



University of Glasgow
DEPARTMENT OF

AEROSPACE
ENGINEERING

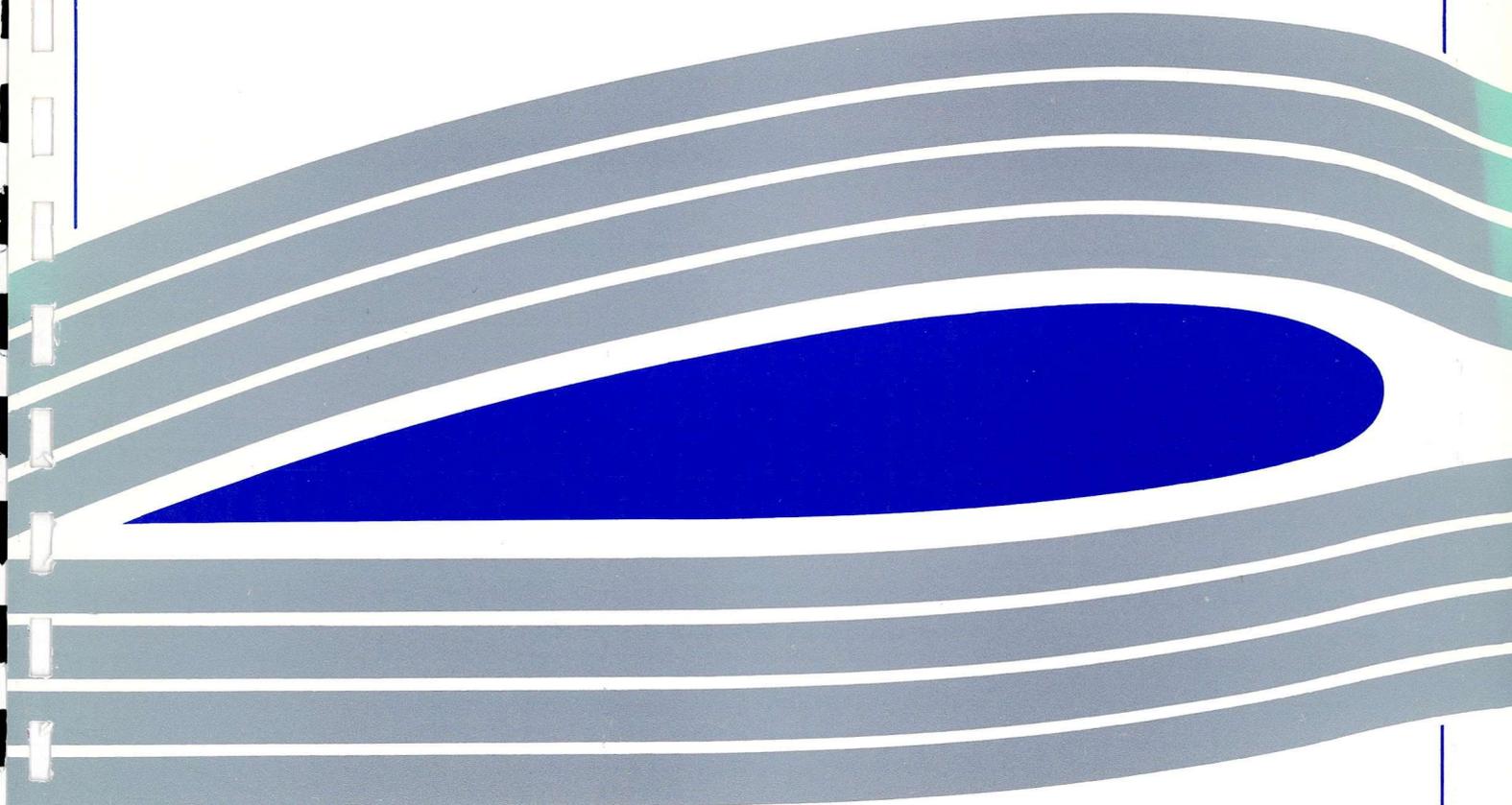


Engineering
PERIODICALS

5000

**A STEPPER MOTOR BASED
SYSTEM FOR RAPID PITCHING
OF WIND TUNNEL MODELS**

G.U. Aero Report 9402



Engineering
PERIODICALS

U5000

**A STEPPER MOTOR BASED
SYSTEM FOR RAPID PITCHING
OF WIND TUNNEL MODELS**

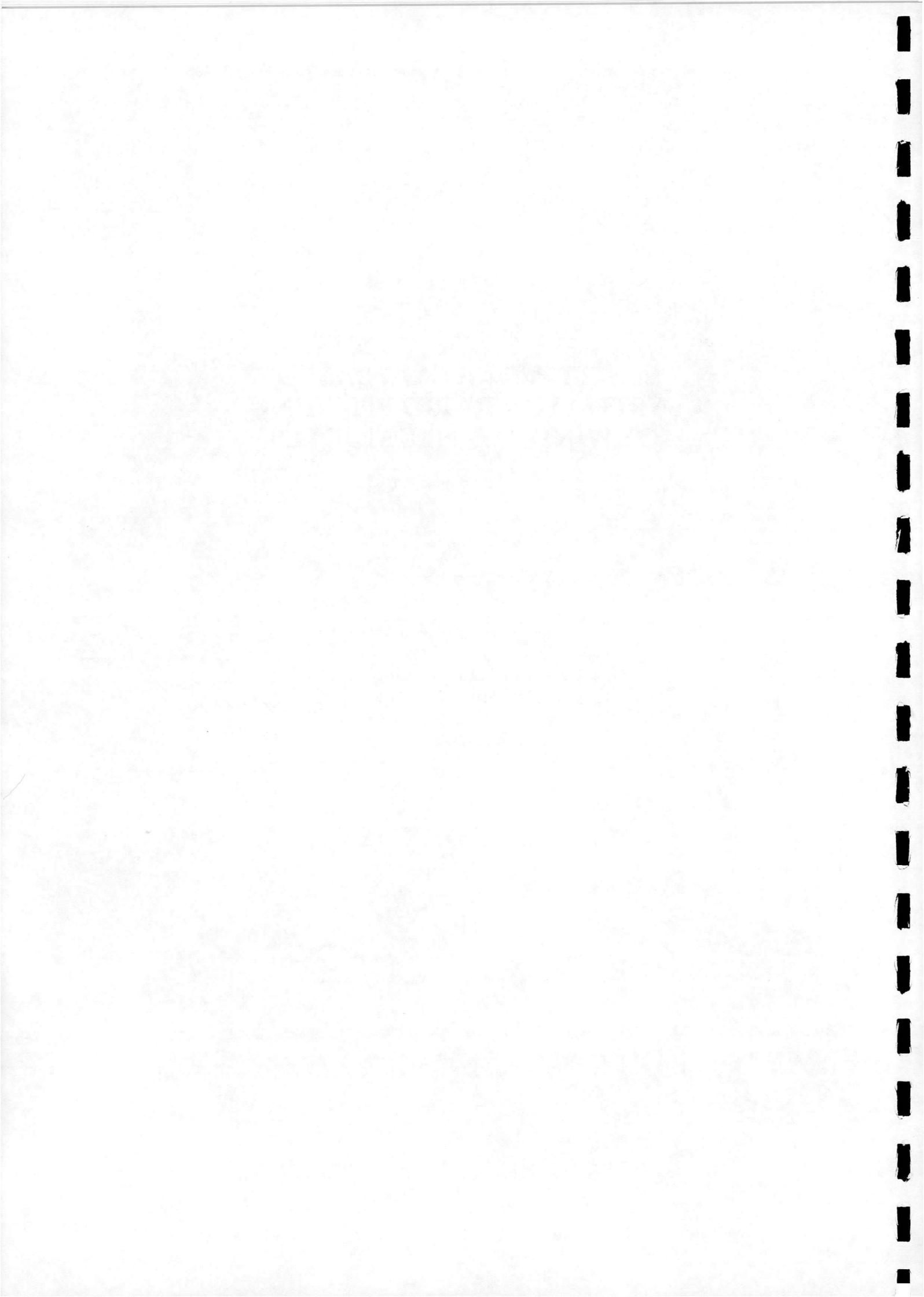
G.U. Aero Report 9402

by

R.B. Green

**Department of Aerospace Engineering
University of Glasgow
Glasgow
Scotland**

18th March 1994



A STEPPER MOTOR BASED SYSTEM FOR RAPID PITCHING OF WIND TUNNEL MODELS

by

R.B. Green

ABSTRACT

This report describes the apparatus and computer codes for the unsteady aerodynamics flow visualisation rig in the Department of Aerospace Engineering's smoke flow visualisation wind tunnel. The purpose of the report is to archive the work done and provide information to those who may need to set up the apparatus or modify any of the running parameters. Loose descriptions of the structure of the LabVIEW computer code are also given.

CONTENTS

INTRODUCTION	2
ELEMENTS OF APPARATUS	2
1. Stepper motor	2
2. Gear box and ball-screw/ flange-nut	3
3. Computer controller	3
4. Associated electronics	4
5. Details of control code	5
SETTING UP THE HARDWARE	14
TROUBLE SHOOTING	14
COMMENTS	15
REFERENCES	15
FIGURES	



INTRODUCTION

A low speed smoke flow visualisation wind tunnel has recently been built in the Department of Aerospace Engineering at the University of Glasgow. This is an ideal facility for investigation of the fundamental aspects of unsteady aerodynamics, e.g. dynamic stall. The maximum speed of the tunnel is some 1ms^{-1} and the tunnel working section is $1\text{m}\times 1\text{m}$, so the maximum practical Reynolds number is some 20 000. Therefore any results are likely to be qualitative in nature, although it is stressed that the value of flow visualisation data cannot be under-estimated.

The unsteady effect most commonly investigated is that of rapid pitching. Two-dimensional aerofoil models are pitched around a convenient location, e.g. the quarter chord point, at a high rate. Such motions allow dynamic stall or unsteady re-attachment to be investigated. Other testing configurations are plunging and three-dimensional motion. Constant rate plunging motions differ from pitching motions in that the incidence undergoes a step change at the start of the plunge, and remains constant thereafter. Three-dimensional models by a sting moving the trailing edge about a fulcrum position via a dog-leg.

This report describes how the above motions are generated. The basic system consists of a stepper motor, which moves the model by reduction gears or a ball screw/ flange nut. The whole system is under computer control.

