ABSTRACT

DEVELOPMENT OF INTERACTIVE OPERATING SYSTEM SOFTWARE FOR TRIPLE QUADRUPOLE MASS SPECTROMETRY

By

Carl Alan Myerholtz

A convenient command structure was needed for the computer-automated control of all instrument parameters and scanning modes of a triple quadrupole mass spectrometer (TQMS) instrument operating in the research environment. The interactive software control system developed to meet this need aids in the optimization of the many parameters that affect the ion beam. It also implements the various scanning procedures used in TQMS applications (daughter, parent and neutral loss scans).

The operating system software was written in the programming language FORTH. Due to the extensible nature of the FORTH language, the resulting software becomes essentially a high level language for programming control of our TQMS instrument.

The capabilities of the control system include: automatic sequencing of experimental modes, scanning any system parameter and generating either a linear or a logarithmic display of the result, and generating three dimensional plots of any two instrument parameters verses ion current.

DEVELOPMENT OF INTERACTIVE OPERATING SYSTEM SOFTWARE FOR TRIPLE QUADRUPOLE MASS SPECTROMETRY

By

Carl Alan Myerholtz

A THESIS

Submitted to

Michigan State University

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

Department of Chemistry

1982

To my parents

ACKNOWLEDGMENTS

I would like to express my thanks to Dr. Chris Enke under whose guidance this research was performed. I thank the members of the Mass Spectrometry Research Group for their support, especially the contributions of Mr. Bruce Newcome.

Financial support for myself was provided by Michigan State University and a grant from the National Aeronautics and Space Adminisration (NASA). Financial support for the TQMS instrumentation project was provided by the Office of Naval Research (ONR).

Finally I would like to express my appreciation and graditude to my parents and my friend Marie for the their support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Introduction to Triple Quadrupole Mass Spectrometry

The triple quadrupole mass spectrometer is a versatile analytical instrument as well as a flexible research tool. It is still a relatively new instrument, with the first one being assembled in our laboratory in 1978 by Rick Yost[1]. A TQMS instrument incorporates many of the features of a conventional quadrupole mass spectrometer, while providing a variety of powerful new operating modes. The basic configuration of a triple quadrupole mass spectrometer is shown Figure 1.1. A TQMS instrument consists of an ion source, two quadrupole mass filters, one quadrupole collision chamber, ion lenses for interquadrupole transmission, and finally a detector after the third quadrupole for measuring the ion current.

The heart of a TQMS instrument is the quadrupole collision chamber. Here ions selected by the first quadrupole mass filter undergo collision with neutral gas molecules. The collision gas can either be chemically reactive toward the selected ions, which results in addition products, or more commonly an inert gas such as argon. Collisions with an inert gas usually produce charged and neutral fragments in a process called collisionally

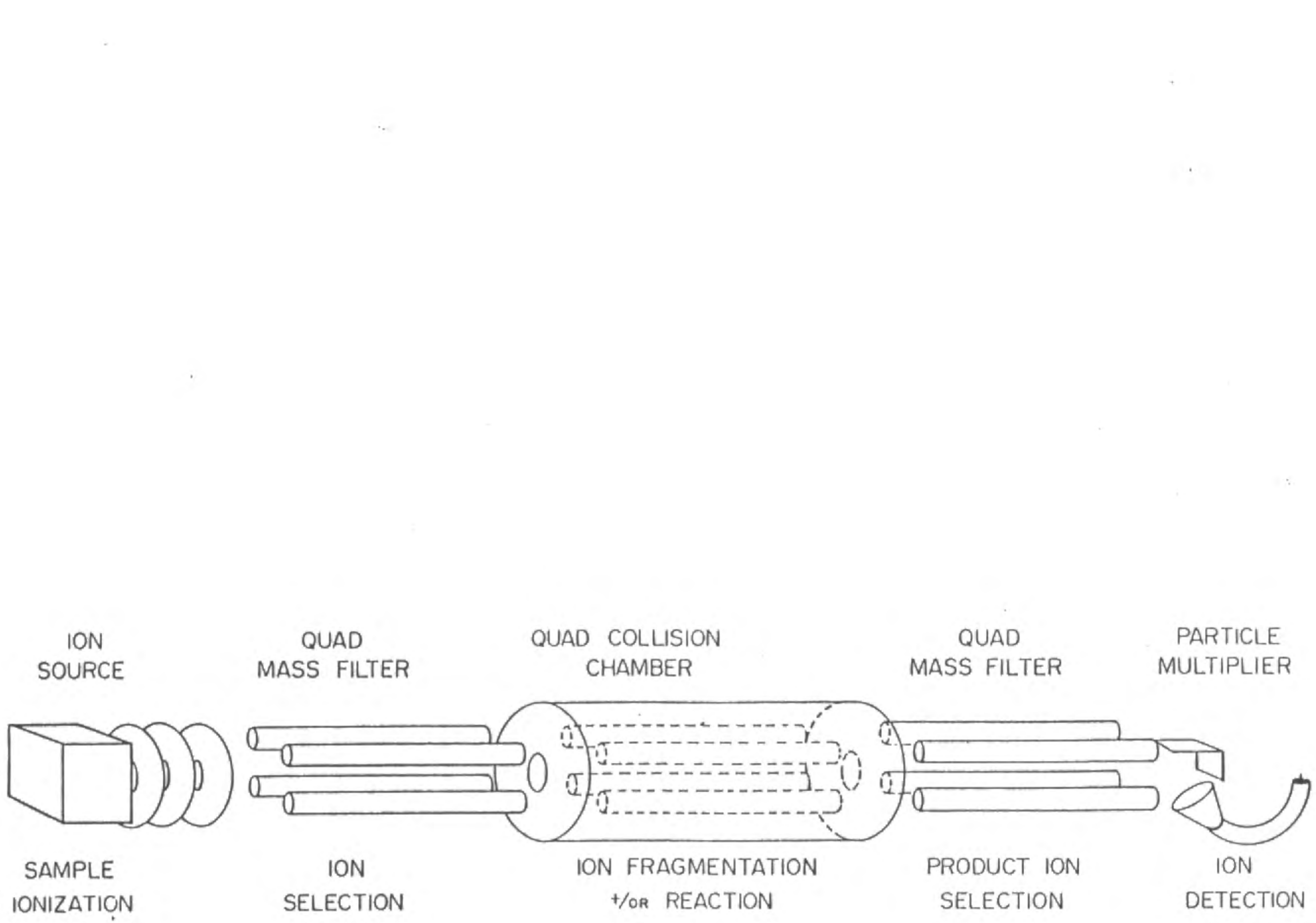


Figure 1.1 TQMS Instrument Block Diagram

activated dissociation (CAD), or collision-induced dissociation (CID)[2]. The resulting ions can then be analyzed using the second quadrupole mass filter.

A quadrupole can be operated in one of two modes which are referred to as the "DC" and "RF" modes[3]. When a quadrupole is operated in the "DC" mode it acts as a mass filter and allows only ions of a selected mass to pass through it. In the "RF" mode, a quadrupole acts as an ion pipe and is transparent to ions of all masses. In a TQMS instrument, each quadrupole can be set to either the "RF" or "DC" mode, the later being used to pass ions of only one mass, or to scan over a mass range to generate a mass spectrum. These various modes for each quadrupole can be combined in many ways to provide different modes of operation for the instrument. For example quadrupole one can be set in the "DC" mode, quadrupole two into the "RF" mode, and quadrupole three into the "DC" mode. This instrument configuration is referred to as the DC/RF/DC mode. The five most useful combinations of the three quadrupole modes are as follows:

Quadrupole One Scan - In this mode the instrument is set in the DC/RF/RF mode and quadrupole one is scanned. In this mode the instrument functions as a conventional quadrupole mass spectrometer.

Quadrupole Three Scan - In this mode the instrument is set in the RF/RF/DC mode and quadrupole three is scanned. In this mode the instrument functions as a conventional quadrupole mass spectrometer.

Daughter Scan - In the mode the instrument is set in the DC/RF/DC mode with quadrupole one set to a user-selected mass while quadrupole three is scanned. This type of scan generates a spectrum of all the fragment ions (daughter ions) which are generated by the selected (parent) ion when it undergoes the CAD process in quadrupole two.

Parent Scan - In this mode the instrument is set in the DC/RF/DC mode with quadrupole three set to a user-selected mass while quadrupole one is scanned. A parent scan generates a spectrum of all the ions formed in the ion source (parents) that generate the selected fragment ion (daughter) upon undergoing the CAD process in the collision cell.

Neutral Loss Scan - In this mode the instrument is set in the DC/RF/DC mode and quadrupoles one and three are scanned together. Quadrupoles one and three are offset by a constant mass with quadrupole one set to pass a higher mass than quadrupole three. This type of scan generates a spectrum of all the ions formed in the ion source (parents) that lose a selected neutral fragment when undergoing the CAD process in the collision cell. The magnitude of the neutral loss is selected by the mass offset between quadrupoles one and three. It is also possible to scan for a mass gain due to a reaction in the collision chamber.
Figure 1.2 diagrams these modes of operation.

In addition to these types of mass scanning experiments, several other useful types of experiments can

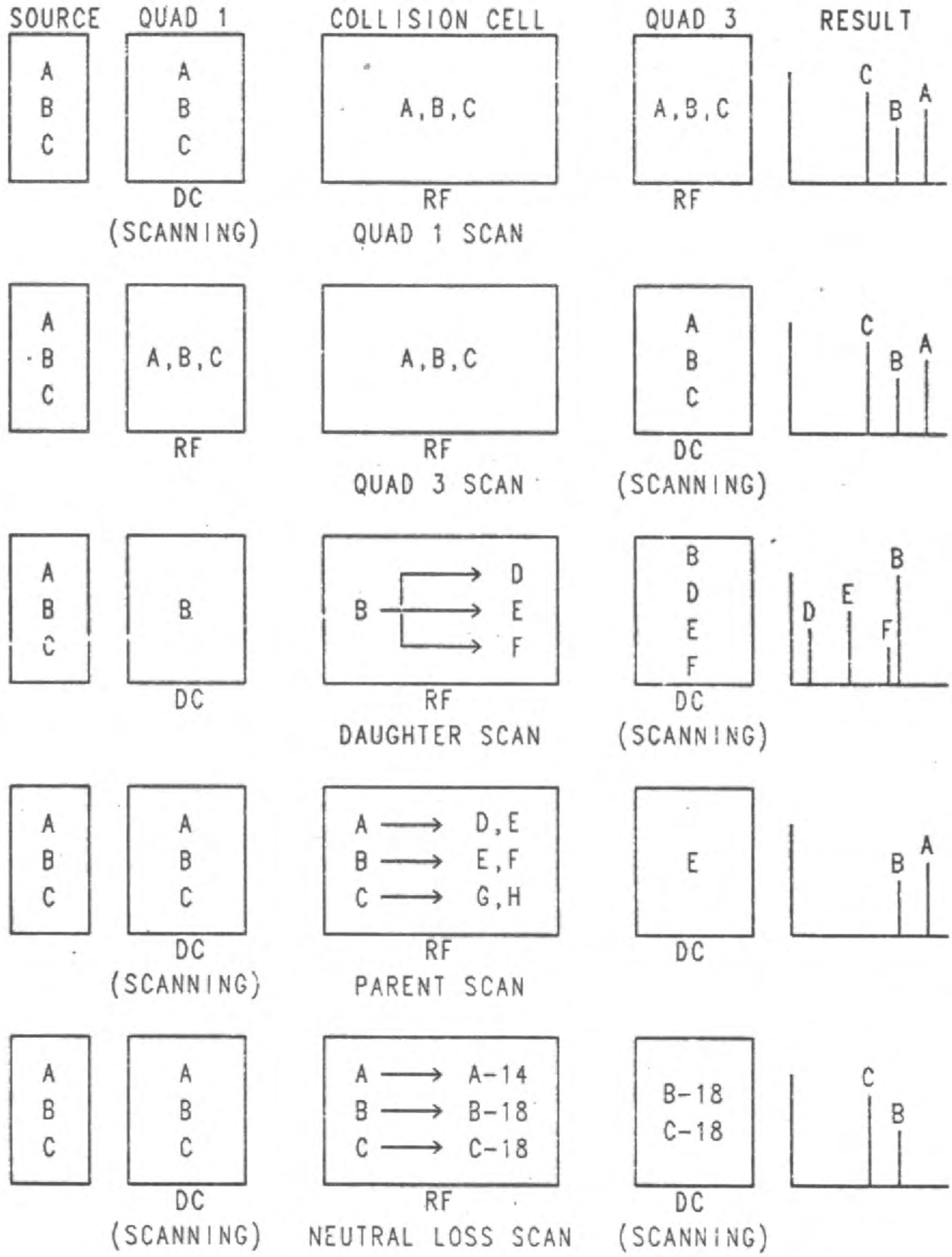


FIGURE 1.2 TQMS OPERATING MODES

easily be performed with a TQMS instrument. The axial or kinetic energy an ion possesses when it undergoes a collision in quadrupole two can readily be varied. From the changes in fragmentation patterns and fragment ion intensities that result from varying the axial energy, information about bond strengths and internal energies of the parent ion can be obtained. By varying the pressure of the collision gas in quadrupole two it is possible to determine the collisional cross section of an ion, identify metastable species, and determine the reaction order of the CAD reactions[4].

As can be seen from the variety of experiments which can be performed , TQMS is capable of providing a great variety and quantity of information. Potential applications of this versatile instrument include; mixture analysis, structure elucidation and screening complex mixtures for specific compound types.

Need for a Control System

The many modes of operation and large numbers of variables involved in the ion path contribute to both the versatility and complexity of a TQMS instrument. The variety of operating modes makes it a very powerful analytical tool, yet this variety greatly increases the difficulty of controlling the instrument. It was evident that in order to become a viable research tool, some degree of computer control had to be introduced into the chemist/instrument interaction loop. Most mass spectrometer systems today

utilize a minicomputer system to acquire and report data[5,6,7]. The computer system and it's associated software are often referred to as a "data system". Many of these data systems have very little control over the instrument, often they can only initiate a mass scan upon command[5]. Only limited control of the instrument is needed by these computer systems since in a conventional mass spectrometer there is only one significant variable, mass. The operator has to optimize only a few lens settings to get a good ion beam, and then use the data system to acquire data.

This is not the case with a TQMS instrument. The major experimental variables include the mass selected by each quadrupole, the axial energy of the ions, and the pressure in the collision cell. In addition, the three quadrupoles introduce the need for many ion lensing elements which adds greatly to the complexity of instrument operation. The demands placed upon the data system by the five different mass scanning modes of TQMS are not greatly different than those of a conventional mass spectrometer. Indeed, several commercial data systems have been modified to function with TQMS instruments[6,8]. However, to take advantage of the other types of experiments TQMS is capable of providing, such as axial energy and collisional cross section studies, a great deal more control of the instrument is needed.

To utilize all of the different operating modes effectivelu and to simplify operation of this very complex instrument, an intelligent control system is needed. This system must go beyond conventional data systems, for in

addition to performing mass scanning experiments, it must support other experimental modes as well. It must also be flexible and expandable enough to incorporate new experimental modes as they are developed. While doing all this, the control system must have the net effect of making the instrument easier and more efficient to use instead of complicating the operation by presenting the operator with more complex control procedures.

Control System Objectives

The primary goal of the control system is to provide an effective interface between the operator and the instrument. The entry and display of information should be in a form the operator is already familiar with or can readily understand[9]. The control system must reduce the complex variety of operations into a few simple, well-understood operations for the operator to learn. Data should be displayed graphically since this is how mass spectral information is traditionally seen. Values of the various parameters should be entered and displayed in terms of familiar units. While accomplishing all of this, the control system must also remain flexible and open ended so that it does not become the limiting factor in the instrument's capabilities and performance.

For ease of operation and for use by operators without extensive expertise, as much of the operation of the instrument as possible should be automated. When switching among the various modes of operation, the control system, not the

operator should be responsible for changing all the necessary parameters. For example, when selecting one of the mass scanning modes, the control system should select the appropriate "RF" or "DC" mode for each quadrupole. In so far as possible, the system should be self calibrating, so that the operator need not be concerned with long term drift of various instrument parameters such as lens potentials. Whenever possible the control system should be capable of detecting malfunctions of error conditions in the hardware of the instrument and be able to report them to the operator.

Our TQMS instrument, operating in the research environment, is in a continuous state of evolution. Various components are constantly being modified or replaced to try new experiments or to improve instrument performance. The control system must be readily adaptable to these changes in instrument configuration. New pieces of electronic hardware should be easily interfaced to the control system and such changes should be transparent to the operator in terms of instrument operation.

In order to be able to take full advantage of the instrument's capabilities, the operator must be able to design and program his or her own experiments. Instructing the control system to perform relatively simple types of new experiments should not require any programming expertise. Ideally, instructions to perform new experiments should not be significantly different from routine instrument

operation. However, the control system must remain open ended enough so that the truly experienced programmer is not limited and can readily implement specialized and more exotic modes of operation.

Some of the basic requirements of an intelligent control system for a TQMS instrument, (which also apply for all instruments) have been covered in this chapter. The following chapters will deal with some of the major factors in developing such a control system for the TQMS instrument in our laboratory.

CHAPTER 2

HARDWARE OF THE TQMS SYSTEM

The hardware necessary to implement and control TQMS can be broken down into six categories: control system computer; mass selection; data acquisition; ion lens control; mass storage; and display. The computer system is the heart of the electronics system; devices in all of the other categories are interfaced to the computer which controls the operation of the entire instrument. The only part of the instrument not under computer control is the vacuum system. Figures 2.1 and 2.2 are block diagrams of the physical layout of the TQMS instrument in our laboratory.

Control System Computer

In our laboratory we have developed our own modular microprocessor system[10]. This system is build around Intel's 8085 microprocessor and its associated family of peripherals. The laboratory computer system consists of a set of small circuit boards each implementing a different function. The following standard modules have been developed to date:

1) Central Processing Unit (CPU)
2) 8K/16K RAM/ROM Board
3) Dual UART
4) Interrupt Controller
5) Chip Select

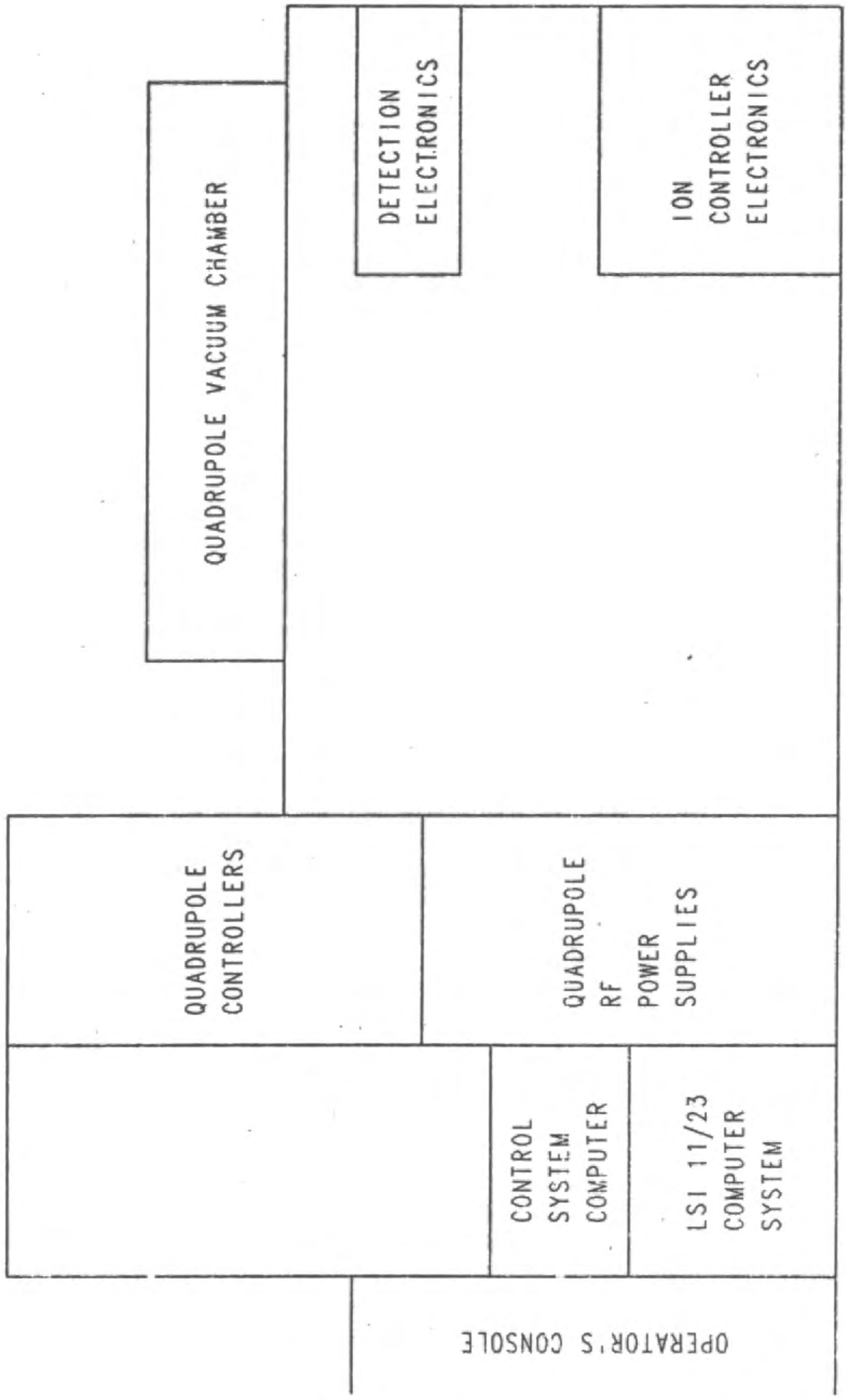


FIGURE 2.1 PHYSICAL LAYOUT

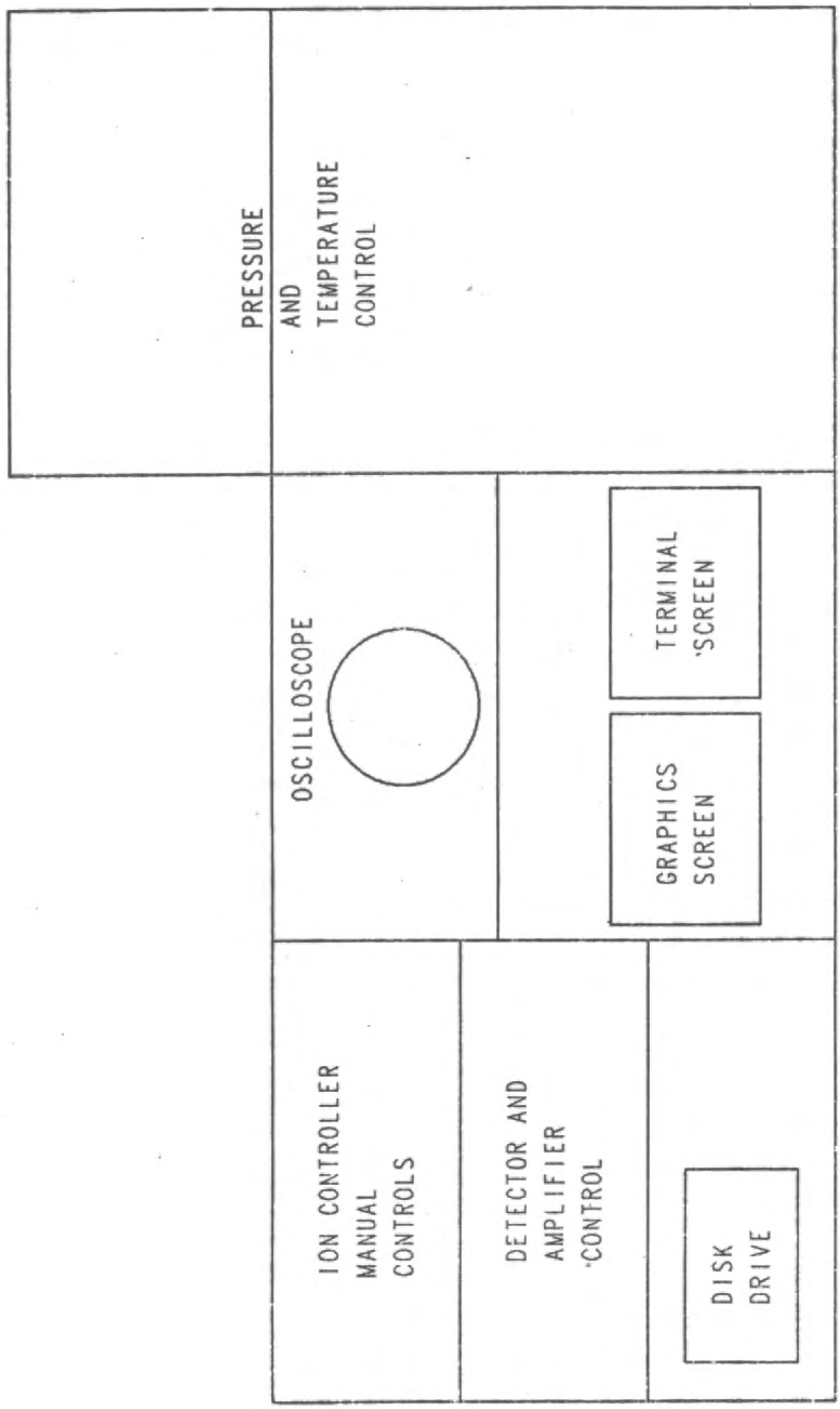


FIGURE 2.2 OPERATOR'S CONSOLE

6) Terminator

7) Parallel Port

8) 8K RAM Memory Board

9) 9513 Counter/Timer

10) Transceiver Board

Modules implementing the desired functions can be mounted on one or more motherboards to provide a computer system with the desired combination of functions.

