MW.1 /94 SLC
IGrH Rep.

Powder Flow Instrumentation

and

Measurements

Daniel A. Ighodaro

A thesis submitted in partial fulfilment of the requirements of
The Nottingham Trent University for the degree of
Master of Philosophy

Department of Electrical and Electronic Engineering
Nottingham Trent University
Burton Street
Nottingham.

Collaborating Body

This research was carried out in collaboration with MINDON Engineering Ltd, Pinxton, Nottingham.

December 1994

ABSTRACT

The objective of the current project was to develop instrumentation for powder flow measurements. This involved reviewing previous research work and identifying areas of further developments.

A review of current theories, explaining the fundamentals of powder flow measurement techniques, was carried out in order to furnish the background knowledge for the current research studies.

The instrumentation is comprised of: a commercial powder flow pneumatic system (for circulating the powder flow in pipes); sensor and signal conditioner units; transputer network (for processing sensor signals); and IBM host transputer development system (for Occam software developments and simulations).

Substantial system developments were carried out during the research work:

The transputer network communications were improved to enable a reliable continuous communications on the links. A low cost, simple and reliable digital interface was designed to link the transputer network to the analogue signal conditioner and sensor units for the purpose of: selecting analogue signal channels, setting channel gains, triggering and controlling the charging of the powder flow. The interface has also provisions for some future powder flow control functions.

Noise problems in the system were reduced by improving the design of the analogue and digital parts of the instrumentation. The sensor head has been developed to neutralise tribo charge noise, resulting from friction during the powder flow.

Implementing parallel processing algorithms on the transputer powder flow instrumentation offers unique advantages over the conventional microprocessor system. The transputer's ability in implementing concurrent processes has been found suitable for processing the sensor signals. The transputer could be programmed in most standard languages but Occam was used during this work. Occam is efficient in exploiting the special properties of the transputer. Occam software modules successfully developed on an IBM transputer host development system were later ported to run the transputer network.

Furthermore, useful areas of further work have been described.

ACKNOWLEDGEMENTS

I wish to express my gratitude to my first Supervisor, Dr B.C.O'Neill, for introducing me to this project on powder flow instrumentation and measurements. I am equally grateful to him for his supervision and various supports (words and deeds) during the course of the project work.

My thanks to my second supervisor, Dr Steve Clark, for his support and contributions to the project work.

I am indeed grateful to my Head of Department, Eur Ing Professor P.G. Holmes for his advice and administrative support, where necessary, during the project work. My thanks, also, to departmental colleagues who may have contributed to the progress of this research work.

My thanks to Mindon Engineering Ltd, Nottingham for providing the Industrial support for this research program.

I wish to acknowledge, with thanks, the sympathies expressed and tremendous support from the departmental staff, during the loss of my research work through fire at the Stephenson building. My christian faith, encouragements from colleagues, family members, friends and the general public enabled me to overcome the fire devastation and restart the project.

Finally, my thanks to my family (Clara, Osa and Erhunmwun) for the domestic support during the project work.

CONTENTS

	ABSTRACT	i
	ACKNOWLEDGEMENTS	ii
	LIST OF FIGURES	vi
1.0	**INTRODUCTION**	1
1.1	Industrial applications	3
1.1.1	Powder paint coating	3
1.2	The project chronological development	5
1.3	Project objectives	6
1.4	Structure of thesis	6
1.5	References	7
2.0	**REVIEW OF POWDER MEASUREMENT TECHNIQUES**	8
2.1	Introduction	8
2.2	Inferential mass flow measurement	8
2.3	Measurement techniques	9
2.3.1	The pulse charge injection technique	10
2.3.2	The tribo-charging technique	11
2.3.3	The flow noise technique	12
2.3.4	The capacitance technique	14
2.3.5	The impact technique	16
2.3.6	The doppler noise technique	17
2.3.7	The nuclear tracer technique	19
2.3.8	The optical technique	20
2.3.9	The thermal technique	21
2.4	References	22
3.0	**POWDER COATING INSTRUMENTATION; CHRONOLOGICAL DEV.**	25
3.1	Introduction	25
3.2	Phase 1 (B.C.O'Neill & C. Willis)	25
3.2.1	Pulse charge technique experimentation	26
3.2.2	The experimental rig	26
3.2.3	The experimental results	30
3.2.4	Experimental conclusions	32
3.2.5	Further work	34
3.3	Phase 2 (B.C.O'Neill & E. Mills)	38
3.3.1	The need for a microprocessor system	38
3.3.2	The transputer instrumentation	39
3.3.2.1	The data acquisition unit	39

3.3.2.2 The data processing array 40
3.3.2.3 The controller unit 41
3.3.3 Experimental results 41
3.3.3.1 Analyses of mass flowrate 42
3.3.4 Experimental conclusions 44
3.3.5 Further work 44

3.4 References 69

4.0 Hardware Design 71

4.1 Introduction 71
4.2 The system design principles 71
4.3 System function 72
4.4 Current system detailed block diagram 72
4.4.1 Introduction 72
4.4.2 Pneumatic powder flow subsystem 74
4.4.3 The signal conditioner 77
4.4.4 The analog to digital converter 80
4.4.5 Digital interface 81
4.4.6 The transputer unit 82
4.5 References 94

5.0 The software design 95

5.1 Introduction 95
5.2 The system design principles 95
5.2.1 Initial verification of software tasks 96
5.2.2 Coding of the various modules 96
5.2.3 Testing of the modules 96
5.2.4 Integration of the modules 97
5.3 The system dataflow 97
5.3.1 Data capture and distribution 98
5.3.2 Signal peak detection and charge int. 105
5.3.3 Data storage and processing 105
5.3.4 Pc Host User interface 106
5.4 References 115

6.0 Discussions of research work achievements 116

6.1 The Hardware changes 116
6.1.1 Powder flow subsystem 116
6.1.2 Signal conditioner 118
6.1.3 Digital interface 118
6.1.4 Communication needs 119
6.2 The software changes 119
6.2.1 Modularisation of software modules 119
6.2.2 Comprehensive documentation of software 119
6.2.3 Channel protocol 120
6.2.4 Suitable software for digital interface 120
6.2.5 Test programs for link communication 120
6.3 Overall system software 121

7.0 Conclusions and further work 122

7.1 Conclusions 122
7.1.1 Review of flow measurement techniques 122
7.1.2 Improved link communication 123
7.1.3 Improved digital interface unit 123
7.1.4 Improved analog interface 123
7.1.5 Software developments 123
7.1.6 Software documentation 124
7.2 Hardware further work 124
7.2.1 Assessment of pneumatic flow subsystem 124
7.2.2 Reassessment of C-V converter 124
7.2.3 Input buffer programmable gain links 125
7.3 Software further work 126
7.4 Installation and verification of the system 126
7.4.1 Performance assessment of hardware 127
7.4.2 Performance of the system software 127
7.4.3 Further studies of the flow materials 128
7.5 Control strategies 128

TABLE OF CONTENTS FOR APPENDICES 130

Contents 130
LIST OF FIGURES FOR APPENDICES 132
Appendix 1 A1.0
Appendix 2 B1.1
Appendix 3 C1.1

LIST OF FIGURES

Chapter 3

Figure

3.1	Experimental Rig Block diagram	48
3.2	Powder Hopper & Venturi	49
3.3	Faraday 'caged' sensor unit	50
3.4	System Electronics Block diagram	51
3.5	Injection Electrode current waveform	52
3.6	Sensor 2 current waveform	53
3.7	Charge waveform	54
3.8a	Low mass flow rate model	55
3.8b	Medium flow rate model	56
3.8c	High mass flow rate model	57
3.9	Experimental setup block diagram	58
3.10	Transputer nodes configuration	59
3.11a	Low mass flow rate model (Reg.1)	60
3.11b	Medium mass flow rate model(New Reg.1)	61
3.11c	High mass flow rate model(Reg.2a)	62
3.11d	Medium mass flow rate model(New Reg.2a)	63
3.11e	High mass flow rate model(Reg.2b)	64
3.11f	High mass flow rate model(New Reg.2b)	65
3.11g	High mass flow rate model(Reg.2c)	66
3.11h	High mass flow rate model(New Reg.2c)	67
3.11i	High mass flow rate model(White Reg.1)	68

Chapter 4

Figure

4.1	The flow system development cycle	84
4.2a	The System block diagram	85
4.2b	The controllable pneumatic processes	86
4.3	Pneumatic powder flow subsystem	87
4.4	The redeveloped sensor unit	88
4.5	The cyclone's vortex activities	89
4.6	Current to Voltage converter circuit	90
4.7	Amp.\buffering of sensor signals	91
4.8	Programmable gain links	92
4.9	The Digital interface	93

Chapter 5

Figure

5.1	The Software development path	108
5.2	Data flow diagram	109
5.3	Typical successful injection waveform	110
Flowchart		
5.3.1	Data acquisition	111

5.3.2 Peak detection 112
5.3.3 Data store and message passing 113
5.3.4 Pc-Host data passing\ display 114

1.0 INTRODUCTION

Various pneumatic processes require reliable instrumentation to effectively measure the quantity of powder or particles transported in pipes. Such instrumentation, often measures mass flowrate and the velocity of powder flow. The main aim of this research is to develop powder flow instrumentation for velocity and mass flow rate measurements.

There are various techniques for measuring powder mass flowrates. These techniques can be classified into intrusive and non-intrusive categories. Intrusive techniques, which impede powder flow in pipes, will not be considered. The non-intrusive techniques which do not obstruct the flow of the powder will be reviewed, with the objective of determining the most appropriate technique suitable for the current research work. Non-intrusive methods include: pulse charge injection, flow noise, capacitance, doppler, nuclear tracers, optical, and thermal techniques. These have various relative advantages and disadvantages. A full review of these techniques is provided in the next chapter.

The flow noise technique relies on tribo-charging of powder flow particles. The mass flowrate is related to the flow noise produced during the tribo charging process. Though non-intrusive, tribo-charging is not easily controllable.

Capacitance techniques monitor the fluctuation of the powder flow density (capacitance noise) and relate this to the mass flow rate. However, the flow density dependent capacitance noise is not easily separated from other sources of flow noise in the pipe.

The doppler technique uses the doppler shift of ultrasonic signals, injected into the powder flow, to measure the flow velocity. The doppler technique does not easily measure mass flow rate.

The impact technique relies on particle impact on a piezo-electric transducer to assess the flow density. This technique is essentially a velocimeter and requires a very sensitive transducer for its operation.

The most suitable non-intrusive technique is that of pulse charge injection. It operates by injecting a controlled charge into the powder flow and monitoring the quantity of charged powder particles induced on downstream sensors. The flow velocity and inferential mass flow rates [2] can be obtained by processing the sensor signals. Absolute mass flow rates can also be measured. This is the basis of both the work carried out in the Nottingham Trent University Research group and the research work described in this thesis.

1.1 Industrial applications

Various industrial applications could benefit from a powder flow instrument based on the pulse charge technique. Typical applications are in: the electrostatic powder coating industries; industrial particle pollution control in the power generating industries; pneumatic drug production. Controlled mass flow rates are desired in these applications.

1.1.1 Powder paint coating

Powder coating is extensively used in the paint industries for metal finishings.

There are two main alternative industrial methods of coating a metallic substance: electrostatic powder coating or solvent-based paint. Powder coating was first introduced in the early 1960's [1]. In view of the merits of powder coating, many studies have been carried out to further enhance the efficiency of the coating process.

The process of powder coating involves spraying pneumatically conveyed charged powder particles on to an electrically grounded target object. This process utilises the basic principle of electrostatic force between two charged bodies. To ensure uniformity of the coating process, the target object is oven heated for fine paint finishing. A single coating is often sufficient for painting metal.

Powder materials commonly used are: epoxy, polyester and nylon (or a mixture of these). These materials possess the necessary properties of chemical inertness and mechanical adherence (during coating).

The advantages [1] of Electrostatic powder coating over solvent based paints are highlighted below:

1. The chemical solvents used in conventional liquid paints can be explosive.
2. The ventilation requirement is substantially reduced.
3. Adequate coating can be obtained from one level of powder coating: a minimum of three coatings is required in the solvent based process for an equivalent quality.
4. Surplus powder in the painting process can be recycled.

The disadvantages of powder coatings are as follows:

1. 'On-line' colour changes are difficult.
2. Recycled powder from an uncleaned booth could be contaminated and rendered unsuitable for use.
3. Limited varieties of powder materials can be used.
4. Not all powder materials can be recycled, making the system uneconomical.

Though 1 & 3 are disadvantages, they could easily be overcome by careful initial specification of the materials and colours required for a particular task. Most users are more interested in the fine metal paint finish provided by the powder coating process. Careful operational handling of the spray booth would largely solve problem 2. Problem 4 is a pointer to the strong need for a conservative system, to minimise the amount of powder wasted during the coating process.

An efficient instrument is needed to regulate the powder flow to avoid incidents of the system over or under spraying. Over-spraying has to be avoided because grit blasting would have to be used to remove excess powder coating.

The current research will, therefore, focus on the task of producing reliable instrumentation, which could be an asset for the powder coating industries.

1.2 The project's chronological development

The chronological details of previous research developments, carried out at Nottingham Trent University, on the pulse charge technique, are explained in chapter 3. The details include: the initiation of the project work; investigation of the pulse charge technique; and the implementation of the microprocessor based system for continuous measurements of mass flow rates.

The future emphasis is to produce a closed loop control system for the powder flow.

1.3 Project objectives

The objectives of the current work were:

1. To review the work carried out by previous Researchers at Nottingham Trent University on the pulse charge technique. Initial work, by B.C.O'Neill and C.Willis, was on continuous mass flow rate and velocity measurements (using analogue processing techniques) of pneumatically conveyed powder. B.C.O'Neill and E.Mills worked on utilising transputer instrumentation (a microprocessor parallel processing system for particle flow).

2. To Identify new areas of hardware and software to improve the transputer instrumentation.

3. To produce a reliable instrument for powder flow rate measurements.

4. To produce a modularised, and fully documented, system software.

1.4 Structure of the thesis

The thesis is comprised of the following: In chapter 2,

various techniques for powder flow measurements are reviewed. Chapter 3 details the chronological history of the work carried out by the previous researchers at Nottingham Trent University.

Chapter 4 describes the hardware developments of the powder flow instrumentation carried out during the current research work.

The software developments are explained in chapter 5.

Chapter 6 discusses the research work achievements.
Conclusions and areas of further work are stated in Chapter 7.

The attached appendices 1,2, and 3 provide further information on the hardware and software developments explained in chapters 4 and 5.

1.5 References

1. **Willis, C.A:** "Continuous Mass flow Rate and Velocity measurement of Pneumatically conveyed powder"
PhD thesis, Trent Polytechnic, Nottingham 1984, P1-8

2. **DeCarlo, J.P:** "Fundamentals of flow measurement" Instrument Society of America, ISBN 0-87664-627-5, P225-227

2.0 REVIEW OF POWDER MEASUREMENT TECHNIQUES

2.1 Introduction

The current study of powder flow instrumentation involves a system predominantly based on the pulse charge technique (discussed at length in chapter 3) but a comparative analytical review of other techniques is provided here.

2.2 Inferential mass flow measurement

Mass flow rate is expressed as the rate of change of mass, m, with respect to time within a defined medium. Mathematically expressed as:

$$dm/dt = k \qquad (2.1)$$

where k is the mass flow rate. Integrating this:

$$m = k(t_2 - t_1) \qquad (2.2)$$

where (t_2-t_1) is the time difference between flow measurements. Also:

$$m = q.l.a \qquad (2.3)$$

where q, l, a are the flow density, medium flow length & cross- sectional area respectively.

Equating 2.2 to 2.3:

$$k = q.l.a/(t_2 - t_1) \qquad (2.4)$$

Equation 2.4 is the basis of the mass flowrate measurement: a function of the density and velocity of powder flow. Direct

measurement of velocity is possible for a sensor purposely built to measure time of flight between two points on the flow. The flow density parameter, q, has to be inferred or estimated through an indirect method, especially in turbulent powder flow. Research studies on mass flow rate require information from the combination of two (velocity and density) measurements. This approach is called inferential mass flow measurement [2a].

Inferential mass flow measurement is different from absolute mass flow measurement. Absolute mass flow rate is the direct measurement of mass over a specified period and this process does not take into account the state of the powder flow, which is required in powder flow applications. The current research is based on inferential flow measurements. Based on this preliminary analysis, various techniques aimed at measuring mass flow rate will be reviewed.

2.3 Measurement techniques

The various techniques used for mass flowrate measurements fall into two broad categories: energy extractive and energy additive [2b]. The energy extractive methods are obstructive or intrusive techniques whilst the additive counterparts are non-intrusive techniques. In an intrusive technique the powder flow is sensed by obstructing the flow stream. A typical example of an intrusive technique is the pressure sensor, which is not often used in powder coating because of its

obstructive nature. In the energy additive techniques some form of energy is introduced into the flow and the effect of the powder particles on this energy is sensed downstream by the appropriate sensors. This is the trend adopted by many research workers because of its non-intrusive nature. It is also the approach adopted in this work.

Non-intrusive [15] techniques could be subdivided into various categories. These include: electrical, ultrasonic, optical and thermal measurement techniques.

2.3.1 The pulse charge injection technique

The current instrumentation is based on the 'pulse charge injection technique' of charging the powder particles. The decision to adopt this technique is based on the following attributes [1a]:

1. It is able to provide a quantitative measurement of absolute mass flowrate.
2. It offers negligible resistance to the particle flow.
3. It is well behaved and stable, hence reliable.

A flowmeter must be stable and accurate over a wide dynamic regime of particle flow measurements. The readings should not fluctuate, irrespective of the humidity of the powder and the powder fluid bed level. It is necessary for the flowmeter to be reliable, irrespective of the environmental conditions.

This technique is explained in detail in chapter 3 (Section 3.1). Briefly, the velocity component of the flow is obtained by using the time of flight of charged powder particles between two sensors. The flow density is empirically inferred from the ratio of the sensed charge downstream, to the injection charge sensed by the first sensor. The mass flowrate can then be derived from the combined measurement of velocity and inferred flow density.

2.3.2 The tribo-charging technique

The tribo-charging technique [4,17] is an electrical method of measuring the mass flowrate. The concept of tribo-charging is described here because of the precautionary measures taken in the current study to reduce the negative effects of tribo charging. Tribo-charging is undesirable in the pulse charge technique because it creates 'noise' on the received signals.

Frictional contact between two substances of different work functions produces positive and negative charges. As a result, powder particles in contact with the walls of a pipe get charged by friction. If the spray gun is designed in such a way that the powder particles produce sufficient contact with the gun, then the particles become positively charged while the gun absorbs the negatively charged electrons. A high electric field strength can be generated from this process [17]. However, neutralisation of the charged bodies can take place easily if the charges are not separated quickly. During

the separation process, the charges can only remain at the respective surfaces of the charging bodies if the separation time is less than the relaxation time of the opposite charges. In this technique, the charging body (corona electrode) is normally earthed, hence, leaving the powder particles charged.

The disadvantage of this method is that the charge quantity produced is not easily controllable [4]. Furthermore, to achieve much frictional contact, powder particles have to travel at a reasonable velocity. Hence, it is difficult to obtain effective results in a low flow regime. Typically amine cured epoxy powders would charge adequately provided the powder is conveyed at relatively high velocities down the gun [4]. At high velocities, positive currents between 2μA and 4μA could be obtained from the flow [4]. Another disadvantage of the tribo-charging technique is that the charging process may continue indefinitely: this results in charge particle deposition on the charging body which can cause the process to be stopped [4].

The flow noise technique (discussed below) operates on the basis of tribo-charging.

2.3.3 The flow noise technique

This is a non-intrusive method, utilising tribo-charging, which has been developed for measuring the average velocity of a flow of charged particles in pipes of pneumatic systems.

Initial work by Gajewski [5,6] on powder flow determined the velocity of the powder flow using a correlation technique.

In this technique, the particles are charged by frictional contact [5,6]. The electrical signals induced in two transducers are used in the non-intrusive measurement of the flow velocity [6]. The transducers, two ring measuring probes aligned and separated by a "correlated" distance [6], are connected to a two-channel storage oscilloscope. The transit time between the voltages induced at the two correlated points (probes) could be measured on the oscilloscope and converted to a velocity reading.

Further work was carried out resulting in an electrostatic method [7,8] for measuring mass flowrate and velocity parameters. The method involves a computer-based measuring system, based on sensing the flow noise. The flow noise is a mixture of average flow superimposed on irregular flow. Experimentally, it was found that there was a direct correlation between the mass flow rate and the flow noise. It was also found that parameters of the flow were statistically dependent on each other (for example the velocity of the flow depends on the volume loading, particle size, etc). The microcomputer based measuring system controls the experiments, processes data (by a cross-correlation method) and displays the data. From the processed data, velocity and mass flow rate information are obtained.

Correlation methods for determination of the flow velocity requires the two or more correlated points to be close, such that the signals correlated are nearly identical. Any disturbance in the flow, between these points, could produce spurious signals which render correlation difficult. Gajewski observed irregularities in the variations of signals at the correlated points which were probably caused by capacitive coupling between the probes and additional disturbances in the particle flow [6].

2.3.4 The capacitance technique

In the capacitance technique, developed by Beck et al [9,10], the capacitance noise of the powder flow is the object of study. Beck used a correlation technique to obtain velocity and mass flow rate information from capacitance noise (measured using a capacitance transducer).

This technique expresses the mass flowrate as being related to the amplitudes of the random fluctuations of capacitance of the particle flow. The combination of the random fluctuations is referred to as the capacitance noise. The variation of the sensor signals with the capacitance noise is used to produce amplitude modulation on the sensor head [1b].

The capacitance is proportional to the permittivity of a medium which changes with powder flow density variations.

Changing powder flow density is hence converted to amplitude modulation or frequency modulation, with the modulation index indicating some parameter of the flow (such as the mass flow rate).

A frequency-modulated transducer for on-line measurement of two component flow was developed by Green et al [11,12]. This technique involves frequency modulation of the sensor signals. In the frequency modulation technique, the modulating frequency varies with the capacitance noise. The frequency generated from the sensor is converted into a voltage reading. The dc component of the voltage is blocked and the ac component provides the flow related signal. A cross correlation of the ac signals provides the velocity information. Green et al rectified and averaged the ac signals. The voltage output of the rectifier and averager was converted into a current signal which indicated the mass flow rates.

However, Willis [1b] observed that the frequency of the random fluctuations is related not only to the powder mass flowrate but also to other parameters of the system such as the hopper powder level (fluidised powder container level) and the feed pipe length. As a result, the capacitance noise signal (related to mass flow rate) is affected by other noise sources generated within the powder conveying equipment.

The main advantage of the capacitance noise technique is that

it is a non-intrusive method. It is purposely designed not to obstruct the path of flow. The difficulty in separating density dependent capacitance noise from other sources of powder flow noise poses a problem in the digital signal processing of the signals, in order to obtain the true mass flow rates.

2.3.5 The impact technique

This is an experimental technique to determine the local flow characteristics in pneumatic conveyors. The technique [13] enables the measurement of local particle flow rates, local average velocity and local velocity distribution among individual particles.

This technique involves counting the number of particle impacts on a piezo-electric transducer. The number of particles impacting per unit time, at a radial position, is proportional to the local flux. A group of random signals generated by the transducer during the collisions (using polystyrene spheres) were fed to a counter which was used to determine the collision rate and the distribution of time between two consecutive collisions. Using this data, from the transducer, the system was calibrated to give velocity measurements. A calibrated transducer (for the particles in use) could be used to determine the profiles of particle flux, velocity and concentration [13].

Mann et al [13] observed that when the particles are uniformly spherical, the velocity distribution among the particles can be determined and when the particles are not identical, the velocity distribution cannot be determined. Furthermore, the success of this technique requires a careful selection of a robust and sensitive transducer which will endure many hours of operation.

2.3.6 The doppler noise technique

This technique involves the transmission of ultra-sonic waves into a flow medium and observing the doppler shift in frequency from the transmitted wave to that received [14,15]. The doppler technique is essentially aimed at producing a velocimeter for the powder flow.

In measuring, say the velocity of a vehicle (using the doppler technique), the moving target is large when compared to a particle. In the case of particle flow, many small targets, which move with different velocities, are involved. As a result, the analysis is far more complex [14] than that of measuring the velocity of a car. However, the same principle of doppler frequency shift is still applicable. In the doppler technique, the sensor receives multiple reflections from the particle flow. The centre of the particle flow reflects more of the ultrasonic incident rays than the particles close to the pipe walls [2c]. A mathematical treatment of the doppler frequency shift, applied to the flow, involves integrating the

frequency shifts from the various particles within a defined volume (such as a flow pipe). The amplitude spectrum of the reflected wave is obtained from the integration of the various signal amplitudes reflected from the individual particles [13]. Signal processing is used to weight each frequency component before calculating an integrated flow velocity. This analysis is possible with homogeneous media; the difficulty arises when the medium is inhomogeneous [2c]. The sound beam might not be able to penetrate down to the particles flowing close to the wall and might be reflected strongly by the central flowing particles and an erroneous high flow rate would be recorded.

Thorn et al [15] explored various doppler techniques (lasers, microwave and ultrasonic) in the velocity measurement of particle flow.

The doppler technique has some disadvantages. Firstly, it is essentially a velocimeter. Mass flow rate can only be determined from a combination of velocity and flow density measurements. Furthermore, the transducer source on the pipe wall can be blocked by deposition of the powder particles, therefore degrading performance. The doppler flowmeter relies strongly on the density of particles to reflect signals to the sensor and this is a condition desired for normal performance. In the ultrasonic methods, difficulties arise from the transmission of the sound waves through the walls of the pipe

to the transducer. In particular, this method is not suitable for powder paints which have clustering tendencies where the high density of the flow tends to block the 'windows' on the pipe walls, where the sound waves pass to the transducer.

2.3.7 The nuclear tracer technique

The tracer technique [15] involves introducing recognisable markers (tracers) into the flowing solid stream and timing their progress between two points. Radio-active particles could be used as tracers. These tracers are commonly produced by irradiating, with neutrons, a sample of the bulk material to be conveyed. This activates naturally occurring isotopes in the material [15]. As a result, the irradiated material retains its original physical and surface properties but now emits radiation. A pulse of radioactive particles are injected into the stream and the resultant cloud of radiation can be detected at two points downstream using scintillation detectors. The velocity of the flow can then be determined from the time of flight.

The major disadvantage of the radioactive tracer method is the problem of contamination [15]. It is desirable that the tracer particles are removed from the conveying system after passing through the measurement section. This is difficult to achieve, resulting in a large proportion of the conveyed bulk material being wasted. Furthermore this technique only measures flow velocity and not mass flow rate.

2.3.8 The Optical technique

This technique involves the transmission of a light beam through the particle flow. The interceptions of the beam are recorded by optical fibre detectors. This technique was used in the pneumatic transportation of coal [16a].

In its basic operation, a beam of light is projected across the powder transport system. The beam is intercepted by the flow and the discrete interceptions of the stream are recorded on optical sensors. These are later translated into a train of digital pulses. When the flow velocity is steady, there is no alteration in the received optical frequency. However, a variation of the flow velocity would result in the corresponding variation in the frequency of interceptions. The velocity of the flow can therefore be determined by processing the frequencies received. The light sources and detectors are placed orthogonally to each other. A digital counter is used to estimate the density of flow using an averaging procedure.

The main disadvantage of this technique is that there are severe restrictions to the density and particle sizes that can be accommodated. A densely populated particle stream might result in the total blocking of the beam, highly degrading the performance of the flow system. Furthermore, the sources of light are mounted on the walls of the pipe and if these sources become covered, due to the coating, then the process

would cease [9].

2.3.9 The thermal technique

Work carried out on this technique explored various parameters of thermal energy to estimate the particle flow [2d,16b] . This is essentially a non-intrusive method because the path of flow is not obstructed by the sensor devices. A differential temperature sensing method is used to estimate the mass flowrate in the region of 0-1000kg/h [16b]. Heat, injected into a dense phase aluminium powder flow, is thermally sensed at points down the flow path and the differential temperatures are recorded. In this process, the objective is to find the relationship between the mass flow and the heat transfer coefficient.

2.4 References

1a, 1b. Willis, C.A: "Continuous Mass flow Rate and Velocity measurement of Pneumatically conveyed powder" PhD thesis of Trent Polytechnic, Nottingham, 1984, P2-3, P4-5

2a, 2b, 2c, 2d. DeCarlo, J.P: "Fundamentals of flow measurement" Instrument Society of America, ISBN 0-87664-627-5, P203-207, P9-22, P167-170, P170-179

3. Beck, M.S., Green, R.G. and Thorn, R: "Non-intrusive measurement of solids mass flow in pneumatic conveying" J.Phys. E:Sci. Instrum. Vol 20, 1987, P835-840

4. O'Neill, B.C: "Arthur Holden Fellowship report on a comparative study of powders in the Electrostatic spraying process. Tribo charging. P28-29" Southampton University, Southampton

5. Gajewski, J.B: "Inductive non-contact method for measuring velocity" J.Phys. E:Sci. Instrum. Vol.16, 1983, P622-624

6. Gajewski, J.B: "Static electricity and measurement of the parameters of a flow in pneumatic conveyances" Inst.Phys.Conf.Ser.No.85, Electrostatics 1987, Oxford, P285-290

7. Gajewski, J.B., Glod, B. and Kala, W: "Electrostatic method for measuring the Two-phase pipe Flow parameters"
IEEE 1990, P897-902

8. Gajewski, J.B., Glod, B. and Kala, W: "Electrostatic flow meter for measuring the two-phase flow parameters in pneumatic transport-results of preliminary tests"
Inst.Phys.Conf.Ser.No.118, Electrostatics 1991, Oxford, P159-164

9. Beck, M.S. and Wainwright, N: "Current Industrial Methods of Solids Detection and Measurement"
Powder Technology Vol. 2, 1968, P189-197

10. Beck, M.S., Draine, J and Plaskowski, A: "Particle Velocity and Mass Flow Measurements in Pneumatic Conveyors"
Powder Technology Vol.2, 1968, P269-277

11. Green, R.G: "Frequency modulated transducer for Gas/Solids flow measurements"
Proc. of IMEKO 7 Practical Meas. for improving Effi. Vol. BFL 251, 1976, P387-391

12. Green, R.G. and Cunniffe, J.M: "Frequency modulated capacitance transducer for On-line Measurement of Two-component fluid flow"

Measurement Vol.1, No.4, 1983, P191-195

13. **Mann, U. and Crossby, E.J:** "Flow measurements of Coarse Particles in Pneumatic Conveyors"
Ind. Eng. Chem, Process Des. Dev., Vol.16, No. 1, P9-13

14. **Bregman, R:** "A Study of the development of an Industrial Doppler Flowmeter"
Flow measurements, Flowmeko, 1983, P171-177
Editor; Spencer, E. A.

15. **Thorn, R., Beck, M.S. and Green R.G:** "Non-intrusive measurement of solids mass flow in pneumatic conveying"
Journal of Physics E: Sci. Instrum. Vol 15, 1982
P1131-1139

16a, 16b. **Mills, E:** "Transputer Instrumentation for Particle Flow Measurements" PhD thesis of Trent Polytechnic, Nottingham 1989, P18-19, P18

17. **Klebar, W., Bauch, H. and Auerbach D:** "Electrostatic powder spraying by means of electrokinetic charging"
Inst. Phys. Conf. Ser. No. 85, Electrostatics 1987, Oxford, P33-38

3.0 POWDER COATING INSTRUMENTATION; CHRONOLOGICAL DEVELOPMENTS

3.1 Introduction

Work in this department on powder coating instrumentation involved several phases of development: initiation of the project by Dr B.C.O'Neill, investigation of the pulse charge technique based instrumentation for the particle flow measurements [1a] and the implementation of a microprocessor based system for continuous measurements of mass flow rates [4a]. The main current emphasis is to produce a closed loop control system for the powder flow.

3.2 Phase 1 (B.C.O'Neill & C. Willis)

The novel pulse charge technique was initiated by Dr B.C. O'Neill and jointly investigated by B.C.O'Neill and Chris Willis in the early 1980's [Continuous mass flow rate and Velocity measurements of pneumatically conveyed powder, Ph.D Thesis, Trent Polytechnic, 1984]. The inherent advantages of this method over the alternative techniques were discussed in chapter 2.

Before initial investigations of the pulse charge technique, the two researchers examined the capacitance noise technique [1a]. It was found that the capacitance flow noise was not a true reflection of the powder flowrate: the flow noise was

dependent on other parameters, for example the hopper powder level and the feed pipe length. In addition, the capacitance noise signal (related to the powder flowrate signal) was found to be below the level of other noise sources generated within the powder pneumatic equipment. Hence, this avenue of investigation was stopped.

3.2.1 Pulse charge technique experimentation

The experimental objectives of the powder flow instrumentation based on the pulse charge technique were as follows [1b]:

1. To maintain a controlled, pulsed, charge injection into the powder.
2. To sense the injected charge on the powder at two detection electrodes.
3. To continuously monitor the powder velocity.
4. To maintain constant flow conditions over a set of readings.
5. To determine the mass flow rate under the conditions above.
6. To control the powder flow rate.

3.2.2 The experimental rig

The experimental equipment used [1b] is shown in figure 3.1. The powder hopper & venturi, given in figure 3.2, and a conveyance system (for powder/air recirculation path) are of

a standard commercial type and constitute part of the experimental rig.

The powder hopper contains a fluid air bed to aerate the powder, to enable its fluidisation. Two air controls (shown in figure 3.2) were used to set the powder flow. The primary control feeds the air to the venturi at the head of the hopper feed pipe; this creates a partial vacuum to suck the fluidised powder from the hopper. The secondary air control adjusts the air to powder ratio by introducing air to the head of the hopper feed pipe; this reduces the primary air suction component to the hopper.

The system was devised to convey powder pneumatically, from the hopper, through a purposely designed charge injector and current sensor detection unit. The powder was separated from the air and then returned to the hopper. With this method it was possible to run the system continuously round the loop for several hours. It was also possible to mechanically switch the powder to a cyclone for absolute mass flowrate measurement. This measurement can be used to assess the efficiency of the inferred mass flowrate obtained from the system electronics connected to the powder flow sensor heads.

The sensor unit (with four sensors (I0, I1, I2, I3)) is shown in figure 3.3. The charging unit, on the sensor unit, comprises of an injection electrode (a needle located along the axis of the pipe) and a collecting electrode, 10, which

forms part of the pipe wall. The powder was charged by applying a high voltage between the injecting electrode and the collecting electrode (I0). The optimum operating voltage pulse was 6kV, for 1ms duration, repeated every 20ms. The charged powder was detected downstream by two electrodes/sensors (I1 & I2). A fourth electrode (S4) was used to neutralise excessively charged powder. I1 and I2 acted as 'Faraday cages' to detect:

1. charge passing through the pipe,
2. charge neutralised on the pipe,

These charges were measured using analogue circuitry attached to the sensor heads. Initially a charge amplifier was used to detect the charges [3] but this was found to be unsuitable, since any small build up of charge on the electrodes, between pulses, interferes with the recordings of the pulses and drives the amplifier off-scale. Furthermore, only short traces of single pulses were obtained and no quantitative measurements were recorded with the charge detector. A low impedance current amplifier was then used to detect the charges, by electrostatic induction. The current amplifier was only slightly affected by any discharge on the electrode. The current amplifiers, connected to peak detectors, were used in the measurement of the mass flow rates. Peak detectors were used to track the peak amplitudes of the current pulses [2,3]

A block diagram of the system electronics is shown in figure

3.4. A brief description of its operation is provided here to explain why the need arose in phase 2 to implement a microprocessor based system for processing the flow signals. The main stages of the system electronics and their operations are described as follows:

1. **Current to voltage convertors:** These convert the currents, from the high tension supply (HT) and those induced on the sensors by the charged powder particles, into voltages for subsequent analogue signal processing.

2. **Peak detectors:** To obtain a mean-peak-height detection on each sensor the charging time constant was designed to be of the order of the pulse repetition period (20ms duration).

3. **HT trigger circuit:** This circuit generated the trigger signal for the HT supply so that the period of the injection current is constant. The pulse time is expressed in terms of the capacitive and injection current pulse times:

$$t_p = t_c + t_{inj}$$

where t_p is the pulse time; t_c is the capacitive pulse time; t_{inj} is the injection pulse time.

4. **Injection current generator:** The HT current is

composed of two components: capacitive and injection currents. The injection current needs to be controlled. Hence, the injection current generator is used as a difference circuit to separate the injection current from the capacitive current.

5. **Powder velocity circuits:** Analogue comparators were used to produce a pulse whose width (t) equals the time difference between the arrival of the charged powder particles on I1 & I2. The pulse is integrated to produce a ramp voltage proportional to the period (t) of the pulse. This peak value of the ramp voltage is held on a peak detector for 20ms duration (the HT pulse cycle). The velocity of the powder is deduced as being proportional to 1/t.

3.2.3 The experimental results

A typical current waveform obtained from the injection electrode (I0) is shown in figure 3.5(a). The first component of the waveform is the capacitive current and the sharp edge is due to the onset of charge injection into the powder/air stream. The capacitive current waveform (figure 3.5b) was simulated by electronic circuitry and subtracted from the I0 current waveform to obtain the injection current (figure 3.5c).

The current waveforms obtained from the first downstream

sensor are shown in figures 3.6a, 3.6b and 3.6c for low, medium and high flow rates respectively (in the range of 0.5 to 6.0 gm/s). The current waveform of the second sensor is similar in shape to that of the first but is smaller in magnitude because it is further from the injection electrode. Figure 3.7 shows the charge waveforms for the same flow conditions, obtained from using charge amplifiers on the sensor head.

The current and charge waveforms were compared with traces produced from an analytical model of the system [2] and the result shows a continuous process of charge powder deposition and re-entrainment on to and off the pipe walls. The rate of deposition and re-entrainment varies with powder flowrate.

Data obtained from the sensor currents were analyzed on a mainframe computer, using an interactive statistical modelling package [7] to produce the following models for the low, medium and high flow rates [1c]:

$R_1 = (I_2/I_{inj})$ for low flow rates

$D_{22} = ((I_2-I_3)/I_{inj})^{1.4}$ for medium flow rates

$S_{22} = ((I_2 / 1_1)^{0.66})$ x Velocity

where I_{inj} is the injection current

I_2, I_3 are sensor 2 and 3 currents respectively.

R_1, D_{22} and S_{22} are models for low, medium and high mass flow rates respectively.

The mass flowrate plots for low, medium and high flow rates using the models are shown in figures 3.7 a, 3.7b and 3.7c. R_1, D_{22} and S_{22} were the parameter models for the various flow regions (low, medium and high respectively). Different models for the various regions were used because of the difficulties in representing the three different flow regions by a common model.

Several empirical models [1c,3] were tested to obtain one model to predict the mass flowrate for a particular region. In these analyses, the data was separated into three models (low, medium and high mass flow rates). The R_1 model was used to correlate the low mass flow rates in the range 0.6 to 2.8 g/s. Over 95% of the data were within 10% error margin. The D_{22} model (medium flow rates) represented mass flow rates in the range 0.6 to 4.0 g/s. Over 85% of the data were within the 10% error margin. The S_{22} model (high flow rates) correlated mass flow rates in the range 3.0 to 8.0 g/s and over 90% of the data were within the 10% error margin. The separation of data into regions (low, medium and high flowrate) was carried out automatically by the statistical modelling package [1c].

3.2.4 Experimental conclusions

The following were the conclusions of the work on the pulse charge technique [1d]:

A. The sensor unit

1. A novel charge injection unit had been produced to inject a controlled amount of charge into the powder flow. The unit was simple and required only low pressure air to maintain a clean injection electrode.

2. Theoretical optimum lengths were derived for the current sensors.

B. Charge simulation

1. The change of linear charge density of the pulse with powder velocity was modelled [1d]. Different length pulses, produced as a result of the velocity changes, were displayed graphically.

