

FILE COPY

4

ADDC-TR-88-218  
Final Technical Report  
October 1988



AD-A206 974

# DEPTH PROFILES AND BULK ANALYSIS OF SEMICONDUCTOR MATERIALS USING ICP MASS SPECTROSCOPY WITH ELECTROTHERMAL ATOMIZATION

GeoChemical Services, Inc.

William Faulkner, William Henderson and Michael Rogers

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

DTIC  
ELECTE  
APR 18 1989  
S E D

ROME AIR DEVELOPMENT CENTER  
Air Force Systems Command  
Griffiss Air Force Base, NY 13441-5700

89 4 17 093

This report has been reviewed by the RADC Public Affairs Division (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-88-218 has been reviewed and is approved for publication.

APPROVED:

*David W. Weyburne*

DAVID W. WEYBURNE  
Project Engineer

APPROVED:

*Harold Roth*

HAROLD ROTH  
Director of Solid State Sciences

FOR THE COMMANDER:

*John A. Ritz*

JOHN A. RITZ  
Directorate of Plans & Programs

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (ESME) Hanscom AFB MA 01731-5000. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notice on a specific document requires that it be returned.

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
1a. REPORT SECURITY CLASSIFICATION <b>UNCLASSIFIED</b>		1b. RESTRICTIVE MARKINGS N/A			
2a. SECURITY CLASSIFICATION AUTHORITY N/A		3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited.			
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE N/A					
4. PERFORMING ORGANIZATION REPORT NUMBER(S) N/A		5. MONITORING ORGANIZATION REPORT NUMBER(S) RADC-TR-88-218			
6a. NAME OF PERFORMING ORGANIZATION GeoChemical Services, Inc.		6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Rome Air Development Center (ESME)		
6c. ADDRESS (City, State, and ZIP Code) 2741 Toledo Street, Suite 201 Torrance CA 90503		7b. ADDRESS (City, State, and ZIP Code) Hanscom AFB MA 01731-5000			
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Rome Air Development Center		8b. OFFICE SYMBOL (If applicable) ESME	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F19628-86-C-0122		
8c. ADDRESS (City, State, and ZIP Code) Hanscom AFB MA 01731-5000		10. SOURCE OF FUNDING NUMBERS			
		PROGRAM ELEMENT NO 65502F	PROJECT NO 3005	TASK NO RA	WORK UNIT ACCESSION NO 52
11. TITLE (Include Security Classification) DEPTH PROFILES AND BULK ANALYSIS OF SEMICONDUCTOR MATERIALS USING ICP MASS SPECTROSCOPY WITH ELECTROTHERMAL ATOMIZATION					
12. PERSONAL AUTHOR(S) William Faulkner, William Henderson and Michael Rogers					
13a. TYPE OF REPORT Final		13b. TIME COVERED FROM Jul 86 TO Mar 87		14. DATE OF REPORT (Year, Month, Day) October 1988	15. PAGE COUNT 92
16. SUPPLEMENTARY NOTATION N/A					
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB-GROUP	Semiconductor Analysis , Electrothermal Atomization , Depth Profile , Inductively Coupled Plasma , ICP Mass Spectrometry ,		
14	02				
09	01				
19. ABSTRACT (Continue on reverse if necessary and identify by block number) The purpose of the contract was to build the equipment necessary to show technical feasibility of a Demand Modulated Electrothermal Atomization System. This system was thought to have advantages over current technology as follows:  1. Prevents excessive analyte concentrations in the plasma giving the analyst control over matrix suppression effects. 2. Allows the analyst to control atomization rates and avoid buildup of deposits in the throat of the sample cone opening. 3. Allows the analyst to work in the optimum counting range for isotopic ratio work regardless of concentration variations. 4. Allows the data to be taken over the temperature dimension, thus resolving isobaric interferences as well as improving the signal to noise ratio resulting in improved detection limits across the entire mass range.					
20. DISTRIBUTION / AVAILABILITY OF ABSTRACT <input checked="" type="checkbox"/> UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS RPT. <input type="checkbox"/> DTIC USERS			21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED		
22a. NAME OF RESPONSIBLE INDIVIDUAL David W. Weyburne			22b. TELEPHONE (include Area Code) (617) 377-4015	22c. OFFICE SYMBOL RADC (ESME)	

①

UNCLASSIFIED

5. Allows the routine use of high dissolved solids (>10%) as well as solid sample introduction.

6. Allows the routine use of small sample sizes - 100 ugms or less.

PERFORMANCE AGAINST PHASE ONE TECHNICAL OBJECTIVES:

Item #1. Software was created to collect and analyze data in three dimensions-AMU, concentration and atomization temperature. This software gives the analyst program control over a comprehensive list of variables. These are as follows:

1. Channels per sweep and dwell time.
2. Default maximum sweeps per summation buffer.
3. Default minimum sweeps per temperature step.
4. Default maximum sweeps per temperature step.
5. Default number of pulses per temperature step.
6. Maximum temperature increments before shutoff.
7. Number of initial sweeps to throw away.
8. Starting and ending temperature.
9. Number of sweeps per temperature increment.
10. Size of temperature increments.
11. Number of sweeps to be stored in each data histogram.
12. Starting and ending amu for scans.
13. Starting and ending amu for threshold control.

See software manual for detailed description.

Item #2. Atomization conditions for direct solid sample introduction were not accomplished due to the complexity and time constraints of the contract. Solid samples were introduced into the plasma under stable conditions and data was taken. Detailed methods development will be needed to determine valid parameters for specific analyses.

Item #3. Solid samples were not analyzed per se, however, analysis of 10% dissolved solids was accomplished. These results were presented at the 1987 Pacific Conference on Chemistry and Spectroscopy in Irvine, California.

Isobaric Interferences were temperature resolved for Cd and Pd at amu 106 and Hg and Pb at amu 204.

Item #4. VG Isotopes was not able to get the instrument to perform as specified. The background was high and the responses were low resulting in detection limit performance for the standard instrument that was three orders of magnitude high. For this reason, no detection limit studies were performed.

Item #5. The software and hardware were designed to accommodate computer control of various gases during an analysis. Specific studies were not accomplished because the instrument in use for this program had limited usable count rates of two million per second. VG's updated design (electronic) will now give up to ten million counts per second.

Item #6. Time did not allow this item to be completed.

Although some of the planned items did not get accomplished, the equipment and software built under this contract has demonstrated the ability to perform these tasks with additional work on specific methods for each type of analysis.

UNCLASSIFIED

TABLE OF CONTENTS

	Page No
Introduction	1
Executive Summary	2
Problem with Standard Instrument	5
Plasma Vacuum Interface	5
Detector Preamp Issues	6
Memory	6
Hardware/Software Development	8
Development of the Tantalum Furnace	8
Development of the Furnace Controller	11
Development of Controlling Software	11
Data Acquisition	12
Software Manual	20
Analytical Performance	29
Addendum	45
System Changes Since Contract Completion	45
Software Code	47



Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By _____	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

## INTRODUCTION

The goal of this contract was to refine GSI's concept of applying feedback control to an electrothermal sample introduction system as applied to a commercially available mass spectrometer manufactured by VG Isotopes. The potential applications of this technology would be

- 1 The bulk trace element analysis of semiconductor materials such as silicon and gallium arsenide
- 2 The bulk analysis of other materials such as acids, solvents, dopants and source materials
- 3 The ability to do semiconductor device depth profiles.

The unique features of this concept are:

- 1 Increased sensitivity
- 2 Resolving concentrations over a third dimension, temperature, eliminates interferences
- 3 Analysis of high dissolved solids (>10%)
- 4 Solid sample introduction.
- 5 Sample to sample memory decay typically exceeds six orders of magnitude
- 6 Prior knowledge of elemental concentrations is not mandatory

The results presented in this report tests the feasibility of these concepts

## EXECUTIVE SUMMARY

### Purpose

The purpose of the contract was to build the equipment necessary to show technical feasibility of a Demand Modulated Electrothermal Atomization System. This system was thought to have advantages over current technology as follows:

- 1 Prevents excessive analyte concentrations in the plasma giving the analyst control over matrix suppression effects.
- 2 Allows the analyst to control atomization rates and avoid buildup of deposits in the throat of the sample cone opening.
- 3 Allows the analyst to work in the optimum counting range for isotopic ratio work regardless of concentration variations.
- 4 Allows the data to be taken over the temperature dimension, thus resolving isobaric interferences as well as improving the signal to noise ratio resulting in improved detection limits across the entire mass range.
- 5 Allows the routine use of high dissolved solids (>10%) as well as solid sample introduction.
- 6 Allows the routine use of small sample sizes- 100 ugms or less.

### Performance Against Phase One Technical Objectives

Item #1. Software was created to collect and analyze data in three dimensions-AMU, concentration and atomization temperature. This software gives the analyst program control over a comprehensive list of variables. These are as follows:

- 1 Channels per sweep and dwell time
- 2 Default maximum sweeps per summation buffer
- 3 Default minimum sweeps per temperature step
- 4 Default maximum sweeps per temperature step
- 5 Default number of pulses per temperature step
- 6 Maximum temperature increments before shutoff.
- 7 Number of initial sweeps to throw away
- 8 Starting and ending temperature.
- 9 Number of sweeps per temperature increment
- 10 Size of temperature increments
- 11 Number of sweeps to be stored in each data histogram
- 12 Starting and ending amu for scans.
- 13 Starting and ending amu for threshold control

See software manual for detailed description

Item #2. Atomization conditions for direct solid sample introduction were not accomplished due to the complexity and time constraints of the contract. Solid samples were introduced into the plasma under stable conditions and data was taken. Detailed methods development will be needed to determine valid parameters for specific analyses.

Item #3. Solid samples were not analyzed per se, however analysis of 10% dissolved solids was accomplished. These results were presented at the 1987 Pacific Conference on Chemistry and Spectroscopy in Irvine, California.

Isobaric interferences were temperature resolved for Cd and Pd at amu 106 and Hg and Pb at amu 204.

Item #4. VG Isotopes was not able to get the instrument to perform as specified. The background was high and the responses were low resulting in detection limit performance for the standard instrument that was three orders of magnitude high. For this reason, no detection limit studies were performed.

Item #5. The software and hardware were designed to accommodate computer control of various gases during an analysis. Specific studies were not accomplished because the instrument in use for this program had limited usable count rates of two million per second. VG's updated design (electronic) will now give up to ten million counts per second.

Item #6. Time did not allow this item to be completed.

Although some of the planned items did not get accomplished, the equipment and software built under this contract has demonstrated the ability to perform these tasks with additional work on specific methods for each type of analysis.

#### Summary of Completed Work

1. Designed and built a torch/furnace assembly that significantly reduced memory and allows the analysis of high dissolved solid and solid samples.
2. Designed and built a computer controlled power supply to provide the power to heat the furnace.
3. Designed and built the data acquisition board to acquire count data over amu and temperature in any form desired under computer program control.
4. Completed the software needed to control the data acquisition.

and data analysis under complete control and definition by the analyst

5 Performed analysis demonstrating that all items worked as designed Data was taken resolving isobaric interferences with temperature as well as the introduction of high dissolved solids and solid samples

6 Two patents were applied for based on the concepts outlined for this contract using specific hardware developed under this contract A copy of these patents is provided in the addendum

Additional work needs to be done to develop method for the analysis of materials of interest This contract has provided all the hardware and software to accomplish this

Work has continued since the end of this contract to refine both the hardware and software for additional capability Please see the addendum for a summary of this progress

## Problems with Standard VC Plasmaquad as Delivered

### Plasma/Vacuum Interface

#### Purpose

The interface is designed to allow as much atmospheric pressure plasma plus analyte into the high vacuum chamber containing the quadrapole and detector as possible

#### Problems

The holes in the cones which provide the entrance to the vacuum chamber must be kept very small, of the order of 1 mm. During the analysis both the hole diameter and the surface roughness of the hole interior must remain constant. This does not happen when analyzing many materials including samples containing total dissolved solids above 1% or strong acids.

#### Discussion

Conventionally, if samples containing high analyte concentration were analyzed, the cone orifice would quickly become clogged and degrade the signal in several ways. The hot analyte ions and atoms (up to 8000 deg K) strike the cold cone surface by traveling through the boundary layer flow adjacent to the cone with some sticking to it. Others will sputter material away. For low analyte concentrations, experience shows that an equilibrium is reached where very little material is either stuck to the cone or sputtered off.

At high analyte concentrations more and more material sticks to the cone creating a disequilibrium that will clog the cone. During this process two things are occurring simultaneously. First, the cone orifice is getting smaller causing the signal to degrade because less analyte is getting through the orifice per unit time. Secondly, the clogging process makes the hole surface much rougher. This in turn induces more turbulence in the plasma flow through the hole and hence a thicker boundary layer. The thicker boundary layer produces more boundary layer mixing with the main plasma flow that further slows the velocity through the orifice. With this slowing the signal is degraded by still less analyte flowing into the vacuum chamber as well as the increased generation of polyatomic species adding noise to the background.

The GSI ETVS (Electrothermal Vaporization System) solves these problems by controlling the rate at which the analyte is introduced at the plasma/vacuum interface and by using a much smaller amount of sample needed (10-100 uL).

## Detector Preamp Issues

### Problems

The detector is not linear and not precisely reproducible in its response to the varying ion fluxes. The pulse width of the preamp (45ns) causes "pulse pileup" or the overlapping of pulses to form longer pulse with the accompanying loss of data.

### Discussion

The detector is a continuous dynode type in which a continuous active surface is biased from one end to the other. An impact causes an electron cascade to avalanche from one end to the other. The bias current is less than 20 uA. If the ion flux striking the active surface causes a cascade current of more than about 10% of the bias current, the bias current cannot replenish the active surface adequately. Thus, later arriving ions do not produce a cascade large enough to trigger a preamplifier response until the surface of the detector has been replenished. This leads to what is called detector fatiguing or loss of signal due to the inability of the detector to respond adequately. This affect begins at about 2 million counts per second.

Pulse pileup occurs whenever the incoming pulses from the detector to the preamp are spaced closer than 45 ns. It is not dependent on the overall rate of ion influx, but on the immediate rate of ion collisions with the detector. That is, a short intense burst of ions will cause pulse pileup even if the average rate of ion influx is low.

### Solution

The GSI ETA feedback control allows lower count rates to be specified so as not to push the cascade current beyond the 10% bias limit. The data acquisition board is designed to handle 10 ns pulses from the preamp. Presently the GSI system does not have the higher speed preamp incorporated into current VC systems.

## Memory in Torches, Spray Chambers and Cones

### Problem

Memory of many elements from a sample that has just been analyzed will carry into the subsequent samples. This memory may take hours to decay to an acceptable level.

## Discussion

The conventional ICP Mass Spectrometer setup uses a Scott spray chamber with a Fassel Torch made of quartz. This combination is fine for extremely dilute solutions but fails when large concentrations of analyte are aspirated into the system. The nebulized sample traveling from the spray chamber to the torch injector tip experiences an increasing temperature gradient and a large surface area of quartz. As the flow moves the contained particles may collide with the wall and be removed from the flow. The residence time of the particles on the quartz is governed by many factors and is not infinite. Thus later sample flow will capture particles released from the quartz surface which will become memory located in the torch.

Cone memory is memory generated by the sputtered material being released and entrained in the plasma flow through the cone orifices. The typical memory decay between samples for the standard spray chamber and torch is about three orders of magnitude or 0.1% of the original sample.

## Solution

The design of the electrothermal furnace and torch was done to reduce memory by at least six orders of magnitude between samples. The feedback control limits the ion flux out of the plasma to eliminate cone clogging and hence a source of memory.

## HARDWARE/SOFTWARE DEVELOPMENT

### Development of a Tantalum Furnace

Work done studying the characteristics of a prototype electrothermal furnace design demonstrated carbon to be an unsuitable surface from which to attempt the atomization of samples. The primary problem is precision. The unpredictable nature of the carbon surface during atomization precluded generating equivalent quantitative data for the same sample in successive runs. This result dictated another suitable surface be found from which atomization could take place. To this end tantalum metal was chosen. The properties allowing tantalum to be chosen include.

- (1) Essentially a monoisotopic element (99.988%)
- (2) High atomic mass (180.9480)
- (3) Very refractory (high melting temperature)

The furnace design is as follows. Two copper electrode blocks have a carbon annular ring between them. When energized, the carbon ring is heated from ambient temperature to greater than 2600C depending on the power applied. Inside this carbon ring is a cylindrical tube with a narrow snout at one end made of tantalum and inside the tube a small tantalum sample spoon may be inserted or retracted from the rear. With the spoon inserted, the furnace becomes a small black body oven in which the sample is uniformly heated until atomization occurs. A variable flow of carrier gas (with or without a vapor pressure enhancer included) is passed over the spoon through the tantalum tube to carry the atomized sample directly into the plasma directly adjacent to the furnace. The complete injection nozzle for the plasma is also constructed of tantalum. The only glassware remaining in the system is the torch glass inside the RF coil.

### Principles of Operation

The GSI Electrothermal Furnace is an atomization furnace and an inductively coupled plasma ion gun all in one unit. The body of the furnace is constructed of tantalum metal (99.99% pure min). Surrounding the oven body is a cylindrical heating element of graphite which is clamped between two large, water cooled copper electrodes. A graphite heat shield surrounds the heating element to contain heat more efficiently within the oven during operation. Samples are introduced to the oven on a small retractable spoon which is breech loaded into one end of the oven. The spoon is also constructed of tantalum. A flow of

gettered argon equivalent to the nebulizer flow in a conventional ICP torch flows over the spoon through the atomization furnace carrying vaporized sample from the furnace to the plasma.

The plasma torch has been uniquely constructed from the same tantalum pieces that comprise the atomization furnace. In this way the shortest possible distance from furnace to plasma has been created. This is the prime contributor to the extremely low memory characteristics of this design. The only quartz in the system is a single outer plasma containing jacket between the load coil and the plasma. Careful design of the furnace body and injection nozzle (equivalent to the quartz injection tip in a conventional torch) allows cool plasma gas to flow backward over the outside of the tantalum oven body but inside the graphite heating element. This eliminates any possibility of unwanted material from the body of the oven or the outside atmosphere getting entrained into the analyte flow. This also prevents material being swept from the plasma and condensing on the body parts of the furnace reappearing later as memory.

#### Repair and Maintenance

The GSI Electrothermal Furnace has been designed to require only minimal maintenance. Periodically the quartz barrel which contains the plasma will have to be replaced due to recrystallization and shattering of quartz when in the RF field. O-rings are used several places within the furnace for sealing gas flows, coolant flows, and aligning the quartz barrel to contain the plasma. All these O-rings are viton with a hardness of 75 durometer. The graphite heating element and heat shield should be occasionally inspected for wear and development of cracks. With proper care and use the heating element will last indefinitely, however, graphite is very delicate material and can easily be broken with misuse.

The tantalum oven body and sample holding spoon are the most susceptible parts of the furnace. Under no circumstances should any gas containing oxygen in any concentration be fed through the oven. Tantalum is a tenacious getter of oxygen and turns to a crumbly white powder when exposed to oxygen at elevated temperatures. The small volumes of acids and aqueous material pipetted onto the sample spoon will eventually erode the spoon due to oxidation. The life of these two parts depends primarily on the type of samples and chemistry of solvents introduced to the oven. With time the tantalum oven body will recrystallize into hexagonal platelets and crumble. This is most aggravated when carbon is present. All replacement parts are available from GSI.

## Furnace Use

The breach of the furnace is pneumatically opened and closed to retract the spoon for sample loading. The pneumatic piston is actuated by the furnace open/close switch on the instrument front panel. This switch energizes gas flow solenoids allowing gas flow to the piston. When closing the breach block holding the tantalum spoon should smoothly engage the fitting in the large electrode block. If it does not, the nylon fitting coupling the spoon holder to the actuating piston should be gently stressed in the appropriate direction to allow the breach blocks to engage. After a short usage period the nylon fitting will take a "set" and provide reliable breach closing.

## Connections

Electrical connections are as follows: the positive terminal from the GSI Electrothermal Furnace Power Controller goes to the ground furnace electrode block which is the one directly bolted to the blue frame of the furnace. The negative terminal from the controller is connected to the insulated furnace electrode block which is the one mounted on the three metal rods by nylon insulation spacers.