The control system computer currently consists of four motherboards configured as follows: Motherboard one contains a CPU board, an 8K/16K RAM/ROM board, a terminator, a chip select board, a dual UART board, an interrupt controller board and a tick clock that generates an interrupt every millisecond. The RAM/ROM board is configured in the 16K mode and contains 8K of ROM and 8K of RAM. The 8K ROM contains the FORTH language system which is discussed in more detail in chapter three. Motherboard two contains an 8K RAM board used for program and data storage, a chip select board and two parallel port boards. The parallel ports are used to interface a variety of devices which will be covered later in this chapter. Motherboard three contains an RAM/ROM board, a chip select board and two transceiver boards. The RAM/ROM board is configured in the 8K mode and is populated with one 2K EPROM chip which serves as a software character generator for the graphics display. The two transceiver boards are used to communicate with special interfaces located outside of the computer rack. Motherboard four contains a chip select board, an 8K RAM board used for

program and data storage and special interfaces used to communicate with the ion controller electronics, which will be discussed later.

Mass Selection

The mass selection achieved by a quadrupole mass filter is a result of the direct current (DC) and radio frequency (RF) fields applied to the quadrupole rods. When only an RF field is applied, a quadrupole passes ions over a wide range of masses. In the mass filter mode, the ratio between the RF and DC field strengths determines the mass selected. The quadrupole control electronics for our Extranuclear quadrupoles feature an external command input which allows a current varying between 0 and 1 milliamp to control the mass selection[3].

Computer control of the quadrupole is accomplished with a special interface located in the same rack (shown in Figure 2.1) as the quadrupole control electronics to minimize the length of the analog signal lines. The interface is actually a limited version of the computer system's address and data bus that is extended by use of one of the transceiver boards on motherboard three. Two 16-bit digital-to-analog converters (DAC's) are connected to this interface and are used to control the mass selection of quadrupoles one and three. A 12-bit DAC is used to control the mass selection of quadrupole two since less accuracy is required here and the 16-bit DACs are relatively expensive. Figure 2.3 is a block diagram of the mass selection control electronics and interface.The interface and 16-bit DACs were

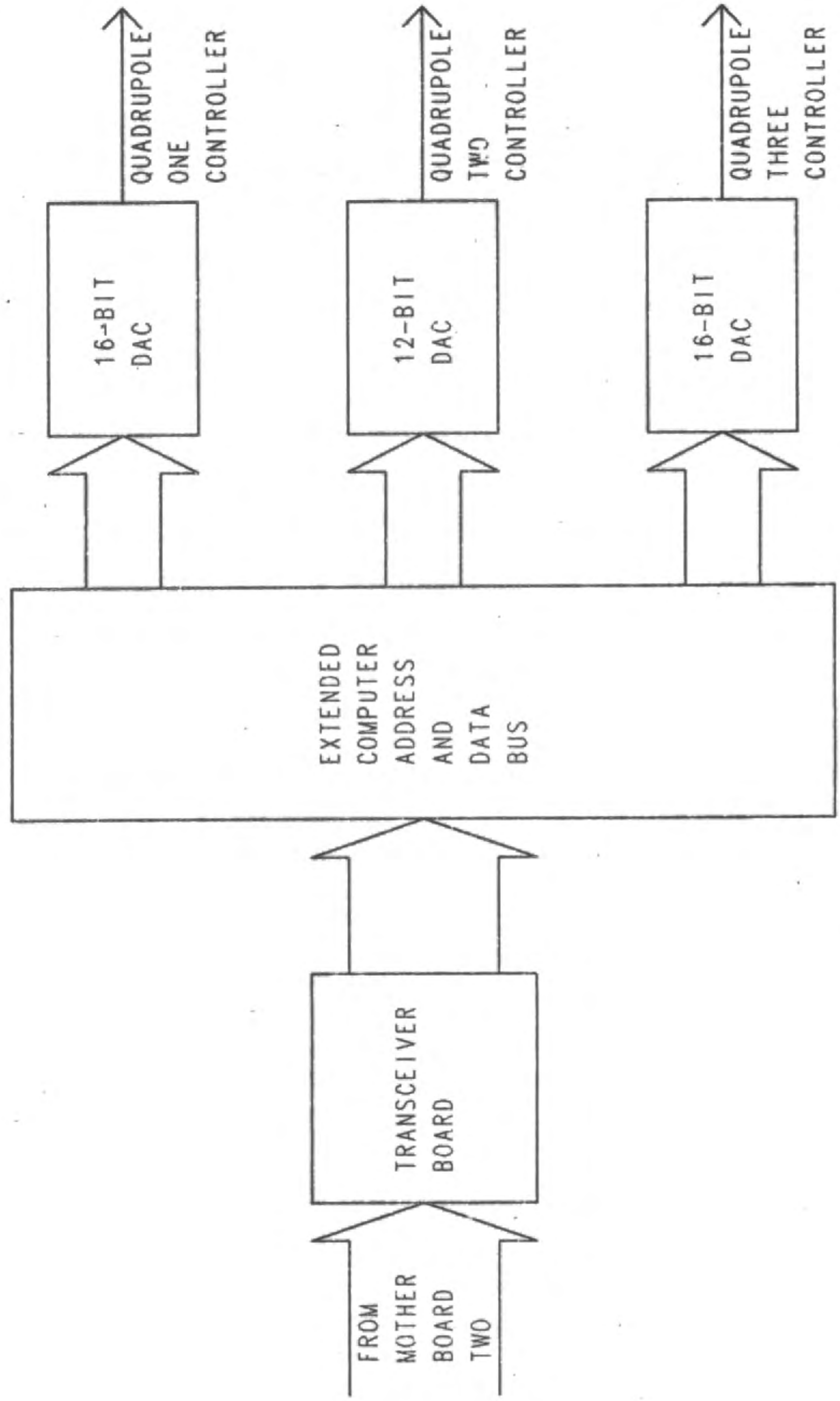


FIGURE 2.3 MASS SELECTION CONTROL

designed and built by Bruce Newcome[11]. Mechanical relays have been added to the quadrupole controllers to allow remote selection of the RF only and DC/RF operating modes. The DC/RF mode is often referred to as the DC mode. These relays are controlled with three bits from one of the parallel ports on motherboard two.

Data Acquisition

The data acquisition hardware consists of an electrometer amplifier with digital range selection and a 12-bit analog to digital converter (ADC) with a 8 usec conversion time[12]. The amplifier has five ranges; each successive range represents a factor of ten increase in gain over the previous range. The amplifier range can be selected by the operator from a switch on one of the front panels or by the computer system. The operation of the electrometer amplifier is discussed in detail in chapter five. Three bits of one of the parallel ports on motherboard two are used to accomplish range selection under computer control. The output of the Channeltron[13] detector is fed into the amplifier and the output of the amplifier is connected to the 12-bit ADC. A block diagram of the data aquisition electronics is shown in Figure 2.4. The box containing the amplifier and ADC is located at the far end of the instrument near the detector to minimize the cable length for the low level Channeltron^{TM} output signal. The analog information is converted to digital form at this point to eliminate noise pickup as the signal is returned through about 15 feet of cable to the computer system. The digital

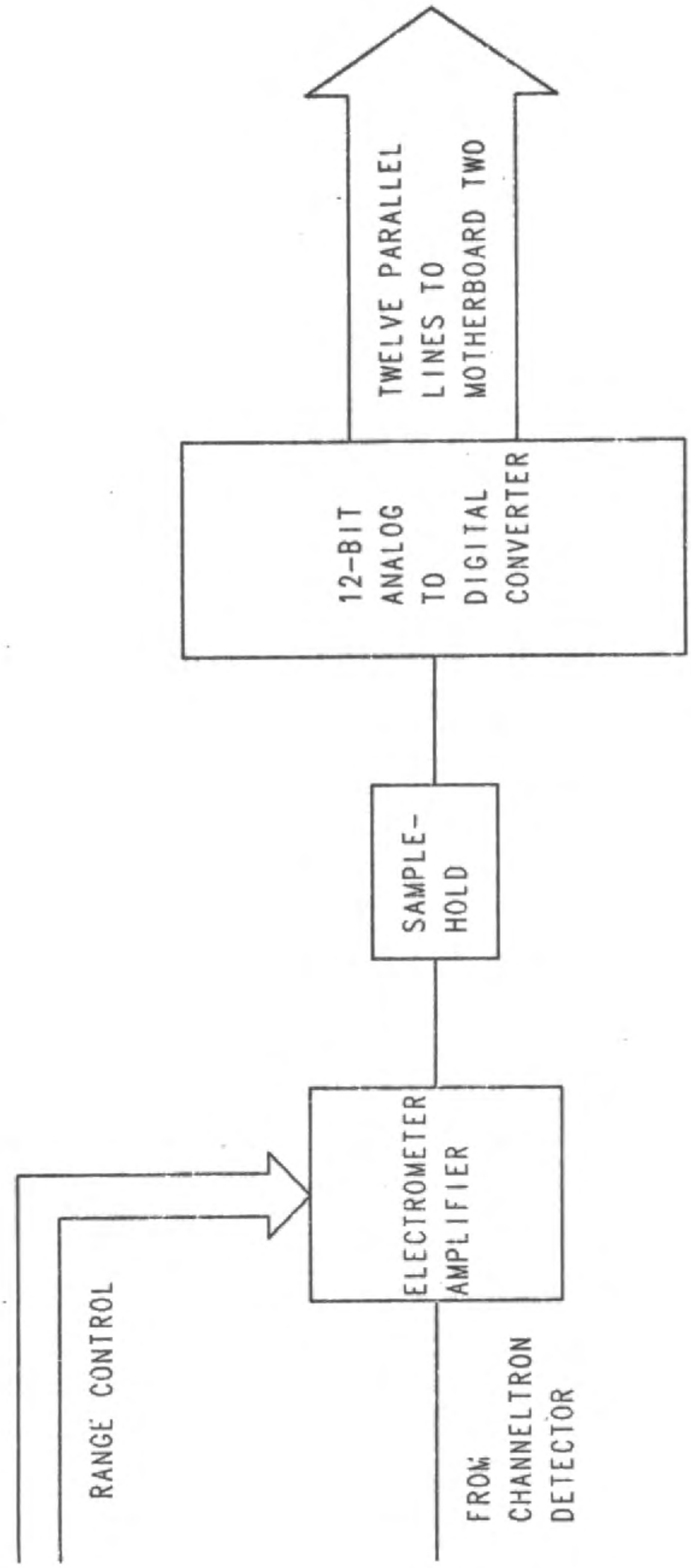


FIGURE 2.4 DATA ACQUISITION ELECTRONICS

output of the ADC is read by the computer system through 12-bits of one of the parallel ports on motherboard two. All of the data acquisition electronics were designed and constructed by me as part of this thesis work.

Ion Lensing

The potentials applied to the various lenses and source elements in the system are controlled by a set of electronics referred to as the ion controller. The ion controller electronics were designed and constructed by Phil Hoffman[14]. The ion controller supports sixteen 12-bit digital to analog converters (DACs), each one controlling the potential of one lens element. The output of the 12-bit DACs varies between -10 and +10 volts. The output of each DAC is fed to a booster amplifier that generates a +/- 200 volt output for a +/- 10 volt input. This output is then connected to the ion path element to be controlled. The resolution of the 12-bit DACs is one part in 4096 which, over the 400 V output range is about 0.1 volts. This is adequate precision for this application. The 12-bit DACs reside in a rack at the far end of the instrument (shown inf figure 2.1) and are interfaced to the microcomputer system through the special interface electronics located on motherboard four.

Mass Storage

Mass storage for programs and data is provided by a Persci model 277 dual floppy disk drive. This drive is capable of accessing two 8-inch floppy disks for a total on-line storage of 0.5 megabytes[15]. The floppy disk drive is

CHAPTER 3

Selection of the FORTH Language System

This chapter will deal with some of the considerations involved in selecting the FORTH language system and some of the logistics involved in implementing FORTH in our laboratory. Some of the requirements of control application programming will be discussed and traditional language systems will be examined to see how they fit these requirements. The features of FORTH will then be examined and the need for software development systems will be discussed.

Programming Languages and Control Applications

Digital computers have been applied in many areas to increase the speed and efficiency of a diversity of operations and to solve problems previously unsolvable. Early in their development, most of the attention in computer hardware and software development focused on the computer's numeric processing capability (commonly referred to as 'number crunching'). This led to the development of the FORTRAN programming language in 1954[20] which emphasized the ability to translate mathematical expressions into a computer program, hence the name FORmula TRANslation. FORTRAN features the ability to manipulate floating point numbers and includes a library of mathematical operations

language is to make the computer easier to use. Since these systems interact with people, they normally use relatively standard input/output (I/O) devices such as video terminals, printers, card readers and disk and tape drives. Interfaces to standard devices are readily available and most operating systems have software to drive these types of devices. These relatively large computing systems systems are often used by many people so that the expense for large amounts of memory and storage is spread over many users.

Computers used in control systems occupy a different place in the total scheme of computer applications. In numeric processing and data management applications the computer is the focus of attention, the tool being used. By contrast, computers used in control applications are just part of the tool, not necessarily even a major part. When computers are the tool being used, tasks can be adapted to fit the computer environment (which for the most part is determined by the software being used). In the control environment the computer system must adapt to the needs of the task. This is a major difference in philosophy which has been largely overlooked in the development of programming languages. Most programming languages are designed to have features or extensions which make programming for certain types of applications easier. In the control environment each application has its own unique problems and needs a customized language to make programming simpler and more efficient. This is where FORTH comes in.

Before discussing the FORTH language system, the needs

of common control applications and the inadequacy of traditional languages to meet these needs will be examined. In a control application, a computer must be interfaced to the system it controls, whether it be a building's heating system, a scientific instrument, or an electronic game. The interface to any given device may be broken down into two components: the hardware, which provides an electronic pathway for communications between the device and the computer, and the software, which is necessary to send and receive information over this pathway. Each device the computer is to control (such as a stepper motor, a relay, or a scientific instrument) that is not a standard computer peripheral device, requires its own specific hardware and software interface. The software portion of an interface is often referred to as a device driver even if it only receives information from the device. Microprocessors are often used in control systems since interfaces to them are relatively simple to design and they are generally inexpensive.

The software to drive a device usually must be written in assembly language to perform the necessary I/O operations or to respond to interrupts. Any language system used to program a control application should have the ability to generate assembly language routines to perform the most basic level of software interface. A control system must be able to respond to changes in the real world as they occur. This makes it necessary to take the speed of program execution into consideration as well as having the ability to use

interrupts to synchronize events. If an analog to digital converter (ADC) presents a new value to the computer every millisecond and the software to process a value takes five milliseconds to execute, data will be lost. Finally, the computer in a control application is only a fraction of the total system and is basically performing only one function. Thus it is difficult to justify spending money on large quantities of memory and mass storage. Thus programs should be kept as small as possible. The following list briefly summarizes the software needs of most control applications.

1) Assembly Language Programming Ability

2) Speed

3) Compact Size

Traditional Languages

A number of traditional language systems are available and have been used to program computer systems in control applications. These can be broken down into three major types of languages systems: assembly-level languages, interpreter and compiler-based higher level languages.

In assembly language programming the programmer writes a program using mnemonic abbreviations for each machine operation which an assembler program then converts directly into machine code. As a consequence, well-written programs in assembly language execute at near maximum speed and make reasonably efficient use of memory. However, assembly language programming is quite tedious and time consuming.

Usually an experienced programmer is needed to program an entire application in assembly language. In addition to the assembler, an editor is needed to enter the source code into the computer. The editor and assembler programs usually run under some operating system which requires the programmer to be familiar with the idiosyncrasies of the operating system as well as the editor and assembler programs.

Since programming in assembly language is slow and tedious, a high level language is often used to make programming easier. Unlike assembly language which allows complete access to the computer and is unique to each type of processor, a high level language divorces the user from the internal operations of the computer and the structure of it's instruction set and allows programs to be written with more English or algebra-like commands. High level languages fall into the remaining two language types: interpreted and compiled.

An interpreter based language is a language in which program execution is controlled by data elements rather than machine code. The most common interpretive language is BASIC. BASIC is an inefficient language system in terms of machine usage and speed. In BASIC, the entire program must be stored in memory in either text or compressed text form which occupies a great deal of memory. The entire BASIC interpreter must be in memory to execute a program written in BASIC. This adds a considerably to the amount of memory needed to execute a BASIC program. Execution of a BASIC program is very slow since each time a line of the program

is executed, the BASIC interpreter must scan the line of text and look up the operation to perform. Table 3.1 is a comparison of the memory usage and execution times of two programs, one written in BASIC and the other assembly language.

	BASIC	Assembly
Bytes of Code		
Interpreter	17,920	0 (1)
Main Program	5,507	2,224
Variables	329	0 (2)
Printer Driver	0 (3)	768
Total Bytes	23,756	2,992
Execution Times		
Each Command	2.0 sec.	0.1 sec.
20 Commands (4)	44.0 sec.	5.4 sec.

Notes:
(1) Assembly-language program need not be present while application program is running.
(2) Variable storage locations included in main program.
(3) Printer driver included in BASIC interpreter.
(4) Includes operator reaction times between keystrokes.

Table 3.1 BASIC and Assembly Language Comparison[21]

Both programs perform the identical function. The memory usage and execution times for the BASIC program are about eight times greater than those of the assembly language version. In addition, BASIC does not support modular programming and most implementations have little or no ability to interface to custom assembly language routines such as special I/O drivers. The main advantage of BASIC is that it is easy to learn and thus very widely used.

High level languages that are compiler based work in the following manner. First, the text is written into a file.

Next, this file is processed by a compiler program which converts the high level program instructions into machine code. Lastly, the machine code is executed. Compilers usually generate two to four times as much machine code as an equivalent program written in assembly language[20]. That is to say, compiled programs are less efficient in memory space and execution speed than programs well written at the assembly level. In some cases this extra code may be a problem because of the additional execution time. Compilers such as FORTRAN allow modules programmed in assembly language to be linked into the main program; thus time-critical routines can be coded in assembly language when necessary. Unfortunately the compiling process requires a great deal of memory and mass storage, and thus requires the support of an operating system. Special loader and linker programs are needed to combine high level programs and assembly programs.

The major language systems available today were originally designed to run on large computer systems and solve numeric processing or data management problems. These traditional programming tools can be applied to control systems, but it is somewhat like trying to use a pickaxe to drive a nail, its too big and not quite the right tool for the job. FORTH does not have this problem.

The FORTH Language System

This section will highlight the major features of the FORTH language system as well as briefly discuss some of it's operating principles. Development of the FORTH lang-

uage started in 1968[22]. The original goal was to create a language that would make programming control applications faster and more efficient. The first implementation of FORTH was on a Honeywell H136 computer system that was to control a radio telescope at the National Radio Astronomy Observatory (NRAO)[22]. This makes FORTH one of the few languages that was developed for small computer systems with control applications in mind.

In its maximum configuration, the FORTH language system incorporates a high level language, editor, assembler, and operating system functions into one integrated package. It is a stand-alone system which requires no other software packages or additional operating systems to function. All functions of the editor, assembler, operating system and high level language are available to the user at all times; there is no need to invoke a separate editor to edit text. An edit command can be issued at any time, by the user from a terminal or even by a program while it is running; FORTH makes no distinction between the two. Most systems that are not integrated are more difficult to learn since each function has its own rules and syntax. For example in the RSX11M operating system used by DEC PDP-11 computers, commands are terminated with a carriage return, yet the text editor TECO which is often used with this operating system terminates its commands with an escape. These kinds of small differences make learning to use new functions more difficult and confusing. The FORTH language system does not suffer these problems since all the functions of the system

are integrated into one package. Once the user has mastered the use of one function, it is easy to learn how to use the others since the structure and syntax are the same.

One of the unique features of FORTH is the use of a push-down stack to pass parameters between routines. A push-down stack operates as a Last-In-First-Out (LIFO) buffer; items placed on the stack are removed in reverse order. Figure 3.1 shows the results of placing the numbers 2, 4, 6, and 8 on the stack, the first item to be removed from the stack would be an 8.

