JUN 02 1997 Sta. 37 (20)

ENGINEERING DATA TRANSMITTAL

1. EDT 619509

2. To: (Receiving Organization) DISTRIBUTION	3. From: (Originating Organization) SESC - Remote Sensing and Sampling Equipment Engineering	4. Related EDT No.: n/a
5. Proj./Prog./Dept./Div.: Characterization	6. Design Authority/ Design Agent/Cog. Engr.: M.F. Erhart	7. Purchase Order No.: n/a
8. Originator Remarks: Approval/Release		9. Equip./Component No.: n/a
		10. System/Bldg./Facility: 200G
11. Receiver Remarks: 11A. Design Baseline Document? [ ] Yes [X] No		12. Major Assm. Dwg. No.: n/a
		13. Permit/Permit Application No.: n/a
		14. Required Response Date: 5/9/97

15. DATA TRANSMITTED

(A) Item No.	(B) Document/Drawing No.	(C) Sheet No.	(D) Rev. No.	(E) Title or Description of Data Transmitted	(F) Approval Designator	(G) Reason for Transmittal	(H) Originator Disposition	(I) Receiver Disposition
1	HNF-SD-WM-CSWD-077		O	Computer Systems and Software Description for Gas Characterization System	SQ	1	1	

16. KEY

Approval Designator (F)	Reason for Transmittal (G)		Disposition (H) & (I)	
E, S, Q, D or N/A (see WHC-CM-3-5, Sec.12.7)	1. Approval 2. Release 3. Information	4. Review 5. Post-Review 6. Dist. (Receipt Acknow. Required)	1. Approved 2. Approved w/comment 3. Disapproved w/comment	4. Reviewed no/comment 5. Reviewed w/comment 6. Receipt acknowledged

17. SIGNATURE/DISTRIBUTION
(See Approval Designator for required signatures)

(G) Reason	(H) Disp.	(J) Name (K) Signature (L) Date (M) MSIN	(G) Reason	(H) Disp.	(J) Name (K) Signature (L) Date (M) MSIN
		Design Authority LE-37	1	1	Other J.D. Johnson 57-14 BE Fang for GDJ 5/6/97
1	1	Design Agent D.D.Tate D Tate 5/5/97	4	6	T.C. Schneider LE-37 T C Schneider 5/6/97
1	1	Cog.Eng. M.F. Erhart M F Erhart 5/8/97	R1-	51	
4	4	Cog. Mgr. R.L. Schlosser R L Schlosser 5/29/97	R1-	56	
1	1	QA L.D. Salsbury L D Salsbury 5/8/97	B7-	41	
1	1	Safety S.U. Zaman S U Zaman 5/8/97	R3-	01	
		Env.			

18.	19.	20.	21. DOE APPROVAL (if required)
P L Phillips 5/5/97 Signature of EDT Originator Date	23 May 97 Authorized Representative for Receiving Organization Date	R L Schlosser 5/29/97 Design Authority/ Cognizant Manager Date	Ctrl. No. [ ] Approved [ ] Approved w/comments [ ] Disapproved w/comments

COMPUTER SYSTEMS AND SOFTWARE DESCRIPTION FOR GAS CHARACTERIZATION SYSTEM

C.V. Vo
SGN Eurisys Serivices Corporation, Richland, WA 99352
U.S. Department of Energy Contract DE-AC06-96RL13200

EDT/ECN: No. 619509 UC: 2030
Org Code: S1200 Charge Code: N2022
B&R Code: EW3120072 Total Pages: 198

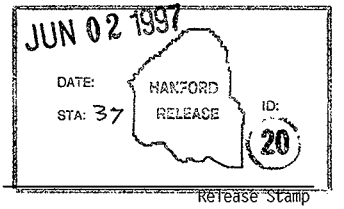
Key Words: Gas Characterization System (GCS)

Abstract:

Computer system design layout description for the Gas Characterization System.

TRADEMARK DISCLAIMER. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof or its contractors or subcontractors.

Printed in the United States of America. To obtain copies of this document, contact: WHC/BCS Document Control Services, P.O. Box 1970, Mailstop H6-08, Richland WA 99352, Phone (509) 372-2420; Fax (509) 376-4989.



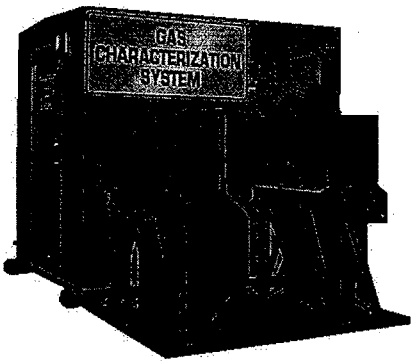
Janis Bishop 6-2-97
Release Approval Date

Release Stamp

Approved for Public Release

COMPUTER SYSTEMS AND SOFTWARE DESCRIPTION
FOR
GAS CHARACTERIZATION SYSTEM

(HNF-SD-WM-CSWD-077)



