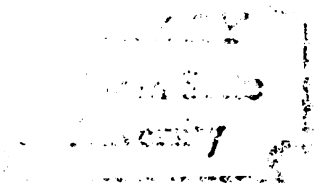


This is to certify that the

thesis entitled

ADAPTIVE SAMPLING FOR ENHANCED PERFORMANCE OF
AN ION DETECTION SYSTEM FOR
A TRIPLE QUADRUPOLE MASS SPECTROMETER

presented by

Robert Stanley Matthews

has been accepted towards fulfillment
of the requirements for

MS degree in EE

P. David Fisher by JBK

Major professor

Date 2/26/82

O-7639

ADAPTIVE SAMPLING FOR ENHANCED PERFORMANCE OF AN ION DETECTION SYSTEM FOR A TRIPLE QUADRUPOLE MASS SPECTROMETER

By

Robert Stanley Matthews

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Department of Electrical Engineering and System Science

1982

ABSTRACT

ADAPTIVE SAMPLING FOR ENHANCED PERFORMANCE OF AN ION DETECTION SYSTEM FOR A TRIPLE QUADRUPOLE MASS SPECTROMETER

By

Robert Stanley Matthews

Detection and tabulation of mass-selected ions are the primary means of data acquisition for mass spectrometers. Previous detection systems have been limited in dynamic range, or in data throughput, so that inefficient and limited use of the ion information resulted.

Analysis of the statistical properties of randomly arriving ions and the use of state-of-the-art technologies have led to the development of an enhanced ion detection system (IDS) for a triple quadrupole mass spectrometer (TQMS). An adaptive sampling technique maximizes data throughput without distorting relative error or limiting dynamic range. This microprocessor-based system measures flux rates from 10^0 to 10^9 ions per second up to a maximum relative error of 1%; sampling periods are reduced to their statistically defined minimums.

Preliminary tests have validated the correctness of the IDS design along with its capability for on-line customized operation through the use of powerful and flexible high-level software.

To my wife and parents

*** ACKNOWLEDGEMENTS ***

I would like to thank Dr. Chris Enke for his support and for his contributions relative to the conceptualization and refinement of the project design. Thanks are also extended to the entire Mass Spectrometry Research Group for their assistance, especially to Mr. Bruce Newcome and Mr. Carl Myerholtz for their insights related to hardware and software development.

Special thanks are extended to Dr. Dave Fisher for his aid in the development of this manuscript, as well as for his support and guidance throughout my academic career. Also, I wish to express my appreciation to Mr. Ron Kraus for providing the fine artwork of this manuscript.

This research was supported by the Office of Naval Research and by the National Institute of Health.

*** TABLE OF CONTENTS ***

page

LIST OF FIGURES vi

I. INTRODUCTION 1
1.1 Mass Spectrometry Overview 2
1.2 Triple Quadrupole Mass Spectrometer (TQMS) 3
1.3 Ion Detection System (IDS) 7
1.4 Thesis Organization 9

II. DESIGN REQUIREMENTS 10
2.1 Input Characteristics 10
2.1.1 Dynamic Range 11
2.1.2 Acquisition Speed and Sampling Precision 13
2.2 Control and Interfacing 16
2.3 Other Considerations 19
2.3.1 Outputs 19
2.3.2 Noise 20
2.3.3 Physical Deployment 20
2.3.4 Stability and Reliability 21
2.3.5 Cost 21

III. HARDWARE 22
3.1 Hardware Overview 22
3.2 Detection Unit (DU) 24
3.2.1 Dual-Mode Channeltron 26
3.2.2 Support Circuitry 28
3.2.3 Housing 31
3.3 Processing Unit (PU) 32
3.3.1 Pulse Channel Circuitry 32
3.3.2 Analog Channel Overview 36
3.3.3 Current-to-Voltage Converter 37
3.3.4 Integration Circuit 39
3.3.5 A/D Converter 43
3.3.6 System Timer/Counter (STC) 44
3.3.7 Analog Control Outputs 47
3.3.8 PU Support Logic 48
3.3.9 Miscellaneous Circuitry 51
3.4 Control Unit (CU) 52
3.4.1 Central Processing Unit 53
3.4.2 Memory 53
3.4.3 Block Selects (Address Decoding) 55
3.4.4 Dual USARTs 55
3.4.5 Interrupt Controller 56
3.4.6 CU-PU Interface 56
3.4.7 Multi-Micro Interface 57

page

IV. SOFTWARE 58
4.1 Languages 59
4.1.1 FORTH 59
4.1.2 8085 Assembly Language 61
4.1.3 Structure 61
4.2 Command Classifications 64
4.2.1 Elementary Functions 65
4.2.2 Data Processing 65
4.2.3 Calibration 66
4.2.4 High-Speed Control 69
4.2.5 Standard Control 70
4.2.6 Development 71
4.3 Numeric Representation 72
4.3.1 Internal Representations 72
4.3.2 I/O Representations 73

V. EVALUATION 76
5.1 Sampling Simulation 77
5.2 Processing Unit (PU) Performance 78
5.2.1 C/V Converter 81
5.2.2 Integration Circuit 82
5.2.3 A/D Converter 83
5.2.4 STC 84
5.2.5 Other PU Circuits 84
5.3 Control Unit (CU) Performance 85
5.3.1 Software 85
5.3.2 Hardware 86
5.4 General Observations 86
5.4.1 Noise Immunity 87
5.4.2 Stability and Reliability 87
5.4.3 Usability 88

VI. SUMMARY 89
6.1 Achievement Of Goals 89
6.2 Further Improvements 91
6.3 Conclusions 92

APPENDICES 93
A. Processing Unit (PU) Calculations 93
B. IDS Command Glossary 97
C. Firmware Source Code 101

BIBLIOGRAPHY 107

*** LIST OF FIGURES ***

		page
1.1	Typical Mass Spectrometer	4
1.2	Triple Quadrupole Mass Spectrometer (TQMS)	5
1.3	Sample spectra	8
2.1	Relationship between ion flux, integration period, and error	15
2.2	TQMS Automated Control Struture	18
3.1	Ion Detection System (IDS) block diagram	23
3.2	Detection Unit (DU) block diagram	25
3.3	Dual-Mode Channeltron Characteristics	27
3.4	Control circuit for Detection Unit (DU)	29
3.5	Processing Unit (PU) block diagram	33
3.6	Pulse Channel Circuitry	34
3.7	Analog Channel block diagram	38
3.8	Current-to-Voltage (C/V) Converter Circuit	40
3.9	Integrator Circuit	42
3.10	A/D Converter Circuit	45
3.11	System Timer/Counter (STC) configuration	46
3.12	PU Support Logic --- Local Decoding Circuit	49
3.13	PU Support Logic --- Control Circuit	50
3.14	Control Unit (CU) block diagram	54
4.1	IDS Memory Map	62

page

4.2 Coordinating Program structure 63

4.3 Data Processing Command example 67

4.4 Data formats 75

5.1 Simulation program 79

5.2 C/V Converter and Integrator Circuit performance 80

*** CHAPTER 1 --- INTRODUCTION ***

The use of microprocessor-based instrumentation, such as by the Mass Spectrometry Research Group at Michigan State University, has become the basis for the continuing development of several advanced spectroscopy techniques. The flexibility, speed, and overall cost-performance benefits offered by microprocessor-based instrumentation in real-time control and data acquisition applications are central to the successful development of one such technique: Triple Quadrupole Mass Spectrometry.

This thesis describes the conceptualization and implementation of a portion of the instrumentation for the Triple Quadrupole Mass Spectrometer (TQMS) at MSU: a microprocessor-based Ion Detection System. The Ion Detection System (IDS) performs the key function of data acquisition for the TQMS using an adaptive sampling technique for enhanced performance.

The remainder of this chapter provides brief descriptions of mass spectrometry, the TQMS, the purpose of the IDS, and an outline of the subsequent chapters on the IDS development and realization.

1.1 MASS SPECTROMETRY OVERVIEW

In order to derive information describing the chemical composition of a substance, chemists may employ one or more of a variety of analytical techniques. Spectroscopy is one such class of techniques; spectra relating various atomic and molecular properties of a substance are used in classifying the substance.

Of all the spectroscopic techniques available to an analytic chemist, mass spectrometry is one of the most versatile because of the wide range of information it can provide [1]. "The analysis of a substance by mass spectrometry involves the conversion of a sample into atomic or molecular ions and subsequent separation of these ions on the basis of their mass-to-charge ratio by magnetic or electrostatic means. Projected ions are selected according to their mass characteristics and are recorded as a function of abundance; the resulting spectrum is characteristic of the original sample" [2].

"Modern mass spectrometers are constructed from elements which approach the state-of-the-art in vacuum systems, magnetic and electrostatic field generation, precision machining, solid-state electronics, and computerized data acquisition and processing. These instruments find applications in studies as diverse as biomolecules, petrochemicals, pharmocology, geochemistry, forensic chemistry, and the environment" [2].

A mass spectrometer can be viewed as composed of four major components: the ion source and its associated inlet, the mass analyzer, the vacuum system, and the ion detector. Figure 1.1 depicts a typical mass spectrometer and its major components [3]. The following section describes the TQMS (first developed at MSU) and provides further insight into the functioning of a mass spectrometer.

1.2 TRIPLE QUADRUPOLE MASS SPECTROMETER (TQMS)

The TQMS (Triple Quadrupole Mass Spectrometer) was originally developed for analytical applications by Yost and Enke during the late 1970's [4]. Figure 1.2 depicts the TQMS. Since contamination of the sample or of the generated ions would result in an erroneous characterization, special consideration is given to the design of the operating environment of the TQMS. Central to this design is the enclosure of the components within a tightly controlled vacuum (10^{-7} torr). Also, since large electrostatic potentials across relatively small dimensions are required, vacuum failure could result in destructive arcing. As a result, vacuum system design is critical.

A sample is converted to atomic or molecular ions within the ion source. These ions are extracted and focused by electrostatic potentials into a beam which is directed toward the mass analyzer section.

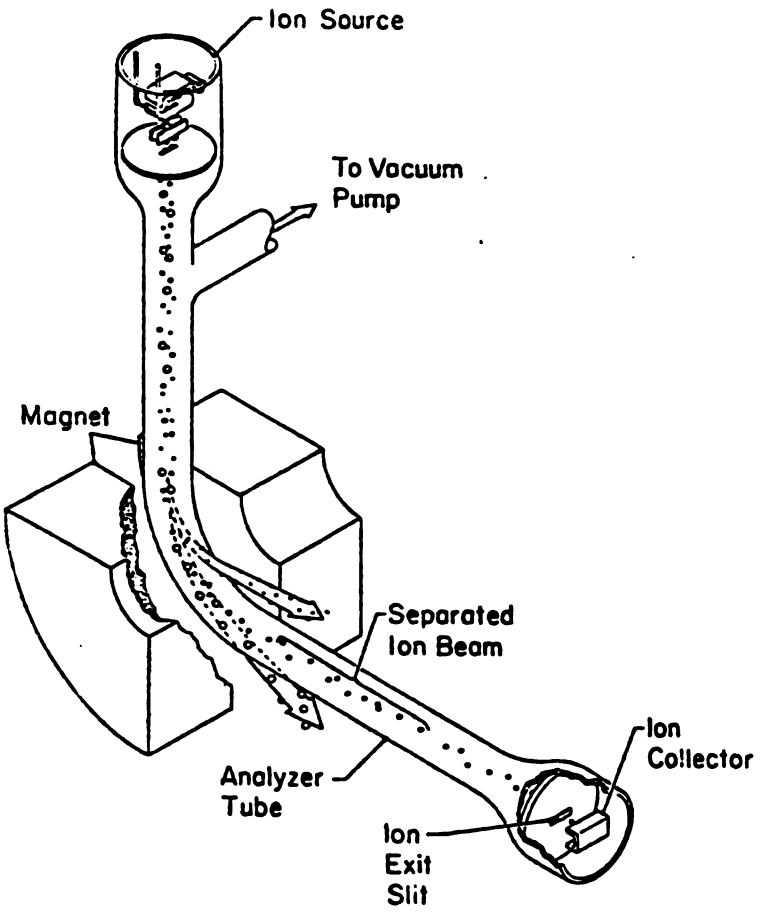


Figure 1.1 Typical Mass Spectrometer [3]

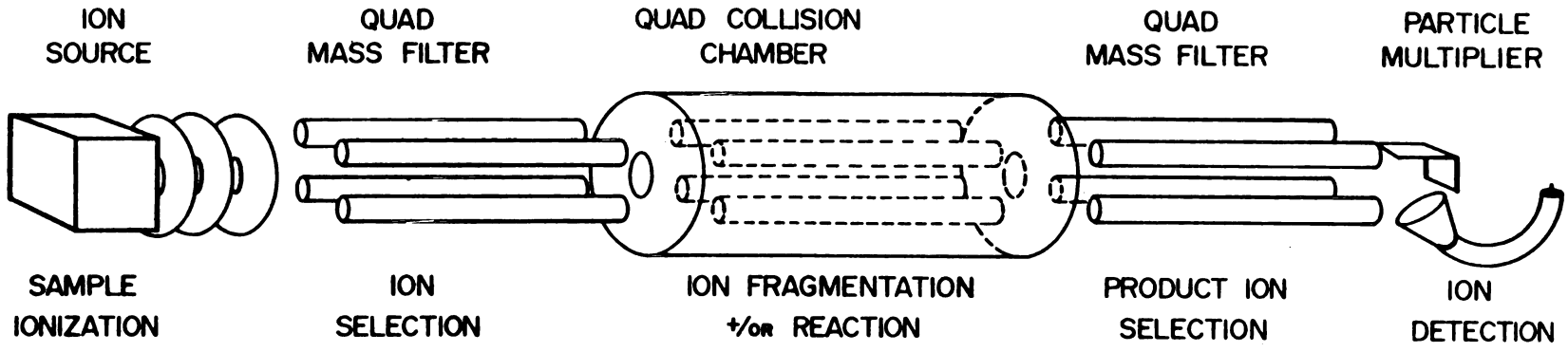


Figure 1.2 Triple Quadrupole Mass Spectrometer (TQMS) [4]

The mass analyzer section of the TQMS distinguishes it from other types of mass spectrometers. The first quadrupole mass filter separates the incoming ion beam such that all ions of a selected mass value (amu) are passed into the quad collision chamber; ions of all other masses are filtered out. This filtering is achieved through the interaction of DC and RF electrostatic fields supplied through the quadrupoles with the mass-charge ratio of the incoming ion beam [5].

In the quad collision chamber, an inert gas is introduced such that the incoming ions fragment upon collision with the inert gas molecules. A quadrupole is used as the collision chamber to provide continued focusing of the ion beam. This collision chamber provides an added dimension in analysis by enabling characterization of substances which are composed of large, complex molecules. A final quadrupole mass filter separates the incoming ion fragments for detection of ion abundance (as a function of mass) by the ion detector.

Most modern mass spectrometers employ an electron multiplier to perform the ion detection. The energy of the incident ions upon the first part of the multiplier's dynode chain is converted into secondary electrons, which upon subsequent collisions with the dynodes generate increasing numbers of electrons. Gains up to 10^7 are routinely obtained (the multiplier acts as a charge amplifier, with

each incoming ion representing one unit of charge). For ion flux determinations, the multiplier's output is encoded in the time domain as current pulses, or if the incident ion rate is faster than the response time of the multiplier, as an analog current. The TQMS was originally equipped with a standard Channeltron™-type of electron multiplier [6]. Though the standard Channeltron™ performed acceptably, it presented obstacles towards the full potential realization of the TQMS.

1.3 ION DETECTION SYSTEM (IDS)

The ion abundance, or more accurately the ion flux, incident upon the detector is correlated with the pass values of the mass filters to generate the characterization spectra; a sample spectra is given in Figure 1.3. The absolute ion flux is a function of several parameters of the TQMS and of the input sample; as a result, the absolute flux may range over many orders of magnitude. Since electron multipliers are physically limited in their dynamic range, active controls are evoked over detector operation to position this limited range "window" over the absolute range of the data. This creates added difficulties for detection system design and spectrometer operation, the details of which are discussed in Chapter 2.

A need to maximize range, accuracy, efficiency, proficiency, and simplicity of TQMS operation provides a

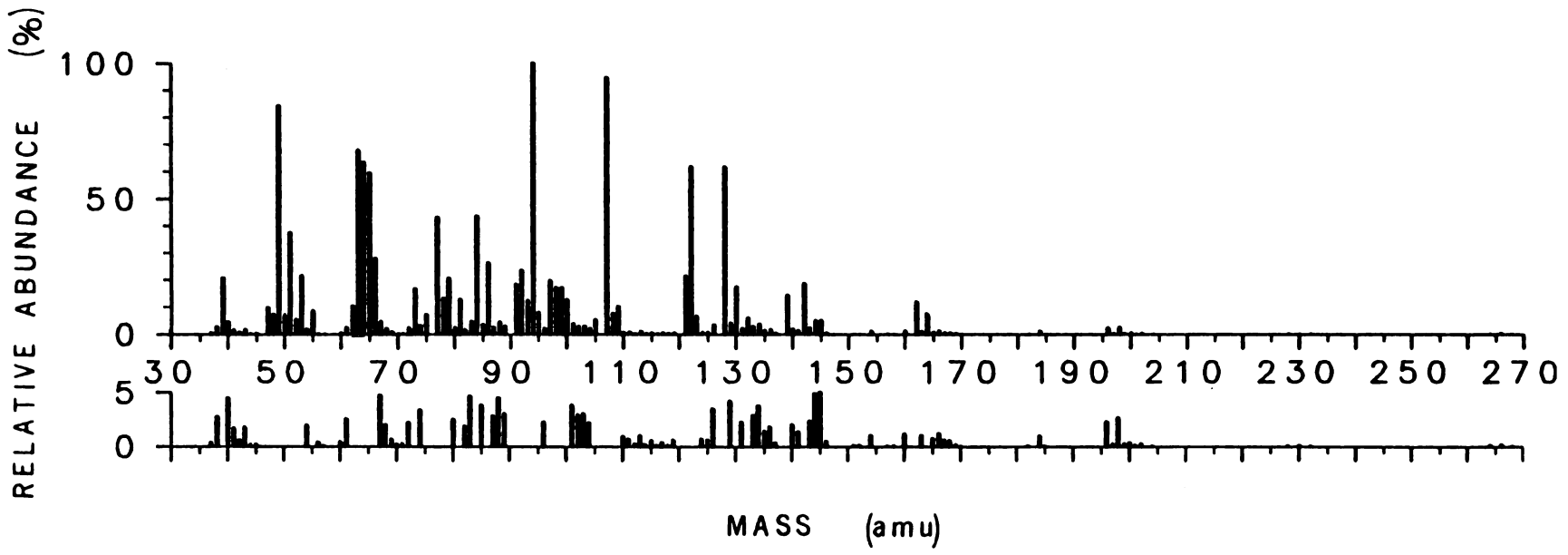


Figure 1.3 Sample spectra

continuing impetus for the TQMS operators/ designers to subject the instrument to a variety of component enhancements utilizing state-of-the-art technologies. Current major enhancement projects include the development of microprocessor-based control and data acquisition systems (the TQMS Multi-Micro System [7]), and an on-line computerized data processing and storage system. These objectives spurred the development of an enhanced ion detection system capable of operation within a sophisticated control environment.

1.4 THESIS OUTLINE

This thesis describes the IDS from its conceptualization to its realization. Chapter 2 explores in detail the set of design constraints imposed upon the IDS design, emphasizing the statistical considerations which form the basis for an adaptive sampling algorithm. Chapters 3 and 4 describe the specifics of the chosen implementation; Chapter 3 concentrates on the electronic and mechanical hardware, while Chapter 4 discusses the microprocessor software structure and programs. Chapter 5 evaluates the IDS performance, and Chapter 6 offers a general summary. Appendices are included which provide information on hardware and software pertinent to an IDS user.

*** CHAPTER 2 --- DESIGN REQUIREMENTS ***

The main function of the IDS is to detect ions directed toward its input and relay the number of ions per second (ion flux) detected to the TQMS data processing/storage system in an appropriate format. The TQMS user also requires very rapid data collection with minimum error and with minimal manual user interaction. In order to maximize these attributes, the design of the IDS is carefully correlated to a large number of TQMS operating parameters. The physical and statistical characteristics of the input sample(s) and its various ionic representations, the limitations of the mass spectrometer and its supporting instrumentation, and the interface to the TQMS control and data processing systems are all prime considerations in the IDS design. These factors in combination with the ever-present considerations of environment, reliability, cost, and simplicity of use are weighed against each other in establishing design requirements implementable with available technology. The following sections develop the key IDS design criteria.

2.1 INPUT CHARACTERISTICS

Performance of the IDS is fundamentally limited by factors characteristic of the ion input; analysis of the

interrelation of these factors provides the insight required for development of an "optimal" detection system. The three most critical factors are the dynamic range, the acquisition speed, and the sampling error.

2.1.1 Dynamic Range

Theoretically, information useable for spectra generation resides in ion flux levels ranging from the infinitesimal to the infinite. In reality, ion flux is limited on the low end by noise produced by the TQMS in the form of stray ions and on the high end by the saturation of sensitive source and detection components. These limitations are common to all types of mass spectrometers.

Some mass spectrometer applications require as little as two decades of dynamic range. Standard detection apparatus can provide up to four decades of range, sufficient for a majority of present applications. Since a sample is characterized not by the absolute, but rather by the relative ion flux per mass point, adjustments in source emission and/or detector sensitivity can provide expansion of the dynamic range beyond the limitations of the detector. Without going into details, this method results in degradation of calibration accuracy and system throughput.

The TQMS is presently designed to limit noise ions to 1 ips (ions per second), thus defining the low end of the

dynamic range (a 2:1 signal-to-noise ratio is acceptable in some analysis modes). Incident ion rates of 1 billion ips will saturate nearly all commercially available electron multipliers. Thus an absolute dynamic range of 10^0 to 10^9 ips is established.

At low ion flux levels (typically less than 10^6 ips), the response time of an electron multiplier is sufficient to permit pulse counting, provided the multiplier is operated at near its maximum gain (typically 10^7). Analog current measurement of the multiplier output is used when pulse overlap occurs for flux levels above 10^6 ips; in this case, the gain is reduced to prevent saturation. Gain is achieved through the application of large voltages across highly resistive dynodes.

Most present detection systems have opted to use only one type of output, reducing control and hardware requirements at the expense of limiting spectrometer performance. Those that opt for pulse counting reduce data throughput since high flux rates (which are quickly measured by analog techniques) require down-scaling and a corresponding increased sampling period. Stictly analog detection systems cannot accurately detect the tiny currents produced at low flux levels; increasing the emission rate of the ion source is often restricted due to the limited amounts of available sample. Thus the design of an optimal

detection system requires hardware capable of processing both pulse and analog signals.