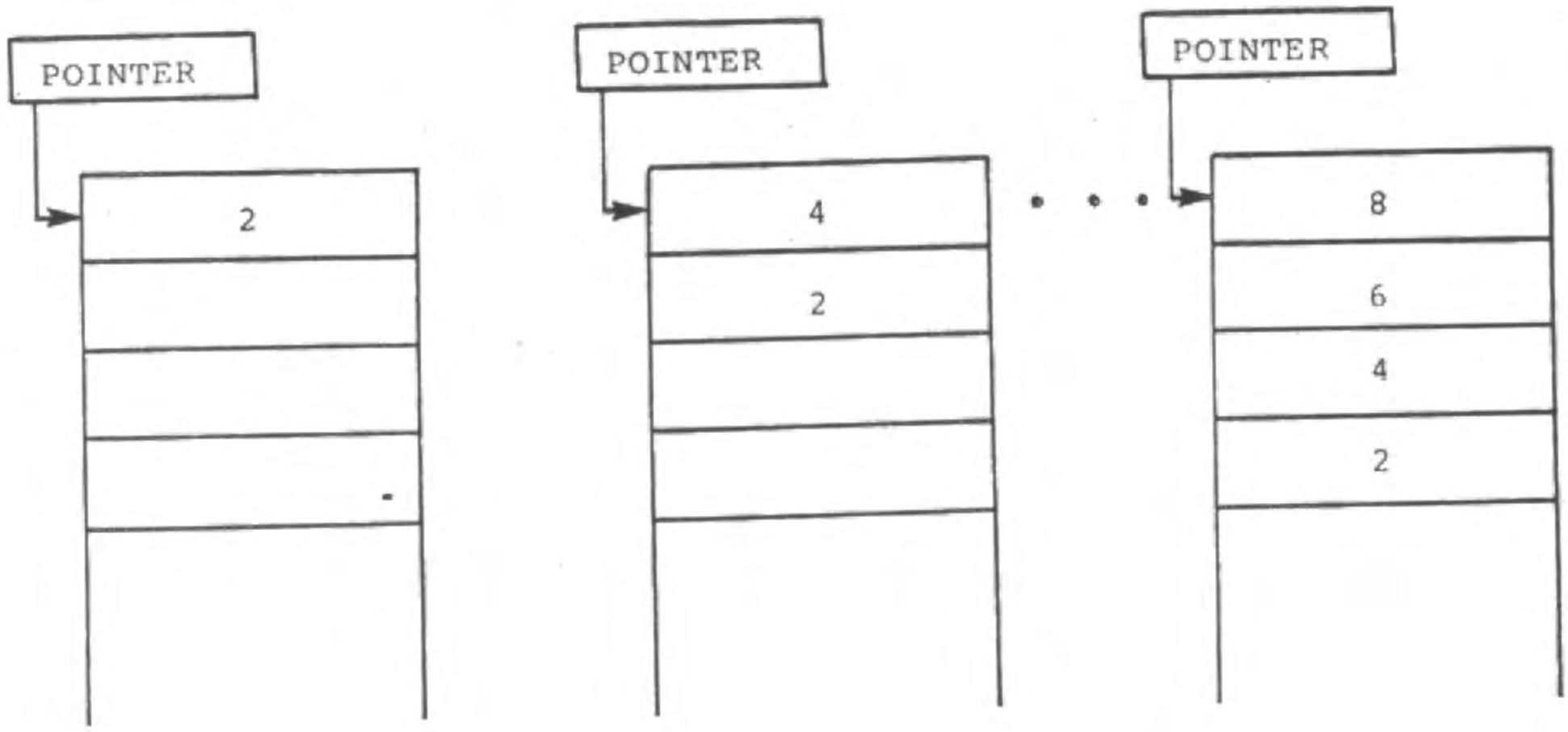


Figure 3.1 Stack Example[23]

FORTH uses the stack to pass parameters between both high level routines and assembly language routines. Thus the interface to an assembly language routine is no different than to a high level FORTH routine.

One of the more dramatic differences between FORTH and traditional languages is its use of post-fix notation rather

than the more familiar algebraic, or in-fix notation. The reverse Polish notation (RPN), used on most Hewlett Packard calculators, is post-fix notation where the operators come after the arguments. FORTH uses the stack for all numeric operations; for instance, addition is performed by removing the top two items on the stack, adding them together, and placing the result on the top of the stack.

Programs written in high level FORTH run very quickly. Typically high level FORTH routines run about 20% slower on 16-bit minicomputers and 75-100% slower on 8-bit microprocessors than the equivalent routines written in assembly language[24]. This is considerably faster than most high level languages which typically run 200-1000% slower than assembly language. If speed is critical, a routine can easily be rewritten in assembly language, and since parameters are passed between routines by way of the stack there is no need to change any other routines that may use the routine that has been converted from high level FORTH to assembly language.

In the FORTH language system, programs are written by creating new program modules called words. A FORTH word is analogous to a subroutine in more conventional languages. FORTH stores words in a linked list called a dictionary. The heart of the FORTH language system is this dictionary, which, like a real dictionary, contains words that are defined by lists of other words. Each word performs some function. The real power behind the FORTH language system is the ability of the user to define new words in the

dictionary. New words are defined by lists of existing FORTH words. The basic FORTH system contains several hundred words. The definition of a new word begins with a colon (:) and ends with a semicolon (;). This ability to add new functions to FORTH makes it an extensible language; it grows on itself.

In FORTH programming, words build on each other, each new word becoming higher and higher level. Eventually a high level language is developed that is specific to a system making programming for that system very efficient. As an example of how this works, examine the FORTH program listing in figure 3.2.

```
0 ( STEPPER MOTOR ROUTINES )
1
2 : STEP  10 @ DROP ;
3
4 : STEPS  0 DO  STEP  LOOP ;
5
6 : DEGREES  4 *  STEPS ;
```

Figure 3.2 FORTH Program Example

These routines were written to drive a stepper motor that advances 0.25 degrees each time it is pulsed. Line 0 is just a comment line indicating that these routines are for the stepper motor. Line 2 defines a new word called STEP, which first places 10 on top of the stack. The hardware interface is designed to pulse the stepper motor 0.25 degrees each time memory location 10 is accessed. The "@" is the FORTH word 'fetch' which takes a number off the top of the stack and replaces it with the contents of that memory location. In this case, the contents of memory location 10

will replace 10 on the top of the stack. This causes the stepper motor to advance 0.25 degrees. Since we have no interest in what is at location 10, the next FORTH word is DROP which discards the number on top of the stack. This is followed by a semicolon which ends the definition of STEP. The new word STEP can be tested from the terminal as soon as it is entered, by typing the word STEP followed by a return and checking the stepper motor to see if it advanced. The word STEPS defined on line 4 pulses the motor n times where n is the number on the top of the stack. This is done by using the new word STEP and the FORTH words DO and LOOP which create a loop which goes from 0 to n thus repeating STEP n times. The final word DEGREES defined on line 6 advances the stepper motor a given number of degrees. It takes the number on top of the stack as the number of degrees to move and multiplies this by four leaving the result on top of the stack. This provides the number of 0.25 degree steps desired which STEPS then uses to advance the stepper motor.

FORTH is somewhat of a cross between a compiled and an interpretive language. When a new word is entered into the dictionary, the list of words that define the new word is compiled into a list of addresses. Each word in the dictionary has an address, this address is entered into the list when a new word is compiled, so the time consuming searching of the dictionary during compilation is done only once. When the new word is executed, FORTH's inner interpreter picks up these addresses and executes the desired

code. The text of a word is not stored, as is common with interpretive languages.

The FORTH language system is very compact; in a maximum configuration, including the editor, assembler, floppy disk and terminal drivers, a FORTH system occupies less than 8K bytes of memory on a microprocessor. This is quite small when compared to many versions of BASIC (with a disk operating system) which, at the barest minimum, occupy about 12K bytes, and are often as large as 24K to 32K bytes. In a dedicated application where the editor, assembler, disk and terminal drivers are not needed, the basic FORTH system can be under 700 bytes for microprocessor systems. In addition to being small itself, the FORTH language system generates extremely compact code. Figure 3.3 compares the memory requirements of applications programs in several different languages, not including the memory requirements of the operating system.

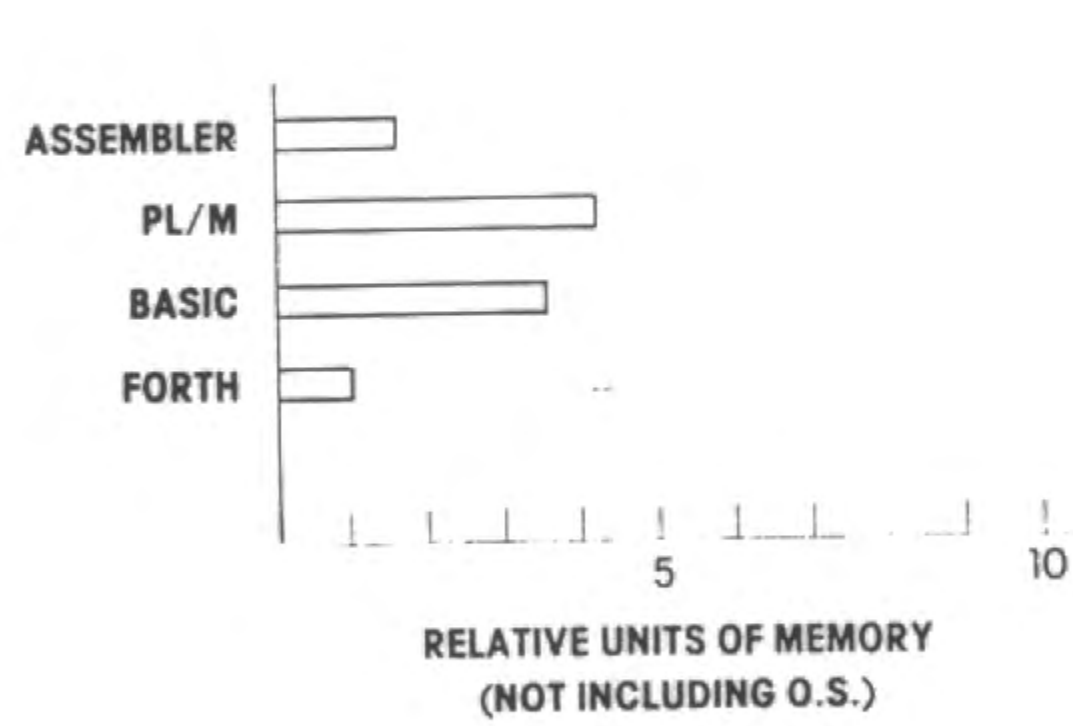


Figure 3.3 MEMORY REQUIREMENTS BY LANGUAGE[25]

PL/M is a high level language compiler. Figure 3.4 is a plot of the memory requirements of FORTH and assembly language programs versus the complexity of the application programs written on minicomputer systems.

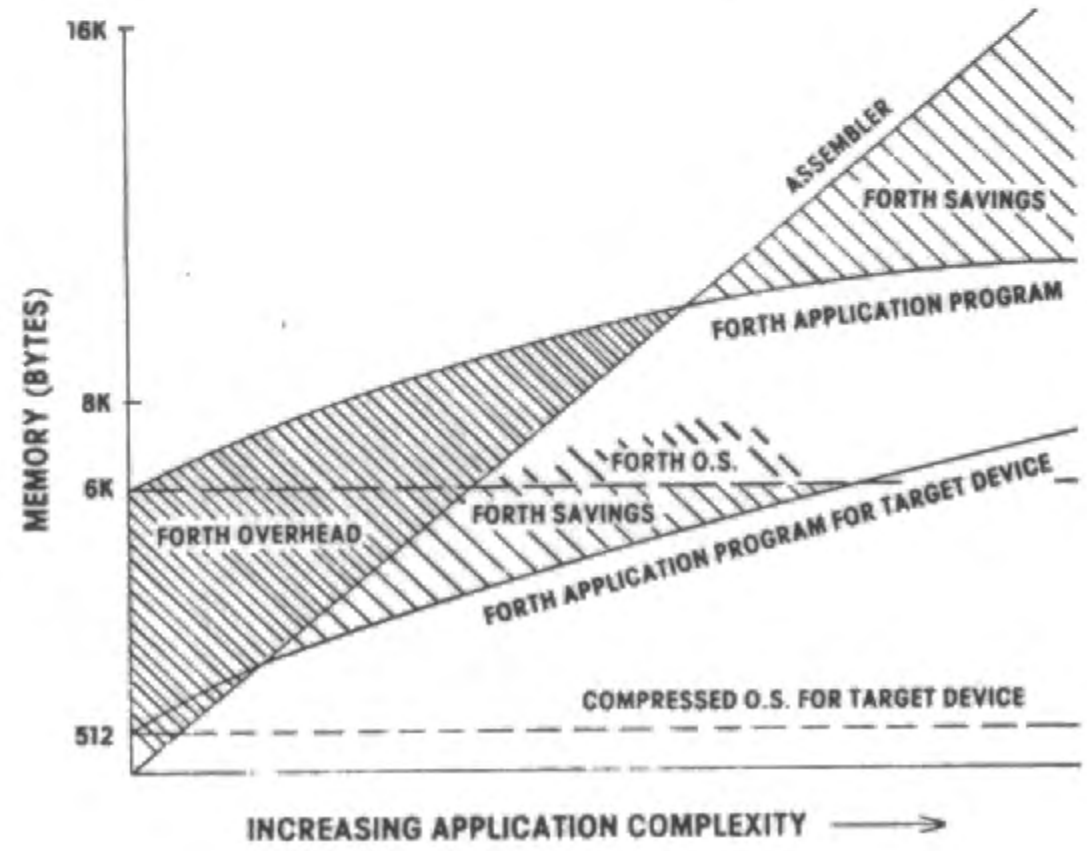


Figure 3.4 MEMORY REQUIREMNETS BY APPLICATION COMPLEXITY FOR ASSEMBLER AND FORTH[25]

There is one line of constant slope representing the memory requirements of assembly language programming. Two curves have been drawn to represent the memory needs of equivalent FORTH programming. The upper curve starting at 6K bytes represents the memory requirements of FORTH in its maximum configuration including the editor and assembler; the lower curve, starting at 512 bytes, represents FORTH in its minimum configuration. In both cases the initial FORTH memory requirements are greater than assembly language needs, but as complexity of the application increases use of

FORTH results in a savings of memory space.

FORTH accomplishes this memory savings in two ways. One is that FORTH compiles each word reference into a two-byte address, whereas in assembly language, a subroutine call requires three bytes on most microprocessors and four bytes on minicomputers. Thus FORTH achieves a 33-50% savings over assembly language when calling another routine. The second memory saving characteristic of FORTH derives from it's extreme modularity. This may be illustrated with the following example. If a constant needs to be added to an 8-bit number, it would probably take at least three bytes of assembly language code to perform this operation. If it is necessary to perform this operation twentyfive different places in a program, this would take seventy-five bytes. Writing this operation as a subroutine would not generate any savings since the subroutine call itself would take three bytes. If a new FORTH word was written to perform the operation, it would take up about sixteen bytes. However, each time it is reference only two bytes are needed. For twenty-five references this would be fifty bytes plus a sixteen-byte overhead gives a memory usage of sixty-six bytes, which is a twelve percent saving over the assembly language version.

Software Development Systems

The role of development systems in the generation of software for control applications is sometimes overlooked, especially in scientific laboratories where writing software

is not a specialty. A development system is a computer system that is used to generate software to run on a control system computer. The development system acts as a programmer's toolbox and work bench. It often includes features which may not be needed or available on the control system computer, such as sophisticated editors, a terminal, a printer, lots of memory, a PROM programmer and mass storage. Since the computer hardware associated with a control system is often non-standard, it is necessary to have a development system on which commercially available software can be modified for eventual implementation in the control system computer. Two disk drives are needed on a development system so that disks can be copied to provide backup copies of software.

In our laboratory, a single board computer featuring a Z80 CPU, 64K bytes of RAM, two serial I/O ports, two 8-bit parallel I/O ports and a floppy disk interface was selected for use as a software development system[26]. This board was connected to two 8-inch floppy disk drives, a video terminal and a PROM programmer. The Z80 board was chosen because it executes the same instruction set as the 8085 processors used in our microcomputer systems and because it is capable of running the CP/M operating system which has become a de facto standard for microcomputer systems. This allowed us to purchase a version of the sophisticated polyFORTH system which was designed to run under the CP/M operating system[27]. One of the features of polyFORTH which sets it apart from

other implementations of FORTH is its target compiler. The target compiler feature allows the user to recompile the basic FORTH system with any additions or changes the user wishes to make. The development system was used to generate a new version of polyFORTH to run on our micro systems. Thus the development system acts as an interface between our non-standard hardware and the products available in the commercial market. In addition to the polyFORTH system, a FORTH screen editor was purchased for use with the development system to aid in program generation. Several new features were added to the screen editor to enhance it's performance. Routines were written in FORTH to drive a PROM programmer, so that when a new FORTH system is generated it can be permanently stored in read-only memory.

CHAPTER 4

SOFTWARE FUNCTIONS DEVELOPED

This chapter will highlight some of the special software features developed for controlling the TQMS instrument in our laboratory. Internal operations of the software will be discussed in some of the more interesting cases.

Design Goals

Prior to the start of this project the instrument functions under microcomputer control were quadrupole scanning, data acquisition and a limited set of graphic display routines. The software was primarily written in assembly language and was stored in EPROM. There was no provision for local programming on the microprocessor. The combination of these factors led to a very rigid inflexible system that was unable to adapt to the changing needs of the research environment. As a result there was a great deal of user frustration with the system, to the extend that an X-Y recorder and manual controls were used for data acquisition because the microprocessor control system could not adapt to the needs of the users. This situation gave rise to the following design goals.

1) Control of Quadrupole Scanning

2) Control of Ion Path Elements

3) Make the Instrument User Friendly

4) Provide a Local Programming Environment

5) Create a Flexible/Adaptable system

6) Create User Documentation

Vectored Execution

Vectored execution is a powerful programming technique that was used extensively in the creation of the operating system software. Its use adds greatly to the flexibility of the system since it allows new functions to be added with little or no changes to existing software. Vectored execution works in the following manner. At a specified memory location a vector (pointer) to a segment of program code is stored. When a function that needs to use that segment of code is executed, this vector is fetched from the memory location and control is transferred to the designated address. This allows different segments of program to be executed by the same function by simply changing the address stored in one location. Figure 4.1 illustrates the use of vectored execution in the word ACQUIRE. The location AMODE contains the address (vector) of either the autoranging routine (as in figure 4.1) or the fixed gain acquisition mode. The word ACQUIRE operates by fetching the contents of the location AMODE and then proceeding to execute the code pointed to by that vector. Thus either mode of acquisition can be used by simply storing the appropriate address in AMODE.

The Device Control Table

All of the various devices discussed in chapter two

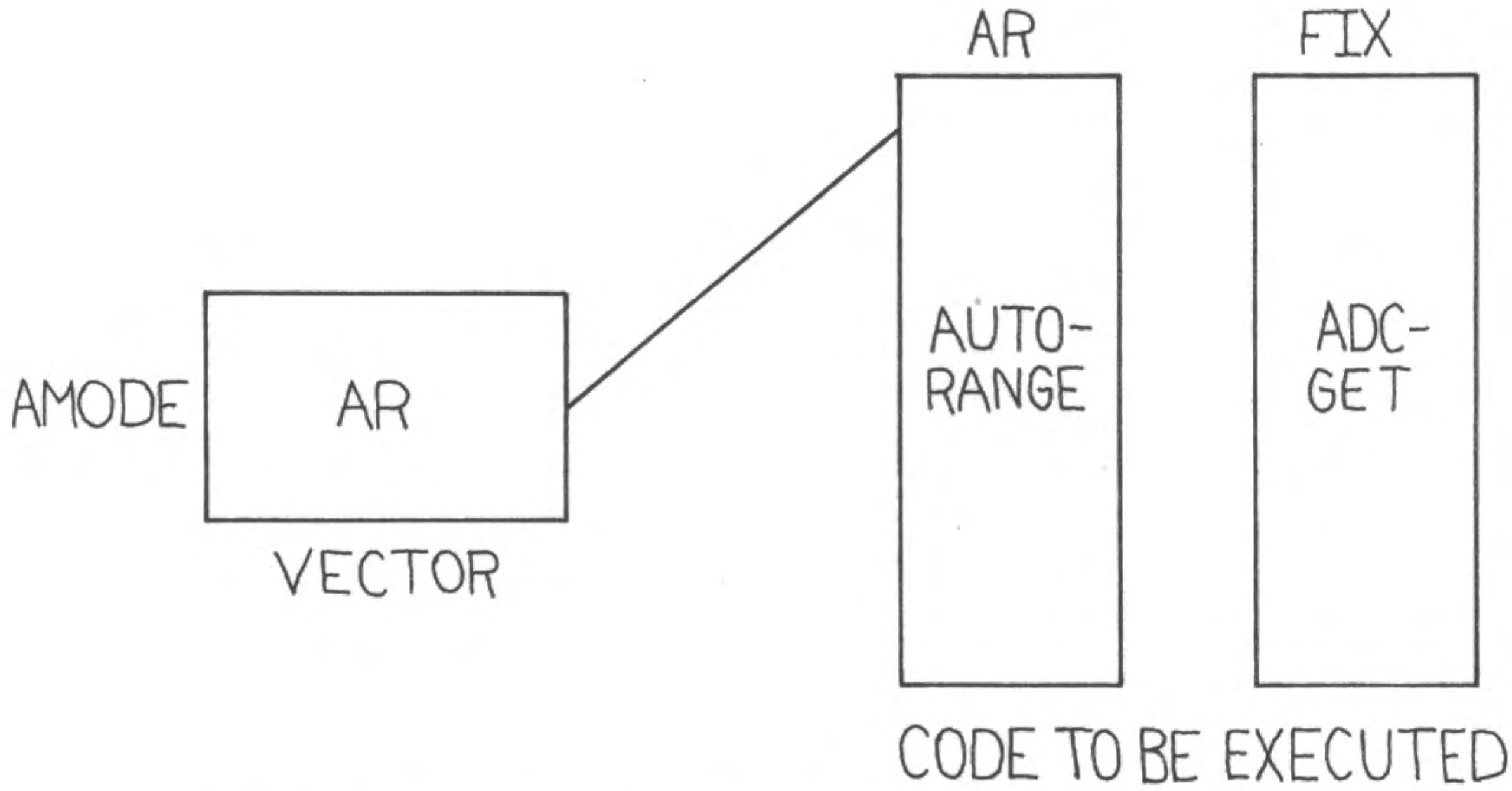


Figure 4.1 Vectored Execution of Acquire

that are interfaced to the control system computer are controlled through a device control table (DCT). The current implementation of this DCT allows for control of up to thirty-two devices. The DCT contains four user-definable entries for each device: 1) a current value where the device is normally set, 2) and 3) start and end values which specify the range over which the device is to be scanned, and 4) a step value which controls the increment of advance when the device is scanned. Four additional entries in the DCT under control of the programmer are as follows: 5) the name of the device, 6) the address of the routine to be executed to send data to the device, 7) the address of the routine that sets up the appropriate numeric formats for input and display values of the device and the appropriate conversion factors between units such as converting volts into a DAC value, and 8) a general purpose entry called flags. The flags entry, currently only used for devices that are part of the ion controller contains the number (0-15) of the ion controller DAC that is being used to control the device in question. Figure 4.2 illustrates the organization of the DCT entries for each device in the system.