Prepared by

S. S. Kanjilal
E. K. Straalsund
D. D. Tate
B. L. Philipp
&
C. V. Vo

April 1997

SGN Eurisys Services Corporation
P.O. Box 840
Richland, Washington 99352

CONTENTS

1.0 INTRODUCTION 6
1.1 Background 6
1.2 Site Environment 6

2.0 SCOPE 7

3.0 SOFTWARE REQUIREMENTS SPECIFICATION 7
3.1 Criteria 7
3.2 Software Functions 9
3.2.1 LabView Data Acquisition Control System 9
3.2.2 Gas Chromatograph 1 and 2 Data Systems 10
3.2.3 FTIR Data System 11

4.0 DESIGN DESCRIPTION & SYSTEM INTERFACES 12
4.1 Principles of Operation 12
4.2 Computers & Software 12
4.2.1 GC1 Computer 13
4.2.2 GC2 Computer 14
4.2.3 FTIR Computer 15
4.2.4 The GCS1HOST Computer 16
4.3 Functions and Flow of the GCS Software 21
4.3.1 Gas Chromatograph 21
4.3.2 FTIR Software 23
4.4 LABHOST Computer 28
4.5 Vendor Information File 29

5.0 OPERATION 30
5.1 Overview 30
5.2 Operating Procedures 30
5.2.1 System Startup, Normal Operation, Shut Down 30
5.3 Discrepancy Reporting & Corrective Actions 30

6.0 CONFIGURATION CONTROL 31
6.1 Configuration Status Accounting 31
6.2 Media Control 31

7.0 SAFETY, RELIABILITY, & QUALITY ASSURANCE 32
7.1 Safety Classification and Signature Designation 32
7.2 Quality Assurance and Quality Control 32

8.0 REFERENCES 33

9.0 BIBLIOGRAPHY 34

LIST OF APPENDICES

LIST OF APPENDICES 35

A -- DEFINITIONS & ACRONYMS 36

B -- OPERATING PROCEDURES 38
1.0 INTRODUCTION 39
2.0 SCOPE 39
3.0 PREREQUISITE 39
4.0 DESCRIPTION 39
4.1 GC1 Hardware/Software 40
4.2 GC2 Hardware/Software 40
4.3 FTIR Hardware/Software 41
4.4 Process Instrumentation Data Logging Hardware/Software 41
5.0 PROCEDURE 42
5.1 Initial Setup 42
5.2 Starting the FTIR 43
5.3 Starting the GC1 43
5.4 Starting the GC2 44
5.5 Starting LabView Data Acquisition System 45
5.6 Verify all systems are operating and communicating with GCS1HOST 46
6.0 Troubleshooting 47
6.1 Troubleshooting Network Communications 47
6.2 Troubleshooting FTIR Operation 47
6.3 Troubleshooting MTI GC Operation. 47
6.4 Troubleshooting Labview Data Acquisition Problems 48

C -- C^{++} SOURCE CODE LISTINGS for GC1 ZIP Routines 49
C.1 GC1_ZIP.INI 49
C.2 GC1ZIP.CPP 50
C.3 GC_FIL.CPP 56
C.4 EXP2.CPP 57
C.5 INDEX3.CPP 59
C.6 GC1ZIP.H 59
C.7 FUNCTION.H 59

D -- C^{++} SOURCE CODE LISTINGS for GC1APP32 60
D.1 GC1.INI 60
D.2 GC1APP32.DEF 60
D.3 GC1APP32.CPP 61
D.4 GC1_INDX.CPP 75
D.5 GC1_TRNS.CPP 76
D.6 APPND.CPP 77
D.7 DIF_CHK.CPP 79
D.8 DIF_FIX.CPP 80
D.9 GC_FIL.CPP 81
D.10 INDEX3.CPP 83
D.11 RPT_TRNS.CPP 84

D.12 TIME_STP.CPP 86
D.13 GC1APP32.H 87
D.14 HEADER1.H 87
D.15 RESOURCE.H 87
D.16 GC1APP32.RC 87

E -- C++ SOURCE CODE LISTINGS for GC2 ZIP Routines 88
E.1 GC2_ZIP.INI 88
E.2 GC2ZIP.CPP 88
E.3 GC_FIL.CPP 95
E.4 EXP2.CPP 96
E.5 INDEX3.CPP 98
E.6 GC2ZIP.H 98
E.7 FUNCTION.H 98

F -- C++ SOURCE CODE LISTINGS for GC2APP32 99
F.1 GC2.INI 99
F.2 GC2APP32.DEF 99
F.3 GC2APP32.CPP 100
F.4 APPND.CPP 113
F.5 DIF_CHK.CPP 115
F.6 DIF_FIX.CPP 116
F.7 GC_FIL.CPP 116
F.8 GC1_INDX.CPP 118
F.9 GC1_TRNS.CPP 119
F.10 INDEX3.CPP 121
F.11 RPT_TRNS.CPP 122
F.12 TIME_STP.CPP 123
F.13 GC2APP32.H 124
F.14 HEADER.H 125
F.15 RESOURCE.H 125
F.16 GC2APP32.RC 125

