INSTRUMENTATION AND CONTROL
FOR
PRECISION GRINDING MACHINES

P.A.Orton B.Sc.

Thesis submitted to the Council for National Academic
Awards in partial fulfilment of the requirements
for degree of Doctor of Philosophy

Nottingham Polytechnic
Department of Computing
in collaboration with
R. H. P. Bearings Ltd (Newark)

October 1990

Acknowledgements

My wife Alison for enabling work at home.

Dr Steve Billings, from Sheffield University Department of Control Engineering, for his continuous enthusiasm.

Dr Jas Sanghera, formally of R.H.P Bearings, for his continuous support even after leaving the company.

Dr Peter Thomas, principal lecturer and manager of the Microprocessor Centre at Nottingham Polytechnic Department of Computing, for his patient reading of the work.

The RHP bearing company, for financial support and the provision of experimental facilities.

The staff of the Nottingham Polytechic Microprocessor Centre and the Department of Computing, for help in general.

Instrumentation and Control for Precision Grinding Machines

Abstract

This thesis examines the present methods of centreless bearing grinding, with a view to applying modern control methods in order to reduce the distribution of output size variations and improve the quality of surface finish.

Established models of the grinding process show that various important parameters describing the process can be derived, providing that the grinding power can be accurately monitored. Several alternative control strategies based on power monitoring are considered.

The drive systems of a typical grinding machine are modelled as they react through the non-linear grinding wheel/work interface. A simplified computer simulation of the process is demonstrated. Suitable methods of instrumentation are evaluated including the optical shaft encoder as a source of high resolution data for the estimation of drive shaft dynamics. A method is devised for on line parameter estimation of the drive system transfer functions. This enables the important grinding power to be estimated from shaft dynamics and also allows the monitoring of the drive system parameters. The optical shaft encoder is also assessed as the source of data for vibration detection.

A development computer with real time multi-tasking operating system software and specialised shaft encoder interfacing has been established as a basis for an expandable control and monitoring system. A plan for feedback control of the grinding, involving several separate control algorithms communicating with an expert system executive controller, is outlined.

Contents

Appendices

List of figures

List of figures continued

Chapter 4

Chapter 5

Chapter 6

List of figures (appendices)

Appendix A

Appendix B

Appendix C

Glossary of Terms

Alias

Harmonic high frequency distortion of a signal due to sampling at too low a frequency (less than half the cutoff frequency). May be prevented by pre-filtering with a low pass anti-alias filter.

Dead time

The time taken for the change of an input to be reflected by some change in output of a system.

Emulation

Microprocessor development system aid. A development system computer interfaces to a target system via the microprocessor mounting socket with the ability to reproduce the behaviour of a microprocessor whilst at the same time providing monitoring and supervisory activity. Selective real time traces of processor activity are a feature which other types of development aids cannot perform.

Grinding - centreless

A grinding wheel is driven against a rotating work-piece where the work-piece surface forms part of the supporting bearing. In the case of the grinder used in this thesis the work-piece is supported by a set of graphite pads which allow the work-piece to slide round when driven by a magnetic plate.

Grinding - cylindrical

A grinding wheel is driven against a rotating work-piece where the work-piece is supported by bearings which are separate from the work-piece surface.

Grinding - plunge

The grinding wheel feed is directly towards the work-piece with no relative axial motion of the wheel and work.

Grinding wheel dressing

This is the means by which the grinding wheel surface is formed. A diamond tool is driven across the surface of the rotating grinding wheel reducing the diameter of the wheel by breaking down the abrasive material of the wheel. The wheel may be shaped in this way with curves and flat sections.

Grinding wheel work interface process

This is the complex process which takes place at the position of contact of the grinding wheel and the work-piece. The process is considered to be confined to a small volume extending around the contact surface. The volume should be sufficiently large as to encompass the elastic distortions of the wheel and work surfaces due to the grinding action.

Incremental shaft encoder

Instrument designed to generate pulses when subject to angular motion. The number of pulses generated is proportional to angular movement. A typical device comprises of a rotating disc with an optical grating around the periphery and a solid state optical emitterr/receiver

Interpolation

The estimation of variables in the interval between measurements. Interpolation algorithms are used in this research to construct constant time sample estimates from measurements collected at irregular intervals.

Interrupt

Facility of microprocessor to halt execution of the current program at the end of the current instruction and start execution at a predefined address. Interrupts can be initiated by the assertion of processor control inputs.

Model

Mathematical representation of a system normally involving some approximation.

Multi-tasking

Capability of an operating system to allow more than one program (known as "tasks") to run concurrently in the time slice sense.

Parameter estimation

Estimation of values for the model coefficients of a system. The approximate structure of the system is normally established prior to parameter estimation.

Peripheral interface adaptor

Programmable parallel input/output interface chip with interrupt and handshaking circuits (MC6821).

Prediction

Estimation of the future values of a signal based on the previous and possibly current system measurements. One or more steps ahead may be predicted.

Programmable interface timer

"PIT" General purpose counter timer and parallel input/output interface chip. Designed as a true 16 bit MC68000 peripheral with vectored interrupt capability.

Real time operating system

Operating system which allows predictable processing time in response to asyncronous external events. A realtime operating system should also have a fast interrupt capability with minimum operating system overhead.

Simulation

The reproduction of the approximate behaviour of a system through the exercising of system models. The simulation in this research is a computer implementation of a discrete time model of the grinding process.

Systems Identification

Method of obtaining the model of a system.

Time sampling

Inverse of normal sampling practice in which time is recorded at equal increments of change in a given variable (an example in part of this research is time being recorded at equal angular increments in position).

Work support shoes

Support structure with graphite pads which controls the motion of the work-piece.

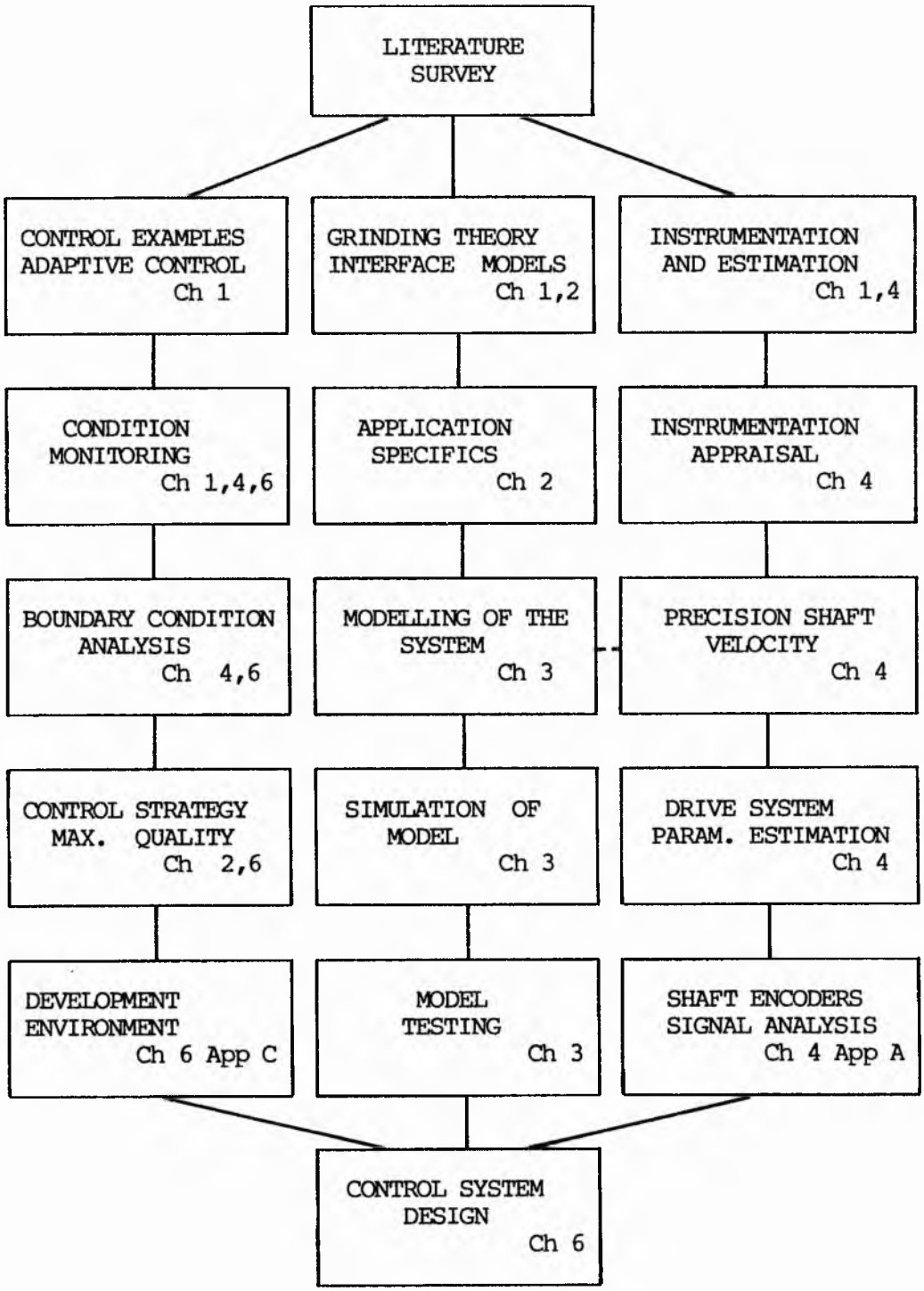
Introduction

The objective of this research is to develop grinding control systems and instrumentation methods. The initiative came from the collaboration of Trent Polytechnic Microprocessor Centre with the sponsor company R.H.P Bearings. The Centre had contributed to a project which resulted in the production of significantly improved grinding machines and control systems. The company had been a participant in the SERC grinding research programme and was interested particularly in implementing advances in the field of closed loop grinder control. The direction this thesis followed was determined to an extent by the particular problems of the sponsors specialised application. Grinding methods are necessarily specialised for particular applications and this inevitably results in the development of a variety of control and instrumentation methods most of which have a valid role. New techniques developed to solve specialised problems can however sometimes create "spin offs". The development of techniques in this thesis for instrumentation could be adapted for other grinding applications and even more generally in engineering not involving grinding.

Figure 1 shows a plan of research in this thesis, illustrated in a block diagram form. This shows three parallel paths; control, modelling and instrumentation. Development along each path included careful consideration of the research as a whole. An example of this is illustrated by the dotted line which is intended to indicate the strong influence of system modelling on the development of appropriate instrumentation. In the corner of each block relevent chapters or appendices are indicated.

Figure 1 Research plan

GRINDER CONTROL SYSTEM



Chapter 1 explains some of the background of modern grinding in general and the influence of skilled labour in the refinement of the process. The most recent developments in grinding theory and adaptive control are considered. The SERC co-ordinated grinding research programme and the contributions relevent to this thesis are assessed.

Chapter 2 considers the particular application which is taken as an example of the sponsor company's work. The factors which influence the choice of appropriate control system techniques are investigated. The need to give a high priority to quality in relation to quantity for small batch production in this application is established.

Chapter 3 is concerned with the modelling of a particular application of grinding bearing rings. This work indicates that the angular velocities of the main drives could be used as a major characteristic of system behaviour.

Chapter 4 assesses available instrumentation methods and develop the idea of shaft velocity measurement for non intrusive system characteristics monitoring. The use of online parameter estimation is established as a powerful technique for system condition monitoring.

Chapter 5 presents an outline for an appropriate integrated hierarchical control and monitoring system.

Chapter 6 deals with the requirements for suitable multiprocessor control system development facilities.

Chapter 7 Conclusions and further work.

Chapter 1
The development of grinding technology

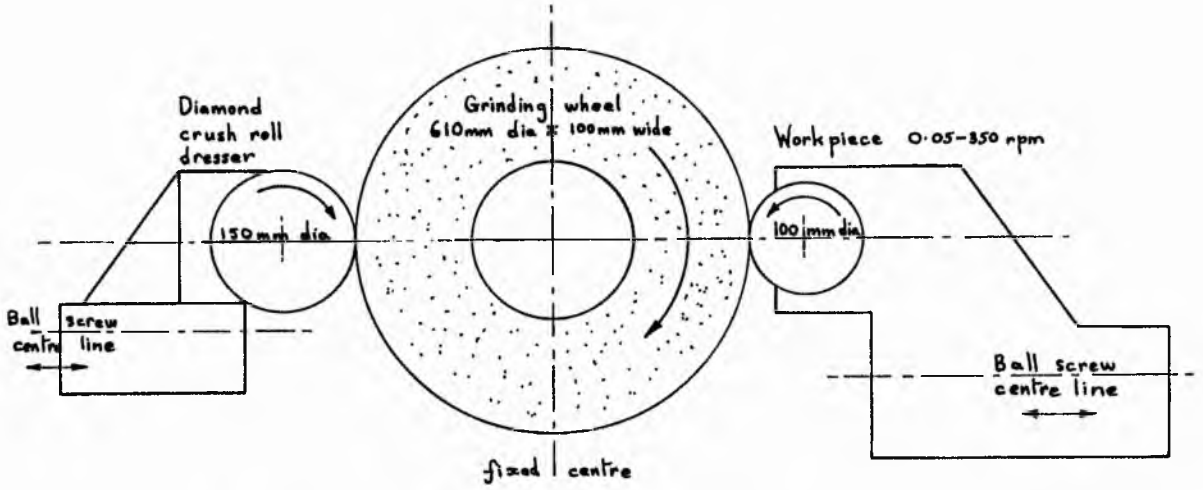
1.1 Historical origins

The grinding of metal has been carried out since man's first attempts to fashion metal into crude weapons. From the start, the effectiveness of the operation depended on the skill and accumulated experience of the individual. It was possibly in the time of very early clockmaking that grinding was first used as a method of metal removal to manufacture components to within parts of a millimeter accuracy. Today much of the industrial precision grinding is still based on empirically gained skills.

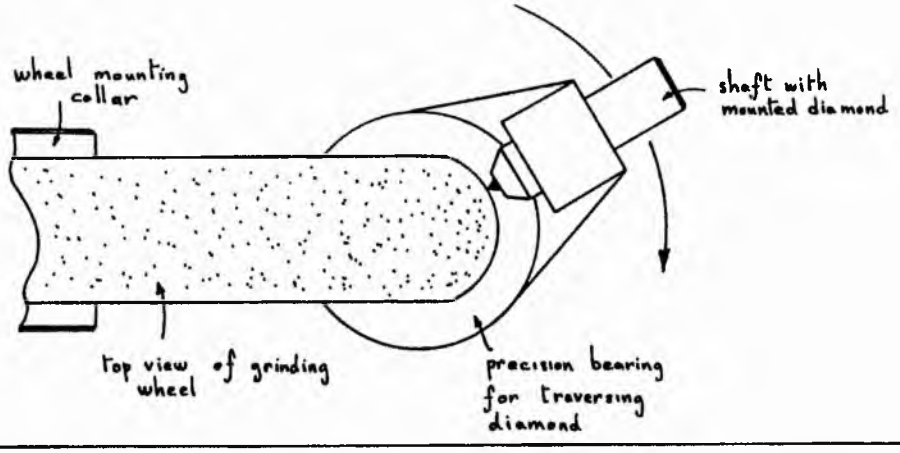
1.2 Grinding used for precision finishing

Grinding is used in the engineering industry for machining components. In a typical grinding application, metal components are loaded into a specialised grinding machine for the purpose of metal removal. Dimensional and surface finish tolerances are maintained. Grinding is the principal method used for the finishing of components which are either hardened or made from particularly tough material. In such cases milling or turning, may not be possible or may not allow surface finish or dimensions to be maintained. Generally the process has been used as a final stage in metal removal where the first stage involves a rough machining of the components to typically within half a millimeter of finish size. Recent developments have been made which allow for the roughing and finish stages to be efficently combined into a single grinding operation (Fig 1.1 creep feed grinding ref 1, see section 1.6). This method has, as yet, only been found appropriate for particular types of component and material conditions. Modern technology has been, until recently, concentrated on the mechanical side of grinding machine technology.

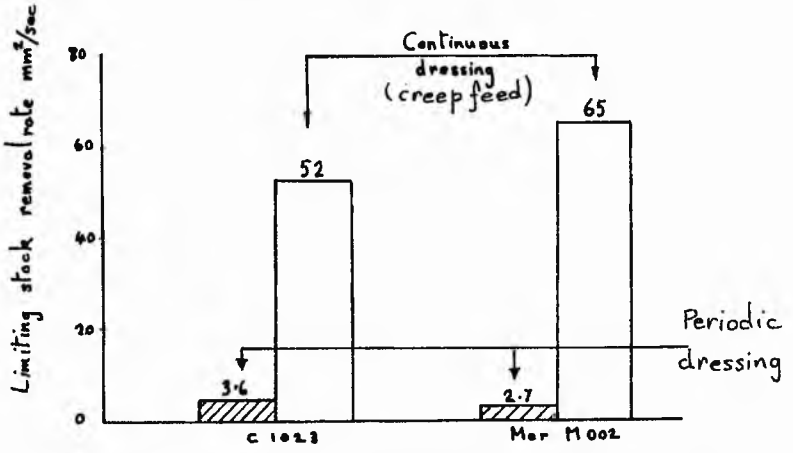
Fig 1.1 Creep feed grinding
Fixed central grinding wheel



Dressing a grinding wheel profile using a single point diamond



Effect of continuous dressing on the stock removal rate



Wheel UN1 WA 60/80 FP 2V 30 m/s 3mm d.o.c.

Machines with more precise bearings, slides and actuators and more rigid structures have been developed. The skilled operator of a modern grinding machine, using tables of feeds and speeds usually empirically produced, is able to manufacture components to sub-micron accuracy. This can be done economically with little or no idea of the complex physics of the process taking place at the grinding wheel work interface.

Automation of component grinding has been common. The control of the machine cycles for the production of up to many thousands of components can be entirely automatic, including the periodic dressing of the worn grinding wheel.

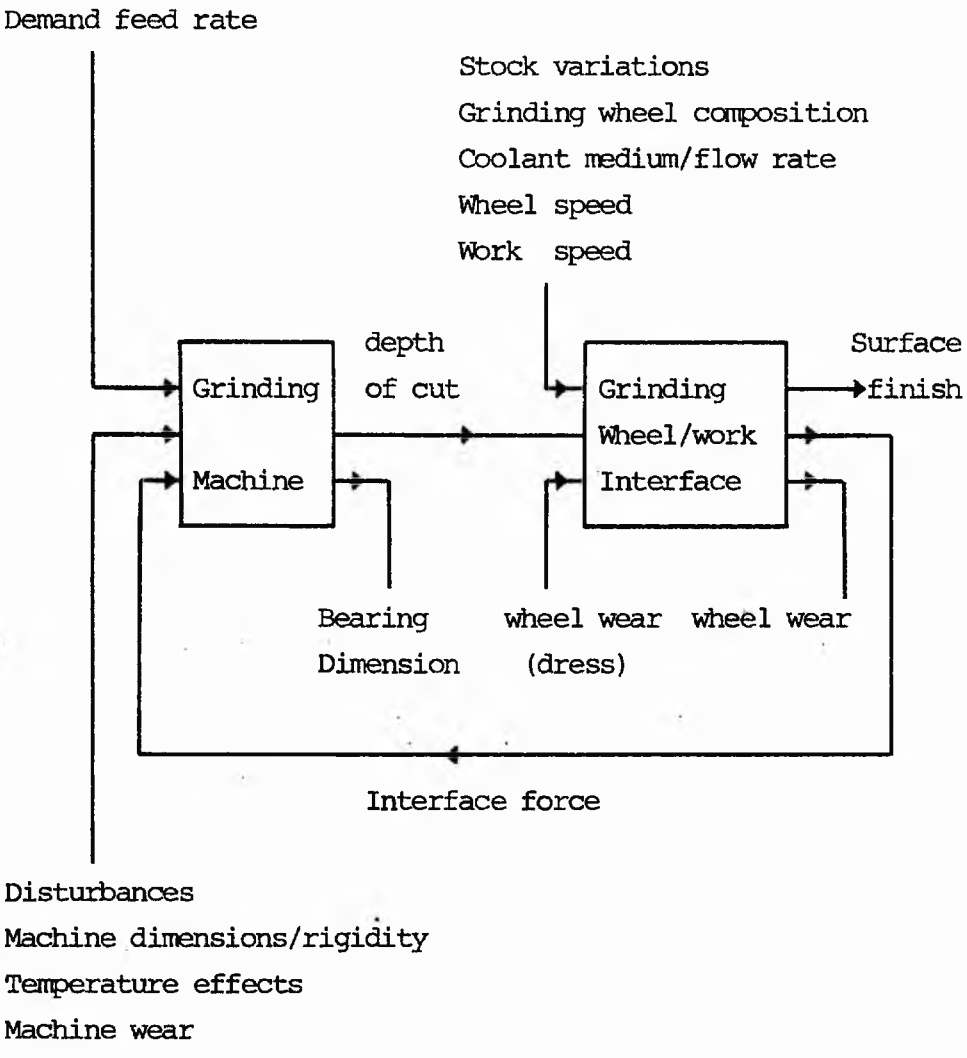
In general, production of this kind is open loop, in the sense that once the equipment is set going there will be no automatic adjustments to the controls. The most common exception to this is the gauging of component dimensions being used to control the grinding wheel final infeed position. This is done in order to compensate for random changes in the machine's structural dimensions. Manually performing this adjustment can also be considered as a form of closed loop control.

1.3 Grinding interface process

At the grinding wheel work interface the removal of metal can be considered on a microscopic level. A rigidly supported small particle of hard crystalline material with sharp irregular shape is driven along the surface of a softer metal material. The edges of the particle in contact with the metal, shear and scrape away fragments of the metal surface. This can be considered as an elemental part of the action of a grinding wheel as large numbers of abrasive particles produce a combined effect which removes metal fragments in large numbers. Numerical methods for describing the kinematics of grinding based on the effects of individual cutting edges have been developed (ref 18) which give the most detailed understanding of this complex process.

Figure 1.2 Typical grinding process

Decomposed into grinding machine
and
grinding wheel work processes



The grinding wheel when properly prepared allows tens of thousands of abrasive particles to be moved across the metal contact area in a continuous stream. The effects depend on the instantaneous vector velocity of the metal contact surface in relation to the wheel. In the case of surface or cylindrical grinding this vector velocity has clearly been arranged to be constant or at least have a straight forward relation to feed rate. A by-product of the grinding interface process is the generation of heat which can damage the metal surface. The conditions that contribute to burn damage have been investigated (ref 6) and related to grinding power or interface force and grinding wheel wear. Fig 1.2 illustrates this process and its inputs and outputs. In most applications the majority of the inputs are fixed or only slowly changing over many numbers of grinding cycles. The main input variable to this process is considered to be the wheel work interaction force (ref 19) or alternatively the infeed rate (ref 2).

1.4 The grinding machine and structure

The principal input to the grinding machine is here considered to be the position of the grinding wheel as determined by its infeed actuator (servo/stepper motor driven ballscrew). This motion results in the changing of the component dimensions once grinding begins but the effective change does not exactly correspond to the infeed position. The grinding process interface reaction force will cause deflections in the structure and bearings of the grinding machine in the order of microns in magnitude. The grinding interface process may be seen as coupled to the grinding machine with input as the effective feed rate and feedback as the interaction force (it may equally be considered the other way round). In certain cases the surface speed of the grinding wheel and work piece are also considered as input variables (ref 9).

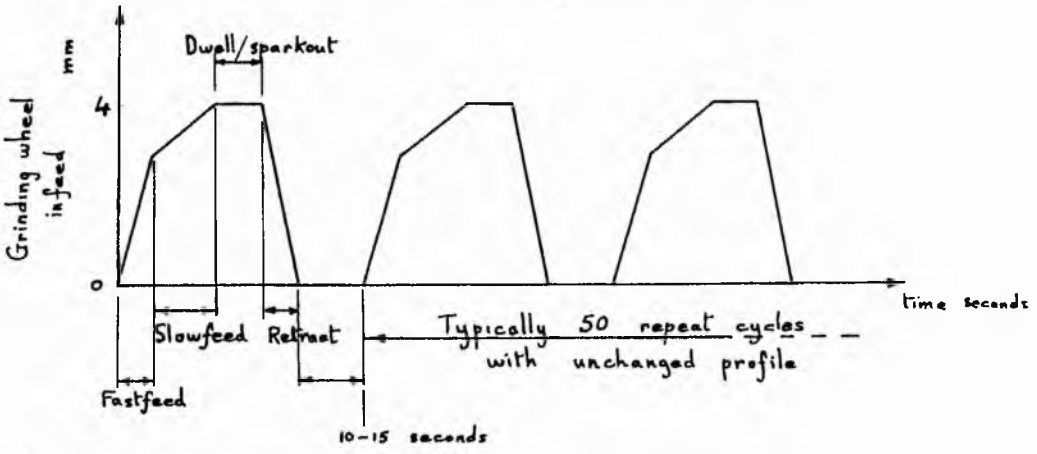
The grinding wheel surface finish properties should be considered as both input and output since it is determined by wheel dressing and as a consequence of the process itself. The analysis of this process is simplified in cases like cylindrical grinding where extended periods can exist in which there is almost constant interface force, depth of cut, surface velocity and thermal balance of heat dissipation.

1.5 Adaptive control

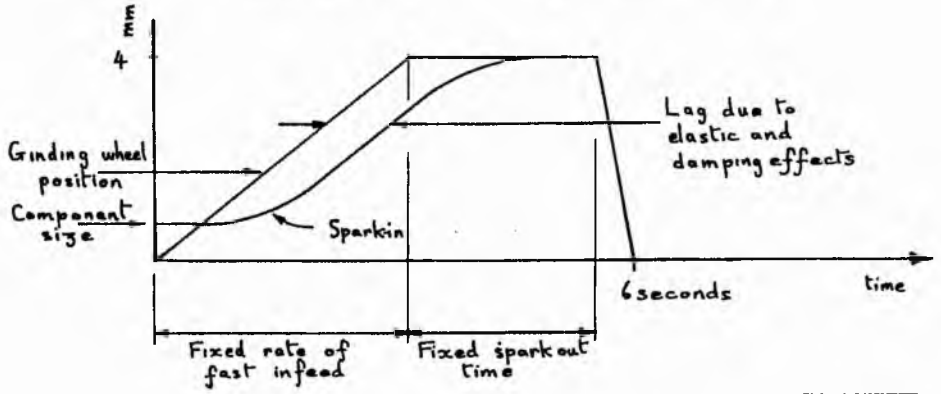
The basic idea of adaptive control for grinding machines is that measurements of the grinding system performance on line or at the end of each cycle should be analysed and the results used to estimate adjustments to the grinding machine control inputs. The objective of the control is to improve some index of performance. This index can be varied but is often as simple as the need to maximise output given constraints on dimensions and surface finish (ref 9,7). In order to devise an appropriate control algorithm it is necessary to have an idea of the physical relationship of the inputs to the output. In the case of some multi-input multi-output systems that have reasonably linear transfer functions it is possible to estimate the transfer functions on line. There does not however appear to be any evidence of research to this end. The established grinding interface models suggest that the process behaviour at the grinding wheel work interface in particular is highly none linear. Fig 1.3a shows a typical grinding wheel feed cycle for a repetative machine. Fig 1.3b shows how an adaptive control system can modify the cycle. In this example as part of the adaptive controlled infeed a level of overshoot is used to reduce sparkout time.

Fig 1.3 Comparison of grinding feed cycles

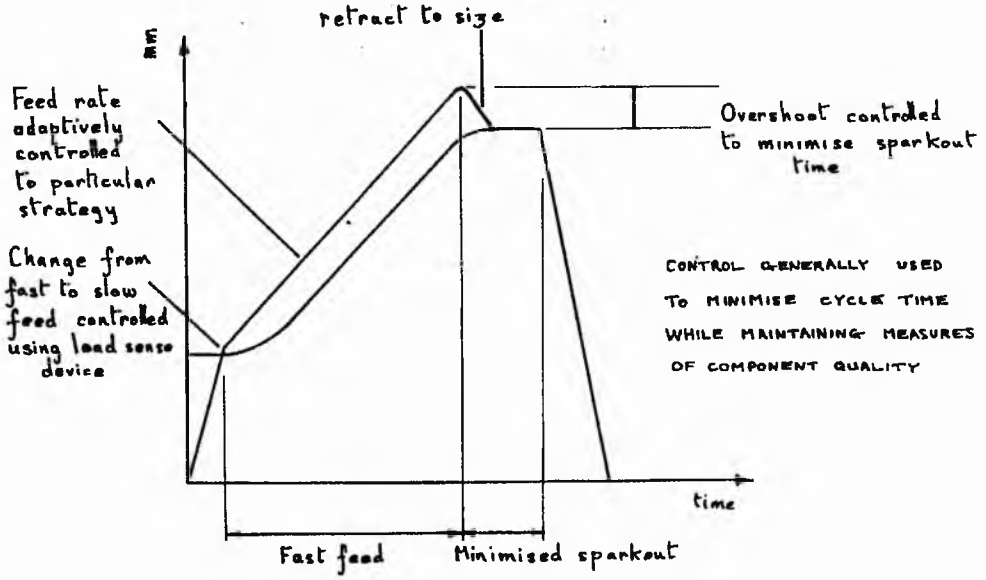
(a) Repetitive – without feedback control



Detail of single cycle showing component size



(b) Feedback controlled cycle



1.6 SERC grinding research program

In 1973 the Manufacturing Technology Commitee of the Science and Engineering Research Council (SERC) undertook a survey of the engineering industry on the role of grinding. The results highlighted the key importance of grinding and the widespread expectation of the industry for future development of the process. The co-ordinated grinding research program was set up in the following year. This involved many leading universities, and polytechnics and was supported by major engineering companies until its finish in 1982 (ref 1).
Under the grinding research program a whole range of work was carried out into aspects of grinding technology which can be divided under the following headings-

a) Economics
b) Machine structure
c) Control systems
d) Drives
e) Geometrical accuracy
f) Abrasive Technology
g) Coolants
h) Grinding wheel dressing
i) Dynamic performance
j) Mechanics of grinding

Most interesting as far as the particular application of bearing grinding (see chapter 2) is concerned are those projects involving efficient methods of control and instrumentation.

The work at Birmingham and Warwick Universities demonstrated modern instrumentation methods suitable for closed loop grinding control. At Birmingham novel techniques were used to develop inprocess size and shape gauging equipment as part of a comprehensive adaptive control system. The aim was to minimise production time whilst maintaining dimension and surface finish tolerences. The output of seven separate transducers was used as feedback for a multi-processor controller (ref 3,5, see chapter 4).

At Warwick University a hydrostatic bearing system with pressure transducers was used as a non intrusive way of measuring the important grinding interface force vector. This method has also been used by He Xiu-shou in an adaptive control implementation. The importance of this vector for feedback control is discussed in chapter 4.

A substantial and successful part of the research aimed at high efficiency high stock removal applications (creep feed grinding at Bristol University ref 1). This developed the technology suitable for roughing and finish forming, difficult to machine materials, in a single machining operation. Fig 1.1 page 5 shows the layout of a creepfeed system. The graph shows the dramatic improvement in removal rate that can be obtained. The idea of large stock removal by grinding had not in the past been seen as economically viable.

Key features of this technique are as follows.

a) Depth of cut is normally the full the amount or nearly the full amount required for finishing grinding. This means only one or two slow speed rotations of the work piece is required in the case of cylindrical grinding.

b) The power flux at the grinding interface can very easily exceed the burning point boundry. To reduce the power the grinding wheel is kept sharp by a continuous crush wheel dressing method. This can facilitate very high removal rates (up to 20 times that possible by using conventional methods). The resulting increase in abrasive material wear rate is only a factor of two. Crush wheel dressing produces a qualitatively better cutting surface than the single point diamond method.

c) The increased depth of cut is benificial in the case of form grinding as there tends to be less loss of form.

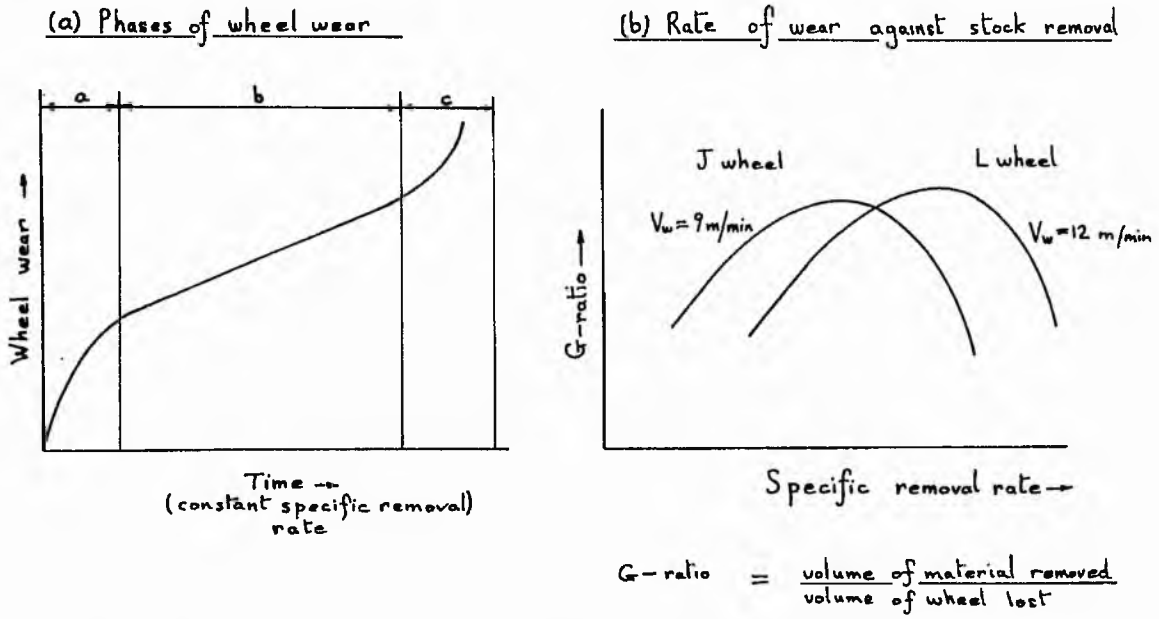
d) The method is most advantageous for difficult to grind materials such as high nickel alloy steels. Bearing steel is considered easy to grind.

e) The precise control of the rotation of the work becomes far more critical as it determines the stock removal rate. In the case of normal cylindrical grinding the stock removal rate is determined by the grinding wheel infeed rate only.

f) Precise control of the continuous dressing is also required in order to ensure economic use of the abrasive material.

Work carried out into grinding wheel wear effects by Bhattacharyya and Moffat is interesting from a control systems point of view (see Fig 1.4). Fig 1.4a shows the phases of wheel wear illustrating that there is clear linear region which could be monitored for control purposes. Fig 1.4b shows that an optimum stock removal rate exists for minimising wheel wear. Fig 1.4d shows that there is an optimum wheel hardness which balances wheel wear against excessive normal grinding force and heat generation.

Fig 1.4 Grinding wheel wear effects



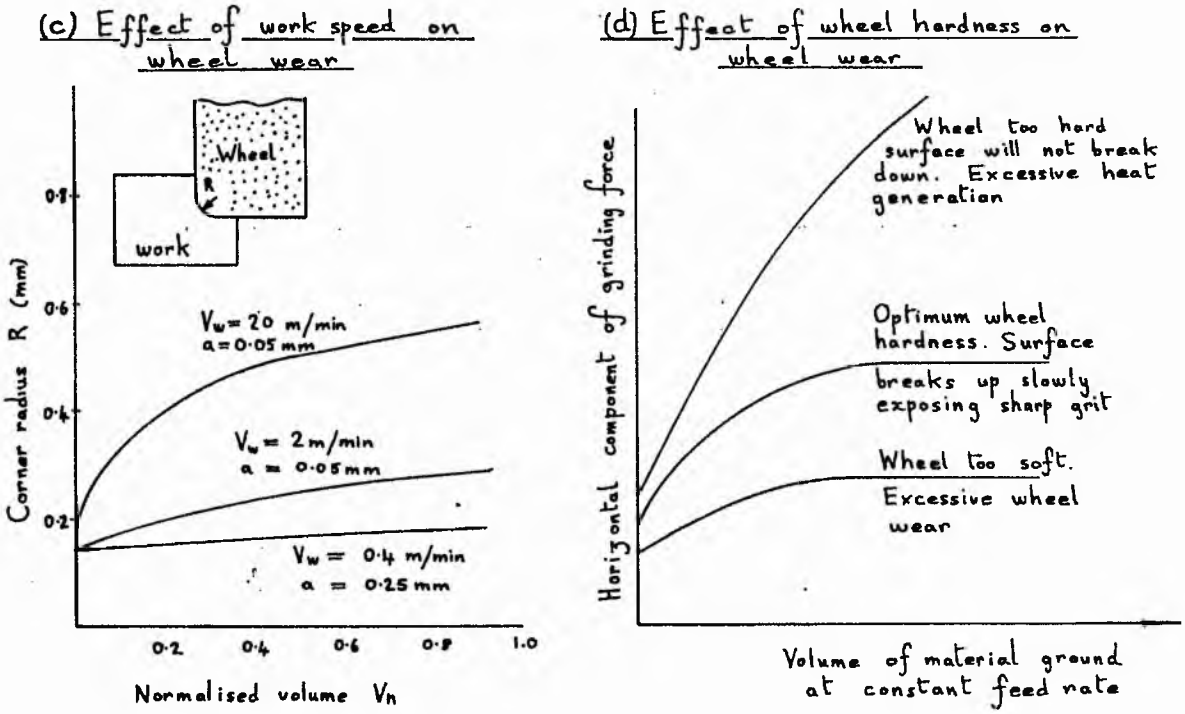
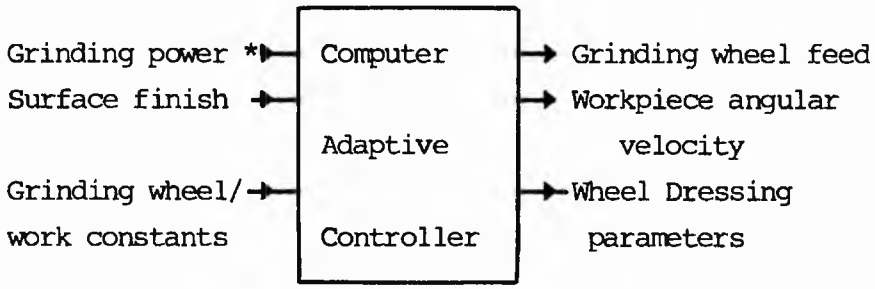


Fig 1.4c indicates that form holding is improved at lower work speeds. This indicates a possible reason for the particular success of creep feed for form grinding. These wear considerations can play an important part in a comprehensive control strategy (chapter 6).

1.6 International grinding research

During the same period Malkin developed an adaptive control system based initially on a single grinding wheel motor power transducer for feedback information (ref 9). Surface finish was measured and the result fed to the controller manually. In this work, as in the Birmingham controller, the objective was to minimise production time whilst maintaining surface finish and size tolerences. This work revolved around the use power series equation models which relate total grinding power to feeds speeds wheel wear factors etc. These models are further discussed in chapters 3,4,5.

Figure 1.5 Malkin Adaptive Control

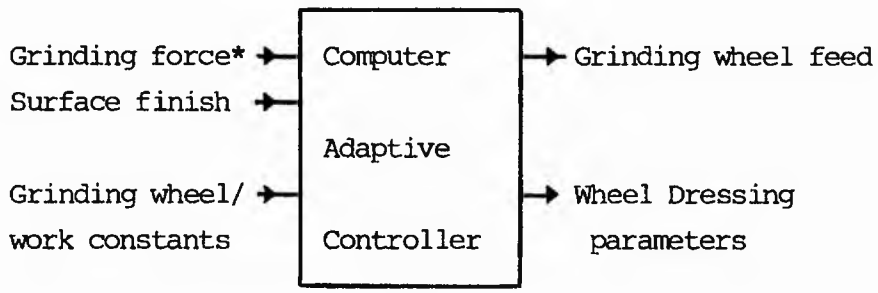


Minimises cycle time
Avoids burn damage
Maintains surface finish
*proportional to tangential
component of grinding
interface force

A further enhancement to Malkin's controller (ref 8) involves the use of inprocess gauging to compensate for wheel work deflections and minimise sparkout time. The reduction of sparkout time reduces the production time but may also have a positive effect on surface finish since evidence exists to show that damaging parametric vibrations can develop during extended sparkout (ref 17). Hahn (ref 11) considers that the need for inprocess gauging is less if interface force is measured since wheel work deflections can be estimated.

An alternative approach is that of controlled force grinding. This has been applied by Hahn and also He Xiu-shou (ref 10). Hahn's "Force adaptive control" used grinding force measurements to estimate wheel work deflection and make compensation as an alternative to inprocess gauging. Hahn considered that the need for inprocess gauging is less necessary using his approach Fig 1.6.

Figure 1.6 Hahn Force-Adaptive Grinding



Compensates for wheel
work deflections
Avoids burn damage
Maintains surface finish
*Vector force

He Xiu-shou's basic control principle was to apply constant force grinding during the roughing stage and constant feed for the finishing. An interesting instrument for measuring radial wheel wear based on a constant pneumatic pressure nozzle floating on the wheel was also used. A differential transformer (LVDT) was used to measure the displacement. Grinding wear was also assessed by analysis of grinding noise (audio frequency). A dull wheel is shown to give increased noise in the 660 Hz region.
He Xiu-shou considers that measurement of the normal grinding force using strain gauges or load cells is not adequate in the production situation and a built in hydrodstatic bearing force measurement method is used. Chapter 4 deals with the instrumentation aspects in more detail.
Production grinding machines are now available with the controlled force feature. The principle results in uniform machine structure deflections as the wheel becomes worn. It also limits the power that can be exerted at the wheel work interface to an almost constant value reducing the possibility of wheel surface break down and workpiece burning. The rate of infeed is automatically reduced as the wheel becomes worn.

1.7 Conclusions

The creep feed method has not been applied to centreless grinding. In the case of shoe type centreless grinding as used for bearing rings it may be feasable but the geometric effects on the rounding of the part with only one or two rotations would probably result in reduced accuracy. To apply the method to bearing grinding would also require significant changes to the conventional equipment available. Modification would have to be made to the workpiece drive for low rotation speed and suitable crush dressing may need to be developed. However if only the existing low levels of low stock removal were kept it may still be possible to dress after each cycle.

An interesting factor is that it is generally implied by this development that lower work surface speeds are better in general. In the case of form grinding most grinding wheel manufacturers recommend low work speeds to reduce form loss. The reason higher work surface speeds have become generally accepted in conventional grinding is probably due to the grinding power flux exceeding burn limit conditions for low speeds as the wheel becomes worn. Another reason the method is not considered appropriate is because the material is comparatively easy to machine. It is possible that large stock removal after heat treatment could cause stress problems. Considerable costly changes to the equipment to improve removal rate are not desirable when the existing removal rate is fast enough. In precision bearing grinding applications the emphasis is towards techniques that lead to an improved quality of output whilst minimising the amount of scrap. Increasing the production rate is seen as having lower order of priority. The particular requirements of bearing grinding is covered in more detail in chapter 2.

The examples of the use of adaptive control are mostly concerned with simple control objectives to minimise production time whilst maintaining dimensional and surface finish tolerences. A different set of control objectives relevent to the application of bearing grinding is discussed in chapter 2.

Malkin and Kaliszer considered the qualitative improvements that can be obtained by using inprocess gauging. In the case of bearing grinding this may not be practical (see chapter 2). Attempts have been made to derive the maximum information from single transducers through the use of mathematical models which facilitate the estimation of parameters not directly measured. Malkin estimates many factors after simply measuring motor current. Hahn uses force measurement to estimate structural deflection etc. In chapter 4 ideas for further development of simplified yet accurate instrumentation are raised.

Chapter 2
Description of a typical grinding process

1.1 Provision for experimental work

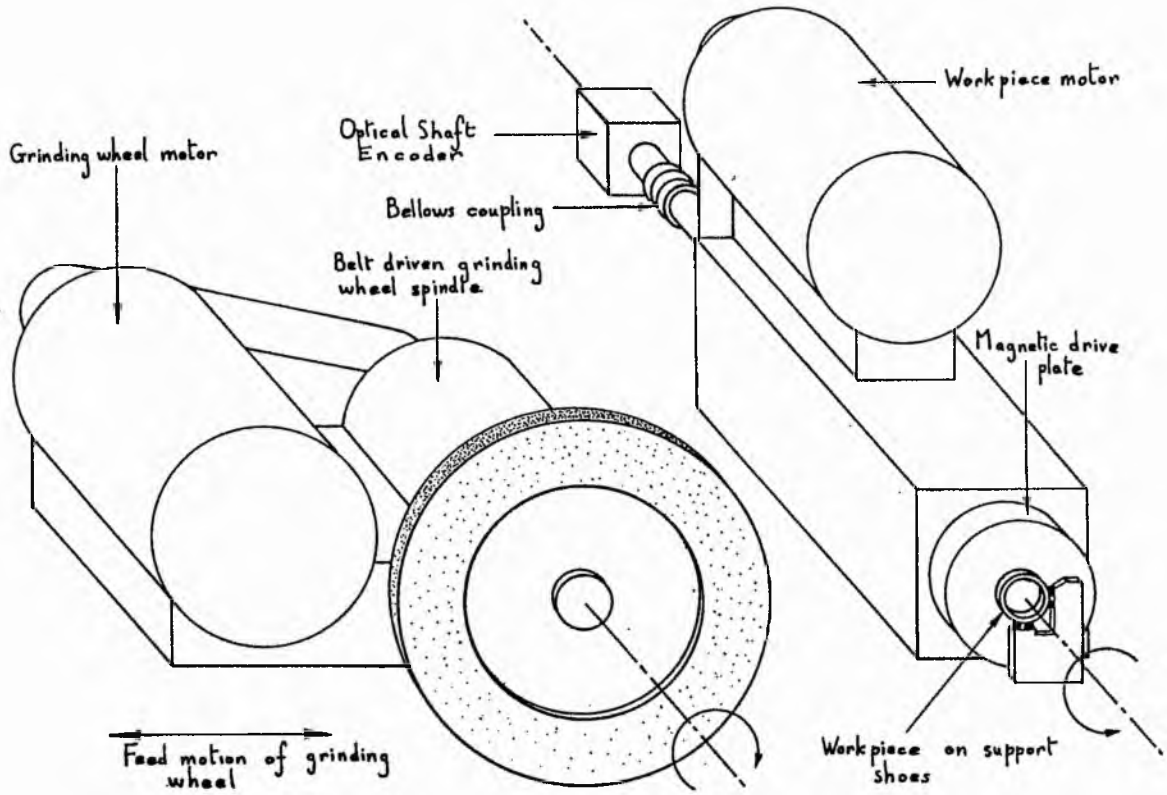
The type of grinding carried out by the industrial sponsor company "R.H.P. Bearings" is of a specialised nature. Of particular interest to the company is the internal and external grinding of bearing rings. The company previously supported a Phd research project concerned with condition monitoring of the internal grinding process (ref 13). The company provided an internal bearing ring grinding machine for the experimental work. In support of the research of this thesis an external ring grinding machine was made available. Some of the important features of this machine are illustrated in Fig 2.1. A significant feature that should be noted is the automatic loading device. This device restricts access to the component and places constraints on the type of in process instrumentation that can be used practically. Details of the equipment are as follows -

Microcentric grinder
External form grinding of the ball bearing grooves
Dia of components typically 50mm
Accuracy - Ground diameter tolerance 20 microns.
Shoe type centreless grinding.
Component driven by eccentric magnetic plate

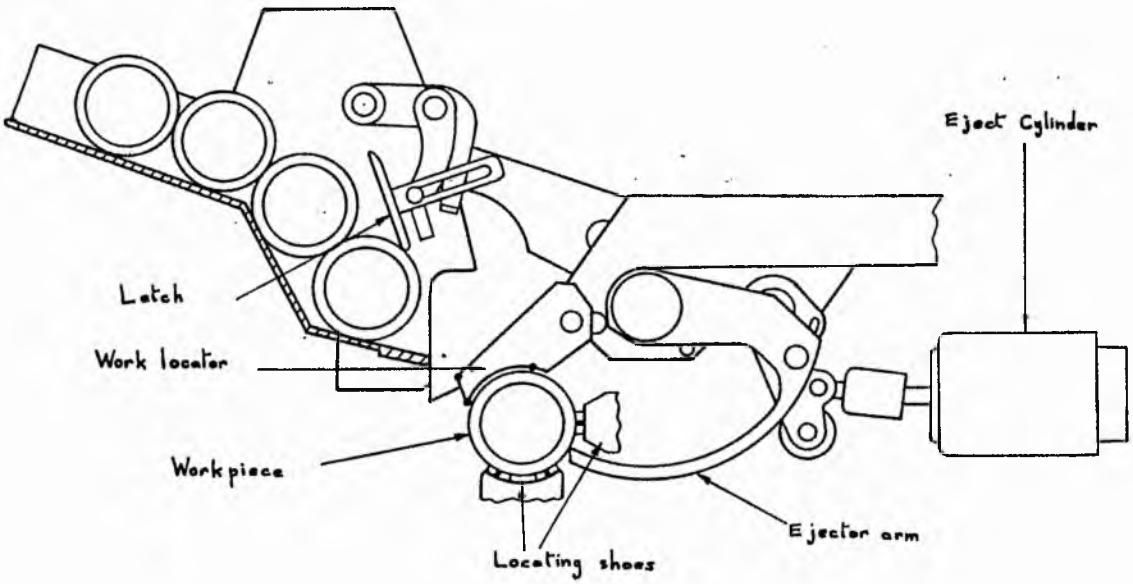
This equipment was specially fitted with an optical shaft encoder as shown in Fig. 2.1. This was used in the experiments with the data logger explained in appendix E.

Figure 1.2a chapter 1 illustrates the timing sequence of the machine and Figure 2.1 shows the general structure. The plunge grinding method is used. That is to say that the grinding wheel (which has previously been diamond shaped with a convex pattern) is forced into the work normally at constant velocity. Fig 1.2a shows the infeed cycle of the grinding wheel. There should be no motion in an axial direction between the grinding wheel and the work. The component revolves on two sets of tough graphite pads, as shown in Fig 2.1, placed as shown at 90 degrees apart. This is an example of centreless grinding. The mechanics of the motion of the work is complex and is the subject of much research (ref 35). Of particular importance from the control point of view is that the loads generated from the grinding wheel/work contact are borne predominantely by these pads. The work piece is driven by a magnetic plate which allows the work piece to slip on its surface and thus rotate on a centre slightly offset from the drive shaft centre. The direction of the offset is carefully chosen so that the work piece is driven constantly and firmly against the two graphite shoes. The actual centre of rotation of the work is fixed by the pads. The location of the pads and the offset are critical and affect the stability of the process.

Figure 2.1 External bearing ring grinding



Automatic loading device



2.2 Machine Cycle

Refer to Figure 1.2a for the actual timing of the following sequence.

1) Work (un-ground bearing is loaded by fast auto-load mechanism into the grinding position). The magnetic drive plate is engaged.

2) Grinding wheel advances rapidly to the work.

3) Grinding wheel reaches a predefined position close to the work and changes velocity to a fraction of the approach speed.

4) Grinding wheel proceeds touching the work soon after slow feed. Grinding takes place. The grinding wheel stops advancing into the work at a preset distance.

5) A preset time delay takes place to allow for what is termed 'spark out'.

6) Grinding wheel rapidly with-draws to its original position.

7) Work is unloaded.

The machine is under the control of sequential logic so that the cycle described above is repeated a predefined number of times.

2.3 Machine cycle adjustable parameters

The following parameters in the present example are set constant for extended periods, being altered mainly when the equipment is set up to grind batches of bearings of different design dimensions. Potentially however, all these parameters could be adjusted or controlled during the cycle.

a) The distance of the fast feed.

b) The slow feed rate (infeed velocity).

c) The final feed position (determines finish size).

d) Dwell or spark out duration.

e) Angular velocity of the work piece drive.