All of the interaction between the user and the devices interfaced to the instrument is routed through the DCT. The user need not be concerned with the different characteristics of each device since by use of the DCT all devices appear to behave the same. Figure 4.3 diagrams the interaction between the user, the DCT, and the various devices in the system. The user only interacts with the DCT,

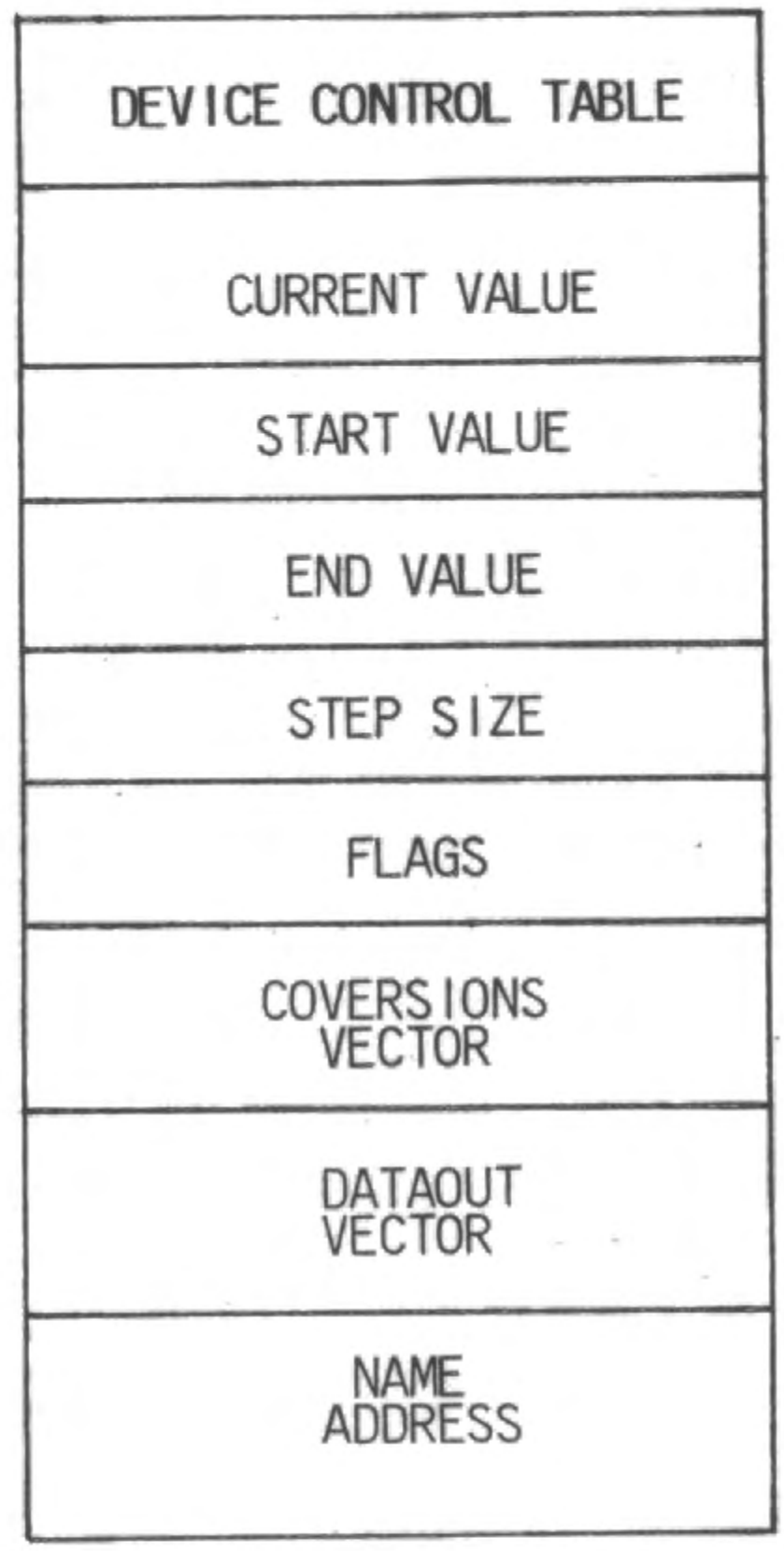


Figure 4.2 Device Control Table Organization

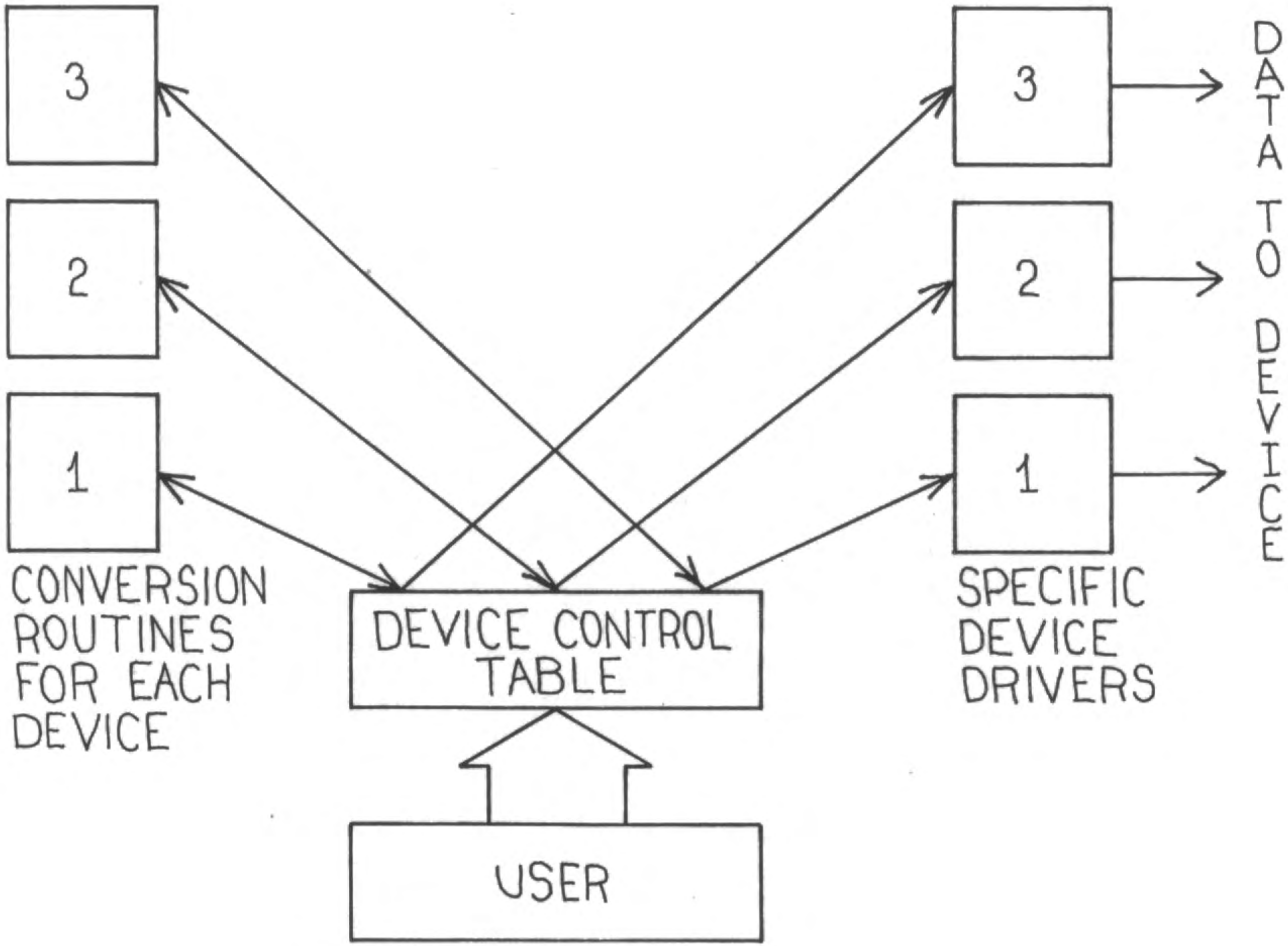


Figure 4.3 DCT Interaction

never directly with a device. The routines pointed to by the DCT vectors perform all the necessary numeric formatting, scaling, and communication protocols for the various devices.

Consider the case of one of the ion lenses controlled by the ion controller as an example. The conversion routine accepts a voltage value on top of the stack and returns a value that will cause the DAC to output the desired voltage. A conversion routine is also provided to perform the reverse operation, accepting a DAC value and returning the corresponding voltage value. This two way communication is symbolized by double ended arrows in figure 4.3. A different set of conversion routine must be developed for each type of device in the system.

To output data to a device the DATAOUT vector is used to point to the device driver specific to the desired device. Communication is only one-way, with data being transferred from the DCT to the device driver and passed on the the device.

To add a new device to the system, only four short routines need to be written. One is the device driver, which is often little more than storing the number on top of the stack in some specific location. A routine that formats the DCT entries for the STAT display (putting decimal points in the right places, etc.) is needed. Finally two conversion routines need to be developed, one to convert values expressed in familiar units (volts, mass, etc.) into DAC values and a second to perform the reverse operation.

The reasons for selecting this table-driven approach are twofold. First and most importantly, it simplifies the operation of the instrument. All of the different types of devices in the system are controlled in the same way. The user only has to learn one set of commands that modify the DCT. This is simpler than having to learn one set of commands to modify the ion lenses, another to control the mass settings of the quadrupoles, and so on. Secondly, our instrument is a research instrument which is undergoing continuous evolution, with new types of devices being added and old ones removed or modified. To add control of a new device to the system, a programmer need only add new entries into the DCT and write software to drive the new device. There is no need to write new commands or modify existing software to add additional devices.

MASS AXIS CALIBRATION

The masses selected by quadrupoles one and three are controlled by two 16-bit digital-to-analog converters (DACs)[28]. The digital values ranging from 0 to 65535 sent to the DACs must somehow be related to the mass value selected by the quadrupole. There are several approaches to this calibration problem, one which puts the burden on the hardware and others that put the burden on the software.

In the first case some arbitrary relationship between the DAC values and the mass values is selected. For example, suppose a change in DAC value of 100 is to correspond to a one amu change in mass. The gains of both the DACs and the

quadrupole controllers are then adjusted to achieve this 100 to 1 relationship over the desired mass range. This is a very tedious operation which can take many hours to perform since there is a great deal of interaction between all the gain and offset controls. The 16-bit DACs as well as the quadrupole controllers are subject to long-term drifts which can cause errors in mass assignments if not corrected regularly. The quadrupole itself is part of a tuned circuit and anytime it or it's connecting wires are moved during cleaning, the impedance of the tuned circuit is changed, and a shift in the mass axis occurs. These drifts and shifts would require that this complex and time-consuming calibration be done much more frequently than anyone would want to. Another drawback to this method is that is assumes that the response of both the DACs and the quadrupole controllers are absolutely linear. In contrast to the demands made on the operator and the system, this method of mass calibration makes programming very easy. To set a quadrupole to a desired mass value, the mass need only be multiplied by a constant (in this case 100) to find the appropriate DAC value.

The mass scan on a quadrupole is theoretically linear with respect to the applied scan voltage. Thus it would be possible to develop a software calibration scheme that would determine the slope and intercept of a line relating DAC voltage to the mass selected. This approach was not taken in the initial implementation of the control system software because of several possible drawbacks. Small errors in

determining the slope could lead to mass assignment errors, though this problem could be eliminated with enough careful programming. Also there would be no correction for any nonlinearities present in the system. The time needed to perform the necessary calculations for each point while scanning would be too great without the addition of specialized hardware. At this time, the control system computer does not have the capability of readily performing the needed least squares fit to allow the computer system to accurately calibrate itself. Perhaps when greater numeric processing power (by the addition of a numeric processor, such as INTEL's 8321) or extra programming time is available, the practicality of this approach can be explored.

The use of lookup tables is impractical in this application due the memory requirements of such an approach. For example, assuming that the smallest desired mass increment was 0.1 amu and the largest mass range to be scanned would be 1000 amu, a 10,000 element lookup table would be needed. Since each element would require two bytes, the lookup table for just one quadrupole would be 20,000 bytes and for all three quadrupoles would be 60,000 bytes. This value is totally impractical considering the 8085 microprocessor can only address 65,535 bytes of memory.

The approach that was implemented in the control software is as follows. A sample generating many fragments over a wide mass range is introduced into the mass spectrometer.

Mixtures or compounds such as perfluorokerosene (PFK) or Perfluorotributylamine (PFTBA) are often used. The DAC controlling one of the quadrupoles is then scanned over it's full range (0-65535) and a mass spectrum is obtained. A number of peaks in the sample, evenly distributed over the entire mass range, are selected as mass calibration points. (In the current implementation the number of points is limited to sixteen.) The known mass values of these peaks and the DAC values at which the peaks occur are entered into a calibration table. To select a given mass, the software uses this table to interpolate the desired DAC value. The interpolation routine was carefully coded to make sure no rounding errors were introduced by the use of integer math. Although this procedure is currently being performed manually, it will be automated in the future. After the calibration compound is entered into the inlet system a calibration program will be executed that will set up the calibration tables for the quadrupoles in just a few minutes.

Speed is a major consideration in implementing mass scanning functions. The desired mass range is scanned by converting the starting mass into a DAC value, then incrementing this value until it reaches the value that corresponds to the ending mass. As peaks are identified during the scan, the DAC value at which they occur is stored. After the scan is completed these DAC values are converted into mass values with the aid of the calibration tables.

Neutral Loss Scanning Algorithm

The ability to scan over a mass range selecting for fragments that are the result of a certain neutral loss provides a powerful technique for screening mixtures. A neutral loss scan is performed by scanning quadrupoles one and three at the same time but at a a constant difference in mass. This is a somewhat difficult function to implement because the responses of the two quadrupoles are not identical. Each quadrupole has it's own mass axis calibration, which can ideally be described by a slope and offset. One method of performing a neutral loss scan is to scan quadrupole one over the mass range of interest, while quadrupole three is slaved to quadrupole one to allow it to follow by some mass difference.

Several laboratories have accomplished this slaving of one quadrupole's scan to another by the use of some additional hardware[29]. The signal that drives the master quadrupole (quad one) is fed into an operational amplifier circuit that has controls for the user to adjust gain and offset. The output of this circuit is then used to drive the slaved quadrupole. The user adjusts the gain of this circuit until a change of one amu on the master causes a change of one amu on the slave. The offset is then adjusted to provide the desired mass difference. This approach has several drawbacks. One is that adjusting the gain of the amplifier circuit to match the two quadrupoles is a tedious operation. The greatest drawback is that manual intervention is needed to adjust the offset for each desired neutral loss.

In the interest of making the system easy to use and to allow for maximum freedom in future automation of mode selection, the neutral loss scanning function was implemented entirely in software. This is consistent with the philosophy that the difficult or tedious tasks be performed by the computer. The implementation of the neutral loss scanning function that was used is the following. The two quadrupoles are initialized to their respective starting values, which is a relatively straight forward operation. Using the calibration tables to determine the appropriate values to send to the DACs, quadrupole one is set to its starting mass value and quadrupole three is set equal to the starting mass of quadrupole one minus (or plus) the desired offset. The difficult part is scanning the two quadrupoles together at a constant rate while allowing the slopes of their calibration curves to be different. The slopes of both calibration curves over the desired mass range can be determined from the calibration tables. These slopes can be used to compute a ratio that relates the setting of quadrupole three's DAC to the value sent to quadrupole one's DAC. It would be possible to use this ratio to calculate a value for quadrupole three each time a new value is sent to quadrupole one. However, even using integer arithmetic, (which is much faster than using floating point routines without the support of special hardware) the necessary calculations could take up to several milliseconds for each point, which would make the scan undesirably slow.

The approach taken was to increment the DAC for quadru-

pole one by an integral quantity. Addition of integers can be accomplished very rapidly with a microprocessor. The DAC's controlling the quadrupoles accept 16-bit data words, requiring the calculations to be done on 16-bit quantities. The 8085 instruction set includes a convenient double add (DAD) command, which performs a 16-bit addition. The ratio of the slopes of the two calibration curves is then used to calculate how much to increment quadrupole three each time quadrupole one is incremented. Unfortunately this result will have a fractional component which is somewhat difficult to work with when performing integer calculations. One solution is to break the current quadrupole three DAC value and its incremental value into two numbers: one represents the integer component and the other represents the fractional component of the number. Thus calculations are extended over more than 16-bits and a binary point is assumed to be between the two values (a binary point is the binary number system equivalent to a decimal point). The accuracy of the fractional component is extremely important since errors can accumulate from the successive additions involved. The obvious choices are between using an 8-bit or a 16-bit fractional component. An 8-bit fraction has a resolution of one part in 256, the smallest fractional part that can be represented is about 0.004. In the worst case an error of this magnitude would cause a two to four amu change in the mass offset during a scan. A 16-bit fraction has a resolution of one part in 65535, the smallest fractional part that can be represented is about 1.5x10-5. In this case

the worst case error would be about an 0.008 amu change in mass offset over the scan, which is a negligible error. Because of this extremely small error, a 16-bit fractional part was implemented in the control software. An excellent discussion of binary integers with fractional parts can be found in reference 30.

Tuning Functions

A great number of factors influence the ion beam in a TQMS instrument. There are many ion lenses, the quadrupole potentials, the resolution of the quadrupoles, and many other factors. All of these interact to make the set up and optimization of the instrument a potentially formidable task. To help simplify the tuning up process, a number of special functions were implemented in the control software.

Chief among these are the fast scanning routines. There are five fast-scanning routines, one for each of the five modes of quadrupole operation. A fast-scan routine does not take any data; it scans a quadrupole rapidly enough that a mass spectrum can be continuously displayed on an oscilloscope screen. This allows the user to readily observe the affects of various adjustments. A special case of the fast scanning operation is a split screen mode, which allows two peaks of greatly different mass to be displayed next to each other on the oscilloscope screen. This feature is particularly helpful in determining if a particular parameter shows any high or low mass discrimination.

Taking advantage of the multitasking capabilities of

polyFORTH the fast scan and split screen routines were implemented as background tasks. This allows the user to enter commands while the computer is displaying the fast scan. The user can change lens voltages from the keyboard and immediately observe the results on the oscilloscope display. A special MANUAL function was implemented that allows the user to increment or decrement the value of any device with a single keystroke. This, combined with the immediate visual feedback provided by a fast scan or split screen display, makes it very easy to manually fine-tune the value of any device.

Data Acquisition Software

The data acquisition software contains all the routines which interact with the hardware that measures the ion current output from the mass spectrometer. Routines in the data acquisition category currently perform the following operations: reading a value from the 12-bit ADC, controlling the range select lines of the variable gain amplifier, and autoranging the amplifier to obtain the best signal to noise ratio for each data point. All other programs or routines that need ion current information are interfaced to the data acquisition routines through the word ACQUIRE. ACQUIRE returns the current value of the ion current.

In the future we expect to be experimenting with different data acquisition hardware as well as testing new algorithms for optimizing data acquisition. Since routines outside the data acquisition group only use the word

ACQUIRE, interfacing to new data acquisition hardware is a relatively simple process. Software is written to drive the new hardware and ACQUIRE is vectored to this new routine, no modifications need be made to any other portions of the control software. This makes it easy to test and implement new data acquisition hardware and software.

Conclusions

All of the original design goals were able to be met. Chapter six contains examples of scanning the quadrupoles and ion path elements cited as goals one and two. The use of the device control table construct has made the instrument much more user friendly in terms of parameter setting and display. The use of the FORTH language system provides a local programming environment. All of the mass spectrometry functions are loaded into RAM from floppy disk. These last two features combined with the extensive use of vectored execution create a highly flexible system that can be readily adapted to new needs. Finally the user's manual in appendix A provides extensive documentation of the use of the system.

CHAPTER 5

TRANSRESISTANCE AMPLIFIER

Use of a current to voltage converter with variable gain can greatly enhance the dynamic range of a data acquisition system. The ability to control the gain of a current to voltage converter by the use of digital signal levels opens new possibilities for intelligent data acquisition. When such a current to voltage converter is interfaced to a laboratory computer system, routines can readily be developed to intelligently autorange the converter to optimize the signal to noise ratio of the data acquired. Such an current to voltage converter is described here. This converter is suitable for use with low current sources such as the photomultiplier tubes and Channeltron electron multipliers often encountered in spectroscopic and mass spectrometry applications.

Circuit Operation

Figure 5.1 is a block diagram of the transresistance amplifier. The amplifier takes a low level current input and generates an output that varies between +/-10 volts. The gain of an amplifier that converts a current into a voltage is referred to as transresistance[31], since in addition to amplifying the signal the amplifier converts it from the current domain into the voltage domain. The transresistance

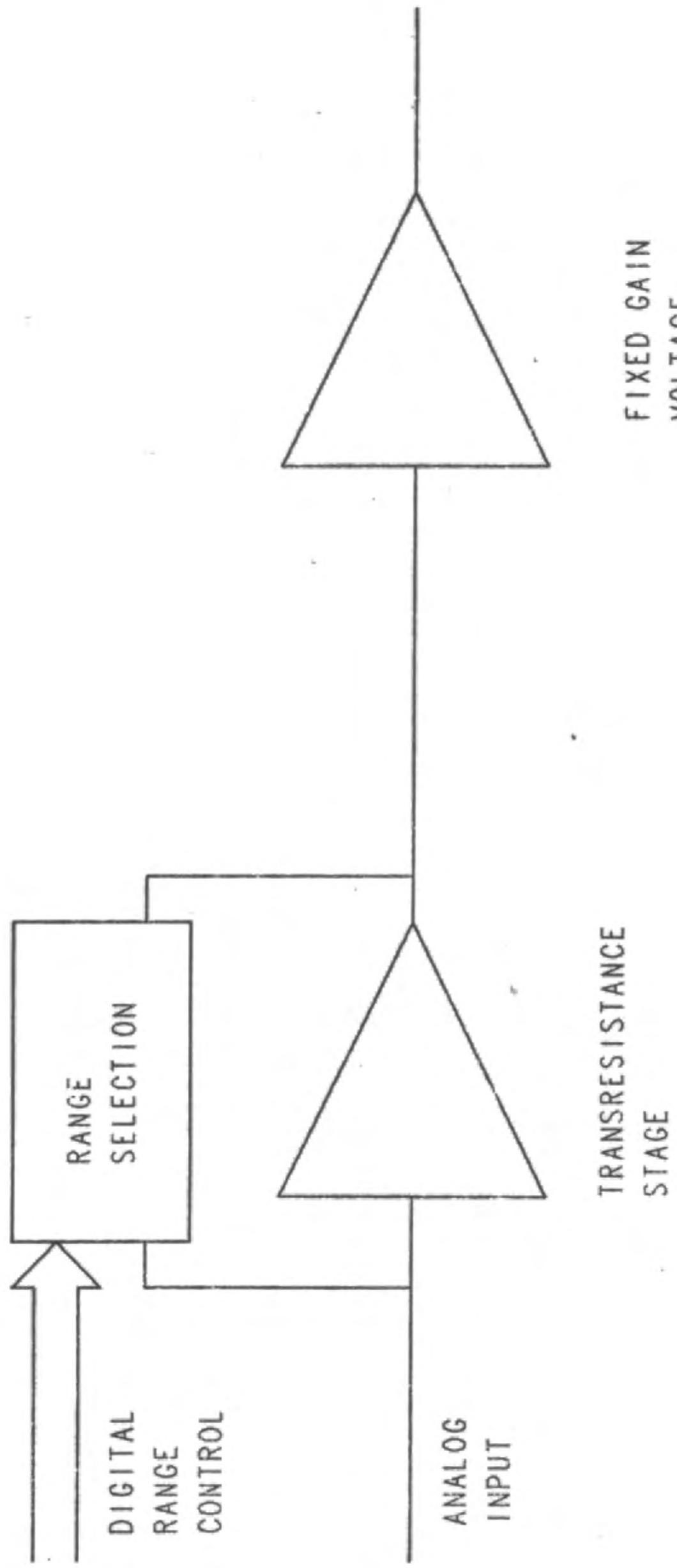


Figure 5.1 Transresistance Amplifier Diagram

amplifier consists of three functional blocks; transresistance stage, range selection for the transresistance stage, and a voltage amplifier with a fixed gain. From here on the amplifier will be treated as a total unit.

The transresistance of the amplifier is controlled by three binary digital range selection inputs. These inputs accept TTL level signals, which can be supplied by a manually operated switch, a parallel port from a computer system, or other types of control electronics. The amplifier has five ranges, each differing from the next by one order of magnitude. Table 5.1 lists the valid range selection codes, the resulting transresistance of the amplifier, and the current input that results in a full scale (-10V) output for each scale.

RANGE	BINARY RANGE CODE	TRANS-RESISTANCE	FULL SCALE
0	000	10^{11}	0.1 pA
1	001	10^{10}	1.0 pA
2	010	10^{9}	10 pA
3	011	10^{8}	0.1 uA
4	100	10^{7}	1.0 uA

Table 5.1 AMPLIFIER TRANSRESISTANCE

If an invalid range is selected (binary values 5 thru 7), the most sensitive range (range 0) is selected.

Circuit Description

Figure 5.2 is a complete schematic diagram of the transresistance amplifier. Operational amplifier U1 and resistors R1 through R6 form the transresistance stage.

