ABSTRACT

DEVELOPMENTS IN TRIPLE QUADRUPOLE MASS SPECTROMETRY

I. A Distributed Processing Control System

II. Screening Applications for Fuel Analysis

by

Carl Alan Myerholtz

A data acquisition and control system for a triple quadrupole mass spectrometer has been developed using several microprocessors in a distributed processing system. This system includes four processors, one acting as the system master controlling three slave processors. In such a distributed processing system each processor is assigned a specific task. Critical to this application is the allocation of the task of data acquisition, ion path control, and peak finding to separate slave processors. This modular approach leads to a system where each major section of the instrument has its own dedicated intelligence.

This parallel processing system allows operations that are often implemented in hardware (for speed considerations) to be performed in software. For an instrument operating in the research environment, the flexibility of a primarily software based system is a great benefit. In this implementation both the hardware and the software become more modular, making it easier to implement and test different data acquisition, peak finding, and scanning algorithms.

The use of triple quadrupole mass spectrometry, an MS/MS technique, to detect selected species in middle distillate fuels has been examined. Nonparaffinic components, which are mainly aromatic and heteroaromatics containing nitrogen or sulfur, contribute to the formation of undesirable deposits during the storage and combustion are of particular interest where aviation fuels are concerned. Collision-activated dissociation (CAD) spectra were obtained for reference compounds from several heteroatom-containing compound classes. These included the thiophenes, thiols, nitrobenzenes, pyridines and anilines. The alkylbenzenes were examined in addition to heteroatom-containing species. The CAD results were used to select screening reactions for each compound class. The effectiveness of these screening reactions was demonstrated by identifying the presence of various species in samples of Jet A aviation fuel, a shale oil derived fuel and No. 2 diesel fuel. Triple quadrupole mass spectrometry can be used to rapidly identify a number of different components in middle distillate fuels. This information can be an aid to studies of fuel composition and stability.

DEVELOPMENTS IN TRIPLE QUADRUPOLE MASS SPECTROMETRY

I. A Distributed Processing Control System

II. Screening Applications for Fuel Analysis

by

Carl Alan Myerholtz

A DISSERTATION

Submitted to

Michigan State University

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

Department of Chemistry

1983

ACKNOWLEDGMENTS

I would like to thank Dr. Chris Enke under whose guidance this work was performed, for providing an environment of freedom and challenge. I thank the fellow members of the research group and the department for making Michigan State a unique place to work. I would like to recognize Dan Sheffield for is contribution to the quality of many of the figures presented here.

Financial support for myself was provided by grants from the National Aeronautics and Space Administration (NASA) and Extranuclear Inc. Financial support for the TQMS instrumentation project was provided by the Office of Naval Research (ONR).

Finally I would like to express my appreciation to my many friends and my parents for their support during my stay in Michigan. I would especially like to thank Uncle Bruce, Tom Atkinson, George Lucas, the makers of Pepsi-Cola, and the people at Bangaway Engineering for helping me preserve my sanity and survive this experience.

Table of Contents

List of Tables

List of Figures

Part I. A Distributed Processing Control System

Chapter 1 : INTRODUCTION

ORGANIZATION

The research covered in this dissertation spans two different areas and the dissertation is divided accordingly. Part I is a description of several instrumentation developments and Part II is a description of the application of triple quadrupole mass spectrometry (TQMS) to hydrocarbon fuel analysis. Part I is made up of Chapters 2 through 5. instrumentation developments and applications. Chapters 2 thru 4 deal with the development of the hardware and software of a distributed processing system for real-time instrument control. The development of a control system for a triple quadrupole mass spectrometer using this distributed processing system is discussed in Chapter 5. Chapters 2,4 and 5 are presented in manuscript form.

Applications of triple quadrupole mass spectrometry to the screening of hydrocarbon fuels for selected species are described in Chapters 6 and 7, which form Part II of this dissertation. These chapters are also presented in manuscript form. Chapter 8 contains comments and suggestions of area for further investigation.

The user's manual for the triple quadrupole mass spectrometer control system is included in Appendix A. This document provides a more complete description of the capabilities of the control system than could have been covered in the manuscript format of Chapter 5.

INTRODUCTION TO THE RESEARCH

The overall objective of the author's research was the development and advancement of laboratory instrumentation. A related goal was to demonstrate the application of TQMS to the detection of selected species in complex mixtures.

Instrument Control System Development

The main focus of the instrumentation work was the development of a distributed processing system for real-time instrument control. This work was a collaborative project with Mr. Bruce Newcome. The results and application of this work is described in Chapters 2 thru 5. The duration, complexity, and sophistication of this project make the separate identification of the author's work and that of Mr. Newcome very difficult. On the first level, it is very simple; all of the software was written by the author, while all of the hardware was developed by Mr. Newcome. However, in the synergism generated by many a late night discussion, many of the author's suggestions were incorporated in the final hardware designs and many of Mr. Newcome's suggestions were incorporated into the software system. In other words, it was a real team effort in the best sense of that phrase. Nevertheless, consistent with our primary responsibilities, the system descriptions contained in this dissertation concentrate on the software aspects of the instrumentation developments.

Chapter 2 is an introduction to distributed processing and some of the special needs of real-time instrument control systems. Chapter

3 is a brief overview of the hardware system developed as part of the distributed processing project. In Chapter 4, the development of an integrated software package for program development and operations in a distributed processing environment are discussed. Software for laboratory instrumentation is an area often overlooked by scientists in the field. Many people fail to look at software and programming languages as tools. This is somewhat due to the fact the software is so flexible it can be bent into just about any shape needed. The suitability of a tool for a job often determines whether or not it is practical to undertake a given task.

Laboratory instrumentation has made a great step forward by moving from the strip chart recorder to the microcomputer for data acquisition. However, the microcomputer acts as only a glorified strip chart recorder if data acquisition is its only function. The next real breakthrough in laboratory instrumentation needs to be and will be in the software field. It is important when developing new tools for laboratory automation that software capabilities be developed along with improved hardware capabilities.

Fuel Analysis Applications

Chapters 6 and 7 describe the application of triple quadrupole mass spectrometry to screening hydrocarbon fuels of selected species. These species are illustrated in Figure 1.1.

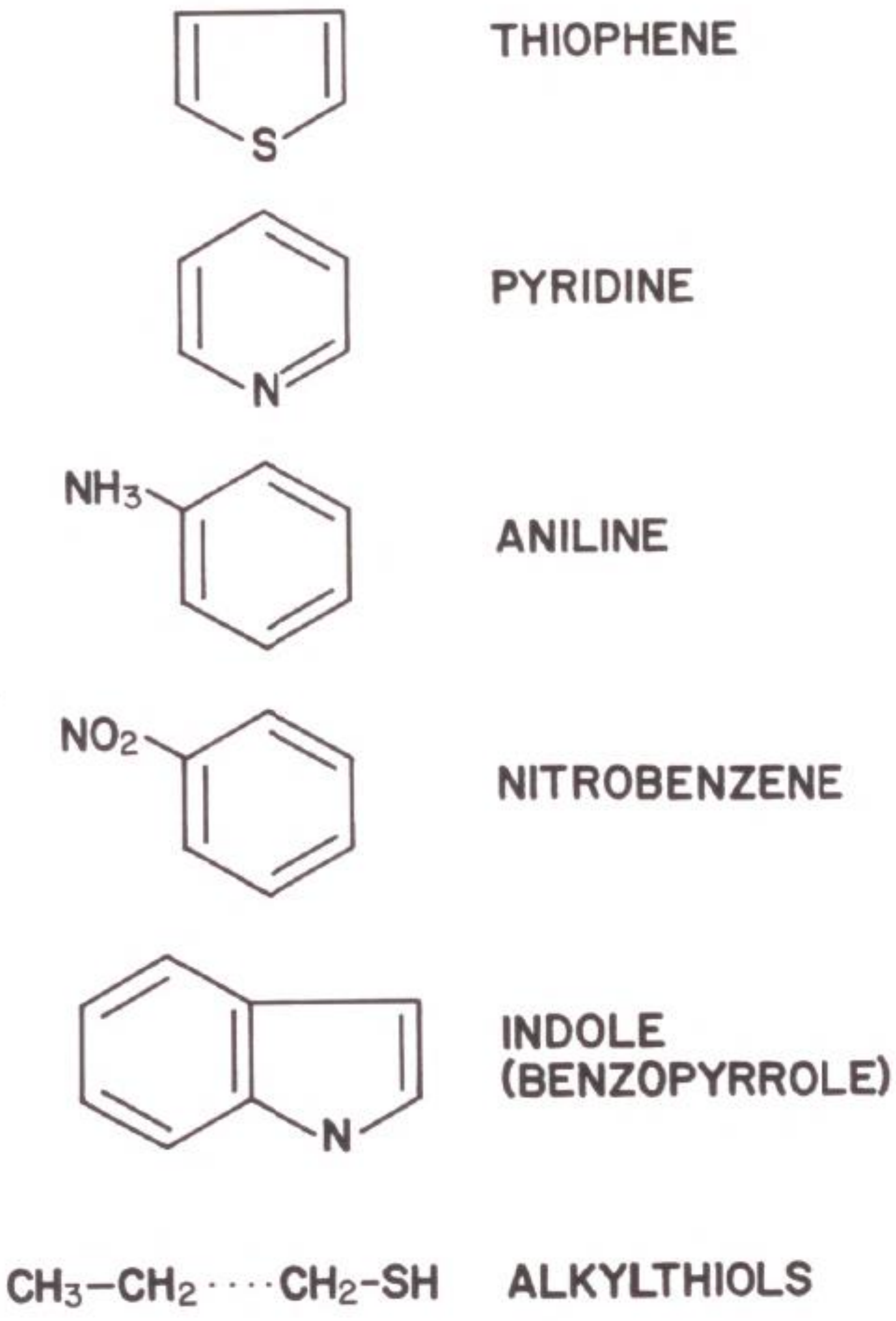


Figure 1.1 Compound Classes Studied

Heteroatom containing species are present in low-levels in most hydrocarbon based fuels. It has been demonstrated that some of these species are detrimental to the storage and thermal stabilities of the fuel. The ability to rapidly identify which species are present in a fuel sample would be a substantial aid to fuel stability studies. The separatory power of TQMS can be used to selectively detect the presence of many, if not all heteroatom-containing species in complex hydrocarbon mixtures. The paper presented in chapter 6 is a detailed study of the determination and confirmation of a screening procedure for thiophenes in jet aircraft fuels. Chapter 7 is intended as an introduction to the application of TQMS for fuels screening and describes the selection and implementation of screening procedures for several classes of compounds.

Chapter 2 : Distribution and Coordination of Tasks

A Distributed Processing System for Real-Time Instrument Control

1. Distribution and Coordination of Tasks

Carl A. Myerholtz

Bruce H. Newcome

Christie G. Enke

Department of Chemistry

Michigan State University

East Lansing, MI 48824

INTRODUCTION

In recent years advances in computer technology have greatly increased the power and sophistication of mini- and microcomputer systems, while at the same time their cost has been dramatically reduced. As a result, the use of small computer systems in the laboratory for instrument control applications has expanded greatly. Laboratory computer systems were initially used principally to perform data acquisition and preliminary data reduction functions, often acting as little more than intelligent strip chart recorders. As the performance/cost ratio increased, small computers were incorporated directly into laboratory instruments and took on increasing responsibility for control operations such as temperature programming and scan generation. In addition, the data acquisition and reduction functions increased in sophistication. The results of this evolution are recent-generation instruments which incorporate real-time control and data acquisition systems and which work interactively with the operator to optimize the data resulting from an experiment. Ideally these systems control as many instrument parameters as practical and automatically record these parameters along with the acquired data to create a complete experimental record. However, as powerful as these small processors are, the demands of high speed data acquisition and instrument control can exceed the processing capabilities of a single processor. It is important for instrument control systems to advance beyond these limitations so that they can be applied to larger and more complex instruments where the need for increasingly intelligent instrument control is extremely great.

DISTRIBUTED SYSTEMS

There are several approaches that can be taken to implement advanced control systems with higher performance than is commonly seen today. One solution to this problem is the development of specialized hardware to solve a specific problem. This approach has several drawbacks among which are the time needed to develop and test the hardware and the difficulty in adapting specially tailored hardware to new experimental demands. A second approach is to employ bigger and faster computer systems that are capable of executing control functions and processing information more rapidly than their predecessors. A problem with this approach is that the increase in computing power becomes more expensive as the level of performance increases. An alternate solution is the use of more than one processor in a system to expand the amount of computing power available to the process. This results in a distributed processing environment where each processor is a assigned a specific task or set of tasks to perform.

Systems utilizing more than one processor typically fall into one of two classes, distributed processing systems and multiprocessing systems. In a distributed processing system, the work load is spread over several processors by assigning a set of tasks to each processor. The processors are not necessarily identical; each may incorporate enhancements such as special interfaces or numeric coprocessors to enable them to perform their allotted tasks. In a multiprocessing system the work load is spread over several "equal" processors by assigning tasks to any unoccupied processor. A number of the advantages of distributed processing systems are listed in Table 2.1[1].

SCHEMES OF PARTITIONING

Partitioning tasks into separate processors takes a variety forms depending on the functions being implemented and the criteria used to separate tasks. Three of the major forms of task partitioning are: horizontal partitioning, vertical partitioning, and partitioning based on data access[2]. Two tasks can be horizontally partitioned if they can be executed without regard to order or can be executed concurrently. Vertical partitioning involves the separation of tasks that have predecessor-successor relationships. These relationships may arise from data dependency or control flow considerations. Partitions based on data access separate tasks by what data they operate on. Temporal order, control and direction of data flow are not considered. Since most tasks involved in real-time instrument control have some form of predecessor-successor relationship, efforts in this project were directed toward developing a distributed processing system utilizing vertical task partitioning. This required the development of efficient communication links between the processors.

Table 2.1 Advantages of Distributed Processing Systems

<u>Faster Execution</u>

- Parallel execution
- Less time spent in "overhead"
- Simpler addition of hardware controllers and processors

<u>Independent Task Execution</u>

- Non-interference of tasks
- Elimination of task interleaving programs
- Elimination of priority assignment programs
- Simpler task program modification

<u>Modularity of Hardware and Software</u>

- Consolidation of related tasks
- Simpler extension of instrument capability
- Simpler debugging and troubleshooting

RELATIONSHIPS AMONG TASKS

Tasks in a distributed processing system can be of several forms, terminal, loosely-coupled, or tightly-coupled. The main differences among these types of tasks are their degree of interaction with other tasks and the time scale of the interaction. It is important not to confuse the discussion of loosely and tightly coupled tasks in a distributed processing system with loosely and tightly coupled communication between processors in a multiple processor system. Loosely-coupled processor systems use shared peripherals to pass messages, while tightly-coupled systems use shared memory to pass messages[3].

Once a terminal task is started, no interaction with the other processors in the system is necessary until the task is completed. The size and complexity of a terminal task is such that only one processor is needed to perform the task. Many computer systems today have some distributed processing characteristics as the result of the use of what are often described as intelligent peripherals. Intelligent peripherals are used to off-load terminal type tasks from the main processor. Most intelligent peripherals owe their intelligence to small microprocessor systems that are used to control the peripheral. A common example of this is the dot-matrix printer which incorporates microprocessor control with a large communications buffer. The main computer and the microprocessor in the printer form a simple two-processor distributed system. The main processor can transfer data to be printed to the control microprocessor, which can then print the data without further intervention from the main processor. This is a good example of a terminal type task. The duration of the task is on the order of seconds or minutes and, once started, little or no communication with other processors in the system is necessary.

Loosely-coupled tasks involve more interaction between tasks on separate processors. These tasks are often components of a task the system is performing instead of separate independent functions as in the case of terminal tasks. Figure 2.1 illustrates a two-processor system performing loosely-coupled tasks. Processor one's task is to acquire 100 data points at some fixed sampling rate.

When 100 points have been acquired they are transferred to processor two. After the transfer is completed processor one is free to start acquiring the next set of 100 data points. During this time processor two writes the data to a disk and is ready to receive the next set of points before processor one has completed acquisition of the second set of points. Loosely-coupled tasks often require interaction between tasks on the millisecond time scale.

Tightly-coupled tasks are similar to loosely-coupled tasks; however, the subtasks each processor is performing form a smaller portion of the overall task being performed by the system. More coordination between tasks is needed and must take place on a shorter time scale. A single-processor system is compared to a two-processor system performing a set of tightly coupled tasks in Figure 2.2. This

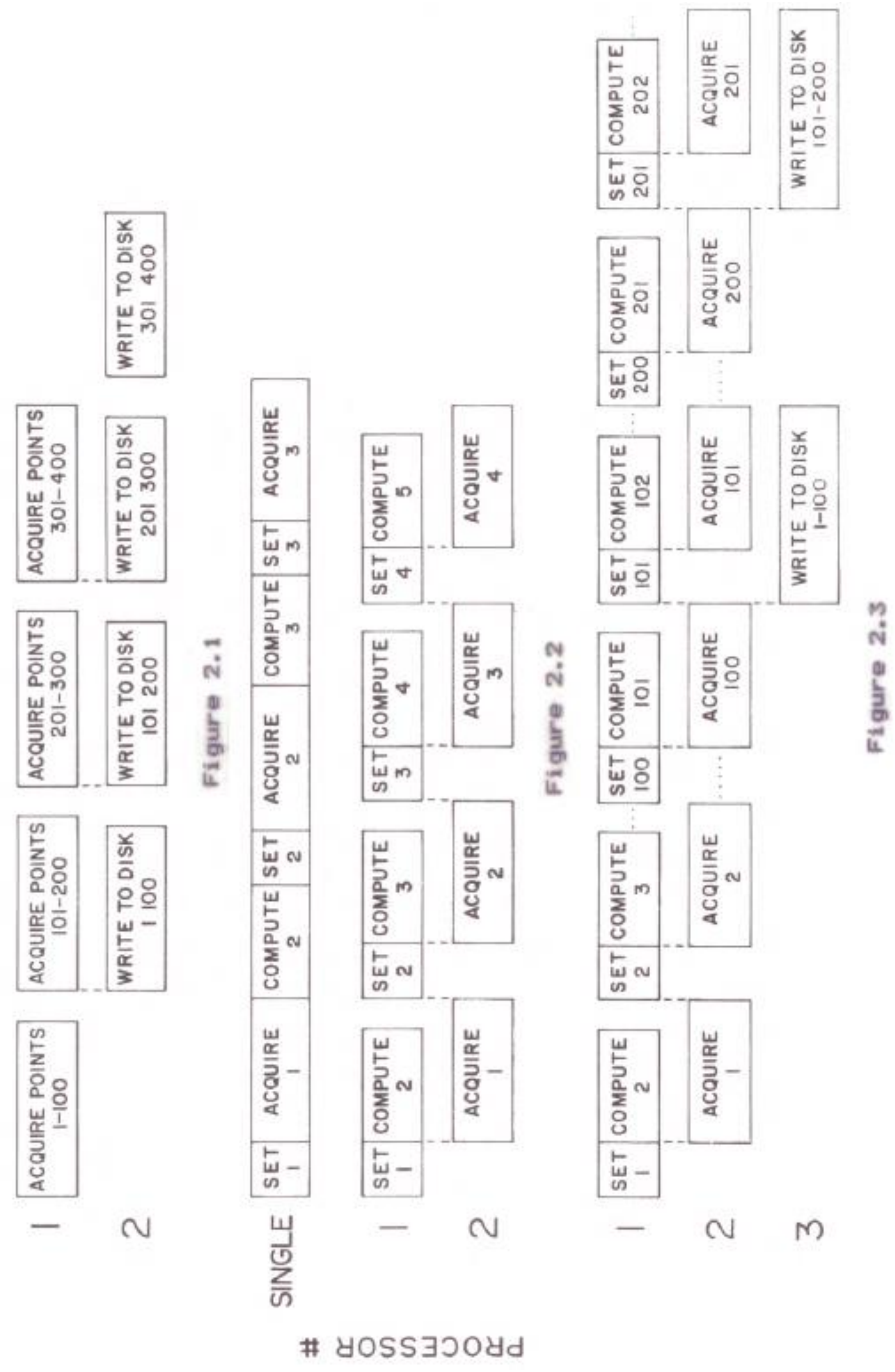


Figure 2.1 Loosely-coupled Tasks

Figure 2.2 Tightly-coupled Tasks

Figure 2.3 System with Loosely and Tightly Coupled Tasks

example involves an experiment that requires that a voltage be sent to a DAC a value be measured, a new voltage calculated, the DAC be updated, and so on. Figure 2.2a illustrates this set of tasks implemented on a single processor system. Figure 2.2b demonstrates how these tasks might be divided between two processors in a tightly-coupled system to increase the sampling throughput. Processor one is assigned the task of controlling the DAC, while processor two is dedicated to acquiring the data. When processor one has set the DAC to a new value, processor two may begin to acquire a data point. At the same time processor one may begin computing the next voltage to send to the DAC so that when acquisition of the first data point is completed a new value may be immediately send to the DAC and the whole process repeated again. Tightly-coupled tasks often require interaction among tasks on the microsecond time scale.

Distributed processing systems can be designed to handle any task type or combination of task types, the primary differences being the amount and speed of the communication needed between tasks on different processors. Figure 2.3 illustrates how, in a distributed system, three processors can interact with each other while performing both loosely and tightly coupled tasks. Processors one and two form a two-processor system, executing tightly-coupled tasks like that shown in Figure 2.2b. Processor three executes a task which stores the data on a disk. That task is loosely coupled to the operation being performed by processors one and two. The resulting system is much like the system described in Figure 2.1. Processors one and two execute tightly-coupled tasks to perform the data acquisition function that processor one in Figure 2.1 performed, while processor three performs a loosely-coupled task to write data to a disk as did processor two in Figure 2.1. This example demonstrates the powerful modularity of distributed processing systems.

ADVANTAGES OF MODULARITY

The inherent modularity of distributed processing systems offers a number of advantages, in both hardware and software, over a large, single-processor system. Among the advantages realized with a distributed processing system are a higher cost-to-performance ratio, simple expansion, simpler, more modular software, and less stringent demands on the hardware design. The higher cost-to-performance ratio comes about because to double the performance of a single processor system usually costs more than twice as much, whereas adding a second processor to the system is more of a linear addition in cost. Also, there is an inherent limit to the performance available in a single processor system, whereas in distributed processing systems the performance can be upgraded until the maximum number of processors capable of being supported is reached. The ready expandability of distributed systems is economical in both cost and more importantly time. Take a case where experimental demands change after a system is in use and a 25% increase in processing speed is needed. In a single processor system, the main processor would need to be replaced, most likely requiring new hardware interfaces and new or rewritten software for the new processor. A distributed processing system could achieve the needed increase in performance with the addition of another processor and with changes only in the routines affected by the additional processor. By spreading the processing demands over several computers, the demands on the hardware system become less strenuous. Five processors operating with 1 MHz bus bandwidths are easier to implement and more noise immune than a single processor system with a 5 MHz bus bandwidth.

Distributed processing systems benefit from the separation of tasks to different processors. In a single processor system which is performing multiple tasks, as the number of tasks increases more and more of the processor's time is spent changing from on task to another[4]. A distributed processing system with tasks running on separate processors does not suffer from this task switching overhead. Since multiple tasks can be split into separate processors, programming complex applications can become much easier. A programmer can create separate routines for each function on different processors without programming multiple tasks into a single loop or concern about interrupt latency and throughput. This independency of tasks can greatly ease the adaptation of the system to new experimental demands. Routines in separate processors can often be more readily modified with less interference to other tasks than can routines in a single processor system. The systems described in Figure 2.2 can be examples of this. If a new, slightly longer and more complex algorithm was needed to calculate the new DAC values, the system in Figure 2.2a would suffer a reduction in the overall sampling rate and additional difficulties in programming could be posed if register use were critical. The acquisition routine might have to be recoded to make up for the additional computational time of the DAC routine. This would not be the case in the system in Figure 2.2b; the DAC routines could be modified without having to disturb the acquisition code at all.

INTERPROCESSOR COMMUNICATION REQUIREMENTS

Once a system has evolved into a distributed processing system, it becomes necessary to define how the different processors in the system will communicate with one another. The various modes of communication needed for real-time instrument control are summarized in Table 2.2. The transfer of blocks of information between processors is necessary so that programs may be loaded into the processors and data sets may be handled efficiently. A mechanism is needed to instruct the various processors in the order in which to execute their designated tasks. The ability to queue up a series of tasks for execution by a given processor is a desirable feature. The assignment of succeeding tasks to a processor should not interfere with the task currently being executed. Parameter passing involves the transfer of small amounts of information between processors. Often these are parameters that modify the execution of tasks assigned to a processor, specifying such information as number of iterations or scanning speed. A method for passing task status information between processors is needed so that the execution of tasks in separate processors can be coordinated. As in the case of task assignment, it is preferable that

Table 2.2 Interprocessor Communication Modes

1. Block data transfer
2. Task assignment
3. Parameter transfer
4. Task coordination

the coordination of processors does not interfere with the execution of tasks.

These different modes of communication between processors could be supported by the use of shared memory. However, there are advantages in both speed and non-interference if the different modes are supported by dedicated hardware. Hardware support for the various communication modes can be implemented so that none of the communication modes interfere with tasks being executed by a processor. However, in the case of block data transfer interference is not a major consideration. This is because these types of transfers take place at system startup when code is loaded into a processor or when a processor requests that a block of data be transferred from its memory and is therefore not executing a time-critical task. Allowing the block data transfer to interfere with task execution on a processor allows the hardware to suspend operation of the second processor during the transfer. This performance concession can greatly reduce the complexity of the hardware needed to support block data transfers.

SOFTWARE CONSIDERATIONS

An important consideration of the software system is its suitability for instrument control applications. An excellent discussion of the features needed in a language for laboratory use can be found in ref.[5] Real-time instrument control requires the ability to control many specialized interfaces in a timely manner. The ease of interaction with specialized hardware and the speed of program operation are important features in software for control applications. In the research laboratory experiment design is constantly evolving. The software for a control system should make it easy to adapt a system to changing experimental needs. Utilization of a high-level language tailored for instrument control applications can facilitate programming of the system by novice users. The use of a consistent high-level language across all processors can reduce the complexity of programming in the distributed programming environment.

Although in some cases, additional processors can readily be added to a system by merely plugging them into the system backplane, developing the software to run a distributed processor system is not as simple. In the first place there are almost no readily available operating systems for small distributed processing systems. Secondly few languages have been developed for programming multiple processor systems. In many distributed processing systems, most of the processors operate more like dedicated intelligent peripherals, usually running programs developed in assembly language that are stored in programmable-read-only memory (PROM). This approach is not very acceptable in a research laboratory where the demands on instrument control systems are constantly changing. The time consuming cycle of assembly language coding, PROM programming, and code testing inhibits the flexibility and adaptability of the system.

The need to be able to readily modify the software running in the various processors gives rise to a number of design considerations for the software system used to run a distributed processing control system in a research laboratory. Among these are the use of processors that mainly run software loaded into random-access memory (read/write memory or RAM) so that the software can be readily modified or replaced. The use of a high-level language to program all of the processors in a system makes programming faster, easier and more efficient.

The system should allow for local program development so that new routines can be developed and tested interactively with the instrument, possibly even during an experimental session. This precludes the use of cross compilers and assemblers that run on larger computer systems. Here only the resulting object code is transferred to the control system, and all program development must be performed offline. The use of cross compilers and assemblers can be particularly frustrating during the testing and debugging stage of software development. As each problem is corrected the programmer must go to a different machine and make the necessary changes, recompile the program, and transfer it to the control system again. This results in the need for an operating system and programming language small enough to reside on the control system but powerful and flexible enough to allow for rapid program development on several processors as well.

SYSTEM DESIGN CONSIDERATIONS

As part of an ongoing investigation of laboratory instrumentation systems, our research group embarked on the development of a distributed processing system for laboratory instrument control. The design goals of this project are summarized in Table 3. The details of this system are presented in two additional manuscripts. The first covers the design and implementation of the system hardware. The second discusses the development of a software system to utilize the power of the distributed processing hardware.

Utilization of distributed processing techniques can provide the scientist with the needed computing power to develop the next generation of intelligent instruments. These systems will include among their extensive capabilities aids to the optimization of experimental conditions, optimization of data acquisition parameters in real-time, and automatic sequencing among different experimental conditions and configurations.

Table 2.3 Summary of Distributed Processing System Design Goals

1. Implementation of Interprocessor Communication Modes
2. Non-interference with Timely Events
3. No resident Monitor Required in Auxiliary Processors
4. Modularity of Hardware and Software Systems
5. Easy Transition from a Single Processor to a Distributed Processing System
6. Readily Programmable by Non-expert Users
7. Readily Adaptable to Changing Experimental Needs
8. Efficient Software Environment for Real-time Instrument Control

REFERENCES

[1]Enke, C.G., Proc. 28th IUPAC Conference, Vancouver, BC (1981)

[2]J.T. Lawson, M.P. Mariani, Proceedings of the IEEE, 358 (1978)

[3]B.C. Searle, D.E. Freberg, Computer, 22. Oct. 1975

[4]Linden, I. Wilson, Microprocessors and Microsystems **4**, 211

[5] Dessy, R.E., Anal. Chem. 55(6), 650A, (1983)

Chapter 3 : INTERPROCESSOR HARDWARE OVERVIEW

Although the hardware for this system has been discussed in detail elsewhere[6], a brief review will aid in setting the background for the software description to follow. The hardware uses modules that were created as part of a project to develop a flexible microcomputer system for the research laboratory environment[7]. Each module implements an individual function; the modules include central processing unit (CPU), memory, parallel interface, analog-to-digital converter, etc. Several of these modules may be mounted on a motherboard that connects them to a common data and address bus. Multiple motherboards may be plugged into a common backplane to configure a computer system with the desired capabilities. Over the three year life of this project, more than twenty function modules have been developed including two CPU modules, one utilizing an Intel 8085 microprocessor and the other an Intel 8088 microprocessor. Five of the modules developed provide interfaces to an interprocessor communications bus.

When this hardware is configured in a distributed processing system through the interprocessor bus, it can support up to eight CPUs running in parallel. Each processor in the system is assigned a processor ID number between 0 and 7. The master processor is given an ID number of 0, while any slave processors in the system are numbered consecutively starting at 1. The prototype distributed processor system which operates a triple quadrupole mass spectrometer consists of one master processor and three independent slave processors. In this system all of the processors utilize the 8088 CPU module. The master processor is interfaced to a CRT terminal and a printer for user interaction, and an 8-megabyte Winchester disk drive for program and data storage. Each slave processor is assigned a specific set of functions and has only the instrument interfaces needed to perform those specific functions. The slave processors are dedicated to specific tasks and are not general purpose or interchangeable processing units as is the case in some distributed processing systems optimized for other purposes.