2.4 Control outside machine cycles

The machine cycle can be considered as the innermost loop of a discrete control system aimed at maintaining bearing size and surface properties. A typical example has been expressed in terms of "Program design language" (ref 53) see figure 2.2

Figure 2.2 Discrete Control Loops

```
loop -1:
    Adjust/change machine bearings/drives
    REPEAT UNTIL  (machine worn)

    loop -2:
        Adjust/change worn wheel dresser
        REPEAT UNTIL  (dresser worn)

        loop -3:
            Change Grinding Wheel
            REPEAT UNTIL (grinding wheel too small)

            loop -4:
                Dress Grinding Wheel
                DO FOR index = 1 to 10

                loop -5:
                    Make size correction
                    DO FOR index = 1 to 50

                    loop -6:
                        Grinding cycle. Grind one bearing
                    END DO
                END DO
            ENDREP
        ENDREP
ENDREP
```

The illustration figure 2.2 shows the typical nesting of the control adjustment loops required to maintain accuracy over an extended period. Often only the machine cycle loop is automatic.

2.5 Size Control (Loop 5)

The finish size of the bearings depends, amongst other factors, on the stability of the machine structure. The repeatability of finish position of the grinding wheel, relative to the work piece centre, depends at the micron level, on temperature changes, bearing wear etc. These random variations, unchecked, can cause a gradual change in output size. The operator checks the size after a number of machine cycles and adjusts the final feed position accordingly. This can be seen as a prediction problem in which an estimate is made for the correction that will minimise the error distribution until the next measurement correction.

2.6 Grinding wheel dressing (Loop 4)

Grinding wheels are manufactured by bonding together graded abrasive particles which can then be formed into required shapes or surfaces on the perifory of the wheel. The grinding wheel may be formed by using a diamond tool driven against the rotating wheel. Microscopically this breaks up abrasive particles and exposes fresh ones. Cutting edges around the periphery of the wheel finish up closely concentric. The grinding wheel can almost be seen as a very hard rotating milling tool with microscopically small teeth. The effect of the grinding operation changes the surface of the grinding wheel, eventually wearing down the sharp cutting edges or breaking them away. As the grinding wheel surface becomes inevitably worn so the components being ground start to show the adverse effects. The ground surface finish gets rougher and the finish size gets less consistent.

The possibility of burn damage may arise. So much heat energy is created at the point of grinding that it is no longer possible for it to be conducted away in time to prevent the temperature rising to a point where the structure of the metal is changed. The point at which the wheel surface needs to be restored is estimated by a skilled operator who decides the number of cycles between redressing the wheel. The effect of dressing is to produce a sudden change in the wheel surface which can be viewed as a discontinuous input disturbance to the wheel wear.

2.7 Grinding wheel change (Loop 3)

The repeated dressing of the grinding wheel reduces its diameter. The wheel must be changed when it falls below a minimum size determined by the geometry of the machine. The problem from the control point of view is that the replacement grinding wheel may have different properties to the previous wheel. The composition of the abrasive particles, their size and geometry, the bonding of the particles and their density, are some of the parameters which the grinding wheel manufacturers try to control to specifications. The variations within a given specification can significantly affect the grinding interface process, requiring changes to be made to the feed, the speeds and the wheel dressing parameters.

2.8 Wheel dresser change (Loop 2)

The diamond wheel dressing tool is subject to wear and tear. A replacement tool may exhibit different dressing properties which will affect the grinding interface process.

2.9 Machine maintenance (Loop 1)

The long term use of the machine eventually results in wear and tear on the bearings and drives. Unchecked, this can result in gradual loss of repeatability of all the component measurements (diameter, concentricity, surface finish). The possibility of breakdown, such as drive system failure or bearing seizure, exists. The maintenance may be implemented simply according to a program based on the machine use or number of components ground or it can be based on monitoring the wear and tear and performing maintenance operations when necessary.

2.10 Control objectives

The control objectives of the operator are basically to minimise the distribution in the component dimensions and surface finish within the tolerance range and to minimise the possibility of components being out of tolerance. Operator time is a high cost factor so that a balance must be found between the quality of the output and the operator attention required to maintain the level. An example of this would be the measurement of each bearing and a control correction each cycle. This would undoubtably reduce the output distribution but may cost more than the benefits. Machine cycle time has a lower cost factor so that control could result in increased cycle time being more than compensated for by a better output distribution.

The control objectives of the operator for this application have been explained. The objective of an appropriate control system could be basically similar. An additional objective sought by the sponsor company is improved setup. Concentration of production on low volume high precision markets makes it important that machines can be set up quickly with output components rapidly moving into the required tolerence ranges (with large scale production initial settling periods in which scrap may result are of less significance).

Based on the particular industrial needs a set of objectives may be stated as follows.

1) Monitor various measurements of the system. Control the system inputs such as feeds and speeds in order to reduce the dimemsion and surface finsh error distributions

2) Converge to tolerance conditions as fast as possible from setup.

3) As a lower priority minimise production time and conserve materials and power.

Chapter 3
System Modelling and simulation

3.1 Introduction

This chapter is concerned with the development of a mathematical model and computer simulation based on the application in chapter 2. The development of this kind of model is an important first step to the development of a control strategy and also provides good insight into appropriate instrumentation methods. The computer simulation is important in this kind of non- linear system because it facilitates the testing of the model. It is also a useful starting point for testing the effect of different control stategies.

The modelling includes some of the structural effects of the machine and also the dynamic reactions of the drive systems.

An important factor which is illustrated by the model is that the velocity variations of the two main drive shafts are characteristic of the forces at the grinding interface. This has important implications for the instrumention of the system and will be dealt with in chapter 4. At the end of this chapter the results of simulation tests are compared to experimental results from the real machine. The somewhat unexpected results from the experiments can be simulated and explained by referring to the mathematical models.

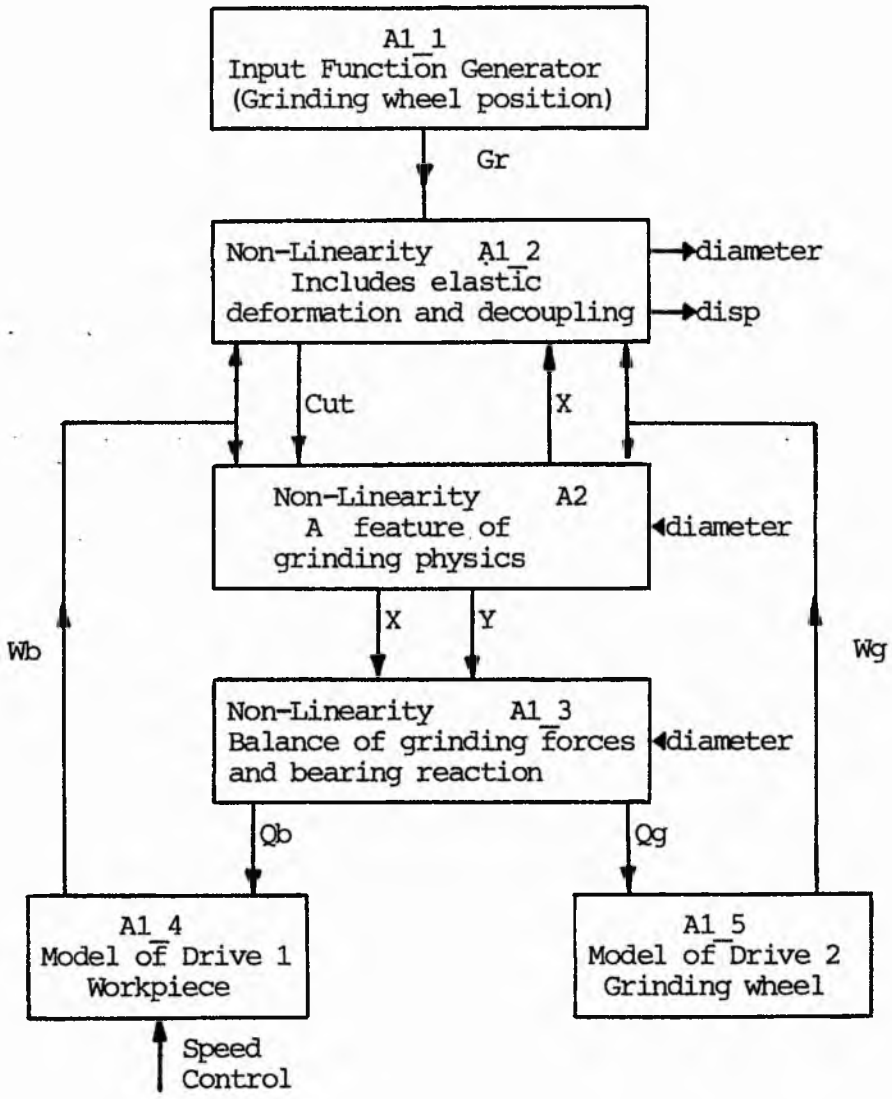
The idea of decomposition of the grinding process into functional blocks (figure 1.2 chapter 1)was introduced. This chapter explores the process in greater depth by further decomposition and the fitting of mathematical models to the parts. A model of the complete process is then created which, although complicated by non-linear and numerical components, provides the basis for a simulation or descrete time numerical model.

The simulation facilitates the testing of the model parts and provides an insight into the possible types of intrumentation that are appropriate. The simulation can also be used as a test bed for the development of computer control.

3.2 Process decomposition

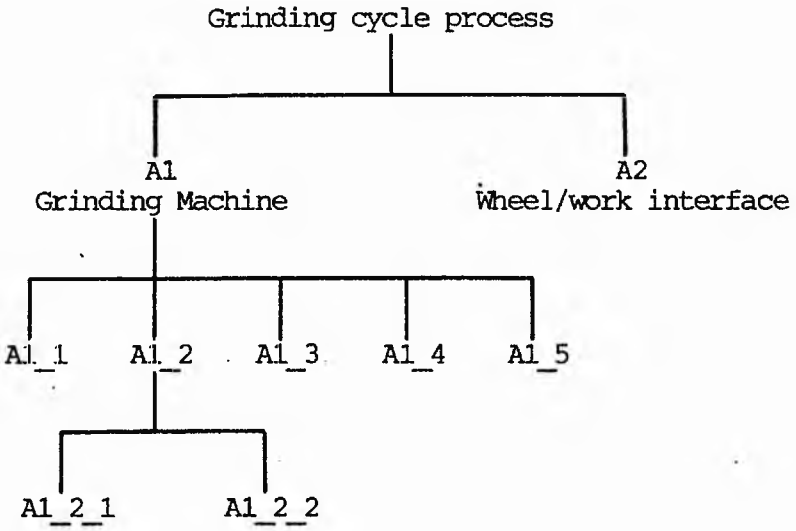
Figure 1.2 chapter 1 shows the first level of decomposition into two processes; A2 the grinding wheel/work interface process and A1 the grinding machine effects. Figure 3.1 shows further decomposition. The processes numbered A1_1 through to A1_4 representing the decomposition of A1. Figure 3.2 shows the decomposition of A1_2. The decomposition has a hierachical tree structure (Figure 3.2). The notation for the processes has been chosen to indicate the position of each process in the structure. Thus A1_2 represents the second child of the first child of the complete process. Process A1_2 represents the relation between the displacement of the component support structure and the force applied.

Figure 3.1 Grinding Machine Decomposition



Gr - Demand input for grinding wheel postion.
Cut - Depth of cut.
Wb - Angular velocity of the workpiece.
Wg - " " " grinding wheel.
X - Horizontal component of grinding force.
Y - Vertical " " "
Qb - Net disturbances torque on drive 1.
Qg - " " " " 2.
disp - Displacement of the structure.
diameter - Workpiece size.

Figure 3.2 Structure of Grinding process decomposition



A1_1 Grinding wheel infeed function generator.

A1_2 Model of rotating part with decoupling effect and elastic deformation of structure (further decomposed).

A1_3 Non-linear balance of bearing forces relation (App D).

A1_4 Work piece drive system.

A1_5 Grinding wheel drive system.

A1_2_1 Rotating part with decoupling.

A1_2_2 Machine structure deformation and damping effects.

3.3 Details of model components

3.3.1 Grinding machine (A1)

Process decomposed for simplification see Figure 3.1 and following components.

3.3.2 Function generator (A1_1)

The motion of the grinding wheel as it moves towards the work may be considered as being driven according to an input demand function. Figure 1.3a chapter 1 can be seen as such a function. With a computer controlled drive system this function is generated according to the control settings for feed rates and distances. The computer generates a digital position value at regular sample times and this is used as the demand input reference for an actuator (servo/stepper motor).

The function is entirely pre-determined and the only small complications are the setting up of the generator according to the input controls. In the case of the experimental machine the following manual control settings determine the output of this process.

1) Slow feedrate.
2) Position of slow feed start.
3) " end.
4) Spark out or dwell time.

The function generator does not have an input and its output is completely determined prior to the start of the grinding cycle. Appendix H includes the software design and implementation of the simulation. Function "A1_1" source "F1_1.C" represents above process. Initialisation of the above variables is taken care of by the setup function within "F1_1.C" called "SetA1_1".)

3.3.3 Process A1_2

This process represents the rotation of the component in the support shoes and the effects of the machine structure and damping. This process is further decomposed into "A1_2_1" and A1_2_2.

3.3.4 Numerical model of rotating component A1_2_1

The component outside dimensions are initially irregular and over size. The rotational motion of the component in centreless grinding is determined by support shoes mounted at ninety degrees to each other (see figure 2.1 chapter 2). The effect of the grinding wheel moving into the component can be seen as a numerical or incremental process (ref 25). The component may be represented as a circular buffer of dimensions (diameters) representing the distances measured across the component at equal angular displacements. The term diameter is used whilst bearing in mind that the unground workpieces are approximately round.

If the angular increment is chosen to be one degree then one hundred and eighty dimensions (taken over half a revolution) are needed to represent the shape. This representation of the component has been used to develop an algorithm to model the changing size of the component as it is being ground. Details of the algorithm are as follows.

3.3.5 Algorithm for model of rotation

At the sample time -

1) Find the angular position of the component in time.

2) Look up the nearest buffer diameter (see Figure 3.4).

3) Find the effective feed ("cut" for short). This is the difference between the position of the grinding wheel and the diameter in step 2 plus any displacement of the structure.

$$cut = Diameter + displacement - Gr$$

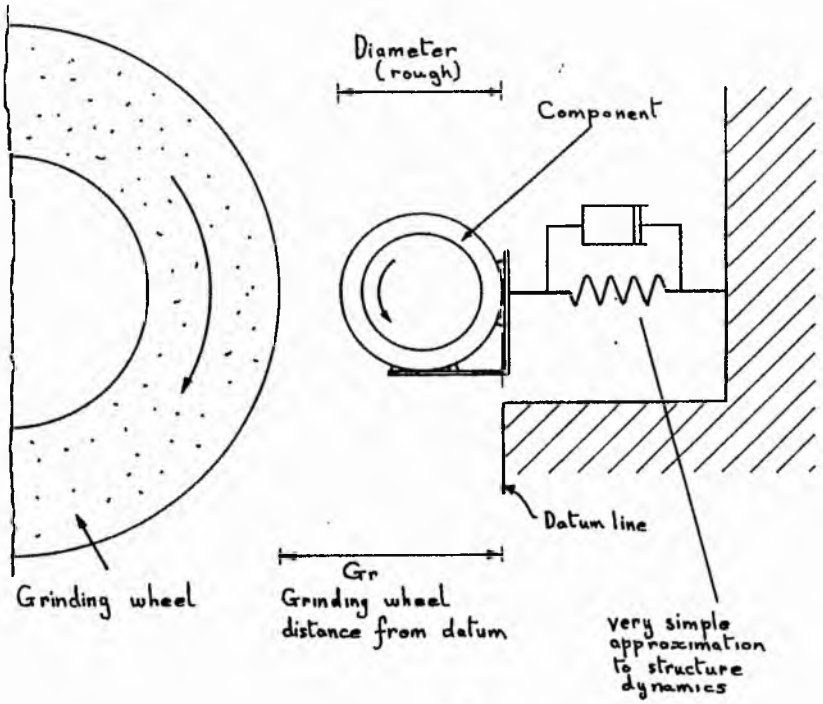
4) Reduce the diameter (referred to in step two) by an amount equal to the cut.

$$Diameter = Diameter - cut$$

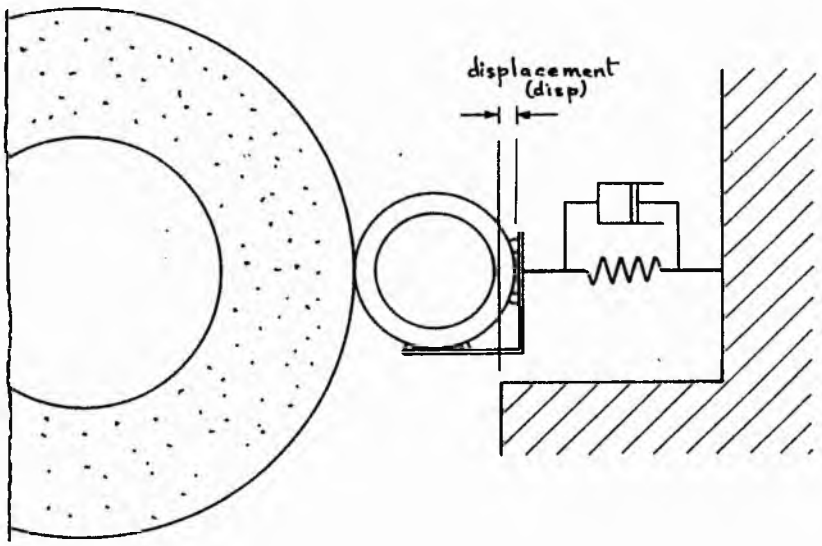
5) Interpolate buffer diameters between current position and last referenced buffer position.

Figure 3.3 Grinding wheel/work interaction
showing machine structure elastic and damping effects

(a) Grinding wheel not in contact with workpiece



(b) Grinding wheel in contact with workpiece



3.3.6 Structural deformation and damping (A1_2_2)

As the grinding wheel is driven into the workpiece those parts of the machine which bear a reaction to the wheel/work interface force deflect or distort. This involves many different parts of the machine bearings/structure which have elastic and damping effects. To represent these combined effects a second order differential equation has been chosen as an initial approximation to simplify the modelling (see Figure 3.3). A more comprehensive model may be more appropriate after a picture of the overall model of the grinding process is obtained.

3.3.7 Non-linear balance of grinding forces (A1_3)

Appendix C is devoted to this relationship which turns out to be quadratic. The relation between the applied torques on the grinding wheel and workpiece shafts and the horizontal component of the interface force is repeated here.

$$X^2 = (Qg + Qb)^2 Hb^2 - Qg^2$$

where

X = horizontal component of grinding interface force

Qg = net applied torque to grinding wheel shaft

Qb = " " " " workpiece "

Hb = workpiece support shoe friction parameter

3.3.8 Workpiece drive system (A1_4)

The design of this drive is similar to the grinding wheel arrangement (drive shaft belt driven by motor). In this case however a high performance "Anyspeed" controlled motor is used. The controller is claimed to maintain the speed within two per cent up or down. Papers have been written describing detailed identification of such devices (ref 56).

The least squares parameter estimation method of appendix G was used to obtain a reasonable sixth order model for this drive.

3.3.9 Grinding Wheel drive system (A1_5)

The heavy grinding wheel and its mounting collars are supported by a large rigid shaft (about 50 cm diameter wheel and 100 mm diameter shaft). The shaft is belt driven by a brushless three phase alternating current motor of approximately fifteen horse power. In the region of the steady state velocity (1500 rpm) this may be considered a linear system. If the flexibility of the belt coupling is taken into account there are two degrees of freedom and this an also be modelled as a linear system. If the belt slips on the drive wheels the system behave in a non linear manner. Appendix F describes a practical method for obtaining the parameters of the drive when it is decoupled from the workshaft. Moving average models have been used effectively and it would be interesting to identify the break down of such models as the system moved into a non linear state such as belt slip.

In the experiments carried out by Malkin (ref 9) this type of drive was used and the motor current was measured in order to estimate grinding power. The relation between the measured current and the grinding power was found by experiment (ref 9) to be approximately that of a second order system with dead time. The dead time could be as high as 0.5 sec. In this thesis the grinding wheel drive is considered as a control block with input as the applied torque and output as the angular velocity (a speed regulator responding to a disturbance input). Since the velocity changes are directly linked to applied torque there is no problem of dead time. The dynamics used for the simulation are those of a second order system. The descrete time model was obtained by using the bilinear transformation (see Appendix H simulation software).

3.3.10 The grinding wheel/work interface (A2)

This process has been modelled using different analytical methods (ref 9,12). Of particular interest here are the methods adopted by Malkin (ref 7). Malkin used appropriate modelling for the implementation of an adaptive controller based on grinding power feedback.

The following equation is the basic grinding power model adopted by Malkin.

Total grinding power per unit width [KW/mm]

$$P' = 13.8v_w a + 10^{-3}v_s + (C_1 + C_2 v_w/v_s d_e)d_e^{1/2}a^{1/2}A$$

where

v_w = Surface velocity of work piece.

v_s = " " grinding wheel.

a = Depth of cut of grinding wheel.

A = Fraction of grinding wheel surface consisting of wear flat area.

C_1, C_2 Constants dependant on the particular wheel work combination.

d_e Constant dependant on the diameters of the grinding wheel and work piece.

In Malkin's work the grinding power which is linked directly to the vertical component of the grinding wheel/work interaction force is considered an output variable from the process "A2" (grinding wheel work interface). The grinding power is the principal feedback variable for Malkin's adaptive controller figure 1.5. The models used by Malkin enable the effective feedrate of the grinding wheel, its angular velocity and that of the work piece to be linked to the vertical or y component of the grinding force.

The relation has been implemented in the simulation (appendix H). The model used to link the horizontal grinding interface force to the grinding feed or depth of cut ("cut" used as variable name) is a simple linear relation. The horizontal force being proportional to the depth of cut, the workpiece angular velocity and the grinding wheel wear flat area. This approximation is possibly a weak point in the simulation which could be improved by using a more refined model recently developed for this relationship (ref 21).

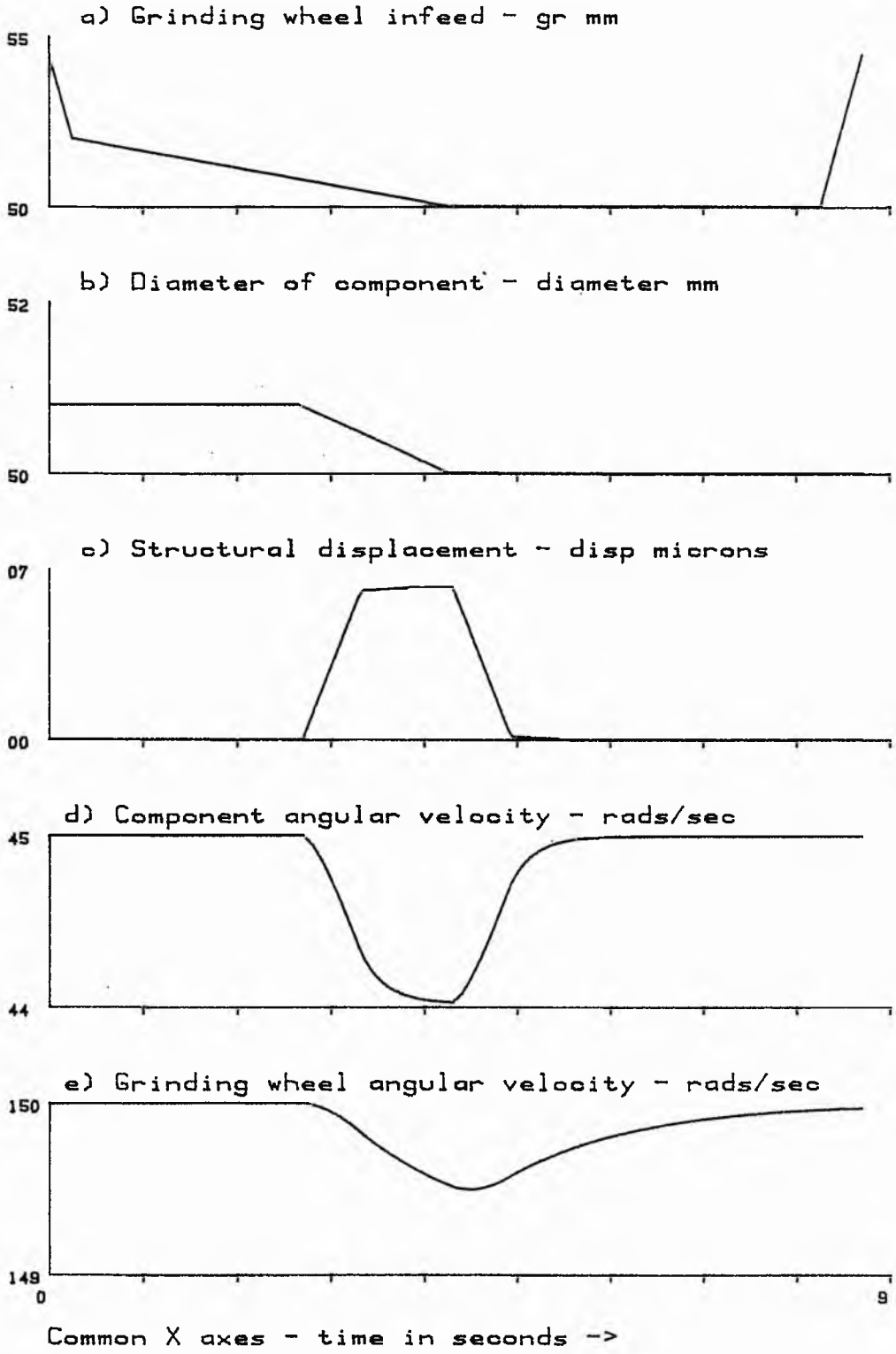
3.4 Computer simulation

The models (paragraphs 3.3 to 3.4) representing the component parts of the grinding process were used as the basis of a computer simulation (see appendix H for details of software design and implementation). Published papers also describe the use of computer simulations to represent this complex non-linear process. The example (ref 21) is the most comprehensive and detailed approach which has only recently been published. The simulation presented in this thesis provides a basic framework of model components which can be easily expanded or improved. The extension of the simulation to include vibrations by modification to the module "A1_2" is suggested.

3.5 Simulation results

The simulator results for various adjustments to machine parameters were recorded using a pen plotter (figures 3.4-3.8). Each figure shows five graphs on a common time axis. Graph "a" is the grinding wheel feed position. This is considered here to be the principal input function for the system. Graph "b" shows the changing diameter of the component. This is the cross sectional size of the component as it rotates. The component can only be round when graph "b" shows a line parallel to the time axis.

Fig 3.4 Grinding Simulation Graph



3.5.1 Figure 3.4

This set of graphs represents the grinding of a 50.7mm diameter completely round component down to 50mm with a cycle time of 9 seconds. Wheel work contact occurs about 2 seconds after the start of slow feed. In this case the structural damping is zero and graph "c" shows clearly the way grinding does not reach a steady state until one revolution after contact. Grinding continues for one complete revolution after the grinding wheel infeed has stopped. There is a small additional grinding time due to the relaxation efect of the structure. Graph "d" shows the response of the work piece drive shaft velocity. The braking force on the shaft exerted by the work piece support shoe friction is clearly greater than the driving force applied by the tangential or vertical component of the grinding force. The value for shoe friction was set greater than 0.2 which is possibly a high value.

3.5.2 Figure 3.5

These graphs are as 3.4 but with structural damping introduced and a higher performance work shaft speed regulator similar to the real machine. Graph "c" shows the effect of the damping on the structural displacement.

3.5.3 Figure 3.6

Work piece shoe friction is here reduced below 0.1 and the work piece or component shaft actually increases speed during grinding See graph "d".

3.5.4 Figure 3.7

This shows the effect of heavy structural damping. The structural displacement (graph "c") has not fully decayed away during sparkout time (graph "a")..

Fig 3.5 Grinding Simulation Graph

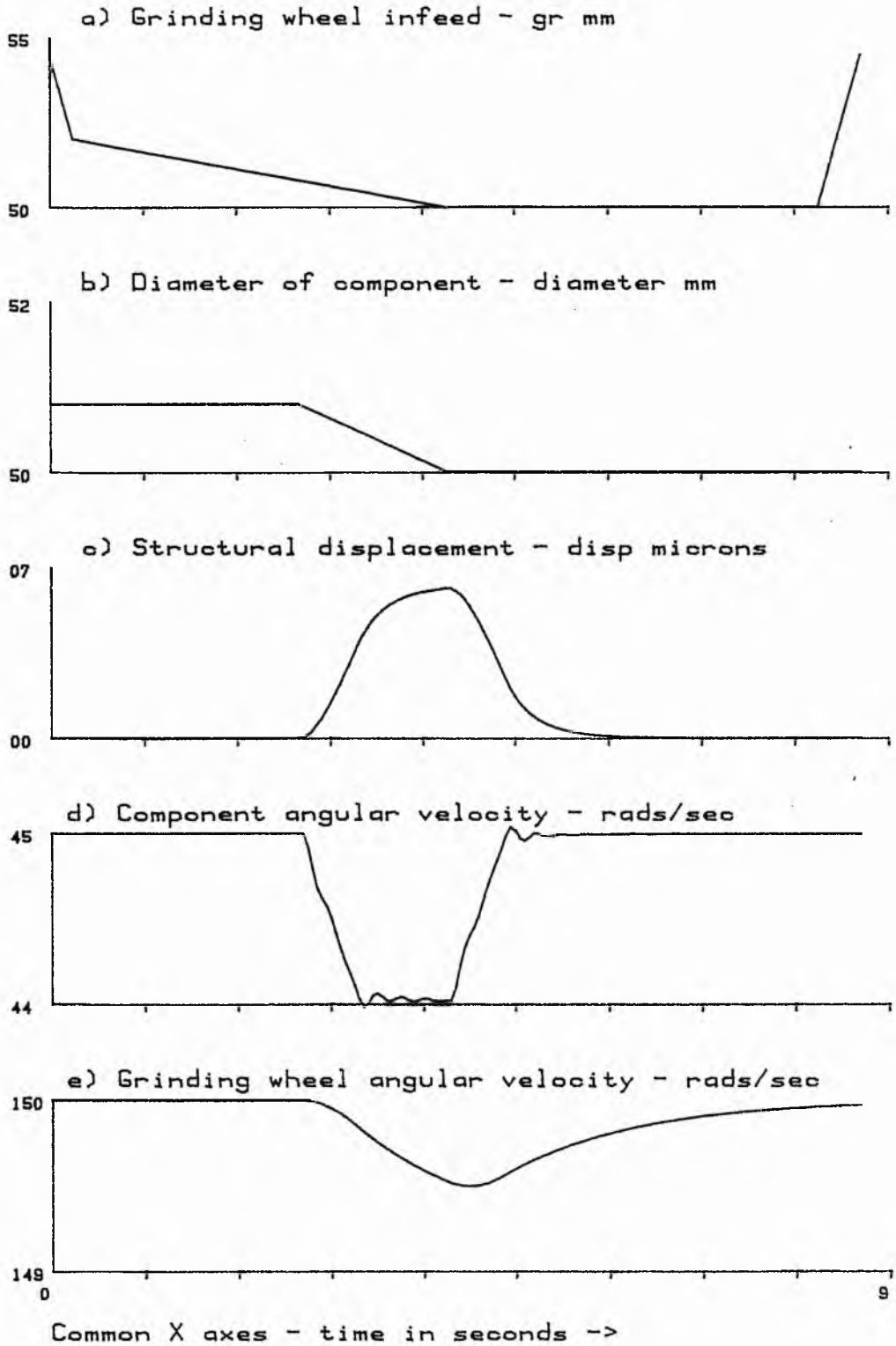


Fig 3.6 Grinding Simulation Graph

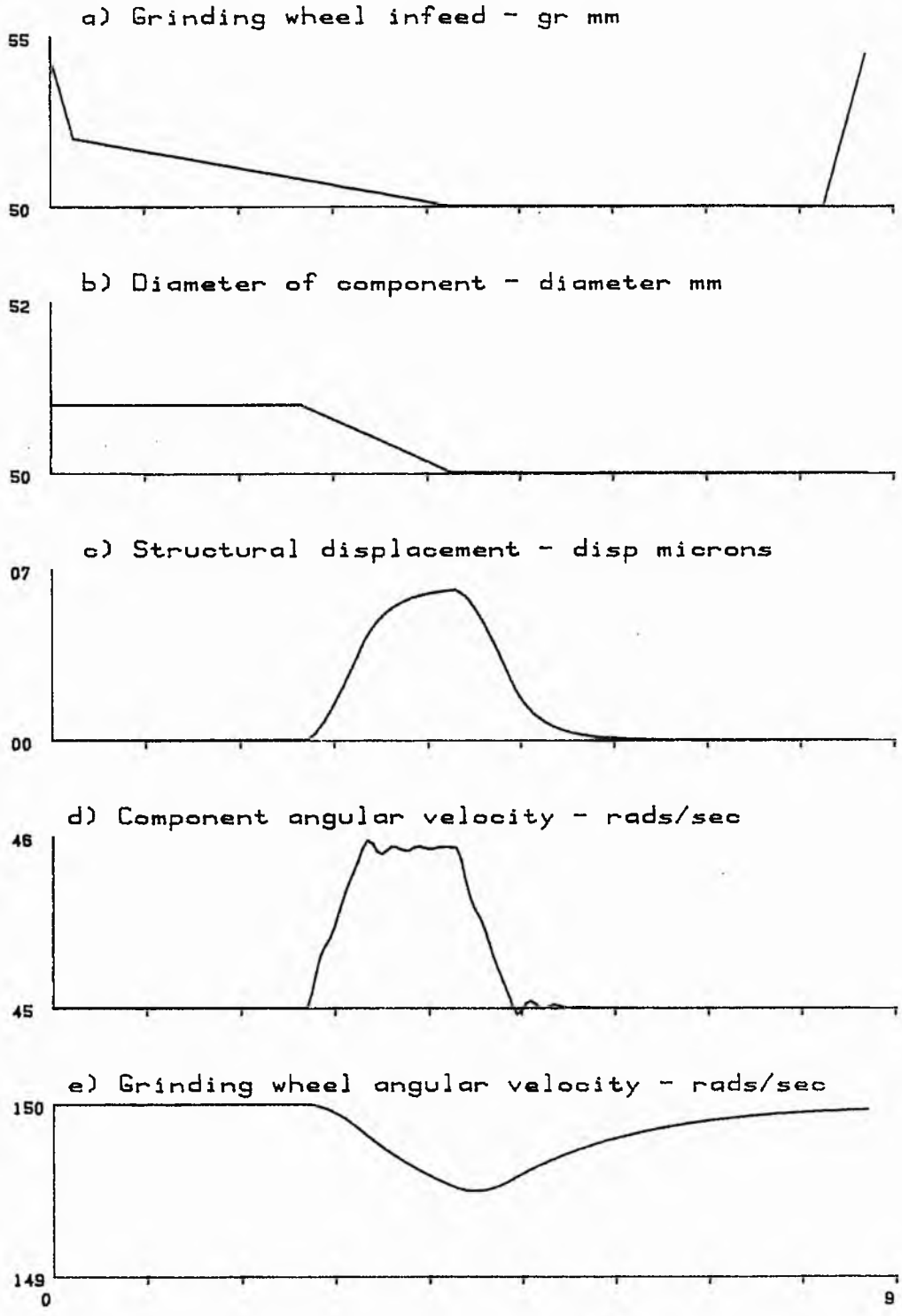


Fig 3.7 Grinding Simulation Graph

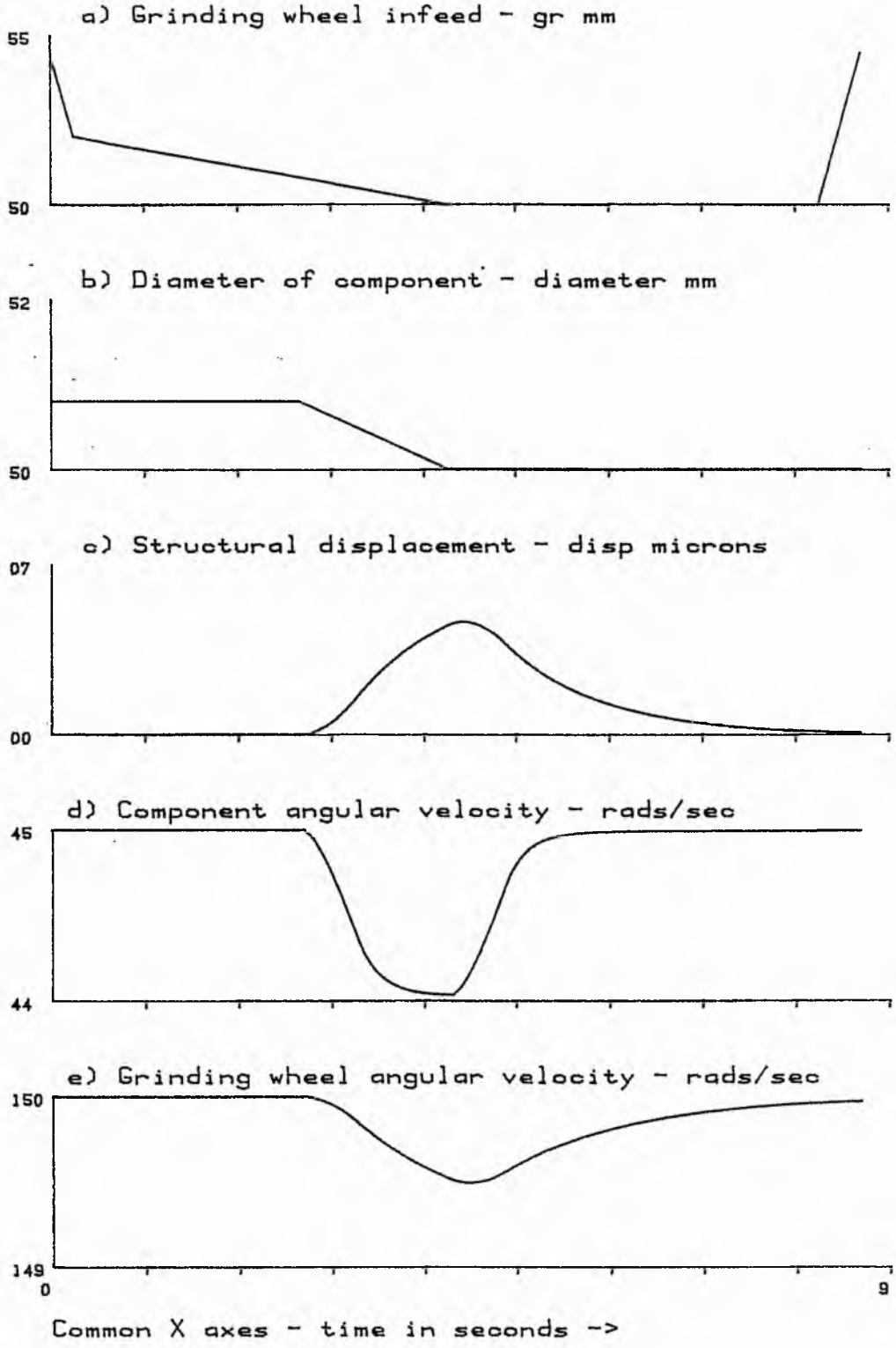
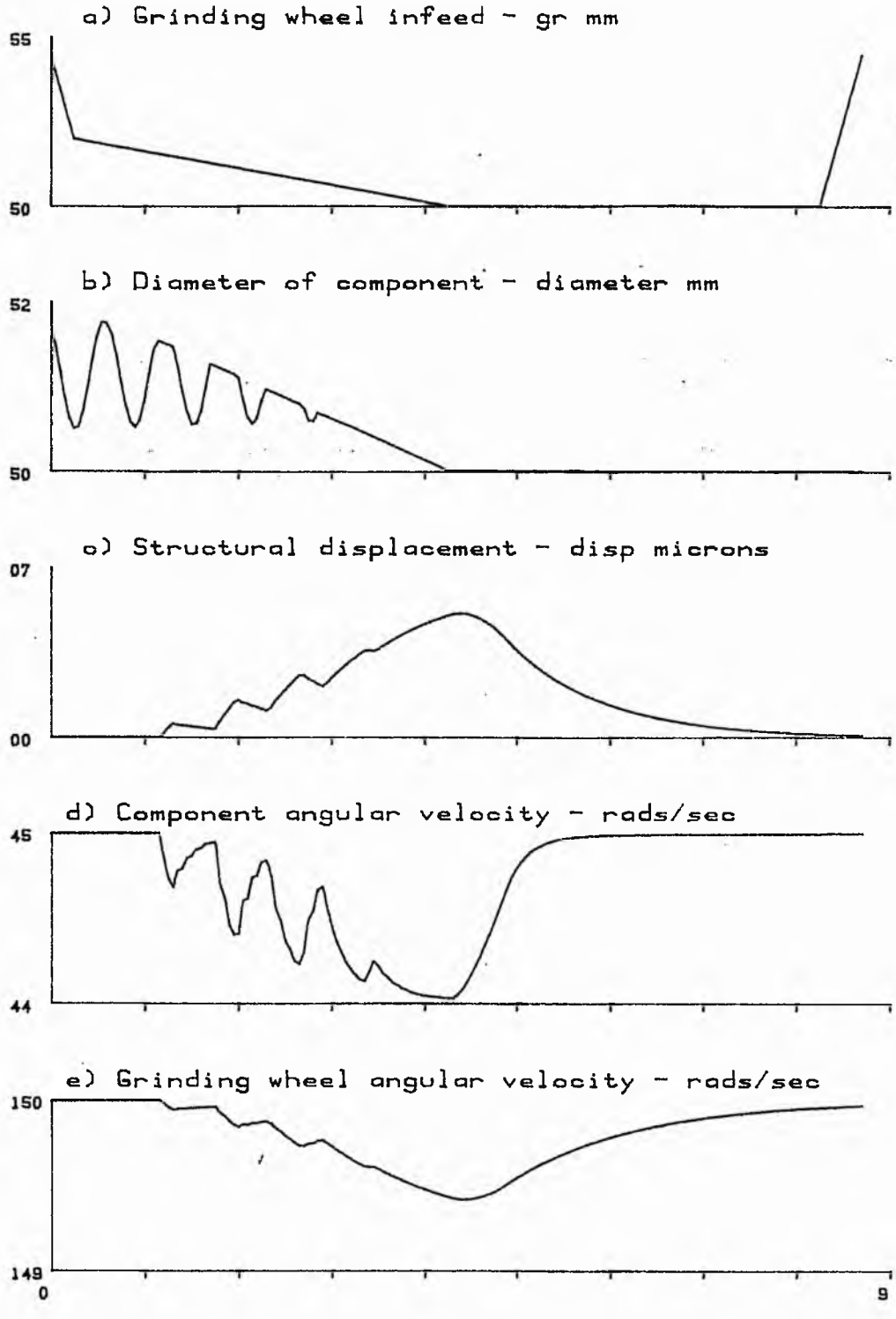


Fig 3.8 Grinding Simulation Graph



3.5.5 Figure 3.8

This shows how the simulator responds to an elliptical component model. The effect of an intermittent grinding action in the early stages produces a dramatic reaction on the two drive shaft velocities.

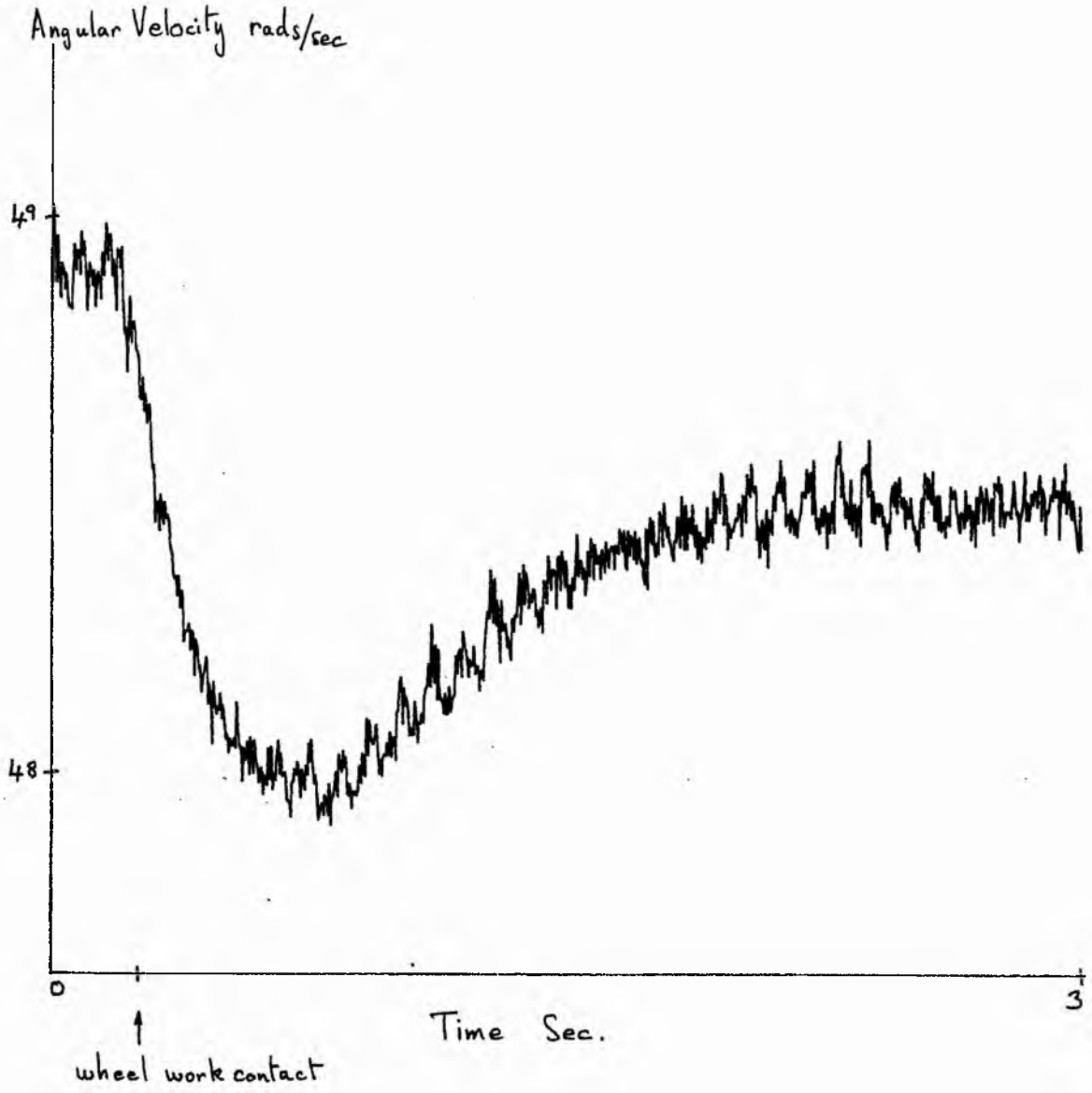
3.6 Comparison of simulation results and experimental results

In order to collect experimental data from a production grinding machine a special purpose data logger was designed and constructed (see appendix E). This device was able to log timing data from a precision optical shaft encoder, which was mounted on the component drive shaft, and several relay states indicating feed rates etc (further work could enable this device to log other measurements such as the grinding wheel timing data and other transducer outputs). The data collected gives a precision picture of the reaction of the component drive shaft velocity in the period just before and after the grinding wheel touches the component. An example is shown in Fig 3.9.

Initially it was anticipated that as the grinding wheel touched the component it would exert a net tangential driving force in the direction of rotation of the component. This would have resulted in a sudden increase in the velocity of the drive shaft. In fact the results Fig 3.9 show exactly the opposite effect. The actual results can be explained by referring to the model which takes into effect the component support shoe frictional braking force. As shown in Fig 3.4 & 3.5 (d) the braking effect produced as the grinding wheel pressses the component into the shoes can be greater than the driving effect of the grinding wheel.

Fig 3.9 Velocity of workpiece shaft

Derived from precision optical shaft encoder timing data (Appendix E)



By choosing appropriate drive system transfer function parameters the simulation component drive shaft reaction corresponds to the experimental results in terms of general shape, drop in velocity and over shoot. This gives some credibility to the model.
Note - It is possible however that a more detailed model of grinding interface region taking into account the actual contact surface of the grinding wheel and work may result in a resolution of grinding force that could itself have a braking action (ref 12).

3.7 Summary

A model of the grinding process for a complete grinding cycle has been established from a collection of linear and non-linear sub systems. This work begins to prepare the way for more detailed modelling in such areas as the wheel/work interface process or the consideration of vibrations.

A computer simulation was used to test the modelling. The structure of the simulator software allows sub processes to be expanded or added easily without disturbing the rest of the software (appendix H).

The simulator provides the basis of a software test bed for feed back control strategies or parameter estimation methods.

The model and simulator behaviour have been compared to the real system in the case of the drive shaft velocities. Effects similar to the experimental results can be reproduced by simulation. Examination of the model provides a plausible explanation of the experimental results

Chapter 4

Measurement and estimation of process parameters

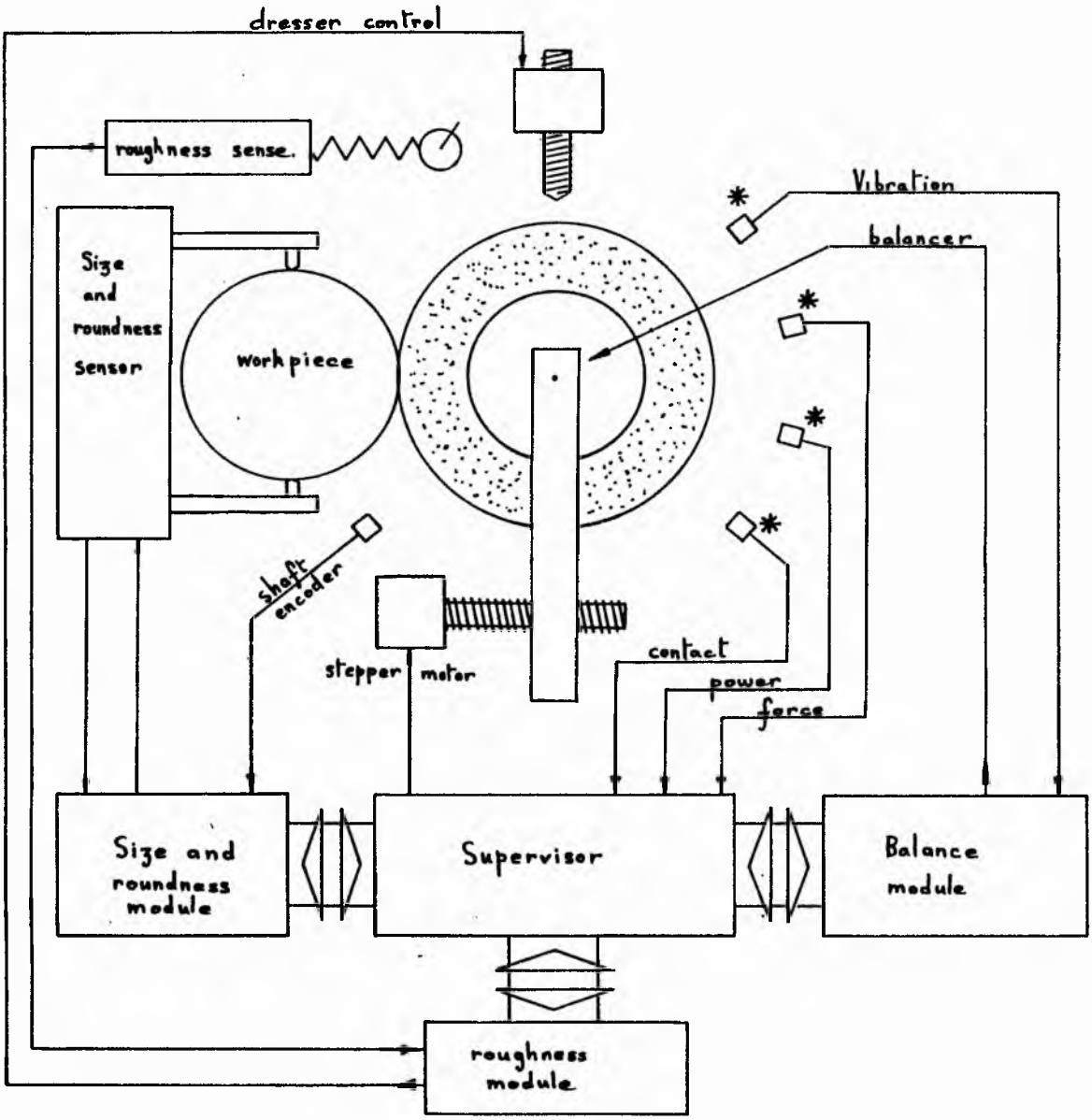
4.1 Introduction

The successful development of a comprehensive control system to improve on the performance of the existing open loop manually controlled grinding machine, will depend partly on the ability to measure and estimate critical process variables (ref 58). The accuracy of measurements and the reliability (ref 61) of the instrumentation are also important factors in the design of a controller. The problems of obtaining a measurement normally incurs some cost, either directly, in terms of transducer price, or more indirectly in maintenance. Periodic calibration may be a costly factor. In some cases the transducer may introduce a form of undesirable change to the system (invasive) which may itself adversely affect the operation. Some types of instrumentation may affect accessiblity or obstruct the normal running of the machine. The more instrumentation or transducers that are used the more it becomes a problem to maintain and calibrate. The choice of appropriate instrumentation therefore requires careful consideration.