2.1.2 Acquisition Speed and Sampling Precision

Because the data acquired (ion flux) by the IDS is based upon the occurrence of random events (ions detected), there is a strict dependency between the time taken to acquire a single piece of data and its resulting relative precision. One characteristic is improved at the expense of the other. The basic argument goes as follows:

Statistical laws predict the magnitude of deviations about an average value, i.e., white noise variation about a "DC" signal such as the "average" ion count [8,9]. The standard deviation for a random count is:

$$SD = \sqrt{COUNT}$$

Similarly, for a count rate (ion flux):

$$SD' = (COUNT / TIME)^{\frac{1}{2}} = \sqrt{FLUX}$$

The percent relative standard deviation is defined as the "typical" (within one standard deviation) error:

$$\%ERROR = \pm (\sqrt{FLUX} / FLUX) \times 100\% = \pm (FLUX)^{-\frac{1}{2}} \times 100\%$$

This inverse relationship between the flux and the error defines the fundamental limits of data acquisition

parameters. For example, as the number of ions counted over a fixed interval increases, the error decreases (and vice versa). Suppose that ions are being detected over a 1 second interval. If 100 ions are detected, then:

$$\%ERROR = \pm (100)^{-\frac{1}{2}} \times 100\% = 10\%$$

Suppose 1,000,000 ions are detected; then,

$$\%ERROR = \pm (1{,}000{,}000)^{-\frac{1}{2}} \times 100\% = 0.1\%.$$

Conversely, specification of the allowable error determines the minimum number of counted ions required for a given precision:

$$\text{MINIMUM COUNT} = (\%ERROR / 100\%)^{-2} = (10000\% / \%ERROR)$$

For example, if the error (precision) is specified as 1%,

$$\text{MINIMUM COUNT} = (10000\% / 1\%) = 10000 \text{ counts}$$

Thus counting can be terminated upon satisfaction of the minimum requirement.

Summarizing, for a fixed counting interval the error varies inversely with the count rate (ion flux). Similarly, setting of the acceptable error limit permits minimization of the sampling period relative to the flux. This intricate relationship between data parameters is shown graphically in Figure 2.1.

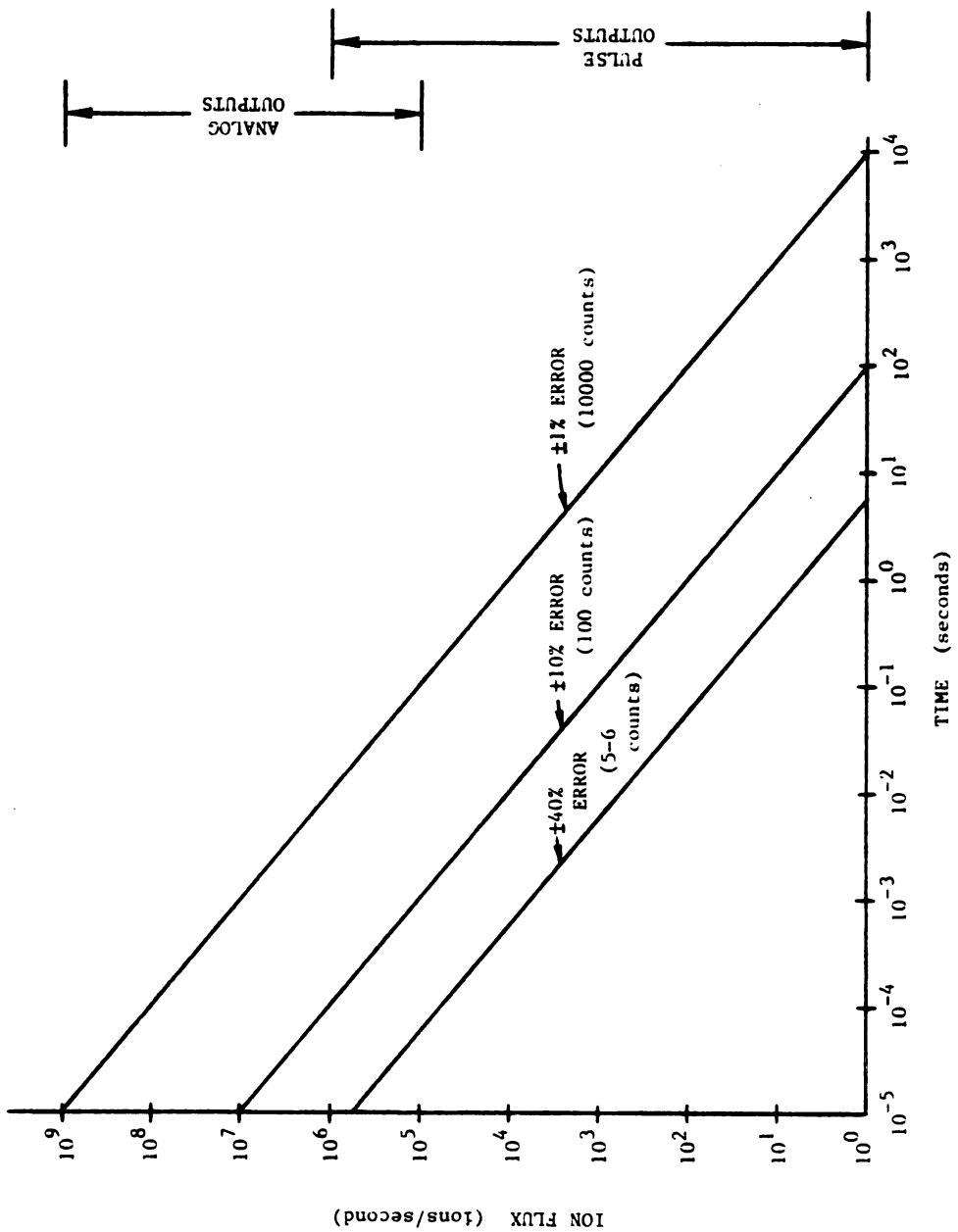


Figure 2.1 Relationship between ion flux, integration period, and error

In order to characterize large molecules, the TQMS is designed to scan samples from 1 amu to 1000 amu. Resolution of 10 parts per amu is sufficient to accurately define an ion flux peak; thus the entire domain of mass values is subdivided into 10000 portions.

The total number of ions emitted is a function of the amount of available sample and the source ionization efficiency. In cases where there are only a limited amount of ions available for filtering and detection, correspondingly quick scanning modes require very fast detection (error considerations are secondary) over a selected mass domain. With the present source, the worst case specification requires a complete scan of the entire mass domain in one second; this sets the maximum acquisition rate at 10 kHz.

Spectra with peaks defined to within 1% relative precision uniquely characterize a sample. In many types of TQMS analysis, such precision is not required. A detection system designed to sample adaptively permits the user to "trade-off" precision versus speed, since these two parameters are inversely related. The graph of Figure 2.1 clearly establishes operating points for the IDS.

2.2 CONTROL AND INTERFACING

Since data acquisition requires a great deal of flexibility on the part of the IDS in response to user

necessitated. A microprocessor-based control scheme compatible with the TQMS Multi-Micro control structure is chosen.

A block diagram of the entire TQMS automated control and data processing structure is given in Figure 2.2 [10]. Data reduction, storage, analysis, and color graphics are handled by the DEC LSI 11/23 minicomputer, independent of ongoing operations of the TQMS. The operation of the TQMS is controlled by the user through the Multi-Micro structure. The option to call upon a variety of pre-programmed operating modes, or to design customized operations is made available to the user. Operating parameters are supplied directly by the user or by the system through default.

In Figure 2.2 the IDS is represented as a single slave-microprocessor block (other slaves control the mass analyzer, vacuum, source, user interface, and other miscellaneous functions). Firmware to control the functions of the IDS exists in local memory. These functions are coordinated by software downloaded from the Multi-Micro System. This permits the user to configure the operating parameters for data acquisition without becoming involved in the time-consuming specifics related to programming the IDS; also, the IDS is relieved of storing all software except that critical to its real-time operation. Additional memory is included for temporary storage of data before transmission to the minicomputer. Also, for calibrating,

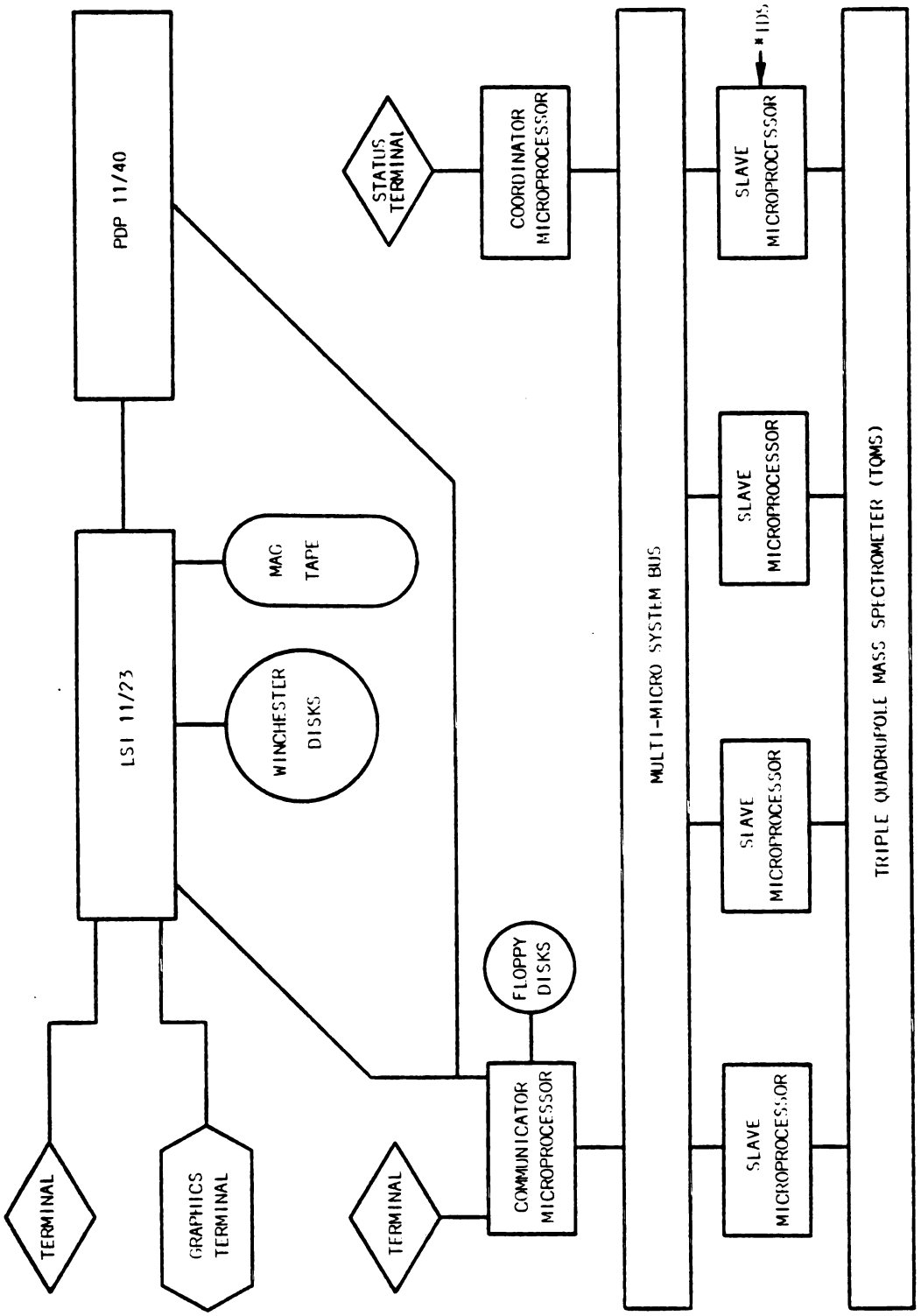


Figure 2.2 TQMS Automated Control Strucure [10]

testing, and debugging purposes, specialized hardware and firmware are included in the design of the IDS. Chapters 3 and 4 examine the specifics of IDS hardware and software.

2.3 OTHER CONSIDERATIONS

Several other considerations affect the design of the IDS. Though all of the following are significant, they are simply satisfied through the selection of proper components and/or through careful applications of standard techniques.

2.3.1 Output Representations

Binary-coded representations of data collected by the IDS are required by the TQMS data processing/storage system. Since acquisition speed is a primary consideration in IDS design, most data processing is carried out by the 11/23 minicomputer, i.e. the IDS output is an integration time and this output is correlated with preset IDS operating parameters by the 11/23 to determine the actual ion flux. For development purposes, the IDS software design provides for on-board flux calculations and formating. Analog representations are also required for monitoring and auxiliary analog data recording (x-y plotter).

2.3.2 Noise

The charge carried by an individual singly charged ion is very small (approximately 1.6×10^{-19} coulombs). The high gain supplied by the electron multiplier still provides only a small output current (no larger than 1 uA). Subsequent processing electronics are designed to guard against unwanted noise sources through the careful application of standard shielding and grounding techniques.

Care is also taken in protecting digital signals from unwanted transients or other disturbances. Also, since some digital as well as analog signals are transmitted over relatively large distances (the reasons for this are discussed below), provisions for the proper driving and receiving of these signals are made.

2.3.3 Physical Deployment

Only two restrictions exist upon the physical deployment of the IDS. One is that the electron multiplier reside in the vacuum chamber of the TQMS. The other is that the initial amplification electronics be located as near as possible to the multiplier output so as to minimize noise and transmission losses.

The remainder of the IDS may be situated as practicality suggests. Where portions of it are separated,

transmission/reception circuitry must be included to insure reliable distanced communication.

2.3.4 Stability and Reliability

Though the enhancement of IDS stability and reliability are considerations left primarily to a second-generation IDS design, these two characteristics have not been totally disregarded. The selection of IC's, components, connectors, mounting and housing hardware, is significantly based on these elements' ability to provide stable and reliable operation against the effects of time, temperature, electrical, and environment.

2.3.5 Cost

Careful analysis of the design criteria developed in this chapter and of the available technological tools demonstrates that the IDS offers a substantial cost-performance benefit over other types of detectors. The key to the realization of this benefit lies in the proper selection and configuration of reasonably-priced hardware and software components. The specifics of the implementation chosen are detailed in the next two chapters.

*** CHAPTER 3 --- HARDWARE ***

The IDS hardware design allows the user to optimize various parameters associated with data acquisition through a programmable approach. The chosen implementation provides the user with ion flux data in a variety of forms dependent on the mode of IDS operation.

A top-down description is used in this chapter; the IDS is viewed first as a few general function blocks which are subsequently developed as more detailed sub-units. The following chapter on software provides additional perspective related to the threading of these sub-units.

3.1 HARDWARE OVERVIEW

The IDS is designed as three distinct functional units; this facilitates the development, testing, and conceptualization of the IDS. These units are titled: the detection unit (DU), the processing unit (PU), and the control unit (CU). Figure 3.1 presents a block diagram of the IDS showing the interconnection of its 3 main units. Interfacing specifics are detailed within appropriate sections.

Central to the operation of the IDS is the Intel 8085A microprocessor [11]. Though any type of general purpose microprocessor may suffice, the 8085 is chosen for its

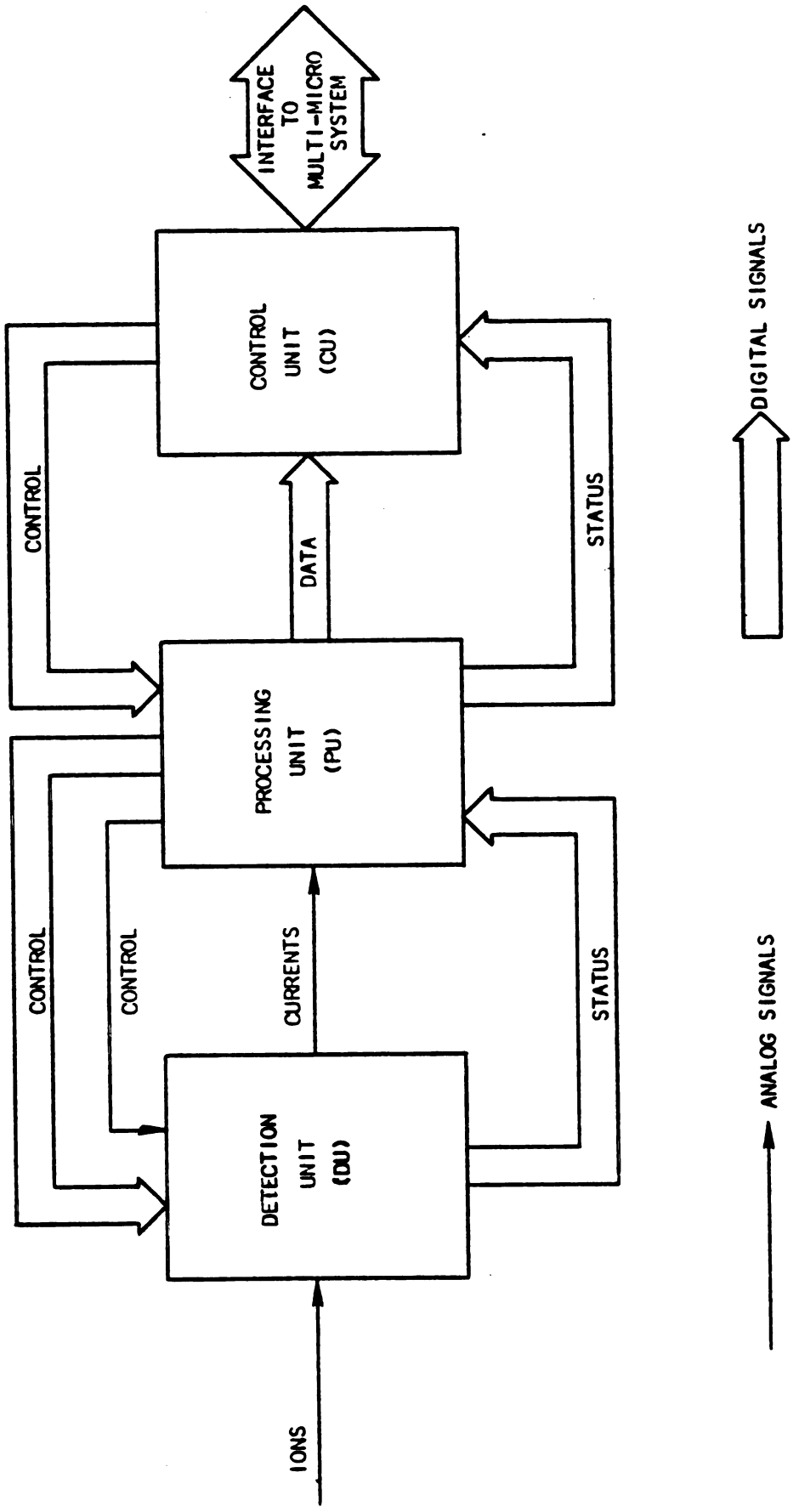


Figure 3.1 Ion Detection System (IDS) block diagram

compatibility with the TQMS Multi-Micro System (which is also 8085-based). This compatability also permits sharing of high-level software and development tools, thus reducing overhead costs.

The electronic components are mounted on several two-sided printed circuit board modules. These modules, except for the initial amplification and conversion stages (which are stand-alone boards), are soldered on several motherboards, which in turn are connected through backplanes or through ribbon cables. This type of construction enhances reliability while permitting ease in development and testing.

3.2 DETECTION UNIT (DU)

The DU is the input stage for the IDS. Ions generated and selected by the TQMS source and mass analyzer are fed to the DU. The DU converts the input ion flux into pulse or analog current for subsequent processing by the PU. Control and status inputs and outputs are also required for protection of the DU components. A block diagram of the DU is given in Figure 3.2.

At the heart of the DU is a new, enhanced type of electron multiplier: the Dual-Mode Channeltron™ recently developed by the Galileo Corp [12]. This type of electron multiplier offers expanded dynamic range, 10^0 to 10^9, without any run-time gain scaling.

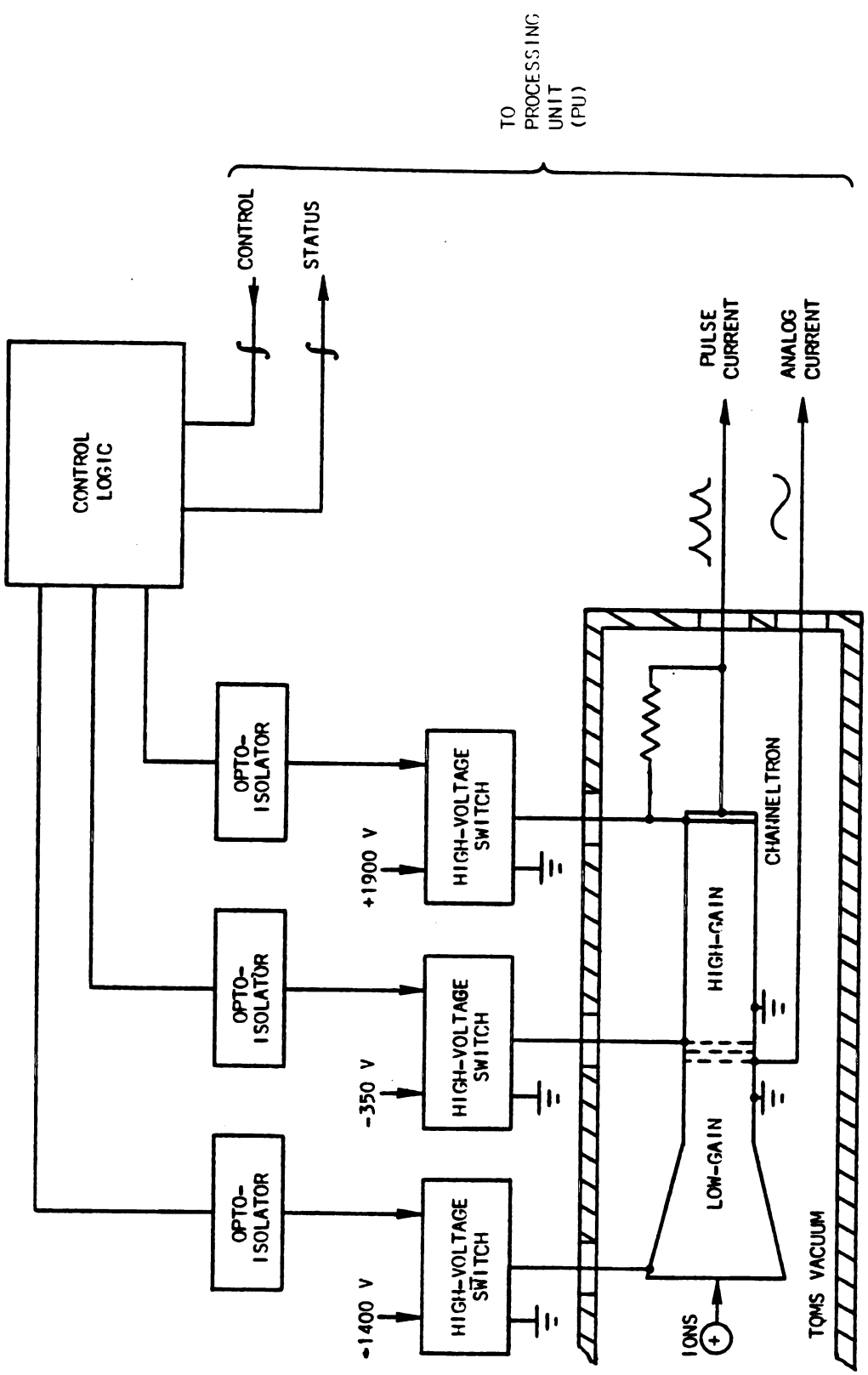


Figure 3.2 Detection Unit (DU) block diagram

Support circuitry and specialized housing comprise the remainder of the DU. Detailed discussion on the DU hardware follows.