G -- C++ SOURCE CODE LISTINGS for FTIR Routines 126
G.1 FTIRZIP.INI 126
G.2 FTIRZIP.CPP 126
G.3 FILENAME.CPP 129

H -- ARRAY BASIC PROGRAM LISTINGS 134
H.1 ICOMPUT.AB 134
H.2 ICOMPUT.AB 144
H.3 SPFTIRII.AB 178

I -- LABVIEW SCREEN AND BLOCK DIAGRAM LISTINGS 181
I.1 VI STRUCTURE 187

LIST OF FIGURES

Figure 4A -- GCS Computer System Layout Diagram 12
Figure 4B -- GCS1HOST I/O 16
Figure 4C -- Gas Characterization System Physical Configuration 19
Figure 4D -- High Level GCS Software/Data Flow 20
Figure 4E -- Interface Program "GCAPP32" Software Flow 25
Figure 4F -- Interface Program "GC_ZIP" Software Flow 26
Figure 4G -- Interface Program "NOTIFY" Software Flow 27

COMPUTER SYSTEMS AND SOFTWARE DESCRIPTION FOR GAS CHARACTERIZATION SYSTEM

1.0 INTRODUCTION

The Gas Characterization System Project was commissioned by TWRS management with funding from TWRS Safety, on December 1, 1994. The project objective is to establish an instrumentation system to measure flammable gas concentrations in the vapor space of selected watch list tanks, starting with tank AN-105 and AW-101. Data collected by this system is meant to support first tank characterization, then tank safety. System design is premised upon Characterization rather than mitigation, therefore redundancy is not required.

1.1 Background

Nuclear waste by-products from Hanford Site weapon production missions are stored in a number of large single and double shell underground tanks. In March of 1990, 23 waste tanks at the Hanford Site were identified as having a potential for build up of several flammable gases including hydrogen, nitrous oxide, and ammonia. Waste tank 241-SY-101 was identified as presenting the most serious safety concern, which prompted the installation of the Gas Monitoring System (GMS-2) with continuous hydrogen measuring. Additionally, for personnel safety and environmental monitoring, the amount of ammonia released from the SY tank farm exhaust stack is monitored. A list of tanks with the potential for approaching the lower flammability limit (LFL) was created, and called the flammable gas watch-list tanks. The Gas Characterization System uses the operationally proven GMS-2 as a design baseline.

The Gas Characterization System (GCS) Computer System consists of several individual computer components necessary to address the control and data acquisition functions of the several specific types of instruments used to monitor the vapor space. These computers are networked, using the Microsoft Windows NT multi-tasking operating system and networking software. This allows the systems components to function in parallel and perform instrument control, data collection, analysis, computer interfacing, and archiving functions.

1.2 Site Environment

The environment to which the Gas Characterization System GCS computer systems will be exposed is classified as an outdoor environment. Environmentally sensitive monitoring components shall be protected by appropriate enclosures. Although the equipment will be protected by enclosed shelters, access to the shelters will expose the equipment to the outdoor environment, with ambient temperatures ranging from -29°C to 49°C (-20°F to 120°F), relative humidity ranging from 5% to 100%, and wind speeds ranging up to 130 kilometers per hour (80 mph) with blowing dust. The background Radiation dose rate in "AW" and "AN" farm is less than 0.5 mR/hr.

2.0 SCOPE

This document provides an overview of the computer systems software that provides analytical calculations, data logging, system control, automated calibration verifications, data transfer, and data archival for the Gas Characterization Systems being installed at the 200 Areas Tank Farms. It outlines the system layout, and contains descriptions of components and the functions they perform. The following system documentation categories are included in the scope of this document:

- Software Requirements Specification
- Computer Software Design Description
- Software Configuration Management Plan

3.0 SOFTWARE REQUIREMENTS SPECIFICATION

The Gas Characterization System requires computers to control and sequence many instruments, sensors, and electric valves. Additionally, the computers must perform mathematical functions (ie. integrate the area under the curve) to produce "Calculated Concentrations". Control and sequencing of the valves and instruments, with these microcomputers, requires not only vendor supplied software, but also configuration, automation, and integration programming. This section will address the requirements, functions, inputs, outputs, format and desirables associated with the GCS product.

3.1 Criteria

At the beginning of the Gas Characterization System project, the following criteria was established as necessary in relation to the GCS software:

3.1.1 The following vapor space gasses shall be monitored at the listed resolutions:

- **H_2** @ 3 to 30 ppm ($\pm$3 ppm)
  @ 30 to 30,000 ppm ($\pm$10% of reading)
- **NH_3** @ 3 to 30 ppm ($\pm$10 ppm)
  @ 30 to 10,000 ppm ($\pm$10% of reading)
- **N_2O** @ 10 to 30 ppm ($\pm$3 ppm)
  @ 30 to 30,000 ppm ($\pm$10% of reading)
- **CH_4** @ 10 to 30 ppm ($\pm$10 ppm)
  @ 100 to 30,000 ppm ($\pm$10% of reading)

3.1.2 The system shall be automated to provide continuous sampling and calculated data.

3.1.3 The system CPU shall be network connected for:

- data retrieval, without entering a radiation zone
- remote system monitoring
- remote diagnostics
- as close to real time availability of data as possible

3.1.4 The system shall be reconfigurable to read other gasses.

3.1.5 The system shall selectively preserve spectral data[Φ] for future review[Φ]

- save all calibration data
- save faulty data sets (0 data)

[Φ] *Notes:*

Selectively Preserve Spectral Data -- is referring to the archival retention of data denoting significant data, changes, calibrations, and errors. Retention of routine data which may remain the same for days at a time, is unnecessary.