Research has been carried out into many possible approaches to monitoring the system (ref 15,16,18,30,31). The research carried out at Birmingham University by Kaliszer uses seven separate transducers based on varied technologies. Fig 4.1 illustrates the layout of the experimental rig. Other researchers typically specialise in the analysis of a single transducer of interest, such as accelerometers for vibration analysis (which alone may amount to a sizable and complex subject (ref 13)). The following section summarises the application of transducers to the grinding problem.

Fig 4.1 Grinding machine instrumentation

Kalizer adaptive control reference 3



Information obtained from sensors marked * may be estimated by processing the shaft encoder signal. An incremental motion encoder could also measure size/roundness. This is believed to be an original idea in this thesis (see Appendix F).

4.2 Transducers

1) Estimation of grinding power by the measurement of grinding wheel motor current using a Hall effect current transducer. Malkin used the grinding power to estimate many important grinding interface parameters including the start of wheel work contact (ref 9)

2) Accelerometers used to measure vibration of the component being ground (ref 13). Mounted under the work support shoes these devices can be used to detect adverse vibration conditions but are somewhat invasive and create obstructions in the production environment.

3) Acoustic pickups (microphones) for vibration/work contact analysis and workpiece burn conditions. This approach is non-invasive but limited to information at or above audio frequency only. Commercially available devices to detect work contact are available and much work has been carried out to estimate grinding interface conditions from the signal (ref 18,30,31).

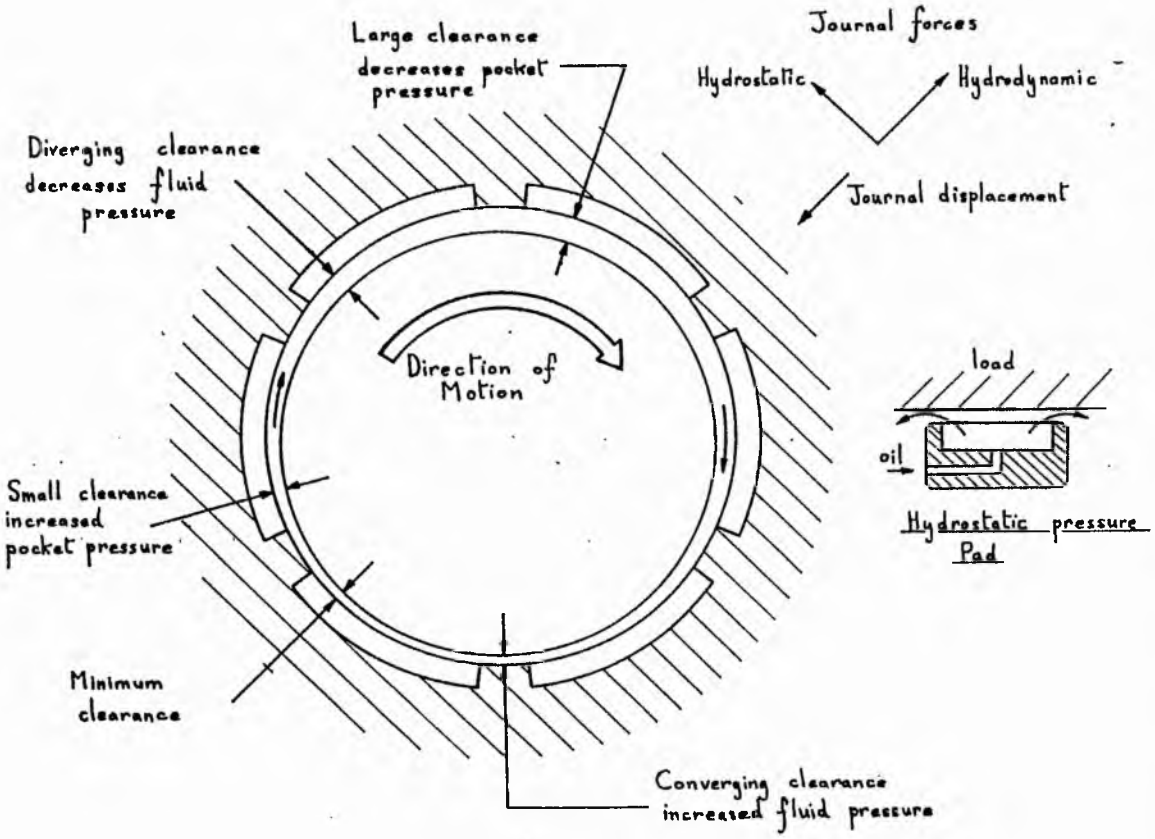
4) In process and off line precision size measurement equipment. The research at Birmingham developed inprocess gauging and roundness sensing Fig 4.1 using differential capacitance.(ref 3,8,27,)

5) Strain gauges/load cell to measure the resolution of grinding forces. These devices are considered by many to be insufficiently robust for the production environment (ref 11).

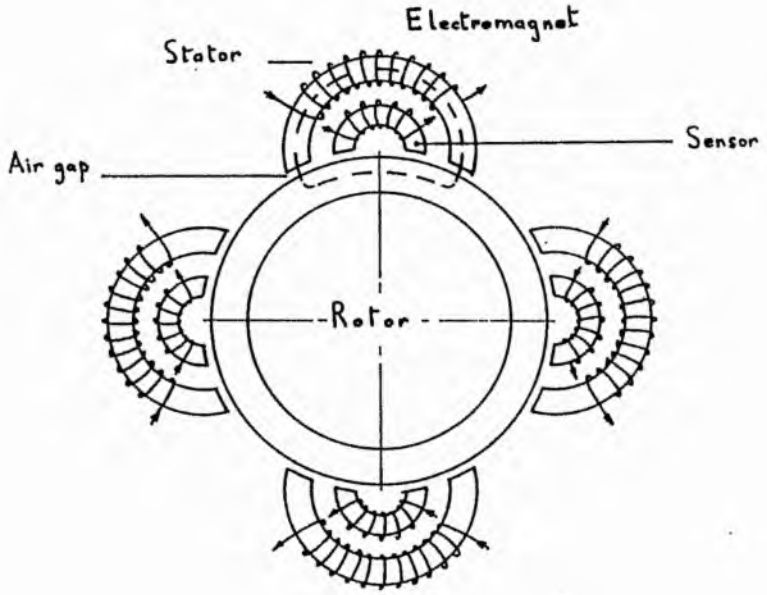
6) Hydrostatic or magnetic automatically controlled bearings Fig 4.2. Fluid pressure or magnetic sensors indicate resolution of grinding force. Hydrostatic bearings were used on Warwick University's adaptive control cylindrical grinding machine for the SERC grinding program. Also see refs 10,26)

Fig 4.2 Automatically controlled bearings

(a) Oil hydrostatic



(b) Active magnetic bearing



7) Infra-red detectors and other optical methods used to measure surface finish by comparision method (ref 29,46). Also used to measure grinding wheel surface temperature (ref 33).

8) Thermocouple to monitor temperatures at various locations on the machine (temperatures linked to movements in the machine structure) (ref 39).

9) Optical shaft encoders for the direct measurement of shaft movement and the estimation of many other parameters (see section 4.5).

10) Power spectrum analysis using laser beams to assess grinding wheel wear (ref 15).

11) Air turbulence effects measured by using a turbulence amplifier close to the grinding wheel to estimate wheel wear (ref 16)

Individually each of the above forms of instrumentation may possess particular merits and collectively would give a most accurate picture of the system variables. It is not practical however to use all the available monitoring techniques. The cost would be excessive and the machine might be instrumentationally cumbersome and impractical from a maintenance point of view.
Fig 4.1 shows the layout of the experimental adaptive control grinding system used by Kaliszer at Birmingham University. There are seven seperate transducers. With this type of system it could be possible that rationalisation of its instrumentation requirements could reduce the number of transducers but maintain the amount of information available.

4.3 Choice of measurements

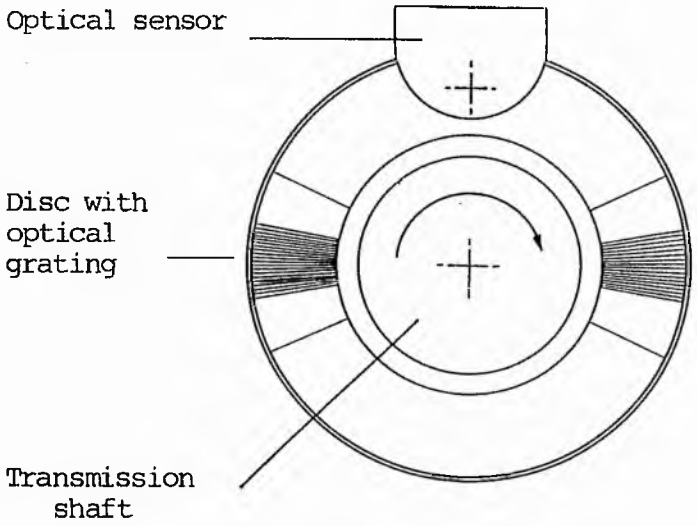
Certain system variables may be considered of key importance in that other useful variables may be estimated from them. This will reduce physical complexity and cost.

Consideration of recent research papers (ref 9 and section 4.5 chapter 4) indicates that a most useful measurement would also be the grinding power (the power consumed at the wheel work interface). Knowledge of the grinding power facilitates the estimation of other features, such as wheel wear, work burning and wheel breakdown. A different line of research (ref 11) places importance on the horizontal or normal force of grinding wheel/work interaction. The vertical and horizontal (normal and tangential) interaction forces are therefore of great interest. In short, we can say that it would be useful if the interaction force vector could be monitored. Ideally a comprehensive control system would be able to analyse machine vibrations, possibly taking steps to reduce the effects. This requires some form of vibration sensing. The component size and its surface finish are clearly of importance and should be established by direct measurement at some stage in the process either on or off line. The following list of key measurements is therefore postulated.

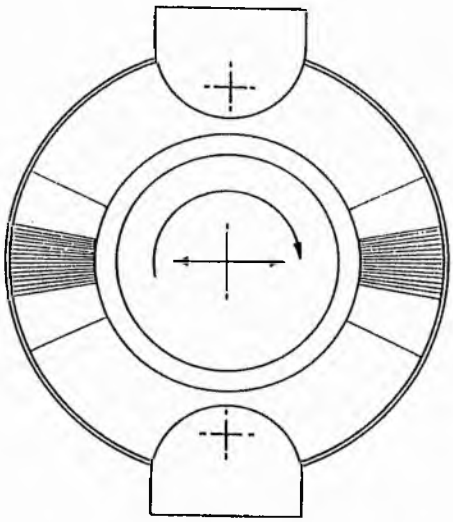
4.4 List of primary measurements

a) Grinding wheel work interaction force vector
b) Bearing size
c) Surface finish
d) Vibration detection

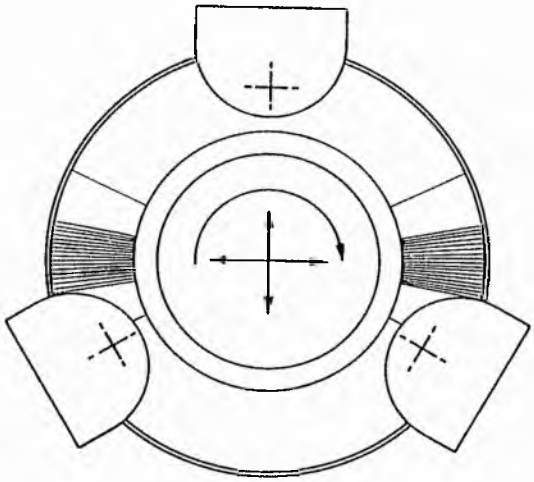
Fig 4. Incremental motion encoder and shaft encoder



Optical incremental encoder with single read head for measurement of angular position



Additional read head enables estimation of small horizontal displacements of the shaft centre



Three or four read heads enable small two dimensional displacements of the shaft centre to be estimated

It is a novel feature of this research however that it is considered that a) and d) can be estimated using a single type of precision digital transducer (the optical incremental shaft encoder) enabling a degree of rationalisation of the instrumentation requirements. Appendix F introduces a development of the shaft encoder called the incremental motion encoder Fig 4.3. This instrument can be used to detect angular position and small two dimensional motions of the shaft in a plain perpendicular to the shaft. This instrument might well be used to gauge size and surface finish (b & d). Further development of this instrument is required however.

4.5 Optical incremental shaft encoder

Optical incremental shaft encoders primarily measure changes in angular position. The output from the shaft encoder can be used to generate a square wave logic level voltage in response to changes in angular position. The absolute change in angular position is proportional to the sum of the pulses generated (there is quantisation error equal to the angle of one encoder increment, see appendix A). Resolution and accuracy can be very high, with 23 bit resolution encoders being commercially available (ref 51).

In the dynamic situation, position must be related to time.
The combination of an encoder plus a precision clock reference can provide high resolution digital data for the purpose of velocity estimation (appendix A). The bandwidth of the velocity estimate may extend from zero to beyond twenty kilohertz. The ability to monitor position and velocity data of this quality makes it possible to consider the estimation of other process features.

4.6 Information from shaft encoders

The following is a list of variables that can be produced directly (1,2,3) or indirectly (4,5,6) using data generated by optical shaft encoders.

1) Position

2) Velocity and derivatives

3) Vibration details from high frequency velocity data. Shaft imbalance

4) Parameter estimation of the drives

5) Drive characteristics monitoring

6) Resolution of grinding forces

7) Wheel work contact detection

A more detailed explanation of the above list follows.

4.7 Position

At first sight this may seem an irrelevant factor, however it is shown (Appendix A) that for high frequency velocity/vibration analysis angular position of the shaft is most important in analysing cyclic (period of complete rotation) disturbances and noise. Since the major machine vibrations are mostly linked to the period of rotation it is most convenient to investigate vibrations by studying the time measured at equal angular increments in position (Appendix A,F).

4.8 Velocity

A precision measurement of velocity (appendix A) would not normally be considered useful for a grinding machine (although the actual velocity in many cases is controlled carefully and set manually to avoid resonances). The approach of this research is that accurate velocity estimates enable the estimation of the dynamics of the drive systems and subsequently the applied loads. This is considered to be a viable alternative to other forms of load measurement such as load cells.

4.9 Vibrations

The incremental motion encoder (Fig 4.3 and Appendix F) can be used for vibration analysis and there is scope for further research to find how far it can practically be exploited in terms of frequency and resolution (Appendix A,F). Compared to using other devices such as accelerometers, strain gauges or acoustic sensors the following observations can be made.

a) Digital non-contact device giving stable and reliable data over long periods with practically no maintainance.

b) Data intrinsically linked to rotational motion enabling straight forward calibration and analysis of rotating part imbalance effects.

c) Can be built into bearings.

d) Requires care in mounting to ensure that relevent vibrations exist in sufficient magnitude between encoder disc and sensors.

e) Can be used for a comprehensive analysis of three dimensional shaft vibrations (Appendix F).

f) Can be calibrated periodically on line (Appendix F).

g) Can be used to detect vibrations over a broad bandwidth extending from practically zero to 20 kilohertz and potentially far higher (Appendix B).

4.10 Parameter estimation for drive systems

Appendix G explains a practical method for obtaining the drive models, on line, based on least squares parameter estimation for a sixth order model approximation. This parameter estimation of the two drive systems takes advantage of the period in the grinding cycle in which the drives are completely decoupled and can be carried out frequently and non-intrusively. The model equation can then be used to estimate values for applied torque disturbances on the shafts given the angular velocities.

4.11 Drive system characteristics monitoring

Analysis of the descrete drive models, with known parameters, enables the estimation of important drive system characteristics such as motor/speed control performance and drive belt tension. Further information can be added by also monitoring motor input power, enabling motor efficiency and possible energy losses in bearings etc to be studied. The drive model parameters can be monitored efficiently by applying recursive least squares method (see appendix G).

4.12 Estimation of grinding forces and some system parameters

The incremental motion encoder can theoretically be used to measure grinding forces. Integrated into the machine bearings it may be possible to detect the deflection due to shaft loading and estimate the load vector. The incremental motion encoder could be integrated into controlled bearing Fig 4.2 and become part of a closed loop system. Further work would be needed to establish the practicality of the approach.

Alternatively the grinding forces may be estimated from the applied disturbance torques on the two shafts.

The object of the rest of this section is to explain how it will be possible to use the data collected from shaft encoders to obtain estimates, firstly, of certain system parameters then the disturbance torques and finally the required grinding forces. It will be shown that not only can estimates be made for the vertical and the horizontal components of the grinding force but that also the friction on the bearing shoes can be estimated along with other important drive system characteristcs. Refer to Fig 4.4 for a block diagram of the technique.

The procedure is based on the idea of first being able to establish the total applied torque on each drive shaft due to the grinding wheel/work interface process. It is possible to find the torques from the velocity data, provided that reliable models of the drive sytems are known.

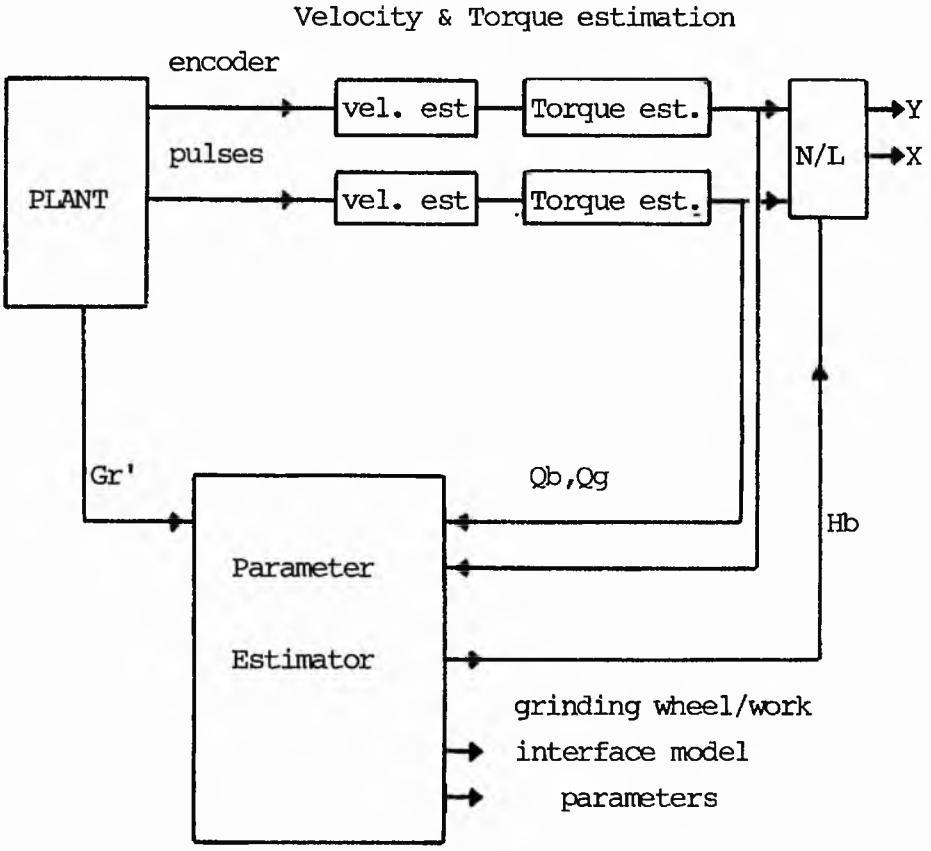
shaft velocity -----> Drive model -----> applied torque

This is explained in section 4.8

It should be noted that velocity changes can be typically at most only a few percent of the steady state values and that precision velocity data of the type provided by the shaft encoder is necessary if small changes in applied torque are to be measured.

Having obtained the shaft torque estimates, the force balance relation established in Appendix D can be applied.

Fig 4.4 Estimation of grinding force and parameters



N/L non-linear model of balance of grinding forces
Y vertical component of grinding force
X normal " " "
Qb,Qg disturbance torque factors on drive shafts
Hb shoe friction factor
Gr' grinding wheel infeed rate

The simplified form of this force relation i.e. equation 6 appendix D (low grinding wheel bearing friction) is repeated here.

$$X^2 = (Qg + Qb)^2 Hb^2 - Qg^2 \quad \text{---------(4.1)}$$

$$Y = Q_g$$

where

X = Horizontal component of grinding force.

Y = Vertical " " "

Qb = Net torque applied to work piece shaft divided by component radius.

Qg = Net torque applied to grinding wheel shaft divided by wheel radius.

Hb = Workpiece/support shoe friction factor.

Thus the vertical component of the grinding force "Y" can be found directly from the grinding wheel torque. Provided the frictional factor Hb is known then the horizontal force "X" can be found also. The friction factor is a function of the coefficient of dynamic friction and it is believed to be reasonably constant over the restricted operating conditions of almost constant speed and with pre -loading due to the nature of the eccentric workpiece drive arrangement.

4.13 Estimation of the work support shoe friction

If values for the horizontal component of the grinding force "X" and the drive torques were known then equation (4.1) could be applied to find a value for the shoe friction. An alternative method of estimating the friction without knowing the "X" force has been developed. This technique also involves the estimation of some of the parameters of the grinding interface model "A2" chapter 3. To explain this method it is necessary to refer to the models of chapter 3.

The starting point is to consider the model "A2" (wheel/work interface). The horizontal grinding force "X" is a complex non linear function of "cut" (the depth of grinding wheel cut). The system is characterised by periods of constant "cut" between starting and ending transients. In a constant "cut" interval the "cut" can simply be related to the grinding wheel feed "Gr" as follows -

$$cut = \frac{Gr'}{Wb}$$

Where Wb is the angular velocity of the workpiece drive.

If "Wb" is constant then -

$$X = f(Gr', A) \text{ ----------(4.2)}$$

where A is the grinding wheel wear factor

The horizontal grinding force is a non linear function of the grinding wheel infeed velocity and the wheel wear factor "A". If we approximate this function to a linear relation (over a limited region) and take "A" to be constant over a short number of grinding cycles then

$$X = SrGr'$$

Where Sr is referred to as the stock removal factor. Hahn (ref 19) uses this factor as a measure of grinding wheel wear. Substituting this relation into equation (1) -

$$(SrGr')^2 = (Qg + Qb)^2 Hb^2 - Qg^2 \quad \text{----(4.3)}$$

The machine feeds and speeds can easily be setup to provide two sets of results with "Gr'" at different values (this can be done over two machine cycles starting with two different settings for "Gr'"). Two independant equations like equation (4.3) may then be written with unknowns Sr and Hb. It follows that Sr and Hb may be evaluated. In practise it will probably be more appropriate to create many more values of "Gr'" and use a least square estimation (see Appendix G) to obtain accurate results. The approximation of equation (4.2) to a linear function can be avoided. If a polynomial is used then all the above estimation may still be carried out with a little modification for additional stock removal rate parameters. If a quadratic approximation is used then equation (4.3) can be written

$$(Sr_1Gr' + Sr_2Gr'^2)^2 = (Qg + Qb)^2 Hb^2 - Qg^2 \quad \text{----(4.4)}$$

Where Sr_1 and Sr_2 are quadratic stock removal factors. The least squares method can again be used yielding Sr_1,Sr_2 and Hb.

The availability of the grinding force vector and a set of values for "Gr'" in the steady state situation makes it feasible to obtain some of the grinding interface model parameters and if necessary monitor for any changes on line.

4.14 Grinding wheel work contact detection

The precision velocity signal obtained from a shaft encoder can be used to detect the sudden contact of the grinding wheel and the work piece. A method for the estimation of the drive shaft applied torques has been explained in section 4.8. The applied disturbance torques for both shafts can therefore be monitored. The estimate of contact time and position can be based on the detection of an increase in grinding wheel applied torque beyond the background noise signal level. The accuracy of the estimate can be enhanced by a combined examination of both shaft disturbance torques but this is a detail point not pursued further in this thesis.

Grinding wheel and workpiece contact detection information can be used to control the grinding wheel infeed velocity (allowing a fast approach speed and changing to a slow feed automatically on detection of contact). Examination of experimental data shows that contact can be detected in less than one hundredth of a second. This compares favourably with the method of using motor current measurement which involves a significant dead time response (ref 9).

The grinding wheel infeed position is a known variable (an input). It is thus possible to estimate the initial component size if the contact instant is found. Contact detection is possible with other sensors than the shaft encoder but possibly less accurately.

4.15 Estimation of grinding wheel wear

Malkin used the following power relationship (ref 1) which links total grinding power to grinding wheel wear.

Total grinding power per unit width [KW/mm]

$$P' = 13.8v_w a + 10^{-3}v_s + (C_1 + C_2 v_w/v_s d_e) d_e^{1/2} a^{1/2} A \quad \text{----(4.5)}$$

where

v_w = Surface velocity of work piece.

v_s = " " grinding wheel.

a = Depth of cut of grinding wheel ("cut" is also used).

A = Fraction of grinding wheel surface consisting of wear flat area.

C_1, C_2 Constants dependant on the particular wheel work combination (grinding wheel composition and workpiece material and hardness).

d_e Constant dependant on the diameters of the grinding wheel and work piece.

The constants "C_1" and "C_2" can be found experimentally (ref 1).

The grinding power "P'" can be found if "Y," the vertical component of the grinding force is known. The variables "v_s","v_w", and "a" are known control outputs and can be adjusted. This means that a least squares parameter estimation method can be applied to equation (4.5) in order to obtain the constant factors C_1 and C_2 and the wheel wear A (this can be done assuming constant wheel wear over a very limited period). In practise it is only possible to obtain a value for the ratio of the two constants "C_1/C_2" using parameter estimation. This means however, that only one constant needs to be found experimentally and this will appear as a linear scaling factor on the estimate of the wheel wear.

Having found a value for "C_1/C_2" equation (4.5) will yield actual values for "C_1A" or "C_2A" which even if neither constant is available provide a useful proportional measure of wheel wear.

Alternatively a recursive form of parameter estimation can be applied to equation (4.5) to estimate "C_1A" and "C_2A".

4.16 Estimation of workpiece burning boundary condition

The modelling of thermal effects at the grinding wheel and work-piece interface (ref 6) relates the feeds and speeds (and other parameters) to the burning point threshold power value. This value of the grinding power is claimed to be the maximum allowable grinding power before workpiece burn occurs. This model can be used together with the model for total grinding power (equation 4.5) to estimate the locus of the burning point boundary in the plane of grinding wheel infeed and work piece velocity. This boundary will change for different values of wheel wear. Thus given the wheel wear factor "A" the burning point boundary in the "v_w" vs "a" plane can be estimated. This is also considered in chapter 5.

4.17 Estimation of structural deflection and damping

The grinding machine structural deflection occurs in response to the grinding force vector. It is an important factor which is accounted for normally by allowing a fixed dwell or sparkout interval. At the end of the grinding wheel infeed the dwell is simply a delay of the order of seconds to allow the structural displacement to decay away as the interface force drops rapidly. This has been considered in Chapter 3 where model "A1_2_2" approximately describes the relaton between deflection and grinding force.

If the model constants can be found (spring constant and damping factor) reduction of the dwell time is possible and this is discussed in chapter 5.

In order to obtain the model parameters the deflection needs to be measured and this can be done by attaching a linear digital transducer between the two drive shaft journals. The problem is that at least part of the structure will have to be avoided (bearing movement for instance).

Alternatively inprocess work-piece gauging equipment can be used. The difference between the actual size and the demand size is then equal to the displacement (this is complicated by the fact that there exists a measurement lag as the instrumentation must be displaced angularly from the grinding wheel). Having access to the deflection and the interaction force enables the evaluation of the model parameters (see Appendix G).

4.18 Summary - measurements and estimation

Various approaches to intrumentation have been considered with a view to the estimation of important process variables. The idea of using optical shaft encoders as an alternative to a collection of other transducers has been introduced. The design of a new instrument called an "Incremental motion encoder" has been introduced. This instrument given further development may facilitate comprehensive vibration analysis and may even be used as part of a gauging instrument.

The method of using regular on -line parameter estimation of the drive systems to monitor drive characteristics and enable the estimation of shaft loading has been suggested. The possibilities for the estimation of non linear model parameters using least squares method have been explored.

Chapter 5
Control Strategy

5.1 Introduction

In chapter 1 it was indicated that much research had been carried out into the control of different aspects of the grinding process. Generally the approach has been to use control methods to effect a solution to an often narrowly viewed problem area. An extreme example of this is perhaps the use of an acoustic sensor being used in a simple infeed control loop to detect wheel/work contact. There are a few examples of a more comprehensive approach to control such as (ref 3,9,11) and more recently (ref 21,23). Kaliszer's work at Birmingham University illustrated the advantages of a comprehensive integrated control approach. The position taken in this thesis is that a promising area of development is to try and integrate and rationalise some of the most effective areas of control and monitoring into a single system in such a way that the intrumentation will be simplified yet reliable and effective. Chapter 4 explores some ideas for the rationalisation of the instrumentation as a basis for this approach.

5.2 Control decomposition

The most successful examples of control application fall into four main catogories. These are as follows -

a) Control based on models of the grinding wheel/work interface process "A2", chapter 3, controlling feeds and speeds.

b) Control to monitor and reduce vibrations.

c) Control based on models of the machine structure, process "A1_2_2" chapter 3, to counteract structural deformation and damping (ref 11,23).

d) Control of finish size through the use of prediction algorithms based on stochastic process models.

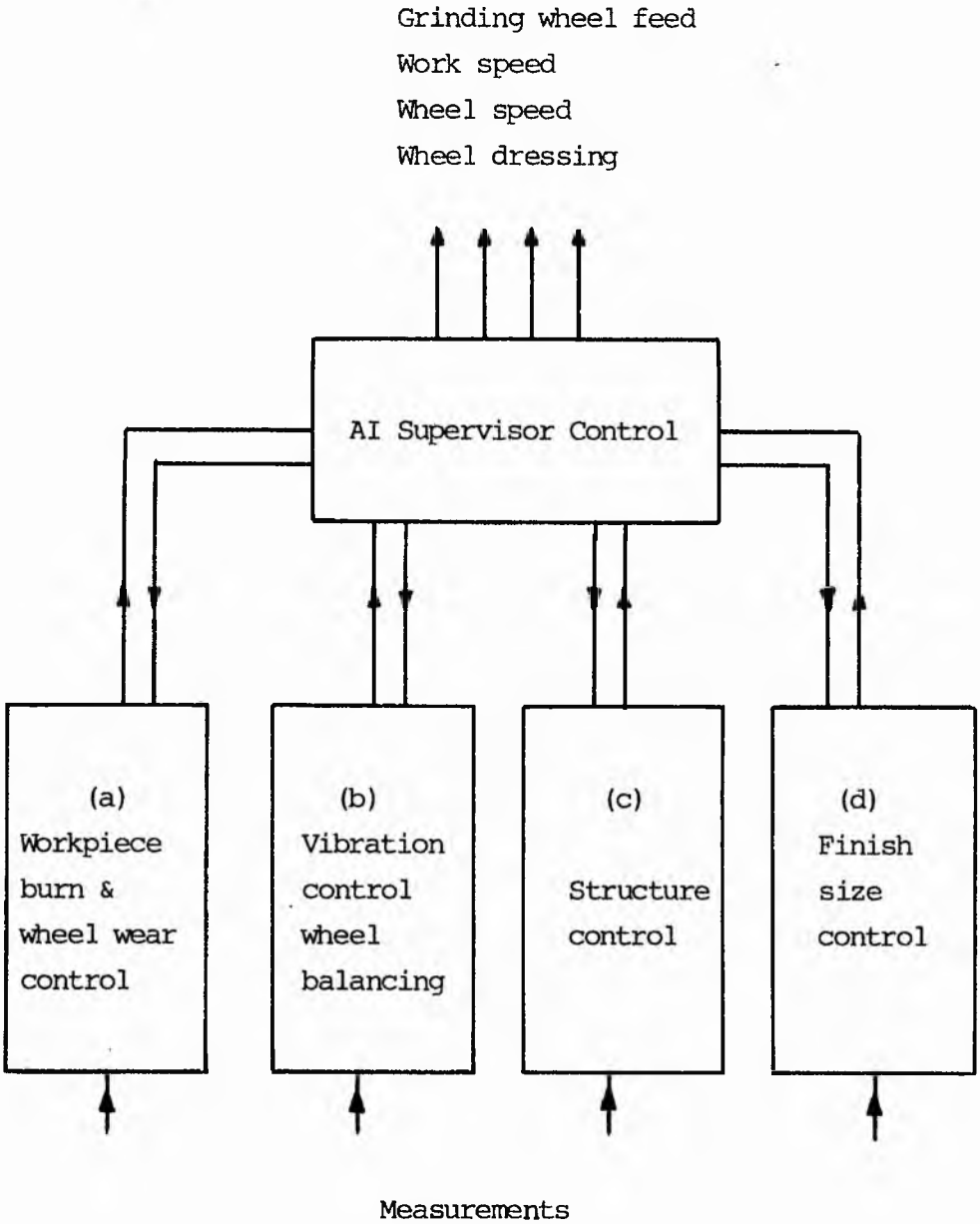
5.3 Supervision of control sub-systems

The four areas of control of section 5.2 may be run in relative isolation on the same system. Attempting to combine the models for the purpose of developing a single control algorithm would be very difficult and costly. A compromise approach might be to adopt a hierarchical control method adopting (a) to (d) as four sub control systems under the supervision of an executive controller. This idea was also mentioned in chapter 1, where it was predicted that such an approach would most likely involve an element of artificial intelligence in the design of an executive controller in order to apply skilled knowledge and experience in the absence of complete models.

5.4 Control sub-system conflicts

The four control sub systems (section 5.2) will react with each other to varying degrees. In the case of controller (a) the speeds and feeds may be controlled in line with a local strategy designed to avoid work piece burn conditions. This choice of speeds may well conflict with another controller's interests. A machine resonance peak for instance (controller b) may have been detected at the same set of speeds. In order to obtain a better view of these sort of problems which the executive will be required to solve, it is appropriate that we look in more detail at the possible design of the subsystems.

Figure 5.1 Grinding process control



5.5 Control subsystems design

5.5.1 Control based on the grinding wheel/work Interface model (a)

The adaptive control system of Malkin (ref 1) is of this type. The models of the wheel/work interface process relate the total power consumed in the process to feeds and speeds and a factor representing the wear on the grinding wheel. Malkin also makes use of a burning limit relationship (ref 6) which gives the maximum allowable grinding power for given feeds, speeds and grinding wheel wear. If the power threshold is exceeded the metal surface overheats and becomes damaged. Malkin's work is concerned primarily with the minimisation of cycle time (with constraints on surface finish). At each level of wheel wear there exists a particular workpiece angular velocity at which the grinding wheel feed is maximised within the burning limit constraint. Malkin's controller tracks the locus of the maximum feedrates for particular wear conditions and sets the feeds and speeds accordingly.

The control objective for this thesis, outlined in chapter 2, places little emphasis on reduced cycle time. This objective is concerned with maximising the possiblity of an acceptable output (component within size and finish tolerance and without burn damage). How the feeds and speeds may be controlled with this objective in mind must be open to a certain amount of speculation. Empirical relationships between feeds and speeds and the surface finish have been proposed (ref 19,20) but the modelling of any mathematical connection between the feeds and speeds and the component output finish size normally leads to difficult complex non-linear expressions.

One approach worth investigating is to adopt the skilled operator approach which historically has been very successful. We know for instance that for each level of grinding wheel wear, "wear flat area", there exists a burning point boundary in the plane of grinding wheel infeed velocity and workpiece angular velocity. The intuitive skilled approach could be to map out the harmful boundary conditions and control the feed and speed to maintain optimum displacement from the boundary conditions. This might be described as a "safe" control method. An alternative strategy could be to minimise the total grinding energy. Excess total grinding energy almost certainly results in greater attrition on the grinding wheel which in turn may affect the output distribution.

A precise strategy for the control of feeds and speeds will undoubtably be decided on the basis of extensive experimental testing. The future of this control subsystem however appears more certain in relation to grinding wheel wear control. The wheel wear parameter can be monitored by applying Malkin's models given the grinding power (see chapter 3). This parameter can be used to control the grinding wheel dressing. There are two possible events that would require redressing of the wheel. The first one is that the wear factor exceeds a threshold level determined by experiment. The other event is that the wheel surface starts to break up, resulting in a sudden fall in the wear parameter, as fresh, sharp, abrasive grains are exposed.
This subsystem should communicate to the executive, information on wheel wear/breakdown and the burning point boundary conditions, with preferred settings for feeds and speeds.

5.5.2 Vibration control (b)

The role of this subsystem is to analyse vibrations and take appropriate action. In chapter 4 it is indicated that appropriate transducers for detecting vibrations would be two optical shaft encoders mounted on the workpiece shaft and the grinding wheel drive shaft. Appendix F gives some details of encoders specialised for vibration measurement.

The subsystem will be required to perform frequency response testing of the structure. The sophistication of this analysis could extend to the use of multi-dimensional fast fourier transform methods. The objectives would be three fold -

1) Identification of Grinding Wheel/shaft imbalance.

2) " of machine structure frequency response.

3) Indication of vibrations related to bearing problems, damaged belt etc.

The frequency response will be two dimensional, based on the frequency of both drive motors.

Based on the frequency analysis, estimates of best settings for shaft speeds should be passed to the executive.

This subprocess could be directly in control of an on line wheel balancing system (ref 17).

5.5.3 Control to counteract structural deformation (c)

The effects of structural deflection have been discussed in chapter 3 and again in chapter 4. The use of the dwell interval to allow for the decay of the displacement has been explained. There are, however, problems with the simple dwell or delay solution. The time taken for the dwell has to be selected on a very conservative basis, representing the maximum possible time required, given the dullest (most worn) grinding wheel. This delay can easily exceed the actual grinding time. Clearly the dwell method is wasteful in terms of grinding cycle time.

There is evidence to suggest also that prolonged sparkout contributes to the increased tendancy toward parametric vibrations (ref 17) and increased grinding wheel wear. This means that there are good reasons, in line with the control objectives, to devise control that will reduce the sparkout time.

Careful consideration of the models of chapter 3, suggests that the fastest possible grinding time for a particular grinding infeed rate, would be achieved by driving the grinding wheel past its normal finish position (overshoot) by an amount equal to the maximum displacement of the structure. The grinding wheel would then have to be withdrawn in a carefully determined way, such that over the next complete cycle the component becomes round. The necessary grinding wheel infeed driving function that would do this can be calculated using information on the structure model "A1_2_2" and the interface process "A2" chapter 3. In order to do this a very fast response feed system is required and it may be that a compromise stategy taking two or more revolutions might be more acceptable (see ref 8).

5.5.4 Control of the finish size (d)

The distribution of the output sizes in the case of the production machines of the sponsor company are controlled by post process manual measurement of the component and adjustment of the grinding wheel final infeed position. This control is carried out at intervals, during which of the order of fifty components may be ground (ref 57). If inprocess gauging is not going to be used then the existing arrangement may be considered for computer control. This is really a prediction problem. The present and past output sizes are known and if the next output size can be estimated then a compensating adjustment can be made to the final grinding wheel position before the next component is ground.

Computer control offers the practical means by which more rigorous prediction methods/algorithms can be implemented. It also becomes practical to implement a compensation adjustment between measurements for all components, using a prediction algorithm that can work for a variable number of steps ahead (interpolation method between one step ahead prediction and the current measurement is possible). If the distribution is modelled as a linear process driven by white noise then a form of Kalman filter can be used to make the prediction and simultaneously estimate the model parameters and error distribution (ref 46). This information is useful to the supervisory controller.

5.6 Supervisor control

The executive or supervisory controller has two main functions.

One job is to resolve possible conflict of feeds and speeds. As mentioned above, the executive should set the feeds and speeds in such a way as to satisfy the requirements of both (a) and (b) sub systems.

The other important function of the executive is to collectively monitor the performance of each sub system and important process variables. It is in this area that artificial intelligence can be considered (ref 63). This part of the control will need further development and could involve considerable investment if the cost of current typical AI projects is a guide (ref 65).

The following simple example demonstrates how expert system features may be included in the executive control. The grinding wheel wear factor (chapter 4) is an important process variable which should not be allowed to rise above a particular threshold level. The setting of this level by an expert represents a simple rule for a rules based system. In practise, far more complex rules might be considered. An example of this is the setting up of confidence factors from different process variables. In this way a decision on wheel dressing for instance, can be taken on a combined assessment of wheel wear, vibration power, the proximity to work burn boundary conditions and the surface finish of the component.

Chapter 6
Implementation of grinding machine instrumentation and control system hardware

6.1 Introduction

In the preceeding chapters the design of a control and monitoring system has been outlined. The complete design of the system will require futher consideration. Sufficient has been achieved however to enable the specification of prototype instrumentation and control equipment to demonstrate the feasibilty of the approach. The objectives would be to set up the infrastructure for the controller and configure and test the instrumentation including the novel shaft encoder developments. The requirements of the controller in terms of processing capability must be examined in order to provide the appropriate resources.

6.2 Control system development environment

The development of the proposed control system requires some provision to be made for the development and debugging of software and hardware. There are many alternative approaches to this problem which all have advantages and disadvantages. The development of the data logger provided some experience of the problems (appendix F). The development environment in this case was the "Hewlett Packcard 64000 system" with microprocessor real time emulation facilities. This arrangement allows for non intrusive debugging and testing in real time. This system however was not selected for the basis of the grinder control development system. After careful consideration, it was felt that the facilties provided by a proprietory stand alone system, running a real time multi tasking operating system, would be more appropriate and could if necessary be supplemented by the emulation system. The system selected was a Motorola 68000 VME bus based arrangement running the "Versados" operating system.

Microprocessor Development System Hardware

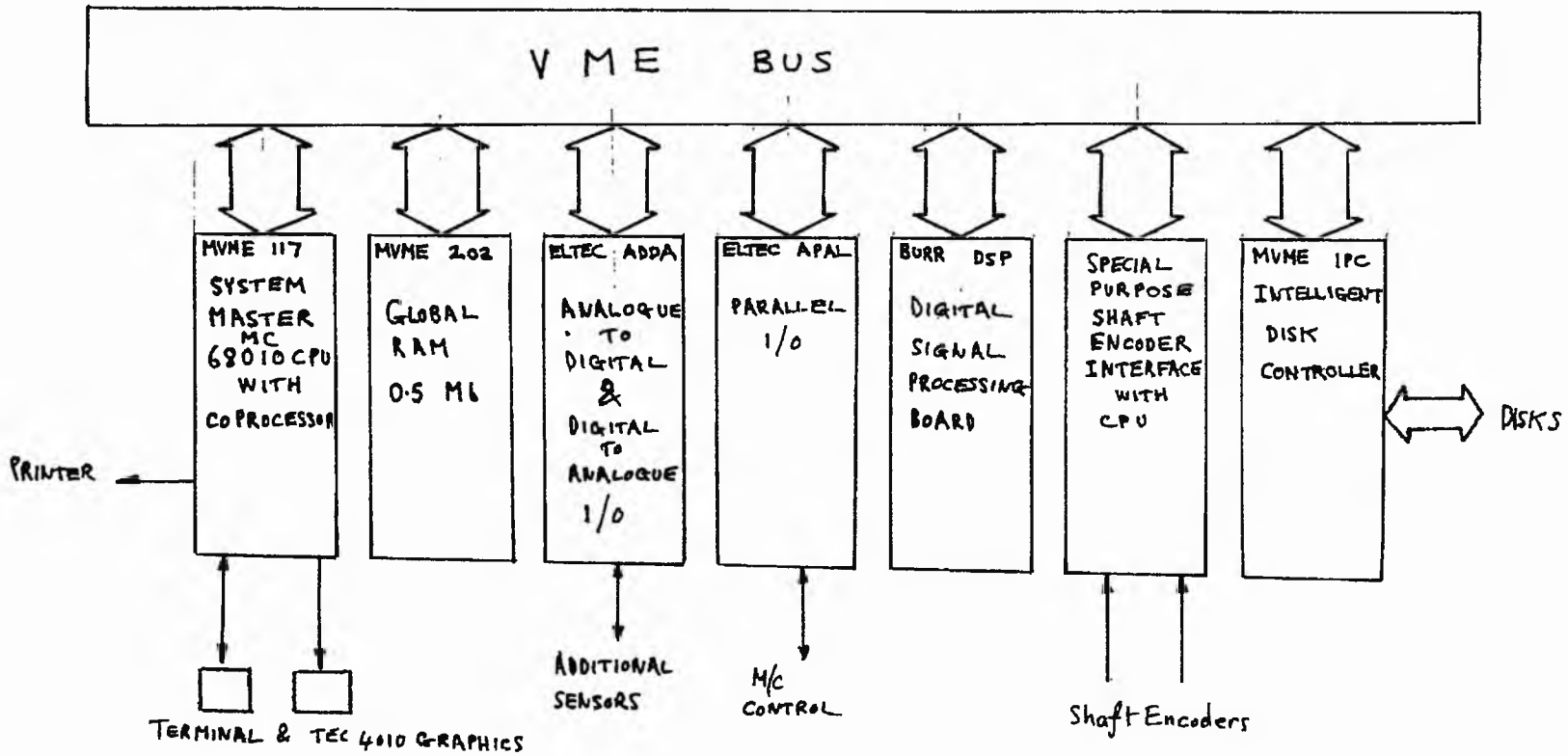


Figure 6.1

Software - Versados, RMS68K, Pascal and assembler

This also provides a "Pascal" compiler, an editor and linker facilities together with extensive software debugging tools. There is also extensive software support for the "RMS68K" operating system. The system runs on the VME multi processor bus (ref 11,12,13,65), which provides for future modular expansion. The "MVME117" processor board provides a system controller function which enables the use of additional processor boards. Interprocessor communication can take place via a section of global shared memory (ref 65,66).

6.3 Instrumentation interfacing

The development system consists of a set of VME cards which include extensive analogue digital and serial interfacing, appropriate for a wide range of instrumentational requirements. The shaft encoders however, require special consideration. If vibrational analysis is required then the two shaft encoders can be generating data at the rate of 80 kilohertz for a basic 20 kilohertz sampling frequency (see appendix A). In practise this level of input places a heavy burden on a single processor based system in which the processor must also carry the overhead of servicing the operating system. This problem may be avoided by using an additional processor, dedicated to shaft encoder interfacing and data aquisition (see appendix B and ref 67). A further processor, dedicated to signal processing including "FFT" frequency analysis is required to process the vibration data efficiently (ref 40).

6.4 Shaft encoder configuration

The experiments carried out with shaft encoders have involved the arrangement of chapter 2 figure 2.3. The encoder was set at the opposite end of the shaft to the workpiece. This arrangement is not ideal for vibration measurement purposes where proximity to the workpiece is necessary.

By mounting the encoder disc rigidly onto the shaft at an appropriate point and mounting the optical sensing device to the machine structure more useful results may be obtained. The mounting of the sensing head is critical and should receive special attention. This is because the vibrations that will be detected will be due to the sum of all the movement between the shaft and the structure, up to the point where the sensor is mounted. The mounting position will determine which vibrations are picked up. To provide comprehensive vibration analysis a minimum of three sensors, mounted at equal angles around the optical grating disc, are required (see appendix F).

Chapter 7
Conclusions

7.1 Introduction

The process of precision cylindrical grinding represents a particular challenge from a control engineering standpoint. The complexity of the process indicated in chapters 2 and 3, has led to most research being limited to the isolated consideration of particular areas of the process. This has however, prepared the way for a better understanding of the process as a whole. This thesis has explored the potential for the integration of recent advances in modern control and instrumentation into a comprehensive control system with novel instrumentation features.

7.2 Achievements

As a basis for the development of a controller, a modelling exercise was undertaken (chapter 3) in order to find a mathematical representation of the process over a complete grinding cycle. The model consists of several linear and non-linear elements including a numerical model of a "centreless" revolving component and a model of the grinding wheel/work interaction. A computer simulation of the system based on the model was developed and tested. The results were compared with data from a real machine, obtained using a specially developed data logger. The interesting results from the data logger results could be explained by detailed reference to the model and could be recreated by computer simulation.

The simulation is a useful testing device for the development of new control methods and improved modelling. Development of the model paved the way for an investigation of new approaches in the field of grinding machine instrumentation and the estimation of process variables.

The use of parameter estimation methods applied to non-linear process models is illustrated. This enables the estimation of features such as grinding forces and grinding wheel wear with the minimum need for experimentally determined constants.

The novel idea (for this application) of using optical shaft encoders to provide information on shaft loading and vibrations has been raised. A new instrument for three dimensional motion analysis has be designed with this purpose in mind (Incremental Motion Encoder). A practical in process method of obtaining drive model parameters is introduced which avoids the awkward problem of having to provide a known excitation input. This facilitates the condition monitoring of the complete drive systems.

Techniques have been developed for the computer interfacing of shaft encoders and the required signal processing.

A scheme for the implementation of a hierachical controller operating with expert system features is introduced. This combines many recent control advances in the field of grinding into a single system.

7.3 Further research

This research provides the basis for further work in several areas. In the field of instrumentation it has been shown that a modified optical shaft encoder (Incremental Motion Encoder appendix F) can be a useful precision digital source of data to describe the three dimensional motion of a shaft at frequencies greater than 20 kilohertz. Work in this field would undoubtedly extend the frequency range and tackle the problems of encoder inter-slot calibration and efficient signal processing. This could lead to the development of a general purpose shaft motion and vibration analysis system.

The work could include consideration of the mounting of shaft encoder sensors for vibration analysis and the correlation of high frequency signals from more than one encoder.

In the field of systems identification further improvements in the accuracy of the parameter estimation (modified inertia approach appendix G) is possible through the application of algorithms which more rigorously account for noise. The extent to which the drive system parameters can be monitored on line to provide details of motor and transmission performance could be further investigated.

The control strategy requires further development if the potential for optimisation are to be established. A significant task will be to find an appropriate cost function that can relate such features as vibration power spectrums to burning point boundary conditions. The development of expert systems techniques applied to the grinding process in real time is a new and potentially rewarding area for further research.

References

1) Challis H, Stanton C, Palmer R, "Grinding - Research on the problems of grinding technology", pub The Science and Engineering Research Council 1982.

2) Kaliszer H, Webster J, "Inprocess measurement of size, shape and surface roughness", p41 int conf on automated inspection and product control April 1982, IFS Ltd.

3) Kaliszer H, Webster J, "In-process measurement and control for the grinding process" 6th Automated Inspection and Production Control Conference, Birmingham, England; 27-29 Oct 1982 I.F.S. (Publications) Ltd

4) Spiewak S (University of Birmingham), Kaliszer H, Kuchta M, "Inprocess wheel balancing" Tech. Pap. Soc. Manuf. Eng. 1984 MR8-533 12p

5) Zhao Y.W, Webster J, Kaliszer H, "In-process size and roundness measuring system for the adaptive control of cylindrical grinding", p289 proc. of 27th int. matador conf. April 1988. pub Macmillan.