3.2.1 Dual-Mode Channeltron™

A Channeltron™ is an electron-multiplying device which is constructed with a single continuous dynode (this type of multiplier is more stable and reliable than discrete dynode types). Depending on the input ion flux and the gain setting, a pulse or analog current is output. Standard Channeltrons™ provide a single output; processing circuitry must multiplex this output if both pulse and analog techniques are required. Gain adjustments are required to expand the dynamic range beyond 10^3 ips.

The Dual-Mode Channeltron™ provides distinct output channels for pulse and analog currents. This Channeltron™ requires an extra power supply (see Figure 3.2) to provide gain for a separate high-gain section, but no voltage ranging is required for either gain section in order to achieve full dynamic range. As a result, the limiting factor in IDS data acquisition speed and range is no longer the electron multiplier.

For brevity, the Dual-Mode Channeltron™ is henceforth referred to as the "Channeltron". Specifications on Channeltron performance are given in Figure 3.3. For ion flux levels up to 10^6 ips, pulse outputs are valid. Above

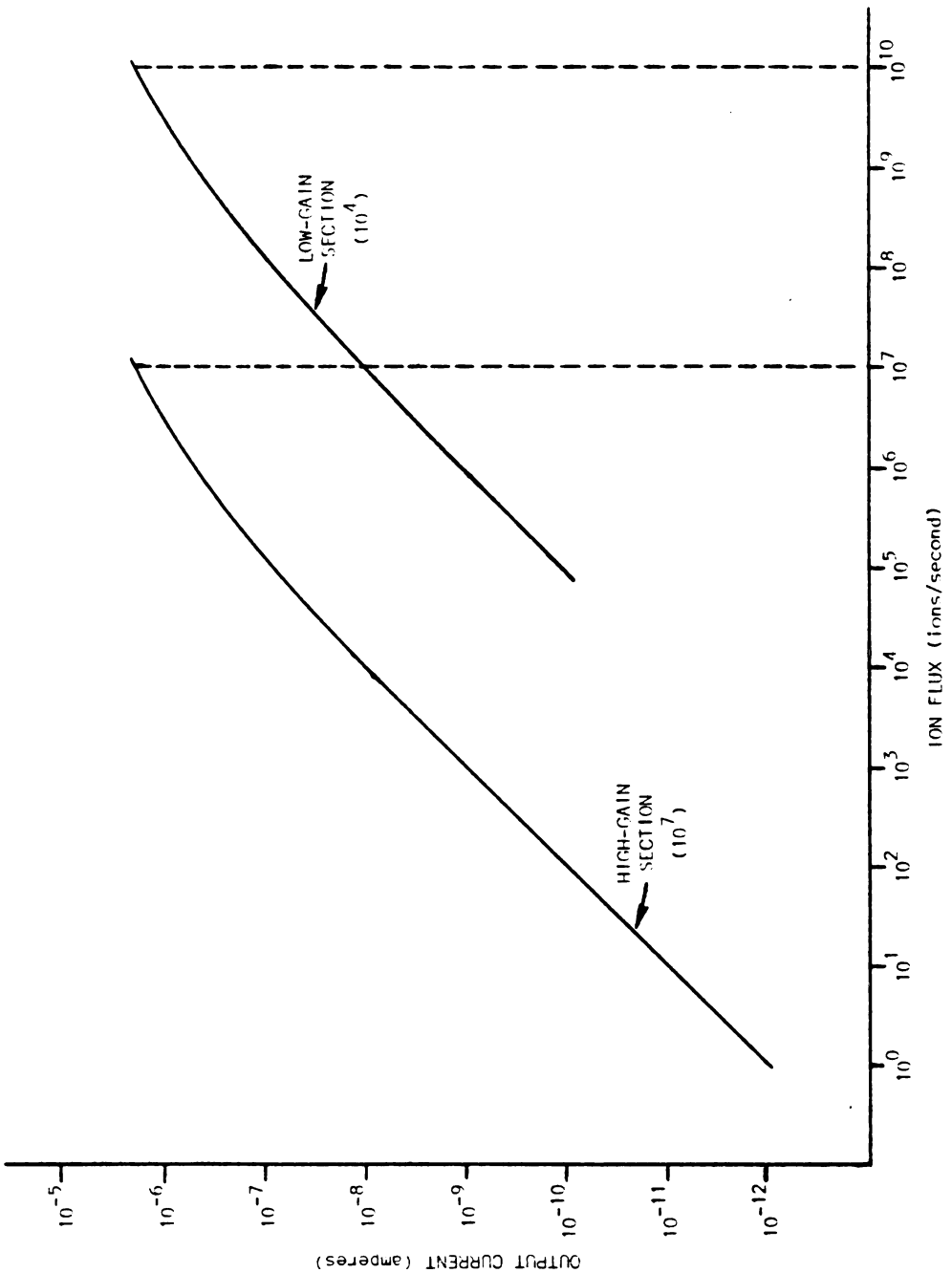


Figure 3.3 Dual-Mode Channeltron Characteristics [12]

this level, the high-gain (10^7) section of the Channeltron may saturate; a protection grid within the Channeltron is activated to prevent this. At flux levels above 10^9 ips the low-gain (10^4) section may saturate; gain (voltage) is removed in this case. Flux levels below 10^5 ips are most efficiently measured by pulse-counting techniques.

3.2.2 Support Circuitry

The Channeltron receives its gain from voltage potentials applied across its highly resistive dynode. The low-gain section requires a voltage of approximately -1400 V, the high-gain section requires +1900 V, and the protection grid requires +350 V, (if in the "protect" state; grounded if not). These voltages are supplied from three commercial high-voltage sources.

Special precautions are taken in order to assure safe Channeltron operation. If either section of the Channeltron is forced to saturate as a result of input overdrive, temporary and/or permanent degradation of operating characteristics may occur [13]. Also, if the vacuum pressure in the TQMS ever drops below a certain level, the Channeltron would be destroyed by arcing across its high-potentials.

The circuitry used to implement these precautions is shown in Figure 3.4. Because the high-voltage supply requires several minutes to bring its output to ground, a

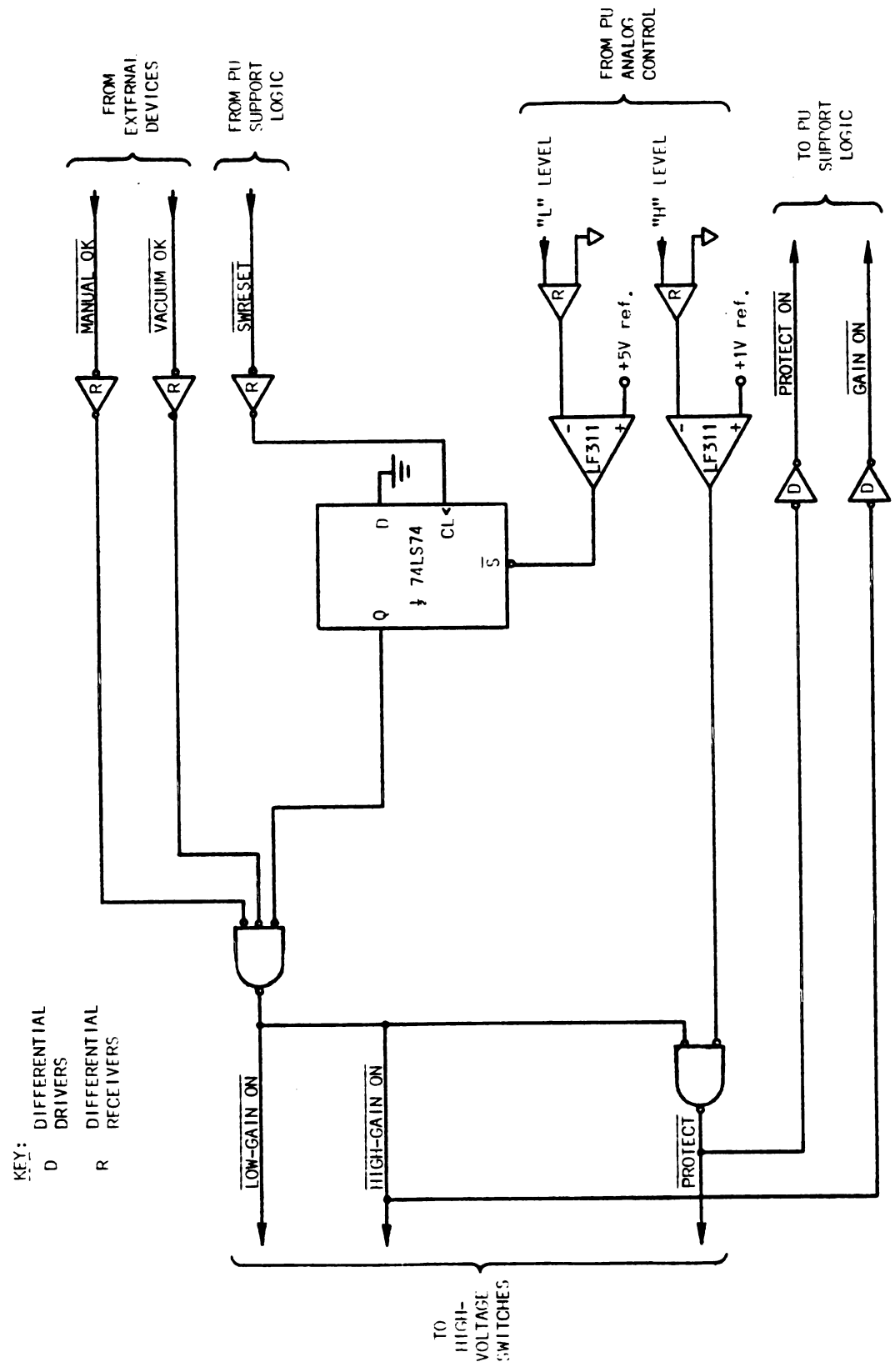


Figure 3.4 Control circuit for Detection Unit (DU)

pair of high-voltage switches are needed to quickly cut-off (or turn-on) the supply voltages to the Channeltron. These switches are isolated from their control inputs, since high-voltage or large-current feedback through these inputs could damage other portions of the IDS; opto-isolators are incorporated as preventions.

Several inputs are used to control the DU high-voltage switches. Feedback of the analog output of the Channeltron, in the form of scaled voltages, is used to detect and protect against Channeltron saturation. Two such inputs, signifying low-gain and high-gain section saturation, are supplied via the PU (Processing Unit); a Channeltron output current of 10 nA is the high-gain section cut-off level, and a current of .5 uA is the overall cut-off threshold. Voltage levels of 1 V and 5 V, respectively, represent these thresholds. These voltages are fed to comparators (with hysterises) which test them against reference voltages (see Figure 3.4). When input "H" exceeds 1 V, a +350 V signal is applied to the protection grid input of the high-gain section. When input "H" drops below 0.9 V, the protection grid input is returned to ground, permitting electrons to enter the high-gain section. Comparator "L", which controls application of the -1400 V and +1900 V used to provide Channeltron gain, is referenced to 5 V. If input "L" exceeds 5 V, the gain voltage inputs of the Channeltron are brought to ground, shutting down the Channeltron operation.

A command from the TQMS user is required to reactivate the Channeltron.

Other control inputs come from the CU (Control Unit) and from manual switches. The on/off state of the high-voltages are controlled through these inputs for purposes of restart, calibration, or testing. Manual, CU, and feedback control signals are all required to be "OK" before the Channeltron will operate.

The DU also supplies two differentially driven outputs which signal the state of the high-voltage switches. These signals inform the other IDS units which channel(s) of the DU, if either, is providing valid output. All support circuitry inputs and outputs are appropriately buffered.

3.2.3 Housing

The Channeltron is mounted inside the TQMS vacuum chamber. Channeltron inputs and outputs are fed through the back-flange of the TQMS to the the remainder of the IDS. The back-flange is specially constructed to permit passage of electrical signals and supplies via BNC, MHV, and SHV connectors, thus permitting simple interface to external circuitry.

The DU support circuitry is housed in a grounded metal box which acts as shielding. Logic circuits are mounted on a printed circuit board and are physically and electrically isolated from the high-voltage switches.

3.3 PROCESSING UNIT

Conversion of the raw analog and pulse currents, supplied by the Channeltron, into data which represent the input ion flux is the primary function of the PU. This data may be in one of several forms depending on the nature of the experiment being performed. These forms are described in detail through the remainder of this section.

Two distinct channels (corresponding to the two types of input supplied to the PU by the Channeltron) form the main body of the PU; they are referred to as the pulse channel and the analog channel. Each channel converts its corresponding type of input into data or control signals required by the IDS. A block diagram of the PU is given in Figure 3.5. Details of the PU design follow.

3.3.1 Pulse Channel Circuitry

Diagrams for the pulse channel are given in Figure 3.6. The pulse channel is composed of an amplifier, a discriminator, and a counter. A commercial pulse-amplifier-discriminator, built by Galileo Corp. especially for Channeltron pulse detection, met the design requirements exactly. This device, the PAD-400 [14], is capable of resolving non-random Channeltron output pulses at rates up to 8 MHz. According to statistics associated with the counting of random events, the resolving time of the

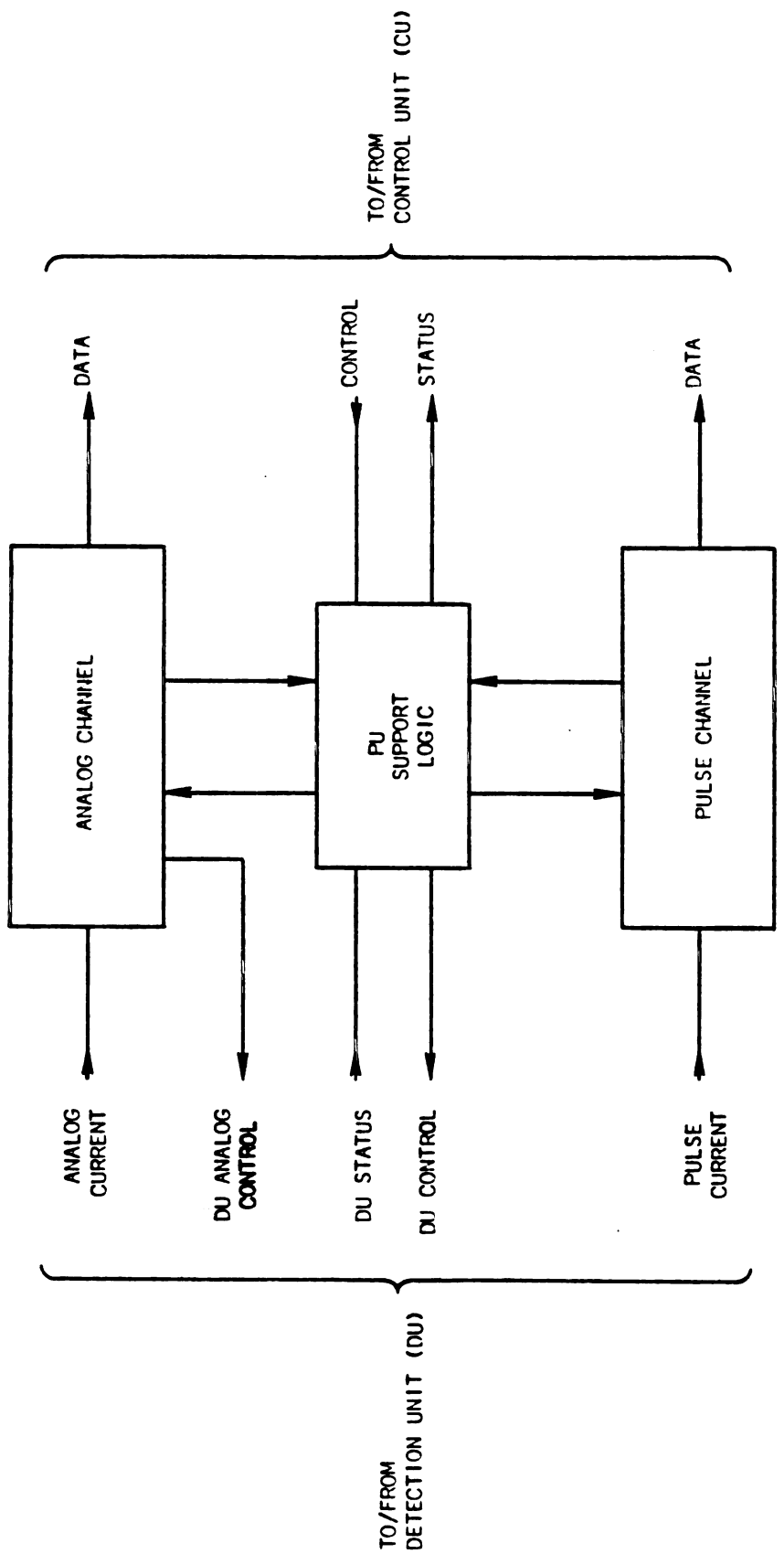


Figure 3.5 Processing Unit (PU) block diagram

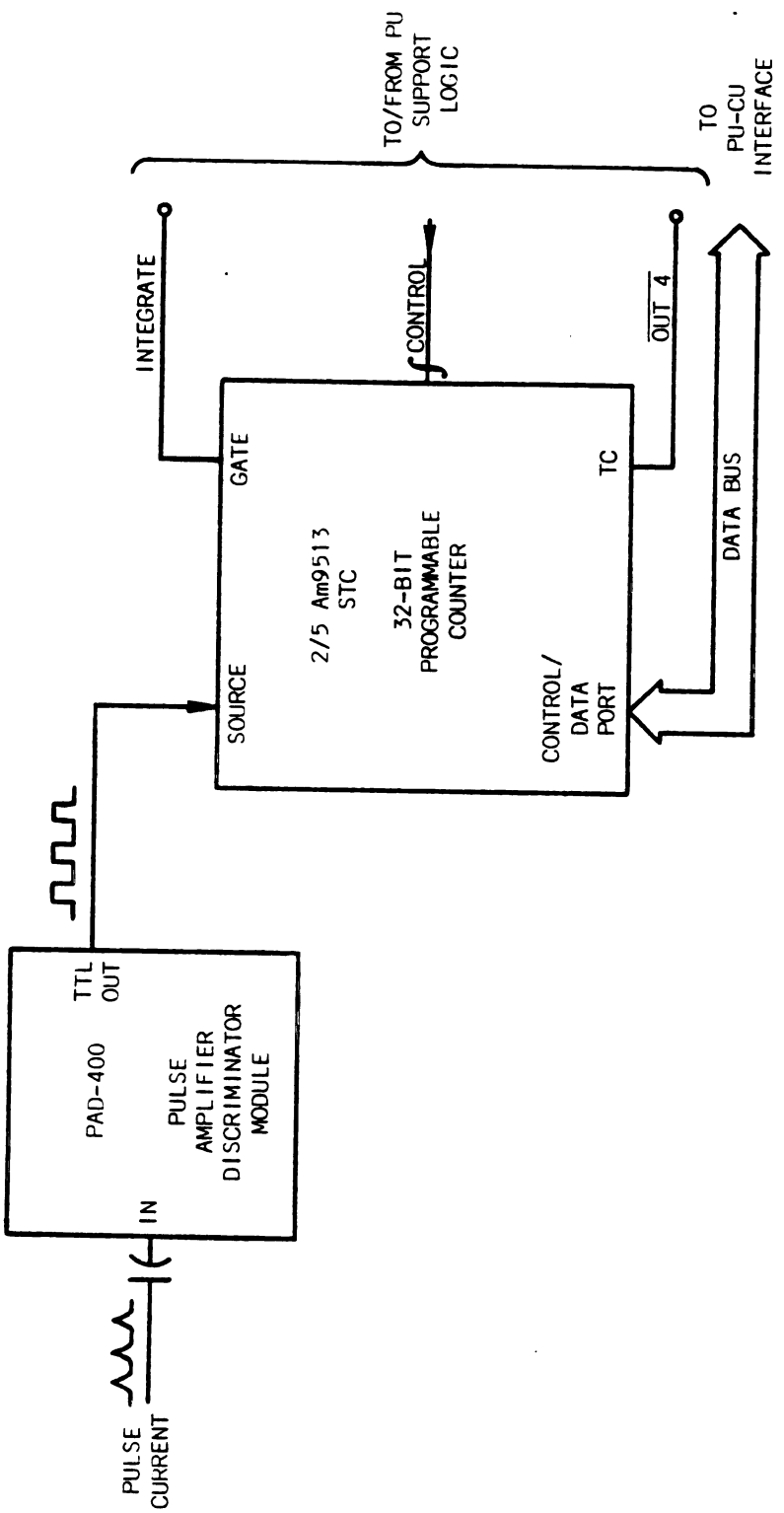


Figure 3.6 Pulse Channel Circuitry

counter and the maximum average count rate determine the probability of missing a count due to pulse overlap [15]. The probability, P(n), that n pulses occur within the resolving time, td, of a pulse counter is given by the Poisson distribution:

$$P(n) = [(R*td)^n/n!] * \exp(-R*td)$$

where R is the average pulse rate. The probability of two or more pulses occurring within the counter resolving time is:

$$P(2+) = 1 - P(0) - P(1)$$

It is desirable that average pulse rates up to 1 MHz be counted to within 1% relative precision (this provides an operational overlap of both channels, necessary for on-line calibration). Application of the above equation shows that a 7 MHz (non-random) counter is the required minimum to insure 1% precision.

In order to record up to 10^6 counts, the pulse counter also requires sufficient width (minimum of 20 bits). Reduction of both speed and width requirements can be attained through the addition of pre-scaling circuitry.

The counter used in the pulse channel design is part of the programmable multi-counter IC from AMD, the Am9513

System Timer/Counter (STC) [16]. A 32-bit counter within the STC is designated as the pulse counter; the counter operates up to a 7 MHz non-random rate. This IC is also used by both channels for timing purposes; a separate discussion on the (STC) is reserved for a later section.

3.3.2 Analog Channel Overview

The analog channel design requires some type of analog-to-digital conversion: input currents ranging from 10^{-10} to 10^{-6} A represent the ion flux range from 10^{5} to 10^{9} ips (see Figure 2.1). Though the Channeltron of the DU is not an exact linear amplifier, a linear model is sufficient for the purpose of analog channel design and descriptions. The non-linearity factors are accounted for in down-line CU or TQMS data processing systems.

As mentioned in the section on the DU (Detection Unit), control voltages representing Channeltron saturation are required by the DU. A direct feedback of the information contained in the analog input current is supplied from the PU; the circuitry which generates the control voltages is considered as part of the analog channel.

From analysis of the design constraints, the following conclusions are drawn relative to the analog channel design: First, the time required to collect a single piece of data (an ion flux level) is dependent upon the flux level and the acceptable error in the data (see Figure 2.1). The IDS must

wait for a sufficient number of ions to be detected before the ions per second value is valid; the acquistion throughput is limited by the data itself. Secondly, since the quality of the data may be traded off in favor of fast acquisition (or vice-versa), programmable control of the analog (and pulse) channel is required for IDS optimization.

A block diagram of the analog channel is shown in Figure 3.7. After initial conversion to a voltage, the analog input is directed along three routes. Route one takes the analog signal to a monolithic A/D converter which provides direct conversion of the input. The second route leads to a programmable integrator which, in conjunction with a timer, supplies an integration time representative of the input signal. This time is correlated with other integrator parameters to determine the ion flux. The final route leads to a pair of op-amps which feedback the analog level to the DU comparators to control the Channeltron. The details and the reasoning behind the configuration of the analog channel are explored in the following sections.