ELEMENTS OF APPARATUS

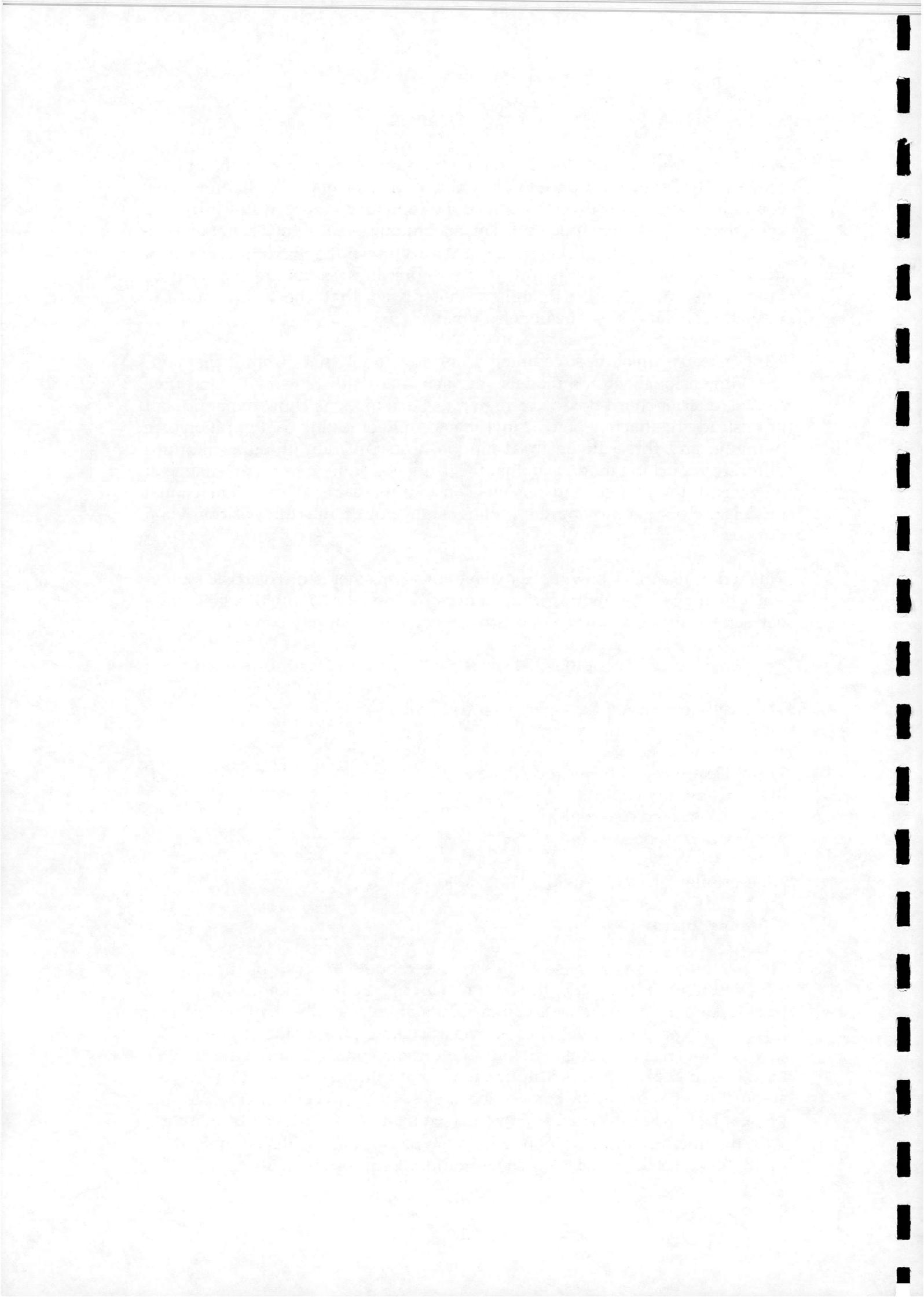
The basic elements of the apparatus are as follows:

- 1) Stepper motor.
- 2) Gear box and ball-screw/flange-nut.
- 3) Computer controller.
- 4) Associated electronics.
- 5) Control code.

A description of each individual item follows.

1. Stepper motor

Stepper motors were chosen to drive the models because of the ease of programming. In principle, the output shaft of a stepper motor rotates by a precise amount in response to an input square wave. The shaft positioning error is non-accumulative. Therefore very accurate positioning is possible in theory. However, to obtain smooth motion, the control box for the stepper motor (which processes the input square waves) must be able to make the stepper motor move in very small step angles. The stepper motor used for the present task is an RS type 34. When driven by a 6A control board operating with the micro-stepping option card, the type 34 motor will step by 0.18° at up to 8KHz, i.e. up to 1440°s^{-1} . The maximum torque is about 3Nm.



The control box for the stepper motor runs in four modes: the first accepts step and direction inputs from an external source (e.g. a computer). The second mode performs ramp-and-hold motions. The third mode performs continuous sawtooth motions and the fourth mode rotates the motor shaft at a constant speed. The speed for the latter three modes is controlled by a frequency generator built into the control box, and the direction control for modes two and three is provided by two reflective optical switches which plug into the back of the control box. These switches must be strategically positioned to limit the range of motion of the model.

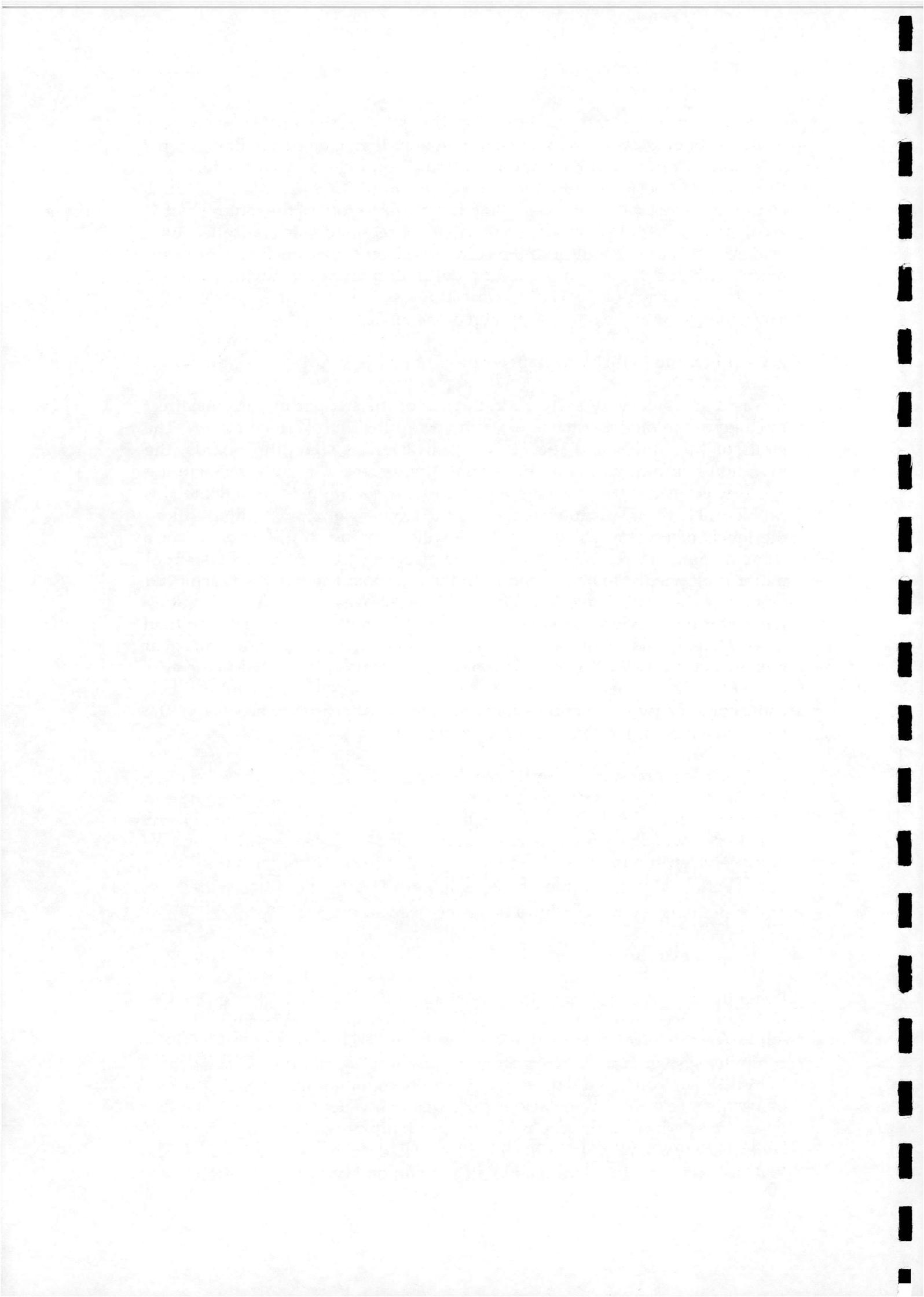
2. Gear box and ball-screw/ flange-nut.

It was discovered very early on in the use of the stepper motor that direct pitching of the model at the quarter chord resulted in severe vibrations. The walls of the tunnel and the model itself acted as sounding boards, the vibration being provided by the stepper motor, even though the stepping angle was only 0.18° . Various methods were attempted to alleviate this problem. Firstly it was found that a reduction gear box drastically reduced the levels of transmitted vibration. However, owing to the tendency of a stepper motor to resonate at particular stepping frequencies, the stepper motor itself needed to be isolated from the gear box. The latter was achieved by using V-belts and pulleys. Additional isolation was achieved by mounting the motor on wooden supports rather than in a metal framework. The final drive system consists of a D50 type, 5:1 reduction, anti-backlash, spur gear box provided by HPC Gears Ltd. The stepper motor is connected to the gear box via 2" dia pulleys on the motor and gear box input shafts and a V-belt connecting the two. The pulleys allow for additional gearing if necessary. The whole assembly is mounted on a plywood box.

Actuation for three-dimensional models is provided by a ball-screw/ flange-nut. Basically, this system converts the rotary motion of the stepper motor to linear motion of the flange nut. The pitch of the ball screw is 40mm per rev. Experience with two-dimensional pitching suggested using pulleys and V-belts between the motor and ball-screw. In addition different pulley sizes allow better rates of motion of the flange nut (a 1:1 pulley size with 0.18° motor stepping is equivalent to a flange nut step accuracy of $20\mu\text{m}$).

3. Computer controller

Being the most expensive element of the apparatus, a cost effective, easy to use system needed to be used for the control of the actuation. Stepper motors eliminate the need for feedback control, so what was required was a computer system capable of supplying square waves at the correct frequency. LabVIEW (an icon based data acquisition and programming language) was to be used, since a departmental standard was to be adopted. At the time this necessitated the use of Apple Macintosh equipment. The final choice of equipment was an Apple LCII with a LAB-LC data acquisition card [1]. The LCII was quickly updated to an LCIII with an on-board math co-processor,



however, since the LCII was too slow. The data acquisition card contained eight analogue input channels, two digital to analogue output channels, three digital input/ output ports of eight lines each, and three on board counters.

A characteristic of stepper motors is that the torque drops as the speed increases, and it was anticipated that, during high speed ramps, the control code would have to be able to accelerate the model. Therefore the motor driving frequency would have to vary as a function of time/ incidence.

The two on-board digital to analogue converters (DACs) on the LAB-LC board can be configured as wave form generators. For the present application one could be configured to generate the square wave driving signal, and the other could provide the direction information, and the two waves would be generated simultaneously. An uncertainty relating to the rate of motion of the model may still exist: the waveforms output by the DACs can be output at an accurate rate, and it is known that, provided the motor has not stalled, one pulse corresponds to 0.18° shaft rotation. Thus, full use of the stepper motor is made (i.e. feedback is not really necessary!).