2. Exponential models were used to simulate the charge decay and re-entrainment on the pipe wall.

3. There was a good correlation observed between the modelled current and the measured charge pulses.

4. Further simulation revealed a relationship between the pulse length and the inherent velocity information obtained from the sensor currents.

C. The data analysis

1. Three mass flowrate empirical models were deduced to suit the various flow conditions and the models were explained based on hypothesis.

2. Applying these empirical models to the powder flow, proved 90% of the 460 data points lay within the 10% error margins.

D. The instrumentation

1. Electronic instrumentation was built to measure and process the signals for the purposes of obtaining powder velocity information and the absolute mass flowrate.

2. Part of the instrumentation provided a constant injection current which enabled a constant charge to be produced.

3. Modification of the powder conveying system was made which enabled the powder to be continuously recirculated while recording the results. This was achieved by introducing a two way powder mechanical switch which diverts the flow powder to the balance for absolute mass flow rate measurement.

3.2.5 Further work

The following were the suggested areas for further work [1d]:

A. Injection electrode

There was a need to reduce the diameter of the injection electrode wire, to ensure the stability of the injection current and enable lower operating voltages to be used.

B. Negative charge injection

1. The pulsed HT supply could only supply a positive pulse. Introducing a negative pulse would have enabled comparison tests of the particle ionisation capabilities of the two methods. Negative charge injection is preferred in electrostatic precipitators [1d]. Availability of positive and negative charge injections would control ac charging.

C. Powder types

Various types and sizes of powder needed to be investigated since the work was confined to one powder type. It would be necessary to verify how the readings are affected with the change in powder type.

D. Environmental humidity.

Humidity effects on the calibration of the instrument needed to be investigated.

E. Continuous charge sensing

Further work on the charge sensing needed to be investigated because only continuous current measurements were possible. The net charge deposition on the sensors saturated the charge amplifiers. The dc drift on the charge amplifier was reduced by including a leakage resistor in parallel with the circuit's capacitor. To measure charge continuously, a technique of sample and hold had to be introduced. This technique would have enabled the capacitor to be discharged during the hold period of the cycle.

Charge measurements were not directly connected with the powder velocity and the investigation of the charge decay pattern may have enabled further understanding of the flow.

F. Improving the modelling

The modelling did not take into account the electrostatic repulsive force between the already deposited charge and subsequent charge deposition on the surface of the sensor. Furthermore, as the deposition layer increased on the pipe wall the electrostatic attractive force between the charged particle and the conducting sensor surface would be decreased.

G. Data separation

The data separation process needed to be further investigated in order to fully understand the separation of the data into the three 'regions'.

I. Feedback control

For mass flow rate measurement, work needed to be carried out, to continuously read and process the powder flow data on a microcomputer. A control system would require the generation of an error signal between the desired mass flow rate and measured flow rate. The error signal would be used to regulate the two feed air valves which control the primary and secondary air to the venturi unit on the powder hopper. The outcome of this work would have enabled the powder flow to be maintained to an operator fixed set value.

3.3 Phase 2 (B.C.O'Neill & E.Mills)

Work was carried out to produce a microprocessor based system for the continuous measurement of mass flowrate of pneumatically conveyed powder based on the pulse charge technique [2].

3.3.1 The need for a multiprocessor system:

Initial studies [5,6] were carried out using a motorola 68000 microprocessor based system. These indicated that the system was not fast enough for the powder flow application. The instrument required 20ms cycle time (charge injection period) to sample all four current waveforms from the sensor head. The 68000 would have been required to sample and store 2000 samples on each current waveform for post processing within the 20ms. The basic stages of post processing would have taken a further 80ms. Future implementation of control functions (decision making) would have taken considerably more time. Infact, the practical requirement of a control system for this application were: 1 second to service up to 8 units giving update information within this period. A single 68000 processor was not able to meet these requirements in real time and hence the idea of a multiprocessor based system was contemplated.

Conventional processors, requiring complex circuitry (FIFO

registers or dual port RAMS) were not suitable for this application and hence the decision was made to implement a multi-transputer based system with Direct Memory Access (DMA) with reduced interface circuit complexities and the possibility of real time processing.

3.3.2 The transputer instrumentation

The transputer instrumentation [4b] was designed to replace the analogue system previously developed by B.C.O'Neill and Willis. The experimental setup used is shown in figure 3.9. The sensor unit is same as in Phase 1 (figure 3.3)

The hardware configuration of the transputer nodes is shown in figure 3.10. The sensor signals were signal conditioned under the control of a T414 transputer. Data from the four sensors were then distributed to 4 other T414 transputers for real time processing. A T800 floating point transputer was used to provide the final results [4a,5,6].

The data acquisition unit, data processing array and the system controller were the main parts of the transputer instrumentation. The functions of the various parts will now be described.

3.3.2.1 The data acquisition unit

The first stage of the processing unit was a multiplexer

for the sensor input signals. The signal is then passed through a gain circuit and an 8-bit analog to digital flash converter. The data acquisition was controlled by a T414 32-bit transputer, operating with a 20 MHz internal clock and 2k internal RAM. External memory was not required since the internal memory was sufficient to run the conditioning program at maximum speed. The control transputer was used to sample the injection current, after searching for the point of injection. When injection was found, the controller transmitted the samples to the array processing transputers. The gain of each channel was controlled, for each sample, from software. The conversion system produced a 16 bit value with a resolution of 8 bits. The sampling rate was about 100 kHz per channel or about 400 kHz per injection period.

3.3.2.2 The data processing array

The four array processors ran programs on the 2k on board RAM, ensuring fast data processing. Using on-board RAM meant the transputers could run without additional wait states being added for memory access, hence, maximum throughput was obtained. The array transputers run identical programs and, in addition, one was used for message passing to and from the data acquisition unit. Each processor accepted data from the acquisition unit. It then formatted it into a 16-bit fixed point

representation of the analog signal, which was then used for peak detection and signal integration. Each processor detects the current peak and integrates the current to obtain the charge. The charge information was then analysed on the controller unit.

3.3.2.3 The controller unit

The controller was a T800 floating point based processor with 1 Mbyte external memory. The basic functions of the controller were:

1. It controlled the processing instrument.
2. It used a digital to analogue converter to set the injection unit's voltage. A TTL control register contained the data for the HT supply which was used for pulse generation.
3. It informed the data acquisition unit of the number of samples to be used during a cycle of HT injection.
4. The processed data from the processing unit were passed onto the controller for mass flowrate analyses.

3.3.3 Experimental results

The transputer system [4c,5,6] provided two modes of operation:

1. Waveform capture and storage.
2. Real time parallel signal processing.

In the waveform capture mode, it was possible to compare transputer signals and to vary the processing algorithms over a wide variety of powder flow conditions.

For each setting of the powder flowrate an absolute value of mass flowrate was obtained from the electronic balance.

Data were collected from the four sensors for 2000 attempted charge injections. Each injection was processed in real time for the following purposes:

1. to detect if injection had occurred,
2. to obtain the integral of the transducer currents,
3 to obtain the positive and negative peaks of the transducer currents and the respective times.

3.3.3.1 Analyses of the mass flowrate.

Data was collected from 198 flow settings in the range 0.2- 6g/s with a velocity range of 3.5 to 14.0 m/s. Some of the readings were automatically rejected, especially at low velocities where strong fluctuations in flow occur.

Several regions [4d] of mass flow predictions were obtained as shown in figures 3.11a, 3.11b, 3.11c, 3.11d, 3.11e, and 3.11f, 3.11g, 3.11h and 3.11i. Linear

prediction plots of mass flow rates readings were obtained against:

1. sensor 2 current waveform (I2), for low mass flow rates Region 1 (figure 3.11a).
2. I1/velocity (where I1 is sensor current 1), for low mass flow rates new region (figure 3.11b).
3. sensor current ratio (I2/I1), for high mass flow rates Region 2a (figure 3.11c).
4. sensor current (I2), for medium mass flow rates new Region 2a (figure 3.11d).
5. sensor current ratio (I2/I1), for high mass flow rates Region 2b (figure 3.11e).
6. I2/velocity for high mass flow rates new region 2b (figure 3.11f).
7. sensor current ratio (I2/I1), for high mass flow rates Region 2c (figure 3.11g).
8. sensor current ratio (I2/I1)/velocity, for high mass flow rates Region 2c (figure 3.11h).
9. I1/velocity for low mass flow rates white region 1 (figure 3.11i).

The data points on the various plots are marked with letters, which indicate the relative primary air settings for the powder flow regulation [4d].

The results, obtained from complex criteria, show that it is possible to obtain the mass flow rate from the transducer readings to an accuracy of 15% of each scale.

3.3.4 Experimental conclusions

The following were the conclusions from the experimental work [4c]:

1. Based on the pulse charge technique, the transputer system allowed the gathering of a large database which was processed at high speed by support software.
2. It was feasible to use a transputer system as a closed loop control system.
4. A new empirical model for the measurement of mass flow rates in the range 0.2 to 6 g/s with a wide range of velocities from 3.5 to 14 m/s was produced. Methods involved an 'on-line' separation of the readings into three regions of flow.
5. The use of transputers had proved more advantageous than conventional processors such as the Motorola 68000. In particular, the system was able to process the sensor signals in real time.
6. A significant change in the sensor readings was observed when changing to a new white epoxy powder. This was attributed to the different particle sizes of the powder types.

3.3.5 Further work

The following were the areas identified for further work [4c]:

A. Different powder types

Effects of the change in the particle sizes should be investigated by using calibrated particle sizes and analyzing the waveforms. Analysis of the white powder will enable a new system to be produced which is less dependent on the wave shapes.

B. Modification of the data acquisition unit

It was recommended that the data acquisition unit be modified in the following ways:

1. Three separate modules should be created for the multiplexers, the gain circuit, and the flash converter (with the transputer control logic). Compact screened modules could reduce inter-stage interferences.

2. The gain circuit should be screened and the analogue data paths routed via screened coaxial cable; to reduce the effects of external interference.

3. The transputer board which interfaces with the A/D could be replaced with an INMOS 16-bit transputer (T212 or T222). This would increase the speed of the operation and reduce the cost. Increases in speed would be as a result of the non-multiplexed address and data busses of

the T2 transputer products, which reduce the external memory access time by one clock cycle (a reduction of 50ns for a 20MHZ transputer)

5. In the system design, the gain computation was carried out using an EPLD for intra-sample adaption. This had the effect of increasing the bandwidth as it doubled the data transmitted down the links. The EPLD could be replaced by a simple latch without loss of functionality. Replacing the EPLD would further reduce the cost of the board.

C. Modification of the processing array

1. The Processing array transputers could be changed for T2 type transputers, further reducing the cost of the system. The difference in the performance needs to be evaluated because 32 bits transputers are more suitable for a 32 bits arithmetic and changing to a 16 bits might create a penalty cost in accuracy.

2. The T414 transputer (with 2k on board RAM) could be replaced by a T425 transputer (with 4k on board RAM). The replacement would enable more software to be produced on the array without additional on board circuitry.

3. Further analyses of the mass flowrate equations could provide better models or algorithms, reducing the number of computations required to produce the same results.

D. Modification of the Controller

The controller's large memory could be re-designed for control application purposes. All of the control logic might be incorporated into the circuit board of the controller making it more dedicated to the flow control task.

E. Modification of the high speed adaptive converter system

The converter operates by calculating the gain of the next sample based on the gain of the current sample and this tracks the signal with a gain. Logic is used to decide the gain value which was stored in high speed RAM.

In the proposed design the logic of the original converter could be replaced by high speed RAM, preprogrammed with a lookup table containing all the information to generate the gains and the adaptations. The new high speed RAM would also enable various adaption techniques to be encoded.

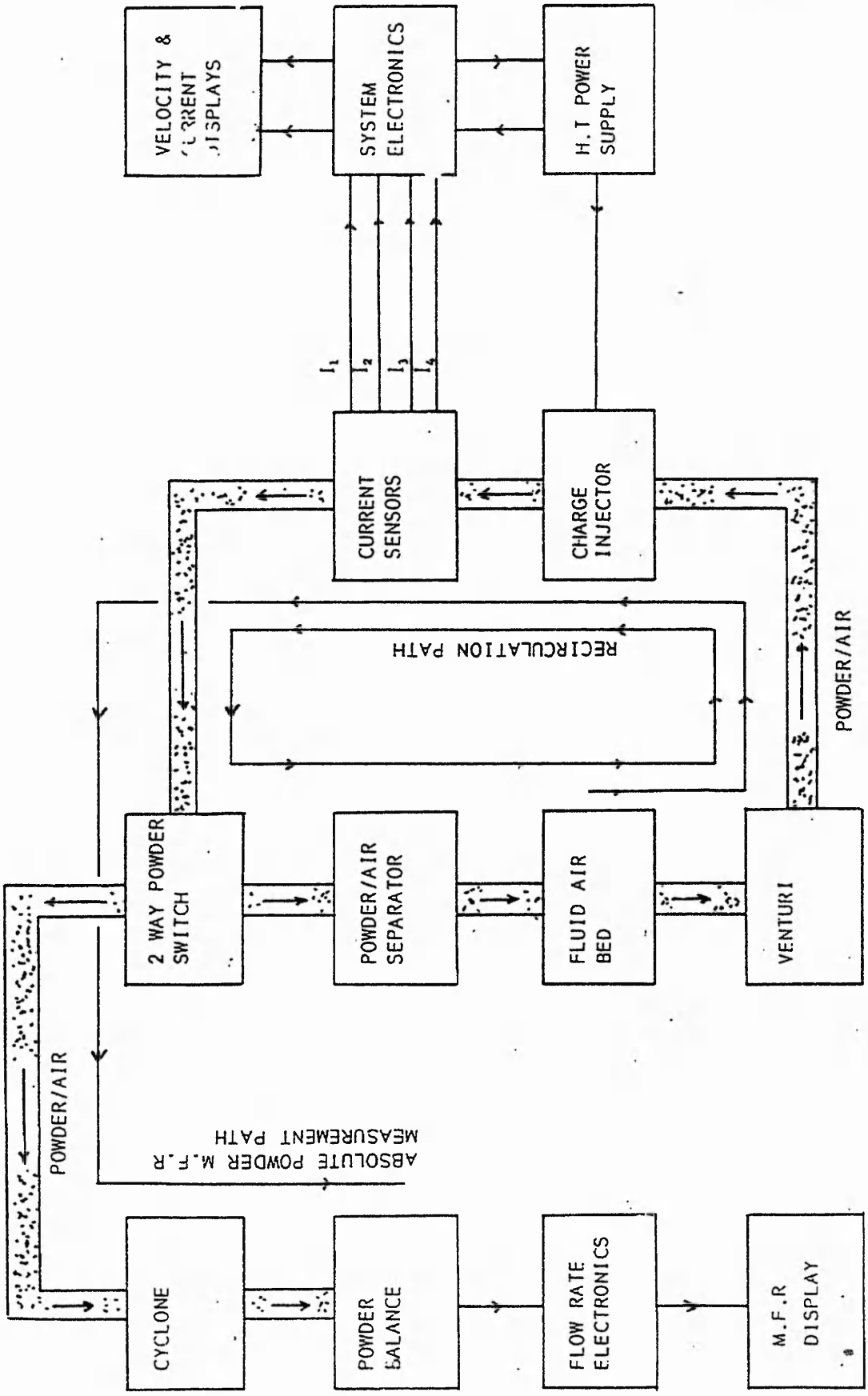


Figure 3.1 Experimental rig block diagram

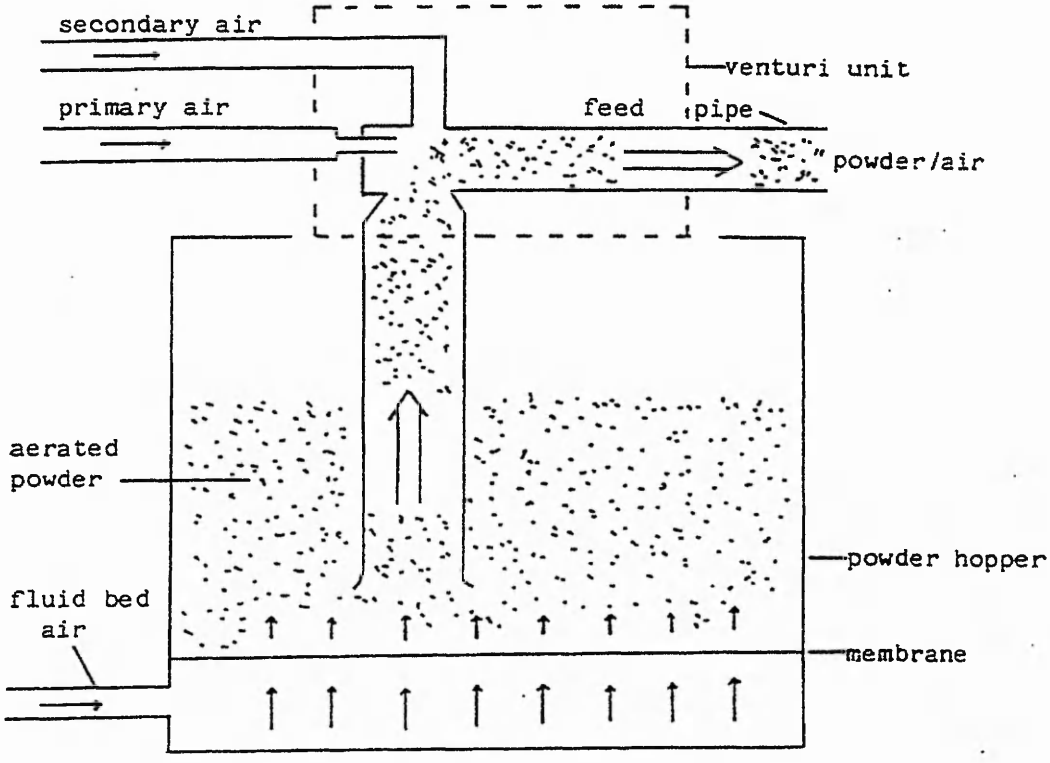


Figure 3.2 Powder Hopper & Venturi

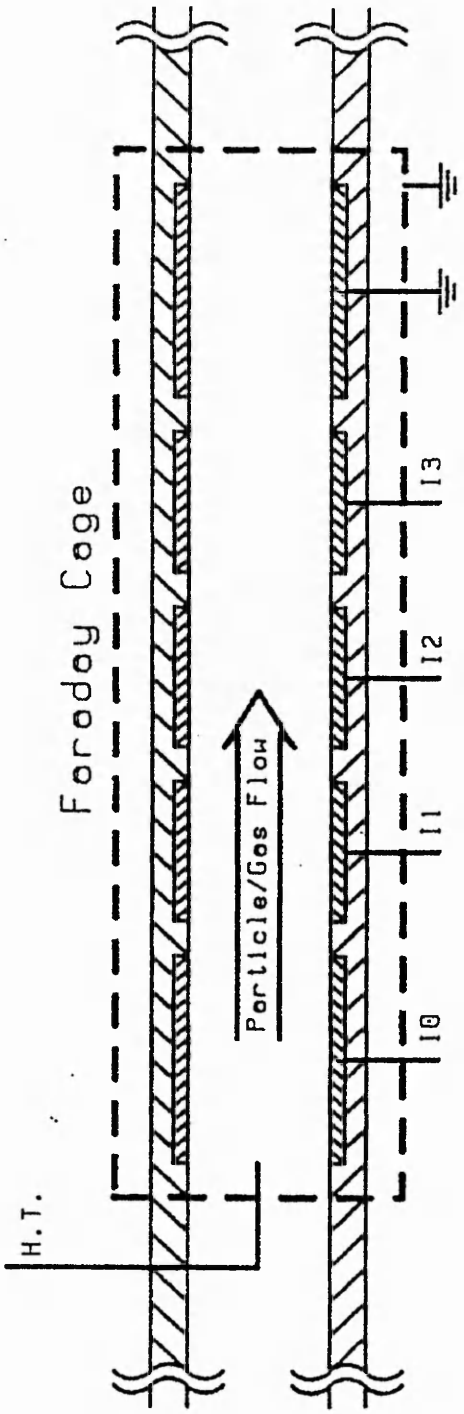


Figure 3.3 Faraday 'caged' sensor unit

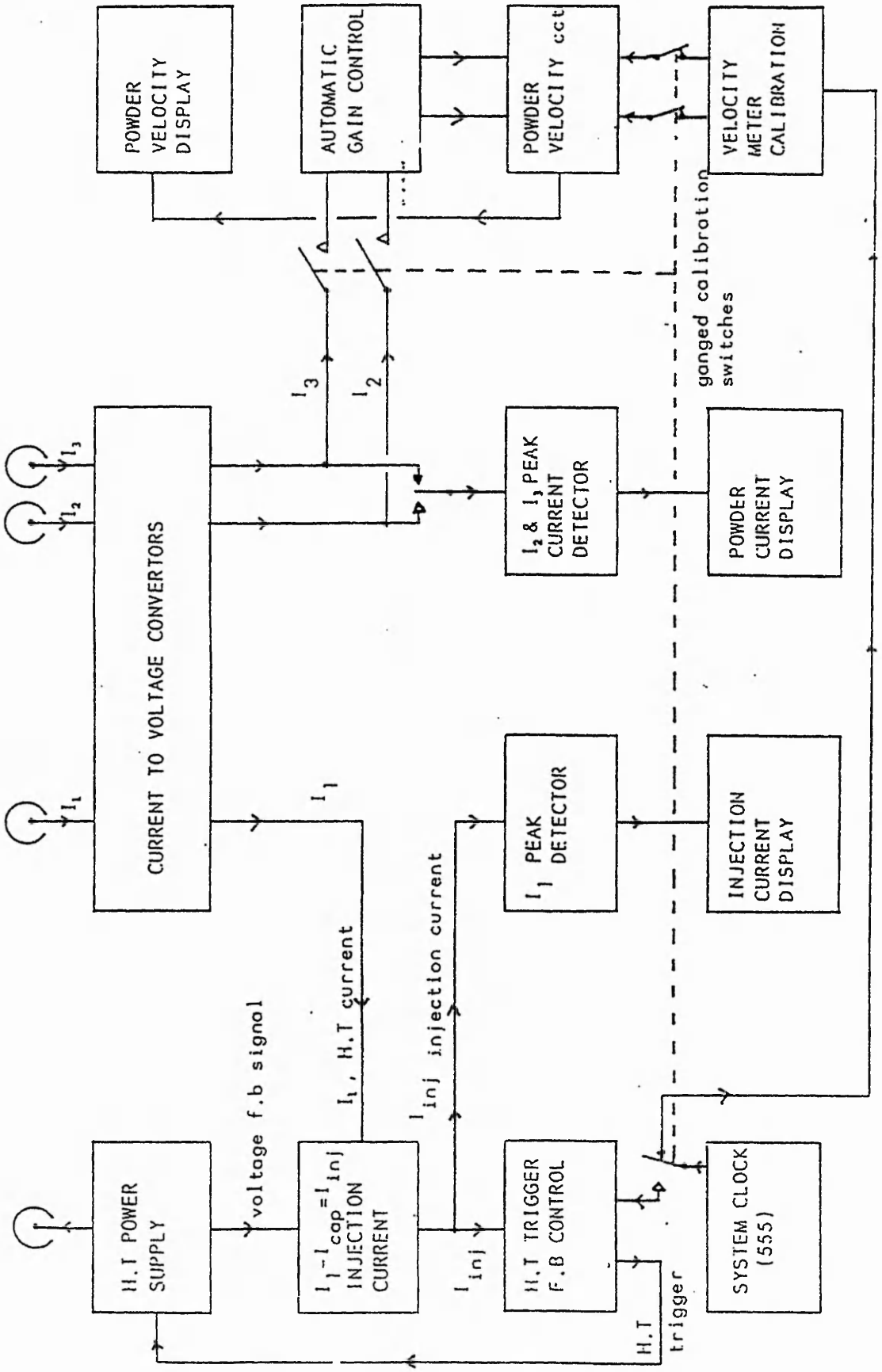


Figure 3.4 System electronics block diagram

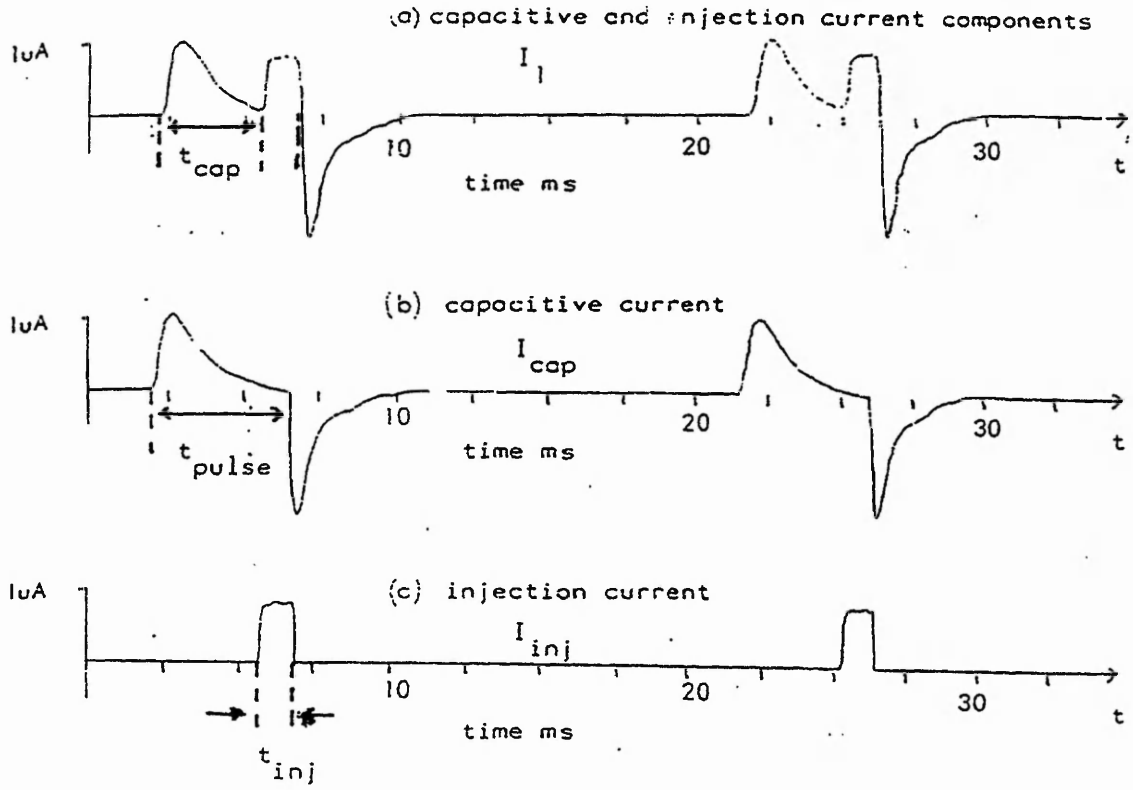


Figure 3.5 Injection electrode current waveform (sensor1)