3.3.3 Current-to-Voltage Converter

Current input to the analog channel will range from approximately 10^{-10} to 5×10^{-7} A. In order to facilitate subsequent analog processing, a current-to-voltage conversion is required. A standard op-amp circuit is used. The converter is set to provide an output range from 1 mV to

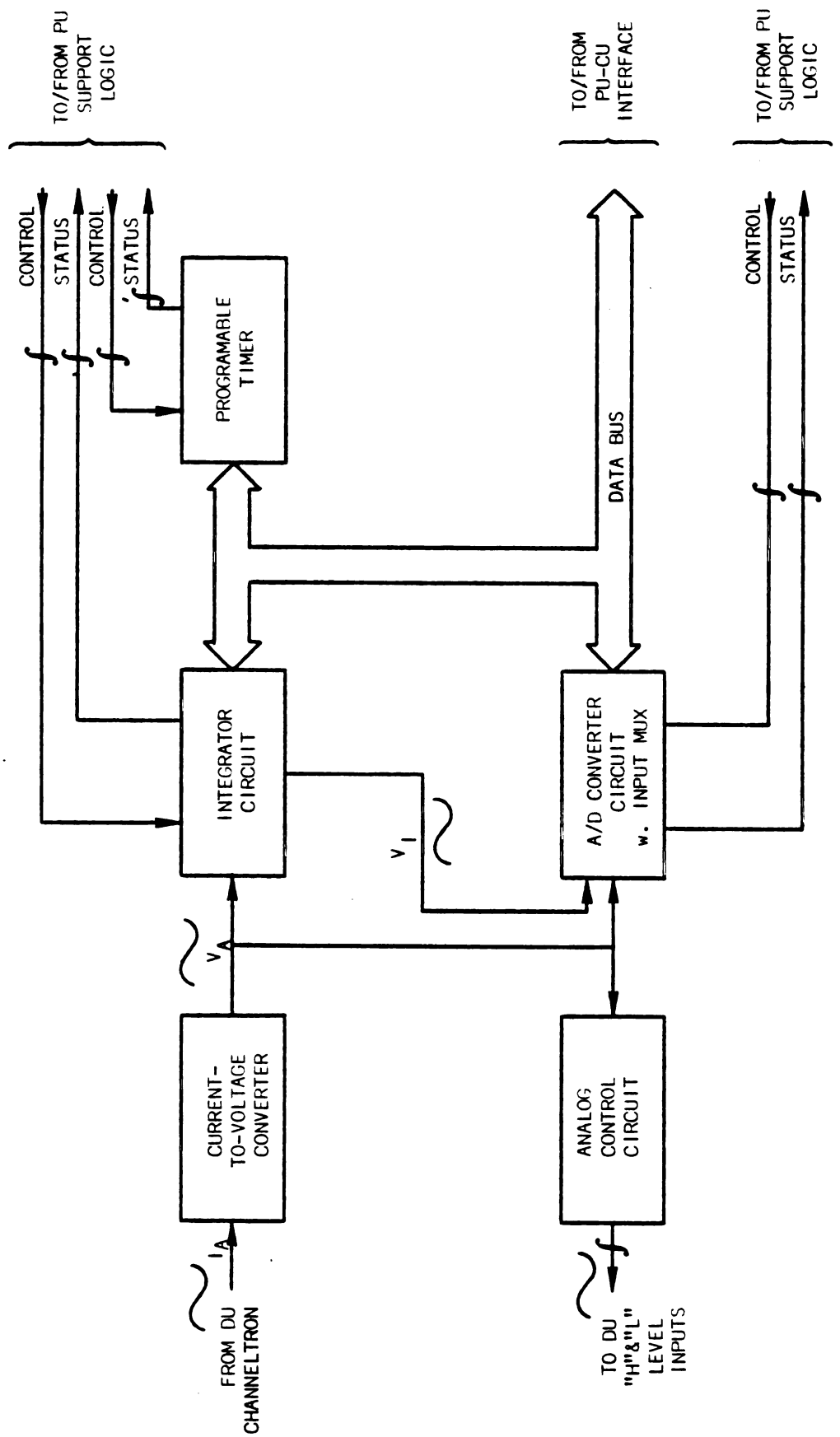


Figure 3.7 Analog Channel block diagram

5 V. Since even slight bias currents and offset voltages would be catastrophic to the functioning of this circuit, special components and construction are required.

Although an instrumentation type of op-amp could accurately provide the required output, all presently lack the slew rate necessary for high-speed IDS operation. A chopper-stabilized op-amp, the ICL7650 from Intersil [17], is selected because of its low bias, low offset, virtually nill drift, low power consumption, good slew rate, and low cost.

The disadvantages in using the 7650 are not significant, but must be compensated for in the remainder of the analog channel design. Operating voltages are limited to ±8 V, extra capacitors are required for the chopper, slight output noise is generated by the chopper action, and output impedance is approximately 1KΩ . Also, a capacitor across the feedback network is required to insure stability.

The op-amp inputs are surrounded by a guard ring to prevent current leakage from the supply pins. Also, since the feedback resistor is of a very large value (10MΩ), the resistor's body is enclosed in a separate shield in order to reduce noise pick-up. Figure 3.8 gives the circuit.

3.3.4 Integrator Circuit

The voltage output of the C/V converter is proportional to the analog current output of the Channeltron, which is in

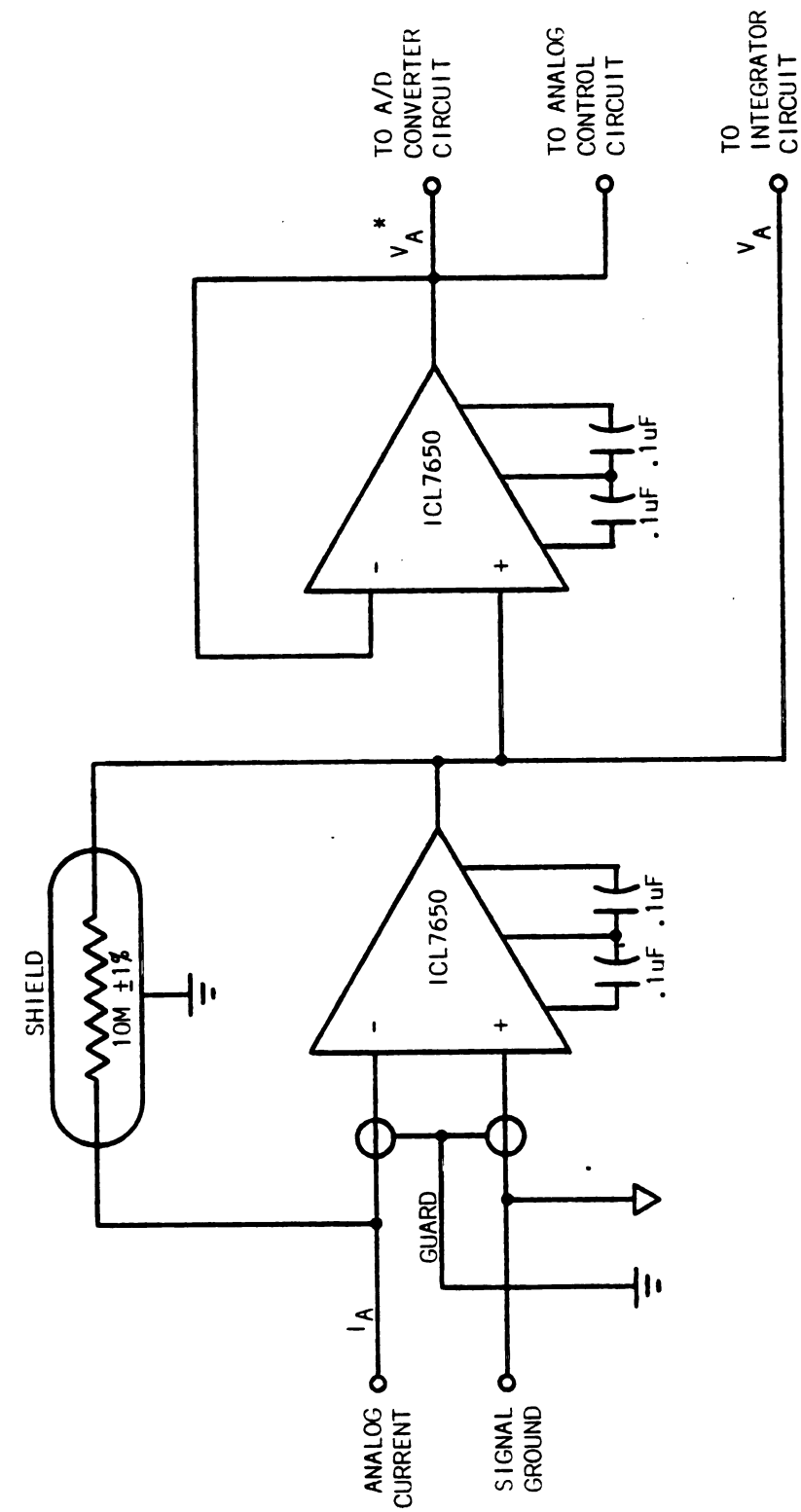


Figure 3.8 Current-to-Voltage (C/V) Converter Circuit

turn proportional to the ion flux. By applying the technique of single-slope integration, the flux level can be calculated from the time required to integrate this voltage to a reference value.

Figure 3.9 shows the integrator circuit. An alternative way of conceptualizing the operation of this circuitry is as follows: the Channeltron, the C/V converter, and the input resistor of the integrator form an ion-to-charge converter (assume that this converter is perfectly linear and has zero response time); the current input to the summing node of the integrator is a collection of discrete charge packets, each equivalent to a single ion. The feedback capacitor of the integrator acts as a charge counter whose output is a linearly proportional voltage. As shown in Section 2.1.2, for a specified sampling error only a minimum number of ions (or charge packets) need be counted; this "count" is divided by the total time of integration to determine the ion flux. A comparator signals a completed integration when the capacitor (charge counter) voltage has reached a reference voltage representative of the required minimum count.

As a result, data acquisition proceeds in an asynchronous manner, maximizing throughput. Maximum integration times may also be programmed for cases where throughput requirements outweigh those of accurate sampling of all data. A connection of the integrator output to an

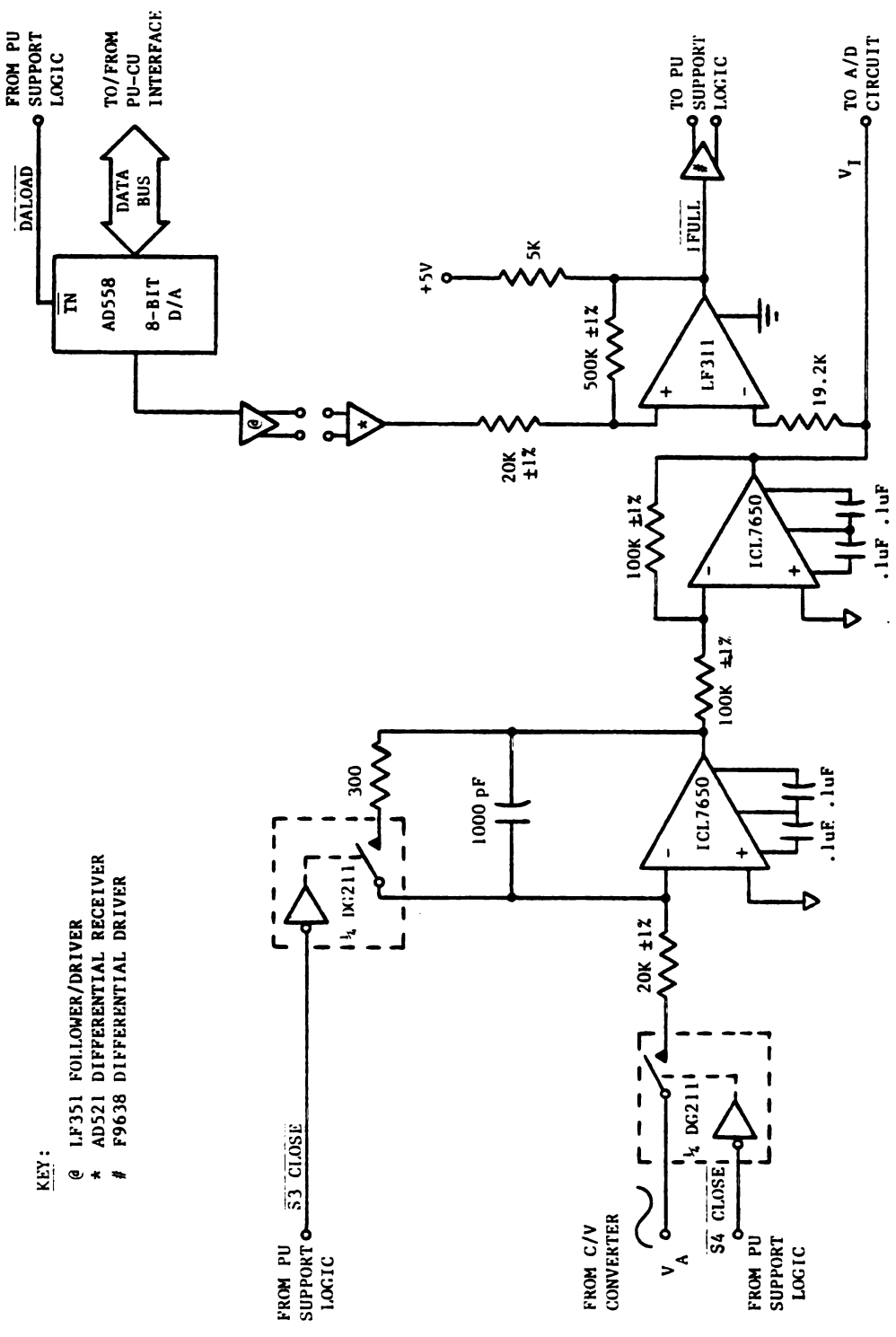


Figure 3.9 Integrator Circuit

A/D converter is provided for evaluation of partially integrated data samples.

Precision components are required. An ICL7650 is used for the integrator's op-amp. Low leakage SPST analog switches, one-half of a DG211 [18], are used. A polystyrene capacitor minimizes dielectric retention and leakage currents. A unity-gain inverter buffers the integrator output to prevent voltage droop. Positive-feedback provides hysteresis for the comparator. A variable voltage reference is supplied by a D/A converter so that the CU (Control Unit) may adjust the analog "count" limit according to the programmed maximum error. Timing is performed by the STC (Am9513 timer/counter, see Section 3.3.6).

The integrator provides the added advantage of a low-pass filter, since input signals above 10KHz (the maximum data collection rate) are considered to be noise.

Calculations to determine the values of the components associated with this circuitry are given in Appendix A. Means for calculating the ion flux from the integration time are undertaken in Section 4.2.3.

3.3.5 A/D Converter

In some types of acquisition modes, speedy data collection is of priority, regardless of the extra error induced by the limited "counting" of random events (ion inputs). Direct measurement of the C/V converter output

voltage is a solution where even partial integrations are too slow.

A 12-bit monolithic A/D converter, the AD574 [19], performs this function. In order to provide measurement of both the C/V converter and the integrator output, these voltages are multiplexed to the A/D input; the CU supplies convert and read strobes and controls the multiplexer. Tri-state drivers buffer the output. The complete A/D circuit is shown in Figure 3.10.

An A/D conversion is also triggered upon completion of an integration; this allows the CU to perform a calibration between the reference and integrator voltages, if desired. Also, CU monitoring of the C/V converter output is useful in noise studies and in IDS development.

3.3.6 System Timer/Counter (STC)

The Am9513 System Timer/Counter (STC) times the integration periods for both analog and pulse channels, and also counts pulses for the pulse channel. Five individually programmable 16-bit counters are contained in it. These registers may be concatenated and can use a variety of gates, sources, active logic inputs, and be operated in a variety of modes.

Figure 3.11 shows an abridged block diagram of the STC internal structure as configured for this application. A 32-bit register is used for integration timing; a second

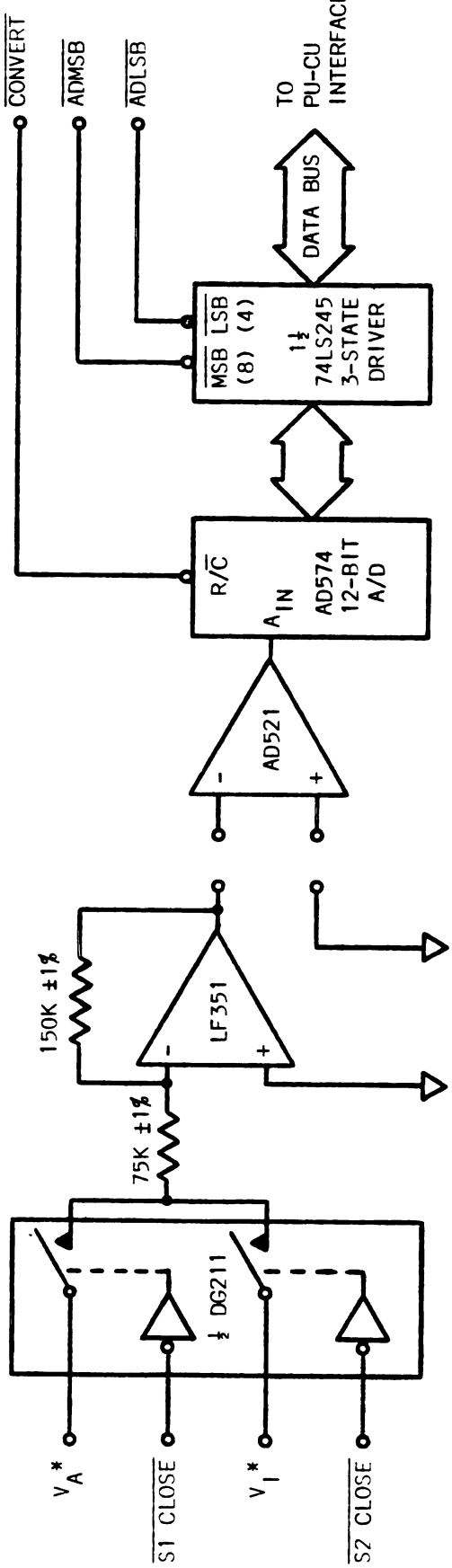


Figure 3.10 A/D Converter Circuit

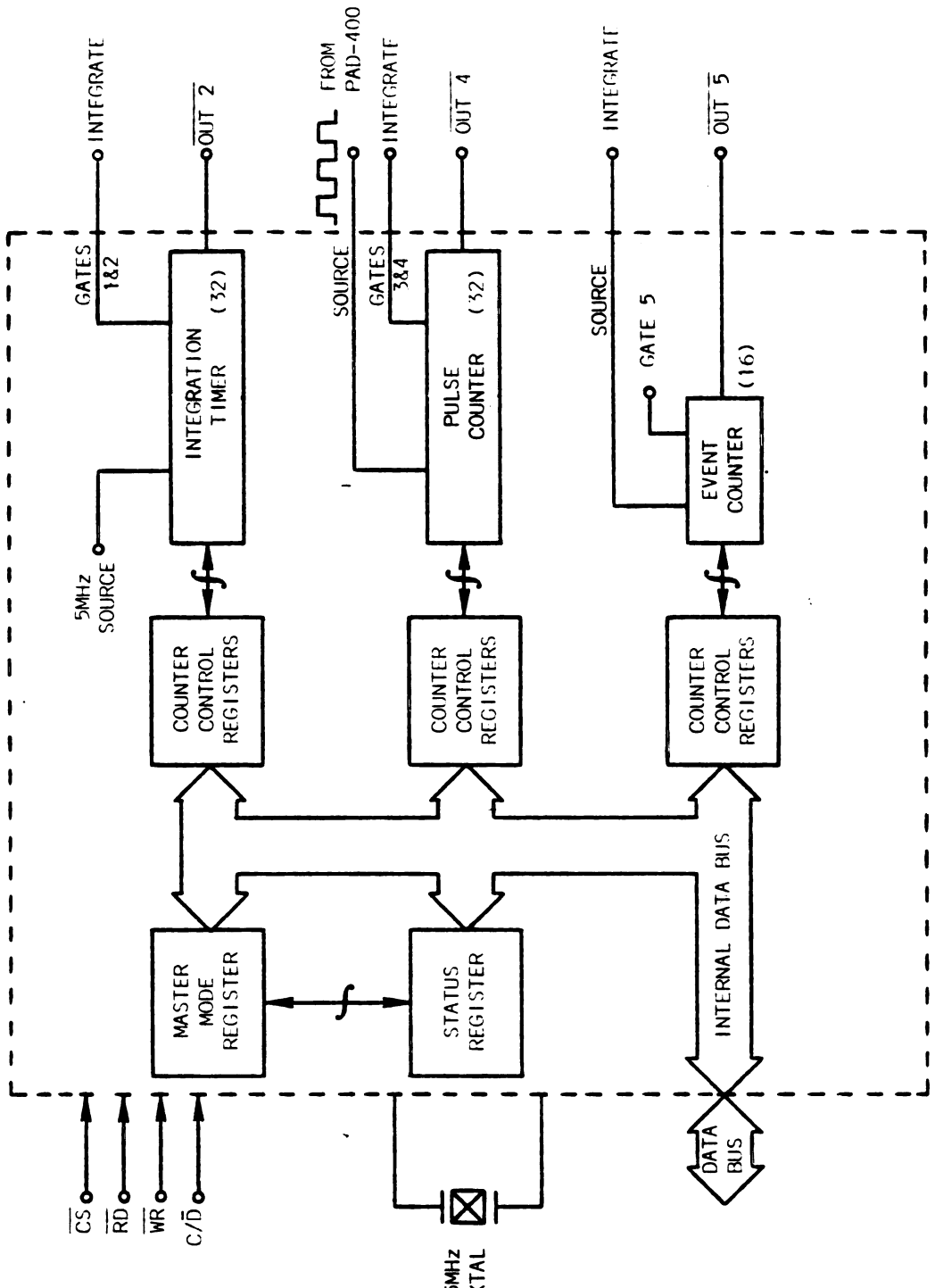


Figure 3.11 System Timer/Counter (STC) configuration

32-bit register for pulse counting; and a 16-bit register for general event-counting. Programming and data retrieval for the STC is directed by the CU. Further information regarding the structuring and programming of the STC will be presented in chapter 4.

3.3.7 Analog Control Outputs

Feedback of the analog current output is supplied to the DU for controlling the operation of the Channeltron. As discussed in Section 3.2.2, signals actuating the shutdown of the gain sections are required.

The "H" amplifier outputs a 100x version of the voltage supplied by the C/V converter. An input of 10 mV (corresponding to the high-gain section overdrive level) produces a 1 V output: this activates the protection grid of the Channeltron. The "H" amp output is limited by a zener diode.

The "L" amplifier is a simple follower. If the C/V converter output reaches 5 V, shutdown of the entire Channeltron follows; restart must be explicitly commanded by the user. Both amplifiers outputs are differentially driven to the DU (Detection Unit).