6) Malkin S, "Burning Limit for Surface and Cylindrical Grinding of Steels", Annals of the CIRP, Vol. 27, 1978, pp.233-236

7) Malkin S, Koren Y, "Off-line Grinding Optimisation with a Micro-Computer", Annals of the CIRP, Vol. 29, 1980, pp 213-216

8) Malkin S, "Grinding Cycle Optimization" ,Annals of the CIRP Vol. 30/1/1981

9) Malkin S, Amitay G, Koren Y, "Adaptive Control Optimization of Grinding", Journal of Engineering for Industry Feb. 1981, Vol. 103 Page 103

10) He Xiu-shou "Research of a practical adaptive control system on external cylindrical grinding process", proc. of 25th int. machine tool design & research conf. p169 April 1985. pub Macmillan

11) Hahn R. S, "An application of force-adaptive grinding" S.M.E Internatioal grinding conference, Aug 27-29, 1984 Fontana, Wisconsin

12) Konig W, "A numerical method to describe the kinematics of Grinding" Annals of the CIRP Vol.31/1/1982

13) Lacey S, Phd Thesis "Condition monitoring of the centreless grinding process" 1985

14) Sangera J, "Inprocess Monitoring & Control of a Microcentric." R.H.P. Internal Paper 1983, unpublished

15) Toshiyuki Kitamura, "Estimation of the Grinding Process By Power Spectrum Patterns around Grinding Wheel Surface". Bull.Japan Soc. of Prec Engg., Vol. 16, No. 2 (June 1982)

16) Radhakrishnan V, "Functional Assessment of the Grinding Wheel Surface Characteristics by Turbulence Amplifier". Journal of Engineering for Industry. Feb 1981, Vol. 103 / 99

17) Mashinostroeniya V, "The Intensification of the Vibration Process in Grinding with 'Sparking out' ". Soviet Engineering Research Vol 2 No 4

18) Hsieh,Jungche,Inasaki,Ichiro,Nippon,Kikai,Gakkai "Acoustic emissions from grinding" Ronbunshu C Hen v 51 n 468 Aug 1985 p2174 -2179

19) Maris M, Snoeys R. and Peters "Analysis of Plunge Grinding of Steels", Annals of the CIRP, Vol. 24, 1975

20) Snoeys R, Peters J. and Decneut A, "The Significance of Chip Thickness in Grinding", Annals of the CIRP, Vol. 23. 1974

21) Konig W, Steffans K, "Closed loop computer technique to simulate grinding processes", Tech. Pap. Soc. Manuf. Eng 1984

22) Pickering C.J.D, Halliwell N.A,"Real-time Vibration measurement by image holagraphy and photo detection", J Sound Vib v 107 n 3 Jun 22 1986 p 471-485

23) Elnon, Kornel F,"New approach to form accuracy control in machining" Int J. Prod Res v 24 Jul-Aug 1986

24) Chownaird Edmond, "Current state ofthe art in non contact inspection of ground surfaces" Measurement Arts Inc., Providence, RI,USA. Tech Pap Soc. Manuf Eng 1984 MR84-537, 13p

25) Chander P, Bhateja, Keene, "Current state of the art of workpiece roundness control in precision centerless grinding" Annals of the CIRP Vol. 33/1/1984

26) Vekteris, V. Yu "Adaptive bearings of cylindrical grinding machine elemants" Sov. Eng Res v 6 n 11 Nov 1986 p79-82

27) Andreiv N, "On-Line Optical Gauging", Control Eng., vol 27,no 11, 1980 p.78

28) Bailey S.J."Optical Sensors Critical to future Productivity" Control Eng., vol 29, no 1, 1982, p72

29) Yong R.D, "Eight Techniques for the Optical Measurement of Surface Roughness", NBSIR 73-219, National Bureau of Standards, Washington D.C. May 1973

30) Eda, Hiroshi, Kishi, Kozo,Usui "In-process detection of grinding burn by means of utilizing acoustic emissions", Bull Jpn Soc. Precis Eng V 18 n 4 Dec 1984 p 299-304

31) Dornfeld D. "Investigation of grinding and wheel loading using acoustic emission" (Univ of Calif USA) J of Eng Ind trans ASME v 101 nl Feb 1984 p28-33

32) Pilinski V.I.,"Grinding forces and friction coefficient", Sov. J Frict. Wear v 5 nl 1984 p55-61

33) Uida, Taskashi, Hosokawe, Akino, Yamamato,"Studies of temperature of abrasive grains in grinding - Application of infrared pyrometer", J.Eng Ind Trans ASME v 107 n2 May 1985 p 127-133

34) Halvorsen J.V, "Our changing grinding machine requirements", Carbide Tool J v 17 n 2 Mar-Apr 1985p 24-28

35) Chien A.Y, "Selection of optimal stable geometrical configuration in centerless grinding", Int J. Mach. Tool Des. Res. v 2 n 2 1984 p87-93

36) Anon, "Advances in grinding technology", Prod. Eng v 63 n 4 April 1984 p16-18

37) Rao, Suren B, "Grinding machine accuracy and its effect on the workpiece", SME Tech Pap Ser MR 82-240 1982 9p

38) Considine M, "Process Instruments and Controls Handbook", pub. McGraw Hill 3rd edition 1985

39) Vekteris, Yu V, "Stabilizing the temperature field of cylindrical grinding machies" Sov Eng Res v 6 n 7 Jul 86

40) Thomas P.D, Milroy I.P, Forsythe W, "Digital Algorithms for Prediction Differentiation and Integration", IMC Journal 1979

41) Peled A. and Liu B, "Digital Signal Processing" John Wiley and Sons 1976

42) Harris C.J. and Billings S.A, "Self-Tuning and Adaptive Control". IEE Control Engineering Series 15 Peter Peregrinus Ltd 1982

43) Kaplinski, C., "Expandable microprocessor bus for distributed processing." Electronic Components & Applications Vol.4, No.4,
August 1982

44) Balph, T., "VME a System Architecture for Industrial Control."
New Electronics, page 55, August 1985 Vol.18 No. 16

45) Simons, B., "VME in a typical application" , New Electronics
August 1985 Vol. 18 No. 16

46) Jones N. B, "Digital Signal Processing" IEE Series 22 Peter Peregrinus Ltd 1982

47) Franklin & Powel "Digital Control of Dynamic Systems" Addison Wesley 1980

48) Turner M, "Real time experts" Systems International January 1986 p55-57

49) Efstatiou J, "Artifical intelligence and the art of expert control" Computer Weekly Oct 24 1985

International 1985

51) "23 bit optical incremental shaft encoder manual" Ferranti plc, Dalkeith EH22 2NG, Scotland

52) Incremental Encoder Interface Manual THCT2000 Texas Instruments 1987

53) Pressman Roger S,"Software Engineering" Mc Graw Hill Computer Science Series

54) Findeisen W, Bailey F.N, Brdys M, Malinowski K, Tatjewski P, Wozaniak A, "Control and Coordination in Hierarchical Systems" International Series on Applied Systems Analysis, Wiley 1980

56) Pincock D.C., Atherton D.P.,"The identification of a two phase induction machine with SCR speed control", Univ of New Brunswick Canada 1972

57) Rossenblatt, Meir J, Hau L, "Comparative study of continuous and periodic inspection policies in deteriorating production systems" IIE Trans v 18 n 1 Mar 1986 p2-9

58) Voelcker J, "Instrumentation" IEEE Spectrum v 23 n 1 Jan 86 p57-60

59) Nayar K.R, "Microprocessor-based instrumentation", J. Inst. Elect Telecom Eng v 29 n 12 Dec 1983

60) Gorbunov Yn. N, "Multistep Proceedure for measuring the parameters of a repeating signal through a stochastic averageing of digital readings" Optoelect Instr Data Proc n3 1985 p101-105

61) Tulumello A, "Study of reliability in instrumentation" (T&T Consulting Associates USA) ISA Trans v 24 n 4 1985 p7-15.

62) Domrachev V.G, Topilski V.B, Meiko B.S, Isaev V.M, Kampenko I.A, "Accuracy checking method for precision angle digitizers", Meas Tech v 29 n 5 May 1986 p374-377

63) Pham D.T, "Expert Systems in Engineering" IFS Pub Springer-Verlag

64) Srinath M.D, Rajasekaran P.K., "An introduction to statistical signal processing with applications", Pub J. Wiley Interscience 1979.

65) Mc Clelland Stephan, "A suitable case for treatment", Sensor Review Vol 8 n 1 1988

66) Mc Clelland S, "Giving AI real-time sensing", Sensor Review Vol 8 n 1 1988

67) "The VMEbus specification" Motorola series in solid-state electronics, Printex Publishing Inc, Rev. C.1 Oct 1985

68) Comer D.J,"Microprocessor-Based System Design",Pub Holt, Rinehart and Winston 1986

69) Cooling J.E, "Real-time Interfacing", Van Nostrand Reinhold (UK) 1986

70) Hardware Operating Manual for SPV100 VMEdsp board - OM100, Burr-Brown Ltd, Livingston, Scotland 1984

71) Kernighan B.W, Ritchie D.M, "The C programming language" Printice-Hall software series 1978

72) Cherry George W, "Pascal Programming Structures for Motorola Microprocessors 1982"

73) Leventhal Lance A., Hawkins Doug, Kane Gerry and Cramer W.D, "68000 Assembly Language Programming" Osbourne McGraw-Hill 1986

74) Wiener R.S, "Introduction to object-oriented programming & C++" pub. Addison Wesley 1988.

APPENDIX A

Shaft encoder for velocity measurement

A.1 Introduction

This appendix examines methods of obtaining velocity estimates from shaft encoders. The particular application of interest explained in Chapters 2 and 3, has two requirements. One requirement is to obtain accurate estimates of velocity at constant time sample intervals at a sampling frequency of approximately 50 hertz. This frequency has been found to be appropriate for the analysis of the low frequency dynamics of the system (ref 9, response of drives to disturbance load, power consumption due to grinding). The additional requirement is to obtain higher frequency data that will describe the relatively low power vibrations of the shaft, relative to particular parts of the machine structure.

A limit of 20 kilo hertz maximum sampling frequency was that which could be easily achieved with readily available hardware (appendix B) and also covered a significant range of the vibration spectrum of interest (ref 13). With further development of the electronics for high frequency timing (50 -100 megahertz clock) and higher resolution encoders, sampling frequencies could go several times higher.

This appendix shows how the high sampling frequency data is appropriately formed by collecting measurements of time at equal angular increments. A method of producing the low frequency constant time sample estimates of velocity from the high frequency timing data is explained. This method results in low quantisation errors.

The appendix also considers the effects of differentiation algorithms on the spectral distribution of the velocity estimates and suggests appropriate algorithms.

A.2 Sampling strategy

An optical incremental shaft encoder can be used to produce digital data from which it is possible to produce estimates of the dynamics of the shaft (chapter 4). Various different methods of collecting the data and interpreting it for position and velocity estimation relative to time may be considered. A position sampling technique is required. The following general points must be considered-

1) The required sampling frequency
2) The allowable error possible in the measurements
3) Does the signal contain significant power in the frequency range higher than half the sampling frequency?

In the case of shaft encoders it is required to decide the resolution of the encoder or number of slots and also the resolution of the base timing reference. Two established ways of using an encoder to estimate position/velocity are as follows. By measuring -

a) Incremental angular position at regular time intervals.
b) Time taken to travel a set angular distance.

Method a) is commonly used in control systems where measurements are all taken at a fixed sampling rate in time and the resulting quantisation error is small enough for the particular application.

Method b) is effectively "time sampled at fixed angular sampling movements". This is particularly useful for relating small velocity changes (vibrations) to the cyclic motion of the shaft to which the information is invariably related. Also this method, because of its synchronisation with the angular rotation, allows for the simpler accomodation or correction for inter-slot angular errors in the shaft encoder (appendix F).

A.3 Quantisation errors of alternative sampling methods

At a particular sampling instant the angular position of a shaft may be estimated with a quantisation error approximately equal to the angle of a single encoder increment.

At a particular angular position in time, indicated by the transition of an encoder output wave, the time can be measured very accurately. In this case the error in the esitmate of the position at the recorded time is due to the quatisation error of the clock. This error is aproximately equal to the anglar travel possible during a single clock period. The error is consequently proportional to the velocity.

If we consider a single position in time estimate it follows that the sampling method with the least error (a or b) depends on the particular conditions. Below a particular velocity there will be more than one clock pulse during a single incremental movement. In this situation the least position error will be given by method (b).

The transition velocity for a 1000 increment encoder and a 20 mega hertz clock is given by -

$$\text{transition velocity} = \frac{60 * 2E7}{1000} = 12E5 \text{ rpm}$$

Below 1200,000 rpm angular error is less if time is measured at incremental angular positions.

The above sampling method was selected for the grinding machine principly because of the potentially high accuracy of position estimation which provides the basis for the investigation of low amplitude vibrations.

If only the low frequency constant time sample velocity estimation were required then it may be considered more appropriate to use method (a) and count encoder increments over fixed periods. Careful selection of differentiation algorithms (ref 40) could help provide estimates of velocity sufficiently accurate for the required analysis of shaft dynamics. Hardware and software requirements would be simplified.

The fact that type (b) sampling is to be used for vibration analysis means however that it is most appropriate to estimate the low frequency constant time sample velocity from the existing data using interpolation methods. It can be demonstrated (see section A.12) that the constant low frequency constant time sample data is infact a sub set of the higher frequency sampled time data.

The type (b) sampling data consists of a series of clock counts recorded at the angular incremental transitions. This record will here after be referred to as the primary position record.

A.4 Encoder interface

(more details in appendix B)

The simplest configuration of a system to record the primary position could involve a simple counter and latch arrangement. A circuit (appendix B) may be arranged so that reference pulses generated by the encoder will latch/save a count on the free running counter which is counting clock cycles from a crystal oscillator timing reference. The temporarily saved values can be conveniently saved by a processor in contiguous order in memory.

A.5 Definitions

Let P[] be an array for the storage of the primary position record or clock counts at the angular reference instants.

Let A[] be an array to store angular pulse counts at equal sample time instants.

Square brackets "[]" is used to indicate an array of unspecified size.

A.6 Processing P[] (the primary position record)

In the simplest system only P[] will be recorded. If P[] overflows it is an elementary matter to create a new record which has a non- overflowing time count simply by totalising differences while moving down the record. Various such non destructive transformations can be useful, as in the case of the early experimental logger data to transform the P[] record from an eight bit record into a sixteen bit record. The size of the data storage if P[] is to be non overflowing can be a problem,in which case it is useful to store only the timing differences which require less memory (this first difference record is referred to as the "k" record and algorithms for velocity estimation can generally be rewritten in terms of these values).

Transforming the record to a centre pivot two term velocity estimate record is also non-destructive and useful. The record P[] will generally be much longer than the A[] record, or to put it another way; between the constant time samples there will be many angular samples. This can be important in that the primary position record may be filtered prior to extracting lower frequency constant time samples (ref 8 decimation) in order to remove frequencies higher than the required sample frequency divided by two. This reduces possible alias problems.

A.7 Estimating velocity at constant time intervals

There are a large number of algorithms for estimating a differential of a discrete signal ranging from simple two term to sophisticated Kalman filter current estimators. Here however no restriction is placed on which could be used. This section only considers examples of simple algorithms applied to the records P[] and A[]. What is considered, are ways of using just A[] for the basis of the estimate, or alternatively using A[] and P[].

Let T = period of sampling

f = frequency of time base clock

Pi = constant 3.1416

n = integer number for time sample reference

m = integer number for encoder pulse count from time zero

N = number of slots on encoder

P[] is reference timer record at constant angular intervals

A[] is angular count " " " time "

v(m) is velocity estimate at angular count m

V(nT) is " " " time nT

If only the A[] record is processed the simplest velocity estimate is given by -

$$V(nT) = (Pi/N)*(A[n+1] - A[n-1])/T$$

The following is an example of using P[] and A[] to find V(nT)
(not forgetting that A[] is really formed from P[])
The principle is simply to produce velocity estimates using the P[] data, which are for P[] instants in time. These estimates are then used for estimating the velocity at the constant time intervals. At time nT angular count is A[n]. This means at time nT the angular position is anywhere between A[n] and A[n]+1. The more accurate time for the angular position A[n] is given by -

P[A[n]]

To find velocity estimate (centre pivot two point as an example)
see figures A.1 and A.2 in this appendix

a) first find half angular travel over the period for estimate

$$Z = A[n+1] - A[n-1] + 1/2 \quad \{ Z \text{ is integer } \}$$

(non integers rounded up)

b) estimate velocity at angular position A[n]

$$v(A[n]) = 4*(Pi/N)*f*Z/(P[A[n] + Z] - P[A[n] - Z])$$

c) estimate velocity at angular position A[n]+1

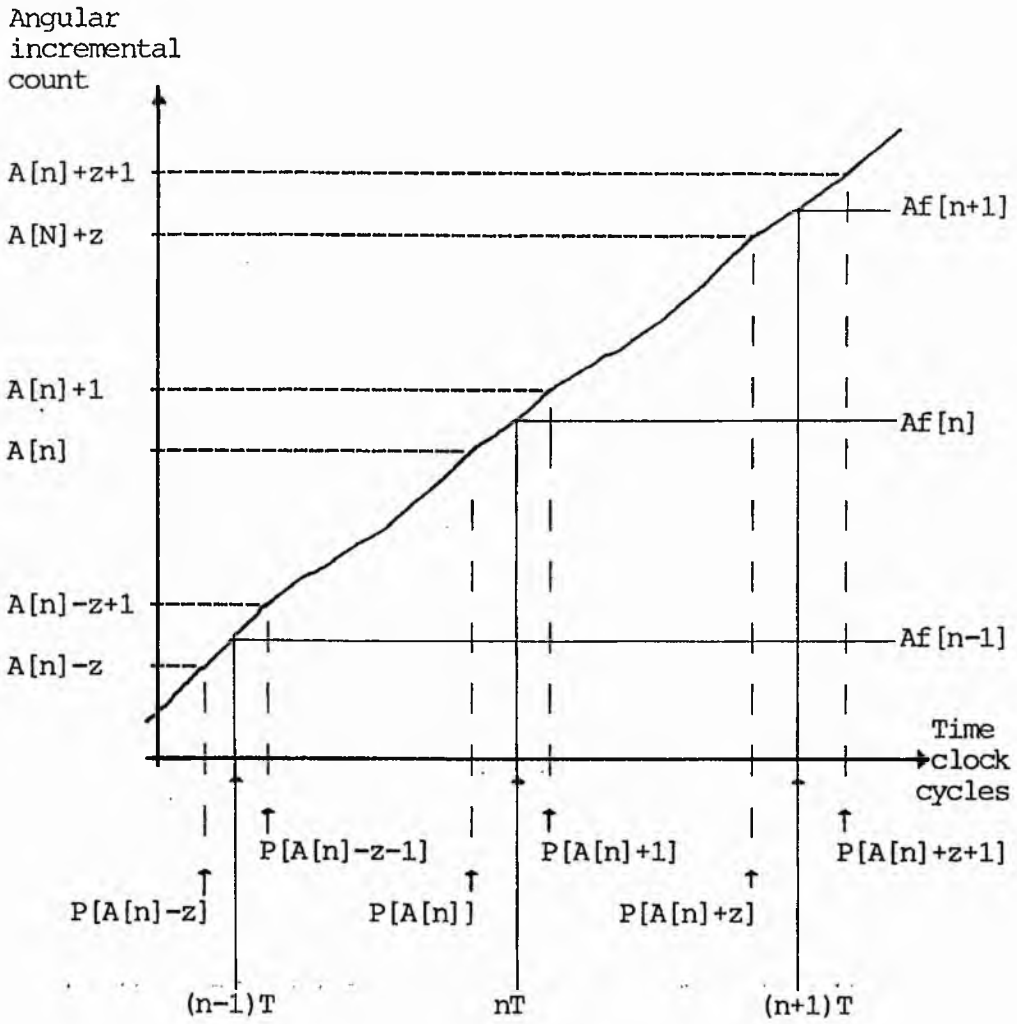
$$v(A[n]+1) = 4*(Pi/N)*f*Z/(P[A[n] + Z + 1] - P[A[n] - Z + 1])$$

d) Interpolate for velocity at nT

$$V(nT) = v(A[n]) +$$

$$(v(A[n]+1)-v(A[n])*(nT-P[A[n]])/(P[A[n]+1]-P[A[n]])$$

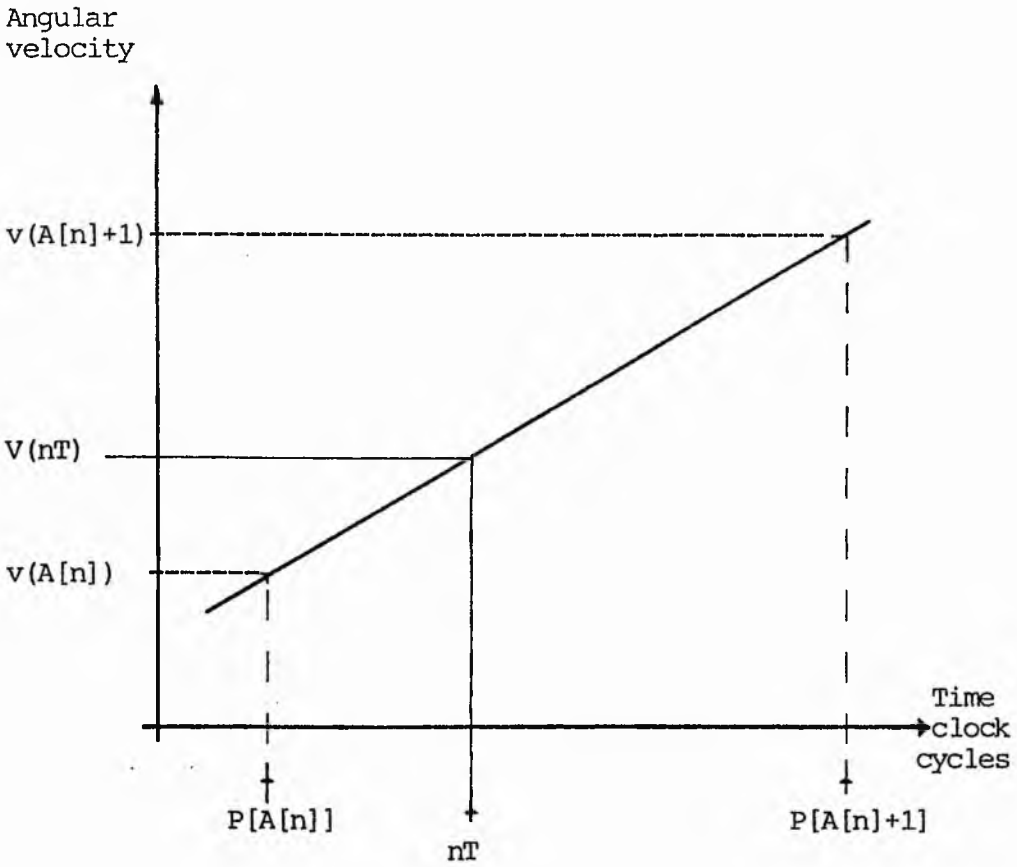
Fig A.1 Angular position versus time graph



nT - nth constant period sample time
P[m] - time at angular incremental position m
z - integer incremental angle from sample (n-1)T to nT
Af[n]- estimate of position at time nT

Graph shows data used in the estimation of position/velocity at regular sample times using a record of times at equal angular increments.

Fig A.2 Angular velocity versus time graph



v(A[n]) - Estimated velocity at last recorded time before nT

v(A[n]+1) - Estimated velocity at next recorded time after nT

V(nT) - Estimated velocity at nT (interpolated)

The above method uses accurate data from constant angle time sampling to interpolate for regular time sample estimates. Additionally the angular sampling interval is checked for each estimate and made close to the angular travel in the time sampling period. Thus this method will have the advantage of low quantisation error while at the same time being based on data collected approximately at the constant time instants. This means that the inherent filtering effect of a velocity estimator is equivalent to that for estimation based on A[] type data alone.

At low angular velocity (when as little as ten units of angular travel exist between sampling intervals) this method will still return a reasonable estimate of velocity.

Note - An alternative approach would be to first find accurate estimates of angular position at the regular sample time instants using a suitable interpolation method. This new record would then be the basis for estimation of velocity.

A.8 Levels of estimate of velocity

(from simple onwards) for regular time sampling

At time nT (1 to 5 based on three point central difference estimate)

1) V(nT) = (Pi/N) * (A[n-1] - A[n+1])/T

2) " = v(A[n]) where Z is a constant chosen for an average velocity.

3) " = v(A[n]) where Z is (A[n+1] - A[n-1])/2

4) " = v(A[n]+1) if (nT - P[A[n]]) > T/2

5) " interpolated value from v(A[n]) to v(A[n]+1)

6) As above but using higher order estimation algorithms up to fifth order.

7) Optimal estimation based on above but considering noise (Kalman filter)

A.9 Comparison of quantisation errors

Quantisation error for 1) is Pi/(T*N) rads/sec

Quantisation error for 3) onwards

error for V(nT) approximately equal to that for v(A[n])

" " 1/v(A[n]) = N/(4*Pi*f*Z) sec/rad. = q

but Z is approx v(A[n])*T*(N/Pi)/2

so that $q = 2/(v(A[n])*T*f)$

Quantisation error for v(n) is given by the difference between

v(A[n]) and $\frac{1}{1/v(A[n]) + q}$

now the second term = v(A[n])*(1/(1 + 2/T*f)

by using a Taylor series expansion

= v(n)*(1 - 2/(T*f) + (2/(T*f)**2 - ....

but 2/T*f will be small so that its larger powers can be ignored. Thus quantisation error for V(n) can be approximated to

2/(T*f)

comparing type 1) to 3) quantisation error

ratio is PI*f/(N*2)

as an example f = 10 E7, and N = 1024
(10 Mhz clock 1024 slot encoder)
then quantisation error is about 15,000 times
less using method 3) onwards.

Further example

Using the following -

Timing clock	4 MHz.
Shaft encoder	1024 clocks per rev.
Average speed	600 rpm.

Using sampling methods 3) onwards. The approximate sampling frequency is 10 Khz. The samples of time will effectively be resolved into approximately 400 elements. If on the other hand we had a 10.24 khz fixed sampling rate and recorded encoder pulse counts we would need an encoder with 400*1024 = 409600 pulses per revolution to rival the adopted method (3 on). Whilst 23 bit encoders do exist they are very expensive and rely on processor hardware and software which introduce conversion time delays.

A.10 Implementation (hardware considerations)

The use of methods beyond 1) depends on the availability of the P[] data. It is clear that between each time sample instant there must be a large number of data recordings in P[]. Memory storage for P[] must be far greater than A[] (typically not less than 50 times). This represents a heavy processing burden to collect the data at over 50 times the required sample rate and find suitable storage. It should be noted however that the velocity estimates as above will not require the use of the whole record, but only a few elements scattered in time close about the regular sample time. This last point could be exploited to reduce processor time, but here it is not pursued because there are other reasons why we wish to maintain the complete P[] record (vibration analysis). The development system (chapter 6) based on VME Versados with RMS68K can only cope with processing data from the encoder at a rate of about 2000 samples per second and this represents a substantial burden on the resources. The rate for a 1024 pulse encoder running at 400 rpm is about 8000. The capability of the single processor is hampered by the over head of the operating system RMS68K. The development system however was selected partly because of the multiprocessor capability of the VME bus. It is therefore possible to extend the system by adding a processor dedicated to shaft encoder logging and associated processing, filtering, estimation etc. The additional processor runs on a separate board on the VME bus as a slave in the master slave configuration. The slave is able to run concurrently with the master and any other slaves whilst sharing an area of global ram (memory available to all processors on the bus).

Software has been developed to allow programs to be produced using the facilities of Versados on the master (editor, compiler, debug etc) and then to download or transfer to the target slave boards local memory for running on the slave processor. Alternative aids exist in the form of the real time emulation system which enables efficient realtime testing and debugging of the slave. The purpose of the slave is primarily to collect and store the P[] data. The slave may also maintain a record of the A[] data and generate interrupts to the master at constant time intervals based on the high frequency reference clock used in the generation of P[].

The P[] data can be organised as 32*2Kbyte cyclic buffer (ideal from a programming point of view since a 16 bit buffer address pointer will wrap around every 64K) A buffer four times as big would be required if the whole grinding cycle were to be saved. If the encoder servicing is by interrupts then the processor is required to service about 10,000 interrupts per second. Ideally the board will calculate a current estimate to pass to the master processer at the sample time and keep a cyclic buffer of estimates at the sample instants. The estimated velocity buffer may be in global ram together with a pointer to the current estimate. The current estimate should be placed in the buffer at the position pointed to as the head, but this estimate and recent estimates should be overwritten as better estimates become available. For future consideration the P[] data can be passed to an addition signal processing board for FFT analysis etc. Care is needed in the use of the global memory because this is only available by using the VME bus which is shared by the other processors. This means that ideally the i/o for encoders which does not need to be accessible to other processors should be taken off the bus and made local to the dedicated processor.

A.11 Velocity Estimates (based on reduced data)

It was stated above that there is a possibility of working in the inverse velocity manner without the overhead of having to save all the angular clocked time record P[]. The method requires a modification to the basic interface (appendix B fig B.2). The interface to the encoder was designed on the principle that there was no requirement to keep a register and counter for the angular position because this would be done effectively by the processor incrementing a memory pointer. In order to keep track of angular position it is necessary to include this additional register either configured so as to be available on a second port or sequentially with the timing data on the single port.

Data is as follows -

A[n] is angle count at time nT (as before)

P[n] is now the actual clocked time at angular count a[]

(P[] is now same size as A[] and may be structured as a two dimensional array with A[n,0] as A[n] and A[n,1] as P[n])

Velocity at time A[n,1] (three point centre pivot for example)

$$V(n) = (A[n+1,0] - A[n-1,0]) / (A[n+1,1] - A[n-1,1])$$

Then velocity at nT

(using simple interpolation from V(n) to V(n+1)

$$= V(n)\,(1 + (V(n)-V(n-1))/(A[n,1] - A[n-1,1]))$$

A.12 Creating an A[] record starting from a P[] record

This involves simply searching down the record, keeping a track of elapsed time since the record start and recording the record number whenever an interval containing a regular sample time instant nT is found.

Let m = angular count — integer
n = time sample clock cycles — integer
e = elapsed time — unsigned long integer
M = length of record — unsigned integer
Process as follows

Code fragment example is using "C" language

```
        n = 1;
        e = 0;

    for (m = 1;m < M;m++)
            /* loop for incrementing down P[] */
       {
         e = e + (unsigned int)( P[m] - P[m-1] );

         if ( e > nT )
              {
              A[n] = m;
              n = n++;
              }
        }
```

If great speed is required then a more efficient search is possible by jumping A[m] - A[m-1] records when A[n] is found. When looking for the next angle count the starting point will then be at or close to the required interval. The search then will typically involve only at worst a few steps up or down the list. Care should be taken however to see that

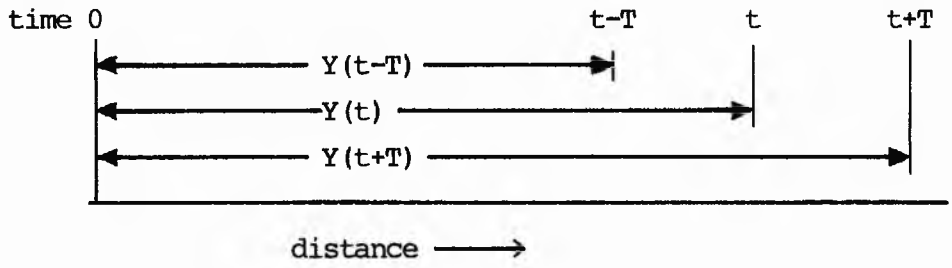
(unsigned int)(P[m + (A[m] - A[m-1])] - P[m])

does not overflow.

A.13 Frequency Response of differential estimates

Having considered in some detail ways of reducing the quantisation error it is now of interest to examine the spectral effects of the algorithms. The line taken here is that the estimators should first be considered theoretically as continuous functions of velocity. The velocity is then considered as being seen as passed through a continuous filter and the gain/frequency response of the filters are examined. In the case of the simple centre pivot continuous algorithms there two areas of interest. Firstly the basic period of integration which is parallel to but not necessarily equal to the sampling period. Secondly the structure or order of the algorithm which can enhance or degrade parts of the velocity spectrum.

Figure A.3



Consider the position of an object moving in straight line (for convenience). The distance covered at a particular time (Fig A.3) is defined as Y(t).

After time "t-T" it has moved a dist "Y(t-T)".
After time "t+T" it has moved a dist "Y(t+T)".

In this case an estimate of the velocity at "t" is given by the two term centre pivot algorithm-

$$Ye'(t) = \frac{Y(t-T) - Y(t+T)}{2T} = \frac{1}{2T}\int_{t-T}^{t+T} Y'(q)\,dq$$

Ye'(t)- estimated velocity
Y'(t) - true "
("q" is an arbitary integration variable)

Now because we are considering centre pivot, differential estimators will get no phase shift in the estimate so that the simple substitution -

$$Y'(q) = SIN(wq)$$

with no complex part may be used to find the gain. Then -

$$Ye'(t) = \frac{-COS(w[t-T]) + COS(w[t+T])}{2wT}$$

$$= \frac{SIN(wT)SIN(wT)}{wT}$$

This means that the normalised gain of the continuous function Ye'(t) over Y'(t) is given by -

$$\text{gain}(w) = \frac{\text{SIN}(wT)}{wT} \qquad \text{---(1)}$$

We can repeat the above for higher order estimates (see ref 40) For a 5 point estimate -

$$\text{gain}(w) = \frac{8\text{SIN}(wT) - \text{SIN}(2wT)}{12\ wT} \qquad \text{---(2)}$$

For a 7 point est.-

$$\text{gain}(w) = \frac{135\text{SIN}(wT) - 27\text{SIN}(2wT) + 3\text{SIN}(3wT)}{180wT} \qquad \text{---(3)}$$

The response of higher derivative estimate continuous functions will follow the same method except that the first expression will now involve the evaluation of multiple integrals. Taking the acceleration as an example and using now the expression for 3 pt second derivative estimate then-

$$Ye''(t) = 1/T_2 \left\{ \int_t^{t+x} \int_0^z Y''(u)\,dz\,du - \int_{t-x}^{t} \int_0^z Y''(u)\,dz\,du \right\}$$

For the present however we will consider the responses of the first derivative.

A.14 Plots of continuous estimate functions

Taking equation (1) and making the following substitution

$$T = \frac{1}{Fs} \quad , \text{and} \quad w = 2\pi f$$

(f is freq c.p.s. and Fs is equivalent to the sampling freq. had the function been sampled)

Then

$$\text{gain}(f) = \frac{Fs\,SIN\,(2\pi f/Fs)}{2\pi f}$$

i.e. unity gain at zero freq. and zero gain at Fs/2, 3Fs/2, 2Fs etc.

The frequency response curves for the above 3 estimates have been plotted (see fig A.4 and the associated program A.1). Thus the spectrum of the true velocity will be enveloped by these functions. The effect of sampling these functions is as usual to create aliassed images of the spectrum about the sampling frequency. The following observations are made.-

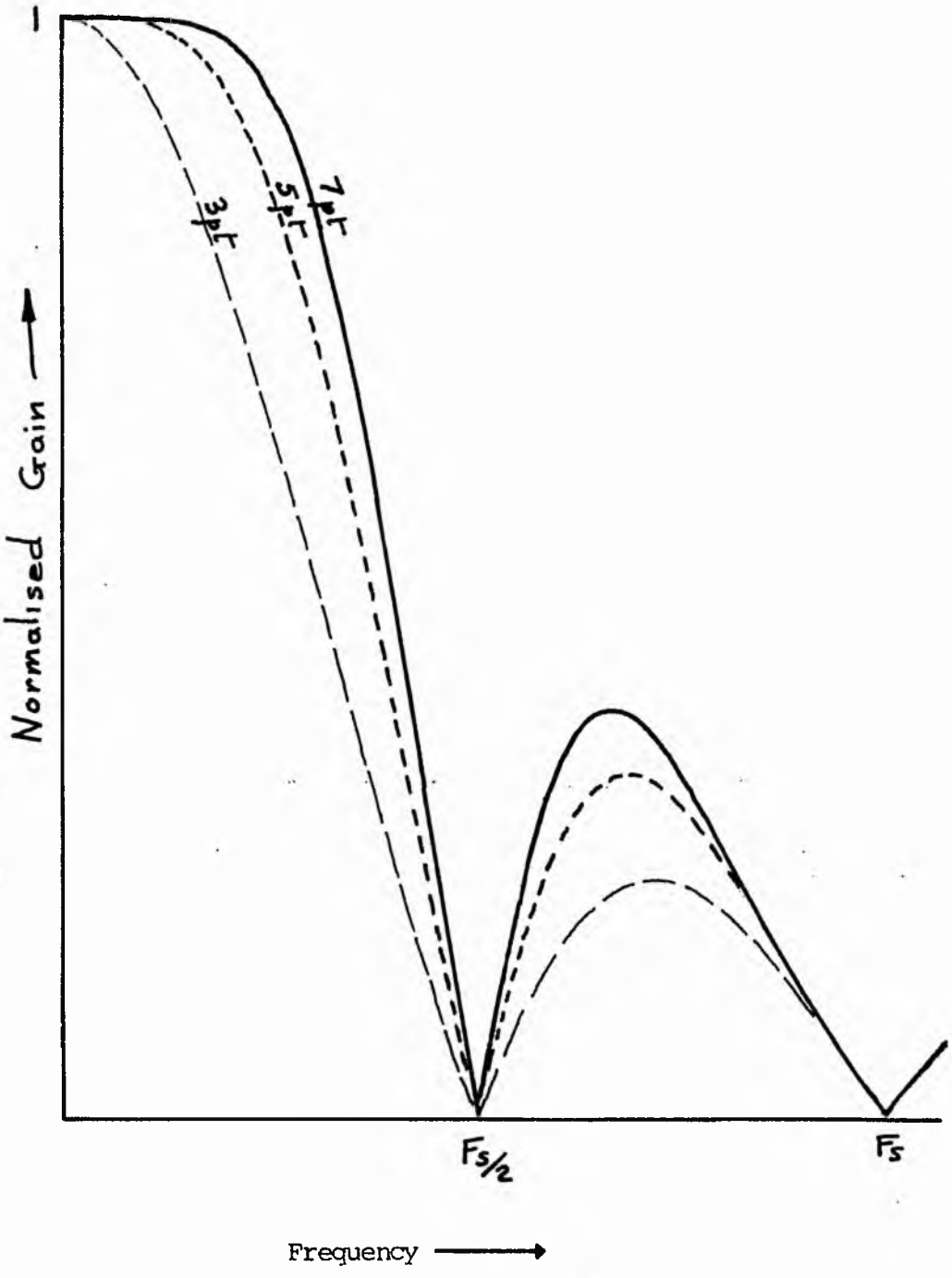
a) The higher the order of the algorithm the better the frequency response becomes as f approaches Fs/2.

b) All the estimates tend to filter out frequencies above Fs/2 (hence playing an anti aliasing role). Unfortunately the higher the order of the estimate the poorer the attenuation becomes.

This is in line with the findings of ref.40. That is that the truncation error (error due to limiting an estimate to a finite no. of terms) is critically dependent on the magnitude of the higher derivatives. Clearly the high frequency components determine the higher derivatives (small high frequency components means small higher derivatives etc).

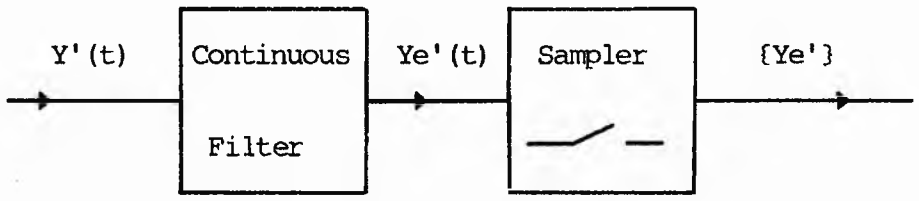
Fig A.4 Frequency response graphs

The continuous time equivalents of the descete time 7,5,3 and point centre pivot velocity estimators. These graphs show the filtering effect of the estimators. The seven point shows the best response towards the half sample frequency mark.



What has been done is to postulate the existence of a continuous function Ye'(t) (the estimate of the function Y'(t) we really want). This function is a simple sum of integrals equivalent to a sampled signal estimate. We can then find the freq. response of this continuous function. The freq. response produced by sampling this function gives what is really the filtering effect of the sampling algorithm on the required signal (Y'(t)). The required signal can be viewed as subjected to a continuous filter followed by a sampler.

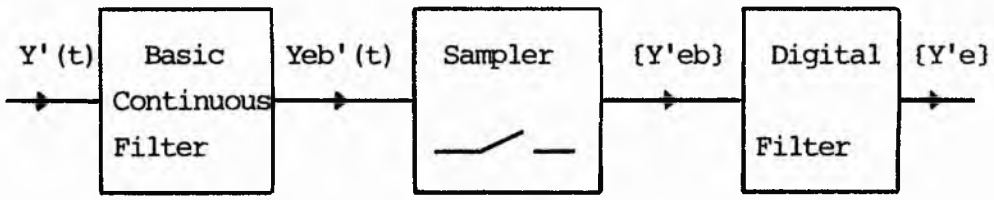
Fig A.5



(curly brackets means sampled signal)

An alternative view is that the continuous filter is only operating in the simplest manner (integrating over only the fundamental period with no sums of integrals). The continuous filter must then be followed by a digital filter which characterises the required algorithm.

Fig A.6



As an example of this approach the three estimates for first derivative as above were used. The digital part of the above illustration was first examined. The input to this part {Y'eb} is in fact the "k" values as already described (the algorithm for the derivative estimate is transformed to a filter of the "k" values or first order estimates). The frequency response of the digital filter is found by considering its "Z" transform -

$$K(z) = \sum_{n=0}^{N} k_n z^{-n} \text{ , and substituting } z = e^{jL}$$

(where L is the digital frequency defined as L=2 f/Fs)

Then the gain is given by -

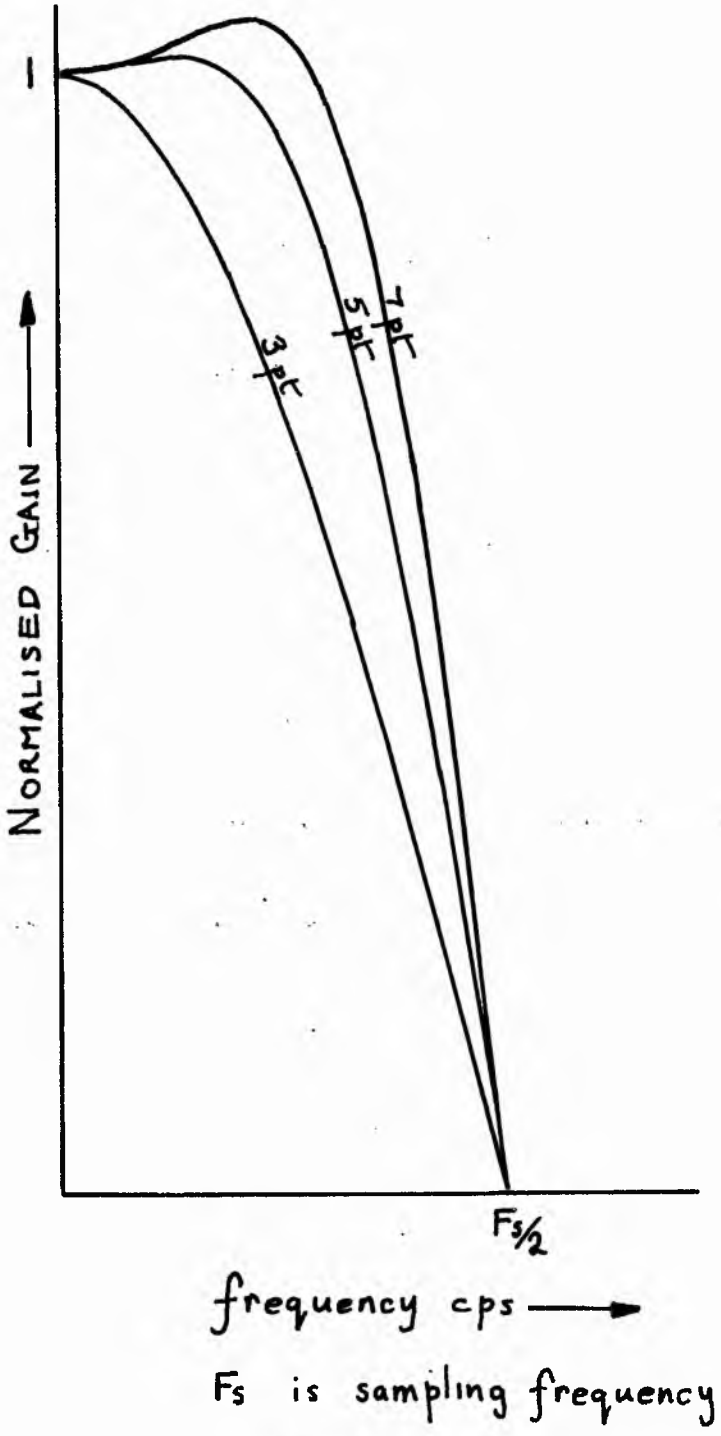
$$|K(e^{jL})| = \sqrt{\{ \sum_{n=0}^{N} k_n COS(nL) \}^2 + \{ \sum_{n=0}^{N} k_n SIN(nL) \}^2}$$

A generalised program was written to plot this gain for different filters.

Figure A.7 shows the results for the three derivative cases. These curves are not the same as the curves in fig A.4, even below the sampling frequency Fs/2. This is because the continuous part of the filter has not been included. If the digital gain function in the range 0 to Fs/2 is multiplied by the continuous gain function of the first part of the filter (fig A.6) then we will get the overall freq. response which should be the same as that obtained in the previous method (fig A.5, freq range 0 to Fs). The drawback with this method is that we will not be able to see easily the anti-aliasing features observed in the continuous filters in fig A.4

Fig A.7 Frequency response graphs
Descrete velocity filters

Graphs illustrate the filtering effect of 3,5 and 7 point centre pivot velocity estimators on the actual velocity.



A.15 Time sampled at equal angular displacements

When we reverse the use of time and displacement, by counting clock cycles between shaft encoder output pulses, problems arise. The system now has a variable sampling frequency. Sampling frequency is directly proportional to the angular velocity. Plots of the encoder pulse periods against angular displacements describe the same relation between time and motion as before. However a direct interpretation of velocity from these results is, in terms of cycles per unit, angular distance.

Analysing this data directly in its inverted form to obtain frequency spectrum is straightforward and useful in that it directly relates oscillation to the periodic shaft rotation.

The first part of this appendix explains how it is possible to easily transform this type of record to a constant time sampled record and also the advantages of having this as the primary source of data.

A.16 Constant angular velocity

The precision grinding machine drive shafts to both the grinding wheel and the work piece according to the original design of the machine are intended to maintain a narrow angular velocity range. The general thinking behind this was that carefully regulating such variables would produce a stable system required to maintain the accuracy of the output. Recently (Reliable Grinders Project R.H.P) improvements were made to still further control the workpiece velocity with the addition of an improved electronic speed regulator. The speed controller is intended to maintain the velocity of the work drive motor within plus or minus 2% and experimental data shows that the shaft speed is within 5%.

A.17 Velocity estimation avoiding need for inversion

An approximation to velocity, avoiding the need to invert the sampled time data, can also be made as follows-

Let Y'av = average steady state angular velocity.
Y'e(r) = angular velocity estimate at time rT (sample period T)

$$1/Y'e(r) = \frac{1}{Y'av + (Y'e(r) - Y'av)}$$

$$= \frac{1/Y'av}{1 + (Y'e(r) - Y'av)/Y'av} \quad \text{------(4)}$$

Now using the Taylor series expansion

$$\frac{1}{1 - x} = 1 + x + x^2 + x^3 + - - -$$

and if Y'e(r) is close to Y'av then (4) can be approximated by-

$$1/Y'e(r) = \frac{2 - Y'e(r)/Y'av}{Y'av} \quad \text{approx.}$$

$$Y'e(r) = \frac{4hY'av - (\text{inverse velocity})}{2h\ Y'av^2}$$

The "inverse velocity" can be found by a suitable differentiating algorithm such as a two term central difference. The first term, being a constant, is dropped for second derivatives. If the approximation method is valid the arithmetic becomes virtually no more computationally difficult than the alternative (dist sampled by time) method.

A.18 Quantisation Errors

The quantisation error of a measurement is related to the least significant bit of the digital word used to express the measurement. In the case of the "k_r" record (time between encoder pulses) the quantisation is the period of one timing clock cycle. This is the max. possible error in the measurement since the true value is always truncated. In an estimate this error is proportional to the sum of the positive coefficients of the algorithm (see ref 40).

Let Q_k = period of one clock cycle.

Z = sum of the positive coefficients

$Y'eq$ = quantisation error of estimate $Y'e$

Then $$Y'eq = Z\, Q_k$$

The following results are obtained if we apply the above expression to the 3, 5, and 7 pt first derivative algorithms (see ref 40).

Estimate		Quantisation error	
3 point 1st derivative	1	x Q_k/h	1.00 x Q_k/h
5 " " "	14/12	"	1.18 "
7 " " "	912/720	"	1.27 "

The above results however are incorrect. These results would be right if the counter which is referenced to establish the "k_r" values was reset after every reading. The free running clock however gives an absolute timing reference which means less quantisation error. This absolute nature of the record is expressed only by the original statements for the estimates in terms of the "Y_r" values.

The correct quantisation errors from these original estimate equations is then given by-

Estimate		Quantisation error	
3 point 1st derivative	1/2	x Q_Y/h	0.50 x Q_Y/h
5 " " "	8/12	"	0.67 "
7 " " "	660/720	"	0.92 "

The quantisation error is directly proportional to the sampling frequency "1/h".

A.19 Coefficient Rounding errors

There are no details given here but to say that such errors, due to the finite word length of the coefficients in the algorithm may need to be considered when using longer algorithms because this is when they start to become more significant.

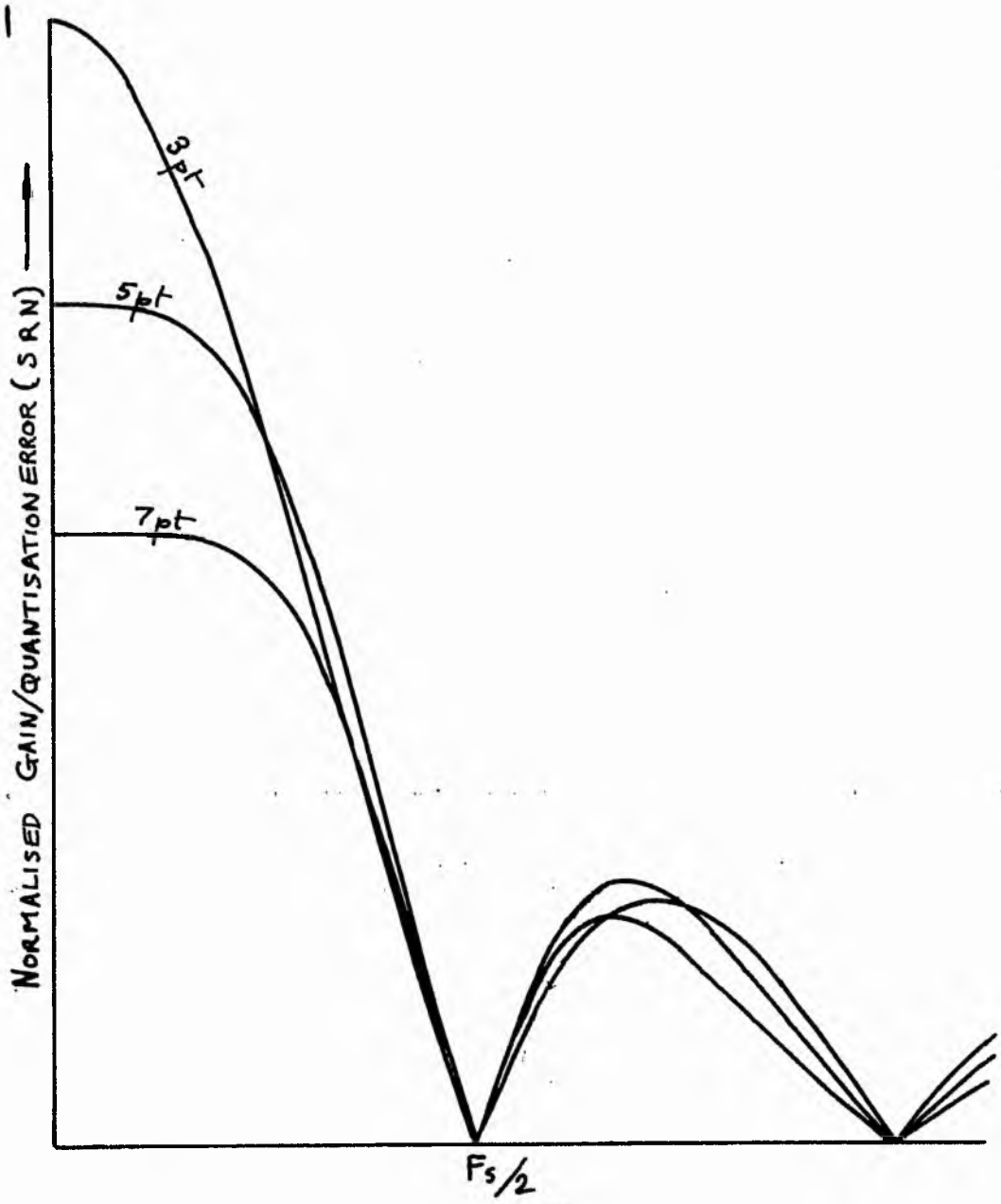
A.20 Signal to noise ratio

In this case the "SNR" for a given frequency is considered to be the ratio of the maximum quantisation error to the gain (actually this is considering the worst case noise situation which is not as useful as taking for instance the mean square value, however further work here could be included to do this). This is very useful when deciding on the choice of algorithm because the relative benefits of different orders of algorithms can now be compared on a single graph in the frequency domain. Figure 8 plots "SNR" against frequency for the three derivative estimators (used above). The best "SNR" for the higher frequencies is given by the high order filter. "SNR" for the low frequencies is better using a low order filter. In order to enhance the spectrum in the range close to sampling freqency divided by two the five point algorithm would be a good choice according to this analysis, although it is anticipated that if the "SNR" were based on the the mean square value of the noise then the seven point algorithm might well give a small improvement (for further information on "SNR" see ref 64).