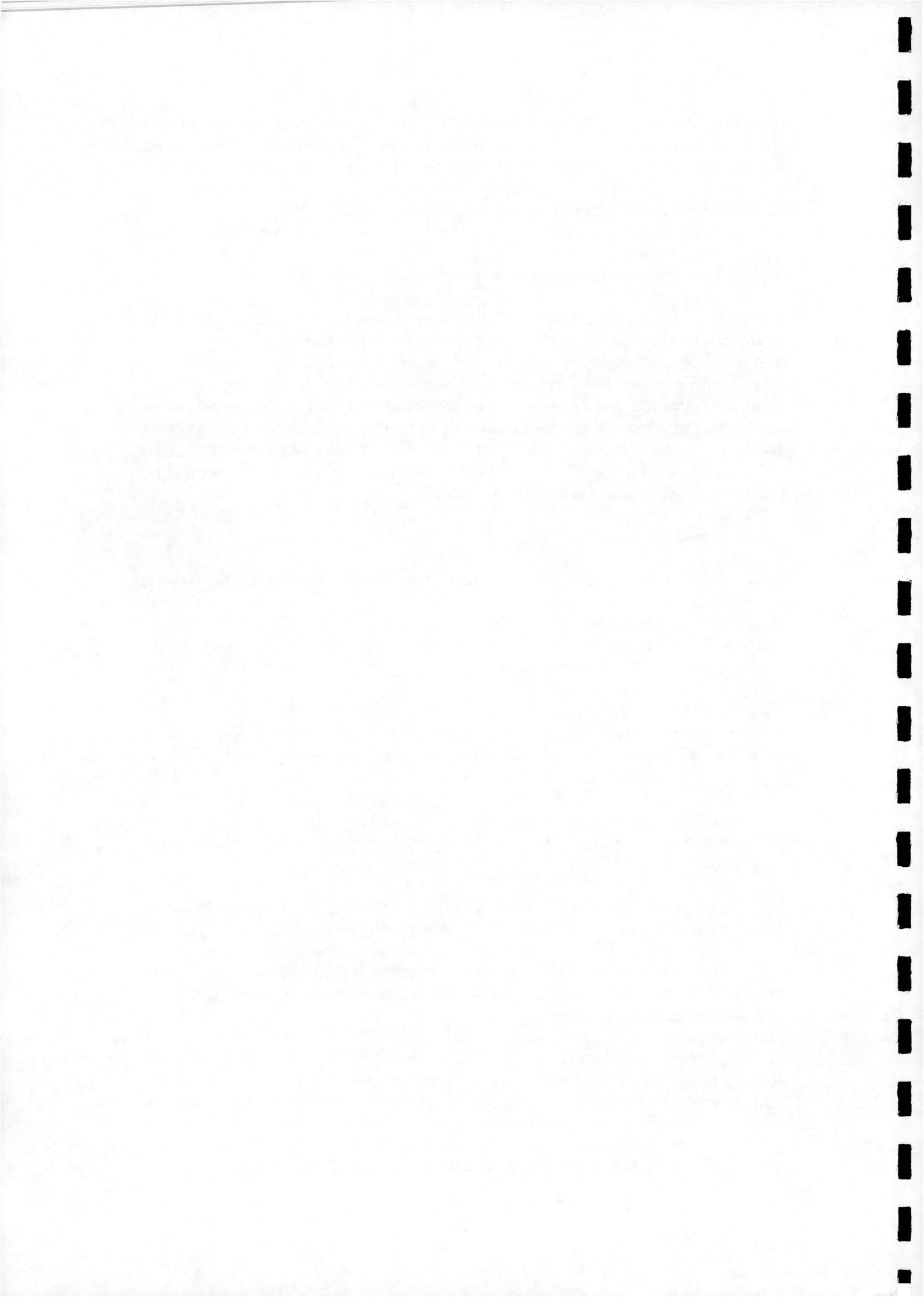
4. Associated electronics

Other items of hardware are an optical encoder (for feedback and datum determination), an LED display for showing the model incidence, and a quad input NAND gate for processing the pulse signals to the stepper motor and to assist in locating the datum. The ball screw needs two datum detectors at each end of the ball screw to prevent the flange nut from running off.

For correct operation of the LED display, the optical encoder must have two output channels 90° out of phase (note that an optical encoder is also mounted on the ball screw). The number of pulses per revolution is not especially important, so long as the user is aware of what the value is. In addition to the above, the encoder for the two-dimensional pitching rig must also supply one synchronisation pulse per revolution. This allows the datum position of the model to be located, so that the model may be automatically positioned at zero degrees.

The arrangement of the micro-switches for the ball-screw is shown in figure 1. These micro-switches act as datum signals and safety signals.

The above datum signals must be accounted for in the design of the processor. During an actual test the datum signal must be ignored in the 2D pitching case, although the hi-lo datum signal must be detected when the datum is being looked for. However, the digital input/ output ports on the LAB-LC board could be used to disable/ enable the datum signal by means of appropriate logic. The datum signal can be fed directly to the computer so that its state may be determined, although the datum signal could also be used to disable the pulses sent to the stepper motor. A simple NAND gate (type 74HC00N) was used and the wiring diagram and truth table are shown in figure 2. Note that the STEP signal (the signal sent to the stepper motor)



does not respond to a change in the CLK (the pulses from the computer) input only when the DATUM ENABLE is high and the DATUM signal is low (i.e. the datum has been found). Note that the CLK input has to be protected by a substantial resistor, since the input is sourced from the analogue output of the LAB-LC.

5. Details of control code

The control code for the two-dimensional pitching apparatus is described. Additional requirements for the ball screw are given later.

5.1. Input parameters

The following input parameters were required for a complete description of a ramp/ oscillatory motion:

Datum angle (so that true zero degrees can be found). The datum position is provided by the optical encoder synchronisation pulse (i.e. one pulse (hi-lo-hi) per rev)

Mean incidence (AM) and amplitude (AR). In addition, for ramps a non-linear (acceleration) region would be required (AH).

Maximum speed of the motor (PR).

Motion type (i.e. ramp-up, ramp-down, sinusoidal, triangle)

The following are input parameters of less significance to above:

Model size

Flow speed

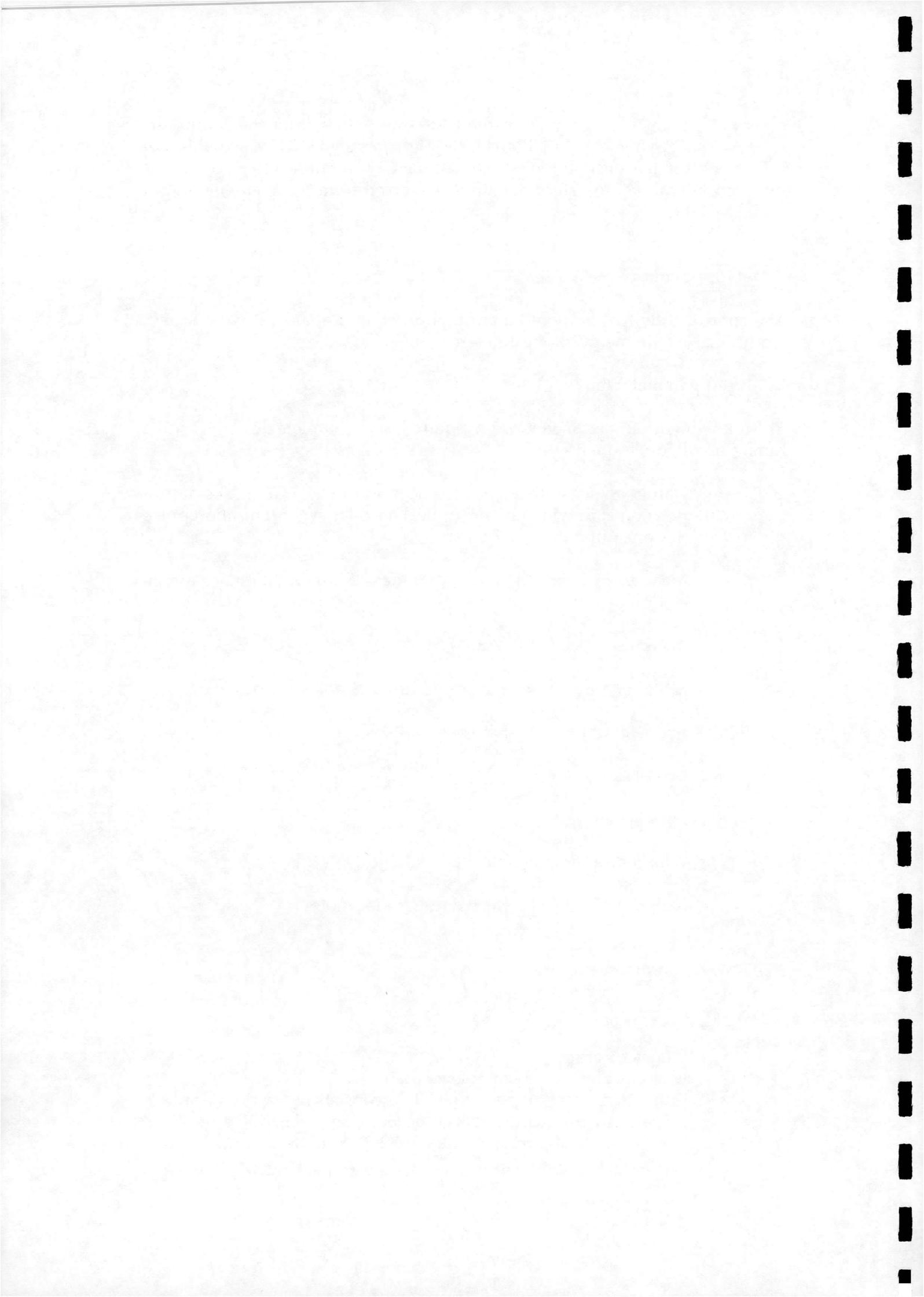
Wait time after ramps (to allow flow to settle)

Number of ramps between datum re-location

5.2. Flow diagram of code

The essential items within the code are as follows:

- i) Locate datum
- ii) Set incidence, for example for static tests.
- iii) Determine driving function. This consists of a series of pulses, a specified time apart. The time between each pulse depends upon the rate of motion of the model. Before the driving function can be calculated the minimum time interval (Δt_{\min}) between



each pulse must be determined (which depends upon the maximum pitch rate). This value then dictates the total number of pulses and the rate at which the LAB-LC board outputs them to the stepper motor controller. The pulse data is stored in an array. In addition to the pulses, the direction information is also calculated. This is only of importance to oscillatory and triangle motions.

iv) Pitch model.

5.3. Details of code

The code was written in LabVIEW. LabVIEW consists of virtual instruments (VIs) which are equivalent to subroutines in any sequential code. The code contains four sub VIs and one calling VI. A description of each follows.