3.3.8 Support Logic

The PU receives commands from the CU and status information from the DU, which properly coordinates execution of PU functions. The PU also provides various status signals to the CU. These signals are supplied and received through a potpourri of decoders, flip-flops, and gates, collectively referred to as the PU support logic.

The logic diagrams are presented in Figures 3.12 and 3.13. CU commands are sent as reads/writes to specified memory locations (memory mapped). The PU support logic decodes the address information, and except for the STC and A/D output (which require use of the CU data bus), it ignores the dummy data while providing strobe signals through the PU. Flip-flops provide control of the PU analog switches, and signal the CU with interrupt requests. A monitor port, through which the CU can read PU and DU status signals, exists as the lower 4 bits of the A/D output drivers (differential inputs are required).

Actuation of most PU functions requires direct commands from the CU. Several exceptions occur: when the integrator enters the "hold" state upon triggering of its associated comparator; while integrating, a "terminal count" (TC) signal is issued by the STC, signifying that the sampling time limit has been reached; upon a power-up reset supplied by a resistor-capacitor-schmitt- triggered gate.

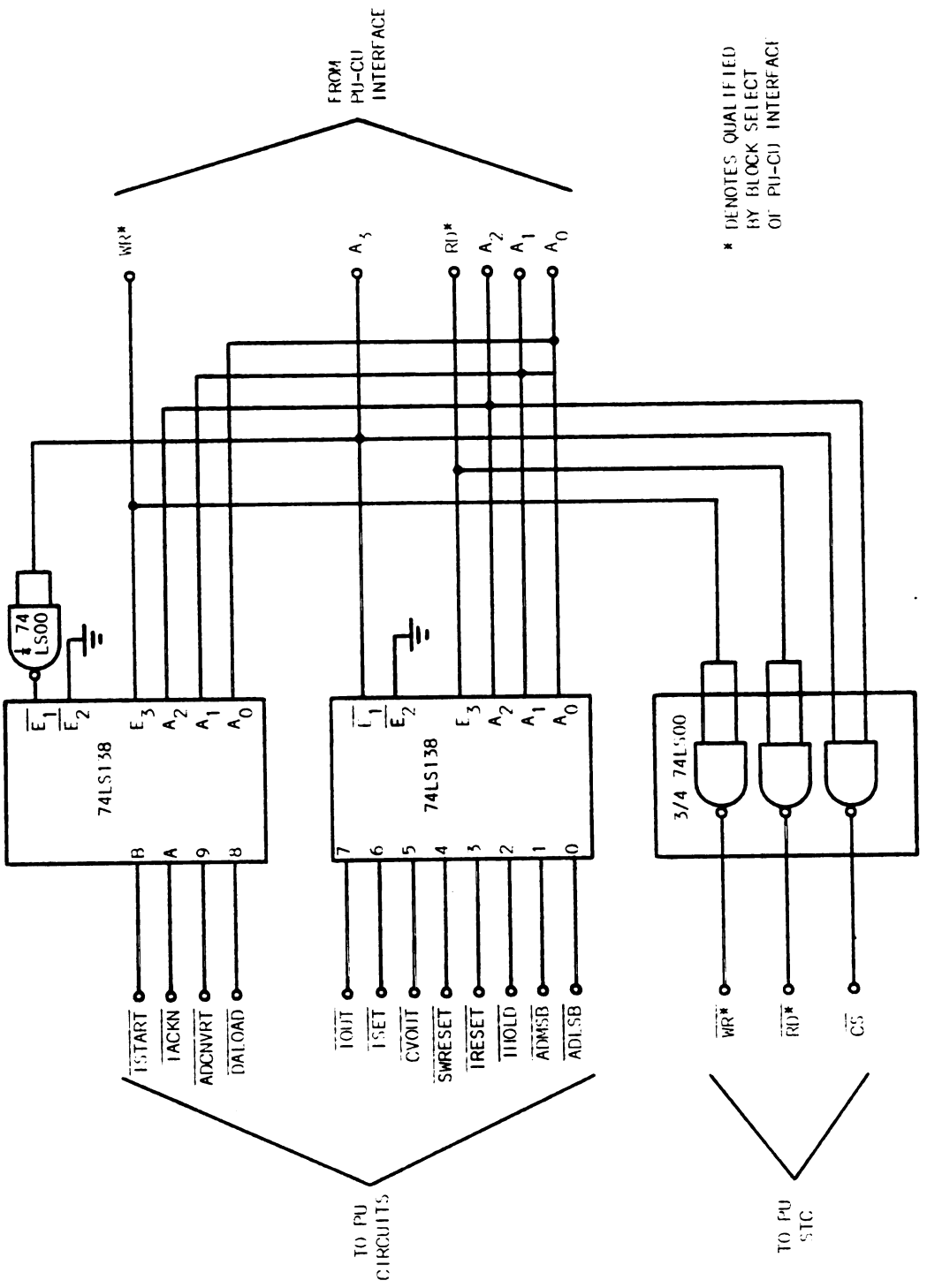


Figure 3.12 PU Support Logic --- Local Decoding Circuit

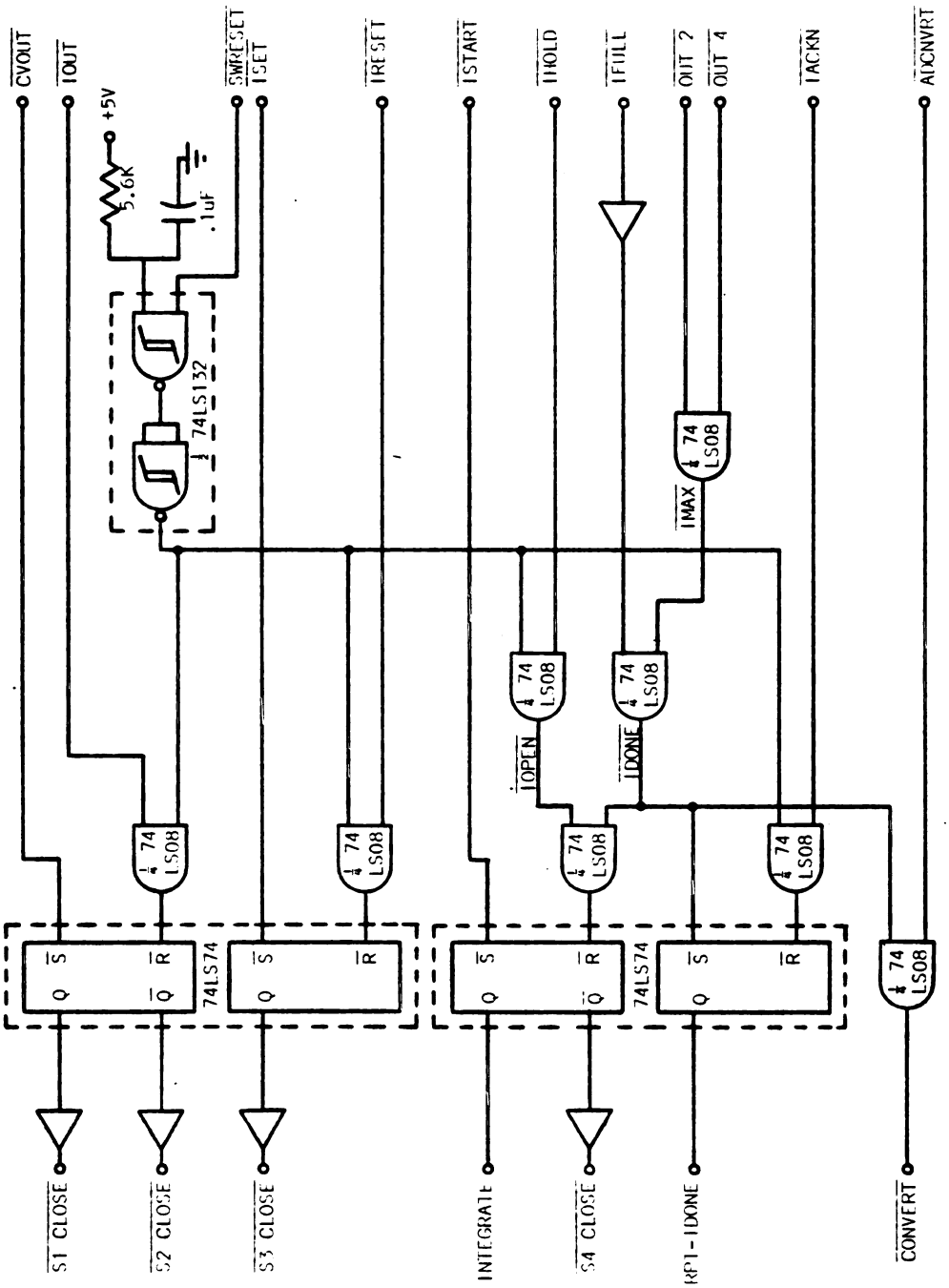


Figure 3.13 PU Support Logic --- Control Circuit

Further information relating the control scheme of the PU is provided in Chapter 4.

3.3.9 Miscellaneous Circuitry

The PU requires interfacing to a group of CU lines. These include the data bus, four address lines (A3-A0), read and write, and two interrupts. Differential transceivers, configured in one and two-way modes, drive/receive these signals. An identical set of transceivers is included on the CU end.

Regulated DC power of several voltages is required for PU operation: +15, -15, +8, -8, and +5 V. Standard LM320/LM340 [20] series IC voltage regulators are used. Regulator inputs are supplied from regulated +24 and -24 V DC sources. Heat sinks are attached to the regulators.

The PU is constructed of three physically separable units: an analog board ("A" board), a digital board ("D" board), and the PAD-400. The "A" board is of single-piece construction with ground plane covering all unused spaces. Connection of the analog input from the Channeltron is through a coaxial cable which is connected to the TQMS back-flange on one end, and soldered directly to the "A" board on the other. A grounded metal box shields the "A" board. The "D" board is an assembly of small boards ("modules") soldered to a motherboard. PU functions are divided the "A" and "D" boards such that all low-level

signals may be processed in a low-noise environment, and that a minimum number of lines are required to interface the boards. All signals are differentially driven between the boards; connections are made through ribbon cables or twisted pairs. The "A" board is housed next to the back-flange, the "D" board is housed about three feet away in a convenient area.

3.4 CONTROL UNIT

A programmable control unit, the CU, directs the operation of the IDS. The basic functional modules used in the CU were designed by other research group members for implementation within portions of the TQMS Multi-Micro System. Incorporation of these modules into the CU provides the necessary "hooks" for linking the IDS into the Multi-Micro and data processing/storage systems.

The 8085A microprocessor and some of its family of peripheral ICs form the core of the CU. Included in the CU are the 8085A CPU, RAM, EPROM, the 8259-5 Programmable Interrupt Controller, dual 8251A USARTs, address decoding circuitry, active bus terminators, and differential transcievers modules [21]. These modules are constructed on small, two-sided printed circuit boards which are then soldered to one of two motherboards (a single motherboard does not provide enough area). A backplane interconnects the motherboards.

Since specifics of the module designs are not part of the work behind this thesis, discussion of the CU hardware will be limited to operational descriptions. Figure 3.14 presents a block diagram of the CU structure.

3.4.1 Central Processing Unit

The 8085A CPU oscillator is set to operate at its maximum speed, 6.144 MHz. The CPU operates upon programs and data stored in ROM and RAM. In the Multi-Micro environment, the CPU may be driven into a hold state by one of the supervisor microprocessors (for DMA), during which programs are downloaded or data is uploaded.

3.4.2 Memory

The memory module has provisions such that a wide variety of both RAM and ROM chips may be used; proper wiring of a few jumpers is required. Maximum storage capacity of a single RAM/ROM module is 16K bytes.

For initial debugging, a single 2716 EPROM (programmed as a system monitor) served as a hexadecimal machine-code interpreter, with 2K bytes of RAM used for scratchpad. The operational CU memory module is comprised of 12K bytes of ROM and 4K bytes of RAM. ROM (specifically, 2716 EPROM's) is programmed with 8K of standard FORTH language [22] and 4K of customized FORTH commands (programs). (FORTH is a unique

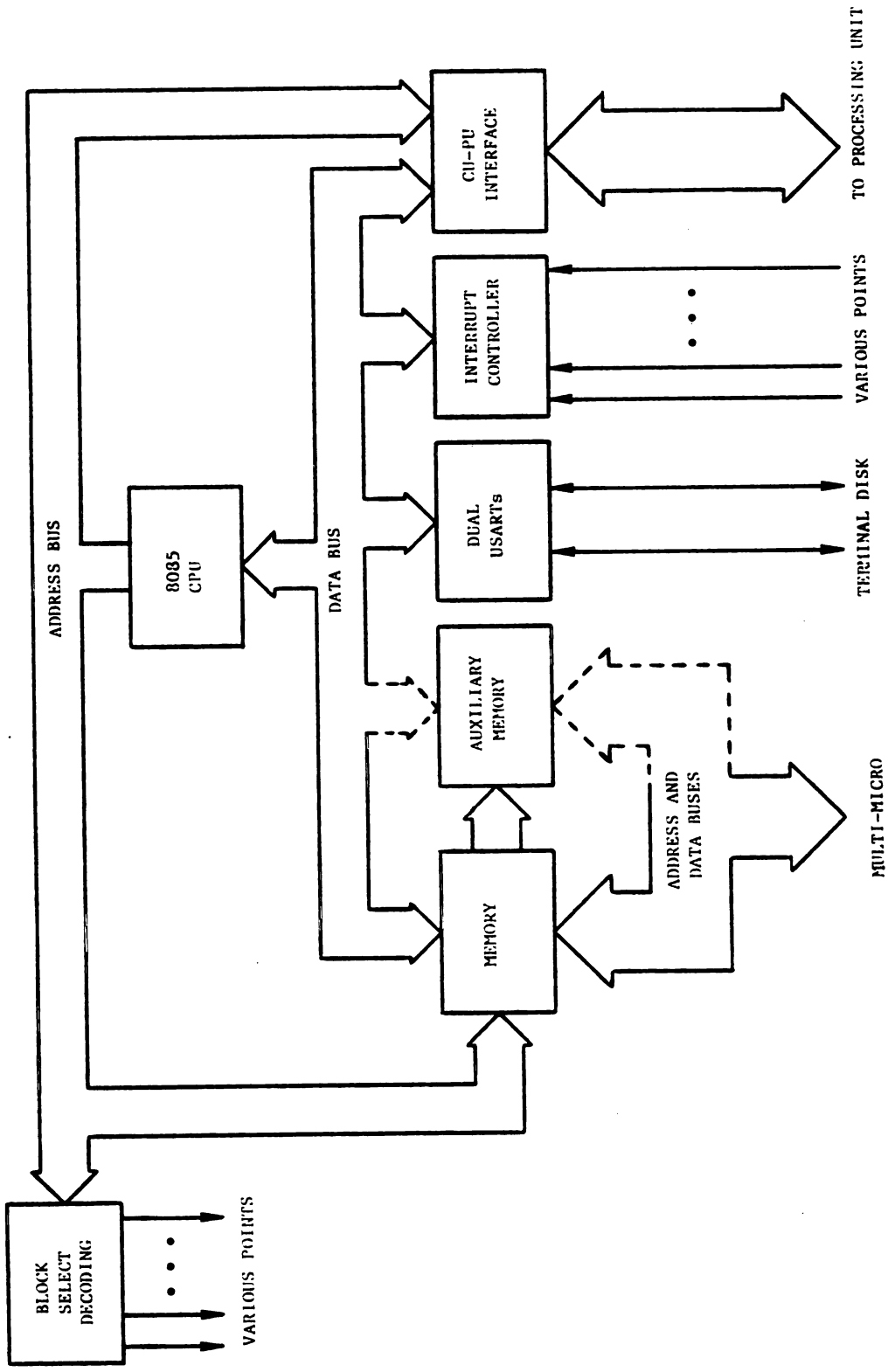


Figure 3.14 Control Unit (CU) block diagram

language, especially designed for real-time microprocessor based control applications; the structure of the ROM programming and of the FORTH language is examined in Chapter 4.) Two 2Kx8 static RAMs form the 4K of RAM.

Memory requirements are subject to variation; finalization of the Multi-Micro System software may increase or decrease them. If more memory is required (such as if large amounts of data must be stored before unloading), provisions in the CU design allow for the simple addition of another 16K memory module.

3.4.3 Block Selects (Address Decoding)

All peripheral and module accessing is performed through a memory map: the required address decoding is performed by the Block Select module. Both read/write qualified and unqualified strobes are available.

Each strobe output corresponds to a block of addresses; only the higher order address lines are decoded. This permits local decoding of lower order address lines by peripheral components. The PU support logic performs this type of local decoding, as do some other modules of the CU.

3.4.4 Dual USARTs

A dual USART module is included to permit interfacing to a development system. During development and testing

phases, one USART connects to a terminal (via RS-232); the other to a floppy disk drive (via a floppy disk controller). Firmware in FORTH ROMs support the development system interface. Within the operational Multi-Micro environment, the USARTs are inactive.

3.4.5 Interrupt Controller

A 8259-5 Programmable Interrupt Controller prioritizes interrupts and vectors the CPU to the appropriate place in memory for the interrupts's coressponding routine. Three external interrupts are required by the CU. The lowest priority interrupt comes from the PU signaling completion of an integration (either by triggering of the comparator or by the timer TC). Next priority belongs to the Multi-Micro system for the purpose of DMA. The highest priority interrupt comes from the DU via PU buffers signaling that Channeltron input overdrive has occurred. This situation will send the CU into a wait loop until a restart command is given.

3.4.6 CU-PU Interface

Differential transceivers are configured in one-way and two-way modes (as on the PU end) for transmission of the data bus, the lower four address lines, read/write, and two interrupt signals.

The standard transciever module permits only the transmission of 16 signal lines, without the inclusion of the block select required by the PU local address decoding. As a solution, the block select line is used to enable/ disable transmissions through the interface, thus eliminating the need for the transmission of the block select.

3.4.7 Multi-Micro Interface

DMA is not the only means of CU/Multi-Micro interaction. Although the hardware associated with this interface is not considered part of the IDS (or of this thesis), a brief description for perspective's sake is appropriate.

The successful operation of the TQMS Multi-Micro System requires a great degree of real-time communication among all segments (source controller, several mass analyzer controllers, vacuum system, the IDS, and the user) of the system. Several types of registers (or ports) are made available to the IDS (and the other segments), which provide Multi-Micro System information. The IDS is programmed to read and write to these registers. For example, the other segments of the Multi-Micro System are able to adjust mass filters in an asynchronous manner corresponding to the IDS's acquisition of data; system throughput is enhanced.

*** CHAPTER 4 --- SOFTWARE ***

Speed, compatibility with Multi-Micro software, compactibility of code, simplicity of use, and development ease are all prime considerations relative to the software design. The key to successful implementation of these requirements is based upon the selection of the proper language(s) for this application.

On-line IDS operation is nothing more than the coordination of independent functions of the hardware; "coordination programs" are supplied through the Multi-Micro System and are experiment-dependent. For clarity (and for consistency with the FORTH language as described later), the sub-routines which form the body of the coordinating programs are stored within IDS firmware and are referred to as "commands". This distinction becomes more apparent as this chapter develops.

This chapter begins with a description of the languages chosen for the IDS, their structures, their relationships to hardware, and the reasons behind their selection. Next, descriptions and classification of the commands are presented. Finally, descriptions of the representations used for encoding parameters and data are given. A glossary of all IDS commands is supplied in Appendix B; a source code listing is provided in Appendix C. A sample

"coordinating program" is given in Chapter 5 in conjunction with the evaluation of IDS performance.

4.1 LANGUAGES

Since flexibility of IDS (and TQMS) operation in a real-time mode is required, even to the point where the user can directly actuate fundamental- level hardware operations, a multi-tiered command structure is necessitated. Code written in 8085 assembly language can provide this capability, but programming time requirements are excessively time-consuming and tedious.

The TQMS designers selected the FORTH language [23,24] as a solution to the entire Multi-Micro programming problem. FORTH is especially designed to operate in a real-time microprocessor-based control environment. Except for a few cases where speed requirements are foremost (assembly language is used here), the entire IDS software is written in FORTH.

4.1.1 FORTH

FORTH is an unusual language, primarily since it is all of the following: a high-level language, an assembly language, an operating system, a set of development tools, and a software design philosophy. FORTH is both a compiling and an interpretive language. FORTH code is very compact;

an application system may be operated with a basic FORTH kernel of less than 1K bytes or with the full-feature version of FORTH which requires only 10K bytes. FORTH code also runs fast, typically one-half the speed of equivalent assembly code on an 8-bit microprocessor (such as the 8085).

The FORTH language is stack based; code is written in post-fix notation. As a result, an initial added difficulty faces the programmer in adapting to writing this style of code. Once this adjustment is made, FORTH programming becomes straightforward.

The entire FORTH language is nothing more than a collection of "words", which upon invocation perform their respective functions. A FORTH application program is simply the creation of a new "word", defined in terms of standard FORTH words. These new words may, in turn, be incorporated into still other new words. Henceforth, these applications-defined words are referred to as the "IDS commands".

Features and functions provided by the FORTH language are alluded to throughout the remainder of this chapter. Further information regarding FORTH can be obtained from a variety of sources, including those supplied by references at the end of this thesis.

4.1.2 8085 Assembly Language

On occasion, data must be acquired at a high rate, possibly the result of limited source sample or inefficient ionization. In such cases, standard FORTH words execute too slowly to perform all necessary operations. A pseudo 8085 assembly language is included within the standard FORTH structure, permitting the coding of high-speed operations.

The assembly subset of FORTH is fundamentally identical to actual assembly language. After invocation of a FORTH word which alters the "vocabulary" (the context in which a word is interpreted), the ensuing code is interpreted as assembly language. The "vocabulary" is returned to FORTH upon interpretation of a special character. An assembly routine is called through invocation of an IDS command, whose definition includes the vocabulary change word and the assembly routine. Only a small percentage of the IDS software requires coding in assembly. FORTH code is also more compact than the equivalent assembly code.

4.1.3 Structure

IDS firmware is loaded with standard FORTH words (8K bytes) and IDS commands (4K bytes). "Coordinating programs" and data are stored in RAM (4K bytes). A memory map for the IDS is given in Figure 4.1. The structure of a coordinating program is given in Figure 4.2.