Fig A.8 Frequency response graphs

Signal to noise ratio

Graphs illustrate the filtering effect of 3,5 and 7 point centre pivot velocity estimators on the actual velocity. In this case the effective gain is shown after quantisation noise is accounted for.



Frequency -------->
Fs is sampling frequency

A.21 Conclusions on encoder sampling

In the case of the grinding machine work there is unlikely to be a problem caused by aliasing of frequencies above Fs/2 because the power of any such elements appears to be falling almost exponentially with rising frequency and the estimation algorithms act as anti alias filters. The choice of algorithms for the estimation of velocity depends then on three conflicting factors

i) The higher order the better the frequencies response.

ii) " " " " more arithmetic operations needed.

iii)" " " " the more the quantisation error .

In the case of ii) the use of special purpose signal processing hardware or even a second processor will mean that there need not be a problem here. The assessment of the conflicting factors i) and ii) can be carried out by comparison of the signal to noise ratio of the various algorithms (ref 64).

The first part of this appendix demonstrates that within the velocity ranges of the grinding machine shafts the most appropriate data to record with the use of an optical shaft encoder is time sampled at equal angular increments.

The processing of this record into a constant time sample velocity (or postion record) is also demonstrated.

Program A.1

```
5  'File name "FILTER"
10 'This basic program plots the normalised frequency response
12 'of continuous time filters.
13 'The three filters here are the continuous equivalents of the
14 'digital centre pivot 3,5, and 7 point differential type
20 LPRINT "IN;SP 1;"
40 YA=6600
50 XA=10000
60 A=YA/6 :B=YA/90
61 LPRINT "PA ";YA;" ";XA;";"
62 LPRINT "PD;PA ";YA;" 0;PA 0 0;"
63 FOR S=1 TO 3
65 LPRINT "PU;PA 0 0;PD;"
70 FOR F=.05 TO 4*3.142 STEP .05
80 Y=F
90 H=F*XA/13
95 ON S GOSUB 150,160,170
100 LPRINT "PA ";YA-Z;" ";H;";"
120 NEXT F,S
130 LPRINT "PU;"
140 END
150 Z=ABS(SIN(Y))*YA/Y:RETURN:'          Centre pivot    3 Point
160 Z=ABS(8*SIN(Y)-SIN(2*Y))*A*.5/Y :RETURN:'            5 Point
170 Z=ABS(135*SIN(Y)-27*SIN(2*Y)+3*SIN(3*Y))*B*.5/Y:'    7 Point
```

Program A.2

```
5  'File name "FILTER2"
10 'This basic program plots the normalised frequency response
12 'of continuous time filters.
13 'The three filters here are the continuous equivalents of the
14 'digital centre pivot 3,5, and 7 point differential type
15 'In this second version of the program the gains are divided
16 'the quantisation error (for each type of filter) giving the
17 'signal to noise ratio.
20 LPRINT "IN;SP 1;"
40 YA=6600
50 XA=10000
60 A=YA/6 :B=YA/90
61 LPRINT "PA ";YA;" ";XA;";"
62 LPRINT "PD;PA ";YA;" 0;PA 0 0;"
63 FOR S=1 TO 3
65 LPRINT "PU;PA 0 0;PD;"
70 FOR F=.O5 TO 4*3.142 STEP .05
80 Y=F
90 H=F*XA/13
95 ON S GOSUB 150,160,170
100 LPRINT "PA ";YA-Z;" ";H;";"
120 NEXT F,S
130 LPRINT "PU;"
140 END
150 Z=ABS(SIN(Y))*YA/Y:RETURN:'          Centre pivot    3 Point
160 Z=ABS(8*SIN(Y)-SIN(2*Y))*A*.5/.67/Y :RETURN:'        5 Point
170 Z=ABS(135*SIN(Y)-27*SIN(2*Y)+3*SIN(3*Y))*B*.5/.92/Y:'7 Point
```

Program A.3

```
20 'This program plots the frequency response of Digital FIR
30 'filters on the Hitachi pen plotter.
40 'The gain is normalised {1 at 0 frequency}.  The coefficients
45 'of the filter are taken from data statements at the end of
48 'the program and may be changed as required.  The three
49 'filters given show the effects of estimating the velocity
50 'using different differentiating algorithms.
51 'They are 3,5 and 7 point centre pivot type.
60 YA=6600:XA=10000
70 LPRINT "IN;SP 1;" :LPRINT "PA ";YA;" 0;"
80 LPRINT "PA ";YA;" ";XA;";" :LPRINT "PD;"
100 DIM C(32)
110 FOR S=1 TO 3
115 LPRINT "PA ";YA;" 0;PA 0 0;"
117 LPRINT "PD ;"
125 LPRINT "LT ;"
130 READ C$
140 SC=0:M=0:LG=0
150 LS=LEN(C$)
160 N=(LS+1)/2:'No of coeffs.
170 CP=1
180 WHILE INSTR(1,C$,",")<>0
190 CP=INSTR(1,C$,",")
200 C(M)=VAL(LEFT$(C$,CP-1)):SC=SC+C(M)
210 M=M+1
220 LS=LS-CP
230 C$=RIGHT$(C$,LS)
235 PRINT C$
240 WEND
250 C(M)=VAL(C$):SC=SC+C(M)
270 REM Coefficients now in the array C(N)
280 ' M is the number of coefficients and SC is their sum
290 D=1.06*SC/YA:' D is a scaling factor for Y axis
295 'L is the Digital frequency {range 0 to 3pi}
300 FOR L=0 TO 12.57 STEP .09821
310 IF L>3.2 AND L<3.3 THEN LPRINT "LT 2 ;"
320 R=0:I=0
330 FOR K=0 TO M
340 R=R+C(K)*COS(K*L)
350 I=I+C(K)*SIN(K*L)
360 NEXT K
370 G=((R*R+I*I)^.5)/D
380 X=XA*L/13
400 LPRINT "PA ";6600-G;" ";X;";"
410 NEXT L
420 LPRINT "PU;PA ";YA;" ";X;";"
430 NEXT S
440 DATA "1,1"
450 DATA "-1,7,7,-1"
460 DATA "12,-96,444,444,-96,12"
470 END
```

Program A.4

```
/*****************************************************************/
/*      Encan  Incremental shaft encoder data analyser.          */
/*      Program  to process  absolute timing data record.        */
/*                                                               */
/*      Data originally   recorded  at equal angular intervals   */
/*      by saving  a free running eight bit counter value.       */
/*      Counter clocked by reference timing oscillator.          */
/*      Estimated values of velocity at equal time intervals     */
/*      are plotted . Sample time and other system constants     */
/*                                  entered by the user.         */
/*****************************************************************/

£include <graf.h>
£include <stdio.h>
£include <math.h>

£define Pi 3.1415
£define Reclen 2000

int P[Reclen];      /*Record of time at equal angular distances */

int A[2*Reclen];      /*Record of angular travel at const sampling

                       intervals (made longer than P[] just to
allow
                       sample rate extropolation       */
int Z;              /*Incremental angular travel over constant
                                        time sampling interval*/

main()
{
unsigned long f;  /*frequency c.p.s */

unsigned int N;   /*Length of angular position record A[]*/
unsigned int M;   /*Length of timing record P[] and c[]*/
unsigned int n;   /*element counter for A[] record    */
unsigned int m;   /*   "       "       "  P[]   "       */
unsigned int T;   /*Sample period in clock cycles     */

unsigned int encres; /*Number of pulses per rev. of enc.*/

int lostbs,lost;  /*variables for bytes lost due to overflow*/
int t;            /*Overflowing elapsed time in clock cycles*/

long e;   /*Non-overflowing elapsed time in clock
                                                        cycles */
long nT;          /*   n*T see n & T */

char c[Reclen];   /*Char array for untreated input data */
int d,prev;          /*Absolute difference variable for c[]*/

double sin();
```

```
float Vcts;
float V;          /*Velocity  (constant time sampling estimate)*/
float vel();      /*Velocity  (at angular position of argument)*/
float velAn;      /* vel(A[n]) used to avoid duplicate call to
                                                        vel()*/

float mhz;        /* Megahertz          cps/1E6            */
float a;          /* Constant factor in velocity estimation  */
float recdur;     /* Duration of record in seconds*/
float samp_freq;  /* Sample freq. in kilohertz */
float maxi,mini;
float autoscale;
float range;
float lastvel;

FILE *fp,*vp,*fopen();

int pt;
int x;    /* x axis graph variable */

printf("\x1B[2J");   /* clear screen and home cursor */

M = Reclen;       /* Length of record              */

printf("Enter clock frequency in Meg. Hz. - ");
scanf("%f",&mhz);

printf("\nEnter number of lost bytes  -");
scanf("%d",&lostbs);

printf("\nEnter encoder resolution -  ");
scanf("%u",&encres);

f= (unsigned long int)(mhz*1E6); /*frequency in cycles per sec */

lost = 256*lostbs;               /* total integer lost figure */


fp=fopen("data.dat","rb");

if ( fp != 0)
{
printf("\nData found\n");
}
printf("Number of bytes found %d\n",fread(c,1,Reclen,fp));
fclose(fp);

/****************    STAGE 1     ******************************
Transform char (byte) record to integer,  c[] -> P[] absolute
                      time  record
************************************************************/
x=1;
for (n=1;n<M;n++)
```

```
{
/* First find the difference c[n-1] to c[n] and form the     */
/* P[] record ( c[] -> P[] ) 16 bit record formed from 8 bit */
if (c[n-1]<c[n])
      {
      d= c[n]-c[n-1];
      }
 else                                  /* deal with overflow */
      {
      d= c[n]-c[n-1] + 256;
      }

      if(d < 50)
       {d = d + 256;}              /* tweek for overflow */

      prev = 5*(d-190);
   P[n] = P[n-1] + d + lost ;      /* add in any lost bytes and
                                          the difference */
}

/* show velocity record  if required */
/*for (n=1;n<M;n++)
   {
      V = (P[n+1] - P[n-1])/2;
      printf("%u,\t",P[n]);
   }
   getchar();
  */
/*********************   STAGE 2  ****************************/
/* Regular time sampling. Searching for the time intervals.    */
/*************************************************************

   First search P[] to find where the sample instants lie.
   Put the angular count int A[]

m is angular count integer
n is time sample number
e is elapsed time (at encoder clock pulse edges)
M is length of record constant

*/
  T=1;

while(T != 0)  /* Plot velocity at chosen sampling frequency */

    {              /*Tempory graph plot variables*/
     unsigned int curr,prev;          /* yaxis variables */

     float X,xstep;             /* xaxis position and increment */

    prev = 0;
    printf("\n enter sample freq in kilohertz  -  ");
    scanf("%f",&samp_freq);
```

```
    T = (mhz*1E3)/samp_freq;

n = 1;              /*start from sample 1 and time 0 */
e = 0;
nT= T;
printf("\n Constant time sample angular position counts \n");

for (m=1; m<M; m++ )
{
     e += (unsigned int)(P[m] - P[m-1]);

     while (e > nT)
      {
     /*  printf("{%d}\t",m); */ /* Display A[] if required */
      A[n++] = m;
         nT += T;
      }


}
N = n;


/* Now print out the record time length and the number of samples
                                                                  */

recdur = e/(1E6*mhz);       /* length of record in seconds */

printf("\nTotal time length of the record is   %5.3f seconds
\n",recdur);

printf("Number of samples is  %u\n",N);

printf("Average Velocity - %10.3f rads/sec
\n",(2*Pi*M)/(encres*recdur));

printf("Average frequency of data - %5.3f kilohertz
\n",M/(recdur*1E3));


/*********************  STAGE 3 ****************************/
/* Now so far we have found P[],A[] the length of A[] and the
total length of time of the record. Next step is to get regular
time  sample estimates of velocity
************************************************************/

   a = 4*(Pi/encres)*f;        /* this is a constant for velocity
                                                   calculation*/
   t = T;

   maxi = 0;     /* set up max and min velocities */
   mini = 1E6;
```

```
   vp = fopen("b:angvel.real","w+b");

   if(vp == 0) printf("Cannot open b:angvel.real\n");


for(n=1; n<N-1; n++ )
  {

      Z = (unsigned int)(((unsigned int)(A[n+1] - A[n-1])
                                        + 1)/2);
                            /*The +1 means that non integers from
                                        the division are rounded up */
         if(Z < 1)  Z = 1;  /* do not let Z go below 1 */

               printf("%d\b",Z);

            velAn = vel(A[n]);

                                  t +=T;
     /* Now perform simple interpolation for improved estimate */
     /*       of velocity at time nT                          */

   Vcts = a*(velAn + (vel(A[n]+1) - velAn)*(unsigned int)(t -
P[A[n]])
                              /(unsigned int)(P[A[n]+1] - P[A[n]]));
   if(n < 600)
       {
       if(Vcts < mini) mini = Vcts;  /* find max and min */
       if(Vcts > maxi) maxi = Vcts;
       }
   if (fwrite(&Vcts,sizeof(float),1,vp) !=1)
           printf("Writerror\n");

 }
 printf("Max velocity = %5.2f rads/sec\n",maxi);
 printf("Min velocity = %5.2f rads/sec\n",mini);


 if(fclose(vp) == -1) printf("Close error\n");


 getchar();  /*wait for keyboard input */
 getchar();

   vp = fopen("b:angvel.real","w+b");

   if(vp == 0) printf("Cannot open b:angvel.real\n");


/*********************************************************/
/*     Plot graph of constant time sampled velocity      */
/*********************************************************/
entergraf();
```

```
cls();          range = maxi - mini;
               autoscale = 512/range;
                lastvel = 0;

             for(n=1;n<750;n++)
                 {
                 if(n>N) break;

                 curr = (unsigned int)(a*V);

                  if(fread(&Vcts,sizeof(float),1,vp) != 1)
                      printf("Read error\n");


                 plot((int)( n ),autoscale*(lastvel-mini),
                      (int)(n+1),autoscale*(Vcts-mini));

                           lastvel = Vcts;
                  }
getchar();
exitgraf();

  }  /* here ends while loop*/

}
/* END OF MAIN */
/*****************************************************************/
/* START OF FUNCTIONS */
/**********************/
          /* Velocity at angular increment m */
float vel(m)
int m;
{         /* Function can be easily modified for more complex
                                                        estimator*/

          return((float)Z/(unsigned int)(P[m+Z] - P[m-Z]));


          /* Velocity at angular incremental position  m       */
          /* Units are (increments/clock cycle)        */
          /* Angular range of estimate is m-Z to m+Z (dist 2*Z)*/
 }
```

Appendix B

Encoder hardware and software interfacing

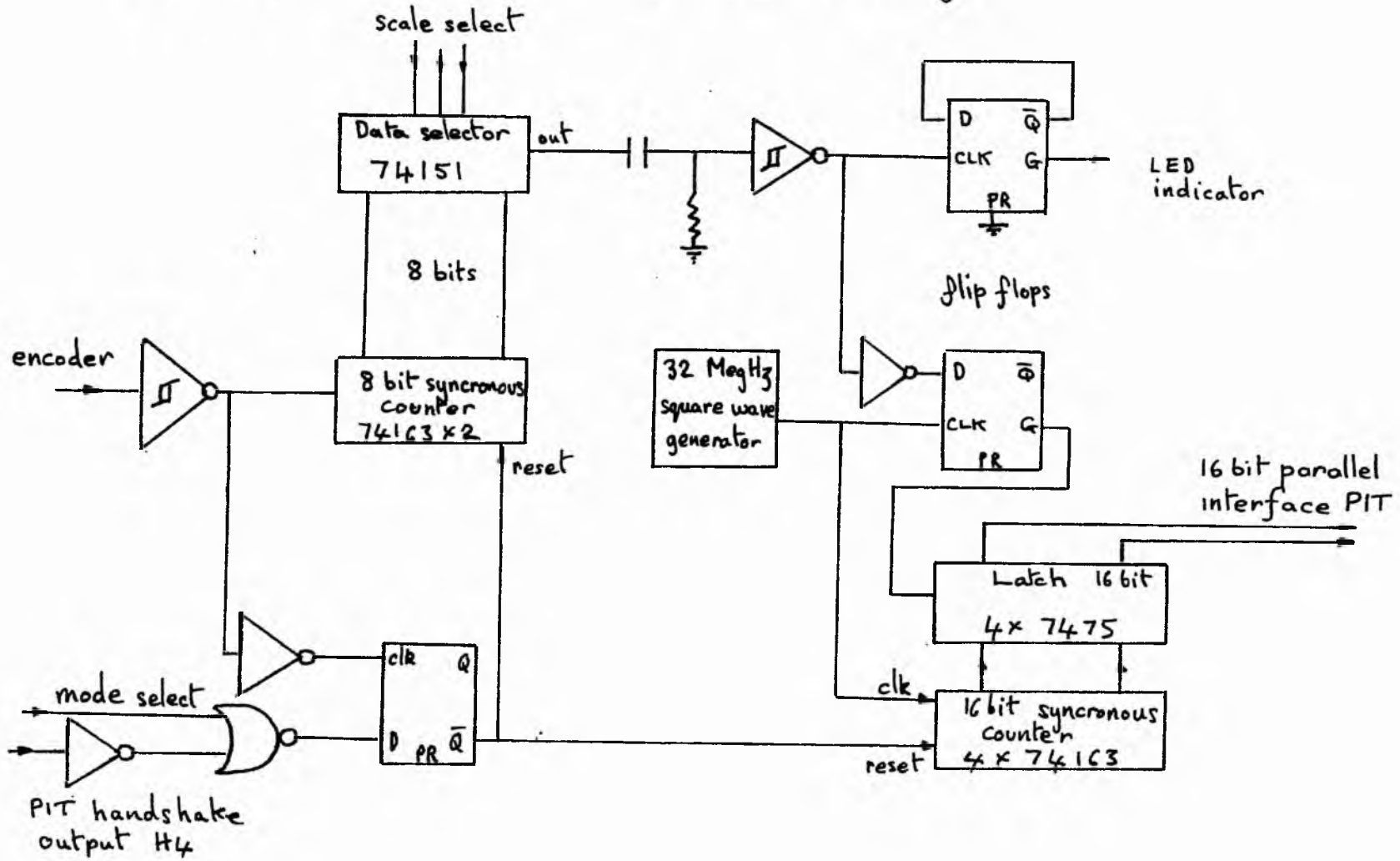
B.1 Dual function encoder interface

The experience gained from the development of the data logger (appendix E) was used in the design of the shaft encoder interfacing for the VME Versados system environment (Appendix C). An interface board was designed and built that would use sixteen bit synchronous counters and would allow sampling frequencies up to 20 kilo hertz (fig B.1). The interface board functions in two distinct modes as follows.

Mode 0 - constant period
Mode 1 - constant angle

Mode 1 is the method as described and used for the data logger (appendix E). A sixteen bit counter is set running continuously counting clock cycles from the reference oscillator. Pulses from the encoder indicating angular position are divided as programmed and the resulting pulses are used to latch and then log (store/save) current clock counter values after an interrupt is generated. Thus a record is formed of time sampled at equal angular intervals. IVEC (see programs at the end of this appendix) the assembler subroutine will collect such a record provided the interface has first been programmed for mode 1. IVEC loads a 128 element array with data. IVEC performs the first differencing which can be efficiently performed in assembler. IVEC2 is as IVEC but in addition there is a syncronising feature. The data logging only starts after a once rotational reference pulse is detected. Thus each array using IVEC2 is composed of timings between the same slots on the encoder. Mode 0 is still based on the interslot timing principle but adapted so that velocity estimates can be recorded at equal sampling time intervals.

Figure B.1 Dual mode single encoder interface



The interface is configured (see interface fig B.1) so that on reading a 16 bit port the counter which has previously been reset is set going when the next encoder pulse is received. The sequence for mode 0 is as follows. (note there are 2 counters one 8 and one 16 bit)

1. 16 bit read of interface port occurs.

2. Read forces both counters to be set ready for next encoder pulse.

3. Next encoder pulse arrives and triggers both 8 and 16 bit counters. The 16 bit counter starting from reset counts reference crystal oscillator pulses. The 8 bit counter starting from reset counts a programmed number of encoder output pulses.

4. The 8 bit counter reaches its programmed count and forces the 16 bit count to be latched and put onto the 16 bit output port.

5. Both counters are reset ready for the next read to occur.

The weaknesses of this method are as follows.

a) The 8 bit counter has to complete its set count before the next port read (set angular displacement must take place). This means a limit on the bottom speed. Also ideally the angular rotation controlled by the 8 bit counter should be as close as possible to the rotation in the sampling period to prevent the quantisation increasing and the sample only representing the average velocity over a fraction of the sampling period.

b) Resolution of encoder must be high enough so that at the highest angular velocity many slots must pass during the sampling interval.

B.2 Improvements to mode "0" sampling.

To avoid the above problems the following is suggested. Use two 16 bit counters that do not reset (see interface fig B.2). One counter should count timing pulses and the other should count encoder pulses. The timing reference count should be latched by the encoder pulses. Thus at the sampling event -

Current encoder count and last latched timing count are saved.

Let T = sample period.
n = integer variable.
Q = angle between slots on encoder (quantisation error)
rads.
P = reference timer clock period (quantisation error)
sec.
Let A0 = encoder count and C0 = timer count at time n*T
Let A1 = encoder count and C1 = timer count at time (n-1)*T
Let A2 = encoder count and C2 = timer count at time (n-2)*T

Now at time (n-1)*T the central difference estimate of the velocity will be -

velocity estimate at (n-1)*T = (A0 - A2)*Q/((C0 - C2)*P)
rads/sec *

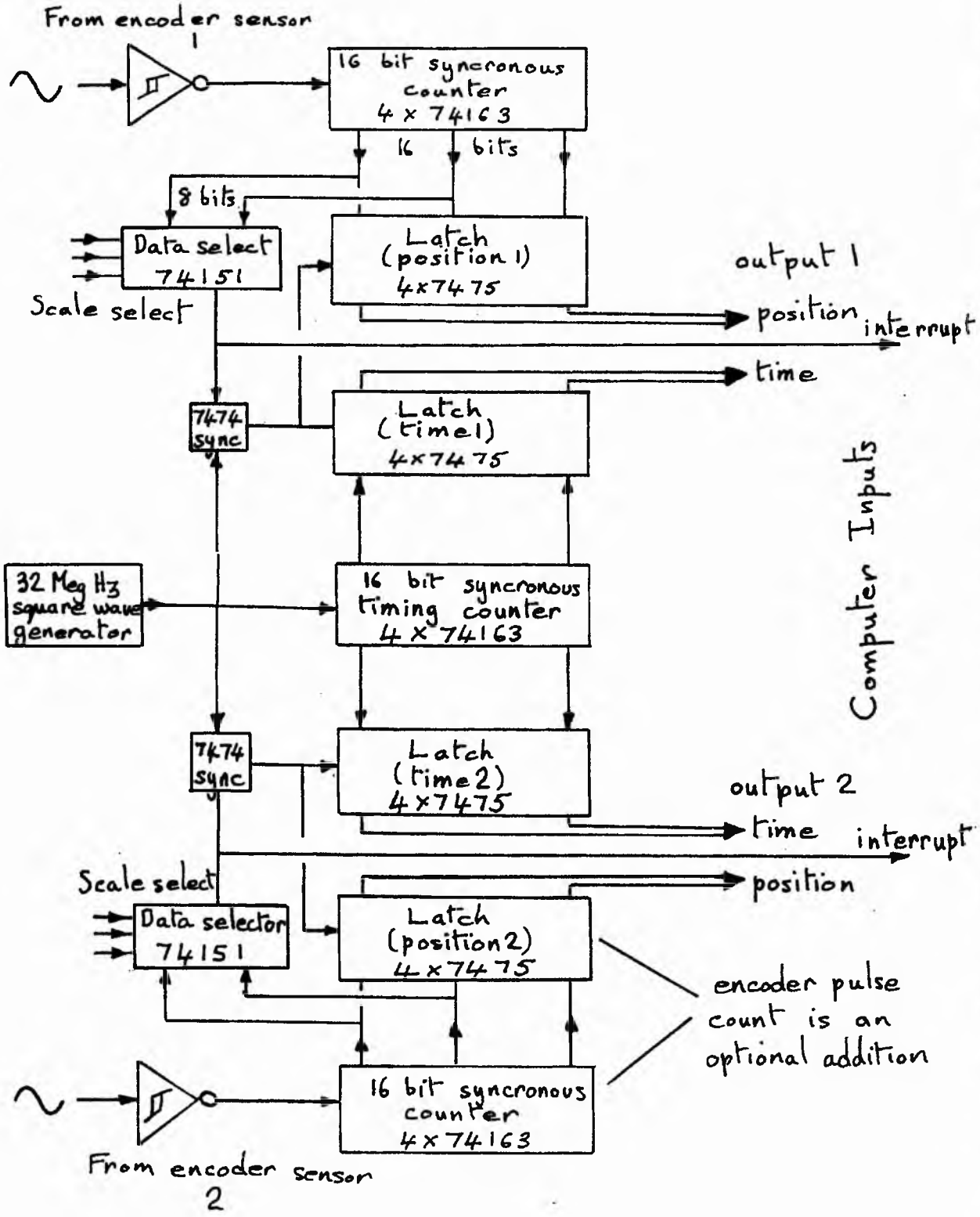
This should be contrasted with the simple estimate -

velocity estimate at (n-1)*T = (A0 - A2)*Q/2*T

This would be used if counting pulses from the encoder only. Note that a more comprehensive treatment of this method is given in Appendix A.

Figure B.2 Shaft encoder interface

Sixteen bit precision with position and time reference



B.3 Improved general purpose interface for multiple encoders

Figure B.2 shows the configuration of an interface for two encoders. Both timing counters share the same clock reference which can also be the source of a regular time sample interrupt. If sampling is only required at the lower frequency constant sample time then a constant sample time interrupt should trigger all latches ready for reading by the processor.

B.4 Computer programs involving encoder interface

1) IVEC Read encoder data mode 1

2) IVEC2 " " " " " synchronous

3) IVEL " " " mode 0

4) SETAPAL Sets up hardware interface

5) SATP sub program of setapal

6) SETP " "

SETAPAL - This is Pascal source for SETAPAL.LO the object of which is to allow simple user friendly programming of the encoder interface. The division of incoming encoder pulses may be set and also the mode. SETAPAL.LO is in the user '0' directory and so can be used in any other directory. SETAPAL calls SETP a Pascal subprogram which in turn calls SATP which is the assembler routine which sets the locations on the board.

B.4.1 Program

```
*****************************************************************
*        Ivec  -   Subroutine to be called from Pascal          *
*   (Used for const. angle sampling of the encoder timing)      *
*  Assumes  the start address of an array[-4..100] of word      *
*     has been passed on to the stack by using the  'var'       *
*     declaration in the forward procedure (or function )       *
*    statement. The two encoder counters do not reset in this   *
*   mode(1) and data is read at the 16 bit port 1 as soon       *
*                                    as it is valid.            *
*****************************************************************

IVEC        IDNT        1,1
            XDEF        IVEC
            SECTION     9
PIT1        EQU         $FF1000         *Base address of PIT port 1

IVEC        MOVE.L      (A7)+,A4        *Save return addr. in A4
            MOVE.L      (A7)+,A1        *Put base  address of array
                                        *         in A1
            MOVE.L      £104,D0         *Set loop counter
            MOVE.L      £PIT1,A2
LOOP        BTST        £7,$1B(A2)      *Test if H4 is set
                                        *    (data ready)
            BEQ         LOOP            *Go back to loop if data
                                        *     not ready(Z=1)
            MOVEP.W     $11(A2),D1      *16 bit read  port 1
            MOVE.W      D1,(A1)         *Put data on the stack
            SUB.W       D2,(A1)+        *Subtract the previous
                                        *  count from
            MOVE.W      D1,D2           *The present count
                                        * (result->array)
            CMP.L       £0,D0
            DBEQ        D0,LOOP
            JMP         (A4)            *Return to Pascal
            END
*
*Note -  Program may be developed to syncronize the start of
*recording to an encoder rev. signal.
```

B.4.2 Program IVEC2

```
*****************************************************************
*     Ivec2  -   Subroutine to be called from Pascal            *
*    (Used for const. angle sampling of the encoder timing)     *
*   Assumes  the start address of an array[0..1028] of word     *
*     has been passed on to the stack by using the  'var'       *
*   declaration in the forward procedure (or function )         *
*   statement   The two encoder counters do not reset in this   *
*  mode(1) and data is read only after the detection of the     *
*   pulse which occurs once per revolution. Thus this data      *
*   is aways synchronised to the same same angular position.    *
*****************************************************************

IVEC    IDNT        1,1
        XDEF        IVEC2
        SECTION     9
PIT1    EQU         $FF1000       *Base address of PIT port 1
PIT2    EQU         $FF1040       *Base address of PIT port2
IVEC2   MOVE.L      (A7)+,A4      *Save return addr. in A4
        MOVE.L      (A7)+,A1      *Put base address of array in
                                  *          A1
        MOVE.L      £260,D0       *Set loop counter
        MOVE.L      £PIT1,A2
        MOVE.L      £PIT2,A0
        CLR         D6
        MOVE.B      $13(A0),D6    *       Port 2B->D1
                                  *(Bit7 changes once per rev.)
        MOVE.B      $13(A0),D6
        MOVE.B      $13(A0),D6
WSTA    CMP.B       $13(A0),D6    *Test for change in bit 7
        BEQ         WSTA          *Test again if no change
LOOP    BTST        £7,$1B(A2)    *Test if H4 is set(data ready)
        BEQ         LOOP          *Go back to loop if data not
                                  *      ready(Z=1)
        MOVEP.W     $11(A2),D1    *16 bit read  port 1
        MOVE.W      D1,(A1)+      *Put data on the stack
        MOVE.W      D1,D2         *  the present count
                                  *   (result->array)
        CMP.L       £0,D0
        DBEQ        D0,LOOP
        JMP         (A4)          *Return to Pascal
        END
```

B.4.3 Program IVEL

```
*****************************************************************
*      Function for Pascal to read the sixteen bit timer        *
*      count latched at the last encoder clock edge             *
*****************************************************************
IVEL      IDNT     1,1
          SECTION  9
*         ASM      APAL
          XDEF     IVEL
PIT1      EQU      $FF1000        *Address of the periferal
*                                  interface and timer.
PIT2      EQU      $FF1040
IVEL      MOVE.L   (A7)+,A4
          MOVE.L   £PIT1,A2       *
*         CLR.L    D3
          MOVEP.W  $11(A2),D3     *read A port1 using
*                                  special 16 bit periferal
*                                  read. This is the time count
*                                  latched at the last encoder
*                                  clock pulse edge.
LAB       MOVE.L   D3,(A7)
          JMP      (A4)           *return to Pascal.
          END
```

B.4.4 Program setapal

```
*****************************************************************}
*      Program to configure the shaft encoder hardware          *
*         interface from the operators terminal.                *
*****************************************************************}
        program setapal(input,output);
            type
               word = -32768..32767;
               byte = -128..127;
                str = packed array[1..80] of char;
                var
                 DI : word;
               mode : byte;
 function  setp  (di  :word; mode:byte)   : str  ; forward;
             begin
 writeln(chr(27),'*');
 writeln('
     Shaft encoder interface initialisation');
 writeln('
Select encoder pulse division -> 1,2,4,8,16,32,64,128');
 readln(di);
 writeln('
Select sampling mode, 1 for const. angle  0 for const. period');
 readln(mode );
writeln(setp(di,mode));  { Setup the PIT and encoder board    }
             writeln;
             end.
```

B.4.5 Program SATP

```
*****************************************************************
*                          SATP                                 *
*  Task to set up the division rate on the encoder board        *
*   and fix the mode (periodic or continuous sampling)          *
*                                                               *
*  !! WARNING, REMEMBER TO PRESERVE A3,A5,A6 FOR PASCAL !!      *
*                                                               *
*****************************************************************

SATP        IDNT     1,1
            XDEF     SATP
            SECTION  9
PIT1        EQU      $FF1000
PIT2        EQU      $FF1040
SATP        MOVE.L   (A7)+,A4
            MOVE.L   £PIT1,A1
            MOVE.L   £PIT2,A2
*********************************
*      Setup Port 1             *
*********************************
            MOVE.B   £0,$7(A1)          *B data direction in
            MOVE.B   £0,$5(A1)          *A  "       "      "
            MOVE.B   £%01100000,$1(A1)  *PGCR
            MOVE.B   £%00110000,$F(A1)  *PBCR, smode00 16 bit
                                        *input,H4 handshake
                                        *output (interlocked)
*********************************
*      Setup Port 2             *
*********************************
            MOVE.B   £$00,$1(A2)        *gen. control reg.  Port 2
            MOVE.B   £$80,$F(A2)        *B control reg.        "
            MOVE.B   £$0F,$7(A2)        *B data dir.           "
            MOVE.B   £$FF,$5(A2)        *A data dir.           "
            MOVE.B   (A7)+,$13(A2)      *B data   out          "
                                        * the Pascal byte argument
                                        * is placed on port 2b
*********************************
            JMP      (A4)
            END
```

B.4.6 Program SETP

```
{*****************************************************************
*                                                                *
*                            SETP                                *
*      Pascal function to configure mode and pulse division      *
*      rate on the encoder interface board.                      *
*      Hardware is set by calling assembler routine SATP.        *
*      This just checks for valid input parameters               *
*          (returns error or conformation message)               *
*                                                                *
*****************************************************************}

     subprogram setp;
           type
              word = -32768..32767;
              byte = -128..127;
              Str  = packed array[1..80] of char;
 procedure satp  (pb2:byte);                    FORWARD;
 function  setp  (di:word;mode:byte): str;
            var
                  d : 1..128;
                  N : byte;
                  F : 0..1;
              begin
               d:=1;
               N:=0;
            while (di div d<>1) and (di mod d = 0) do
               begin
               d:=d*2;
               N:=N+1;
               end;
               if (di mod d = 0) and ((mode =1) or (mode=0))then

               begin
               satp(N+mode*8);
               setp:='OK';
               end
               else
               begin
                 setp:='Invalid   entry.   No   change   to encoder
board.';
               end;
               end.
```

Appendix C

C.1 Development System

The Development of the data logger was an important step in terms of initial data collection and validation of techniques for handling the shaft encoder signals. Valuable experience in real time system development was obtained. In particular, experience in the use of the H.P. 64000 real time emulation system (ref 70) made software and hardware debugging and development very efficient.

The problems encountered arose mainly from detailed tasks involved in designing a reliable disc operating system and working within the restrictions imposed by hardware (in particular the shortage of memory, and the speed of the MC6800)

The entire system code was purpose built and its development was costly in terms of research time and effort.

To overcome the problems of this kind of purpose built software and hardware it became necessary to look for a more generalised development environmemt, that would be capable not just of data logging but handling the entire proposed control and monitoring system.

The requirement was estimated to be -

a) Powerful 16 bit CPU
b) Large Memory, at least 250 k bytes ram
+Winchester disc +backup storage ie floppy discs
c) Real time multitasking operating system, with possibility for direct user control of I/O.
d) Analogue and digital I/O. Programmable timer.
e) Standard proven bus rack based card setup.
f) Development tools:
High level language compiler
Assembler and linker
Editor
Debug package (trace facility)
g) Possible use of graphics interface.
h) " " DSP (Digital signal processing) by using standard DSP co-processor board

The VME bus based system promoted by Motorala, Mostek, Sygnetics etc satisfies these requirements. This has the ability to transfer data at 24-Mbytes/s rates and handle all the signals for interrupts, arbitration of bus usage and provision for distributed intelligence. Modules are available from many different manufacturers, offering a wide choice of configurations.

C.2 VME Versados Development Kit

A survey was made of real time operating system software available for this equipment. This revealed that either Versados or OS9 68K would be suitable. Considering also availability and price it was decided to adopt the MVME 315 (Versados kit).

This included the following-

MVME 101 CPU Board
MVME 201 256 Kbyte RAM Board
MVME 315 Intelligent Floppy disc controller
with DMA and SASI Interface.
MVME 101 BUG Firmware for CPU Board (Ver. 3)
M68KOVDOS Versados (Real Time, Multi-Tasking
Multi User Operating System.

Unfortunately at least 384Kb of RAM is needed to allow the use of the assembler and high level languages. However the supplier subsequently agreed to change the 201 256k byte Board for a 202 512k byte Board, free of charge.

To complete the basic system two 1 Meg byte capacity floppy disc drives were purchased together with container and power supply (this was easily the cheapest way and high quality drives were ensured).

The following further additions were made

a) A Pascal Compiler
b) Digital I/O card with programmable timer.
c) Analogue I/O card with 7 microsec conversion time.

Standard VME digital signal processing boards exist (Burr-Brown VMEdsp ref 40)) with on board specialised cpu and software. This would allow the efficient concurrent application of-

1) Spectrum Analysis
2) Digital filtering
3) Correlation
4) Convolution
5) Matrix Inversion

The VME 101 board was not supplied with a printer port. A centronics compatible interface was specially constructed and fitted to the CPU board.

C.3 Configuration

This follows very closely the MVME 101 System Versados Hardware and software Configuration Manual. The change of memory caused only a small problem. The 202 - 512 kb memory board could only be located at an address boundary which corresponds to board capacity. This means that if the memory is to cover the sysgened* versados load address (10000hex), then the start address for the memory will have to be zero. Thus 12kb of 202 board RAM will duplicate memory on the CPU board. This conflict is resolved by the fact that the address decoder chip will direct all requests for memory to the on board static RAM and not put any addresses to the 202 board below address 3000 hex. This means that the system has effectively 512k RAM. (The possibility exists of mapping all memory to the 202 board thus avoiding the use of the 12k static ram on the cpu board.)

The I/O modules were put into the Short I/O address range for Global Devices (see memory map at end of appendix).

Problems encountered in setting up the Versados system-

a) Incorrect MAD101V address decoder PROM caused catastrophic system crash (had to be replaced by supplier).

b) Mitsubishi floppy disc drive hardware configuration not compatible with the intelligent disc controller. (modifications were required to internal jumpers).

c) Hard disc configuration parameters (cylinder, heads) different to those sysgened on supplied software. It was not feasible to modify the sysgen because this requires the hard disk. This was undertaken on the suppliers VME10.

d)Terminals available (VT100,VT220 & VT52) not compatible with Motorola editor.

e)RS232 transmission problems. Sysgened Xon/Xoff chars not standard for serial printers and Apricot. These are now modified by an IOS configuration command.

C.4 Versados and RMS68K

The advantages of adopting this operating system are enormous. The use of proven hardware and software right up to I/O boards means that far more of the valuable research time can be spent on new development. The operating system encourages a high degree of modularity in software development. Many independant tasks* can be run concurrently with virtually no interaction. On the other hand tasks may communicate with the executive or other tasks using a powerful set of operating sytem calls (see RMS68K manual).
Individual tasks may be developed and debugged (SYMBUG) before inclusion in a larger more complex system. Tasks may be assigned different priorities. As an example a task concerned with important real time control can take priority over a background task concerned with processing an FFT (Fast Fourier Transform) algorithm (ref 8). Tasks can also be assigned to respond as user interrupt service routines with a minimal overhead in terms of the executive.

The data logger developed in the first part of the research required the design and construction of a front panel (controls and display). There is no essential requirement for a purpose built front pannel with this type of development system.
Terminal control and display (facilities which come as a standard part of the operating system) offers extensive possiblities for man machine interface. Multi-screen output using extra terminal/s or graphics system (PLUTO) is easy.
(Graphics VME card can be added to the system at any time)

The system is clearly expandable so that there exists a straight forward upwards development path, which could possibly go as far as a distributed control and monitoring system covering many separate grinding machines .

The powerful builtin communications facilities of Versados mean that communication with other RS232 devices is straightforward.

Possible uses are.

a) Communication with a central computer monitoring many different workstations.

b) Input data such as that from post process measurement instrumentation.

The number of serial channels is at present only two but this may easily be extended by the addition of extra serial I/O boards. The system may by expanded to make use of the VMS bus for high speed serial communication (3.2 Mbits/s). The VMS bus is intended to handle the short, urgent messages typical of system control, timing and resource management.

One problem with the use of an operating system is that there is always an overhead on processor time due to the executive control functions. The important factor is that this over head be known and accounted for in any critical real time task. The latest version 4.5 of Versados now gives considerable savings in this overhead.

C.5 Applications of multi-tasking using RMS68K

Constant sample time activation of tasks

The control system requirement constant time sampling can be dealt with by the RMS68K periodic activation facility. This enables specified tasks to be started at regular intervals. The tasks can be put into the suspended state and restarted by periodic activation. Program "ACT" section C.11.1 is a sample Pascal function to simplfy the setting up of the periodic activation call to RMS68K. Program "START" section C.11.2 is a pascal function to suspend the calling task (the name START actually marks the point in the Pascal calling program where the program restarts).

Using RMS68K controlled interrupts

The real time executive "RMS68K" has the role of supervising user interrupts. The handling of these interrupts is demonstrated by three sample programs.

1) Program TISR section C.11.3 setups interrupt service routine and informs the executive.

2) Program SINT section C.11.5 configures timer chip to generate vectored hardware generated interrupts.

3) Program INTR section C.11.4 simulates an interrupt.

(programs at end of appendix)

TISR - uses RMS68K to configure an interupt.

TISR - also includes the interrupt routine. This should sit in memory waiting to be used. In order to use this demonstration task it is required that TISR be run in the background mode. This means using the @ operator (see Versados manual). Thus @TISR will first tell RMS68K that there will be an interrupt and

the level, and also the address of the interrupt service routine within TISR code. The program code for TISR then sits in memory in the dormant state ready for the interrupt.

INTR - simply simulates an interrupt using RMS68K

SINT - sets up the a timer on the "APAL" board to give out periodic interrupts.

The demonstration involves first configuring the APAL input output board then running @TISR followed by SINT. Then you should observe port lights counting interrupts.

The parallel io ports on the APAL can similarly be set to give an interrupt receipt of data read or write.

Multiprocessor applications

Additional processor boards on the VME bus can communicate with APAL gobal memory space. A write to the timer from an additional board may thus force a vectored interrupt on the master processor board. The facilities of the VME bus also allow the reverse configuration of the master interrupting the slave processor.
The parallel ports may also be used to force interrupts.

Example of using multi-tasking (development aid)

Program C.6 is a Pascal program that can be run as a background task with output to a secondary terminal. This uses periodic activation to repeatedly display the i.o. registers. This is helpful in the development of interfacing hardware and software.

C.6 Extensions to development environment

During the course of development using the original configuration of the VME based system, experience was gained with the use of Pascal and RMS68K. A special purpose encoder interface was designed and built in order to test the feasibility of a far more complex control and monitoring system (see section on encoder hardware design). From this work certain lessons were learnt regarding the development environment.

a) The development of programs at the higher language level "Pascal" and subsequent running as concurrent tasks (in the time slice sense) is an efficient way of producing code. The application requires the development or aquisition of certain specialised library functions which require some time to establish and test (matrix,graphics etc).

b) The Pascal must inevitably call assembler language functions and the experience to do this and develop the assembler code has been aquired.

c) The Pascal programs must be properly interfaced to the operating system. There were no established routines available for this purpose. This meant creating a library of functions such as "act()" a call to RMS68K to periodically start a task (used in constant time period sampling applications).

d) In terms of ability to control and communicate with the operating system the system potential was easily able to cope with the forseeable development.

c) The overiding problem encountered was the inability of the system to process the data at a sufficiently high rate for the practical real time application (infact the system could not aquire encoder data at, as fast a rate as, the special purpose data logger based on the old MC6802 8 bit processor).

This last problem is due to the requirement to receive data at such a high rate from the encoder. This data is required for vibration/noise analysis and is not essential for all the control algorithms which can run on a far lower fixed sample frequency.

The solution to this problem is to draw on the multi-processor capabilty of the VME bus.

C.7 Multi processor expansion

The application clearly requires more processing power. This may be achieved to an extent by choosing a faster or more powerful processor. Adding processors is a way which allows even further improvment to the total processing power but requires the means of interprocessor communication to be established.

In deciding the general configuration of a new more powerful system based on RMS68K and the VME bus, lessons from the first system were applied. In particular it was felt that to allow for multi-processor expansion the bus would need to be kept free of traffic except for essential interprocessor communication and that the new system should not carry the file handling transfers on the VME bus. Also the operating system should not (as in the old system) occupy VME address space.

The VME bus is essentially designed as a multi-processor bus. Each VME card may contain its own processor which is arranged to share a portion of VME address bus memory space with a combination of the other processors on the bus. The processors can all run asyncronously together provided there is the hardware in the form of a "system controller" which performs the necessary bus arbitration. There are various ways of organising the hierarchy of the processors (see VME manual). Here we are using the master/slave relation in which the main processor board containing the "system" controller will have control of the bus allowing slave processors to have the bus when the main board does not require it (slave boards are prioritised by their slot position in bus).

C.8 Improvements to development system hardware

The new system as follows was acquired -

a) MVME117 board with 68010 processor and floating point co-processor. This board included hardware for disk interfacing mapped off the VME bus (also a SCSI interface, 0.5 Megabyte local on board RAM, and a system controller.)

b) FORCE 68-1B VME processor board to act as the slave device.

c) 0.5 Megabyte of RAM for use as global memory.

d) Accessories (Winchester disc & controller, parallel and analogue I/O, nine slot VME back plane etc).

Having configured the system as above to run Versados and RMS68K the remaining problem was how to include the multi-processor features in a convenient and generalised way to pave the way for future development.

It was considered that desirable features for compatibility with the operating system would be as follows.

1) Where possible to allow tasks to be compatible between processors so that a task can be developed in the time slice Versados system and then run either as a Versados task or a slave task on the slave processor.

2) To allow interrupt driven interprocessor communication (ie master able to interrupt slave and slave able to interrupt master).

3) Facility for tasks to be passed into slave local memory in order to reduce VME traffic.

4) Slave firmware monitor program to allow basic interrupt responses such as starting a task, stopping a task, or grabbing a task from global into local RAM.

Much of the above requirements were met except for item 4). This meant that slave programs required starting from terminal control via the use of conventional monitor firmware bug.
The arrangement of interprocessor interrupts was by the use of the PIT chips on the parallel i/o board. The PIT can be programmed so that a write to a register can cause a vectored interrupt. This VME back pane can be configured with jumpers to directly connect the interrupt to any processor on the bus. Thus one register is set for the master and one for each slave. Direct board to board connection was not possible. An example task to set up the slave task is given in the programs section.

C.9 Development of Slave software and interfaces

To a degree this is possible using the development facilities of the Versados system. A basic requirement is that the slave input/output devices be mapped into global memory. The task program and data area must also be in global memory. The type of processor on the slave board must also be compatible with the master processor. Then the task can easily be run by either the slave or the master. The master processor will clearly not be able to dedicate as much time to the task as the slave since it has to service other tasks and also carries the executive overhead of the operating system.

There are clear advantages in organising slaves so that as much as possible they use their own local address space for program memory and input/output. If this is done then development of the slaves can still be efficiently carried out with the use of an in circuit microprocessor emulation system. In fact particularly in the debugging of assembler code input/output and in real time applications a good emulation system is argueably the most valuble and productive development aid.

Application of slave processor as dedicated shaft encoder analyser / input logger

The detail design of this system is also considered in Appendix A

Desirable features -

a) Potential for up to four shaft encoder hardware interfaces mapped into local address space.

b) Continuous logging of timing data at variable high frequency (of the order of 20 khz). Data to be logged into large circular buffers in local address space.

c) Noise cancelling filtering.

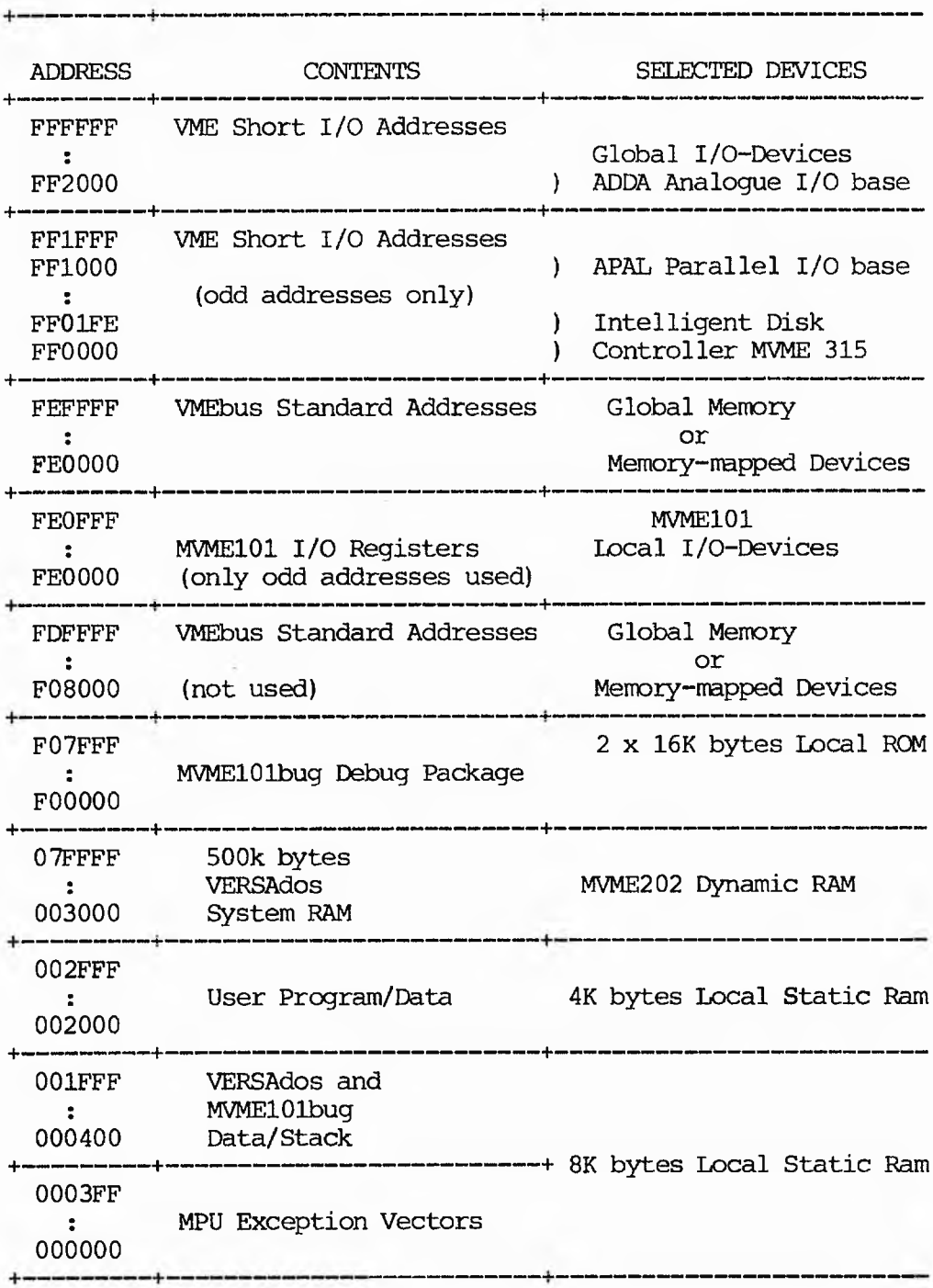
d) High performance regular time sample velocity estimates to be written into global ram circular buffer.

e) Facility to pass timing data to Digital Signal Processing board for two dimensional power spectrum analysis.