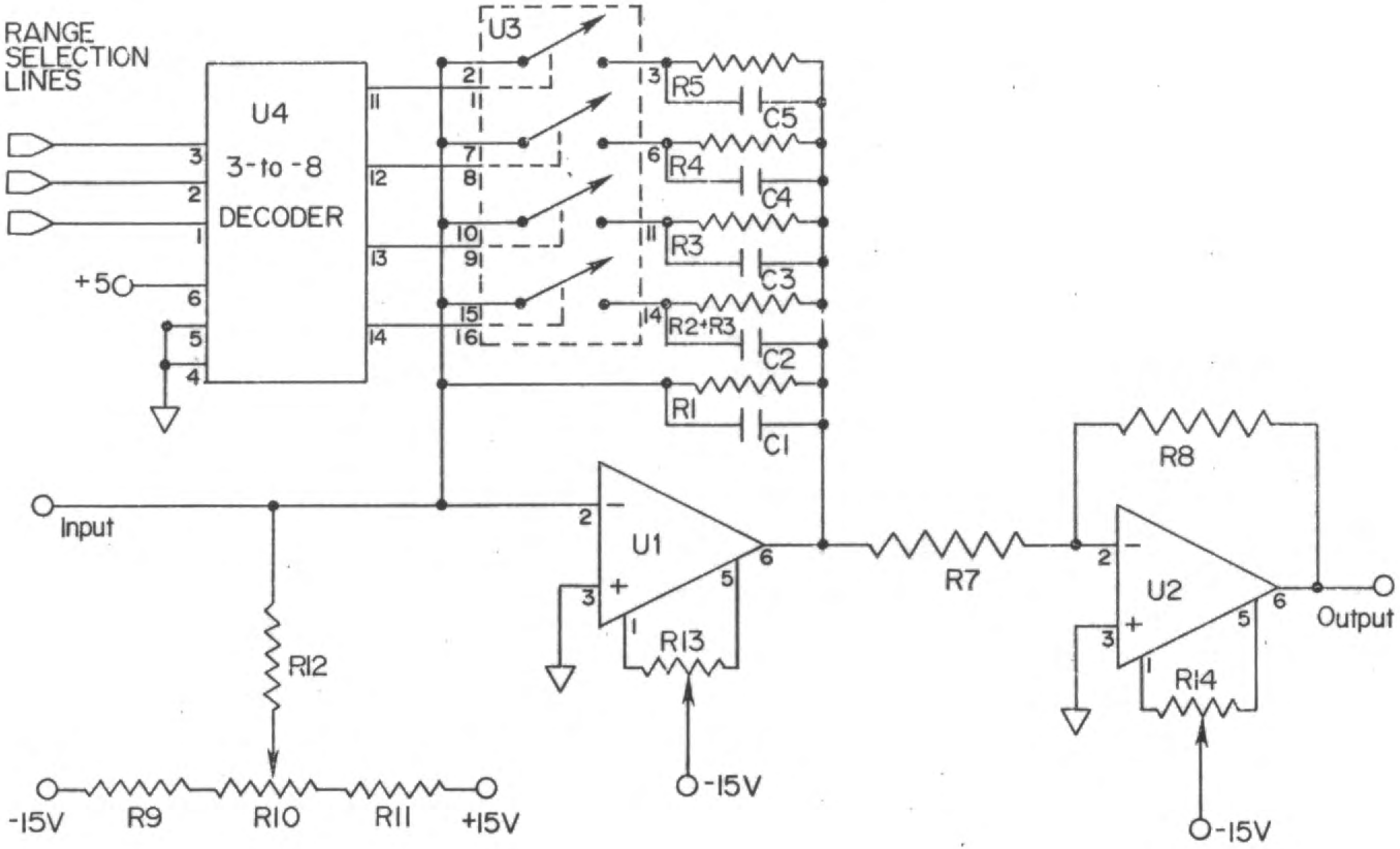


Figure 5.2 Transresistance Amplifier Schematic

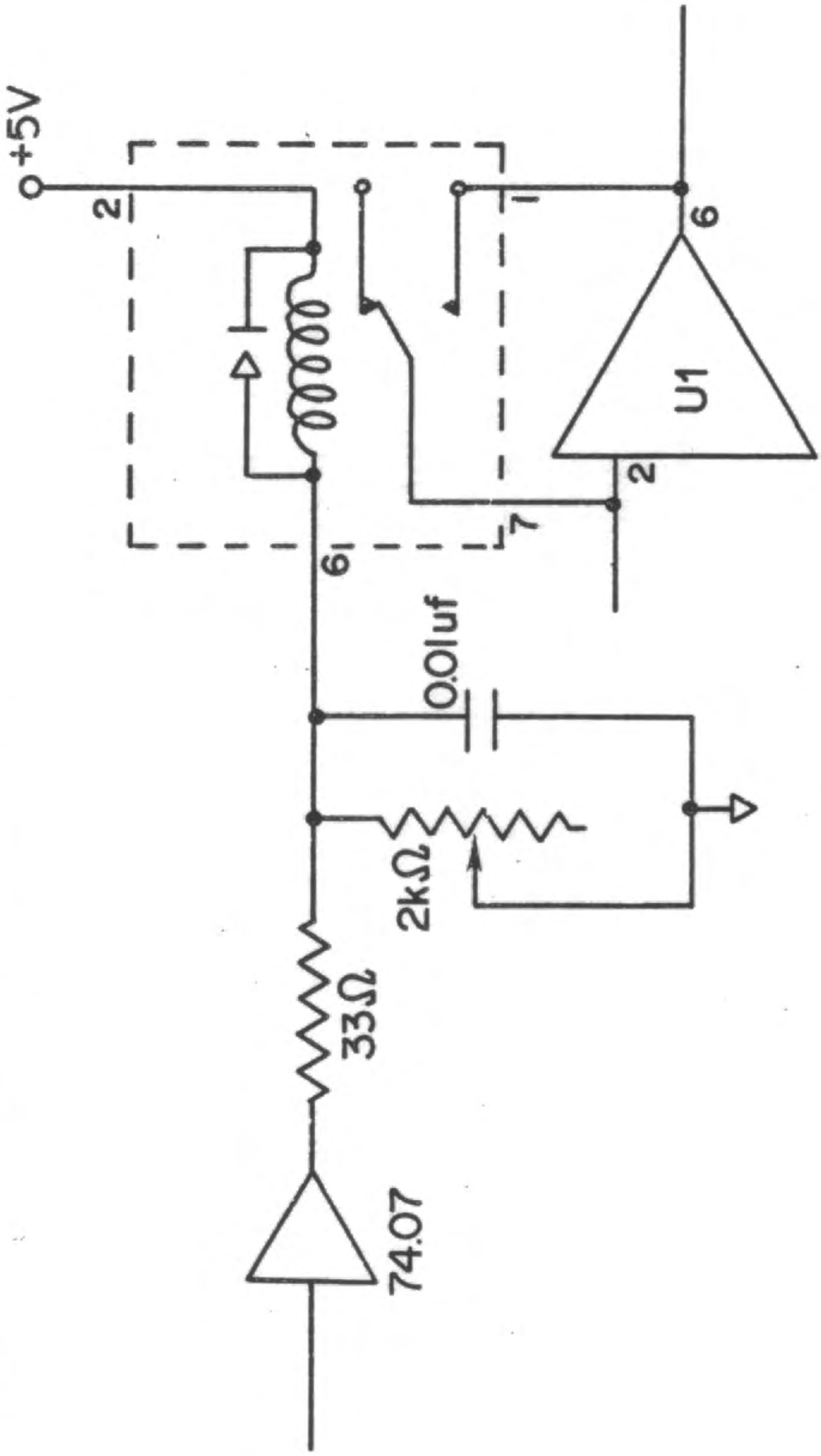


Figure 5.3 Additions for Improved Settling Times

Component List for

Electrometer Amplifier

U1,U2	LF351 Operational Amplifier
U2	DG211 Quad Analog Switch
U4	74138 3-to-8 Decoder
R1,R12	1000M 1% Resistor
R2	100M 1% Resistor
R3,R4	10M 1% Resistor
R5,R6	1M 1% Resistor
R6	100k 1% Resistor
R7	10k 1% Resistor
R9,R11	22k 5% Resistor
R10	2k 10 Turn Trim Pot
R13,R14	10k 10 Turn Trim Pot

TABLE 5.2 COMPONENT LIST

Range selection is accomplished with integrated circuits U3 and U4, while U2, R7 and R8 form the final amplifier stage. Resistors R9 through R12 are used to generate a trimming current to correct any offset current present at the input of U1. R13 and R14 are used to trim the balance of operational amplifiers U1 and U2.

Integrated circuit U4 accepts the three binary range select inputs and decodes them into four separate select outputs that are used to control the four analog switches contained in U3. These switches are used to place resistors R2 through R6 in parallel with R1, in the feedback loop, to control the gain of the transresistance stage. Resistor R1 is always in the feedback loop to prevent the amplifier from saturating if an invalid range is selected and to allow for five levels of gain when only four levels of switching are available. Table 5.3 shows the results on the trans-resistance stage of switching various resistors in parallel with R1.

RESISTOR	FEEDBACK RESISTANCE	TRANSRESISTANCE
R1	1000MΩ	$1.00x10^9$
R2+R3	99MΩ	$0.99x10^8$
R4	9.9MΩ	$0.99x10^7$
R5	1.0MΩ	$1.00x10^6$
R6	100kΩ	$1.00x10^5$

Table5.3 AMPLIFIERFEEDBACK RESISTANCE

The 1% error introduced in two of the ranges due to the parallel resistor R1 is not significant because the

tolerances of the components used are no better than 1%. The final stage configures U2 as an inverting amplifier with a fixed gain of 100.

Capacitors C1 through C5 prevent amplifier U1 from becoming an oscillator. To prevent oscillation, the RC product of the feedback loop must approximately equal the RC product of the input[32]. The output impedance of the signal source forms the R component, and the parasitic capacitance introduce by coaxial cables and other sources forms the C component of the input RC product. The optimum values of capacitors C1 through C5 have to be selected in each application. Parasitic capacitance should be kept to a minimum, since the feedback capacitance adversely effects the bandwidth of the amplifier.

Output settling time after switching between different ranges also is affected by these capacitors. An adverse characteristic of analog switches is charge injection on the switched signal. Whenever an analog switch is closed a small amount of charge (about 10^{-12} coulombs) is injected into the signal being switched[33]. Normally, charge injection is not a major problem, but in this amplifier with it's extreme sensitivity to low level currents it can be quite troublesome. When ranges are changed, the amplifier sees this injected charge as a current large enough to drive it into saturation. The time it takes the amplifier to recover varies between 0.1 and 100 msec depending on the range selected and the size of the capacitor involved. Settling times for each range must be determined experimentally for

each application.

Calibration

Since fixed resistors were used in the feedback loops of the operational amplifiers there is no way of adjusting the gain, only the offsets can be adjusted. To calibrate the amplifier, first ground the input of U1 through a 1000MΩ resistor and select range 0. Adjust R10 until the output of U1 is zero. Adjust R14 until the output of U2 is zero. At this point U2 is properly balanced. Connect the input of U1 to the signal source the amplifier is to monitor. Setup the signal source to give a zero output. It is important to do this portion of the calibration procedure with the amplifier connected as it will routinely be used, since changes in cable routing and length can affect the offset current at the input of the amplifier. Adjust R10 until the output of U1 is zero. Change the amplifier range to range 1. Adjust R13 if necessary to zero U1. Change the range back to 0 and readjust R10 if necessary. Iterate this procedure until a zero reading is obtained on both ranges.

Improving Switch Settling Times

Settling times on all ranges can be reduced to under 2 msec. with addition of the mechanical relay circuit shown in Figure 5.3. Range switching is then accomplished by the following procedure. Close the relay, change the range, open the relay. When the relay is closed it shorts out the feedback loop, preventing the amplifier from going into saturation. The network formed by C1 and R1 is needed to

smooth out the edge of the relay control signal so that it does not induce enough current into the amplifier to cause it to saturate. R1 is adjusted to minimize the effect.

This amplifier has been in use with the triple quadrupole mass spectrometer in our laboratory for the past year. The amplifier's performance has been comparable to the Keithley model 18000 amplifier, which it replaced. The only maintenance needed was an occasional recalibration to correct for long term drift in the components.

CHAPTER 6

PERFORMANCE DEMONSTRATION

Introduction

This chapter will highlight some of the commands of the control system and how they reduce the complexity of instrument operation. Examples will be given of how readily other types of information, such as appearance potentials, can be obtained, instead of just mass spectra. Finally the ability and advantages of generating new commands will be demonstrated. A complete description of all the commands and operations of the control system can be found in appendix A.

Setting Parameters

The current settings of the four user entries in the device control table (current,start,end, and step) for all the devices in the system can be displayed with the STAT command. See figure 6.1 for an example of this display. There are a number of commands that can modify different entries in the DCT, however the operator need only learn one to make full use of the instrument. The PARAM command allows the operator to examine the four user entries for a selected device and modify them if so desired. The PARAM command is executed by first entering the mnemonic name of a device followed by the command PARAM. The values of the four user entries are each displayed in turn, and the operator is

```
QUAD MODE  RF  RF  RF

ACQUISITION MODE AR   RANGE 10 PEAK WIDTH   16

DISPLAY MODE LOG    RANGE 10    # FIELDS 0    AMU/FIELD    0    START MASS    0

 #  NAME  CURRENT   START    END      STEP     #  NAME  CURRENT   START    END      STEP
 0   EV      70.0  -100.0   100.0       4     16   M1     219.3   500.0   630.0       4
 1   RP      45.0  -100.0   100.0       4     17   M3     219.3    40.0   230.0       4
 2   IV      10.0  -100.0     0.0       4     18
 3   L1     -10.0  -100.0   100.0       4     19
 4   L2      10.0  -100.0   100.0       4     20   M2       6.0     0.0   100.0      33
 5   L3       5.0  -100.0   100.0       4     21
 6   Q1     -10.0  -100.0   100.0       4     22
 7   L4      10.0  -100.0   100.0       4     23
 8   L5       0.0  -100.0   100.0       4     24
 9   L6     -40.0  -100.0   100.0       4     25
10   Q2      15.0  -100.0   100.0       4     26
11   L7      10.0  -100.0   100.0       4     27
12   L8       5.0  -100.0   100.0       4     28
13   L9     -30.0  -100.0   100.0       4     29
14   Q3     -40.0  -100.0   100.0       4     30
15   L10     60.0  -100.0   100.0       4     31
```

FIGURE 6.1 STAT DISPLAY

given the option of modifying any of the values. For an example of the use of this command and descriptions of the other commands that affect the parameter settings, see chapter two of the operator's manual in appendix A.

As stated earlier the quadrupoles can be operated in either the RF only mode or DC/RF mode. To allow the operator to set the quadrupoles into a desired mode, a series of commands of the general form $X_1X_2X_3$ were implemented. Where X=R to select the RF only mode and X=D to select the DC/RF mode. For example, the command DRD sets quadrupole one into the DC/RF mode, quadrupole two into the RF only mode, and quadrupole three into the DC/RF mode.

Thus by learning only the PARAM command and the general form of the quadrupole mode selection commands, an operator can set the instrument into any desired configuration. This greatly simplifies the task of setting up an instrument with so many variables.

Taking Data

The control systems allows for the acquisition of two types of data. One is continuous data, where the value of a device is swept over some range and a data point is recorded each time the device is advanced. This mode of operation is referred to as sweeping and is initiated by the SWEEP command. A sweep of a given device is performed in the following manner. Using the PARAM command the start, end, and step values are setup for the desired device. Then the mnemonic name of the device is entered followed by the SWEEP command. Figure 6.2 shows the typical result of a SWEEP

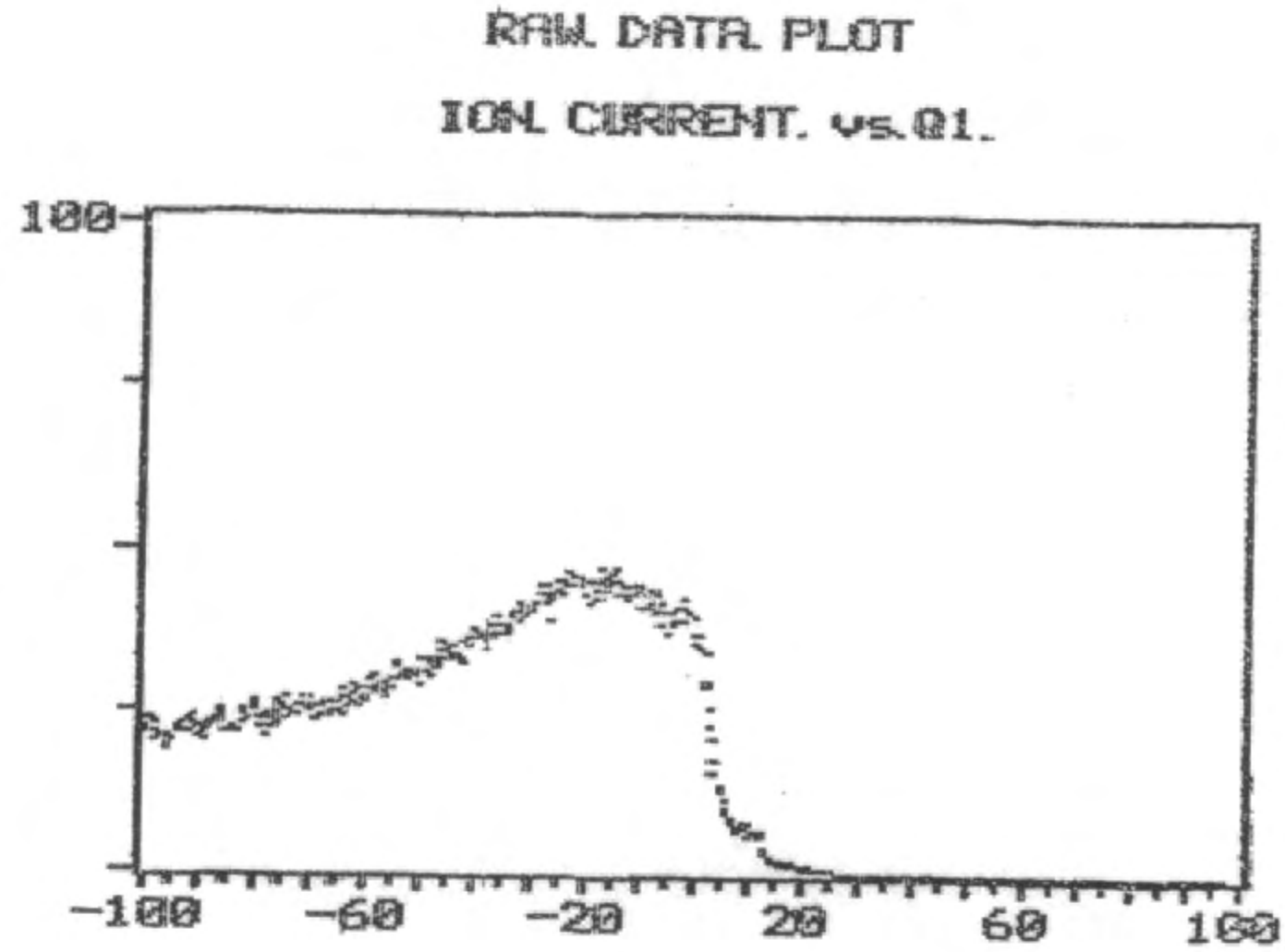


FIGURE 6.2 QUAD 1 SWEEP

operation. For further information on the SWEEP command see chapter four of the operator's manual in appendix A.

The second type of data the control system can acquire is mass spectral data where one or two of the quadrupoles are scanned over a mass range. During the scan, the position of each peak encountered is determined and recorded along with it's intensity. The resulting data is not continuous since data points are recorded only when a peak is located. This type of operation, which generates a mass spectrum, is referred to as a scan.

As mentioned in chapter one, there are five modes in which a TQMS instrument can be operated to obtain a mass spectrum. As a result, there are five scan commands of the form xSCAN, where x=1,3,D,P, and N corresponding to respectively, Quad one scan, Quad three scan, Daughter scan, Parent scan, and Neutral loss scan. Each SCAN command automatically selects the appropriate RF or DC/RF mode for each quadrupole. Figure 6.3 shows the typical result of the DSCAN command. A more in depth discussion of the scanning commands can be found in chapter three of appendix A.

Displaying Data

Either continuous or mass spectral data can be displayed on the graphics screen with the DISP command. The DISP command determines what type of data is trying to be displayed and automatically formats the display accordingly. There are a number of parameters which effect the data display format. However only the selection of the Y axis plotting mode will be discussed here. Details of the display