Future Review - The need to retrieve the spectral data associated with significant calculated data is to be considered on the order of years.

3.1.6 The Gas Chromatograph shall sample at a variable minute rate

3.1.7 The FTIR sample rate shall, at 64 scans, be 10 minutes or better

3.1.8 The computer operating systems shall be compatible with the site standard and shall provide for preemptive multi-tasking. Windows[1] NT 3.5 recommended. The software shall be Windows 3.1 or Win NT compatible.

3.1.9 System software shall control calibrations and flag them in the data set

3.1.10 The output calculated concentrations and associated spectral data shall be given unique file and record names based upon date and time.

3.1.11 Process data (such as pressure, flow, & temperature) shall be logged at one minute intervals. Channels shall be denoted in the saved data, and the records shall be identified by date and time. Each process data file shall be uniquely named using a date stamp.

3.1.12 Each computer controlling a directly connected instrument shall save necessary data for a minimum of 24 hours, in the event of a failure of the network link.

3.1.13 The GCS host computer (local to the instruments) shall preserve unarchived data for a minimum of one week, in the event of a failure of the central archival system (LABHOST) or related telecommunications.

---

[1] Windows is a trademark of Microsoft Corporation.

3.1.14 No data shall be deleted from the local system without verifying the data is resident on the receiving computer.

3.2 Software Functions

The GCS computer System consists of several individual computer components necessary to provide control and data acquisition functions for the GCS analytical instruments. Two gas chromatograph instruments (GC1 computer and GC2 computer) are used to monitor Hydrogen, Nitrous Oxide, and Methane in the tank vapor space. The FTIR instrument (FTIR computer) monitors Ammonia content in the same gas stream.

The GC1, GC2 and FTIR computers are networked to the host computer GCS1HOST using the Microsoft Windows NT multi-tasking operating system and networking software. (Note: Currently, for the two systems being installed, the Host computer for the first system will be designated GCS1HOST and the Host computer for the second system will be designated GCS2HOST. This document will refer to the Host computer as GCS1HOST). This allows the system components to function in parallel to control the instruments, sensors, and electric valves. Additionally, the computers must perform mathematical functions (i.e. integrate the area under the curve) to produce "Calculated Concentrations". Controlling the instruments with these microcomputers, requires not only vendor supplied software, but also configuration, automation, and integration programming. Integration software consists of several systems. LabView VI system provides process data acquisition and control of the automatic calibration feature for the gas chromatograph instruments. GC1 and GC2 software systems provide an automated process for data configuration and handling. A modified vendor supplied software version (Win-IR) gives the FTIR the capabilities of continuous sampling and data configuration.

3.2.1 LabView Data Acquisition Control System

3.2.1.1 Operator Interface Control and Indicator Function

- Provide real time display on front panel for process data such as pressure, flow, and temperature.

- Accept control inputs from front panel for data file recording and gas chromatographs calibration. Details to these control inputs will be described in section 3.2.2.2 and 3.2.2.4

3.2.1.2 Data File Recording and Archiving Function

- Data files will be logged based on the file configuration control inputs as follows:

  - Requires complete archive path.
  - Data frequency.
  - Toggle button to Save or Not Save to file.

- The process data file will be automatically generated once at the beginning of each day and uniquely named using a date stamp.

- At the beginning of each day, the data file of the previous day will be archived across the HLAN per the predefined archive path.

3.2.1.3 HLAN network failure mode

- Data will be preserved by saving locally.

- Provide adequate data storage for a minimum of one week.

3.2.1.4 GCs Calibration Control Function

- Accept calibration control inputs on front panel as follows:

  - Time start
  - Duration

- Provide signals to control auto calibration valves.

- Mark data file to identify calibration sample.

3.2.2 Gas Chromatograph 1 and 2 Data Systems

The GC1 and GC2 data systems are design to have the same functionality except that GC1 measures hydrogen gas and GC2 measures methane and nitrous oxide gases.

3.2.2.1 EZChrom 200 Data System Function

- Controls and acquires data from MTI Gas Chromatograph. Software is vendor supplied.

- User defined method programs are used to analyze chromatogram data and calculate component concentrations.

3.2.2.2 Organize and Format EZChrom Data

- Custom software (GC1APP32.EXE and GC2APP32.EXE) collects and organizes EZChrom data.

  - Organized by date
  - Organized by run or index number

3.2.2.3 Organize Data Following a System Outage or Date Change

- Software automatically indexes filenames following a system outage or upon date change.

3.2.2.4 Interface with Labview VI

- Data from GCs is passed to and displayed in Labview[2]

- Calibration data is flagged based upon external valve operation initiated from Labview.

3.2.2.5 Data Compression and HLAN Transfer

- At a pre-defined time each day, all GC data will be compressed and sent to archive over HLAN connection.

3.2.2.6 Data Handling During Network Outage

- System is required to store archive data locally in the event of a network failure or interruption.

3.2.3 FTIR Data System

3.2.3.1 Win-IR Data System Function

- Controls and acquires data from BIO-RAD Infra-Red Spectrometer. Software is vendor supplied.

- User defined method programs are used to analyze interferogram data and calculate component concentrations.

