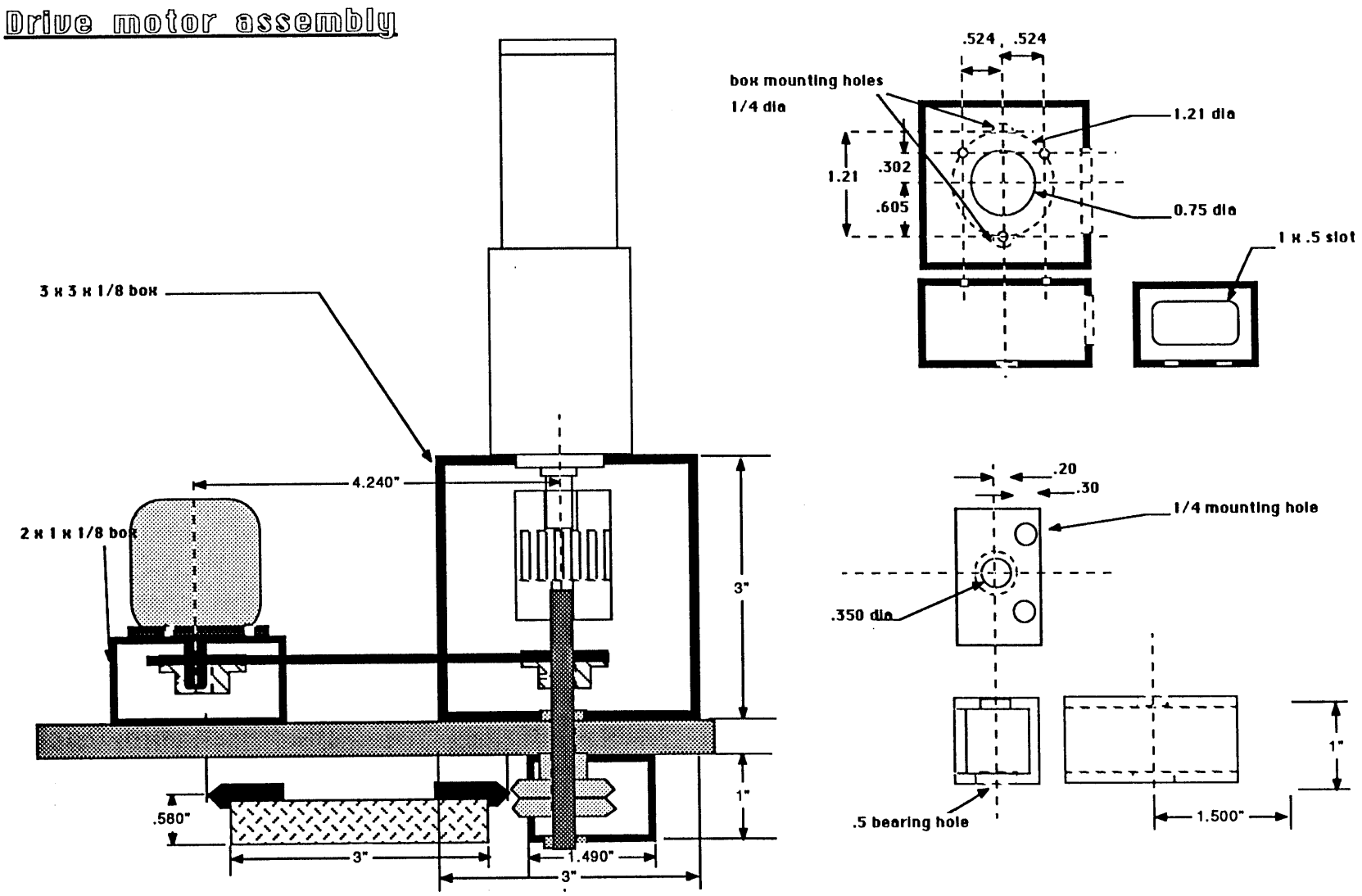
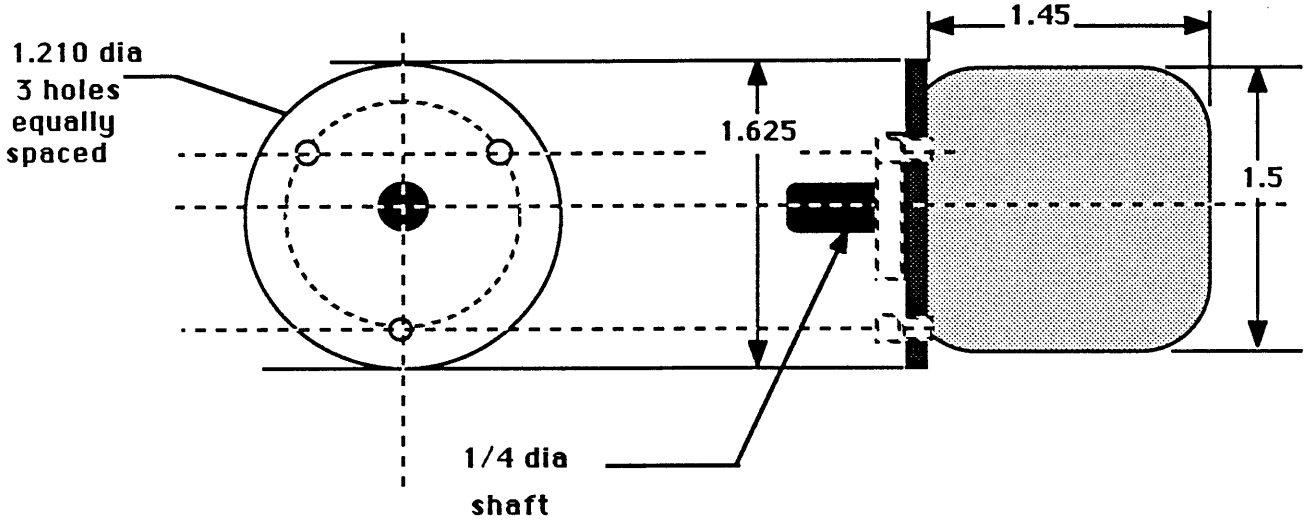
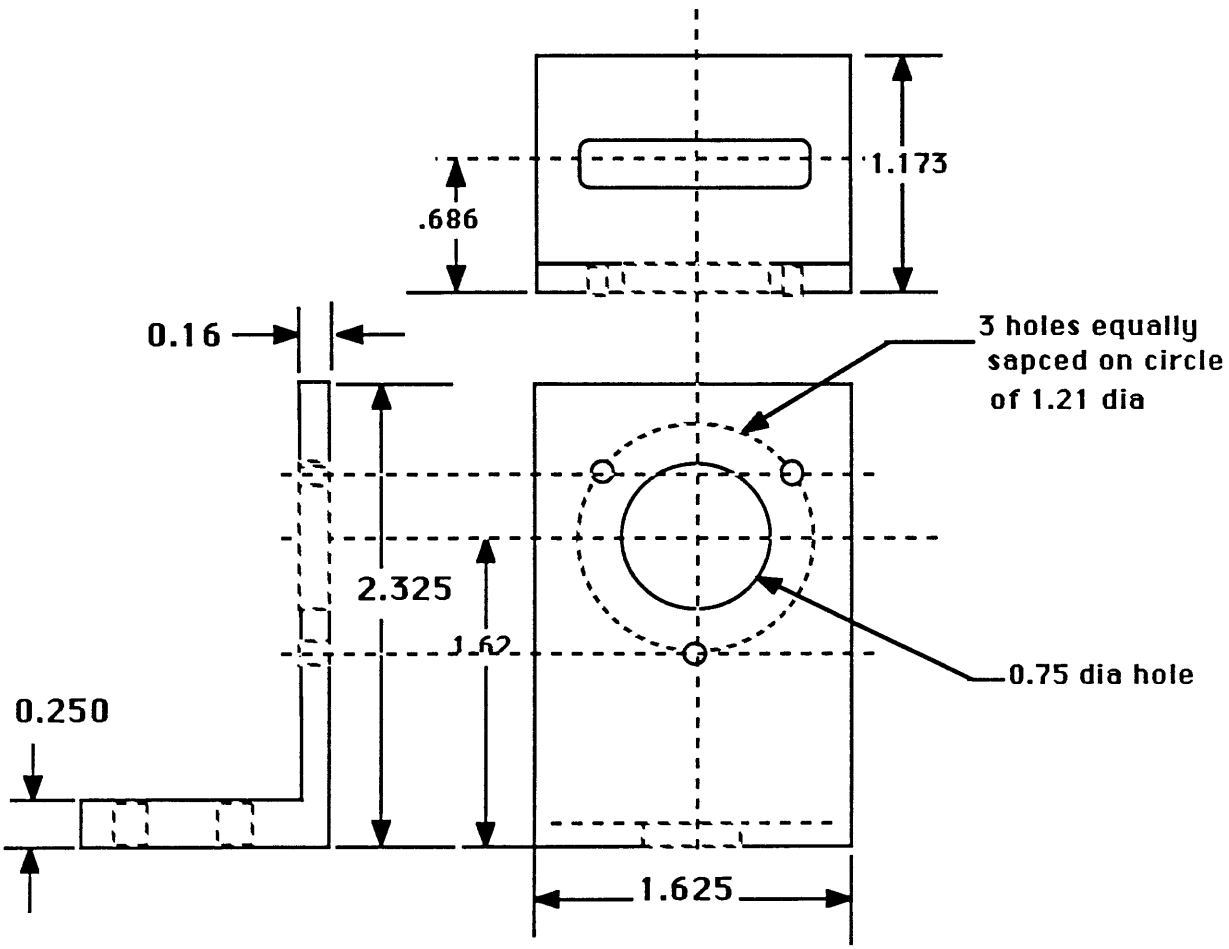