Input argon is required in two places. At the tantalum plasma nozzle input and at the carrier gas input on top of the tantalum spoon holder assembly. Nominal flows providing the best results have been found at about 16 liters per minute for the plasma nozzle and about 1 liter per minute for the carrier gas flow.

Cooling water is input in series through three of the furnace blocks. The input should be to the round front block shielding the tantalum plasma nozzle. The output from this block goes to the input of the ground furnace electrode block and finally the output from the ground electrode block goes to the input of the electrically hot electrode block. The output from this block goes to the drain.

## Development of the Furnace Controller

Complete manual control of the GSI Direct Electrothermal Atomization Furnace is impossible. Therefore, the furnace is controlled by the GSI Feedback Furnace Controller. This device is used to control the power applied and consequently the temperature of the furnace. An IBM PC-XT is used to control all instrument functions including furnace control. The prototype design uses a power triac solid state device and the necessary digital electronics to convert an eight bit output word from the computer into a power level for the furnace. Raw power is brought to the controller via a 208 volt line. This is converted to 7.5 volts using a 5kVA transformer with appropriate core biasing to drive the furnace load of about 15 milliohms.

## Development of Controlling Software

The development of controlling and data acquisition software for the GSI Direct Electrothermal Atomization Furnace, the GSI Feedback Furnace Controller and the GSI Real Time Data Buffer Card was contracted. The software must be able to collect and store data in real time, analyze the data sufficiently to determine appropriate furnace control and interface with existing instrument control and data handling software.

Control of the instrument during data collection is performed in the way VG Isotopes originally designed the VG Plasmaquad mass spectrometer to operate. The GSI software, in the design phase, works in parallel with the VG Plasmaquad data collection software but with several additions. Parameters governing a sample analysis are completely set by the GSI software. These parameters are then passed to the VG Plasmaquad instrument control software for the initiation of a sample run. In addition, the initial furnace control parameters are set in the furnace controller.

Actual data collection parallels the VG Plasmaquad system by tapping the output signal from the preamplifier of the VG instrument. The data is produced by successive sweeps across the mass region of interest. After each sweep or set of sweeps (controlled by initial parameters) the computer then decides whether or not the furnace may be incremented in temperature. This decision is based on the number of ion counts in each channel. If any mass channel in the mass scan region exceeds a preset threshold value for that channel, the material is deemed to be atomizing at too great a rate and the furnace is not allowed to increment in temperature. The furnace will be held at its present temperature value until all mass channel values are within their respective thresholds. All data is written to the hard disk in real time.

## Data Acquisition Hardware/Software Operation and Interface

In August 1986, the VG mass-spectrometer was equipped with an Olivetti Z8000 based CPU and the standard Tracor-Northern multichannel analyzer. The MCA was accessed from the Olivetti through a IEEE-488 bus. The time required to poll and extract the data stored in the MCA was about 3 seconds. This data extraction needed to be done once for each sweep, or about every 0.2 seconds (2048 channels/sweep \* 100 uSec/channel). This forced a data collection path to be developed, that was parallel to the MCA, but allowed real time access to the data.

Once this decision for custom electronics to collect the data was made, the next decision was which CPU and language to choose for the host computer and the software. Since VG already had an IBM PC based system that replaced the Olivetti Z8000 and their new software was written in Logitec's Modula-2, this was examined and the CPU and language were determined to be very desirable.

At this stage, late August and early September 1986, a search for off the shelf hardware was conducted that would allow a direct memory access (DMA) of the data from the collection device to computer memory to occur. No reasonable DMA boards were found.

The decision was then made to input data with a parallel I/O card (made by John Bell Engineering) and collect the data on a custom in-house board. The John Bell board required one slot in the IBM and the in-house board required the end slot and the space next to it.

The inputs to the data collection board are:

- 1) the next channel pulse (from the PQ controller).
- 2) the count pulses (from the La Croix amplifier)
- 3) control signals from the host computer (through the John Bell card)

The outputs from the data collection board are:

- 1) eight (8) bits of temperature control to the furnace controller (allowing 256 levels of power/temperature), controlled from the IBM computer.
- 2) data and control signals to the IBM through the John Bell board.

The heart of the data collection board consists of a set of two counters: the current temperature counter and the current count. The current counter is a free running event counter that counts pulses from the La Croix amplifier. On a next channel pulse the current count is passed into an on-board FIFO and the

counter is reset. When a start of sweep is detected, the current temperature is passed into the FIFO. The data from both of these counters is sixteen (16) bits wide. The temperature data has the top two bits forced on, this marking is used to align the beginning of sweeps and control the summation of data into temperature bins. The count data has the high bit clear.

Whenever data is present in the FIFO, the IBM is notified through a control line. The IBM then extracts data from the FIFO through the John Bell card.

The VG software was modified to work with two GSI routines. The first routine, METHOD, is the main control routine. The second routine, GETDATA, is the data collection, summation, and storage routine. GETDATA is called by METHOD, after the mass-spectrometer begins collecting data.

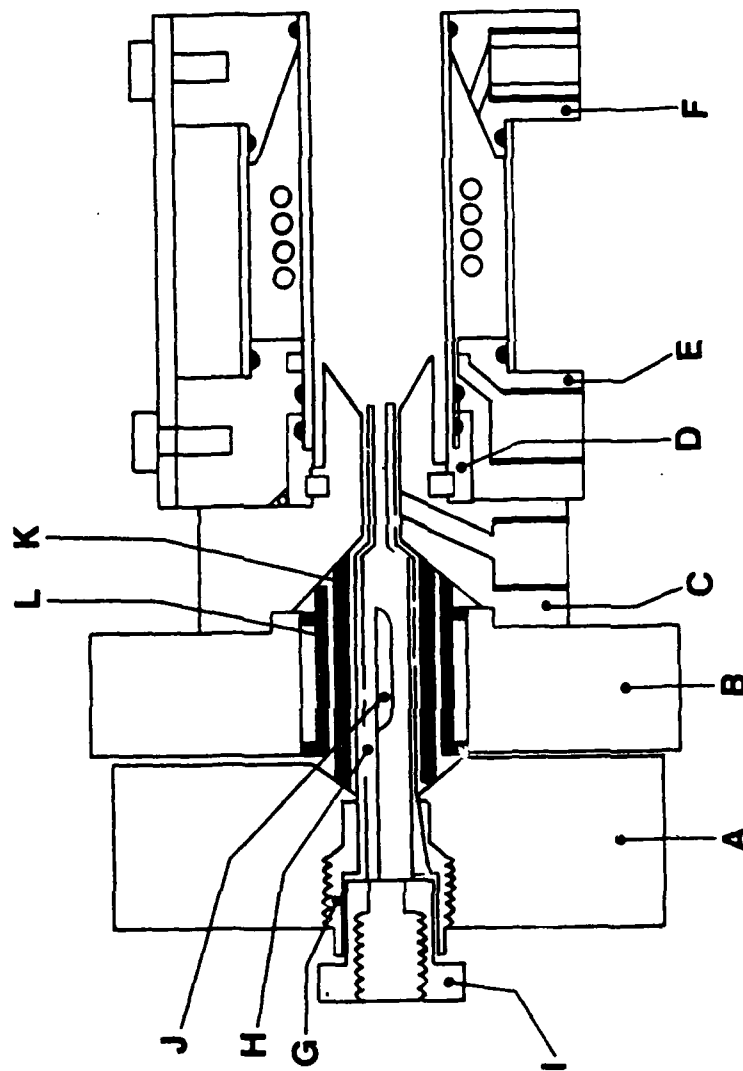
METHOD prompts for the type of action requested.

- 1) data collection or
- 2) data analysis

In the data collection mode, METHOD prompts for and inputs a method or control file. A method file contains the parameters needed to run the mass-spectrometer through the VG control software, control the power to the furnace, and control the summation of the data. Method files are created and modified off-line with a text editor. The furnace control is achieved by allowing the temperature/power setting to be incremented by a scheduled amount, only when the counting rate for each channel in the sweep falls below thresholds set in the method file. Additional lower and upper limits on the number of sweeps at each temperature/power setting are imposed. Data is summed within a temperature setting into bins. The number of sweeps within each summation bin is controlled. E.g., one sweep per bin allows very fine granularity in the data and many bins, while at the other extreme, all sweeps at one temperature summed into one bin allows 256 bins total. The amount of granularity required is determined by the disk space available. Each channel is summed into a 32 bit integer in the summation bins, 2048 channels \* 4 bytes = 8kb/summation bin of data + a header. The original disk space allocated was 10 Megabytes. This gave 10M/8K or about 1100 bins total. Each scan required 10 Megabytes of storage!

The data analysis allowed reduction of the data into a VG formatted raw data file, analysis of the temperature data with the background (zero temperature) subtracted from the remainder of the data, and plotting of the raw data or zero reduced data.

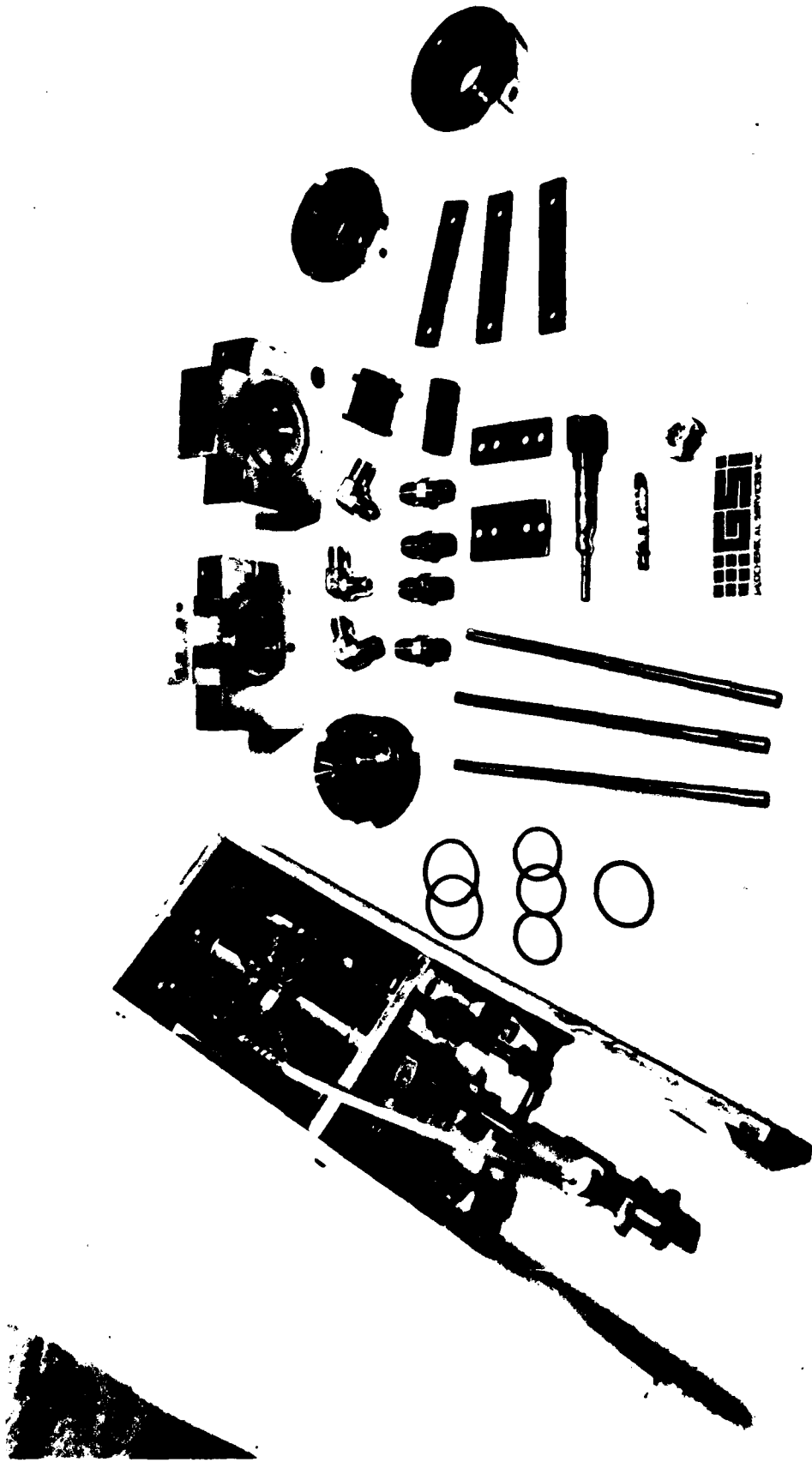
# TORCH/FURNACE ASSEMBLY



I - Breach Block  
 J - Sample Spoon  
 K - Heater Element  
 L - Heat Shield

E - Cooling Manifold  
 F - Outer Manifold  
 G - Tube Holder  
 H - Tantalum Tube

A - Moveable Block  
 B - Oven Block  
 C - Plasma Nozzle  
 D - Sealing Ring

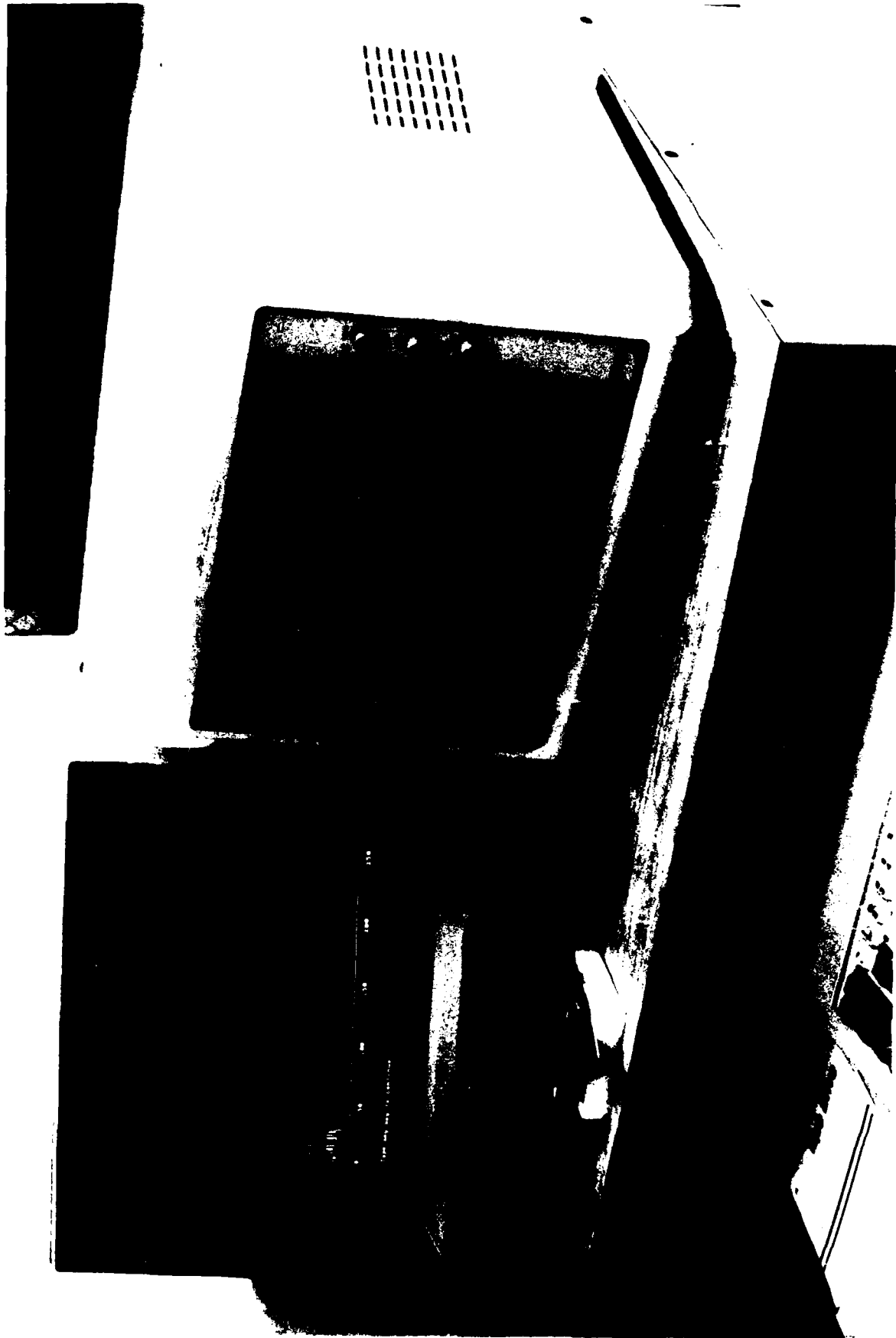


**GSI FURNACE/TORCH ASSEMBLY**

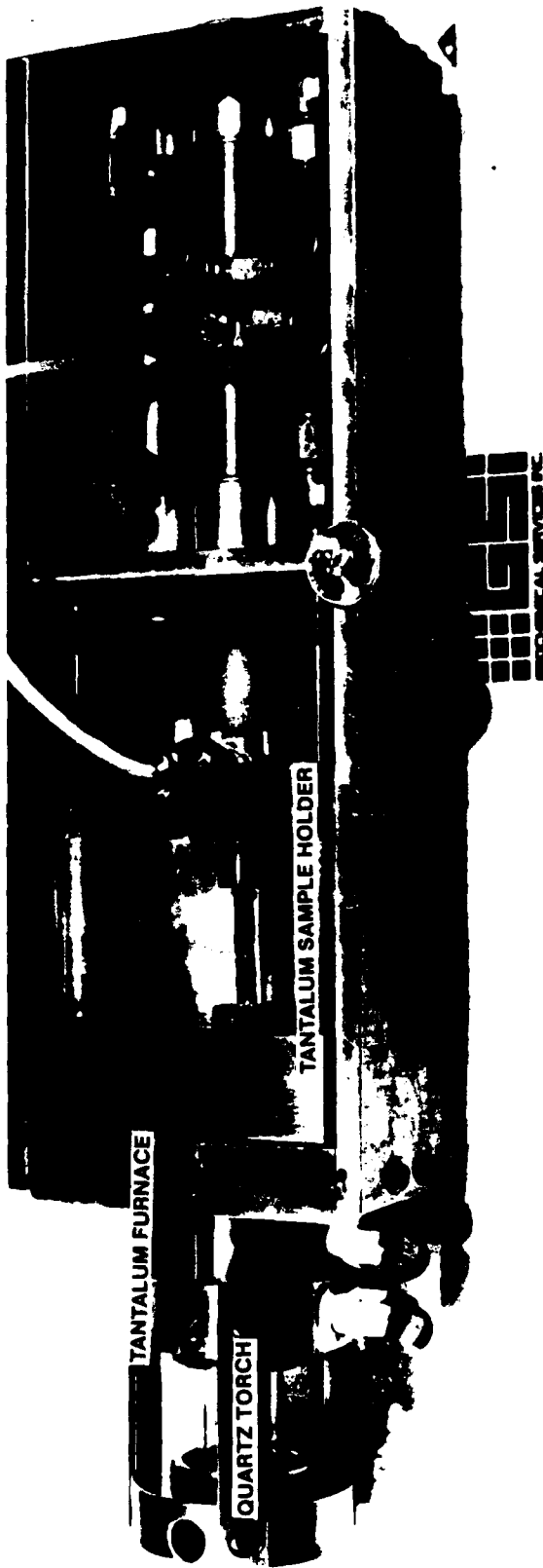




**GSI DATA ACQUISITION ELECTRONICS**  
(FITS IBM PC EXPANSION SLOTS)



DATA GRAPHICS USING GSI SOFTWARE



GSI FURNACE/TORCH ASSEMBLY

GSI ICP/MS SOFTWARE

USERS MANUAL

## The GSI Package

The software package includes several programs which are written in modula 2 and in IBM assembly. This package has been designed to be compatible with the standard VG Plasmaquad ICP/MS software. The instrument is still brought to operational mode and returned to standby mode using the VG Plasmaquad software.

The GSI package consists of the following files

- (1) m2.exe
- (2) getdata.exe
- (3) dataread.lod
- (4) method.lod
- (5) turbo.exe
- (6) turbo.msg
- (7) markbad.com
- (8) gwbasic.exe
- (9) plot.bat
- (10) method.txt
- (11) config.sys
- (12) autoexec.bat

With these twelve programs you enter another dimension in ICP/MS data collection.