In the distributed processor environment, three types of interprocessor communication are supported by specialized interface modules. The interconnection of the processors by these communication links is illustrated in Figure 3.1. Three interprocessor communication modules are mounted on a motherboard to form links between a processor's local data and address bus and an interprocessor communications bus. Each processor in the system has one interprocessor motherboard plugged into its backplane. One of the three interprocessor modules supports the direct memory transfer (DMT) communication mode. The DMT mode allows the master processor to transfer data between any two processors in the system. The master processor can perform a transfer between the memory on any two slaves or between memory on the master and any slave. This is accomplished with bus switching hardware that puts any slave processor involved in a transfer on hold and connects its local data and address bus to the interprocessor communications bus. When the transfer is completed, the slave processor's bus is released, and program execution in the slave is resumed.

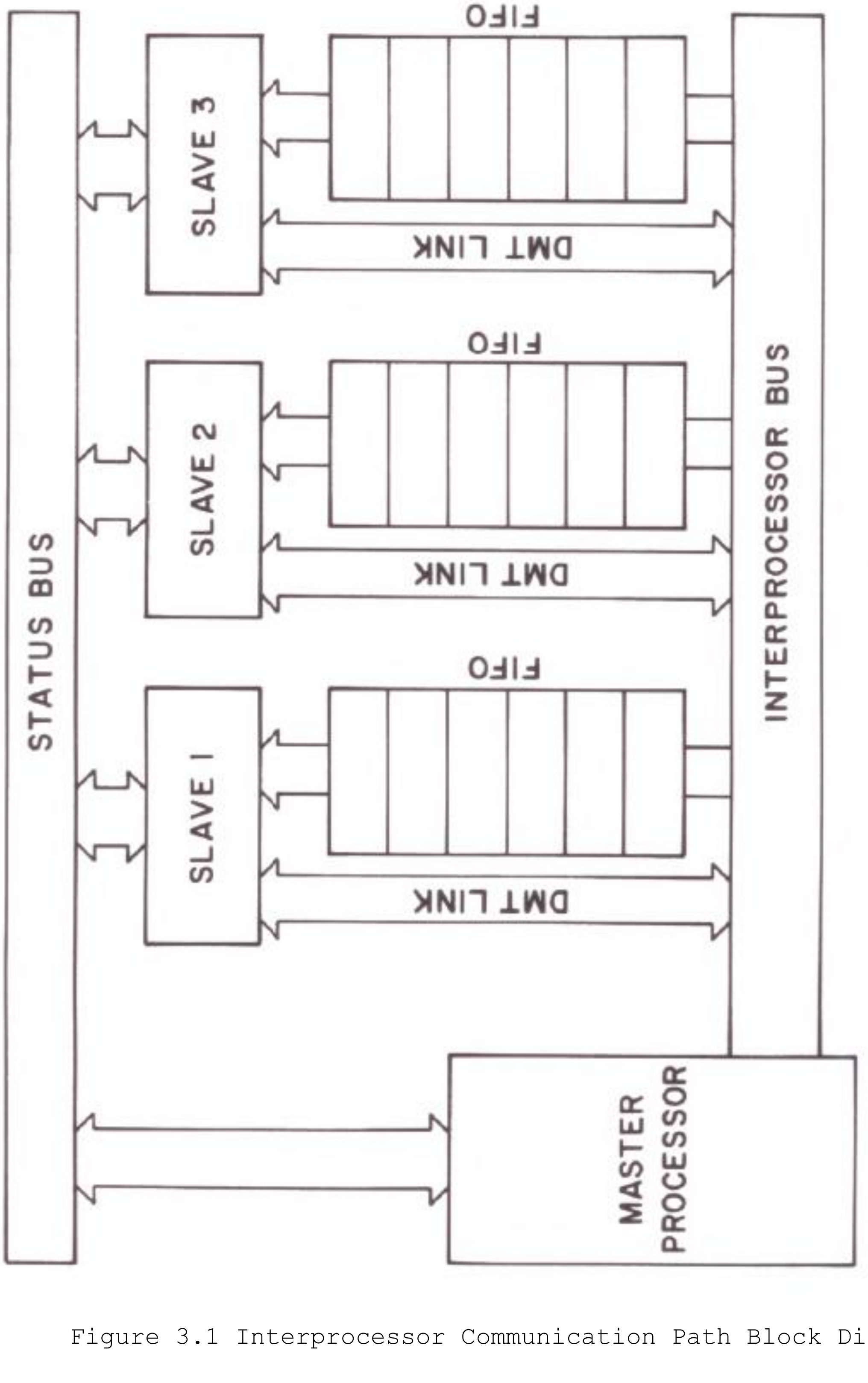


Figure 3.1 Interprocessor Communication Path Block Diagram

A second interprocessor module supports a method of communication called the command transfer mode. The hardware on this module consists of first-in-first-out (FIFO) buffers for each slave processor. Each FIFO is 24 bits wide and 32 elements deep and is referred to as a command buffer. The master processor can write data into any slave's FIFO which then can be read out and interpreted by the slave. The low order 20 bits written by the master are stored in the FIFO as data. The four high order bits are hardware control signals that can be used to immediately reset, hold or interrupt the slave processor.

The third interprocessor module provides a communication mode called status transfer. The hardware on this module involves 16 bytes of dual-port memory. Each processor in the system (maximum of eight) is assigned two bytes of status information: one hardware status byte and one software status byte. These are maintained up to date in each computer's dual-port memory. The hardware status byte for each processor includes such information on the processor as command buffer full or empty, and processor halted. The software status byte is used to synchronize tasks between different processors by using program-driven flags or codes to indicate program status. This hardware and software information on each processor is updated every 4 usec in each processor's interface through a special status bus, which is part of the interprocessor communications bus.

REFERENCES

[6] B.H. Newcome, C.G. Enke, Rev. Sci. Inst. Submitted for publication 1984.

[7] B.H. Newcome, C.G. Enke, Rev. Sci. Inst. Submitted for publication 1984.

Chapter 4 : An Integrated Software System

A Distributed Processing System for Real-Time Instrument Control

3. An Integrated Software System

Carl A. Myerholtz

Christie G. Enke

Department of Chemistry

Michigan State University

East Lansing, MI 48824

The hardware for a distributed processing system for laboratory instrument control has been described earlier[8]. This hardware allows control systems to be assembled easily and provides communication and control pathways among several processors in a tightly-coupled distributed processing system. In order to utilize these capabilities effectively, a software environment had to be created which could effectively develop real-time research instrument control applications.

SOFTWARE SELECTION CONSIDERATIONS

The software developed to operate with this hardware needed to be as well suited for operation in a single processor environment as in a distributed processing environment. Since the hardware provides an easy path for upgrading to a distributed processing system, the software selected also needed to be able to make the transition easily. This constrained our software development to systems that can operate effectively in a small single processor environment as well as a larger multiple processor distributed environment. It is also important that programming language features and structure should be such that transferring a task to a separate processor would not require extensive recoding of existing programs.

In addition to its suitability for operation in a distributed processing system, the software system should be adaptable to instrument control applications. Real-time instrument control often involves the need to control many specialized interfaces very rapidly.

The ability to interact directly with these specialized interfaces and the speed of execution of programs are important considerations. In a research laboratory, experiment design is constantly evolving so that it is important that the control software for an instrument be able to be reconfigured easily to meet new experimental demands.

Another goal for our software system was that it provides the user with the ability to develop and test programs destined for the slave processors, load programs into the slaves, and provide methods of accessing the slave processors for debugging purposes. The software environments of the master and slave processors should be as similar as possible, so that programs may be tested on the master processor, where the user can readily interact with the program, prior to being loaded into a slave processor.

In a few cases, groups have endeavored to develop from scratch, languages and operating systems designed specifically for distributed processing applications[9],[10]. This approach was viewed as too costly and time consuming an undertaking for a scientific laboratory. Instead, only currently available languages and operating systems were considered and evaluated for their suitability and adaptability to the distributed processing environment. What was needed was a language that would operate effectively on small single processor systems, but could easily be enhanced to operate in a distributed processing system.

SELECTION OF THE FORTH LANGUAGE SYSTEM

The high-level programming language FORTH was chosen as the basis for this system[11],[12],[13]. Although FORTH is not a widely used language, its popularity has increased steadily since its inception. Originally developed in 1968 to control a radio telescope at the National Radio Astronomy Observatory, FORTH is one of the few languages developed for small computer systems with control applications in mind[14]. Implementations of FORTH are now available for nearly all popular microprocessor and minicomputer systems. Some of the implementations include such advanced features as multi-tasking[15] and floating point processor support[16].

A FORTH program, or "word", consists of a series of previously defined words. These "words" are stored with their definitions in a "dictionary"; definitions for new words are merely lists of previously defined words. When executed, each word (program) performs a function that can be as simple as addition or as complex as plotting an entire graph of acquired data. Even a novice user can write programs (new words) by merely concatenating existing high-level words. All previously defined words, from assembly language to the highest level can be used in any given program. When executed directly, each word acts like a separate program. When used as a command in another program, it is like a subroutine. A typical FORTH system has several hundred words in the "vocabulary" of the core system. The modularity of FORTH words is similar to that of distributed processing systems.

Simple modules (or words) can be readily combined into complex modules that can be combined into even more complex modules.

An unusual feature of FORTH is the use of a push-down stack to pass parameters between words when they are acting as subroutines in a larger program. A push-down stack acts as a Last-In-First-Out (LIFO) buffer; items placed on the stack are removed in reverse order. FORTH uses this parameter stack to pass values between both high level and assembly language routines. Thus, the interface to an assembly language routine is no different from that to a high level FORTH routine.

It's extensibility, the ability to add new commands to the language, is one of the most powerful features of FORTH. It is generally recognized that high-level languages aid in program development. But, in order to be of significant aid to a programmer, a language must contain those high-level functions that are needed for the given application. FORTRAN is an example of a high-level language well suited for numeric processing. FORTRAN, which stands for formula translation, was designed to make computational programming easier. Most control applications do not need a great deal of computation. Instead, they need many specialized input/output (I/O) functions and the ability to respond rapidly to real-time events. When FORTRAN is used to program control applications, the special I/O functions often end up being coded as assembly language subroutines. The result is that most of the actual control software is not written in a high-level language. When developing control application software, as in all projects, it is important for the tool to fit the job. Most of the major programming languages available today were originally designed to run on large computer systems and solve numeric processing or data management problems. These more traditional programming tools can be applied to control systems, but this is like trying to use a pickaxe to drive a nail, neither the scale nor the function are quite right for the job.

FORTH can be the basis for the development of high-level languages which are specific for each application. In FORTH programming, words build on each other, each new word becoming a higher and higher level of operation. The end result is a high-level language that is specific to the application at hand. As programs for a given level of operation are achieved, these become the "high-level" words for the next level of program capability. This makes programming that application on any level very efficient. Figure 4.1 is a listing of a short FORTH program that illustrates how this comes about. The application to be controlled is a stepper motor that advances 0.25 degrees each time memory location 10 is accessed. Line 0 is just a comment line to indicate these routines are for the stepper motor. Line 2 defines a new word STEP that fetches (@) the value from location 10 and then discards (DROP) it. This causes the stepper motor to advance 0.25 degrees. The word STEPS, defined on line 4, pulses the motor n times, where n is the number on top-of-the-stack (TOS). This is done by using the new word STEP and the standard FORTH words DO and LOOP which create a loop that goes from 0 to n, thus repeating STEP n times. The final word, DEGREES, advances the stepper motor a

```
( STEPPER MOTOR ROUTINES )

: STEP  10 @ DROP ;

: STEPS  0 DO  STEP  LOOP ;

: DEGREES  4 *  STEPS ;

: DEMO 360 0 DO ACQUIRE CR . 10 DEGREES 10 +LOOP ;
```

Figure 4.1 FORTH programming examples illustrating how words build into more and more powerful commands.

given number of degrees. It takes the number on TOS as the number of degrees to move and multiplies this by four leaving the result on TOS. This provides the number of 0.25 degree steps desired which the word STEPS then uses to advance the stepper motor. This new word DEGREES can be used to create a simple data acquisition experiment that acquires and displays a data point for every 10 degrees of stepper motor rotation. Line 8 in Figure 4.1 defines the word DEMO that performs this experiment. A loop is created with the DO and +LOOP words that will go from 0 to 360 by steps of 10. The user written word ACQUIRE acquires one data point and returns it on top of the stack. This value is displayed on a new line with the "CR" and "." commands.

FORTH is a very compact and powerful language system. A basic FORTH system which contains the FORTH compiler, an editor, an assembler, disk accessing functions and terminal I/O routines typically occupies only 8 Kbytes (K=1024) of processor memory. For applications not requiring all of the above features, the FORTH kernel can be reduced to under 1 Kbyte. The small memory requirement of this language system makes it ideal for small microcomputer systems such as dedicated slave processors.

In a standard configuration, the FORTH language system incorporates a high-level language, editor, assembler, and operating system functions in an integrated package. It can be a stand-alone system that does not require the use of other software packages or an additional operating system 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 by a program while it is running; FORTH makes no distinction between the two operations. The major functions of high-level language, editor, and assembler all follow the same rules of form and syntax which makes the system internally consistent and easier to use than non-integrated systems.

FORTH allows the user to access memory and input/output (I/O) ports on peripheral devices directly. There is no operating system standing guard (or interfering), between the user and the system hardware. Many I/O tasks can be taken care of in high-level FORTH. In addition, memory and I/O ports can be accessed readily from a terminal, greatly simplifying the testing and debugging of hardware interfaces.

Typically, programs written in FORTH execute 30-50% as fast as their assembly language counterparts. This is much faster than conventional interpretive languages such as BASIC that execute programs hundreds of times slower than assembly language. The speed of FORTH, its ability to interact directly with the processor and its peripheral devices allow much more efficient use of machine resources than many compiler language systems such as FORTRAN and PASCAL.

Availability of source code for the language was also an important consideration since some modification would be necessary to adapt the language to support interprocessor and intertask communication. Source code for FORTH is readily available for most popular microprocessors from a number of vendors. Two of the sources are the Forth Interest Group (FIG) and FORTH Inc. of Hermosa Beach, CA. The Forth Interest Group offers source code for FORTH implementations on a variety of processors for a nominal cost. However, the source code available from FIG has a number of drawbacks for this application. The principal drawback of the FIG implementations is that the basic FORTH system is written in conventional assembly language and requires a separate assembler to generate a FORTH system. Unlike the FIG implementation, the source code for the polyFORTH system from FORTH Inc. was written in FORTH and includes a target compiler that allows one FORTH system to generate a new FORTH system. An additional feature of the polyFORTH system that make it attractive is support of multitasking. Although the polyFORTH system is quite a bit more expensive than the FIG implementations, the performance, target compiler, training support, and other advanced features make it well worth the expense in this application.

ADAPTING FORTH TO A DISTRIBUTED ENVIRONMENT

The basic FORTH system which normally operates in a single processor environment had to be expanded to handle the interprocessor tasks of block data transfer, parameter passing, and task assignment. This involved additions to the FORTH system running on the master processor and development of modified FORTH system that would operate within the slave processors. It was also necessary to develop a method of compiling the modified FORTH system and loading it into the slave processors.

Compilation of Slave Processor Software

The first step in adapting FORTH to a distributed processor environment involved modifications to the polyFORTH target compiler that is executed on the master processor. Several interprocessor memory access words were developed that utilize the direct memory transfer (DMT) hardware to transfer data between the memory of different processors in the system. These words, which are described in Table 4.1, form the basis for the transfer of large blocks of information between processors. Their use follows the sequence: select a slave processor with the #SLAVE command, then transfer data between the master's parameter stack and the selected slave's memory. Two slave processor control words, RESET and HLD, were developed. These words utilize the control signals provided by the command transfer hardware to reset a selected processor or put a processor into a HOLD state, preventing any program execution. These interprocessor control and access words were used to create a modified version of the target compiler that directly downloads the code into the desired slave processor instead of writing the compiled code to the disk.

There are several advantages to having a compiler that directly downloads code into the target processor. One advantage is that the compiler can initialize variables and tables in the slave processor during the compilation process. This eliminates the need for special

Table 4.1 Interprocessor Memory Access Words

SLAVE - select the slave whose number is on top of the stack (TOS) for access by the interprocessor memory operations.

I@ - Fetches a 16-bit value from the address specified by the master's TOS in the slave selected by #SLAVE to the master's stack.

! - Stores the second value on the master's stack at the address on top of the stack in the slave previously selected with #SLAVE. This operation transfers a 16-bit value.

IC@ - Fetches an 8-bit value from the address specified by the master's TOS in the slave selected by #SLAVE to the master's stack. I! - Stores the low-order byte of second value on the master's stack at the address on top of the stack in the slave previously selected with #SLAVE.

SLAVE - Selects the source and destination processors for the IMOVE function. The TOS value is the destination processor number and the second stack value is the source processor number.

IMOVE - Transfers n bytes from a source address in the source processor to a destination address in the destination processor selected by the >SLAVE command. Usage: source add, dest add, number bytes IMOVE.

slave initialization routines which take up memory and are only used once. The devices interfaced to the slaves are all memory-mapped; thus, the compiler can also be used to initialize all of them appropriately. Since the compiler interprets code from the disk, the entire specialized initialization procedure may reside on the disk rather than occupy any system memory.

For the master processor to instruct a slave processor to execute a selected routine, the master processor should have some knowledge of what routines are available in a given slave. To accomplish this, a word was added to the compiler which records information about selected slave commands into a Slave Command Access Table (SCAT). This new word is called COMMAND and is a FORTH "immediate" word. This type of word is executed at compilation time instead of being compiled and is thus useful in adding new functions to the compiler. When it is executed, COMMAND records into the slave's SCAT the first three characters of the name, the length of the name, and the code field address of the last word compiled into the slave. This information is later used to direct the execution of a task in a slave processor.

DIRECTING SLAVE PROCESSORS

The source code of the basic FORTH system to be used in a slave processor had to be modified so that the slave processor can receive instructions from the command FIFOs. The code that normally interprets commands and data from a terminal was replaced with new code which performs the same function using the command FIFOs. The result is a slave interpreter that operates in the following manner: using the command buffer hardware, the slave monitors the condition of its command FIFO. When the FIFO becomes not empty, the high 8 bits of the 24-bit value in the FIFO are read. A value of one indicates that an immediate control operation using the high four bits was executed. In this case the lower 16 bits are discarded. If the value is two, this indicates that the lower 16 bits contain the code field address of a FORTH word to execute. Control is then transferred to the routine at this address. When execution is completed, control will return to the command FIFO interpreter. A value of three in the high byte of the command FIFO signals that the lower 16 bits are data. This value is then transferred to the slave's parameter stack. This method of moving numeric data to the stack and initiating the execution of a word appears to the rest of the FORTH system to be identical to interpreting text from a terminal. This new command interpreter was installed in such a manner as to preserve the multitasking capabilities of the polyFORTH system. Thus, a slave can be running a background task such as an oscilloscope display and still be able to act on new commands sent to it through the command buffers.

The slave processors do not possess terminals or disk drives; thus, the support code for these devices is not normally down-loaded into a slave. This reduces the basic FORTH slave system to approximately 2 Kbytes. However, for debugging purposes, terminal support software can be loaded into a slave processor to allow a programmer to interact with it directly. Normally the user can directly interact with only the master processor. Since the slave interpreter operation mimics normal operation from a terminal, a programmer may test a routine on the master and then, when it is down-loaded into a slave, he or she can be confident it will behave in the same manner.

MASTER PROCESSOR FORTH EXTENSIONS

To allow programs running on the master to pass data and commands to the various slave processors, several new words were defined for the master processor FORTH system. The first of these follows the form nPUSH, which pushes the value at the top of the master processor's stack to the top of the nth slave processor's parameter stack. This becomes the primary method of parameter passing from the master to the slave processors. The second type of word developed was designed to ease access of variables and arrays in a slave processor from the master processor. It follows the form nLABEL and is used to define a new word on the master processor that when executed selects the appropriate slave with the #SLAVE command and leaves the address of a variable in the slave on the master's stack. The execution of words defined by this command prepares the master processor for use of one of the interprocessor memory access words described in Table 4.1. The third major type of word added follows the form SLn, and allows commands to be passed to a slave in the form of addresses of FORTH words to execute. When executed, a word of this form looks up the code field address for the word that follows it in the Slave Command Access Table for slave n. If the look-up is successful, the address is transmitted to slave n's FIFO as a command to be executed. The SLn and nPUSH commands use the status hardware to determine if a slaves FIFO is full or not. If the FIFO is full the command passes control to the next task in the multitasking loop. When control is returned to the command it checks the FIFO status again. this process is repeated until data can be transferred to the slave's FIFO.

A brief example of how some of these commands are used is presented in Figure 4.2. In the code for slave one the word SQUARE is defined. It squares the number on top of its stack. After the definition of SQUARE the word COMMAND appears that causes the compiler to make an entry for SQUARE in the SCAT for slave one. Similarly the code for slave two contains the definition of the word CUBE. On the master processor the word POWERS is defined and it behaves as follows: first it duplicates the number on the top of the stack, then it transfers that number to the stack of slave one and to the stack of slave two with the 1PUSH and 2PUSH commands. Then the SL1 command is used to direct slave one to execute the SQUARE function while the SL2 command directs slave two to execute the CUBE function.

```
SLAVE 1:
     : SQUARE  DUP * ;  COMMAND
SLAVE 2:
     : CUBE  DUP DUP * * ;   COMMAND
MASTER:
     : POWERS DUP 1PUSH 2PUSH  SL1 SQUARE  SL2 CUBE ;
```

Figure 4.2 Sample Multiprocessor Program

RESULTS AND CONCLUSIONS

Three of the major interprocessor tasks, block data transfer, parameter passing, and task assignment, can easily be accomplished by the use of the new words added to the master processor's FORTH vocabulary. Tables 4.1 and 4.2 summarize the new FORTH commands that a user needs to learn to develop programs to run in a distributed environment. Block data transfers can be readily programmed utilizing the LABEL commands and the interprocessor memory access words described in Table 4.1. The passing of parameters from the master to the slaves is accomplished by the use of the PUSH commands. Assignment of tasks to the slave processors is made simple by the use of the SL commands and the ability to refer to the slave tasks by name.

The use of this type of software system is not limited to the dedicated hardware developed in our laboratory. This software approach could be implemented on any multiprocessor system with shared memory. It would require the development of additional software to emulate the various communications modes implemented in hardware. The specialized communications hardware developed as part of this project offloads some of the burden from the software and offers some time savings, which, when trying to accomplish a great deal of real time instrument control, can be vital.

Table 4.2 User Extensions to FORTH and the Target for the master processor

COMMAND - A directive to the target compiler to record information of the last word compiled into the slave command access table.

1PUSH,2PUSH,3PUSH - Transfer the TOS value from the master's parameter stack to the specified slave's parameter stack.

SL1,SL2,SL3 - Direct the specified slave to execute the word whose name follows the SLn command. Example: SL1 + would cause slave one to perform an addition.

1LABEL,2LABEL,3LABEL - Usage: 1LABEL slave-name master-name. Creates a word in the master's dictionary (master-name) that when executed selects the appropriate slave number for interprocessor memory access and leaves the address of the parameter field of slave-name on the master's stack.

HLD - Causes the slave processor whose number is on TOS to be put in a hold state, preventing program execution.

RESET - Performs a hardware reset of the slave processor whose number is on TOS.

The primary goal of this effort was to develop a programming environment for a distributed processing system that was simple and easy to use. The use of FORTH for program development on both the master and the slave processors makes the system internally consistent and easy to use once the fundamentals of FORTH are mastered. Although a moderate amount of sophisticated systems level programming was needed to develop this system, it is simple and straight forward to use. This system has been used to implement a four-processor system to control a triple quadrupole mass spectrometer. Use of this system demonstrates: the speed of programming, use by novices to design automated experiments, speed of execution and gains of parallel processing, completely modular software that is non-interactive with parallel tasks except where it should be.

REFERENCES

[8] B.H. Newcome, C.G. Enke, Rev. Sci. Inst.

[9] R. Taylor, P. Wilson, Electronics **55**, No. 24, 89, (1982)

[10] S.H. Fuller, J.K. Ousterhout, et. al. Proceedings of the IEEE **66**, No. 2, 216

[11] C.H. Moore, Astrom. Astrophys. Suppl. **15**, 497, (1974)

[12] J.S. James, Byte **5**, No. 8, 100 (1980)

[13] S.M. Hicks, Electronics, March 15, 1979, p. 114

[14] C. Moore, Byte **5**, No. 8, 76 (1980)

[15] R.E. Dessy, M.K. Starling, American Laboratory, 21, Feb. 1980

[16] Forth Inc, Hermosa Beach, Ca.

Chapter 5 : A Distributed Processing Control System

New Directions in Computer Control of Chemical Instrumentation: An Application to Triple Quadrupole Mass Spectrometry

Carl A. Myerholtz

Bruce H. Newcome

Christie G. Enke

Department of Chemistry

Michigan State University

East Lansing, MI 48824

Advances in modern computing technology have had a dramatic impact on the analytical research laboratory in recent years[17], [18],[19]. The introduction of minicomputer systems made it possible to bring computing facilities into the laboratory where the scientist could utilize them directly. Many of the early computer systems performed mainly data acquisition and data reduction functions. As the cost of laboratory computer facilities decreased while their capabilities increased, laboratory computer systems were given more responsibilities. Computers systems were interfaced more intimately to instruments in order to perform control operations such as scan generation and temperature programming in addition to data acquisition and data reduction. Advances in storage technology have allowed greater amounts of data as well as reference libraries to be stored in laboratory computer systems. These systems began to be known as "data systems", and as they were paired up with instruments, we began to see terms such as MS/DS appearing referring to the combination of a mass spectrometer and a data system."

When the functions of a conventional data system are examined, it can be seen that they can be classified into two general groups; first the real time functions which are involved in instrument control and data acquisition and secondly, the non-real time functions such as data analysis, reference library searching, archival data storage, and tabulation and presentation of results. These two groups have very different requirements in both CPU response time and types of resources needed. The control group requires a fast response time and dedicated interfaces to the instrument while the data analysis group requires large mass storage devices (e.g. disk and tape drives), graphics display devices, and facilities for hard copy generation (e.g. printers and plotters) and can tolerate a moderate CPU response time. Examining these different requirements suggests that the needs of a data system might be best met by two computers operating in a hierarchical manner[20],[21].

The real time functions can be performed by an intelligent instrument control system which can be dedicated to the particular instrument and which will handle all of the instrument control and data acquisition functions. The data analysis functions can be performed by a second system which is configured to handle these tasks in an efficient manner. This scheme allows the data analysis computer to be operated as a multi-user system which can analyze the data from more than one instrument. The cost and capabilities of expensive peripherals can then be shared by more than instrument or function. This distribution of data system functions also allows analysis of stored data and acquisition of new data simultaneously thus maximizing the use of an expensive instrument. In our laboratory we have implemented such a hierarchical system using microcomputers for the intelligent control systems and a minicomputer for the data analysis system.

Modern microprocessors are ideally suited for dedicated control systems since they offer high performance at a low cost. A further advantage of these microprocessors is that they are supported by a large variety of inexpensive LSI (large scale integration) peripherals such as serial ports, parallel ports, counter/timers, and microprocessor-compatible signal converters. The availability of these low-cost interface devices is important since it allows a much larger number of instrument parameters to be controlled at a reasonable cost. The programmable nature of many of these devices is also an advantage since it allow a custom interface to the instrument to be easily implemented from standard devices.

Data analysis systems need to be able to support large amounts of storage for both data and reference libraries. Support for languages such as FORTRAN and PASCAL for data manipulation and number crunching applications must be provided as well as multiuser, multitasking operating systems. Graphics capabilities are often added to these systems to aid in data display and interactive data analysis. Relatively expensive features such as large storage devices, floating point processors, and graphic displays can be most effectively utilized in a multiuser environment, where several users can take advantage of these capabilities at one time. Commercial minicomputer systems such as the PDP-11 series from Digital Equipment Corp.[22] with the RSX-11M operating system are well suited for such applications. Figure 5.1 illustrates the hierarchical system of computing facilities in use in our laboratory. Microprocessor based systems are used to control instruments and acquire data. The data is then passed up to a PDP-11 computer system for data reduction, display and archival storage[23]. This computer is also part of a department-wide distributed network which allows access to even a greater variety of resources if they are needed.

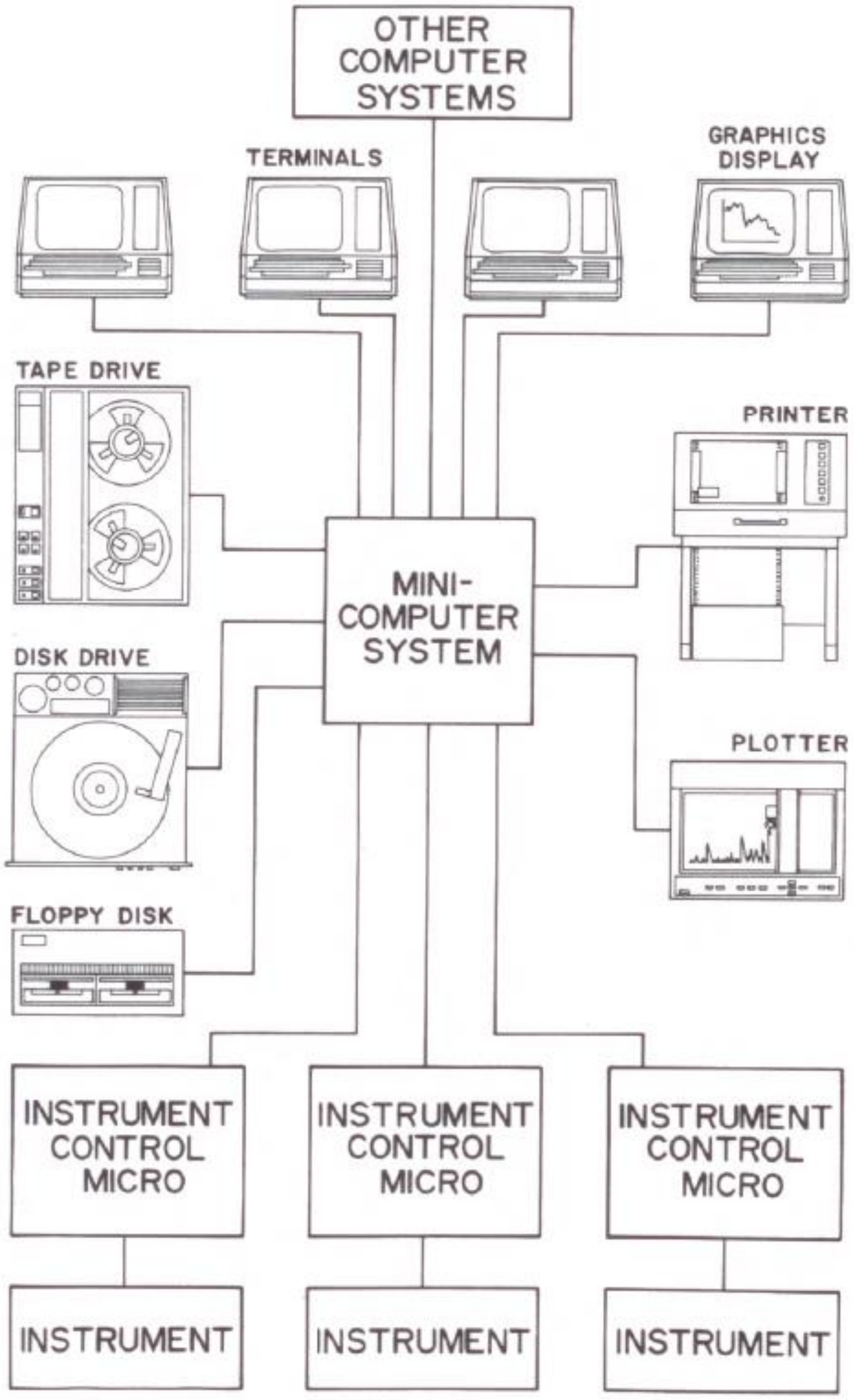


Figure 5.1 Distributed Intelligence

This separation of instrument control and data analysis tasks frees each of the computer systems to provide enhanced capability. Once freed of real-time processing constraints, the data analysis system can be expanded to provide more sophisticated data reduction and retrieval functions. Expert systems utilizing artificial intelligence concepts can be developed. The control system can also grow in sophistication and take on even more aspects of instrument control. Ideally these systems would control as many instrument parameters as possible and record these parameters along with the acquired data to create a complete experimental record. This paper will focus on the control system part of the data system and show how substantial advances in control capability and operator assistance can be achieved.