The velocity estimate buffer is available to the master processor via the global or shared ram. The slave also writes the current buffer pointer value into global ram.

Figure C.1

C.10 MEMORY MAP OF THE VME VERSADOS SYSTEM



C.11 Programs

C.11.1 Program ACT

```
*****************************************************************
*           RMS68K       function - for Pascal                  *
*        Request that the operating system periodically         *
*        activate the calling task. Period determined by        *
*        parameter (number of milli sec.) passed as a           *
*        function argument.                                     *
*****************************************************************
ACT             IDNT        0,0
                XDEF        ACT
                SECTION     9
ACT             MOVE.L      (A7)+,A4
                LEA.L       PIM(PC),A1
                MOVE.L      (A7)+,(A1)   *Take period from stack and
*                                         place in parameter block
                MOVE        £29,D0       * code for RQSTPA -> D0
                LEA         PRMBLK(PC),A0
                TRAP        £1           *EXECUTE RMS68K CALL(RQSTPA)
                MOVE.W      D0,(A7)
                JMP         (A4)         *Return to Pascal calling
*                                         program.
PRMBLK:         DC.L        0            *Task to be activated
*                                         0 means requesting task.
                DC.L        0
                DC.W        $4400        *0100 0100 0000 0000
*                                         flags see RMS68K manual
                DC.L        0
PIM             DS.L        1            *Period in milli sec.
*                                         to be stored here.
                DC.L        0
                DC.L        0
                END
```

C.11.2 Program START

```
*************************************************************
* A subroutine for a Pascal program.                        *
* Function is to call RMS68K to suspend the task            *
*           RMS68K - SUSPND                                 *
*************************************************************
START       IDNT        1,1
            XDEF        START
            SECTION     9
START       MOVE.L      (A7)+,A4     *save return address in A4
            MOVE.L      £17,D0
            TRAP        £1           *SUSPND task suspends itself
            JMP         (A4)         *return to Pascal
            END
```

C.11.3 Program TISR

```
*****************************************************************
*  Assembler  program, to set up an interrupt service routine *
*  Priority 5 , Vector No. $60                                  *
*****************************************************************

ISMOD       IDNT      0,0
            XDEF      GO
            SECTION   0
            DS.B      128           *Reserve stack space
STACK       DS.W      1
PRMBLK      DC.L      0             *Target is this task
            DC.L      0             *(Sess.)N/A for user task


            DC.W      0             *Allocate vector to ISR
            DC.B      0             *Reserved
            DC.B      $60           *Vector no.
SRA         DC.L      ISR           *Int. ser. address
            DC.L      0             *Argument in A1 when int.
                                    *      occurs.
ERR         DS.L      1
PIT1        EQU       $FF1000       *Base address of PIA
GO          LEA.L     STACK(PC),A7  *Initialise stack ptr.
            MOVE.L    £PIT1,A6      *Put PIA base into A6 for
                                    *    offset

            LEA.L     SRA(PC),A2    *3 lines to pop absolute
                                    *  address
            LEA.L     ISR(PC),A3    *   of ISR (int. ser.)
                                    * into the A3
            MOVE.L    A3,(A2)       *   parameter block for
                                    *   CISR  call
            MOVE.L    £61,D0        *Set up  for CISR
            LEA.L     PRMBLK(PC),A0
            TRAP      £1            *CISR
            BNE       FAULT(PC)     *Branch to FAULT if failed
            MOVE.B    £$0F,$13(A6)  *Turn on leds if success
            BRA       FAULT(PC)



FAULT       MOVE.B    D0,$11(A6)    *Display error code
            MOVE.L    £17,D0
            TRAP      £1            *SUSPND suspend self
```

Program TISR (continued)

```
*********************************************************
* Interrupt Service Routine   (flash leds on apal board) *
*********************************************************
*
FRED          DC.W       $AAAA
ISR           ROL        FRED
              ADDI.B     £1,PIT1+$13 *Turn on more leds
              MOVE.L     £0,D0
              MOVE.B     £$01,PIT1+$35 *CLR ZDS bit in TSR reg.
              TRAP       £1          *RTE return from exception.
              END  GO
```

C.11.4 Program INTR

```
*****************************************************
* TASK  INTR  - Simulates an interrupt (SINT) to be  *
*   serviced  at a routine within Task TISR.         *
*       Priority 7      -      Vector  $60  No.      *
*****************************************************

INTMOD        IDNT       0,0
              XDEF       GO
              SECTION    0
PRMBLK        DC.W       0              *Not used
              DC.B       6              *Priority level
              DC.B       $60            *Exception vector no.

PIT1          EQU        $FF1000

GO            MOVE.L     £PIT1,A6
              MOVE.L     £62,D0
              LEA.L      PRMBLK,A0
              TRAP       £1             *SINT  Interrupt.
              BNE        FAULT(PC)

              MOVE.B     £$F0,$11(A6)   *Indicate success
FIN           MOVE.L     £15,D0
              TRAP       £1             *TERM  Terminate self.

FAULT         MOVE.B     D0,$11(A6)     *Display the error code
              BRA        FIN

              END        GO
```

C.11.5 Program SINT

```
*****************************************************************
* This task Configures the PIT timers on the APAL I/O board     *
* Port 1 timer is set to give out periodic interrupts.          *
* Port 2 timer is set to give out a square wave on TOUT/C3      *
* The APAL jumpers JA3 & JB3 (int. req. & int. ack.) should be* 
* set to one of 7 interrupt lines (excluding 3).   The  same   *
* level should also be jumpered (K5) on the MVME101 board.      *
*         ( Level 6 interrupt  has been set.)                   *
*****************************************************************

CALLMOD   IDNT     0,0
*         SECTION  9
*         ASM      APAL
PIT1      EQU      $FF1000
PIT2      EQU      $FF1040
START     MOVE.L   £PIT1,A6
          MOVE.L   £PIT2,A7
          MOVE.B   £$00,$1(A6)        *gen. control reg.
          MOVE.B   £$80,$F(A6)        *B control reg.
          MOVE.B   £$FF,$7(A6)        *B data direction
**********************************
* Timer 1 control                *
**********************************

          MOVE.B   £$60,$23(A6)       *Timer1 int. vec. reg.
          MOVE.L   £$EEEE,D0
          MOVEP.L  D0,$25(A6)         *Preload
          MOVE.B   £$A0,$21(A6)       *Timer1 control
                                      * reg.(periodic ints.)
          MOVE.B   £$A1,$21(A6)       *Enable counter
**********************************
* Timer 2 control                *
**********************************
          MOVEP.L  D0,$25(A7)         *Preload
          MOVE.B   £$40,$21(A7)       *Timer2 control reg.
                                      * (square wave)
          MOVE.B   £$41,$21(A7)       *Enable counter
**********************************
          MOVE.L   £15,D0             *TERM Terminate self
          TRAP     £1
          END      START
```

C.11.6 Program SCAN

```
*********************************************************
* Pascal utility program to scan the parallel and       *
* analogue io boards. Can be run as a background task   *
* displaying io registers continuously on additional    *
*                    terminal                           *
*********************************************************

    program vos(input, output);
    type
      word = -32768..32767;
    var
    ER,CH: word;
     AV:-128..256;
     Z : string [300];
     TI: integer;
     PROCEDURE START; FORWARD;
     FUNCTION CHAN(CH:WORD) :WORD ; FORWARD;
     function act(TI:integer):word;forward;
      procedure con(X:integer);
    var
     T : array[0..7] of integer;     L : 0..15;
     R : 0..15;
    LB : string [4];
    RB : string [4];
    LH : string [1];
    RH : string [1];
     N : -1..3;
     A : 0..255;
 begin
   T[0]:=1;   T[1]:=2;   T[2]:=4;   T[3]:=8;
   T[4]:=16;  T[5]:=32;  T[6]:=64;  T[7]:=128;

    begin
      L:=0;       R:=0;
      LB:='';     RB:='';

    for N:= 7 downto 4 do
          begin
           if X >= T[N] then
              begin
                LB:=Concat(LB,'1');
                X:=X-T[N];
                L:=L+T[N-4]
              end
              else
                LB:=Concat(LB,'0');
              end;
          end;
          begin
             R:=X;
           for N:=3 downto 0 do
              begin
```

```
                if X>=T[N] then
                   begin
                      RB:=Concat(RB,'1');
                      X:=X-T[N];
                    end
                    else
                      RB:=Concat(RB,'0');
                    end
                end;
                if L<=9 then
                begin
                  LH:=chr(48+L);
                end
                else
                begin
                 LH:=chr(55+L);
                end;
                  if R<=9 then
                begin
                  RH:=chr(48+R);
                end
                else
                begin
                  RH:=chr(55+R);
                end;
          if (AV<32) or (AV>126) then
              begin
              AV:=32;
              end;
       Z:=Concat(chr(27),'i',Concat(LH,RH));
       Z:=Concat(Z,chr(27),'i',Concat(LB,' ',RB));
       Z:=Concat(Z,chr(27),'i',chr(AV));
       writeln(z);
        end;      {end of the procedure "con"}
      {************************************************************}
      {Start of Program vos}
     label
        10;
    type
       Byte = -128..127;
     var
          C : string [2];
          ACIB[origin 16£FF2000] :array [0..32] of integer;
          ACIA[origin 16£FF1000] :array [0..54] of Byte;
          S :array [0..15] of string[33];
          T : 0..100000;
          J : 0..15;
          V : word;
         RG : array[0..15] of integer;
      begin
          ER:=act(2000);
          C := chr(31);
        write(Concat(chr(27),'*')); {erase page and set curser to
home}
```

```
   write(chr(27),'&'); {set protect mode}

write('        ',chr(27),'1');
writeln('APAL-1   Register Dump',chr(27),'m');
writeln;
writeln(' ':31,'ADR    HEX     BINARY        ASCII');

writeln('Port General-Control-Register   1');
writeln('Port Service-Request-Register   3');
writeln('Port A Data-Direction-Register  5');
writeln('Port B Data-Direction-Register  7');
writeln('Port C Data-Direction-Register  9');
writeln('Port Interrupt-Vector-Register  B');
writeln('Port A Control-Register         D');
writeln('Port B Control-Register         F');
writeln('Port A Data Reg.               11');
writeln('Port B Data Reg.               13');
writeln('Port Status Reg.               1B');
writeln('Timer Control-Register         21');
writeln('Timer Interrupt-Vect.-Register 23');
writeln('.L Preload Register            25');
writeln('.L Count Register              2D');
writeln('Timer Status-Register          35');
writeln;
writeln(' Anologue input      AD0  volts =');
writeln('   "       "         AD1    "   "');
writeln('   "       "         AD2    "   "');
writeln('   "       "         AD3    "   "');
write(chr(27),'Y3 ');                 { end  inv. vid.}
write(chr(27),'k')  ;
write(chr(27),'Y"o');                 {start inv. vid.}
write(chr(27),'j')  ;


write(chr(27),'Y£E',chr(27),'1');   {set first tab}
write(chr(27),'Y£M',chr(27),'1');   {set sec.  tab}
write(chr(27),'Y£Z',chr(27),'1');   {set third tab}

   begin
      { start;}
RG[0] :=1;  RG[1] :=3;   RG[2] :=5;   RG[3] :=7;
RG[4] :=9;  RG[5] :=11;  RG[6] :=13;  RG[7] :=15;
RG[8] :=21; RG[9] :=23;  RG[10]:=27;  RG[11]:=33;
RG[12]:=35; RG[13]:=41;  RG[14]:=49;  RG[15]:=53;


    end;
10:


      write(chr(27),'H');       {set cursor home (no
erase)}
```

```
    for J:= 0 to 15 do
         begin

           AV:=ACIA[RG[J]];                    if AV< 0 then
                begin
                   AV:=256+AV;
                end
                else
                begin
                end;
          con(AV);
         begin
                end;
           end ;

       begin
            end;
     writeln;
        START;
      writeln(chr(27),'i',CHAN(2)*0.00486:5:3);
      writeln(chr(27),'i',CHAN(3)*0.00486:5:3);
      writeln(chr(27),'i',CHAN(0)*0.00486:5:3);
      writeln(chr(27),'i',CHAN(1)*0.00486:5:3);
  goto  10;
        end.
\CMD\
```

Appendix D

Non-linear relation of balance of grinding forces

D.1

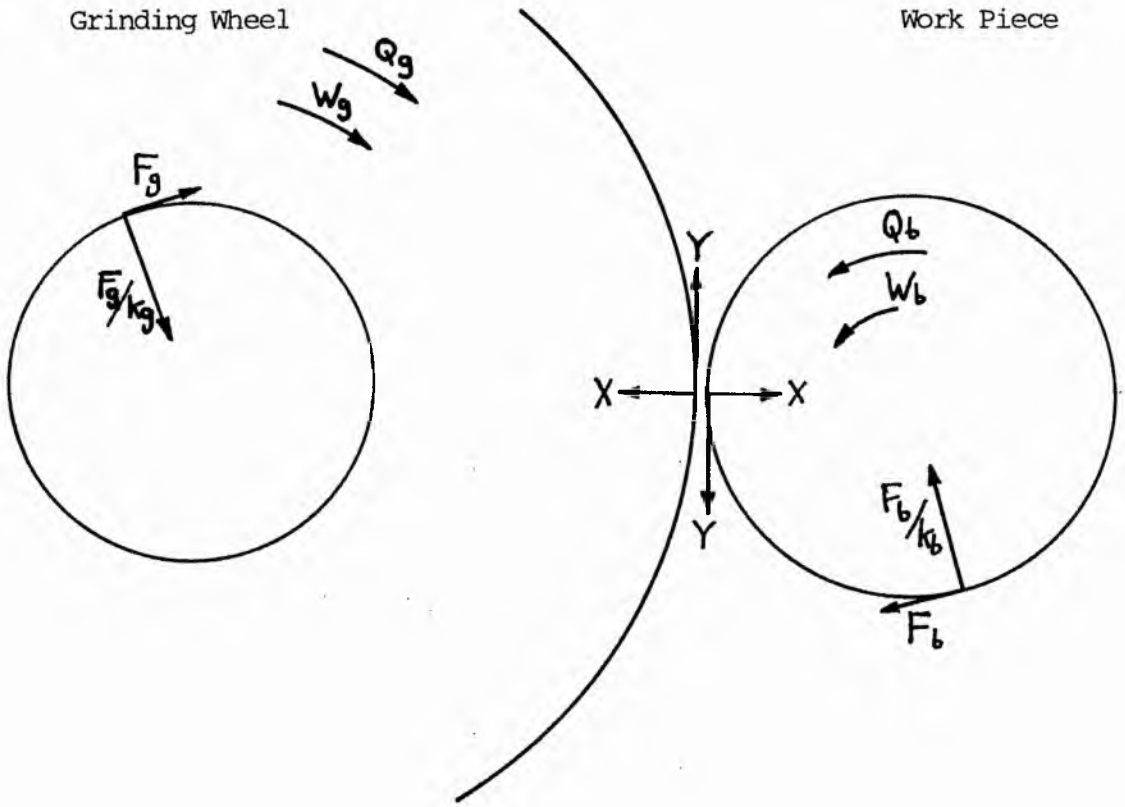
As the grinding wheel applies pressure to the work piece it also creates a reaction on the bearings. Thus the work done by each motor as a result of grinding includes the sum of the work done in grinding and the work done against the bearings.

The general system is described in chapter 2. Here we consider a schematic force diagram fig D.1. This diagram represents a simple balance of forces approximation. It is important to emphasize here that, by definition, the moments M_g and M_b are only the sums of the applied moments due to grinding and friction forces F_g,Y and F_b,Y respectively. At this stage the effect of frictional forces due to the work rotating offset from the drive shaft are not included. This can be included later but for now it is worth pointing out that such effects are likely to be fairly constant throughout the process. The result of the mathematics given the various approximations is as follows.

The disturbance inputs Q_b and Q_g to the two drive systems (fig D.1) are related by a non linear function which depends on the bearing frictional properties. This means that by analysing data of the two shaft speeds (given the transfer functions of velocity to disturbance torque are known, see appendix G) estimates of the resolution of forces at the wheel work contact point, and the dissipation of energy at that point can be obtained. However we will need to know the coefficient of friction of the bearings. The need to establish the friction is not considered to be a bad thing because details of this kind can only add to the possibilities for the overall condition monitoring of the system.
(The friction coefficient on the work support shoes is an important factor in determining the stability of the work during grinding. A complex matter in centre-less grinding, ref 35)

The frictional coefficients may be obtained by separate experimentation. They are likely to be a function of angular vel. but since this varies only a small amount this is not likely to be a problem. They will also be a non-linear function of the load, but for only small pertabations of load it is believed that the relation will be linear. In chapter 4 a method is described for the on line estimation of the workpiece support shoe friction.

Figure D.1 Balance of grinding forces



	Grinding Wheel	Work Piece
Angular velocity	W_g	W_b
Resultant torque	Q_g	Q_b
Tangential component of friction	F_g	F_b
Coefficient of friction	K_g	K_b
Radius	r_g	r_b

Horizontal component of grinding force	X
Vertical " " "	Y

D.2 Definitions

W_g Angular velocity of the grinding wheel.

W_b " " " work piece.

F_g Tangential component of friction on gr. wheel bearings due to reaction to Y and X.

F_b Tangential component of friction on work support shoes due to reaction to Y and X.

k_g Coefficient of friction of gr. wheel bearings

k_b " " work support shoes.

r_g Radius of the grinding wheel.

r_b " " work piece.

X Horizontal component of the grinding force.

Y Vertical " " " "

M_g Defined as $F_g r_g + Yr_g$

M_b " " $F_b r_b - Yr_b$

D.3 Grinding Force equations

See fig D.1 for illustration.

Balancing forces

$$F_g^2 (1 + k_g^{-2}) = F_b^2 (1 + k_b^{-2}) \qquad \text{----------(D1)}$$

$$" \quad = X^2 + Y^2 \qquad \text{----------(D2)}$$

By definition

$$(F_g + Y) r_g = M_g \qquad \text{----------(D3)}$$

and

$$(F_b - Y) r_b = M_b \qquad \text{----------(D4)}$$

To tidy up the expressions it is convenient at this stage to define two new variables. Q and h as follows -

$$Q_g = M_g / r_g \quad , \quad Q_b = M_b / r_b$$

and

$$h_g = (1 + k_g^{-2})^{1/2} \quad , \quad h_b = (1 + k_b^{-2})^{1/2}$$

Then eliminating F_g and F_b in (1) by using (3) and (4)

$$(Q_g - Y) h_g = (Q_b + Y) h_b$$

Therefore

$$Y = \frac{Q_g h_g - Q_b h_b}{h_g + h_b} \qquad \text{----------(D5)}$$

From (2) and (4)

$$X^2 = F_b^2 h_b^2 - Y^2$$

and

$$F_b = Q_b + Y$$

we obtain

$$X^2 = \frac{Q_b^2 h_b^2 (h_g^2 - 1) + 2Q_b Q_g (h_g^2 h_b^2 - h_g h_b) + Q_g^2 h_g^2 (h_b^2 - 1)}{(h_g + h_b)^2} \qquad \text{----------(D6)}$$

If two unequal sets of estimates for Q_g, Q_b and X are available then estimates may be determined for the the friction factors h_g and h_p.

If the coefficient of friction of the grinding wheel bearing can be assumed very small then (5) and (6) become greatly simplified--

$$Y = Q_g \qquad \text{----------(D7)}$$

and

$$X^2 = (Q_g + Q_b)^2 h_b^2 - Q_g^2 \qquad \text{----------(D8)}$$

Appendix E

Data Logger

E.1 Introduction

Having decided that the use of optical shaft encoders could provide a good source of data describing the dynamic behaviour of this system it was felt that it would be useful to fit an encoder on a grinding machine and gain some practical experience in the required interfacing and interpretation of the data acquired. In order to do this a data logger was built to record information from a shaft encoder running on a production machine on site at the sponsor company (R.H.P).

E.2 Objectives of data logger

a) To record a series of files (programmable length) that would contain timing data generated by an optical shaft encoder.

b) To allow for recording to be repeatedly triggered by some external event.

c) To allow files to be selectively transferred via RS232 for analysis.

The first version of the logger included the following features.

a) Ability to load byte data from PTM at a rate of up to 10Khz.

b) File length 60K- 10 files.

c) File handling system from front panel (files randomly selected for read or write).

d) Switch selectable division of encoder pulses (1 to 128).

e) Trigger isolation using filter and opto-coupler with isolated supply.

f) RS232 port for file transfer.

E.3 Displaying results.

A utility program was written for the BBC micro, with five functions.

i) To allow the receiving of RS232 signals and write the data into a block of memory (10k).

ii) Monitor- Dump hex values to the display.

iii) Process byte data (differencing etc.).

iv) Display graphically, processed data with facilities to scan up and down the memory data block.

v) To drive the Ink Jet printer in the graphics mode to produce
hard copy of the results (see copy of the program).

E.4 The Data Logger (Construction & operation) Version 1

Details of the design and construction of the first version of the logger follow. In the light of experience improvements were made which are documented under Version 2.

E.4.1 Data logger hardware

Storage medium -

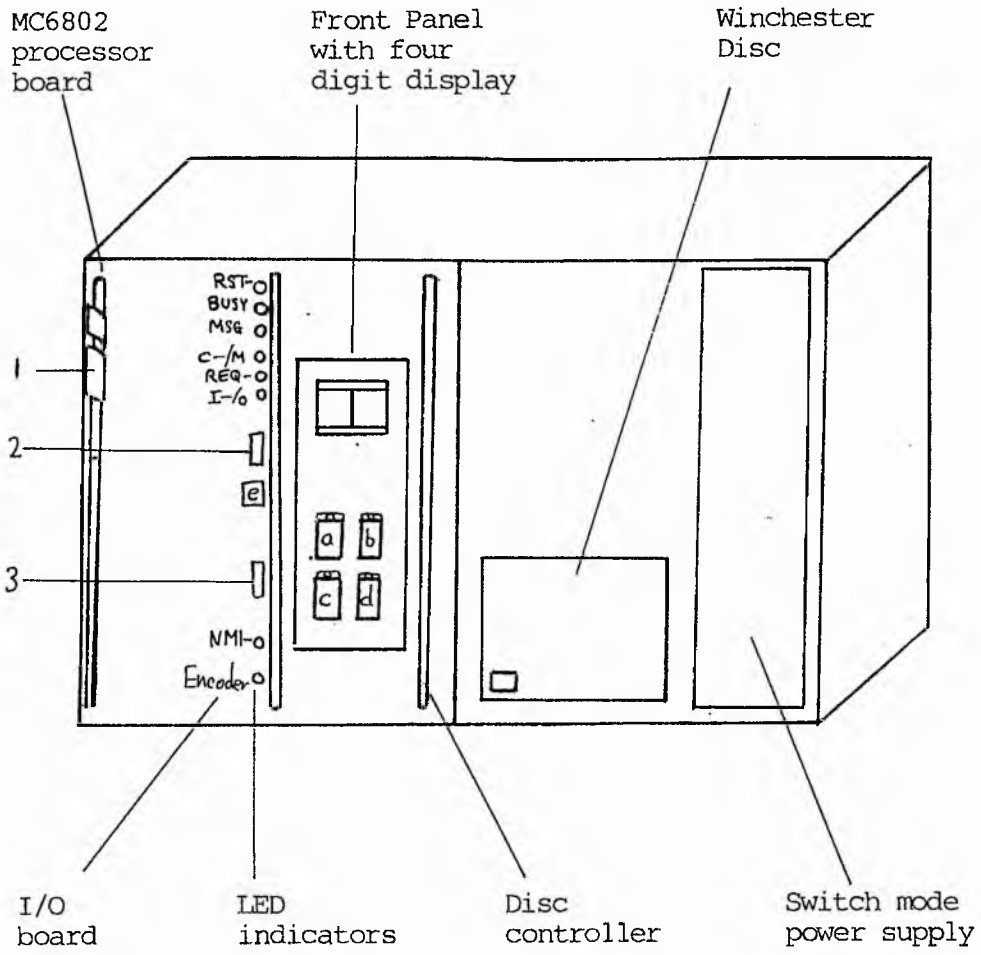
Winchester Disk - Seagate Technology ST406-Micro-Drive

Details--	5.25"	Dia.
Capacity	6.38	Megabytes Unformated.
	5	" Formated
	2	Heads - 1 disc- (tot. 612 tracks)
	32	Sectors per track.
	2.5	Megabytes per surface.
	5.0	M bits per second.

An XEBEC S1410 Disc Controller board was also available.

MC6802 general purpose industrial controller board. This was on a single bus mounted card and included four MC6820 (Peripheral interface adaptor PIA ref 51), two serial ports and a programmable timer chip MC6840 (Programmable timer module PTM).

Figure E.1 Data logger



Front panel Controls

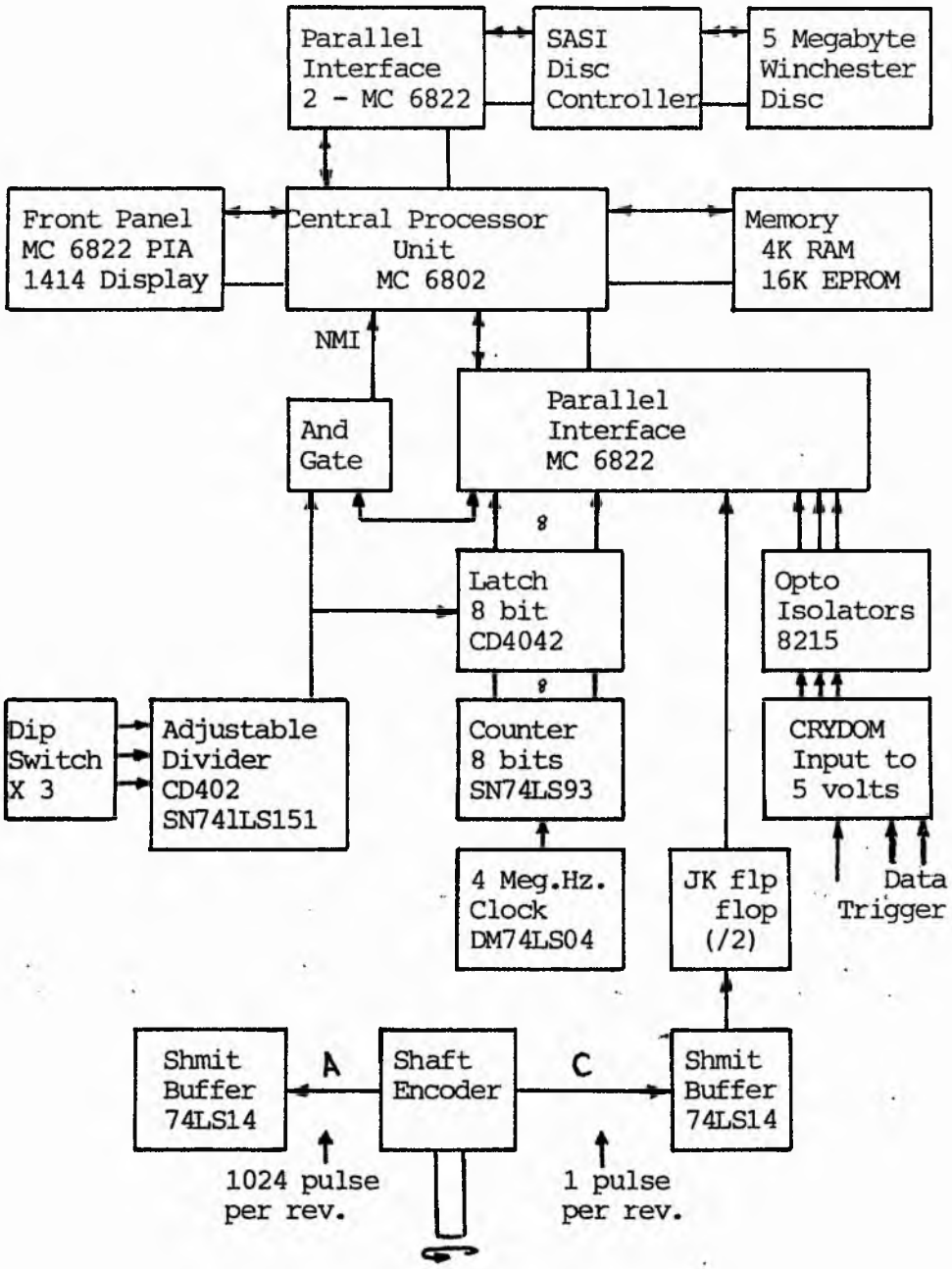
a) Reset
b) Select file number
c) Start recording *
d) Dump selected file to serial port
e) Trigger override button

* default, if no file selected, is tthe auto increment mode

Conectors

1) Serial port
2) Parallel input data
3) Shaft encoder input

Fig E.2 Data Logger General Circuit Configuration



E.4.2 The Disc Interface Signals

P.I.A. Disc Controller

8 lines Tri state <==========> DATAO- } Complement
<==========> " } of Data
<==========> " }
<==========> DATA7- }

SEL- low to select the drive --------->

ACK- low means data ready --------->
or data received

RST- low resets the drive --------->

(The following disc control outputs are open collector.)

<-------- BUSY- Drive is selected if low.

<-------- C-/D Command if low, data if high.

<-------- MSG- Command complete.

<-------- I-/O If low data is coming to disc.

<-------- REQ- Controller has received
data or has got data ready for host.

In all the circuits using these outputs were pulled up by 1K Ohm Resistors.

E.4.3 Hardware design of logger

The block diagram (fig E.2) illustrates the general layout and shows the disc/computer and encoder/computer interfaces which are of particular interest because they represent the special purpose hardware designed just for this project.

Disc interfacing

Central to this is the use of the MC6821 PIA (Peripheral interface adaptor chip ref 50).

There are many ways the PIA. can be programmed to achieve the required handshaking.

The PIA handshack pins (CA1 and CB1) handle those signals from the controller which indicate that :-

a) Data sent from disc is ready for the computer to read.

b) Data sent from computer has been read by the disc controller.

These signals do not exist directly. They have to be produced from the I-/O and REQ- signals (see section E.4.2).

The PIA may be programmed to react to negative or positive transitions on CA1 and CB1. The general response of the PIA to these inputs is to set a flag bit 7 of the relevant control register and to set the IRQ- output low if bit 0 (interrupt mask bit) is high. The flag bit 7 and IRQ- only reset (i.e. both high) after a periferal data registor read operation on the relevant port. The PIA has two eight bit parallel ports and these were both used. One port was set up for input and the other for output.

The pins CA2 and CB2 can be programmed as either inputs or outputs. In this application however there are only the three output modes of interest. They are:-

a) Direct control over CA2/CB2 by use of the control register

b) Pulse output (short fixed length low pulse produced after a read on A side or a write B side).

c) Set high by CA1/CB1, low by data Read/Write.

It was decided to use method c) because it allowed the interlocking of the outputs with the REQ-. Both method b) and c) only allow for port B being the output and A the input port.
The sequence of events for data out is then as follows (assuming control reg. B has been set with the correct code and neg. transitions on CBI were selected to be active).

i) Neg. edge on CBI sets interrupt flag bit 7 low, IRQ- goes low and CB2 goes high

ii) Interrupt flag is cleared by a data read periferal data reg.B (IRQ- goes high).

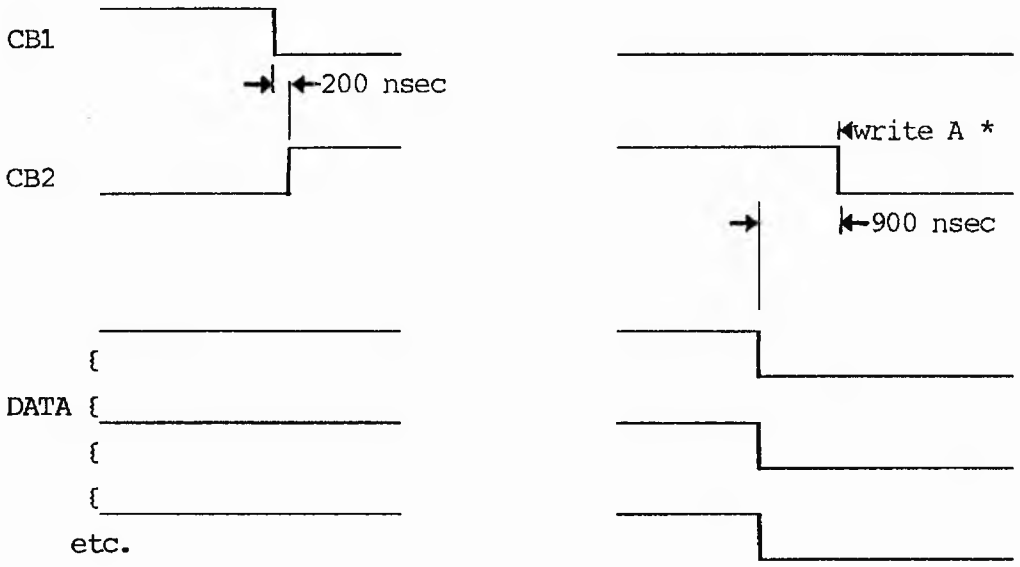
iii) Data Write to port B causes CB2 to go low.

E.4.4 PIA Control Register Settings (CRB and CRA the same)

Bit	7	6	5	4	3	2	1	0
Set	X	X	1	0	0	1	0	1
	don't care		handshake mode			select PRA	-ve edge	IRQ- mask

HEX value is 25

E.4.5 PIA timing sequence after CB1 input goes low



* Interrupt flag bit 7 must have been cleared by a read port A
See further timing diagram figures E.3,E.4.

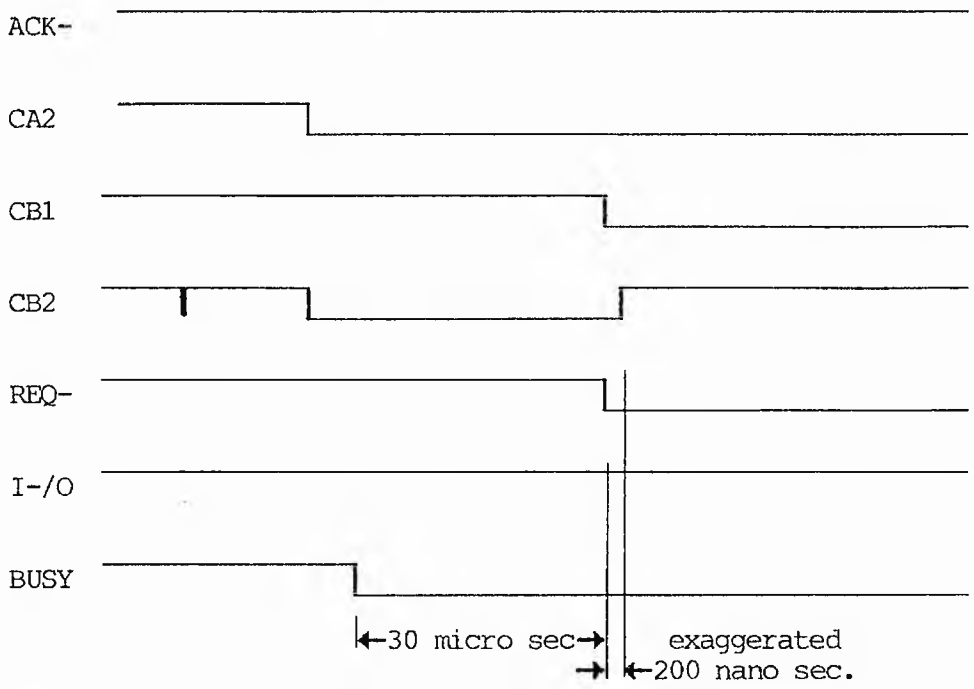
E.4.6 Adaption of the Disc Controller Signals

The normal duration of the REQ- signal was found to be about 200nsecs. This was found to be too short for the PIA to recognise. To solve this problem a pulse stretcher circuit was designed to give a minimum 1 microsecond on that line. With the chosen mode of operation of the PIA output data changes on port B, 900nsecs. prior to CB2 going low. The negative edge on CB2 signals data valid. This means that in order to produce the required ACK- signal, we need a circuit that will set an output low when given a negative edge from CB2 and set its output high if the level on REQ- goes high (see data out handshake diagram and circuit). The REC- signal was fed into a multiplexor controlled by I-/0. This made it possible to automatically direct REC- to the correct port.

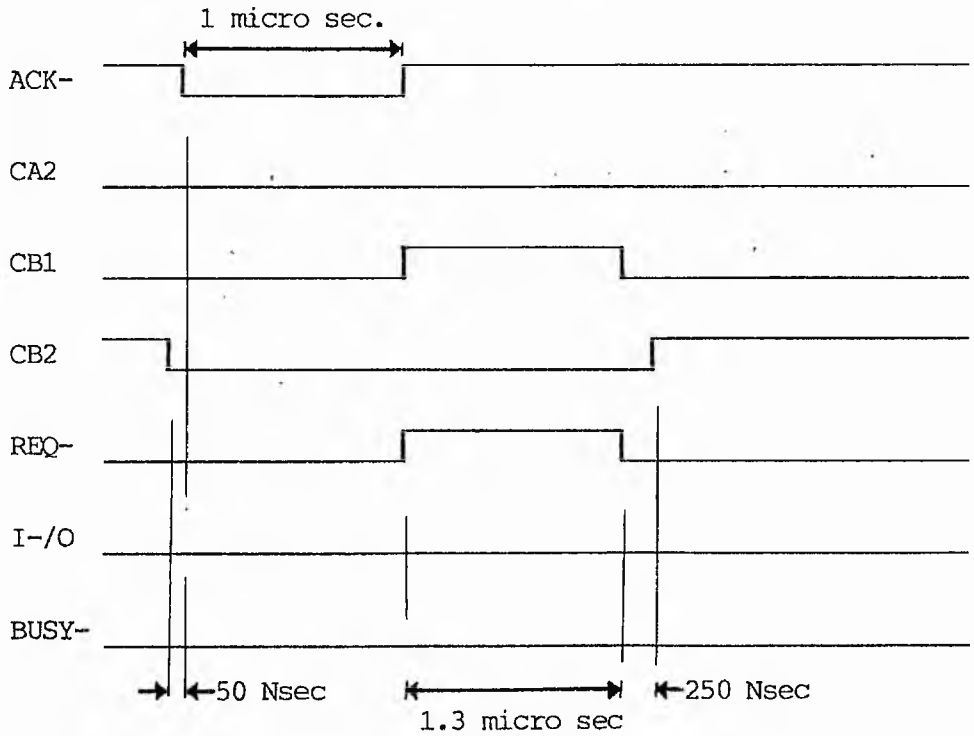
Figure E.3 Data Logger Winchester Disc interface

(Sasi controller to MC 6821 Parallel interface adaptor)

Selecting the drive — Logic analyser 10 u sec/clock, 20 u sec/division



Data sent to Disc handshake — 50 n sec./clock, 1 u sec./division



E.4.7 Prototype logger

The single board MC6802 processor computer, the disc and controller, a switch mode power supply and a special interface card were built into a KM6 rack system. This produced obvious benefits in terms of general reliability and convenience, especially since provision had been made to allow the easy connection of the emulation pod via a side hole. All interface lines of interest were connected in an orderly manner to the rear wire rap panel to allow easy use of the logic analyser. Previous work had successfully established the basis of communicating with the disc drive using IRQ- interrupt driven routines. This work was restricted however due to the shortage of parallel i/o. The MC6802 processor board was now set up with four PIA's and allowed the full disc control bus to be easily monitored using a second PIA. It was also possible to configure an output for RST-, an output to operate a switch to inhibit NMI- and an input to act as a trigger to syncronise the logger to an external event. A front panel was added to allow stand alone operation and crude file handling (see fig E.1). A line of LED's was added to give easy indication of control line status and encoder function. Configuring the ACIA for the RS232 was left until last because it was possible initially to write the files directly into the HP64000 emulation memory.

E.4.8 Logger Timing Principle

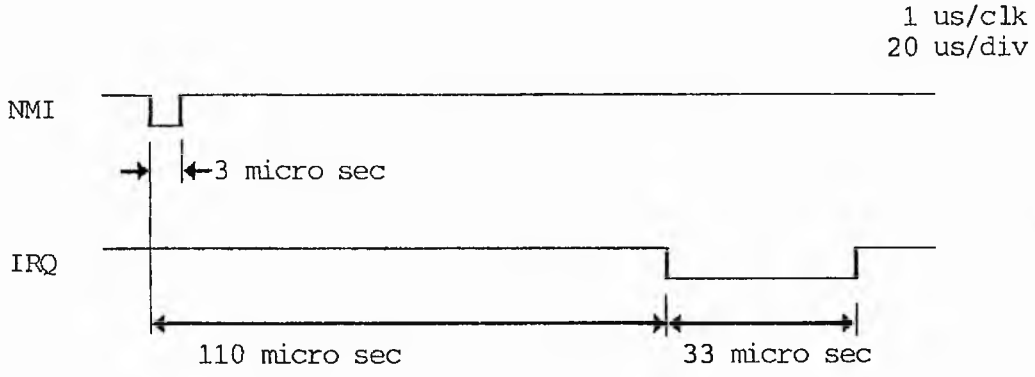
The timing reference was the system clock (1Mhz), although provision was made for the use of an 8Mhz clock (faster PTM needed). The PTM was programmed to operate in a free running mode, resetting to FFFF hex after each cycle. After triggering the system by an external event the NMI- (non-maskable interrupt) would become activated each time a pulse was received from the shaft encoder interface. This would cause the processor to read the PTM and subsequently store the data on disc.

Figure E.4 Data Logger Interrupts

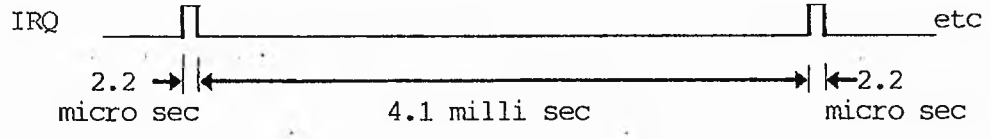
NMI - Non maskable interrupt active low.
IRQ - Interrupt request " ".

NMI is activated by the shaft encoder and the current timer count is saved in the buffer.

IRQ which eminates from the PIA (parallel interface adaptor) goes low when the data has been sent to the SASI controller and indicates that the controller is ready for the next byte.



Sending data via RS232



E.4.9 Buffer

Although the Disc controller board contained a full sector 256 byte data buffer, provision had to be made for the time between sectors/tracks. To allow for this a cyclic buffer (approx. 1.5k) was built into the software.

The processor loads the buffer using NMI- and unloads it in response to IRQ- (interrupt request)(see figures E.5,E.6). No pulses should be missed. If the shaft encoder however, gives out pulses too fast there will not be enough time to properly unload the buffer and it will overflow. This event is provided for in the software and the front panel responds with the error message "BUST".

As an example, if the shaft encoder. gives out 900 interrupts per rev. and the revolutions per minute is 600 then -

Interrupts per sec = 9000

Therefore there will be 1000,000/9000 or approx. 110 clocks per interrupt. Now there must be enough time to allow at least one NMI and more than one IRQ-. From the program running on the HP system the routines were measured for time. NMI- approx. 50 micro secs SWI- approx. 50 micro secs. The above case is clearly near the limit of the system.

Many bugs were found in the logger before it reached a reasonably reliable state. Debugging was performed with the aid of the HP 64000 in circuit emulation system.

E.5 Data Logger Version 2

See figure E.2 for schematic layout

E.5.1 Timing reference

In the first version of the logger an error is produced as a result of the uncertain response of NMI (because the mpu must complete the current instruction). The magnitude of error is between 0 and 4 micro seconds (the difference between the shortest and the longest instruction). It is apparent in the form of a quantisation like effect (a tendency for the values to fall into four micro second quantized levels). To solve this problem the version 2 logger was fitted with a separate 4 megaherz clock
reference counter and latch (fig E.2). The timer count is latched by the encoder reference pulse which also asserts the NMI. The software is re-organised so that the response to an NMI is now to read the data held by the latch.

E.5.2 Buffer modifications

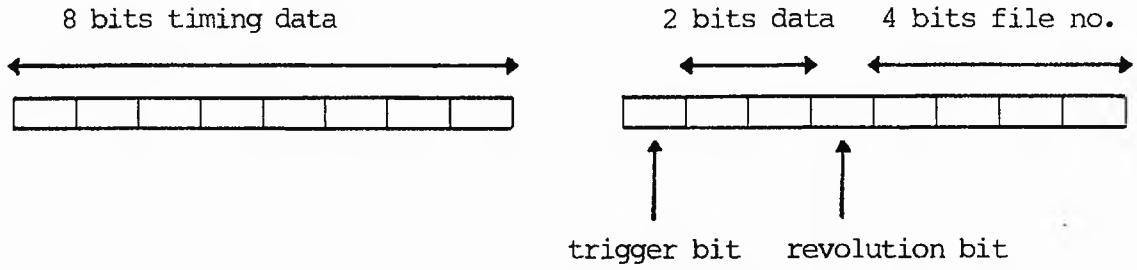
Software has been modified to enable the cyclic buffer to run continuously while awaiting a trigger event. The buffer read pointer follows the buffer write pointer around the buffer at a programmable lag (256 to 1024 bytes). This means that a short record of events prior to triggering are now saved. To indicate the buffer running but awaiting trigger event, the four button pannel led's turn on together with the disc busy light. Only when a trigger is received will the button led's go out and the data be sent to the disc.

E.5.3 Triggering modification

Version 1 triggering (start of file writing) is active for high or 5 volt level on the trigger input. This caused problems if the triggering signal was not reset prior to the end of the file. The software was changed so that triggering was by a positive going edge only.

E.5.4 New Data format

Version 1 logger used fixed length files composed of single bytes of timing data. Version 2 had data interleaved with the following format.



Thus every alternate byte contains the file number.

Bit 7 is the status of the trigger.

Bits 6 & 5 are available to record other machine status.

Bit 4 is the revolution bit. This changes status each revolution.

With this new format it is easy to check the correct functioning of the logger. The revolution bit should change status only when the exact number of bytes since the last change is equal to the number of encoder pulses per revolution.

E.5.5 BBC micro logger software version 2

Changes were made as follows

a) Routines were added to validate the data as described above (counting bytes per revolution).

b) Modifications were made to display the value of the file number byte data together with the timing data. This was done for both the colour graphic display and the ink jet printer.

c) As explained in appendix 1 further routines were added to modify the 8 bit record to a 16 bit record,the values being stored in a 16 bit word buffer. This buffer is large enough for the use of the various filters used (filters are changed simply by loading the required coefficient values).