This software allows you to collect data in three dimensions in real time from your mass spectrometer. The dimensions are atomic mass units, ion counts and temperature of the electrothermal atomization furnace. In this way you are able to observe directly when each element volatilizes from your sample of interest. You have infinite flexibility in programming the parameters under which data is collected. This is done by means of a method file. This file is created using any word processing editor you have available (we use the Turbo editor).

In this file you specify all the parameters necessary to run the instrument and control the furnace during the run. You then start data collection and watch!

### How the system works

The system consists of several pieces of hardware and software. The GSI Direct Electrothermal Atomization Furnace replaces the conventional ICP torch and is controlled by the GSI Feedback Furnace Controller. Data collection is performed using the GSI Real Time Data Buffer Card which is installed in your IBM PC-XT or PC-AT. The software then collects data, analyses it to determine the rate of atomization of the sample and controls power to the furnace. The advantages of this system over the

standard VG software and hardware are

- (1) the detector remains within linear counting range
- (2) maintains a stable vacuum plasma interface orifice
- (3) matrix effects in the plasma are controllable
- (4) allows resolution of isobaric interferences

### System requirements

In order to run the GSI ICP/MS software your computer must be an IBM PC-XT or PC-AT or compatible. You must have a regular monochrome monitor as well as the Princeton (or NEC multisync) monitor. You will also require at a minimum a 20Mbyte hard disk drive and preferably a 30 Mbyte drive.

### Getting started

To begin, load the two programs config.sys and autoexec.bat in the root directory of your hard disk and reboot the system.

#### Load the files

```
C:\>copy a:autoexec.bat
C:\>copy a:config.sys
```

Reboot the system by pressing the CTRL, ALT and DELETE keys simultaneously. Next create a subdirectory called \M2 as follows

```
Creating \M2 subdirectory
C:\>md m2
```

```
Go into this subdirectory
C:\>cd m2
```

Now load the following programs from the floppy disk

- (1) m2.exe
- (2) getdata.exe
- (3) turbo.exe
- (4) turbo.msg
- (5) markbad.com
- (6) method.txt

This is done using the copy command. The LOD modules from this package must be loaded in a subdirectory labeled \M2LOD. Do this as follows

Return to the root directory  
C:\M2>cd\

Create subdirectory \M2LOD  
C:\>md m2lod

Go into \M2LOD subdirectory  
C:\>cd m2lod

Now copy the LOD files from the floppy into this subdirectory.

The next step is optional but we prefer it for housekeeping purposes on the hard disk. Return to the root directory as above and create another subdirectory called \VG. Go into this subdirectory and copy all the original VG Plasmaquad LOD modules and the VG Plasmaquad version of m2.exe from the root directory into this subdirectory. Then return to the root directory and delete all of the VG Plasmaquad files in this directory. You are now ready to run (assuming all the necessary hardware is installed correctly on your instrument)

To run, start the instrument normally using the VG Plasmaquad software. When stable and running break out of the VG Plasmaquad software and return to MS-DOS by pressing CTRL BREAK

Using GSI ICP/MS software

Go to the \M2 subdirectory  
C:\M2>cd\m2 (return)

Run the program getdata.exe  
C:\M2>getdata (return)

Run the program method lod

C:\M2>m2 method (return)

The method program is menu driven and quite similar to the VG Plasmaquad menu driven programs. The method file is created using the turbo word processor and is covered in the method file section of this manual. In order to run select the menu choice listed in the initial screen of the method program which displays as follows.

```

:
:                               GSI DMA SOFTWARE
:                               Copyright 1986 Geochemical Services, Inc
:
: R      Read method file
:
: C      Control and run from a method file
:
: V      VG data format write of zero sum data
:
: Q      Quit and return to DOS
:
:                               Enter your selection [R,C,V, or Q]
:
:
```

R displays a method file on the screen for you to read. This is useful in reviewing your sample analysis method.

C passes analysis parameters to the instrument at the beginning of data collection and controls the furnace during sample analysis according to the method file. A normal termination of the instrument and data storage in a background corrected VG Plasmaquad binary data file occurs at the end of data collection.

V allows the hard disk data storage area to be interrogated and a background corrected VG Plasmaquad type binary data file is written.

Q exits the GSI ICP/MS software and returns to the MS-DOS operating system

#### Creating and editing method files.

Complete control of data collection and the electrothermal atomization furnace is done by means of a method file. In this file all the parameters that are required to run the VG Plasma-quad during data collection and control the furnace during this time are specified in a method file. The format of a method file is as follows

```
2048 0250      channels per sweep and dwell time(usec)
0000          default max sweeps per summation buffer
0000          default min sweeps per temperature step
0000          default max sweeps per temperature step
0000          default # of pulses per temp step
0210          maximum furnace increment before shutoff
0125          # of initial sweeps to throw away
0000 0000 0010 0100 0001 0120
              min/max tmp,min/max sweeps,tmp inc,sums per step
(this control line may be repeated several times)

-1           start amu and threshold definitions
0007 0125    starting and ending amu for scanning
0023 0027 0500 starting and ending amu, threshold
```

The first line defines the number of channels per sweep the quadrupole has and the dwell time at each channel. The next four lines define default parameters which would be used only in the simplest methods. With these four values you can define a simple staircase ramp for data collection. The sixth value is the maximum power increment the furnace is allowed to attain. The range is from 0 to 255 representing 0 to 100 percent power to the furnace. The number of throw away sweeps defines how many of the initial sweeps are to be ignored. This feature allows you to ignore sweeps when the detector is first turned on to allow for any transients or initial fatiguing that may occur

#### (a) furnace control parameters

The next six numbers in a line define control parameters for the furnace and are repeated as necessary in order to produce the desired furnace temperature ramp for the particular analysis

being performed. The first two numbers in each of these lines define the starting and ending temperature for the part of the temperature profile controlled by the respective lines. These numbers must range between 0 and 255. The next two numbers in each of these lines define the minimum and maximum number of sweeps across the mass spectrum that may be collected at the present temperature increment before the furnace is commanded to the next temperature increment. The fifth value in each line commands the size of the increments between successive temperature steps for the furnace in the temperature interval specified in that line. The sixth value defines how many sweeps across the mass spectrum are collected in a histogram before the data is stored and the next histogram is begun.

The -1 value defines the end of the furnace control data. The next line identifies the starting and ending values for the sweep across the mass spectrum and is specified in atomic mass units (amu). These values are calibrated from the VG Plasmaquad software calibration file resident on the hard disk. The calibration may be updated using the VG Plasmaquad software.

(b) feedback threshold control

The final set of lines with three values each define the feedback control threshold parameters. The first two values identify the region of the mass scan region to which the control threshold is applied and is specified in amu. The last value is the threshold value above which no mass channel inside the mass range for that threshold value is allowed to go. If the threshold is exceeded the furnace is held at its present temperature provided the maximum number of mass scan sweeps specified for that temperature has not been exceeded. Any number of thresholds may be set but remember that thresholds specified after set thresholds in lines previous will be overwritten by the next threshold value if there is overlap.

An example:

```
0023 0067 0500
0064 0093 0800
```

In this example the region from amu 64 to amu 67 appears to be controlled by two threshold values. However, only the last value - 800 - is recognized in this region of the mass spectrum.

(c) using the Turbo editor

This editor is the Turbo Pascal editor and contains many features. For those not familiar with this editor the following guidelines

To enter the editor

```
C \M2>turbo
```

Enter no (N) to the question do you want error messages - these messages are for the pascal compiler that is included with the editor. Unless you are writing pascal programs these messages are not useful. Next a menu with several choices will appear. Enter E to invoke the editor. Next you will be asked for a file to edit. You may enter either a preexisting filename or a new filename for the title of your method file. The file METHOD.TXT included with this software may be used and edited as a model. To move the cursor the arrow keys on the numeric keypad of your IBM PC-XT or PC-AT are used. The home command moves the cursor to the beginning of the present line and the end key moves the cursor to the end of the present line.

There are two modes (present mode is displayed at the top of the screen) while editing insert and overwrite. Insert mode allows text to be inserted in preexisting text while overwrite mode simply overwrites preexisting text. Characters are deleted using the delete key of the numeric keypad. When in insert mode lines may be inserted or deleted using the enter and backspace keys respectively.

To exit the editor the following sequence is necessary.

```
To exit to command mode hold down the CTRL button  
and type K. Then type D. The cursor prompt is now
```

```
)
```

```
To save your editing
```

```
)S
```

```
When finished saving
```

```
)Q
```

You are now back to MS-DOS and are ready to run the data collection software. It is a good idea to now copy method.txt to another filename meaningful to the sample analysis it pertains to. In this way you may build a library of method files while maintaining the file method.txt as a skeleton guide when editing.

Processing data

The program method writes a binary data file equivalent to

the VG Plasmaquad raw data file except that a background constant for each mass channel in the sweep is subtracted from the data. These constants are determined by the number of background sweeps specified in the method file. Each of the stored mass scan sweeps are interrogated and the total sum of these sweeps on a channel by channel basis are divided by the number of sweeps making up the sum. Since statistically this value can lead to negative values in the data file, any value that is determined to be negative by this operation is set to zero.

The program "dataread" interrogates the hard disk storage area and writes a set of data files that store response versus temperature of the furnace for any desired amu.

## ANALYTICAL PERFORMANCE

Several types of analysis were done in order to demonstrate that the software and hardware perform as designed. Because of the contract time and money constraints, detailed methods development for the semiconductor analysis in the original proposal was not possible.

An analysis was done to demonstrate the ability of the system to resolve isobaric interferences over a temperature profile which was a major goal of the contract. Figure A shows the analysis of a sample containing both Cd and Pd. At amu 111, Cd shows a signal vs. furnace temperature. At amu 106, both Cd and Pd have an isotope that can not be resolved on the normal mass spectrometer. Figure B shows the resolution of both Cd and Pd vs. temperature at mass 106.

Figures C through L show the responses of the rare earth elements vs. temperature. These elements are of interest for mining as well as the electronics industry. With the normal spray chamber, oxides of the lighter rare earth elements can cause significant interferences with the heavier elements. With the temperature resolved system the oxides would be expected to be temperature resolved. No oxides are present because of the lack of water in the sample introduction.

Figures 1 through 3 show data collected with software developed by GSI after the contract work. These outputs show the resolution of Hg and Pb at mass 204 on a complex United States Geological Survey reference standard, GXR-1. This standard has 4 ppm of Hg and 700 ppm of Pb. The sample was run as a 10% dissolved solid. Figure 1 shows Hg at mass 208. Figure 2 shows the Hg at mass 202. Figure 3 shows Hg at mass 204 resolved against Pb at mass 204.

## PATENTS

Patents were applied for on the furnace/torch assembly and the feedback control system as described in the contract proposal. The concepts on which the patents were based were documented prior to the contract award, however the filing dates were after the award and some drawings developed during the contract period were used.

Sample	Y, Sum	--	345045.0	11384935.0
Standard	Y, Sum	--	174221.0	4019627.0
Blank	Y, Sum	--	5736.0	55360.0

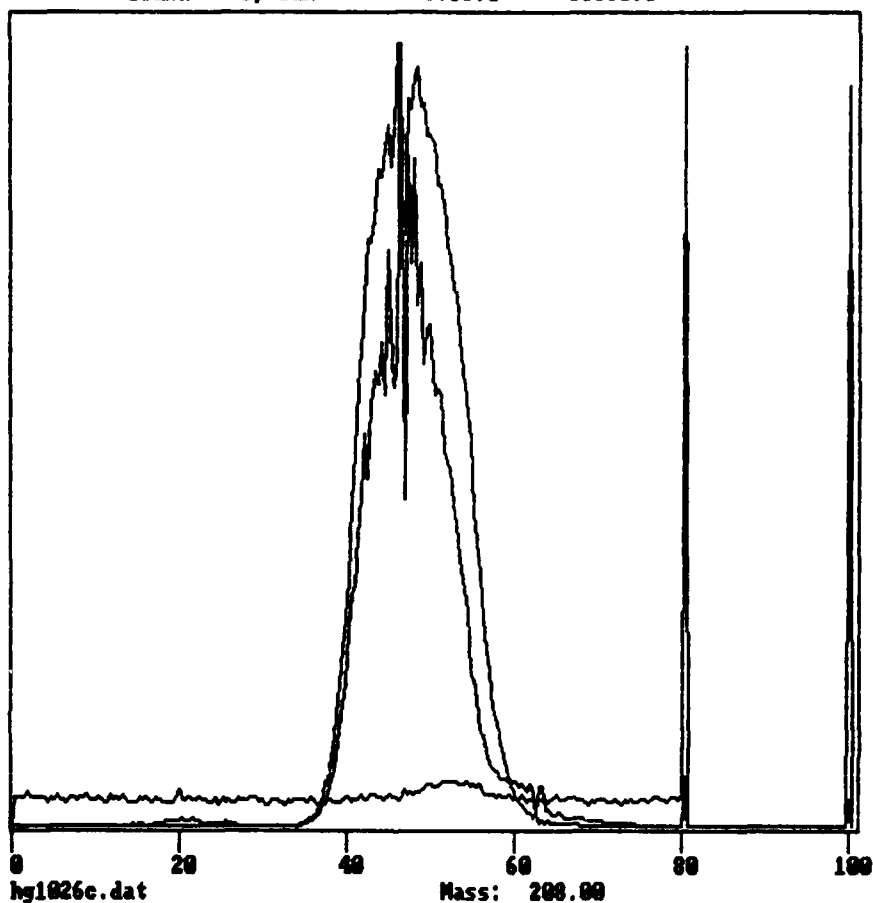


Figure 1: Temperature profile at atomic mass 208Pb. Three profiles are plotted: sample, standard and blank. Values displayed are the peak value are a single temperature and total integrated counts under the temperature profile respectively. The two spikes at temperatures 80% and 100% are due to integrated background during the furnace purge cycle at the end of analysis.

Sample	Y, Sum	--	10608.0	144382.0
Standard	Y, Sum	--	44532.0	473112.0
Blank	Y, Sum	--	1928.0	17296.0

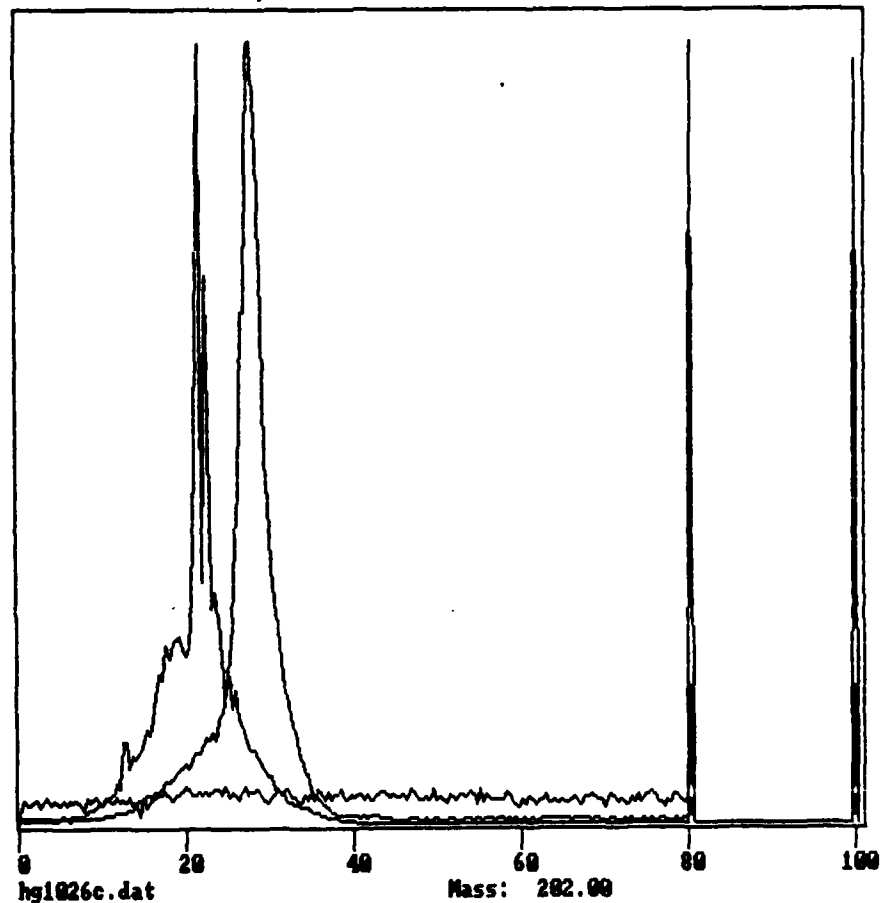


Figure 2: Temperature profile at atomic mass 202Hg. Plots are analogous to figure 1 with sample, standard and blank set. The two signal peaks are slightly offset due to matrix chemistry during analysis.

Sample	Y, Sum --	51448.0	1016478.0
Standard	Y, Sum --	17458.0	271631.0
Blank	Y, Sum --	1850.0	14862.0