DESIRABLE CONTROL SYSTEM FEATURES

As control system tasks evolve from simple data acquisition to full instrument control, an expanded number of capabilities and features are needed to ensure that the operator and an instrument can effectively interact to produce the best possible results from the experiment. A number of these features are listed in Table 5.1. An example of the desirability of software control of the instrument over simple computer starting or monitoring of existing control hardware is seen in the generation of a scan control signal. In the case of computer-triggered scan generator, the computer can only control the start of a scan whereas software generation of the

Table 5.1 Desirable Attributes for a Control System

- Maximum flexibility through programmed software control of the instrument
- Appropriate and friendly user interface
- Aids in optimization of instrument parameters
- Adaptable to changing experimental needs
- User programmable for experimental procedures

complete scan signal allows simple linear scans and also logrithmic, square-law, step, recursive, or any other function. Efficiency might be improved by scanning at a fast rate until a signal peak appears then backing up and scanning slowly over the peak. Total generation of a scan control signal by the control system allows any scanning algorithm to be implemented in the future without any modifications to the hardware.

The existence of an appropriate and friendly user interface to the control system is very important since an awkward or difficult user interface discourages operator interaction with the system and leads to higher error rates and increased frustration. A control system may employ a number of devices to interface with the operator such as keypads, terminals, or touch panels. The choice of device depends greatly of the use of the particular system. Other factors which contribute to a friendly user interface are the use of simple, yet descriptive commands and the provision for more than one method of modifying an instrument parameter. For example, the system might provide a simple command, a video editor, and a menu, any of which could change the parameter of interest. This would allow different operators with a wide range of expertise to effectively operate the instrument. Another feature that helps operator confidence is checking for inappropriate commands and issuing descriptive warning and error messages.

An important function of the control system is assistance in or the automation of the optimization of the instrument parameters. The control system can perform this function on a number of levels depending of the needs of the experiment and the ability of the operator. The existence of tuning aids which allow the operator to interactively optimize the instrument are very helpful when the instrument is performing non-routine experiments and/or when the optimum tuning criteria are not clearly definable. When the optimum or standard performance can be clearly defined by the operator the control system could perform an automatic tuning operation. Another possible feature of an advanced control system is the dynamic optimization of the instrument during data collection using a predefined set of goals specified by the operator.

A system which allows for changing experimental needs is useful since this allows the system to expand if new ancillary equipment is added to the instrument or if the instrument itself is modified. By allowing such modifications to the control system, the frustration of using an instrument which is only partially under computer control is avoided. If the hardware of the computer system is modular, new interfaces can be easily added to the system. Equally important is that the software for the system should be open to modification so new interfaces can be integrated into the system.

A useful feature for a control system is the ability to define new experimental procedures so that routine analyses can be performed easily and with few errors. It is desirable if the new experiments can be defined by using the normal commands for the instrument instead of a separate command language since this helps minimize the amount of information that must be learned. As instruments become more complex and their control systems become more sophisticated the amount of computing power that is needed increases rapidly. As developed in the next section, one possible way to achieve this increase in computing power is to use several smaller computers connected together as a distributed processing system instead of one bigger or faster computer.

DISTRIBUTED PROCESSING SYSTEMS

Distributed processing systems offer several advantages over single processor systems[24],[25]. Some of these advantages are listed in table 5.2 and can be grouped in three classes: faster execution, independent task execution, and modularity of hardware and software[26]. Systems utilizing more than one processor typically fall into one of two classes, distributed processing systems and multiprocessing systems. In a distributed processing system the work load is spread over several processors by assigning a fixed set of tasks to each processor. The processors are not necessarily identical; each may incorporate enhancements to enable them to perform their allotted tasks. These enhancements may include special interfaces, additional memory or numeric coprocessors. This is called static load sharing, where assignment of a task to an individual processor occurs during the design phase. In a multiprocessing system the work load is spread over several "equal" processors by assigning tasks to any unoccupied

Table 5.2 Advantages of Distributed Processing Systems

Faster Execution

- Parallel execution
- Less time spent in "overhead"
- Simpler addition of hardware controllers and processors

Independent Task Execution

- Non-interference of tasks
- Elimination of task interleaving programs
- Elimination of priority assignment programs
- Simpler task program modification

Modularity of Hardware and Software

- Consolidation of related tasks
- Simpler extension of instrument capability
- Simpler debugging and troubleshooting

processors. This is called dynamic load sharing, where assignment of a task to a processor occurs during program execution.

Systems utilizing more than one processor can be loosely or tightly coupled[27]These terms refer to the method of communication the processors use to pass information. Loosely coupled systems use shared peripherials to pass messages, while tightly-coupled systems use shared memory to pass messages[28] Multiple processors systems can be interconnected in a variety of ways[29],[30],[31]. Several of these interaction schemes are shown in Figure 5.2. In our laboratory we chose to implement a distributed processing system utilizing a global bus structure as the principal means of interprocessor connection[32]. The bus structure developed can support one master processor and up to seven slave processors. The master processor is equipped with interfaces to interact with the user and to direct the operations of the slave processors. The slave processors possess the interfaces to the instrument, but cannot interact with the user directly.

PARTITIONING OF TASKS

The partitioning of tasks into separate processors takes on a variety forms depending on the function being implemented and the criteria used to separate tasks. Three of the major forms of task partitioning are horizontal partitioning, vertical partitioning and partitioning based on data access[33]. Two tasks can be horizontally partitioned if they can be executed without regard to order or can be executed concurrently. Vertical partitioning involves the separation

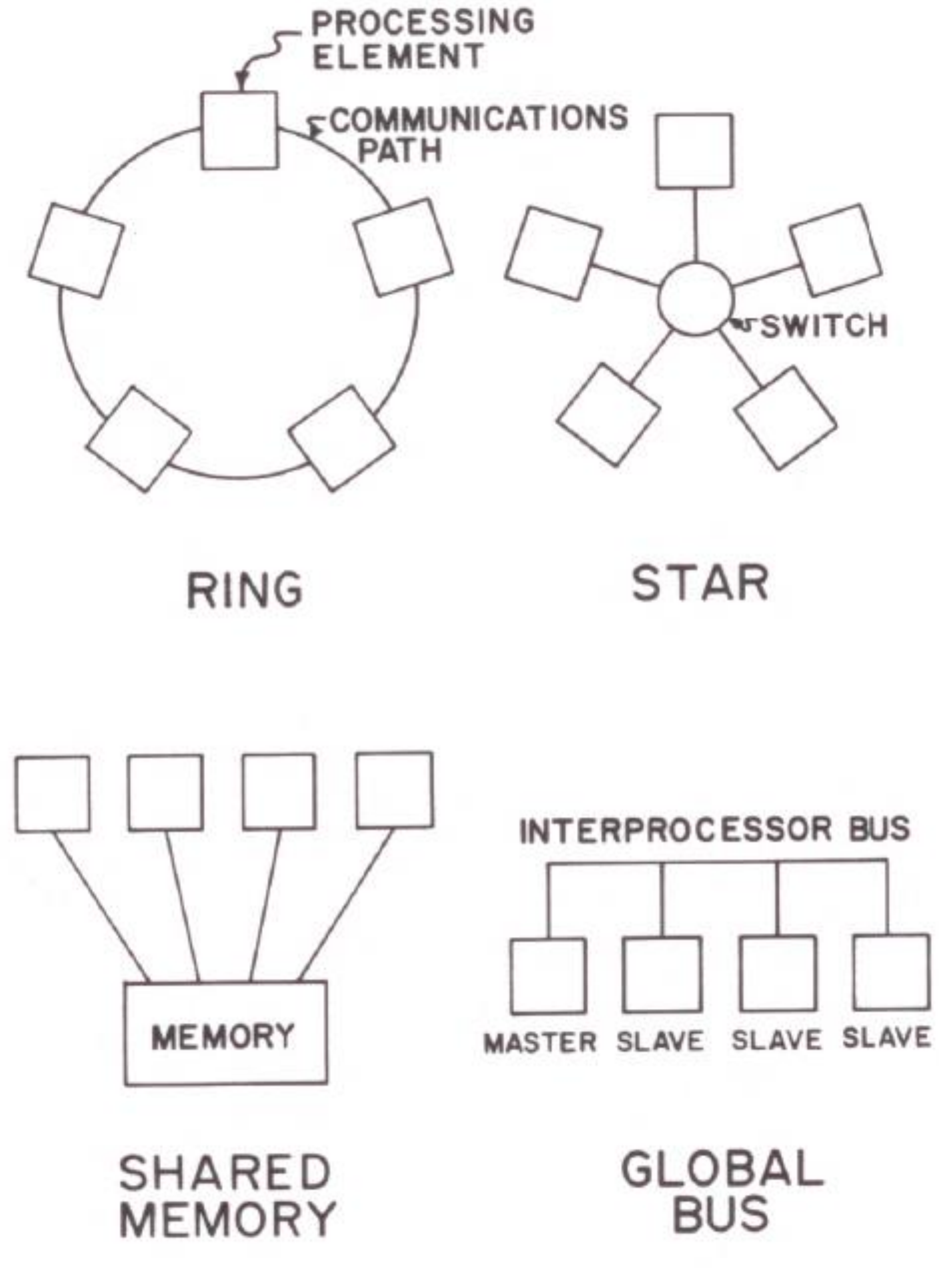


Figure 5.2 Multiprocessor Topologies

of tasks that have predecessor-successor relationships. These relationships may arise from data dependency or control flow considerations. Partitioning based on data access separates tasks by what data they operate on. Temporal order, control and direction of data flow are not considered. Since most tasks involved in real-time instrument control have some form of predecessor-successor relationship, efforts were directed toward developing a distributed processing system utilizing vertical task partitioning. This required the development of efficient communication links between the processors.

Vertical partitions in real-time systems can be classified into three major types according to transform concepts proposed by Yourdon and Constantine[34]: afferent, central, and efferent. In afferent data flow an input stream is prepared for internal processing. This involves such actions as acquiring data points, signal processing (averaging), formatting and conversions necessary to transform the data into a usable form. Efferent data flow involves the preparation of internally computed data for output transmission. Efferent transforms perform the conversions, scaling, and formatting necessary to present data to the output interface (display, DAC, printer, etc.) in the required form. Central transforms are those between the afferent and efferent data types. These central transforms are where the main data processing, such as peak finding, of the system take place. The afferent/central/efferent partitions may indicate classes of tasks that can be separated into different processors.

MICROPROCESSOR SYSTEM HARDWARE

A system of modular microcomputer components was developed to aid the construction of laboratory control systems[35]. In this system each function (CPU, memory, ADC, parallel interface, etc.) is implemented as a separate module. These modules are interconnected by mounting them on a "motherboard". Several motherboards can be connected together by plugging them into a backplane module. The structural relationships of these system elements is shown in Figure 5.3. To date, more than 20 function modules have been developed for this system, including two CPU modules (Intel 8085, 8088[36]), three types of DACs, parallel and serial interfaces, an analog multiplexer, a programmable gain amplifier, and a 12-bit ADC. Figure 5.3 contains all the essential modules for a small control system.

Several advantages arise from the use of such a system to implement the necessary computer hardware for a control application. One is that a system can be implemented with only the needed functions. There are no multifunction boards that require the space and price of the functions when only one of the functions is needed. A second advantage of a flexible modular system is the ease of expansion and upgrading. Experimental demands in the research environment are constantly changing. The ability to incorporate additional modules to extend areas of the instrument operation that are under computer control is important. The capabilities of an existing control system can be expanded by the addition of modules which add more memory,

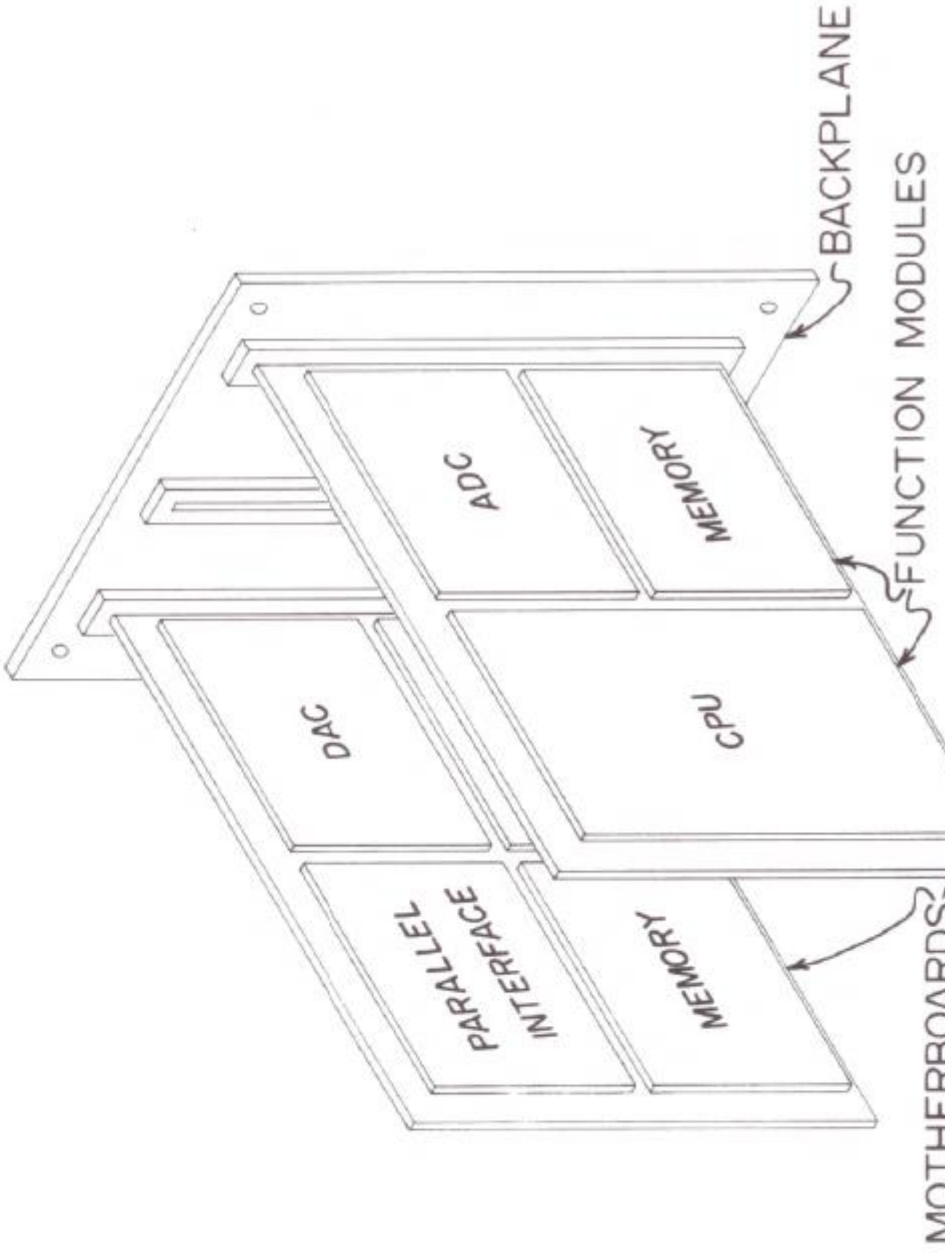


Figure 5.3 Modular Hardware System

graphics display capabilities, or an advanced numeric processor. Common functions such as "chip-select" logic can be implemented on one module for use by all the other modules on each motherboard. This reduces the board size and the complexity of other modules. By minimizing the complexity of these modules custom interfaces for special applications may easily be developed. This hardware system has been utilized to implement an number of small control systems in our laboratory[37],[38],[39].

INTERPROCESSOR HARDWARE

Additional modules were developed for this system to allow processors to be linked together to form a distributed processing system (Figure 5.4). These modules implement three paths of communication which are used to support the interprocessor communication modes outlined in Table 5.3. The first interprocessor module (Figure 5.4A) supports a direct memory transfer communications path. This hardware allows transfers between the master processor and memory on any slave. This is accomplished using bus switching hardware which places any slave processor involved in a transfer into a hold state and then connects the slave processor's address and data bus to an interprocessor communications bus. When a transfer is completed, the slave processor's bus is released, and program execution in the slave is resumed. This hardware is used to implement the block transfer of data between processors and to load software into the slave processors at system startup.

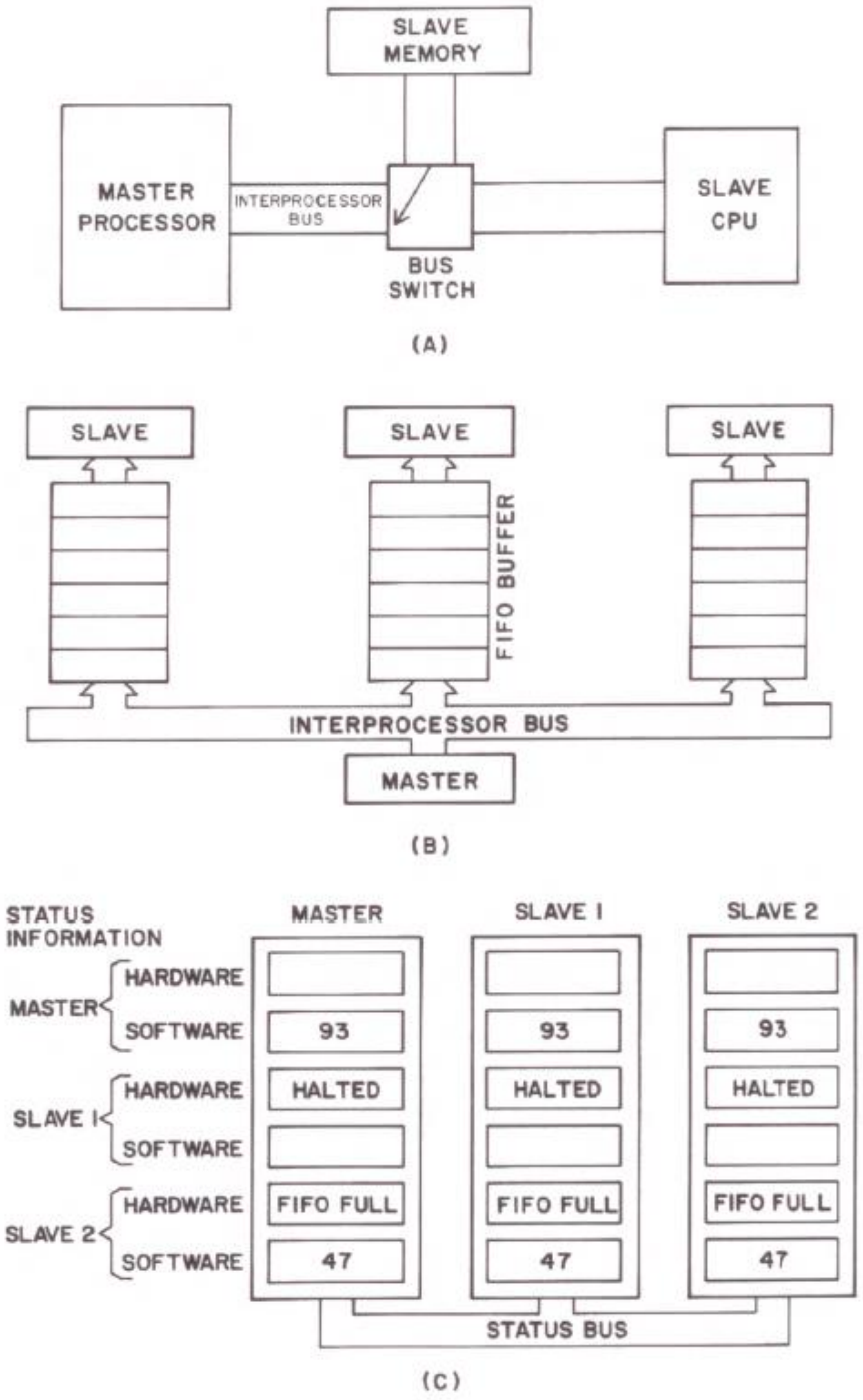


Figure 5.4 Distributed Processing Modules

Table 5.3 Modes and Paths of Interprocessor Communication

Modes of Interprocessor Communication

- Block data transfer
- Task assignment
- Parameter transfer
- Task coordination

Hardware Paths for Interprocessor Communication

- Direct memory transfer
- Command transfer
- Status transfer

The second interprocessor communications path supported in hardware is called the command transfer path (Figure 5.4B). This path is implemented as a set of first-in-first-out (FIFO) buffers for each slave processor in the system. Each FIFO buffer is 32 elements deep and is referred to as a command buffer. The master can write data into any slave's FIFO buffer, which can then be read out and interpreted by the slave. In addition to the command FIFO buffer this module provides control lines that allow the master processor to reset, hold or interrupt a slave processor. The task assignment and parameter passing functions needed for interprocessor communication are implemented utilizing this hardware module.

The third interprocessor module provides a communications path called the status transfer (Figure 5.4C). Each processor in the system (maximum of eight) has one of these modules which contains 16 bytes of dual-port memory. Two bytes are assigned to each processor: one for hardware status and one for user definable software status. Each processor's hardware status byte includes such information as command buffer full or empty, and processor halted. The value of the software status byte can be written by application routines to indicate what function is under execution or when a task has been completed. These values are updated every 4 usec through a portion of the interprocessor communications bus called the status bus, so that any change of status information in the system is known by all processors within 4 usec. This module provides the principal means of interprocessor task coordination.

SOFTWARE

Once this efficient modular hardware system was developed, it was necessary to develop a software operating environment that complements the capabilities of the hardware. The attributes of a good laboratory language have been discussed in some detail in references [40] and[41]. In our laboratory we have chosen to use the high-level programming language FORTH[42],[43] ,[44] to develop instrument control applications. FORTH implements many of the attributes of a good laboratory language. Among FORTH's more important features are its small size, its ability to directly access machine resources, and its extensibility. A standard FORTH system containing the FORTH compiler, an editor, an assembler, disk access functions and terminal I/O routines typically occupies only 8 Kbytes (K=1024) of processor memory. For applications that do not require all of the above functions, the FORTH kernel can be reduced to under 1 Kbyte. This small size makes FORTH well suited for small control system where FORTH can be programmed into read-only memory (ROM) so that it is available upon system powerup. The standard FORTH system implements all of the functions necessary for program development. Thus, each FORTH based instrument can be programmed without the use of additional software packages such as cross compilers, and without the need for a separate development system. FORTH allows the user to access memory and input/output (I/O) ports on peripheral devices directly. There is no operating system standing guard, or interfering, between the user and the system hardware. This is important in instrument control applications where much of the work is interfacing to and operating non-standard peripherals. Many I/O tasks can be taken care of in high-level FORTH. In addition, memory and I/O ports can be directly accessed from a terminal, greatly simplifying the testing and debugging of hardware interfaces.

FORTH is an extensible programming language, one to which new commands and structures can readily be added. A FORTH program, or "word", consists of a series of previously defined words. Each word performs a function such as multiplication, fetching data from memory, acquisition of a data point or plotting a set of acquired data. When executed directly, each word acts like a separate program or function of the system. When used in defining a new "word" or program, existing words act like subroutines in languages such as FORTRAN. A typical FORTH system has several hundred words already defined in it's "vocabulary". Programs are easily written, even by novice users, by merely concatenating existing words: and in the process, a new, higher-level word is added to the vocabulary.

An unusual feature of FORTH is the use of a push-down stack to pass parameters between words. A push-down stack acts as a Last-In-First-Out (LIFO) buffer; items placed on the stack are removed in reverse order. FORTH uses this parameter stack to pass values between both high level and assembly language routines. Thus, the interface to an assembly language routine is no different than to a high level FORTH routine. Whenever a number is entered on a command line it is pushed onto the top of the stack.

In FORTH programming, words build on each other, each new word becomes a higher and higher level of operation. The end result is a high-level language that is specific to an application, which makes programming that application more efficient. Figure 5.5 is a listing of a short FORTH program that illustrates how this comes about. The application to be controlled is a stepper motor that advances 0.25 degrees each time memory location 10 is accessed. Line 0 is just a comment line to indicate these routines are for the stepper motor. The definition of a word is begun with a ":" followed by the name of the new word. The definition of a word is terminated by a ";". Line 2 defines a new word STEP that fetches (@) the value from location 10 and then discards it (DROP). This causes the stepper motor to advance 0.25 degrees. The word STEPS, defined on line 4, pulses the motor n times, where n is the number on top-of-the-stack (TOS). This is done by using the new word STEP and the standard FORTH words DO and LOOP which create a loop that goes from 0 to n, thus repeating STEP n times. The final word, DEGREES, advances the stepper motor a given number of degrees. It takes the number on TOS as the number of degrees to move and multiplies this by four leaving the result on TOS. This provides the number of 0.25 degree steps desired which the word STEPS then uses to advance the stepper motor. This modularity of software complements the modularity of the hardware, where small simple components can be readily combined together to perform a complex function.

```
( STEPPER MOTOR ROUTINES )

: STEP  10 @ DROP ;

: STEPS  0 DO  STEP  LOOP ;

: DEGREES  4 *  STEPS ;
```

Figure 5.5 Sample FORTH Program

For operation in the distributed processing environment, a modified version of FORTH was developed for operation on the slave processors. First the editor, assembler, and disk access functions were removed since they would not be needed on the slave processors. This reduced the size of the basic system to less than 4 Kbytes. The standard command interpreter was then modified to accept commands and parameters from the command buffer FIFOs. Extensions were made to the FORTH system running on the master processor to allow the master to download code into the slave processors, perform interprocessor memory transfers and direct the execution of tasks in the various slave processors. This yielded an integrated system where all processors are programmed in a common language with little deviation from single processor implementations of FORTH[45].

CONTROL SYSTEM DESIGN CONSIDERATIONS

One of the first considerations in designing a control system is the determination of the useful dynamic range of data that the instrument can produce. In many cases the dynamic range of the instrument is not limited by noise from the detector or the electronics but is limited instead by "chemical noise" in the system. This noise results from the chemical background in the sample being analyzed. One method of increasing the dynamic range of the instrument is to add more than one dimension of selectivity. An example of this is the use of chromatography to separate the sample in time before it is analyzed by the instrument. In the triple quadrupole mass spectrometer, there are two stages of selection which results in a usable dynamic range on the order of 10^8. That is, it is possible to have a noise level that is eight orders of magnitude below the level of maximum usable signal.

When a gas chromatograph is used with the triple quadrupole mass spectrometer, a mass scanning rate of 1000 amu/sec or greater is desirable to give adequate resolution of the GC profiles. At these scanning speeds analog current measurement techniques are used with a resulting dynamic range of from 10^5 to 10^6. This range is limited by ion statistics at the low end and detector saturation at the high end. Other sample introduction techniques that provide a less transient sample allow slower scan rates. At slower scan rates, pulse counting techniques can be used to extend the dynamic range to its full value of 10^8. This wide dynamic range must be reflected in the number representations and peak finding algorithms so that significant data is not lost.

The triple quadrupole mass spectrometer also provides a challenge to the designer of the control system because of the large number of parameters that need to be controlled. These devices are listed in Table 5.4 along with the simple mnemonic names that have been assigned to them. The TQMS instrument is capable of operating in three different ionization modes: electron impact ionization, positive chemical ionization, and negative chemical ionization. It can rapidly change between these different modes under computer control. This provides an added level of complexity since the optimum

Table 5.4 TQMS devices and their mnemonic names

NAME	DEVICE	NAME	DEVICE
EV	Electron energy	Q2	Quad 2 DC rod offset
REP	Repeller	Q3	Quad 3 DC rod offset
EIV	EI ion volume	MHV	Multiplier high voltage
CIV	CI ion volume	M1	Mass selected by quad 1
EXT	Extractor	DM1	Quad 1 delta mass
L1	Ion source lens 1	RS1	Quad 1 resolution
L2	Ion source lens 2	M2	Mass selected by quad 2
L3	Ion source lens 3	M3	Mass selected by quad 3
L4	Interquad lens 1-2	DM3	Quad 3 delta mass
L5	Interquad lens 2-3	RS3	Quad 3 resolution
Q1	Quad 1 DC rod offset		

value of all of the devices may be different for each of the different ionization modes and thus the control system must be capable of storing the desired device settings for each the mode. To achieve optimum performance the value of some of the devices should be changed as the mass command is scanned. This tracking of several devices with mass increases the demands put on the control system since it increases the number of calculations that must be performed in real time as the data is acquired.

Other capabilities that are needed in the control system are the ability to view the raw ion signal in real time and the need to display the data that the control system has acquired. The display of the raw ion signal is very important for proper tuning of the instrument. The control system needs to offer a number of aids to the operator so that the TQMS instrument can be interactively tuned using the judgment of the user to optimize the instrument for the experiment. The ability to display the acquired data allows the operator to adjust the acquisition parameters so that the data is collected properly and also permits the course of an experiment to be monitored. Utilities for managing the data that the system acquires are also important features of the control system.