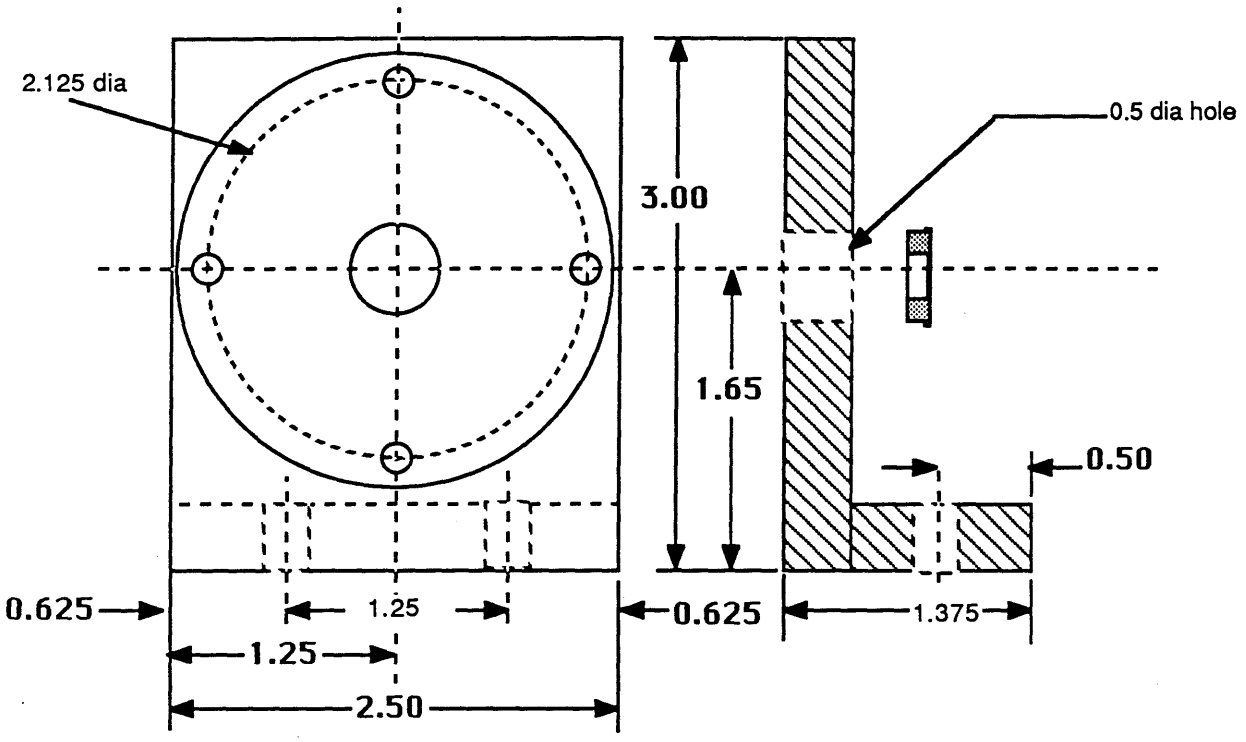
Figure 6: drive motor assembly

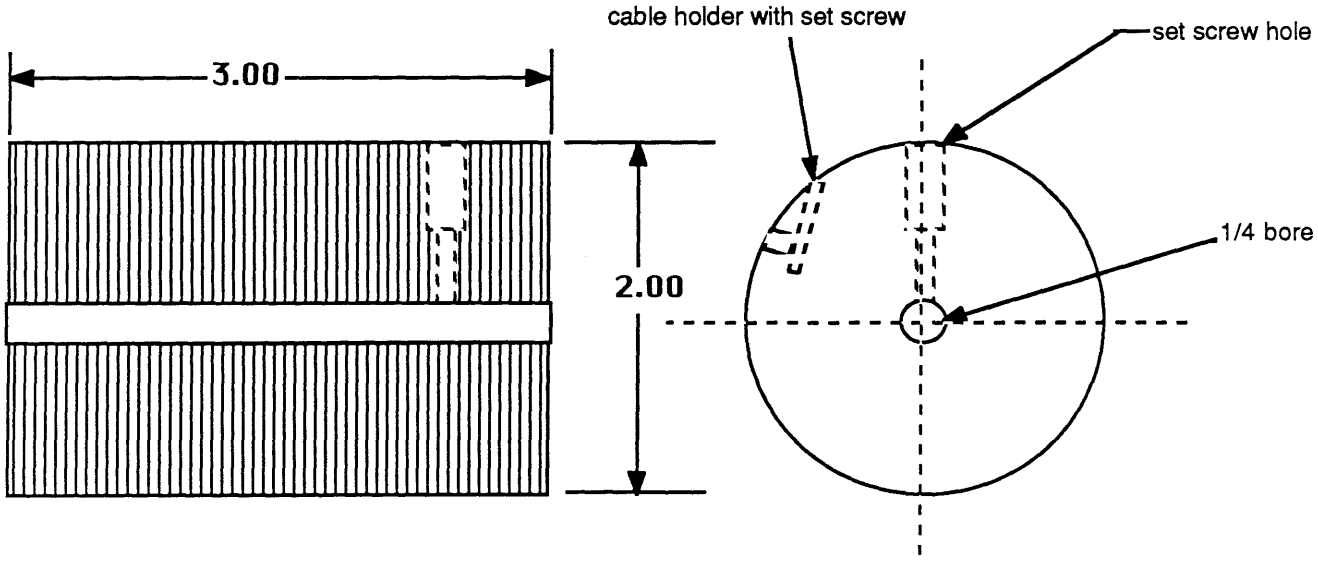




Encoder and mount

Figure 7: encoder and mount





DRUM AND MOUNT

3/29/87 - built

Figure 8: drum and mount

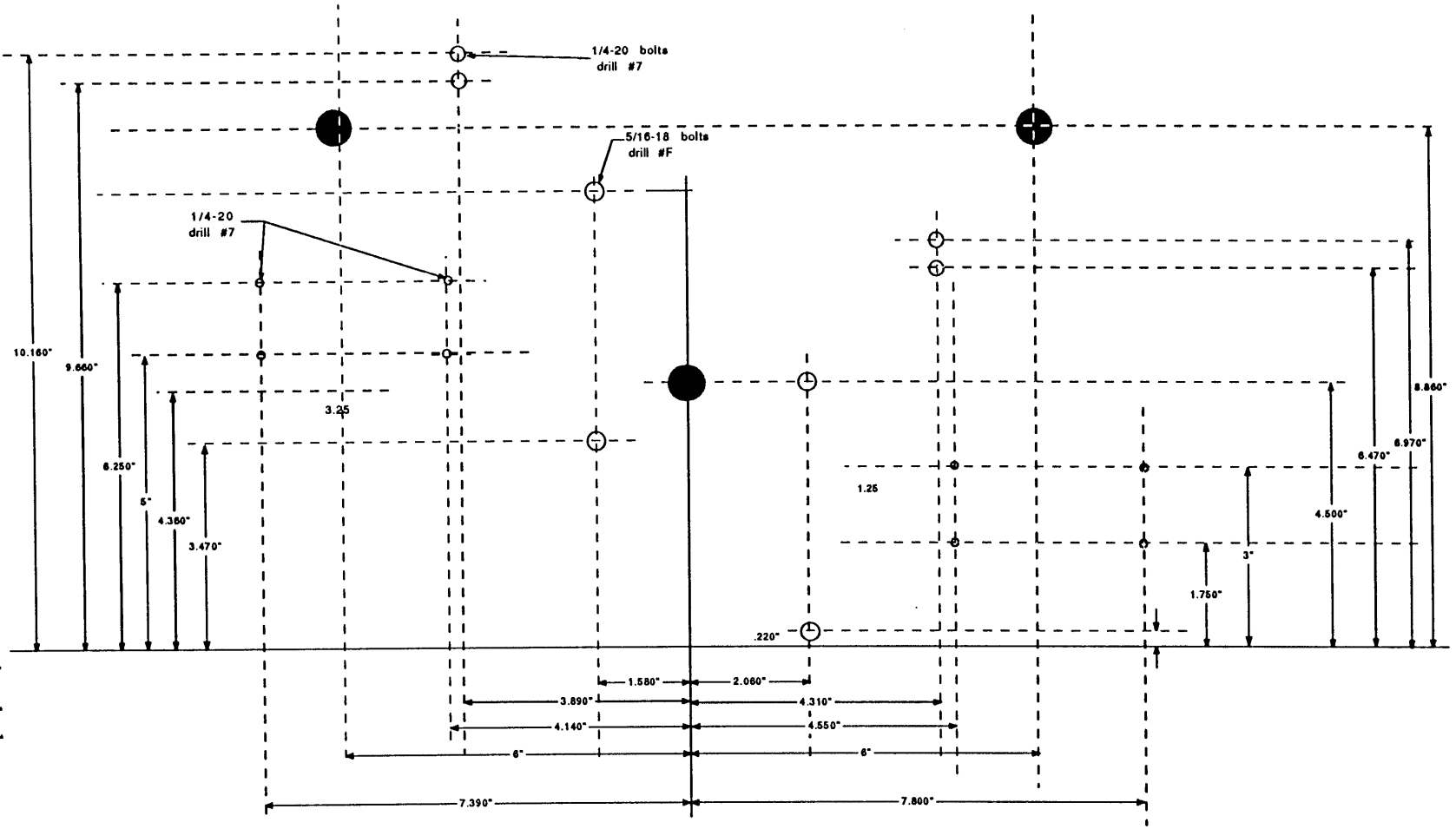
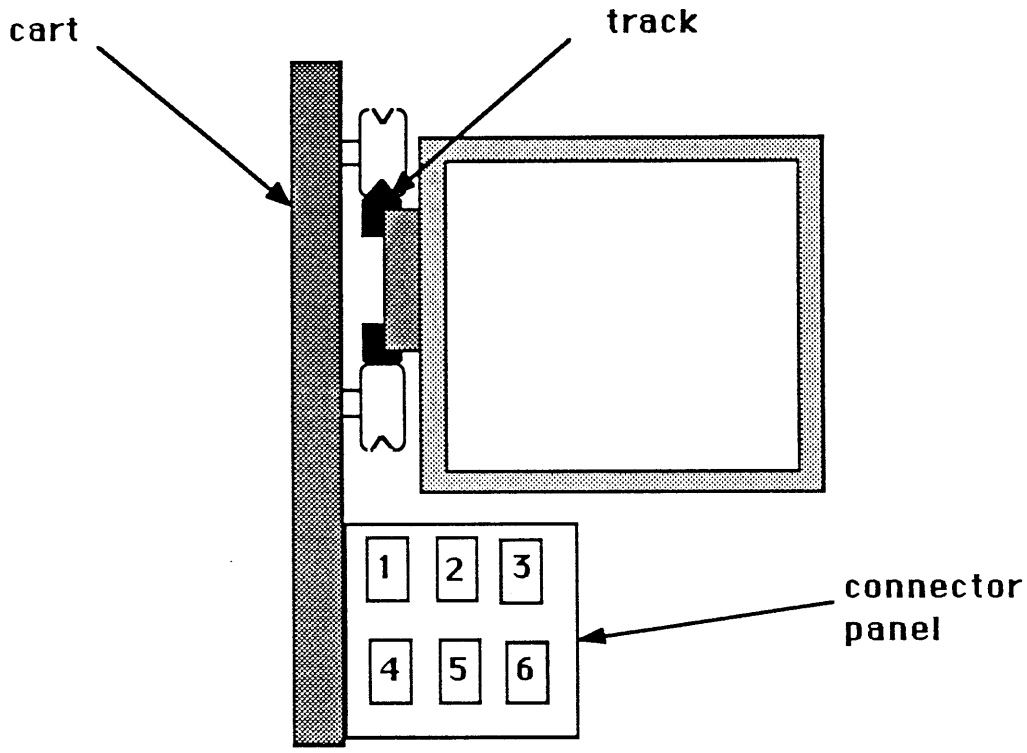