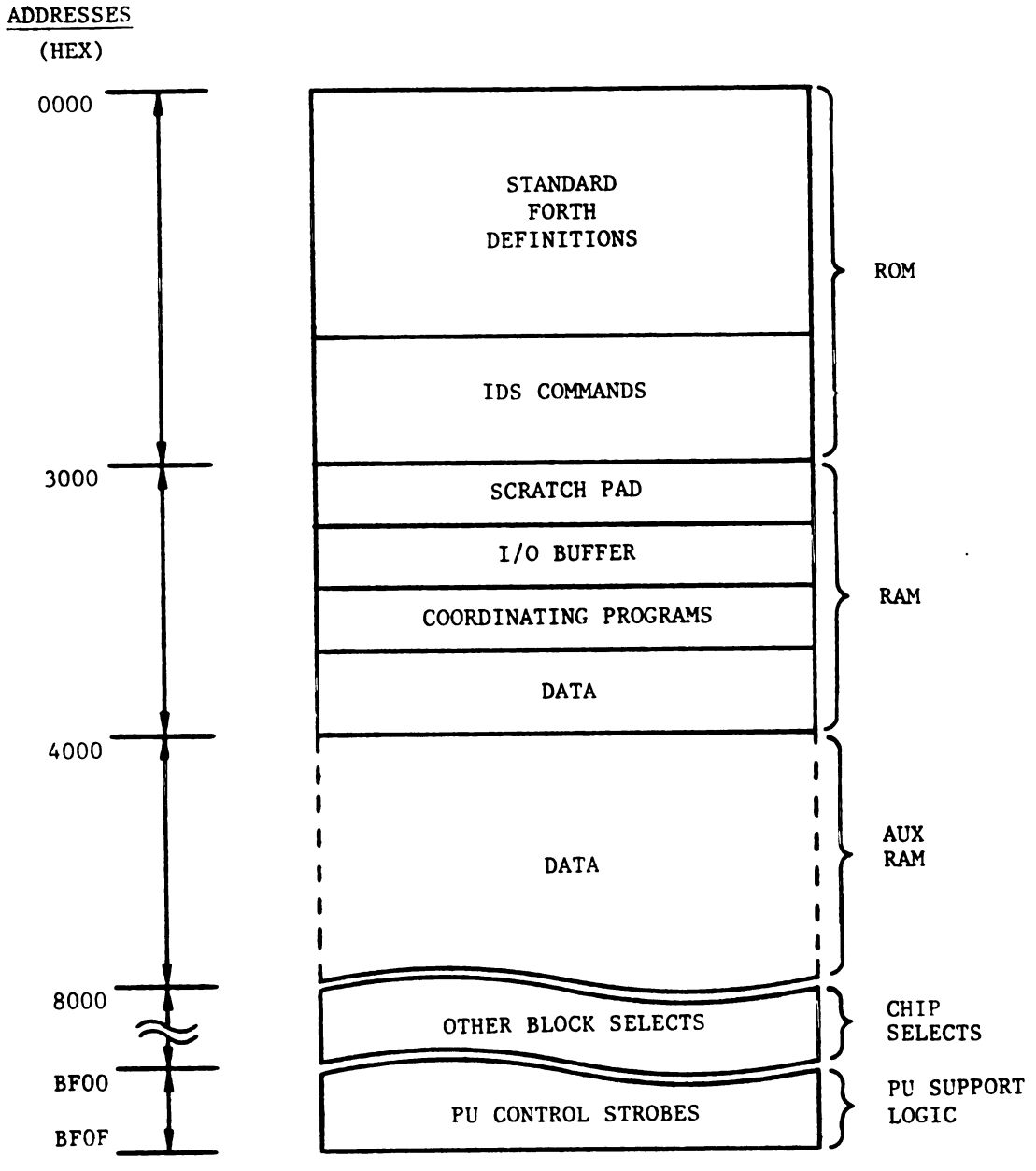


Figure 4.1 IDS Memory Map

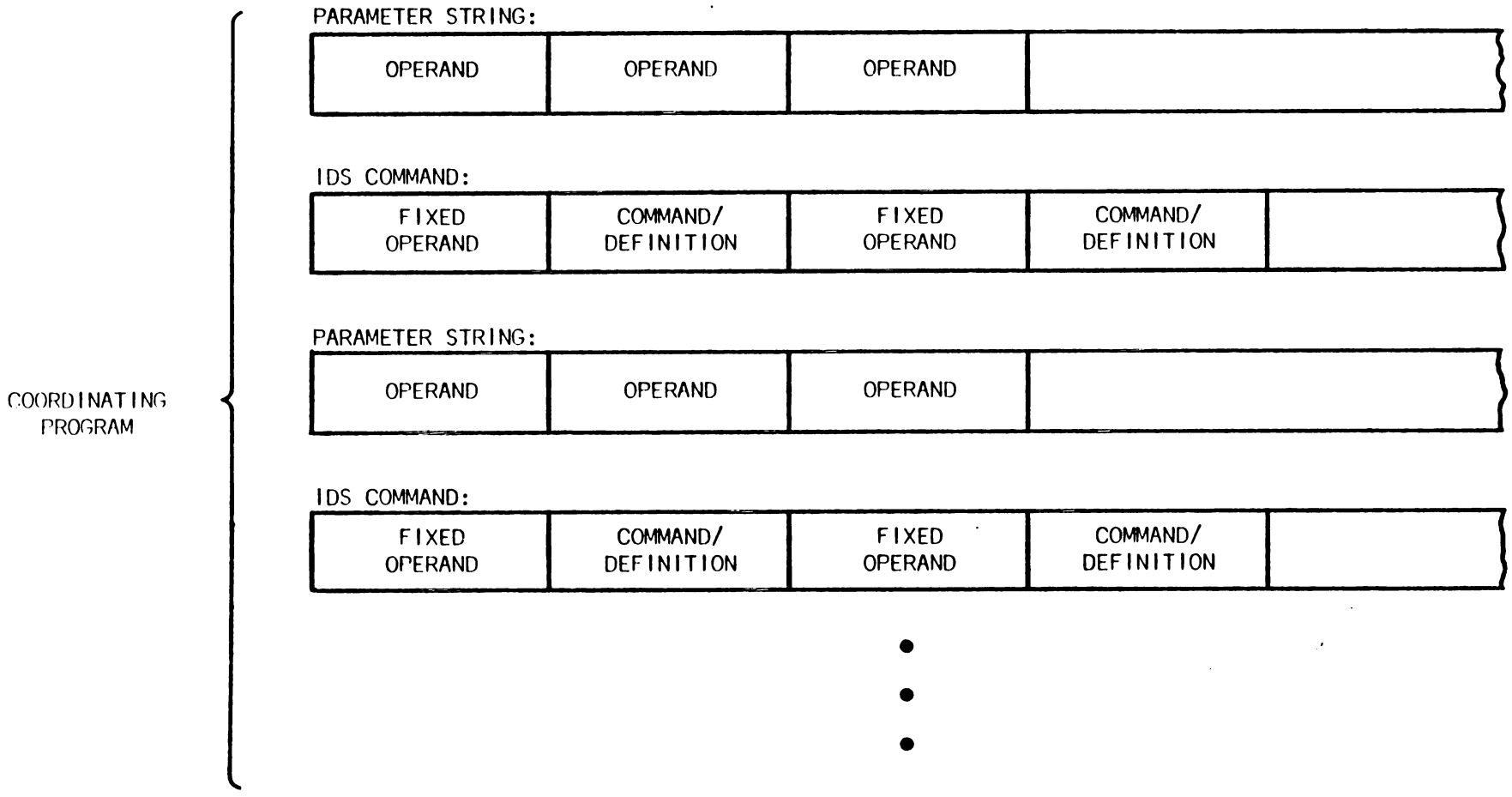


Figure 4.2 Coordinating Program structure

Once an IDS command is defined, it may be used in the definition of another command. Coordinating programs are simply high-level words composed of a sequence of IDS commands and standard FORTH words. The user may execute a word of any level at any time in any sequence; this is the basis for the great degree of flexibility in IDS/TQMS operation.

The firmware is a compiled version of FORTH source code. An interpreter is included within the compiled firmware for interpretation of coordinating programs (the interpreter is one of the standard FORTH words). This interpreter is also very useful in evolving "development commands" (see section 4.2.6) associated the with development and testing of the IDS.

4.2 COMMAND CLASSIFICATION

IDS functions are executed through the invocation of IDS commands; all IDS commands are defined in terms of standard FORTH words and/or other IDS commands. These commands fall into one of six classes: elementary functions, standard control, high-speed control, data processing, calibration, and development.

In the source code, some IDS commands are prefixed with a single special character which designates their relationship to hardware for simplified development; renamings are likely in the Multi-Micro environment.

4.2.1 Elementary Function Commands

The CU (Control Unit) of the IDS generates the addresses and control signals which are required by the PU (Processing Unit) for local control strobe generation. The class of commands referred to as "elementary function commands" directs this signal generation.

Functions executed by these commands include: setting/resetting of integrator switches and the D/A reference; A/D input MUX control, A/D conversion and reading; monitor port reading; "integration-done" interrupt reset; STC (System Timer/Counter) data/command/status loading and unloading; and master-reset. The definitions of these commands are very short, since they are simply a read or write to a location in the memory map. All of these commands are incorporated into the definitons of the other IDS commands (coordinating programs rarely require direct use of elementary function commands).

4.2.2 Data Processing Commands

"Data processing commands" execute calculations and conversions upon hardware-collected data and input control parameters. These commands are defined primarily in standard FORTH words which perform arithmetic and logical operations. Data processing commands are usually found in

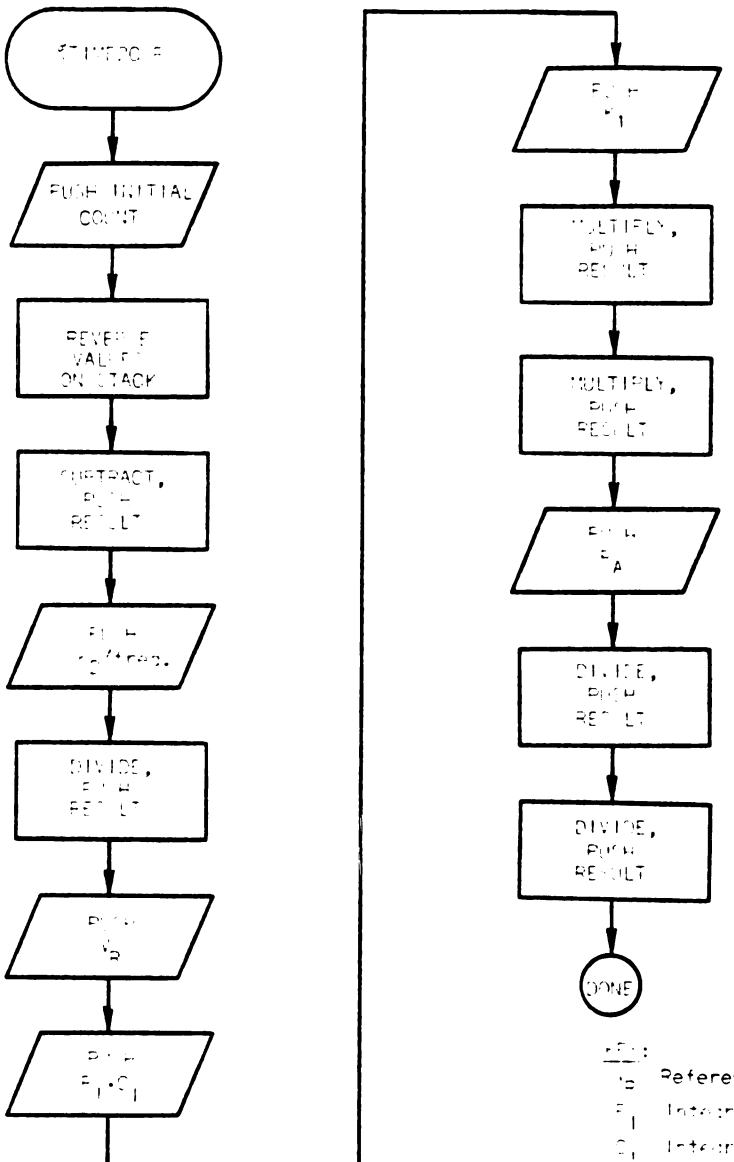
the definitions of "standard control" and "calibration" commands.

An example of a data processing command, used to calculate the input analog current, is given in Figure 4.3. Here an integration is performed with the STC running a count-down counter to determine the integration time. Upon triggering of the comparator, the timer count is read. This value is placed on the "parameter stack" (see FORTH manual: this LIFO stack is used to supply operands to the IDS commands) by an IDS command. A data processing command called "%TIME2CUR" accepts this count, along with several other required parameters, and performs the calculation required to convert the count value into the measured current value.

Due to the fact that FORTH provides only integer arithmetics, special programming precautions are taken in performing calculations and conversions to prevent loss of significance. The representations and formating of parameters and data, which are necessitated by the structure and style of memory and FORTH, are discussed in section 4.3.

4.2.3 Calibration Commands

Two general types of calibration are required of the IDS: adjustments between measuring and threshold setting hardware; and calibration between pulse and analog measuring channels. The former is commonly used in the



Where:

V_R Reference Voltage

R_I Integrator Resistor

C_I Integrator Capacitor

R_A Σ/Δ Resistor

T_I Integration Time

K_1, K_2 Scaling Factors

The analog output current may be calculated as follows:

$$I_A = \frac{V_R \cdot R_I \cdot C_I}{T_I \cdot R_A} \cdot K_1$$

where,

$$T_I = \frac{(\text{initial counter value} - \text{final counter value})}{(\text{source frequency})} \cdot K_2$$

Figure 4.3 Data Processing Command example

testing and development phases with external measuring devices to establish absolute hardware operation points. The latter is required in the on-line accurate sampling of an ion input. Examples are as follows:

"◇COMPCHECK" , a calibration command, compares the reference input of a comparator to the value of the integrator output which triggered it. The binary value latched into the D/A converter (the comparator reference source) is subtracted from the value read by the A/D converter immediately upon comparator triggering (the A/D automatically performs a conversion anytime the comparator fires). If the result is not equal to the calculated hysteresis of the comparator, an error is signalled. Corrections may be implemented via IDS offsetting of the D/A input, hardware adjustment of an appropriate trimpot, or through software compensation of acquired data values.

A range of ion flux levels, between 10^5 and 10^6, provides both pulse and analog current outputs from the Channeltron. By supplying a known ion input to the IDS within this range, "◇CHANLDIFF" finds the difference between the ion flux levels calculated from data obtained by each channel.

Calibration commands are defined primarily in terms of elementary function and low-level data processing commands. Calibration commands are found in the definitions of

standard control, high-level data processing, and development commands.

4.2.4 High-Speed Control Commands

For those data acquisition modes where speed is of the essence, a class of "high-speed control commands" are available. These commands simply alter the "vocabulary" and then execute the assembly code contained within their definitions.

An example is the "!MAXSCAN" command. This command instructs the IDS to wait no longer than 100us to collect a single value. The IDS waits for a signal from the TQMS status port to begin integration. Upon receipt of the "start" signal, integration is begun. The timer, which has been set to a maximum of 100us, counts until either the comparator triggers (unlikely) or time expires (a TC signal is generated by the STC). Depending on the input operands to this command, the C/V output, the integrator output, or the timer value is immediately stored, and a "next" signal is sent to the TQMS status port. This procedure continues for a pre-defined number of acquisitions. Note that at this rate of data collection, no data from the pulse channel is usable (see Figure 2.1).

A standard scan proceedure is highly similar to the above example, except that more operations may be executed between acquisitions since extra time is allotted. In fact,

most high-speed control commands have a "dual" in a standard control command; but the converse is not true. The Command Glossary notes the duals of these commands. High-speed commands are usually contained in the definitions of high-level, standard control commands.

4.2.5 Standard Control Commands

"Standard control commands" act as the "executive" for the IDS. Sequencing of elementary functions, chaining of data and calibration commands, calling of high-speed routines, programming of references and the STC, I/O communications, and many other operations are executed through standard control commands.

An example of a standard control command is "$ECYCLE". The programming of the STC requires a sequential writing to it of mode, load, and hold register information for each of its five 16-bit counters. By loading the proper code to these registers, counters 1 and 2 are concatenated as a 32-bit timer, counters 3 and 4 are concantenated as a 32-bit pulse counter, and counter 5 is an event counter. "$ECYCLE" takes time limit values (stored in variables by a data processing command), and programming code (stored in firmware "constants"), and performs the correct sequence of write operations to the STC.

The definitions for this class of commands are composed of all other command types. "Standard control commands"

typically compose the majority of commands in a coordinating program.

4.2.6 Development Commands

Development of the IDS is a highly interactive process. It begins with the individual assembly and electrical testing of various modules. These modules are then assembled into the 3 main units of the IDS. Upon demonstration that the CU is electrically and logically operative, EPROMs with standard FORTH code are burned (by another computer) and installed in the CU memory. Interactive development of customized FORTH code on a terminal connected to the CU USART is facilitated, since the standard FORTH code includes an interpreter. All IDS commands are developed in this manner. The debugged commands are retained on a floppy disk (which is connected through the second USART); these commands are eventually compiled and burnt into EPROM. The structure of the FORTH language is extremely conducive to development in this manner.

A set of commands which are useful development aids, but are not required in the operational IDS are classified as "development commands". To aid future testing and maintainance, these commands are also compiled into the CU firmware. Conversion into ASCII output formats for terminal display, looping of strobe signals, and other testing

utilities are included among the development commands. Usage of these commands by the IDS while in the Multi-Micro environment is permitted, but is not particularly useful.

4.3 NUMERIC REPRESENTATION

The standard FORTH words provide only for integer arithmetic. Through proper scaling of control and data parameters, computational and representational errors are avoided. The following sections describe both the internal numeric representations used by the IDS and the representations of parameters and data transmitted through IDS I/O.

4.3.1 Internal Representations

Standard FORTH arithmetic words include provisions for the use of 16 and 32-bit operands. Since the largest number range the IDS deals with is 10^9, 32 bits is sufficient for IDS programming. For data processing commands which multiply or divide 32 bit numbers, there are standard FORTH words which generate partial products 48 bits in length.

In order to facilitate integer arithmetic, representations which do not require fractions are used. Examples are as follows:

voltage in millivolts (mV)
current in picoamperes (pA)
time in microseconds (us)
ion flux in ions/second (ips).

According to the calculations of Chapter 2, none of these values ever exceed 10^9. Fractional portions are not required since usuable IDS parameters and data is at least 100 times greater than a fundemental unit, thus providing sufficient resolution (100:1) over all values.

Rounding errors are avoided primarily because errors of up to 1% (in general) are tolerable. Non-linearities associated with data representing real physical parameters may or may not be taken into consideration, depending on the specific data processing command which is utilized; this distinction is made in the glossary.

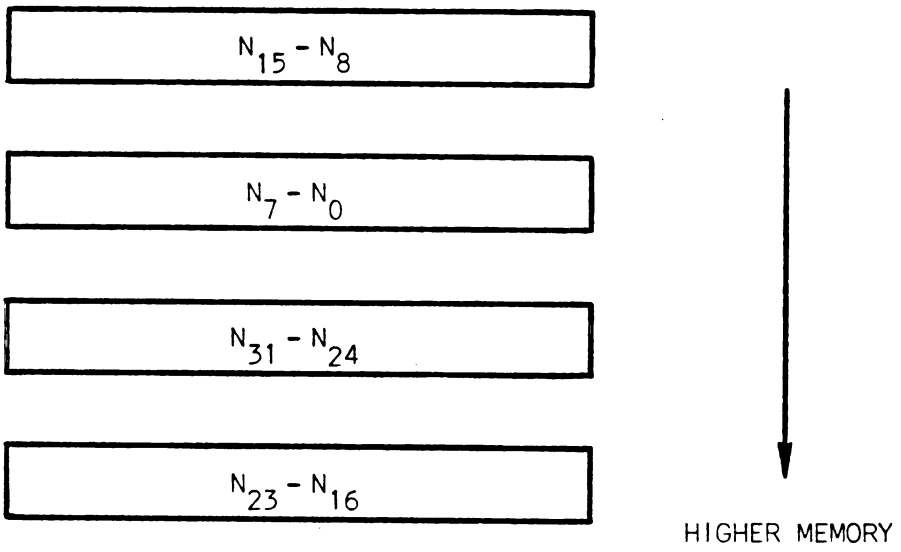
4.3.2 I/O Representations

Integer format I/O to system and development terminals often unnecessarily burdens the user. Standard FORTH words provide a solution to this problem: string-to-number and number-to-string conversion words are included in firmware. These words convert an integer to an ASCII string, which can be printed in fractional format at the terminal; an ASCII character representing the radix point is inserted into the appropriate position in the string.

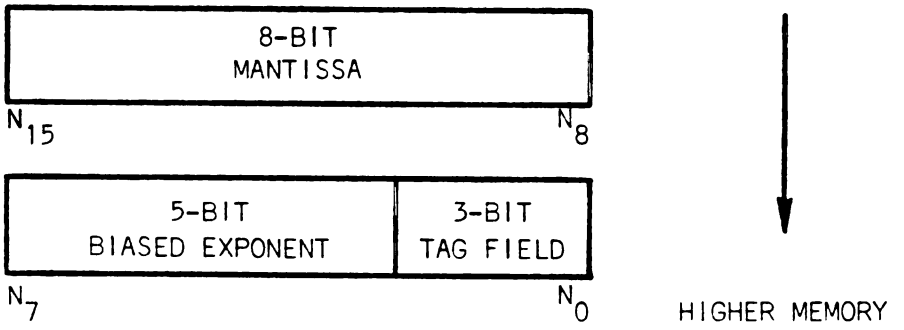
Integer formats of parameters with large ranges of values consume a great deal of memory space (4 bytes per 32-bit number) compared to a floating point format (8 bits of mantissa and 5 bits of exponent provide sufficient resolution and range for the IDS). Data processing commands

which convert an integer to a floating point format are included in the IDS firmware. A shift and count routine is used to perform this function. In cases where high-speed data collection and outputting are being executed, such floating point formating cannot be done because these operations slow down the data acquisition process.

Some data formats include tag bits with these bits indicating which PU channel, or which A/D input, generated that specific piece of data. This type of tagging is most appropriate for high-speed operations or for hardware calibration. Figure 4.4 depicts the integer and floating point formats; the sequencing of bytes in memory is a characteristic of the 8085 and FORTH structures.



32-BIT INTEGER FORMAT



16-BIT FLOATING POINT FORMAT

Figure 4.4 Data formats

*** CHAPTER 5 --- EVALUATION ***

The Ion Detection System (IDS) is presently in final development stages. Full-scale testing and evaluation require interfacing of the IDS into the TQMS/Multi-Micro System environment, with known ion inputs and their spectra providing benchmarks for complete IDS evaluation. This interfacing will require a significant amount of time, since the Multi-Micro System is yet to be fully developed and tested. Because of the impracticality of full-scale IDS testing in light of current user demands upon the TQMS, and because several adjustments are still required to the IDS (as described later in this chapter), evaluations in this chapter are limited to those from a "bench" environment. Since operation of the Detection Unit (DU) can only be achieved inside of the TQMS vacuum, only the Processing Unit (PU) and the Control Unit (CU) are evaluated here.

This chapter begins with a description of a typical sampling operation. Simulation of this operation is facilitated by the use of a picoampere source in place of the DU (Channeltron) analog output. Following sections highlight the performance of critical circuits; their responses under under sampling simulation are particularly stressed. A final section discusses IDS performance from an overall perspective.

5.1 SAMPLING SIMULATION

The IDS is required to collect up to a maximum of 10000 samples during a single scanning operation (the term "sample" refers to the collection of a single data value corresponding to the ion flux). Programming of the number of samples to be acquired, the data precision, and the maximum sampling time is normally done by the Multi-Micro System through the loading of a coordinating program (see Chapter 4) into the IDS memory. For bench-testing purposes, the IDS is interfaced to a terminal through which a user supplies commands to the on-board FORTH interpreter.

IDS hardware and firmware are responsible for measuring and storing (locally) data values. Once instructed to sample, the IDS normally requires no further interaction with external systems during the sampling period. By simulating this sampling operation, all primary circuit functions can be monitored both through terminal outputs and by external test apparatus.