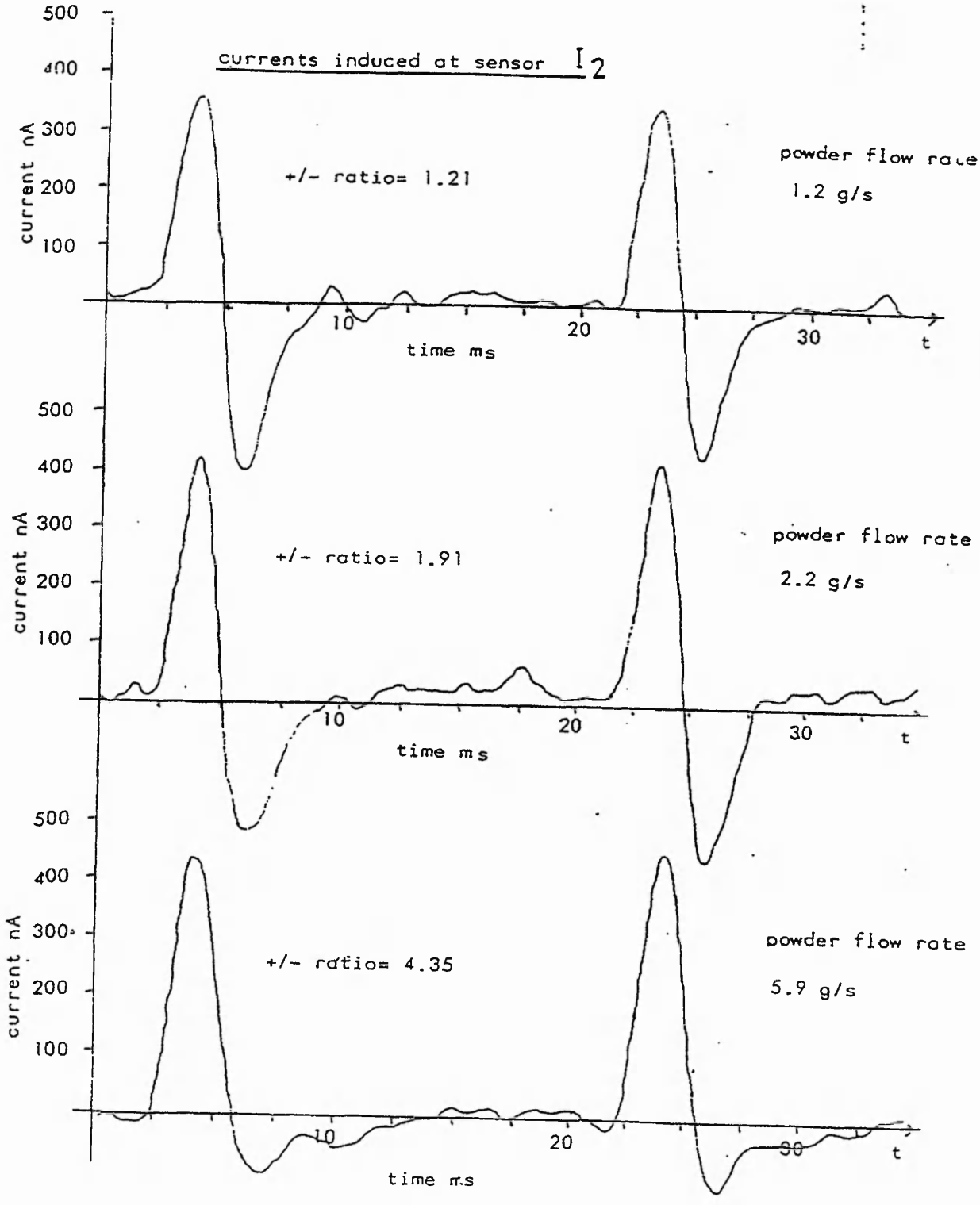


Figure 3.6 Sensor 2 current waveform

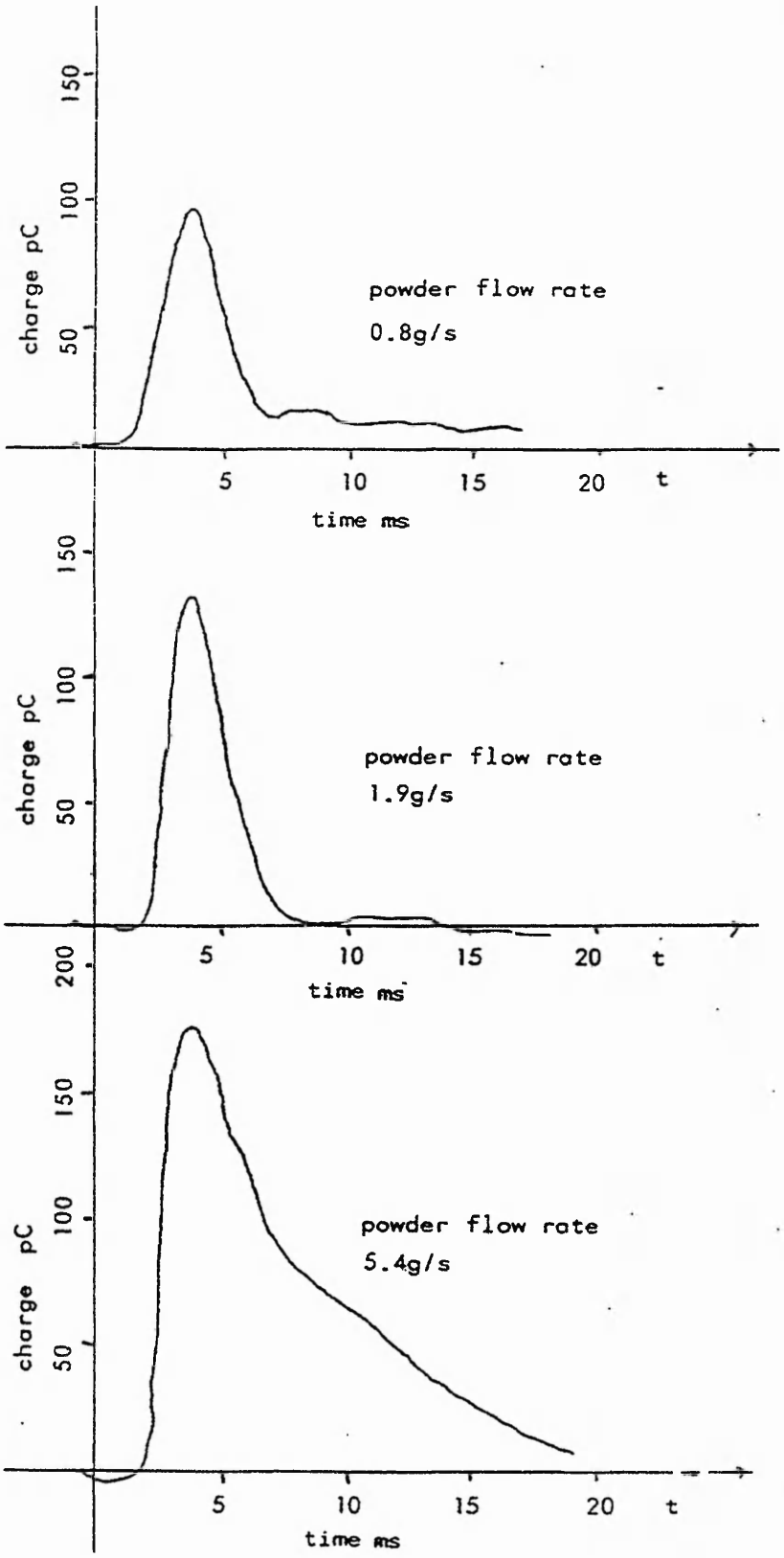


Figure 3.7 Charge waveforms

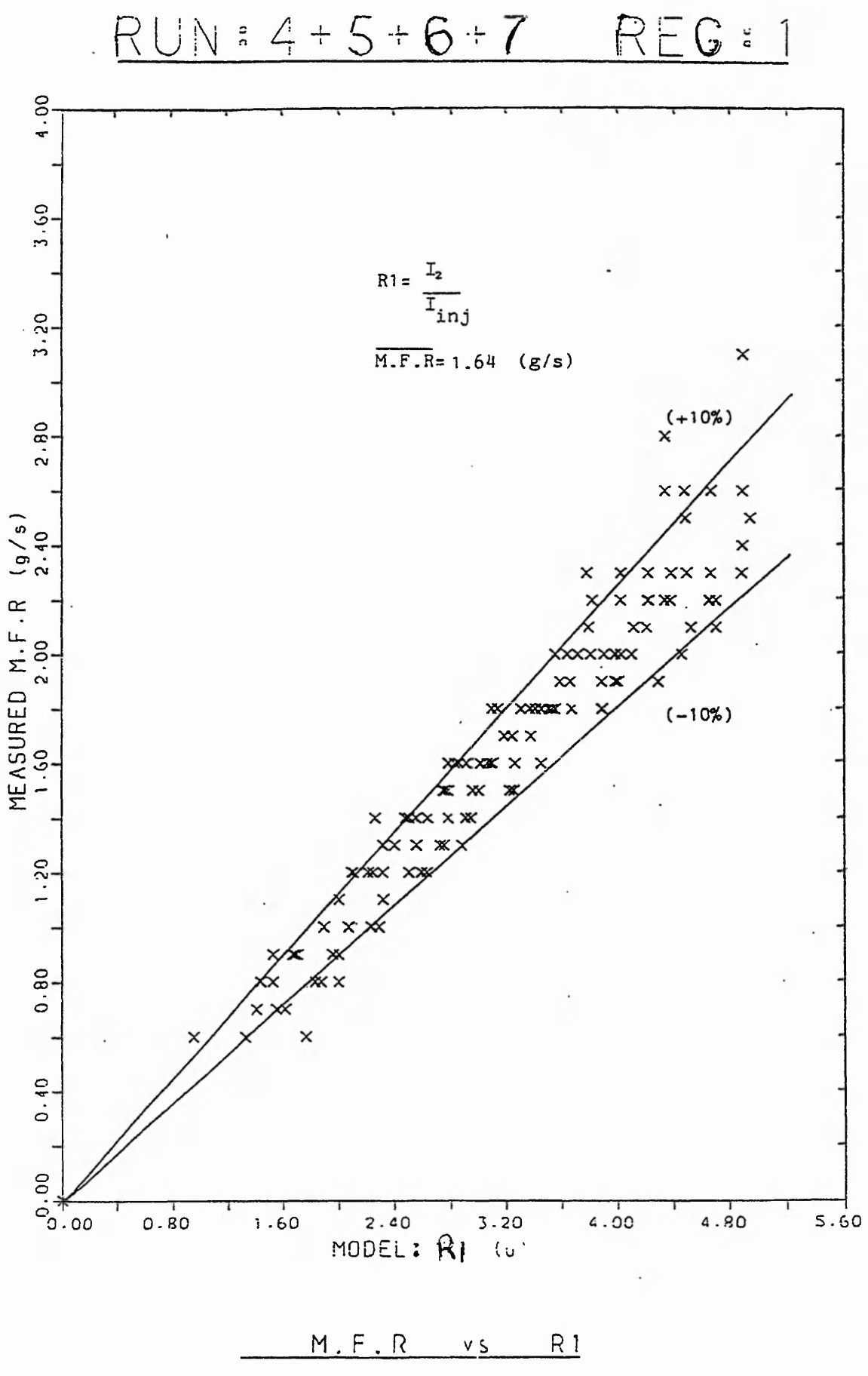


Figure 3.8a Low mass flow rate model

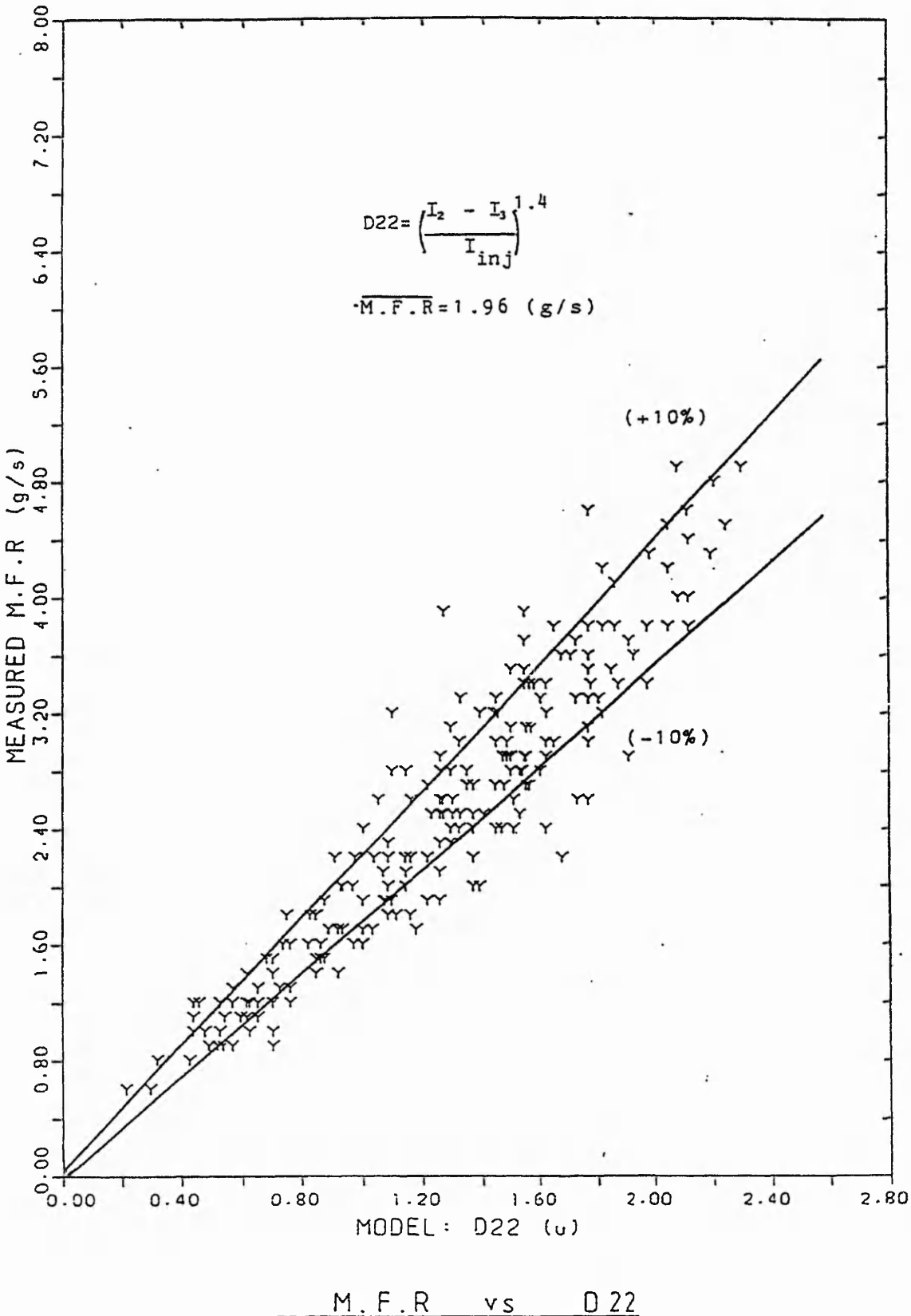


Figure 3.8b Medium mass flow rate model

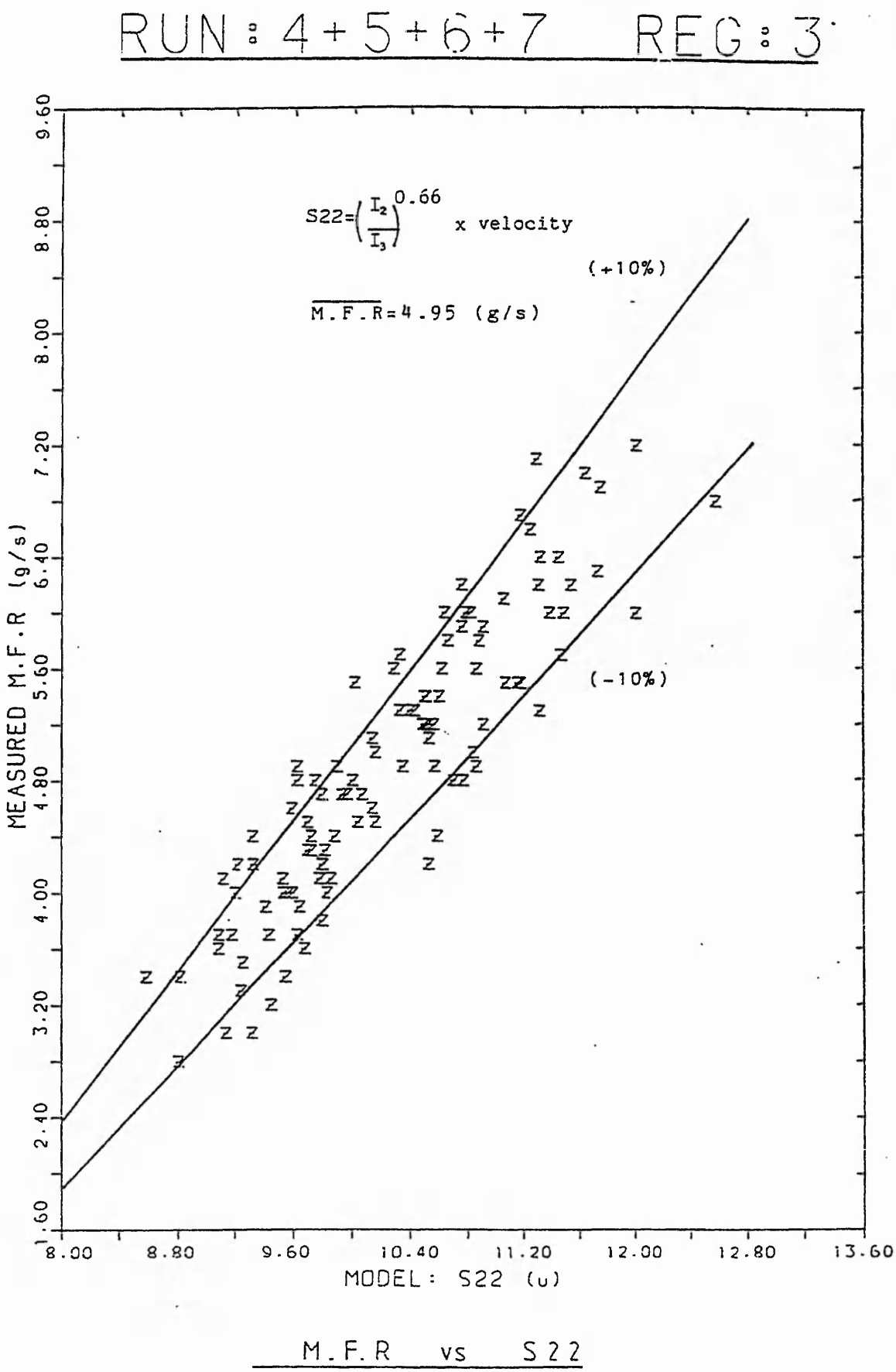


Figure 3.8c High mass flow rate model

PHASE 2 DIAGRAMS (B.C.O'Neill and E.Mills)

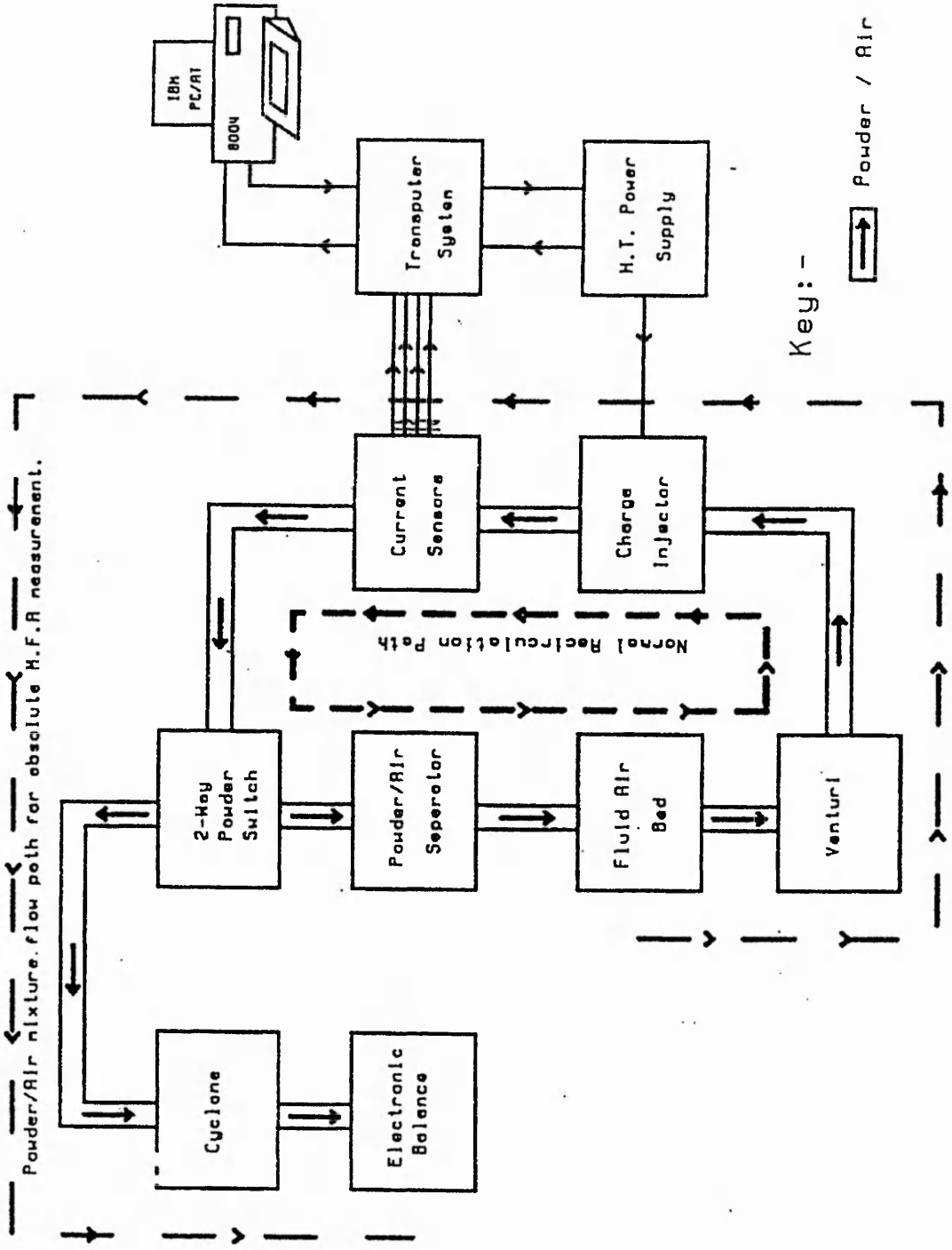


Figure 3.9 Experimental setup block diagram

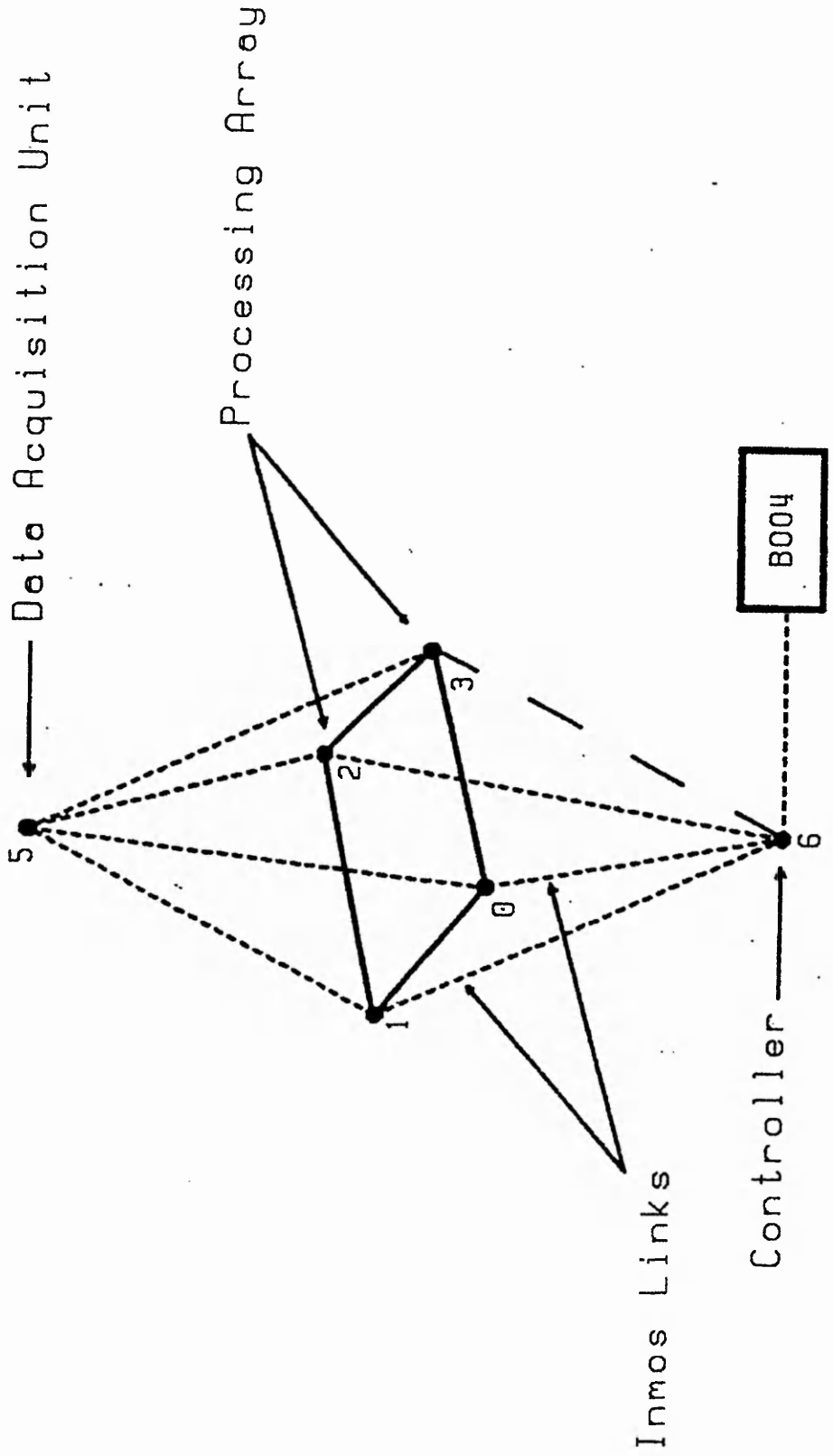
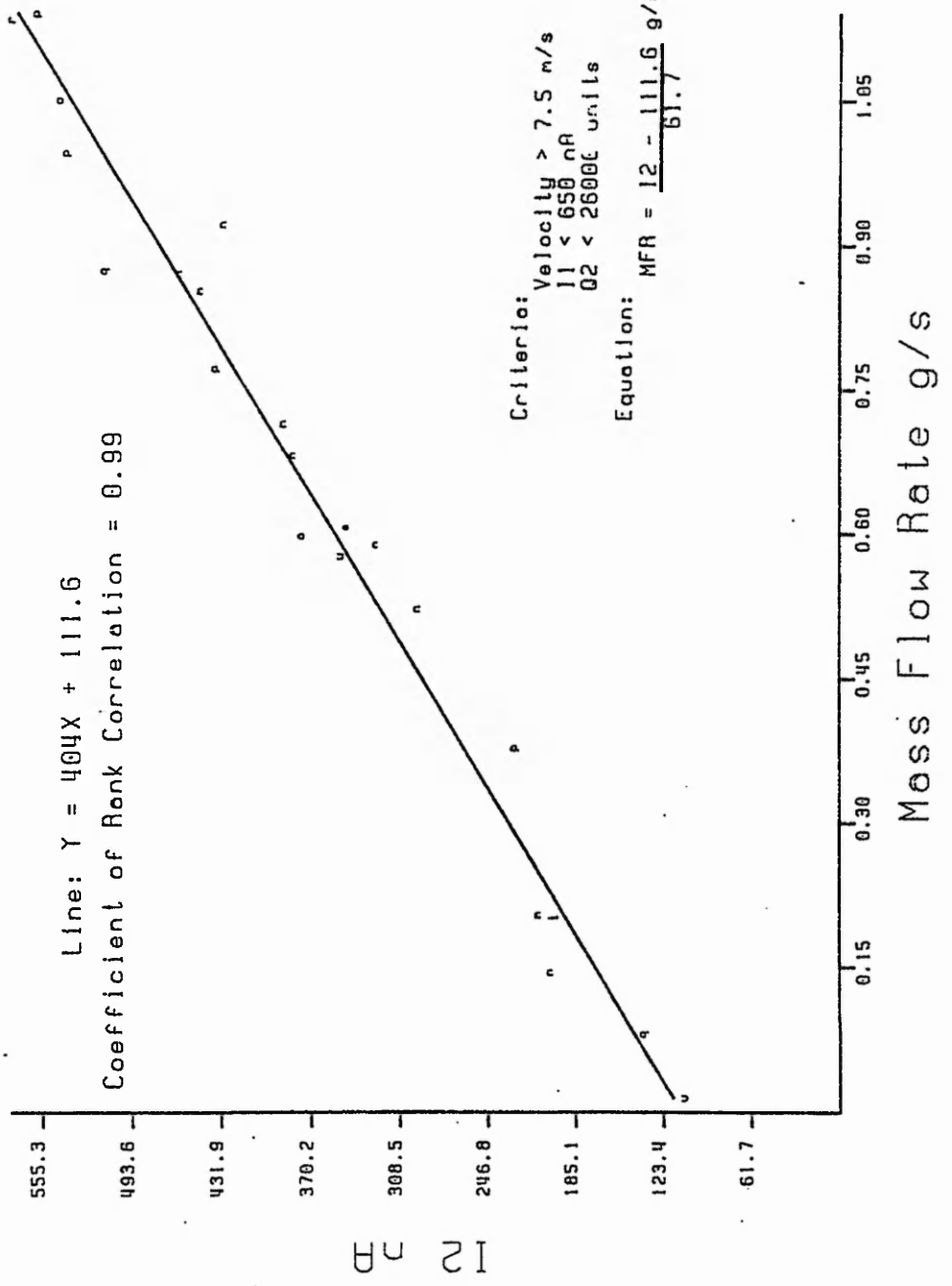
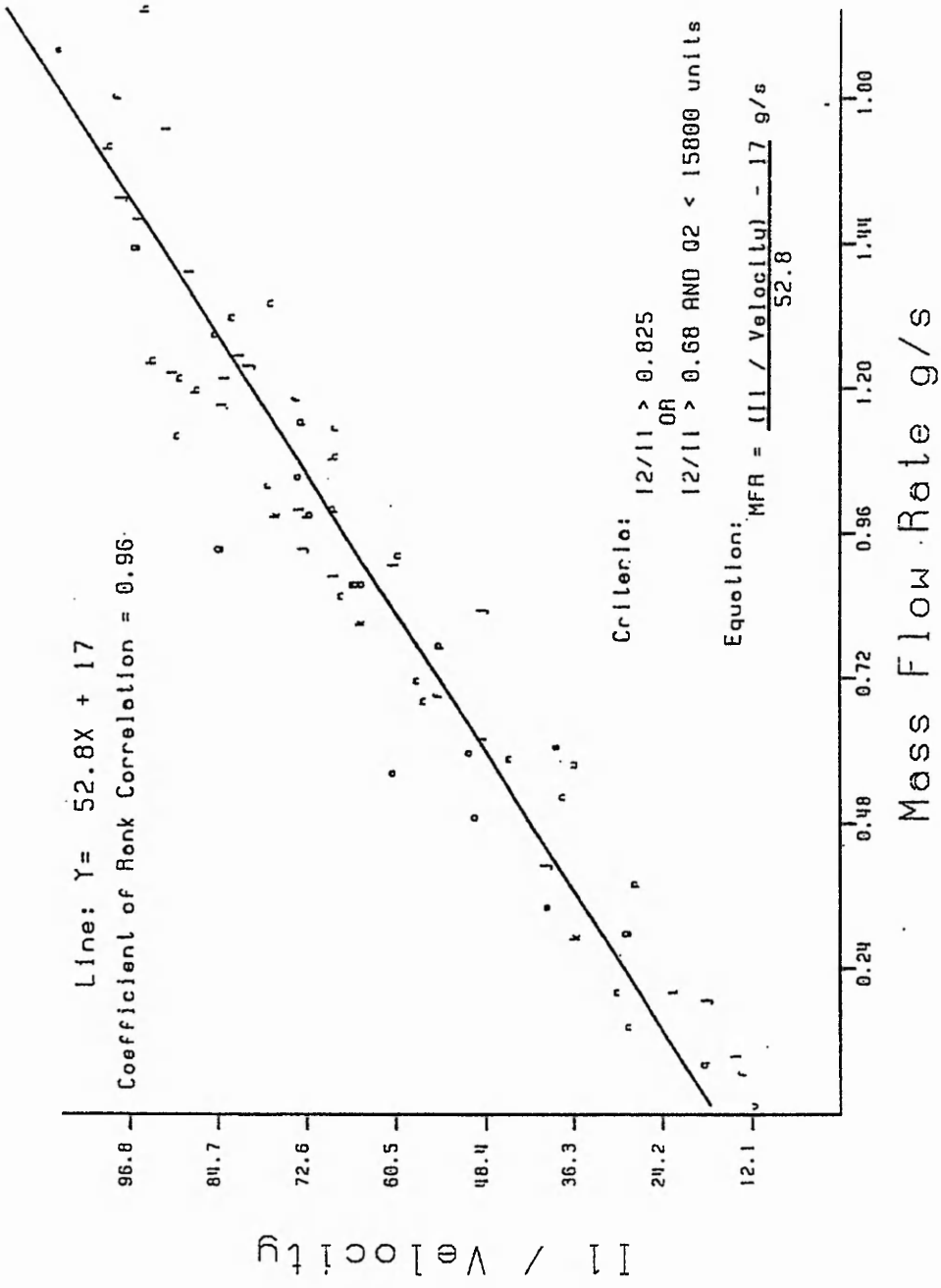


Figure 3.10 Transputer nodes configuration



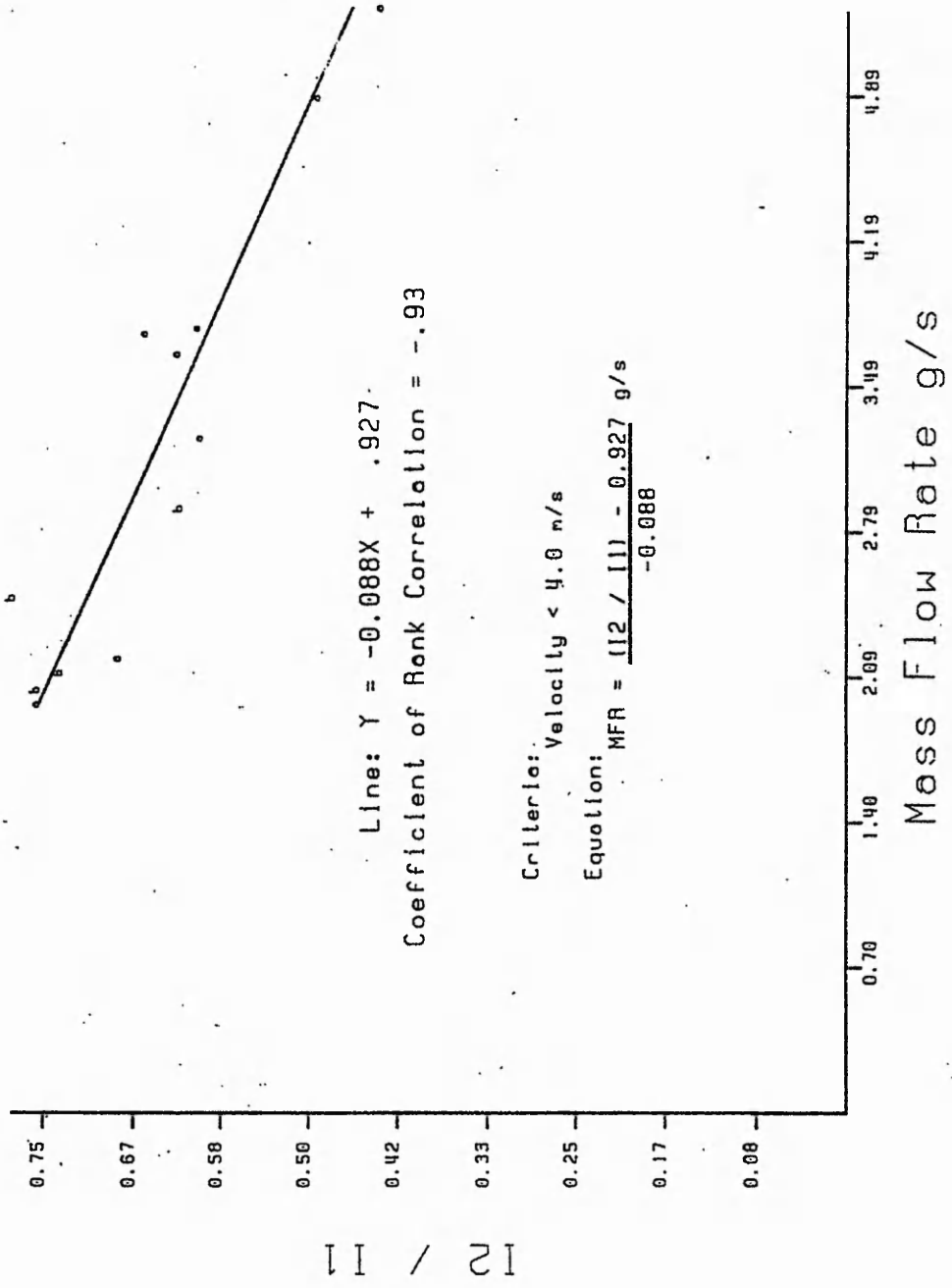
Region 1.

Figure 3.11a Low mass flowrate plot (Region 1)



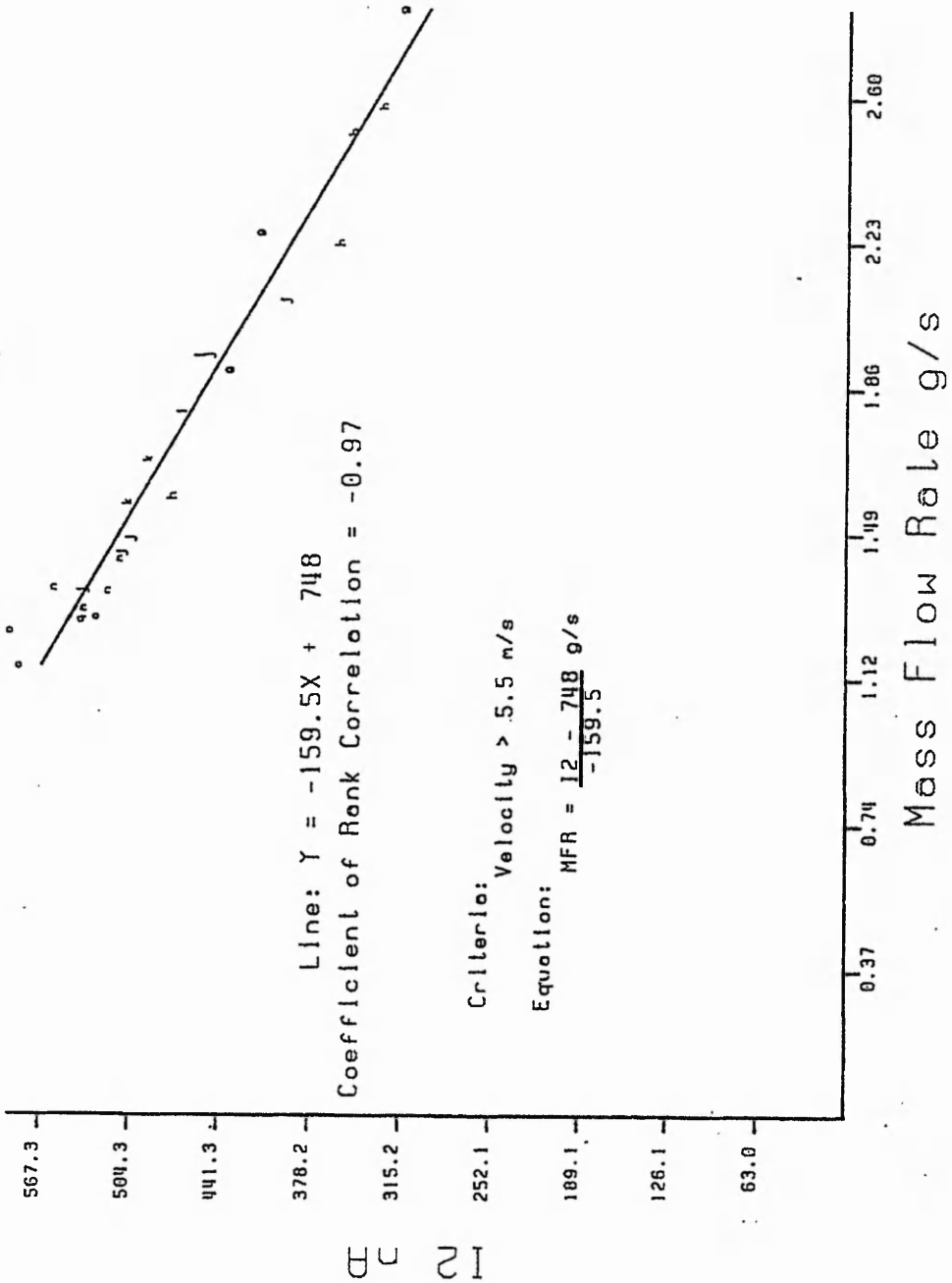
New Region 1.

Figure 3.11b Low mass flowrate plot (New Region 1)



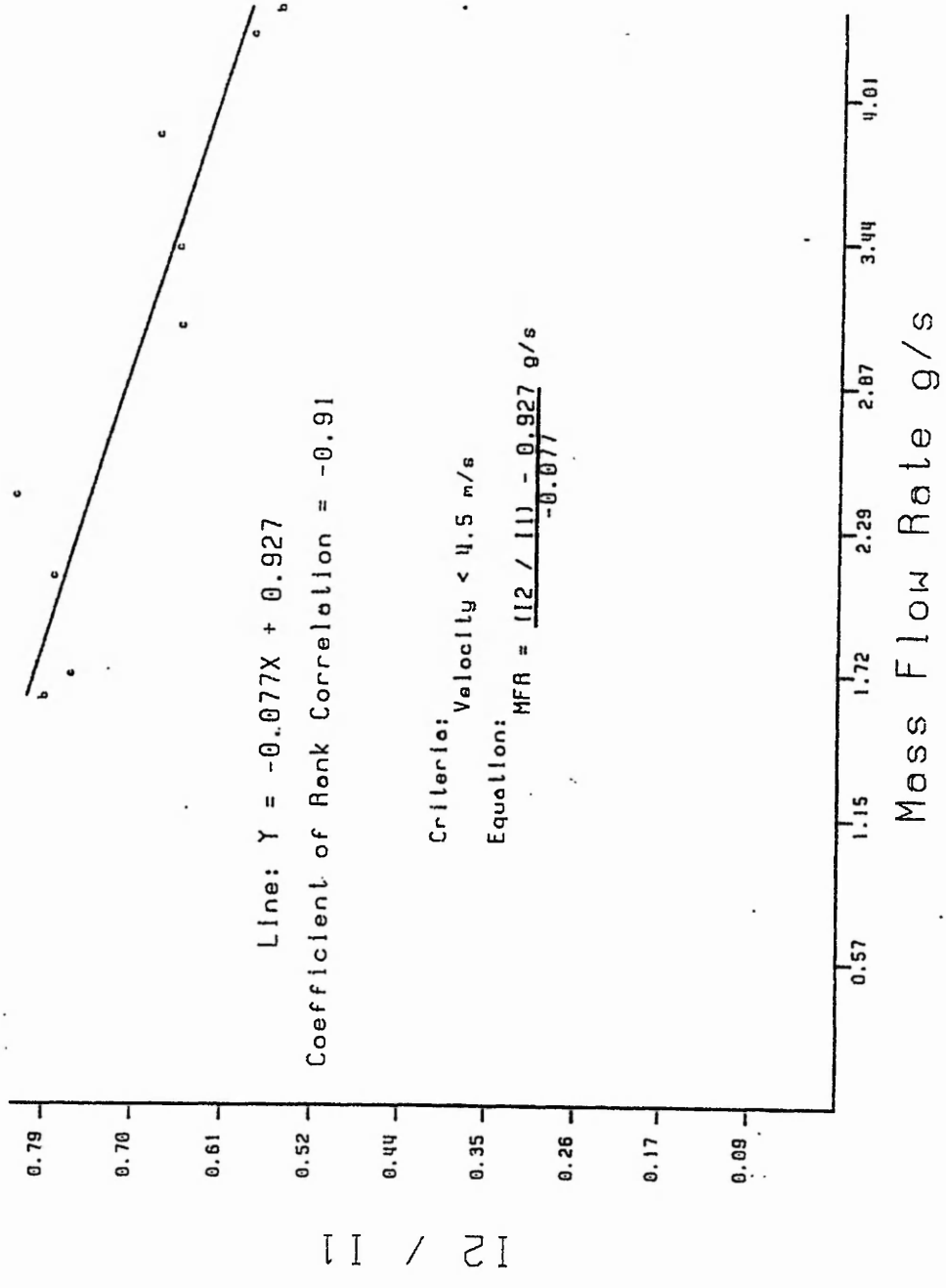
Regicn 2a.

Figure 3.11c High mass flowrate plot (New Region 2a)



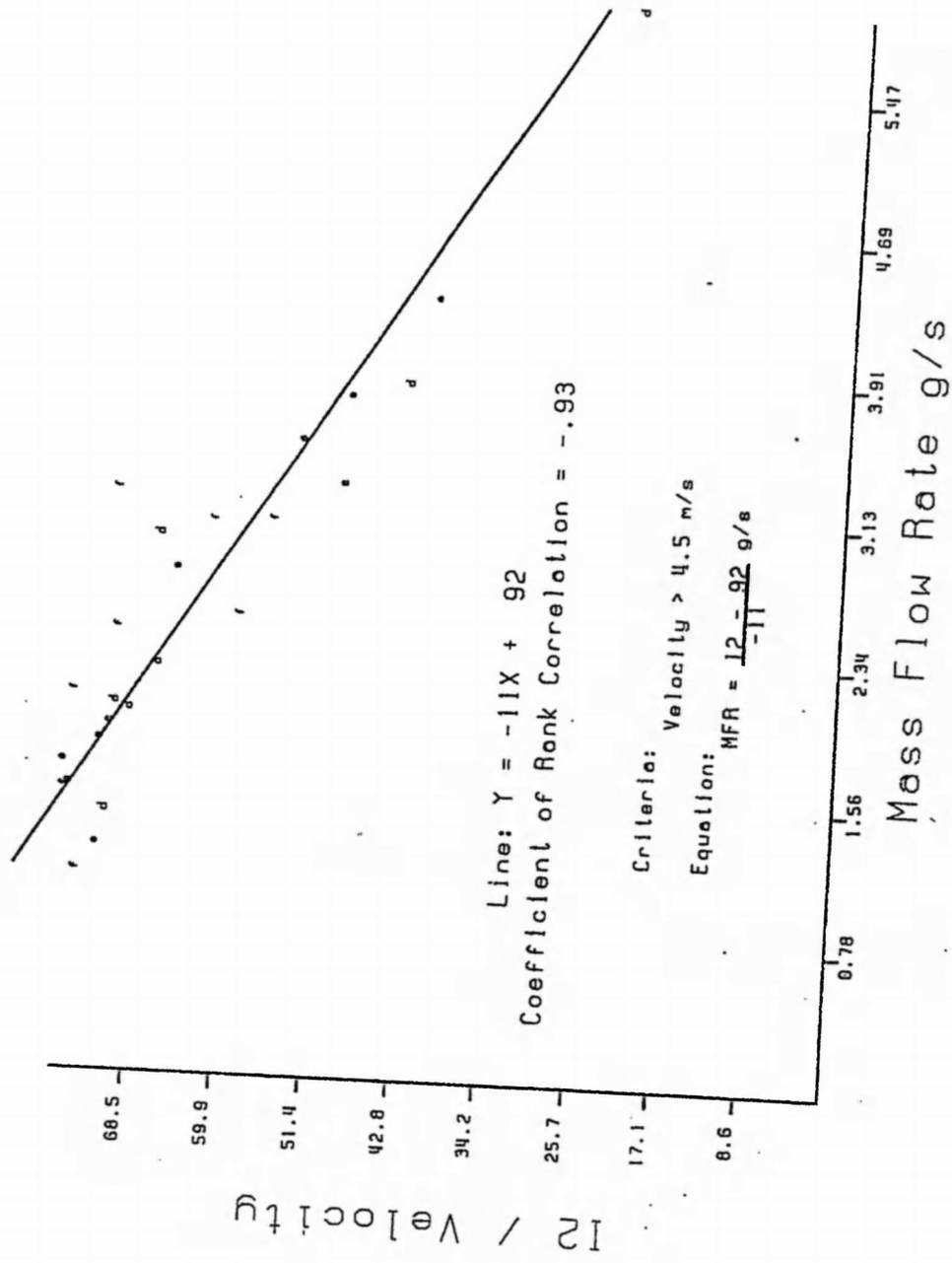
New Region 2a.

Figure 3.11d Medium mass flowrate plot (New Region 2a)



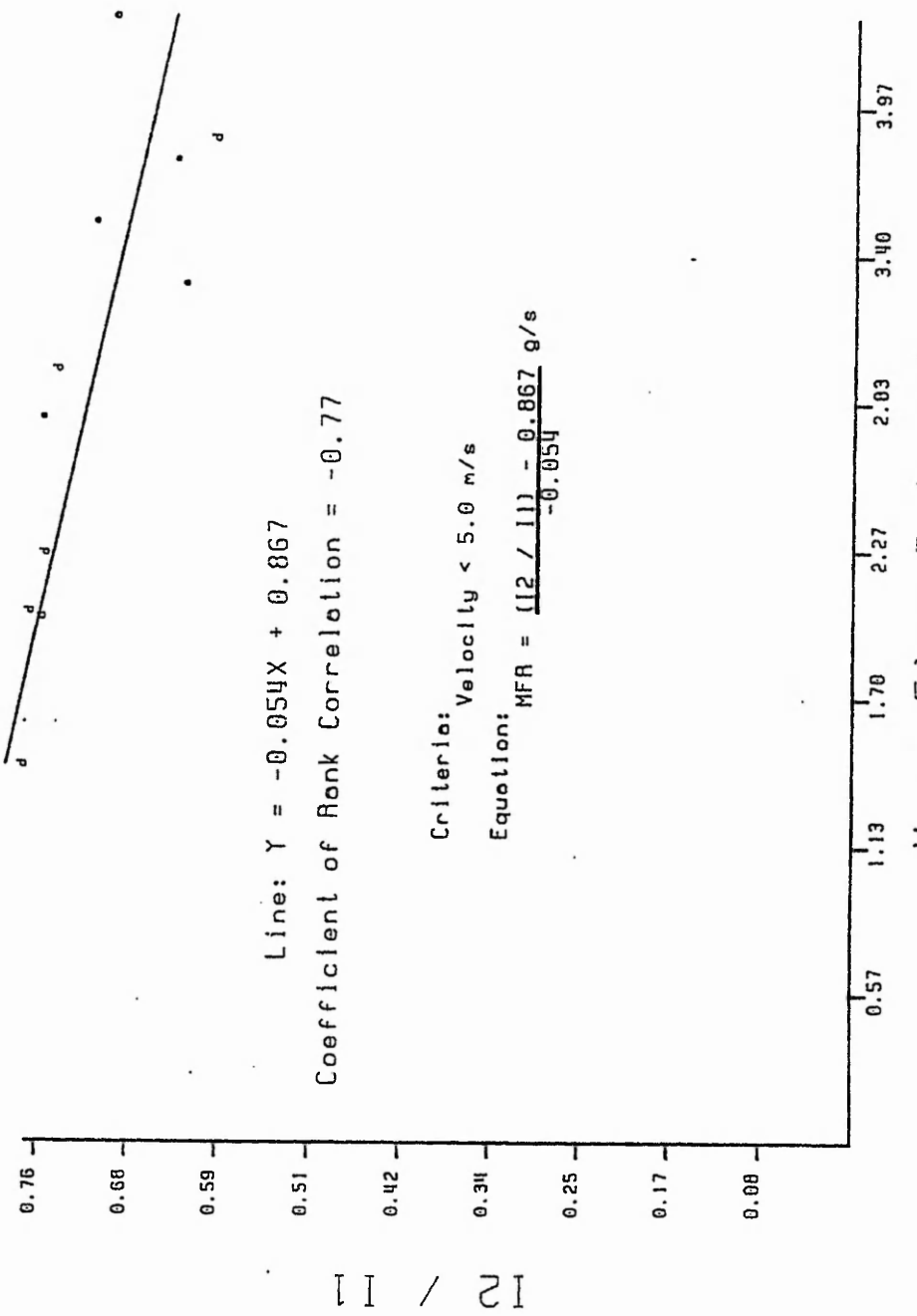
Region 2b.