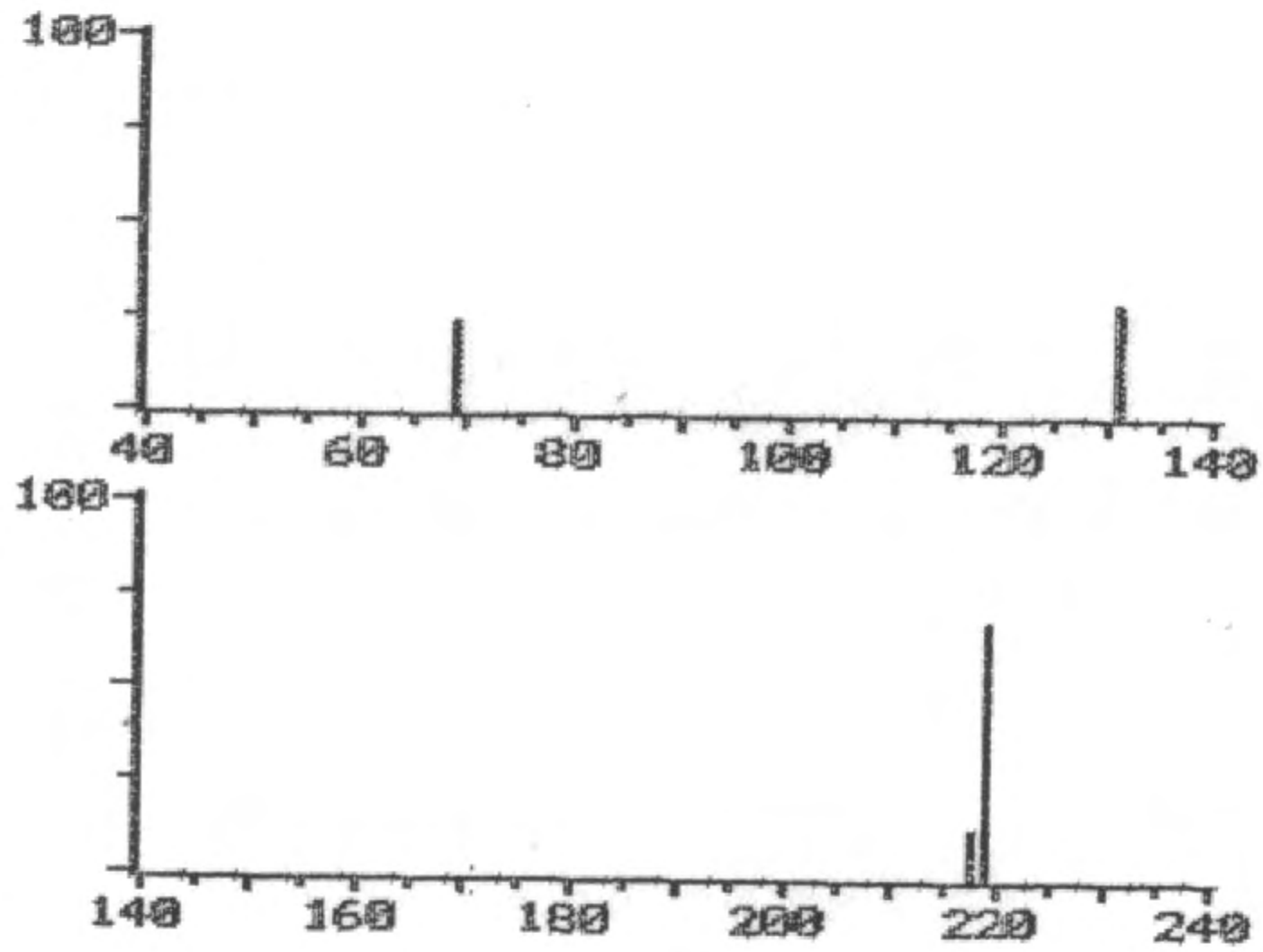


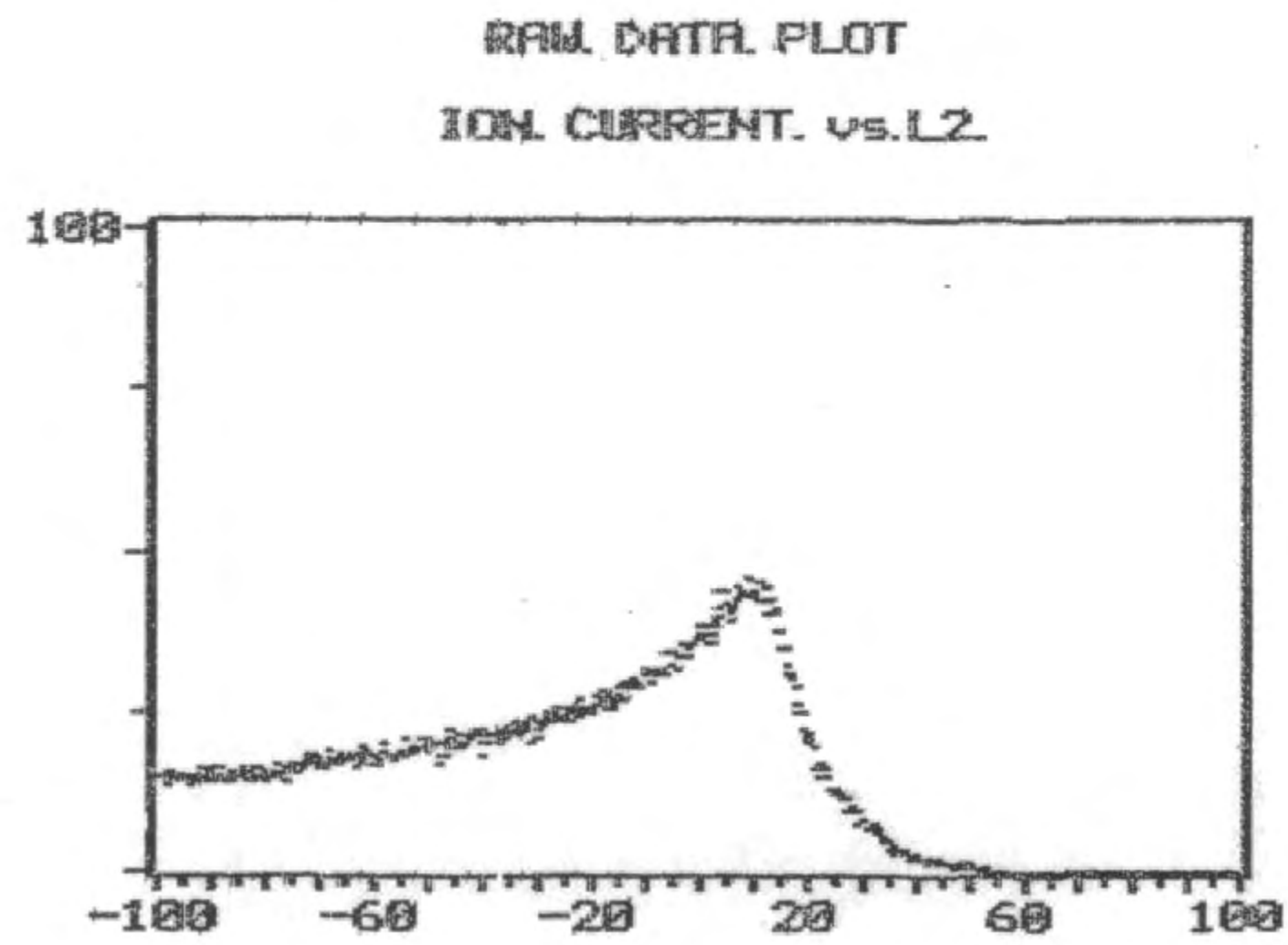
FIGURE 6.3 DAUGHTER SCAN

parameters, how to modify them and their effects can be found in chapter six of the operator's manual in appendix A. The Y axis mode can be selected to be either linear or logarithmic. In the logarithmic mode, the base 10 logarithm of the Y component of each data point is taken prior to plotting the data on the graphics display. This allows data covering several orders of magnitude to be displayed clearly on the limited resolution graphics screen. In the linear mode, the data is plotted without any special processing. Figure 6.4 is a comparison of a logarithmic plot and a linear plot of the same data taken with the SWEEP command.

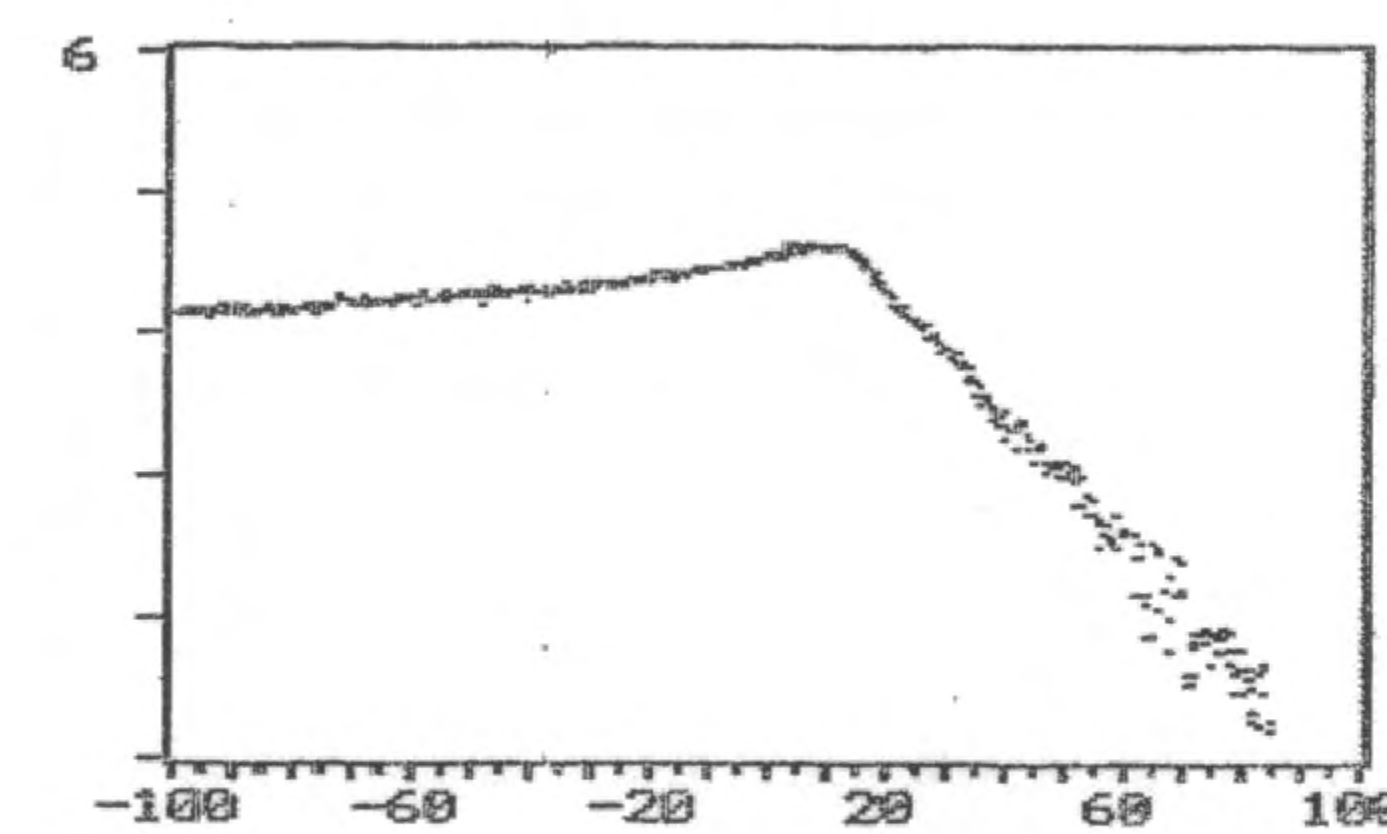
Creating New Commands

One of the most powerful features of this control system is the ability to generate new commands and modes of operation using existing commands without extensive programming. The commands are actually just an extension of the FORTH language, generating a new high level language for programming TQMS operations. This feature is particularly important and helpful on an instrument operating in the research environment. The capability of readily generating software to implement new modes of operation greatly enhances the ability to investigate the capabilites and limitations of the TQMS instrument.

Consider the FORTH program listing given in figure 6.5. Line 2 contains the definition of one of the existing commands, MS1. This command interprets the number on top of the stack as a mass value and sets quadrupole one to select



LINEAR DISPLAY



LOGARITHMIC DISPLAY

FIGURE 6.4 LINEAR/LOG DISPLAY COMPARISON

that mass. The definition of MS1 consists entirely of other

```
0 ( Mass Spec Examples)
1
2 : MS1  M1 SET ;
3
4 : DAUGHTERS  BEGIN  MS1 DSCAN DISP
5    DUP 0= END ;
```

FIGURE 6.5 MASS SPEC EXAMPLES

commands created for the TQMS control system. M1 selects the mass of quadrupole one as the value to be modified, and SET uses the number on top of the stack to change the current value of the device. Notice that the new command is written entirely in high level mass spec commands that have been added to FORTH.

The command DAUGHTERS, whose definition begins on line 4 of figure 6.5, expects a list of one or more masses on the stack. It sets quadrupole one to the first mass on the stack and then performs a daughter scan and displays the result. This procedure is repeated until the list is exhausted. We are already familiar with the three high level mass spec words MS1, DSCAN, and DISP. The other words are basic FORTH words documented in reference 23. The word BEGIN marks the beginning of a loop, while the other three words are used to test for the end of the loop when the list of masses has been processed. This is an example of how high level mass spec words can be combined with more conventional programming constructs to create a powerful new command.

Non-spectral Data

Unlike most other data systems the TQMS control system

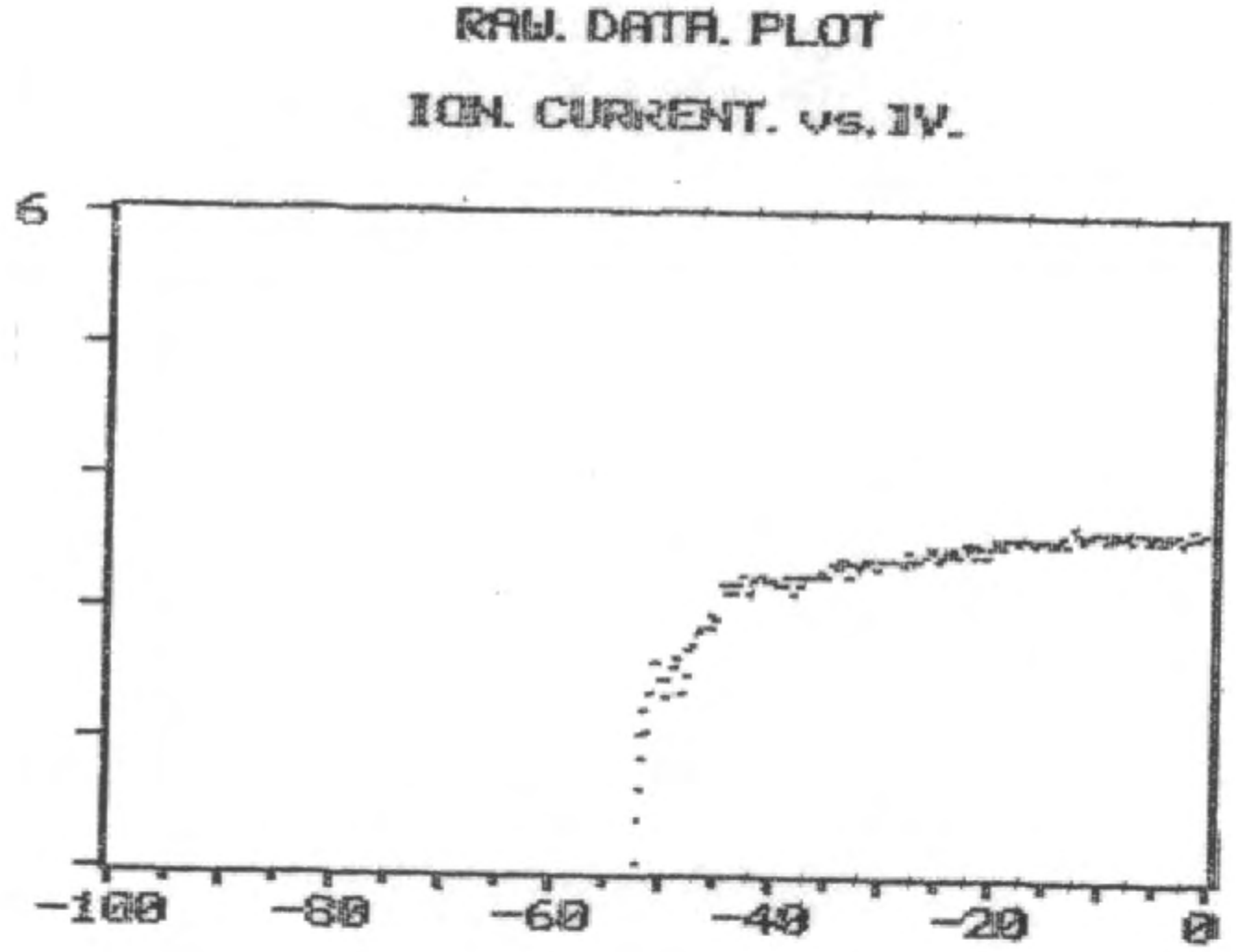


FIGURE 6.6 ION VOLUME SWEEP

can easily obtain non-spectral data from the instrument. One type of information the experimenter may be interested in is the appearance potential of an ion. The appearance potential of an ion is the energy needed to generate the ion from a neutral species. This energy is supplied by an electron beam in the ion source. The energy of the electrons (which generate the ions) is determined by the potential difference between the filament and the ion volume. Figure 6.6 shows the result of sweeping the ion volume potential with the filament set at -70 volts while selecting the 219^+ ion from PFTBA. No ion current is detected until the sharp rise at about -52 volts, indicating the potential at which the formation of 219^+ ions begins. Taking the difference between this turn-on potential and that of the filament, we get the appearance potential of the 219^+ ion from PFTBA as 18 volts.

The fragmentation of ions in the collision cell is dependent on the axial energy of the collision, which is determined by the potential difference between quadrupoles one and two. It often is important to know the axial energy dependence of a given fragmentation reaction. This can be done by selecting the parent ion with quadrupole one and the desired daughter ion with quadrupole three. Then the potential of quadrupole two is swept. The result of such an experiment for the reaction of 84^+ from thiophene going to 45^+ is shown in figure 6.7. To convert the X axis to axial energy the difference must be take with the potential of quadrupole one, which in this case is -10 volts.

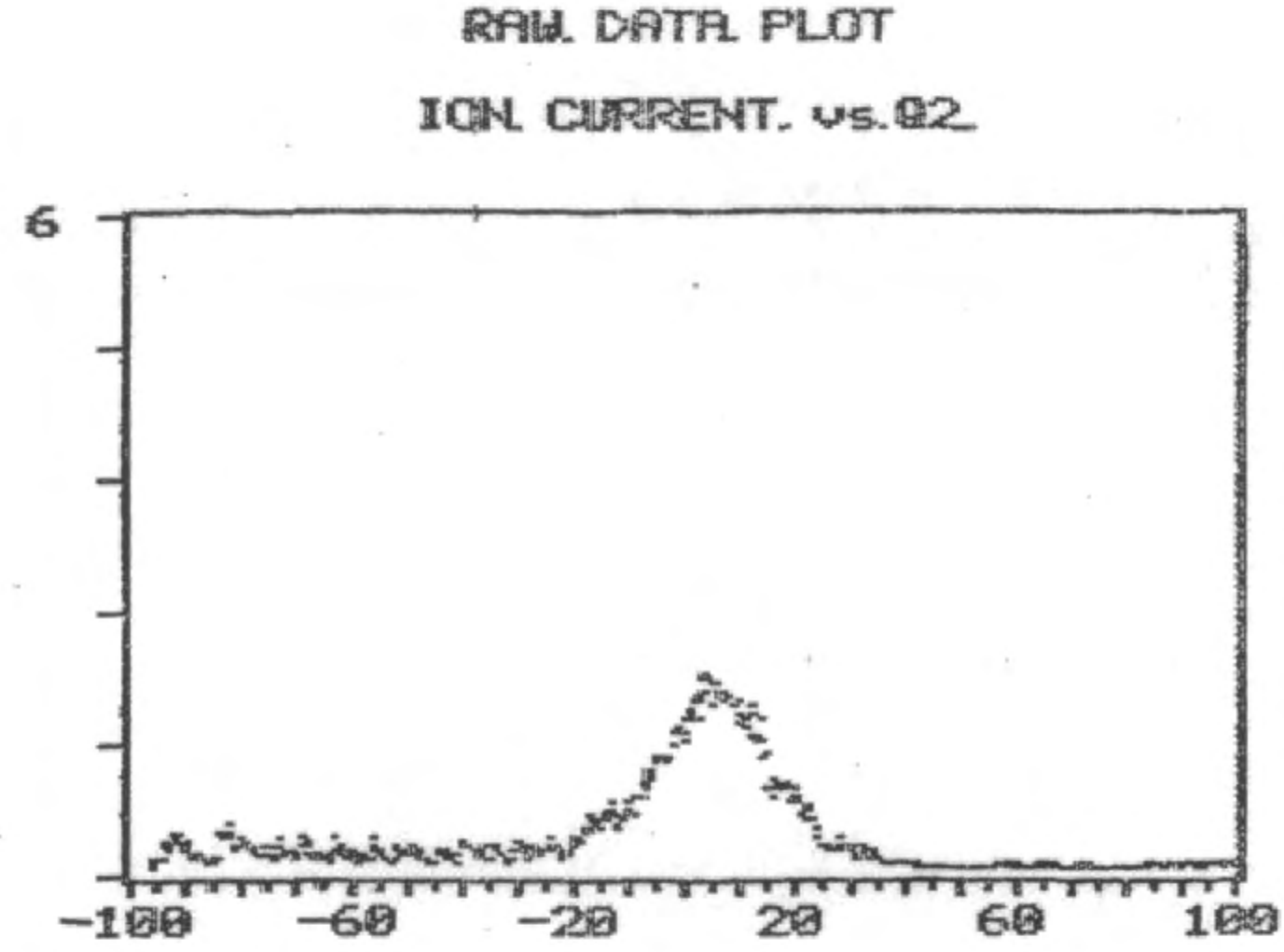


FIGURE 6.7 QUAD 2 SWEEP

The control system is capable of generating three dimensional plots of any two ion path parameters verses ion current. This is done using the CONTOUR-DATA and 3D-PLOT commands described in chapters four and six of the operator's manual found in appendix A. These plots are useful in understanding the interactions of the numerous lens elements. Figure 6.8 is an example of such a plot for lens four potential verses lens six potential.

Review of Objectives

A number of objectives for the control system were outlined in chapter one. The software developed for the control systems meets most of these objectives in whole or in part.

Operating mode selection is completely automatic. Whenever a xSCAN command is executed the control system selects the appropriate RF or DC/RF mode for each quadrupole and initializes the mass selection of each quadrupole to the appropriate values. Automatic calibration of the quadrupole mass selection was not implemented initially due to the instrument's extreme discrimination against higher masses. Recent modifications to the instrument have improved the high mass sensitivity dramatically making it possible to implement automatic mass calibration in the future.

The device control table structure allows new devices to readily be added to the system. During the recent instrument modifications, when several new ion lenses were added to the system, only a half hour was needed to place

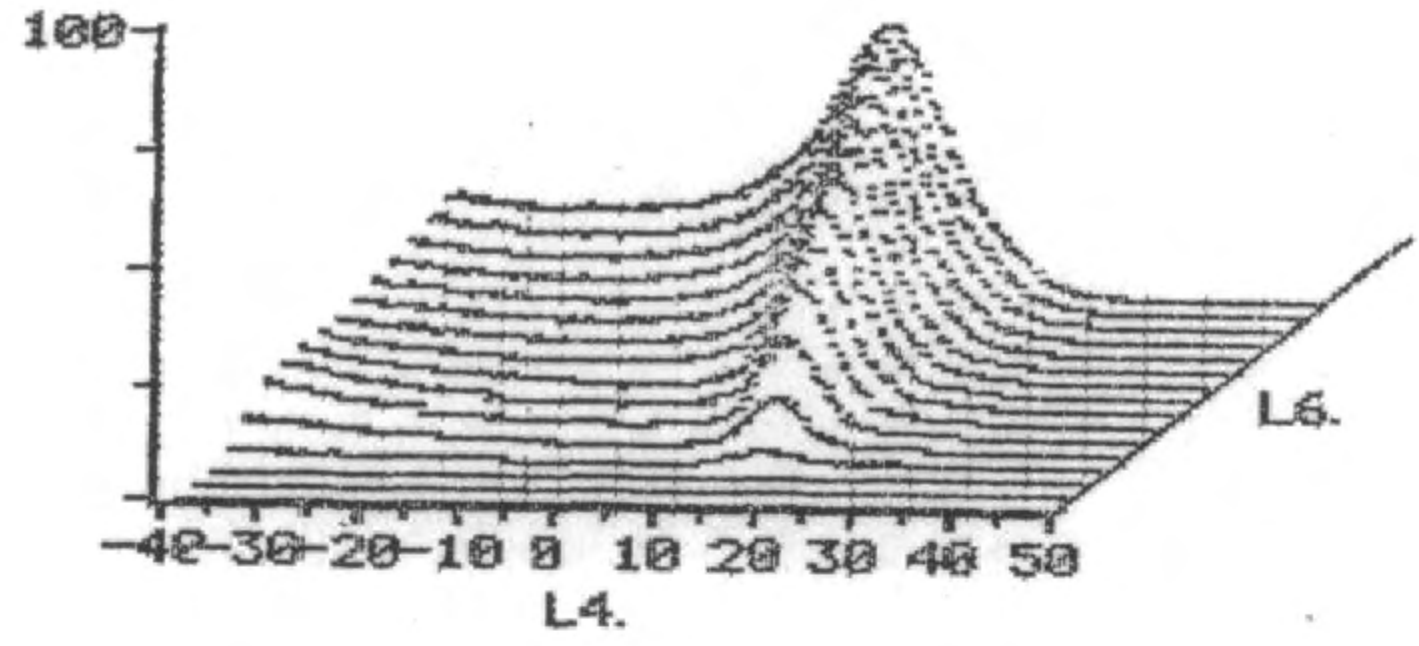


FIGURE 6.8 THREE DIMENSIONAL PLOT

these lenses under computer control. Instrument operators were able to take advantage of these lenses immediately since they are controlled through the DCT by already existing commands.

The use of FORTH allows new commands to be constructed easily out of existing commands by non-programmers. However all the constructs (conditionals, flow control, etc.) of a high level language as well as assembly language programming ability are available for use by the experienced programmer. The FORTH language proves to be an excellent environment for programming laboratory control applications.

Closing Remarks

Portions of this control system have been in use over six months, with the system being essentially complete and debugged for the past three months. Productivity on the instrument has increased dramatically, so that generally the control system has not been the limiting factor in instrument performance. The full capabilities of the control system have not been exploited due to other limitations on instrument performance. As these problems are overcome the control system will greatly facilitate the research projects on the TQMS instrument.

APPENDIX A

OPERATOR'S MANUAL

FOR THE

TQMS CONTROL SYSTEM

CHAPTER 1

CONTROL SYSTEM INTRODUCTION

1.1 HOW TO USE THE CONTROL SYSTEM

1) Read this document.

2) Try it.

3) Read it again.