Figure E.5 Data Logger Cyclic Buffer Design

Labels (pointers to locations of data)

Top of Buffer ----------TB
Bottom " ----------BB
Write into Buffer ------WPO
pointer
Read from Buffer -------RPO

Buffer input occurs after a NMI interrupt from the shaft encoder

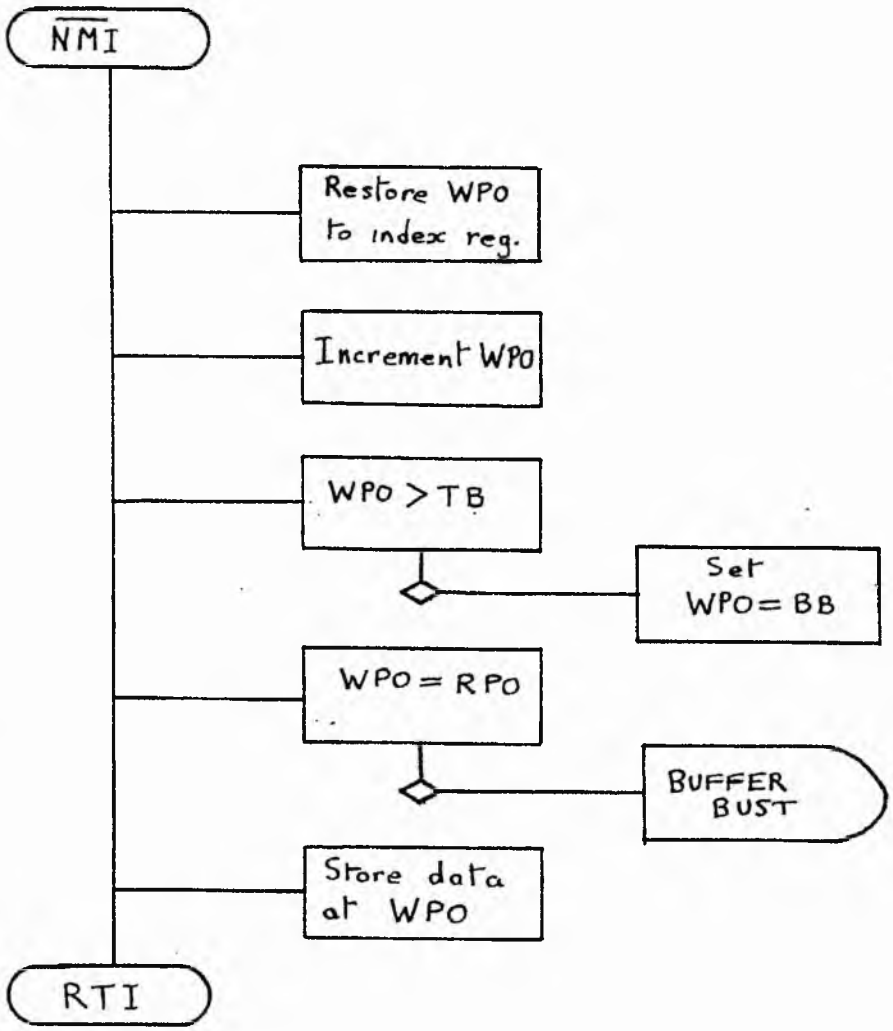


Figure E.6 Cyclic Buffer Output (Subroutine GETA)

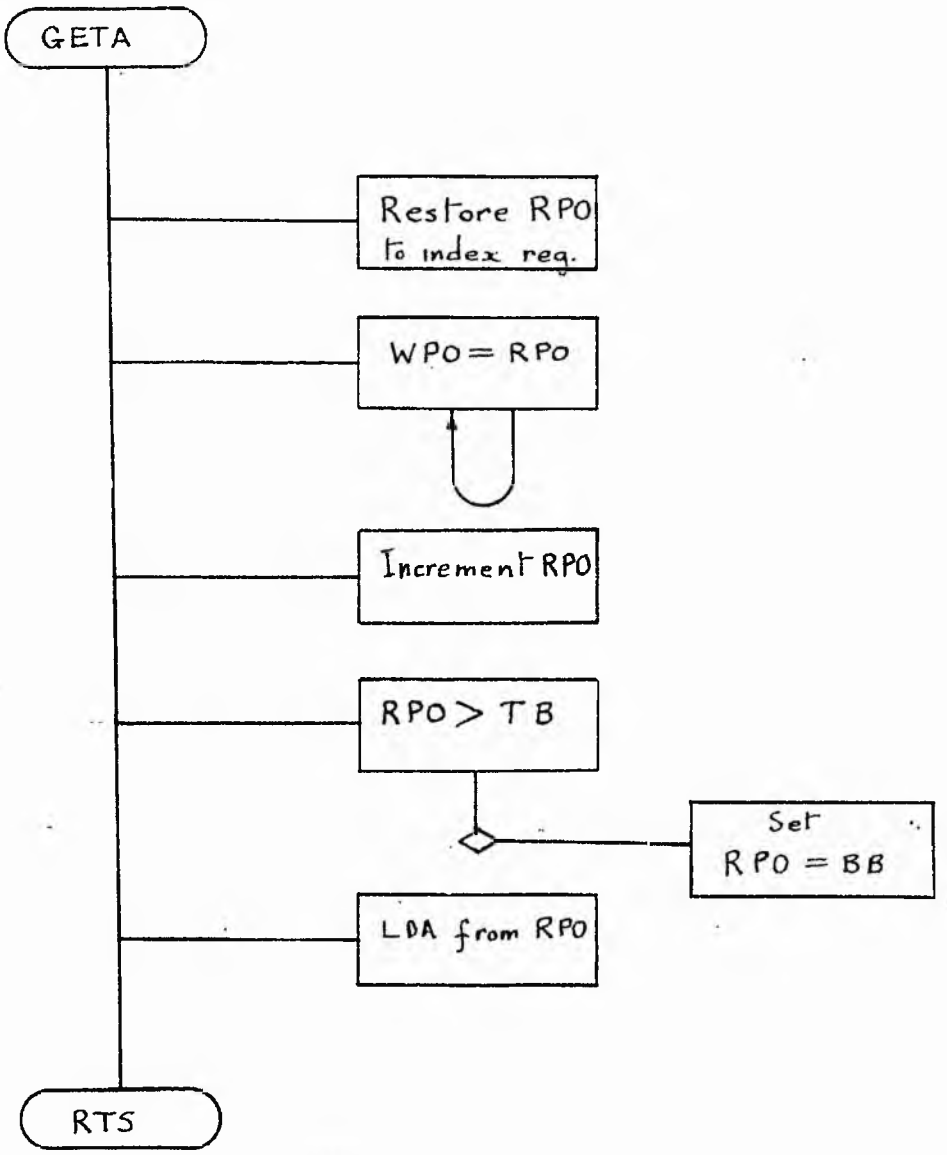
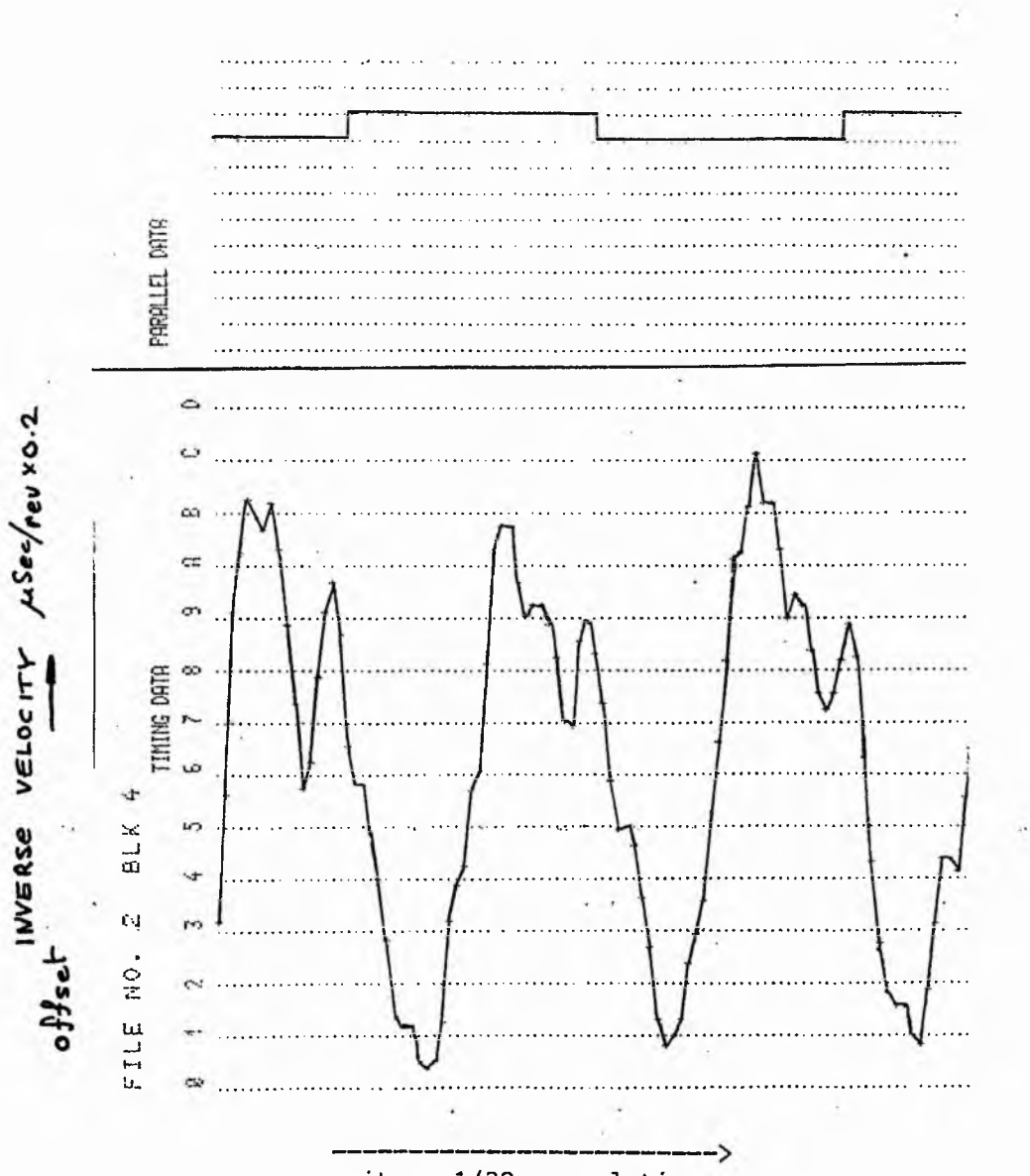


Fig E.7 Data logger record

Timing and parallel data

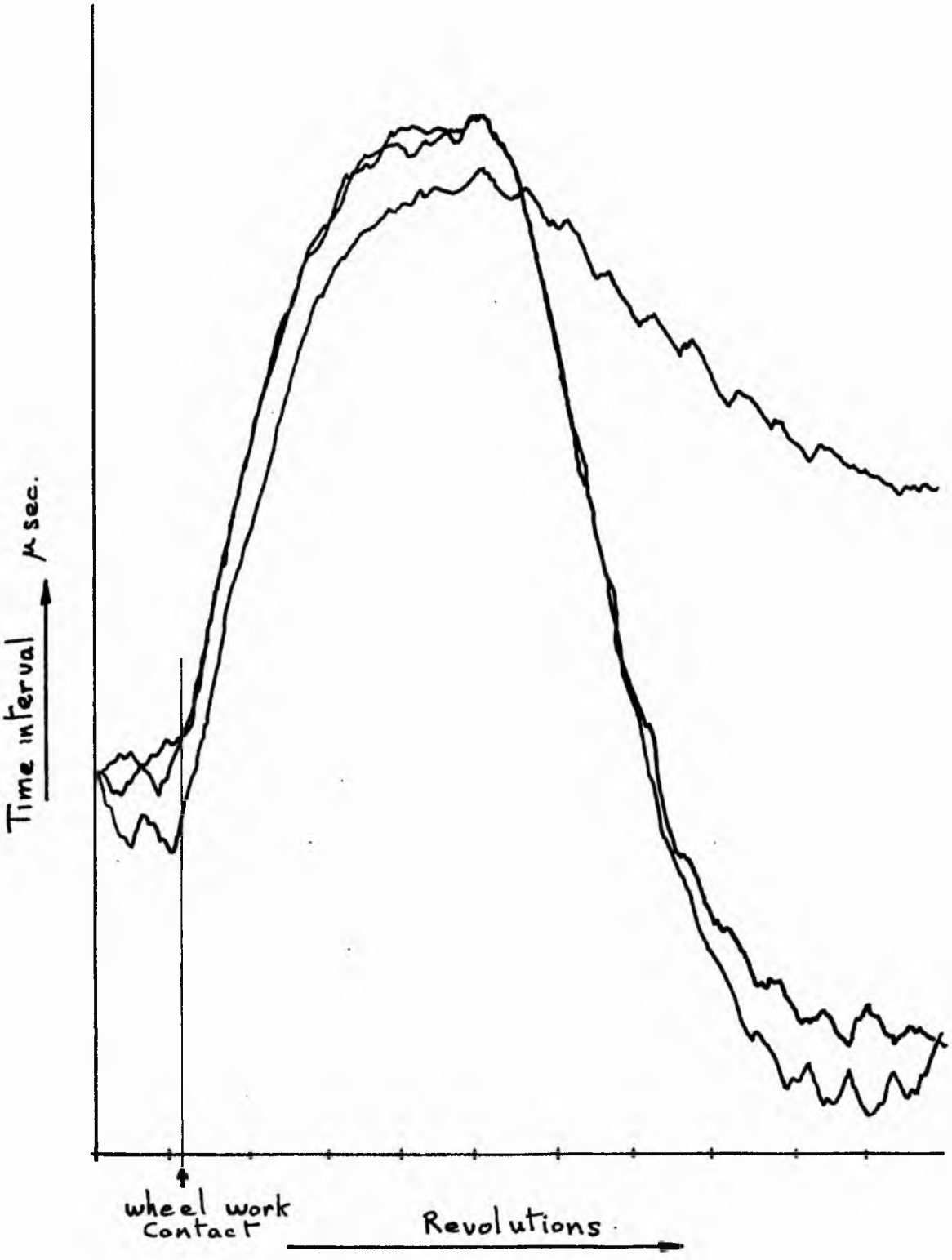
4 bit parallel data channel.
Distance from edge to edge indicates complete revolutions



Timing data is shown as first difference record

Figure E.8 Inverse velocity record

Period showing wheel work contact



```
 "6802"
*      THIS PROG. A/ CONFIGURES THE 6821 PIA
*                 B/ SELECTS THE DISC DRIVE
*                 C/ WRITES DATA FROM A TABLE CONTROLLER
*                 D/ READS DATA FROM CONTROLLER TO TABLE
*****************************************************************
*      DATA TRANSFERS ARE INTERRUPT DRIVEN BY THE REC- SIGNAL
*           FROM XEBEC S1410 DISK CONTROLLER.
*****************************************************************
*      PIA MEMORY LOCATIONS
*      PERIF. DATA REG.A/DATA DIR.    4COOH       DRA
*      CONTROL REG.A                  4C01H       CRA
*      PERIF.DATA REG.B/DATA DIR.     4C02H       DRB
*      CONTROL REG. B                 4C03H       CRB
*****************************************************************
DRA          EQU 4COOH    ;IIAL     input data
CRA          EQU DRA+1    ;IIAL
DRB          EQU DRA+2    ;IIA2     output data
CRB          EQU DRA+3    ;IIA2
DRAS         EQU 5000H    ;IIA3     control status
*bit - 7      6      5      4      3      2      1      0
*NMSWITCH   TRIG    I/O-   REQ-  C/D-    MSG-   BUSY-  RST
* See  disc drive notes for codes 0-5 . TRIG is signal
*  that starts disc  writes.
CRAS         EQU DRAS+1  ;IIAS
CPAN         EQU 5800H   ;IIA7 Control panel, 4 highbits LEDs
*                              4 low bits push buttons.
CPCR         EQU CPAN+1  ;IIA7
CEX          EQU 5400H   ;     External counter & latch data reg.
CEXCR        EQU CEX+1   ;        "       "     control   "
PLS          EQU CEX+2   ;     Control data from grinder +file no.
PLCR         EQU CEX+3   ;        "         control reg.
*****************************************************************
CD           EQU 4400H   ;0 dig. of display cd+1 is 1 etc.
ACIACR2      EQU 6000H   ;Terminal ACIA control reg.
ACIADR2      EQU ACIACR2+1 ;Terminal ACIA data reg
TB           EQU 7FFH           ;top of buffer     (7ffh for the
2K ram)
BB           EQU 200H          ; bottom  of buffer
*****************************************************************
*RESERVED LOCATIONS (flags pointers & data tables )
*****************************************************************
             ORG 154OH
TBS          RMB 2
BBS          RMB 2
BSIZE        RMB 2             ;buffer size
OPOIN        RMB 2               ;output to disc command pointer
IPOIN        RMB 2                 ;input from data    "       "
SPOIN        RMB 2                 ;control status     "       "
*****************************************************************
WPO          RMB 2             ;buffer write pointer
RPO          RMB 2             ;buffer read pointer
RRR          RMB 2
```

```
CDP         RMB 2           ;character display digit pointer
XR          RMB 2           ;   "      out table      "
FL          RMB 1           ;write flag
BFBU        RMB 1           ;buffer bust flag
TOFAST      RMB 1           ;max. R.P.M exceeded flag
KEYFL       RMB 1           ;wait for key up flag
FILN        RMB 1           ;current file no.
FROM        RMB 2           ;block trans. from pointer
TO          RMB 2           ;   "     "     to    "
BYTES       RMB 2           ;no. of bytes to be moved
WRRM        RMB 6           ;write command start
RERM        RMB 6           ;read command start
JT          RMB 12          ;start of jump table
SFLAG       RMB 1           ;cold start flag
RSFL        RMB 1           ;flag to divert data out to RS232
DIV         RMB 1           ;data division location
BC          RMB 1           ;byte counter for data out
TRFL        RMB 1           ;trigger flag, set to 1 by PIA
                            ;   trig. bit
************************
            ORG  OF800H
START:      SEI
            LDS £14FH       ;set up stack pointer
            LDS £NULL
            JSR ASC         ;clear the display
************************
            JSR SPTM        ;set up the PTM
*     INIT. BUFFER
IB:         LDX £BB
            STX WPO
            STX RPO
            CLR BFBU        ;clear buffer bust flag
************************
            CLR FL          ;clear the write flag
************************
            CLR FILN        ;reset the file pointer
************************
            CLR RSFL        ;reset the RS232 flag
************************
            CLR DIV
************************
*SET UP PIAs
************************
*GET BUFFER SIZE
            JSR CALBS
************************
            CLR CRA         ;set CRA for DDRA
            CLR CRB         ;set CRB for DDRB
            CLR CRAS        ;set CRAS for DDRA
            CLR CPCR        ;set CPCR for DD.
            CLR CEXCR       ;set CEXCR for DD.
            CLR PLCR
            CLR DRA         ;set DDRA to inputs
```

```
            CLR CEX             ;set  CEX to inputs
            LDA A £OFH
            STA A PLS           ; set PLS to half out half in
            LDA A £OFFH
            STA A SFLAG
            STA A DRB           ;set DDRB to outputs
            LDA A £81H
            STA A DRAS          ;set status port for inputs except
bit 1
            LDA A £OFOH
            STA A CPAN
            LDA A £025H         ;(0010 0101, see notes)
            STA A CRB           ;setCRB
            STA A CRA           ;set CRA
            LDA A £34H
            STA A CPCR          ;panel port control reg.
            STA A CRAS          ;set CRAS
            STA A CEXCR         ;ext. counter set to allow data
                                 read.
            STA A PLCR
            LDA A £1            ;reset disc
            STA A DRAS          ;
*****************************************************************
BFR         JSR SHIFT           ;move commands to ram
*****************************************************************
*    WAIT FOR KEYBOARD INPUT
*****************************************************************
            JSR DELAY
JK:         JSR KSCAN
            BRA JK
*       RESET THE DISC DRIVE (by putting a 1 in bit 1 of DRAS)
*****************************************************************
*       Note that on power up disc reset is active low, but it
*        appears to be necessary to do an additional reset to get
*         it to work.
*****************************************************************
DRSET:      CLR A
            CMP A SFLAG
            BEQ WARMST
            CLR DRAS        ;reset not active
            JSR DELAY
            INC DRAS        ;reset active
            JSR DELAY
            CLR DRAS        ;reset not active
*****************************************************************
SEND:        SEI
*         SET TABLE POINTER AT OPOIN ,IPOIN
*         AND INIT. DRIVE CHAR.
*****************************************************************
*   TEST FOR DRIVE READY
TAG         LDX £TDR
            STX OPION
            LDX £BB
```

```
             STX IPOIN
              JSR SEL
CHRES         LDX BB
              BNE TAG                ;test drive again
HALT:         LDX £ID
              STX OPOIN
              LDX £BB
              STX IPOIN
              JSR SEL
NOID          JSRCALIB
WARMST        LDX £RSET
              JSR ASC
              CLR FILN
              CLR RERM+2
              CLR WRRM+2

              LDX£AUT
              STX JT+7
REND          RTS
              NOP
              NOP
              NOP
*****************************************************************
* DISPLAY FILE NUMBER AND INCREMENT IT
*****************************************************************
FINC          LDX £W
              STX JT+7
              LDX £FILO
              JSR ASC
JINC          LDA A FILN
              STA A WRRM+2
              STA A RERM+2
              INC FILN
              JSR DIG
              CMP A £46H
              BNE FINCE
              CLR FILN
FINCE         RTS
*****************************************************************
* SUB TO GET FILE NO. FROM COMMAND AND DISPLAY IT
*****************************************************************
DIG           LDA A WRRM+            ;get file no.
              STA A PLS              ;put file no. in PIA
              ADD a £30H             ;convert to ASC
              CMP A £3AH
              BMI DISP
              ADD A £7
DISP          STA A CD
              RTS
*****************************************************************
* AUTOMATIC FILE INCREMENTING FILE WRITER (DATA FRM ENCODER)
*****************************************************************
AUT           LDX £AUTO
```

```
                JSR ASC                 ;display AUTO
NFL             JSR JINC
AUTIL           LDA A DRAS
                AND A £40H
                BNE AUTIL
AUTEST          JSR W                   ;inc the file number
                CLR A
                CMP A FILN
                BNE AUT                 ;continue if file is 0
                JSR DRSET
AUTE            RTS
*****************************************************************
* WRITE THE SHAFT ENC. DATA ON THE DISC (file number is FILN)
*****************************************************************
W:              JSR SPTM
                JSR RESBUF
                JSR WRT
                LDX ENDO
                JSR ASC                 ;display ENDO
                JSR DIG                 ;  "     file number
                RTS
*****************************************************************
* DUMP output data on RS232
*****************************************************************
DUMP        JSR CTCRES                  ;set up PTM
                JSR ACIARES             ;set up ACIA
                LDX £DMPO
                JSR ASC
                JSR DIG
                INC RSFL
                JSR READ
                CLR RSFL
                LDX £GOTO
                JSR ASC
                JSR DIG
                RTS
*****************************************************************
*               SELECT DISC
*****************************************************************
RESET           LDA A £1
                STA A DRB
                INC DRAS                ;reset active
                JSR DELAY
                DEC DRAS                ;reset not active
SEL             LDA A £2H               ;hard wire requires that bits
                                         1&0=10
                STA A DRB
CHBUSY          LDA A DRAS
                AND A £2                ;check for drive busy
                AND A £2
                BNE RESET
                CLR DRB
                CLR BFBU
```

```
TS          LDA A DRAS
            AND A £2            ;is the drive busy ?
            BEQ TS              ;go to TS if busy
ECH         CLR A
            CMP A FL
            BEQ TSE
            CMP A BFBU
BEQ TSE
            LDX £BUST
            JSR ASC
TSF         JMP TSF
SEI
TSE:        RTS                 ;end of SEL subroutine
*       CHECK THAT THE CRB FLAG IS SET AND CRA IS NOT
*****************************************************************
            LDA A CRA           ; A=CRA
            AND A 080H          ; Z=1      if flag set
            BEQ ANSET           ; continue from ANSET if CRA
                                  flag not set
            NOP
ANSET       LDA A CRB           ; A=CRB
            AND A £80H          ; flag set if Z=0
            BNE BSET            ; continue from BSET if CRB flag
not set
            NOP
BSET        NOP
*****************************************************************
*         WAIT IN LOOP
*****************************************************************
LOOP        LDA A CRB
            AND A £80H          ;check if int flag set
            BNE OUTAB           ;go to outab if set
            LDA A CRA
            AND A £80H          ;check if int A flag set
            BNE INTAB           ;goto intab if set
            BRA LOOP
*         INTERRUPT SERVICE
*****************************************************************
INTSERV:    LDA A CRB
            AND A £80H          ;check if interrupt flag set in
                                 CRB
            BNE OUTAB           ;go to OUTAB if set
INTAB:      LDA A DRA           ;read PDRA
            LDX IPOIN
            COM A               ;complement A
            LDA B RSFL
            BEQ TORAM
            DEC DIV             ;return from int. without
                                 putting data out
            BLE LD              ;return for a count stored in
                                 DIV.
            RTI
LD:         LDA B £0            ;set DIV for no. of missed data
```

```
                                    bytes.
                STA B DIV
WFP             LDA B ACIACR2       ;wait for the trans. buff. empty
flag
                AND B £2
                BEQ WFP
                STA A ACIADR2       ;put out serial byte
                RTI
*****************************************************************
*OLD CODE TO DUMP TO MEM IPOIN INSTEAD OF RS232
*****************************************************************
TORAM           STA A ,X            ;store PDRA
                INX
                STX IPOIN
                RTI
*****************************************************************
OUTAB;          LDA A DRAS          ;check the control bus
                AND A FL            ;is C/-D high ?
                BNE DOUT            ;if high go and send data

                LDX OPOIN           ;get pointer for output command
                LDA A ,X
                COM A               ;change A to its complement
                LDA B DRB           ;dummy read
                STA A DRB           ;send byte to the controller
                INX                 ;increment the pointer
                STX OPOIN
                LDX £BB             ;reset the write pointer so
                                     after command
               STX WPO                  ;-bytes are sent buffer is
initialised

                RTI                   ;(above will give trouble on
track changes)
*****************************************************************
* Checking for the neg edge of TRIG
*****************************************************************
DTD             LDA A DRAS          ;check for trigger event
               AND A £40H
                BEQ NGETA
                INC TRFL            ;set trigger flag if triggered
                CLR CPAN            ;turn panel leds off
                BRA DOUT
*****************************************************************
*****************************************************************
NGETA           LDA A WPO+1
                SUB A £OFFH
                STA A RPO+1
                LDA A WPO
                SBC A £00
                STA A RPO

                LDX RPO
```

```
                CPX £BB
                BGE XXX
                LDA A RPO+1
                ADD A BSIZE+1
                STA A RPO+1
                LDA A RPO
                ADC A BSIZE
                STA A RPO
XXX             LDX RPO
                STX RRR
                BRA DCH
                CPX WPO
                BEQ NGETA
                INX
                CPX £TB
                BLE NRC
                LDX £BB
NRC:            NOP
                STX RPO
                BRA DCH
*****************************************************************
*DOUT is the entry to the code that services the buffer checks
*  for trig. and sends data to disc after triggering.
*****************************************************************
DOUT            NOP
*****************************************************************
* Mod. to load next byte from buffer into accum. A
*****************************************************************

GETA:           LDX RPO
BA:             CPX WPO
                BEQ GETA
                INX
                CPX £TB
                BLE RC
                LDX £BB
RC:             LDA A ,X
                STX RPO
*****************************************************************
NSB             COMA
DCH             LDA B TRFL              ;branch back if trigger has not
been tripped
                BEQ DTD
                LDA B DRB               ;dummy read clears IRQ flag
                STA A DRB               ;output A to the disc
IRQE;          RTI

*****************************************************************
*      DELAY SUBROUTINE
*****************************************************************
DELAY:          LDA A £08H
DY              DEC A
                BNE DY
```

```
            RTS
*****************************************************************
*  CALIBRATE SUB
*****************************************************************
CALIB:      LDX £CAL              ;get the start of 5 byte
                                   calibration code
            STX OPOIN
            LDX £BB
            STX IPOIN
            JSR SEL
            LDX £RSS
            STX OPOIN
            JSR SEL
            RTS
*****************************************************************
* READ SUB
*****************************************************************
READ:       LDX £RERM
            STX OPOIN
            LDX £BB
            STX IPOIN
            JSR SEL
            RTS
*****************************************************************
*WRITE SUB
*****************************************************************
WRT         LDA A £OFOH
            STA A CPPAN
            CLR TRFL
            LDA A £80H
            STA A DRAS
            LDX £WRRM
            STX OPOIN             ;set the command data pointer
                                   for write
            LDA A £8
SPOT        STA A FL               ;set flag to show data is to
come from encoder
            JSR SEL
            CLR FL                ;clear above flag
            CLR DRAS
            RTS
*****************************************************************
*Cyclic buffer interrupt input routine
*****************************************************************
NM          LDX WPO
            INX
            CPX £TB                ;is X > TB (top of buffer)
            BBLE RB                ;if so reset the buffer
            LDX £BB
RB          CPX RPO
            BEQ BST                ;buffer bust
RBN         LDA A CEX
            STA A ,X
```

```
              INX
              LDA A PLS
              STA A ,X
              STX WPO
NME:          RTI
BST           CLR BFBU
              DEC BFFBU                ;set BFBU flag to FF (buffer
is bust)
              BRA RBN
*****************************************************************
*ASCII Display load subroutine (put 4 letter word pointer in XR)
*****************************************************************
ASC:          STX XR
              LDX £CD+4
              STX CDP
MC:           LDX XR
              LDA A ,X
              INX
              STX XR
              LDX CDP
              DEX
              STA A ,X
              STX CDP
              CPX £CD
              BNE MC
              RTS
*****************************************************************
* KEY SCAN subroutine       to jump table on key pressed
*****************************************************************
 KSCAN:       CLR B
              LDX £JT
BOU           LDA A CPAN
              CMP A £OFH
              BEQ KEYUP
              CLR B
              CMP B KEYFL
              BNE KSCAN
LS5:          LSR A
              BCC JPO
              INX
              INX
              INX
              CPX £JT+12
              BEQ KSCAN
              BRA LS5
JPO:          LDA A £OFFH              ;set flag so that KEYUP will
be needed for
              STA A KEYFL              ;next full key check
              JMP ,X
*****************************************************************
KEYUP:        INC B
              CMP B £OFFH
              BNE BOU
```

```
                CLR KEYFL

                BRA BOU
NULL:           ASC "    "
RSET:           ASC "RSET"
BUST:           ASC "BUST"
ENDO:           ASC "ENDO"
FILO:           ASC "FILO"
AUTO:           ASC "AUTO"
DMPO:           ASC "DMPO"
GOTO:           ASC "GOTO"
*****************************************************************
*  BLOCK TRANS.     (max. OFFH blocks)
*****************************************************************
* Load FROM with start address of data to be moved.
* Load TO with the new start address.
* Load BYTES with number of bytes to be moved.
*****************************************************************
SHIFT           LDX £WR
                STX FROM
                LDX £WRRM
                STX TO
                LDX £24
                STX BYTES
NLLLOC:         LDX FROM
                LDA A ,X
                INX
                STX FROM
                LDX TO
                STA A ,X
                INX
                STX TO
                LCX BYTES
                DEX
                STX BYTES
                BNE NLOC
SHEND:          RTS
*****************************************************************
*   ACIA (MC6850) set up (subroutine) ALSO SETS PTM
*****************************************************************
ACIARES:        LDA A £03H              ;ACIA master reset
                STA A ACIACR2           ;reset terminal control reg
                LDA A £14H              ;System set up
                STA A ACIACR2           ;Terminal control reg.
                RTS
CTCRES          CLR 4801H               ;set to write to chan 3
                LDA A £82H              ;SET BAUD RATE CHAN 3CR
                STA A 4800H             ;PUT IN CTC         3CR

                LDA A £83H              ;SET SPARE CHAN     2CR
                STA A 48801H            ;PUT IN CTC         2CR
                LDX £0CFH               ;SET BAUD RATE  3L 1200BPS
                STX 4806H               ;PUT IN CTC          3L
```

```
*****************************************************************
* SET PTM FOR TIMING ENCODER PULSES
*****************************************************************
SPTM:       CLR 4801H              ;set reg 2 so that reg 3
                                    may be written
            LDA A £3
            STA A 4800H            ;set CR3 for int.Clk./8
            STA A 4801H            ;set CR2 for  "   "  and
 *                                   select CR1 on 4800H.
            LDA A £2                ;release preset and select
Ex.Clk.
            STA A 4800H            ;  reg. 1
            LDX £OFFFFH
            STX 4802H              ;load chanel 1 counter latch
with FFFF
            RTS
*****************************************************************
*  RESET THE BUFFER     (Subroutine)
RESBUF      LDX £BB
            STX WPO
            STX RPO
            RTS
*****************************************************************
* READING THE RS232 PORT  (put data in buffer beginning at BB)
*****************************************************************
RIN         LDX £BB
RSIN        LDA B ACIACR2              ;waiting for buffer full
flag
            AND B £1
            BEQ RSIN
            LDA A ACIADR2              ;get serial byte.
           STA A,X
            INX
            CPX £TB
            BLE RSIN
            BRA RIN
*****************************************************************
* DATA TABLE TO INITIALISE DRIVE CHARACTERISTICS
*****************************************************************

ID:         FCB  OCH,0,0,0,0,0,1,32H,4,0,80H,0,40H,0BH
*****************************************************************
*  TEST DRIVE READY
TDR:        FCB 0,0,0,0,0,0
*  RECALIBRATE
CAL:        FCB 1,0,0,0,0,4
*  REQUEST SENSE (ERROR DATA)
RS:         FCB 3,0,0,0,0,0
*  FORMAT A TRACK    (0)
FAT:        FCB 6,0,0,0,0,4
*  FORMAT FROM 0
FOR:        FCB 4,0,0,0,0,4      ;Note that 5th value determines
the interleave
```

```
*                                       the value should be 1-31 ?
*  REQUEST SENSE STATUS
RSS:        FCB 3,0,0,0,0,0
*  SEEK (TRACK 0)
SE:         FCB OBH, 0,0,0,0,4
*  WRITE (1 BLOCK OR SECTOR TO SECTOR 0 TRACK 0)
WR:         FCB OAH, 0,0,0,100,4    ;each file is 25k long
*  read (1 BLOCK OR SECTOR FROM SECTOR 0 TRACK 0)
RE:         FCB 8,0,0,017H,41,4     ;read 10k+1 sector
*****************************************************************

*  KEY BOARD JUMP TABLE  (part of KEYSCAN)
*****************************************************************
            JMP FINC                ;button 2
            JMP DRSET               ;  "    1
            JMP W                   ;  "    3
            JMP DUMP                ;  "    4
*****************************************************************
* SUB TO CALCULATE BUFFER SIZE AND STORE RESULT IN BSIZE
*****************************************************************
CALBS       LDX £TB
            STX TBS
            LDX £BB
            STX BBS
            LDA A TBS+1
            SUB A BBS+1
            STA A BSIZE + 1
            LDA A TBS
            SBC A BBS
            STA A BSIZE
            RTS



*NMI, IRQ & RST vectors
*****************************************************************
            ORG OFFFCH                  ; location of the NMI
vector
NVEC        FDB NM
            FDB START               ; reset vector
            ORG OFFF8H              ; location of the IRQ
                                      vector
            FDB INTSERV
*****************************************************************
            END         (START)
```

Appendix F

F.1 Shaft encoders for measurement of vibrations and cyclic noise (Incremental motion encoder)

The data logger results from appendix E of grinding machine shaft motion, show signals from a shaft travelling at a velocity which varies only a few percent in value from its mean (figure E.8 page E.19). When the shaft is not loaded cyclical noise can clearly be observed with a period equal to the period of rotation. The most significant source of this noise are as follows:-

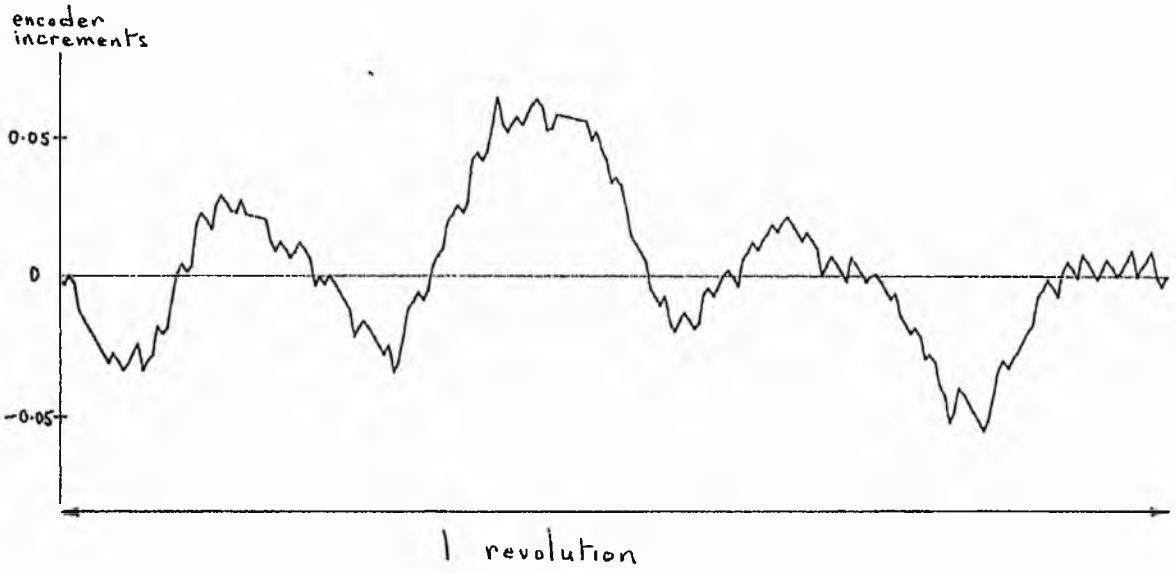
1) Small changes in the angular velocity of the shaft (angular vibrations).

2) Angular motion of the encoder relative to the shaft.

3) Inaccuracy in interslot positions of encoder.
For the Ferranti "24 rh" encoder the manufacturers claim that the maximum output signal angular error will not exceed plus or minus half a count in one revolution non-cumulative.

In the case of experiments carried out on the grinding machine, the shaft encoder was a stand alone device, connected to the shaft by a "bellows" coupling in an arrangement designed to ensure that only angular motion is received by the encoder. If the disc of the shaft encoder is mounted directly onto the shaft then there is the possibility of non angular vibrations between the optical sensor and the shaft being superimposed on to the encoder signal.

Figure F.1 Shaft Vibration and Noise



Mean angular velocity is approximately 46 rads/sec

Number of encoder increments per revolution 225

F.2 Estimation of vibrational displacement using velocity

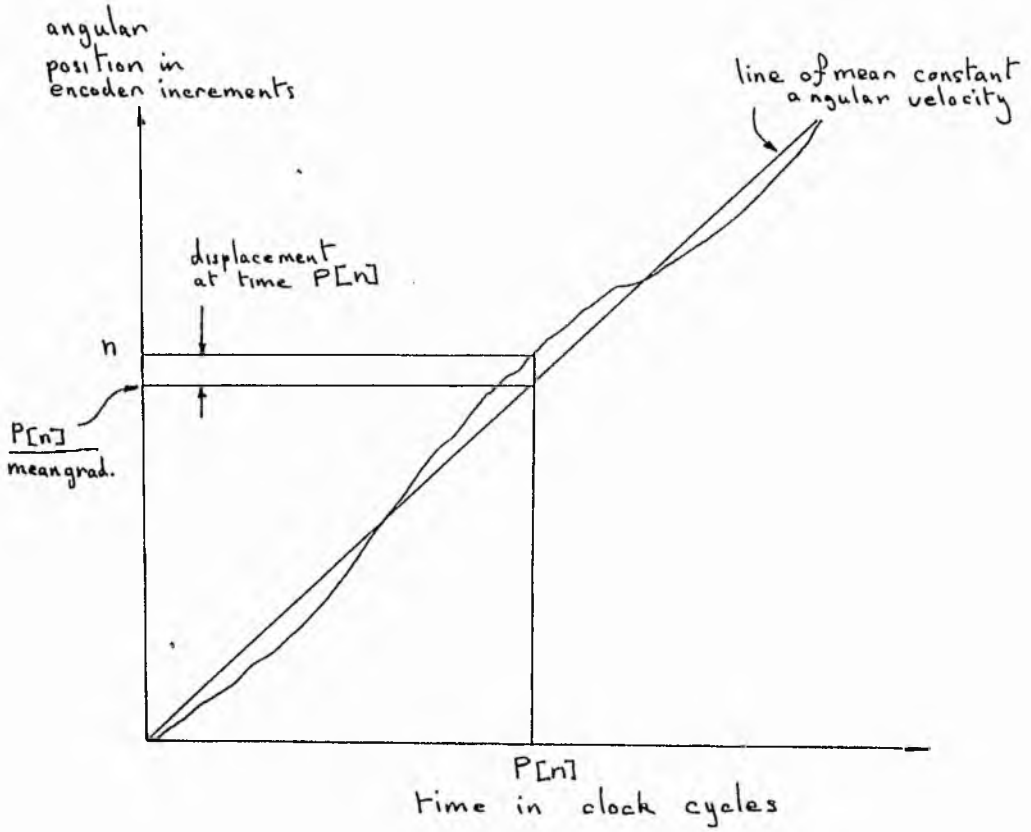
In order to assess possible shaft vibrations or encoder error noise it is necessary to decompose the signal by removal of the mean velocity (various ways of doing this are well documented ref 46). The simple mean cycle value is used as an example. The integral of the velocity fluctuations from the mean will then provide a measure of the angular vibrations. The estimated displacement obtained will contain error components of the type mentioned in section F.1. The principle error is the interslot angular error which because of its constant nature can be found and compensated for. Angular displacement may then be considered as -

The integral of (measured velocity - mean velocity)

F.3 Estimation of vibrations using positon/time data

The data logger (appendix E) recorded measurements of time at equal angular increments (within limits of error see F.1 (3)). This is expressed graphically as a sketch in figure F.2. The encoder directly records position so there is no need for velocity integration. The integral of the mean velocity is the trend line passing through the data.

Figure F.2 Estimation of Displacement



Definitions
(see also appendix A)

n = angular position incremental count

P[n] = timing clock count at incremental position n

mean_grad = mean gradient of angular position time graph
i.e. mean velocity

encres = number of increments one encoder revolution

Then at time P[n] the angular vibrational displacement in units of encoder increments is given by -

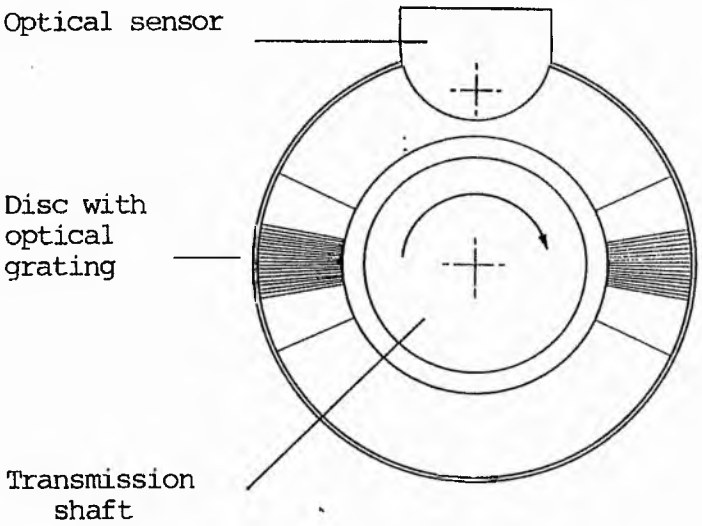
$$n - P[n]/mean\_grad$$

or in degrees

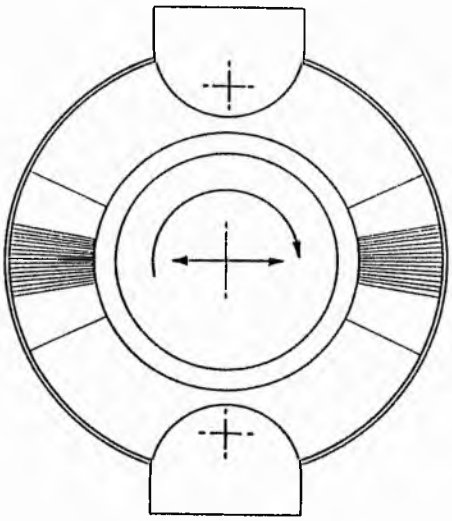
$$\frac{(n - P[n]/mean\ grad)encres}{360}$$

Figure F.1 shows results of applying the above equation on the experimental data. The maximum displacement is only a fraction of one increment and the resolution in this case is about 0.001 of one degree (far higher resolution would be obtained using a faster timing clock). This represents an approximate linear displacement of the read head of the order of 0.0005 mm. This kind of resolution means that encoder errors F.1 (3) will be significant and should be accounted for.

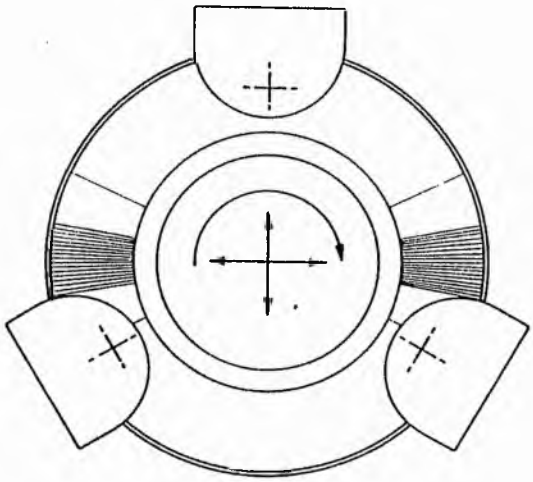
Fig F.3 Incremental motion encoder and shaft encoder



Optical incremental encoder with single read head for measurement of angular position



Additional read head enables estimation of small horizontal displacements of the shaft centre



Three or four read heads enable small two dimensional displacements of the shaft centre to be estimated

F.4 Further analysis of shaft motion

The conventional application of shaft encoders is limited to the estimation of angular position using a single optical sensing head. The encoder is normally arranged mechanically so as to eliminate, as much as is reasonable, any non-rotational motion of the encoder disc relative to the sensor. An alternative approach is to allow for a small two dimensional motion of the encoder disc centre and utilise additional sensors to estimate the position of the shaft and its angle.

It is considered in this thesis that the potential exists for the development of a specialised encoder with three or more read heads. The timing data from such a device could be processed to provide estimates of angular position of the disc and its two dimensional centre position to sub-micron accuracy. The literature survey does not show that such a device has previously been developed. The detailed development of this instrument here called "IME" or incremental motion encoder will require further research both for its physical design and signal processing requirements. The following are some initial considerations.

F.4.1 Analysis of data from multiple sensor encoder

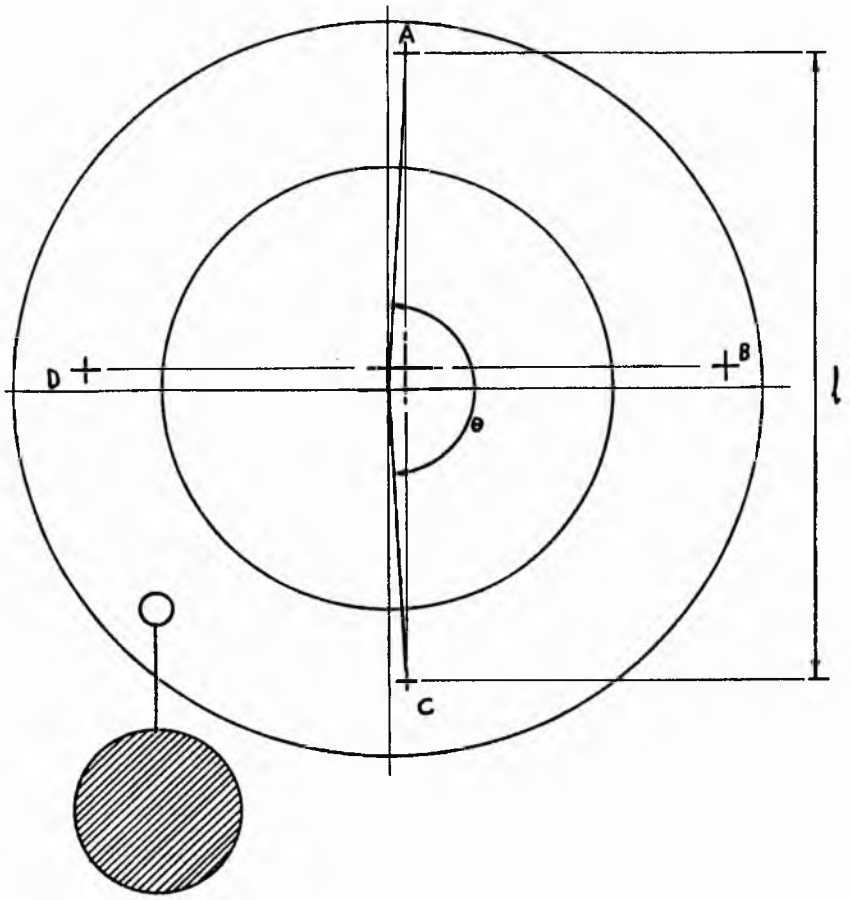
The proposed "IME" instrument will provide timing data indicating the position of three or more read heads on the encoder disc. A starting point for the estimation of the centre position of the disc could be to consider the read heads as lying on a circle whose centre is that of the disc.

(continued on page F9)

Fig F.4 Optical shaft encoder with multiple read heads
(Incremental motion encoder)
For three dimensional shaft motion measurement

The illustration shows a device with four heads A,B,C, and D at 90 degrees apart. The optical disc is shown displaced from the centre of the read heads.

Θ = measured/estimated angular deviation from A to C in radians



Because the disc is displaced from the centre of the read heads the angle Θ is no longer π radians. The centre of the disc now lies on the large radius arc traced by all points at an angle Θ between A and C. For small movements the horizontal displacement approximates to -

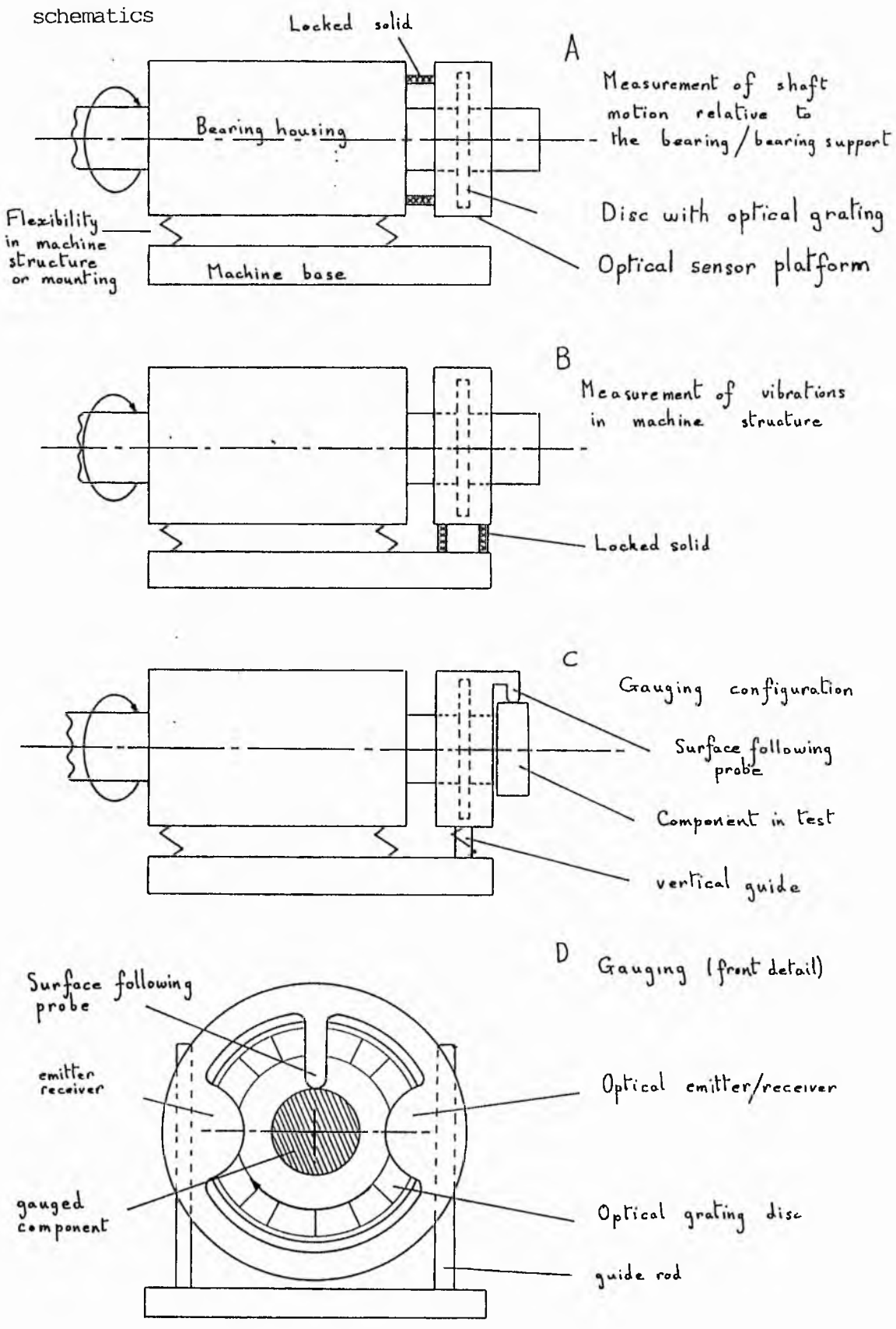
$$\frac{(\pi - \Theta)l}{4}$$

This is an approximation which can be made for centre displacements of the order of tens of microns if the disc is of the order of tens of millimeters in diameter. If four read heads are used spaced at ninety degrees apart (see Fig F.3) then two diametrically opposite read heads will provide sufficient position information to allow the estimation of the centre displacement of the disc in one dimension. The other pair of read heads will enable the estimation of the centre position in a second dimension. The error in this estimation due to radial movement of the sensors can be almost eliminated by an iterative method once an initial estimate for the displacement of the encoder disc centre has been found. A detailed consideration of the trigonometry reveals that three read heads at 120 degrees would be sufficient but would require greater data processing facility. Figure F.3 shows how any two read heads can be used to determine an arc upon which the disc centre must lie. The centre positon can be determined by calculation of the intersection of two such arcs. Three or more read heads are required.

F.4.2 Calibration of encoder disc

The encoder disc will require special consideration in order to account for inter-slot errors. The calibration of the disc (the estimation of interslot errors) may be carried out using a method of repeated averaging (ref 46,60,62). The interslot distance errors cause highly consistent distortions in position estimates with a period of one revolution. Repeated averaging of the vibration estimate over many revolutions (figure F.1) recovers consistent errors and consistent cyclic vibration signals. By using repeated small angular movements of the optical encoder disc relative to the shaft it is possible to average out the consistent vibration signal. This leaves a measure of the interslot errors.

Fig F.5 Incremental Motion Encoder Applications



F.4.3 Incremental encoder applications

Fig. 3 shows various schematic applications of the principle. There are three basic fields:-

a) Measurement of motion of shaft relative to bearing/bearing loading. The sensor platform is rigidly secured to the bearing mounting. This arrangement can be used to estimate perpendicular loads on the bearing. It is also most suitable for use with automatically controlled bearings such as "oil hydrostatic" or "magnetic" (see Fig 4.2 Chapter 4)*. Malfunction of the bearings can also be detected.

b) Measurement of vibrations in machine structure. The sensor platform is rigidly secured to the machine base. (typically vibrations due to shaft imbalance can be analysed)

c) Gauging of cylindrical components. A probe attached to the sensor platform is used to ensure that the relative motion (between sensors and optical grating) is determined by the shape of the workpiece. The probe is kept in contact with the surface of the work by using a small spring preload force.

*The I.M.E is ideal for use as the position sensor in automatically controlled bearings. It is more stable than backpressure sensors as used with hydrostatic bearings or magnetic sensors used with magnetic bearings.

Appendix G

Parameter estimation (least squares)

G.1 Introduction

Parameter estimation is the method for determining the coefficients of a control system transfer function model by processing the input and output values. The structure of the process must be known or approximated prior to parameter estimation and this can present a problem. In the case of the grinding machine, descrete time moving average models have been tried. In order to find the order of the systems, trial and error methods were used. Choosing a system model of a higher than necessary order simply results in the higher order parameters being very small and an overhead in processing time. The main requirement for successful application is that the input is such that the dynamics of the system are sufficiently excited and for a sufficient duration. The required fundamental mathematical relationship is -

$$\hat{a} = (A^T A)^{-1} A^T y \qquad \text{--- (g.1)}$$

where $\hat{a}$ is the vector of parameter estimates, A is a matrix of input and output values and y is a vector of output values (ref 47). "T" indicates transpose matrix. In the particular case of moving average systems, A reduces to a matrix of input values.

G.2 Non-linear least squares estimation

The relation applies not just to linear systems but also non-linear cases. If a record of input/ouput activity enables a set of simultaneous equations to be written with the coefficients as the unknowns (vector a) then it is possible least squares may be applied. The conditions are that the number of equations is greater than or at least equal to the number of coefficients (additionally it must be possible to form at least one matrix "A" of rank equal to the size of vector "a", or to put it simply, there must be enough independant information in the equations to provide a sound numerical solution). See also Astrom (1966) where it is shown that "y" must be a "persistently exciting" input.

If the number of equations exceeds the number of parameters the least square method provides a solution which minimises the resulting mean squared error.