Figure 3.11e High mass flowrate plot (Region 2b)



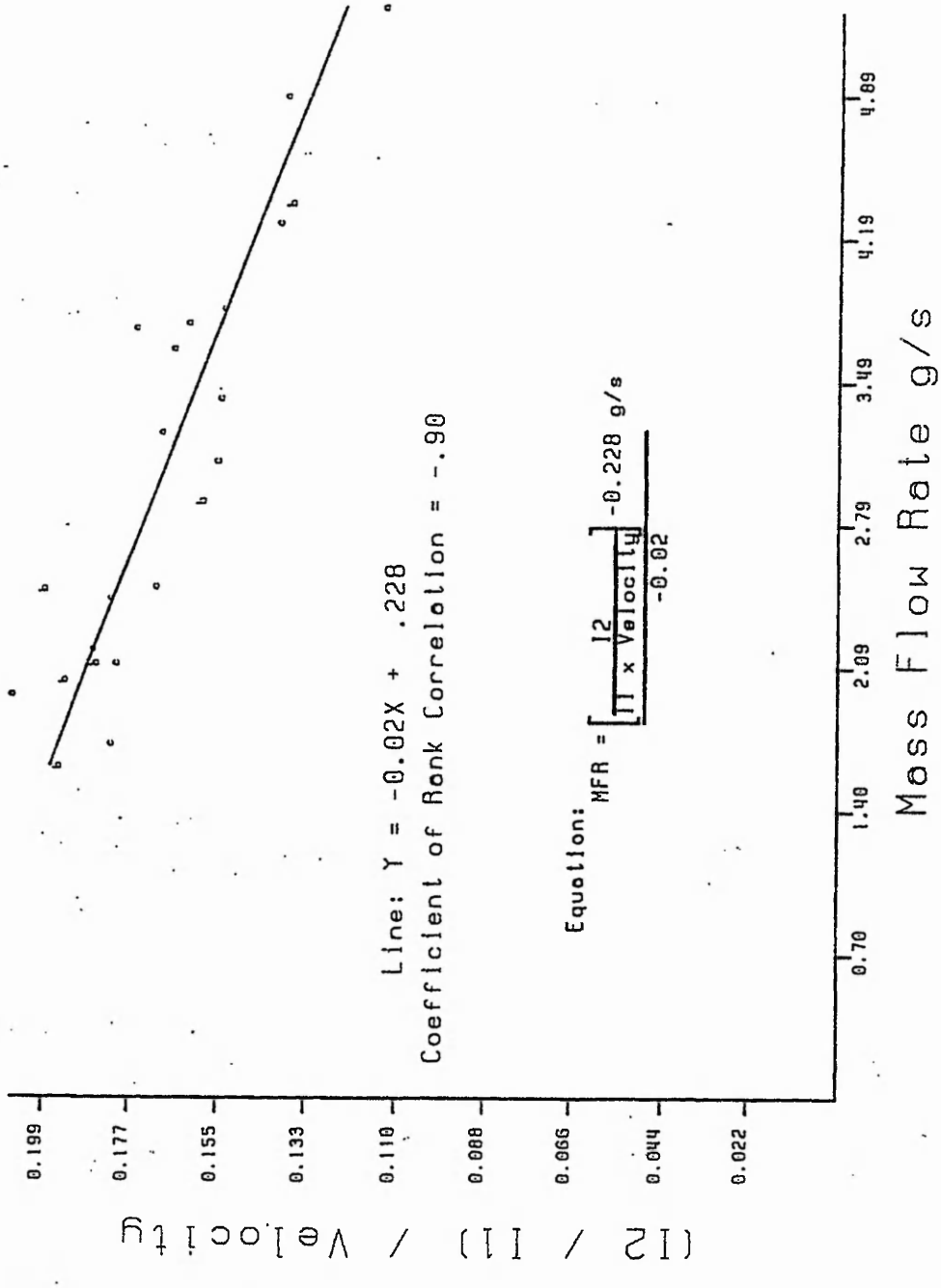
New Region 2b.

Figure 3.11f High mass flowrate plot (New Region 2b)



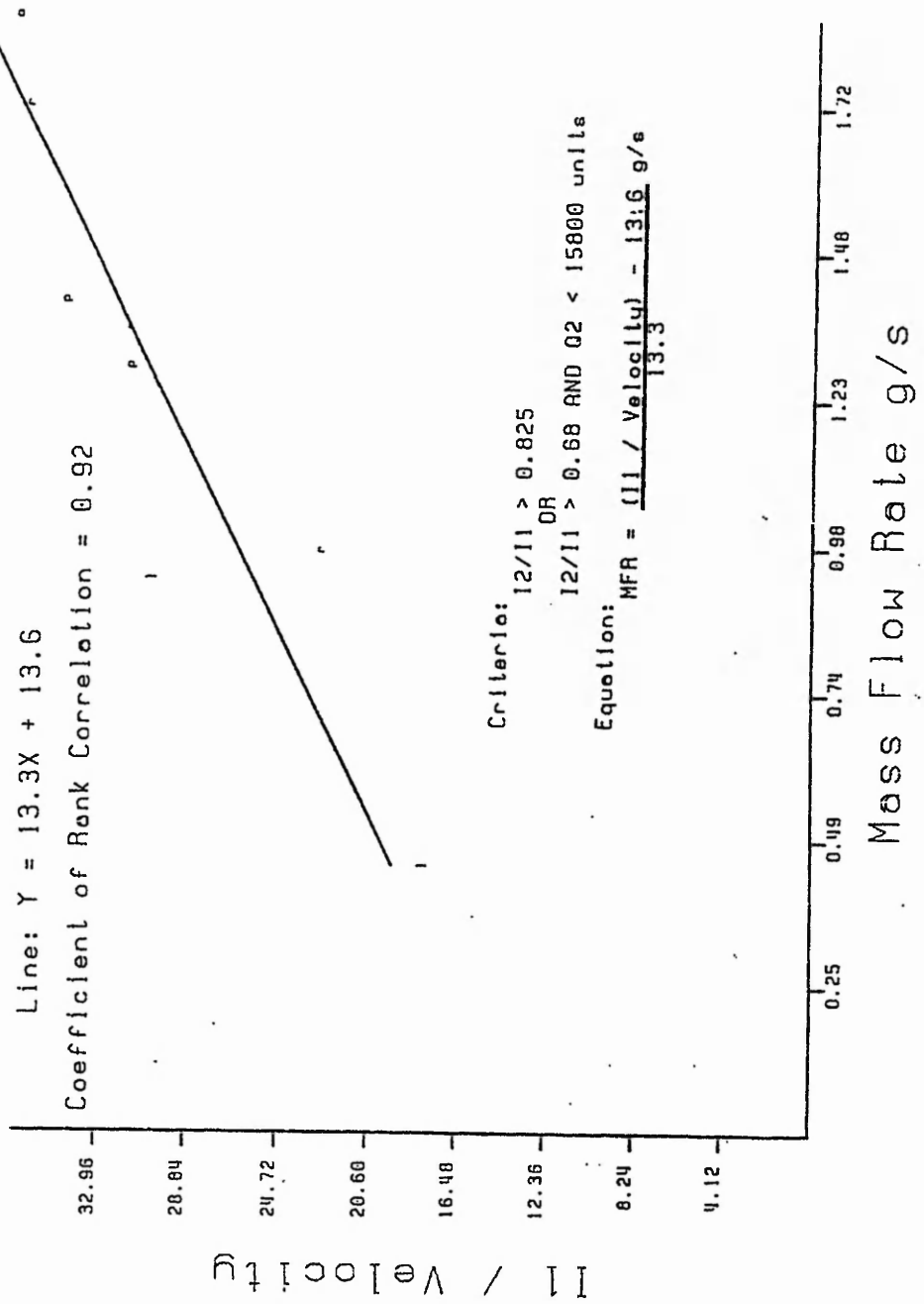
Region 2c.

Figure 3.11g High mass flowrate plot (Region 2c)



New Region 2c

Figure 3.11h High mass flowrate plot (New Region 2c)



White Region 1.

Figure 3.11i Low mass flowrate plot (white Region 1)

3.4 References

1a, 1b, 1c, 1d. Willis, C.A: "Continuous mass flow rate and velocity measurements of pneumatically conveyed powder" PhD Thesis; Trent Polytechnic, Nottingham, 1984, P9, P63-97, P111-121, P139-143

2. O'Neill, B.C. and Willis, C.A:
"Corona charging of pneumatically conveyed powders"
Journal of Electrostatics, June 1985, P99-107

3. O'Neill, B.C. and Willis, C.A:
"An electrostatic method for the measurements of powder flow rates in pipes"
Inst. Phys. Conf. Ser. No 85, Electrostatics 1987, P303-306

4a, 4b, 4c, 4d. Mills, E: "Transputer Instrumentation for particle flow measurements" PhD Thesis; Trent Polytechnic 1989, P31-41, P21-28, P183-188, P159-177

5. Mills, E and O'Neill, B.C:
"Transputer Instrumentation applied to Electrostatic powder flow measurements"
Proc. of the 8th Technical Meeting of the Occam User Group, March 1988, P55-62

6. Mills, E and O'Neill, B.C:

"Particle flow Instrumentation"

Proc. of the Second International Conference on Applications of transputers, July 1990, P56-62

7. Baker, R.J. and Nelder, J.A. "The GLIM System, generalised linear interactive modelling manual" Numerical Algorithm's Group, Oxford, 1978

4.0 HARDWARE DESIGN

4.1 Introduction

The current work is a development of previous research [1,2]. The objective was to enhance the existing powder paint spraying system by making measurements and providing instrumentation to improve the quality control of the process [8]. As a result, there was a need to review the system hardware.

4.2 The system design principles

The system development cycle [4a], adopted for the powder flow instrumentation, is illustrated in figure 4.1. The design is top down, progressing from general to the specific. The specifications for the powder flow system was developed in the review (chapter 3). The preliminary system design [4b] deals with the major functions to be performed by the various system modules. Further down the cycle, the system is partitioned into hardware and software development. The detailed analyses of each module is performed and then subjected, where necessary, to a design review before construction. The individual hardware and software modules are then integrated into the complete system.

4.3. System function

The primary function of the powder flow instrumentation is to continuously measure mass flow rates in the region 0.2-6 gs^{-1} with a velocity range of 3.5-14.0 ms^{-1}. Various parts of the instrumentation were designed to meet performance and cost reduction objectives.

4.4 Current system detailed block diagram

4.4.1 Introduction

The block diagram of the current system, given in figure 4.2a, contains several updates on the system used by previous Researchers. The updated sections are :

1. Pneumatic powder flow subsystem
2. Signal conditioner unit
3. A/D converter unit
4. Digital interface unit
5. Transputer unit.

Commercial pneumatic processes which could be controlled by the instrumentation are illustrated in figure 4.2b. A brief explanation of these processes are as follows:

A. Powder coating

A better quality powder coating can be produced by well controlled, sprayed powder particles. In electrostatic powder coating, a metal object (to be coated) is initially earthed. The charged powder particles from the spray gun are attracted to the object. In an industrial application, the thickness of the coating is regulated by controlled mass flow rates from the spray gun.

B. Industrial particle pollution control & solid mass metering

An instrument which senses and controls industrial pollution would be valuable to many industries. In the power generating industry, the exhaust gases from power station chimneys need to be analysed for pollution. This instrument could be readapted to sense the quantity of the pollution and filter the pollutant particles from the exhaust gases. The exhaust gases (from burning coal machinery in power generating industry) could be used to drive a power generating turbine. The quantity of solid particles from the gases need to be metered and controlled to prevent possible blockage of the turbine and to minimise environmental pollution from the escaped gases.

C. Commercial powdered drugs manufacture

The instrument could serve a very useful purpose in pneumatic commercial drug manufacture. The quantity of powdered drugs could be controlled by a process similar to electrostatic powder coating. Cost savings can be achieved by effectively controlling the mass flow rates of the powdered drugs.

4.4.2 Pneumatic powder flow subsystem

The operation of the pneumatic powder flow subsystem has been detailed in the review chapter.

Figure 4.3 contains the details of the pneumatic powder flow subsystem. The specific units updated, within this subsystem, are the sensor and cyclone (shown within dashed lines on the diagram). Work carried out on these units is described below.

A. The Sensor Unit

The detailed structure of the sensor unit is provided in appendix C. In the previous work, the pulse charge technique was used to inject charge into the powder flow. Three downstream sensors detected the charged powder flow. A fourth downstream electrode (earthed) was used to neutralise excessive charged powder.

One of the requirements of the pulse charge technique is to sense controlled pulse charged particles. Hence, any charge generated, prior to the injection of the controlled pulse, constitutes a source of error. From digital oscilloscope observation, the signal output from the sensor head was noisy. This is partly due to the tribo-charging noise, superimposed on the injection pulse. Hence, there was the need to reduce sensor noise by minimising the tribo charging effect.

An earthed tribo-charge neutralising electrode has been inserted before the pulse charge injection unit. The electrode is expected to neutralise the tribo charges which would interfere with the controlled pulse injection. The redeveloped sensor unit is illustrated in figure 4.4.

The hypodermic needle, on the sensor unit, is currently, not rigidly fixed to the sensor. Hence, it can be dismantled easily from the entire sensor body for occasional cleaning.

B. The Cyclone

Another section of the pneumatic powder flow subsystem that required close examination was the cyclone. The cyclone performs the following basic function:

The absolute mass flowrate (see figure 4.3) can be directly measured using the electronic balance and the mechanism, used in directing the flow to the balance, is the cyclone. The basic feature of the cyclone is to recover powder out of the powder/air flow. The essential requirements of the cyclone are as follows:

1. It must be highly efficient (ideally 100%) in recovering the powder from the powder air mixture.
2. There must be efficient air exhaustion device from the cylinder of the cyclone.
3. There must be efficient matching of the pipe inlet of the cyclone to the flow pipe of the sensor unit.

The powder/air mixture, diverted from the flow pipe enters, tangentially, the cyclone through a trapezium shaped pipe and creates a powder/air vortex [9], within the cyclone's cylinder. Due to the spinning motion of the vortex, the powder particles migrate downwards under the influence of centrifugal and gravitational effects. The air from the mixture spins upwards due to the inner vortex effect. A diagrammatic illustration of these vortex activities is as shown in figure 4.5. The powder particles from the cyclone are collected on the electronic balance for absolute mass flowrate measurement.

The particle collecting efficiency of the cyclone depends on the following factors:

1. the difference in density of the powder particles and the conveying gas.
2. the solids (powder) concentration
3. the inlet (trapezium) gas velocity.
4. the diameter of the cylinder

Item 4 was a major factor of consideration which was optimised in line with current theories.

4.4.3 The signal conditioner

The signal conditioner is another section of the system (figure 4.2a) where updates were made. The operation of the conditioner was reviewed in chapter 3. Analogue signals generated by the powder flow sensors have to undergo signal conditioning prior to sampling and processing. The following modules of the instrumentation were investigated with a view to improving their signal-to-noise ratio.

A. Current to voltage conversion

The circuit diagram of the current to voltage converter (C-V) is shown in figure 4.6. The currents, induced on the 4 sensors (the injection and 3 downstream sensors) by the charged powder particles, are of the order of 1-5uA. These currents are converted to voltage for real time signal processing.

The C-V should possess low offset, low noise, high input resistance and high sensitivity (sensitive to low input current) [5a]. To meet these requirements the National JFET op-amp such as the 351 was chosen for the circuit. An alternative option is an instrumentation op-amp such the FET OP111

The frequency of the sensor currents ranges from dc to 1kHz. Hence, a simple, first order, low pass filter was incorporated on the C-V. The filter is comprised of a high stability feedback resistor (R_F) with 1% tolerance, placed in parallel with a polyester capacitor (C_F). In addition to filtering, the feedback resistor also provides stable gains.

In the previous research work protection diodes were used to protect the circuit from high current upsurge from the flow. However, from experimental observation, the protection diodes constituted an unwanted source of negative feedback. This led to circuit instability and oscillation, particularly when loaded with the buffer output stages. Hence a simple series resistor (R_2) was used for limiting the sensor current upsurge. The various components used (passive and non-passive) were of very close tolerance (1%) and this takes account of the sensitivity of the sensor currents.

B. Voltage amplification

For a typical voltage of 0.8mV (expected from the output of the C-V converter) to be successfully transmitted to a remote processing unit, in a noisy environment, it is necessary to buffer, amplify and differentially transmit the signals. The circuit design incorporates differential input, and output, instrumentation amplifiers [5b, 6a] and buffers on the transmission line. In view of the fluctuating nature of the powder flow signals, the gains of the amplifiers are variable such that when the signal level is low then the gain is increased. Ideally, the gain ranges from unity to 1000.

The IN102 instrumentation amplifier used was of the low power and high accuracy Burr Brown type.

In the current mode of operation (figure 4.7), the instrumentation amplifiers (IN102) at the transmitting (output of the C-V) and receiving ends are linked through screened and twisted transmission lines. This will further reduce the noise interference [6b]. At the receiving end, voltage level shifting is carried out to align voltages at the buffer output before further signal conditioning to attain the +/-1.28V A/D inputs.

C. Multiplexing and programmable gain units.

The multiplexing and programmable gain functions have been discussed in the review chapter. The updates that were made here are as stated.

1. The CMOS analogue multiplexer (National 74HCT4353) was set up from the digital interface. This enabled the sensor channels to be selected.

2. Sensor channel gains were set up, on a programmable gain circuit, using a demultiplexer (National 74HCT4353).

A functional diagram of the programmable gain circuit links to the A/D and the multiplexer is illustrated in figure 4.8.

4.4.4 The analogue to digital converter

A high resolution A/D requires a tighter performance from the signal conditioner. Typically a 10 bits A/D requires an accuracy of 0.1%. An 8 bits A/D for the above voltage range would allow 4mV resolution with an accuracy of 0.4%. An 8 bits flash converter was adopted for the system (same as previous work), taking into account that the input signal from the signal conditioner would largely determine the accuracy expected from the system.

Some of the updates carried out on the A/D module were:

1. to initiate its conversion from the digital interface (to be discussed in section 4.4.5)

2. to minimise bit errors on its conversion output by proper grounding of the module (separating digital ground from analog ground).

4.4.5 Digital input interface

The digital interface is a programmable unit on the system block diagram. An EPLD digital interface unit was originally used for gain computation and intra-sample adaptation. The EPLD unit was unduly complex and expensive for this application. Hence, the unit was replaced with a simple programmable digital interface capable of:

1. Initiating the A/D conversion.
2. Sequential switching of the multiplexer channels
3. Controlling the channel gains
4. Buffering the A/D data prior to processing.
5. Controlling the pulse charge injection.

A functional diagram of the digital interface is shown in figure 4.9. Part of the interface is a memory mapped decoder. This enables the hardware devices (A/D, latches) to be memory

mapped on the first stage processing transputer. The devices could be programmed to initialise the A/D, configure the gain, select the appropriate channel, control the pulse injection. The programming aspects will be discussed in the chapter 5.

4.4.6 The transputer unit

The transputer unit is required to acquire data samples and process the signals Its 4k on board RAM is suitable for running the system program.

The INMOS links use duplex and synchronised communications [7]. A message on the link is transmitted as a single byte communication. A byte transmitted on a link is normally acknowledged by the receiving end. Single byte buffers are required, on the link ends, to buffer information.
One aspect of the transputer processing units that required careful investigation was the serial communication links. Initially, continuous communication on the channels was unreliable. The initial step was to ensure that the ends of the links were properly buffered. The data acknowledgement on the transputer consists of a 1 and 0 bits. This state was continuously monitored, during the run of the instrumentation.

The link signals are TTL compatible and signals could be successfully transmitted over a short distance by using a simple wire connection. To eliminate noise and cross talk during the transmission, screened twisted pairs of conductors

were used to connect the links.

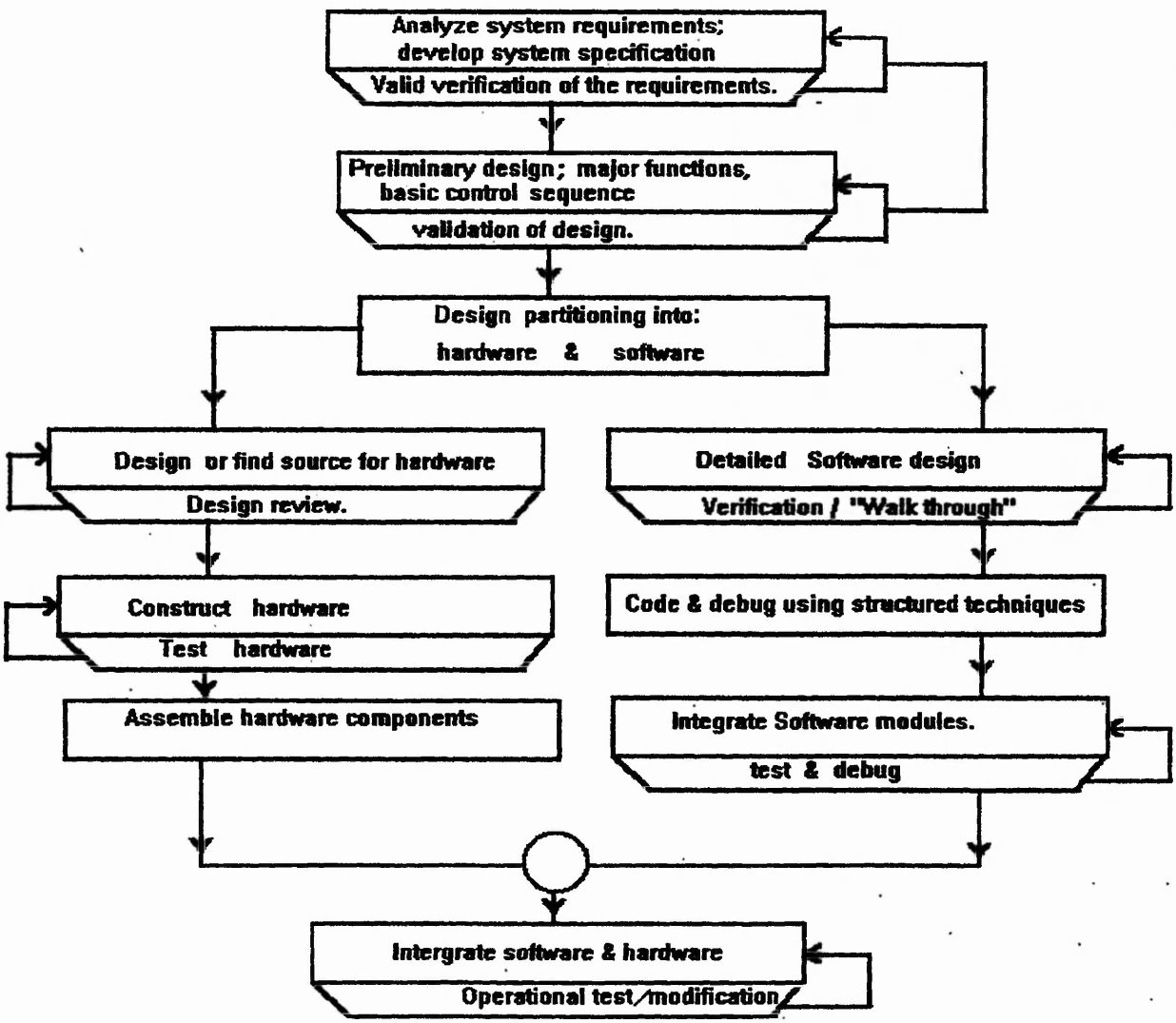
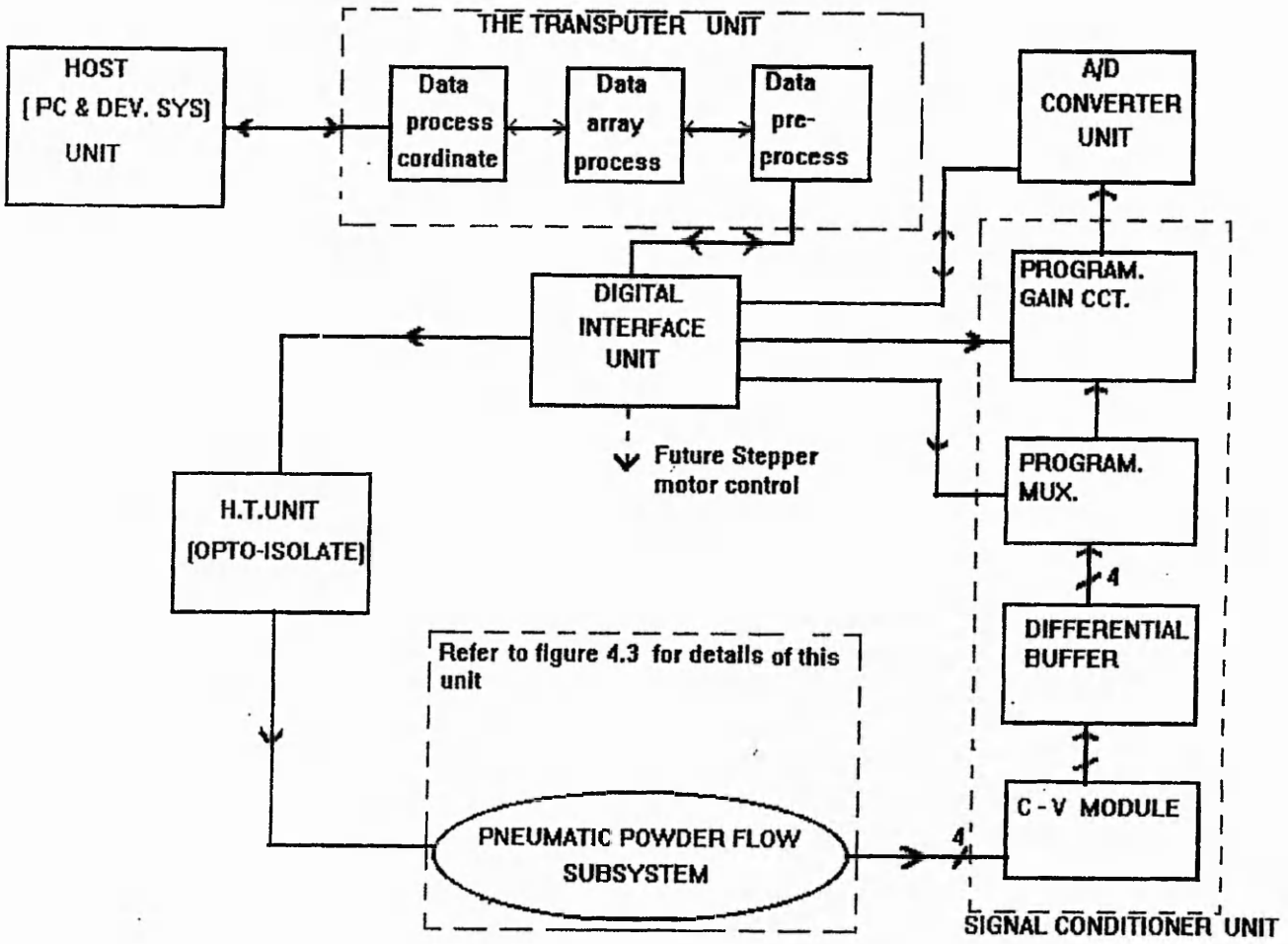


FIGURE 4.1 : THE POWDER FLOW SYSTEM DEVELOPMENT CYCLE

FIGURE 4.2 a. THE SYSTEM BLOCK DIAGRAM



THE CONTROLLABLE PNEUMATIC PROCESSSES

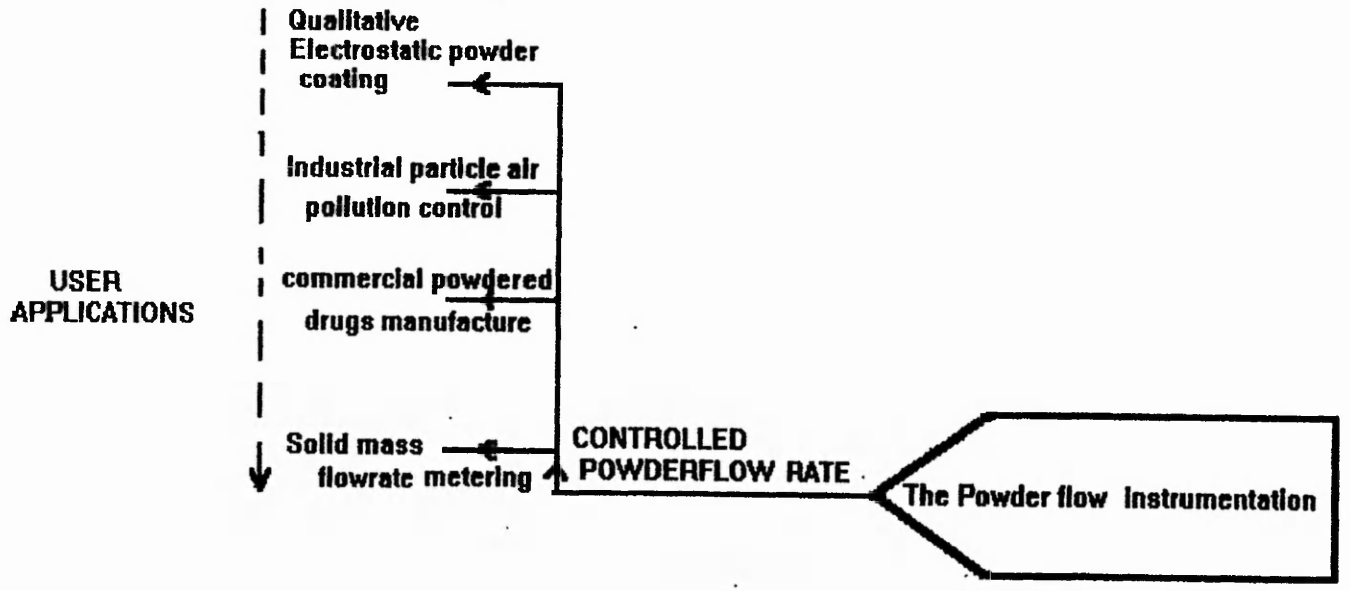
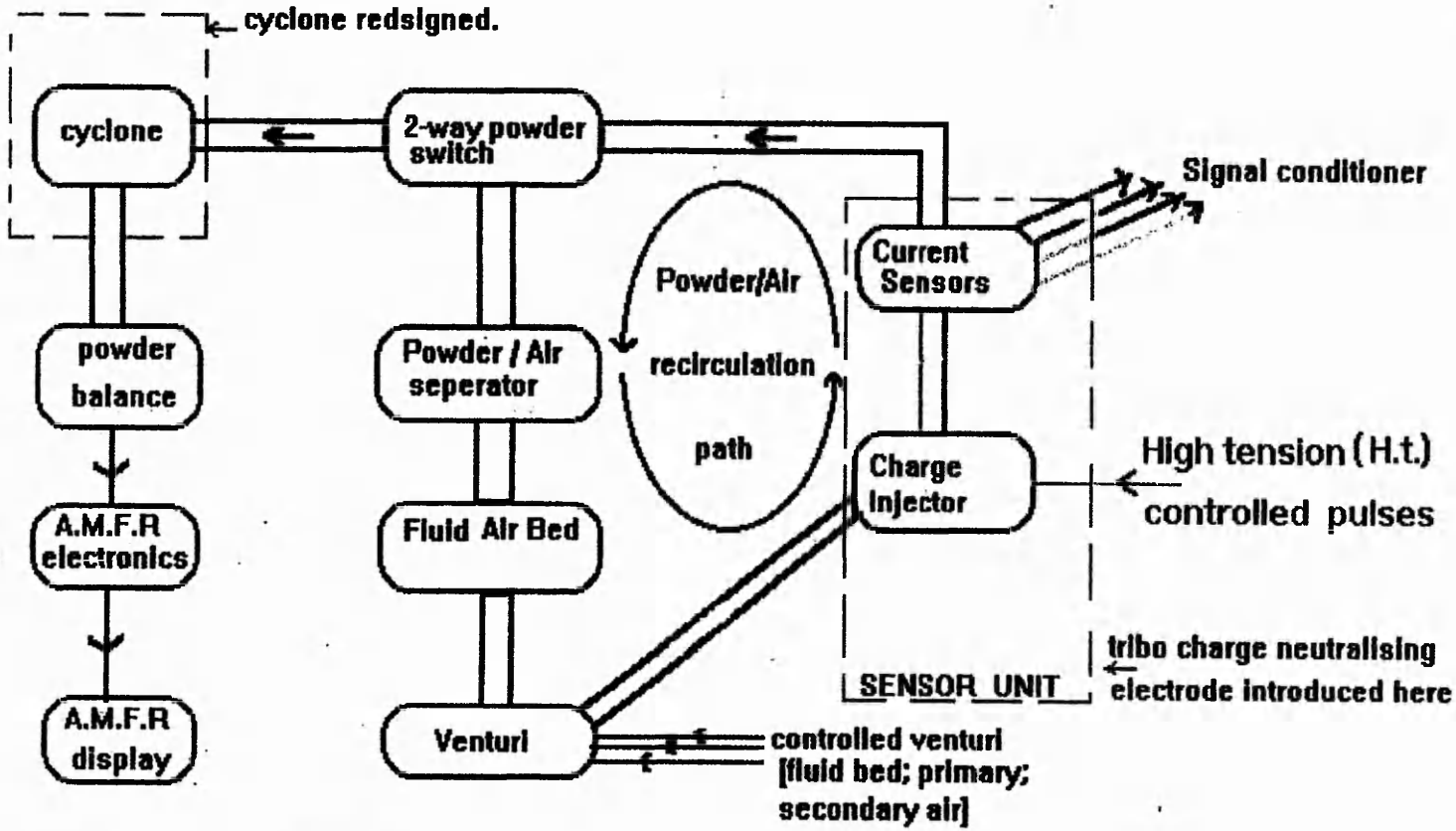


FIGURE 4-2b



A.M.F.R: absolute mass flow rate

FIGURE 4.3: DETAILS OF THE PNEUMATIC POWDER FLOW UNIT

AS REFERRED FROM FIGURE 4.2 a [SYSTEM BLOCK]