3.2.3.2 Organize and Format Win-IR Data

- Custom software with modified vendor supplied software continuous collects and organizes data.

  - Organized by date
  - Organized by index number in sequential order

3.2.3.3 Organize Data Following a System Outage or Date Change

- Software automatically indexes filenames following a system outage or upon date change.

3.2.3.4 Data Compression and HLAN Transfer

- At a pre-defined time each day, all FTIR data will be compressed and sent to archive over HLAN connection.

3.2.3.5 Data Handling During Network Outage

- System is required to store archive data locally in the event of a network failure or interruption.

---

[2] LabView is a trademark of National Instruments Corporation.

4.0 DESIGN DESCRIPTION & SYSTEM INTERFACES

4.1 Principles of Operation

The general organization of the GCS computer system components are shown in the GCS Computer System Layout Diagram (Figure 4A). An individual processor is dedicated to each gas measuring instrument (Gas Chromatograph and FTIR), as recommended by the equipment vendors. Each of these dedicated CPUs runs the software supplied with the instrument. Another CPU is responsible for collecting the data from the three GC and FTIR processors as well as reading the analog sensors (flow, pressure, temperature). This fourth CPU reduces and compresses the data, then passes the calculated results with source data via HLAN to a central computer that shares the data real time to the networked data users and archives the data to CD-ROM.

4.2 Computers & Software

This subsection defines the hardware and software composition of each of the four computers in each GCS field unit (GC1, GC2, FTIR, and GCSHOST computers).

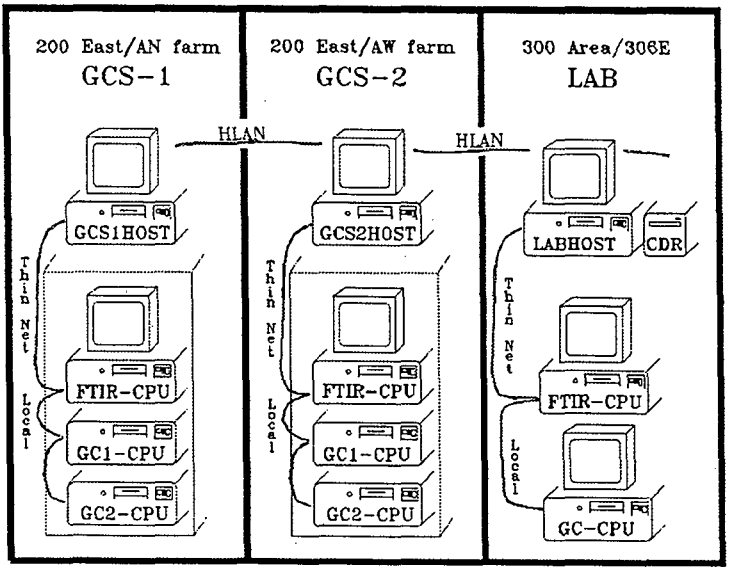


Figure 4A -- GCS Computer System Layout Diagram

4.2.1 GC1 Computer

The GC1 Computer supports an MTI dual column Gas Chromatograph with a control and computational program which records accurate measurements of trace Hydrogen (H_2 at 3-30ppm-low & 30-30,000ppm-high).

4.2.1.1 GC1 Computer Hardware

Model: AST[3] Premmia 486/66
Memory: 16 megabytes of RAM
Disks: A: - 3.5" HD floppy
C: - 340 megabyte hard drive
Ethernet: 3COM Etherlink III card on thin-net (BNC connector)
(initial setup: P300, IRQ-5)
Ports: connected to MTI dual column GC via RS-232 cable on COM1

4.2.1.2 GC1 Computer Software

Windows NT or Windows 95 (EZChrom 200 must run under one of these 32 bit operating system environments to allow the GCAPP call, from GCSHOST, to check for the existence of the desired file)
NETBEUI network protocol
EZChrom 200 Data System software, version 4 (supplied with GC)

4.2.1.3 GC1 Computer Hard Disk Organization (essential first level directories)

Other directories may be present as needed to support a support function such as a video card (ie. If the video card fails, another available card will be used with its own support files directory).

```
C:\   (340 Meg. hard drive with one partition)
  ├──CAL
  ├──DOS
  ├──GC1
  ├──MTI
  ├──TEMP
  ├──USERS
  ├──WINNT35
  └──WIN32APP
```

4.2.1.4 GC1 Network Shares and Connections

Shares from GC1 include:

Share Name	Path	
C	C:\	(no password)
GC1	C:\GC1	(no password)

Connections from GC1 are a necessary part of configuration and will be necessary for testing, diagnosing problems, or reloading initial configuration from the CD ROM drive on GCSHOST.

[3] AST is a trademark of AST Research, Inc.

4.2.2 GC2 Computer

The GC2 Computer supports an MTI dual column Gas Chromatograph which records accurate measurements of Nitrous Oxide (N_2O at 10-30ppm-low & 30-30,000ppm-high), and Methane (CH_4 at 10-100ppm-low & 100-30,000ppm-high).