Since the DU is unavailable for simulation, a picoampere source is substituted for the analog current output of the Channeltron. No simulation of the pulse channel is performed on the bench, since the major component of this channel is a commercially-built module which is specifically designed to operate in conjuction with a channeltron output.

The sequence of IDS commands and parameters, the

coordinating program, used to perform this simulation is given in Figure 5.1; the development system terminal substitutes for the Multi-Micro System. After initialization, the IDS sends a prompt to the development system terminal, signaling that it is ready to sample. After the picoampere input is set, a command from the terminal initiates a sampling. The IDS returns the C/V converter output voltage, the integrator output voltage, and the integration time to the terminal (instead of an array in memory). These values are logged; the voltage values are compared with externally measured values, and the integration times are plotted to determine linearity in comparison with measured input voltages. This procedure is performed over the entire range of analog inputs.

Several of these simulations are performed, and their results are compared. The variance from ideal expected values is attributed to various circuits: these deviations are discussed in conjunction with their contributing circuits in the next several sections. Figure 5.2 shows several plots of the collected data.

5.2 PROCESSING UNIT (PU) PERFORMANCE

The C/V converter, the integrator, the A/D converter, and the STC (timer/counter) are the key circuits within the analog channel responsible for the precise measurement of DU output currents and conversion into digital data. The

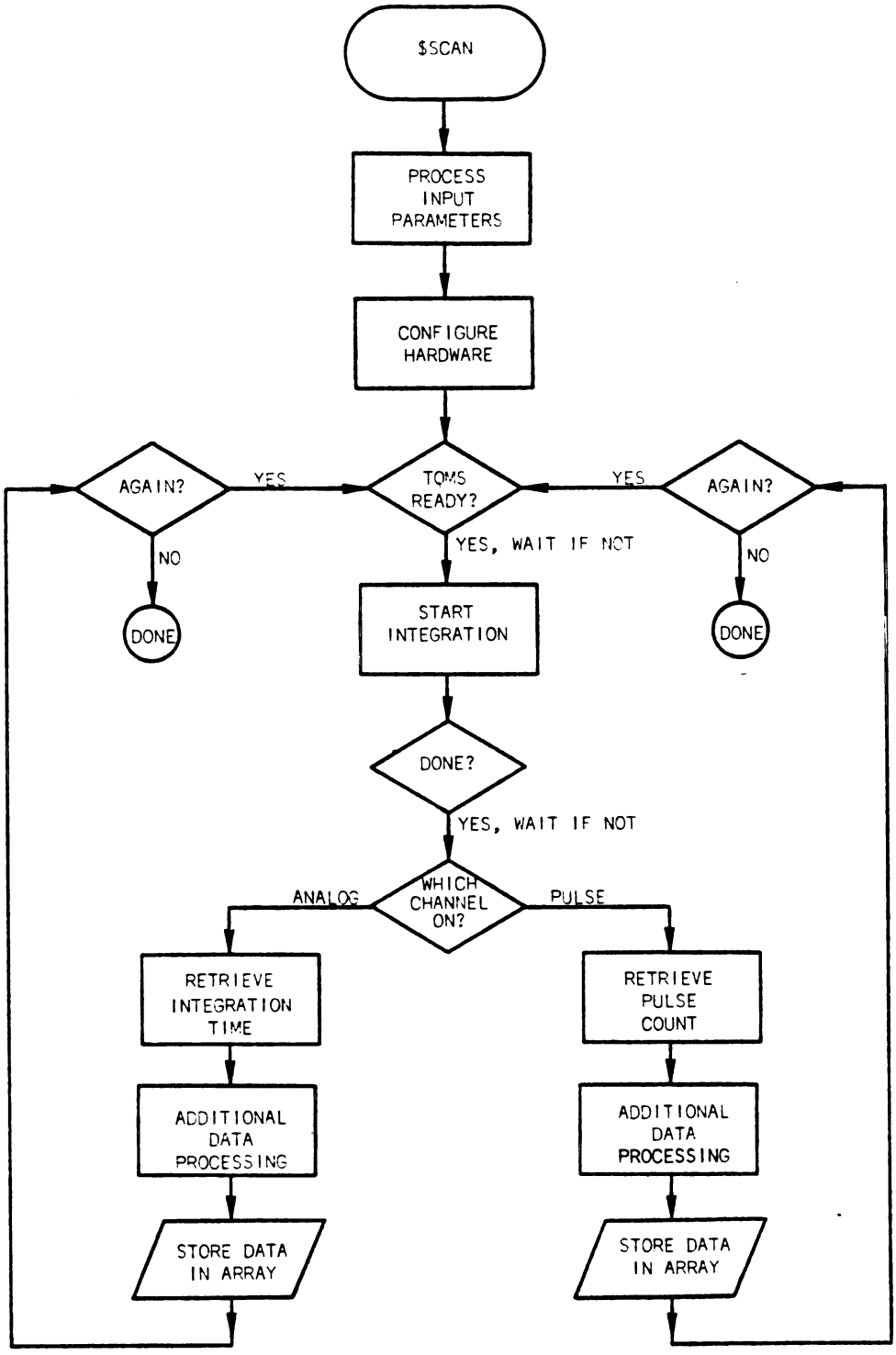
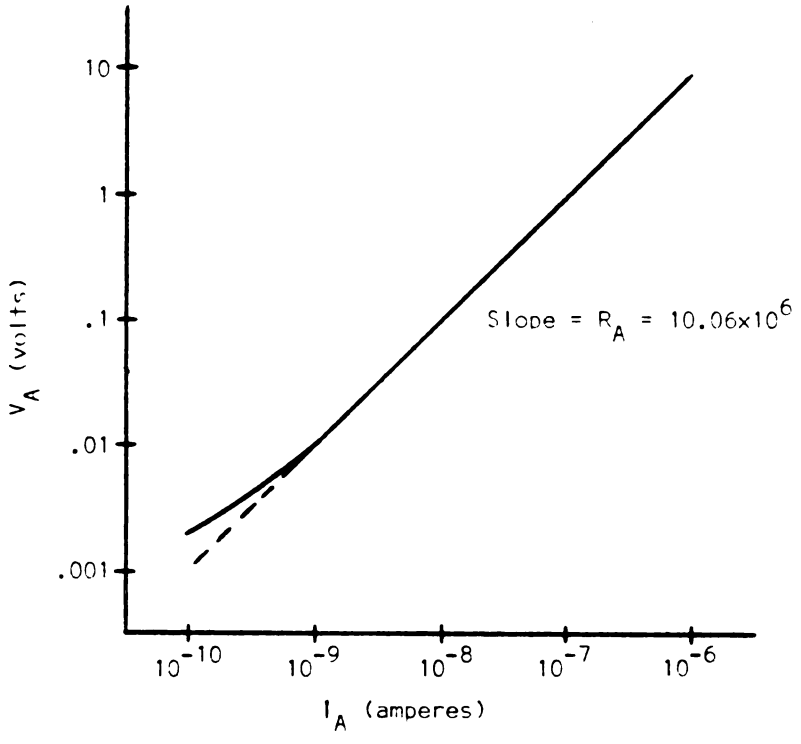


Figure 5.1 Simulation program



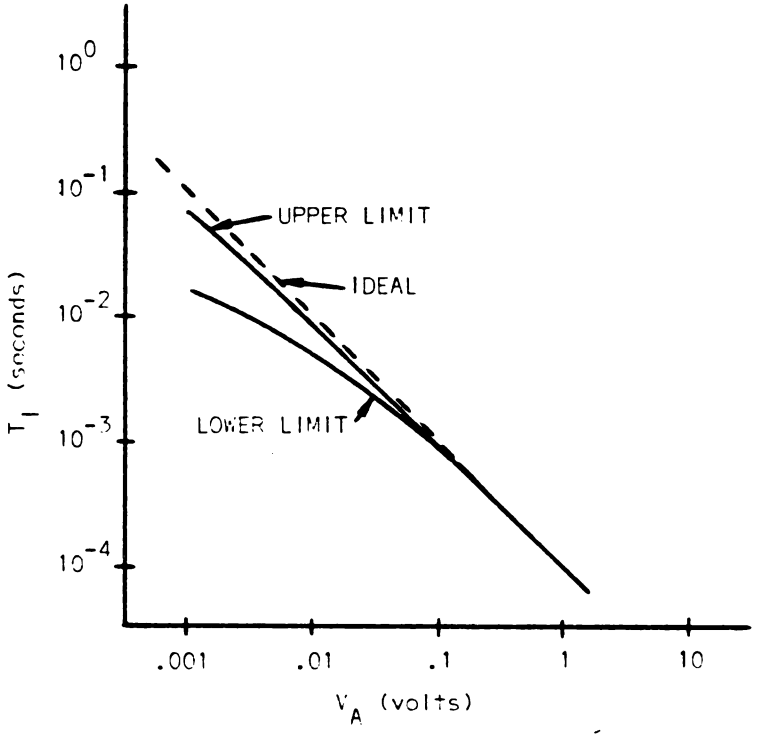


Figure 5.2 C/V Converter and Integrator Circuit performance

following sections individually examine the performance of each of these circuits in regard to the simulations.

5.2.1 C/V Converter

Since the C/V Converter must convert currents ranging from 10 pA to .5 uA into voltages ranging from 1 mV to 5 V, small amplitude noise may easily corrupt the signal. The integrator circuit filters a significant amount of mid- and high-frequency noise, but offsets or drifts greater than 1% in the C/V output distort the data by an equivalent amount. This degree of error exceeds design criteria. Interference noise, in the presence of 60 Hz from lab sources, must also be guarded against admission to the integrator input.

Measurements with a DVM and an oscilliscope are performed without the analog board housed in its shielded box (which is presently under construction). As seen in Figure 5.2, C/V output values for low input currents are somewhat offset from the ideal. Specifications for the ICL7650 op-amp show bias currents of 10 pA maximum, not enough to be responsible for the error. Evidence points to the picoampere source as uncalibrated; measurements with an electrometer should validate this conclusion. Interference noise poses a greater problem: sources include 60-cycle, digital generated rf, and the op-amp itself as a result of its chopping action. Oscilliscope measurements show noise up to 50 mV (p-p); data returned by the A/D substantiate

this. This noise level is significantly reduced from the 100 mV (p-p), which was observed in the circuit before shielding was added to the feedback resistor. Enclosure of the entire circuit board in a shielded housing should further reduce this noise.

5.2.2 Integrator Circuit

Three sub-cicuits comprise the integrator circuit: the op-amp integrator, the comparator, and the D/A reference. Evaluations of these sub-circuits are given individually as follows:

The op-amp integrator exhibits a constant offset (0.6%) from its calculated slope of integration; this offset falls easily within the range of allowable component (capacitor and resistor) tolerance error. Also, at low input voltages the integrator output resembles a staircase. This may be the result of the noisy input signal, or the chopping action of the integrator op-amp. Further investigation is required. Enclosure within a shielded box, or capacitive loading of the input may reduce this effect.

No oscillations of the comparator's output are observed at even the slowest threshold crossings. Incorrect triggering of the comparator does occur at slow ramp inputs as a result of the associated staircase effect of the integrator. This provides an explanation for the increasing variation of integration times in conjuction with the

increasing amplitude of the staircase steps. Also, since the 50 mV (p-p) noise common to the entire board is also observed in the comparator input signals, premature triggering of the comparator results. This premature triggering produces integration times which taper away from expected values as the input voltages decrease (see Figure 5.2). It is unlikely that this type of error is the result of circuit leakage currents, unless manufacturer's specifications for components are incorrect.

The D/A converter provides a 0 to 10 V reference ranging in 256 steps (8-bit converter); the analog reference may vary from the desired value by ±19.5 mV. At low reference settings (the reference value represents the minimum-required "analog count", see section 3.3.4), a significant error may result. This error may be compensated for in software by performing a "◇COMPCHECK" operation (the determination of the actual comparator triggering point) and adjusting collected data values appropriately.

5.2.3 A/D Converter

The 12-bit A/D converter provides a resolution of 1:4096; since a 0-to-10 V range is used, precision to ±1.22 mV is achieved. Since the outputs of the C/V converter and the integrator never exceed 5 V, a 2x gain is applied to the analog signal as it is driven off the analog board to the "D" board, where the A/D converter resides. This makes use

of the full A/D input range (10 V). Lower level signals have increased resolution, while all signals are measured with increased noise margins.

The data generated by the A/D exhibit variations which are partially attributable to simple twisted-pair connection between boards in the development stage (to be replaced with coaxial cables). Also, since input levels are relatively stable over the integration period, no sample-and-hold circuit is included.

5.2.4 STC (System Timer/Counter)

The STC (System Timer/Counter) timer is gated by the same signal which controls the integrator input switch. The time counted by the timer is designed to be exactly equal to the period from which the integration switch was thrown, to the time when the comparator triggers (assuming the timer has not reached its preset maximum limit). Switch delays and jitters contribute insignificant errors, even over minimal integration periods.

5.2.5 Other PU Circuits

All other PU circuits, the support logic, the analog control circuit, and the tranceivers of the PU-CU interface function according to specification. Gating delay times within the logic circuits never add to create a switching

delay (such as with the integrator circuit switches) which could distort the integration time by more than 0.2%, far less than the 1% allowable.

5.3 CONTROL UNIT (CU) PERFORMANCE

The following sections present evaluations of the Control Unit (CU), highlighting software and CU-PU interfacing aspects.

5.3.1 Software

The FORTH-based software facilitates many on-line innovations in the programming of the IDS. The commands listed in the glossary are the basic building blocks for the development of customized acquisition modes in IDS operations.

FORTH is extremely valuable as a system-debugging aid. Since commands are developed interactively on the IDS, and since FORTH lends itself well to the establishment of high-level routines, software development is accomplished in an extremely short period of time.

The present set of IDS commands is sufficient for the performance of all basic operations. "High-speed-control commands" (assembly routines) are yet to be written, as they require a closer interaction with the Multi-Micro System. Other command classes also have room for further development

in conjunction with the evolving definition of on-line IDS requirements.

5.3.2 Hardware

The use of a memory-mapped control structure is an effective means of actuating IDS functions. Simple interfacing and programming requirements facilitated debugging and development. The Dual USART's also provide the additional option of sending data to a storage device outside of the Multi-Micro System. The Interrupt Controller's full capabilities are untapped (the IDS only uses 2 interrupts); a simpler interrupt structure may have been used, but with limited expansion capability and elongated development time. CU-PU interface limitations on the number of control lines are successfully circumvented by the use of the interface-enable proceedure and local power-on reset circuitry for the PU. All other CU circuits function according to specifications.

5.4 General Observations

The following general observations on IDS performance address the secondary design criteria established in Chapter 2. They include noise immunity, stability and reliability, usability, and cost.

5.4.1 Noise Immunity

As previously demonstrated, the analog circuitry (in its present state) of the IDS is easily corrupted by noise at low input levels. Even upon construction of final IDS housing, additional adjustments may still be required to insure sufficient guarding against unwanted disturbances.

If noise problems still persist, data processing techniques may be employed to compensate for errors. This would be viable, since those inputs which are most affected (low-level signals) require significantly greater integration times. Slow-downs to perform software compensation are negligble compared to integration periods. High-speed data collection only detects high-level signals, which are sufficiently immune to noise.

5.4.2 Stability and Reliability

PU circuit op-amps which provide high-gains require capacitors across their feedback resistors in order to suppress oscillation. Also, if the CMOS op-amps (ICL 7650's) are overdriven, latch-up may occur; protection is provided by zener diodes across feedback paths.

Several components require hardware compensation of offsets via trimpots. After initial calibration, most components exhibit stable operation. The differential analog receivers tend to drift significantly with

temperature. Calibration after an appropriate warm-up period compensates effectively.

5.4.3 Usability

The IDS exhibits all the required properties associated with flexibility and user friendliness in operation. The FORTH software provides the framework for this. The tailoring of IDS command mnemonics to resemble the functions which they actuate, facilitates development of user dexterity with the IDS.

From a hardware standpoint, all boards and units connect simply via ribbon cable, coaxial cable, and twisted pair wires. These cables/wires are terminated using standard connectors.

*** CHAPTER 6 --- SUMMARY ***

This final chapter provides a general summary of the Ion Detection System (IDS) design goals, implementation, and performance. Further improvements are suggested, and a conclusion relative to the IDS's value to the Triple Quadrupole Mass Spectrometer's (TQMS) Multi-Micro System is provided.

6.1 ACHIEVEMENT OF GOALS

The major goal in undertaking development of the IDS was to create a detection system which operates with optimal efficiency and detection capability. This translates to providing a detector with a dynamic range of 10^9, and which uses the minimally required integration period (as prescribed by the acceptable error) to acquire a data sample. Combination of a detector of this capability with a Multi-Micro control structure will result in greatly enhanced performance of the Triple Quadrupole Mass Spectrometer.

The expansion of dynamic range is to be achieved through the use of a newly developed Channeltron™ (electron multiplier) in conjunction with appropriate signal-processing electronics (both pulse counting and analog single-slope integration techniques are used).

Testing of the dynamic range capability is still under way. Preliminary results indicate that noise problems, which induce errors greater than 1% to relative precision, exist at low analog signal levels (which provide information about ion flux levels between 10^5 and 10^9 ips). Eventual housing of sensitive circuits in a shielded environment should solve this problem. Pulse current detection circuitry (for flux levels less than 10^6) is yet untested.

The control of asynchronous data acquisition (according to the statistical relationship between the incidence of random ions and the error involved in counting them) is fully implemented and successfully exploits the throughput advantages in this type of scheme. Fundamental firmware which directs hardware operations has been developed and debugged. Further software developments to customize operations and to operate within the Multi-Micro environment may be simply developed using the firmware command kernels (written in FORTH).

Circuitry which performs analog integration exhibits some premature and variant integration time outputs. Although not critical, these variations create errors in excess of 1%. These have been attributed to a defective, specialized IC and to noise problems which permeate the entire board. Again, appropriate shielding measures are being undertaken.

Secondary design goals of stability and reliability, usability, data structures, noise immunity, and cost minimization are only general guidelines in the IDS design. Although evaluations will have to wait until the IDS is operational within the Multi-Micro System, precautions were taken to minimize unwanted effects.

6.2 FURTHER IMPROVEMENTS

Assuming that the prescribed analog circuit adjustments fail to resolve the dynamic range problems, several alternatives exist. One is the replacement of the ICL7650 op-amps with more expensive instrumentation amplifiers, even with the reduced slew rates. New developments in IC technology may possibly provide suitable circuits. Another alternative is the redesign of the analog circuit board to provide increased shielding of individual, critical components. A more complex and costly scheme would involve the use of multiple amplification, integration, and A/D converter stages to segment the analog range portion; various combinations of these circuits could be employed. Also, the use of a dual-slope integration technique could enhance integration precision.

Any further improvements from a design consideration standpoint would first require fundamental improvements in the design of the TQMS itself before increased precision would be meaningful, or increased throughput could be

realized (throughput is statistically limited for a given flux range). The present IDS hardware supplies all necessary circuitry, even for auto- calibration, that is meaningful for this application.

Software enhancements are dependent on the type of acquisitions which may be required. Since FORTH provides more than sufficient flexibility, most software improvements could be made in the development of assembly-coded routines (for high-speed data acquisition), or in the areas of user communications (which would aid the inexperienced user).

6.3 CONCLUSIONS

An ion detection system for a triple quadrupole mass spectrometer, optimally suited for operation within a multi-micro control structure and for data acquisition in the most expedious manner, has been successfully developed. Fundamental to the success of the IDS design is its basis upon the statistical characteristics of the measured quantity (ions), permitting adaptive data sampling for enhanced performance. Although some problems have been identified in preliminary test phases, continuing development is resolving these difficulties. Expectations are that the Ion Detection System (IDS) will provide a greatly enhanced data acquisition capability for the TQMS.

*** APPENDICES ***

*** APPENDIX A --- PROCESSING UNIT (PU) CALCULATIONS ***

The parameter and component values used in the Processing Unit (PU) were calculated as follows:

Defined limits of $5x10^{8}$ for the maximum ion flux, $IFLX_{mx}$ and 1% for the minimum "typical" error (Section 2.1.2), e_{mn}, specify the minimum integration time (Figure 2.1) to be 20 us. To insure 1% accuracy of the timer, 100 non-random counts, Ct_{mn}, must count the time. Thus a 5MHz source is supplied to the timer. This is generalized as:

$$f_{tmr} = IFLX_{mx} * e'_{mn}$$

where $e'_{mn} = e_{mn} / 100\%$ and f_{tmr} = timer source frequency. Thus any flux rate can be calculated:

$$IFLX = f_{tmr} / (e'^{2} * Ct)$$

where Ct = timer counts elapsed during integration period, and e' = programmed "typical" error for the sample. This error is achieved by integrating until the required amount of ions has been detected (Section 2.1.2).

For the pulse channel, flux determination simply requires counting the discrete current pulses to a

set number (related to the error) while simultaneously timing the integration (counting) period.

The analog channel operates in an analogous manner. As described in Section 3.3.4, analog input current can be thought of as chained "charge packets" proportional to the ion flux. A capacitor "counts" this charge to a set voltage while this integration period is timed; this set (or reference) voltage is similarly a function of the programmed error, e' (i.e., charge proportional to 10000 detected ions must be "counted" to achieve 1% accuracy).

The integrator capacitor output voltage is:

$$V_I = Q_I / C_I$$

Since the maximum "analog count" occurs for the minimum error setting, 1%, the corresponding reference voltage will also be at its maximum. To comply with electrical requirements, this value is chosen to be 5 V. This voltage will represent 10000 ion counts, provided we select proper capacitor and charge amplification components. (Note that linearity is assumed; in reality this is not quite accurate, but software compensates for this.) Lower settings of the reference voltage achieve lesser "counts" (thus greater errors).

The component values are calculated as follows. With V_A = integrator input (C/V output), R_I = integrator input resistor, and:

$$I_I = V_A / R_I$$

then:

$$Q_I = I_I * t_{int}$$

where t_{int} is the integration time. Figure 2.1 shows that for the maximum flux measured to 1%, a 20 us integration time is required. Also, an input current, I_A = .5 uA, is supplied at this flux level (assuming linearity, see Figure 3.3). Since:

$$V_A = I_A * R_F$$

and:

$$V_I = (V_A * t_{int}) / (R_I * C_I)$$

then:

$$R_F / (R_I * C_I) = V_I / (t_{int} * I_A)$$

$$= 5 / (20x10^{-6} * 5x10^{-7})$$

$$= 5x10^{11} F^{-1}$$

The resistor and capacitor values are chosen to optimize operation against effects of leakage, biases, offsets,

loading, and noise, over the entire input current range. Practical values chosen are:

$$R_F = 10\ M\Omega$$

$$R_I = 20\ K\Omega$$

$$C_I = 1000\ pF.$$

*** APPENDIX B --- COMMAND GLOSSARY ***

polyFORTH GLOSSARY

WORD	VOCABULARY	BLOCK	STACK	ENTERED

#ACOUNT DETECTOR 55 0-1 25 FEB 1982
Variable: The value which is read out of the timer is stored here (32-bit).

#ADCELL DETECTOR 50 0-1 25 FEB 1982
Variable: Holds the combined MSB and LSB of the A/D, right-justified (12 bits in a 16 bit cell).

#ADVOLTS DETECTOR 50 0-1 25 FEB 1982
Variable: Holds a value of the A/D in a decimal, weighted format.