IMPLEMENTATION OF TQMS CONTROL FUNCTIONS

An examination of the tasks involved in controlling a TQMS instrument led to the identification of three major groups of functions; ion path control, signal acquisition or detection, and signal processing (peak finding)[46]. These functions correspond to the efferent, afferent, and central types of vertical partitions discussed earlier. The initial decisions of how to partition functions among processors in a distributed system are the most critical in determining overall system performance and capabilities. A distributed processing system utilizing four Intel 8088 processors was implemented to support these partitioning decisions. In this system one processor acts as the system master and the other three are operated as dedicated slave processors. Figure 5.6 illustrates the interconnection of the master to the slave processors and the connections of the slave processors to the instrument. The horizontal arrows between the slave processors represent the status communication hardware which is used to synchronize the slave processors during scanning operations. The double-ended arrows connecting the master to the various slaves indicates the path of data transfer through which the master can directly access the memory in each slave. Finally, commands and parameters are transferred from the master to each slave through the command buffers depicted as a series of small boxes between the master processor and each slave processor in Figure 5.6.

All of the communication between the instrument and the outside world is provided by the master processor. The master processor supports a number of devices in order to carry out its role. A CRT terminal and keyboard are provided as the primary means of interaction between the operator and the instrument. The master processor accepts commands from the operator and directs the operation of the slave processors to carry out these commands. A printer is provided so that

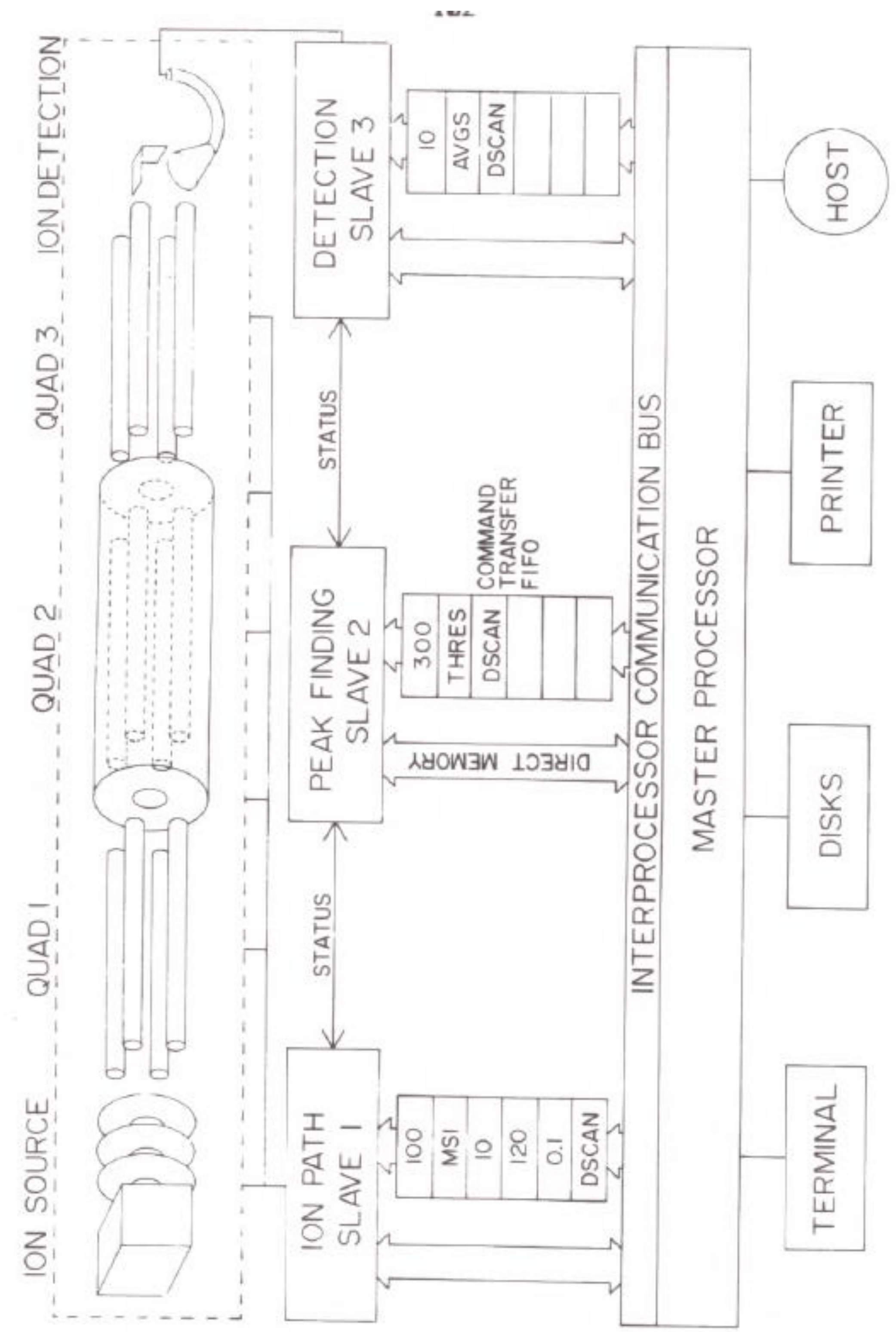


Figure 5.6 TQMS Control System Block Diagram

the operator can obtain a permanent record of results and system parameters to consult during the course of an experiment. An 8 Mbyte Winchester disk is utilized to store data acquired during an experimental session. A link to a "host" minicomputer allows data to be directly transfer to the "host" upon completion of an experiment. An 8-inch floppy disk is provided to allow different operators to maintain their own sets of instrument parameter settings and pre-defined experimental procedures. The master is the most complex processor in the system and as a result contains the largest amount of memory, a total of 48 Kbytes. A basic FORTH system programmed into EPROM (eraseable programable read only memory) occupies 8K bytes of this memory space, leaving 40K available for control software and data space.

The ionpath processor performs all of the efferent functions of the control system. It contains interfaces to provide all of the output signals needed to control the instrument. These interfaces consist mainly of digital-to-analog converters to provide control voltages and a few bytes of parallel output to control the mode selection of the ionizer and quadrupole controllers. All of the DACs have 12-bits of resolution and provide a +/- 10 volt output range with the exception of the DAC's controlling the mass selection of quadrupoles one and three. The mass control DACs have 16-bits of resolution and an output range of 0-10 volts. In addition to many interfaces, this processor is provided with 32 Kbytes of RAM (random access memory) for programs and calibration tables. The software to run the ion path slave is loaded into its memory by the master processor.

One of the main functions of the ionpath slave processor is to perform the transformation of information from a value in a given set of units to a number to send to a DAC to generate the desired value in the instrument. The ion path slave accepts values in experimental units (mass, volts, etc.) from the master, performs the necessary transformations and outputs the result to the instrument. The master is never directly involved in setting instrument parameters and never needs to deal with the values in any other terms than volts or mass.

The second function of the ion path slave is the generation and control of the scanning functions. This is a logical task for this processor since it has local control over the entire instrument. The ion path slave accepts information from the master processor such as: type of scan, range to be scanned and mass step increment to use, and scanning rate. The slave processor then sets up the specified instrument conditions and generates the desired scanning function. During the course of the scan the ion path slave coordinates the advancement of the scan with the operation of the other slave processors. The ionpath slave also signals the completion of the scan to all other processors.

The detection slave processor performs the afferent functions of data acquisition and formatting. This processor controls the interfaces that convert the analog output of the instrument into a digital form. The principal function of this processor is to a acquire a data point. It performs this function by sampling the ion current from the instrument, possibly averaging the result to improve the signal to noise ratio, and then converting the result to a standard numeric format recognized by the other processing elements in the system. Currently only analog data conversion is supported, however this processor provides the system with the processing power need to handle pulse counting hardware when it is installed. The detector slave is provided with 16 Kbytes of RAM. However, with the limited range of function currently demanded of this processor less than 8K bytes are utilized.

The peak-finding slave processor is the main signal processing element in the system. This processor combines positional output from the ion path slave with intensity information from the detection slave. Various criteria are used to determine the presence and position of a peak in the mass spectrum. When a peak is detected this processor records the position and intensity of the peak. This information is transferred to the master processor when a scan is completed. The peak finding slave has no direct connections to the instrument. This processor is supplied with 24 Kbytes of memory, of which 8 Kbytes are allocated to storage of information detected during a scan.

In addition to illustrating the interconnections in the control system, Figure 5.6 illustrates the sequence of commands each processor might receive in order to acquire a daughter scan of a parent ion at mass 100. The detection slave is passed the parameter 10 along with the AVGS command to set the number of times to average the ion-current value to 10. The detection slave is then instructed to prepare for a daughter scan. The peak finding slave is passed the parameter 300 along with the command THRES. This sets the threshold for peak detection to 300 counts. This processor is also instructed to prepare for a daughter scan. When told to prepare for a daughter scan the peak finding and detection slaves look to the ion path slave for synchronization during the scan. Finally, the ion path slave is passed the parameter 100 along with the MS1 command, which selects the parent mass of 100 for the daughter scan. The parameters for the start (mass 10), end (mass 120), and increment of the scan (0.1 amu) are passed to the slave processor prior to issuing the daughter scan command DSCAN.

Figure 5.7 illustrates the synchronization and overlap of tasks in the distributed processing system. This figure also illustrates the increased throughput that can be achieved by the use of several processors in place of a single processor. The data acquisition cycle has been divided into five principal tasks; calculation of ionpath values, setting the ionpath values, acquisition of a data point, format conversions of an acquired data point, and peak finding operations. In a single processor system these functions must be performed in series. The distributed processing system allows these tasks to be overlapped to some extent increasing the system throughput.

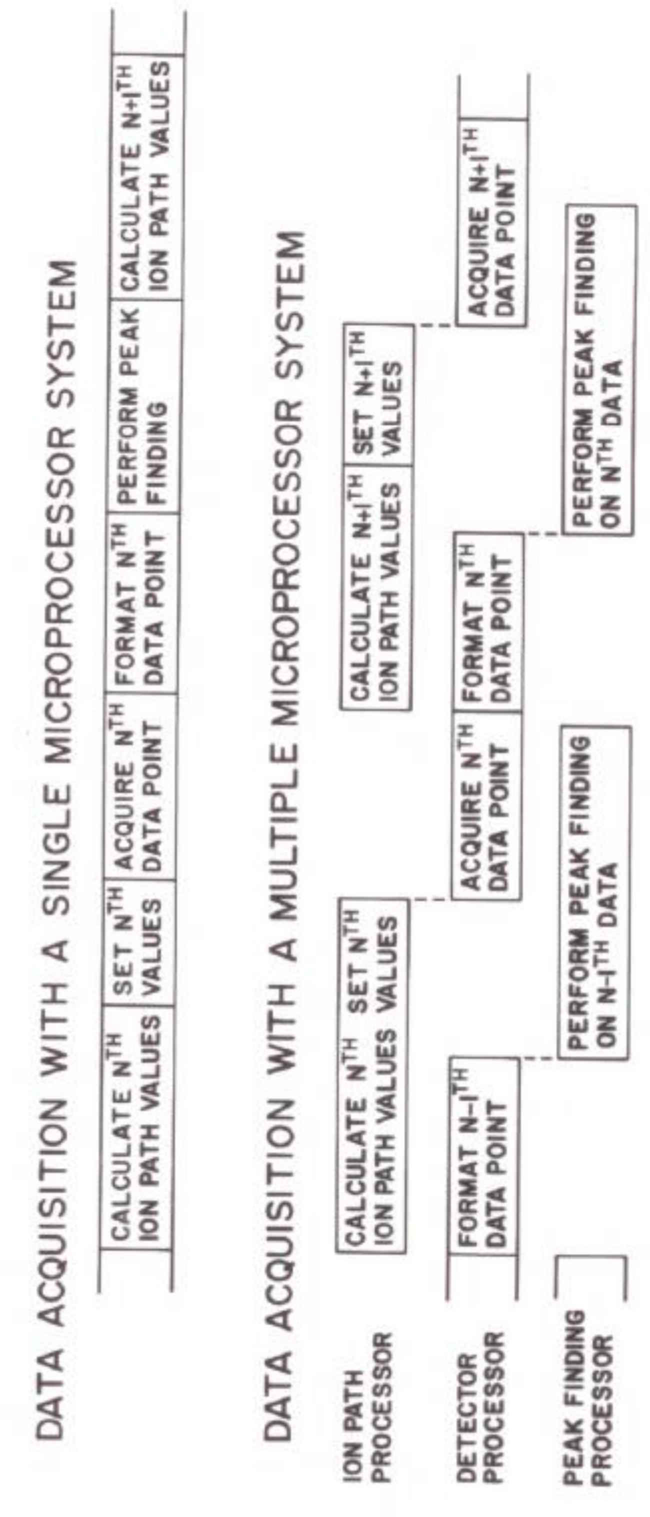

DATA ACQUISITION WITH A SINGLE MICROPROCESSOR SYSTEM
CALCULATE NTH ION PATH VALUES
SET NTH VALUES
ACQUIRE NTH DATA POINT
FORMAT NTH DATA POINT
PERFORM PEAK FINDING
CALCULATE N+1TH ION PATH VALUES
DATA ACQUISITION WITH A MULTIPLE MICROPROCESSOR SYSTEM
ION PATH PROCESSOR
CALCULATE NTH ION PATH VALUES
SET NTH VALUES
CALCULATE N+1TH ION PATH VALUES
SET N+1TH VALUES
DETECTOR PROCESSOR
FORMAT N-1TH DATA POINT
ACQUIRE NTH DATA POINT
FORMAT NTH DATA POINT
ACQUIRE N+1TH DATA POINT
PEAK FINDING PROCESSOR
PERFORM PEAK FINDING ON N-1TH DATA
PERFORM PEAK FINDING ON NTH DATA

Figure 5.7 Parallel Processing Timing Diagram

USER INTERFACE

In order to manage a large number of instrument parameters effectively, special attention must be placed on the interface between the user and the instrument. This involves consideration of how parameters are displayed and modified, how commands are identified, and the feedback provided the user. An important area of operator/instrument interaction is how the user puts information into the system. A wide variety of input options are available today including predefined pushbuttons, keyboards, lightpens, voice, touchscreens, joysticks and trackballs. One or more of these input options may be used in a single system. The choice of input method can greatly affect the ease of instrument use.

When defining system commands, one must attempt to reach a compromise between short cryptic command names and long descriptive command names. Short names may be easy to input, but they suffer from a number of drawbacks. Among these drawbacks are the need for extensive memorization by the user, cryptic command lines and easy confusion between commands. Consider an example where the starting and ending values of a scan must be set as well as the increment or step size to use while scanning. If a two-letter command scheme is used the resulting commands might look like SS, SE, and SI, for set start, set end, and set increment respectively. The resulting commands can hardly be described as descriptive of their function. Long descriptive command names can be easy to learn and make command line easy to read. Using the above example, descriptive command names might take on the form SET-START, SET-END, and SET-STEP. While these are very descriptive of their function, they are too long to be easily entered on a keyboard frequently.

A useful approach to the definition of commands is to partition the name into two sections; a general function class and a specific function class, then abbreviating only one of these sections to help keep the command both short and clear. Two examples of this from the implementation of the TQMS control system are a group of parameters setting functions and the scanning functions. The parameter setting functions are like those outlined in the example above, which are used for setting a start, end and step value. In this case the general function was the setting of a parameter and the specific function was the parameter to be set. The abbreviation "!" was chosen for the general setting function yielding the command names !START, !END, and !STEP. When the general abbreviation "!" is learned these commands are read set-start, set-end and set-step. The second example concerns the commands to direct the system to perform one of the five basic scan types; a quad one scan, a quad three scan, a parent scan, a daughter scan or neutral loss scan. In this case the general function "scan" was retained as the descriptive function and the specific type of scan was abbreviated to 1, 3, P, D, and N respectively. This results in the five scanning commands 1SCAN, 3SCAN, PSCAN, DSCAN, and NSCAN.

To help the user deal with the large number of parameters to be set in a TQMS instrument, interactive screen editors were developed. An example of this is the parameter editor PED (Figure 5.8) that displays a table of the current settings of all devices in the system as well as their start, end and step size scanning parameters. There are more than seventy values in this table. To modify any value, it is first selected by using cursor control keys on the terminal to move a cursor to the desired value. The current cursor position is indicated by displaying the selected value in reverse video. Once a value is selected, the user may enter a new value at that location by merely typing it in from the keyboard. This results in a "what you see is what to change" type system, where there is no command structure needed to identify the parameter of interest. This approach gives the user access to many parameters at one time while making it easy to select and modify an individual parameter. The use of such a screen editor approach is superior to a menu drive parameter selection and modify scheme, since the user can directly modify the parameter of interest without first having to go through several levels of menu selection.

To aid the user in instrument setup, tuning, and operation, three types of display functions are provided by the control system. The simplest is the numerical display of acquired data on the control system terminal. A variety of text displays can be called up. A DLIST command lists the data from the last scan acquired as a table of scanned value (mass, lens voltage, etc.) and ion intensity. The SDIR command displays a directory of all the scans taken during an

```
#1 MSU TUNE                                                      MODE: EI

 # DEV CURRENT  START    END   STEP   # DEV CURRENT  START    END   STEP
 0 EV     70.0    0.0    0.0    0.0  16 M1     219.0   50.0  250.0   0.1
 1 REP    34.3    0.0    0.0    0.0  17 M2       0.0    0.0    0.0   0.0
 2 CIV    36.3    0.0    0.0    0.0  18 M3      69.0   20.0   80.0   0.1
 3 EIV    17.0    0.0    0.0    0.0  19 DM1    -15.1    0.0    0.0   0.0
 4 EXT     9.2    0.0    0.0    0.0  20 DM3      0.4    0.0    0.0   0.0
 5 L1    -16.6    0.0    0.0    0.0  21 RS1      5.8    0.0    0.0   0.0
 6 L2    -23.7    0.0    0.0    0.0  22 RS3    -70.7    0.0    0.0   0.0
 7 L3    -21.0   50.0  -50.0   -0.2  23
 8 Q1     -0.6  -20.0   20.0    0.1  24
 9 L4      1.9 -100.0  100.0    0.5  25
10 Q2      8.4    0.0    0.0    0.0  26
11 L5     -8.3 -100.0  100.0    1.0  27
12 Q3      0.8    0.0    0.0    0.0  28
13 MHV  2450.0    0.0    0.0    0.0  29
14                                   30
15                                   31
```

Figure 5.8 Parameter Editor Display

experiment, listing the type of scan, the range of the scans, and how many data points were recorded. This type of text display provides the user with some immediate quantitative feedback during and immediately after a set of experiments.

The graphical display of acquired data is a second means of immediate feedback the control system provides for the user. A graphics display has been interfaced to the system to allow the display of mass spectra and voltage sweeps. The display routines are primarily used to produce an immediate visual representation of the acquired data. Interactive display functions are not provided, nor are extensive formatting capabilities available. These functions are more appropriate for the data analysis system. However, it is important for a user to be able to view his or her data during the course of an experiment in order to evaluate the performance of the instrument or determine the course of the experiment.

A third type of data display is provided to aid in the tuning of the instrument. This display utilizes an oscilloscope to display the ion current signal to the user in real-time. This allows the user to directly observe the effects of changing various parameters. The x-axis of the oscilloscope display is controlled by the ion path processor using a DAC. The ion path processor also can control the gain of the amplifier which supplies the ion current signal to the y-axis of the oscilloscope. This system allows a flexible tuning aid call a split screen display to be developed. The split screen display can display between one and five mass windows 5 amu wide on the oscilloscope screen. This provides side by side comparisons between peaks that may be widely separated in mass. In addition, the user can specify the individual gain to be used in displaying each mass window. This allows mass peaks of greatly different intensities to be clearly displayed simultaneously. The selection of mass windows to display, the scan type to use (quad one, parent, daughter, etc.), and the gain for each mass window are set up using an interactive screen editor similar to the parameter editor described above. This editor provides control of five mass window and gain selections for each of the five basic scan types, yielding a total of 25 mass-gain pairs. The split screen editor runs on the master processor while the ion path slave processor controls the oscilloscope display. Whenever an entry is changed in the split screen editor, this information is transferred to the ion path slave to control the format of the display.

These three types of display provide the user with a variety of immediate feedback about his or her experiment and the performance of the instrument. The split screen display is a very helpful aid when tuning the instrument. The ability to rapidly switch between different scan types while tuning makes it much easier to optimize the instrument performance for the various scan modes.

One of the most comfortable and useful forms of instrument control implemented in this system is a function we have been calling "softknobs". The "softknob" is an optical rotary encoder that generates two streams of pulses when it is rotated. From these pulse streams it is possible to determine the direction and degree of rotation. One of the features of these rotary encoders is that they have no limit on their rotation. The shaft can rotate in either direction indefinitely. Four of these encoders have been interfaced to the master processor as another form of user input.

The principal use of these softknobs is to allow the user to manually adjust devices in the system. Figure 5.9 illustrates the operation of the softknobs in the distributed control system. "Softknobs" are connected to the master processor through an interface that converts the two pulse streams to direction bits and a strobe signal. The strobe signal is used to interrupt the master processor each time a knobs position is changed. The processor then can read the direction bits to determine whether to increment or decrement a variable assigned to a device. Every 16 ms another task on the master processor reads this variable and adjusts the parameter settings for that device. It then sends any changes to the ionpath slave which outputs the appropriate value to a DAC to change a device setting.

These "softknobs" offer several advantages over conventional knobs and other input devices. Since their action is controlled by software these knobs act as variable resolution devices. Each revolution of a knob generates 250 pulses; however, the software is free to interpret this in anyway necessary. One turn can generate a full-scale change of 0-10 volts at the output by assigning 0.04 volt value to each pulse. However, a knob could be programmed to generate a full-scale change for 10 revolutions by assigning a 0.004 volt value to each pulse. Non-linear and cyclical functions can be assigned to

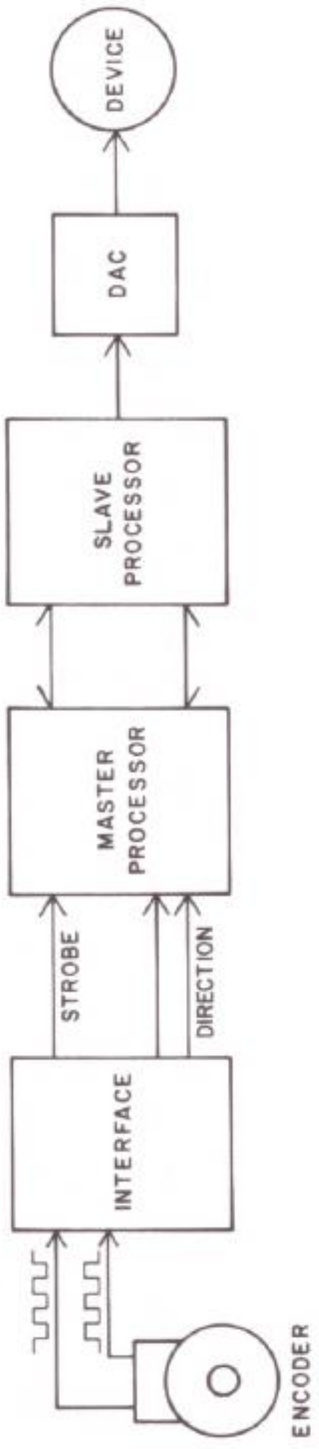


Figure 5.9 Softknobs Functional Diagram

a knob with the appropriate software. The software could be programmed to ramp a signal up to its maximum and back down again in a cyclical manner as a knob is continuously rotated in one direction. The knobs can be reassigned in software, so that a limited number of knobs can control a large number of devices in any combination, thus reducing the costs while retaining manual control features. More than one knob may be assigned to a single device to provide coarse and fine adjustments. The software can limit the output range of a knob so that when a limit is reached no change occurs with further rotation of the knob. The possibilities are endless.

The most important feature of the "softknobs" is that they provide a familiar and effective method by which the user may set instrument parameters without having to develop a separate set of manual and computer controls. As a result of the computer system actually setting the values, the computer knows the value each device is set to and can allow the user to store and retrieve them. In addition, the device setting can be recorded along with the data to form a complete data base.

A final area of user interaction being investigated is the usefulness of voice output from the control system. A Votrax Type-n-talk[47] has been interfaced to the master processor. This unit can accept text information from the control system and convert it into synthesized speech. We are investigating the usefulness of this technique for error reporting. The audible output of the speech synthesis module allows the operator to move about the lab performing other tasks and still be apprised of any error conditions that may develop.

USER PROGRAMABILITY

The ease with which a user can define new commands can greatly influence how effectively an instrument can be utilized. Table 5.5 summarizes function of several of the commands available in the TQMS control system. These are all high-level mass spectrometry commands that a user must learn to use the instrument. Commands can be entered singly or several commands may be entered at once on one line. The use of FORTH allows a novice user who understands these basic commands to create new commands to aid him or her during an experiment.

Figure 5.10 contains a number of examples of new command definitions generated from the basic set in Table 5.5 and how they might be used. Line 1 contains the definition of one of the existing commands given in Table 5.5, MS1. This command interprets the number on top of the stack as a mass value and sets quadrupole one to select the specified mass. The definition of MS1 consists entirely of other high-level commands created for the control system. M1 selects the mass of quadrupole one as the parameter to be modified, and SET outputs the value on top of the stack to the device, in this case quadrupole one. Notice that the creation of this new command required the user to have no knowledge of the FORTH language other than how to define a new word.

Table 5.5 Selected control system commands and their function.

COMMAND	FUNCTION
1SCAN	Quad one scan
3SCAN	Quad three scan
PSCAN	Parent scan
DSCAN	Daughter scan
NSCAN	Neutral loss scan
DISP	Plot acquired data
DLIST	List acquired data points
KNOBS	Activate softknobs
PED	Activate parameter editor
SPLITS	Setup split screen display
SET	Set a selected device to a value
MS1	Set quad one to a specified mass
MS3	Set quad three to a specified mass
ADD	Add two spectra together
SUB	Subtract two spectra
SDIR	Display directory of acquired scans
EI	Select EI ionization mode
+CI	Select positive chemical ionization mode
-CI	Select negative chemical ionization mode
HELP	Display help information

```
( Mass Spec Programming Examples )
: MS1  M1 SET ;

   58.0 MS1 DSCAN  72.0 MS1 DSCAN  100.0 MS1 DSCAN

: DEMO1  58.0 MS1 DSCAN  72.0 MS1 DSCAN  100.0 MS1 DSCAN ;

: DAUGHTERS  BEGIN  MS1 DSCAN  DUP 0= END ;

   58.0 72.0 100.0 DAUGHTERS

: DEMO2  58.0 72.0 100.0 DAUGHTERS ;
```

Figure 5.10 Mass spectrometry programming examples

Line 3 in Figure 5.10, demonstrates how the MS1 command might be used. In this example, the experiment requires that daughter spectra be obtained for the parent masses 58, 72, and 100. As mentioned earlier, a series of commands can be issued on one line. In this case MS1 is used to first select the parent ion at mass 58, then a DSCAN command causes a daughter spectrum to be acquired. This type of command sequence is repeated for the parent masses at 72 and 100 amu. If the command sequence given in line 3 needed to be repeated frequently, a new word could be created to perform this function. Line 5 illustrates the definition the word DEMO1 which performs this command sequence. The definition of this new command consists of merely enclosing the sequence of mass spec commands between a name for the new command (DEMO1) preceded by a ":" and a ";". Now the three different scans can be acquired by issuing the single command, DEMO1.

Although this approach to creating a new command is easy, it is not very versatile since each mass must specified in the definition. If it is frequently necessary to perform a series of daughter scans at a number of different parent masses, it would be advantageous if there were a command that allowed this to be done easily. The definition of such a command, DAUGHTERS, appears on line 7 of Figure 5.10. This command accepts a list of one or more parent masses on the stack, and performs a daughter scan for each parent mass. The definition of DAUGHTERS includes both high-level mass spec commands and basic FORTH words for program flow control. Reference [48] describes the operation of most standard FORTH words in detail. The word BEGIN marks the beginning of a loop. MS1 takes a number from the stack and sets quadrupole one to that mass, then DSCAN performs the daughter scan. DUP makes a copy of the number on top of the stack, and the 0= tests if this number equals zero. This is used to detect the end of the list of parent masses since the top of an empty stack always has a zero in it. If the number does not equal zero the word END causes the commands following the BEGIN to be repeated. When the last parent mass is processed, as indicated by a zero on the stack, the command is terminated. The use of the DAUGHTERS command is illustrated on line 9. This new word can then be used to create another new command DEMO2 that performs the same functions as the previous DEMO1 command. The definition for DEMO2 is given on line 11 of Figure 5.10. This is an example of how high-level mass spec commands can be combined with more conventional programming constructs to create a powerful new command.

SUMMARY

In an advanced control system, the interaction between the operator and the system can occur on a number of different levels. These shells of complexity allow an unsophisticated user to successfully operate the instrument with a small number of commands while providing more capabilities to the more knowledgeable user. In the system that controls the TQMS instrument there are three different levels; the simple command level, an intermediate level which provides facilities for the generation of new experimental methods, and the most advanced level which allows new instrument capabilities to be added to the system. The knowledge that is needed for each level is simply an expansion of that required for the previous level.

In light of the continued rapid pace with which the electronics and computer industries are developing, the future possibilities for chemical instrumentation are very bright. A possibility is the development of an intelligent control system which would be capable of dynamic optimization of the instrument during the course of an experiment. This system would make its decisions on the tradeoffs between sensitivity, selectivity, analysis time, and sample size based on information about the experimental goals. A second stage might be the connection of such a control system to an intelligent data analysis system which would interactively work with the control system to perform an analysis. The data analysis system would first define a set of rules for the control system by interaction between the operator and an "expert system" program. These rules would be used to acquire a set of data which would then be interactively analyzed by the data analysis system and the operator. This analysis would produce a new set of rules that the control system could employ for data collection. The cycle of data collection and experiment definition would continue until further improvements in the information obtained from the instrument was not possible. This could lead to an "integrated" laboratory which would combine the intelligent instruments with automated sample handling. In this approach all of the instruments in the laboratory would be connected together to an "expert system" data analysis system. The operator would then interact with this system to define the objectives of the analysis.

Interactive analysis of the sample could then be performed on a number of instruments and the combined results of all of the experiments would be used to complete the analysis of the sample.

REFERENCES

[17] Betterridge, D., Goad, T.B., Analyst 106(1260), 257, (1981)

[18] Perrin, D.D., Talanta 24(6), 339, (1977)

[19] Peixin, H., Avery, J.P., Faulkner, L.R., Anal. Chem. 54(12), 1313A, (1982)

[20] Dessy, R.E., Anal. Chim. Atca 103, 459, (1978)

[21] Ziegler, E., Anal. Chim. Acta 147, 77, (1983)

[22] Digital Equipment Corp., Maynard,MA

[23] Hoffman, P.A., Enke, C.G, Computers & Chemistry 7(2), (1983), 47-50

[24] Myerholtz, C.A., Newcome, B.H., Enke, C.G., Rev. Sci. Inst. submitted 1983

[25] Susaki, H., Minami, S., Appl. Spec. 36(5), 553, (1982)

[26] Enke, C.G., Proc. 28th IUPAC Conference, Vancouver,BC (1981)

[27] Fathi, E.T., Krieder, M. Computer, March 1983, P. 23.