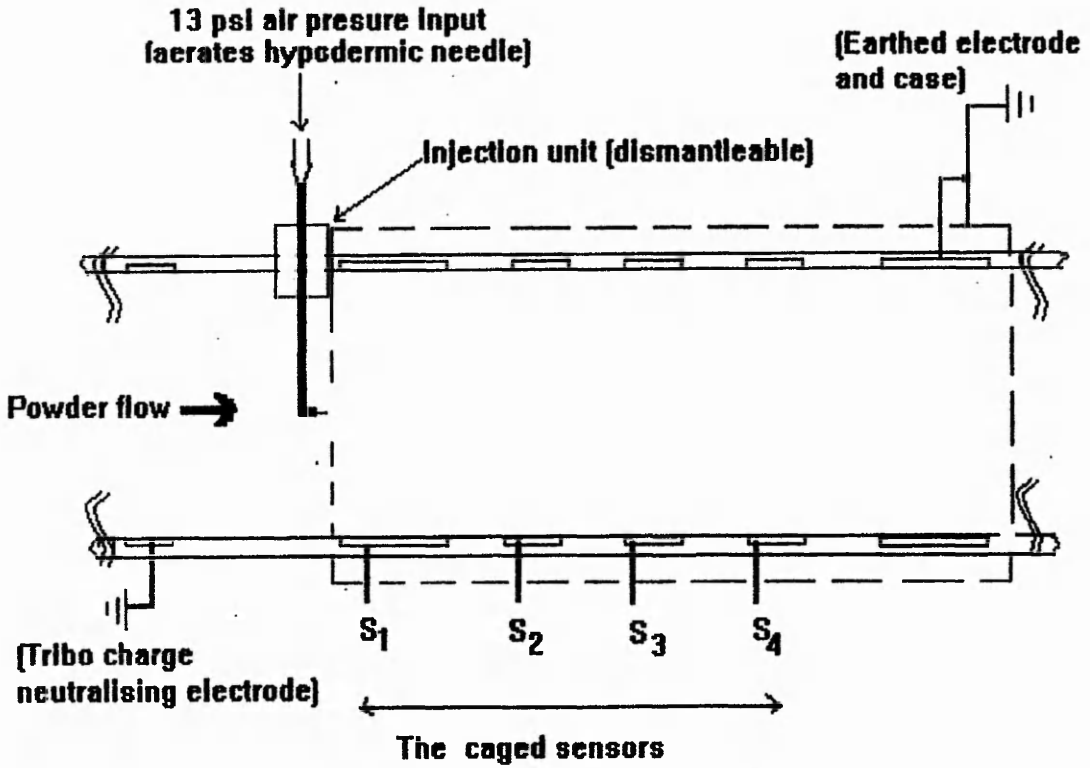


FIGURE 4.4: THE REDEVELOPED SENSOR UNIT

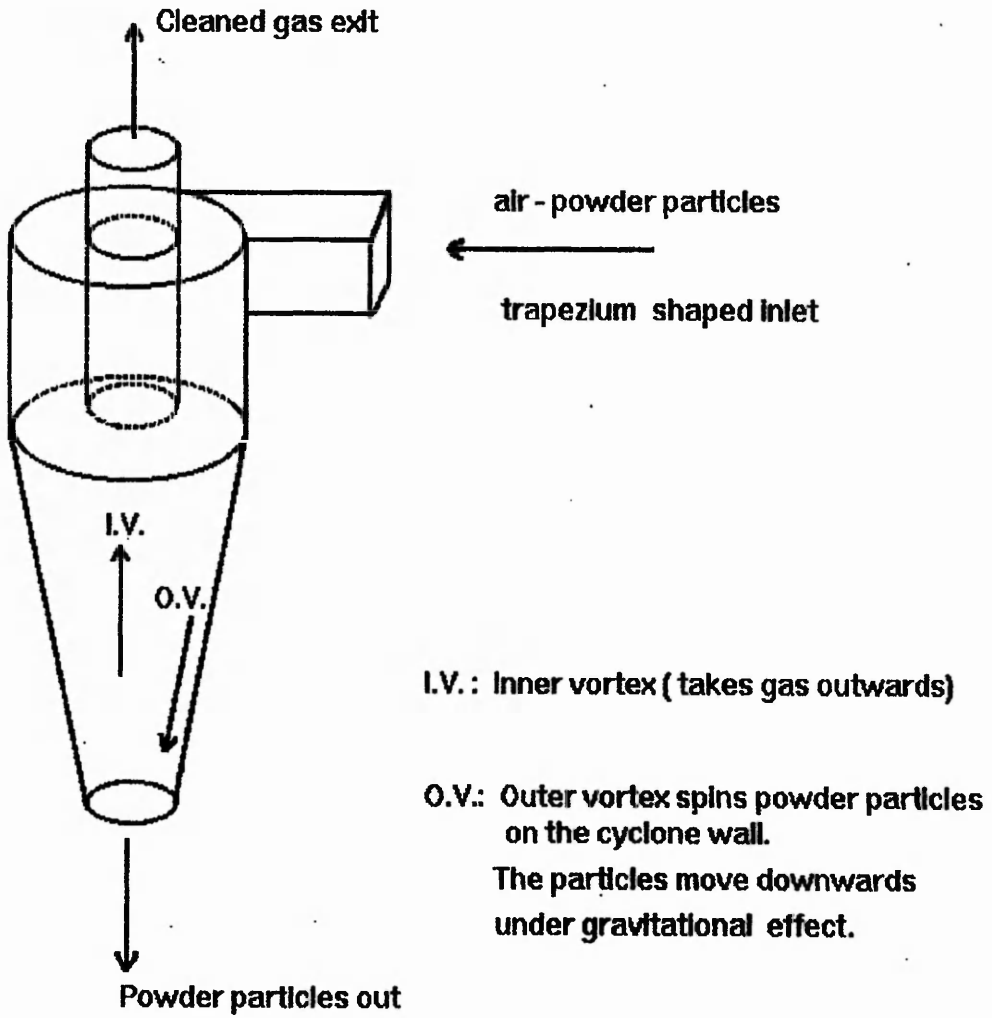


FIGURE 4-5: THE PRINCIPLE OF THE CYCLONE'S VORTEX ACTIVITIES FOR PARTICLES-AIR SEPARATION

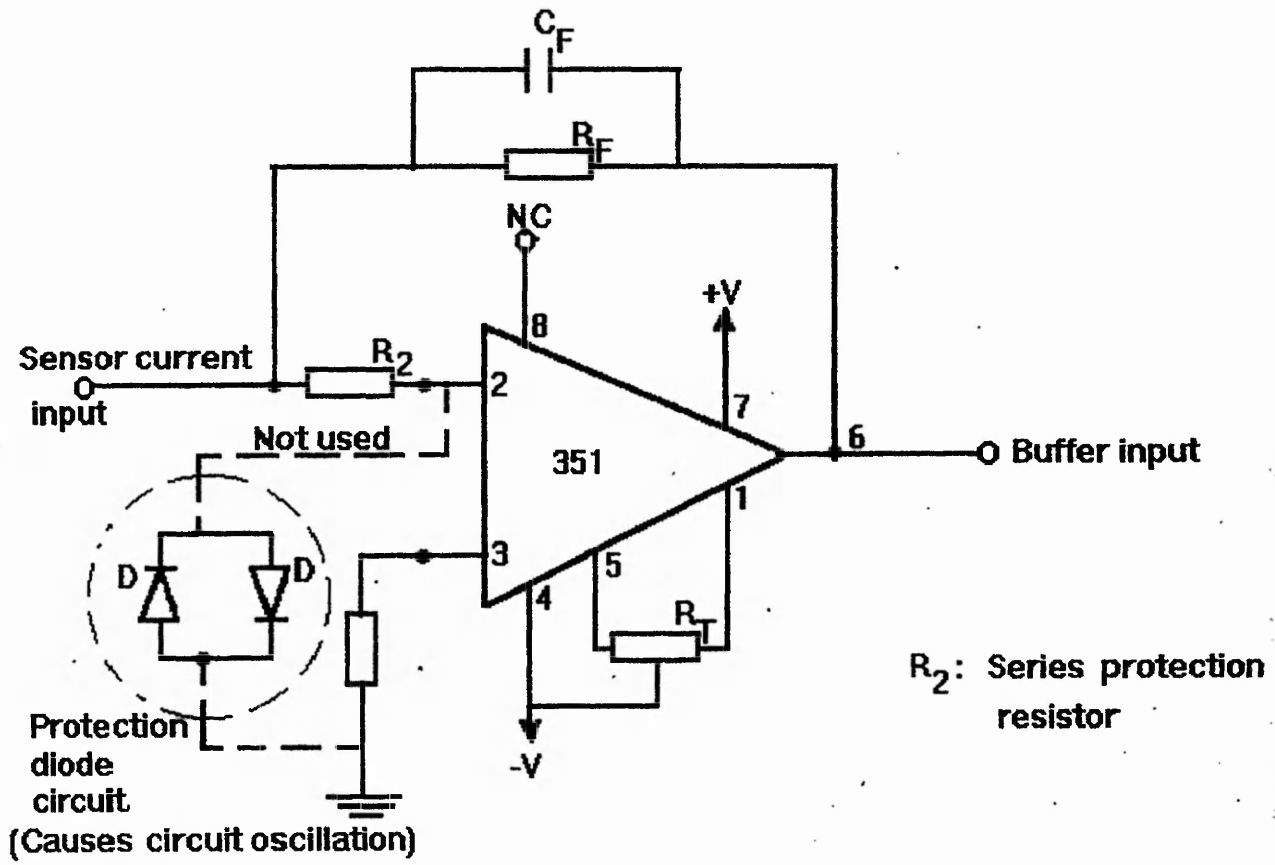
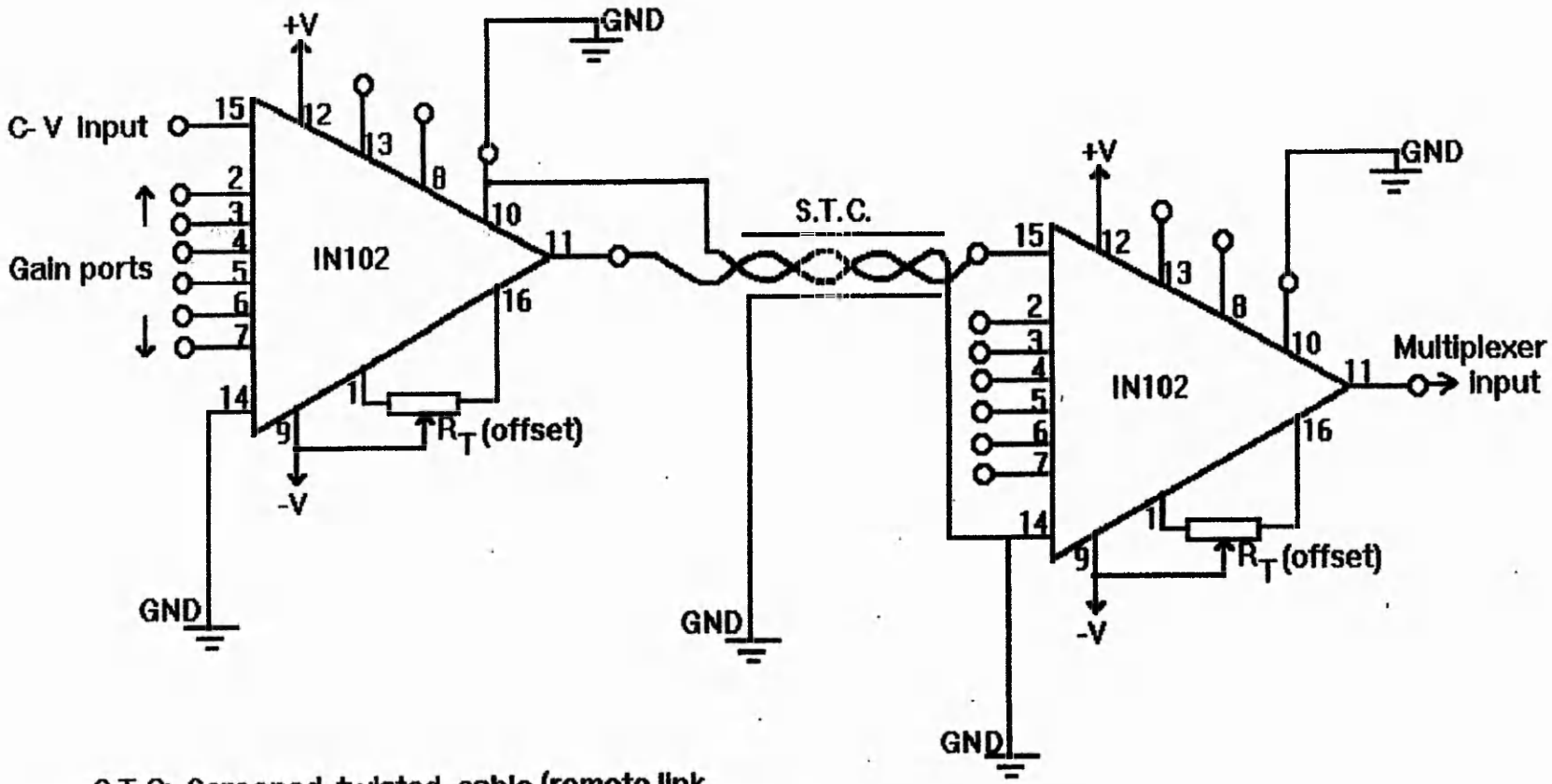
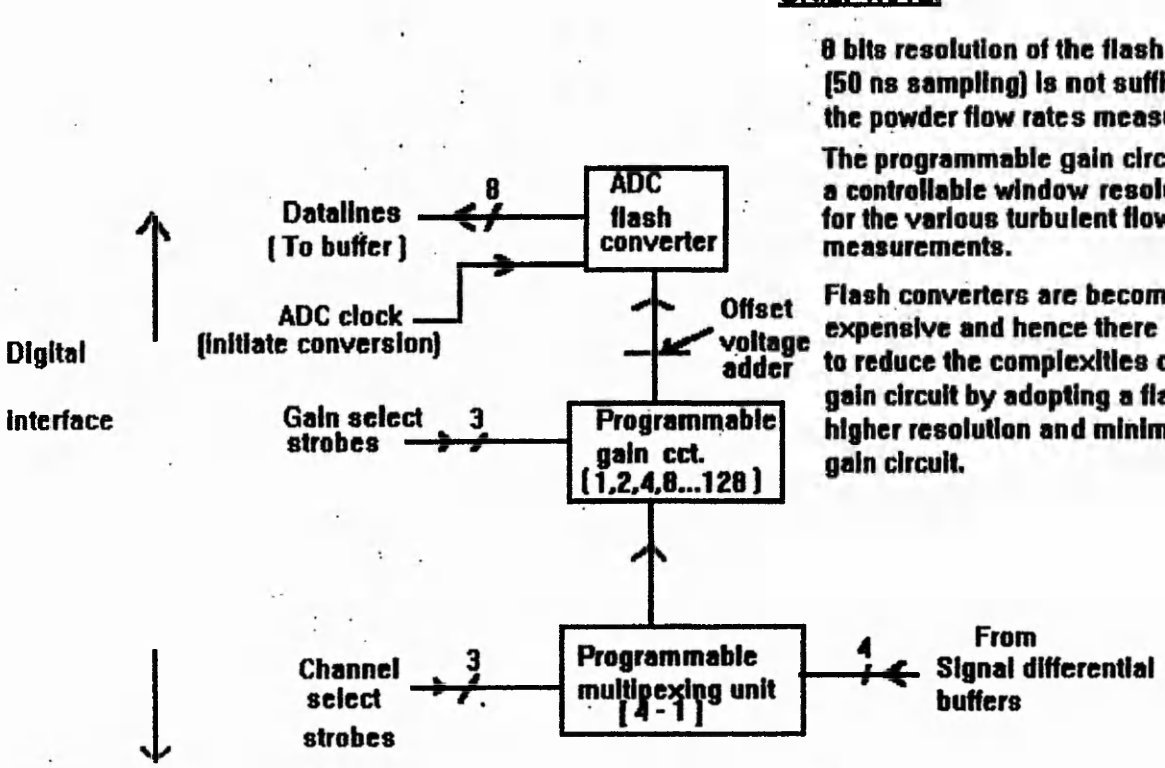


FIGURE 4.6: THE CURRENT TO VOLTAGE CONVERTER CIRCUIT

FIGURE 4.7: AMPLIFICATION AND BUFFERING OF THE SENSOR SIGNALS (Single channel)



S.T.C: Screened twisted cable (remote link between the buffer stages)



BRIEF NOTE:

8 bits resolution of the flash converter (50 ns sampling) is not sufficient for the powder flow rates measurements.

The programmable gain circuit provides a controllable window resolution suitable for the various turbulent flow rate measurements.

Flash converters are becomming less expensive and hence there might be need to reduce the complexities of the programmble gain circuit by adopting a flash converter of higher resolution and minimising the gain of the gain circuit.

FIGURE 4.8: PROGRAMMABLE GAIN LINKS TO THE A/D, DIGITAL INTERFACE AND MULTIPLEXER

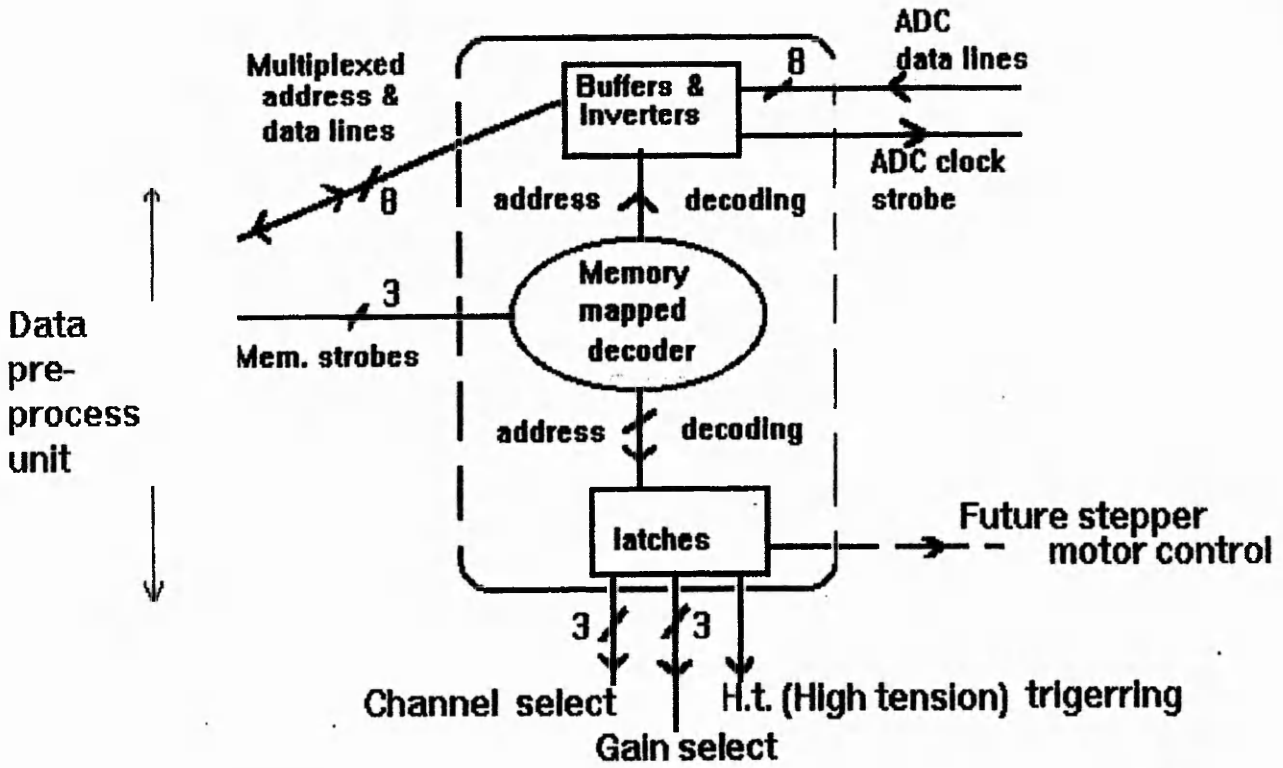


FIGURE 4.9 : THE DIGITAL INTERFACE

4.5 References

1. **Willis, C.A:** "Continuous Mass flow Rate and Velocity measurement of Pneumatically conveyed powder"
PhD thesis of Trent Polytechnic, Nottingham 1984

2. **Mills, E:** "Transputer Instrumentation for Particle Flow Measurements" PhD thesis of Trent Polytechnic, Nottingham 1989

3 **O'Neill, B.C:** "Arthur Holden Fellowship report on a study of powders in the Electrostatic spraying process, Tribo charging, P28-29" Southampton University, Southampton

4a, 4b. Lawrence, P.D., Konrad, M: "Real-Time microcomputer system design: An Introduction" McGRAW Hill International Edition, Computer Series, ISBN 0-07-100561-7, P69-73, P74-86

5a, 5b. Walter G.J: "IC Op-Amp Cookbook: Amplifier Techniques" Third Edition. HOWARD W. SAMS & Company, ISBN 0-672-22453-4, P387-388, P331-354

6a, 6b. Horowitz, P., Winfred, H: "The Art of Electronics: Precision circuits and low noise techniques" Second Edition. CAMBRIDGE University Press, ISBN 0-521-37095-7, P421-430, P455-466

7. INMOS Limited: "The Transputer Applications Notebook: Systems and Performance" SGS-THOMSON, June 1989, INMOS document Number: 72-TRN-205-00, P30-47

8. Ighodaro, D.A., O'Neill, B.C:"Powder flow-rate measurements by means of a pulse charge injection technique" Inst.Phys. Conf. Ser. No. 118, Electrostatics, 1991, Oxford, P135-140

9. Cross, J.A., Cetronio, A: "Advances in electrostatic coating techniques" INSPEC Internat. Conf.,1978, Abington, England, P169-73

5.0 THE SOFTWARE DESIGN

5.1 Introduction

The current software is designed to meet the needs of the system, discussed in chapter 4. It is comprised of various modules which perform functions relevant to the powder flow measurements.

5.2 The system design principles

In chapter 4 the system design principles (illustrated in figure 4.1.) were described. Part of diagram 4.1, relevant to the current chapter, is shown in figure 5.1. Figure 5.1 illustrates the software development. This involved the following steps [5]:

1. Initial verification of the software tasks.
2. Coding of the various modules
3. Testing of the modules
4. Integration of the modules on the transputer development system.
5. Integration of the software and the hardware.
6. Testing of the integrated module

Items 5 & 6 were not completed in the current work and hence left for further work.

5.2.1 Initial verification of software tasks

The verification of the system software was carried out by earlier workers [1,2] and discussed in chapter 3. Briefly, the various objectives of the software are to: capture valid sensor data, preprocess the data, perform mass flowrate analyses.

5.2.2 Coding of the various modules

Occam (6,7,8) was chosen for the development language, in continuation of the previous research software. Occam is purposely designed for efficient running on a transputer [3,4a]. Furthermore, Occam offers the programmer the ability to map an application onto a yet-to-be determined number of processors, maintaining all the potential for parallelism that exists in the underlying application [4a].

The transputer types used were the T800 and the T414. The T800 is a 32 bit floating point processor with multiplexed address and data buses [4a]. The T400 is a non-floating point processor.

5.2.3 Testing of the modules

Testing of the various modules was carried out after coding. The Occam 2 compiler and debugger, from direct experience,

were found to be inefficient and not particularly 'user friendly' during the programming phase. This led to significant delays during the research. The situation would have been improved by the use of a more efficient compiler and debugger.

5.2.4 Integration of the modules

The integration of the modules is illustrated in figure 5.2 (data flow diagram). The integration is the configuration of the system software for interactive working of the modules.

On successful compilation and debugging of the individual modules, the system modules were interlinked and configured to initially run on the transputer development system, before being downloaded on the system hardware. This process was carried out to enable easy debugging of the system.

With a simple programmable interface it was possible to memory map the analog hardware devices such as the A/D, multiplexer, HT supply and signal conditioner gain units.

5.3 The system data flow diagram

During the current study, the data flow diagram (figure 5.2.) was partitioned into four modules which run on a transputer network. These are :

1. Data capture and distribution
2. Signal peak detection and charge integration
3. Data storage and processing
4. Pc host user interface

The details of the processes and the areas covered during the current study are provided below.

5.3.1 Data capture and distribution

This is the datacapture and preprocessing module. This software runs on the T400 transputer which addresses the digital interface (linked to the analog to digital converter, the multiplexer, and the gain amplifiers). Various processes are carried out on this module. To perform the processes, the addresses of the hardware devices (A/D, latches, multiplexer and buffers) were memory mapped on the first transputer. The devices were read, as memory locations.

This module was programmed to perform the following functions:

1. injection pulse triggering
2. sensor channel and gain selections
3. fast datacapture and distribution
4. injection detection

An explanation of the functions are provided below.

1. Injection Pulse triggering

The powder flow is charged through the High Tension (HT) voltage (0-15kV) source which is pulsed by the software. This is memory mapped and hence programmed from a latch on the digital interface. A digital pulse generator was programmed to trigger the HT supply at repetitive intervals. The trigger pulses were timed to obtain charge injection on the powder flow. The parameters for pulse timing are supplied by the user interface program. HT pulse triggering is carried out simultaneously with sensor channel and gain selection (described in section 2).

HT pulse triggering has not been automated in the current work.

2. Sensor channel and gain selections

The multiplexer is programmed to select the sensor channel and the appropriate gain values for the input amplifiers. This process is carried out through the latches on the digital interface. The software is not automated to control the channels and set the gains. However, the user can input the gains and channel desired using programs written for the server and the development system. The network program passes the user message through the datacapture transputer and the digital

interface unit, to the channel multiplexer and gain select units. During operation, the user can switch to any of the four sensor channels and select gains of 2^n (where n = 0 to 7).

3. Fast datacapture and distribution

During data capture, 4000 samples are taken from within the 20ms HT injection cycle. This is repeated for any desirable number of cycles, on any of the four channels. The rate of datacapture is improved by the implementation of a 'round-robin' algorithmic structure. A fast algorithm is required because of the rapidly changing statistics of the sensor waveform. To avoid transputer links waiting on the processor, the software design should decouple link communication from computation [4c]. This helps to maximise the multiprocessor performance.

A typical round-robin algorithm that achieves improved performance is illustrated below:

```
[data arraysize] BYTE p1,p2,p3:
  ...proc receive
  ...proc process
  ...proc send
  SEQ
    input (p1)  -- start sequence
```

```
PAR
  input (p2)
  process (p1)
  WHILE no.of samples less request number
    SEQ
      PAR
        input (p3)
        process (p2)
        send (p1)
      PAR
        input (p1)
        process (p3)
        send (p2)
      PAR
        input (p2)
        process (p1)
        send (p3)
```

From the structure, the buffers (p1, p2, p3) pass a round-robin between the input (p1), transformer (p2) and output (p3). But in this application only two buffering processes (p1 and p2) are used to capture (receive) and distribute (send) data to other processes in the network. The injection detection process (p3) is removed from this round-robin technique. Process p3 is explained in section 4.

The data array slicing, at the top of the structure,

involves defining a protocol variable which could be used for an array slice of 16 or 32 bytes. The effect of array slicing can be explained as follows [4c]:

> The transputer loops in about 1 microsecond but adds in about 50 ns. Abbreviations, such as the one shown below, are used to open out loops, which in effect speeds up the speed of execution. Consider a structure such as this which performs addition 3000 times.

```
SEQ i= 0 FOR 3000
  a[i] := b[i] + c[i]
```

Such a process is inefficient. To reduce the execution time array slicing abbreviations such as the structure below is recommended:

```
VAL sliceloop IS 3000 >> 4: -- 3000/16
VAL remainder IS 3000-(sliceloop*16): -- rem. after division

SEQ
  SEQ i = 0 FOR  sliceloop
    VAL base IS i TIMES 16:
    aslice IS [a FROM base FOR 16]
    bslice IS [b FROM base FOR 16]
    cslice IS [c FROM base FOR 16]
```

```
    SEQ i = 0 FOR 15
      aslice[i] := bslice[i] + cslice[i]
SEQ i = 3000 - remainder FOR remainder
  a[i] := b[i] + c[i]
```

This standard software structure provided a basis for implementing fast datacapture and processing algorithms on the modules.

In these algorithms, array slicings were used to improve the speed of the software.

Opening out loops in slices or offsets of 16 is a significant feature of the transputer because optimal code, with no prefix instructions, is generated for each addition statement [4]. The transputer's DMA (direct memory access) works faster reading packets of 16 bytes than reading packets of 32 bytes or higher.

4. Injection detection

A typical powderflow waveform, showing a successful injection, is shown in figure 5.3. The waveform is comprised of capacitive and injection currents.

1

The terms aslice, bslice and cslice are abbreviations (individually representing arrayslice of 16 bytes). Buffers were used to store the sliced arrays.

However, the data samples captured are not all from valid injections. A prefiltering algorithm is used to filter out invalid data. A 'track injection' algorithm sets limits to discriminate against sporadic or undesirable powder flow noise. Threshold levels are set for acceptable powder flow levels.

In the current approach to injection detection, a slight time delay is introduced to avoid the initial capacitive current before searching for a point of injection. A window length is defined and used to analyse various regions of the waveform to verify injection. If, after 1000 samples, a defined threshold injection point has not been found then the search is abandoned.

The injection detection algorithm includes software to read the status of the flow at various times and to display when the point of injection has been detected. After detection of injection, data capture occurs, using an array slicing algorithm.

Variant protocol is a feature incorporated in this module. This has been used mainly for tight parameter identification on the channels (see section 6.2.3).

Data is serially received from the A/D and transmitted in parallel to the array processing unit (module 2) using the

array slicing method.

The flowchart which explains the software design of the data acquisition algorithm is shown in figure 5.3.1.

5.3.2 Signal peak detection and charge integration

This module deals with the processing of the signals to obtain the flow velocity and the quantity of charge injected into the powder flow. To obtain the flow velocity, array slices of 16 or 32 bytes, from the data acquisition and preprocessing unit, are analysed for peak amplitudes. The four powder flow sensors are addressed by this module for the necessary data capture and peak detection. Charge calculation is also carried out here by integrating the sensor current over a period of time.

The flowchart which explains the structure of the above algorithm is shown in figure 5.3.2.

5.3.3 Data storage and processing

The function of this module is to gather a large amount of data from the array processing unit for analysis of the mass flow rate. The software has been designed to capture array slices of data of 16 or 32 bytes from the array processing unit and then store them for future analysis. Currently this module is capable of storing 2 Mbytes of raw data but the algorithm to analyse and control the mass flow rate has not

been developed (see chapter 7).

The flowchart which explains the structure of the Data storage algorithm is shown in figure 5.3.3.

5.3.4 Pc Host user interface

This is a module which runs on the host transputer and it is the user-software interface to the development system and the embedded network system. It is mainly designed to read the status of the powder flow in the pneumatic system and control the other transputer's network programs. The parameters for the network are fed in from this module. A feature of this module is a small lookup table algorithm for running and testing the system. Its basic functions was to set the gains of the data acquisition at 2^n where n ranges from 0 to 7. Ideally gains of upto 128 could be achieved.

Also provided in this software is the interactive 'user friendly' interface which provides various requests, such as the desired number of data acquisition samples, the number of HT (high tension) pulses requested for the injection, the sensor channel desired on the multiplexer switch.

The status report of the powder flow can be read using a small algorithm. This monitors the following status:

1. A/D overflow; this indicates an input signal greater

than the level of signal that can be analysed.

2. A/D underflow; this indicates a condition of no sensor input on one of the sensors.

3. Increasing sensor signal; this indicates a condition likely to detect a point of injection.

4. Decreasing sensor current; this parameter could also be used to track the point of injection on the reverse edge.

5. Constant change in the sensor current; this indicates a point away from injection in the powder flow analyses.

6. Injection detected: this indicates adequate charging of the powder flow.

7. Injection not detected: this indicates insufficient charging of the powder flow.

The flowchart which explains the structure of the User interface algorithm is shown in figure 5.3.4.