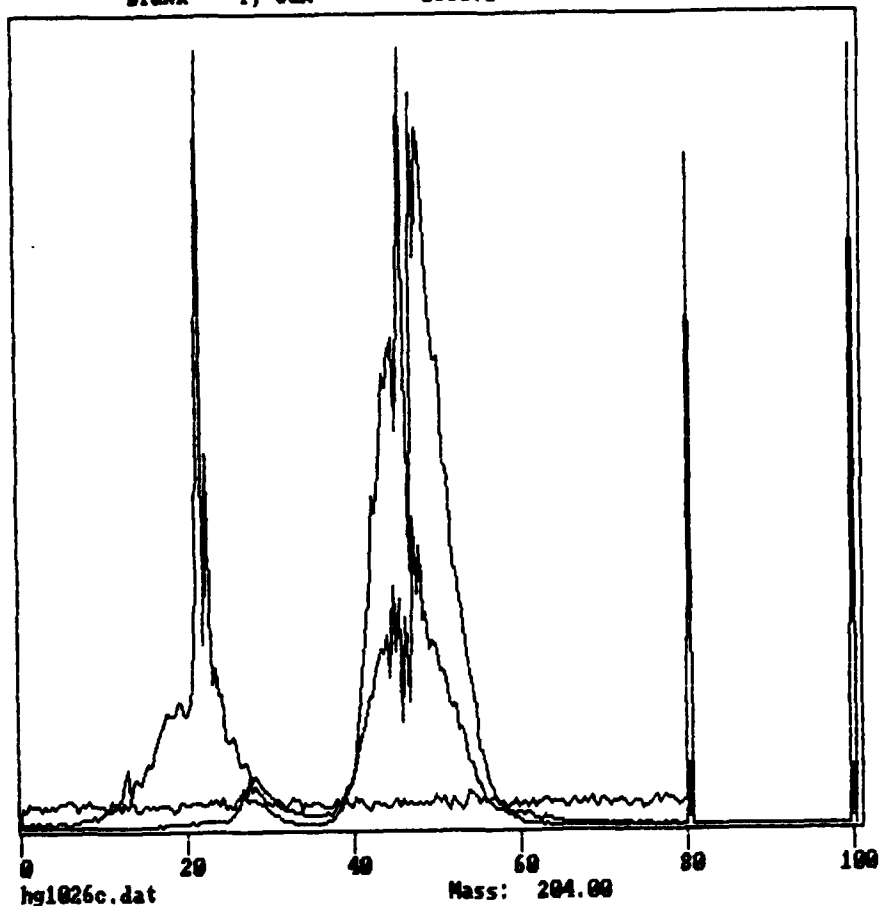
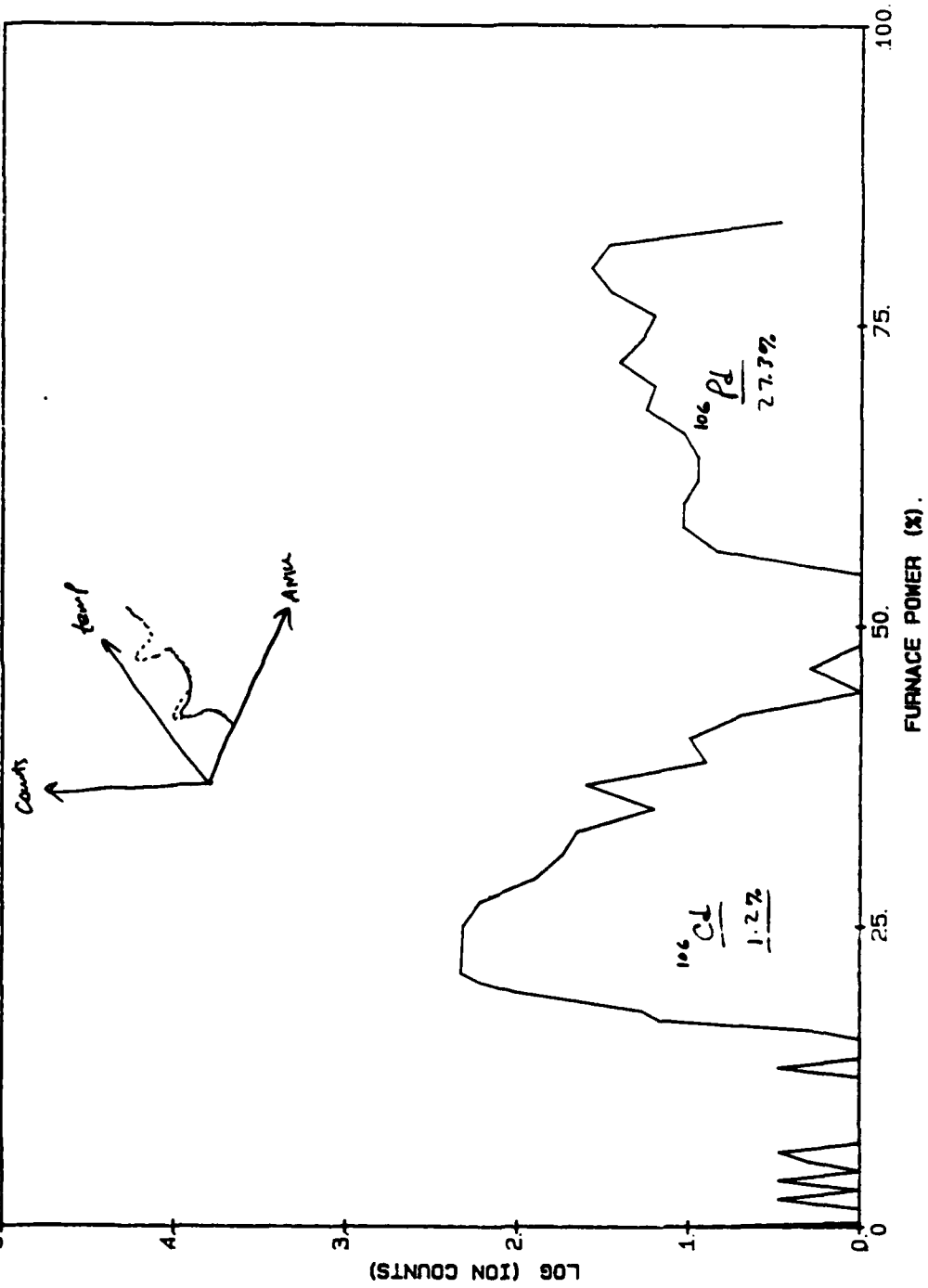
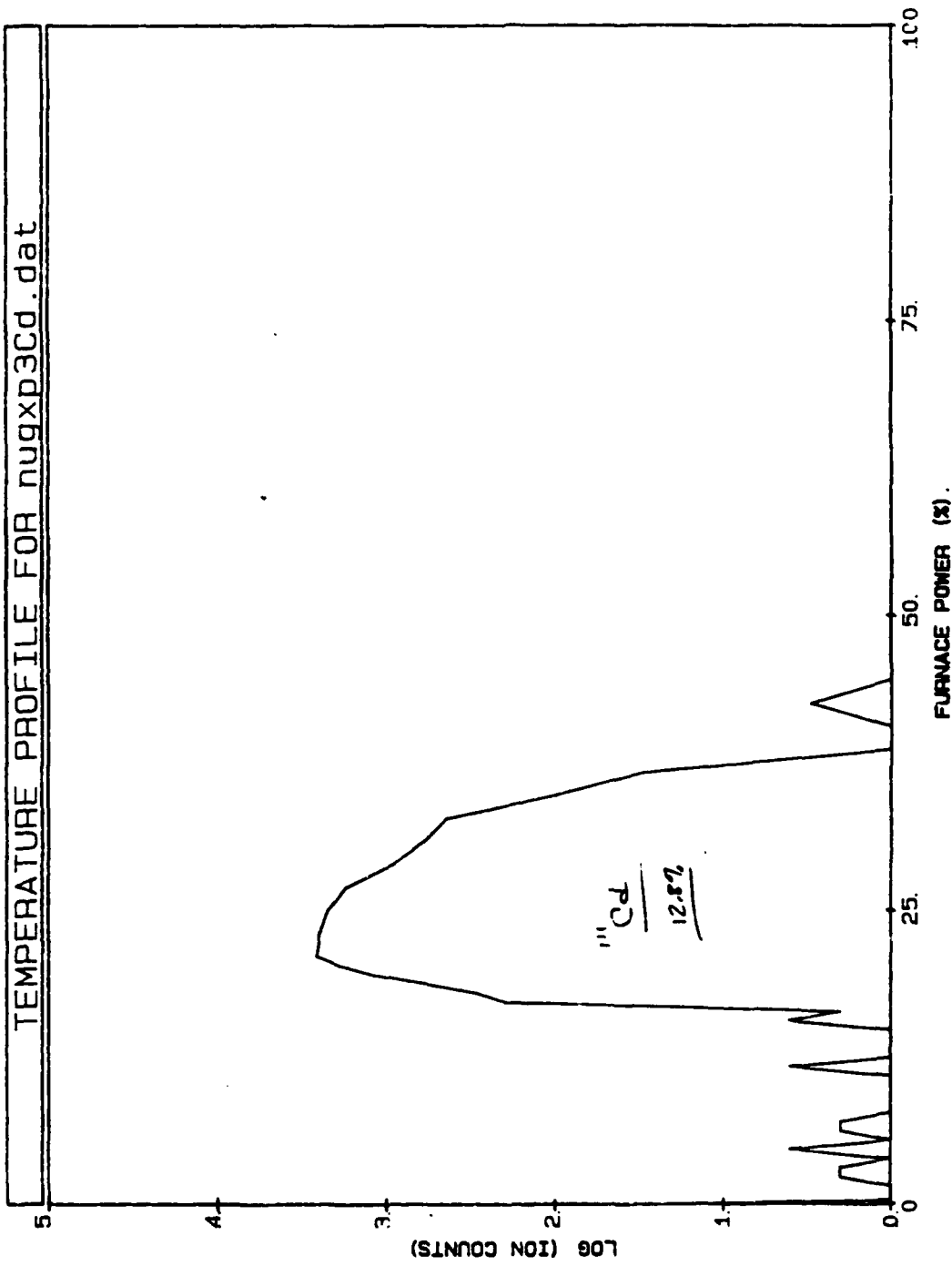


Figure 3: Temperature profile at atomic mass 204. This figure demonstrates the method of isobaric resolution between the 204 isotopes of Hg and Pb. Hg begins volatilization at about 10% power and ends at about 34% while Pb begins at about 39% and ends at about 60%.