[28] Searle, B.C., Freberg, D.E., Computer, 22, Oct. 1975.

[29] Anderson, G.A., Jensen, E.D., Computer Surveys, 7(4), (1975), 197:

[30] Dessy, R.E, Anal. Chem. 54(11), (1982), 1167A

[31] Dessy, R.E., Anal. Chem. 54(12), (1982), 1295A

[32] Newcome, B.H., Enke, C.G., Rev. Sci. Inst.

[33] J.T. Lawson, M.P. Mariani, Proceedings of the IEEE, 358 (1978)

[34] Yourdon, E., Constantine L.L., "Structured Design", Yourdon Incl., 1133 Ave. of the Americas, New York, N.Y., February 1976.

[35] Newcome, B.H., Enke, C.G., Rev. Sci. Inst.

[36] Intel Corp., Bowers Rd., Santa Clara, CA

---

[37] Jones, L.M. Leroi, G.E., Myerholtz, C.A., Enke, C.G., Rev. Sci. Inst. Accepted for publication 10/83

[38] Aiello, P.J., Enke, C.G, ACS Symposim "Image Devices in Spectroscopy", in press

[39] Stults, J.T, Newcome, B.H., Myerholtz, C.A., Enke, C.G., 31st Annual ASMS conference, Boston, MA, 1983

[40] Dessy, R. Anal. Chem 55(6), 1983, 650A

[41] Dessy, R. Anal. Chem 55(7), 1983, 756A

[42] C.H. Moore, Astrom. Astrophys. Suppl. **15**., 497, (1974)

[43] J.S. James, Byte **5**, No. 8, 100 (1980)

[44] S.M. Hicks, Electronics, March 15, 1979, p. 114

[45] Myerholtz, C.A., Enke, C.G. Rev. Sci. Inst.

[46] Myerholtz, C.A., Newcome, B.H., Enke, C.G., 31st Annual Conference, American Soc. for Mass Spec., Boston, MA, (1983)

[47] Federal Screw Works, Troy, MI

[48] Brodie, L.,"Starting FORTH", Prentice Hall, Englewood Cliffs, NJ, 1981

Part II. Screening Applications for Fuel Analysis

Chapter 6 : Screening Aviation Fuels for Thiophenes

Screening Aviation Fuels for Thiophenes with Triple Quadrupole Mass Spectrometry

Carl A. Myerholtz

Adam J. Schubert

Christie G. Enke

Department of Chemistry

Michigan State University

East Lansing, MI 48824

ABSTRACT

The application of triple quadrupole mass spectrometry (TQMS) to the screening of jet aircraft fuels for thiophenes is examined. Jet aircraft fuels present a complex hydrocarbon matrix which makes identification of low-level components difficult. Spectra of the collisionally activated dissociation (CAD) fragments of several thiophenes and potentially interfering compounds were obtained. The fragmentation reactions found to be characteristic of the thiophenes are the loss of a 45 amu neutral fragment and the generation of a 97+ daughter ion. These screening reactions were applied to untreated samples of JET A aviation fuel, and readily detected the presence of thiophene with 0-4 carbons in side-chains. The validity of these screening reactions was confirmed by the use of GC/MS/MS.

A number of studies have shown that low level concentrations of heteroatom containing species can affect two important characteristics in jet aviation fuels[49],[50],[51],[52]. These are the storage stability and thermal stability of the fuel. The storage stability of a fuel is a measure of how well a fuel can tolerate long term exposure to temperatures between 50-125 C, without polymerizing and forming sediments. The thermal stability of a fuel is a measure of the amount of deposit formation the fuel produces when subjected for short periods of time to temperatures in the 200-400 C range. These are the conditions a fuel experiences in a jet engine where significant deposit formation can have serious consequences. Knowledge of the presence and distribution of various heteroatom containing species can aid studies of fuel storage and thermal stability. This information becomes more important when the suitability of alternate sources of feedstocks such as shale oil are considered.

Jet fuels present a complex hydrocarbon matrix which makes identification of low-level components difficult. The probability of finding unique molecular ion peaks (with a unit mass resolution mass spectrometer) for each compound or class of compounds in the raw fuel mass spectrum is very low. Triple quadrupole mass spectrometry offers several modes of component specificity other than the molecular ion mass. For particular components, characteristic combinations of parent and daughter masses can provide a high degree of selectivity. In addition, particular daughter ion or neutral loss masses can be highly indicative of certain compound classes.

In the past, the use of mass spectrometry to detect small quantities of heteroatom species in complex hydrocarbon mixtures such as jet fuels required the separation of the aromatic and polar species from the aliphatic species that make up the bulk of the fuel. This can be a very time consuming operation. The goal of this project was to utilize the separatory power of the triple quadrupole mass spectrometer to determine if a screening procedure for selected heteroatom-containing species which did not include prior separation could be developed. The thiophenes were selected as the target compound class since their presence has been reported in jet fuels[53] and there is evidence that they affect the thermal stability of fuels[54],[55]

EXPERIMENTAL SECTION

Two TQMS instruments were employed in the course of this study. The initial exploratory studies were performed on a triple quadrupole mass spectrometer built in our laboratory[56], while the remainder of the work was performed on a model ELQ-400-3 instrument manufactured by Extranuclear Inc. of Pittsburg, PA.

<u>Instrumental Parameters.</u> Both instruments used in this study were operated in the electron impact (EI) ionization mode, with 20 eV of electron energy. The ion source was operated at 100 C. Argon of 99.9% purity was used as the collision gas in the second quadrupole region for all collisional activated dissociation experiments. The argon pressure was maintained between 0.5 and 1.5 mTorr. Ion energies

in the range of 10-20 eV were employed to induce fragmentation. All data were acquired using a scanning rate of 100 amu/sec.

<u>Sample Introduction.</u> Samples of the reference compounds and fuels were introduced in liquid form into a heated inlet system by use of a syringe. The heated inlet system had a 500 mL expansion volume that was maintained at 150 C. Sample sizes varied between 1-5 uL. The gas chromatograph was operated at a 1 mL/min. flow rate of helium and a split ratio of 20:1. During the GC experiments the injector, transfer line and ion source were operated at 200 C, 250 C, and 250 C respectively. The GC oven temperature was programmed for 4 C/min heating rate starting at 50 C, continuing to 250 C and holding at 250 C. Sample injections into the GC were between 0.5 and 1.0 uL.

<u>Chemicals.</u> All samples of pure compounds were purchased from either Chemservice Inc. or the Aldrich Chemical Co. The fuel samples were obtained from Dr. Gary Seng at NASA's Lewis Research Center.

RESULTS AND DISCUSSION

The general formula for the thiophenes is CnH2n-4S, forming a homologous mass series of 84, 98, 112, 126, etc. The parent mass alone is inadequate to identify the thiophenes since the cycloalkanes and olefins are isobaric with the thiophene family. Conventional mass spectrometers are unable to resolve these peaks unless they have a high resolving power (greater than 1/31,000 at 100 amu) or are combined with a separation technique such as GC or LC.

Mass spectral data from collision activated dissociation spectra obtained for a number of thiophenes and compounds that are isobaric with the thiophenes are presented in Table 6.1. These data were obtained by setting quadrupole 1 to transmit the parent ion, operating quadrupole 2 in the radio frequency only mode while it was filled with argon, and scanning quadrupole 3 to obtain the spectrum of all the collision activated dissociation (CAD) generated daughter ions[57]. The daughter spectra of the various thiophenes were studied to identify dissociation reactions that are unique or characteristic of thiophenes. The loss of a neutral fragment of 45 amu (representing CHS) was common to all the thiophenes and was not observed in any of the isobaric compounds that may be present in a complex hydrocarbon mixture. The formation of a 45+ fragment ion representing CHS+ was also observed. In addition, all of the substituted thiophenes generate an intense 97+ daughter ion. Figure 6.1 illustrates some proposed decomposition mechanisms for the loss of a 45 amu neutral fragment and the generation of +97 daughter ions from different thiophenes. Table 6.2 shows, in a condensed form, the results of the search for identifiable characteristics of thiophenes. As indicated earlier, parent mass alone is inadequate to identify the presence of a thiophene. However, the neutral loss of 45 amu, qualified by the parent mass, provides a unique identification for the thiophenes.

A TQMS instrument is operated in the neutral loss mode to identify all the ions in a mixture that undergo a fixed neutral loss. In the neutral loss mode, quadrupoles 1 and 3 are both scanned in unison at a fixed mass separation. To be detected, an ion selected by

Table 6.1 CAD spectra of reference compounds

COMPOUND	PARENT ION	FRAGMENT IONS
Thiophene	84 (100.0)	69 (3.2), 58 (24.0), 45 (5.6), 39 (1.1)
2-Methylthiophene	98 (36.1)	97 (100.0), 69 (0.8), 54 (3.7), 53 (3.6), 45 (1.9), 39 (1.6)
2,5 Dimethylthiophene	112 (100.0)	97 (87.2), 67 (1.1), 53 (1.0), 45 (0.7)
2-Ethylthiophene	112 (11.7)	97 (100.0), 69 (0.7), 67 (0.4), 53 (4.2), 45 (1.1)
Cyclohexane	84 (90.6)	69 (24.6), 56 (100.0), 55 (35.0), 42 (24.1), 41 (56.2), 39 (2.0), 29 (1.7)
Methycyclohexane	98 (79.7)	96 (3.6), 83 (100.0), 82 (37.0), 70 (14.8), 69 (19.5), 56 (27.0), 55 (79.3), 42 (18.8), 41 (24.5), 29 (1.9)
1-Hexene	84 (62.3)	69 (33.1), 56 (52.9), 55 (100.0), 42 (47.1), 41 (34.2), 29 (8.2)
2-Heptene	98 (59.8)	70 (22.9), 69 (57.4), 56 (100.0), 55 (70.6), 43 (4.7), 42 (8.9), 41 (34.9)
2-Octene	112 (54.6)	83 (23.9), 70 (57.6), 56 (43.8), 55 (100.0), 42 (17.8), 41 (34.6)
n-Hexane	86 (48.5)	57 (100.0), 56 (59.1), 43 (62.9), 41 (50.8), 29 (52.8), 27 (6.6)
n-Heptane	100 (33.4)	71 (45.7), 70 (21.9), 57 (47.5), 55 (12.1), 43 (100.0), 41 (24.0), 29 (17.3), 27 (2.1)
n-Octane	114 (14.1)	85 (25.2), 84 (10.9), 71 (14.4), 70 (6.7), 57 (26.0), 55 (7.6), 43 (100.0), 41 (12.7), 29 (7.8

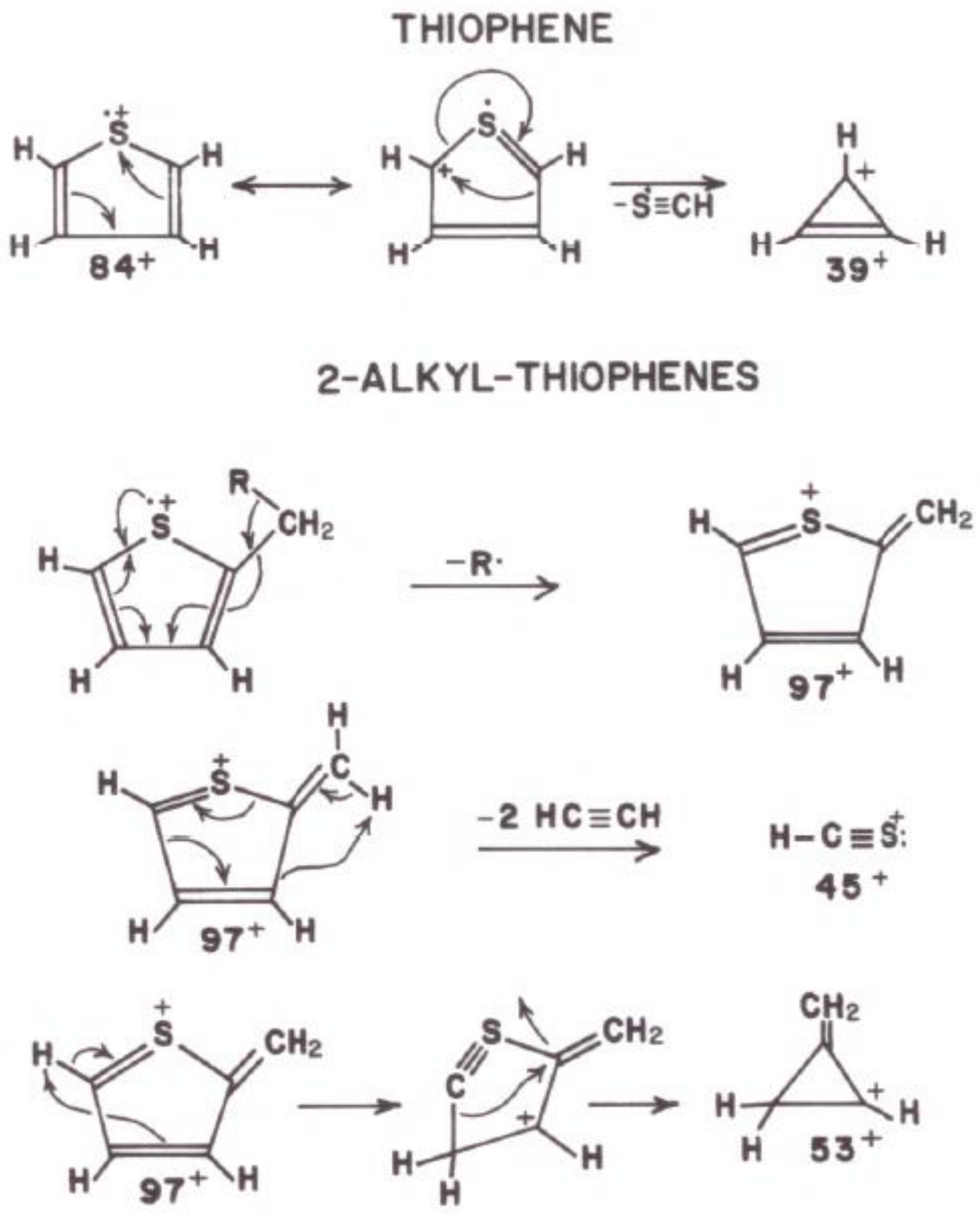


Figure 6.1 Thiophene Decomposition Mechanisms

Table 6.2 Summary of characteristic ions

	PARENT MASS						NEUTRAL LOSS	DAUGHTER ION	
	84	98	112	86	100	114	45	97	45
THIOPHENE	X						X		X
2-METHYL-THIOPHENE		X					X	X	X
2-ETHYL-THIOPHENE			X				X	X	X
DIMETHYL-THIOPHENE			X				X	X	X
1-HEXENE	X								
CYCLO-HEXANE	X								
2-HEPTENE		X							
METHYL-CYCLOHEXANE		X							
2-OCTENE			X						
n-HEXANE				X			X		
n-HEPTANE					X		X		
n-OCTANE						X	X		

quadrupole 1 must undergo the selected neutral loss in quadrupole 2 in order to be transmitted by quadrupole 3 to the detector. In the present instance, the neutral loss mode produces a spectrum of all the parent ion masses which undergo a loss of 45 amu upon collision. In order to obtain a scan of all ions that generate a given daughter ion, such as 97+, the parent scan mode is used. The parent scan involves selecting the desired daughter ion with quadrupole 3 while quadrupole 1 is scanned over the desired mass range. An ion must be transmitted by quadrupole 1, undergo CAD in quadrupole 2, and generate the particular fragment ion selected by quadrupole 3 to be detected.

As suggested by the study of the fragmentation of pure thiophenes and potentially interfering compounds, the loss of 45 was used as a primary screening characteristic and the formation of a 97+ daughter ion was used as a secondary characteristic to confirm the presence of thiophenes in the sample. In Figure 6.2, the results of scanning samples of Jet A and a shale oil derived fuel for a neutral loss of 45 are presented. Low ionization potentials (20 eV) were used in an attempt to minimize fragmentation of the samples in the source[58]. The peaks at 84, 98, 112, 126, 140, 154, and 168 amu indicate the presence and distribution of thiophenes with 0-6 carbons in side chains. Figure 6.3 shows a portion of the raw mass spectra for Jet A and the shale oil fuel covering the same mass range as the neutral loss scans presented in Figure 6.2. Comparisons of the spectra in Figures 6.2 and 6.3 illustrate dramatically how readily a TQMS instrument can detect components in complex mixtures. Figure 6.4 illustrates the results of scans of the two fuel samples for parents of the +97

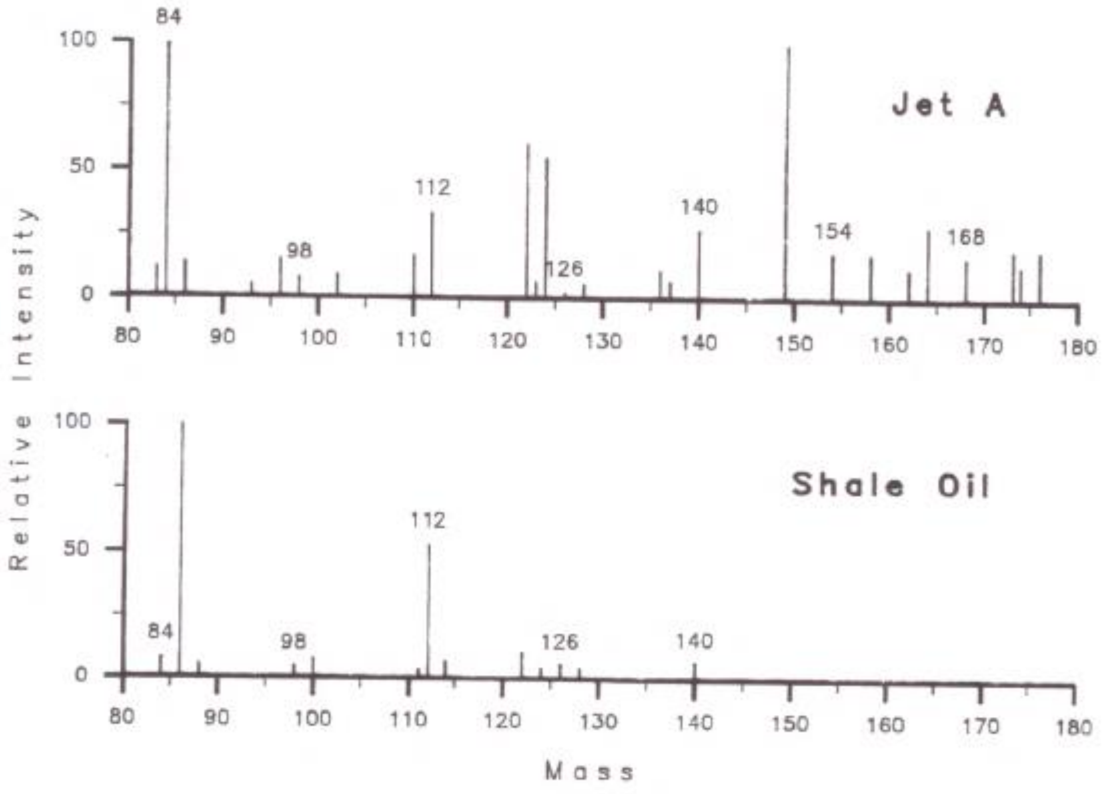
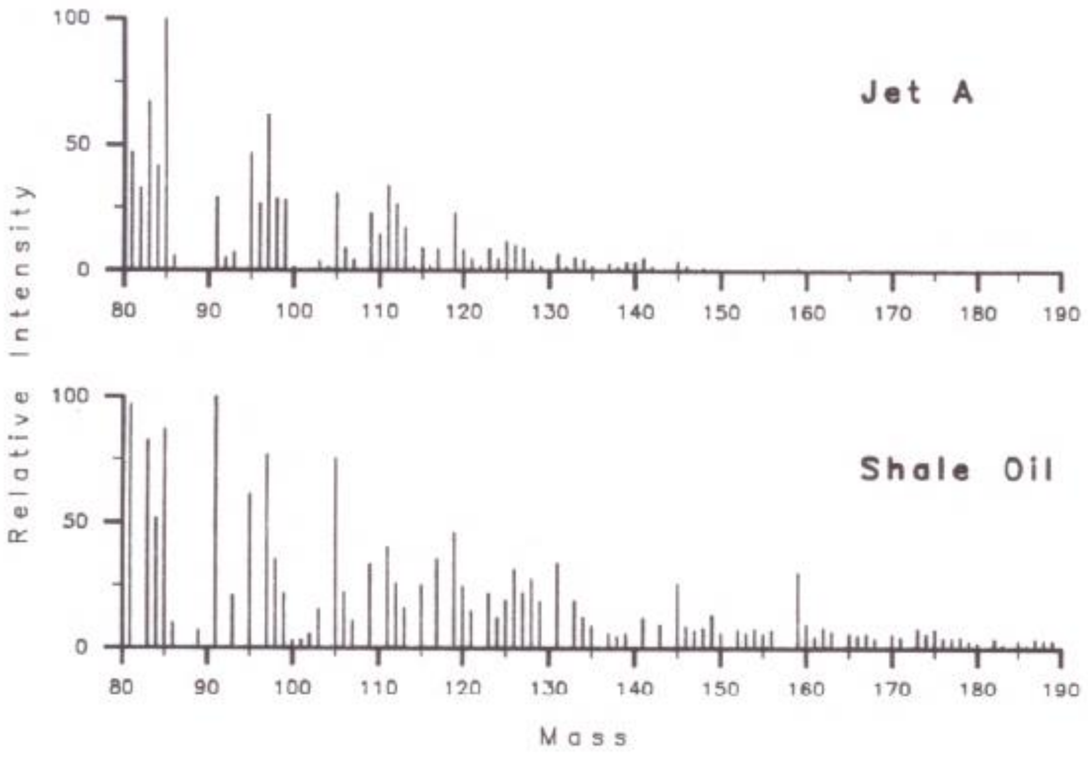


Figure 6.2 45 Neutral Loss from Jet A and Shale oil

F



igure 6.3 Raw Spectra of Jet A and Shale oil

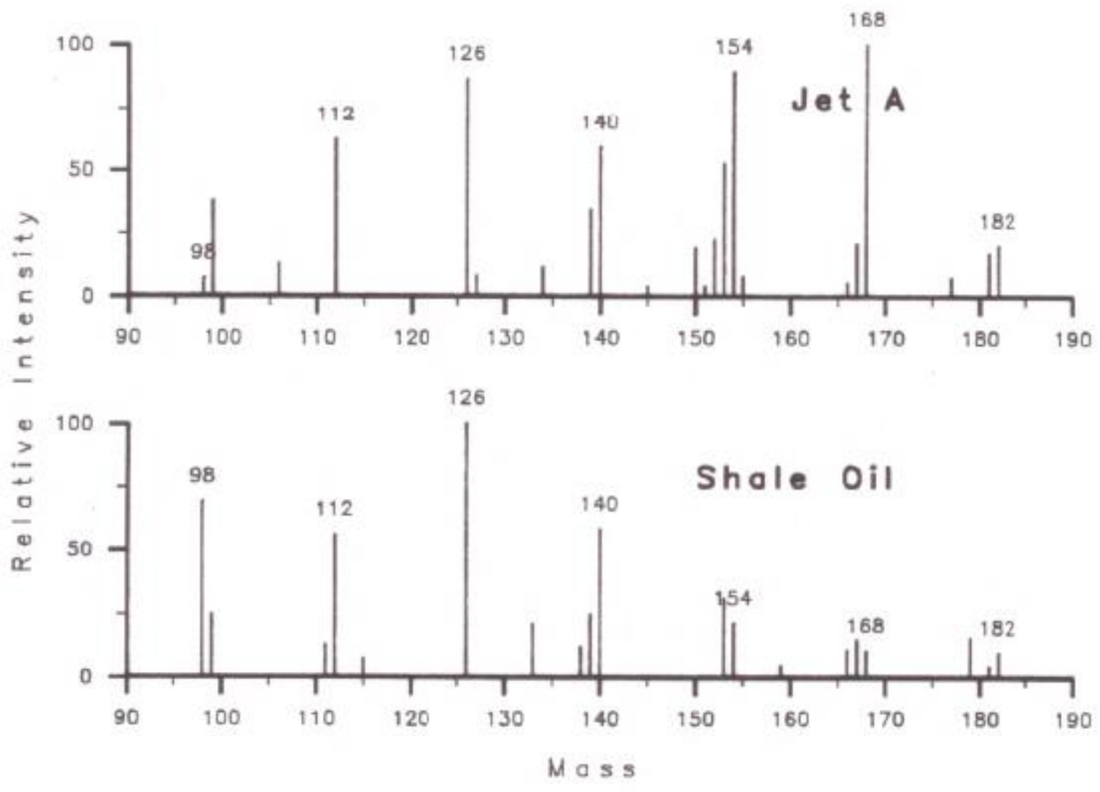


Figure 6.4 97+ Parent scans of Jet A and Shale oil

daughter ion used to confirm the presence of substituted thiophenes. The peaks at 98, 112, 126, 140, 154, and 168 are readily apparent and support the results presented in Figure 6.2.

To confirm the identification of the thiophenes detected in Jet A by the rapid screening technique, additional experiments were performed with the aid of a gas chromatograph interfaced to the triple quadrupole mass spectrometer. These experiments involved the use of a technique known as multiple reaction monitoring (MRM). MRM is the GC/MS/MS analog of multiple ion monitoring (MIM) routinely performed with conventional GC/MS instruments. Instead of selecting a series of fixed mass settings at which to repetitively monitor the ion current, a series of parent-daughter mass pairs are selected to monitor the ion currents arising from selected CAD reactions in the collision cell. For example, to monitor for the 45 neutral loss from thiophene, quadrupole one would be set to transmit ions of mass 84, quadrupole two would be pressurized with collision gas, and quadrupole 3 would be set to transmit ions of mass 39. Similarly, several other parent-daughter reaction pairs can be monitored sequentially and repetitively in an MRM experiment.

To determine the retention times of the thiophenes, a standard mixture of four thiophenes was prepared in a dodecane solvent. The resulting MRM chromatograms are presented in Figure 6.5 and were used obtained the retention times of the reference thiophenes. This experiment was repeated with a 1.0 ul sample of Jet A. Table 6.3 lists the thiophenes present in the mixture, the reactions used to monitor

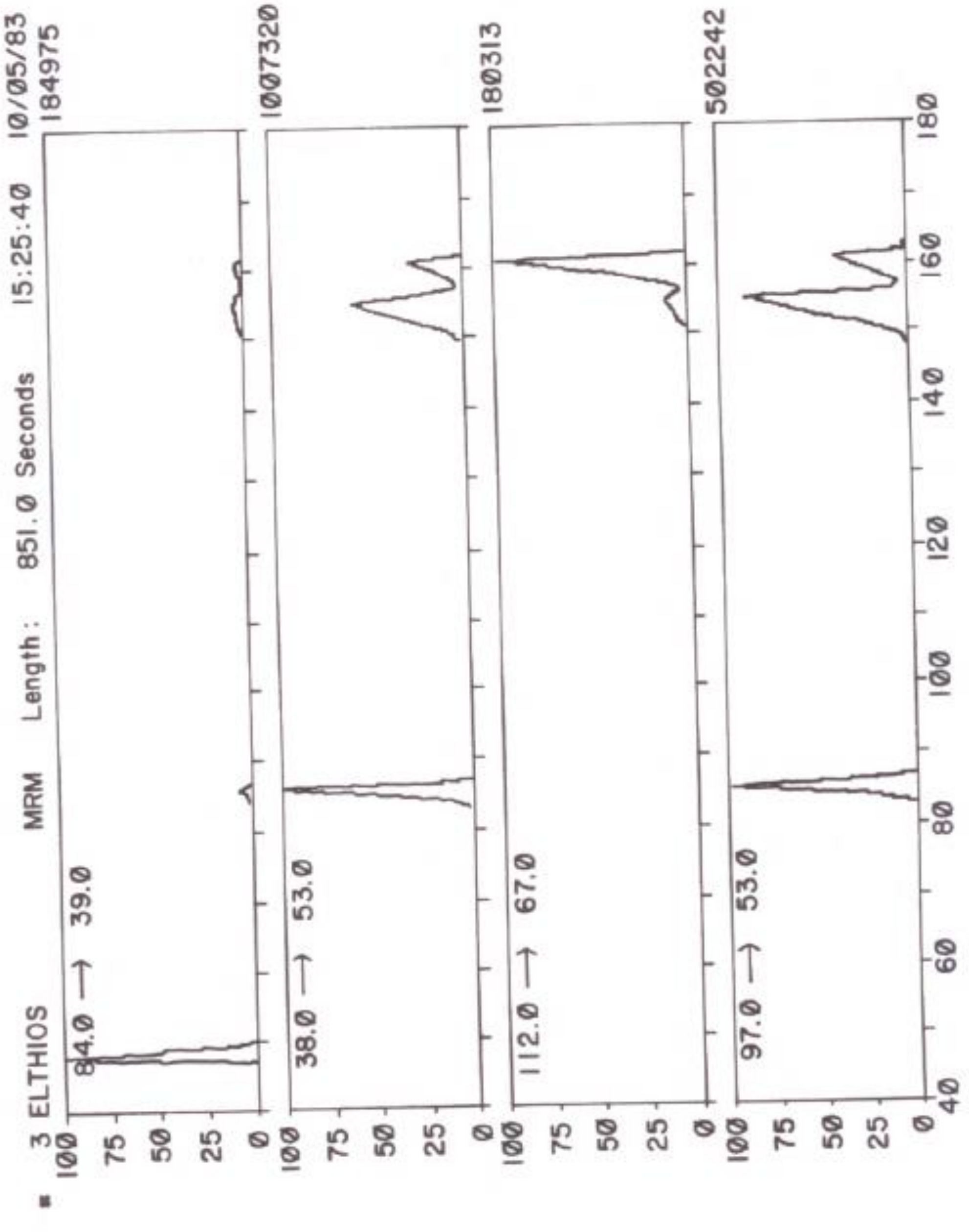


Figure 6.5 Multiple Reaction Monitoring Chromatograms

Table 6.3 Thiophenes, reactions, retention times

Compound	Monitoring Reaction	Retention Time (sec.) Std.	Jet A
Thiophene	84 -> 39	48.0	51.0
2-Methylthiophene	98 -> 53	85.8	87.9
2-Ethylthiophene	112 -> 67, 97 -> 53	154.9	155.2
2,4 Dimethylthiophene	112 -> 67, 97 -> 53	161.1	163.0

each thiophene, and the reference retention times and the retention times observed from Jet A. The retention times of the components of Jet A monitored by the reactions attributed to the thiophenes match the observed retention times for the reference thiophene mixture. These results confirm the ability of the selected screening reactions to detect the presence of thiophenes in complex mixtures such as jet aviation fuels.