4.2.2.1 GC2 Computer Hardware

Model: AST Premmia 486/66
Memory: 16 megabytes of RAM
Disks: A: - 3.5" HD floppy
C: - 340 megabyte hard drive
Ethernet: 3COM Etherlink III card on thin-net (BNC connector)
(initial setup: P300, IRQ-5)
Ports: connected to MTI dual column GC via RS-232 cable on COM1

4.2.2.2 GC2 Computer Software

Windows NT or Windows 95 (EZChrom 200 must run under one of these two 32 bit operating system environments to allow the GCAPP call, from GCSHOST, to check for the existence of the desired file)
NETBEUI network protocol
EZChrom 200 Data System software, version 4 (supplied with GC)

4.2.2.3 GC2 Computer Hard Disk Organization (essential first level directories)

```
C:\   (340 Meg. hard drive with one partition)
  ├──CAL
  ├──DOS
  ├──GC2
  ├──MTI
  ├──TEMP
  ├──USERS
  ├──WINNT35
  └──WIN32APP
```

4.2.2.4 GC2 Network Shares and Connections

Shares from GC2 include:

Share Name	Path	
C	C:\	(no password)
GC2	C:\GC2	(no password)

Connections from GC2 are a necessary part of configuration and will be necessary for testing, diagnosing problems, or reloading initial configuration from the CD ROM drive on GCSHOST.

4.2.3 FTIR Computer

The FTIR Computer supports a Bio-Rad FTS-175 Fourier Transform -Mid-InfraRed Spectrometer, with Michelson mechanical bearing interferometer, which records accurate measurements of Ammonia (NH_3 at 10-30ppm-low & 30-10,000ppm-high) and supplies spectral data suitable for future calculation of other gasses from the retained interferogram.

4.2.3.1 FTIR Computer Hardware

Model:	AST Premmia 486/66
Memory:	16 megabytes of RAM
Disks:	A: - 3.5" HD floppy C: - 340 megabyte hard drive
Ethernet:	3COM Etherlink III card on thin-net (BNC connector) (initial setup: P300, IRQ-5)
Video:	VGA
Other Cards:	Bio-Rad EISA bus card, connecting FTS-175 FTIR Spectrometer via a ribbon cable

4.2.3.2 FTIR Computer Software

Windows for Workgroups version 3.11
NETBEUI network protocol
Win-IR software (version 3.0 or later) (supplied with FTIR)
[note: Grams386 is similar software and may be used to review and interpret the chromatograms created by this system.]
Configuration File
Background
Training Spectra

Three configuration files are necessary for GCS mode. These three were project developed or modified, using Quant Basic (the development tool supplied with the WIN-IR software.

- ♦♦ ICOMPUT.AB - Converts Interferograms to Absorbance Spectrum
- ♦♦ JUANTBAS.AB - Uses Absorbance to calculate concentrations
- ♦♦ SPFTIRII.AB - Configures system for continuous operation and data storage format

4.2.3.3 FTIR Computer Hard Disk Organization (essential first level directories)

```
C:\   (340 Meg. hard drive with one partition)
 ├──CAL
 ├──DOS
 ├──FTIR
 ├──WIN_IR
 ├──MOUSE
 └──WINDOWS
```

4.2.1.4 FTIR Network Shares and Connections

Shares from FTIR include:

Share Name	Path	
C	C:\	(no password)
FTIR	C:\FTIR	(no password)

4.2.1.5 Connections from FTIR include:

Connection Name	Drive Letter Used
GCS1HOST\FTIR	D:

4.2.4 The GCS1HOST Computer

The GCSHOST Computer is the focal point for the instruments within the GCS enclosure, performing 3 main functions. (note: The use of GCS1HOST in this section is representative of the other field GCS units, GCS2HOST or GCS3HOST)

4.2.4.1 It handles the discrete sensors and valves associated with GCS via the National Instruments I/O cards internal to it. Continuous temperature, pressure, and flow measurements are made, displayed, and logged. Additionally, periodic calibration verifications are automatically called for, which requires closing the sample line valve and opening the calibration gas valve.

4.2.4.2 GCS1HOST communicates with both the local instrument network and with HLAN, as isolated networks, using two network interface cards. The traffic from either network can not reach the other. A GCS process can communicate to either but not both. GCS1HOST communicates with the GC1 computer and instrument, the GC2 computer and instrument, and the FTIR computer and instrument via ethernet on thin-net cable. This network is local to the GCS enclosure only.

4.2.4.3 GCS1HOST retrieves the spectral data files from the instrument computers and performs labeling, compression, and other functions and stores the data on LABHOST for network access by authorized data users and long term archiving.