5.3.1. Timing sub VI.

The rate at which the elements of the array are output by the LAB-LC needs to be determined. To make the stepper motor step, one hi-lo pulse needs to be performed. With the 5:1 reduction gear box, 0.036 degrees of model motion therefore occupies two array points. Therefore the rate at which the pulses must be supplied is

$$FB=PR/0.018 \quad \text{where FB is the base frequency.}$$

The time base was chosen as 10 μ s between pulses, so the actual time interval is

$$UA=1.0/(FB*1.0e-05)$$

UA must be supplied in integer form, so the actual update interval is

$$UB=int(UA)$$

in which case the actual maximum pitch rate is

$$AP=PR*UA/UB$$

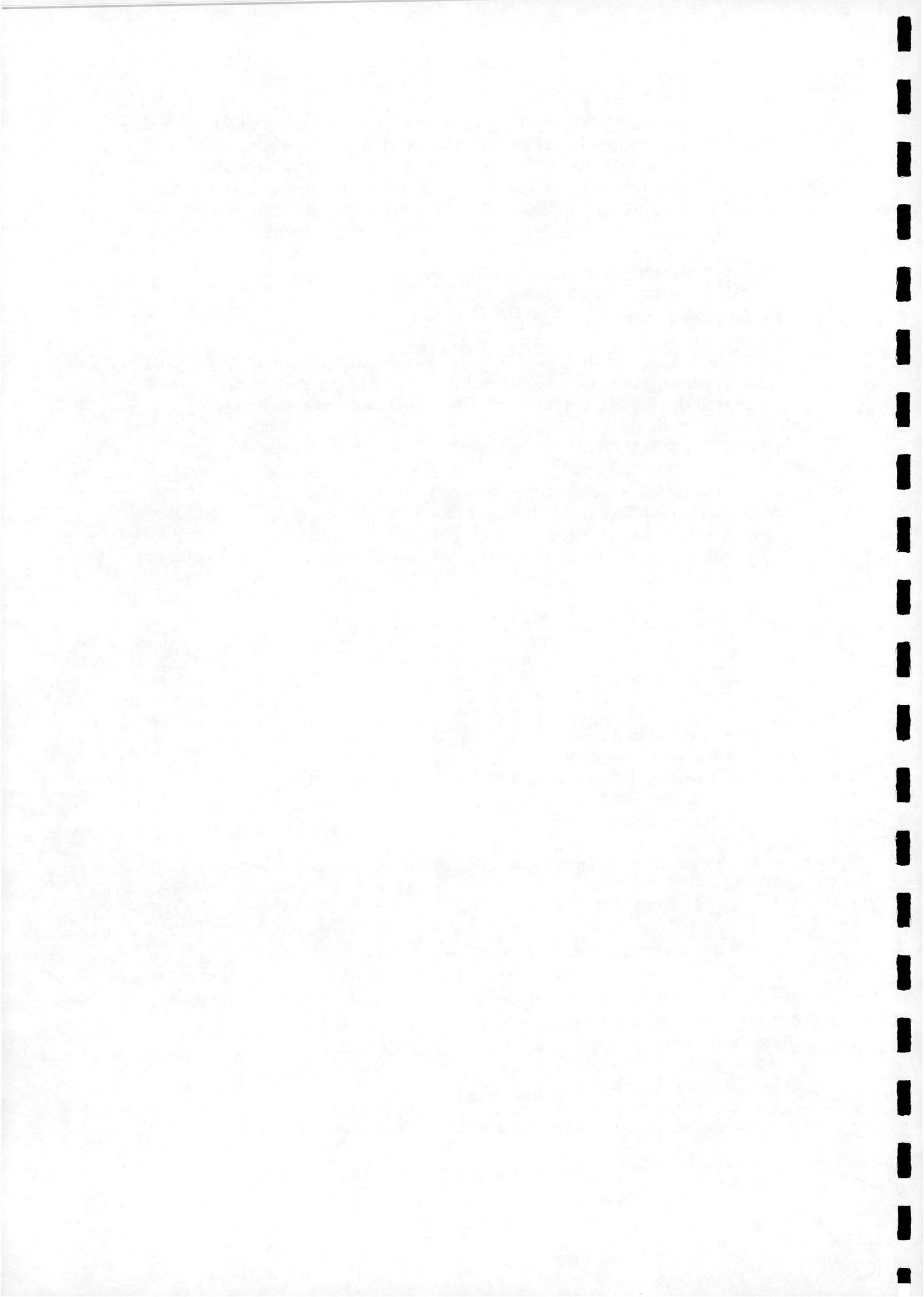
Finally the total elapsed time of the test is

$$TE=((\pi-2.0)*AH + AR)/PR$$

and therefore the number of elements in the drive array is

$$NE=int(FB*TE)$$

The above code is written into the LabVIEW VI set_rate_parameters.



5.3.2. Drive array sub VI

Details of the above are fed into the drive array VI, form_array_WFG_DACDIRN. The inputs to it are as above. Note that for a sinusoidal test, the amount of non-linearity (AH) is half the range of motion (AR). The details of the drive array must account for the period of acceleration and deceleration of the motor.

The non-linear zones in the ramp test have a sinusoidal shape and the circular frequency of the non-linear part is

$$OM=PR/AH$$

Again the total time for the test is

$$TE=(AH*\pi + AR-2.0*AH)/PR$$

The time at which acceleration ends is

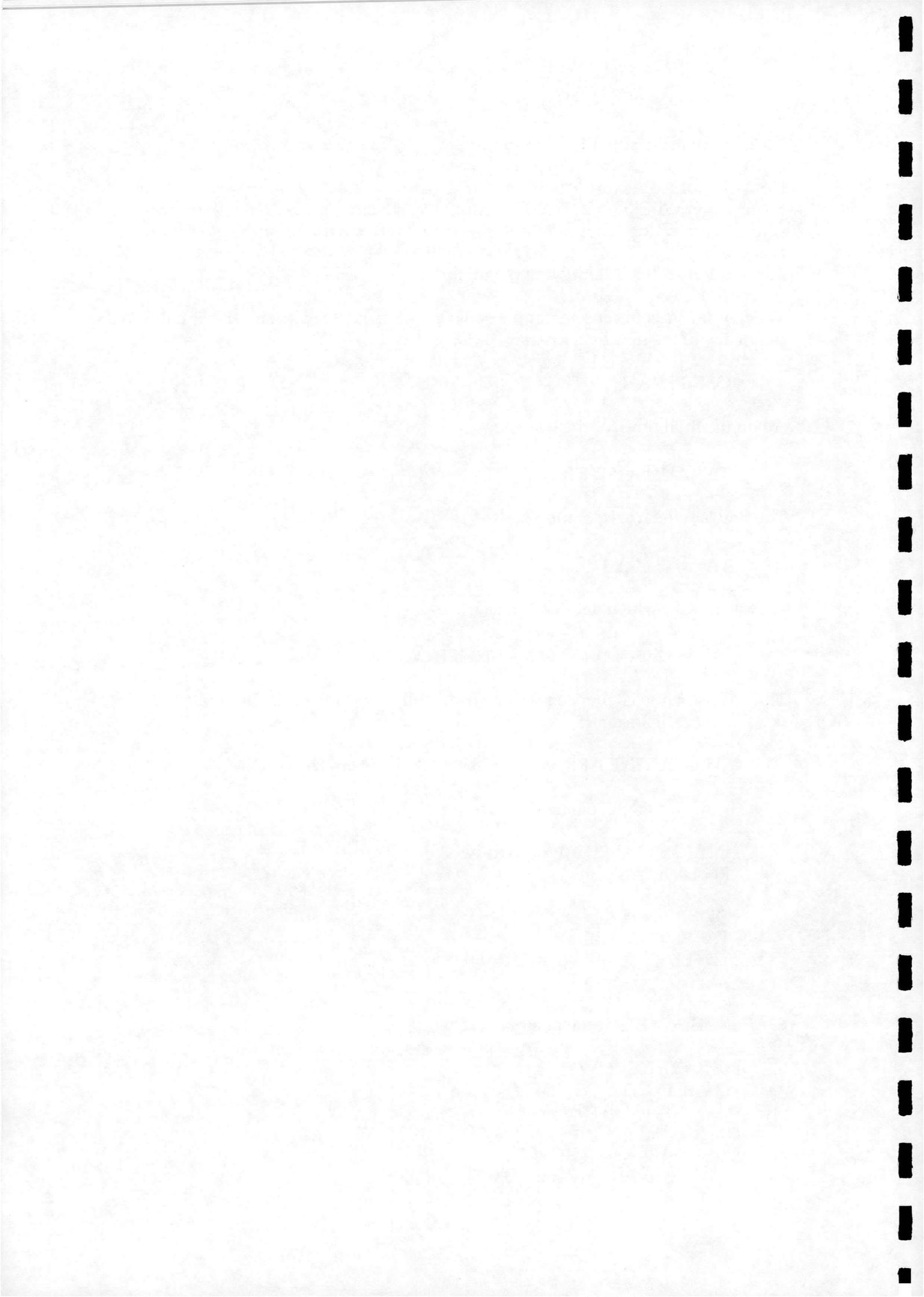
$$TA=\pi/(2.0*OM)$$

and the time at which deceleration starts is

$$TD=\pi/(2.0*OM) + (AR-2.0*AH)/PR$$

The drive array is then constructed as follows (presented in sequential FORTRAN code format):