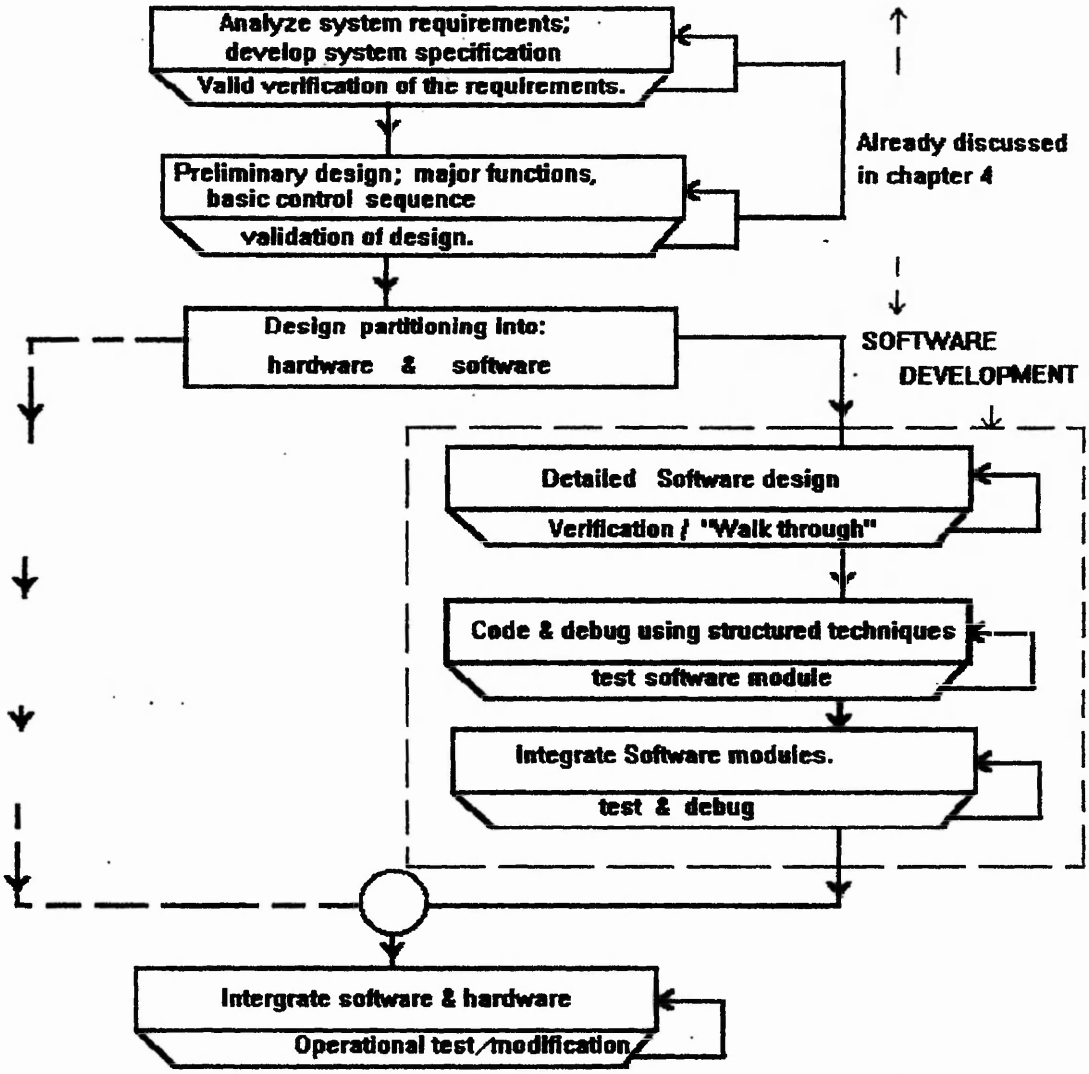


FIGURE 5.1 : The Software development path

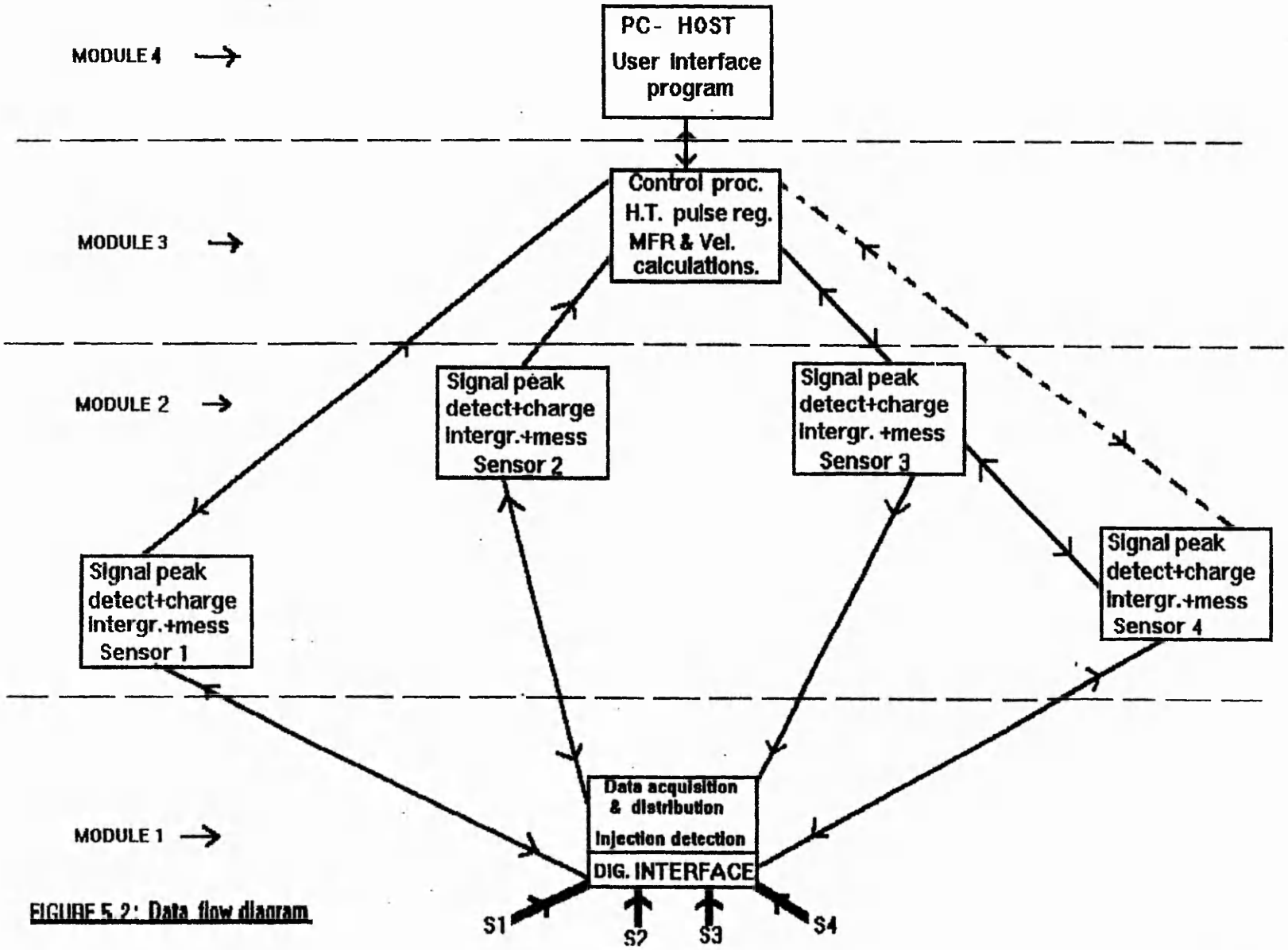


FIGURE 5.2: Data flow diagram

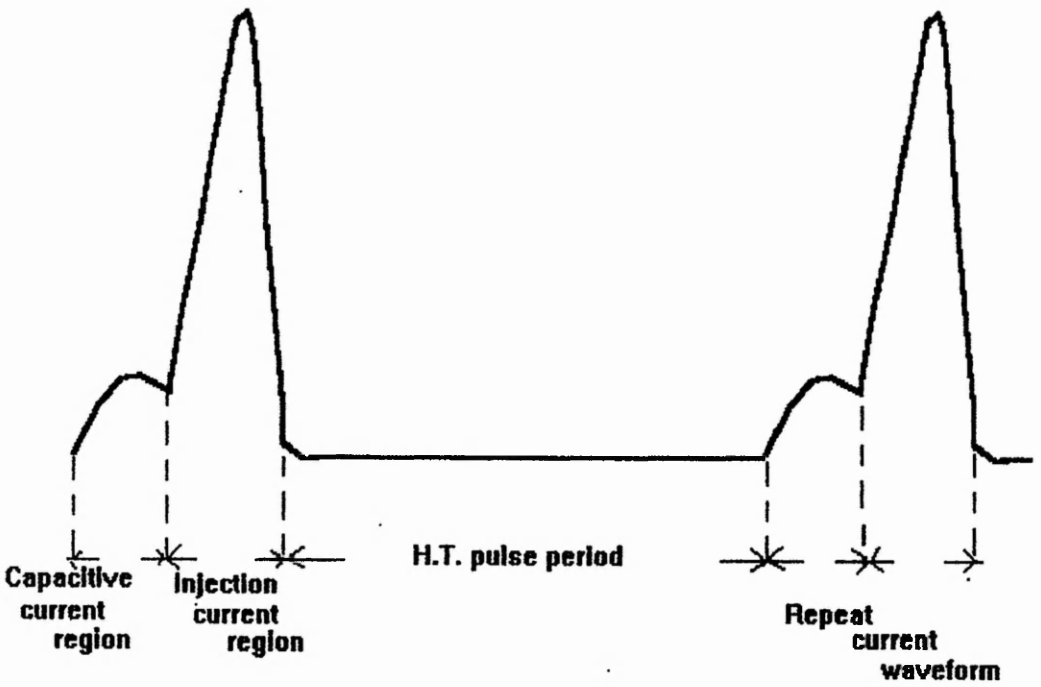


FIGURE 5.3: TYPICAL SUCCESSFUL INJECTION WAVEFORM

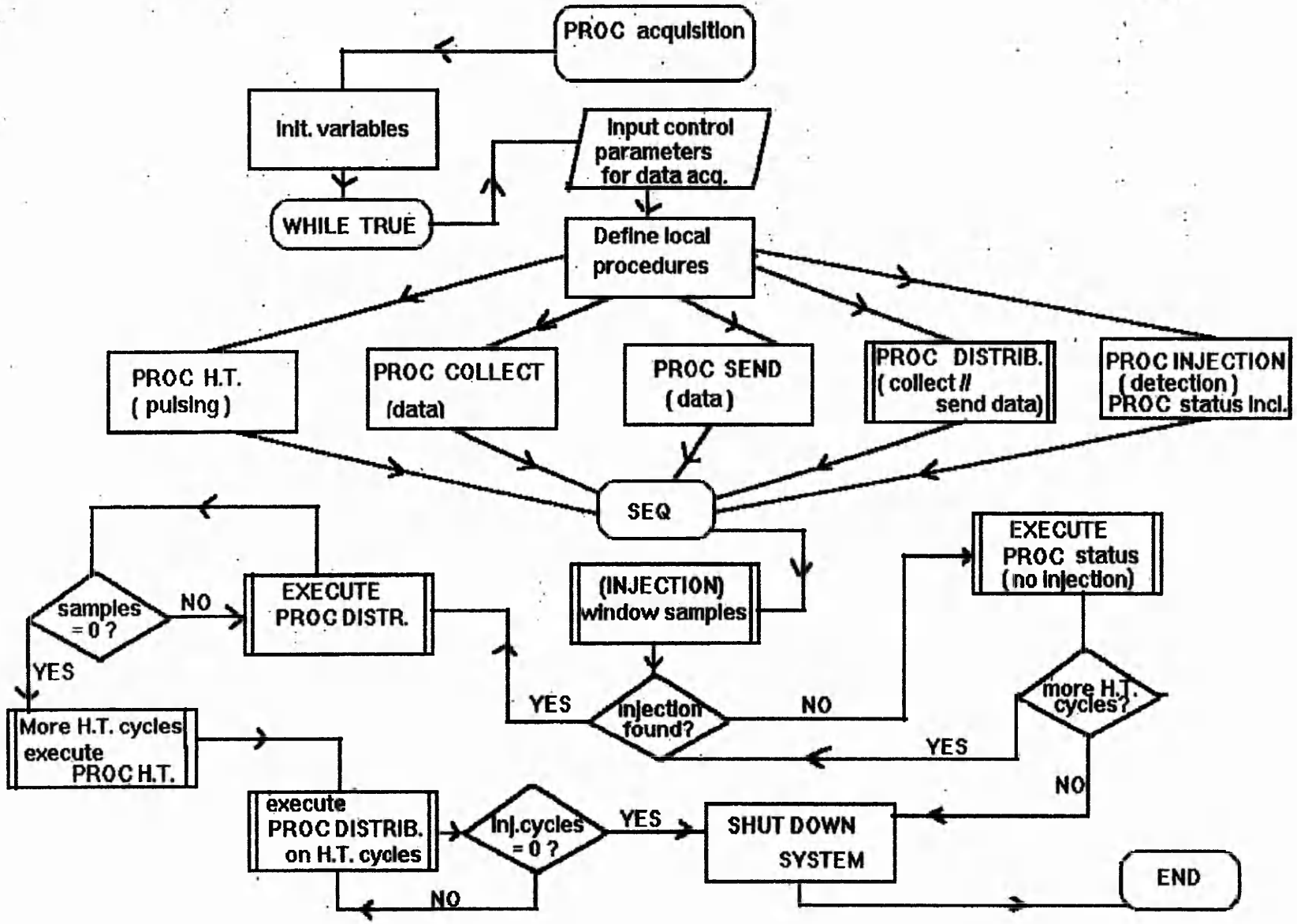


Figure 5.3.1: Data acquisition

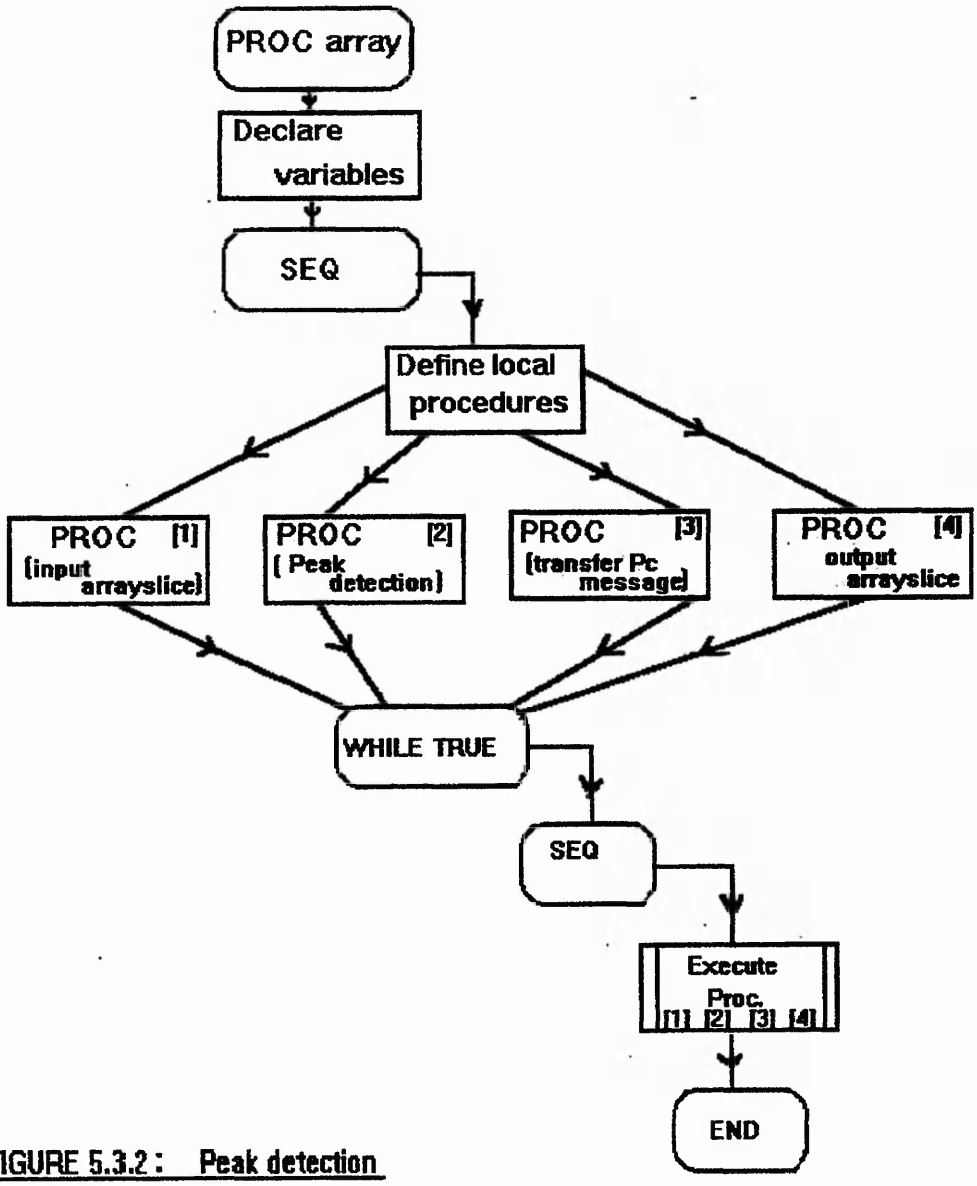


FIGURE 5.3.2 : Peak detection

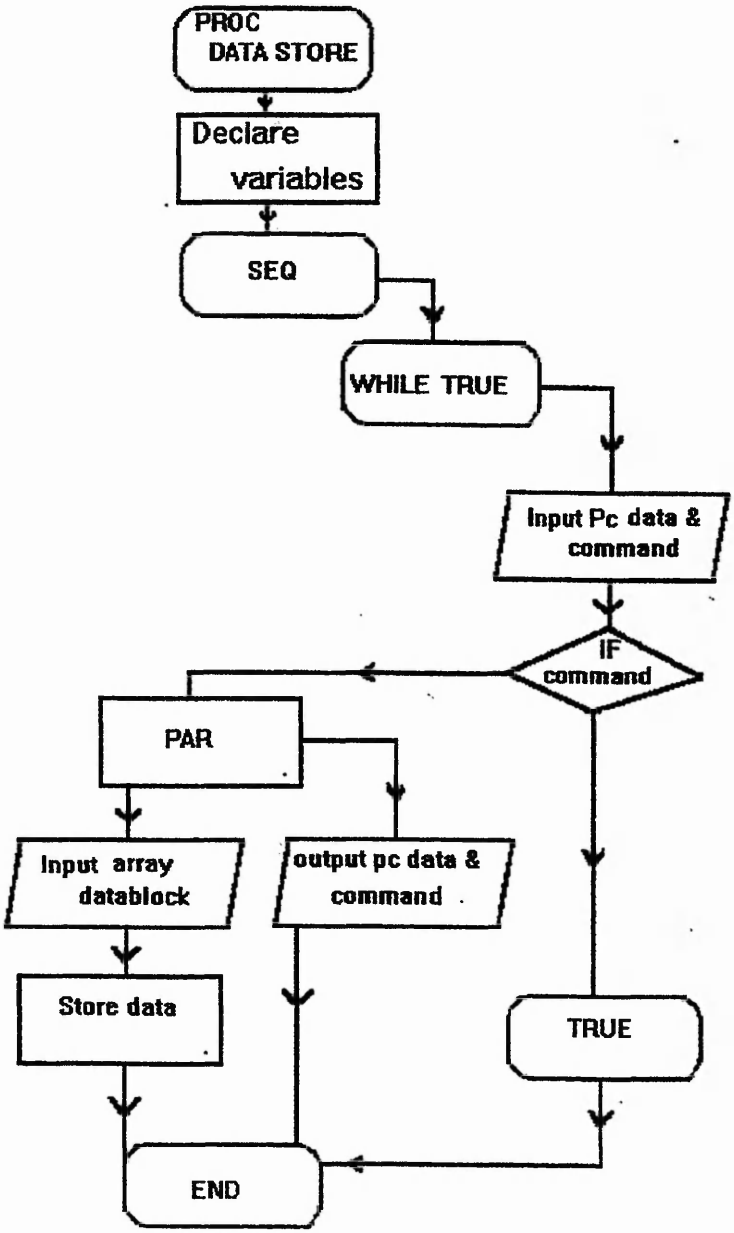


FIGURE 5.3.3: DATA STORE & MESSAGE PASSING

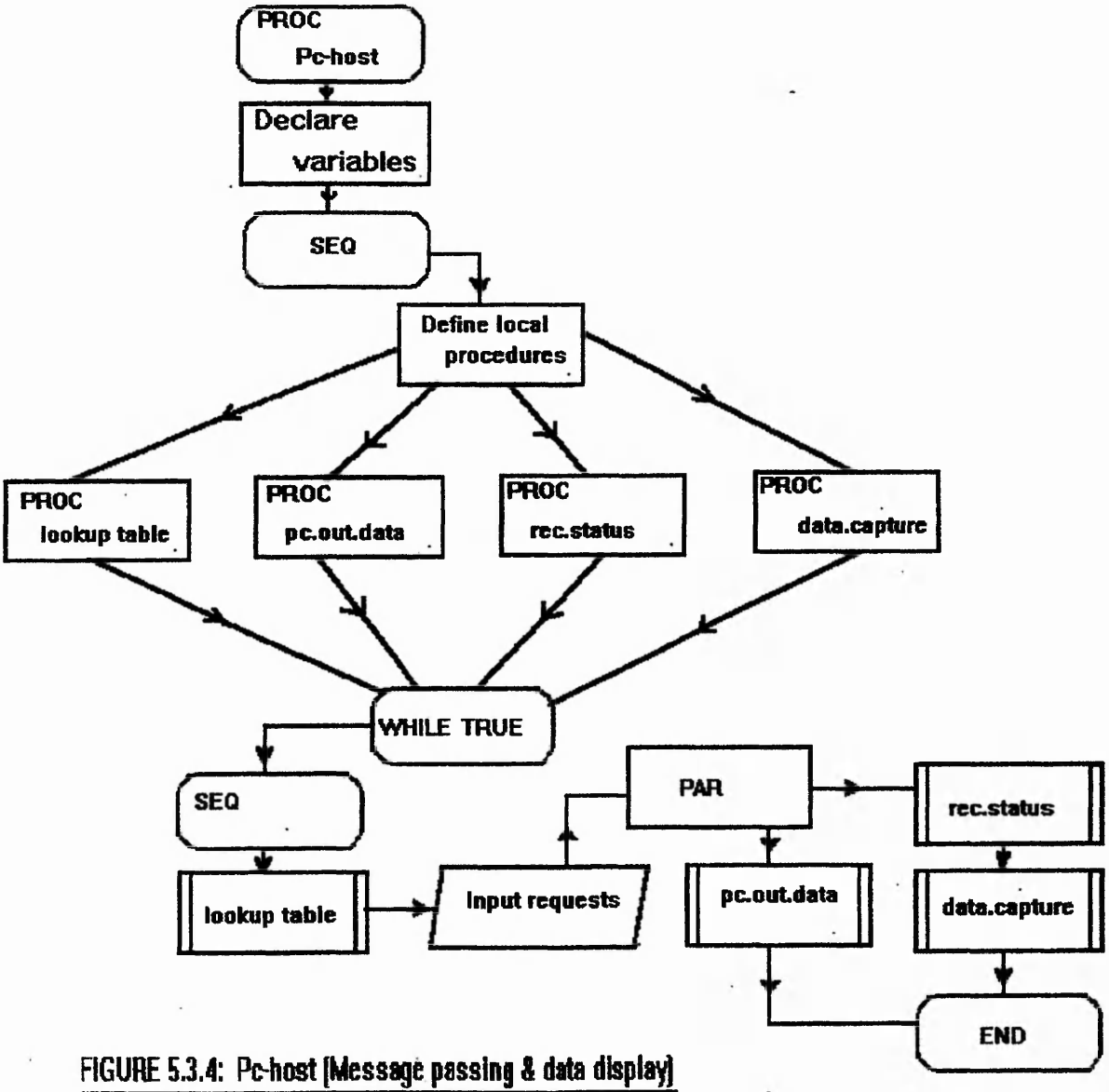


FIGURE 5.3.4: Pc-host [Message passing & data display]

5.4 References

1. **Willis, C.A:** "Continuous Mass flow Rate and Velocity measurement of Pneumatically conveyed powder"
PhD thesis of Trent Polytechnic, Nottingham 1984

2. **Mills, E:** "Transputer Instrumentation for Particle Flow Measurements" PhD thesis of Trent Polytechnic, Nottingham 1989

3. **INMOS Limited:** "Product information. the transputer family" Prentice Hall, June 1986, 72 TRN 094 00, P6-7

4a, 4b, 4c. INMOS Limited: "Transputer technical notes", Prentice Hall, 1989, ISBN 0-13-929126-1, P159-161, P2-21, P228-245

5. **Lawrence, P.D., Konrad, M:** "Real-Time microcomputer system design: An Introduction" Computer Series McGRAW Hill International Edition, ISBN 0-07-100561-7, P69-71

6. **INMOS Limited:** "Occam 2 Reference Manual"
Prentice Hall, 1988, ISBN 0-13-629312 -3

7. **INMOS Limited:** "Occam 2 toolset Manual"
INMOS Limited, 1989, 72 TDS 184 00

8. **INMOS Limited:** "Occam and the toolset INMOS Training Course manual" INMOS limited, 1989

6.0 DISCUSSIONS OF RESEARCH WORK ACHIEVEMENTS

This chapter discusses the major developments in the current work compared to previous research.

6.1.1 Powder flow subsystem

The hardware changes carried out on the powder flow subsystem were explained in section 4.4.2. The cyclone and sensor unit (on the subsystem) were re-designed. The improvements on these units are discussed below.

A. The cyclone

There were two main reasons for the redesign:

1. There was the need to regularly dismantle the cyclone for cleaning. Difficulty was experienced in dismantling the cyclone previously used in this research. The cyclone was redesigned to allow the cylinder unit (body of cyclone) to be dismantled easily from the nozzle unit (particle outlet) for regular inspection and cleaning. The cyclone needs to be regularly cleaned because the particles sticking to the interior of the cyclone affect the vortex activities of the cyclone. The new design should improve the particle collecting efficiency of the cyclone, when operating in the powder flow subsystem.

2. In the previous work, there was not sufficient information on the cyclone development. In the current design, the cyclone was built in accordance with the standard literature design structure for a cyclone (as detailed in section 4.4.2b). This will greatly improve the particle-air separation efficiency of the cyclone when used in the powder flow subsystem.

B. The sensor unit

The sensor unit needed improvement because, from experimental observation, the sensor signals were noisy. Tribo-charging of flow particles was found to be the main source of noise interference (see section 2.2.2). Previous research did not consider a means of reducing the tribo-charging noise. In the current work, a tribo charging noise neutralising electrode has been included in the redesigned powder flow unit.

There was also a need to be able to dismantle easily the hyperdermic needle from the sensor, for occasional inspection (to look for damage or remove sticking powder particles). In the current work, the hyperdermic needle is easily removed from the sensor body.

6.1.2 Signal Conditioner

Changes were made to the signal conditioner (see section 4.4.3), to improve its signal to noise ratio. This was important because internal and external noise interference is undesirable in the measurement of sensitive sensor readings. As a result instrumentation differential amplifiers were used for buffering. These amplifiers possess improved dc response, from experimental observation (see appendix A).

The performance of the protection diodes on the current to voltage converter was studied. These diodes caused circuit instability and hence were removed from the current to voltage converter in order to minimise the internal noise interference on the sensor signals. An alternative circuit protection was utilised.

6.1.3 Digital interface

The digital interface (see section 4.4.5) is now much simplified when compared to the EPLD, used by the previous researcher. The simple digital interface meets the needs of the flow system without loss of functionality. The digital interface could now be programmed to:

1. trigger the HT supply
2. select the multiplexer channels,

3. set the appropriate gain for each sensor channel.

6.1.4 Communication Needs

In the initial stages of the research, the link communications on the transputer network were not reliable for continuous transmission of data. In the current work, the transputer network links were improved. Continuous transmission of sensor data is now possible.

6.2 The software developments

6.2.1 Modularisation of the Software

The algorithms used have been simplified into modules. As a result, it will be easier for users to understand the basic software functions and hence, modify the algorithms, where necessary.

6.2.2 Comprehensive documentation of the software

As part of the comprehensive documentation, flowcharts were produced for the various software modules. This will enable future researchers to understand the objectives of the algorithms. Flowcharts for the full system software were not provided by previous researchers and hence, considerable difficulty was experienced during the current work in interpreting the software used. Furthermore, extended comments

were inserted into the various software modules which enhanced the software documentation (see appendix B).

6.2.3 Channel protocol

Previous research work used an 'alien' or 'anarchic' protocol which allows input and output to communicate without checking for compatibility of message formats. A defined channel protocol (adopted during the current work) communicates values between two concurrent processes and hence, it is necessary for the input and output to be compatible with the protocol. Channel protocol enables the compiler to check the usage of channels.

Variant protocol was introduced for single channel message communication with different formats. A variant protocol specifies a number of different formats for communication on a single channel.

6.2.4 Suitable software for the simplified digital interface.

Previous software addressed a complex EPLD digital interface (Erasable Programmable Logic Device). The software was rewritten to address the simple digital interface.

6.2.5 Test programs for link communication reliability

Short programs were written to test continuous data

transmission on the transputer links. This involved data transmission on the channels using the necessary variant protocol for the parameter passings.

6.3 Overall system software

The overall system software is now geared for reliable data capture and display. Had the laboratory replacement not been delayed, after the fire, the full system would have been demonstrated and verified.

7.0 Conclusions and further work

This section deals with the conclusions of this work and areas of further work that could be addressed to extend the current studies.

7.1 Conclusions

Much has been achieved in the project aim of developing a powder flow instrumentation for powder flow rate measurements.

Hardware and software developments have been carried out, with the objective of realising a reliable powder flow instrumentation.

The following section contains the main conclusions of the research work.

7.1.1 Review of Powder flow measurement techniques

A comprehensive review of powder flow measurement techniques was carried out. A detailed chronological history of the development of powder flow measurements, based on the pulse charge technique, has also been provided.

7.1.2 Improved link communication

The communication problems on the serial transputer links were solved. Hence, the system became reliable for continuous data transmission on the links. Test programs were written and used to confirm the communication reliability.

7.1.3 Improved digital interface unit

A low cost, simple and very effective programmable digital interface unit was designed to replace the complex Erasable Programmable Logic Device (EPLD), used by previous reseachers. The interface unit was programmed easily for gain computation, triggering of injection pulses, and channel selection. The unit is essentially a multifunctional programmable element for various powder flow parameter settings.

7.1.4 Improved analog interface

The analog input unit was re-designed to reduce noise interference on the powder flow signal. Differential instrumentation buffers were introduced, on the signal conditioner, to reduce external noise interference.

7.1.5 Software developments

The system software was adapted to meet the needs of the new

hardware design.

Variant protocol was implemented on the software, providing tight checks on parameter and channel usage (previous research used anarchic channel protocol which allows input and output of any format without checking).

As part of the comprehensive documentation, modular software was adopted. This will enable future researchers to understand easily the objectives of the algorithms.

7.1.6 Software documentation

The software structure was simplified and the software was fully documented for future use and further developments. Flowcharts were provided to explain the software.

7.2 Hardware further work

The areas to be addressed on the hardware are stated below:

7.2.1 Real time performance assessment of the redesigned pneumatic powder flow system.

The real-time performance of the powder flow system can only be fully tested when the system is set up in the near future.

Currently, the powder flow is mechanically switched to the

cyclone for absolute mass flow rate measurements. Future work should replace the current mechanical switch with a programmable digital mechanical switch. The switching process could then be automatically controlled from a transputer.

7.2.2 Reassessment of the current to voltage converter

The use of protection diodes needs to be examined. Currently, the current to voltage converter is protected against upsurge of high current by high resistance series resistors. This provides an alternative to the protection diodes, which caused circuit instability, or oscillations, when circuit performance was investigated. Protection diodes could be studied further, with the intent of improving circuit stability whilst incorporating the diodes.

7.2.3 Input buffer programmable gain links.

The differential amplifiers on the sensor unit have programmable gain links. These are manually adjusted to suit the transmission of the sensor voltages from the current to voltage converter. These links could be programmed from the digital interface. An analog switch (demultiplexer) could be used to interlink the digital interface to the differential amplifiers.

7.3 Software further work

The current software runs on the transputer development system. For future use, the software has to be reconfigured to run on an appropriate transputer network. All the modules could then be integrated and tested for the necessary flow measurements.

The software could be further developed to perform the necessary analyses to obtain a 'reference mass flow rate'. Empirical models suitable for the future control system could then be established. Analysis of various flow regimes (low, medium and high) need to be carried out during experimentation.

A major area of further research, could be image processing of the flow data, to graphically display the actual particle flow's dynamic state at any time. The display would be of interest to the instrument's operator or control Engineer. Image processing may require more sensors at various points of the flow and more transputers to cater for the processing. On-line graphical display of the flow's dynamic state would enable a better understanding of the system's flow mechanisms.

7.4 Full installation and verification of the system

Verification of the current powder flow subsystem (detailed in

section 4.4.2) revealed areas (see sections 7.4.1 & 7.4.2) for future developments.

7.4.1 Real time performance of the redesigned hardware.

The sensor unit has been redesigned to include, amongst other changes, a tribo-charge neutralising electrode to reduce the tribo charge noise interference on the pulse charge injection. When the whole system is set up, in the near future, the real time performance of this redesigned sensor unit needs to be fully tested and possibly further developed to reduce the internal sensor noise.

The current to voltage converter and buffers on the signal conditioner have also been redesigned. The signal conditioner has been structured to adapt to the new digital interface (see section 7.1.3).

When the system is fully installed, on-line performance tests should be carried out on the integrated system hardware to confirm correct operation and fully characterise the system.

7.4.2 Real time performance of the system software

The current software is capable of: fast data capture, controlling the charge injected into the powder flow, controlling the gains of the sensor channels and channel switching. Further development of the software would enable

real-time mass flow rate and flow regime analyses to be performed. Analyses of experimental results would verify improved accuracies. Furthermore, the system has to be operated on-line for full commercial utilisation. Research work (previous and current) has been operated off-line.

7.4.3 Further studies of the powder flow materials

The material used in the current study was epoxy powder. A fresh epoxy produced significant difference in the sensor readings. A continuously re-used epoxy produced a 'reduced' performance due to changes in average particle size. The instrument must be able to cope with any likely changes in materials used for powder coating. Hence, there is the need to consider the state of the materials used in the analyses of the results.

In addition, the response of other powder materials needs to be investigated. An alternative material might provide enhanced chemical, mechanical and electrical qualities more suitable to this application.

7.5 Control strategies

Part of the current system will be used as a base to implement a future control system.

Currently, the digital interface performs many functions. A

future control system could be addressed from the latch on this interface. Stepper motors linked to this could be programmed to mechanically control the primary and secondary pneumatic air valve settings, which in effect controls the mass flow rates. These settings are currently controlled manually.

Improved measurement accuracy opens up the possibility of on-line control of the powder mass flow rates. This is a difficult task because of the fast response and stability considerations of a closed loop control system.

TABLE OF CONTENTS FOR APPENDICES

Contents

1.0 Appendix A (dc assessment of Current to Voltage converter and buffer) A1.0

1.1 Introduction A1.0

1.2 Detailed explanation of test 1 A1.1

1.2.1 dc response of the Silicon protection diode A1.2

1.2.2 dc response of the zener protection diode A1.2

1.2.3 dc response of the Schottky protection diode compared with the silicon. A1.2

1.3 Detailed explanation of test 2 A1.3

1.3.1 dc response of the Current to Voltage converter, with the transmit buffer gain set to unity. A1.4

1.3.2 dc response of the Current to Voltage converter, with the transmit buffer set to gain 10. A1.4

1.3.3 dc response of the Current to Voltage converter, with the transmit buffer set to gain 100. A1.5

1.3.4 dc response of the Current to Voltage converter, with transmit and receive buffers set to unity. A1.5

1.3.5 dc response of the Current to Voltage converter, with the transmit and receive buffers gain set to 10. A1.6

1.3.6 dc response of the Current to Voltage converter, with the transmit and receive buffers gain set to 100. A1.6

2.0 Appendix B (software listings) B1.1

2.1 A/D program for fast datacapture and processing B1.1

2.2 Pc host user interface program B1.16

2.3 Data storage and message passing program B1.36

2.4 Signal peak detection and message passing program B1.41

2.5 Data transfer program B1.45

2.6 The Protocol file B1.49

2.7 The Configuration file B1.50

2.8 Occam 2 product compiler B1.53

3.0 Appendix C (Detailed structure of Sensor unit) C1.1

3.1 Complete sensor unit assembly C1.1

3.2 Area view of the sensor unit C1.1

3.3 The charge injection unit C1.2

3.4 The side view of the injection unit C1.2

3.5 The hypordermic needle C1.2

3.6 The discharging electrodes C1.3

LIST OF FIGURES FOR APPENDICES

Appendix A

Figure A1.0	Silicon protection diode characteristics	A1.7
Figure A1.1	Zener protection diode characteristics	A1.7
Figure A1.2	Comparative plots of Schottky and Silicon diode protection characteristics	A1.8
Figure A1.3	dc response of C-V on transmit buffer load with unity gain	A1.9
Figure A1.4	dc response of C-V on transmit buffer with gain of 10	A1.9
Figure A1.5	dc response of C-V on transmit buffer with gain of 100	A1.10
Figure A1.6	C-V on transmit and receive buffer load [unity gain] dc response	A1.10

Appendix C

Figure C1.1	The sensor unit complete assembly	C1.4
Figure C1.2	Area view of the sensor unit	C1.5
Figure C1.3	The Charge injection unit	C1.6
Figure C1.4	The injection unit (side view)	C1.6
Figure C1.5	The hypodermic needle	C1.7
Figure C1.6	The discharging electrodes	C1.8

1.0 APPENDIX A

1.1 Introduction

In chapter 4, the hardware design of the system was discussed. The function of the current to voltage converter, in the system, was explained. This appendix describes the dc performance analyses tests carried out on the current to voltage converter and buffer circuits.

The aim of the tests was to investigate the linear dc response of the current to voltage converter (C-V) and the buffer circuits with or without protection diodes.

The following are the brief outlines of the tests.

1. **Test 1**

   Investigation of the relative performances of the protection diodes (silicon, zener and schoktty) in the C-V circuit.

2. **Test 2**

   Investigation of the relative dc response performances, when the C-V circuit (without protection diodes) is on load (buffer at the output)

The detailed explanations of these tests are provided below.

1.2 Detailed explanation of test 1.

These tests (the relative performances of the diodes) were carried out to determine which of the diodes is more appropriate for use as part of a protection circuit for the current to voltage converter. The significance of these diodes as protective devices have already been explained in the review chapter (chapter 3). The diode protects the circuit against accidental upsurge of the powder flow currents. These tests were aimed at determining the relative turn on points (forward conducting voltages) of three diodes (silicon, zener and schottky), when used as protection circuits against upsurge of powder flow sensor currents. A relatively low turn on point will be essential to avoid damage to the current to voltage converter circuit and possibly other parts of the signal conditioner.

To simulate the powder flow input sensor current, a power supply unit was used to supply voltages (ranging from low to high values) across a 100 Mohm resistor. The current input could be derived from these two parameters (voltage input and source resistance). The output voltage response across a 100K resistor, on the C-V, was monitored. The investigations carried out and results obtained from the various diodes will

now be explained. Figures 4.6 (chapter 4) contains the schematic circuit diagram of the C-V.

1.2.1 dc response of the silicon protection diode

Figure A1.0 is the graph drawn from the experimental data, using the silicon diode. The silicon diode has a good turn on point at about 2.2 volts, with a circuit offset voltage of 2.4 mV (marked on the graph). As a result, the circuit is protected against any upsurge of sensor current which causes a voltage drop above 2.2 volts.

1.2.2 dc response of the zener protection diode

Figure A1.1 is the graph drawn from the experimental data, using the zener diode. The response obtained is identical to that of the silicon diode and hence it is also capable of protecting the circuit against voltages above 2.4V.

1.2.3 dc response of the schottky protection diode compared with the silicon.

Figure A1.2 is the graphical plot of the Schottky diode characteristics compared with silicon. From the graph the turn on point for the schottky diode is quite high (about

7.8 volts) and hence not suitable for this project application. However, the schottky diode protection circuit possesses very low circuit offset voltage. The silicon diode circuit is acceptable because of its low turn on point.

1.2.4 Conclusions

From these investigations silicon and zener diodes possess good turn on points and low offsets. The schottky diode possesses a much lower offset, but the high turn on point is a disadvantage to this application. The silicon or zener diode protection circuit is more suitable to this application.

1.3 Detailed explanation of test 2

From test 1, the response obtained from the current to voltage converter was quite encouraging. The response was essentially linear. This diode protection circuit would have been desirable but the performance of the current to voltage converter degraded when loaded with the instrumentation buffers. The circuit became unstable when loaded with buffers. The source of instability was traced to the diode protection circuits. When the diodes were removed, the oscillation or instability problem was then solved. A decision was taken in the current research work not to include the protection

diodes. An alternative protection circuit was discussed in the chapter 4 (hardware design). After removing the diode protection circuit, the linear response of the current to voltage converter (cascaded with the instrumentation output buffers) was investigated. These responses were observed with the buffer gain set at various values. Figure 4.7 (chapter 4) is the cascaded circuit of the buffer stages (transmit and receive). Brief explanations of the various investigations are provided below.

1.3.1 dc response of the C-V on load with the transmit buffer gain set to unity

Figure A1.3 is a graph of dc response of the C-V loaded

with the transmit buffer. The input current was simulated, as in test 1, by using a source resistance of 100Mohm. The gain of the buffer was set to unity.

A linear response was obtained with no offset problem from the circuit. A zero offset can be obtained by proper tuning of the offset trimmers on the buffers.

1.3.2 dc response of the C-V on load with the transmit buffer gain set to 10.

Figure A1.4 is the response from this circuit with

configuration essentially the same as that used in section 1.3.1. The gain of the buffer is set to 10. The dc response is again linear with a slight offset of 0.2 mV.

1.3.3 dc response of the C-V on load with the transmit buffer gain set to 100.

Figure A1.5 is the response from this circuit with same configuration as that of section 1.3.1. The gain of the buffer is now set to 100.

The dc response is linear with a slight offset of 0.4 mV.

1.3.4 dc response of the C-V on load with the transmit and receive buffers and gain set to unity.

Figure A1.6 is the response from this circuit (C-V loaded with the transmit and receive buffers. The circuit gain was set to unity.

The dc response is linear and the offset is lower than that from section 1.3.1 circuit configuration. However there is slight delay in the circuit turn on point.

1.3.5 dc response of the C-V on load with the transmit and receive buffers and gain set to 10.

Figure A1.7 is the response from this circuit (circuit configuration same as that of section 1.3.4). The circuit gain was set to 10.

The dc response is linear with very low circuit offset voltage.

1.3.6 dc response of the C-V on load with the transmit and receive buffers and gain set to 100.

Figure A1.8 is the response from this circuit (circuit configuration same as that of section 1.3.4). The circuit gain was set to 100.

The dc response is linear with low circuit offset voltage.

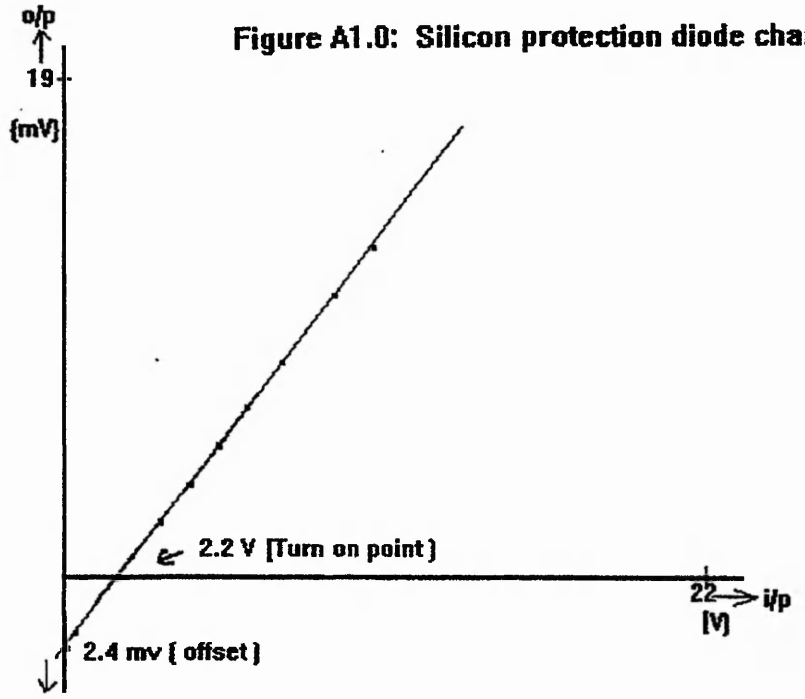


Figure A1.0: Silicon protection diode characteristics

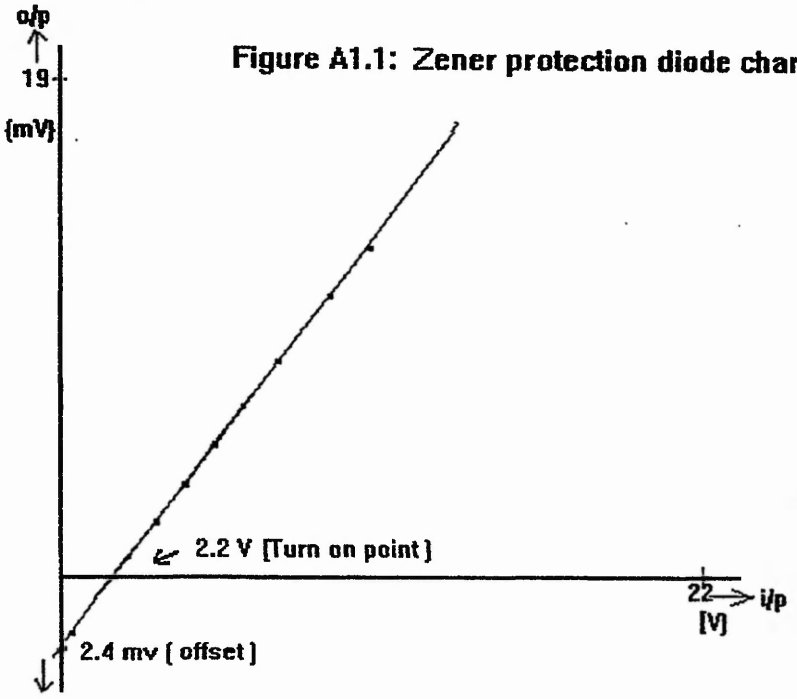


Figure A1.1: Zener protection diode characteristics

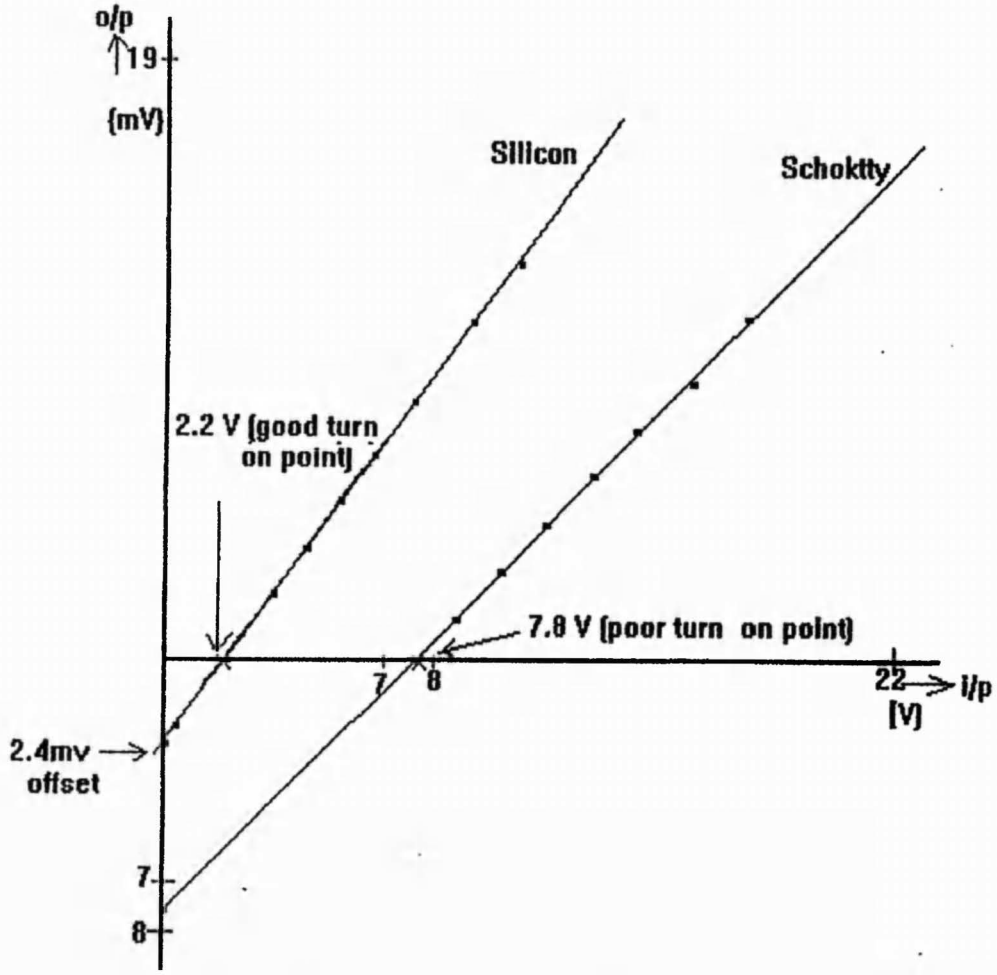


Figure A1.2: Comparative plots of Schottky and Silicon diode protection charcteristics

Figure A1.3: dc response of Current to Voltage converter on transmit buffer load with unity gain

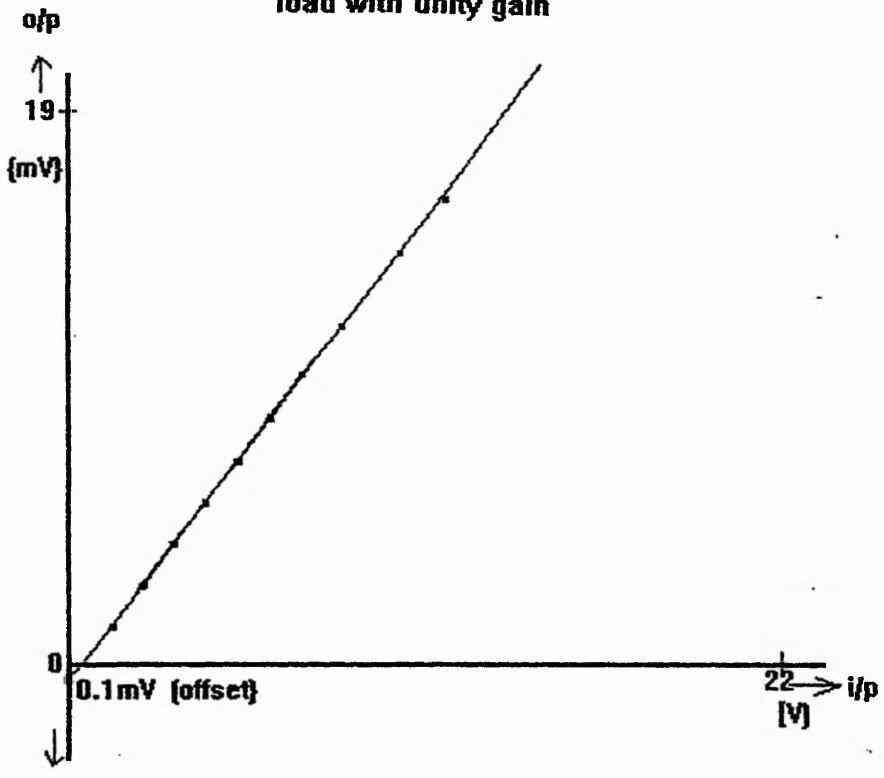


Figure A1.4: dc response of Current to Voltage converter on transmit buffer load with gain 10