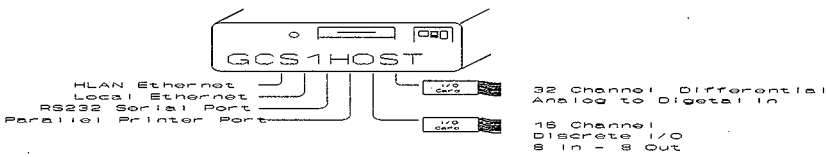


Figure 4B -- GCS1HOST I/O

4.2.4.4 GCS1HOST Computer Hardware

Model:	AST Premmia GX P-100, EISA/ISA/PCI bus compatible
Processor:	Pentium[4] at 100 MHz
Memory:	32 megabytes of RAM
Disks:	A: - 3.5" HD floppy
	1.2 Gig.HD partitioned to C: 340 megabyte D: 680 megabyte
	E: - CD ROM drive (internal)
Ethernet:	AMD on the motherboard for HLAN access, AUI connector (initial setup: P320, IRQ-5, DMA-3)
	3COM Etherlink III network card on thin-net (BNC connector) (initial setup: P300, IRQ-10) (not bridged or forwarded to HLAN)
Ports:	Parallel port on P378h, IRQ-7
	Serial ports: Com1 disabled, Com2 on P278h, IRQ-3
Video:	VGA
Others:	National Instruments "AT-MIO-16X", internal, for analog I/O (initial setup: P220, IRQ-11, DMA 6&7)
	National Instruments "PC-DIO-24", internal, for parallel I/O (initial setup: P210, IRQ-4)
	National Instruments "AMUX-64T", external, expands 16 channel Analog I/O to 64 channels

4.2.4.5 GCS1HOST Computer Software

Operating System:	Windows NT version 3.5
Network Protocol:	NETBEUI on 3COM card, for instrument subnet access
	TCP/IP on AMD net port, for HLAN access
Commercial:	LabView for Windows & NT, by National Instruments, version 3.1
Resource Tools:	SCHEDULE.EXE -- launches specific application routines at designated times

Developed Interface Apps:

FTIRZIP.EXE	--	Zips previous days interferograms & calculated data files and sends the result to the designated network directory across HLAN
GC1ZIP.EXE	--	Purges every N^{th} chromatogram unless it meets change or significance criteria and zips the previous day's data files into one, then sends the result to the designated network directory
GC1APP32.EXE	--	Retrieves data from GC1 CPU, reformats data, relabels the files, and stores data locally
GC2ZIP.EXE	--	Same as GC1ZIP.EXE except adds two fields of data
GC2APP32.EXE	--	Same as GC1APP32.EXE except adds two fields of data
GCSVI.EXE	--	Reads process variable data directly from the sensors (temperature, pressure, & flow) then displays and logs it

[4] Pentium is a trademark of Intel.

4.2.4.6 GCS1HOST Computer Hard Disk Organization (only the essential first level directories are shown)

```
C:\   (340 Meg.)
  ├──BIN
  ├──DOS
  ├──NETWORK
  ├──NIDAQNT
  ├──RESKIT35
  ├──SCSI
  ├──USERS
  ├──TEMP
  ├──WINNT35
  └──WIN32APP

D:\   (680 Meg.)
  ├──CAL
  ├──FTIR
  ├──GC1
  ├──GC2
  └──NI
```

4.2.4.7 GCS1HOST Network Shares and Connections

Shares from GCS1HOST include:

Share Name	Path	
ADMIN$	C:\WINNT35	(password protected)
C	C:\	(password protected)
D	D:\	(password protected)
FTIR	D:\FTIR	(password protected)
GCS1CD	E:\	(no password)
DJet	Printer (HP DeskJet 540)	

Connections from FTIR include:

Connection Name	Drive Letter Used	
GC1\GC1	F:	
GC2\GC2	G:	
LABHOST\GCS1DATA	H:	
WHC129\COMMON	S:	(may be any 200 Areas general purpose server)