CONCLUSIONS

This work illustrates a method for the development of a rapid screening procedure for a particular compound class using triple quadrupole mass spectrometer. The separatory power of a TQMS instrument allows detection of preselected trace components in complex matrices without the need for prior sample work up or additional chromatographic techniques. The capability of TQMS to analyze fuel samples directly can reduce sample handling and analysis time to about 5 minutes. The ability of a triple quadrupole mass spectrometer to detect a minor component in such a complex hydrocarbon mixture can make it a valuable tool in fuel stability studies.

ACKNOWLEDGMENTS

The authors would like to thank the National Aeronautics and Space Administration for financial support of this project, and the people at Extranuclear Inc. for their assistance and use of their equipment.

REFERENCES

[49] Dahlin. K.E.; Daniel, S.R.; Worstell, J.H. Fuel 1981, 60, 477-480

[50] Tutubalina,V.P.; Gabdrakhmanov, F.G.; Korotkova, E.G. Deposited Doc. 1980, SPSTL 460Khp-D80

[51] Hazlett, R.N.; Hall, J.M. Prepr. - Am. Chem. Soc., Div. Pet. Chem. 1981, 26(2), 613-619

[52] Worstell, J.H.; Daniel, S.R. Fuel 1981, 60, 481-484

[53] Bol'shakov,G.F., Izv. Vyssh. Uchebn. Zaved., Neft Gaz 1976 19(5), 51-3

[54] Chertkov, Y.B., Chem. Technol. Fuels Oils, (1976),154

[55] Heneman, F.C. Gov. Rep. Announce. Index 1981, 81(26), 5656

[56] R.A. Yost, C.G. Enke, Anal. Chem. 50, 1251A (1979)

[57] R.A. Yost,C.G. Enke,D.C. McGilvery,J.D. Morrision, Int. J. of Mass Spectrom. and Ion Physics, 30, 127 (1979)

[58] Aczel, T., Rev. Anal. Chem., 1, 226 (1972)

Chapter 7 : Screening Fuels for Selected Species

Screening Middle Distillate Fuels for Selected Species by Triple Quadrupole Mass Spectrometry

Carl A. Myerholtz

Christie G. Enke

Department of Chemistry

Michigan State University

East Lansing, MI 48824

ABSTRACT

The use of triple quadrupole mass spectrometry, an MS/MS technique, to detect selected species in middle distillate fuels has been examined. Collision-activated dissociation (CAD) spectra were obtained for reference compounds from several heteroatom-containing compound classes. These included the thiophenes, thiols, nitrobenzenes, pyridines and anilines. The alkylbenzenes were examined in addition to heteroatom-containing species. The CAD results were used to select screening reactions for each compound class. The effectiveness of these screening reactions was demonstrated by identifying the presence of various species in samples of Jet A aviation fuel, a shale oil derived fuel and No. 2 diesel fuel. Triple quadrupole mass spectrometry can be used to rapidly identify a number of different components in middle distillate fuels. This information can be an aid to studies of fuel composition and stability.

In recent years, a number of studies have shown that the thermal and storage stabilities of middle distillate fuels are affected by low level concentrations of heteroatom containing species[59],[60] ,[61] ,[62] Thermal stability is the resistance of a fuel to deposit formation when stressed at temperatures encountered in an operating engine. Storage stability is the resistance of a fuel to the formation of gums under storage conditions that are less strenuous, but of longer duration than the thermal stresses encountered in an engine. Studies of thermal and storage stabilities of fuels would be aided by knowledge of the presence and distribution of various heteroatom-containing species and how they change under thermal stress. This information becomes even more important when the suitability of alternate sources of feedstocks such as shale oil are considered.

Triple quadrupole mass spectrometry (TQMS) is a tandem mass spectrometry or "MS/MS" technique. An MS/MS instrument consists of two mass analyzers separated by an ion-molecule collision region. Ions of a selected mass are allowed to pass through the first mass analyzer and into the collision region. There the ions undergo collision-activated dissociation (CAD). The second mass analyzer is then used to select particular masses of fragment ions for detection. A TQMS instrument utilizes two quadrupole mass filters as mass analyzers. The collision cell is also a quadrupole, but one that is operated in the "RF only" mode. This mode provides a minimum of mass discrimination and acts rather as an "ion pipe" to contain the reactant ions and all the ionic products of the ion-molecule reaction. Without the quadrupole collision chamber, losses due to scattering would be excessive. A block diagram of a TQMS instrument is shown in Figure 7.1. The three quadrupoles are identified by number, beginning at the source. Ions undergo CAD in the center quadrupole (number 2) at much lower energies (5-30 eV) than other MS/MS techniques which use magnetic and electric sectors and operate at collision energies in the 3-10 keV range[63].

A TQMS instrument can be operated in a variety of modes to perform different types of experiments. Figure 7.2 illustrates the operation of a triple quadrupole mass spectrometer for three types of experiments useful in mixture analysis. In addition to mixture analysis applications, TQMS can be a useful tool in the elucidation of the structure of pure compounds[64].

In MS/MS the term "parent" is used to describe an ion selected from the ion source by the first mass analyzer. The term "daughter" is used to describe an ion that is a result of fragmentation of a "parent ion". In TQMS, the daughter ions are produced by CAD in the the collision region. The operation of a TQMS instrument to perform the three types of MS/MS experiments utilized in this study is described below.

<u>Daughter Scan.</u> Quadrupole one is set to transmit only ions of a selected mass from the source. These ions undergo CAD in the collision cell and generate a number of fragment or daughter ions. Quadrupole three is scanned to allow these fragment ions to be detected. The result is a spectrum of all the daughter ions generated by the parent ion selected by quadrupole one. This type of experiment is

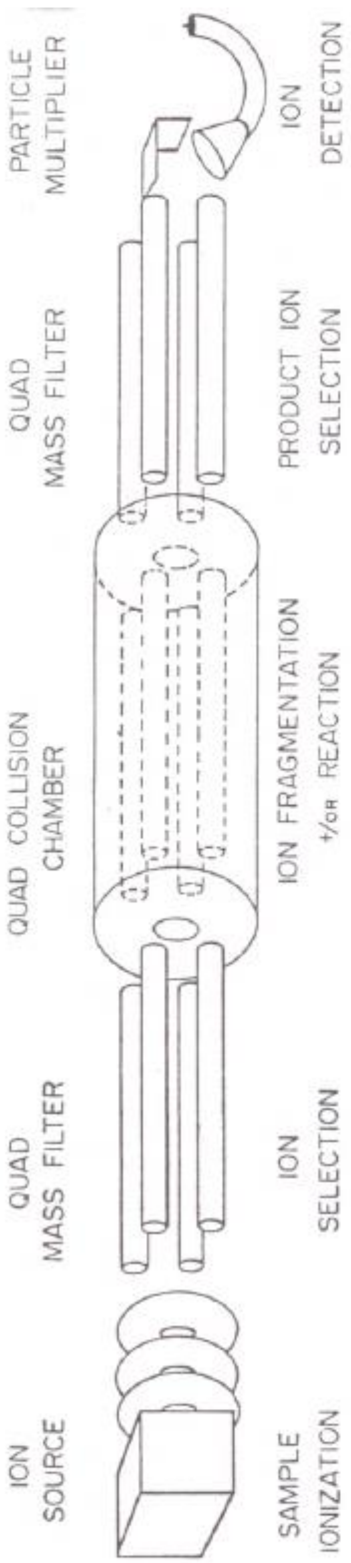


Figure 7.1 TQMS instrument block diagram

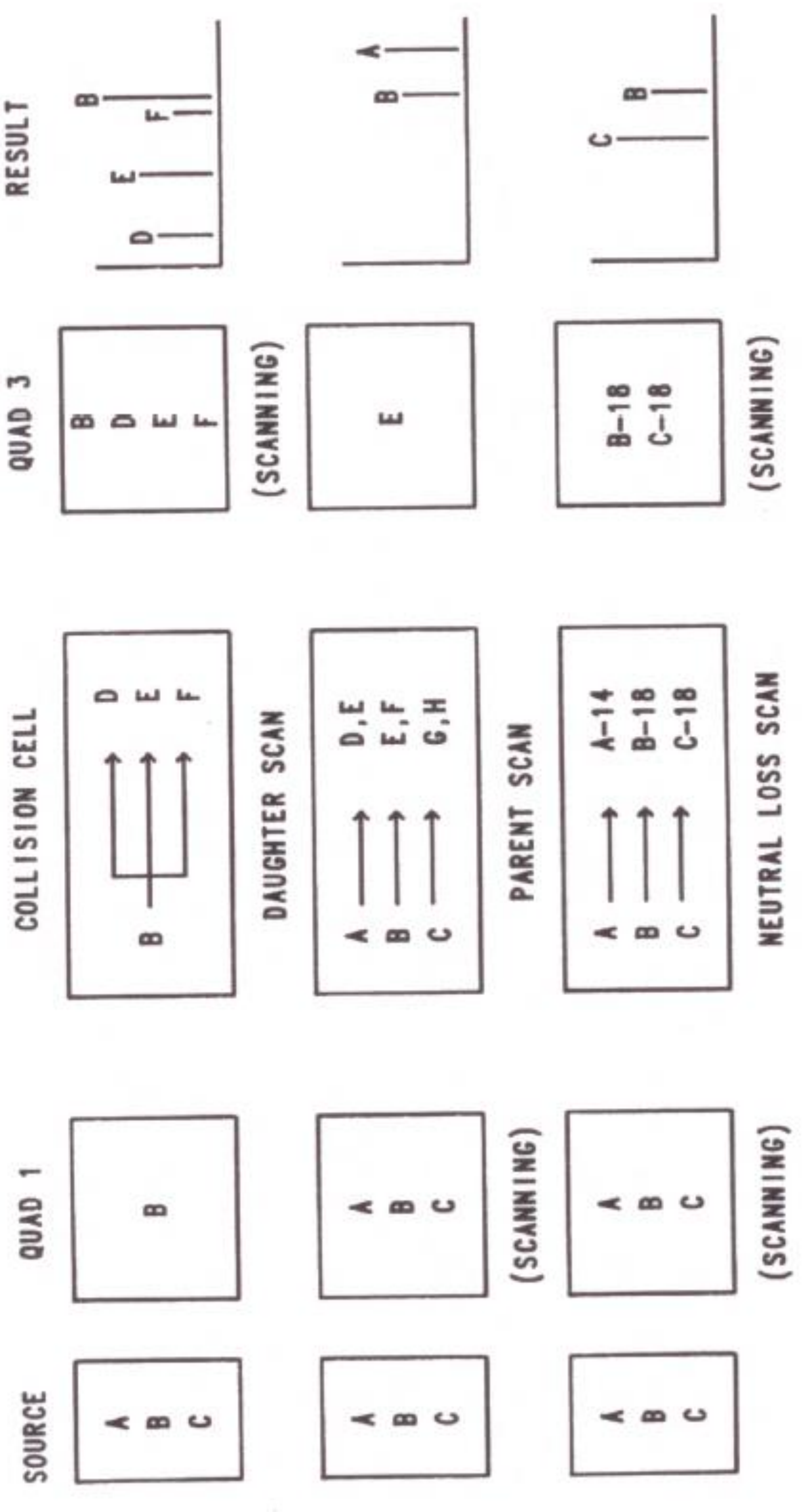


Figure 7.2 TQMS modes used in Mixture analysis

particularly useful during the investigative and developmental phase of a study such as described in this paper.

Parent Scan. This type of experiment generates a spectrum of all parent ions in the source that produce a selected daughter ion. Quadrupole three is set to transmit only ions of a selected mass from the collision cell. Quadrupole one is scanned over the desired mass range. All ions that undergo CAD to generate a daughter ion of the mass selected by quadrupole three are detected. The resulting spectrum consists of parent ion masses which fragment to form the selected daughter mass.

Neutral Loss Scan. Quadrupoles one and three are scanned in unison, but with a constant difference in selected mass. All ions that undergo a neutral loss equal to the mass offset of quadrupoles one and three will be detected. The result is a spectrum of parent ion masses which undergo the selected loss of neutral mass upon fragmentation.

The ability to perform several types of experiments on a sample with a single instrument makes TQMS an attractive option for rapid screening applications.

The complex hydrocarbon matrix of the middle distillate fuels makes the determination of low-level components difficult. The chance of finding unique molecular ion peaks in the mass spectrum of a raw fuel for each compound or class of compounds is almost negligible. The different types of experiments that can be performed with a triple quadrupole mass spectrometer offer several modes of component specificity other than the mass of the molecular ion. Characteristic combinations of parent and daughter ion masses can provide a high degree of selectivity for particular components. In addition, particular daughter ion or neutral loss masses may be highly indicative of certain compound classes.

The detection of selected aromatic and heteroatom species in a complex hydrocarbon matrix such as Jet A by mass spectrometry has generally required the separation of the aromatic and polar species from the aliphatic species that make up the bulk of the fuel. The goal of this project was to utilize the tremendous separatory power of the triple quadrupole mass spectrometer to develop screening procedures for selected aromatic and heteroatom species that do not require prior separation. The compound classes selected for investigation included the thiophenes, alkylbenzenes, pyridines, anilines, and the nitrobenzenes. The pyridines and thiophenes are of particular interest since their presence has been reported in middle distillate fuels and there is evidence that they have an adverse effect on the stability of these fuels[65],[66] ,[67] ,[68].

EXPERIMENTAL

Instrumental. The instrument employed in this study was a model ELQ-400-3 triple quadrupole mass spectrometer manufactured by Extranuclear Inc. of Pittsburg, PA. Electron impact (EI) ionization with 20 eV of electron energy was used throughout the course of this study. The ion source was operated at 100 C. Argon of 99.9% purity was used as the collision gas in the second quadrupole region for all collisionally-activated dissociation experiments. The argon pressure was maintained at 1.5 mTorr. Ion energies in the range of 10-20 eV were employed to induce fragmentation. Data were acquired using scanning rates between 100 and 250 amu/s.

Sample Introduction. An all-glass, batch type inlet system with a 500 ml expansion volume was used to introduce liquid samples into the mass spectrometer. Samples varying between 1-5 ul were injected into the inlet system by means of a syringe. Several minutes were allowed for the sample to completely vaporize before a valve on the inlet system was opened to introduce the sample into the ionization region of the mass spectrometer.

Chemicals. All pure compounds used in this study were purchased from either Chemservice Inc or the Aldrich Chemical Co. The Jet A, and shale oil derived fuel samples were obtained from NASA, Lewis Research Center, Cleveland, OH, and the diesel fuel sample was obtained from a local service station.

RESULTS AND DISCUSSION

In order to identify those fragmentation characteristics which may be useful for screening purposes, several compounds in each class were examined. The daughter scan mode of the TQMS instrument was used for this study to obtain the fragmentation spectra of the species of interest. The results of these fragmentation experiments are listed in Table 7.1. These results were then examined to identify characteristic CAD reactions for each class of compounds. It is desirable to have two characteristic reactions for a compound class to help ensure the accuracy of the screening procedure.

Examination of the CAD results for the thiophenes led to the identification of two reactions suitable for screening applications. One of these is the loss of 45, representing the loss of CHS from fragmentation of the thiophenic ring. The second screening reaction is the generation of a 97+ daughter ion by all of the substituted thiophenes. This leads to a screening procedure which includes a neutral loss scan for the loss of 45 as the primary screening reaction and a parent scan for the parents of 97+ daughter ions as a secondary screening reaction to confirm the presence of the thiophenes[69]. All of the thiophenes also generated a 45+ daughter ion. This fragment ion could be used as additional confirmation of the results obtained by the principal screening reactions outlined above.

The study of the alkylbenzenes yielded two characteristic daughter ions suitable for screening purposes. These daughter ions appear at 65 and 91 amu. The 65+ daughter ion is a result of fragmentation of the aromatic ring. The daughter ion at 91 amu is a result of the formation of the well known resonance-stabilized tropylium ion that is particularly characteristic of the alkylbenzenes[70].

Table 7.1 Daughters of Reference Compounds.

COMPOUND	PARENT ION	FRAGMENT IONS
Thiophene	84 (100.0)	69 (3.2), 58 (24.0), 45 (5.6), 39 (1.1)
2-Methylthiophene	98 (36.1)	97 (100.0), 69 (0.8), 54 (3.7), 53 (3.6), 45 (1.9), 39 (1.6)
2,5 Dimethylthiophene	112 (100.0)	97 (87.2), 67 (1.1), 53 (1.0), 45 (0.7)
2-Ethylthiophene	112 (11.7)	97 (100.0), 69 (0.7), 67 (0.4), 53 (4.2), 45 (1.1)
Benzene	78 (100.0)	77 (24.3), 76 (3.6), 63 (3.1), 52 (2.6), 51 (2.6), 50 (3.4)
Toluene	92 (37.2)	91 (100.0), 66 (1.9), 65 (8.6), 63 (1.0)
o-Xylene	106 (56.8)	91 (100.0), 79 (1.1), 78 (1.4), 77 (1.6) 65 (3.8)
m-Xylene	106 (46.7)	91 (100.0), 78 (1.4), 77 (1.4), 65 (2.9)
p-Xylene	106 (37.7)	91 (100.0), 79 (1.0), 78 (1.4). 77 (1.5) 65 (3.0)
Ethylbenzene	106 (17.6)	91 (100.0), 78 (2.4), 65 (3.1)
t-Butylbenzene	134 (0.9)	119 (100.0), 91 (52.9), 65 (1.2)
Nitrobenzene	123 (7.0)	93 (42.1), 77 (100.0), 65 (33.0)
m-Nitrotoluene	137 (12.0)	107 (16.2), 91 (100.0), 79 (12.2), 77 (8.8), 65 (9.4)
Ethylnitrobenzene	151 (12.0)	136 (2.7), 134 (8.5), 121 (20.4), 106 (1.5), 105 (100.0), 103 (23.4), 93 (21.3), 91 (14.7), 79 (9.2), 78 (4.6), 77 (15.5)
1-Octanethiol	146 (100.0)	112 (78.8), 84 (10.8), 83 (19.3), 82 (12.6), 70 (15.2)
1-Dodecanethiol	202 (87.9)	168 (100.0), 111 (12.4), 97 (27.9), 96 (27.4), 83 (27.4), 82 (21.5), 70 (5.0)
Indole	117 (100.0)	91 (1.0), 90 (49.1), 89 (34.2), 63 (1.3)
2-Methylindole	131 (86.8)	130 (100.0), 116 (1.5), 104 (4.2), 103 (8.0), 91 (1.8), 89 (1.5), 78 (2.8), 77 (6.3)
2,3-Dimethylindole	145 (100.0)	144 (73.4), 130 (52.6), 128 (5.0), 118 (3.1), 117 (2.4), 116 (2.6), 115 (7.2), 103 (3.2), 91 (2.4), 89 (1.1), 78 (1.3), 77 (5.2)
Pyridine	79 (100.0)	77 (23.6), 52 (22.7), 51 (2.6), 50 (2.2)
4-Methylpyridine	93 (100.0)	92 (24.3), 78 (3.6), 67 (18.1), 66 (30.6), 65 (17.9), 64 (1.7), 53 (1.3), 40 (2.1)
2,4-Dimethylpyridine	107 (100.0)	92 (10.2), 80 (4.4), 79 (15.6), 78 (5.3), 77 (5.3), 66 (2.2), 65 (3.4)
2,6-Dimethylpyridine	107 (100.0)	92 (10.8), 80 (2.2), 79 (8.3), 78 (2.3), 77 (3.6), 66 (12.0), 65 (5.0)
Aniline	93 (100.0)	92 (4.8), 78 (1.9), 77 (1.1), 76 (0.2), 67 (1.8), 66 (58.1), 65 (9.7), 54 (1.4)
o-Methylaniline	107 (100.0)	106 (94.1), 90 (1.0), 89 (2.5), 80 (2.8), 79 (5.9), 78 (3.0), 77 (6.0)
m-Methylaniline	107 (87.8)	106 (100.0), 90 (1.5), 80 (2.9), 80 (3.2), 79 (8.3), 78 (3.9), 77 (9.7)
2,4-Dimethylaniline	121 (100.0)	120 (62.1), 106 (91.7), 104 (1.9), 103 (2.4), 94 (1.2), 93 (1.5), 92 (2.0), 91 (2.5), 80 (1.1), 79 (5.0), 78 (2.8), 77 (8.6)
2,6-Dimethylaniline	121 (100.0)	120 (35.1), 106 (69.7), 104 (1.6), 103 (1.6), 94 (0.9), 93 (1.5), 92 (1.3), 91 (1.8), 80 (1.1), 79 (4.1), 78 (2.7), 77 (6.9)

The fragmentation results for the nitrobenzenes show that the loss of 46 and 30 is characteristic of this class of compounds. These correspond to the loss of NO2 and NO from the parent ion[71]. The loss of 46 seems particularly well suited as a primary screening characteristic, since in the CAD spectra of the pure compounds this loss generates the most intense ion.

The alkylthiols, octanethiol and dodecanethiol, exhibit a strong loss of 34, which corresponds to a loss of SH2. Since the thiol group is the only unique feature in the alkyl chain, only one screening reaction can be identified.

The indoles exhibit the loss of 27 and 54 as well as the formation of a 91+ daughter ion. The loss of 27 and 54 correspond to CHN and C4H6 respectively. Since the indole structure contains a benzene ring the formation of the 91+ daughter ion is expected.

Unique identification of the presence of the pyridines and/or anilines is made difficult because these two classes of compounds have the same molecular formulas. The major structural difference between the two compound classes is whether or not the nitrogen is incorporated into the aromatic ring structure. Both the anilines and the pyridines exhibit a neutral loss of 27 amu upon fragmentation. The loss of 27 (CHN) can be used as a primary screening reaction to locate nitrogen-containing aromatic species. The anilines undergo a loss of NH3 as indicated by the loss of 17 from the parent ions. This loss can be used to determine if any anilines are present at the masses detected by the loss of 27. If any loss of 17 is observed, anilines are present in the sample with pyridines possibly present. If no loss of 17 is observed the presence of pyridines is indicated by the loss of 27.

The indoles exhibit the loss of 27 and 54 as well as the formation of a 91+ daughter ion. The loss of 27 and 54 correspond to CHN and C_4H_6 respectively. Since the indole structure contains a benzene ring the formation of the 91+ daughter ion is expected. The loss of 27 can be used for screening for indoles in the presence of pyridines and anilines since the mass series for the indoles does not overlap the mass series of the pyridines and anilines.

Table 7.2 summarizes the screening reactions selected for the compound classes studied. Three types of fuels were examined in this study; Jet A a jet aviation fuel, a shale oil derived jet fuel, and No. 2 Diesel fuel. Mass spectra of raw Jet A and diesel fuel are shown in Figure 7.3. The complexity of the spectra illustrates the impracticality of attempting to use parent masses alone to identify trace components present in the fuels.

The general formula for the thiophenes is $C_nH_{2n-4}S$, forming a homologous mass series of 84, 98, 112, 126, 140, etc. The results of screening all three fuel type for substituted thiophenes are presented in Figure 7.4. These spectra show the presence of thiophenes with 1 to 7 carbons in side chains. General trends can also be observed in these results. In Jet A the thiophenes seem to be more highly substituted than in the other fuels. In the shale oil spectrum the thiophene distribution favors the low end of the mass range. Whereas

Table 7.2 Summary of Screening Reactions

NEUTRAL LOSS	COMPOUND CLASS
-17	Anilines
-27	Anilines, Pyridines, Indoles
-30	Nitrobenzenes
-34	Alkylthiols
-45	Thiophenes
-46	Nitrobenzenes
-54	Indoles

DAUGHTER ION	COMPOUND CLASS
65+	Alkylbenzenes
91+	Alkylbenzenes
97+	Thiophenes

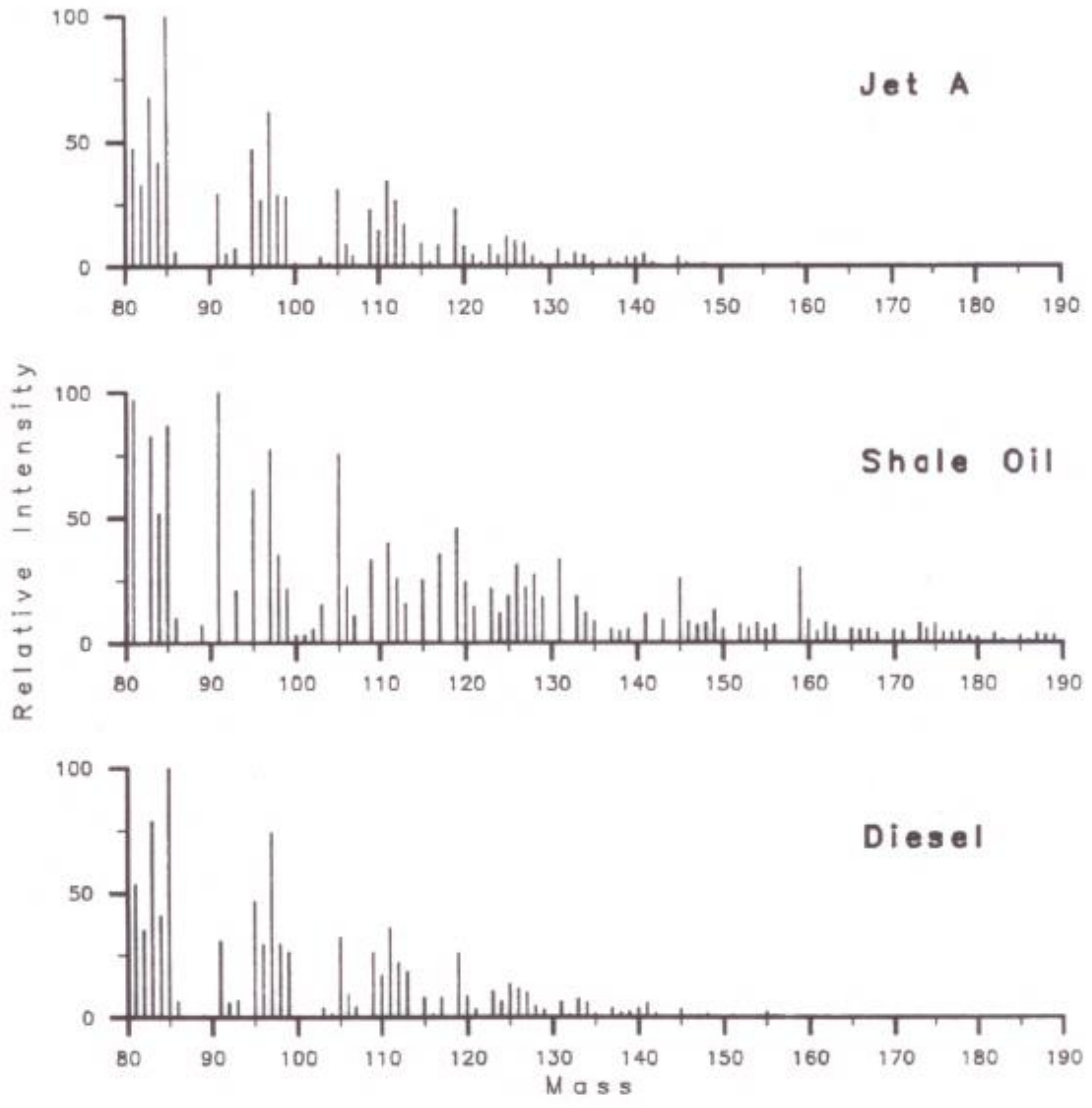


Figure 7.3 Raw MS of Jet A, Shale Oil and Diesel fuel

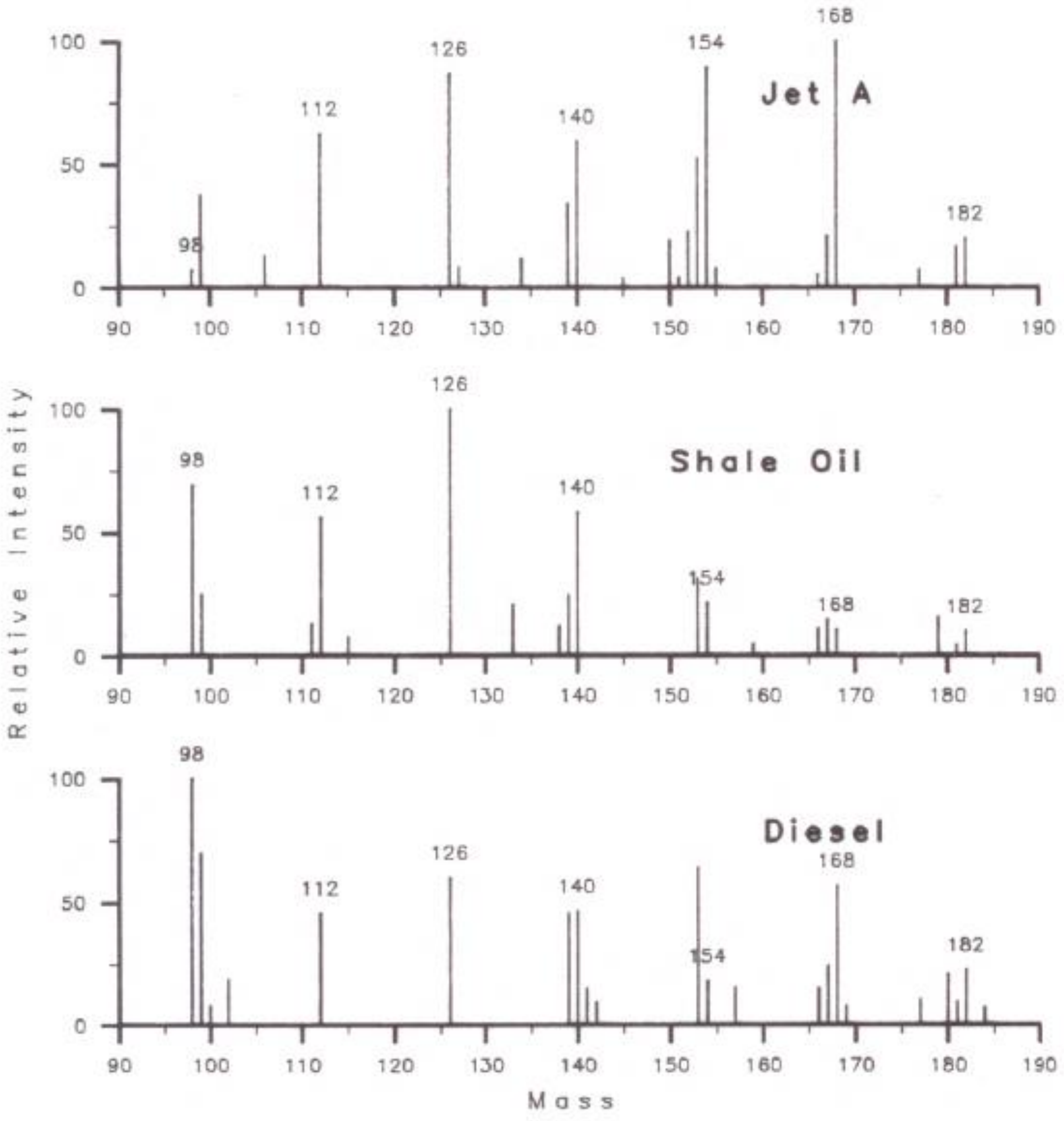


Figure 7.4 Parents of 97+ for Jet A, Shale Oil, Diesel

in the case of the diesel the thiophene distribution seems to be fairly even over the entire mass range.