Figure 10: layout of cart mounting holes

Baseplate layout

3/25/87
BASEPLATE

Connections between cart and outside world



1 connector to left encoder

2 connector to right encoder

3 connector to drive motor encoder

4 power to motors and cart brakes

5 clineomenter connections

6 power to platform brakes

Figure 12: connector panel on the side of cart

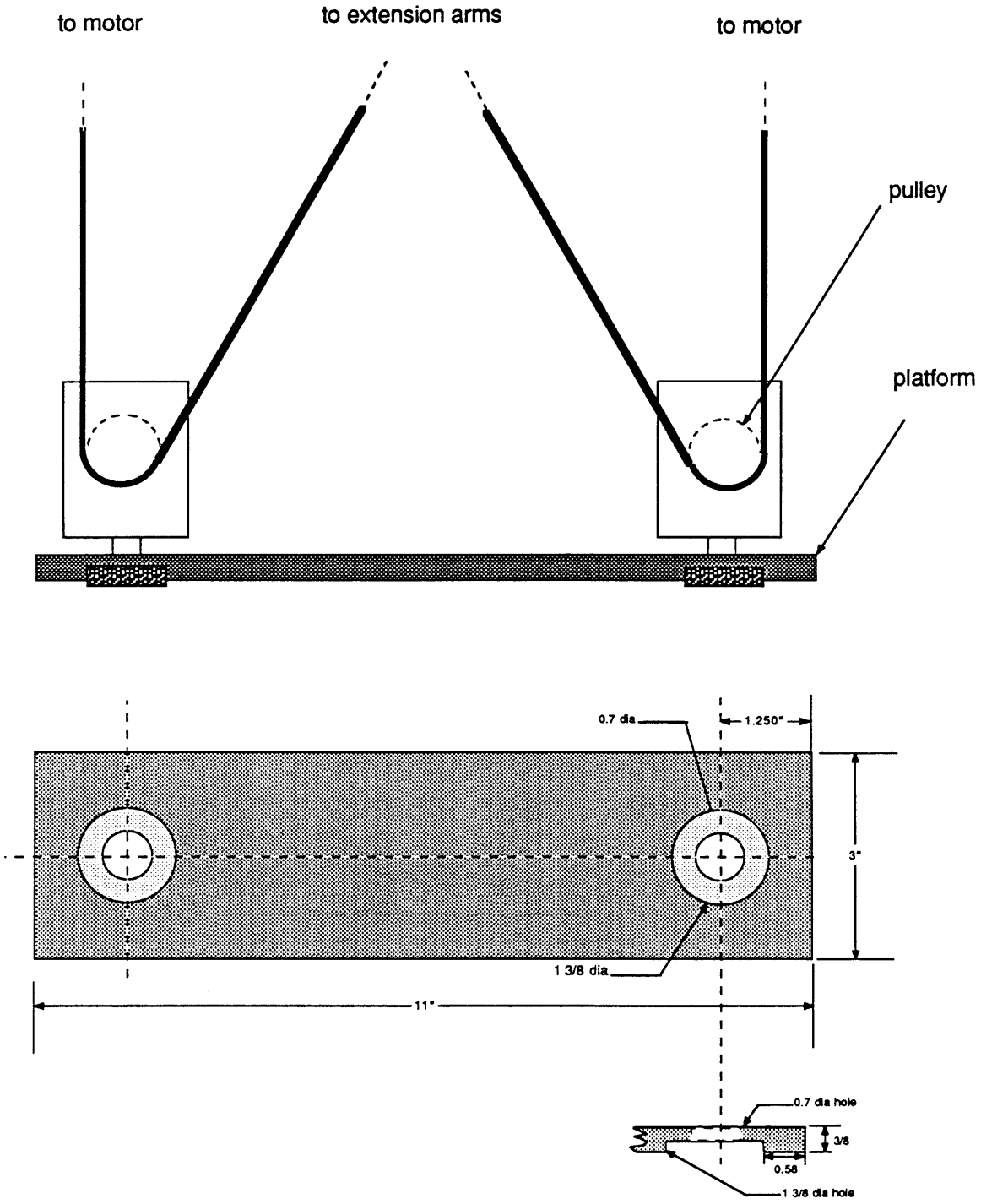


Figure 13: drawing of platform

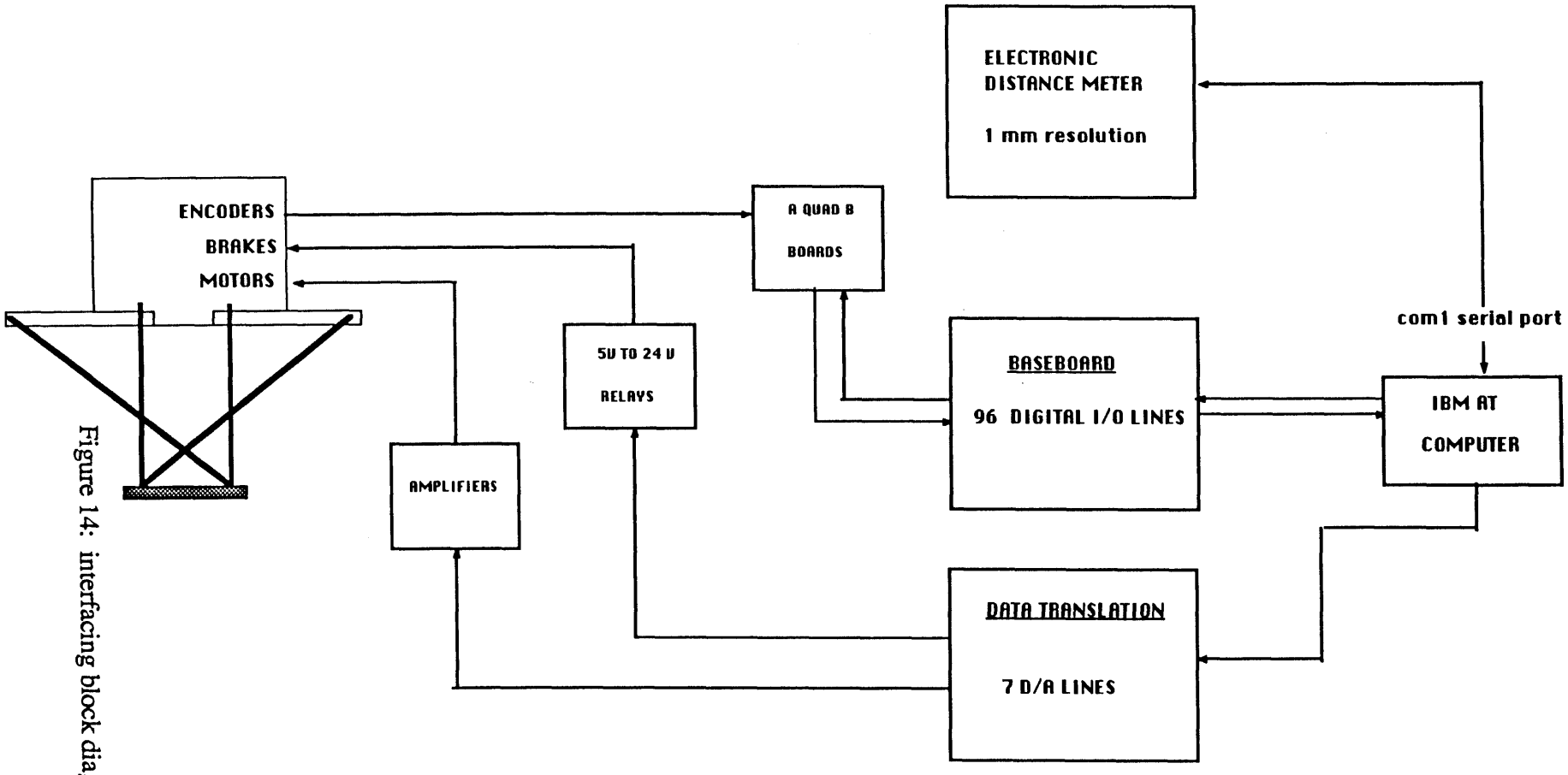
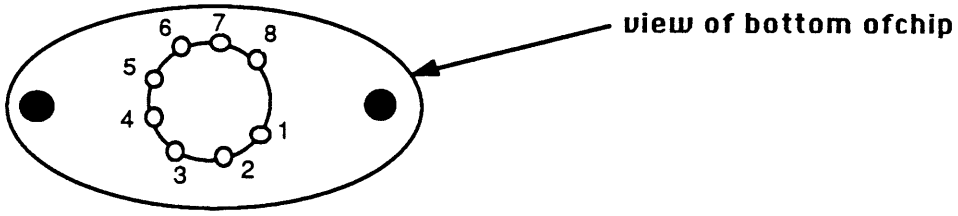