4) Look it up in the index.

5) When all else fails, but infrequently as possible, ask the programmer.

6) If the programmer is irritable, try a bribe.

1.2 TURNING THINGS ON

The basic control system consists of the microprocessor system in the computer rack, two display screens (the righthand one act as a terminal and the other as a graphics display), a keyboard, and a dual floppy disk drive. The microprocessor system and the floppy disk drive electronics are normally left on all the time, however if no lights are lit on the microprocessor CPU board the power has been turned off. If this is the case consult a higher authority before proceding any further. The display screens and the keyboard are normally left turned off, the small toggle switch above the lefthand screen controls the power to the screens and keyboard. When this switch is placed in the on position (up) several red lights on the keyboard will come on immediately, while it will take several minutes for the display screens to warm up. When a cursor is visible on the righthand screen, press the RETURN key on the keyboard. An "ok" should be printed on the screen, if not try reloading the software. Now enter the STAT command followed by a RETURN, a list of all current parameters settings should be displayed, if not try reloading the system software. If the system software cannot be reloaded, seek help.

1.3 RELOADING THE SYSTEM SOFTWARE

It may be necessary to reload the system software under two conditions, one is if the software is not present when the display screens are turned on. This is indicated by failure to respond to the STAT command. The other is if the system crashes while in use. A crash usually makes itself known by causing the graphics display to flicker and by

refusing to respond to any input from the keyboard. To reload the system software, perform the following steps:

1) Press the reset button on the microprocessor. This is a small pushbutton switch on the computer board with the small LEDs. This should cause the terminal display screen to clear and the message "polyFORTH" to be printed at the top of the screen, if not you need help.

2) Located the floppy disk labeled MASS SPEC SYSTEM. Use this and only this disk. Currently this disk is usually found hiding in the box labeled CARL MYERHOLTZ SOFTWARE.

3) Insert the floppy into drive 0 (the lower drive) with the label side facing down.

4) Type in 9 LOAD followed by a RETURN. The system should start loading itself, reporting each block number as it is loaded. This will take awhile since there is quite a bit of software.

5) Reloading is now complete. Before performing any experiments be sure to reload the current calibration data. A copy of this data is usually stored in block 0 of the MASS SPEC SYSTEM disk. See chapter 7.

6) Return the floppy to its original location.

1.4 THE FLOPPY DISK DRIVE

The control system uses 8 inch floppy disks for data and program storage. Each disk contains 250 data blocks numbered 0-249. The dual disk drive is capable of holding two floppy disks at a time. The lower drive is drive 0 and the upper drive is drive 1. Single sided floppy disks must be inserted with the label side facing down. To remove a disk, press the eject button for the desired drive. The eject buttons are located on the drive. The data blocks of a floppy in drive zero are numbered 0-249 while a floppy in drive one is normally accessed with block numbers 250-499. It is possible to acess data on drive one using block numbers 0-249 by using the FORTH command DRIVE. See a copy of "Using FORTH" for details. Located next to the disk drive are two write protect switches, one switch associated with each drive. When a write protect switch for a given drive is in the up position it is not possible to write anything on to the disk. This feature is used to avoid accidentially destroying the contents of a disk. Normally both write protect switches are left in the off (down) position so that data may be written to the disk. When

reloading the system software it is recommended that drive zero be write protected.

1.5 TYPING AT THE KEYBOARD

All Commands are issued by the user from the keyboard. The control system does distinguish between upper and lower case letters. All of the system commands use only upper case letters, for convenience sake the keyboard should be left in the TTY mode so that all alphabetic characters are upper case. The TTY mode is indicated by the red light in the TTY LOCK key being on. The TTY mode may be toggled on and off by pressing the TTY LOCK key. When the TTY mode is toggled off the keyboard generates lowercase characters. Also note that the keyboard is auto repeating, any key held down for several seconds will start repeating. If a mistake is made while typing in a line the RUBOUT key can be used to backup to where the mistake was made. Each time the RUBOUT key is pressed the cursor is moved left one position. All characters to the right of the cursor are meaningless.

1.6 ENTERING COMMANDS

One or more of the commands discussed in the following chapters may be entered on one line, a space must separate each command. None of the commands are performed until the RETURN key is pressed, then the commands are executed in the order they appear from left to right. When all the commands have been completed the message "ok" is printed.

1.7 THE BREAK KEY

Any command or string of commands may be aborted by pressing the BREAK key. The BREAK key is located just above the righthand display screen. After pressing the break key press the RETURN key to obtain the "ok" message.

CHAPTER 2

SETTING DEVICE PARAMETERS

This chapter covers routines used to set and change the parameters associated with any of the scannable devices present in the system. A scannable device is defined as any device whose value the computer can vary continously (actually in small increments). Each device is given a two or three character abbreviated or mnemonic name. A list of scannable devices presently under computer control is given in table 2.1.

NAME	DEVICE	UNITS	DAC STEP	RANGE
RP	REPELLER	VOLTS	0.1 V	-200.0 to 200.0
IV	ION VOLUME	VOLTS	0.1 V	-200.0 to 200.0
L1	LENS 1	VOLTS	0.1 V	-200.0 to 200.0
L2	LENS 2	VOLTS	0.1 V	-200.0 to 200.0
L3	LENS 3	VOLTS	0.1 V	-200.0 to 200.0
Q1	QUAD 1 POTENTIAL	VOLTS	0.1 V	-200.0 to 200.0
Q2	QUAD 2 POTENTIAL	VOLTS	0.1 V	-200.0 to 200.0
Q3	QUAD 3 POTENTIAL	VOLTS	0.1 V	-200.0 to 200.0
L4	LENS 4	VOLTS	0.1 V	-200.0 to 200.0
L5	LENS 5	VOLTS	0.1 V	-200.0 to 200.0
L6	LENS 6	VOLTS	0.1 V	-200.0 to 200.0
L7	LENS 7	VOLTS	0.1 V	-200.0 to 200.0
L8	LENS 8	VOLTS	0.1 V	-200.0 to 200.0
L9	LENS 9	VOLTS	0.1 V	-200.0 to 200.0
L10	LENS 10	VOLTS	0.1 V	-200.0 to 200.0
M1	QUAD 1 MASS SELECTION	AMU	~1/128 AMU	0.0 to 999.9
M3	QUAD 3 MASS SELECTION	AMU	~1/128 AMU	0.0 to 999.9

TABLE 2.1

All devices are controlled through a device control table, which has entries for the parameters of each device in the system. The user need not be concerned with this table, but should be aware of its existance. There are four parameters associated with each device, the CURRENT value, the START value, the END value, and the STEP size. The CURRENT, START, and END values are expressed with the units of the device (see table 2.1). The STEP size is expressed in DAC units with the value of one DAC unit varying between different types of devices. In the case of all of the ion path potentials one DAC unit equals exactly 0.1 V, thus a STEP of 5 be a 0.5 V change. Only in the case of the mass settings of the quadrupoles is the values of one DAC unit inexact, although typicaly one DAC unit is aproximately 1/128 of an AMU.

CURRENT - The CURRENT value is the value at which the device is normally left set. This is also the value the device is returned to after it has been scanned or swept.

START - The START value is used to control the value at

which a SCAN or SWEEP is begun.

END - The END value controls the ending point of a SCAN or a SWEEP function.

2.1 SET COMMANDS

Each of the four device parameters may be set to a new value with one of the following SET commands.

mn - device mneumonic name
i - an integer number
n - any number in the valid range

2.1.1 SET

n mn SET - Sets the device mn to n, the new value is sent out to the device and entered into the device control table.

2.1.2 SET-START

n mn SET-START - Sets the START value of device mn to n.

2.1.3 SET-END

n mn SET-END - Sets the END value of device mn to n.

2.1.4 SET-STEP

i mn SET-STEP - Sets the STEP size of device mn to i.

2.1.5 SET-CURRENT

mn SET-CURRENT - Outputs the CURRENT value, already stored in the device control table, to device mn. This command is used when there is some doubt as to the value at which the device is currently set.

2.1.6 INIT-DEVICES

INIT-DEVICES - Sets all devices to their CURRENT values. This command is used after any portion of the system (such as the ion controller) has been turn off then on again or when the operator is confused.

2.2 MODIFYING PARAMETERS

There are several commands that allow the user to examine and modify various parameters of one or more devices.

2.2.1 PARAM

The PARAMeter command allows the user to examine all the parameters associated with a given device and alter them if so desired.

PARAM - After a RETURN has been entered the CURRENT value of device mn will be displayed and the cursor will move several spaces to the right. At this point the user may either enter an new value followed by a RETURN or enter only a RETURN to retain the old the value. The user will be similarly queried with the START, END, and STEP values. For example:

```
Q1 PARAM
CURRENT  -10.3   -12.2 (cr)
START    -20.0    (cr)
END       50.0    60 (cr)
STEP         2    (cr)
```

Changes the CURRENT value from -10.3 to -12.2, retains the START value, changes the end value from 50.0 to 60.0 and retains the STEP value.

2.2.2 PARAM-LIST

PARAM-LIST - Performs a PARAM operation for each device in the system.

2.2.3 ION-PATH

The ION-PATH command allows the user to examine and modify the CURRENT value of each device controlled by the Ion Controller, this currently includes L1, L2, L3, IV, RP, Q1, Q2, Q3, L4, L5, L6, L7, L8, L9, and L10. After the CURRENT value for a device has been displayed, the cursor is moved several places to the right. At this point the user may enter a RETURN to retain the displayed value or enter a new value followed by a RETURN. After each device is examined, the CURRENT value is sent out to the device to insure that it is properly set.

2.3 PARAMETER DISPLAY

2.3.1 DEV-STAT

DEV-STAT - Dispalys the entire contents of the device control table, listing the parameters for every device in the system.

2.3.2 STATUS

STAT - Displays the status of all user controlled parameters including aquisition, display, and device parameters.

2.4 SPECIAL PARAMETER OPERATIONS

2.4.1 MS1

n MS1 - Set Quad 1 to mass n, also set the current value in the device control table to n. This command is exactly equivalent to the command sequence n M1 SET.

2.4.2 MS3

n MS3 - Set Quad 3 to mass n, also set the CURRENT value in the device control table to n. This command is exactly equivalent to the command sequence n M3 SET.

2.4.3 MANUAL

mn MANUAL - Gives the user manual control of the CURRENT value of device mn. Manual control is accomplished with the use of the four arrow (cursor control) keys. Pressing either the up-arrow or the right-arrow key causes the STEP value to be added to the CURRENT value, and device mn is set to this value. The CURRENT value of the device is displayed on the terminal screen and is also continously updated in the device control table. Pressing either the down-arrow or the left arrow key cause the STEP value to be subtracted from the CURRENT value. To exit the manual mode press the space bar.

2.4.4 ALL-SET

i ALL-SET - Sets all ion path devices to the integer potential i.

CHAPTER 3

MASS SCANNING FUNCTIONS

3.1 SCANNING MODES

There are five modes of scanning the quadrupoles available to the user. These are as follows.

QUAD 1 SCAN - In this mode the instrument is set into the DC/RF/RF mode and Quad 1 is scanned.

QUAD 3 SCAN - in this mode the instrument is set into the RF/RF/DC mode and Quad 3 is scanned.

DAUGHTER SCAN - In this mode the instrument is set into the DC/RF/DC mode with Quad 1 set to a user selected value and Quad 3 is scanned.

PARENT SCAN - In this mode the instrument is set into the DC/RF/DC mode with Quad 3 set to a user selected value and Quad 1 is scannned.

NEUTRAL LOSS SCAN - In this mode the instrument is set into the DC/RF/DC mode and Quads 1 and 3 are scanned at a constant offset from each other.

The phrase SCAN appears only in command words that scan the mass setting of one or more quadrupoles. There are three scanning functions available, one is Fast SCANning for oscilloscope display, another is SCANning with data acquisition, and the third is the SPLIT screen display mode.

3.2 SCANning with Data Acquisition

There is one SCAN command word for each of the five scanning modes.

1SCAN - Performs a Quad 1 Scan by putting the quads in the DC/RF/RF mode and scanning Quad 1 from its START to END values using STEP as the increment. Data is acquired using the current acquisition parameters and stored in the data buffer.

3SCAN - Performs a Quad 3 Scan by putting the quads in the RF/RF/DC mode and scanning Quad 3 from its START to END values using STEP as the increment. Data is acquired using the current acquisition parameters and stored in the data buffer.

DSCAN - Performs a Daughter Scan by putting the quads in the DC/RF/DC mode, setting Quad 1 to its CURRENT set value and scanning Quad 3 from its START to END values using STEP as

the increment. Data is acquired using the current acquisition parameters and stored in the data buffer.

PSCAN - Performs a Parent Scan by putting the quads in the DC/RF/DC mode, setting Quad 3 to its CURRENT set value and scanning Quad 1 from its START to END values using STEP as the increment. Data is acquired using the current acquisition parameters and stored in the data buffer.

NSCAN - Performs a Neutral Loss Scan by putting the quads in the DC/RF/DC mode and scanning both Quad 1 and Quad 3 together at a constant offset. The offset is determined by the difference between the START values of Quads 1 and 3 in the parameter table. Both Quads are started at their respective START values. The range of the scan is determined by the START and END values of Quad 1. The increment is also controlled by Quad 1. Data is acquired using the current acquisition parameters and stored in the data buffer.

A scanning mode may be selected and subsequently performed with the general command SCAN.

SMODE xSCAN - Sets the scanning mode to the given mode. To select a Quad 3 scan for the scanning mode execute the command SMODE 3SCAN (cr).

SCAN - Implements the scanning mode selected with the MODE command.

3.3 FAST SCANS

The Fast SCANning mode allows the quadrupoles to be scanned fast enough for the user to see a mass spectrum on the display oscilloscope. No data is stored during a Fast SCAN operation. There is one Fast SCAN command word for each of the five scanning modes. All Fast SCAN modes use the same parameters as their SCAN counterparts.

F1SCAN - Fast Quad 1 Scan

F3SCAN - Fast Quad 3 Scan

FDSCAN - Fast Daughter Scan

FPSCAN - Fast Parent Scan

FNSCAN - Fast Neutral Loss Scan

A fast scanning mode may be selected and subsequently

performed with the general command FSCAN.

FMODE FxSCAN - Sets the fast scanning mode to the given mode. To select a Quad 3 scan for the fast scanning mode execute the command FMODE F3SCAN (cr).

FSCAN - Implements the fast scanning mode selected with the FMODE command.

The speed of a fast scan can be controlled with the command FSPEED.

n FSPEED - Sets the fast scanning speed parameter to n.

Valid values of n range from 1 to 255. The value of n has no direct physical significance, it should be thought of as a slowness factor. A value of n=1 gives the fastest possible scan, while a value of n=255 gives the slowest possible scan (255 times slower than n=1).

The Fast SCAN commands unlike most other functions in this software package initiate a background task which performs the actual scanning of the quads. This means the user can perform operations from the terminal while the quads are being scanned, however the speed of execution and output will be adversely affected. This feature allows the user to vary ion path parameters while simultaneously observing the results on the oscilloscope display. To terminate a backgound task performing a Fast SCAN enter the command FSTOP.

FSTOP - Terminates a Fast SCANning operation.

3.4 SPLIT SCREEN MODE

The SPLIT screen mode allows the user to display any two peaks side by side on the display oscilloscope, making it possible to compare two peaks at greatly different masses during tuning operations. A three AMU window centered around each user selected peak is displayed on the oscilloscope and the quadrupoles are scanned using the smallest possible step size. The SPLIT screen mode is a special case of the fast scanning mode and is similar in many ways. SPLIT mode commands start a background task just like Fast SCAN commands. The scanning speed may be varied with the FSPEED command and a SPLIT command is terminated with the FSTOP command. No data is stored in any of the SPLIT modes. There are four SPLIT screen commands as follows:

m1 m2 1SPLIT - Performs a fast scan of Quad 1 displaying peaks centered at masses m1 and m2.

m1 m2 3SPLIT - Performs a fast scan of Quad 3 displaying peaks centered at masses m1 and m2.

m1 m2 DSPLIT - Performs a fast daughter scan selecting the parent ion as the CURRENT set value of M1 (Quad 1) and displaying daughter peaks centered at m1 and m2 that are scanned by Quad 3.

m1 m2 PSPLIT - Performs a fast parent scan selecting the daughter ion as the CURRENT set value of M3 (Quad 3) and displaying parent peaks centered at m1 and m2 that are scanned by Quad 1.

CHAPTER 4

DEVICE SWEEPING FUNCTIONS

Routines that scan any device controlled by the Device Control Table have the phrase SWEEP as part of their command name. All SWEEP functions acquire data and store it in the data buffer while they scan the selected parameter. SWEEP functions are limited to taking only 500 data points, if an attempt is made to take more than 500 data points only the first 500 points are stored. None of the SWEEP functions affect the settings of the RF/DC realys.

mn = a device mnemonic name. e.g. Q1,L2,EV ect.

4.1 SWEEP

mn SWEEP - Scans the device mn from its START to END values taking a data point each time the value is incremented by the value of STEP. When the scan is complete the device is reset to its CURRENT set value.

4.2 LINKED SWEEP

The Linked SWEEP mode allows up to sixteen ion path potenials to be linked and swept together. Devices to be linked are specified with the LINK-SET command. One device is designated the master device and controls the range and increment of the sweep. A Linked SWEEP is accomplished by setting all of the linked devices to their START values and then incrementing all of them by the STEP value of the master device until the master device reaches it's END value. A data point is taken after each increment and is stored in the data buffer along with the value of the master device. When the Linked SWEEP is completed all linked devices are returned to their CURRENT set values.

4.2.1 LINKING DEVICES

Up to sixteen ion path devices may be linked using the LINK-SET command. The user enters a list of device names to be linked followed by the command LINK-SET. The first device is taken to be the master device.

dev1 dev2 ... devn LINK-SET - Links devices dev1 through devn using dev1 as the master device. e.g. Q2 Q1 Q3 LINK-SET

Occasionally this command will generate the error message

"Invalid Device Selection". When this occurs just reenter the device list and LINK-SET.

4.2.2 LINK-LIST

To obtain a list of all devices currently linked together use the command LINK-LIST. This command list the mnemonic names of all the linked devices with the first device's name highlighted in reverse video to indicate that it is the master device.

LINK-LIST - List the currently linked devices.

4.2.3 LSWEEP

LSWEEP - Performs the linked sweep function sweeping all the devices linked with the LINK-SET command and storing the results in the data buffer.

4.3 TUNE

The TUNE function is designed to aid the user in tuning up the ion path. It automatically sets the instrument into the RF/RF/RF mode, scans the selected ion path device from -100V to +100V and plots the results on the graphics display. The parameters of the selected device are not affected. However the instrument is left in the RF/RF/RF mode, the acquisition mode is set to AutoRanging down to range 10 and the display mode is set to LOG down to range 10. The previous contents of the data buffer are destroyed.

mn TUNE - Perform the TUNE function for the ion path device mn.

4.4 TUNE-UP

The TUNE-UP function allows the user to quickly setup the ion path. TUNE-UP starts with the first device in the ion path and sequentially sweeps each device from -100 to +100 volts. After each sweep is completed it is displayed on the graphic screen. The current value of the device is displayed on the terminal screen and the cursor is moved

several places to the right. At this point the user may enter a RETURN to retain the displayed value or enter a new value followed by a RETURN. The control system then procedes to sweep the next ion path device.

TUNE-UP - Sequentially sweep all ion path devices, display the results and allow the user to modify the current settings of each device.

4.5 CONTOUR-DATA

The CONTOUR-DATA command allows the user to obtain data needed to generate three dimensional contour maps. One dimension is intensity; the other two are obtained by setting one deivce to a specified valued, SWEEPing a second device, then incrementing the first device and repeating the process.

mn1 mn2 CONTOUR-DATA - Sets device mn1 to its START value then SWEEPs device mn2. The data is then stored with the SDATA command (see chapter 7), the data header is printed (also see chapter 7). Then mn1 is incremented by its STEP value and the process is repeated until the END value of mn1 is reached.

CHAPTER 5

DATA ACQUISITION PARAMETERS

There are three data acquisition parameters under user control. These are Acquisition mode, Range, and Peak Width.

5.1 ACQUISITION MODE

Currently there are only two valid acquisition modes.

FIX - FIXed range

AR - AutoRanging

In the FIXed mode the gain of the detection amplifier is set to one of the five possible ranges and left there during the entire data gathering operation. In the AutoRanging mode the computer switches the gain between the 6 range and a lower limit set by the user. This switching of ranges allows the computer to optimize the signal intensity for each data point, giving the instrument a much larger dynamic range for one scan. When using the AutoRanging mode it is essential that the computer/manual switch associated with range selection be in the computer position, otherwise the data is meaningless. This switch is located next to the manual range selection switch.

5.2 ACQUISITION RANGE

This parameter determines the amplifier range used in the FIXed mode or the lower limit used in the AutoRanging mode. The valid values for range are 6 through 10.

5.3 PEAK WIDTH

The value of PEAK WIDTH determines how wide a peak must be in DAC units in order to be considered a valid peak. Peaks narrower than PEAK WIDTH are rejected as noise. As a general rule the value of PEAK WIDTH should be at least three times the STEP value of the quadrupole being scanned. This requires that at least three consecutive data points be above threshold value before a peak is identified.

5.4 SETTING ACQUISITION PARAMETERS

All three acquisition parameters can be set or changed with the Acquisition PARAMeter command A-PARAM. After the command A-PARAM is entered followed by a RETURN the current acquisition mode will be displayed and the cursor will move several spaces to the right of the mode listing. At this point the user may either enter a new mode followed by a RETURN or enter only a RETURN to retain the old mode entry. Note that only FIX and AR are valid responses for the mode. Next the current value of the acquistion range is displayed. The user has the same option of entering a new value or retaining the old setting. Finally the PEAK WIDTH is displayed and can be modified or retained. For example:

```
A-PARAM
ACQUISITION MODE    AR     (cr)
RANGE                8     10 (cr)
PEAK WIDTH          30     40 (cr)
```

Retains the AutoRanging mode, changes the aquisition range from 8 to 10 and changes the peak width from 30 to 40.

5.5 LISTING ACQUISITION PARAMETERS

The current status of the acquisition parameters may be displayed with the A-STAT command.

A-STAT - List the current acquisition parameters

The current acquisition parameters are also displayed as part of the STATus command, along with the display parameters and the device parameters.

5.6 NO-AVERAGE

NO-AVERAGE - Disables averaging during data aquisition. This speeds up SCANs and SWEEPs, but in the case of a mass scan may cause some peaks to be missed.

5.7 AVERAGE

AVERAGE - Enables averaging during data aquisition. This should be the normal mode of operation.

CHAPTER 6

DISPLAY FUNCTIONS

This chapter deals with commands and parameters which display data on the computer graphics display. All data taken with either a SCAN function or a SWEEP function is stored in the data buffer, the routines discussed in this chapter all involve displaying the currrent contents of the data buffer on the graphics screen. Two types of data can be displayed on the graphics screen, mass data taken with a SCAN function or raw data taken with a SWEEP function. Mass data can be displayed in up to five fields. That is, the x axis can be divided into up to five segments when plotted on the graphics screen. Raw data is displayed in only one field.

6.1 DISPLAY PARAMETERS

There are five parameters that control how data is displayed on the graphics screen. These are MODE, RANGE, # FIELDS, AMU/FIELD and START MASS. MODE and RANGE affect the plotting of both mass and raw data, while # FIELDS, AMU/FIELD and START MASS only affect the display of mass data.

6.1.1 DISPLAY MODE

The display mode controls the plotting of the y axis, which is always ion current. There are only two valid display modes.