In the case of the grinding machine equation (1) has been applied to a sixth order model of the workpiece drive system using program "BATCH" page g15 (this method is often referred to as the "least squares batch estimation method). The results are presented graphically at the end of this appendix page g11.
The application of least squares to the non-linear model of the interacting drive systems is explained in chapter four. It is shown how various machine constants and the resolution of forces of interaction might be estimated.

G.3 Recursive least squares

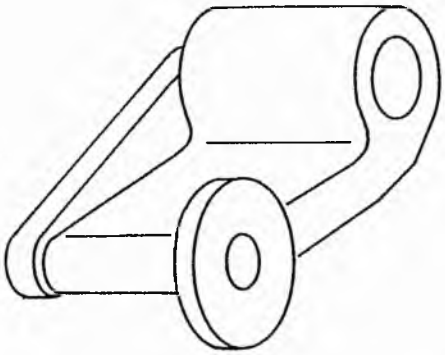
This is a method of producing repeated estimates of the parameters without having to evaluate the inverse of the matrix A. If the parameters are slowly changing in time recursive least squares is a method of tracking the changes. The program "RLS" is an example of the software required to implement such an algorithm. In the case of the grinding machine all algorithms for estimation of parameters can be of the type which updates previous estimates since the history of the machine can be saved and the parameter shifts are not likely to be very large.

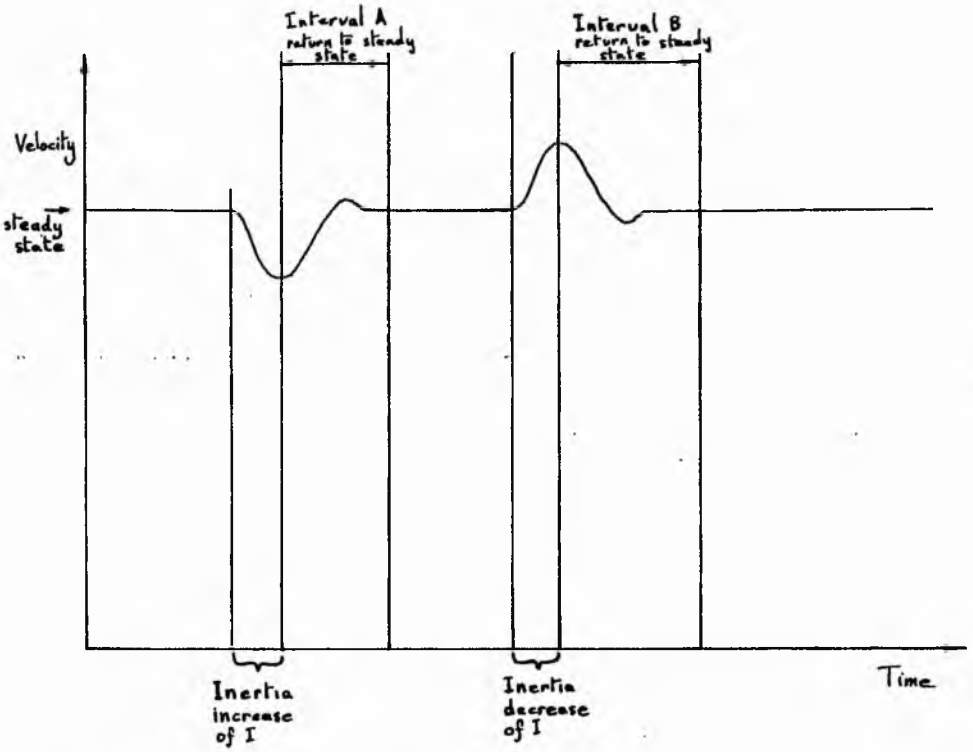
G.4 Kalman filter parameter estimator

It is possible to use a form of Kalman filter to estimate the parameters and this may be more suitable than the above method in the case of noisy signals. The intention in this research is to apply this method (ref 64) to the prediction problem of adjustment of the bearing size (see also chapter 2). In this case the parameters of the model, the prediction and the variance are all generated by the Kalman filter. The model in this case is the model of the stochastic process of output bearing sizes based on a linear filter operating on a white noise input.

Fig G.1 Parameter estimation using modified inertia

There are two main drives on the grinding machine, the grinding wheel and the work piece drives. Both drives are tested when decoupled (ie not grinding). In this illustration the grinding wheel drive is used.





Method — Estimate parameters over interval A

" " " " B

Use parameters A and B plus known inertial change I to estimate full transfer function.

G.5 Parameter estimation for the grinder drives

(without a known input)

The grinding machine has two important drive systems. One is to drive the grinding wheel and the other is to drive the workpiece or bearing to be ground. What is required is the transfer function of input applied load or torque to output velocity. This would be straightforward if it were practical to provide a known exitation load and examine the output. It is possible to arrange some kind of calibrated load but maintaining the accuracy of such a device is likely to be very difficult in the production environment. An alternative method has been devised which should avoid as much as possible those load calibration problems. The devised technique relies only on the time base crystal oscillator and a known inertial load for its calibration. Both the crystal and the inertia are highly stable and provide an accurate basis for the parameter estimation. The method requires that each drive is arranged to have a variable inertial load characteristic on the final drive shaft. This can be done in the simplest manner by attaching weights to the shaft, or automatically by the powered movement of weights from the shaft centre. Figure A.1 illustrates the technique. The graph shows the reaction of the decoupled grinding wheel drive system to sudden changes in inertia. Two periods of return to steady state are shown for the different inertia values.

The required transfer function of interest is the drive shaft velocity (input) to applied torque (output) although a more comprehensive method may consider other inputs to the system such as power voltage to take into account possible mains noise. Experiments have shown that a sixth order moving average structure is appropriate.

G.6 Drive parameter estimation (details)

The data on which the estimation is based is as usual the input output record. However in this case the input is the zero disturbance input and the output is recorded after the end of a disturbance input (ie the no load return to the steady state). In this case equation (g.1) cannot be solved.

If y is the input and u the output the discrete time system at time k sampling periods is given by -

$$y(k) = a1*u(k-1) + ... + a6*u(k-6) \quad \text{----------(g.2)}$$

(parameters a1,a2....)

Because the estimates are not required in tightly critical real time the model does not need to be physically realisable and so the following centre pivot alternative is appropriate to equation (G.2)

$$y(k) = b1*u(k-3) + ... + b6*u(k+2) \quad \text{----------(g.3)}$$

The additional force due to changing the inertial load by Q is given approximately by the centre pivot algorithm -

$$\text{additional torque} = \frac{Q*(u(k+1) - u(k-1))}{2*T} \quad \text{----------(g.4)}$$

(T being the sample period)

Then equation for the modified system is given by

$$\begin{aligned} y(k) = \quad & + b1*u(k-3) \\ & + b2*u(k-2) + (b3-Q/2)*u(k-1) \\ & + b4*u(k\ \ ) + (b5+Q/2)*u(k+1) \\ & + b6*u(k+2) \end{aligned} \quad \text{----------(g.5)}$$

Now in the no input [zero y(k)] situation equation (G.3) maybe rewritten as follows -

$$u(k-3) = b2/b1*u(k-2) + ... + b6/b1*u(k+3) \quad -----(g.6)$$

This may be considered as a fifth order system with input u(k-3).

Similarly the modified system may be reduced to a fifth order system. Thus we have two fifth order systems and for both of them least squares can provide a solution for the parameters.

Thus the output record with no input can be used to find actual values for the parameters (from the unmodified system)-

b2/b1, b3/b1, b4/b1, b5/b1, b6/b1,

and

b2/b1, (b3-Q/2)/b1, b4/b1, (b5+Q/2)/b1, b6/b1,

Let these values found from the estimation be-

BU1,BU2,....

and

BM1,BM2,.... respectively

Thus there are now 10 equations and 6 unknowns. Six of the equations should be independant and yield sound numerical solutions.

Using the second parameters from each set and dividing one by the other we have -

$$b3 - Q/2 = b3*BM2/BU2 \text{ ----------(g.7)}$$

ie

$$b3 = \frac{Q/2}{1 - BM2/BU2} \text{ ------(g.8)}$$

also

$$b1 = \frac{Q/2}{BU2 - BM2} \text{ -------(g.9)}$$

Similarly all the parameters b1,...b6 may be found

It is possible that accuracy could be gained by using a higher order algorithm for the approximation of the acceleration in equation (G.4) for the added inertial load. There is redundancy in the data (10 equations 6 unknowns) and this could be exploited to provide a mean estimates and assess the reliability of the estimates to a limited extent.

G.7 Drive systems parameter estimation summary

To summerise, the method can be broken down into the following steps.

1) Obtain the velocity profile data over a period just after a disturbance torque has been removed (alternatively in some cases this can be the same response as the control input return to set point response and could be obtained by a temporary disconnection of drive power or speed control input).

2) Repeat step "1" after modifying the system parameters by adding or removing a known inertial load.

3) Reduce the system model to the next lowest order and perform parameter estimation (recursive least squares may be used to update estimates from previous tests).

4) Use equations "6" to "9" (or similar) to solve for full set of system parameters.

G.8 Software and results

1) Pascal (Versados program) "RLS" demonstration of recursive least squares running on the Versados development system. Page 10. See (ref 15 for algorithms)

2) Pascal sub program "SIM" sample test model of control system. Page 13

3) Pascal program "BATCH" uses batch method to partially estimate parameters (see section G.5) of drive system by processing response of system to the release of an unknown load. Page 14.

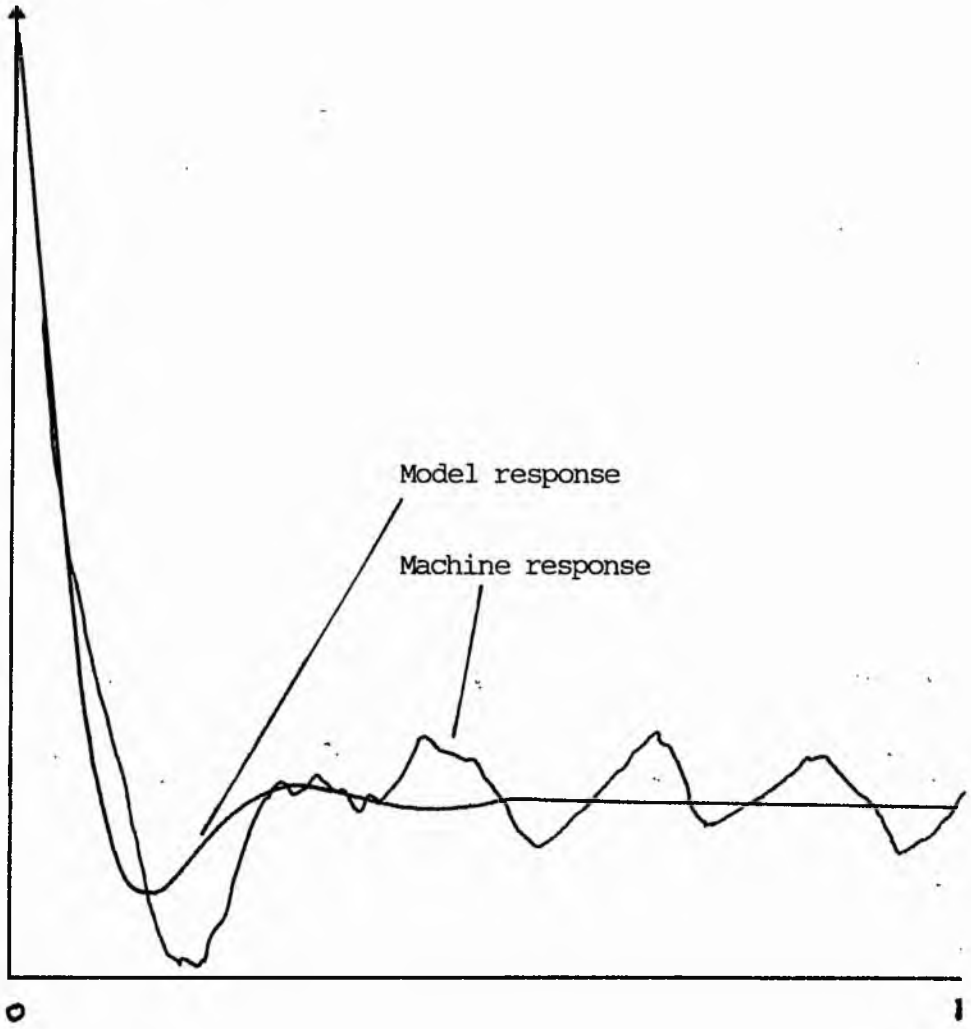
4) Results from 3) tested on a grinder workhead. Page 20 Fig G.2

Fig G.2 Parameter estimation (grinding wheel drive)

Velocity time graph showing model and machine reaction to the removal of a disturbance load torque.

Fifth order model parameter estimates are -

B1	B2	B3	B4	B5
1.226	0.137	-0.631	0.16	35.9



Time seconds ⟶

approx 7 revolutions in 1 second

sampling rate 1 Kilo Hz, 100 samples shown

Program G.1 "RLS"

```
{*****************************************************************
*       Recursive  Least Squares Parameter  Estimation          *
*       This is really a software test.  This program           *
*       operates  only on a model  of a control system.         *
*       The model is arranged as a function "SIM".              *
*       This program  has no  access to  the  function          *
*       parameters  but is able to estimate them quite          *
*       accurately  by feeding  an excitation input to          *
*       the function and processing the output with an          *
*       RLS algorithm. It is a simple change to switch          *
*       from the model to real time or file data input          *
*       output records.  Parameter estimates are sent           *
*       to the standard output and quickly converge to          *
*       the preset test function values.                        *
*****************************************************************}



program RLS(FI,input,output);
     type
                    count= 1..100;
                     mat = array[-1..100] of real;
                  string4 = packed array[1..4] of char;
       var
        I,L,PR,BR,D,DT,PD : mat;
                    DF,FI : text;
                    N,M,H : integer;
       AF,WF,IWF,SC,SB,yd : real;
                        X : char;
         {$F=sim}                  { model/filter }
function  Mtran   (Z       : mat)             :mat ;     forward;
function  Msub    (Z,X     : mat)             :mat ;     forward;
function  Mult    (Z,X     : mat)             :mat ;     forward;
function  Vmult   (Z,X     : mat)             :real;     forward;
function  Smult   (Z       : mat;S :real)     :mat ;     forward;
function  Minv    (Z       : mat)             :mat ;     forward;
function  Mint    (Z       : mat;R,C :real)   :mat ;     forward;
function  Munit   (D       : real)            :mat ;     forward;
function  Mpush   (Z,X     : mat)             :mat ;     forward;
procedure Mread   (var Z   : mat;var F:text)      ;     forward;
procedure Mprint  (Z       : mat;S:string4)       ;     forward;

{_____________________________________________________________}
```

```
                begin
                reset(FI);
                I:=Munit(3);
                D:=Mint(D, 3,1);
                int;
{-------------------------------------------------------------}
              H:=0;
              Mread(PR,FI);       {initial PR matrix          }
              Mread(BR,FI);       {   read in the starting    }
                                  {estimates of the parameters}
              Mprint(PR,'PR =');
              WF:=0.8;            { set waiting factor        }
                                  { (estimator damping factor)}

              { pre-calculate 1-WF and 1/WF to avoid
              {               repetition                      }

              AF:=1-WF;
              IWF:=1/WF;          { inverse waiting factor    }

              while 1>0 do        { loop always }
              begin
 writeln('    A1 = ',BR[1]:4:4,'    A2 = ',BR[2]:4:4,'    A3 =
',BR[3]:4:4);
              {******* obtain the latest data*********}
              yd:=sim;
              D[1]:=Y2;   D[2]:=Y1;  D[3]:=U2;
            H:=H+1;
```

```
{        Recursive least squares          }
PD:= Smult(Mult(PR,D),IWF);
DT:= Mtran(D);
SB:= Vmult(DT,PD);
SC:= 1/((1/AF)+SB);
L := Smult(PD,SC);
SB:= Vmult(DT,BR);
SC:= SB-yd;
BR:= Msub(BR,Smult(L,SC));  { new parameters }
PR:= Smult(Mult(Msub(I,Mult(L,DT)),PR),IWF);
```

{ Excitation function generator U2 }

```
{ U2 is 0 for 5 periods ramps down till 12 ramps up till}
{ 18 then repeats }
          if H=5 then
               begin
               U2:=U2-2;
               end;
           if H=12  then
             begin
              U2:=U2+4;
             end;
           if H=18 then
              begin
             U2:=0;
             H:=0;
             end;
           end;
    end.
```

Program G.2 "SIM"

```
{*****************************************************************}
{*  Digital implementation of a second order filter or control   *}
{*  system (Test model for recursive least squares).             *}
{*  Transfer function -                                          *}
{*                                                               *}
{*                                                               *}
{*                          W*W                                  *}
{*                 ------------------------                      *}
{*                 S*S  +  2*G*W  +  W*W                         *}
{*                                                               *}
{*        W is the fundamental frequency                         *}
{*        G is the damping factor                                *}
{*        G =  1 for critical damping                            *}
{*        G => 1 for no overshoot                                *}
{*                                                               *}
{*        "int" initialises the filter                           *}
{*        "sim" takes past values of the input U                 *}
{*        and the output Y to evalute a new value for Y.         *}
{*        U2 is the last input. Y3,Y2,Y1 are present and         *}
{*        values of the output.                                  *}
{*****************************************************************}

   T,E,W,G,Y1,Y2,Y3,U2,A1,A2,A3 : real;
 function sim :real;
          begin
             Y1:=Y2;  Y2:=Y3;
          Y3:=A1*Y2+A2*Y1+A3*U2;
         sim:=Y3;
         end;

procedure int; {Initialisation procedure}
    begin
     T:=0.2; E:=0; W:=0.7; G:=2;
     Y1:=E; Y2:=E; Y3:=E; U2:=E;
     U2:=65;
           A1:= (2-T*W*W)/(G*W+1);
           A2:=   (G*W-1)/(G*W+1);
           A3:=     T*W*W/(G*W+1);
            end;
```

Program G.3 "BATCH"

```
program Batch(input,output);
{Batch Parameter Estimation of the Grinder Drives
(Input  - Disturbance torque to drive head.
 Output - Angular velocity of     "     ")
 Model is fifth order moving average.
```

Main program to perform estimation upon the detection of a disturbance input to the drive. The disturbance is unknown in magnitude and only the velocity record as the system returns to its set point is used. It is assumed that the disturbance has been completely removed by the time sampling is taking place. The input to the system in this interval is therefore zero. This method does not provide enough information to determine all the parameters (a known non-zero input record would be required for that). The intention is to completely describe the system by using the zero-input as described, but also repeating the procedure with a modified inertial load (see notes). }

```
{$S=40000}   {Additional stack/heap memory bytes are
        required due to recursive matrix function calling}
      type
                      pword = 0..65535;
                       word = -32768..32767;
                      count = 1..300;
                       byte = -128..127;
                        mat = array[-1..300] of real;
                        ar2 = array[-2..100] of pword;
                    string4 = packed array[1..4] of char;
                      coord = array[0..1] of real;
        var
           I,L,P,PR,Y,A,AT : mat
            B,BR,D,DT,PD,C : mat;
                 J,N,M,H,T : integer;
          VEL,MD,U,SP,AF,WF : integer;
            IWF,SC,SB,yd,ud : real;
                      FI,GT : text;
                         CH : char;
                       V,VT : ar2 ;
                         ER : word;
                    MIN,MAX : pword;
                         XY : coord;
```

```
procedure start                                          ;      forward;
function  act     (T       : integer)          :word;     forward;
function  Mtran   (Z       : mat)              :mat ;     forward;
function  Msub    (Z,X     : mat)              :mat ;     forward;
function  Mult    (Z,X     : mat)              :mat ;     forward;
function  Vmult   (Z,X     : mat)              :real;     forward;
function  Smult   (Z       : mat;S :real)      :mat ;     forward;
function  Minv    (Z       : mat)              :mat ;     forward;
function  Mint    (Z       : mat;R,C :real)    :mat ;     forward;
function  Munit   (D       : real)             :mat ;     forward;
function  Mpush   (Z,X     : mat)              :mat ;     forward;
procedure Mprint  (Z       : mat;S:string4)         ;     forward;
function  scan    (mode    : byte)             :ar2 ;     forward;
function  plot    (XY      :coord;X,Y:real)  :coord;      forward;
procedure graph   (Z       : ar2;CLS :byte)         ;     forward;
procedure gdump                                          ;     forward;
procedure sync                                           ;     forward;
function  ivel                                    :pword;     forward;
procedure tor     (var Z   : ar2;var X:mat)          ;     forward;

{_____________________________________________________________}
                  begin
                 rewrite(FI,'results');
write(FI,'   B1         B2         B3         B4         B5');
writeln(FI,'                SP');
writeln('Set sampling rate in milli seconds');
readln(T);
       if T=0 then
       begin
        T:=10
       end;
       writeln('Period is ',T,'milli seconds');
                ER:=act(T);
                  U:=1;
```

```
                    while 1>0 do
   {The start of the main loop which performs one batch
   parameter estimation and then prompts a repeat
   experiment. Use "control C" to leave.}

                     begin
   writeln('Graphics dump ?   (Y/N/Q)');
               reset(input,'£');                read(CH);
                 while (CH='T') or (CH='t') do
                 begin
    V:=scan(0);
           for N:=-2 to 100 do
           begin
             V[N]:=V[N]-MIN;
           end;
    graph(V,0);
            V[0]:=V[1];
          for N:=3 to 97  do
            begin
VT[N]:=round((V[N-3]+V[N-2]+V[N-1]+V[N]+V[N+1]+V[N+2]+V[N+3])/7);
            end;
           for  N:=3 to 97 do
            begin
             V[N]:=VT[N];
           end;
             graph(V,1);
           tor(V,B);
      V[-1]:=V[1]; V[-2]:=V[1];
        for N:=2 to 100  do
        begin
      writeln(V[N]);
      if V[N]>V[-1]  then
          begin
        V[-1]:=V[N];
          end;
{ if V[N]<V[-2]  then
          begin
          V[-2]:=V[N]
          end;}
          end;
```

```
            V[-2]:=0;
            writeln(V[-2],V[-1]);
            writeln('************************************');
            graph(V,1);
            reset(input,'£');
            readln(CH);
              end;
              if (CH='Y') or (CH='y') then
              begin
              gdump;
                 reset(input,'£');
                 writeln('Graph title text to printer (Y/N)');
                 read(CH);
               if (CH='Y') or (CH='y') then
                 begin
                   rewrite(output,'£cn01');
              writeln('          ',chr(18));
writeln('Max sample',MAX,'   Min sample',MIN,
                                '     Set point =',SP:4:1);
write('Steady state velocity =',VEL:4:1,' R.P.M.');
writeln(' Max deviation  =',MD:2:1,'% .');
write('X axis covers 1 second -
Sampling rate is 100 Hz- 100 samples');
writeln(' shown');
writeln('Y axis units are 100 Nano second.');
writeln('There are 1024 encoder pulses per rev.
 and the time for 64 to');
writeln(' pass after the sampling instant
is taken by counting the 100 N/s');
writeln(' clock pulses.');
              writeln(chr(20));
              end;
              end;
              if (CH='Q') or (CH='q') then
               begin
                halt(0);  end;
```

```
            rewrite(output,'£cn01');
                 Y:=Mint(Y,60,1);
                 A:=Mint(A,60,5);
                 I:=Munit(5);
        {****** load Y and A *********}
                 M:=round(Y[-1]);
                 H:=round(A[ 0]);
         writeln(chr(27),chr(12));
writeln; writeln; writeln; writeln; writeln; writeln;
writeln; writeln; writeln; writeln; writeln; writeln;
writeln('                    WAITING FOR DISTURBANCE INPUT');
                 sync;
                 V:=scan(0);
                MIN:=V[-2];
                MAX:=V[-1];
                for N:=-2 to 100 do
                 begin
                   V[N]:=V[N]-MIN;   {offset sample results
                   by  the minimum sample to avoid the use of
                   large numbers with small differences creating
                   numerical problems in the estimation}
                  end;
                  V[-2]:=-V[-1];
                   graph(V,0);
               XY[0]:=0; XY[1]:=0;
               XY:=plot(XY,0,0);  {set graphics cursor home}
                for N:= 1 to M do
                 begin
                   {load up the A matrix with the sample
                                   results}
                   Y[N]:=V[N+4];
                   J:=H*N;
                   A[J]:=U;  A[J-1]:=V[N]; A[J-2]:=V[N+1];
                   A[J-3]:=V[N+2]; A[J-4]:=V[N+3];
                 end;

                AT:=Mtran(A);  {Transpose sample matrix A}
```

```
{Use least square parameter (batch method) estimation equation}

          B :=Mult(Mult(Minv(Mult(AT,A)),AT),Y);
        {Vector B is the estimate of the parameters}

writeln('              B1          B2          B3            B4
B5');
writeln('         ',B[1]:4:4,'    ',B[2]:4:4,'    ',B[3]:4:4,
'    ',B[4]:4:4,'    ',B[5]:4:4);
                rewrite(output,'£');
             SP:= B[5]*U/(1-B[1]-B[2]-B[3]-B[4]);
               {Set point less the minimum value in the samples}
            VEL:= 1E7*60/((SP+MIN)*32);
               {Set point velocity}
             MD:= (MAX-(SP+MIN))*100/(SP+MIN);
               {Maximum deviation of the sample}
writeln(FI,B[1]:4:4,'    ',B[2]:4:4,'    ',B[3]:4:4,
'    ',B[4]:4:4,'    ',B[5]:4:4,'    ',round(SP)+MIN);
{ The discrete equation of motion should now be as follows
          V[3]  = B[1]V[2] + B[2]V[1] + B[3]U
  To check that this really is correct the next section takes the
starting
  point of the sample and constructs the estimated response. }
         for N:=4 to 100  do
             begin
V[N]:=round(B[1]*V[N-1]+B[2]*V[N-2]+B[3]*V[N-3]+B[4]*V[N-4]+B[5]*
U);
             end;
            graph(V,1);
       end;
       end.
```

Appendix H
Grinding simulator software design

H.1 Software design

A simulator program was written in "C" language (ref 71). The program structure follows the system decomposition as illustrated in chapter 3 Fig 3.1 with functions being written to represent each process.

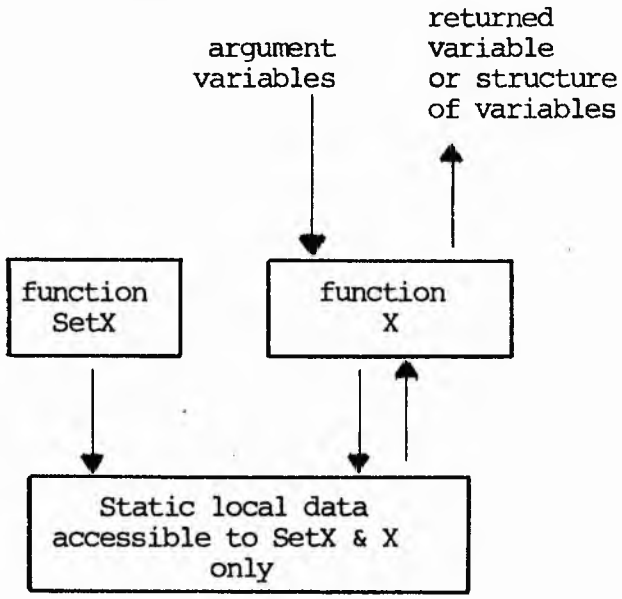
The program has been designed to minimise the possibility of programming errors through handling large numbers of global varibles. Each of the functions requires one or more input variables which are passed as arguments. Each function is required to return one or more outputs and this is done by returning a variable or a structure of variables. In this way the function can only see the input argument variables from the calling level of the program.

A function may require its own variables (state) which are not lost, as would normally be the case on leaving the function (stack variables some times referred to as automatic variables ref 71). The function is effectively required to have a memory. This problem was solved by using static variables declared global in the source files for each function. With this arrangement any additional functions written in the source file can share the static variables but the variables are hidden from the calling level of the code. The approach here is towards object orientated programming with each control process being a distinct object with carefully arranged interfaces designed to allow private process variables which can be accessed by special initialisation functions only. The new "object-orientated" C++ language would enable an improved implementation with all the processes being created as different objects of a single "control class type" (ref 74).

Figure H.1 Simulation

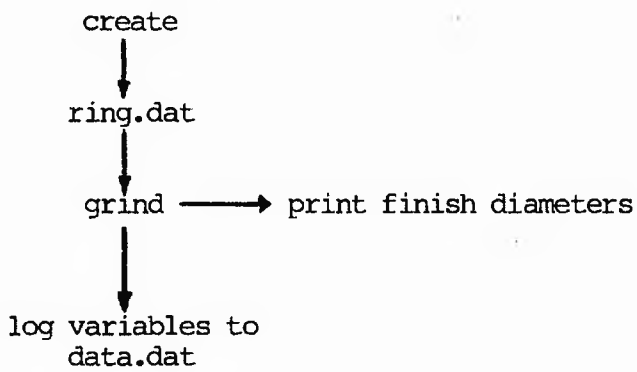
a) Initialsation using "Set" functions

Example for function X



Function SetX called to initialise function X's local static data area (no other access to X's private data to be allowed).

b) Simulator input and output



Command "create" generates a "ring.dat" file.
" "grind" runs the simulation.

H.2 Initialisation functions

The local static variables are required to be set up or initialised at some stage and this is done by using special set up functions which are written in the same source file as the function which represents the process.

The notation for a process function is the code used in the decomposition illustrations ie "A1()". The set up function is "SetA1()". Unfortunately the source code files may not be given the function names because this causes the definition of duplicate symbol names (compiler errors). The source files are therefore coded with an "F" in the place of the "A". The functions "A1()" and "SetA1()" are thus contained together in the source code file "F1.C"

H.3 Simulator input/output

The simulator operates on a mathematical model of the component. For further details see description of "A1_2_1". This model is kept as a file of numbers representing measurements across the theoretical part (diameters). The current version of the simulator operates on a single component file, "ring.dat" each time the program is run. The file "ring.dat" is copied into an array and it is this array which is changed. The simulator reduces the dimensions held in the array and finally after the completion of a theoretical machine cycle prints the contents of the array. The values should all be the same as the finish size parameter set in the simulator (within plus or minus 0.0005mm). The simulator records important process variables on a file "data.dat", the model equivalent of a data logger.

The file "ring.dat" representing the unground component can be modified by running a program called "create" which enables the setting up of data representing round or elliptically distorted components of variable dimensions.

H.4 Source programs for simulator ("C" lanuage)

H.4.1 Index

```
/*****************************************************************/
/* GRINDING SIMULATION                                           */
/*****************************************************************/

£include <graf.h>
£include <stdio.h>
£include <math.h>
£include <defs.h>

/* External functions */
extern A1OUT  A1();     /* Grinding machine effects     */
extern A2OUT  A2();     /* Grinding wheel/work
                                interface process       */
extern int SetA1();     /* Initialise A1,A2's state     */
extern int SetA2();     /* and other local variables    */

/* External variables */
extern float Dia[];     /*Array model of component shape*/


/* Global variables */
long t;                 /* Time measured in sec/100     */
long Ts;                /* Sampling period    "         */
long timeup;            /* End of cycle time  "         */

RECORD data;            /* Structure for log file to    */
                        /* record variables             */

main()
{
A1OUT  A1out;
A2OUT  A2out;
float dia;              /* Diameter of component            mm*/
float gr;               /* Position of grinding wheel       mm*/
float Xforce,Yforce;    /* Components of   "    force       N */
float cut;              /* Depth of cut                     mm*/
int timeout;            /* Time limit                   sec/100*/
int n;                  /* local counter variable             */
FILE *fdata;



Ts = SAMPLING_PERIOD;   /* 0.01sec units        */

A2out.Xforce = 0;  /* set grinding force vector to zero  */
A2out.Yforce = 0;

SetA1();                                /* intialise A1 */
SetA2();                                /* intialise A2 */

Xforce = 0;
                    /* open data file for logging variables */
```

```
fdata = fopen("data.dat","wb");

if(fdata == 0)
        {
        printf("data.dat file not found");
        exit(1);
        }


for (t=Ts; t < timeup ; t=t+Ts) /* Time increments by sampling*/
        {                       /* period each time round this*/
                                /*          loop.             */


        A1out = A1(A2out); /* grinding machine effects   */

        printf("( %d )",A1out.contact);

        A2out = A2(A1out); /*    "  wheel/work interface */


                                /*          log  variables
*/
        if(fwrite(&data,sizeof(RECORD),1,fdata) != 1)
                {
                printf("Unable to write to data.dat");
                exit(1);
                }


        }

printf("\n\n\n");

       /* Show dump of bearing model array */
for(n=0; n<REVRES; n++)
        {
        printf("%5.3f\t",Dia[n]);
        }
}
```

```
/*****************************************************************/
/* A1 GRINDING MACHINE    (mechanical effects)                   */
/*****************************************************************/

£include <graf.h>
£include <stdio.h>
£include <math.h>
£include <defs.h>

extern long Ts;
extern long t;

extern float    A1_1();  /* function generator                   */
extern A1_2OUT  A1_2();  /* rotation and elastic structure       */
extern TORQUE   A1_3();  /* balance of grinding forces           */
extern float    A1_4();  /* work piece drive                     */
extern float    A1_5();  /* grinding wheel drive                 */

extern int   SetA1_1();
extern int   SetA1_2();
extern int   SetA1_3();
extern float SetA1_4();
extern float SetA1_5();

static A1OUT     A1out;

int SetA1()
{
SetA1_1();
SetA1_2();
SetA1_3();
A1out.velocity.workshaft  = SetA1_4();
A1out.velocity.grindshaft = SetA1_5();
}

A1OUT A1(A2out)
A2OUT A2out;
{
TORQUE torque;              /* vector of grinding wheel & work
                               piece drive applied torques Qg,Qb */

A1_2OUT    A1_2out;
float          gr;
float         cut;
float Xforce,Yforce;

    gr = A1_1();                          /* function generator */

                                  /* structure + rotation model */
    A1_2out = A1_2(gr,A2out,A1out.velocity);

    torque = A1_3(A2out,A1_2out.diam);    /* non-linear force balance   */
```

```
                                /*grinding wheel drive system */
    A1out.velocity.workshaft = A1_4(torque.workshaft);

                                /* work piece drive system    */
    A1out.velocity.grindshaft = A1_5(torque.grindshaft);

    A1out.cut           = A1_2out.cut;
    A1out.diam          = A1_2out.diam;
    A1out.contact       = A1_2out.contact;

    return(A1out);

}
```

```
/*************************************************************/
/* A2 GRINDING WHEEL WORK INTERFACE PROCESS                  */
/*************************************************************/
/* Model based on grinding power model (Malkin ref 1)        */

£include <stdio.h>
£include <math.h>
£include <defs.h>

static float    C1;     /* Sliding power constant       */
static float    C2;     /*    "       "       "          */
static float    PCH;    /*  Chip      "       "          */
static float    PPL;    /* Ploughing "       "          */
static float    XCN;    /* Horizontal force constant    */
static float    cv;     /* dimension conver. factor     */

static float    A;      /* Proportion of wear flat area */

SetA2()
{
/* Set up variables for Grinding interface model ref (1)*/

C1 = 7.55E-3;    /* Sliding power constants       */
C2 = 2.10E-3;

PCH = 13.8;      /* Chip power constant           */
PPL = 9.6;       /* Plowing "     "               */
XCN = 8;         /* Horizontal force parameter    */

A = 0.25;    /* proportion of wear flat area wear */
cv = 2000;   /* dia mm to radius m constant       */

}

VECTOR A2(A1out)
A1OUT A1out;
{
VECTOR grindforce;
float de;                   /* equivalent diameter see ref 1 */
float Pow;                  /* net grinding power      Kw/mm */
float ds,dw,vs,vw,a;        /* diameters, velocities & cut mm*/
float psl;                  /*        as follows             */

 dw = A1out.diam.workshaft;
 ds = A1out.diam.grindshaft;

 vw = A1out.velocity.workshaft*dw/cv;
 vs = A1out.velocity.grindshaft*ds/cv;


 a  = A1out.cut;

                          /* Power series equations of Malkin */
```

```
if (A1out.contact == 1)
{
     de = ds*dw/(dw + ds);     /* equivalent diameter see ref 1 */


                    /* Net grinding power Kw/mm */
    if(a > 0.00001)
      {
      psl  = (C1  +  C2*vw/(vs*de))*sqrt(de*a)*A;
      }
   else
      {
      psl = 0;
      }

      Pow = PCH*vw*a   +   PPL*vs  +  psl;


grindforce.Yforce = Pow/(100*vs);

/* Grinding wheel wear modelling can be included here by       */
/* linking the wear flat area to a total volume removal        */
/* variable and of course feeds and speeds                     */


grindforce.Xforce = XCN*a*vw*A;       /* linear approximation  */
                /* for horizontal component of grinding force */

                                          /* Newton's          */
}
else
{
grindforce.Xforce = grindforce.Yforce = 0;
}

return(grindforce);              /*returns grinding force vector*/

}
```

```
/*****************************************************************/
/* A1_1  FUNCTION GENERATOR                                      */
/*****************************************************************/

£include <graf.h>
£include <stdio.h>
£include <math.h>
£include <defs.h>

extern RECORD data;
extern long  Ts;
extern long  t;
extern long  timeup;

static float gr;              /*position of grinding wheel mm */
static float grstart;         /*  start                       */
static float findist;         /*  finish                      */
static float incslow;         /*  slow feed increment         */

static float incfast;         /*  fast   "     "              */
static long  t1;              /*  time at end of fast infeed  */
static long  t2;              /*  time at end of slow infeed  */
static long  t3;              /*  time at end of dwell        */
static long  t4;              /*  time at end of fast retract */



SetA1_1(arg)
float arg;
{

float clear;                  /* mm                           */
float slowdist;               /* mm                           */
float ds;
float fastrate;               /* mm/sec                       */
float slowrate;               /* mm/sec                       */
float sparktime;              /* seconds                      */

findist = 50;
clear = 5;
slowdist = 2;
fastrate = 10;
slowrate = 0.5;
sparktime = 4;

/* need to find time at end of fastfeed and time at
end of slow speed- t1,t2, and t3 the end of sparkout*/
/* start value gr */
gr = grstart =findist + clear;

/* slow feed start dist */
ds = findist  + slowdist;
```

```
/*time to ds */

t1 = ((gr - ds)/fastrate)*100;          /*seconds/100*/
t2 = t1 + ((ds - findist)/slowrate)*100;/*    "      */
t3 = t2 + sparktime*100;
t4 = t3 + ((gr - findist)/fastrate)*100;

timeup = t4;

printf("times %ld - %ld - %ld - %ld\n",t1,t2,t3,t4);

/* incrememental movement in sample period for fast and slow feed */
incfast = Ts*fastrate/100;
incslow = Ts*slowrate/100;
}

float A1_1()
{
double x;
if (t<= t1)
        {
        gr = gr - incfast;
        }

if (t>t1 && t<t2)
        {
        gr = gr - incslow;

        if (gr < findist)
                {
                gr = findist;
                }
        }

if ( t>=t2 && t<t3)
        {
        gr = findist;
        }

if (t>t3 && gr < grstart )
        {
        gr = gr + incfast;
        }

    data.gr = gr;               /*file record*/

    return(gr);
}
```

```
/*****************************************************************/
/* A1_2 ROTATIONAL EFFECTS & MACHINE STRUCTURAL ELASTICITY       */
/*****************************************************************/

£include <stdio.h>
£include <math.h>
£include <defs.h>

extern int SetA1_2_1();
extern int SetA1_2_2();

extern A1_2_1OUT   A1_2_1();
extern float A1_2_2();



SetA1_2()
{
SetA1_2_1();
SetA1_2_2();
}

A1_2OUT A1_2(gr,grindforce,velocity)
float gr;
VECTOR grindforce;
VELOCITY velocity;
{
A1_2OUT A1_2out;
A1_2_1OUT A1_2_1out;
float disp;

                        /*response ofstructure to applied load */

disp  = A1_2_2(grindforce.Xforce);

A1_2_1out = A1_2_1(gr,disp,velocity);        /* numerical rotation model */

A1_2out.diam    =     A1_2_1out.diam;
A1_2out.cut     =     A1_2_1out.cut;
A1_2out.contact =     A1_2_1out.contact;

return(A1_2out);
}
```

```
/*****************************************************************/
/* A1_3 BALANCE OF GRINDING FORCES (Non-linear)                  */
/*****************************************************************/

£include <graf.h>
£include <stdio.h>
£include <math.h>
£include <defs.h>



static float cfric;     /*coefficient of friction of the shoes */

SetA1_3()
{

cfric = 0.1;                        /* 0.1 SHOE_FRICTION; */

}



TORQUE A1_3(grindforce,diam)
VECTOR grindforce;
DIAM   diam;
{
TORQUE torque;
float Xforce;
float Yforce;

 /*    The torques calculated here are the net torques on each
  shaft due to the grinding interface force and the additional
    friction created by the reaction to the interface force   */

Xforce = grindforce.Xforce;
Yforce = grindforce.Yforce;

torque.grindshaft = Yforce*diam.grindshaft/2;

torque.workshaft =
  (Yforce - sqrt(Xforce*Xforce + Yforce*Yforce)*cfric)
                                                  *diam.workshaft/2;

/* printf("%f\t",torque.workshaft);*/

return(torque);     /*returns net disturbance torques on
                          work shaft and grinding wheel shaft */
}
```

```
/*****************************************************************/
/*A1_4 COMPONENT ANGULAR VELOCITY                                */
/*****************************************************************/
£include <stdio.h>
£include <defs.h>

extern RECORD data;
extern long Ts;        /* sampling period */

static float y1,y2,y3;     /* values of present & past output*/
static float a1,a2,a3;     /* model parameters */
static float st;           /* steady state velocity */

SetA1_4()
{
float g;         /* damping factor  g = 1 critical */
float w;         /* fundamental frequency */
float per;
/* Setup parameters and initialise second order process */

g = 0.7;       w = 5;        per = 0.05;

/* Model parameters set up using central difference approx */

a1 = (2 - per*w*w)/(g*w + 1);
a2 = (g*w - 1)/(g*w + 1);
a3 = per*w*w/(g*w + 1);

y1 = y2 = y3 = 0;

st = 45;         /* steady state angular velocity rads/sec*/

printf("\n%f\t%f\t%f\n\n",a1,a2,a3);

return(st);
}

float A1_4(Qb)
float Qb;
{
float wb;
float u2;

printf("\t%5.3f",Qb);

u2 = Qb/45;    y1 = y2;    y2 = y3;
y3 = a1*y2 + a2*y1 + a3*u2;

wb = data.wb = y3*5 + st;    /* add steady state velocity */

  return(wb);
}
```

```
/*****************************************************************/
/* A1_5 GRINDING WHEEL VELOCITY                                  */
/*****************************************************************/

£include <stdio.h>
£include <defs.h>

extern RECORD data;
extern long Ts;        /* sampling period */


static float y1,y2,y3;          /* output                */
static float u1,u2,u3;          /* input                 */
static float st;                /* steady state          */

static float c1,c2,c3,c4,c5;    /* difference equation   */
                                /*    coefficients       */
SetA1_5()
{
float g;              /* damping factor  g = 1 critical */
float w;              /*    fundamental frequency       */
float per;
static float a1,a2,a3;
static float b1,b2,b3;

/*  Setup parameters and initialise second order process      */

per = 0.05;        /* sampling period in seconds               */

g = 0.2;           /* damping factor, g <= 1 for no over shoot */
w = 20;   /*        resonant frequency                        */

/* difference equation for second order system is of the form */
/*      a1*y1 + a2*y2 + a3*y3 = b1*u1 + b2*u2 + b3*u3;       */
/*  if g is the damping factor and w the resonant frequency   */
/*  then the following coefficients may be obtained by using  */
/*            the bilinear transformation method              */

a1 = (per*per*w*w/4 - g*w*per + 1)*4/(per*per*w*w);

a2 = (per*per*w*w/2 - 2)*4/(per*per*w*w);

a3 = (per*per*w*w/4 + g*w*per  + 1)*4/(per*per*w*w);

b1 = 1;          b2 = 2;          b3 = 1;

c1 = b3/a3;      c2 = b2/a3;      c3 = b1/a3;

c4 = -a1/a3;     c5 = -a2/a3;

y1 = y2 = y3 = 0;
u1 = u2 = u3 = 0;
```

```
printf("\n%f\t%f\t%f\n\n",a1,a2,a3);
st = 150;                /* steady state angular velocity rads/sec*/
}

float A1_5(Qg)
float Qg;
{
float wg;

u3 = Qg;                 /*   applied input torque        */


y3 = c1*u1 + c2*u2 + c3*u3 + c4*y1 +c5*y2;


y1 = y2;
y2 = y3;

u1 = u2;
u2 = u3;

wg = st - y3/50;  /* steady state at 150 rads per second */

data.wg = wg;

  return(wg);
}
```

```
/*****************************************************************/
/* A1_2_1 CENTRELESS ROTATION - NUMERICAL MODEL       */
/*****************************************************************/

£include <stdio.h>
£include <math.h>
£include <defs.h>


/*REVRES
   model parameter for number of angular increments in 1/2 rev*/

extern long t;
extern RECORD data;


float Dia[REVRES];


static  int     Last;
static  int     n,m;
static  A1_2_1OUT A1_2_1out;              /* output structure */
static  float diameter;


SetA1_2_1()
{
FILE *fp;

int n;       /* loop counter */

fp = fopen("ring.dat","rb");

if(fp == 0)
        {
        printf("\nRing file not found");
        exit(1);
        }
if(fread(Dia,sizeof(float),REVRES,fp) != REVRES)
        {
        printf("\nError reading ring file");
        exit(1);
        }
/* set grinding wheel size */

A1_2_1out.diam.grindshaft = 600;  /* mm */

Last = 0;
}


A1_2_1OUT A1_2_1(gr,disp,velocity)
float gr;
float disp;
```

```
VELOCITY velocity;
{
float   dum;                    /*dummy argument variable     */
float   angpos;                 /* angular position of shaft  */
float   efeed;                  /* depth of cut    mm         */
float   diacurrent;             /* diameter variables mm      */
float   dialast;
float   diadiff;
float   diastep;
float   rpt;                    /* revs. per tenth of a second*/
int     current;
float     x;                            /*  Dia[] - gr  */

int     diff;
char *p;

rpt = 45/(200*Pi*4);        /*revs/tenth of sec*/

     /* angular position at time t in fraction of revolution */
angpos = modf(t*rpt,&dum);      /* dum not used */

                                /*nearest current angular index */
current = (int)(floor(angpos*REVRES+0.5));

if(current == REVRES) current = 0;

/* printf("%d\t",current);*/

x = Dia[current] - disp;


if ( x < gr )               /* check for wheel work contact */
        {
        A1_2_1out.contact = FALSE;
        }
     else
        {
        A1_2_1out.contact = TRUE;
        }



if  ( x  <= gr)
        {
        efeed=0;
        diameter = Dia[current];
        }
     else
        {
        diameter = gr +  disp;
        efeed = Dia[current]-diameter;
        Dia[current] = diameter;
        }
```

```
        /* interpolate from current to last */
        diff = current - Last;

        diacurrent = Dia[current];
        dialast    = Dia[Last];

        if(diff<0)
                {
                diff = diff + REVRES;
                }

                diadiff = dialast-diacurrent;
                diastep = diadiff/diff;

        for (n=1;n<diff;n++)
                {
                m= n+Last;
                if(m>REVRES-1) m = m - REVRES;
                Dia[m] = dialast = dialast - diastep;
                }
Last = current;

data.diameter = diameter;


A1_2_1out.cut = efeed;
A1_2_1out.diam.workshaft = diameter;

printf("**%5.4f**",disp);


return(A1_2_1out);

}
```

```
/*****************************************************************/
/* A1_2_2 GRINDING MACHINE STRUCTURAL DEFLECTION                 */
/*****************************************************************/

£include <stdio.h>
£include <defs.h>

extern long Ts;        /* sampling period        */
extern RECORD data;

static float y1,y2,y3; /* displacement vector    */
static float a1,a2,a3; /* diff. equation coeff. */
static float spr_con;

SetA1_2_2()
{
float g;         /* damping factor  g = 1 critical */
float w;         /* fundamental frequency */
float per;
/* Setup parameters and initialise second order process */

g = 1;           /* damping        */
w = 7;           /* resonant freq. */
spr_con = 0.004;
per = 0.05;

a1 = (2 - per*w*w)/(g*w + 1);
a2 =  (g*w - 1)/(g*w + 1);
a3 = per*w*w/(g*w + 1);

y1 = y2 = y3 = 0;

printf("\n%f\t%f\t%f\n\n",a1,a2,a3);
}

float A1_2_2(Xforce)
float Xforce;
{
float disp;
float u2;

u2 = spr_con*Xforce;

y1 = y2;   y2 = y3;   y3 = a1*y2 + a2*y1 + a3*u2;

disp = y3;

/*scale up to microns for filed value */
data.disp = 1000*disp;

return(disp);  /* total value of distortions due to Xforce*/
}
```

```
/*****************************************************************/
/* CREATE A PART (generate a component dimension array file)  */
/*****************************************************************/

/* Displays and modifies ring file circular or elliptical     */

£include <graf.h>
£include <stdio.h>
£include <math.h>
£include <defs.h>

main()
{
FILE    *fp;
float   ring[REVRES];
float   angle;
float   sinang;
float   cosang;
float   a;
float   b;
int     n;
char    c;

/* Display current ring file if any and
                                        prompt for modification */

fp = fopen("ring.dat","w+b");

if (fp == 0)
        {
        printf("Ring file not found");
        exit(1);
        }

if(fread(ring,sizeof(float),REVRES,fp) != REVRES)
        {
        printf("Warning!  error reading ring file");
        }
else
        {
        printf("\t   ARRAY MODEL OF COMPONENT");
        printf(" (diameters in milimetres)\n");
        printf("\t                        Unground part\n\n");
        for(n=0; n<REVRES; n++ )
        printf("%5.3f\t",ring[n]);
        }

printf("\n\n  Do you wish to alter dimensions y?\n");

c = getche();

if (c != 'y')
       {
```

```
        if (fclose(fp) == -1)
         printf("\nClose file error\n");
        return;
        }

printf("\nEnter max. daimeter  -  ");
scanf("%f",&a);

printf("\nEnter min. daimeter  -  ");
scanf("%f",&b);


for (n = 0; n<REVRES; n++)
        {
        angle    = Pi*n/REVRES;
        cosang   = cos(angle);
        sinang   = sin(angle);
        ring[n]  = sqrt(a*a*cosang*cosang + b*b*sinang*sinang);
        }

if(fseek(fp,0L,0) !=0)
        printf("Seek error !");

/*
for (n = 0; n<REVRES; n++)
        ring[n]=9.0;
*/

n = fwrite(ring,sizeof(float),REVRES,fp);

if (n !=REVRES )
        {
        printf("Error data not written to file %d",n);
        exit(1);
        }

if (fclose(fp) == -1)
        printf("\nClose file error\n");

}
```

```
/*****************************************************************/
/*          STRUCTURE AND CONSTANT DEFINITIONS                   */
/*****************************************************************/


£define REVRES                  200     /*dimensions across the*/
                        /* component at equal angular increments */

£define SHOE_FRICTION           0.1
£define SAMPLING_PERIOD         5               /*sec/100*/
£define Pi                      3.141593
£define TRUE                    1
£define FALSE                   0


                        /* Grinding force vector */

typedef struct vector{
                          float Xforce;
                          float Yforce;
                        } VECTOR;

                          /* Shaft torque vector */

typedef struct torque{
                          float grindshaft;
                          float workshaft;
                        } TORQUE;


typedef struct  diam{
                          float grindshaft;
                          float workshaft;
                       } DIAM;

typedef struct  velocity{
                          float grindshaft;
                          float workshaft;
                       } VELOCITY;

typedef struct  A1out{
                         DIAM diam;
                         VELOCITY velocity;
                         float cut;
                         int  contact;
                       } A1OUT;


typedef struct A2out{
                          float Xforce;
                          float Yforce;
                       } A2OUT;
```

```
typedef struct A1_2out{
                        DIAM diam;
                        float cut;
                        int contact;
                      } A1_2OUT;

typedef struct A1_2_1out{
                        DIAM diam;
                        float cut;
                        int contact;
                      } A1_2_1OUT;

                        /* Log record structure */
typedef struct record{
                        float gr;
                        float diameter;
                        float disp;
                        float wg;
                        float wb;
                     } RECORD;
```