Figure 14: interfacing block diagram

Wiring of 3573am amplifiers

Burr-Brown 3573 am high-power op-amp



Simple op-amp non-inverting amplifier circuit

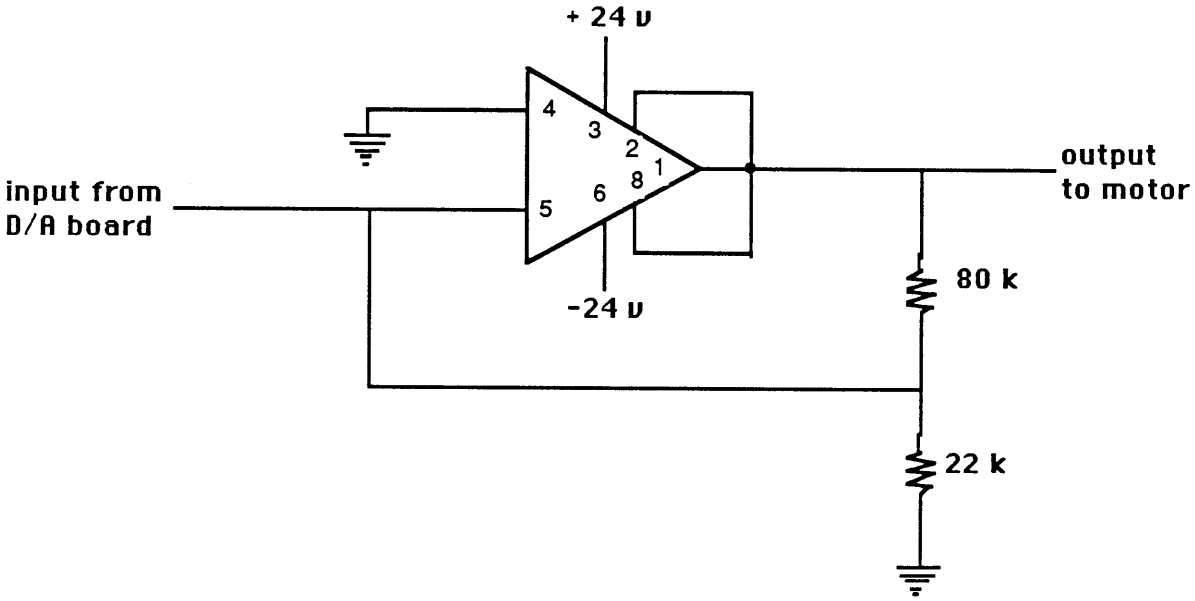


Figure 15: op-amp circuit

Interfacing of the Data Translation

DT 2815 Digital/Analog board

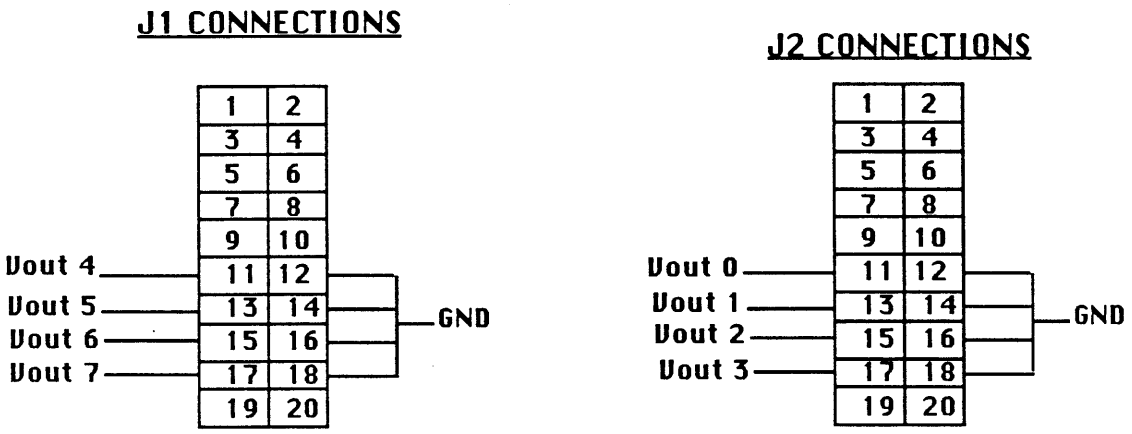
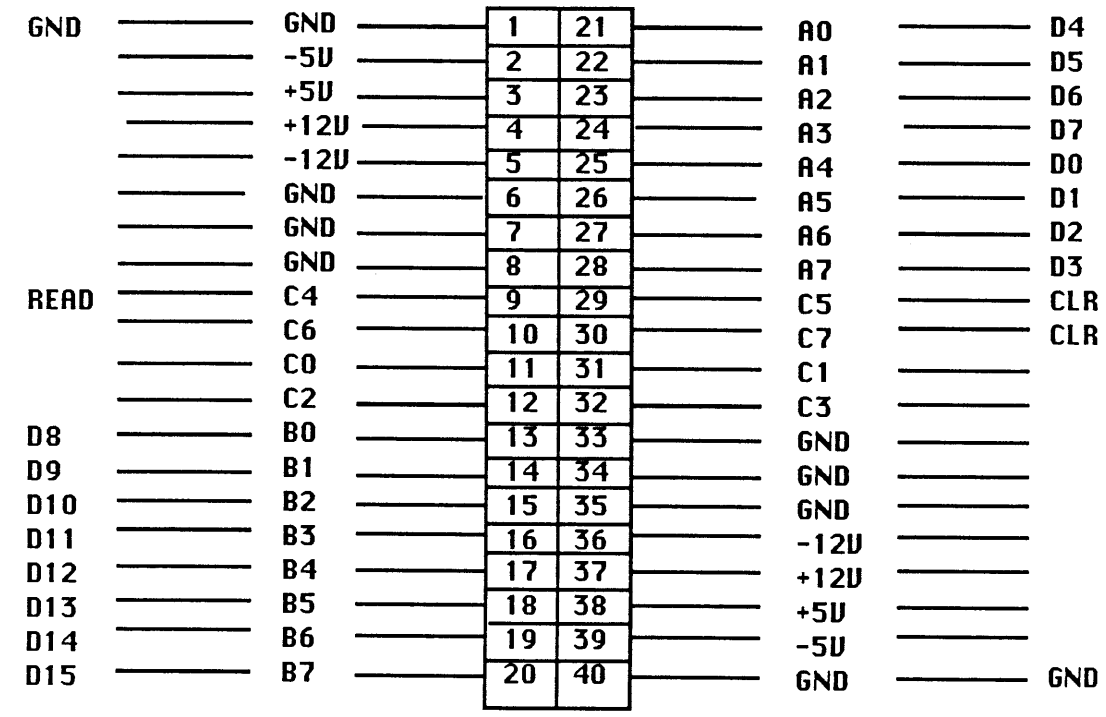


Figure 16: connections to DT2815 board

Connections between the A QUAD B BOARDS and the IBM PC BASEBOARD



The A QUAD B boards have 16 output data bits (D0- D15)