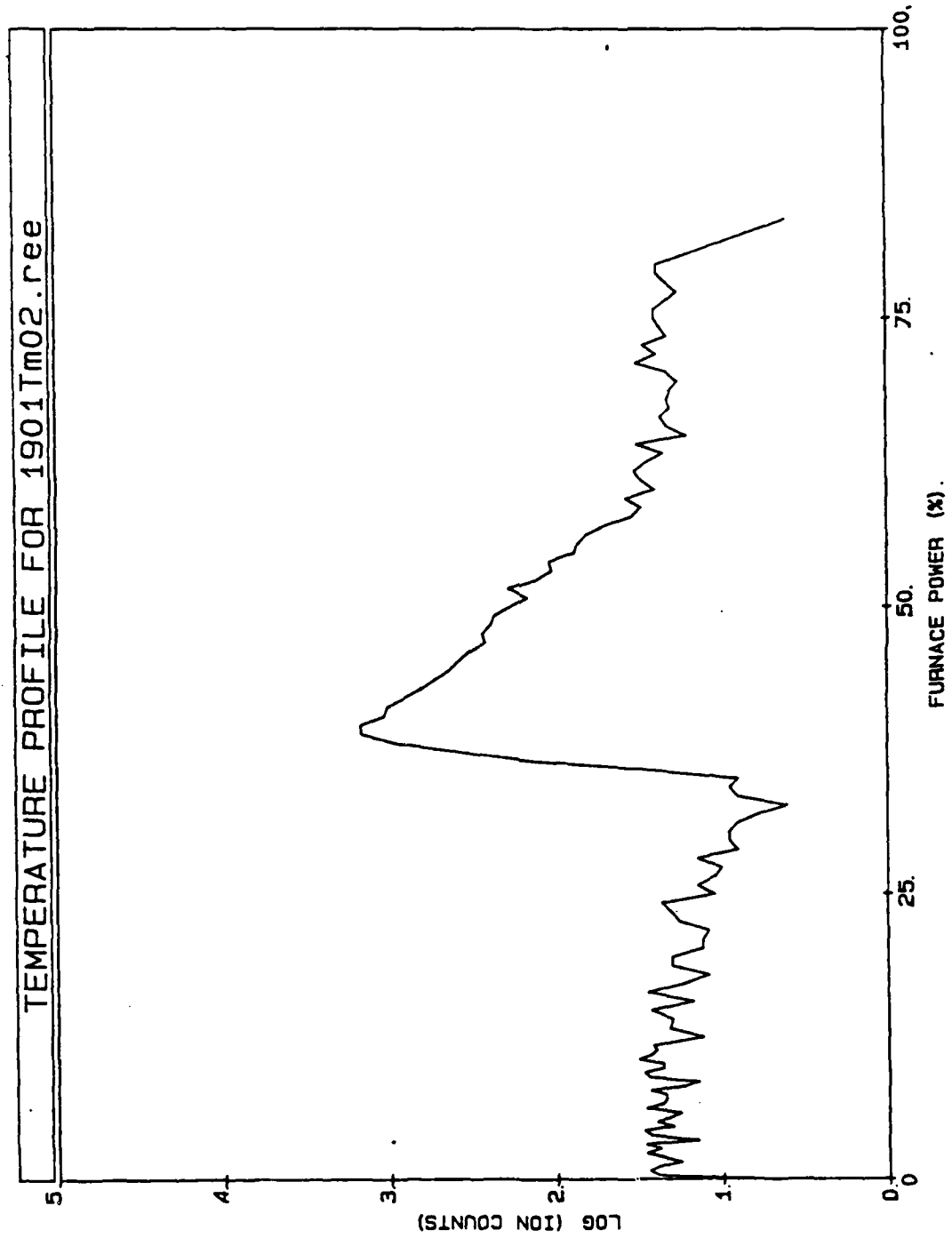
TEMPERATURE PROFILE FOR nugxp3Pd.dat



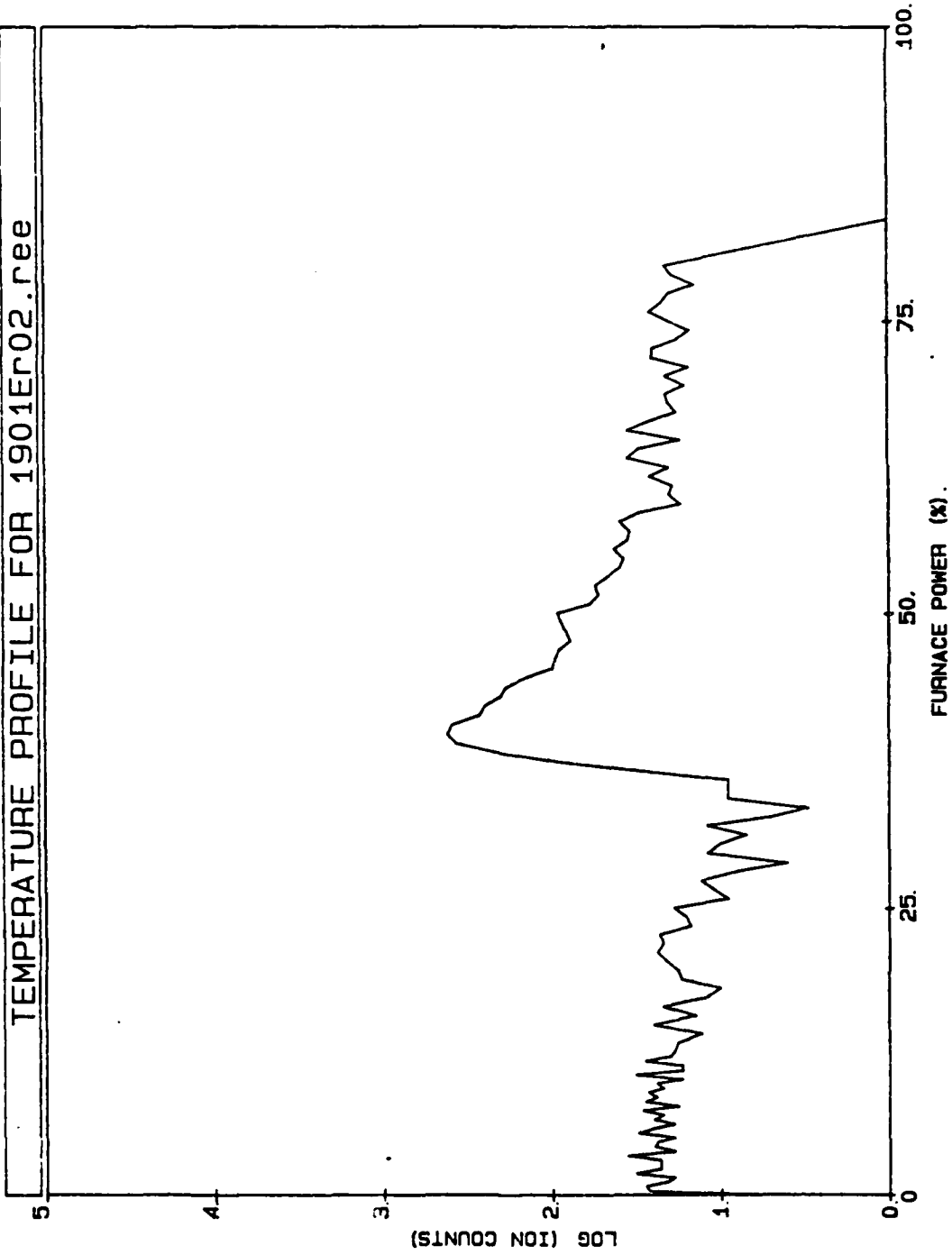
TEMPERATURE PROFILE FOR nugxp3Cd.dat



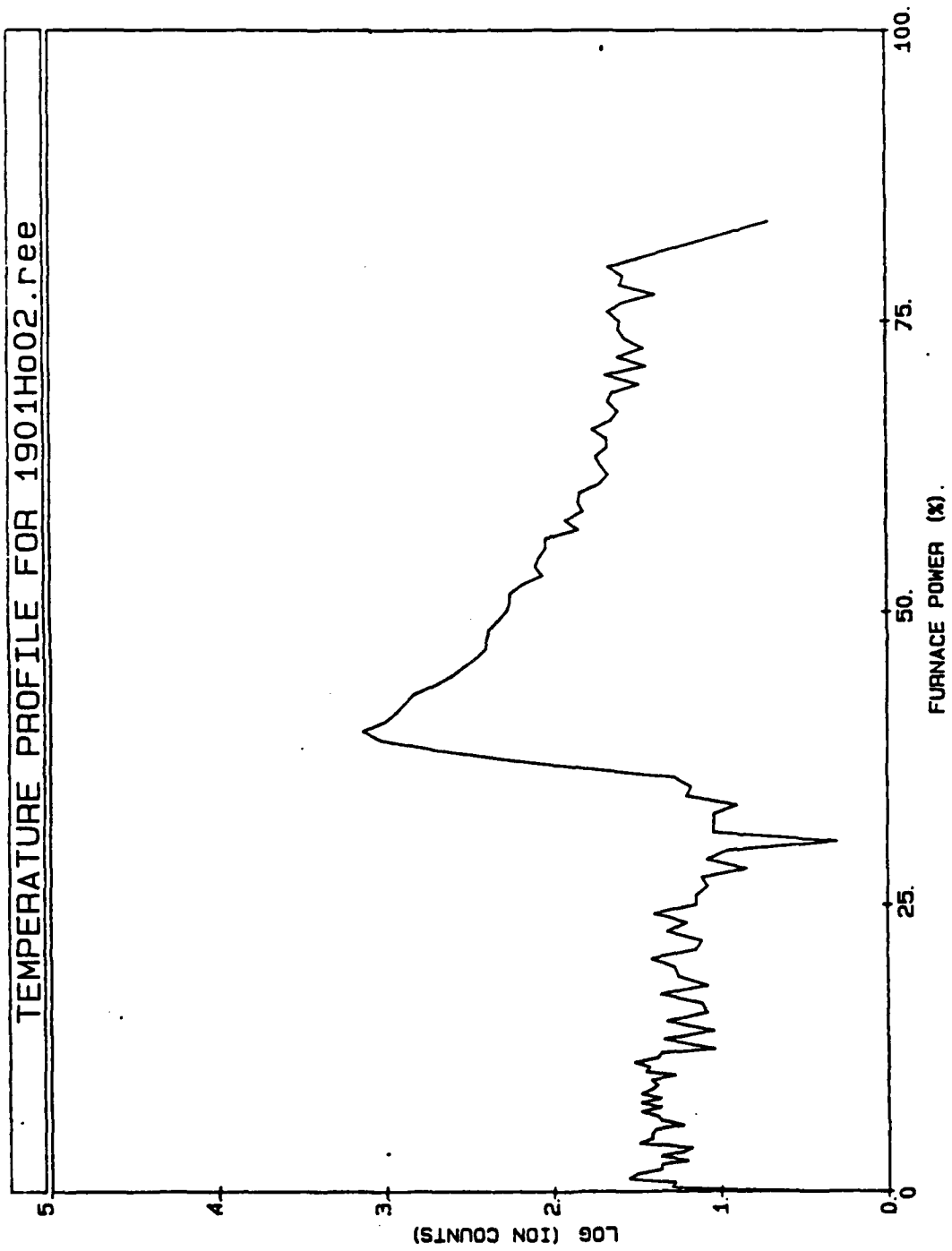
TEMPERATURE PROFILE FOR 1901Tm02.ree



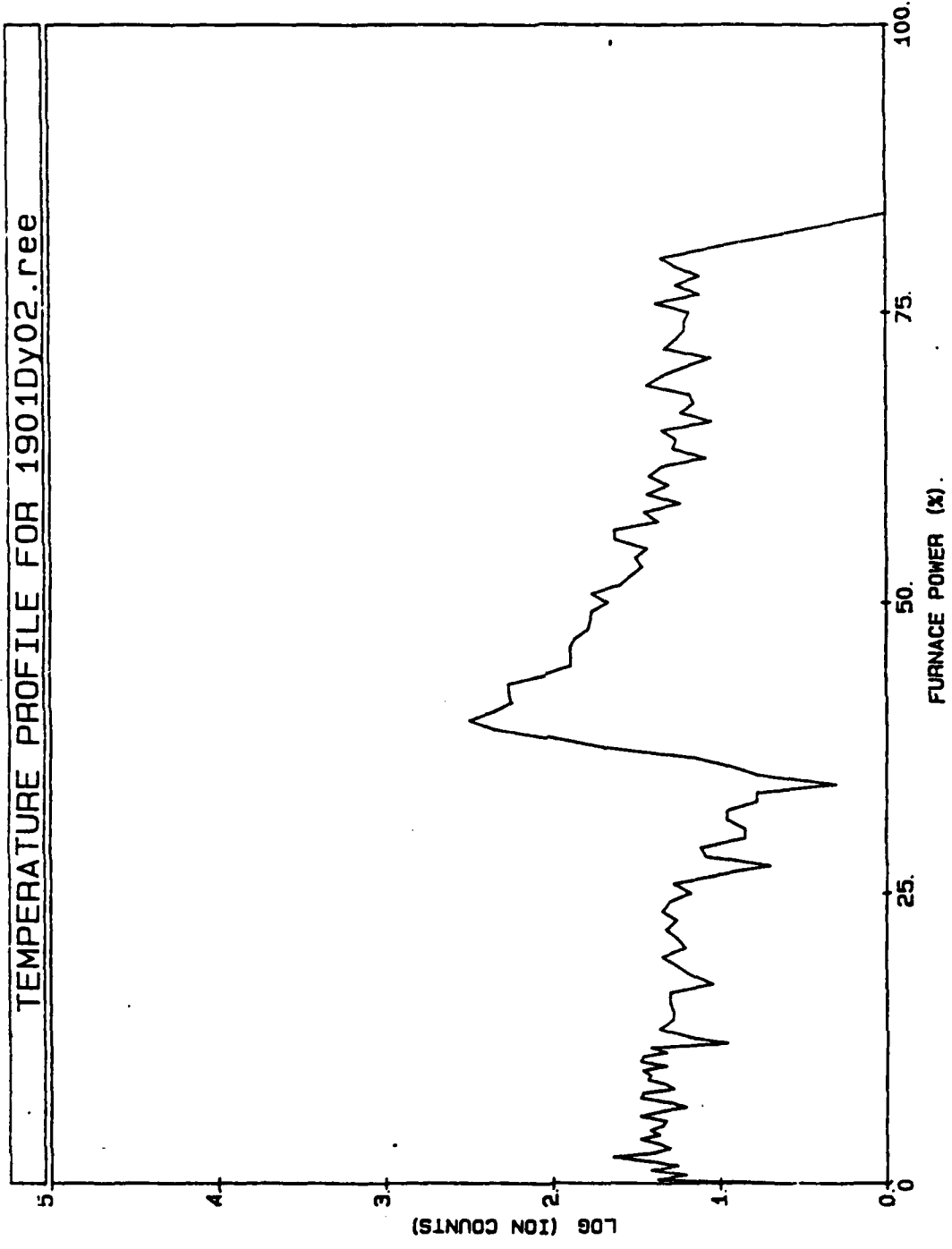
TEMPERATURE PROFILE FOR 1901Er02.rce



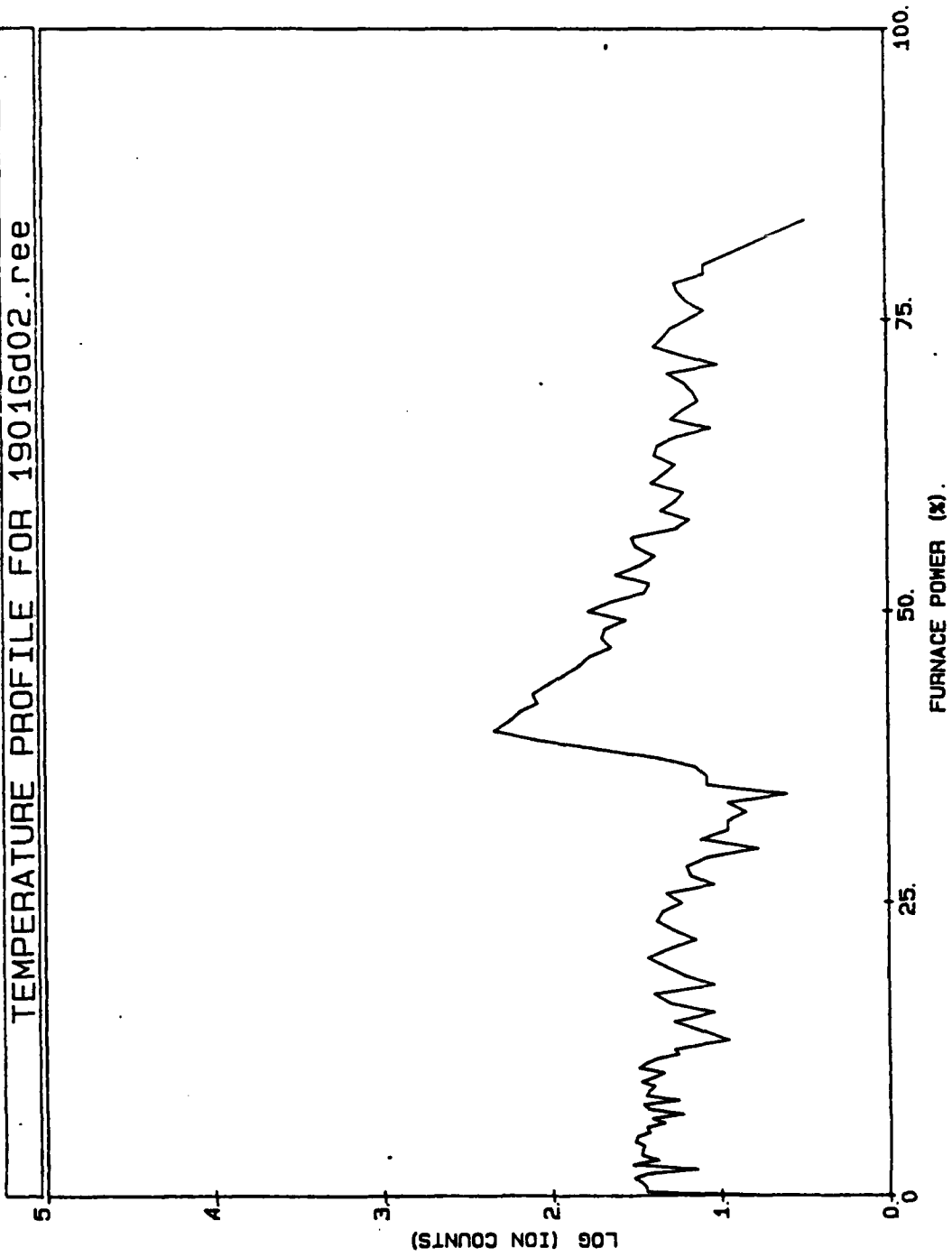
TEMPERATURE PROFILE FOR 1901H002.ree



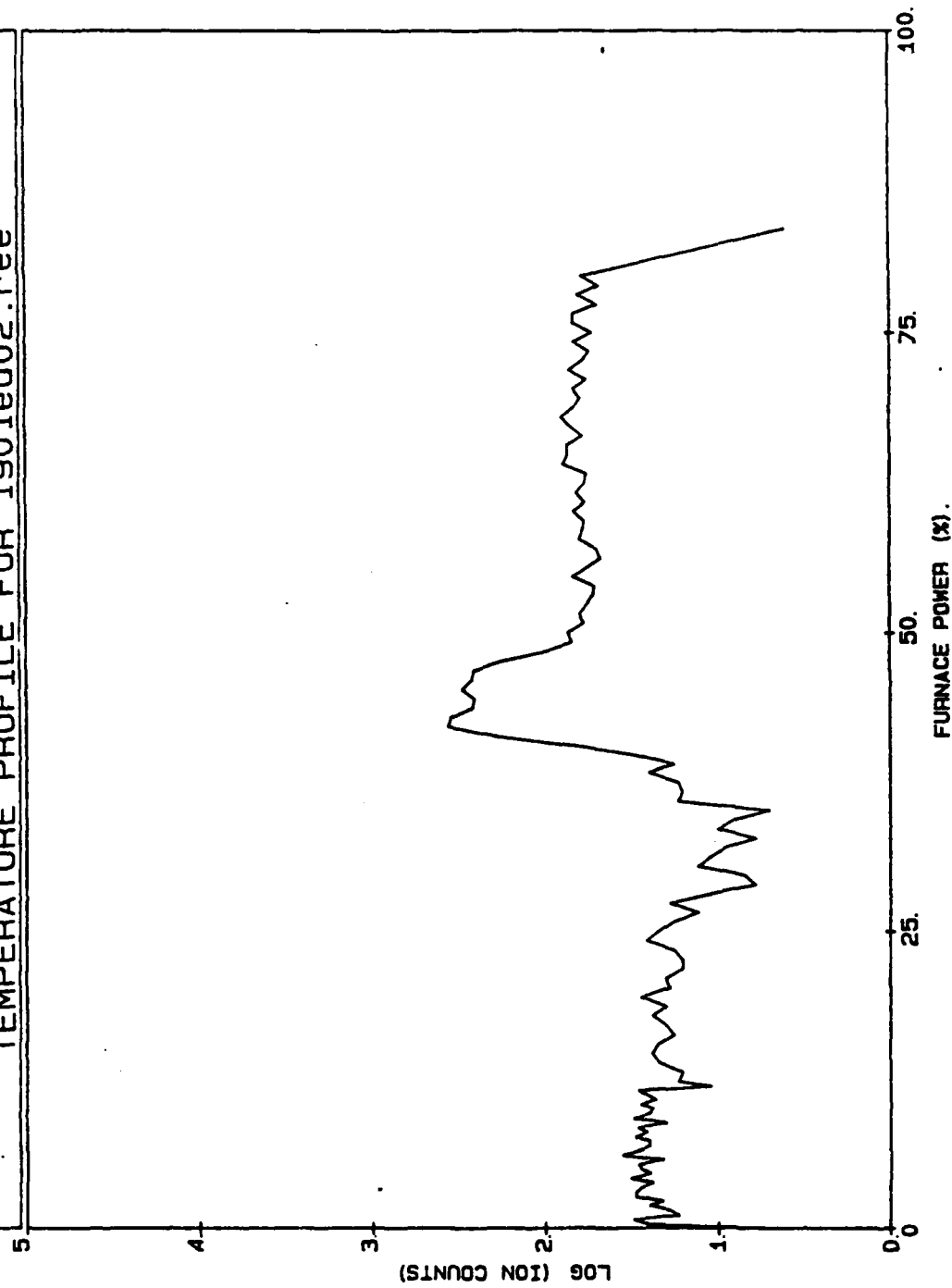
TEMPERATURE PROFILE FOR 1901DY02.ree



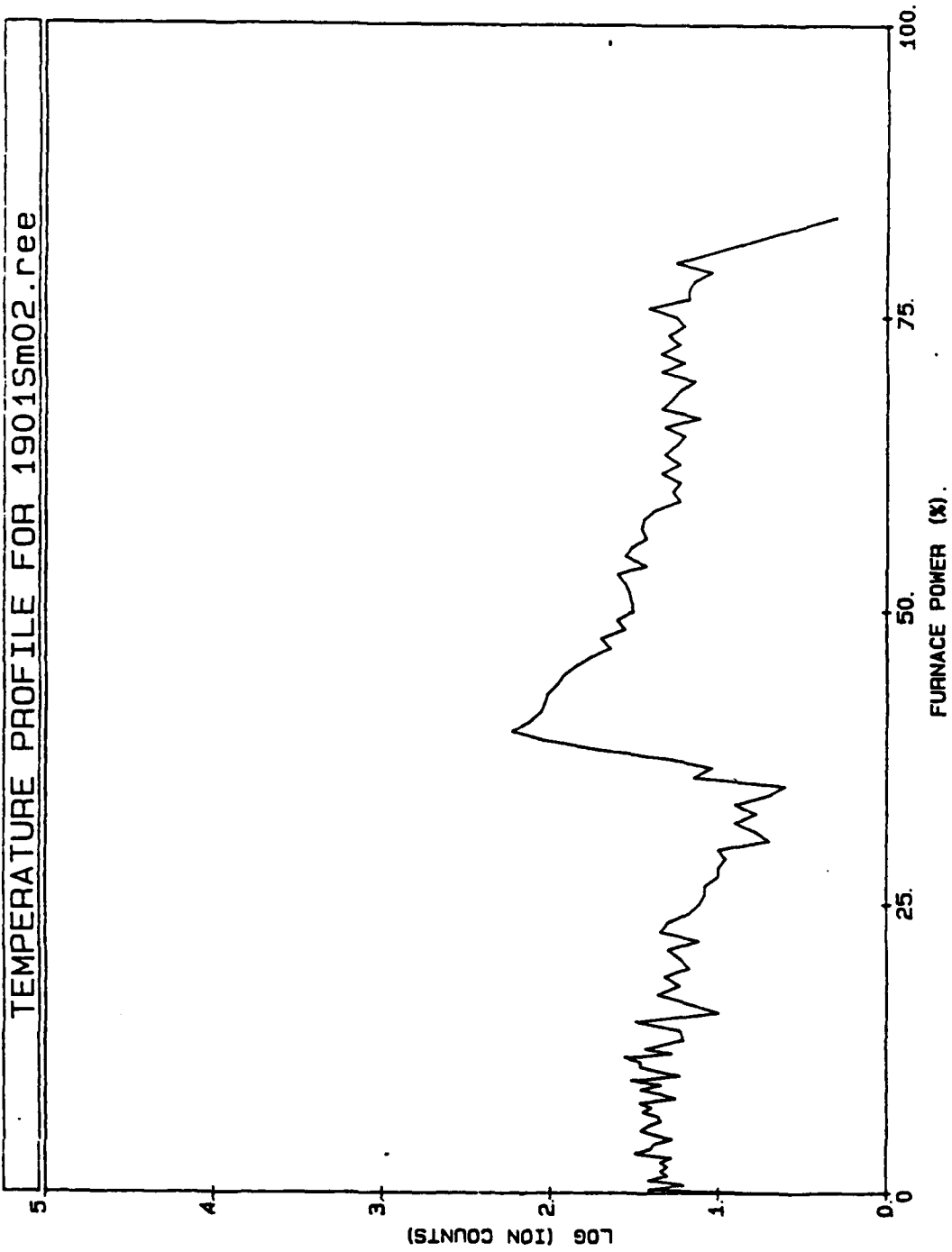
TEMPERATURE PROFILE FOR 19016d02.ree

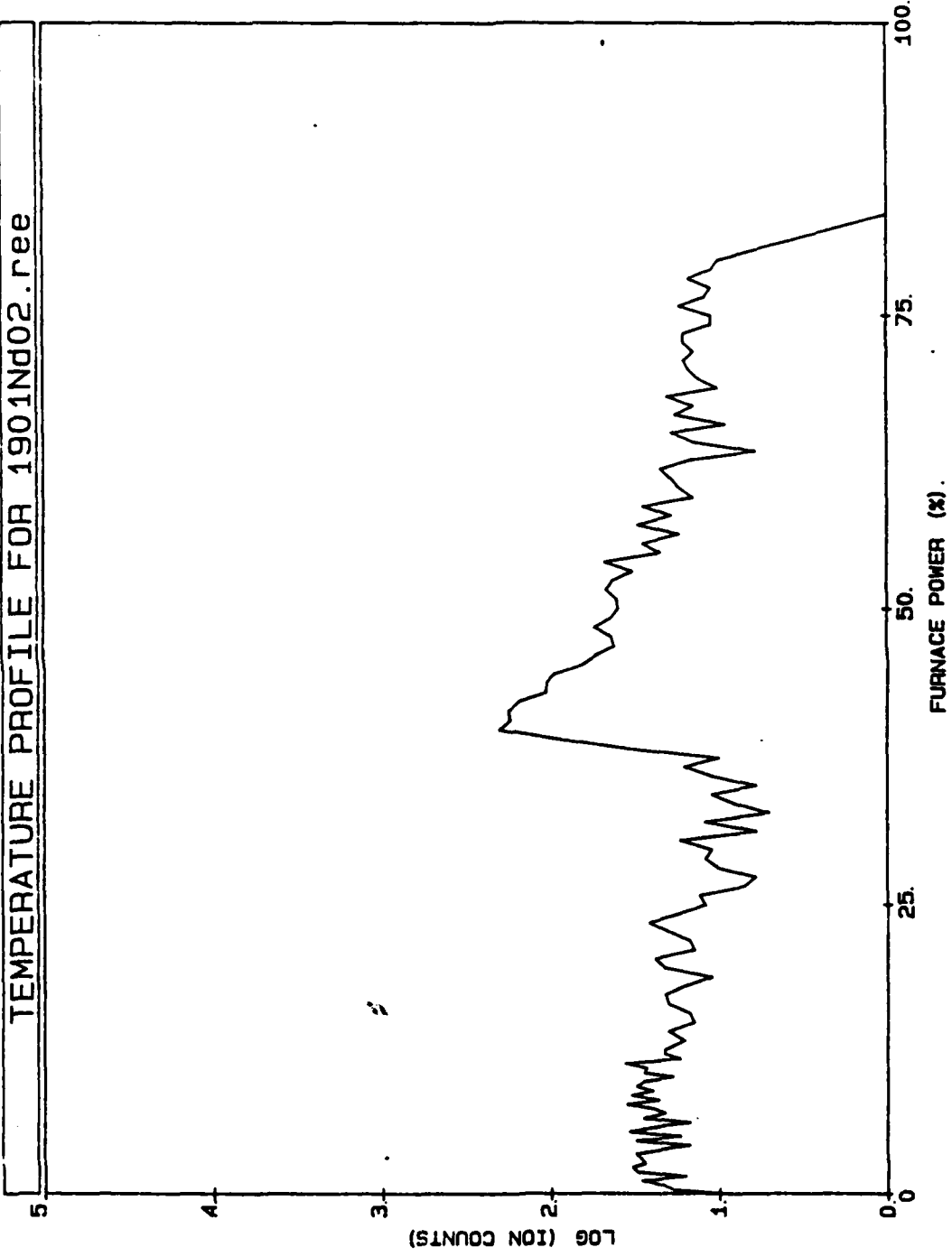


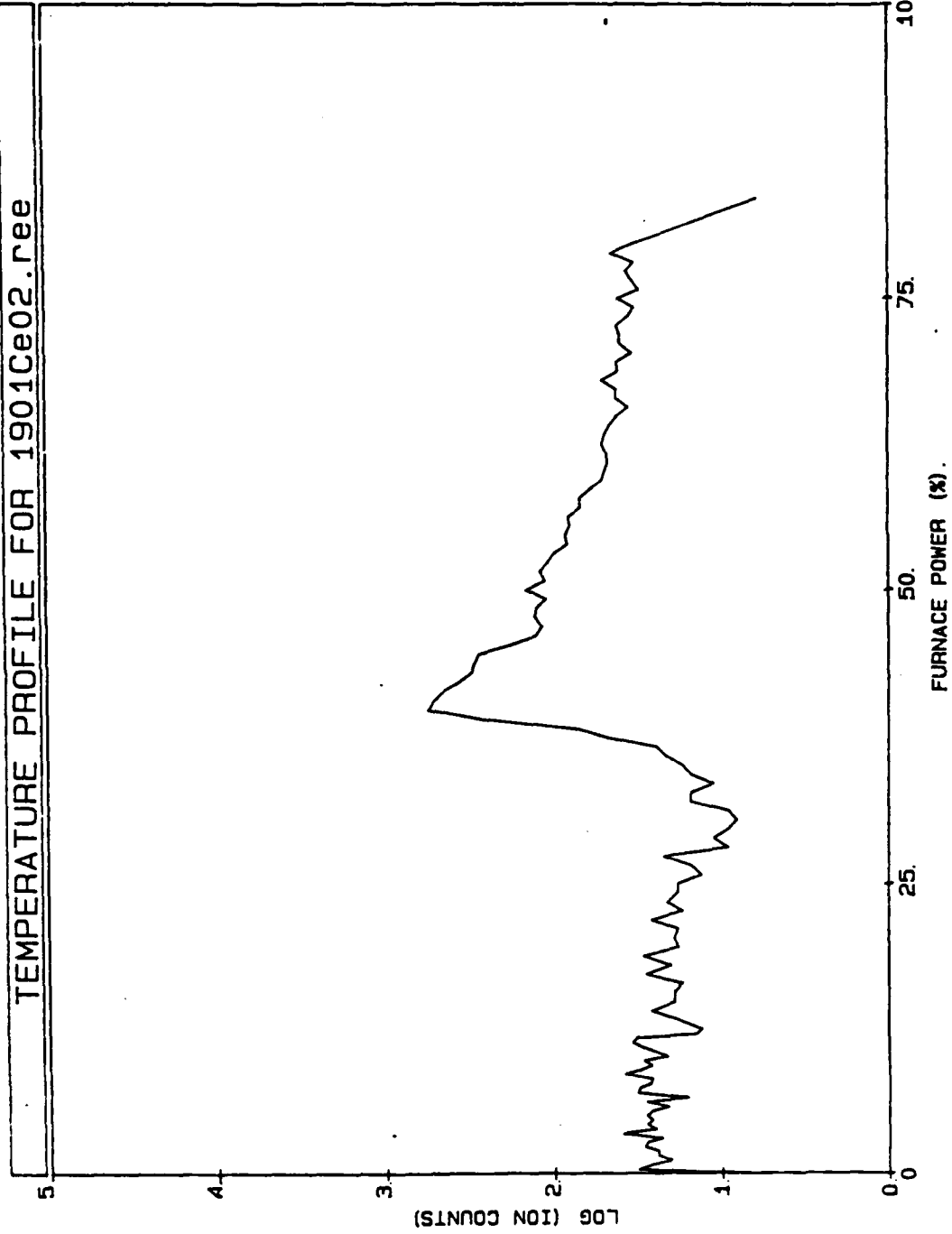
TEMPERATURE PROFILE FOR 1901eu02.ree



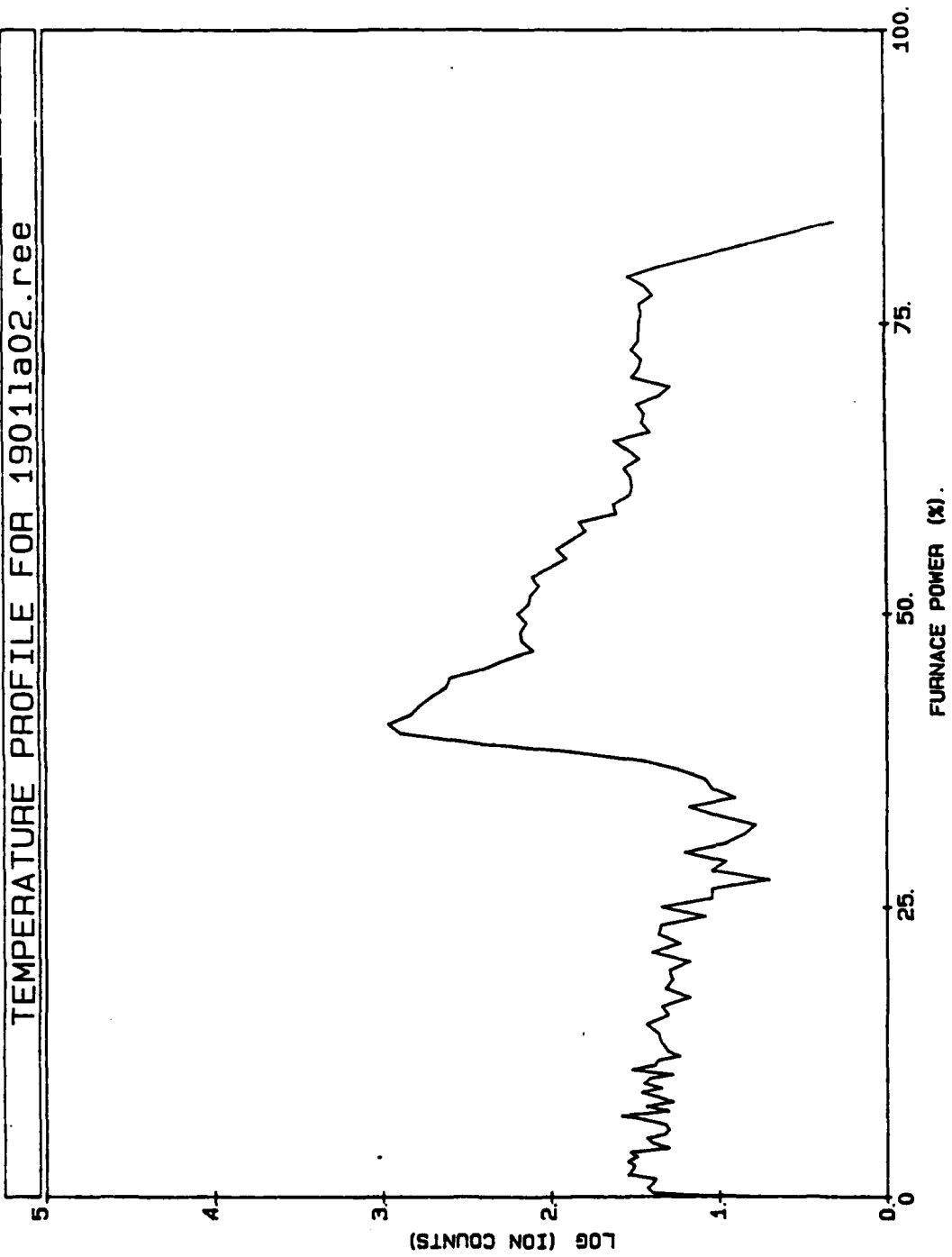
TEMPERATURE PROFILE FOR 1901Sm02.ree







TEMPERATURE PROFILE FOR 19011a02.ree



## ADDENDUM

### System Changes Since Contract Completion

#### Software

-Computer controlled selection of gas flows over sample during analysis

-Streamlining of the data collection process. Data is stored for elements and/or isotopes only if they are specified in the Method File.

-The structure of the sample analysis has been formalized into two formats a) an unknown sample and b) a uniform matrix.

a) Unknown sample. The time-temperature profile of this sample is stored and then imposed on the standard and the blank. If quantification is desired, the data file must contain a sample, a standard and a blank analysis.

b) Uniform Matrix: This assumes that the sample(s) for analysis are of a known matrix. The first sample develops the time/temperature profile curve and the analysis proceeds as follows: Time/temperature developing sample, standard, blank, sample 1, sample 2, etc.

-Development of quantification software and screen graphics capability. Data is quantified by displaying a plot of the signal vs. furnace temperature for a given atomic mass unit (amu). Integration limits for the plot are set on the screen under cursor control. These limits are then applied to all analyses in a single run (sample, standard, blank, etc.) for that amu. Similarly, limits are set for each amu of interest. A printout may be obtained for each amu containing:

- concentration +/- error
- response +/- error
- detection limits +/- error

The "error" above accounts for statistical errors in the data only. All graphics displays may be printed in hard copy.

-Scanning in both the sweep mode and single ion mode is possible.

Critical issues were the amount of data generated and saved, the type of analysis needed, a need for controlling the mixture of gases flowing at different temperatures, and how to analyze data when the time at temperature varied in the sample, standard, and/or blank. A second major development effort for the software and hardware began.

From the crude proof of principle software developed during

the first effort, a user friendly, fully functional analysis product was defined. The enhancements include:

- 1) automated data collection in two modes: Sample 1, Standard 1, Blank 1, Sample 2, Standard 2, Blank 1, Standard 1, Sample 1, Sample 2,
- 2) a simple built-in editor for the method files,
- 3) display and/or plotting of the temperature spectra,
- 4) graphic selection of the temperature region used for analysis,
- 5) analysis of the data, computing concentration and detection limits,
- 6) summation of data at 20 AMU's with individually selected half-widths for summation (not all data kept),
- 7) summation of the data was simplified to a total of 257 bins (numbered -1 for detector stabilization at 0 temperature, and 0 to 255), one for each temperature,
- 8) single ion mode,
- 9) valve control for gases,
- 10) time collection of data for non-furnace applications.

The hardware was productized from the original two board set into a single board that included 4 bits to control valves for gases.

### HARDWARE

#### Data Acquisition Board

-The data acquisition and IBM PC/XT/AT interface hardware has been developed into a finished PC board and professionally laid out and assembled.

-The electrothermal furnace power controller power supply system has been completely redesigned incorporating a switch mode solid state DC power supply. The power output is much smoother and more precise than the initial unit built for the contract.

-Design refinements in the furnace/torch assembly have been implemented, the most significant being a reduction in the amount of tantalum required to construct the furnace body.

## SOFTWARE CODE

```
dummy method text file for testing
2048 0100 channels per sweep, integration
time in usec
0020 default max sweeps per summation
buffer
0010 default min sweeps per temp
0100 default max sweeps per temp
0010 default # pulses per temp step
0210 max temp. stop when this is
exceeded
0100 initial sweeps to throw away
0000 0000 0100 0100 0001 0055 start/end temp, min/max sweeps,
temp pulses, max sweeps/sum
0001 0210 0020 0100 0010 0105
-1 start amu definitions
0002 0220 full range of amu's to scan
0002 0002 1000 starting amu, ending amu,
threshold
0003 0010 0500
0011 0226 0100
```

DEFINITION MODULE GSI,

FROM DataBase       IMPORT ShortName, LongName;  
FROM PGfiles        IMPORT DataHeaderRCD, DriveTypes,

EXPORT QUALIFIED        Config,  
                          ConfigRCD,  
                          RCFile,  
                          DetermineDaacsAndStep,  
                          AbortPQ;

TYPE    ConfigRCD =  
        RECORD  
          NumPrAv        CARDINAL;  
          NumPiAv        CARDINAL;  
          NumPersAv      CARDINAL;  
          NumMCAAv       CARDINAL;  
          PrinterNo      CARDINAL;  
          PlotterNo      CARDINAL;  
          PeriphNo      CARDINAL;  
          NumAutoS       CARDINAL;  
          AutoNo         CARDINAL;  
          RackCode       CARDINAL;  
          NumXTubes      CARDINAL;  
          NumY Tubes     CARDINAL;  
          RackYi         CARDINAL;  
          RackXi         CARDINAL;  
          RackYStep      CARDINAL;  
          RackXStep      CARDINAL;  
          Height         CARDINAL;  
          UpTime         CARDINAL;  
          WTime         CARDINAL;  
          Drive         DriveTypes,  
          MCANo         CARDINAL;  
          AutoShut       BOOLEAN,  
          DummyA,  
          DummyB         BOOLEAN,  
          DeadTime       REAL,  
          SkipMasses     ARRAY [1..36] OF REAL  
        END,

VAR Config                                ConfigRCD,

PROCEDURE DetermineDaacsAndStep(VAR HeaderBlock   DataHeaderRCD),  
PROCEDURE RCFile,  
PROCEDURE AbortPQ,

END GSI

page 65,132

TITLE GET DATA

GET DATA Summation and Storage Program

Written by J R Hanratty

Updated 16-Nov-1986

Updated 24-Dec-1986

Updated 11-Jan-1987

Updated 13-Jan-1987

Copyright 1986 by James R Hanratty and Geochemical Services, Inc.

325 Mariposa Drive  
Camarillo, CA 93010  
(805) 484-1414

ALL RIGHTS RESERVED

No\_Buffers = 14 , number of buffers reserved to hold data  
old No\_Buffers = 21

hardware definitions

smaxt equ 0b000H+(19\*10)+4  
stemp equ 0b000H+(20\*10)+4 , crt locations for  
text messages  
stimet equ 0b000H+(21\*10)+4  
stime equ 0b000H+(22\*10)+4  
schan equ 0b000H+(23\*10)+4

port\_a1 EQU 3A0H , control out  
A1\_I\_ENB EQU 01H , IBM START SCAN  
A1\_NEXT EQU 02H , IBM NEXT CHANNEL  
A1\_TEMP EQU 04H , STEP TEMP  
A1\_RESET EQU 08H , RESET  
A1\_I\_DIS EQU 00H , INTERRUPTS DISABLE, 1ST OF 3 BITS  