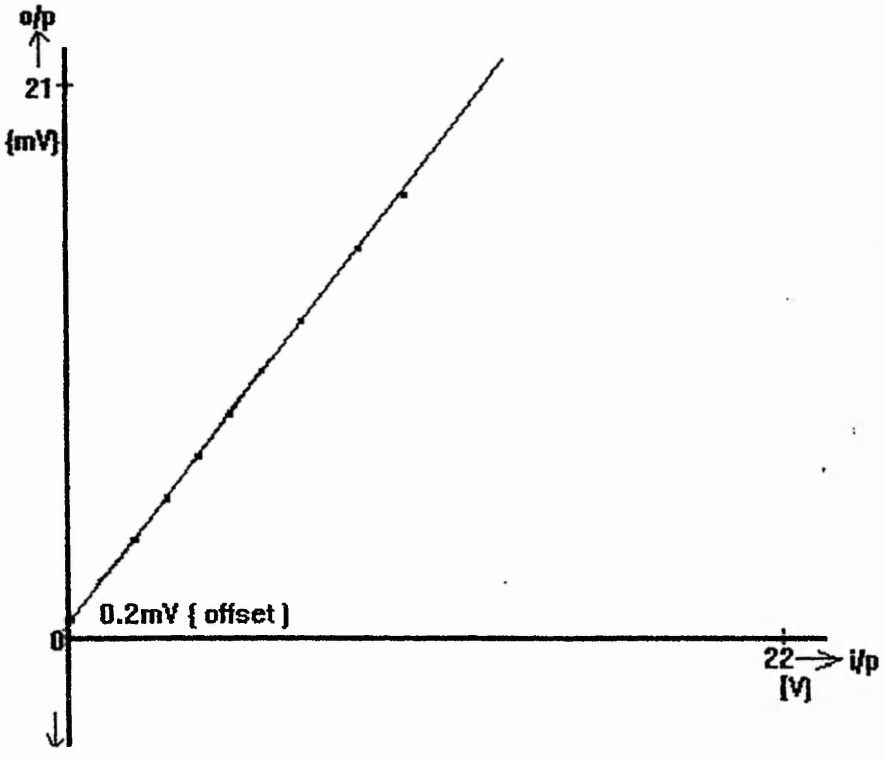


Figure A1.5: dc response of Current to Voltage converter on transmit buffer load with gain 100

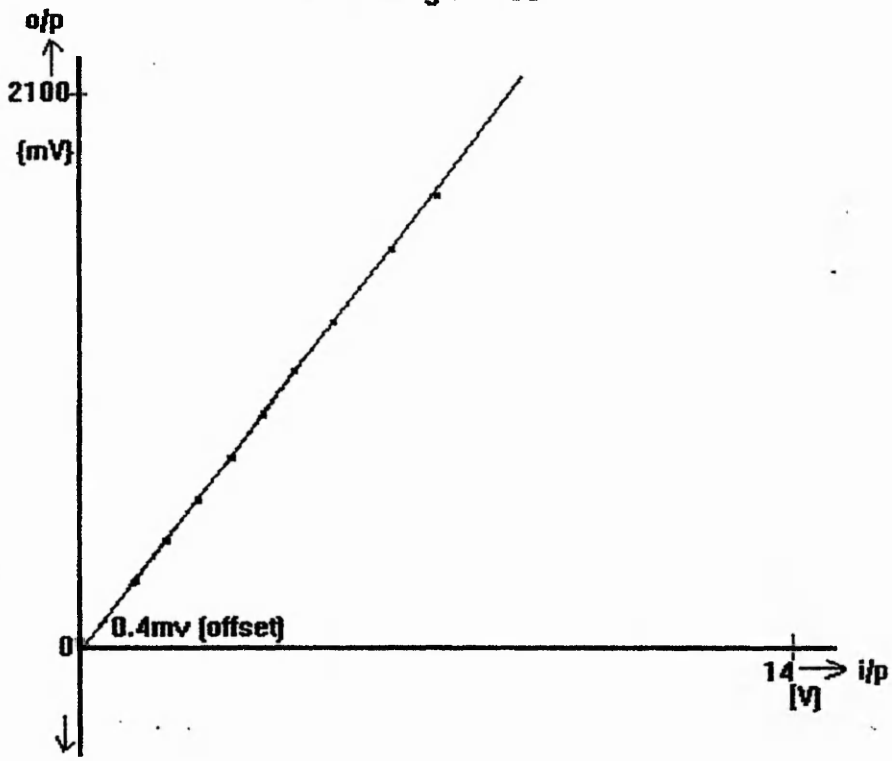
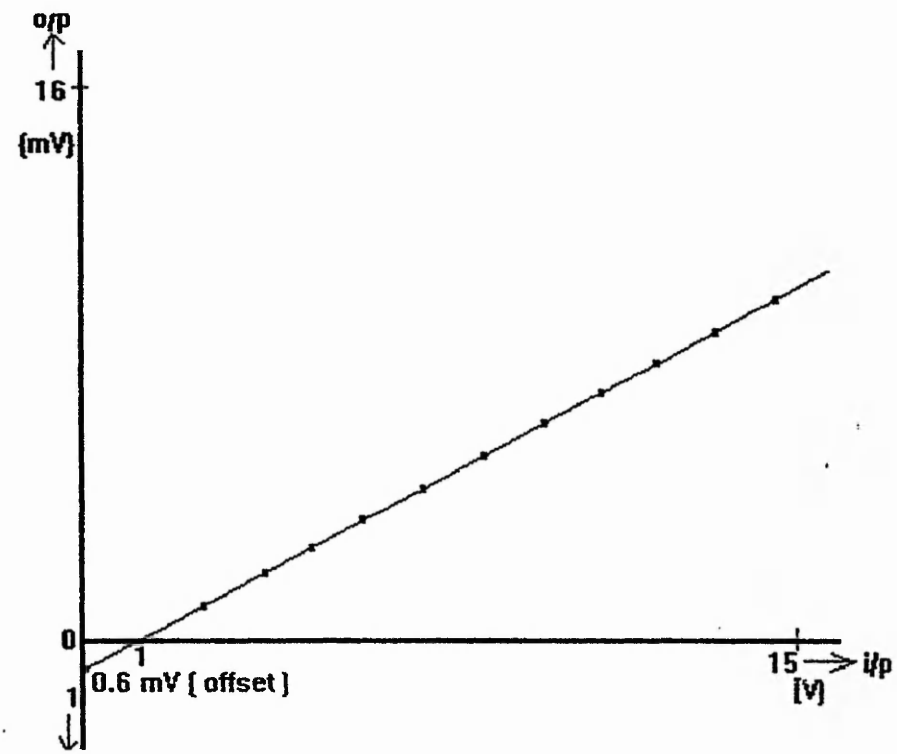


Figure A1.6: dc response of Current to Voltage converter on transmit and receive buffer load with unity gain



2.0 APPENDIX B

2.1 A/D program for fast data capture & preprocessing

```
--#COMMENT "This is the data acquisition & preprocessing"
--#COMMENT  "Software"

--#COMMENT "HT can be  pulsed at various injection periods."
--#COMMENT "At 5 microsec. sampling period 4000 samples"
--#COMMENT "can be collected in 20 ms"

--#COMMENT "With 400khz (2.5us sampling period) sampling"
--#COMMENT "frequency, 4000 samples will be captured"
--#COMMENT "in 10ms"

--#COMMENT "High sampling rate is required in order to track"

--#COMMENT "the essential features from the sensor waveform"
--}}}

#OPTION "A"

#INCLUDE "protoco.inc"

PROC atod.preprocess     ( CHAN OF ATOD.CHANS array.out.atod0,

                           CHAN OF ATOD.CHANS array.in.atod0,

                          CHAN OF ATOD.CHANS array.in.atod1,

                         CHAN OF ATOD.CHANS array.in.atod2 )

  --#COMMENT "Hardware machine map addresses translated to"
  --#COMMENT "occam map addresses"

  VAL occam.addr0 IS (#00007801 >< #80000000 ) >> 2:

  VAL occam.addr1 IS (#00007802 >< #80000000 ) >> 2:

  VAL occam.addr2 IS (#00007803 >< #80000000 ) >> 2:

  VAL occam.addr3 IS (#00007804 >< #80000000 ) >> 2:


 --#COMMENT "Placing variables on the addresses"

  INT signal.gainset:

  INT atod.int:

  INT read.adc:

  INT ht.control:
```

```
PLACE signal.gainset  AT occam.addr0:
PLACE atod.int        AT occam.addr1:
PLACE read.adc        AT occam.addr2:
PLACE ht.control      AT occam.addr3:
--#COMMENT "Retype int to byte for hardware read"
[]BYTE adc.b RETYPES read.adc:
adcbuffer   IS    adc.b[0]:
[]BYTE signal.b retype signal.gainset:
setlatch    IS    signal.b[0]:
[]BYTE ht.b retype ht.control:
ht          IS    ht.b[0]:
--COMMENT "On & off values for HT pulses"
VAL BYTE data1 IS #01(BYTE):
VAL BYTE data2 IS #00(BYTE):
--#COMMENT "Declare variables"
TIMER   clock:
TIMER   clock1:
INT     timenow:
INT     tim.atod:
--#COMMENT "Message passing variables"
BYTE     byte.char:
INT      number:
[2]INT   pulsewidth:
INT      nsamples:
[4]BYTE  latch:
--#COMMENT "Clock timing"
VAL tick IS 1:
VAL pulsetime IS 10000/tick:
```

```
VAL INT processor5 IS 5:
--#COMMENT "Input data from host Pc"
WHILE TRUE
  SEQ
    array.out.atod0 ? CASE
      chanout.4ints.4byte; byte.char; number; pulsewidth;
      nsamples; latch
        --#COMMENT "The local procedures"
        SEQ
          --{{{  PROC h.t. pulses  --Testing HT no. of
          cycles
          --#COMMENT "Pulsing HT"
          PROC h.t.control(INT number)
            SEQ n= 0 FOR number
              SEQ
                --#COMMENT  "Pulse on time"
                clock ? timenow    -- PULSE ON TIME
                ht := data1    -- latchset for off
                clock ? AFTER timenow PLUS pulsewidth[0]
                --#COMMENT "Pulse off time "
                clock ? timenow
                ht := data2
                clock ? AFTER timenow PLUS pulsewidth[1]
                --#COMMENT "CAUSEERROR() is a predefined"
                --#COMMENT "occam procedure for debugging"
                --CAUSEERROR()
          :
          --}}}
          --{{{  PROC h.t. pulse
          --#COMMENT "Proc h.t. pulses, more than once,"
          --#COMMENT "the high tension voltage supply"
          --#COMMENT "connected to the hypodermic"
          --#COMMENT "needle of sensor unit"
          --#COMMENT "The h.t. is linked to latch"
          PROC h.t.pulse()
                                                    --B1.3
```

```
    SEQ
      --{{{  pulseon time

      clock ? timenow

      ht := data1

      clock ? AFTER timenow PLUS ((pulsewidth[0]))

      --}}}

      --{{{  pulseoff time

      clock ? timenow

      ht := data2

      clock ? AFTER timenow PLUS ((pulsewidth[1]))

      --}}}
:

--}}}

--{{{  PROC input.data

--#COMMENT "Proc collect: serially collects data"
--#COMMENT "from the atod by slicing the samples"
--#COMMENT "from each sensor channel into"
--#COMMENT " arrayslice of 16 or higher"

--#COMMENT "At every channel switching, some"
--#COMMENT "settling time delay is allowed before"
--#COMMENT "reading atod "

--#COMMENT "Mux switching is done on the latch"

--#COMMENT "Opening out loops in arrayslice of 16"
--#COMMENT "has a knock on effect on the"
--#COMMENT "transputer performance"

PROC input.data ( []BYTE data)

  VAL delay IS 0:
  INT temp:

 --#COMMENT "Slice data array into"
 --#COMMENT "buffers (datafor0-3), If"
 --#COMMENT "4 sensors are used "

 --#COMMENT "3 buffers are currently used"

  []BYTE datafor0  IS [data FROM 0 FOR
    arrayslice]:
```

```
[]BYTE datafor1  IS [data FROM arrayslice FOR
  arrayslice]:

[]BYTE datafor2  IS [data FROM (2*arrayslice)
  FOR arrayslice]:

--[]INT datafor3  IS [data FROM (3*arrayslice)
--FOR arrayslice]:

SEQ

  setlatch := #00(BYTE)

  SEQ i= 0 FOR arrayslice

    SEQ

      --{{{  atod read of sensor 2

      SEQ

        setlatch := latch[2]   -- mux chan1 with
                               -- desired gain

        SEQ j= 0 FOR delay

          temp := i

        datafor2[i] := adcbuffer

      --}}}

      --{{{F ZX -- Filed for debugging purposes
      --:::F ZX
      --}}}

      --{{{  atod read of sensor 0

      --#COMMENT "Set latch, delay & read"
      --#COMMENT "sensor1 "

      setlatch := latch[0]

      --clock1 ? AFTER tim.atod
      --tim.atod := tim.atod PLUS delay

      SEQ j= 0 FOR delay
        temp := i

      datafor0[i] := adcbuffer  --i + 1
                                --adcbuffer

      --}}}
```

```
            --{{{  atod read of sensor 1

            --#COMMENT "Set latch, delay & read"
            --#COMMENT " sensor2"

            setlatch := latch[1]    -- mux chan1 with
                                    -- desired gain


            SEQ j= 0 FOR delay
              temp := i

            datafor1[i] := adcbuffer  --i + 2
                                      --adcbuffer

            --}}}

            --{{{  atod read of sensor 2

            --#COMMENT "Set latch,delay & read"
            --#COMMENT "sensor3"

            setlatch := latch[2]    -- mux chan1 with
                                 -- desired gain

            SEQ j= 0 FOR delay
              temp := i

            datafor2[i] := adcbuffer  --i + 3
                                      --adcbuffer
            --}}}

:
--}}}

--{{{  PROC sendout DATA.

--#COMMENT "Proc Send: parallel data out in"
--#COMMENT "sliced arrays of 16 or"
--#COMMENT "higher. The slices are"
--#COMMENT "obtained from one channel "
--#COMMENT "to another. The slicing goes"
--#COMMENT "on until the number of samples "
--#COMMENT "desired from each channel is "
--#COMMENT "is exhausted "

PROC send.data ([]BYTE data)

  SEQ

    []BYTE datafor0  IS [data FROM 0 FOR
       arrayslice]:
   []BYTE datafor1  IS [data FROM arrayslice FOR
       arrayslice]:
```

```
    []BYTE datafor2  IS [data FROM (2*arrayslice)
       FOR arrayslice]:

    --[]BYTE datafor3  IS [data FROM
    --(3*arrayslice) FOR arrayslice]:

    PAR

      array.in.atod0 ! chanout.byte.array0;
                       datafor0

      array.in.atod1 ! chanout.byte.array1;
                       datafor1

      array.in.atod2 ! chanout.byte.array2;
                       datafor2

:
--}}}

--{{{  PROC input.send.data 1 (fast data collect
         & send for 1 H.T. cycle

        --#COMMENT "Proc distribute decouples "
        --#COMMENT "the computation process"
        --#COMMENT "activites of collect and"
        --#COMMENT "senddata from the link"
        --#COMMENT "communications. As a result"
        --#COMMENT "the link throughput  "
        --#COMMENT "performance is maximised "

PROC input.send.data1 ( VAL INT nsamples )

   --#COMMENT "datasize arrays for "
   --#COMMENT "collect & "
   --#COMMENT "send on four channels"

  [4*arrayslice] BYTE A,B:
  SEQ

    --#COMMENT "Start up data collect process "

    --#COMMENT "Collects & store data in"
    --#COMMENT "buffer A"

    input.data(A)

    --#COMMENT " Now Collect more data & send"
    --#COMMENT " previous in parallel"

    PAR

      input.data(B)

      send.data (A)
```

```
      --#COMMENT "Then collect & send the leftover"
      --#COMMENT "samples in parallel"

      SEQ

        --#COMMENT "Use each buffer to collect &"
        --#COMMENT "send leftover "
        --#COMMENT "samples divided by 2*arrayslice"

        SEQ i = 0 FOR  (((nsamples - (2*arrayslice))
                          >> 1 ) >> 4)

          SEQ

            --#COMMENT "The round-bobbin data"
            --#COMMENT "collect and send process"

            PAR

              send.data (B)

              input.data(A)

            PAR

              send.data (A)

              input.data(B)

        send.data(B) --Final buffer send
:

--}}}

--{{{  PROC input.send.data 2 (fast data collect
       & send for n H.T.cycles


--#COMMENT "Proc distribute decouples the"
--#COMMENT "computation process"
--#COMMENT "activites of collect and send"
--#COMMENT "data fromthe link"
--#COMMENT "The link throughput"
--#COMMENT "performance is maximised"


PROC input.send.data2 ( VAL INT nsamples )

  --#COMMENT "datasize collect & send"

  --#COMMENT "on the four channels"

  [4*arrayslice] BYTE A,B:
```

```
SEQ

  --#COMMENT "Number(H.T.cycles desired)"

  SEQ j= 0 FOR number

    SEQ

      --#COMMENT " Pulse the H.T. "

      h.t.pulse()

      --#COMMENT "Collect data start up"
      --#COMMENT "process "

      input.data(A)

      --#COMMENT " Now collect more data & send"
      --#COMMENT " previous in parallel"

      PAR

        input.data(B)

        send.data (A)

      --#COMMENT " Then collect & send the"
      --#COMMENT "leftover samples in par."

      SEQ

        --#COMMENT "Use each buffer to collect"
        --#COMMENT "& send leftover "

        --#COMMENT "samples divided by
        --#COMMENT "2*arrayslice "

        SEQ i = 0 FOR (((nsamples -
                  (2*arrayslice)) >> 1 ) >> 4)
          SEQ

            PAR
              send.data (B)
              input.data(A)
             PAR

              send.data (A)

              input.data(B)

        send.data(B) --Final buffer send
```

```
--}}}

--{{{  PROC out powderflow status

--#COMMENT "Proc Print is used in"
--#COMMENT "sending out injection waveform"

--#COMMENT "status"
--#COMMENT "for screen display on the Pc."
--#COMMENT "Also used for message "
--#COMMENT "communications to the array or"
--#COMMENT "controller procs "
--#COMMENT "Message tags are decoded in the"
--#COMMENT "destination procs."

PROC print (VAL INT flow.status)

  array.in.atod0  ! printer1; flow.status

:

--}}}

--{{{  PROC dummy

--#COMMENT "PROC dummy is a timeout program,"
--#COMMENT "which runs in parallel with"
--#COMMENT "a process to be debugged "

PROC dummy ()
  TIMER clock2:
  INT tim.c:
  SEQ
    WHILE TRUE
      SEQ
        clock2 ? AFTER tim.c
        tim.c := tim.c PLUS #80000
:
--}}}

--{{{  Program track injection

--#COMMENT "PROC detects injection by statistcal"
--#COMMENT "decision thresholding "
--#COMMENT "Proc trackinjection: analyses the"
--#COMMENT "injection waveform "

--{{{  track injection (status.report included)

PROC  track.injection  (  BYTE  byte.char,  INT
                                nsamples)

  SEQ
```

```
--{{{  Boolean variables

BOOL signal.underflow, signal.overflow,
                signal.injection.found:

BOOL injection.timing:

BOOL spike:

--}}}

#COMMENT "Signal analyses "
#COMMENT "interger variables

--{{{  INT declarations

INT datasample:

INT analyse.samples:

INT change:

INT amplitude1:

INT amplitude2:

INT amplitude3:

INT lastdatasample:

INT sum.sample.change:

INT trend:

INT process.time.start:

INT sample.sum:

INT injectionsamples:

INT injection.searchtime.start:

INT injection.searchtime:

INT process.time.finish:

--}}}

--#COMMENT "Parameter definition"
--#COMMENT "for statistical analyses of the"
--#COMMENT "flow"

--{{{  Threshold values

VAL  threshold.level IS 8:
```

```
VAL up.sum.sample.change IS (threshold.level
    + (threshold.level >>2)):  --1.25 * w.l.

VAL injection.positive IS   2 *
          up.sum.sample.change:

VAL down.sum.sample.change IS
              threshold.level:

VAL injection.negative IS   (2 *
        down.sum.sample.change):

VAL sum.constant.sample.change IS 0:

--}}}

--#COMMENT "Data change storage"

[4800]INT store.change.in.data:

--#COMMENT "Extract out byte field from"
--#COMMENT "integer variable"

[]BYTE Bdatasample RETYPES datasample:
datasample.b  IS  Bdatasample[0]:


--#COMMENT "PROC out status report of the"
--#COMMENT "powder flow to Host"

SEQ

  PROC statusreport (BYTE byte.char)

    --{{{  local procedure to detect status of
           signal

    VAL INT n1 IS 1:     --  signal.underflow

    VAL INT n2 IS 2:     --  signal.overflow

    VAL INT n3 IS 3:     --  up.sum.change

    VAL INT n4 IS 4:     --  down.sum.change

    VAL INT n5 IS 5:     --  constant trend

    VAL INT n6 IS 6:    --  injection.positive

    VAL INT n7 IS 7:    --  injection.negative

    VAL INT n8 IS 8:     --   ???

    VAL INT n9 IS 9:     --   no injection
```

```
--#COMMENT "Condition monitory of "
--#COMMENT "the flow using the status"
--#COMMENT "tags"

SEQ

  IF

    sum.sample.change >
         up.sum.sample.change

      SEQ

        trend := up.sum.sample.change

        array.in.atod0 ! printer1; n3

            sum.sample.change    >
          down.sum.sample.change

      SEQ

        trend := down.sum.sample.change

        array.in.atod0 ! printer1; n4

    sum.sample.change >
          sum.constant.sample.change

      SEQ

       trend :=sum.constant.sample.change

        array.in.atod0 ! printer1; n5

    sum.sample.change>injection.positive

      SEQ

        trend := injection.positive

        array.in.atod0 ! printer1; n6

    sum.sample.change>injection.negative

      SEQ

        trend := injection.negative

        array.in.atod0 ! printer1; n7

    TRUE
      array.in.atod0 ! printer1; n8
--}}}

:
                                        --B1.13
```

```
SEQ
  --{{{  initialisation
  clock ? process.time.start   --clock time
          --for process starts
  ht := data1     --Start HT
  setlatch := latch[0]  --Set gain & channel
  setlatch := latch[0]  -- some delay
                        -- introduced
  datasample := 0         --initial data
  lastdatasample:=datasample
  #COMMENT " Initial AtoD read "
  datasample := adcbuffer/\#FF(INT)
  lastdatasample := datasample
  #COMMENT "start summing samples"
  sample.sum := datasample
  #COMMENT "data change initialise to zero"
  sum.sample.change := 0
  --}}}
  SEQ i = 0 FOR nsamples
    SEQ
      #COMMENT "Request for injection"
      setlatch := latch[0] --setup channel
      datasample := adcbuffer/\#FF(INT)
      change := datasample -lastdatasample
      sum.sample.change :=
             (sum.sample.change + change )
      store.change.in.data[i]  :=   change
      lastdatasample :=   datasample
      sample.sum := sample.sum + datasample
```

```
          --#COMMENT "Call status subroutine"

          SEQ

            --CAUSEERROR()
            statusreport ( byte.char)

        --}}}
:
--}}}

--}}}

--#COMMENT "execute Atod program"

SEQ

  IF
    byte.char = 'B'

      SEQ
        PAR

          h.t.pulse()

          input.send.data1 (nsamples)

        --track.injection(byte.char, nsamples)

    byte.char = 'C'
      SEQ
        track.injection(byte.char, nsamples)
    TRUE
      SKIP
```

2.2 Pc host user interface program

```
--#COMMENT "Software for continuous read and display of "
--#COMMENT "data from sensors (1,2 & 3)"

--#COMMENT "Declare i/o library"
--#COMMENT "protocol file"

#INCLUDE  "hostio.inc"
#INCLUDE  "protoco.inc"

--#COMMENT " Server  &  Protocol  channel declarations "


PROC pc.driver.code ( CHAN OF SP fs, ts,
                      CHAN OF ATOD.CHANS tc, fc)

  --#COMMENT "Declare standard i/o & string libraries"

  #USE "hostio.lib"
  #USE "string.lib"


  --{{{  Declare variables (Bools, bytes, array, int)

  BOOL going:

  [2]INT pulsewidth:

  INT number:

  INT command:

  [4]BYTE latch:

  [arrayslice]BYTE datastream:

  [sensor.no.samples]BYTE databuffer:

  BYTE key, byte.char:

  BYTE result:

  INT nsamples:

  [10]INT gainint:

  BOOL running:

  --}}}
```

```
INT count:

SEQ

  --{{{  PROC lookup table   --LATCH SETUP FOR CHANS & GAINS

  --#COMMENT "Lookup table for setting the sensor channels"
  --#COMMENT "& respective channel gains on the latch"

  PROC table ()

    SEQ

      --{{{  latch lookup table

      so.write.string (fs, ts, " *c*n")
      so.write.string (fs, ts, "LOOKUP TABLE(chan/gain)
                                                *c*n")
      so.write.string (fs, ts, "..................*c*n")
      so.write.string (fs, ts, "append hex symbol to gain
                                          request.. *c*n")
      so.write.string (fs, ts, "*c*n")
      so.write.string (fs, ts, "G1 G2 G4 G8 G16 G32 G64 G128
                                             *c*n")
      so.write.string (fs, ts, "*c*n")
      so.write.string (fs, ts, "CH0 08 18 28 38 48 58 68
                                              78*c*n")
      so.write.string (fs, ts, "CH1 09 19 29 39 49 59 69
                                              79*c*n")
      so.write.string (fs, ts, "CH2 0A 1A 2A 3A 4A 5A 6A
                                              7A*c*n")
      so.write.string (fs, ts, "CH3 0B 1B 2B 3B 4B 5B 6B
                                              7B*c*n")
      so.write.string (fs, ts, "CH4 0C 1C 2C 3C 4C 5C 6C
                                              7C*c*n")
      so.write.string (fs, ts, "CH5 0D 1D 2D 3D 4D 5D 6D
                                              7D*c*n")
      so.write.string (fs, ts, "CH6 0E 1E 2E 3E 4E 5E 6E
                                              7E*c*n")
      so.write.string (fs, ts, "CH7 0F 1F 2F 3F 4F 5F 6F
                                              7F*c*n")
      so.write.string (fs, ts, " *c*n")

      --}}}

  :
  --}}}

  --{{{  PROC pc.out.data

  --#COMMENT "The keyboard.screen Process"

  PROC pc.out.data ()

    [4]BYTE latch:

                                                    --B1.17
```

```
SEQ

  so.write.nl (fs, ts)
  so.write.string.nl (fs, ts, " Hit key for requests ")
  so.write.nl ( fs, ts )
  so.getkey (fs, ts, key, result)
  IF
    result = spr.ok

      SEQ

        --{{{  host dependent constructs( keyboard read
               screen display)

        so.write.string.nl (fs, ts, " New Pulse Timing
                                     requests ?.. y/n")

        so.getkey ( fs,ts, key, result)

        SEQ

          IF

            (key = 'n' ) OR ( key = 'N')
              SEQ
                SKIP

            (key = 'y') OR ( key = 'Y')
              SEQ
                --{{{ (keyboard read & ht lookup table
                       display)

                so.write.nl (fs, ts)
                so.write.nl (fs, ts)
                so.write.string.nl (fs, ts, " no of ht
                                    cycles ? .. *c*n")
                so.read.echo.int (fs, ts, number, going)
                so.write.nl (fs,ts)
                so.write.string (fs, ts, "*c*n")

                --}}}

                --{{{  set latch parameters ( ht pulses,
                            gains, flowsamplses)

                --{{{ (keyboard read & display of pulse
                                          on time)

                so.write.nl (fs, ts)
                so.write.nl (fs, ts)
                so.write.string.nl (fs, ts, "
                     pulse.on.time (ms) ? .. *c*n")
                so.read.echo.int (fs, ts, pulsewidth[0],
                                          going)
```

```
so.write.nl (fs,ts)
so.write.string (fs, ts, "*c*n")

--}}}
--{{{ (keyboard read & display of pulse
        off time)

so.write.string.nl    (fs,    ts,    "
   pulse.off.time (ms) ? .. *c*n")
so.read.echo.int (fs, ts, pulsewidth[1],
   going)
so.write.nl (fs,ts)
so.write.string (fs, ts, "*c*n")

--}}}
--{{{ (keyboard read & display of pulse
      samples)

so.write.string.nl ((fs, ts, " Request
  160n flowsmples ( n = 1, 2 ..30 )
   *c*n"))

so.read.echo.int  (fs,  ts,  nsamples,
                          going)
so.write.nl (fs,ts)
so.write.string (fs, ts, "*c*n")

--}}}
--{{{(keyboard read & display of 1st
   latchset (hex))

so.write.string.nl  (fs,  ts,  "  enter
                  latch0.. *c*n")
 so.read.echo.hex.int    (fs,    ts,
              gainint[0], going)
so.write.nl (fs,ts)
so.write.string (fs, ts, "*c*n")

--}}}
--{{{  (keyboard read & display of 2nd
        latchset (hex))

so.write.string.nl  (fs,  ts,  "  enter
                  latch1 .. *c*n")
 so.read.echo.hex.int    (fs,    ts,
                  gainint[1], going)
so.write.nl (fs,ts)
so.write.string (fs, ts, "*c*n")

--}}}
--{{{  (keyboard read & display of 3rd
      latchset (hex))

so.write.string.nl (fs, ts, " enter
                  latch2 .. *c*n")
```

--B1.19

```
                so.read.echo.hex.int    (fs,    ts,
                            gainint[2], going)
              so.write.nl (fs,ts)
              so.write.string (fs, ts, "*c*n")

              --}}}
              --{{{ (keyboard read & display of 4th
                                    latchset (hex))
              so.write.string.nl  (fs,  ts,  "  enter
                                 latch3 .. *c*n")
                so.read.echo.hex.int    (fs,    ts,
                                 gainint[3], going)
              so.write.nl (fs,ts)
              so.write.string (fs, ts, "*c*n")

              --}}}

              --}}}

              --#COMMENT "Type convert Hex data
              --#COMMENT " i/p to Int"

              SEQ

                SEQ j = 0 FOR 4

                  SEQ

                    latch[j] := BYTE (gainint[j])


              --#COMMENT "Parallel send & receive"
              --#COMMENT "data to/from other"
              --#COMMENT "processes. Without"
              --#COMMENT 'WHILE TRUE' at the"
              --#COMMENT "receive the process can "
              --#COMMENT "run discretely"
              --#COMMENT "The 'WHILE TRUE ' is "
              --#COMMENT " required for  "
              --#COMMENT "continous read & display of"
              --#COMMENT  "the AtoD data."

              SEQ

                tc ! chanout.4ints.4byte; byte.char;
                     number;  pulsewidth;  nsamples;
                      latch
          TRUE
            SKIP
      --}}}
    TRUE
      SKIP
:
--}}}
```

```
--{{{  PROC rec.status

--#COMMENT "PROC status displays powder flow"
--#COMMENT "status using various tags"
--#COMMENT "to report the various states"
--#COMMENT " of the powder flow as "


PROC  rec.status ()

  INT intn:

  SEQ

    --{{{  Declare injection parameter tags

    VAL n1 IS 1:  -- underflow
    VAL n2 IS 2:  -- overflow
    VAL n3 IS 3:  -- up.sum.change
    VAL n4 IS 4:  -- down.sum.change
    VAL n5 IS 5:  -- constant trend
    VAL n6 IS 6:  -- unknown
    VAL n7 IS 7:  -- ???
    VAL n8 IS 8:  -- injection.spike
    VAL n9 IS 9:  -- no spike

    --}}}

    SEQ

      --#COMMENT "Input the Status tag from the AtoD"

      fc ? CASE

        printer1; intn

          CASE intn

            --{{{ Various injection status flags

            n1
              so.write.string.nl (fs, ts, "warning! ..
                atod underflow")
            n2
              so.write.string.nl  (fs,  ts,  "warning!  ..
                atod overflow")
            n3
              so.write.string.nl (fs, ts, "warning! .. up
              sum of sample change")

            n4
              so.write.string.nl  (fs,  ts,  "warning!  ..
                down sum of sample change")
```

```
                n5
                  so.write.string.nl (fs, ts, "warning! .. sum
                      sample change constant")
                n6
                  so.write.string.nl (fs, ts, "warning!
                      ..Injection positive")
                n7
                  so.write.string.nl   (fs,   ts,   "warning!
                    ..Injection negative")
                n8
                  so.write.string.nl  (fs,  ts,  "warning!  ..
                    unknown trend ???")
                n9
                  so.write.string.nl (fs, ts, "warning! .. No
                    injection found")

                --}}}

                ELSE

                  SKIP

        so.write.nl (fs, ts)
        so.write.nl (fs, ts)
        so.write.nl (fs, ts)

:

--}}}

--{{{  PROC datacapture MENU  --Out parameters & receive
                                --data

--#COMMENT "Keyboard read & screen display"
--#COMMENT  "the respective data i/p & o/p"
--#COMMENT "Currently, processes are parallel run"
--#COMMENT " at low  priority Timers"
--#COMMENT "... for the sake of simulations"
--#COMMENT "and will be changed to PRI PAR"
--#COMMENT "required for the fast processing of real time
--#COMMENT "flow process "
--#COMMENT ".. PRI PAR means clock ticking"
--#COMMENT "at 1us in contrast to low "
--#COMMENT "priority PAR which ticks at 64us "

PROC menu ()
  SEQ
    so.write.nl (fs, ts)
    so.write.string.nl (fs, ts, "Hit key for requests ")
    so.write.nl ( fs, ts )
    so.getkey (fs, ts, key, result)
    IF
      result = spr.ok
```

```
SEQ
  so.write.string.nl (fs, ts, " New Pulse Timing
  requests ?.. y/n")
  so.write.nl (fs, ts )
  so.write.nl (fs, ts )
  so.write.nl (fs, ts )
  so.getkey ( fs,ts, key, result)
  SEQ
    IF
      (key = 'n' ) OR ( key = 'N')
        SEQ
          SKIP

      (key = 'y') OR ( key = 'Y')

        SEQ
          --{{{  keyboard read & ht lookup table
                 display

          so.write.nl (fs, ts)
          so.write.nl (fs, ts)
          so.write.string.nl (fs, ts, " no of ht
          cycles ? .. *c*n")
          so.read.echo.int (fs, ts, number, going)
          so.write.nl (fs,ts)
          so.write.string (fs, ts, "*c*n")

          --}}}

          --{{{  set latch parameters ( ht pulses,
                 gains, flowsamples)

          --{{{  keyboard read & display of pulse
                 on time

          so.write.nl (fs, ts)
          so.write.nl (fs, ts)
          so.write.string.nl (fs, ts, "
          pulse.on.time (ms) ? .. *c*n")
          so.read.echo.int (fs, ts, pulsewidth[0],
                            going)
          so.write.nl (fs,ts)
          so.write.string (fs, ts, "*c*n")

          --}}}
          --{{{  keyboard read & display of pulse
                 off time

          so.write.string.nl (fs, ts, "
           pulse.off.time (ms) ? .. *c*n")
          so.read.echo.int (fs, ts, pulsewidth[1],
           going)
          so.write.nl (fs,ts)
          so.write.string (fs, ts, "*c*n")
          --}}}
```

```
--{{{  keyboard read & display of pulse
       samples

so.write.string.nl (fs, ts, " Request
160n flowsmples ( n = 1, 2 ..30 )
*c*n")
so.read.echo.int (fs, ts, nsamples,
 going)
so.write.nl (fs,ts)
so.write.string (fs, ts, "*c*n")

--}}}
--{{{  keyboard read & display of 1st
       latchset (hex)

so.write.string.nl (fs, ts, " enter
latch0.. *c*n")
so.read.echo.hex.int (fs, ts,
gainint[0], going)
so.write.nl (fs,ts)
so.write.string (fs, ts, "*c*n")

--}}}
--{{{  keyboard read & display of 2nd
       latchset (hex)

so.write.string.nl (fs, ts, " enter
latch1 .. *c*n")
so.read.echo.hex.int (fs, ts,
gainint[1], going)
so.write.nl (fs,ts)
so.write.string (fs, ts, "*c*n")

--}}}
--{{{  keyboard read & display of 3rd
       latchset (hex)

so.write.string.nl (fs, ts, " enter
latch2 .. *c*n")
so.read.echo.hex.int (fs, ts,
 gainint[2], going)
so.write.nl (fs,ts)
so.write.string (fs, ts, "*c*n")

--}}}
--{{{  keyboard read & display of 4th
       latchset (hex)

so.write.string.nl (fs, ts, " enter
 latch3 .. *c*n")
so.read.echo.hex.int (fs, ts,
 gainint[3], going)
so.write.nl (fs,ts)
so.write.string (fs, ts, "*c*n")
--}}}
--}}}
```

```
--#COMMENT "Type convert Hex data "
--#COMMENT  "i/p to int"

SEQ

  SEQ j = 0 FOR 4

    SEQ

      latch[j] := BYTE (gainint[j])


--#COMMENT "Parallel send & receive data
--#COMMENT " to/fro other"
--#COMMENT "processes. Without 'WHILE"
--#COMMENT " TRUE' at the"
--#COMMENT "receive the process can be"
--#COMMENT "run discretely"
--#COMMENT "The 'WHILE TRUE ' is"
--#COMMENT "required for"
--#COMMENT "continous read & display of
--#COMMENT "the AtoD data."
--#COMMENT "It is currently commented"
--#COMMENT "out"

SEQ
  PAR
    SEQ
      --{{{  pc sendout set parameters

      tc ! chanout.4ints.4byte;
             byte.char; number;
          pulsewidth; nsamples; latch

      --}}}

    SEQ

      --WHILE TRUE
      SEQ
        --{{{  variable declarations

        BYTE result:
        INT  point,  length1,  length2,
        length3:
        INT32 id1, id2, id3:

        --}}}

        SEQ

          --#COMMENT "This F old2 is"
          --#COMMENT "currently not"
          --#COMMENT  "used & "
```

```
--#COMMENT "is only reads"
--#COMMENT "data"
--#COMMENT "to A drive "


--#COMMENT  "Store 4000"
--#COMMENT  "bytes in the C"
--#COMMENT  "drive"
--#COMMENT "& simultaneously"
--#COMMENT  "display 32 ints"
--#COMMENT"datarespectively"
--#COMMENT  "from the 3"
--#COMMENT  "powder"
--#COMMENT "flow Sensors on"
--#COMMENT "the screen "


--{{{  display 32 ints from
      sensor 1 & store C dri.

fc ? CASE

  chanbuffer2; databuffer

    SEQ
      so.write.string.nl(fs,
       ts, " Reading the
      sensor (1)  data .. " )
      so.write.nl ( fs, ts )

      SEQ
        --{{{   TYPE convert
              bytes to int32

        []INT32 slice RETYPES
        databuffer:

        --}}}

        SEQ i = 0 FOR 32


          --{{{  screen
             display ints
             ANDed 256
          SEQ

            so.write.int(fs,
            ts, INT(slice[i]
            /\#FF(INT32)),8)

          --}}}
        --{{{  rtt
        so.write.nl (fs, ts)
        so.write.nl (fs, ts)
```

```
            --{{{  open file
                 c:\store\flow1

          so.open(fs,ts,
             "c:\store\flow1",
           spt.binary, spm.output,
             id1, result)

            --}}}

            --{{{  write
                   datasamples

            so.write(fs, ts, id1,
                     databuffer,
                     length1)

            --}}}

            --{{{  close file

            so.close(fs, ts, id1,
            result)

            --}}}
            --}}}
--}}}

--{{{  display 32 ints from
       sensor 2 & store C dri.

fc ? CASE

  chanbuffer2; databuffer

    SEQ
      so.write.string.nl(fs,
        ts, " Reading the
        sensor (2) data.. " )
      so.write.nl ( fs, ts )

      SEQ
        --{{{   TYPE convert
                bytes to int32

        []INT32 slice RETYPES
        databuffer:

        --}}}

       SEQ i = 0 FOR 32

          --{{{screendisplay
               ints ANDed 256
```

```
            SEQ
              so.write.int( fs,
              ts, INT(slice[i]
              /\#FF(INT32)),8)
          --}}}
        --{{{  rtt
        so.write.nl (fs, ts)

        so.write.nl (fs, ts)

        --{{{     open  file
             c:\store\flow2

        so.open(fs, ts,
         "c:\store\flow1",
        s p t . b i n a r y ,
        spm.output,    id1,
        result)

        --}}}

        --{{{  write
             datasamples

        so.write(fs,ts,id1,
        databuffer, length1)

        --}}}

        --{{{  close file

        so.close(fs,ts,id1,
                  result)

        --}}}
        --}}}
--}}}

--{{{   display 32 ints from
       sensor 3 & store C
        dri.

fc ? CASE

  chanbuffer2; databuffer

    SEQ
      so.write.string.nl(fs,
        ts,  "  Reading  the
        Sensor (3) data.." )
      so.write.nl ( fs, ts )
```

```
SEQ
  --{{{  TYPE convert
         bytes to int32

  []INT32 slice RETYPES
       databuffer:

  --}}}


  SEQ i = 0 FOR 32

    --{{{        screen
       display    ints
       ANDed 256
    SEQ

      so.write.int(fs,
       ts,INT(slice[i]
       /\#FF(INT32)), 8)

    --}}}

  --{{{  rtt
  so.write.nl (fs, ts)

  so.write.nl (fs, ts)

  --{{{  open file
         c:\store\flow3

  so.open(fs, ts,
  "c:\store\flow1",
  spt.binary,
  spm.output,
  id1, result)

  --}}}

  --{{{  write
         datasamples

  so.write(fs, ts, id1,
    databuffer, length1)

  --}}}

  --{{{  close file

  so.close(fs, ts, id1,
           result)

  --}}}
  --}}}
```

```
                                --}}}

                                --rec.status()

                                --#COMMENT " This fold is"
                                --#COMMENT " not used "

                                --{{{F OLD3
                                --:::F OLDO
                                --}}}

                    --#COMMENT "TRUE guard necessary if"
                    --#COMMENT "keyboard variables are "
                    --#COMMENT "mistyped. However, for"
                    --#COMMENT "continous rescrolling "

                    --#COMMENT "of the flow menu the exit"
                    --#COMMENT "library should"

                    TRUE
                      SEQ
                        so.write.string.nl (fs, ts, "Wrong data
                         entry ! .. *c*n")

                        SKIP
                        --so.exit(fs, ts, sps.success)  --SKIP


          --#COMMENT "This TRUE guard redisplays the top"
          --#COMMENT "Hierachical Commands"
          --#COMMENT " menu & hence the SKIP after the guard"
          --#COMMENT "statement "

          TRUE
            SKIP

:
--}}}

--#COMMENT "The keyboard.screen Process"

PROC display.status ()


  [4]BYTE latch:

  --WHILE TRUE
  SEQ
    so.write.nl (fs, ts)
    so.write.string.nl (fs, ts, "Hit key for requests ")
    so.write.nl ( fs, ts )
    so.getkey (fs, ts, key, result)
    IF
      result = spr.ok
```

```
SEQ

  --{{{  host dependent constructs( keyboard readm
         screen display)

  so.write.string.nl (fs, ts, " New Pulse Timing
                       requests ?.. y/n")

  so.getkey ( fs,ts, key, result)

  SEQ
    IF
      (key = 'n' ) OR ( key = 'N')
        SEQ
          SKIP

      (key = 'y') OR ( key = 'Y')

        SEQ

          --{{{  keyboard read & ht lookup table
                 display

          so.write.nl (fs, ts)
          so.write.nl (fs, ts)
          so.write.string.nl (fs, ts, " no of ht
          cycles ? .. *c*n")
          so.read.echo.int (fs, ts, number, going)
          so.write.nl (fs,ts)
          so.write.string (fs, ts, "*c*n")

          --}}}

          --{{{  set latch parameters ( ht pulses,
                 gains, flowsamplses)

          --{{{  keyboard read & display of pulse
                 on time

          so.write.nl (fs, ts)
          so.write.nl (fs, ts)
          so.write.string.nl    (fs,    ts,    "
             pulse.on.time (ms) ? .. *c*n")
          so.read.echo.int (fs, ts, pulsewidth[0],
             going)
          so.write.nl (fs,ts)
          so.write.string (fs, ts, "*c*n")

          --}}}
          --{{{  keyboard read & display of pulse
                  off time

          so.write.string.nl    (fs,    ts,    "
             pulse.off.time (ms) ? .. *c*n")
```

```
so.read.echo.int (fs, ts, pulsewidth[1],
                  going)
so.write.nl (fs,ts)
so.write.string (fs, ts, "*c*n")

--}}}
--{{{  keyboard read & display of pulse
       samples

so.write.string.nl (fs, ts, " Request
  160n flowsmples ( n = 1, 2 ..30 )
   *c*n")
so.read.echo.int  (fs,  ts,  nsamples,
   going)
so.write.nl (fs,ts)
so.write.string (fs, ts, "*c*n")

--}}}
--{{{  keyboard read & display of 1st
       latchset (hex)

so.write.string.nl (fs, ts, " enter
   latch0.. *c*n")
 so.read.echo.hex.int    (fs,    ts,
   gainint[0], going)
so.write.nl (fs,ts)
so.write.string (fs, ts, "*c*n")

--}}}
--{{{  keyboard read & display of 2nd
       latchset (hex)

so.write.string.nl  (fs,  ts,  "  enter
   latch1 .. *c*n")
 so.read.echo.hex.int    (fs,    ts,
   gainint[1], going)
so.write.nl (fs,ts)
so.write.string (fs, ts, "*c*n")

--}}}
--{{{  keyboard read & display of 3rd
       latchset (hex)

so.write.string.nl  (fs,  ts,  "  enter
       latch2 .. *c*n")
 so.read.echo.hex.int    (fs,    ts,
       gainint[2], going)
so.write.nl (fs,ts)
so.write.string (fs, ts, "*c*n")

--}}}
--{{{  keyboard read & display of 4th
       latchset (hex)
so.write.string.nl  (fs,  ts,  "  enter
                  latch3 .. *c*n")
```

```
                        so.read.echo.hex.int (fs, ts,
                          gainint[3], going)
                        so.write.nl (fs,ts)
                        so.write.string (fs, ts, "*c*n")

                        --}}}
                        --}}}

                        --#COMMENT "Type convert Hex data" I/P
                        --#COMMENT "to Int"

                        SEQ

                          SEQ j = 0 FOR 4

                            SEQ

                              latch[j] := BYTE (gainint[j])


                        --#COMMENT "Parallel send & receive"
                        --#COMMENT " data to/fro other"
                        --#COMMENT "processes. Without 'WHILE

                        PAR  --SEQ

                          SEQ

                            tc ! chanout.4ints.4byte; byte.char;
                                    number;
                                  pulsewidth; nsamples; latch

                            --CAUSEERROR()

                          rec.status()

                    TRUE
                      SKIP
                --}}}
          TRUE

            SKIP

:
--}}}

--#COMMENT "The various agorithmic tasks to required for"
--#COMMENT "the powder"

WHILE TRUE
  SEQ

    --#COMMENT "The byte.char is always reset after each"
    --#COMMENT "run, thus"
```

```
--#COMMENT "initialising this synchronisation switch"
--#COMMENT "for a repetitive read of data"

byte.char := 0(BYTE)


SEQ

  so.write.string.nl (fs, ts, "MENU")

  so.write.nl (fs, ts)
  so.write.string.nl (fs, ts, " Press desired
     character")
  so.write.nl (fs, ts)
  so.write.string.nl(fs, ts, "B: Block data (raw)
  capture & save on HARD drive C ")
  so.write.nl (fs, ts)
  so.write.string.nl(fs, ts, "C: Injection detection
       (Atod preprocess)")
  so.write.nl (fs, ts)
  so.write.string.nl(fs, ts, "D: Signal peak detect
       (Array processdata)")
  so.write.nl (fs, ts)
  so.write.string.nl(fs, ts, "M: Mass flowrate
       estimate (System process)")
  so.write.nl (fs, ts)
  so.write.string.nl(fs, ts, "Q: Quit Menu")
  so.write.nl( fs, ts )
  so.write.string.nl(fs, ts, "Enter request..")
  so.write.nl( fs, ts )
  so.write.nl( fs, ts )
  so.write.nl( fs, ts )
  so.write.nl( fs, ts )
  so.write.nl( fs, ts )
  --so.write.nl( fs, ts )
  so.getkey( fs, ts, byte.char, result )

  --#COMMENT "This renders the task commands CASE"
  --#COMMENT "insensitive"

  SEQ

    --{{{  --#COMMENT "execute the various commands"

    IF
      byte.char >= 'a'

        SEQ
          byte.char := (BYTE (( INT byte.char) - ((INT
            'a') - (INT 'A'))))
      TRUE
        SKIP
```

```
IF
  --#COMMENT " Exit command to DOS "

  byte.char = 'Q'

    so.exit (fs, ts, sps.success)

  --#COMMENT " Block datacapture command; already
  --#COMMENT " explained "

  byte.char = 'B'

    SEQ

      table ()    -- procedure call
      menu()

  --#COMMENT "Flow status command; already"
  --#COMMENT "explained above "

  byte.char = 'C'

    SEQ

      display.status ()

  --#COMMENT "Closing Guard for the Conditional"
  --#COMMENT "IF construct"

  TRUE

    SKIP
--}}}
```