The board requires a 2 Meg clock and a CLEAR and READ/LATCH signal

Figure 17: connections to BASEBOARD

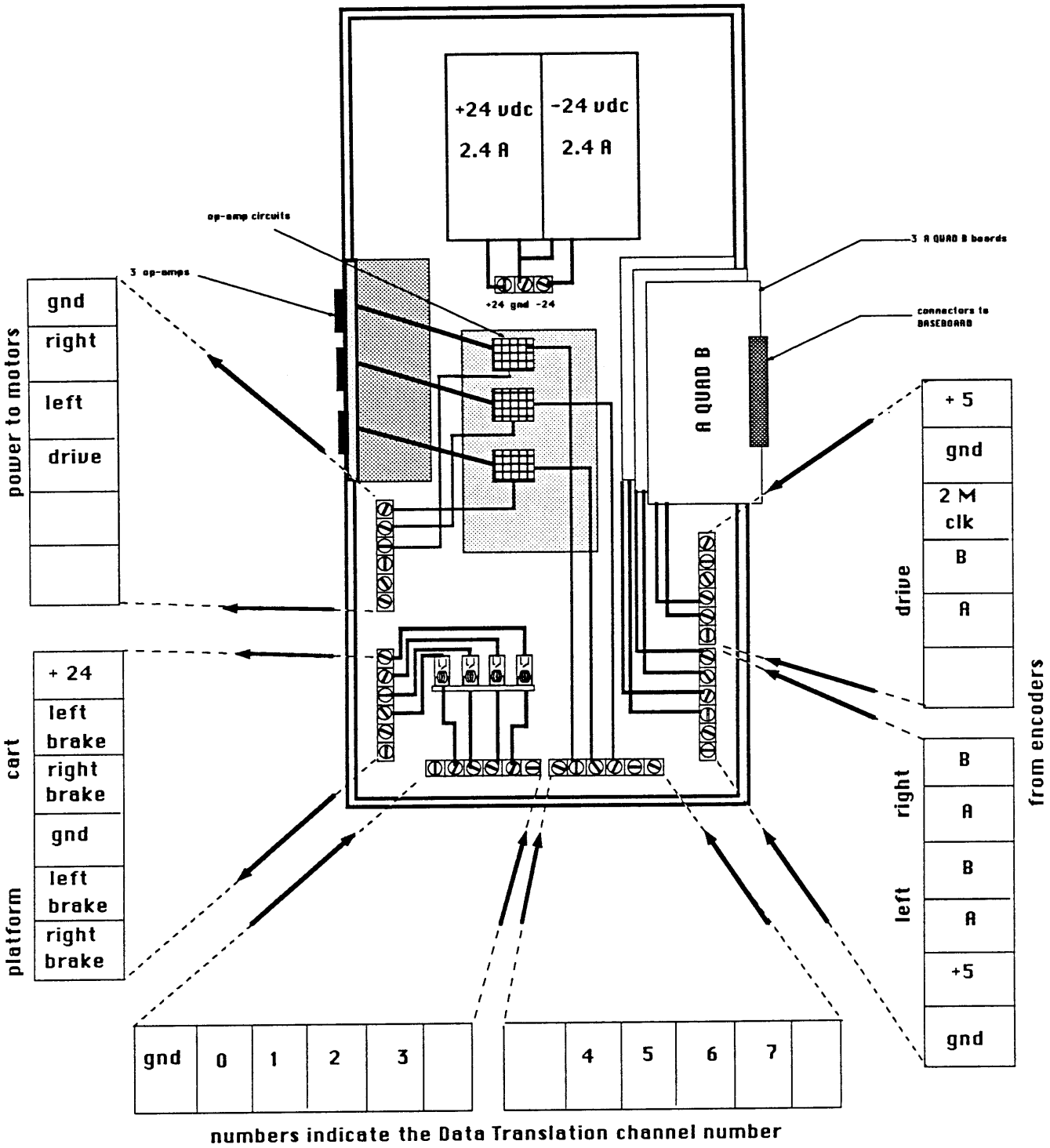


Figure 18: electronics board

Main Program

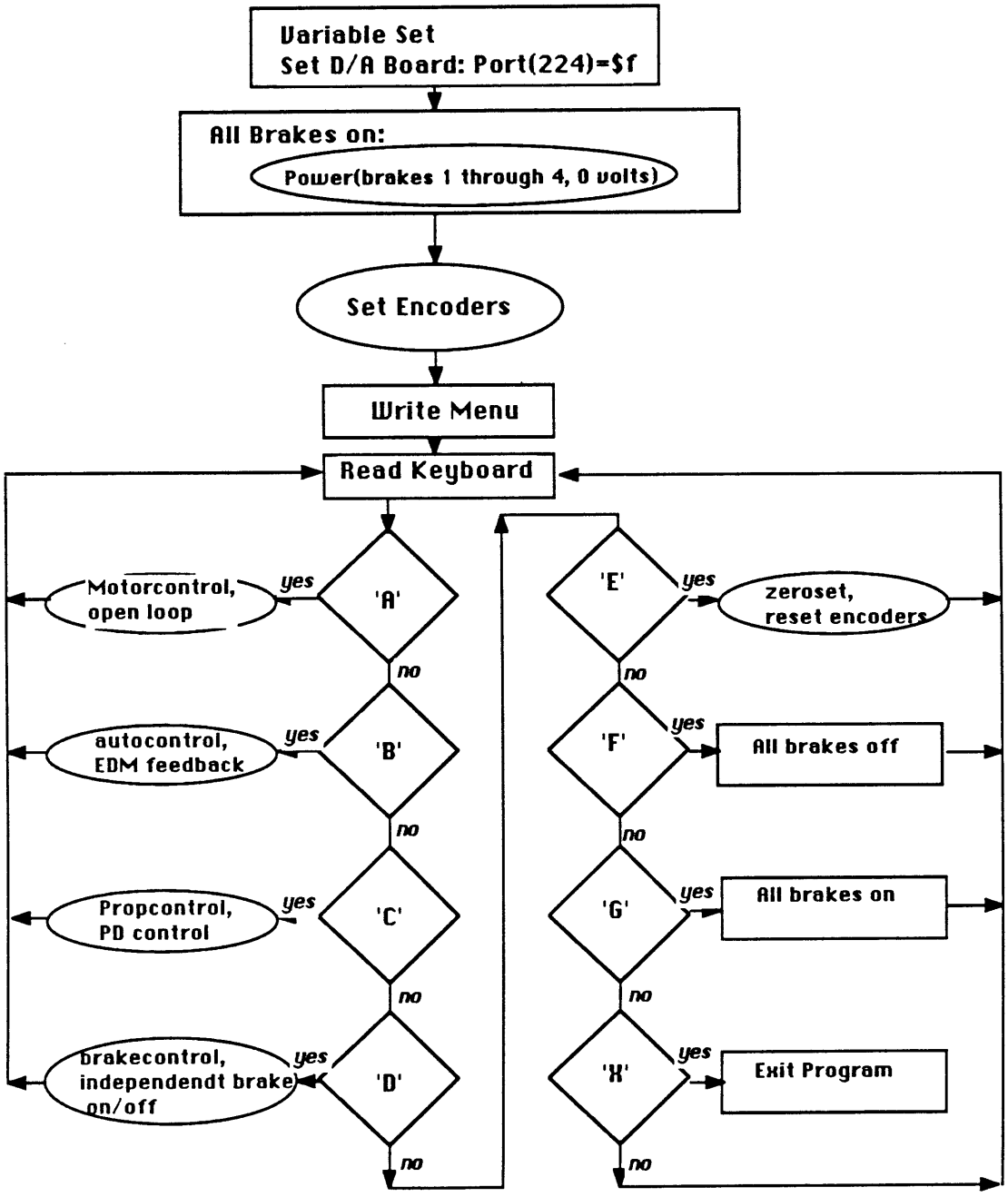


Figure 20: main program flowchart

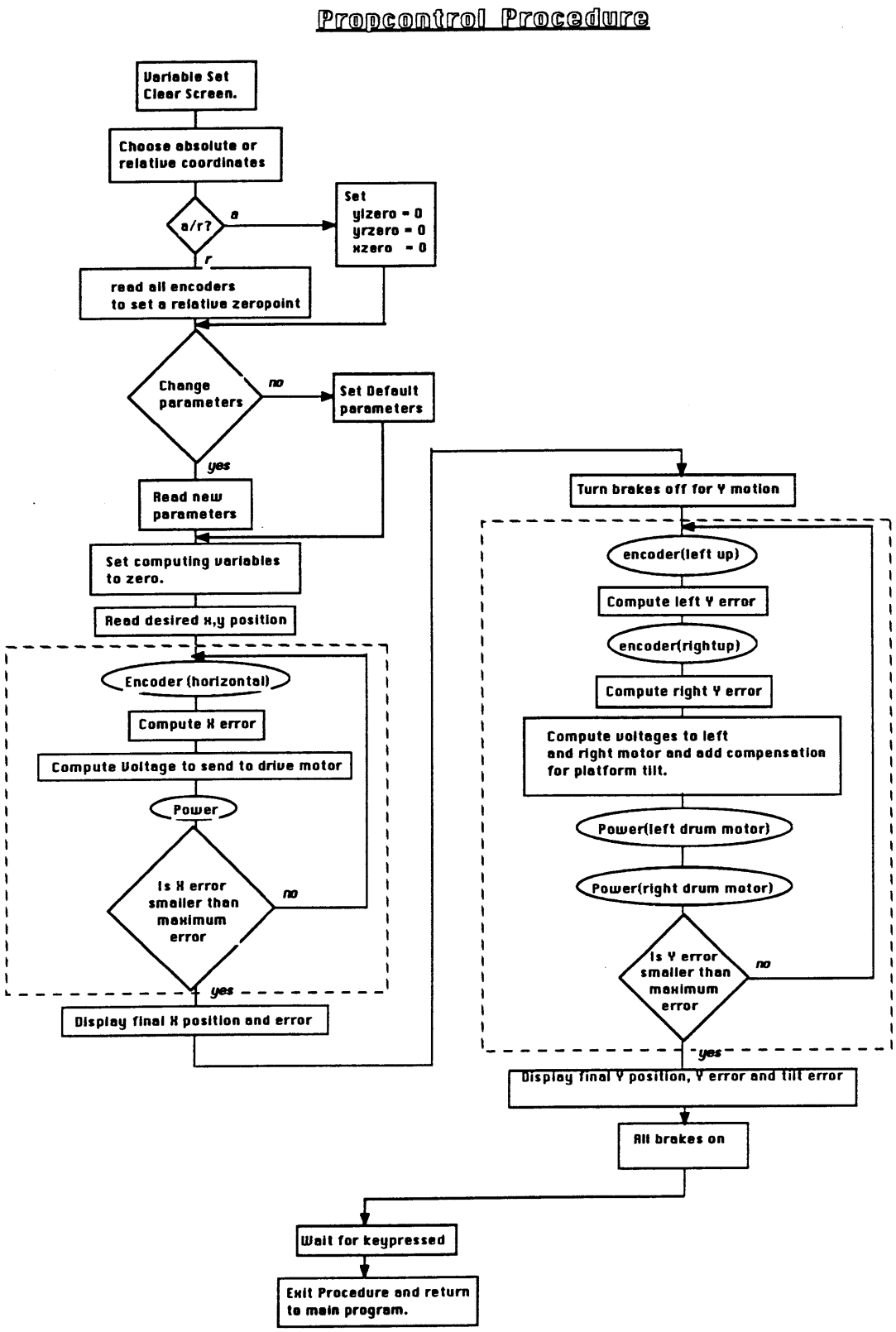


Figure 21: PD-control flowchart

Appendix A

```
program dtcontrol;

type      { define data types used globally in program }
  hexstringtype  = string[4];

var       { define global variables }
  a , b ,  voltage                    : real;
  sendvoltage, ba                     : integer;
  xzero,ylzero,yrzero                 : real;
  brakes                              :boolean;
  xinch,yinch                         : real;
  channel ,c,i,g,v                    : integer;
  hexvoltage                          : hexstringtype;
  lbyte                               : integer;
  hbyte                               : integer;
  choice                              : char;
  motor ,dir                          : integer;
  edm , x , time, total               : integer;
  average,startdistance               : real;
  t       ,auto                       :boolean;
  going ,q                            :integer;
  horizontal, leftup, rightup         :integer;
  accelrate,deaccelrate, volt,distance   :real;
{*************************************************************}


procedure lowbyte;
  var
    i0 , f0 , stat0 : integer;
  begin;
    i0:=0;
    f0:=0 ;
    repeat;
      stat0:=port[$225];
      if stat0=$0 then f0:=1;
      i0:=i0+1;
    until (i0=20) or (f0 = 1);
    if f0=0 then write('ERROR in LOW BYTE');
  end;

{*************************************************************}


procedure highbyte;
  var
    i1 , f1 , stat1 : integer;
  begin;
    i1:=0;
    f1:=0 ;
  repeat;
    stat1:=port[$225];
    if stat1=$10 then f1:=1;
    i1:=i1+1
  until (i1=20) or (f1 = 1);
  if f1=0 then write('ERROR in HIGH BYTE');
end;

{*************************************************************}

procedure reset;
  var
    count , i2 , f2 , test : integer;
  begin;
    i2:=0;
    count := 0;
```

```
    f2:=0;
    repeat
    count:=count+1;
    repeat
      i2:=i2+1;
      test:=port[$225];
      if test=$4 then f2:=1;
    until (i2=20) or (f2 = 1);
    if f2=0 then port[$225]:=$0;
    until (count = 2) or (f2=1);
    if f2 = 0 then writeln('RESET ERROR')
  end;

{*************************************************************************}


function hex( num : integer) : hexstringtype;
  var
    hexvalue , temp : string[16];
  begin
    hexvalue := '0123456789ABCDEF';
    temp     := '';
    repeat
      temp := concat( copy(hexvalue,((num mod 16)+1),1) , temp);
      num  := num div 16;
    until num div 16 = 0;
    temp := concat( copy(hexvalue,(num+1),1) , temp);
    hex := copy(temp,1,4);
  end;

{*************************************************************************}

function getabsolvoltage ( num : real ) : integer;
  begin
    getabsolvoltage := round(((5+num)/10*4095));
  end;

{*************************************************************************}

procedure power(channel:integer;voltage:real);
 begin;
  sendvoltage := getabsolvoltage(-voltage);
  lbyte := (lo(sendvoltage*16) + (channel*2) + 1);
  hbyte := hi(sendvoltage*16);
  lowbyte;                                    {check if ready for low byte}
  port[$224]:=lbyte;                          {send low byte}
  highbyte;                                   {check if ready for high byte}
  port[$224]:=hbyte;                          {send high byte}
end;

{*************************************************************************}



 procedure readedm;
  var
    a,b,c       : string[1];
    e           : string[20];
    d,got,f1,lp,pr : real;
  begin
    str(3:1,c);
    e:='';
    f1 := 0;
    got:=0;
```

```
   d:=0;
   repeat
     read(aux,a);
     d:=d+1;

     if ( f1=1) and (a='m') then
       begin

         f1 := 0;
         val (e,edm,x);
         got:=1
       end;
     if f1 =1 then    e:=e+a;
     if (a='+') and (f1 = 1) then e:='';
     if a='+' then f1:=1;
   until (got=1) or (d = 10);
 end;                  {end of readedm}
{*************************************************************}

procedure measure;

begin
time := 0;
total := 0;
  for  x := 1 to 3 do
    begin;
      time := time + 1;
      readedm;
      total:=total+edm;
    end;
    average:=total/time;

  if not auto then
   begin;
    gotoxy(1,20);
    write('Average is ');
    writeln(average);
   end;
end;


{*************************************************************}
procedure brakecontrol;



    begin
     clrscr;

     writeln('1) Cart - left brake on          2) Cart - left brake off');
     writeln('3) Cart - right brake on         4) Cart - right brake off');
     writeln('5) Platform - left brake on      6) Platform - left brake off');
     writeln('7) Platform - right brake on     8) Platform - right brake off');
     writeln('');
     writeln('A) All brakes on                       0) All brakes off');
     writeln('R) Read EDM');
     writeln('M) Return to motor control ');
writeln('');

     write('Enter choice ');
     repeat;

     if keypressed then read(kbd,choice);
         if choice = #49 then power(0,0);
         if choice = #50 then power(0,5);
         if choice = #51 then power(1,0);
```

```