A1\_I\_1 EQU 11H , INTERRUPTS ENABLE ON EVERY 1  
A1\_I\_2 EQU 21H , INTERRUPTS ENABLE ON EVERY 2  
A1\_I\_4 EQU 31H , INTERRUPTS ENABLE ON EVERY 4  
A1\_I\_8 EQU 41H , INTERRUPTS ENABLE ON EVERY 8  
A1\_I\_S EQU 71H , INTERRUPTS ENABLE ON EVERY START

port\_b1 EQU port\_a1 + 1 , control in  
B1\_FULL EQU 01H , FIFO FULL  
B1\_NEAR EQU 02H , FIFO NEAR FULL  
B1\_HALF EQU 04H , FIFO HALF FULL  
B1\_EMPTY EQU 08H , FIFO EMPTY  
B1\_ERROR EQU 10H , WRITE ERROR

port\_c1 EQU port\_b1 + 1 , not used  
port\_l EQU port\_c1 + 1 , control word 82h mode 0

```

port_a2 EQU port_1 + 1 , lsb
port_b2 EQU port_a2 + 1 , msb
port_c2 EQU port_b2 + 1 , hand shaking
C2_BUFFER_FULL EQU 02 , THERE IS DATA IN THE BUFFER
port_2 EQU port_c2 + 1 , control word B6H MODE 1
p2_INTR EQU 05H , INTERRUPT on

HEADER EQU 0
flags EQU 0 , flags (hi 4 bits) +
, data length (low 12 bits)
flags_bad_size equ 08000h , data buffer wrong length
flags_data_over_run equ 04000h , no free data buffer
flags_error2 equ 02000h
flags_error1 equ 01000h

TEMP EQU 2 , TEMP FOR THE CURRENT BUFFER
START_TIME EQU 4 , STARTING TIME IN SWEEPS
, FROM START OF SUMMATION
ELAPSED_TIME EQU 6 , ELAPSED TIME IN BUFFER IN SWEEPS
data_offset EQU 8 , offset to start of data in buffer

buffer_length_b equ (511+data_offset+8192) and 0fe00h , round
up to next
buffer_length_w equ buffer_length_b/2 , sector
boundary

public summation
SUMIT SEGMENT PARA 'CODE'
ASSUME CS:SUMIT, DS:SUMIT, ss:a_stack
; assume es:dummy , debug code only
; DATA
;
active_flag dw 0 , status flag for data collection
, 0 -> inactive,
, 1 -> collecting

summation_buffer_ptr dd 0 , -> summation buffer
data_capture_ptr dd 0 , -> data capture

buffer_index dw 0 , index into tables: 4 bytes

buffer_chain DD No_Buffers dup (0)

MAX_sweeps_Sum DW 256 DUP (0) , max allowed sweeps per
summation

Max_Sweep_Temp dw 256 dup (0) , max sweeps per temp
, while above threshold or
not
Min_Sweep_Temp dw 256 dup (0) , min sweeps per temp,
regardiess

```

```

Temp_Ramp          dw  256 dup (0) , of the thresholds
commanded step    , # temp pulses per com-

THRESHOLDS        DD  2048 DUP (0) , the thresholds

Summation_Flags   DD  2048 dup (0) , data capture and summation
flags

Commanded_Temp    dw  0 , commanded temperature
new_temp          dw  0 , new temp setting

MIN_SWEEP         DW  0 , min sweeps/temp
TEMP_INC_PULSES   DW  0 , # temp inc pulses
max_sweep_buffer  dw  0 , max # sweeps/buffer

```

, the following words must be contiguous

```

CHANNEL_PER_sweep DW  0 , size of data requested
Max_Temp          dw  0 , Commanded stopping Temp
MAX_SWEEP         DW  0 , max sweeps/temp
first_sector      dw  101 , first sector on drive for
data
disk_drive        dw  2 , the disk drive to use

```

,end of sacred area, caused by the order of copy from the callers  
buffer

```

inhibit_temp_inc  dw  0 , data is above threshold
Channel_Threshold dw  0, 0 , exceeded channel and
threshold
Time_at_Temp      dw  0 , elapsed time at tempera-
ture
temp_bumped       dw  0 , the temp has been bumped
CHANNEL_PER_sweep_4 DW  0 , size of data requested*4
starting_sector   dw  -1 , the sector for next write

```

```

time_Prompt       db  'xxxxx' Current sweep',0
temp_Prompt       db  'xxxxx' Current temp',0
Max_Temp_prompt   db  'xxxxx' Commanded Maximum temp',0
sameTime_Prompt   db  'xxxxx' Sweeps at current temp',0
Channel_Prompt    db  'xxxxx xxxxx' Channel and threshold
exceeded',0

```

START PROC

SUMMATION

```

mov sp, offset a_stack top_stack , setup a stack
mov ax, seg a_stack
mov ss, ax
PUSH CS , SETUP DATA SEGMENT
POP DS

```

```

MOV      AH,37          , SET UP INTERRUPT VECTOR 4Fh
MOV      AL,4fH        , CHANGE INTERRUPT REQUEST
MOV      DX, OFFSET SUMIT.INT_ENTRY
INT      33

mov      ai,082H       , set up ports
mov      dx,port_1
out      dx,AL

mov      ai,0B6H
mov      dx,port_2
out      dx,AL

mov      ai,p2_INTR    , enable interrupts
mov      dx,port_2
out      dx,AL

mov      ai,AI_RESET   , reset the hardware to start new
mov      dx,port_ai
out      dx,AL

mov      dx,port_ai
xor      ax,ax        , clear reset pulse and temp
out      dx,AL

mov      dx,seg a_stack , set up pointers in tables
add      dx,00ffff     , cross next 64k boundary
and      dx,0f000h     , and fall back to the boundary
add      dx,2000h      , reserve two 64k buffers
, old    add      dx,3000h , reserve three 64k buffers

exit2
mov      ax,03100h     , function code and return status
INT      33

exit    mov      dx,seg summation
       jmp      exit2

```

```

INT_ENTRY:      SOFTWARE INTERRUPT ENTRY
                CMP      AL,0          IDENTIFY THE FUNCTION CODE
                JZ       START_SUM_R   , START
                CMP      AL,2          , DISK READ
                JZ       DISK_READ
                CMP      AL,3          , DISK WRITE
                JZ       DISK_WRITE
                cmp      al,4          , get status
                jz       Get_status    , yes
                cmp      al,5          , terminate and stay resident,
remove our code
                jz       exit
                CMP      AL,1          , STOP
                jnz      summation_exit
                Jmp      STOP_SUM      , YES

START_SUM_R:
                JMP      START_SUM

DISK_READ:
                CALL     READ_DISK
summation_exit
                IRET          , UNKNOWN RETURN

DISK_WRITE:    CALL WRITE_DISK
                JMP SUMMATION_EXIT

get_status:
                sti
                push     bx
gs_1:
                mov     bx,cs:active_flag    , collection status
                or      bx,bx
                jnz     gs_1

                pop     bx
                jmp     summation_exit

Disp_Temp:
                push    ax
                push    bx
                push    cx
                push    dx
                push    si
                push    di
                push    ds
                push    es
                mov     ax, stemp
                mov     es, ax
                xor     bx, bx
                mov     ax, cs Commanded_Temp

```

```

call    i2bin
pop     es
pop     ds
pop     si
pop     di
pop     dx
pop     cx
pop     bx
pop     ax
ret

```

#### Disp\_Max\_Temp

```

push    ax
push    bx
push    cx
push    dx
push    di
push    si
push    ds
push    es
mov     ax,smaxt
mov     es,ax
xor     bx,bx
mov     ax,cs: Max_Temp
call   i2bin
pop     es
pop     ds
pop     si
pop     di
pop     dx
pop     cx
pop     bx
pop     ax
ret

```

#### Disp\_Times

```

push    ax
push    bx
push    cx
push    dx
push    di
push    si
push    ds
push    es
les     bx,cs summation_buffer_ptr
mov     ax,es start_time[bx] , and time
add     ax,es elapsed_time[bx]
mov     bx,stime
mov     es,bx
xor     bx,bx
call   i2bin

```

```

mov     ax,cs    Time_At_Temp
mov     dx,StimeT
mov     es,dx
xor     bx,bx
call    i2bin
pop     es
pop     ds
pop     si
pop     di
pop     dx
pop     cx
pop     bx
pop     ax
ret

```

Disp\_Thresh:

```

push   ax
push   bx
push   cx
push   dx
push   di
push   si
push   ds
push   es
xor    bx,bx
mov    ax,cs:Channel_Threshold
mov    cx,schan
mov    es,cx
call   i2bin

mov    ax,cs:Channel_Threshold+2
mov    bx,schan
mov    es,bx
mov    bx,12
call   i2bin
pop    es
pop    ds
pop    si
pop    di
pop    dx
pop    cx
pop    bx
pop    ax
ret

```

mov\_text\_crt:

```

mov     es,di
xor     di,di
mt2:
mov     al,cs:[si]      ; get the next character
or      al,al          ; done?
jnz    mt1              ; no