Figure 7.5 presents the results of screening for the alkylbenzenes that form the homologous mass series 92, 106, 120, 134, 148, etc. The presence of alkylbenzenes with up to 6 carbons in side chains (mass 162) is indicated in Jet A and the shale oil sample where the maximum is 5 carbons (mass 148) in the diesel fuel.

The results of screening for the presence of indoles by observing the loss of 54 are show in Figure 7.6. In none of the fuel samples was the presence of indole (117) indicated. The presence of at least three indoles in Jet A is indicated by the peaks at masses 131, 145, and 159. Two indole peaks appear in the spectra for shale oil (145, 159) and diesel fuel (131, 145).

Figure 7.7 compares the results of screening for the nitrobenzenes in Jet A and diesel fuel. The Jet A appears to have two nitrobenzene components at 123 and 137 amu. Diesel fuel has three nitrobenzenes as indicated by the peaks at 123, 151, and 165 amu. No nitrobenzenes were observed in the shale oil fuel. On the subject of negative results, no alkylthiols were sucessfully detected in any of the fuel samples.

Finally, Figure 7.8 presents the results of screening the shale oil derived fuel for the pyridines and anilines. The spectrum of the loss of 27 indicates the possible presence of pyridines or anilines at masses 107, 121, 135, and 149. The only mass the spectrum of the

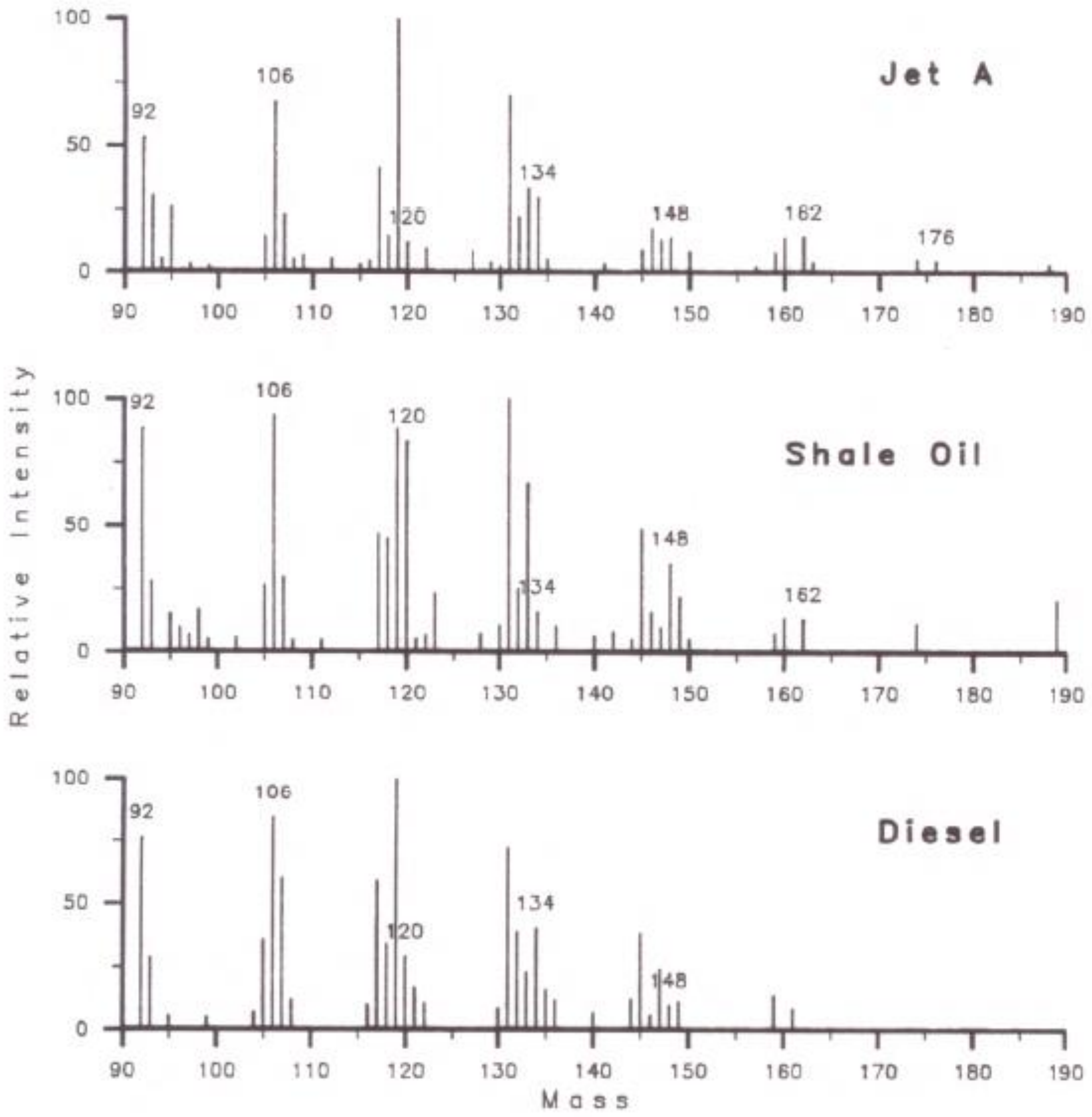


Figure 7.5 Parents of 91+ for Jet A, Shale Oil, Diesel

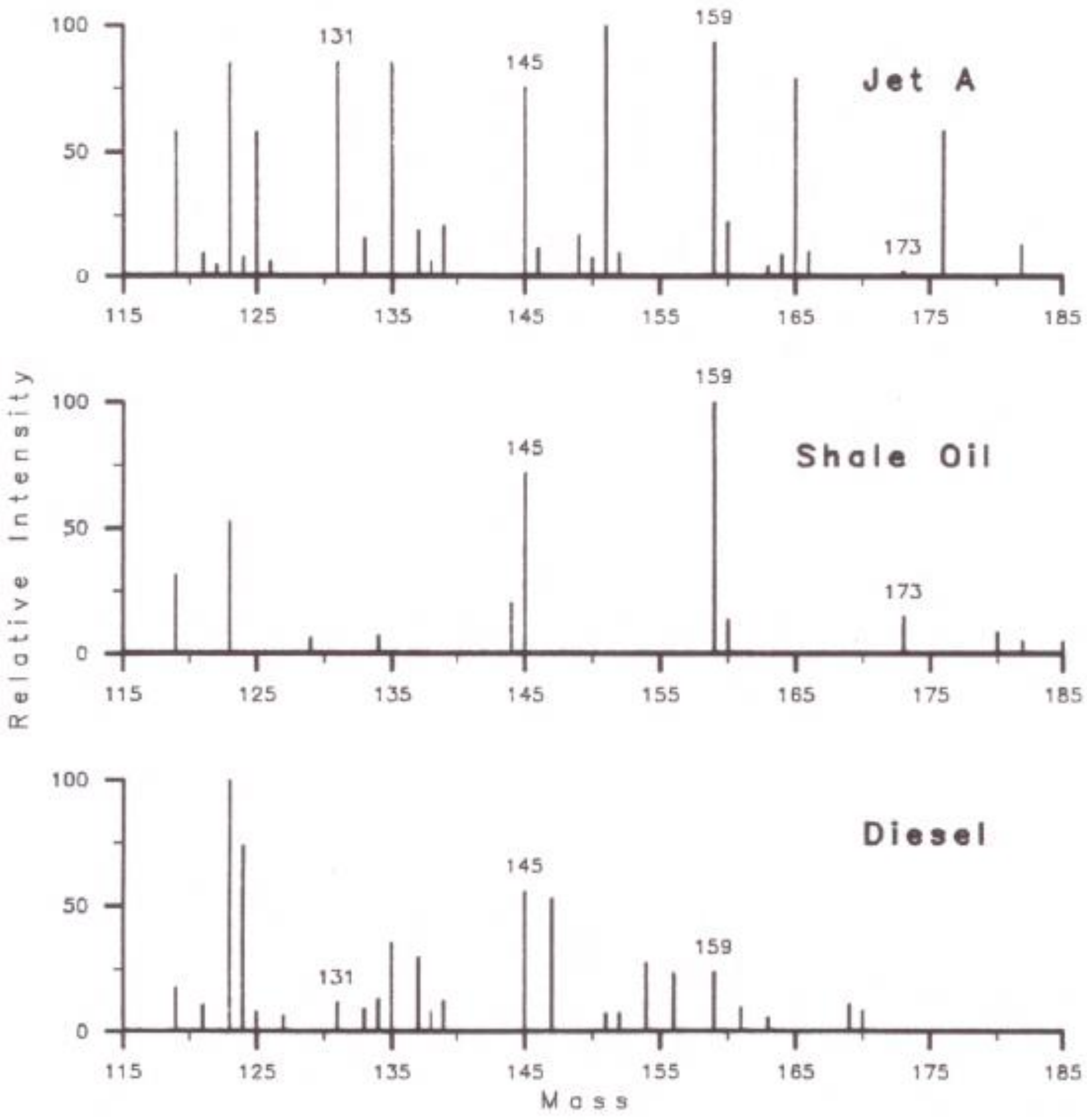


Figure 7.6 Loss of 54 from Jet A, Shale Oil, Diesel

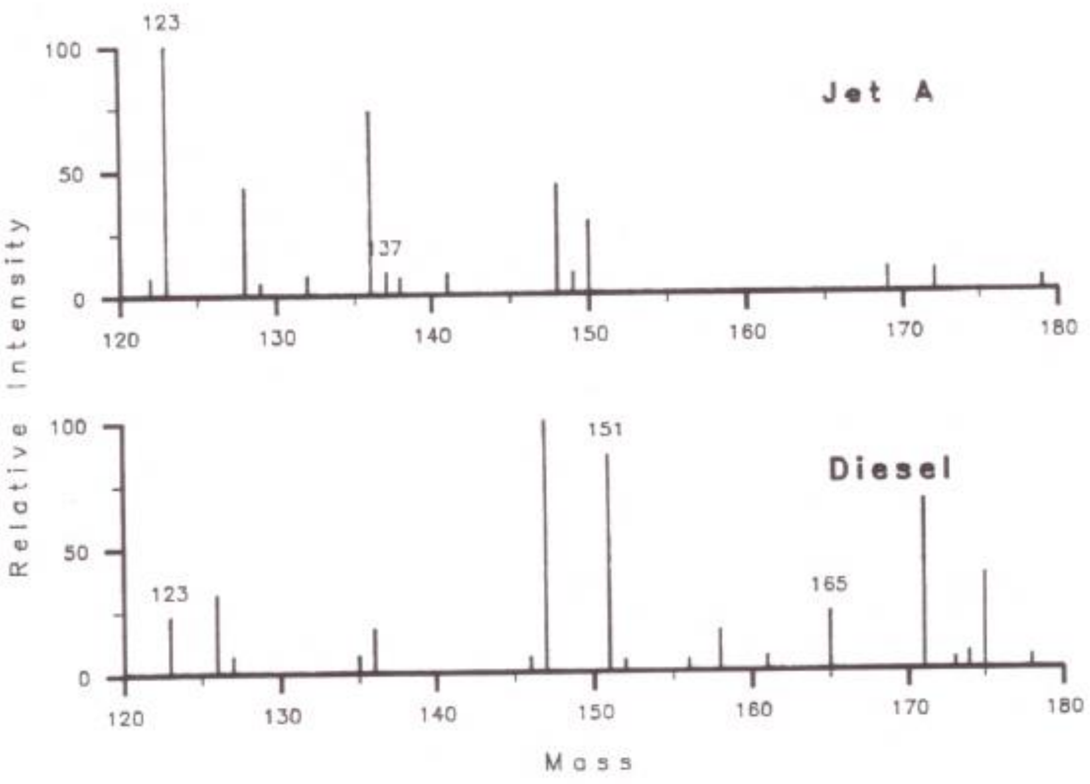


Figure 7.7 Loss of 46 from Jet A, Diesel

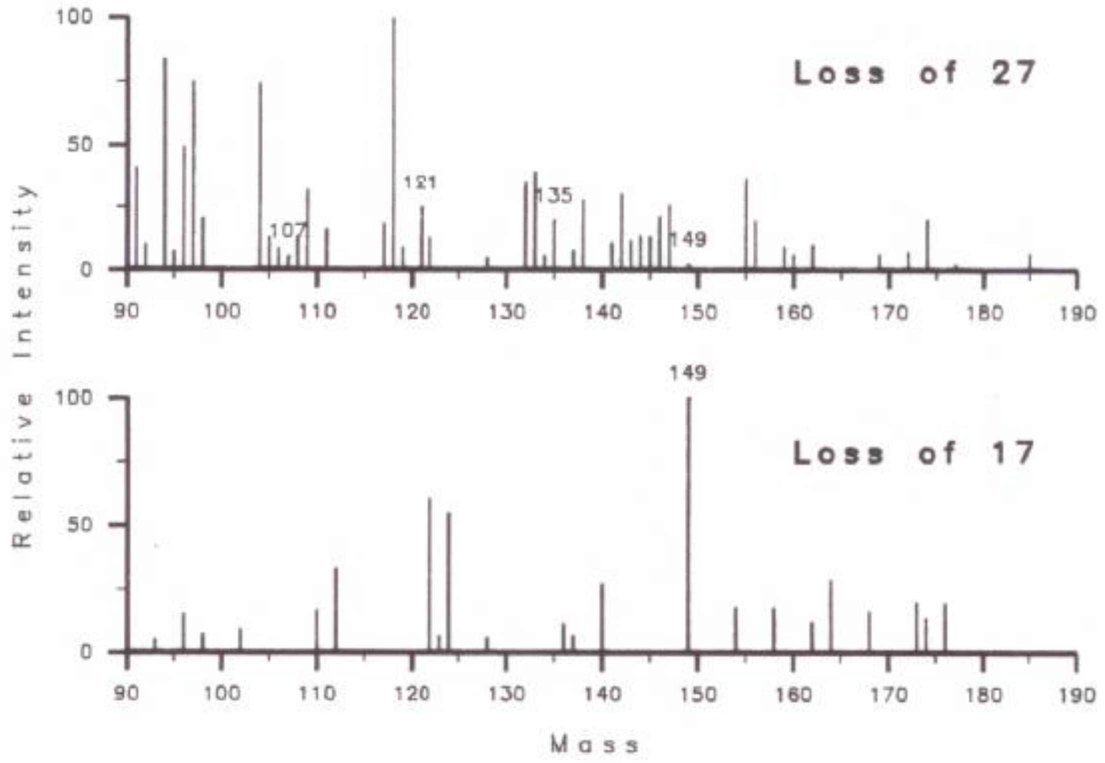


Figure 7.8 Loss of 27 and 17 from Shale Oil

loss of 17 (used to screen for anilines) has in common with this list, is 149. This suggests that there are at least the pyridines and one aniline to be found in the sample.

These results demonstrate the ability of TQMS to detect selected species in hydrocarbon fuels. Although no attempt was made to quantify these results, the ability to readily detect single ring thiophenes that are present in these types of fuels at below the 10 parts per thousand range gives a rough indication of the sensitivity of the technique. The use of internal standards such as isotopically labeled species or standard addition experiments are effective approaches for quantitation by TQMS when necessary. Another useful approach for quantitation would be to use multiple reaction monitoring (MRM), the TQMS analog of multiple ion monitoring in GC/MS, to monitor the ion currents for the reactions of interest and then use the area of the chromatographic peaks for quantitation as in GC/MS.

CONCLUSIONS

The separatory power of a TQMS instrument permits the detection of trace components in fuels without prior sample work-up or additional chromatographic techniques. Sample handling and analysis time can be reduced to about 5 minutes because of the ability of TQMS to analyze fuel samples directly. Since each scan takes only a few seconds to complete, screening for many compound classes can be accomplished in a very short time. The ability of a triple quadrupole mass spectrometer to detect minor components in such complex mixtures can make it a valuable tool in fuel stability studies.

ACKNOWLEDGMENTS

The authors would like to thank the National Aeronautics and Space Administration for financial support of this project, and the people at Extranuclear Inc. for their assistance and use of their equipment.

REFERENCES

[59] Dahlin. K.E.; Daniel, S.R.; Worstell, J.H. Fuel 1981, 60, 477-480

[60] Tutubalina,V.P.; Gabdrakhmanov, F.G.; Korotkova, E.G. Deposited Doc. 1980, SPSTL 460Khp-D80

[61] Hazlett, R.N.; Hall, J.M. Prepr. - Am. Chem. Soc., Div. Pet. Chem. 1981, 26(2), 613-619

[62] Jones, L.; Hazlett, R.N.; Li, N.C.; Ge, J. Preprint Am. Chem. Soc. Fuels Div. 1983, 28(1), 196

[63] Yost, R.A., Enke, C.G., McGilvery, D.C., Morrision, J.D., Int. J. of Mass Spectrom. Ion Phys, 30, 127 (1979)

[64] Yost, R.A., Enke, C.G., American Lab, June 1981

[65] Bol'shakov,G.F., Izv. Vyssh. Uchebn. Zaved., Neft Gaz 1976 19(5), 51-3

[66] Chertkov, Y.B., Chem. Technol. Fuels Oils, (1976),154

[67] Worstell, J.H., Daniel, S.R., Fuel, 1981, 60, 481-4

[68] Wenzel, B., Aiken, R.L., J. Chrom. Sci., 1979, 17, 503-9

[69] Myerholtz, C.A.; Enke, C.G. Am. Soc. Mass. Spec. 30th Conference Honolulu, 1982

[70] McLafferty, F.W. "Interpretation of Mass Spectra third edition"; Turro, N.J. Ed.; University Science Books: Mill Valley,CA, 1980, p.187

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

Chapter 8 : COMMENTS AND SUGGESTIONS

Short terms goals in the instrumentation area could focus on consolidating the gains made in our laboratory over the past several years. A number of advanced capabilities such as low-cost graphics, floating point coprocessor support, Winchester disk support, distributed processing capability, and the softknobs, have been made available for use in the laboratory. To date, only the TQMS control system takes advantage of these capabilities. The low-cost hardware that is available needs to be utilized to provide more graphics and data reduction functions on control systems. The greater and more immediate the feedback of experimental results to the operator the more efficiently a series of experiments can be performed.

In the area of hardware development two projects might be considered for future development. One is an interprocessor module similar to the status module featuring a greater amount (1K) of dual-port memory. This would be helpful since in a distributed system there are a number of pieces of information that all the processors need to know. A larger dual-port memory store would simplify the dissemination of this type of information and reduce redundancy in the system. The second project would be the development of a CPU module based on the 68008 microprocessor chip. The 68008 provides all of the features of a 68000, except it utilizes only an 8-bit data bus instead of a 16-bit data bus. This processor allows large memory spaces to be accessed more easily than the 8088 processor currently being used. This would allow less-expert programmers to develop larger applications which could include more functions to aid the operator.

In order for triple quadrupole mass spectrometry to gain more widespread acceptance and use as an analytic tool, more practical applications need to be demonstrated in the literature. The fuel analysis capabilities demonstrated in chapters 6 and 7 provide a fertile ground for new applications. These types of applications are of particular interest to the petroleum companies which are among the largest users of mass spectrometry.

To continue to expand the applications of TQMS to fuel studies, screening procedures for additional compound types need to be investigated. Compound classes of interest include the pyrroles, furans, benzofurans, quinolines, tetrahydroquinolines, tetralins (tetrahydronapthalenes), and indans. Members of these compound classes are illustrated in Figure 8.1. The furans, pyrroles and quinolines are of particular interest since they have been shown to contribute to deposit formation in jet fuels.

72

If these studies are undertaken, the screening reactions could be used with GC/MS/MS to attempt to develop methods of quantifying the species present and identifying specific isomers. The screening and quantification methods could then be applied to fuel samples before and after thermal stressing. The deposit formation results obtained from experiments run on a jet fuel thermal oxidation tester (JFTOT) can then be correlated with the detailed composition of the fuel obtained by the TQMS techniques to identify the species and mechanisms involved in deposit formation.

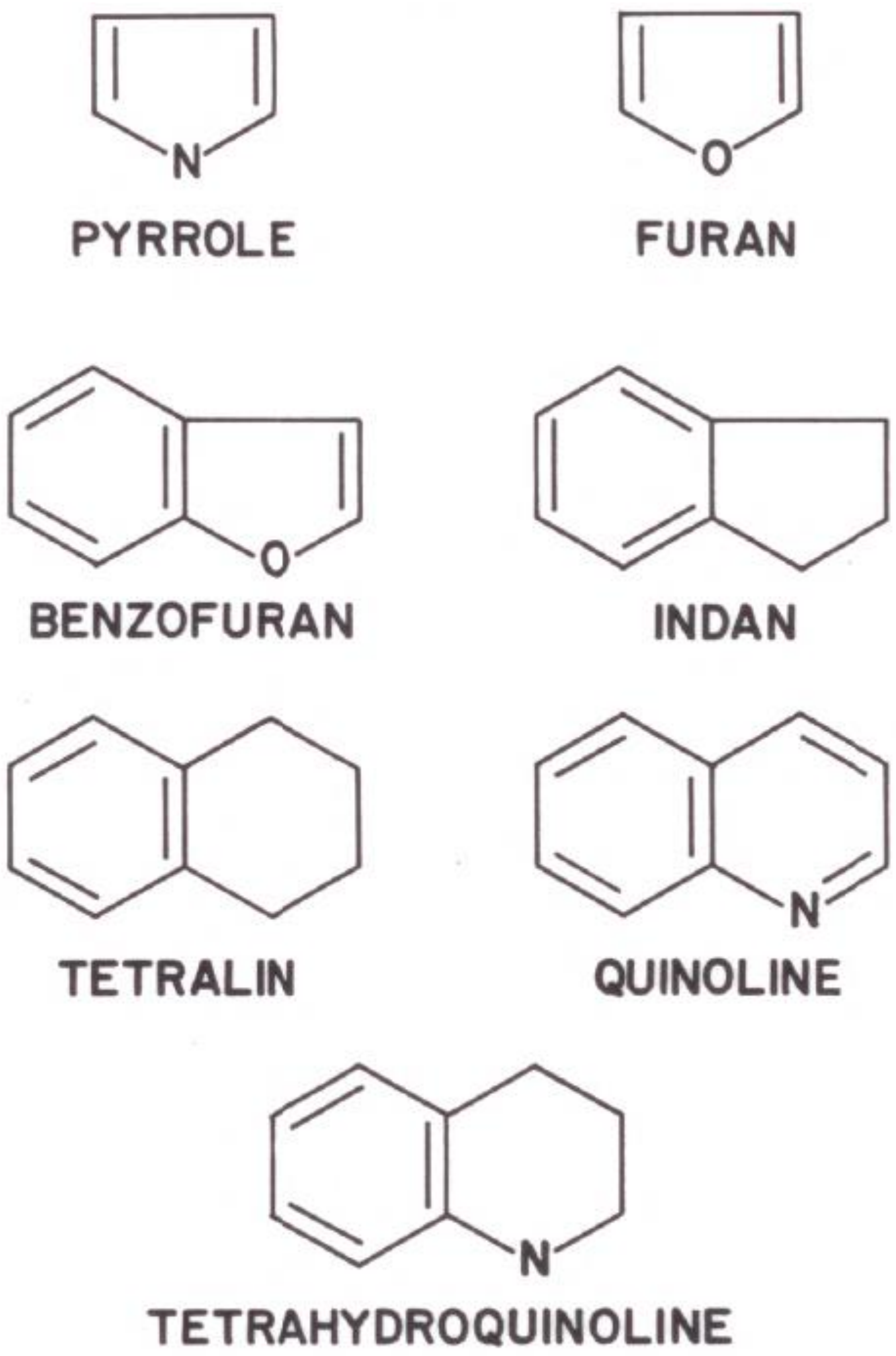


Figure 8.1 Compound Classes for Future Study

An area of potential application of TQMS to fuels not directly related to stability studies is fuel contamination. Leakage between storage areas, especially in naval vessels can cause relatively clean fuels such as aviation fuels to be contaminated by dirtier fuels such as diesel fuel and bunker oils. Studies of these dirtier fuels should lead to the identification of species not found in jet fuels. These species could then be screened for in the jet fuels to detect any fuel contamination.

In the process of the author's work several interesting fragmentation patterns were observed. Since the author's work already covered several diverse topics no fragmentation studies were undertaken. These observations will be stated briefly for whomever might find them interesting to investigate. The first, the fragmentation of benzenethiol produced the following ion intensities; 110 (100.0), 84 (4.4), 66 (6.7). The interesting fragmentation is the loss of 26 (C2H2) from the parent to generate the 84+ daughter ion. The expected loss of 33 (SH) or 34 (SH2) was not observed. The 84+ daughter ion is suggestive of a thiophene ring, indicating that the benzene ring may have ruptured lost C2H2 and formed a thiophene ring. The 84+ ion also formed in the source could be fragmented to see if it is indeed a thiophene ring. If so, this leads to the question of why formation of a thiophene ring is favored so strongly over losing the SH group and leaving the highly stable benzene ring structure intact.

The second area that might yield interesting information on fragmentation mechanisms is the observation that the methyl and dimethyl substituted pyridines form a 92+ daughter ion upon CAD. It is possible that this ion may be a pyridine analog of the tropylium ion so commonly found among the alklybenzenes.

Finally, a few closing comments, remember "No guts, No glory". If you are unfortunate enough to have read this entire dissertation, I would like to bestow a curse upon you "May you live in interesting times". May the Force be with you.

Worstell, J.H., Daniel, S.R., Fuel, 1981, 60, 481-4.

APPENDIX A

ENKE TRIPLE QUADRUPOLE MASS SPECTROMETER

OPERATOR'S MANUAL

VERSION 1.0

September 13, 1983

Carl Myerholtz

Table of Contents

DEVICE PARAMETERS

The computer has control over a great number of the devices that affect instrument performance. There are a variety of methods available to the user to set up these parameters, but first a discussion of their organization is in order. The user may configure up to sixteen sets of parameters at one time. These sets of parameters are stored on the disk and can be identified with a 64-character title. Only one parameter set can be active at a time. This is accomplished by loading a selected set into memory. In addition, each set of parameters is divided into four sections referred to as modes. These mode sections contain complete sets of values for all device for different ionization modes. These are identified as EI, +CI, -CI, and USR corresponding to electron impact, positive chemical ionization, negative chemical ionization, and a spare user ionization mode. Thus, in a given parameter set, values can be set for all devices for each different ionization mode available. Once a parameter set is in memory it is possible to switch between the device settings for the different ionization modes very rapidly without having to access the disk again. Each device in the system has four parameters associated with it. These are referred to as the CURRENT, START, END, and STEP values. The CURRENT value is the value to which the device is presently set to and to which it is returned to if the device is scanned. The remaining values control the device when it is scanned. The STEP value is the step size the value of a device is incremented as it is scanned from the START value to the END value.

PARAMETER STORAGE AND RETRIEVAL

As mentioned above up to sixteen sets of parameters can be stored on disk, they are numbered 0 to 15. Before a set can become active it must be loaded into memory. This section deals with moving parameter sets between disk and memory.

PDIR - PARAMETER DIRECTORY

This command displays a list of the titles assigned to the 16 parameter sets.

PGET - GET PARAMETERS

n PGET - Moves parameter set n into memory overwriting the set that is in memory. All the devices are updated to the new values.

PSAVE - PARAMETER SAVE

n PSAVE - Stores the current parameter settings residing in memory into parameter set n on the disk. None of the device settings are affected.

TUNESAVE

n TUNESAVE - copies the CURRENT value of the device settings for the currently selected ionization mode into the same ionization mode of parameter set n on the disk.

ALLSAVE

Copies the CURRENT value of the device settings for the currently selected ionization mode into the settings for the same ionization mode of all the parameter sets on the disk.

IONIZATION MODE SELECTION

Whenever a new ionization mode is selected out of the parameter set currently in memory all device values are updated and the source and detection electronic are put in the proper mode if they are under computer control.

EI

Selects the electron impact ionization mode.

+CI

Selects the positive chemical ionization mode.

-CI

Selects the negative chemical ionization mode.

USR

Selects the user ionization mode. Currently this turns all filaments off.

PARAMETER EDITOR

The parameter editor allows the user to modify any of the device parameters of the parameter set in memory using a screen oriented editor. The changes go into effect as soon as the editor is exited. The changes do not affect the setting of the parameter set on the disk unless the appropriate PSAVE command is issued.

PED

PED - activates the parameter editor. The editor displays the parameter set number and title of the active parameter set transferred into memory with the PGET command, the ionization mode selection when the editor was invoked, and the values of all the device parameters. Note that PED always returns to the ionization mode that was in effect when it was invoked. A cursor, which is highlighted in reverse video, will be placed at the first entry in the parameter set. The arrow keys on the terminal are used to move the cursor to the parameter value to be modified. A return entered alone will cause the cursor to move the CURRENT value entry of the next device in the list. To change the value on which the cursor is positioned, type in the new value followed by a return. In addition to the arrow keys the parameter editor responds to several single character commands as follows:

M - TOGGLE MODE

This command toggles the parameter editor between the values for the different ionization modes. Each time M is entered it toggles the editor to the next ionization mode in a circular list that follows the order EI, +CI, -CI, USER, EI, etc.

T - ENTER TITLE

This command erases the current title associated with the parameter set and moves the cursor to the title line to allow a new title to be entered. A new title may be up to 64 characters in length with spaces allowed and is terminated with a return.

C - COPY

Copies the values at the current position into a single entry buffer.

I - INSERT

Inserts the value in the buffer into the value at the current cursor position.

R - REPAINT

Redisplays the entire screen.

Q - QUIT

Exit from the parameter editor.

.PED - DISPLAY PARAMETERS

Displays all of the parameter settings for the current ionization mode without entering the parameter editor. Often used in conjunction with the PRINT command.

STAT - PARAMETER STATUS

Equivalent to .PED.