:

2.3 Data storage & message passing program

```
--#COMMENT "PROC cordk.occ stores 4000 bytes from each sensor"
--#COMMENT "for transmission to the HOST system"
--#COMMENT "This program serves, also, as "
--#COMMENT "a multiplexing switch for the sensor signals"

--#COMMENT "Variant protocol declaration "
#INCLUDE  "protoco.inc"

--#COMMENT " 2 Server channels & 4 protocol channels declared"

PROC cordinate       ( CHAN OF ATOD.CHANS tc,
                       CHAN OF ATOD.CHANS c.out.array0,
                       CHAN OF ATOD.CHANS c.in.array0,
                       CHAN OF ATOD.CHANS c.in.array1,
                       CHAN OF ATOD.CHANS c.in.array2,
                       CHAN OF ATOD.CHANS fc)
  --{{{  Variable declarations
  BYTE byte.char:
  INT number:
  [2]INT pulsewidth:
  INT nsamples:
  [4]BYTE latch:
  [4][arrayslice]BYTE datastream:
  [6][sensor.no.samples]BYTE store:
  INT intn:
  --}}}
```

```
--#COMMENT "Int count is used for debugging purpose "
INT count:
SEQ
  --#COMMENT "Initialising count to zero"
  count := 0
  --#COMMENT " Input data message from the Host "
  WHILE TRUE
    SEQ
      tc ? CASE
        chanout.4ints.4byte; byte.char; number; pulsewidth;
        nsamples; latch
          IF
            --#COMMENT "IF Pc-Host data is received "
            --#COMMENT "then out message to Array Processor
            --#COMMENT "0, otherwise "
            --#COMMENT " stop communication "
            (byte.char = 'B')
              --{{{  store data from worker procs + output
                     to Pc
              --{{{
              PAR
                SEQ
                  --#COMMENT " This count Construct does the
                  --#COMMENT "looping; either"
                  --#COMMENT " crashes on transputer error"
                  --#COMMENT "flag or keeps "
                  --#COMMENT "waiting on the IF construct
                  --#COMMENT "guard; "
                  count := (count + 1)
                  IF
                    count = 15
                      CAUSEERROR()
                    TRUE
                      SKIP
```

```
  --#COMMENT "Transmit blended message to"
  --#COMMENT "Array Process 0 "
  --#COMMENT "on no error tracked on count"

 SEQ
   c.out.array0    !    chanout.4ints.4byte;
            byte.char; number; pulsewidth;
            nsamples; latch

SEQ


  --#COMMENT "WHILE TRUE will be needed for"
  --#COMMENT "continous "
  --#COMMENT "data store & processing.
  --#COMMENT "currently commented"
  --#COMMENT " out since not being used"

  --WHILE TRUE

  SEQ

    --{{{  Val & Int declarations
    VAL blocksize IS nsamples:
    --sensor.no.samples:
    VAL no.of.loops IS (blocksize /
                        arrayslice):

    INT pointer.block.byte.read:

    --}}}

    SEQ

      --#COMMENT "This pointer run from 0"
      --#COMMENT "to 4000 "
      --#COMMENT "in steps of 16"
      --#COMMENT "(arrayslice value)"


      pointer.block.byte.read := 0  --init.
       --pointer
      INT store.offset, store.offset1:
      INT  store.offset2:

      --{{{  array store redeclared as
            --variables

      --#COMMENT "The stores are the
      --#COMMENT "respective arrays "

      --#COMMENT "for storing the"
      --#COMMENT "respective sensor data"
```

```
store0 IS store[0]:

store1 IS store[1]:

store2 IS store[2]:

--}}}

SEQ  i = 0 FOR no.of.loops
  SEQ

    --#COMMENT "Zero offset"
    --#COMMENT "initialise"
    --#COMMENT "array stores "

    store.offset := (i* arrayslice)

    store.offset1 := store.offset

    store.offset2 := store.offset1


    --{{{  Receive & store 4000 Bytes
           from each sensor

    --#COMMENT "Parallel receive &"
    --#COMMENT "store data from "

    --#COMMENT "channels 0 -2 Sensors
    --#COMMENT "(1 - 3 ) "

    PAR   --ALT

      c.in.array0 ? CASE
             chanout.byte.array0;
           datastream[0]
          SEQ j = 0 FOR arrayslice
            SEQ
              store0[(store.offset
                + j ) ]      :  =
                datastream[0][j]


      c.in.array1 ? CASE

             chanout.byte.array1;
           datastream[1]

          SEQ j = 0 FOR arrayslice
            SEQ
                store1[(store.offset
                +j)] :=
                datastream[1][j]


                                    --B1.39
```

```
                        c.in.array2 ? CASE
                              chanout.byte.array2;
                            datastream[2]

                            SEQ j = 0 FOR arrayslice
                              SEQ
                                  store2[(store.offset
                                  +j)] :=
                                  datastream[2][j]

                          --#COMMENT "16 step increment"
                          --#COMMENT "until 4000"
                          pointer.block.byte.read :=
                         (pointer.block.byte.read +
                          arrayslice)

                          -}}}
                          --SEQ
                            --c.in.array0 ? CASE  printer1;
                            --intn

                      --#COMMENT "Then output total 12000"
                      --#COMMENT "bytes to HOST "
                      --#COMMENT "reading 4000 bytes from"
                      --#COMMENT "each Sensor chan."
                      SEQ
                        --{{{  Sequential o/p of collected
                               data to HOST
                        fc ! chanbuffer2; store[0]
                        fc ! chanbuffer2; store[1]
                        fc ! chanbuffer2; store[2]
                        --fc ! printer1; intn
                        --}}}
          --}}}
          --}}}
        byte.char  =  'C'
          SEQ
            PAR
              --#COMMENT "This process is desirable for
              --#COMMENT "Injection status"
              SEQ
                c.out.array0   !   chanout.4ints.4byte;
                        byte.char; number;
                       pulsewidth; nsamples; latch
              SEQ
                c.in.array0 ? CASE
                  printer1; intn
                    SEQ
                      fc ! printer1; intn
        TRUE
          SKIP
:
```

2.4 Signal peak detection & message passing program

```
#OPTION "A"

#INCLUDE "protoco.inc"

PROC    sensor3.process    (    CHAN    OF    ATOD.CHANS
                          array.in.atod1,
                        CHAN OF ATOD.CHANS c.in.array1)
  --{{{  variable decls.(1)  -- used for message passings
  [arrayslice]BYTE datastream:
  [4]BYTE latch:
  [2]INT pulsewidth:
  INT number, nsamples:
  BYTE byte.char:
  INT intn:
  INT count:
  --}}}
  --{{{  variable decls.(2)  -- used for signal peak detection
  --#COMMENT " Data peaking variables "
  INT data:
  [2]INT min:
  [2]INT max:
  [2]INT sum.data:
  --#COMMENT " Peak timing variables "
  TIMER time:
  INT start:
  [2]INT peaktime:
  INT finish:
```

```
INT elapsed:
--}}}
[]BYTE Bdata RETYPES data:
data.b IS Bdata[0]:
SEQ
  PROC data.process ()
    SEQ
      min[0] := 0(INT)
      max[0] := 255(INT)
      sum.data := max
      time ? start
      maximum IS max[0]:
      minimum IS min[0]:
      sum  IS sum.data[0]:
      new.peak.time IS peaktime[0]:
      --{{{  LOCAL PROC 1 ( recieve 16 bytes data )
      PROC recieve.data ([arrayslice]BYTE datastream)
        SEQ
          array.in.atod1    ? CASE    chanout.byte.array2;
             datastream
      :
      --}}}
      --{{{  LOCAL PROC 2 (process recieved data)
      PROC process.data ([arrayslice]BYTE datastream )
        SEQ
          data := 0
          SEQ i = 0 FOR arrayslice
            SEQ
              data.b := datastream[i]
```

```
          IF
            data > maximum
              SEQ
                maximum := data
                time ? new.peak.time
            TRUE
              SKIP
          IF
            data  < minimum
              minimum := data
            TRUE
              SKIP
          sum := sum + data
:
--}}}
--{{{  LOCAL PROC 3 ( recieve & process in parallel)
PROC peaking ()
  [arrayslice]BYTE A,B:
  SEQ
    recieve.data (A)
    PAR
      recieve.data (B)
      process.data (A)
    SEQ i = 0 FOR ((nsamples -32) >> 1) >> 4
      SEQ
        PAR
          recieve.data(A)
          process.data(B)
```

```
                PAR

                  recieve.data(B)

                  process.data(A)

              process.data (B)

        :
        --}}}

        --{{{  LOCAL PROC 4 ( transfer block of data )

        --#COMMENT "I/p packetised 16bytes from atod & output"
        --#COMMENT "same to controller"

        PROC transfer()

          SEQ

            WHILE TRUE
              SEQ

                --#COMMENT "Packetised 16 bytes continous i/p"

                array.in.atod1  ? CASE

                  chanout.byte.array2; datastream  ---packet
                    transmission of 16 bytes

                    SEQ
                      --#COMMENT " Packetised 16 bytes
                      --#COMMENT "continous o/p "

                      c.in.array1 ! chanout.byte.array2;
                                          datastream
        :
        --}}}

        SEQ
          --peaking ()
          transfer()
    :

    SEQ

      data.process()

:
```

2.5 Data transfer program

```
#COMMENT "This module runs on one of the array processors"
#COMMENT "It passes data to and from A/D"

#OPTION "A"


#INCLUDE "protoco.inc"

PROC sensor1.process     (CHAN OF ATOD.CHANS, c.out.array0,
                             CHAN OF ATOD.CHANS array.out.atod0,
                           CHAN OF ATOD.CHANS array.in.atod0,
                         CHAN OF ATOD.CHANS c.in.array0)

  --{{{  variable decls.(1)  -- used for message passings
  [arrayslice]BYTE datastream:
  [4]BYTE latch:
  [2]INT pulsewidth:
  INT number, nsamples:
  BYTE byte.char:
  INT intn:
  INT count:
  --}}}
--#COMMENT "These parameters are not used here"
  INT data:
  [2]INT min:
  [2]INT max:
  [2]INT sum.data:
  TIMER time:
  INT start:
  [2]INT peaktime:
```

```
INT finish:

INT elapsed:

--}}}

[]BYTE Bdata RETYPES data:

data.b IS Bdata[0]:

INT count:

SEQ

  WHILE TRUE
    SEQ
      c.out.array0  ? CASE

        chanout.4ints.4byte; byte.char; number;
                             pulsewidth; nsamples; latch

          SEQ
            SKIP


            --{{{  message passing (injection parameter)

            IF
              byte.char = 'C'

                PAR

                  array.out.atod0  !  chanout.4ints.4byte;
                            byte.char;  number;
                            pulsewidth; nsamples; latch

                  SEQ
                    SEQ
                      array.in.atod0 ? CASE
                        printer1; intn
                          SEQ
                            c.in.array0 ! printer1; intn


             --#COMMENT "Data passing to and from A/D"


             byte.char = 'B'

               SEQ

                  array.out.atod0  !  chanout.4ints.4byte;
                       byte.char;    number;    pulsewidth;
                            nsamples; latch

                  SEQ
                    PAR
```

```
WHILE TRUE

  SEQ

    c.out.array0  ? CASE

      chanout.4ints.4byte;byte.char;
                number;  pulsewidth;
                nsamples; latch


        SEQ
              array.out.atod0    !
                chanout.4ints.4byte;
                     byte.char;number;
                          pulsewidth;
                        nsamples;
                     latch
SEQ
  count :=0

  WHILE TRUE

    SEQ
      --{{{ rec.atod / out.controller
      array.in.atod0  ? CASE

            chanout.byte.array0;
               datastream
               --packet transmission
               --of 16 bytes

          SEQ

            --CAUSEERROR()

            --#COMMENT "The "
            --#COMMENT "16 bytes"
            --#COMMENT "packetised"
            --#COMMENT "continous
            --#COMMENT "o/p to"
            --#COMMENT "controller"

                 c.in.array0     !
               chanout.byte.array0;
               datastream

            count := count + 1

            IF
              count = 2560

                CAUSEERROR()

              TRUE
                SKIP
                                  --B1.47
```

```
                                        --}}}

                  TRUE
                    SKIP
                --}}}
:
```

2.6 The Protocol file

```
--#COMMENT "This variant protocol is used in the flow"
--#COMMENT " software for declaring the global variables"


--#COMMENT "Arrayslice used all through the software for 16"
--#COMMENT "bytes databuffs"

VAL INT arrayslice IS 16:

--#COMMENT "Fixed number of data samples desired from each"
--#COMMENT "atod sensor"

VAL INT sensor.no.samples IS  4800:

PROTOCOL ATOD.CHANS

  CASE

    --#COMMENT "Extensively used tag for messsage passing"
    --#COMMENT "(Host - atod)"

    --#COMMENT " ...Refer  processes for the data  messages "

    chanout.4ints.4byte; BYTE; INT; [2]INT; INT; [4]BYTE

    --#COMMENT "Buffering Tag used on Controller & Host"
    --#COMMENT "processes"

    --#COMMENT "4000bytes are stored on the controller before
    --#COMMENT "output to Host"

    chanbuffer2; [sensor.no.samples]BYTE

    --#COMMENT "16bytes databuffs.for datatransfer from ATOD"
    --#COMMENT "to other procs."

    --#COMMENT "The 16bytes packets are processed on the"
    --#COMMENT "arrays & controller"


    chanout.byte.array0; [arrayslice]BYTE
    chanout.byte.array1; [arrayslice]BYTE
    chanout.byte.array2; [arrayslice]BYTE


    --#COMMENT "Printer Tag communicates the flow status from"
    --#COMMENT "AtoD to HOST"

    --#COMMENT "Extensively used in the atod data preprocess"
    --#COMMENT "& analyses"
    printer1; INT
:
```

2.7 The Configuration file (Soft1.pgm)

```
#COMMENT "The configurattion file links and configures the"
#COMMENT "various #modules to run on the developemnt system"


--{{{F pc2.occ
--:::F PC2.OCC
--}}}


--{{{F cordk.occ
--:::F CORDK.OCC
--}}}

--{{{F flow2.occ
--:::F FLOW2.OCC
--}}}

--{{{F flow2a.occ
--:::F FLOW2A.OCC
--}}}

--{{{F flow2p.occ
--:::F FLOW2P.OCC
--}}}

--{{{F flow2b.occ
--:::F FLOW2B.OCC
--}}}

--{{{F atod1.occ
--:::F ATOD1.OCC
--}}}

--{{{F protoco.inc
--:::F PROTOCO.INC
--}}}


--{{{  INCLUDES ( standard libraries)


#INCLUDE "linkaddr.inc"

#INCLUDE "hostio.inc"

#INCLUDE "protoco.inc"


--}}}
```

```
#USE  "pc2.c8h"
#USE  "cordk.c8h"

#USE  "flow2.c8h"
#USE  "flow2a.c8h"
#USE  "flow2p.c8h"
--#USE  "flow2b.c8h"
#USE  "atod1.c8h"
#COMMENT "Declaration of the iserver channels"
CHAN OF SP   fs, ts:
#COMMENT "Placement of links"
PLACED PAR
  PROCESSOR 0 T8
    PLACE ts AT link0.out:
    PLACE fs AT link0.in:
    --internal channel declarations
    CHAN OF ATOD.CHANS tc, fc:

    --{{{  Non-array channel declarations ( Used )
    CHAN OF ATOD.CHANS c.out.array0:
    CHAN OF ATOD.CHANS c.out.array1:
    CHAN OF ATOD.CHANS c.out.array2:
    CHAN OF ATOD.CHANS array.out.atod0 :
    CHAN  OF  ATOD.CHANS  array.in.atod0,  array.in.atod1  ,
     array.in.atod2:
    CHAN OF ATOD.CHANS c.in.array0, c.in.array1, c.in.array2:
    --}}}
    INT  memory:
    []INT free.memory  RETYPES memory:
```

```
PAR
  --{{{  Non-array placement of software channels (used)
  pc.driver.code        ( fs, ts, tc, fc )
  cordinate             (tc, c.out.array0,  c.in.array0,
                         c.in.array1, c.in.array2, fc)
  sensor1.process       (c.out.array0, array.out.atod0,
                          array.in.atod0,  c.in.array0)
  sensor2.process       (array.in.atod1,  c.in.array1)
  sensor3.process       (array.in.atod2,  c.in.array2)
  atod.preprocess       (array.out.atod0, array.in.atod0,
                        array.in.atod1, array.in.atod2 )
  --}}}
```

2.8 Occam 2 product compiler (19th September 1988)

```
PROCESSOR 0 0 T8 1

SC 0 8268 8268 0
SC 1 504 504 0
SC 2 360 360 0
SC 3 108 108 0
SC 4 352 352 0
SC 5 1132 1132 0

CODE 216 0 1380 53668
```

3.0 APPENDIX C: DETAILED STRUCTURE OF THE SENSOR UNIT

This appendix provides the detailed diagrammatic structure of the sensor unit. The detailed description of the sensor unit has been provided in chapter 4 of the thesis main section. The description of the sensor structure is provided below.

3.1 The complete sensor unit assembly

Figure C1.1 shows the complete assembly of the sensor unit with the various sections. The sensor body is made up of perspex and the sensor screws are of mild steel. All the screws are of 4mm diameter. The sensor pipe connections are carried out to fix into the internal diameter of the reinforced rubber tube. The sensors are separated by 0.2mm acetate insulation spacers.

3.2 Area view of the sensor unit

Figure C1.2 shows the area view of the sensor unit. The diagram shows outlines of the tribo-charging neutralising (discharging) electrodes, charge injection unit, and the sensors unit with the respective separating acetate materials.

3.3 The charge injection unit

Figure C1.3 shows the detailed structure of the injection unit. This unit, which includes the hypodermic needle, can be dismantled easily from the sensor main body. A compressed air pressure of about 1.6 bars can be applied, through a small pipe, to aerate the needle. This regular aeration of the hypodermic needle prevents build up of powder, thus maintaining a consistent performance of the injection process.

3.4 The side view of the injection unit

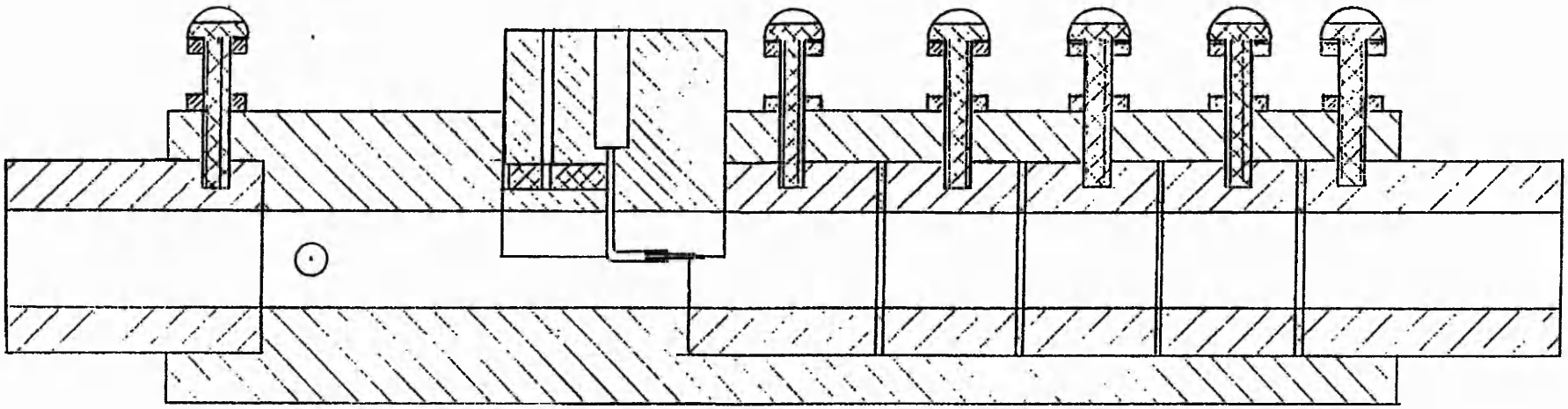
Figure C1.4 shows the side view of the injection unit with the various dimensions provided. This diagram reveals the outlines of the hypodermic needle and the adjusting screw for holding in position the high voltage connection.

3.5 The hypodermic needle

Figure C1.5 provides the detailed dimensioned diagram of the hypodermic needle. This needle was bent, to align with the direction of the downstream powder flow. A thin piece of wire, with a fine radius tip, was inserted and held rigidly to the needle tip. The electric field intensity of the injection is related to the radius of the tip. The smaller the radius, the higher the intensity of the electric field strength.

3.6 The discharging electrodes

Figure C1.6 is the diagram of the discharging electrodes, with the respective dimensions. The usefulness of this discharging electrodes has already been discussed in chapter 4. The build up of charges on the powder in the powder hopper and venturi unit can be discharged by these electrodes, before the injection unit.



COMPLETE ASSEMBLY

Figure C1.1: The sensor unit complete assembly

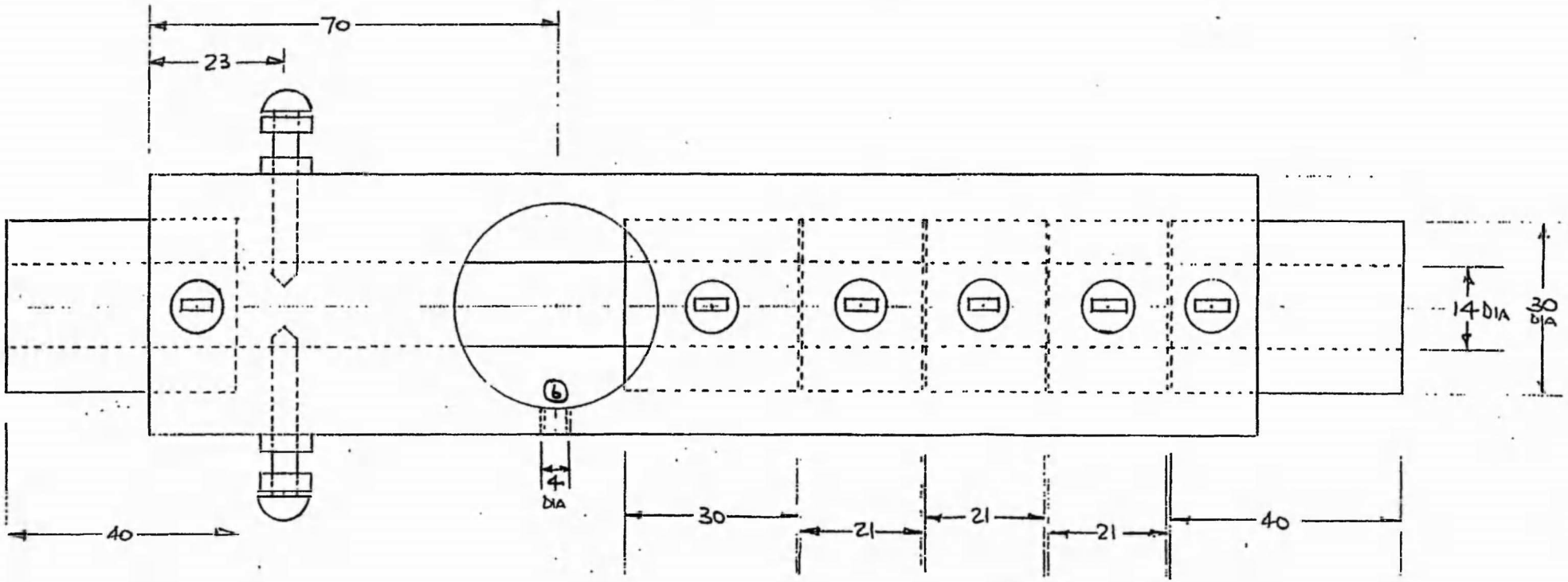


Figure C1.2: Area view of the sensor unit

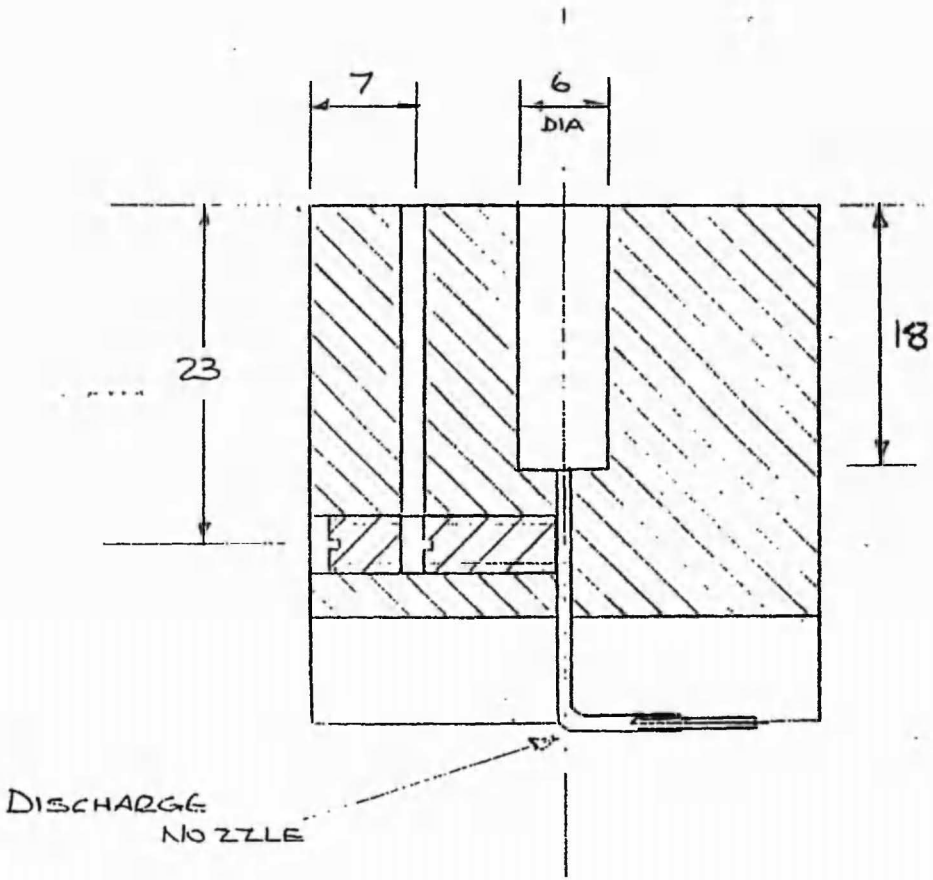


Figure C1.3: The charge injection unit

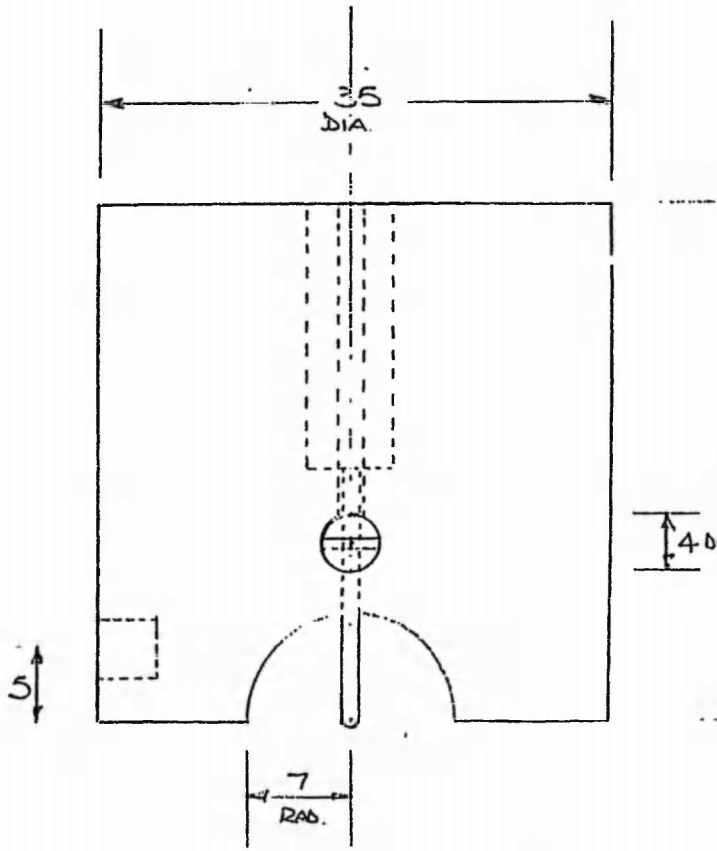


Figure C1.4: The injection unit side v

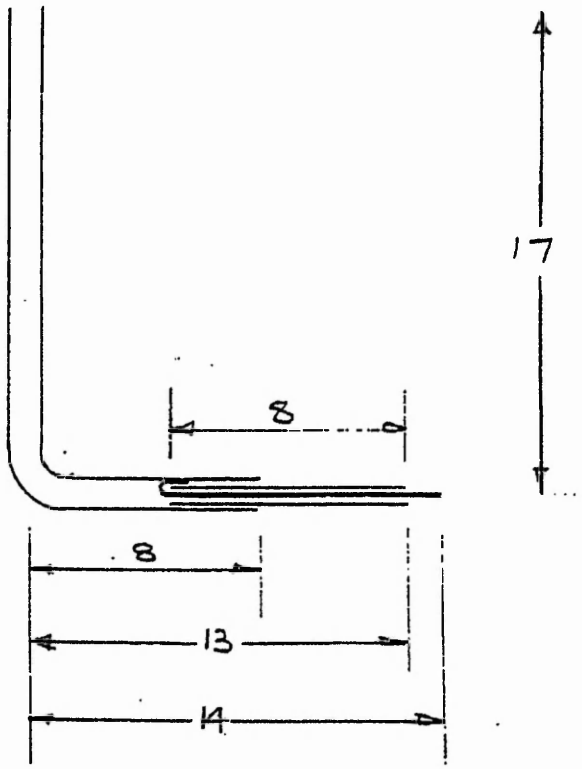


Figure C1.5: The hypodermic needle

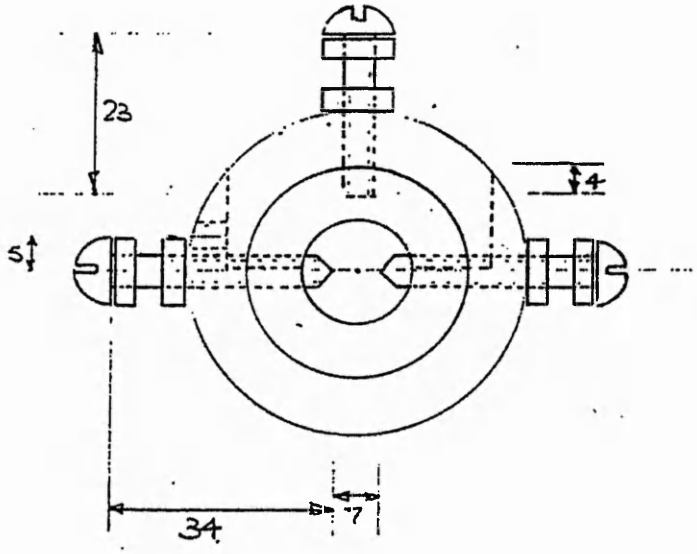


Figure C1.6: The discharging electrodes