```

```

ret

mt1: mov     es [di], al
      inc     si
      add     di, 2
      jmp     mt2

I2BIN:  , INPUTS AX      = INTEGER TO BE DISPLAYED
        , ES:BX      = STARTING POSITION IN DISPLAY

        MOV     DL, ' ' , SET THE STARING STRING TO
BLANKS
        MOV     CX, 10000 , GET FIRST DIGIT, TENS OF
THOUSANDS
        CALL    GET_DIGIT

        MOV     CX, 1000 ,
        CALL    GET_DIGIT , THOUSANDS

        MOV     CX, 100 ,
        CALL    GET_DIGIT , HUNDREDS

        MOV     CX, 10 ,
        CALL    GET_DIGIT , TENS

        OR      AL, '0' , INSURE UNITS ARE SHOWN AS
INTEGER,
        MOV     ES, [BX], AL , NO LEADING BLANKS
        RET

GET_DIGIT:
        MOV     DH, DL , SET DEFAULT CHARACTER
GET_DIGIT_1:
        CMP     CX, AX , DO WE HAVE A DIGIT?
        JA      GET_DIGIT_3 , NO, DONE
        SUB     AX, CX
        OR      DX, '00' , SET TO DIGIT
        ADD     DL, 1 , BUMP DIGIT
        JMP     GET_DIGIT_1 , MORE?
GET_DIGIT_3:
        MOV     ES [BX], DL , OUTPUT THE CHARACTER
        MOV     DL, DH
        ADD     BX, 2 , POINT TO THE NEXT CHARACTER
POSITION
        RET

, ax = 1
dp_sum_ptr EQU 00 , summation flags buffer pointer
dp_min_ptr EQU 04 , thresholds ptr
dp_Ramp_ptr EQU 08 , Temp ramp pointer

```

```

dp_min_sweep_p EQU 12 ,      min sweeps/temp
dp_max_sweep_p EQU 16 ,      max sweeps/temp
dp_max_ptr     EQU 20 ,      max sweeps per summation ptr
dp_channels    EQU 24 ,      channels/sweep
dp_Max_temp    EQU 26 ,      max commanded temp
dp_Throw_Away equ 28 ;      # sweeps to throw away at 0 temp

```

```

; es sec pointer

```

```

START_SUM:

```

```

sti
cld      , all transfers in the forward direction
push    bz      ; save context
push    cx
push    dx
push    si
push    ds
push    es
push    di

push    es      , save these for now
push    di

mov     ax,cs
mov     ds,ax      ; point to the proper segment
mov     ah,37      , hardware interrupt setup
mov     al,08h+3   , vector to our summation code
mov     dx,offset sumit:hard_int , our entry
int     33

mov     commanded_Temp,0 , start temp ramp from 0

pop     di      , get pointer to our data
pop     es      , and segment info

push    di      , save for later
mov     bx,di   , copy

; copy control variables Summation_flags, Thresholds,
;                          Temp_Ramp,
;                          min_sweep_temp, max_sweep_temp,
;                          Max_sweeps_sum, channel_per_sweep,
;                          Max_temp, throw_away

lea     si,es:dp_channels[di] , starting location
mov     di,offset sumit:channel_per_sweep
mov     cx,10      , 5 words = 10 bytes

mov     ax,es
mov     ds,ax
mov     dx,cs
mov     es,dx

```

```

rep movsb ; move all 2 words

; copy summation flags in
;
mov     di,bx ; get parameter index
mov     ds,ax
lds     si,ds:dp_sum_ptr[di] ; target address
mov     di,offset sumit:Summation_flags ; source address
mov     cx,size Summation_flags ; length of array
rep movsb ; move the array

; copy max sweeps/sumation
;
mov     di,bx ; index to parameters
mov     ds,ax
lds     si,ds:dp_max_ptr[di] ; source address
mov     di,offset sumit:max_sweeps_sum ; destination
mov     cx,size max_sweeps_sum ; length
rep movsb ; move the array

; copy thresholds
;
mov     di,bx ; index to parameters
mov     ds,ax
lds     si,ds:dp_min_ptr[di] ; source address
mov     di,offset sumit:thresholds ; destination address
mov     cx,size thresholds ; length
rep movsb ; move the array

; copy temp ramp
;
mov     di,bx ; index to parameters
mov     ds,ax
lds     si,ds:dp_ramp_ptr[di] ; source address
mov     di,offset sumit:Temp_Ramp ; destination address
mov     cx,size Temp_ramp ; length
rep movsb ; move the array

; copy Min Sweeps per temp
;
mov     di,bx ; index to parameters
mov     ds,ax
lds     si,ds:dp_min_sweep_p[di] ; source address
mov     di,offset sumit:Min_sweep_Temp ; destination
address
mov     cx,size Min_Sweep_Temp ; length
rep movsb ; move the array

```

```

,copy Max Sweeps per temp
;
    mov     di,bx                ; index to parameters
    mov     ds,ax
    lds     si,ds:dp_max_sweep_p[di] ; source address
    mov     di,offset sumit.Max_sweep_Temp ; destination
address
    mov     cx,size Max_Sweep_Temp ; length
    rep movsb                    ; move the array

    mov     es,ax                ; es -> parameters

    mov     ax,cs:Max_Sweep      ; set max and mins for
1st sweep
    mov     cs:min_sweep,ax
    mov     cs:Max_sweep_Buffer,ax ; and sum them all up
    mov     cs:Temp_Inc_Pulses,0 ; inhibit temp inc for
first step

    mov     ax,seg a_stack       ; set up pointers in tables
    add     ax,00ffff            ; cross next 64k boundary
    and     ax,0f000h           ; and fall back to the boundary

    mov     bx,cs
    mov     ds,bx
    mov     es,ax                ; use the new segment address of the
buffers

    mov     buffer_index,0      ; point to the first buffer index
    mov     word ptr summation_buffer_ptr,BUFFER_LENGTH_B ;
summation -> DUMMY
    mov     word ptr summation_buffer_ptr+2,ax
    mov     word ptr data_capture_ptr,0 ; data capture -> 1ST
REAL
    mov     word ptr data_capture_ptr+2,ax

    push    ds
    push    ax

; old    mov     dx,3            ; show three buffers to clear
    mov     dx,2                ; show two buffers to clear
    mov     ds,ax

LL4: mov     cx,08000h          ; length of each buffer in words
    mov     ax,0                ; data value for start
    mov     bx,0                ; initial index

LL5: mov     ds:[bx],ax
    add     bx,2
    lopp   LL5

    mov     ax,ds                ; get and bump data seg

```

```

add      ax,1000h
mov      ds,ax
dec      dx
jnz      LL4

pop      ax
pop      ds

MOV      word ptr es:(buffer_length_b+TEMP),-1
          ; MARK THE CURRENT TEMP
          ; OF DUMMY SUMMATION AS -1

mov      word ptr es:flags,0
mov      word ptr es:temp,0
mov      word ptr es:start_time,0
mov      word ptr es:elapsed_time,0

mov      word ptr es:flags+buffer_length_b,0
mov      word ptr es:start_time+buffer_length_b,0
mov      word ptr es:elapsed_time+buffer_length_b,0

mov      bx,offset buffer_chain
call     set_pointers
add      ax,1000h
call     set_pointers
,       add      ax,1000h
,       call     set_pointers

mov      ax,CHANNEL_PER_sweep
add      ax,ax , * 4
add      ax,ax
sub      ax,10 ; remove a few sweeps just in case
mov      CHANNEL_PER_sweep_4,ax

mov      ax,first_sector          , init disk xfer
address
inc      ax
mov      starting_sector,ax      , the first is really
for header

pop      di

mov      al,082H          , set up ports
mov      dx,port_1
out      dx,AL

mov      al,0B6H
mov      dx,port_2
out      dx,AL

mov      al,p2_INTR      ; enable interrupts

```

```

mov     dx,port_2
out     dx,AL

mov     al,A1_RESET      , reset the hardware to start new
mov     dx,port_a1
out     dx,AL

xor     ax,ax            ; remove the reset signal
out     dx,al

x:      mov     dx,port_c2      , clear any data that's left
in      al,dx
and     al,C2_Buffer_Full
jz      a
dec     dx
in      al,dx            , msb
dec     dx
in      al,dx
jmp     x

a:

mov     dx,port_a1
mov     al,A1_I_s        , clear reset pulse and arm interrupts
out     dx,AL

xor     ax,ax            , 0 -> interrupt mask reg of 8259
mov     dx,21h
out     dx,al

mov     di,stemp         , label the data area
mov     si,OFFSET temp_prompt      ; current temp prompt
call   mov_text_crt

mov     di,stime
mov     si,OFFSET time_prompt      , current temp prompt
call   mov_text_crt

mov     di,stimet
mov     si,OFFSET sametime_prompt   , current temp prompt
call   mov_text_crt

mov     di,schan
mov     si,OFFSET channel_prompt    , current temp prompt
call   mov_text_crt

mov     di,smaxt
mov     si,OFFSET max_temp_prompt   , current temp prompt
call   mov_text_crt

xor     ax,ax            , zero initial displays
mov     cs,Commanded_Temp,ax
mov     cs,Channel_Threshold+0,ax

```

```

mov     cs, Channel_Threshold+2, ax
mov     cs, Time_At_Temp, ax

call    Disp_Thresh
call    Disp_Temp
call    Disp_Times
call    Disp_Max_Temp

mov     cs:active_Flag, 1 ; show active

pop     di
pop     es
pop     ds
pop     si ; restore context
pop     dx
pop     cx
pop     bx

jmp     summation_exit

set_pointers:
mov     cx, 7
xor     dx, dx
17:    mov     cs:[bx], dx
mov     cs.2[bx], ax
add     bx, 4
add     dx, buffer_length_b
loop   17
ret

STOP_SUM
int     08h+3 ; fake data ready to collect any
data
" mov   ax, ai_reset ; reset temp and turn interrupts
off
mov     dx, port_ai
out     dx, al
xor     ax, ax ; turn off interrupts
out     dx, al
call    flush_to_disk

mov     cs:active_Flag, 0 ; show inactive

jmp     summation_exit

hard_int:
cld ; all transfers in + direction
push   di
push   si ; save context
push   ds
push   dx

```

```

push    ax
push    bx
push    es
push    cx

lds     bx,cs:data_capture_ptr
les     di,cs:summation_buffer_ptr

xor     ax,ax      , turn off interrupts
mov     dx,port_a1
out     dx,a1

mov     al,20h    , send eoi to 8259
out     20h,a1
sti     ,allow other interrupts to occur

more_data
mov     si,ds:flags[bx] , index into data
more_data_2:
mov     dx,port_c2    , test for errors and data
in     al,dx
test    al,c2_buffer_full ; is there data?
jz     more_data_2    ; jrh debug loop until done
      jz     empty    , no, so why the interrupt?

dec     dx          ; port_b2 to get msb
in     al,dx
mov     ah,al      ; put in msb of result
dec     dx          ; get lsb form port_a2
in     al,dx

or     ax,ax      , if 2 ms bits on, then new
frame
js     more_data_1 , new frame

cont:
mov     ds:data_offset[bx][si],ax , store the data
mov     word ptr ds:data_offset+2[bx][si],0
cmp     WORD PTR cs:thresholds[si],ax ; is this too high?
;     jns    cont_1    ; no, below threshold, get it
      jae    cont_1    , no, below threshold, get it
cmp     cs:inhibit_temp_inc,1 ; all ready flagged?
      je     cont_1    , yes

mov     cs:inhibit_temp_inc,1 ; inhibit possible temp inc

mov     word ptr cs:Channel_Threshold+0,ax , jrh debug
value exceeding thresholds
mov     word ptr cs:Channel_Threshold+2,si , jrh debug
and the index too
sar     word ptr cs:Channel_Threshold+2,1
sar     word ptr cs:Channel_Threshold+2,1

```

```

        call    disp_Thresh

cont_1:
        add     si,4           , bump data pointer
        jmp     more_data_2   , not new, continue summation
more_data_1:
        and     ax,03ffff     , clear two top bits from temp
        or      si,si        , if index = zero, continue
summation
        jz      more_data_3   , no data, begin collection
        cmp     cs:channel_per_sweep_4,si , continue collecting
data?
        jg     more_data_2   , yes, we found a fast sweep
break

        call   sum_new       , test and sum or flush as
needed
        mov    cs:inhibit_temp_inc,0 , remove inhibit possible
temp inc
        jnc   more_data     , and begin looking for data
        jmp   empty_and_done

more_data_3:
        mov    ds:temp[bx],ax   , show the temp
        jmp   more_data_2     , and start collecting the data

empty
        mov    ds:flags[bx],si , save pointer

        mov    dx,port_a1
        mov    al,A1_I_s      , arm interrupts
        out   dx,AL

empty_and_done:

        pop    cx
        pop    es
        pop    bx
        pop    ax
        pop    dx
        pop    ds
        pop    si
        pop    di
        jmp   summation_exit

```

```

, subroutines
,

```

```

new_temperature:
    mov     cs:Time_at_Temp,0      , start timing again
    mov     cs:temp_bumped,0      , show new start
new_temperature_2:
    mov     ax,cs:new_temp        , save the new temp for data
summation
    mov     ds:temp[bx],ax
    call   disp_Times
    cld
    ret

sum_new: ; if summation buffer and data_capture are the same,
sum it up
; if not same, send summation, and start new summation
with data_capture
    mov     ds:flags[bx],si      , save the index into the data
buffer
    mov     cs:new_temp,ax       , save new temp
    les     di,cs:summation_buffer_ptr
    test    es:flags[di],3fffh   ; data in summation
buffer?
    jz      no_data              , no, use data as is, no
summing
    mov     ax,ds:temp[bx]
    cmp     es:temp[di],ax       ; if temp matches
    jz      same_temp            ; continue summation
    cmp     cs:max_temp,ax       , are we now done?
    jg     no_data              , no, continue
    call   flush
    call   flush_to_disk        , yes we are done, finish
and exit
    mov     al,a1_Reset          , kill temp and all
    mov     dx,port_a1
    out    dx,al
    xor     ax,ax
    out    dx,al

    mov     cs:active_Flag,0     , show inactive

    stc
    ret                          , show done
                                , return

no_data:
    call   flush                 , otherwise, flush sum buffer
    jmp    new_temperature       , and start with a new sum

same_temp
    inc    word ptr cs:time_at_temp , time at this temp
    call   disp_Times

    mov     ax,es:elapsed_time[di] , insure in range of
summation buffer limits

```

```

    cmp     ax,WORD PTR cs:max_sweep_buffer
,   jns     flush_it           , enough data summed, start
new sum
   jae     flush_it           , enough data summed, start new
sum
   mov     ax,cs:temp_bumped   , have we bumped this temp?
   or      ax,ax
   jnz     same_temp_4
   mov     ax,cs:time_at_temp  ; too long at temp?
   cmp     ax,cs:max_sweep    , exceeding max thresholds?
,   js      same_temp_i       ; no, more tests follow
   jb      same_temp_1       , no, more tests follow
same_temp_2:
   mov     cs:temp_bumped,1    ; show we've bumped temp
   call    bump_temp
   jmp     same_temp_4       , continue summation

same_temp_i
   cmp     ax,cs:min_sweep    , have we been here long
enough?
,   js      same_temp_4       , no, we must wait more
   jb      same_temp_4       , no, we must wait more
   test    cs:inhibit_temp_inc,0ffffh ; can we inc temp?
   jz      same_temp_2       ; yes, so do it and continue
same_temp_4
   mov     ax,ds:flags[bx]    , and length match
   cmp     ax,es:flags[di]    ; ?
   jnz     flush_it         , data -> send it

sum_it_up
   mov     ax,cs:new_temp     , save new temp in slot for
next temp
   mov     ds:temp[bx],ax
   inc     word ptr es:elapsed_time[di] , time in this
summation
   mov     cx,ds:flags[bx]    ; set counter
   mov     word ptr ds:flags[bx],0    ; start next data
capture fresh
   sar     cx,1
   sar     cx,1
   mov     si,data_offset    , starting address of data
   add     si,bx

   push   di                 , save the summation buffer
address
   add     di,data_offset

add_i
   mov     ax,ds:[si]        , get lsw
   add     es:[di],ax        , form lower sum
   jnc     add_2
   inc     word ptr es:2[di] , for carry, inc msw

```

```

add_2:
    add     si,4           ; bump pointers
    add     di,4
    loop   add_1         ; loop counter

    pop     di
    cld
    ret

flush_it:
    call   flush        ; output the summation buffer
    jmp    new_temperature_2 ; and start anew

bump_temp:
    push   bx
    mov    bx,cs:Commanded_Temp ; get commaned temp
    add    bx,bx          ; form word offset
    mov    ax,cs:temp_ramp[bx] ; get # pulses for next step
    mov    cx,cs:Temp_inc_pulses ; # pulses this step
    mov    cs:Temp_inc_pulses,ax ; and set for next step
    add    cs:Commanded_Temp,cx ; form new temp
    add    bx,cx          ; and index into the tables
    add    bx,cx
    mov    ax,cs:Min_Sweep_temp[bx] ; get new max and mins
    mov    cs:min_sweep,ax
    mov    ax,cs:Max_Sweep_temp[bx]
    mov    cs:max_sweep,ax
    mov    ax,cs:Max_Sweeps_Sum[bx] ; max sums/buffer
    mov    cs:Max_Sweep_Buffer,ax
    pop    bx
    mov    dx,port_ai    ; control port
    mov    al,ai_temp    ; leave interrupts alone,
                        ; and work with temp

    mov    ah,ai_temp

    or     cx,cx         ; if count = 0, 1st inc = throw
away
    jnz    bump_temp_1
    mov    cs:Time_at_temp,0
    mov    cs:Temp_bumped,0
    jmp    bump_temp_4

bump_temp_1:
    out    dx,al
    xor    al,ah         ; set hi
    out    dx,al
    xor    al,ah
    loop   bump_temp_1

bump_temp_4
    call   disp_Temp
    cld
    ret

```

```

flush
    lds    bx,cs:data_capture_ptr
    les    di,cs:summation_buffer_ptr

    mov    ax,es:elapsed_time[di]      ; update time stamp
    add    ax,es:start_time[di]
    mov    ds:start_time[bx],ax
    mov    word ptr ds:elapsed_time[bx],1 ; start elapsed
time off

    sar    word ptr es:flags[di],1     ; convert byte
offset to count
    sar    word ptr es:flags[di],1

    mov    word ptr cs:summation_buffer_ptr+0,bx ; dc -> sb
    mov    word ptr cs:summation_buffer_ptr+2,ds

    mov    bx,cs:buffer_index         ; get next dc buffer
    add    bx,4
    cmp    bx,(4*No_Buffers)          ; wrap around?
    jne    flush_1
    xor    bx,bx                       ; yes, start again
flush_1
    mov    cs:buffer_index,bx         ; save index for next
time
    lds    bx,cs:buffer_chain[bx]
    mov    word ptr cs:data_capture_ptr+0,bx ; and the dc ->
too
    mov    word ptr cs:data_capture_ptr+2,ds
    mov    word ptr ds:flags[bx],0 ; start data capture fresh

    cmp    di,(6*buffer_length_b)    ; do we need to write?
    jnz    flush_2                    ; (> 0 -> no

flush_3
    push   ax
    push   bx
    push   cx
    push   dx
    push   ds

    mov    cx,di                       ; transfer length in
sectors
    add    cx,buffer_length_b         ; in bytes to
    sar    cx,1                        ; words
    mov    cl,ch                       ; and to sectors
    and    cx,07fh

    mov    dx,cs:starting_sector     ; starting sector
    add    cs:starting_sector,cx     ; update for next
transfer

```

```

mov     bx,es           , segment address
mov     ds,bx

XOR     BX,BX           , SEGMENT OFFSET

call    write_disk

pop     ds
pop     dx
pop     cx
pop     bx
pop     ax

flush_2
clic
ret

flush_to_disk:
call    flush           ; output current
summation
mov     word ptr ds:elapsed_time[bx],0
call    flush           ; second with 0
et
cmp     word ptr cs:summation_buffer_ptr,0 ; did we do a
write?
jnz     flush_3         ; no, do it now
clic
ret

READ_DISK:
mov     ax,cs:Disk_drive , drive #
int     37              ; do disk write

jnc     RD_4
nop                    , no error checking
RD_4:
pop     ax              , remove the garbage
left by dos
RET

WRITE_DISK:
mov     ax,cs:Disk_drive , drive #
int     38              ; do disk write

jnc     WD_4
nop                    , no error checking
WD_4:
pop     ax              , remove the garbage
left by dos
RET

END_SUMMATION.