        if choice = #52 then power(1,5);
        if choice = #53 then power(2,0);
        if choice = #54 then power(2,5);
        if choice = #55 then power(3,0);
        if choice = #56 then power(3,5);
        if choice = #114 then measure;
        if choice = #105 then
           begin;
           repeat;
             writeln(' Enter voltage ');
             readln(volt);
             power(4,volt);
           until volt = -1.23;
          end;
        if (choice = #97) or (choice = #111) then
           begin;
           v := 5;
           g := 0;
           if choice = #97 then v := 0;
             repeat;
                 power(g,v);
                 g := g + 1;
             until g = 4;
           end;
   until choice = #109;
   end;

procedure accel;
          begin;
            if abs(volt) < 5 then
               begin
                     if motor <> 4 then
                        begin
                              power(motor-2,5);
                              power(motor-4,5);
                              power(motor-5,5);    {ensure  brakes are off}
                              power(motor-3,5);
                        end;
                     volt := volt +(dir*accelrate);
                     if abs(volt) > 5 then volt := (5*dir);
                     power(motor,-volt);                    {send power to motor}
                     if (motor <>4) and t then power (motor+1,volt); {both motor
s up/down}
                     going := 1
               end;
          end;

procedure deaccel;
          begin;
                if abs(volt) > 0.05 then
                   begin
                         volt := volt - (dir*deaccelrate);
                         if  ((volt*dir) <= 0 ) then
                                begin;
                                 volt := 0;
                                 going := 0;
                                 if motor <> 4 then
                                   begin
                                    power(motor-5,0);
                                    power(motor-4,0);
                                    if brakes then power(motor-3,0);
                                    if brakes then power(motor-2,0);
                                   end;
                                end;
                        power(motor,-volt);
```

```
                end
            else
                begin
                        going := 0;
                        volt := 0;
                        if motor <> 4 then
                        begin
                          power(motor-5,0);      {ensure brakes are on}
                          if brakes then power(motor-3,0);
                          if brakes then power(motor-2,0);
                          power(motor-4,0);
                        end;
                end

        end;
{*************************************************************************}


procedure autocontrol;          {closed loop control}
  var
    xdes,ydes,xpos,ypos     :real;
    startx,starty           :real;
begin;
clrscr;
gotoxy(1,5);
writeln('Auto Control Mode  (Closed Loop Control)');
gotoxy(1,7);
write('Enter desired x position  ');
readln(xdes);
write('Enter desired y position  ');
readln(ydes);
write('Enter accel rate  ');
readln(accelrate);
write('Enter deaccel rate ');
readln(deaccelrate);

gotoxy(1,15);
writeln('Setting coordinates');
{measure;}
startx := average;
xpos := 0 ;
ypos := 0;

repeat
 {  measure;}
   xpos := (average - startx);
   gotoxy(1,15);
   writeln(' X position = ',xpos);
   motor :=  4;       {drive motor - x- first}
   dir := round((abs(xdes-xpos)/(xdes-xpos+0.00000001)));
   accelrate := abs(xdes-xpos);
   deaccelrate := accelrate;
   for time := 0 to abs(round((xdes-xpos)*20)) do accel;    {accelerate}
   repeat
    deaccel;                        {deaccelerate}
  until going = 0;
until abs(xdes-xpos) < 0.1;
writeln('OK, X position set');
measure;
starty := average;

repeat
{   measure;  }
   ypos := (average-starty);
   gotoxy(1,18);
```

```
    writeln('Y position = ',ypos);
    motor := 5;
    t := true;
    dir := round((abs(ydes-ypos)/(ydes-ypos+0.000001)));
    accelrate := abs(ydes-ypos);
    deaccelrate := accelrate;
    for time := 0 to abs(round((ydes-ypos)*20)) do accel;
    repeat
     deaccel;
    until going = 0;
  until abs(ydes-ypos) < 0.1;
  writeln('Ok, Y position set');
  readln(choice);

  end;




  {*************************************************************************}


  {*************************************************************************}


  procedure encoder(num:integer);
  var
    tba,readinga,readingb:integer;
    hib,lob,hia,loa:real;

    answer,inchperunit:real;
  begin
    tba:=ba+num*4;
    port[tba+1]:=128+32+16;
    delay(5);
    port [tba +1]:=128+32;                          {the data comes in as}
    delay(5);                                        {msb.............lsb}
    readinga:=port[tba +3];                          {hib...lob...loa...hia}
    hia :=round((readinga and $f0)/16);
    loa := readinga and $0f;
    readingb:=port[tba +2];
    hib:=readingb and $f0;
    lob:=readingb and $0f;
    answer:= (((hib + lob) * 255) + loa*16 + hia);
    if num = horizontal then inchperunit:=xinch else inchperunit:=yinch;
    distance:=inchperunit *(answer);
  end;

  {*************************************************}