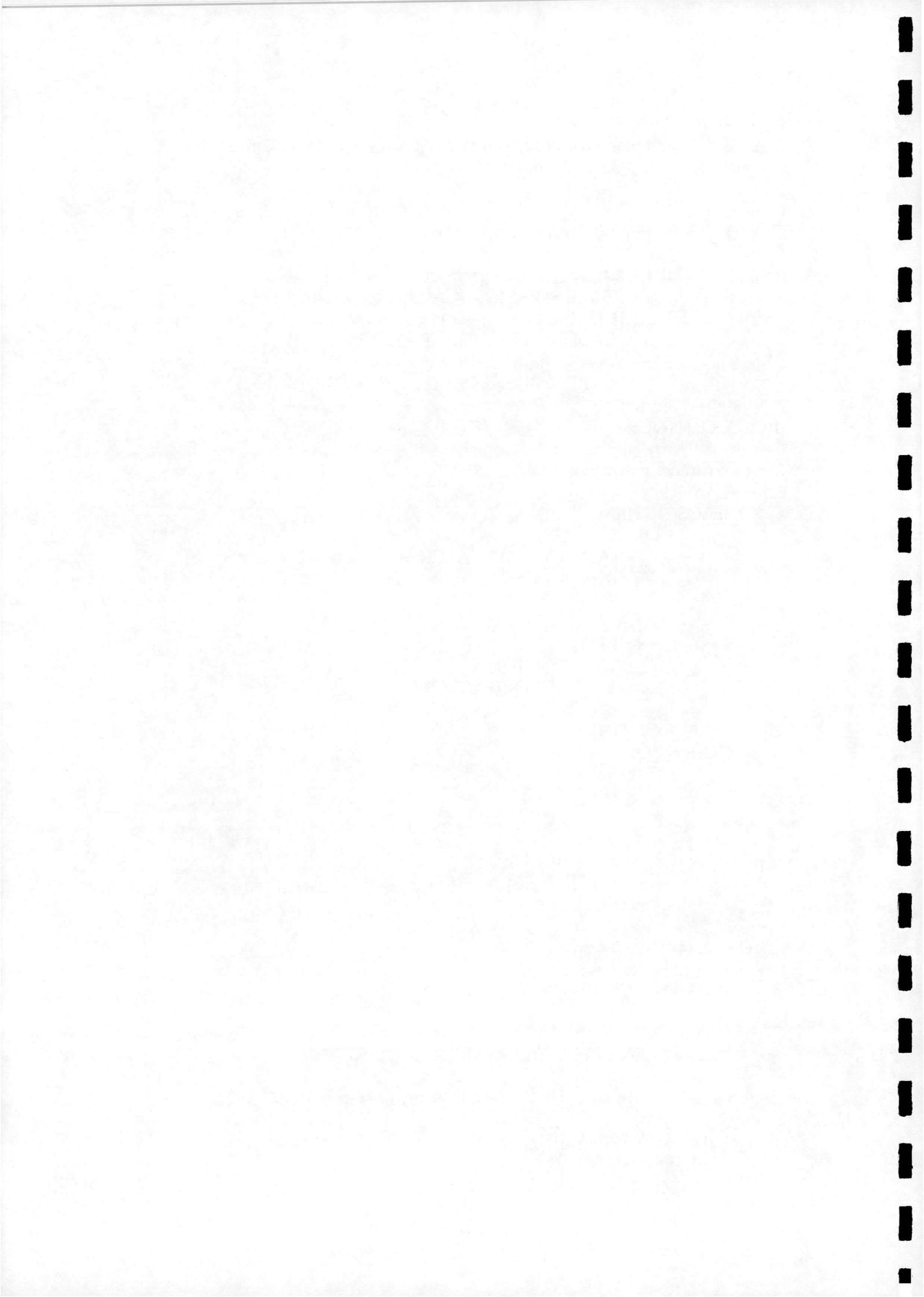
```
      SUBROUTINE ARRAY(AR,AH,PR,FB,NC,IALPHA,IDIRN,OSC, UP)
C-----
C NC is the number of elements in array = NE*2
C-----
      INTEGER IALPHA(NC), IDIRN(NC)
      PI=4.0*ATAN(1.0)
C
      OM=PR/AH
      TA=PI/(2.0*OM)
      TD=PI/(2.0*OM) + (AR-2.0*AH)/PR
      AP=0.0
C-----
C AP is the previous value of alpha
C-----
      IF (OSC) NC=NE/2
      DO 10 I = 1,NC
          T=FLOAT(I)/FB
          AC=AH*(1.0-COS(OM*T))
          AL=AH+PR*(T-TA)
          AF=AR+AH*(SIN(OM*T-TD))-1.0
```



```

C -----
C AC is the incidence during acceleration, AL is the incidence during the
C linear phase, and AF is the incidence during the deceleration phase.
C -----
C -----
C check phase of motion and take appropriate action
C -----
      IF ( T .LE. TA ) THEN /model accelerating/
        CALL STEPCHECK(AP,AC,IALPHA(I))
      ELSE IF ( T .LE. TD ) /linear phase of motion/
        CALL STEPCHECK(AP,AL,IALPHA(I))
      ELSE /deceleration phase of motion/
        CALL STEPCHECK(AP,AF,IALPHA(I))
      ENDIF
10  CONTINUE
C -----
C set up direction array
C -----
      IF (OSC) THEN
        DO 20 I = 1, NE
          IALPHA(I+NE)=IALPHA(I) /motion is symmetrical/
          IF (UP) THEN
            IDIRN(I)=2047
            IDIRN(I+NE)=0
          ELSE
            IDIRN(I)=0
            IDIRN(I+NE)=2047
          ENDIF
20  CONTINUE
      ELSE
        DO 30 I = 1, NE
          IF (UP) THEN
            IDIRN(I)=2047
          ELSE
            IDIRN(I)=0
          ENDIF
30  CONTINUE
C -----
C end of array calculation
C -----
      RETURN
      END
C
      SUBROUTINE STEPCHECK(APR,ALPHA,ISTEPVAL)
C -----
C subroutine for calculating whether to perform a step or not.....
C
      DIFF=ABS(APR-ALPH) /previous alpha - current value/
      STEP=DIFF-0.036

```



```

REM=ALPHA-STEP
  IF ( STEP .GE. 0.0 ) THEN /perform step/
    ISTEPVAL=0
    APR=ALPHA-DIFF /step over 0.036° carried over/
  ELSE /no step/
    ISTEPVAL=2047
  ENDIF
RETURN
END

```

5.3.3. Motor driver sub VI

Once the drive arrays have been determined, then moving the model is a case of writing the arrays to the LAB-LC, which is taken care of by a built in LabVIEW VI. The motor driver VI has the name DRIVER_wvfg_DACDIRN. Inputs to this VI are the update interval, the pulse and direction arrays, the number of samples in each array, the time base and the drive type (i.e. continuous or single execution of the array of pulses).

5.3.4. Setting incidence and finding the datum sub VI

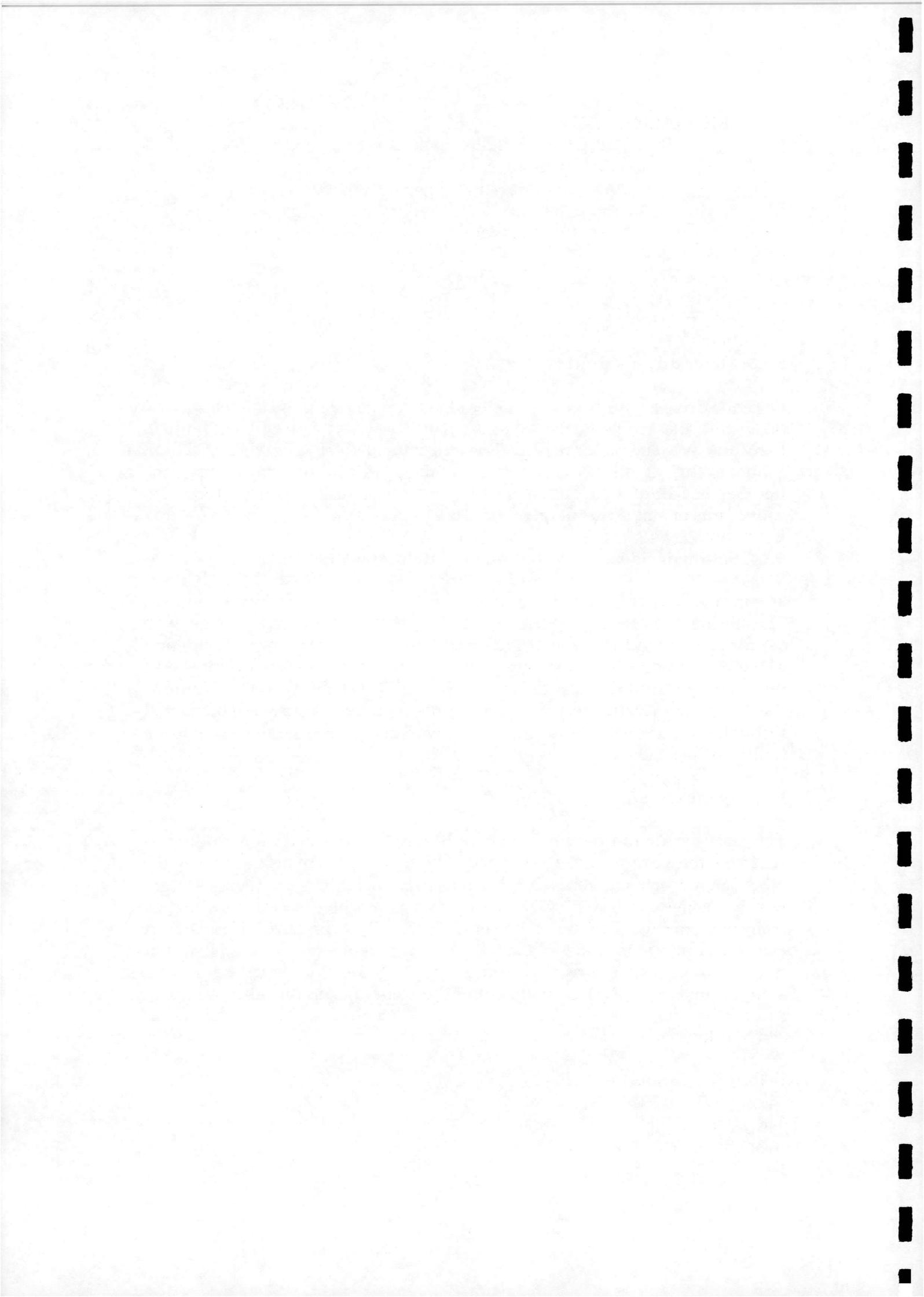
The incidence may be set to any value in the main program control window. This allows static tests to be performed. In addition, start incidences must be set for ramps, and sinusoidal tests may have a non-zero mean incidence. Therefore a VI must exist for setting the incidence to any chosen value. The VI written to perform this task is named SET_ALPH+DATUM_WFG. Inputs to this VI are an update interval for timing, the number of degrees to be moved through, and a logical input indicating whether or not the datum is to be located before the incidence is set.

Finding the datum.

The position of the datum is generally not known. If datum finding is required, the VI form_array_wvfg_DACDIRN is called. An array to move the model in 10° steps at 20°s^{-1} is formed. Digital port 0 line 2 is then set high, which enables the NAND gate to stop stepping when DATUM switches from high to low. The motor driver VI DRIVER_wvfg_DACDIRN is then called in continuous mode. While the model is moving the state of the datum is continuously monitored via port 1 line 0. Ideally the model should be stopped by the NAND gate switching the stepping off, and the computer should then read that the datum has actually been reached when the datum signal is low.

Setting the incidence

The chosen incidence is set through individual high-low-high pulses sent through the analogue output port.



5.3.5. Calling VI

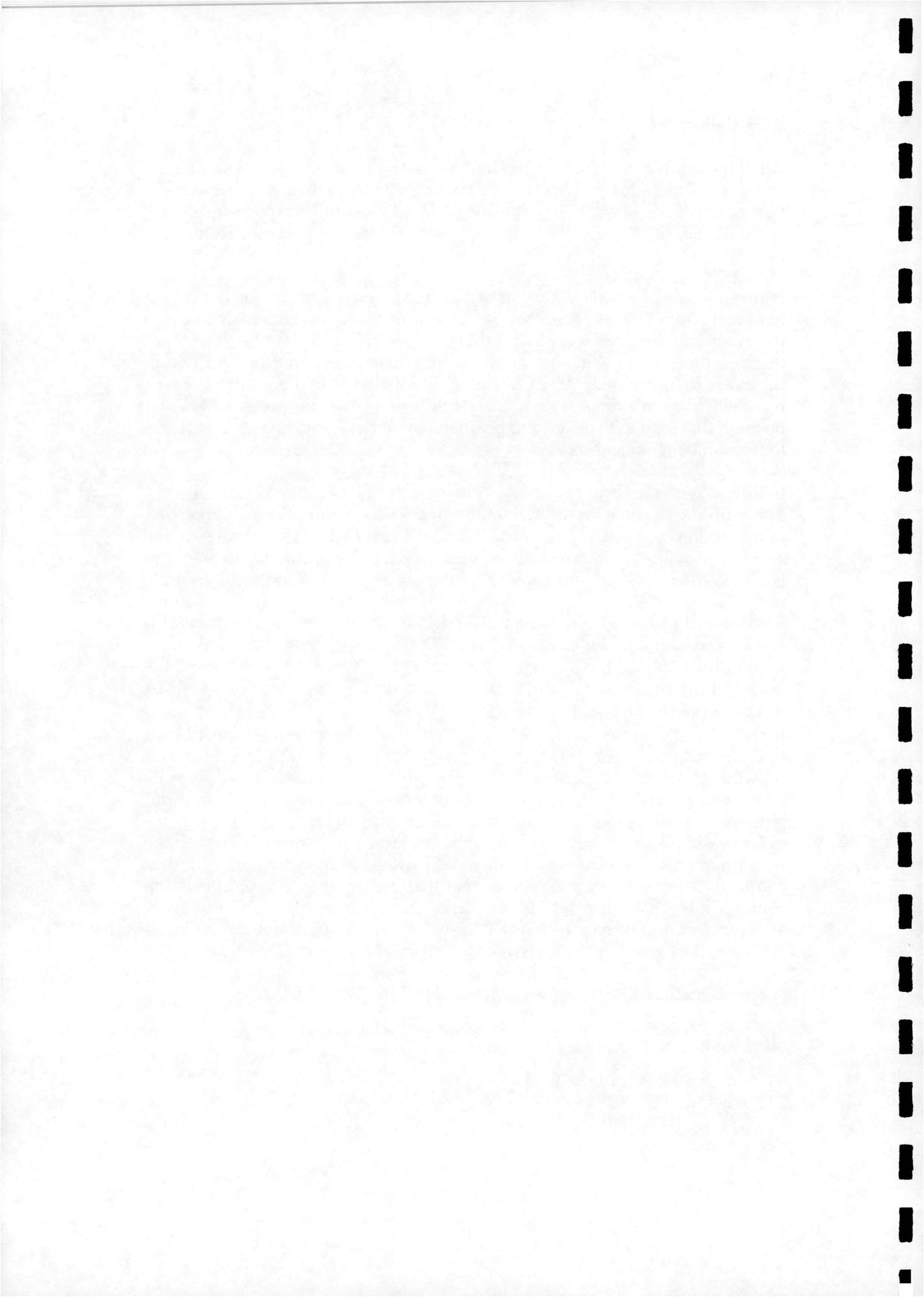
All of the input parameters are set in the calling VI. The method of setting the variables is intended to signify their importance in the running of the code. For example it is vital that the mean incidence and amplitude are set correctly, although the value of the flow speed does not affect the movement of the model.