LOG - LOGarithmic

LIN - LINear

The LOG mode plots the data on a true base 10 logarithmic scale. The upper limit corresponds to a full scale reading on the 6 range and the lower limit corresponds to 10% of the range selected by the range parameter. This mode is most useful in displaying data taken in the AutoRanging mode. In the LINear mode the upper limit corresponds to 100% of the range selected by the range parameter and the lower limit corresponds to zero. A 6 displayed at the upper end of the y axis indicates the data was plotted in the LOG mode while a 100 indicates the LIN mode was used.

6.1.2 DISPLAY RANGE

This parameter determines the intensity range of data to be plotted, as described in the display mode section. Valid values for the range are 6 through 10.

6.1.3 NUMBER OF FIELDS (# FIELDS)

This parameter controls the number of fields used to display mass data. If the value is zero the computer determines how many fields to use. The user may specify the number of fields to be used by entering a value between 1 and 5.

6.1.4 AMU/FIELD

This parameter controls the number of amu each field covers. This number must be a positive integer and should not exceed 1000. If the value is set to zero the computer will determine how many amu to display in each field.

6.1.5 START MASS

This parameter controls the starting point of the x axis, which is the left most edge of the first field displayed. This number must be a positive integer and should not exceed 1000. If the value is set to zero the computer will use the START value of the scan being displayed as the START MASS.

6.2 SETTING DISPLAY PARAMETERS

All display parameters can be set or changed with the Display PARAMeter command D-PARAM. This command behaves in the same way for the display parameters as the A-PARAM command does for the acquisition parameters. For example:

```
D-PARAM  (cr)
MODE      LOG   LIN (cr)
RANGE      10    8  (cr)
```

```
# FIELDS     0  (cr)
AMU/FIELD    0  (cr)
START MASS 20   0 (cr)
```

Changes the mode from LOG to LIN, changes the display range from 10 to 8, retains the zero values of # FIELDS, and AMU/FIELD and changes the START MASS from 20 to 0.

6.3 LISTING DISPLAY PARAMETERS

The current status of the display parameters may be seen by entering the D-STAT command.

D-STAT - List the current display parameters

The current display parameters are also displayed as part of the STATus command, along with the acquisition parameters and the device parameters.

6.4 DISPLAYING DATA

The current contents of the data buffers can be displayed with the command DISP.

DISP - Display the current contents of the data buffer.

This command will display either mass data or raw data, whichever is currently in the data buffer. This command is most often used immediatly after a SCAN or SWEEP function. After this data has been displayed the display parameters can be altered and the same data replotted by issuing another DISP command.

6.5 OVERLAYING DATA PLOTS

Sometimes it may be desirable to superimpose a set of data on an already exsisting display. This most often involves data taken with several different SWEEP functions. Do not attempt to plot raw data on a mass plot or vice versa. To superimpose the current contents of the data buffer on a existing display use the command OVER-PLOT.

OVER-PLOT - Display the current contents of the data buffer without erasing the screen.

6.6 DISPLAYING CONTOUR DATA

Data taken with the CONTOUR-DATA command can be displayed with the 3D-PLOT command to generate a three dimensional contour map.

m n 3D-PLOT - Generates a pseudo three dimensional plot using data stored in data blocks m through n that was generated with the CONTOUR-DATA command.

6.7 LABEL

It is possible to display a line of text at the top edge of the graphics display with the LABEL command. The text is limited to 42 characters and is completly erased with the next DISP or CLEAR command. This is useful for putting a label on a plot to be hardcopied with the video printer.

LABEL text - Displays text at the top edge of the graphics display.

6.8 CLEARING THE GRAPHICS DISPLAY

The graphics display may be erased with the CLEAR command.

CLEAR - Erases the graphics display to all black.

The user should clear the graphics display whenever he or she is done using the system.

CHAPTER 7

STORAGE AND RETREIVAL

This chapter deals with commands which allow the user to read or write the contents of either the parameter buffer or the data buffer to or from a floppy disk. The act of writing either buffer does not effect the current contents of that buffer.

7.1 PARAMETER BUFFER OPERATIONS

The parameter buffer contains the device, display, and data acquisition parameters as well as the list of linked devices, the Fast scan MODE, the Scan MODE, the value of FSPEED, and the Quad calibration tables. The entire contents of the parameter buffer can be stored in one data block.

7.1.1 STORING THE PARAMETER BUFFER

The entire parameter buffer can be written onto a floppy disk with the SAVE command.

n SAVE - Write the parameter buffer into data block n.

It is suggested that blocks 1 thru 8 be used to store different versions of the parameter buffer on one diskette, however any block 0 - 249 may be used if so desired.

7.1.2 LOADING THE PARAMETER BUFFER

The parameter buffer may be loaded from any data block which has already had data written into it with the SAVE command by using the GET command.

n GET - Load the parameter buffer with the contents of data block n.

The GET command recovers all the stored parameters except the Quad calibration tables. The current Quad calibration tables remain unaffected.

7.1.3 OLD-GET

The OLD-GET command allows the user to recover device settings that were SAVEd before the nasty programmer changed versions of the software.

n OLD-GET - Loads device settings into the parameter buffer from block n.

7.1.4 LOADING THE CALIBRATION TABLES

The Quad calibration tables can be loaded from a floppy disk with the CAL-GET command. This command only loads the Quad calibration tables and does not affect any of the other parameters.

n CAL-GET - Loads the Quad calibration tables from the data stored in data block n.

About the only time this command is needed is after rebooting the system.

7.2 DATA BUFFER OPERATIONS

The data buffer contains the data acquired by the last SCAN or SWEEP command. The data buffers is 2K bytes long and thus occupies two data blocks when stored on disk. The contents of the data buffer can either be written to, or read from floppy disk. Since the data buffer requires two consecutive data blocks for storage, it is recommended that data always be stored starting at an even block number.

7.2.1 STORING DATA

The data currently in the data buffer can be stored on a floppy disk with the following commands:

n WDATA - Writes the current contents of the data buffer into two consecutive data blocks on a floppy disk starting at block n.

SDATA - Writes the contents of the data buffers into the next two data blocks. Whenever a WDATA, RDATA, or EXPT,

command is performed the computer stores the block number used. The SDATA command adds two to this number and writes the data buffer out to this new block numbers saving the new number. The user should exercise caution when using this command. For instance, the sequence

```
100 WDATA
20 RDATA
SDATA
```

would result in the storage of data in blocks 22 and 23, not 102 and 103.

7.2.2 READING DATA

Data which has previously been stored on a floppy disk can be read into the data buffer with the following command.

n RDATA - Reads the contents of two consecutive data blocks from a floppy disking into the data buffer starting at block n.

7.3 EXPERIMENT HEADER

The experiment header allows the user to store information about an experiment on a disk. An experimental run may consist of one or more data files. The EXPT command queries the user for information about the experiment and then writes this information and the current contents of the parameter buffer onto a floppy disk. The user is queried for the following information:

```
TITLE:              max of 8 characters
DATE:               max of  8 characters
TIME:               max of 12 characters
OPERATOR:           max of 16 characters
IONIZATION MODE:    max of 16 characters
NOTE1:              max of 64 characters
NOTE2:              max of 64 characters
NOTE3:              max of 64 characters
NOTE4:              max of 64 characters
```

7.3.1 EXPT

n EXPT - Queries the user for the header information and then stores it in block n and then stores the parameter buffer in n+1.

7.3.2 EXPT-PRINT

n EXPT-PRINT - display the experiment header information stored in block n.

7.4 DATA-HEADER

Whenever a data file is written to disk, along with the data, some header information is stored. Among other things this header includes the current title, scan #, mode (3SCAN,SWEEP, etc), device scanned, START END and STEP values of the device scanned, the values of the quad 1 and quad 3 mass setting and the number of data points taken. If a quad was in the RF mode, RF is printed instead of the mass value. Likewise if the quad was scanned, SC will be printed. The DIR and CONTOUR-DATA commands print out this header in the order listed above prefixed by a block number. The data header for the current contents of the data buffer may be displayed with the DATA-HEADER command.

DATA-HEADER - Displays the data header for the current contents of the data buffer.

7.5 DISK DIRECTORY

The DIR command is used to find out what data has already been stored on a floppy disk. If no data has been stored in a given block, just the block number is displayed. If parameter data is present, the block number and the message " PARAMETER DATA" is displayed. If an experiment header file is encountered, the TITLE, DATE and TIME are displayed. If a data file is encountered, the data header is displayed.

n DIR - Lists the contents of the disk starting at block n. If any key is pressed, output stops.

CHAPTER 8

MISCELLANEOUS

8.1 RF/DC RELAY CONTROL

The relays that control whether or not a quadrupole is in the RF (ion pipe) or DC (mass selection) mode have been placed under computer control. The following commands allows the user to control the settings of these relays .

n RF - Switches Quad n to the RF mode

n DC - Switches Quad n to the DC mode

RRR - Selects the RF/RF/RF mode

RRD - Selects the RF/RF/DC mode

DRR - Selects the DC/RF/RF mode

DRD - Selects the DC/RF/DC mode

RDR - Selects the RF/DC/RF mode

8.2 QMODE

QMODE - Displays the current status of the quadrupole RF/DC relays.

8.3 USER DEFINABLE KEYS

The five function keys f1,f2,f3,f4, and f5 are user definable. Each of these five keys may be set equivalent to any command word. Once the function of a key has been defined, that key may be used in place of the command word with the same results. The use of a user defined key is governed by the same rules as command words, they must be separated from other command words by a space and are not executed until a return is entered. It should be noted that none of the function keys generate any printing on the terminal when they are pressed, so care must be exercised when they are used.

The commands that assign a command to a user definable key are as follows:

F1= command - Set the f1 key equal to command

F2= command - Set the f2 key equal to command

F3= command - Set the f3 key equal to command

F4= command - Set the f4 key equal to command

F5= command - Set the f5 key equal to command

8.4 TALK

It is possible to communicate with the LSI 11/23 from the mass spec terminal. This is accomplished by issuing the TALK command followed by two RETURNs. At this point a prompt from the 11/23 should appear on the terminal screen.

TALK - Connect the user with the LSI 11/23.

To return to communicating with the microprocessor press the BREAK key. Note that it is not possible to communicate with both the 11/23 and the mass spec at the same time.

APPENDIX B

The Multiplexer/Offset Board

In a conventional quadrupole mass spectrometer, the potential that controls the mass selection of the quadrupole is also connected to the X axis control of an oscilloscope. When the detector output is connected to the Y axis of the oscilloscope, a mass spectrum can be displayed on the oscilloscope. If the mass range is scanned very rapidly (> 10 times a second) the mass spectrum is displayed continuously and allows the operator to easily observe the effects of tuning various parameters. In a TQMS either or both of the quadrupoles may be scanned and the choice of which quadrupole's sweep potential should be connected to the X axis on the oscilloscope depends on the nature of the experiment. In the case of a daughter scan experiment, the sweep potential from quadrupole three should be used; for parent or neutral loss scans, the sweep potential for quadrupole one should be connected to the oscilloscope.

When a conventional oscilloscope is used to monitor a mass spectrum as described, it is often difficult to closely compare or observe two peaks widely separated in mass. For example, when using PFTBA to tune up the instrument it is often desirable to monitor the intensity and peak shape of the mass 69 and 219 fragments. However when the X axis gain

on the oscilloscope is increased sufficiently to observe the peak shape in detail, the peaks are spread so far apart that only one peak can displayed on the screen. In order to display two widely separated peaks side by side on the oscilloscope screen it is necessary to subtract an offset voltage from the sweep potential driving the oscilloscope while scanning the higher mass peak. This causes the peak to shift toward the lower mass peak on the oscilloscope screen.

Instead of providing the operator with a switch to select which sweep potential drives the X axis on the oscilloscope, an analog multiplexer was interfaced to the microprocessor address and data bus. This interface was designed so that when a quadrupole number (1, 2, or 3) is written into a specific memory location, the sweep potential for that quadrupole is selected to drive the oscilloscope. The control system uses this ability to automatically select for the user which quadrupole sweep potential will drive the oscilloscope depending on the type of experiment being run.

An operational amplifier in the difference amplifier configuration is used to subtract an offset voltage from the sweep potential. Since the microprocessor system controls the sweep potential, an 8-bit digital to analog converter (DAC) was interfaced to the data bus to generate the offset voltage. This allows the microprocessor to subtract the offset voltage at the desired point in the scan.

Figure B.1 shows a block diagram of the system. The latch is loaded by the computer with the desired quadrupole number, this number is used by the multiplexer to select the

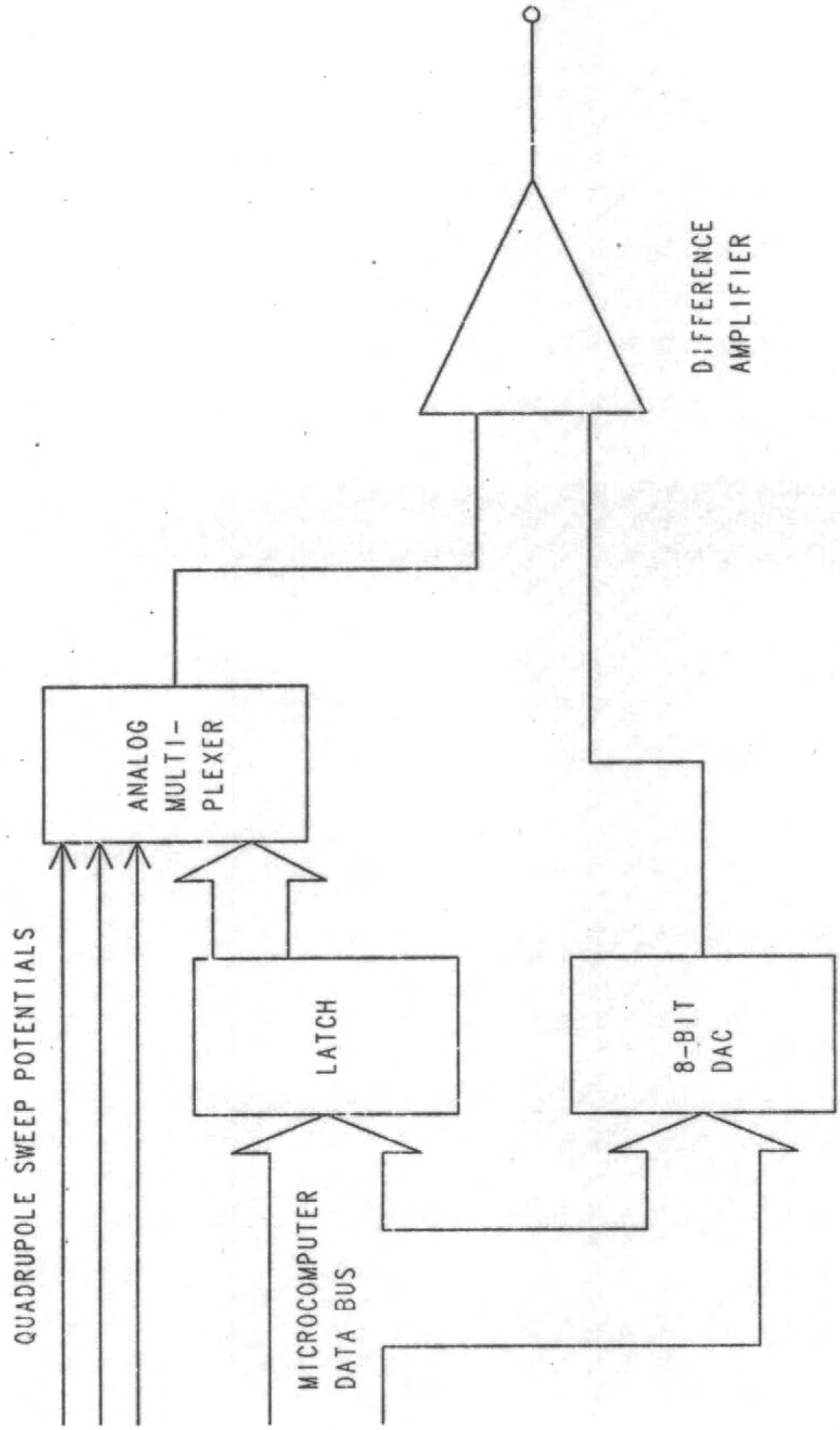
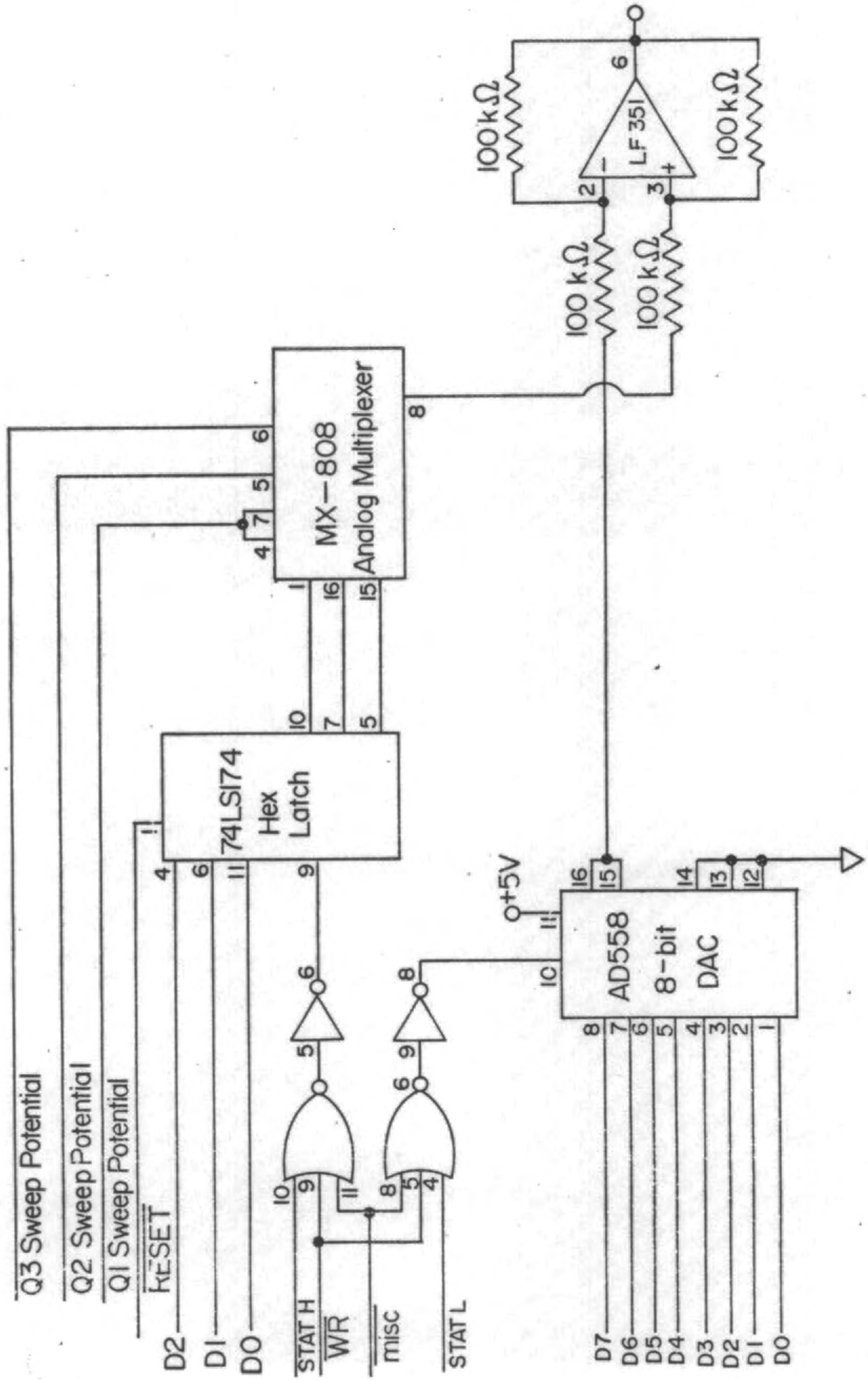


FIGURE B.1 MULTIPLEXER/OFFSET BLOCK DIAGRAM

sweep potential from the desired quadrupole. The output of the multiplexer is fed to the positive input of the difference amplifier. The DAC generates an offset voltage programmed by the computer which is fed into the minus input of the amplifier. The algebraic difference of these two voltages appears at the output of the amplifier and this signal is used to drive the X axis of an oscilloscope. These two features combined together allow the control system to display the desired information to the user without the user needing to make any adjustments of controls. This allows the user to concentrate of the experiment rather than on instrument setup. Figure B.2 is a complete schematic diagram of the multiplexer/offset board.


100kΩ
6
LF351
2
3
100kΩ
100kΩ
100kΩ
MX—808
Analog Multiplexer
74LS174
Hex
Latch
Q3 Sweep Potential
Q2 Sweep Potential
Q1 Sweep Potential
RESET
D2
D1
D0
STAT H
WR
misc
STAT L
+5V
AD558
8-bit
DAC
D7
D6
D5
D4
D3
D2
D1
D0

REFERENCES

1. R.A. Yost, and C.G. Enke, Anal. Chem. 50, 1251A (1979)

2. R.A. Yost, C.G. Enke, D.G. McGilvery, D. Smith, and J.D. Morrision, Int. J. Mass Spectrom. Ion Phys. 30, 127 (1979)

3. "Quadrupole Power Supply Manual", Extranuclear Laboratories Inc., Pittsburg, PA. (1974)

4. R.K. Latven, Ph.D. Thesis, Michigan State University (1981)

5. Kratos Analytical Instruments, Westwood, New Jersey

6. Finnigan MAT, Sunnyvale, California

7. V.G. Datasystems Limited, Cheshire, England

8. Sciex, Thornhill, Ontario, Canada

9. E. Ziegler, Anal. Chim. Acta 122, 315 (1980)

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

11. Bruce Newcome, Dept. of Chemistry, Michigan State University (1982)

12. Model ADC-HY12BC, Datel Systems Inc., Mansfield, MA.

13. Model 4870, Galileo Electro-Optics Corp., Sturbridge, MA.

14. Phil Hoffam, Dept. of Chemistry, Michigan State University (1982)

15. Product specification on model 277 disk drive, Persci Corp., Marina Del Rey, CA. (1976)

16. Product specification on model M7142 display module. Digital Equipment Corporation, Maynard, MA. (1980)

17. Product specification on model RG-SBC-GG1 graphics generator, Raster Graphics, Tigard, Oregon (1981)

18. R. Matthews, Ms. Thesis, Michigan State University (1982)

19. Ralph Thiim, Dept. of Chemistry, Michigan State Univeristy (1982)

20. P. Wintz, Digital Design 10(11), 30 (1980)

21. P. Turvill, Microcomputing 61, 32 (1982)

22. C. Moore, Byte 5(8), 76 (1980)

23. E. Rather, L. Brodie, and C. Rosenberg, "Using FORTH", Second Ed., FORTH Inc, Hermosa Beach, CA (1980)

24. C. Moore, E. Rather, "The Use of FORTH in Process Control" presented at the International '77 Mini-Micro Computer Conference in Geneva, May 26,1977

25. S. Hicks, "FORTH: A Cost Saving Approach to Software Development", FORTH Inc., Hermosa Beach, CA

26. The Big Board Computer available from Digital Research Computers of Texas, Garland, Texas

27. FORTH Inc., Hermosa Beach, CA

28. Model DAC71-CSB-V, Burr-Brown Research Corporation, Tucson, AZ

29. D. Zakett, P.H. Hemberger, R.G. Cooks, Anal. Chim. Acta 119, 149 (1980)

30. H. Chamberlin, "Musical Applications of Microprocessors", Hayden Book Company, Rochelle Park, NJ, Chapter 13 (1980)

31. "IEEE Standard Dictionary of Electrical and Electronic Terms", Institute of Electrical and Electronics Engineers Inc., New York, NY, 742 (1977)

32. "Linear Databook", National Semiconductor Corporation, Santa Clara, CA, 3-39 (1978)

33. "Analog Switches and Their Applications", Siliconix Inc., Santa Clara, CA, 2-28 (1980)