  procedure motorcontrol;

  var
    key             :char;
  {  choice          :char;}

  begin
    going := 0;
    clrscr;
  repeat
     gotoxy(1,5);
     writeln(' CONTROL CENTER ');
     writeln(' ---------------')  ;
     writeln(' u-up  d-down  l-left  r-right');
```

```
   writeln(' t-change from independent control to ');
   writeln(' s-stop everything          q-brake control');
   writeln(' e-read EDM                 a-Auto Control Mode ' );
    encoder(horizontal);
    writeln('Horizontal encoder ',distance:5:5);
    encoder(leftup);
    writeln('Left vertical encoder ',distance:5:5);
    encoder(rightup);
    writeln('Right vertical encoder ',distance:5:5);
    writeln('Voltage    ',volt:5:4)   ;
    writeln('motor      ',motor,'                ');
    if not t then writeln('Mode : control platform LEFT MOTOR ONLY   ')
                                                                    ');
             else writeln('Mode : platform motors in SYNC                ');
    writeln('    Accel rate = ',accelrate:5:3,'                          ');
    writeln(' Deaccel rate = ',deaccelrate:5:3,'                         ');
    writeln('');
gotoxy(1,1);
    if going = 0 then  writeln('STOPPED and BRAKES APPLIED');

    read(kbd,choice);
gotoxy(1,1);

  case choice of
      #97:begin;                                          ');          {a}
          writeln('AUTO CONTROL
          auto := true;
          autocontrol;
       end;                                                ');          {l}
      #108:writeln('moving left.                           ');          {r}
      #114:writeln('moving right.                          ');          {u}
      #117:writeln('platform up.                           ');          {d}
      #100:writeln('platform down.                         ');          {s}
      #115:writeln('EMERGECY STOP.
      #116: t := not t;
      #112:begin
        gotoxy(1,20);
            write('Enter new deacceleration rate ');
            readln(deaccelrate);
            write('Enter new acceleration rate ');
            readln(accelrate);
            gotoxy(1,20);                                                ');
            writeln('                                                    ');
            writeln('
           end;
     #101:begin
           gotoxy(1,18);
           writeln('READING EDM');
           measure;
           gotoxy(1,18);                          ');
           writeln('
          end;
    end;

    if (choice <> #115) and (going =1) and (choice <> key) then deaccel     {de
celerate}
      else
   case choice of
       #108: begin;
                motor := 4;
                going := 1;                                {accel left}
                dir := 1 ;
                key := choice;
                accel;
               end;
```

```
      #114: begin;
               motor := 4;                          {accel right}
               dir := -1;
               going := 1;
               key := choice;
               accel;
            end;
      #117: begin;                                  {move up}
               motor := 5;
               dir := -1;
               going:= 1;
               key := choice;
               accel;
            end;
      #100: begin;                                  {move down}
               motor := 5;
               dir := 1;
               going:= 1;
               key := choice;
               accel;
            end;
      #115: begin;
               for i:= 0 to 6 do
                  begin;
                  power(6-i,0); {stop everything...motors first}
                  end;
               volt := 0;
            end;
    end;
   until choice = #113;          {quit}

end;


{*************************************************************************}

procedure setencoders;
var
p,num2,  num,tba :integer;
begin

 for num:=0 to 3 do begin
    tba:=ba +num*4;
    port[tba] := $92;  {control word a-in, b-in, c-out}
    delay(50);
    port[tba+1] := 128+32;
    delay(50);              {delay of 50 ms}
    port[tba +1]:=0;
    delay(50);
    port[tba +1] := 128+32;
    delay(50);
 end;
end;

{*************************************************************************}

procedure zeroset;
  begin
    clrscr;
{   measure;}
    startdistance:=average;
    setencoders;
    encoder(horizontal);
    xzero:=distance;
    encoder(leftuo);
```

```
    ylzero:=distance;
    encoder(rightup);
    yrzero:=distance;
    writeln(xzero);
    writeln(ylzero);
    writeln(yrzero);

  end;

{*************************************************************************}

procedure propcontrol;
var
  x,y :real;
  volt,xerror,ylerror,oldvoltage,cx,cy,bx,by,kx,ky,yrerror,oldxerror:real;
  oldylerror,xmaxerror,ymaxerror:real;
  voltleft, voltright,v1,v2:real;
  oldyrerror,oldylvoltage,oldyrvoltage, xdistance, ydistance:real;
diff1,diff2,  differror,olddifferror,gain2 :real;
  sign : integer;
  rel : string[1];
begin
  clrscr;
  writeln('P-D Control Mode  (closed loop control) ');
  gotoxy(1,5);
  write('Would you like absolute or relative coordinates (a/r) ');
  readln(rel);
 if rel = 'a' then
    begin
      ylzero := 0;
      yrzero := 0;
      xzero  := 0;
    end
   else
     begin;
       encoder(horizontal);
       xzero := distance;
       encoder(leftup);
       ylzero := distance;
       encoder(rightup);