Figure xx shows a representation of the front panel. The datum is the first item that should be set. After the datum is found, the model moves through the number of degrees specified. If this does not return the model to the zero degrees, then a new value of datum position may be set and the find datum button is then pressed. When the user is satisfied that the model is being returned to zero degrees, then the large button may be pressed. The set position dial may then be used to position the model at any chosen incidence, for example for static tests.

Before a dynamic test is run the settings in the main panel need to be set. Three "motion type"s are available: ramp, sine and triangle. For high speed ramp motions the "ramp non-linearity" must be set to a non zero value, or the motor will stall. The "ramp wait" input specifies the amount of time the flow is allowed to settle before the model moves again for ramp and triangle motions. (either to reset or start the test). "Runs between relocate" is used for triangle and ramp tests, and datum is relocated every n runs. This may also allow the running parameters to be reset. The "speed" setting is the maximum speed of the motor, the "range" is the number of degrees that the model is moved through and the "mean alpha/ start control" is the position at which the model starts the test. Therefore for a 40° to 0° ramp-down test the range is set at 40° and the start is set at 40°. The "ramp-up/ down" button must be set to the correct direction. After the inputs are set correctly the "press to run test" button may be depressed. Once running each function is sitting in its own execution (while) loop. Therefore to stop a test any of the "continue" buttons needs to be depressed. Note that for ramps and triangles the number of loops set has to be executed before the continue status is read. To stop the test altogether, "continue run" must be pressed before any of the other continue buttons is operated. Note also that moving the mouse affects the running of the test, since data is transferred through the bus, which ultimately affects the output of data from the LAB-LC. Finally actual run information is output as the "speed" and the "reduced pitch rate/frequency".

A representation of the calling VI MOVE_MODEL_WVFG follows:

```
SEQUENCE 0:
{
CONFIGURE PORT 0 AS OUTPUT
CONFIGURE PORT 1 AS INPUT
} /END SEQUENCE 0/
SEQUENCE 1:
{
```



```

DO WHILE("PRESS DATUM FOUND" IS FALSE):
{
IF ("FIND DATUM" IS TRUE) THEN
    {
        SET ALPHA(FIND DATUM & MOVE TO "DATUM POSITION)
    }
}
} /END SEQUENCE 1/

SEQUENCE 2:
{
SETINC=0
DO WHILE("PRESS TO RUN TEST" IS FALSE):
{
INC="NUMBER OF DEGREES"-SETINC
IF (INC IS NON-ZERO) THEN
    {SET ALPHA(DO NOT LOCATE DATUM & MOVE TO INC)}
SETINC="NUMBER OF DEGREES"
}
} /END SEQUENCE 2/

SEQUENCE 3:
{
DO WHILE("CONTINUE RUN" IS TRUE):
{
IF (RAMP) THEN
    {
        SEQUENCE 0:
        {SET ALPHA(DO NOT LOCATE DATUM & MOVE TO "START")}
        SEQUENCE 1:
        {
            DO WHILE ("RAMP CONTINUE" IS TRUE AND LOOP < "RUNS
            BETWEEN RELOCATE")
                {
                    SEQUENCE 0:
                    {RESET WAVE FORM GENERATOR}
                    SEQUENCE 1:
                    {SET TIMING}
                    SEQUENCE 2:
                    {WAIT}
                    SEQUENCE 3:
                    {CALCULATE DRIVE ARRAYS}
                    SEQUENCE 4:
                    {RUN TEST}
                    SEQUENCE 5:
                    {DO WHILE (TEST NOT FINISHED)}
                    SEQUENCE 6:
                    {WAIT}
                    SEQUENCE 7:

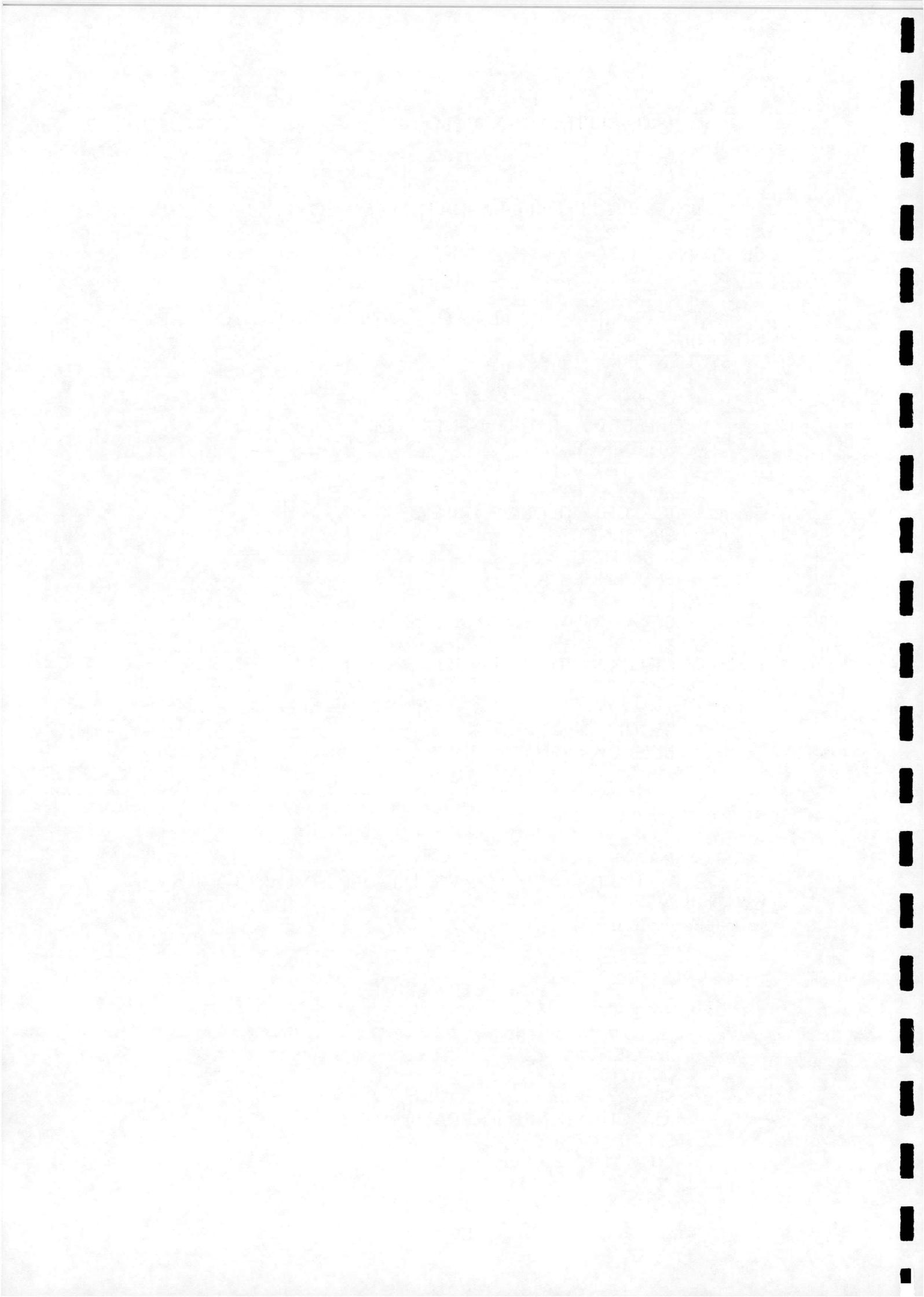
```



```

        {SET ALPHA(DO NOT LOCATE DATUM & MOVE TO
START)}}
        }
        SEQUENCE 2:
        {SET ALPHA(FIND DATUM & MOVE TO DATUM
POSITION)}}
ELSE IF (SINE) THEN
    {
    SEQUENCE 0:
    {SET ALPHA(DO NOT LOCATE DATUM & MOVE TO START
POSITION)}}
    SEQUENCE 1:
    {
    SEQUENCE 0:
    {RESET WAVE FORM GENERATOR}
    SEQUENCE 1:
    {SET TIMING}
    SEQUENCE 2:
    {CALCULATE DRIVE ARRAYS}
    SEQUENCE 3:
    {RUN TEST}
    SEQUENCE 4:
    {
    DO WHILE (OSCILLATORY CONTINUE)
    {
    CHECK STATUS OF OSCILLATORY CONTINUE
    }
    }
    SEQUENCE 5:
    {RESET WAVEFORM GENERATOR}
    }
    }
ELSE IF (TRIANGLE)THEN
    {
    SEQUENCE 0:
    {SET ALPHA (DO NOT LOCATE DATUM & MOVE TO START
POSITION)}}
    SEQUENCE 1:
    {
    SEQUENCE 0:
    {RESET WAVE FORM GENERATOR}
    SEQUENCE 1:
    {SET TIMING}
    SEQUENCE 2:
    {WAIT}
    SEQUENCE 3:
    {CALCULATE DRIVE ARRAYS}
    SEQUENCE 4:
    {RUN TEST}
    }
    }

```



```

SEQUENCE 5:
{
DO WHILE (TEST NOT FINISHED)
{
CHECK STATUS OF WAVE FORM GENERATOR
}
}
}
ENDIF
OUTPUT PITCH RATE DATA
}
} /END SEQUENCE 3/
SEQUENCE 4:
{STOP}