```

```
START   ENDP
SUMIT   ENDS
```

```
a_stack segment   PARA stack 'STACK'
          dw       50 dup (0)
top_stack dw       0
a_stack ends
```

```
END           SUMMATION
```

(\* Copyright 1986 Michael Rogers and GeoChemical Inc \*)  
(\* all rights reserved \*)  
(\* sC-\*)

MODULE DATAREAD,

FROM SYSTEM

IMPORT BYTE, WORD, ADDRESS, ADR, SIZE, TSIZE, CODE,  
INBYTE, OUTBYTE, INWORD, OUTWORD,

FROM System

IMPORT Terminate, Status,  
ProcessPtr, ProcessDescriptor, curProcess;

FROM MemoryOperations

IMPORT FillChar, Move, Hi, Lo, Swap, And, Not, Or, Xor, Shr,  
MemGet, MemWGet, PortGet, PortWGet, MemSet, MemWSet, PortSet,  
PortWSet, Ptr, Ofc, Seg, Cseg, Dseg, Sseg, MemAvail,

FROM FloatingUtilities IMPORT Frac, Int, Round, Float, Trunc,

FROM ScreenHandler

IMPORT ClrEol, ClrScr, DelLine, InsLine, GotoXY, WhereX,  
WhereY,

CrtInit, CrtExit, LowVideo, NormVideo, HighVideo, SetAt-  
tribute,

GetAttribute, normalAtt, boldAtt, reverseAtt, underlineAtt,  
blinkAtt, boldUnderlineAtt, blinkUnderlineAtt, boldBlinkAtt,  
reverseBlinkAtt, boldUnderlineBlinkAtt,

FROM TFileIO

IMPORT Append, AssignFile, Close, Erase, Flush, Rename,  
Reset, Rewrite, Truncate, Eof,

FROM TRealIO IMPORT ReadReal, WriteReal;

FROM TTextIO

IMPORT ReadInt, ReadCard, ReadChar, ReadString, ReadLn,  
ReadBuffer,

WriteInt, WriteCard, WriteChar, WriteString, WriteBool,  
WriteLn,

Eoln, SeekEof, SeekEoln;

FROM TBinaryIO

IMPORT Read, BlockRead, Write, BlockWrite, BlockRWResult,  
FilePos, FileSize, Seek, LongFileSize, LongFilePosition,  
LongSeek,

FROM TKernelIO

IMPORT File, FileType, OptionMode,  
StatusProc, ReadProc, WriteProc, ErrorProc,  
stdinout, input, output, con, trm, kbd, lst, aux, usr,  
conStPtr, conInPtr, auxInPtr, usrInPtr, conOutPtr, lstOutPtr,  
auxOutPtr, usrOutPtr, errorPtr, IOresult, KeyPressed,

IOBuffer,

IOCheck, DeviceCheck, CtrlC, InputFileBuffer, OutputFileBuf-  
fer;

FROM TDOS IMPORT MsDos, Intr, RegPack;

CONST

maxint = 32767;

Sec1 = 0a000H,

DiskDrive = 2;

```
NMaxSweeps = 30000,  
Solid = 0FFFFH,
```

TYPE

```
ShortString = ARRAY [0..25] OF CHAR;  
OptionsAvailable = (FirstSet , SecondSet , ThirdSet );  
IDType = ARRAY [1..5] OF CARDINAL;  
PointerToWorld = POINTER TO INTEGER;  
string80 = ARRAY [0..80-1] OF CHAR;  
doubleInteger = ARRAY [0..1] OF INTEGER;
```

VAR

```
AMUM,  
AMUI,  
realcounts,  
sum,  
I,  
FirstAmu,  
LastAmu          : REAL;  
throwAway,  
numberOfprofiles,  
MAXTEMP,  
StartingAmu,  
EndingAMU,  
xtemp,  
elapsedTime,  
rsize,  
counter,  
Difference,  
X,  
j,  
i                : INTEGER;  
diskBlock       : CARDINAL;  
comma,  
achr            : CHAR;  
Done,  
done            : BOOLEAN;  
result          : RECORD  
    ax,  
    bx,  
    cx,  
    dx,  
    bp,  
    si,  
    di,  
    ds,  
    es,  
    flags       : INTEGER;  
END;
```

```
DriverParameterBlock : RECORD  
    SummationPtr      : POINTER TO
```

```

doubleInteger;
doubleInteger;
INTEGER;
INTEGER;
INTEGER;
INTEGER;

ThresholdsPtr      : POINTER TO
TempRampPtr        : POINTER TO
TempMinPtr         : POINTER TO
TempMaxPtr         : POINTER TO
MaxSweepsPtr       : POINTER TO

ChannelsPerSweep   : INTEGER;
MaxTemp            : INTEGER;
ThrowAwayNum       : INTEGER;
FirstSectorNum     : CARDINAL;
DiskDriveNum       : INTEGER;
MethodFileName     : string80;
SAMU               : REAL;
EAMU               : REAL;

(* SPARES ARE USED ONLY TO PROVIDE SPACE TO RECEIVE DATA TRANS-
FERRED WHEN A
SINGLE SECTOR IS READ DIRECTLY FROM THE DISK *)
SPARES              : ARRAY [1..(512--
130)] OF BYTE;

END;

```

```

dataRecord          : RECORD
    size            : INTEGER;
    temp            : INTEGER;
    stime           : INTEGER;
    etime           : INTEGER;
    dataA           : ARRAY [0..4095] OF INTEGER;
    spares          : ARRAY [0..512] OF INTEGER;
END;

```

```

firstetc           : ShortString;
outFileName        : string80;
Y                  : ARRAY [0..5] OF ShortString;
dataFile           : File;
halfwidth,
lower,
upper,
Channelofinterest : ARRAY [0..5] OF INTEGER;
realZero,
realSum            : ARRAY [0..2048] OF REAL;
realTemp          : ARRAY [0..256],[0..5] OF INTEGER;
realCounts        : ARRAY [0..256],[0..5] OF REAL;

```

```
PROCEDURE sumToReals;
```

```
VAR
```

```

        i: INTEGER;
BEGIN
    FOR i := 0 TO rsize-1 DO
        realSum[i] := Float(dataRecord.dataA[i*2])+Float(dataRecord-
d dataA[i*2+1])*65536.0+realSum[i],
        END,
    END sumToReals;

PROCEDURE ReadDiskSectors ( VAR StartingSector : CARDINAL;
                            NumberOfSectors : INTEGER;
                            DataRecord : Pointer-
ToWord);
BEGIN
    WITH result DO
        cx := NumberOfSectors;
        dx := StartingSector;
        INC(StartingSector, cx);
        ds := Seg(ADR(DataRecord));
        bx := Ofs(ADR(DataRecord));
        ax := 2; (*read*)
        Intr(04fH, result);
    END;
END ReadDiskSectors;

PROCEDURE ReadDiskHeader;
BEGIN
    diskBlock := Sec1;
    ReadDiskSectors(diskBlock, 1, ADR(DriverParameterBlock));
END ReadDiskHeader;

PROCEDURE readDisk;
BEGIN
    ReadDiskSectors(diskBlock,
                    17, (*2048*4+512 / 512 # sectors*)
                    ADR(dataRecord));
END readDisk;
PROCEDURE FirstEtc;
BEGIN
    Y[1] := 'first';
    Y[2] := 'second';
    Y[3] := 'third';
    Y[4] := 'fourth';
    Y[5] := 'fifth';
    firstetc := Y[i];
END FirstEtc;

PROCEDURE WriteDataToDiskFile;
BEGIN
    FOR i := 1 TO numberofprofiles DO
        FirstEtc;
        WriteString(stdinout, '          Enter file for ',0);
    
```

```

WriteString(stdinout, firstetc,0);
WriteString(stdinout, ' raw temperature profile ', 0);
ReadBuffer(on);
ReadString(stdinout, outFileNames),
ReadLn(stdinout);
ReadBuffer(off);
AssignFile(dataFile, outFileNames, text);
Rewrite(dataFile, 0);
sum := Float(0);
WriteReal(dataFile, realZero[Channelofinterest[i]], 10, 4);
WriteLn(dataFile);
WriteInt(dataFile, (counter-1), 6);
WriteLn(dataFile);
FOR j := 1 TO counter-1 DO
  WriteInt(dataFile, realTemp[j][i], 4);
  WriteReal(dataFile, realCounts[j][i], 8, 0);
  WriteLn(dataFile);
END;
Close(dataFile);
END;
END WriteDataToDiskFile;

PROCEDURE ChannelOfInterest;

BEGIN
  WITH DriverParameterBlock DO
    FirstAmu := DriverParameterBlock.SAMU;
    LastAmu := DriverParameterBlock.EAMU;
    WriteLn(stdinout);
    WriteString(stdinout, ' Starting
Mass = ',0);WriteReal(stdinout,FirstAmu,10,5);
    WriteLn(stdinout);
    WriteString(stdinout, ' Ending
Mass = ',0);WriteReal(stdinout,LastAmu,10,5);
    AMUM := FLOAT(ChannelsPerSweep-1)/(LastAmu-FirstAmu);
    AMUI := 1.0-(FirstAmu*AMUM);
  END;
  WriteLn(stdinout);
  WriteLn(stdinout);
  WriteString(stdinout, ' You may create up to 5 Temp
profiles at once ',0);
  WriteLn(stdinout);
  WriteString(stdinout, '
Your Choice ',0);
  ReadBuffer(on);
  ReadInt(stdinout, numberofprofiles);
  ReadLn(stdinout);
  ReadBuffer(off);
  WriteLn(stdinout);
  WriteString(stdinout, ' You may choose the channel
and halfwidth of each profile',0);
  WriteLn(stdinout);

```

```

FOR i := 1 TO numberofprofiles DO
  FirstEtc,
  WriteString(stdinout, '                Enter ',0);
  WriteString(stdinout,firstetc,0);
  WriteString(stdinout, ' center channel and profile halfwidth
', 0);
  ReadBuffer(on);
  ReadInt(stdinout, Channelofinterest[i]),
  ReadInt(stdinout, halfwidth[i]),
  ReadLn(stdinout);
  ReadBuffer(off);
END;
Determinehalfwidth;
WriteLn(stdinout);
END      ChannelOfInterest;

```

```

PROCEDURE Determinehalfwidth;
BEGIN
  FOR i := 1 TO numberofprofiles DO
    lower[i] := Channelofinterest[i] - halfwidth[i];
    upper[i] := Channelofinterest[i] + halfwidth[i];
  END;
END Determinehalfwidth;

```

```

PROCEDURE ReadDataAndZeroSumIt;
BEGIN
  FOR i = 0 TO 2047 DO
    realSum[i] := 0.0;
    realZero[i] := 0.0;
  END;
  ReadDiskHeader, (* put data in header of file *)
  throwAway := DriverParameterBlock ThrowAwayNum;
  rsize := DriverParameterBlock ChannelsPerSweep;
  MAXTEMP := DriverParameterBlock MaxTemp;
  ClrScr;
  WriteLn(stdinout);
  WriteLn(stdinout);
  WriteString(stdinout, '                Maximum Temperature for
analysis ',0);
  WriteInt(stdinout,MAXTEMP,5);
  WriteLn(stdinout);
  WriteString(stdinout, '                Number of channels per
sweep ',0);
  WriteInt(stdinout, rsize,5);
  WriteLn(stdinout);
  done := FALSE;
  WITH result DO
    WITH dataRecord DO
      elapsedTime = 0;
      REPEAT
        readDisk,

```

```

        INC(elapsedTime, etime),
    UNTIL ((elapsedTime >= throwAway) OR (etime = 0) OR (size
= 0));
    WriteString(stdinout, '                Number of throwaway
sweeps read ', 0),
    WriteInt(stdinout, elapsedTime, 5),
    WriteLn(stdinout);
    elapsedTime := 0;
    REPEAT
        IF size = rsize THEN
            sumToReals;
            INC(elapsedTime, etime);
        END;
        readDisk;
    UNTIL ((etime = 0) OR (size = 0) OR (temp <> 0));
    WriteString(stdinout, '                Number of zerocorrec-
tion sweeps read ', 0),
    WriteInt(stdinout, elapsedTime, 5),
    WriteLn(stdinout);
    FOR i := 0 TO 2047 DO
        realZero[i] := realSum[i]/Float(elapsedTime);
        realSum[i] := 0 0;
    END;
    elapsedTime := 0;
    counter := 1;
    ChannelOfInterest;
    REPEAT
        REPEAT
            IF (elapsedTime=0) THEN
                xtemp := dataRecord.temp;
            END;
            IF size = rsize THEN
                sumToReals;
                INC(elapsedTime, etime);
            END;
            readDisk;
        UNTIL ((temp<>xtemp) OR (etime=0) OR (size=0));
        FOR i := i TO numberofprofiles DO
            realcounts := 0 0;
            FOR j := lower[i] TO upper[i] DO
                realcounts := realcounts+(realSum[j]/Float(elapsed-
Time));
            END;
            realTemp[counter][i] = xtemp;
            realCounts[counter][i] := realcounts;
            FOR j := lower[i] TO upper[i] DO
                realSum[j] := 0 0;
            END;
        END;
        counter := counter+1;
        elapsedTime := 0;
        IF (xtemp)=MAXTEMP THEN

```

```

        done = TRUE
    END,
    UNTIL done,
    WriteString(stdinout, '          Finished reading data
', 0),
    WriteLn(stdinout);
    WriteDataToDiskFile,
    END,
END;
END ReadDataAndZeroSumIt;

BEGIN (*mainline*)
    AMUI = 0.0;
    AMUM = 1.0;

    REPEAT
        ClrScr,
        WriteLn(stdinout);
        WriteLn(stdinout);
        WriteString(stdinout, '          GSI DMA
SOFTWARE', 0), WriteLn(stdinout);
        WriteString(stdinout, '          Copyright 1986 Geochem-
ical Services, Inc', 0), WriteLn(stdinout);
        WriteString(stdinout, '          Program
Dataread', 0),
        WriteLn(stdinout);
        WriteLn(stdinout);
        WriteLn(stdinout);
        WriteLn(stdinout);
        WriteString(stdinout, '          R Read data from hard
disk and write a temp profile', 0),
        WriteLn(stdinout);
        WriteLn(stdinout);
        WriteLn(stdinout);
        WriteString(stdinout, '          Q Quit and return to
DOS', 0), WriteLn(stdinout);
        WriteLn(stdinout);
        WriteLn(stdinout);
        WriteLn(stdinout);
        WriteString(stdinout, '          Enter your
selection [R or Q] ', 0),
        ReadBuffer(on);
        ReadChar(stdinout, achr);
        ReadLn(stdinout);
        ReadBuffer(off);
        WriteLn(stdinout);
        WriteLn(stdinout);
        CASE achr OF
            'R' : ReadDataAndZeroSumIt;
            'Q' :
            'X' : Terminate(normal);
        ELSE

```

```
        END, (*case*)
        WriteLn(stdinout),
UNTIL FALSE,
END DATAREAD
```

```
( Copyright 1986 James R. Hanratty and GeoChemical Inc.)
( all rights reserved )
```

```
( $C-)
```

```
PROGRAM MarkGood,
```

```
VAR
  done           : boolean;
  zchr           : char;
  i,
  j,
  disk_block     : integer;
  result         : record
                ax,
                bx,
                cx,
                dx,
                bp,
                si,
                di,
                ds,
                es,
                flags : integer,
                end,

  data_record    : record
                data  array [0 : 255] of integer,
                end,
```

```
procedure read_disk;
begin
with result do
begin
  cx = 1,
  dx = disk_block,
  ds = seg(data_record),
  bx = ofs(data_record),
  ax = 2, (read)
  intr ( $4f, result),
  (writeLn('read ', disk_Block),)
end,
end,
```

```

procedure write_disk;
begin
with result do
begin
cx := 1;
dx := disk_block;
ds := seg(data_record);
bx := ofs(data_record);
ax := 3; {write}
end;

(writeLn('write disk block = ',disk_Block));
intr ( $4f, result);
end;

BEGIN (mainline)
repeat
writeLn,
writeLn,
writeLn('Mark good those hard disk bad blocks from markbad'),
writeLn,
writeLn,
writeLn('1 Examine FAT for bad blocks in data area');
writeLn,
writeLn('2 Mark last 1/3 of hard disk good');
writeLn,
writeLn('3 Examine manually a sector');
writeLn,
writeLn('4 Exit');
WriteLn,
write ( '                                Enter your
selection = ');

achr := ' ';
readLn(achr);
case achr of
'1' begin
j := 0;
for Disk_block := 40 to 64 do
begin
Read_Disk;
for i := 0 to 255 do
if j <= 10
then
begin
if data_record.data[i] = 0
then (all is ok)
else
begin
writeLn('Possible bad disk area or data in

```

```

reserved area said ',
                Disk_Block:=6,
                i:=6,
                data_Record:=data[i]:6);
j := j + 1,
if j > 10 ( too many errors?)
then
begin
write('Continue display? Y/N = ');
readln(achr);
if achr in ['y','Y'] then j := 0;
end;
end;
end;
end;
'2' begin
for Disk_block := 40 to 64 do
begin
Read_Disk;
for i := 0 to 255 do data_record:=data[i]:60000;
Write_disk;
end;
end;
'3' begin
write('Sector to read = ');
readln(Disk_block);
Read_disk;
for i := 0 to 255 do
begin
write (data_record:=data[i]:4);
if (i mod 16) = 15 then writeln;
end;
end;
else ;
end; (case)
until Achr = '4',
END.

```

Copyright 1986 James R. Hanratty and GeoChemical Inc )  
( all rights reserved )

{ \$C- }

PROGRAM MarkBad;

```
VAR
  done          boolean;
  achr          char;
  i,
  j,
  disk_block   : integer;
  result       record
    ax,
    bx,
    cx,
    dx,
    bp,
    si,
    di,
    ds,
    es,
    flags      integer;
  end;

  data_record  record
    data       array [0 .. 255] of integer;
  end;
```

```
procedure read_disk,
begin
with result do
begin
  cx = 1,
  dx = disk_block,
  ds = seg(data_record);
  bx = ofs(data_record),
  ax = 2; (read)
  intr ( $4f, result),
  (writeln('read ', disk_Block));)
end,
end,
```

```
procedure write_disk,
begin
with result do
begin
  cx = 1,
  dx = disk_block,
```

```

    ds := seg(data_record);
    bx := ofs(data_record);
    ax := 3, (write)
end;

(writeLn('write disk block = ', disk_Block));
intr ( $4f, result),
end,

BEGIN (mainline)
repeat
writeLn;
writeLn;
writeLn('Mark hard disk bad blocks');
writeLn;
writeLn;
writeLn('1 Examine FAT for bad blocks in data area');
writeLn;
writeLn('2 Mark last 1/3 of hard disk bad');
writeLn;
writeLn('3 Examine manually a sector');
writeLn;
writeLn('4 Exit');
WriteLn;
write ( '                               Enter your
selection"= '),

achr := ' ';
readLn(achr),
case achr of
'1' begin
j := 0;
for Disk_block := 40 to 64 do
begin
Read_Disk,
for i := 0 to 255 do
if j <= 10
then
begin
if data_record data[i] = 0
then (all is ok)
else
begin
writeLn('Possible bad disk area or data in
reserved area s: i d ',
Disk_Block:6,
i:6,
data_Record data[i]:6);
j = j + 1;
if j > 10 ( too many errors?)
then
begin

```

```

"
        write('Continue display? Y/N = '),
        readln(achr),
        if achr in ['y','Y'] then j := 0;
        end,
        end,
        end,
        end;
    end;
end;

'2' begin
for Disk_block := 40 to 64 do
begin
Read_Disk;
for i := 0 to 255 do data_record.data[i] := $fff7;
Write_disk;
end;
end;

'3' begin
write('Sector to read = '),
readln(Disk_block),
Read_disk;
for i := 0 to 255 do
begin
write (data_record.data[i].4),
if (i mod 16) = 15 then writeln,
end,
end,
end,

else ,

end; (case)
until Achr = '4';
END.

```



*MISSION*  
*of*  
*Rome Air Development Center*

*RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control, Communications and Intelligence (C<sup>3</sup>I) activities. Technical and engineering support within areas of competence is provided to ESD Program Offices (POs) and other ESD elements to perform effective acquisition of C<sup>3</sup>I systems. The areas of technical competence include communications, command and control, battle management, information processing, surveillance sensors, intelligence data collection and handling, solid state sciences, electromagnetics, and propagation, and electronic, maintainability, and compatibility.*