       yrzero := distance;
     end;

   gotoxy(1,7);
   write('Would you like to change gain parameters and accuracy (y/n) ');
   readln(choice);
     if choice = 'y' then
      begin

        write('Enter k for x motion ');
        readln(kx);
        write('Enter b for x motion ');
        readln(bx);
        write('Enter c for x motion ');
        readln(cx);
        write('Enter maximum final error for x ');
        readln(xmaxerror);
        bx := 0.2;
        cx :=0.1;
        kx := 2      ;
        xmaxerror := 0.01;

        write('Enter k for y motion ');
        readln(ky);
        write('Enter b for y motion ');
        readln(by);
```

```
      write('Enter c for y motion ');
      readln(cy);
      write('Enter maximum final error for y ');
      readln(ymaxerror);
      write('Enter gain 2 ');
      readln(gain2);
    end

    else
     begin
       bx := 0.2;
       cx :=0.1;
       kx := 2       ;
       xmaxerror := 0.01;
       by := 0.1;
       cy := 0.02;
       gain2 := 1.2;
       ky := 0.6;
       ymaxerror := 0.05;
      end;
   gotoxy(1,18);
  oldvoltage:=0;
  oldxerror := 0;
  oldyrerror :=0;
  oldylerror :=0;
  oldylvoltage :=0;
  oldyrvoltage :=0;

  write('Enter desired x position ');
  readln(x);
  write('Enter desired y position ');
  readln(y);

  clrscr;
  repeat
    encoder(horizontal);
    xerror := x - (distance -xzero) ;
    volt := -cx*oldvoltage + kx*(bx+1)*xerror + kx*(cx-bx)*oldxerror;
    oldvoltage:= volt;
    oldxerror := xerror;
if volt <> 0 then
  begin
    volt := volt + 0.95*(abs(volt)/volt);
    if abs(volt) > 5 then volt:=5*(abs(volt)/volt);
  end;                        {so as to avoid  div by zero error}
    power(4, -volt);

 until abs(xerror) < xmaxerror;
{ measure;}
{ xdistance:=average - startdistance;}
 power(4,0);
 writeln(' X position is ',distance:5:5,'  ');
 writeln(' Error is ',xerror:5:5,'  ');
 writeln(' EDM position reading (meters) :',xdistance);
                             {turn off brakes for y motion}

  power(0,5);
  power(1,5);
  power(2,5);
  power(3,5);
  oldvoltage := 0;
  oldylerror := 0;
  olddifferror := 0;
  differror := 0;
  repeat
    encoder(leftup);
    ylerror := y -(distance -ylzero);
```

```
    diff1 := distance;
    encoder(rightup);
    diff2:= distance;
    differror := -diff2+diff1;
    yrerror := y -(distance -yrzero);
    voltleft := -cy*oldylvoltage + ky*(cy-by)*oldylerror + (by+1)*ylerror;
    oldylvoltage:= voltleft;
    oldylerror := ylerror;
{    differror := -ylerror+yrerror;}
{    voltright := -cy*oldyrvoltage + ky*(by+1)*differror + ky*(cy-by)*olddifferr
or;}
    if voltleft < 0 then voltleft := voltleft -0.9;
    voltleft := voltleft + 0.1;
    if abs(voltleft) > 2 then voltleft:= 2*(abs(voltleft)/voltleft);

    voltright :=  gain2 * differror + voltleft;

    oldyrvoltage:= voltright;
    olddifferror := differror;
{    voltright := voltright + voltleft;}
{    if voltright > 0 then voltright := voltright + 0.4;}
    if abs(voltright) > 5 then voltright:=5*(abs(voltright)/voltright);
    power (5,-voltleft);
    power (6,voltright);
  until (abs(ylerror) <ymaxerror) and (abs(yrerror) < ymaxerror ) or keypressed;
  power(0,0);
  power(1,0);
  if brakes then   power(2,0);
  if brakes then   power(3,0);
  power(4,0);
 { measure;}
  ydistance:=average -xdistance;

gotoxy(1,18);

encoder(leftup);
writeln('Done');
writeln(' Y left distance is ',distance:5:5,' Error is ',ylerror:5:5 );
encoder(rightup);
writeln(' Y right distance is ',distance:5:5,' Error is  ',yrerror:5:5);
writeln(' EDM y position reading (meters) :',xdistance);
gotoxy(1,21);
write('Hit RETURN to continue');

read(x);

end;

 {MAIN PROGRAM}


begin;
  brakes := true;
  t := true;
  ba:=784;              {base address for baseboard}
  horizontal:=0;        {setting port values for encoders}
  leftup:=3;
  rightup:=1;
  accelrate := 1;
  deaccelrate := 1;
  c := 0;
  clrscr;                                    {tests to see if port nee    as
    reset;                                                                  48
tting}
 port[$224]:=$f;                            {set D/A to program 2}
```

```
repeat                                          {turn all brakes on};
 power(c,0);
   c := c + 1
until c = 7 ;

voltage:=0;
volt:=0;
xinch:=0.01887810/16;
yinch:=0.0143283582/16;                 {setting inches per unit of encoder movement}
setencoders;
repeat;
      clrscr;
      gotoxy(1,5);
      writeln('SCAFBOT MAIN MENU');
      writeln;
      writeln(' A) Open loop keyboard control.');
      writeln(' B) Closed loop control with EDM.');
      writeln(' C) Proportional control with encoders.');
      writeln(' D) Independant brake control.');
      writeln(' E) Zero set coordinates of platform.');
      writeln(' F) Turn platform brakes off.');
      writeln(' G) Turn platform brakes on.');
      writeln(' X) exit.');
      read(kbd, choice);
      case choice of
        #97: motorcontrol;
        #98: autocontrol;
        #99: propcontrol;
        #100:brakecontrol;
        #101:zeroset;
        #102:begin
               brakes := false;
               power(2,5);
               power(3,5);
             end;
        #103:begin
               brakes := true;
               power(2,0);
               power(3,0);
             end;

      end;
until choice = #120 {x- exit}
end.

C:\BRAD\ERIC>
```