```

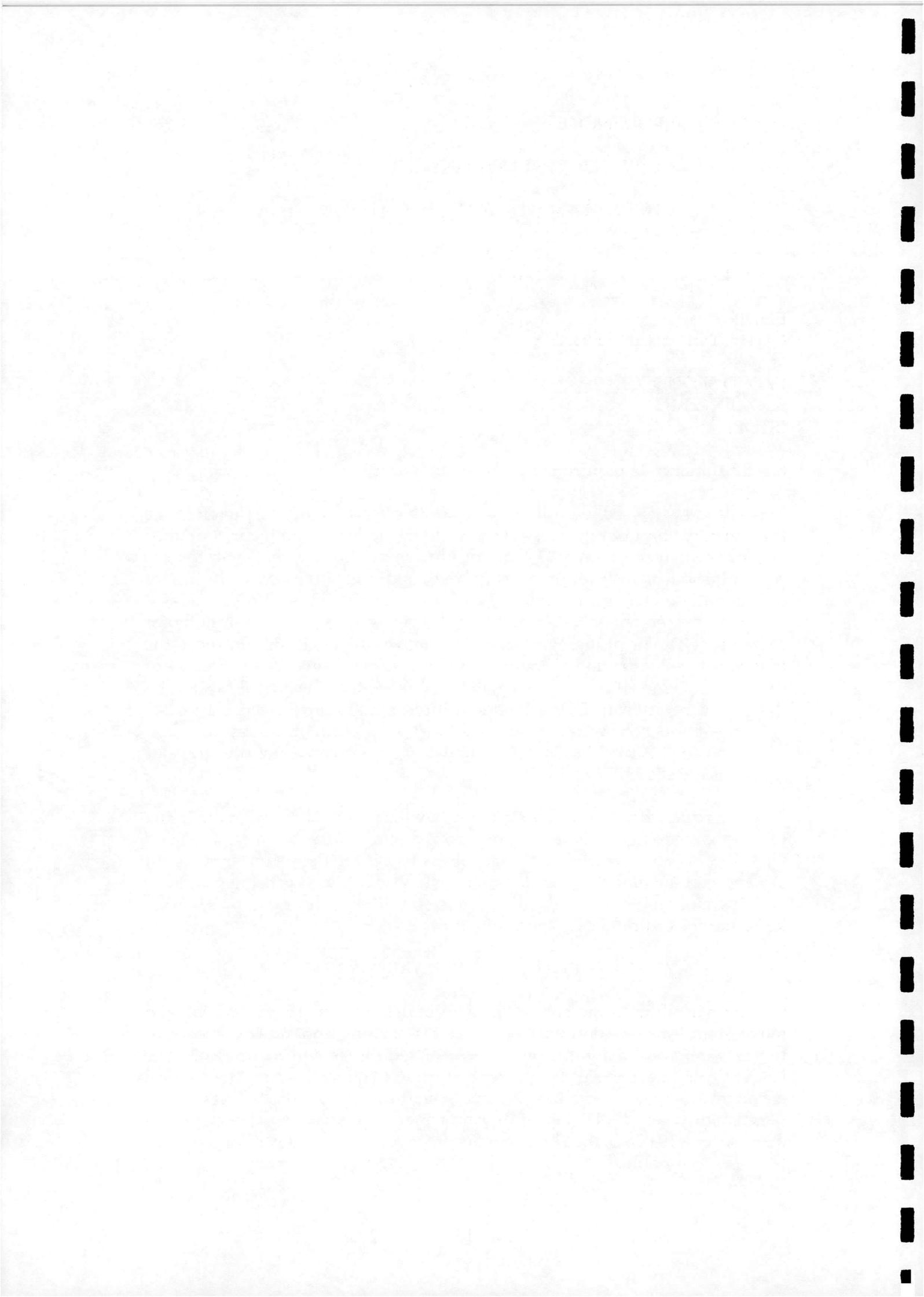
6. Additional code requirements for the ball-screw

The ball screw is used to plunge models or to pitch 3D models. Therefore the transfer function from the stepper motor to the ball screw to the model must be included in the above VIs. In addition in setting up the logic for the configuration of the stepping array, the condition for pulsing the motor changes; the motor must be pulsed each time the model position requires a flange nut increment of $0.02/n$ mm (n is the pulley wheel gearing from motor to ball screw). For plunging motions the above process is simple, since the transfer of motor motion to model motion is linear. However in the case of driving a 3-D model, the transfer function will be non-linear, and to find the flange nut position from the model attitude may require interpolation. A version of the Lab VIEW code does exist at present for plunging, and extension to 3-D pitching is anticipated to be relatively easy (the transfer functions are shown in figure 3).

A further difference in the coding is the way the datum signal is treated. The micro-switches act as safety limits, so during normal running DATUM ENABLE must be switched on. This means that if the flange nut hits a limit switch, then the model will stop moving whether DATUM is being sought or not. Therefore the status of the datum must be checked during ramp and sine tests, and the error indicated accordingly.

SETTING UP THE HARDWARE

Correct wiring up of the apparatus is absolutely essential if the model is to move at all. A basic configuration of the 2D pitching apparatus is shown in figure 4, and the diagram for the NAND gate is shown in figure 2. The NAND gate needs a +5V supply and a ground connection, and the optical encoder uses the same voltage inputs. Note that pin 2 on the NAND gate is permanently set HIGH (i.e. +5V). The trigger threshold on the incidence display is +10V, so the TTL signals from the optical encoder need to



amplified, and the amplifier (NE5532 type) is supplied from a +/- 15V supply.

The wiring from the terminal block is as follows:

Terminal 10; DAC0 CLK signal to NAND gate

Terminal 12; DAC1 Direction signal to stepper motor

Terminal 16; Digital Port A, line 2. DATUM ENABLE signal to NAND gate

Terminal 22; Digital Port B, line 0. Direct read of DATUM signal.

The ball-screw requires the micro-switches to be wired correctly, and figure 1 shows the required wiring diagram.

On setting up, it is the user's responsibility that the wires are connected to the correct places. Note that incorrect wiring may irreversibly damage the LAB-LC and the optical encoder, so great care is required before switching on. Take particular care that power supplies are not inadvertently connected to the LAB_LC or the output lines of the optical encoder; such mistakes are easily made. All pre-prepared wiring has the individual wires clearly marked.

Note that, to reduce interference, the stepper motor controller must be placed as far from the computer as possible.

TROUBLE SHOOTING

If the model does not run even after correct operation of the code, check the following:

Stepper motor controller switched on, enabled and set to mode 1.

STEP and DIRECTION wires connected to remote input of stepper motor controller.

Power supply to NAND gate switched on and connected properly.

All wiring correct (INCLUDING GROUND CONNECTIONS).

If the model still refuses to run, then it is possible that the NAND gate has malfunctioned. Replace with a type 74HC00N.

If the model still refuses to run, then switch the stepper controller to mode 4 (continuous mode), set the frequency the generator to low frequency, and run the model. A lack of response here indicates that either the stepper motor or the controller has malfunctioned. If this is the case, the electronics staff must be contacted. If the model does run, then correct operation of the board must be verified. The check board VI may be run (this simply checks to see if the computer can see the board). If the computer can see the board then at this level, diagnostic tools (e.g. an oscilloscope) must be used as the code runs. If the computer cannot see the board, then the system parameters must be



checked (e.g. are the National Instruments drivers in the right places?). If the user is confident that the board is not functioning correctly, then National Instruments technical support ^[1] must be contacted.

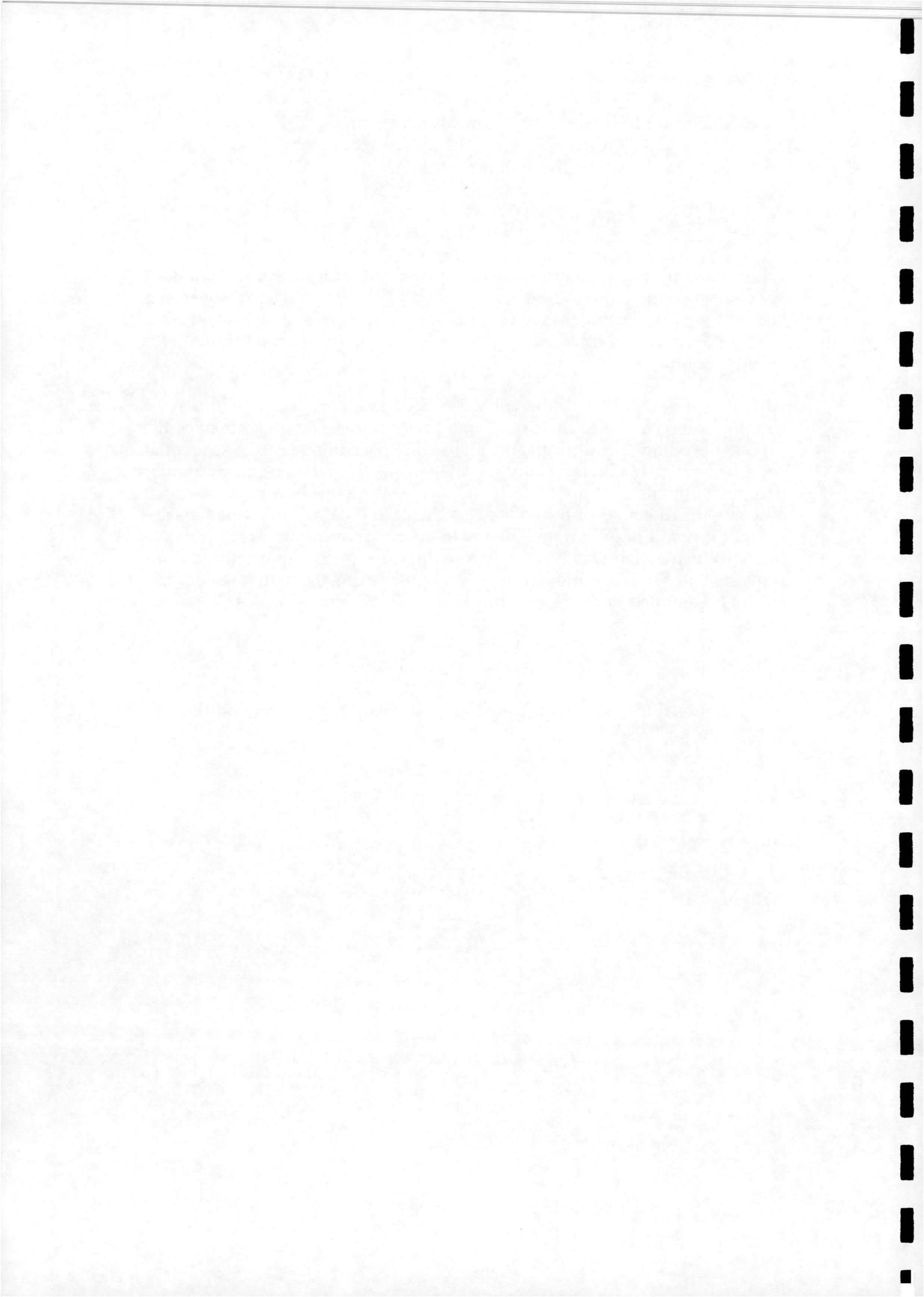
COMMENTS

When running the apparatus, take care that the chosen rate of motion does not coincide with a motor resonance frequency. When the motor resonates, it becomes very noisy and does not step properly. Resonance may explain why, in some circumstances, the model does not reach its required position in, for example, a ramp test.

Although the front end of the LabVIEW code is a very convenient way of running a test, at times the use of this programming language makes the process of coding very difficult and clumsy. Simple editing tasks in sequential code become a major re-wiring job in LabVIEW, and logic in particular becomes difficult to follow if there are a lot of steps on the way to the final outcome. Hard copies of the code are difficult to obtain, are not necessarily straight forward to follow, and, owing to the need for colour, are impossible to reproduce economically in a report. In view of the above the author suggests that VIs are written in sequential code (e.g. 'C') where possible, and are then compiled and linked with the LabVIEW front end, if this facility exists.