#CLOCK DETECTOR 47 0-1 25 FEB 1982
Constant: STC clock frequency.

#CMR DETECTOR 52 0-1 25 FEB 1982
Constant array: Values define mode register values to be loaded (in STC).

#COUNTMAX DETECTOR 47 0-1 25 FEB 1982
Variable: Holds the timer count-limit, proportional to "#MAXTIME".

#ECOUNT DETECTOR 55 0-1 25 FEB 1982
Variable: The value which is read out of the event counter is stored here (16-bit).

#ERSQUINV DETECTOR 47 0-1 25 FEB 1982
Variable: Holds the calculated value of the scaled, inverse-squared-error.

#HLR DETECTOR 52 0-1 25 FEB 1982
Constant array: Values define hold register values to be loaded (in STC).

#IGCOUNT DETECTOR 47 0-1 25 FEB 1982
Variable: Holds the calculated elapsed counts of integration.

#IGTIME DETECTOR 47 0-1 25 FEB 1982
Variable: Holds the calculated elapsed time of integration (in microseconds).

#MAXTIME DETECTOR 47 0-1 25 FEB 1982
Variable: Holds the programmed time-limit for integration (in microseconds).

#MINERROR DETECTOR 47 0-1 25 FEB 1982
Variable: Holds the programmed error value, an integer between 1 and 100 representing the percent relative standard deviation.

#PCOUNT DETECTOR 55 0-1 25 FEB 1982
Variable: The value which is read out of the pulse counter is stored here (32-bit).

#VLSB DETECTOR 50 0-1 25 FEB 1982
Constant: Represents the value of the A/D least significant bit, .0024414 V (12-bit A/D over 0-10 V range). Value is scaled for use in integer arithmetic.

$ASTORE DETECTOR 55 4-0 25 FEB 1982
Stores the bytes read from the timer into a variable. Performs the sequencing required to represent the bytes as a FORTH 32-bit number.

$BYTER DETECTOR 53 1-0 25 FEB 1982
Loads the LSB followed by the MSB into an STC register. Used within the "$ECYCLE" routine.

$CV? DETECTOR 58 0-0 25 FEB 1982
Connect, convert, and print the value of the C/V converter output (via the A/D).

$ECYCLE DETECTOR 54 0-0 25 FEB 1982
Supervises the loading of the STC registers during an initialization. The master-mode register bit which permits auto-incrementing of the data-pointer must be active in order to perform an "element cycle" (see STC data sheet).

$EMIN? DETECTOR 58 0-0 25 FEB 1982
Prints the error-limit value.

$ERRORSET DETECTOR 48 1-0 25 FEB 1982
Interprets the top stack value to represent the error-limit setting for integrations (in integer $). Converts this into a scaled, inverse-squared-error value (convenient for later calculations). Stores both values into variables.

$ESTORE DETECTOR 55 2-0 25 FEB 1982
Stores the bytes read from the event counter into a variable. Performs sequencing of the bytes in order to represent a FORTH 16-bit number.

$EVENTER DETECTOR 54 0-0 25 FEB 1982
Loads the event counter (#5) registers; presently not in use.

$FILER DETECTOR 56 0-0 25 FEB 1982
Coordinates the storing of stack values, representing the hold register values, into the proper variables.

$HCYCLE DETECTOR 56 0-0 25 FEB 1982
Coordinates the "hold cycle" operation of the STC (reading the hold registers) and the storing of these values into variables. The master mode register bit which permits auto-incrementing must be active for "hold-cycling".

$HODER DETECTOR 53 0-0 25 FEB 1982
Loads the hold registers of the STC. Used within the "$ECYCLE" routine.

$IGCOUNT DETECTOR 49 0-0 25 FEB 1982
Calculates the elapsed integration count by subtracting the post-integration timer value from the pre-integration timer value (the count-limit). Stores this result in a variable. Note: the timer operates in a count-down mode.

$IGTIME DETECTOR 49 0-0 25 FEB 1982
Converts the elapsed integration count value into elapsed integration time (in microseconds). Stores result in a variable.

$INT-TIME? DETECTOR 58 0-0 25 FEB 1982
Calculates and prints the integration time (in microseconds).

$INT? DETECTOR 58 0-0 25 FEB 1982
Connect, convert, and print the value of the integrator output (via the A/D).

$LODER DETECTOR 53 0-0 25 FEB 1982
Loads the load registers of the STC. Used within the "$ECYCLE" routine.

$MODER DETECTOR 53 0-0 25 FEB 1982
Loads the mode registers of the STC. Used within the "$ECYCLE" routine.

$PSTORE DETECTOR 55 4-0 25 FEB 1982
Stores the bytes read from the pulse counter into a variable. Performs the sequencing required to represent the bytes as a FORTH 32-bit number.

$READER DETECTOR 56 1-0 25 FEB 1982
Sequentially reads all hold registers (the counter outputs providing a "save" command was issued to the STC) onto the stack.

$RESET-STC DETECTOR 54 0-0 25 FEB 1982
Performs a software reset of the STC.

$TIMESET DETECTOR 48 1-0 25 FEB 1982
Interprets the top stack value to represent the time-limit setting for integrations (in microseconds). Converts this value to a count-limit and stores both values into variables.

$TMAX? DETECTOR 58 0-0 25 FEB 1982
Prints the time-limit value.

.S DETECTOR 51 0-0 25 FEB 1982
Prints out the contents of the stack non-destructively.

ADCNVRT DETECTOR 46 0-0 25 FEB 1982
Initiates A/D conversion (25 us maximum conversion time).

ADLSB DETECTOR 45 0-1 25 FEB 1982
Reads A/D LSB (and monitor port) and places value on the stack.

ADMSB DETECTOR 45 0-1 25 FEB 1982
Reads A/D MSB output and places value on the stack.

ADOUT? DETECTOR 57 0-0 25 FEB 1982
Prints the binary output of the A/D converter value stored in "#ADCELL".

ADSTART DETECTOR 44 0-1 25 FEB 1982
Constant: Address to start A/D conversion.

ADVOLTS? DETECTOR 57 0-0 25 FEB 1982
Prints the formatted decimal value of the A/D converter value stored in "#ADVOLTS".

ADWEIGHT DETECTOR 50 0-0 25 FEB 1982
Reads and formats the A/D output bytes into a single cell (16 bit format, right-justified).

AMPOUT DETECTOR 45 0-0 25 FEB 1982
Connects C/V output to A/D input via input MUX.

BINARY DETECTOR 51 0-0 25 FEB 1982
Changes the system number base to 2.

BTOV DETECTOR 50 0-0 25 FEB 1982
Converts the binary value of the A/D output into a decimal, weighted format.

CLOAD DETECTOR 46 1-0 25 FEB 1982
Top stack value is loaded into the STC control port.

CNTRLPORT DETECTOR 44 0-1 25 FEB 1982
Constant: Address to STC control/status port.

DALOAD DETECTOR 46 1-0 25 FEB 1982
Top stack value is latched into D/A and the D/A output follows accordingly.

DASTART DETECTOR 44 0-1 25 FEB 1982
Constant: Address to load D/A.

DATAPORT DETECTOR 44 0-1 25 FEB 1982
Constant: Address to STC data port.

DLOAD DETECTOR 46 1-0 25 FEB 1982
Top stack value is loaded into the STC data port.

DONEACKN DETECTOR 46 0-0 25 FEB 1982
Resets "integration-done" interrupt flip-flop.

DREAD DETECTOR 46 0-1 25 FEB 1982
A data register value is read from the STC data port and placed on the stack. The setting of the data pointer selects the specific output register; the data pointer is programmed via the control port.

IGRATER DETECTOR 44 0-1 25 FEB 1982
Constant: Address to switch integrator output thru the A/D input MUX.

IHOLD DETECTOR 45 0-0 25 FEB 1982
Disconnects C/V output from integrator input, i.e holds integrator output voltage.

INTGOUT DETECTOR 45 0-0 25 FEB 1982
Connects integrator output to A/D input via input MUX.

IRESET DETECTOR 45 0-0 25 FEB 1982
Zeros integrator output, i.e. integrator reset.

IRPTOFF DETECTOR 44 0-1 25 FEB 1982
Constant: Address to acknowledge interrupt.

ISET DETECTOR 45 0-0 25 FEB 1982
Allows integrator to charge (unzeros).

ISTART DETECTOR 45 0-0 25 FEB 1982
Connects C/V output to integrator input, i.e. starts integration.

LSBLATCH DETECTOR 44 0-1 25 FEB 1982
Constant: Address to read A/D LSB and monitor port.

MPORT? DETECTOR 57 0-0 25 FEB 1982
Inputs and prints the bits of the 4-bit monitor port.

```
MSBLATCH      DETECTOR        44      0-0      25 FEB 1982
          Constant:  Address to read A/D MSB.

PREAMP        DETECTOR        44      0-1      25 FEB 1982
          Constant:  Address to switch C/V converter output thru the
          A/D input MUX.

S3CLOSE       DETECTOR        44      0-1      25 FEB 1982
          Constant:  Address to close Switch 3.

S3OPEN        DETECTOR        44      0-1      25 FEB 1982
          Constant:  Address to open Switch 3.

S4CLOSE       DETECTOR        44      0-1      25 FEB 1982
          Constant:  Address to close Switch 4.

S4OPEN        DETECTOR        44      0-1      25 FEB 1982
          Constant:  Address to open Switch 4.

SREAD         DETECTOR        46      0-1      25 FEB 1982
          Status register value is read from STC control/status
          port and placed on the stack.

SWRESET       DETECTOR        45      0-0      25 FEB 1982
          Resets all switches:  S1 open, S2 close, S3 close, S4 open.
          The integrator is held zeroed with its input disconnected and
          with its output connected to the A/D input.

SWSTART       DETECTOR        44      0-1      25 FEB 1982
          Constant:  Address to switch reset.

UD.           DETECTOR        51      1-0      25 FEB 1982
          Prints out the top stack value (if a 32-bit number) in
          an unsigned format.
```

*** APPENDIX C --- SOURCE CODE ***

```
L 44 58 LP:

Block Number:  44
0   ( PU ADDRESS MAP---CHIP SELECTS )
1
2   HEX
3   BF80 CONSTANT LSBLATCH       BF81 CONSTANT MSBLATCH
4   BF82 CONSTANT S4OPEN         BF83 CONSTANT S3CLOSE
5   BF84 CONSTANT SWSTART        BF85 CONSTANT IGRATER
6   BF86 CONSTANT S3OPEN         BF87 CONSTANT PREAMP
7   BF88 CONSTANT DASTART        BF89 CONSTANT ADSTART
8   BF8A CONSTANT IRPTOFF        BF8B CONSTANT S4CLOSE
9   BF8E CONSTANT DATAPORT       BF8F CONSTANT CNTRLPORT
10  DECIMAL
11
12
13
14
15

Block Number:  45
0   ( ELMENTARY HARDWARE COMMANDS )
1
2   : AMPOUT   PREAMP C@ DROP ;  ( CONNECTS PRE-AMP TO A/D INPUT,
3     SWITCH 1 CLOSED, SWITCH 2 OPEN )
4   : INTGOUT   IGRATER C@ DROP ;  ( CONNECTS INTEGRATER TO A/D
5     INPUT, SWITCH 2 CLOSED, SWITCH 1 OPEN )
6   : IRESET   S3CLOSE C@ DROP ;  ( ZEROS INTEGRATER OUTPUT )
7   : ISET   S3OPEN C@ DROP ;  ( FLOATS INTERGRATER OUTPUT )
8   : ISTART   0 S4CLOSE C! ;  ( CONNECTS PRE-AMP TO INTEGRATER )
9   : IHOLD   S4OPEN C@ DROP ;  ( DISCONNECTS PRE-AMP FROM INTGRTR )
10  : SWRESET   SWSTART C@ DROP ;  ( RESETS ALL SWITCHES TO POWER-UP
11    STATE, SWITCHES 1 AND 4 OPEN, SWITCHES 2 AND 3 CLOSED )
12  : ADMSB   MSBLATCH C@ ;  ( A/D OUTPUT, DO11-DO4 ---> STACK )
13  : ADLSB   LSBLATCH C@ ;  ( A/D OUTPUT AND MONITOR REGISTER,
14    DO3-DOO. M3-MO ---> STACK )
15

Block Number:  46
0   ( ELEMENTARY HARDWARE COMMANDS )
1
2   : ADCNVRT   0 ADSTART C! ;  ( INITIATES A/D CONVERSION )
3   : DALOAD   DASTART C! ;  ( STACK ---> DI7-DIO, SETS D/A )
4   : DONEACKN   0 IRPTOFF C! ;  ( RESETS DONE INTERRUPT )
5   : CLOAD   CNTRLPORT C! ;  ( STACK ---> STC CONTROL PORT )
6   : DLOAD   DATAPORT C! ;  ( STACK ---> STC DATA PORT )
7   : SREAD   CNTRLPORT C@ ;  ( STC STATUS REGISTER ---> STACK )
8   : DREAD   DATAPORT C@ ;  ( STC DATA PORT ---> STACK )
9
10
11
12
13
14
15

Block Number:  47
0   ( DATA PROCESSING VARIABLES AND CONSTANTS)
1
2   DECIMAL
3
4   ( STC clock frequency )
5       5,000,000 2CONSTANT #CLOCK
6
7   ( Data parameter variables )
8       VARIABLE #MINERROR
9       VARIABLE #ERSQUINV
10      VARIABLE #MAXTIME 2 ALLOT
11      VARIABLE #COUNTMAX 2 ALLOT
12      VARIABLE #IGCOUNT 2 ALLOT
13      VARIABLE #IGTIME 2 ALLOT
14
15
```

```
Block Number:  48
0   ( PARAMETER SETTING )
1
2   DECIMAL
3
4   ( Convert time-limit into count-limit )
5       : $TIMESET    2DUP #MAXTIME 2! #CLOCK 1000 M/ 1000 M*/
6                     #COUNTMAX 2! ;
7
8   ( Convert error-limit into inverse-error-squared )
9       : $ERRORSET    DUP #MINERROR ! DUP * 10000 SWAP /
10                     #ERSQINV ! ;
11
12
13
14
15

Block Number:  49
0   ( PARAMETER CALCULATIONS )
1
2   DECIMAL
3
4   ( Calculate elapsed counts during integration period )
5       : $IGCOUNT    #COUNTMAX 2@ #ACOUNT 2@ D-
6                     #IGCOUNT 2! ;
7
8   ( Convert elapsed counts to elapsed time in us )
9       : $IGTIME    #IGCOUNT 2@ 1 #CLOCK 1000 M/ 1000 / M*/
10                   #IGTIME 2! ;
11
12
13
14
15

Block Number:  50
0   ( A/D RELATED )
1
2   DECIMAL
3
4   ( A/D LSB Volts = .0024414 Volts )
5       24414 CONSTANT #VLSB
6
7   ( A/D variables )
8       VARIABLE #ADCELL
9       VARIABLE #ADVOLTS
10
11  ( Format A/D bytes into single cell )
12      : ADWEIGHT   ADMSB 16 * ADLSB 16 / + #ADCELL ! ;
13
14  ( Multiply A/D binary output x LSB value )
15      : BTOV   #VLSB #ADCELL @ M* 10000 M/ #ADVOLTS ! ;

Block Number:  51
0   ( UTILITIES )
1
2   HEX
3
4   ( Non-destructive stack printout )
5       : .S   CR 'S S0 @ 2 - DO I @ . -2 +LOOP ;
6
7   ( Set base to binary )
8       : BINARY   2 BASE ! ;
9
10  ( Print unsigned 32-bit number from stack )
11      : UD.    <# #S #> TYPE ;
12
13
14
15
```

```
Block Number:  52
0   ( STC REGISTERS' ASSIGNMENTS )
1
2   ( Counters 1 and 2 are concatenated as a single 32-bit
3     counter, as are counters 3 and 4; counter 5 is used as an
4     event counter.  Counters 1-4 are programmed in operating
5     mode "B" with active low gate, 5MHz source, and active
6     low TC.  Counter 5 is programmed in operating mode "A"
7     with TC inactive, output low. )
8
9   HEX
10
11  ( Mode registers' assignments: )
12      CREATE #CMR BB05 , B005 , BB05 , B005 , 0B00 ,
13
14  ( Hold registers' assignments: )
15      CREATE #HLR 0 , 0, 0 , 0 , 0 ,

Block Number:  53
0   ( STC INITIALIZATION )
1
2   HEX
3
4   ( Load LSB followed by MSB into the STC registers )
5       : $BYTER   DUP C@ DLOAD 1+ C@ DLOAD ;
6
7   ( Load "mode" register with firmware values )
8       : $MODER   I' 2 * #CMR + $BYTER ;
9
10  ( Load "load" register with programmable count-limit values )
11      : $LODER   I' 2 MOD IF #COUNTMAX $BYTER
12                 ELSE #COUNTMAX 2+ $BYTER THEN ;
13
14  ( Load "hold" register with firmware values )
15      : $HODER   I' 2 * #HLR + $BYTER ;

Block Number:  54
0   ( STC INITIALIZATION )
1
2   HEX
3
4   ( Event counter loader, presently not in use )
5       : $EVENTER   10 DROP ;
6
7   ( Software reset STC )
8       : $RESET-STC   FF CLOAD 5F CLOAD ;
9
10  ( Initialize STC by performing element cycle )
11      : $ECYCLE   01 CLOAD 4 0 DO $MODER $LODER $HODER LOOP
12                  $EVENTER ;
13
14
15

Block Number:  55
0   ( STC READING )
1
2   HEX
3
4   ( STC Counter output variables )
5       VARIABLE #ECOUNT           ( Event count )
6       VARIABLE #PCOUNT 2 ALLOT   ( Pulse count )
7       VARIABLE #ACOUNT 2 ALLOT   ( Timer-analog count )
8
9   ( Store counter outputs, on stack, into memory )
10      : $ESTORE   0 1 DO I 2 MOD #ECOUNT + C! -1 +LOOP ;
11      : $ASTORE   2 5 DO I 4 MOD #ACOUNT + C! -1 +LOOP ;
12      : $PSTORE   2 5 DO I 4 MOD #PCOUNT + C! -1 +LOOP ;
13      ( Bytes on stack are sequenced into FORTH-compatible
14        representations for 16- and 32-bit numbers )
15
```

```
Block Number:  56
0   ( STC READING )
1
2   HEX
3
4   ( Execute "hold" cycle, put all values on the stack )
5       : $READER   19 CLOAD A 0 DO DREAD LOOP ;
6
7   ( File the values from the stack into memory )
8       : $FILER    $ESTORE $PSTORE $ASTORE ;
9
10  ( Read and store "hold" register values )
11      : $HCYCLE   $READER $FILER ;
12
13
14
15

Block Number:  57
0   ( I/O )
1
2   DECIMAL
3
4   ( Print the binary A/D output value )
5       : ADOUT?    BASE @ ADWEIGHT #ADCELL @ BINARY 0 <# 12 0
6                   DO # LOOP #> TYPE ." (AD11-AD0) " BASE ! ;
7
8   ( Print the decimal A/D output value )
9       : ADVOLTS?    BASE @ BTOV #ADVOLTS @ DECIMAL 0 <# # # #
10                    46 HOLD # #> TYPE SPACE ." V " BASE ! ;
11
12  ( Print the monitor port status, in bits )
13      : MPORT?    BASE @ ADLSB 16 MOD BINARY
14                . ." (M3-M0) " BASE ! ;
15

Block Number:  58
0   ( I/O )
1
2    DECIMAL
3
4   ( Print the time-limit or error-limit setting )
5       : $TMAX?    #MAXTIME 2@ UD. ." us " ;
6       : $EMIN?    #MINERROR @ . ." % " ;
7
8   ( Print the elapsed integration time )
9       : $INT-TIME?    $IGTIME #IGTIME 2@ UD. ."  us " ;
10
11  ( Connect, convert, and print the C/V or integrator output )
12      : $CV?    AMPOUT ADCNVRT ADVOLTS? ;
13      : $INT?    INTGOUT ADCNVRT ADVOLTS? ;
14
15
```

*** BIBLIOGRAPHY ***

1. Porter, C.J., Benson J.H., Ast T., "The Modern Mass Spectrometer --- A Complete Chemical Laboratory", Organic Mass Spectrometry, Vol. 16, No. 3, 1981.

2. Ligon, W.V., "Molecular Analysis by Mass Spectrometry", Science, Vol. 205, July 13, 1979.

3. McLafferty, F.W., Interpretation of Mass Spectra, 3rd ed.; University Science Books, 1980.

4. Yost, R.A., Enke C.G., "Triple Quadrupole Mass Spectrometry for Direct Mixture Analysis and Structure Elucidation", Analytical Chemistry, Vol. 51, No. 10, October, 1979.

5. Fall School in Practicle Quadrupole Mass Spectrometry (notes), Extranuclear Laboratories, Inc., October, 1980.

6. Kurz, E.A., "Channel Electron Multipliers", American Laboratory, Vol. 11, No. 3, March, 1979.

7. Newcome, B.H., Enke C.G., "A Modular Twin Bus Microprocessor System for Laboratory Automation", in preparation for submission to Review of Scientific Instrumentation.

8. Willard, H.H., Merrit, L.J., Dean J.A., Settle, F.A., Instrumental Methods of Analysis, 6th ed.; D. Van Nostrand Co., 1981.

9. Darland, E.J., Leroi, G.E., Enke C.G., "Maximum Efficiency Pulse Counting in Computerized Instrumentation", Analytical Chemistry, Vol. 52, No. 4, April, 1980.

10. Gregg, H.G., et. al., "An Automated Triple Quadrupole Mass Spectrometer", presented at 29th ASMS Annual Conference on Mass Spectrometry and Allied Topics, May, 1981.

11. "8085A Single Chip 8-Bit N-Channel Microprocessor", MCS-85 User's Manual, Intel Corp., September, 1978.

12. Dual-Mode Channeltron Electron Multiplier (prelim. data), Galileo Electro-Optics Corp., October 1980.

13. Kurz, E.A., "The Effect of High Input Current Pulses Upon Channel Electron Multipliers", Review of Scientific Instrumentation, Vol. 50, No. 11, November, 1979.

14. PAD-400 Pulse Amplifier Discriminator (eng. spec.), Galileo Electro-Optics Corp., September, 1978.

15. Malmstadt, H.V., Enke, C.G., Crouch, S.R., Electronics and Instrumentation for Scientists; The Benjamin/Cummings Publishing Co., Inc., 1981.

16. Am9513 System Timing Controller (data sheet), Advanced Micro Devices, Inc, 1980.

17. ICL7650 Chopper Stabilized Operational Amplifier (data sheet), Intersil, Inc., 1980.

18. "DG211 Quad Monolithic SPST CMOS Analog Switch", Analog Switch Data Book, Siliconix, Inc., 1980.

19. AD574 Fast, Complete 12-Bit A/D Converter with Microprocessor Interface (data sheet), Analog Devices, 1980.

20. Voltage Regulator Databook, National Semiconductor Corp., 1980.

21. Peripheral Design Handbook, Intel Corp., August, 1980.

22. Kimball, B., "FORTH 'Breaks the Ice' in Software Development", Digital Design, June, 1981.

23. Brodie, L., Starting FORTH; Prentice-Hall, Inc., 1981.

24. Rather, E., Brodie, L., Rosenberg, C., Using FORTH; FORTH, Inc., 1980.