SETTING SINGLE PARAMETERS

The following four commands all the user to set one of the parameters for a device without entering the parameter editor. To work properly the device whose value is to be modified must first be selected by entering it two or three letter mnemonic name. For example, to change lens three to 12.3 volts one would enter "L3 12.3 SET" or "12.3 L3 SET". Note that the device name may be entered before or after the numeric value. The only requirement is that it appears before the command word.

SET

n SET - sets the current value of the selected device to n.

!START

n !START - sets the start value of the selected device to n.

!END

n !END - sets the end value of the selected device to n.

!STEP

n !STEP - sets the step value of the selected device to n.

SETTING QUAD DC/RF MODES

DC

n DC - sets quad n in the DC mode.

RF

n RF - sets quad n in the RF-only mode.

RRD - RF/RF/DC

Places quads 1 and 2 in the RF-only mode and quad 3 in the DC/RF mode.

DRR - DC/RF/RF

Places quad 1 in the DC/RF mode and quads 2 and 3 in the RF-only mode.

DRD - DC/RF/DC

Places quads 1 and 3 in the DC/RF mode and quad 2 in the RF-only mode.

MISC

MS1 - SET MASS QUAD 1

n MS1 - sets the mass selected by quad one to n. Equivalent to "n M1 SET".

MS3 - SET MASS QUAD 3

n MS3 - sets the mass selected by quad three to n. Equivalent to "n M3 SET".

LOSS - SET NEUTRAL LOSS

n LOSS - sets up for a neutral loss of n. Start mass for quad three is set equal to (start mass quad one) - n.

DEVINIT - INITIALIZE DEVICES

Sets all devices to their current value in the parameter table. This is automatically done after each PGET.

DATA FILE OPERATIONS

The software system maintains a large file (4,192,000 bytes) on the Winchester disk for data storage. This file is divided into 65500 records, each 64 bytes long. All the data acquired by the system is stored in this file. A distinction is made between two types of data in the file. These are SCANS and EXPERIMENTS. A SCAN is a set of data collected with one of the commands discussed in the next chapter. Each time a data SCAN is acquired it is written into the data file. An EXPERIMENT is a set of sequentially acquired scans, hopefully, but not necessarily related. Since a great number of scans may be written on the disk EXPERIMENT records can be written to group scans together. Each EXPERIMENT has a name up to 16 characters in length (spaces may be included). After data has been acquired, data can be retrieved from any experiment. However new data is always stored only in the last experiment on the disk. The various commands used to examine, select, and enter EXPERIMENT information in the data file are discussed below.

?DISK - CHECK DISK SPACE

Displays the number of records currently in use in the data file and the number of records still free.

EDIR - EXPERIMENT DIRECTORY

Experiment Directory - Lists all of the experiments in the data file along with the time and date they were created, the number of scans, and the total number of records used in the experiment. When displayed on the terminal EDIR will pause each time the screen is full. Strike the space bar to continue the experiment directory or the return key to leave the directory.

SELECT - SELECT EXPERIMENT

SELECT "experiment name" - Selects an existing experiment as the experiment from which to retrieve data. No matter which experiment is selected new data will always be added to the last experiment on the disk.

?EXPT - DISPLAY EXPERIMENT NAME

Displays the name and header information for the currently selected experiment.

EXPT - ENTER A NEW EXPERIMENT

EXPT "name" - defines a new experiment called "name" on the disk. The name of an experiment may be up to 16 characters and may contain any character except "\". All data acquired subsequently to the EXPT command will be entered into this new experiment. The new experiment will also be made the current experiment for any of the display commands.

DELETE - DELETE AN EXPERIMENT

DELETE "experiment name" - deletes all the data in the specified experiment from the disk.

NOTES

Along with other information in the header for an experiment is space or the user to enter four 64-character lines of notes about the experiment, his or hers love life etc. Notes may be entered into the experiment selected as the current experiment at any time. Thus, notes may be entered at the start, or end of an experiment or days later when you think up an excuse for the poor results. Each note line is entered by a separate command, and can be written over by repeating the command. The form of the note commands is as follows.

```
1NOTE This is the text for note one.
2NOTE This is the text for note two.
3NOTE This is the text for note three.
4NOTE This is the text for note four.
```

?NOTES - DISPLAY NOTES

?NOTES displays all of the notes for the current experiment.

SDIR - SCAN DIRECTORY

Scan Directory - lists the header information for all of the scans in the current experiment. This header information includes the scan type, range of the scan, mass settings of quads 1 and 3, ionization mode, number of data points acquired and the Total Ion Current measured in the scan. When a Scan Directory is displayed on the terminal it will pause each time the screen is full. Strike the space bar to continue, or a return to leave the directory.

SS - SELECT SCAN

n SS - Selects Scan n of the current experiment as the current scan for display and reporting purposes.

?SCAN - DISPLAY CURRENT SCAN

?SCAN - Displays the header information for the currently selected scan.

INITIALIZE

INITIALIZE - removes all data files from the winchester disk.

DATA ACQUISITION

Data acquisition operations can be broken down into two basic types. Those that perform peak finding while scanning a parameter and those that do not. All data acquisition operations automatically write the data to the disk when the operation is complete. The number of data points that a single scan can obtain is limited to 1024. The ion current measured by the system is expressed as a number between 0 and 1,048,575, giving the data acquisition system a dynamic range of six orders of magnitude.

ACQUISITION PARAMETERS

RATE

n RATE - sets the rate at which the scan is performed, where n specifies one of the scanning rates given in the table below.

RATE	POINTS/SEC	TIME/POINT	AMU/SEC (0.1 amu steps)
0	10,000	100 usec	1,000
1	5,000	200	500
2	2,500	400	250
3	1,000	1 msec	100
4	500	2	50
5	250	4	25
6	100	10	10
7	50	20	5
8	25	40	2.5
9	10	100	1
10	5	200	0.5
11	2.5	400	0.25
12	1	1 sec	0.10
13	0.5	2	0.05
14	0.25	4	0.025

As the scanning rate is decreased the number of times the ion current is sampled before an average value is obtained for a given data point is increased. Thus, as the scanning rate is lessened the signal to noise ratio of the data improves. This rate parameter controls the scanning rate of all data acquisition, those that perform peak finding and those that do not.

PEAK FINDING PARAMETERS.

There are three parameters that control the performance of the peak finding algorithm. These are threshold, minimum width, and maximum width. Currently there are no provisions for saving and recalling these parameters from the disk. The user is advised to check these parameters before performing any data acquisition functions.

THRESHOLD

The threshold is the value which the ion current must become greater than before a peak can be recognized. It may have any value in the range of 0 to 65535.

MINIMUM WIDTH

This parameter defines the minimum acceptable peak width in terms of step size. That is if it is set to two and the step size is 0.1 amu than the minimum acceptable peak width is 0.2 amu. The main function of this parameter is to filter out narrow noise spikes.

MAXIMUM WIDTH

This parameter the maximum width of the peak in terms of step size. If the ion current has not returned to below threshold by this point the peak is terminated and the software beings searching for the isotope peak. To indicate that a peak exceeded the maximum width value and was abnormally terminated the flags value for that peak is set to one.

ASET - SET ACQUISITION PARAMETERS

ASET first displays a table of rate setting similar to the one given earlier. ASET then displays the values all four acquisition parameters. ASET will then display a ? next to each parameter in sequence. To change a parameter type in a new value, to leave it unchanged just strike return.

.ASET - DISPLAY ACQUISITION PARAMETERS

Displays the same information as ASET without asking the user to change any of the values.

!THRESHOLD

n !THRESHOLD - set the threshold parameter to n.

!PWIDTH

n !PWDITH - set the minimum peak width parameter to n.

!MWIDTH

n !MWIDTH - set the maximum peak width parameter to n.

MASS SCANNING

The control system allows the user to scan the mass filters of the TQMS in five different ways: mass scan of quad one, mass scan of quad three, parent, daughter, and neutral loss scans. These modes are abbreviated as 1, 3, P, D, and N respectively.

1SCAN - QUAD ONE SCAN

Sets the instrument into the DC/RF/RF configuration and then scans quad one from its START to END values while performing peak finding.

3SCAN - QUAD THREE SCAN

Sets the instrument into the RF/RF/DC configuration and then scans quad three form it's START to END values while performing peak finding.

PSCAN - PARENT SCAN

Sets the instrument into the DC/RF/DC configuration. Sets quad three to its CURRENT value and scans quad one from its START to END values while performing peak finding.

DSCAN - DAUGHTER SCAN

Sets the instrument into the DC/RF/DC configuration. Sets quad one to it's CURRENT value and scans quad three from its START to END values while performing peak finding.

NSCAN - NEUTRAL LOSS SCAN

Sets the instrument into the DC/RF/DC configuration. It then scans quads one and three together. Both start at their respective START values and the scan proceeds with peak finding until quad one reaches its END value. The step size is controlled by the STEP value for quad one.

1SCANS

n 1SCANS - performs 1SCAN n times. n must be in the range 0-32767. This causes n quad one scans to be sequentially acquired and recorded on the disk.

3SCANS

n 3SCANS - performs 3SCAN n times. n must be in the range 0-32767. This causes n quad three scans to be sequentially acquired and recorded on the disk.

PSCANS

n PSCANS - performs PSCAN n times. n must be in the range 0-32767. This causes n parent scans to be sequentially acquired and recorded on the disk.

DSCANS

n DSCANS - performs DSCAN n times. n must be in the range 0-32767. This causes n daughter scans to be sequentially acquired and recorded on the disk.

NSCANS

n NSCANS - performs NSCAN n times. n must be in the range 0-32767. This causes n neutral loss scans to be sequentially acquired and recorded on the disk.

PARAMETER SCANNING

The computer system can perform a type of data acquisition scan called a SWEEP for any device under its control. In a sweep a data point is recorded each time the current value of the selected device is incremented by its step size. The total number of data points taken is given by (END -START)/STEP and must not exceed 1024.

SWEEP

device SWEEP - scans the specified "device" from its START value to its END value recording the ion current each time the value of the device is incremented by STEP.

OSCILLOSCOPE DISPLAY FUNCTIONS

The mass spectrometer system includes a display oscilloscope that enables the user to view the output of the detector preamplifier in real time. The X axis of the oscilloscope is driven by the computer and normally represents a mass axis. The computer can be used to select one of five gain factors to apply to the output of the preamplifier before displaying the ion current on the oscilloscope. The principle use of the oscilloscope is to give the user an immediate visual feedback when tuning up the instrument.

SCOPE - OSCILLOSCOPE GAIN CONTROL

n SCOPE - selects one on the five gains to apply to the ion current from the table below.

n	GAIN FACTOR
0	256
1	64
2	16
3	4
4	1

FAST SCANNING FUNCTIONS

Fast scanning functions perform scanning operations for display on the oscilloscope. No data acquisition or storage is performed. Fast scanning functions operated as a background task, that is once started they continue to display data on the oscilloscope until specifically turned off or another fast scanning function is started. The terminal will remain active while a fast scanning function is in operation. The fast scanning operations use the same START, END and STEP values that are used when scanning with data acquisition.

FSTOP - STOP OSCILLOSCOPE DISPLAY

Halts any display on the oscilloscope screen.

FSPEED

n FSPEED - controls the scanning speed of all oscilloscope display functions. n is an arbitrary value between 1 and 32767. The larger the value of n the slower the scanning speed.

F1SCAN

Performs a quad one scan repeatedly for oscilloscope display.

F3SCAN

Performs a quad three scan repeatedly for oscilloscope display.

FPSCAN

Performs a parent scan repeatedly for oscilloscope display.

FDSCAN

Performs a daughter scan repeatedly for oscilloscope display.

FNSCAN

Performs a neutral loss scan repeatedly for oscilloscope display.

SPLIT SCREEN OPERATIONS

Split screen functions allow the user to display up to five peaks side by side on the display oscilloscope. Split screen displays are a background task like the fast scanning functions and once started continue to display on the oscilloscope while the terminal remains active. There are five basic split screen operations corresponding to the five standard mass scanning modes (quad1, quad3, parent, daughter, and neutral loss). The user may assign between one and five masses to each split screen mode. In addition, the scope gain function to be used when displaying a given mass may also be specified. When activated the system will display a five amu region centered around each specified mass using the specified gain factor. Note that when a parent, daughter, or neutral loss split is activated the appropriate values for parent ions and neutral losses are extracted from the currently active parameter set as describe in the mass scanning section earlier. To set up all of these parameters an interactive routine called SPLITS is used.

SPLITS - SPLIT SCREEN SETUP

SPLITS activates a split screen setup editor. It displays the center mass and gain function for the five display window for each of the five scanning modes along with a brief menu of commands. In addition, a sixth set of parameters is display for a TUNE mode that will be explained later. A reverse video cursor will appear at the first enter for the quad 1 scan mode. This cursor can be moved to the different mass windows displayed on the screen using the arrow keys. To change the mass window selected by the cursor, enter the new value followed by a return. When a split screen display is activated it starts by displaying mass window 1, then mass window 2 and so on until it encounters a mass window whose value is set to zero. Thus, by setting mass window 4 to zero for quad 1 scanning mode only the first three mass windows will be displayed. In addition to the arrow keys SPLITS recognizes ten single character commands.

Z - ENTER ZERO

Enter a zero for the mass window selected by the current cursor position.

G - GO

Activate the split screen display for the scan mode specified by the line the cursor currently is located on. When a scan mode is being actively displayed on the scope an asterisk is displayed next to the mode name to indicate which mode is active.

A - AMPLIFICATION TOGGLE

Toggle the gain factor for the mass window currently selected by the cursor. Each time A is pressed the gain will be advanced one through the sequence 256, 64, 16, 4, 1.

< - SHIFT LEFT

Increments the value in the mass window currently selected by the cursor 0.2 amu causing the peak to shift to the left on the display. This command will start the split screen display running if a Go command hasn't already been issued.

> - SHIFT RIGHT

Same as the shift left command above except that it decrements by 0.2 amu causing the peak to move to the right.

P - PARENT

Allows the user to specify a new parent ion to be used for the daughter scan display.

D -DAUGHTER

Allows the user to specify a new daughter ion to be used for the parent scan display.

M - IONIZATION MODE TOGGLE

The ionization mode currently in effect is displayed at the upper righthand corner of the split screen display. This command toggles the system to the next ionization mode in the circular list EI, +CI, -CI, USR. The selected ionization mode remains in effect after exiting SPLITS.

T - SELECT TUNE VALUES

There is a sixth set of mass windows labeled TUNE. When a T is entered a split screen display is started for the scan mode specified by the line the cursor is currently on. However instead of using the mass windows specified for that mode the mass windows specified on the tune line are used. An asterisk is then displayed next to the active scan mode and the on the TUNE line to indicate which scan mode is active and that it is using the TUNE parameters. The primary use of this function is to keep a set of mass windows in the TUNE set for a standard reference compound, thus making it easy for the user to quickly check on known reference peaks.

Q - QUIT

Exits from the SPLITS command.

The SPLITS command is very powerful and somewhat complex. The easiest way to understand its operation is to experiment with it for a while using a familiar reference compound.

.SPLITS - PRINT SPLIT SETTINGS

Generates the same display as SPLITS without entering the interactive mode.

SSAVE - SPLITS SAVE

Copy the currently active splits settings into a reserved storage area on the disk.

SGET - SPLITS GET

Copy the splits setting from the disk area into memory making them active. Most often used to recall a set of default settings from the disk.

1SPLIT

Activate a quad 1 split screen display using the parameters setup with the SPLITS command.

3SPLIT

Activate a quad 3 split screen display using the parameters setup with the SPLITS command.

PSPLIT

Activate a parent split screen display using the parameters setup with the SPLITS command.

DSPLIT

Activate a daughter split screen display using the parameters setup with the SPLITS command.

NSPLIT

Activate a neutral loss split screen display using the parameters setup with the SPLITS command.

XSCANS

As an additional tuning aid there is a special class of fast scanning operations called XSCANS. They operate similarly to FSCANS. However instead of utilizing the STEP value specified in the parameter table the mass scanning DACs are incremented by one. These scans operated 3 to 5 times slower than fast scans and are primarily intended to be used to examine small mass ranges in great detail.

X1SCAN

Performs a quad one xscan repeatedly for oscilloscope display.

X3SCAN

Performs a quad three xscan repeatedly for oscilloscope display.

XPSCAN

Performs a parent xscan repeatedly for oscilloscope display.

XDSCAN

Performs a daughter xscan repeatedly for oscilloscope display.

XNSCAN

Performs a neutral loss xscan repeatedly for oscilloscope display.

SOFT KNOBS

In order to place the numerous computer controlled devices under a more convenient form of control for the user a set of four knobs have been supplied. These four knobs may be connected by the computer to any device in the system.

KNOBS - ACTIVATE SOFTKNOBS

This command activates the softknobs. It displays ten sets of knob definitions, that is which knob is assigned to which device. These knob definitions are numbered 0 - 9. To select a knob definition enter its number. The device assignments and values for each knob will then be updated to reflect the new definition. The values of the devices the knobs control are continuously updated on the terminal display. To exit from the knobs display enter a Q. Note that the softknobs will have no effect on the system unless the KNOBS command is active.

KSET - DEFINE A KNOB SET

m KSET dev1 dev2 dev3 dev4 - defines a new set of devices for knobs definition n. Dev1 is assigned to knob one, dev2 is assigned to knob two and so on. The names of four device must always be entered. If a knob is to be inactive in a definition assign it to the NUL device. Example: 3 KSET Q1 Q2 Q3NUL causes knobs definition three to assign Q1, Q2 ,Q3 to knobs one, two, and three respectively. Knob four is assigned to the NUL device and is thus inactive.

.KNOBS - DISPLAY KNOBS DEFINITIONS

.KNOBS - displays the knob definitions without activating the knobs.

KSAVE

Saves the current set of knobs definitions so that they can be recovered if changes are made.

KGET

Recovers the previously saved set of knob definitions.

DISPLAY FUNCTIONS

All display functions use the last scan acquired or the scan selected as the current scan by the SS command from the current experiment specified by the SELECT command. The data in a scan can be displayed in two forms: numeric or graphic. The DLIST command generates a numeric display. The DISP command generates a graphical display. In the graphical displays, the data is normalized. That is the top edge of display is equivalent to the largest point in the data set. The raw value of the point is always displayed at the top righthand edge of the plot. The intensity axis of the plot may be on either a linear or a logarithmic scale.

DLIST - DATA LIST

Displays the x value (mass or voltage), raw intensity, normalized intensity, and flag values for each data point in the scan. A flag setting of 1 indicates the mass peak width was greater than the maximum limit in effect at the time of acquisition. DLIST will pause each time the terminal screen becomes full. Strike the space bar to continue the display or hit return to exit from DLIST.

LIN - LINEAR DISPLAY

Selects a linear intensity scale for graphical displays. This is indicated on a plot by tic marks on the y axis at 25%, 50%, 75%, and 100%.

LOG - LOGRITHMIC DISPLAY

Selects a logarithmic intensity scale for graphical displays. This scale covers three orders of magnitude indicated by the tic marks at 3.000, 2.000, 1.000.

DISP - DISPLAY

Generates a graphical display of the data in the scan. For voltage sweeps, a single box will be displayed, and the data will be plotted as a continuous curve. For a mass spectrum the x axis will be divided up into segments no more than 200 amu long (maximum of 5). The data will then be plotted as a histogram. The user can over-ride the automatic formatting of mass spectral data using the commands described below.

DSET - SET DISPLAY PARAMETERS

There are three parameters that control the format mass spectral information is displayed. Start mass - is the first mass at which to start displaying the spectrum. Amu/field - is the number of amu to display in each box plotted on the screen. #Fields - is the number of boxes into which the mass spectrum is divided. Each one of these values has a status associated with it. Either Auto or Preset. In the Auto mode the computer selects the most optimal value for the parameter. In the Preset mode the user specifies the values the computer is to use. DSET will display the settings of all of the parameters then ask the user if he want to change each one by displaying a ? next to the value. A response of -1 sets a parameter into the auto mode, a return leaves the parameter unchanged, and entering a new value sets the parameter to that value. Remember that the first two values are in terms of mass and must be entered including the tenths place. i.e 30.0 not 30 for a 30 amu setting.

.DSET - DISPLAY DISPLAY PARAMETERS

Displays the settings of the display parameters.

OPLOT - OVER PLOT

Displays the current scan in graphical form without erasing the display or plotting axis. This command is useful for comparing data by causing one scan to be displayed on top of another. Note that all data displayed with OPLOT is normalized to the maximum value of the first data set plotted.

DTIC - DISPLAY TIC

n1 n2 DTIC - plots the total ion current (TIC) for scans n1 thru n2. Only the TIC for scans of the same type as scan n1 will be plotted.

DATA MANIPULATION

ADD

n1 n2 ADD - add scans n1 and n2 together and create a new scan. All masses will be rounded to the nearest nominal mass. The new scan will be appended to the end of the last experiment.

SUM

n1 n2 SUM - sums all scans between n1 and n2 of the same type as scan n1. All masses are rounded to the nearest nominal mass. A new scan is created and added to the end of the last experiment.

SUB

n1 n2 SUB - subtract scan n2 from scan n1. All masses are rounded to the nearest nominal mass. The new scan of the difference between n1 and n2 is added to the end of the last experiment. Any negative peak intensities are set to zero.

METHODS AND SEQUENCES

METHODS

A METHOD is a series of instructions that are stored on the disk and can be executed with a single command. A method consists of 16 lines of text up to 64 characters in length. Up to 50 methods may be defined, numbered 0-49. Methods can be nested, that is one method can initiate the execution of another method. However, under no circumstances should a method try to execute itself. Death and destruction will surely follow. Any command that can be entered from the terminal can be used in a method. However, any text strings such as experiment names must be terminated with the \ character. Comments may also be included in a method. Comments are used only for reference and identification purposes and do not affect the execution of a method. Comments enclosed in { } are printed out each time the method is executed and comments enclosed in ( ) are ignored. When used the ( ) and { } must be separated from any text by at least one space. A useful command when printing text from a method is CR which causes the terminal to start printing on a new line. There are three command used to create, display, and execute methods all of which must be proceeded by a method number. These are MED, MLIST, and METHOD.

MED - METHOD EDITOR

n MED - Selects method n and activates the method editor. The editor displays the contents of the method with the method number at the top of the screen. The contents of the method can be changed with a series of editor commands. These commands are entered and scroll by on only the four lower lines of the terminal screen. The contents of the methods are always displayed on the upper portion of the screen. Any changes will appear in the method listing immediately. Most of the editor commands are single letters, all of the commands are terminated with a return.

METHOD EDITOR COMMANDS

n T - Select line n as the current line.

P text - Put text on current line, replacing any existing text.

U text - Put text on line Under the current line moving all other lines down. The last line is lost.

X - Delete the current line moving all remaining lines up.

F text - Find the first occurrence of text starting from the current cursor position. Can be repeated by just typing F.

E - Erases the last text string found with the F command.

R text - Replace the last string found with the F command with text.

D text - Delete the next occurrence of text.

I text - Insert text at the current cursor position. Any text pushed off the end of a line is lost.

TILL text - Deletes all text on a line from the current cursor position through and including the string text.

WIPE - Erase all the text in the method.

Q - Quit. Exit the method editor.

MLIST - METHOD LIST

n MLIST - displays the text for method n.

MCOPY - METHOD COPY

n1 n2 MCOPY - copies method n1 into method n2.

MDIR - METHOD DIRECTORY

MDIR - displays the first line of all 50 methods. It is considered good form to make the first line of a method non-printing comment to identify the function of the method. When MDIR is displayed on the terminal it will pause each time the screen is full. Strike the space bar to continue the method directory or a return to abort the directory.

METHOD

n METHOD - Causes method n to be executed.

SEQUENCES

A sequence is a series of methods each of which is repeated for specified number of seconds before proceeding to the next method. The primary application of sequences is GC/MS experiments. Up to 16 sequences can be defined at one time, numbered 0-15. A sequence can contain up 16 methods. Each method can be executed from 1 to 65535 seconds. Sequences can be nested, that is for example sequence A may contain a method that causes the execution of sequence B. One should use caution when nesting sequences and methods that a recursive system is not developed. A recursive system is one that ends up executing itself over and over again until world war three or the computer system crashes. If less than 16 methods are to be executed in a sequence the duration of the method following the last method to be executed must be set to zero.

SED - SEQUENCE EDITOR

n SED -Display sequence n and activates the sequence editor. Use the arrow keys to move the cursor to the method number or duration you wish to change. Enter the new value followed by a return. Enter a Q to exit the editor.

SLIST - SEQUENCE LIST

n SLIST - Displays the method numbers and durations for sequence n.

SEQUENCE

n SEQUENCE - Causes sequence n to be executed. Execution of a sequence can be terminated after the completion of a method by entering a Q. The current method be finished before the Q takes effect.

MULTIPLE REACTION MONITORING

MASS CALIBRATION

Quads one and three each have an interpolation table used to convert mass values (amu) to the digital-to-analog (DAC) values which are used to control the quadrupole power supplies. These tables contain up to sixteen masses and their corresponding DAC values. The control system uses these entries and a linear interpolation algorithm to make mass assignments. To set up these interpolation tables the user must enter a list of calibrations masses to be used and then perform the calibration function while a reference compound is in the mass spectrometer.

CALIBRATION MASSES

The user may store five sets of calibration masses on the disk. The calibration masses are not entered into either interpolation table until the CALIBRATE functions is performed.

CGET

n CGET - Make calibration mass set n active by loading it from disk into memory.

CSAVE

n CSAVE - Store the active calibration masses into set n on the disk.

.CAL - DISPLAY CALIBRATION MASSES

.CAL - displays the active calibration masses.

CALSET

CALSET - Allows the user to enter a new list of calibration masses. This new list becomes the active set of calibration masses. Each time a "?" is displayed a new calibration mass can be entered. To enter less than 16 masses enter a 0 after the last desired mass. When the entry of new values is completed the list of new calibration masses is displayed.

INTERPOLATION TABLE OPERATIONS

LINCAL

n LINCAL - forces a linear calibration into the interpolation tables for quads one and three. The value n is the upper mass limit of quadrupole control electronics in use.

.ITABLE - DISPLAY INTERPOLATION TABLE

n .ITABLE - Displays the mass and dac entries in the interpolation table for quad n, where n can be 1 or 3.

ISAVE

ISAVE - stores both interpolation tables on the disk. This is used to save a calibration so that it is not lost when the computer is turned off or reloaded.

IGET

IGET - recovers the interpolation table values stored on the disk with the ISAVE command.

CALIBRATE

n CALIBRATE - calibrates quad n (1 or 3) by the following procedure. A calibration mass is selected and the system pauses for the user to adjust the multiplier gain if necessary. A mass window 4 amu wide is scanned over the peak ten times. After each scan the DAC value on intensity of the peak is displayed. An average DAC valued is then displayed and the user has a choice of accepting or rejecting the value. If rejected the averaging step is repeated. If accepted the system proceeds to the next calibration mass. Note that the calibration scans are performed at the currently selected scan rate. For optimal calibrations the scanning rate should be the same for the calibration and the data to be acquired.

FLOPPY DISK OPERATIONS

The floppy disk drive attached to the system can used to store parameter set, methods and other user configured information. Information that is normally stored on the fixed disk can be transfer to or from the floppy disk by the commands described in this chapter. The information on the floppy disk cannot be accessed directly, it must first be transferred back to the fixed disk. All of the transfer commands are prefixed with a } or { to indicate the direction of transfer. } indicates a transfer from the fixed disk to the floppy, while { indicate a transfer from the floppy to the fixed disk.

}SPLITS

}SPLITS - transfers the splits settings stored on the fixed disk with the SSAVE command to the floppy disk.

{SPLITS)

}SPLITS - transfers the splits settings from the floppy disk to the reserved storage area on the fixed disk. These setting do not become active until a SGET command is issued.

}KNOBS

}KNOBS - transfers the current knob set definitions to the floppy disk.

{KNOBS

{KNOBS - transfers a set of knob definitions from the floppy replacing the current set of knob definitions.

}ITABLE

}ITABLE - transfer the calibration interpolation table stored on the fixed disk with th ISAVE command to the floppy disk.

{ITABLE

{ITABLE - transfers the calibration interpolation table from the floppy disk to the reserved storage area on the fixed disk. These calibration values are not active until an IGET command is issued.

}PARAM

n }PARAM - transfers parameter set n to the floppy disk.

{PARAM

n {PARAM - transfers parameter set n from the floppy to the fixed disk. The parameter set is not active until a PGET command is issued for that parameter set.

}PARAMS

}PARAMS - transfers all 16 parameter sets to the floppy.

{PARAMS

{PARAMS - transfers all 16 parameter sets from the floppy to the fixed disk.

}METHOD

n }METHOD - transfer method n from the fixed disk to the floppy disk.

{METHOD

n }METHOD - transfer method n from the floppy to the fixed disk.

}METHODS

n1 n2 }METHODS - transfer methods n1 thru n2 to the floppy disk.

{METHODS

n1 n2 {METHODS - transfer methods n1 thru n2 from the floppy to the fixed disk.

MISC

HELP

Online help is supported in the following manner. Entering the command HELP displays instructions on how to use the help facility. HELP "word" displays help information for the function "word". To find out what words help recognizes type HELP WORDS.

TRANSMISSION

n TRANSMISSION - computes the transmission of the instrument at mass n. This is done by setting quads 1 and 3 to the desired mass and measuring the ion current in the DC/RF/DC mode. Then quad 1 is set to 50% of quad 3 and the ion current is measured in the RF/RF/DC MODE. The ratio of the DC/RF/DC ion current to the RF/RF/DC ion current is computed and displayed. An example of this command follows.

```
219.0 TRANSMISSION

   21876    89678  TRANSMISSION AT MASS 219.0 = 24.3 %
```

The first number is the ion current measured in the DC/RF/DC mode and the second number is the ion current in the RF/RF/DC mode. If either of these number falls below 2000 or above 1,000,000 the calculated value for the trans mission may be unreliable.

PTATRANS

PTATRANS - checks the transmission at masses 69.0, 131.0, 219.0, and 502.0. This is useful when perflurotributlyamine is being used as a calibration compound.

Notes on Migration to MS Word

For easy of archiving the original text was migrated into MS Word in February 2021. The migration attempted to preserve the look and feel of the original document while not creating excess effort.

The original text was written for a Text Formatter called ReadiWriter that used commands embedded into simple text to direct the formatting. The formatting was some what specific to the printer being used at the time, an Epson MX series dot-matrix printer that had only one font size along with bold, italics and underlining highlights. 11pt Courier New font was used as being fairly close to the original font. The only additional highlight used was a slight increase in the font size of some headings.

There were a surprising number of typos and other small errors in the text, many of which were corrected. This document is not meant to be an exact reproduction of the original, rather it is an attempt to preserve the original information and look and feel. No effort was made to clean up or enhance the scanned figures.