REFERENCES

[1] National Instruments LAB-LC User Manual, January 1993 Edition. NI part number 320380-01.



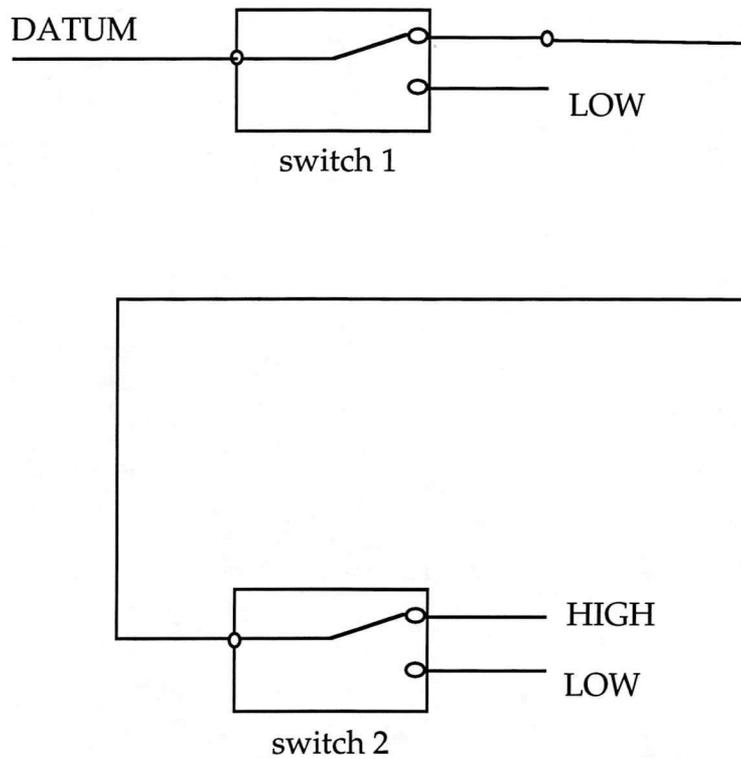
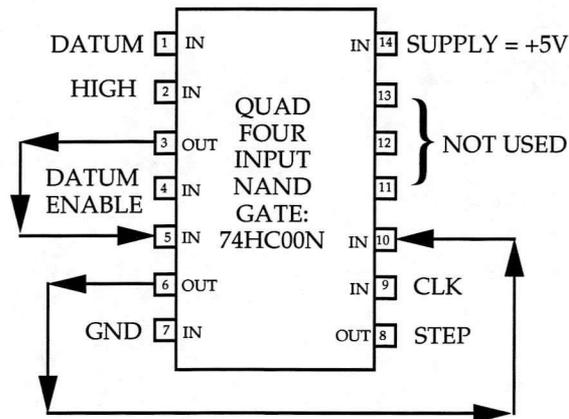
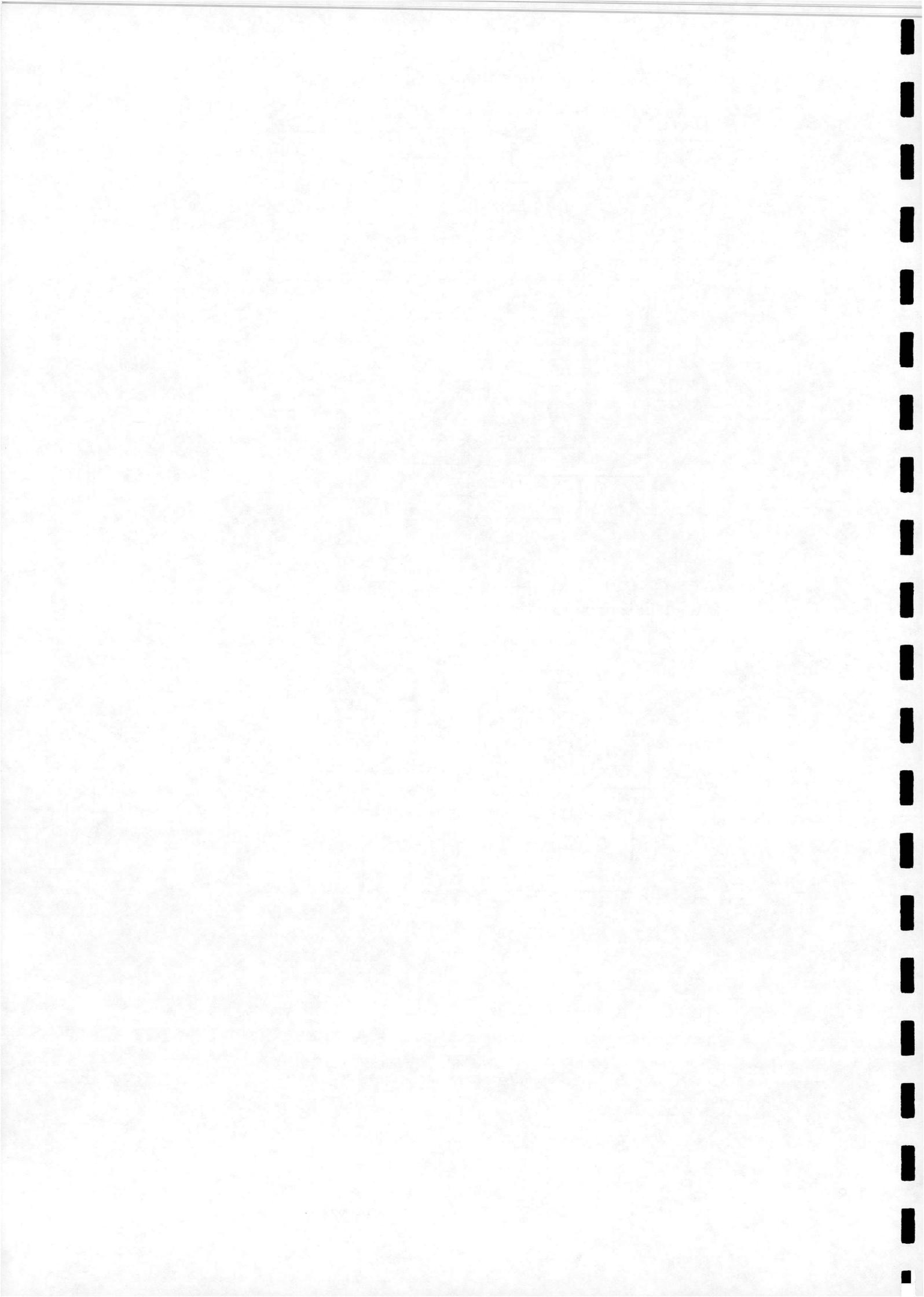


Figure 1 Wiring of micro-switches for ball-screw safety/datum



PIN	STATE			
2	1	1	1	1
DATUM	1	1	0	0
OUTPUT 3	0	0	1	1
DATUM ENABLE	1	0	1	0
OUTPUT 6	1	1	0	1
CLK	1	0	1	0
STEP	0	1	1	0

Figure 2 Datum enabling and finding circuit with truth table



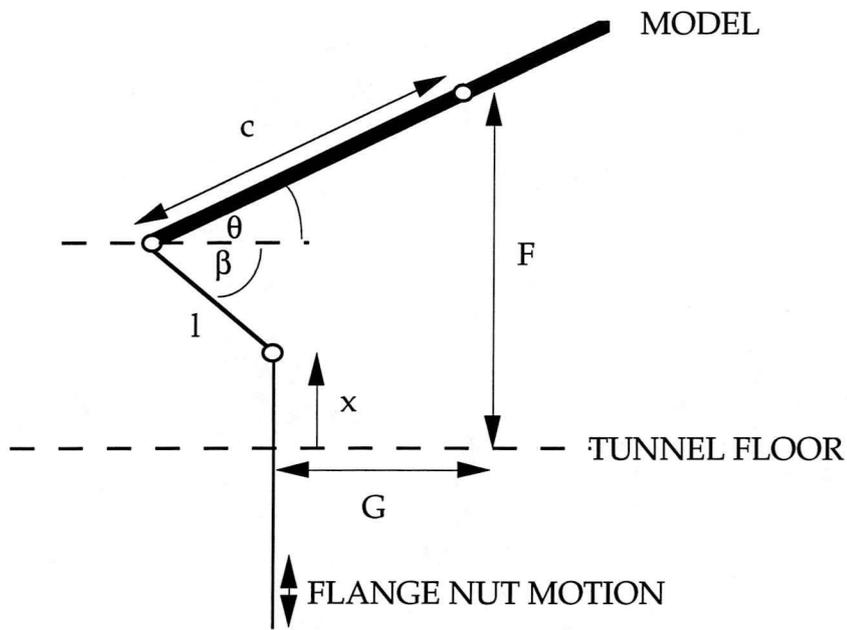


Figure 3 Drive mechanism for the 3-D model.

$$G=c \cos \theta - l \cos \beta, F=x + l \sin \beta + c \cos \theta$$

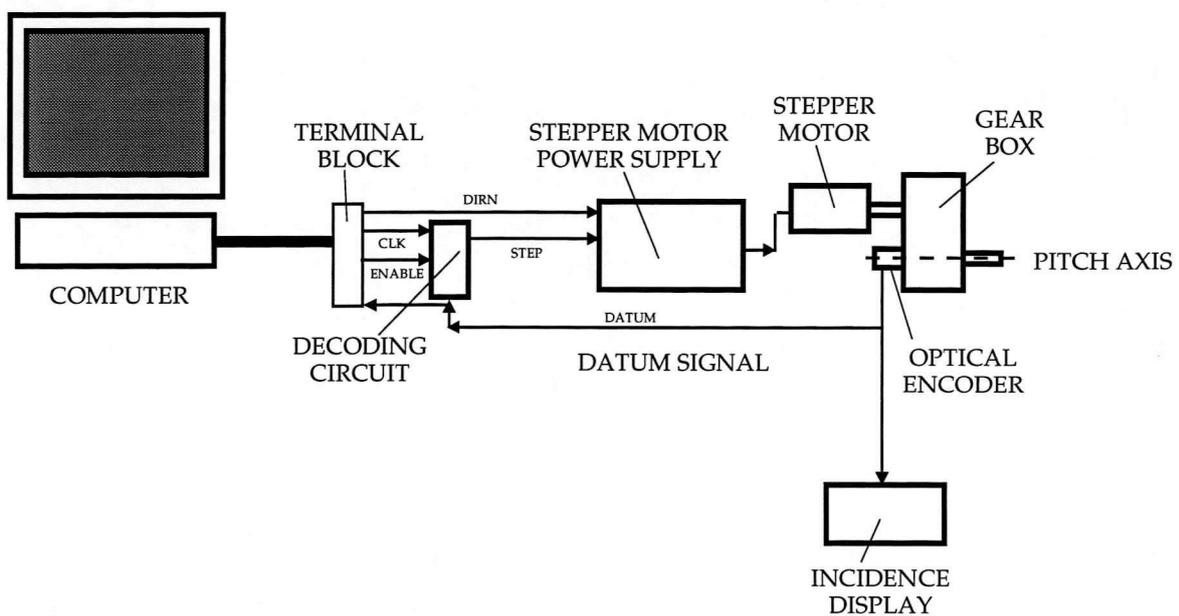


Figure 4 Schematic of pitching apparatus.