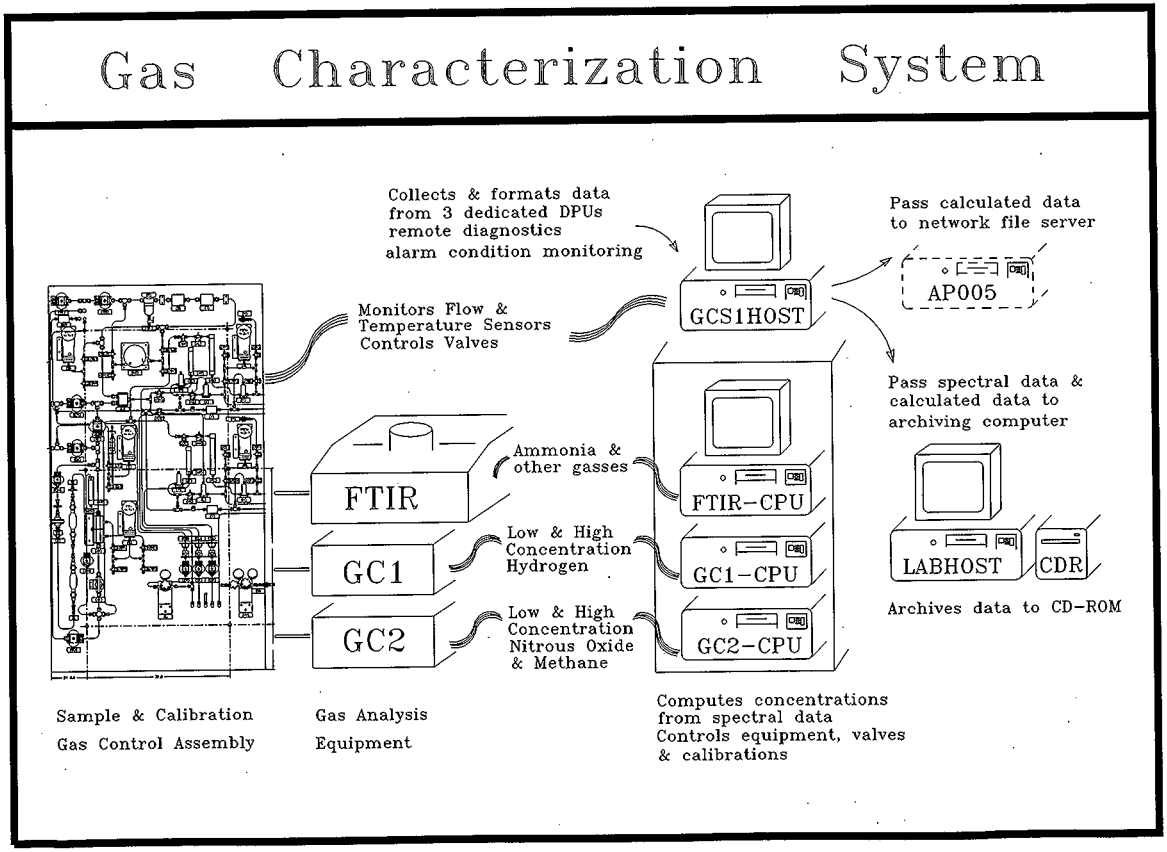


Figure 4C -- Gas Characterization System Physical Configuration

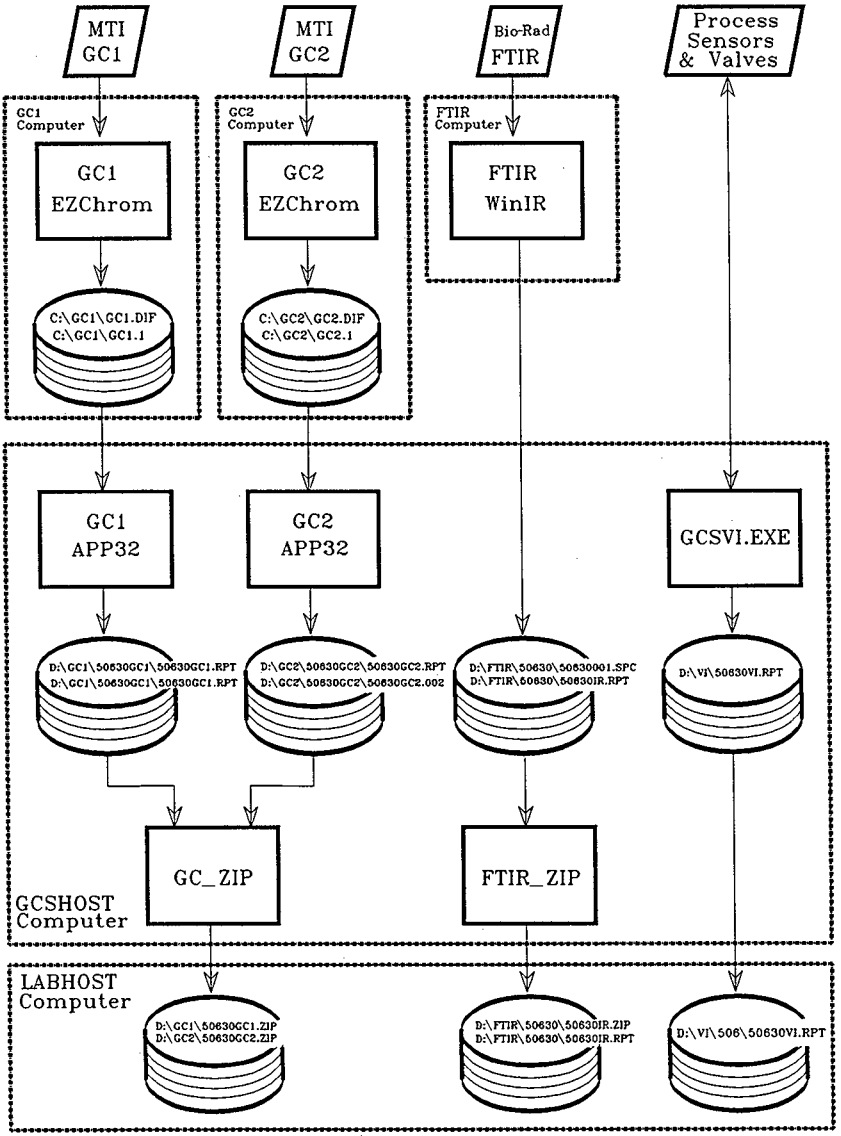


Figure 4D -- High Level GCS Software/Data Flow

4.3 Functions and Flow of the GCS Software

The Gas Chromatograph instrumentation hardware was delivered from the vendor with the EZChrom 200 Data System software package included. Similarly the Fourier Transform Infrared Spectrometer was shipped with the Grams 386 software package included for control of the instrumentation hardware and processing of its output. Integration software to automate the vendor systems and turn the output into customer usable data, was developed on site. The following register defines the software modules developed on site and points to it's source code listing.

Software App Name	Source Code Listing/ Development Language	Description
GC1ZIP.EXE	Appendix "C" / C^{++}	Purges every N^{th} chromatogram unless it meets change or significance criteria and zips the previous days data files into one, then sends the result to the designated network directory
GC1APP32.EXE	Appendix "D" / C^{++}	Retrieves data from GC1 CPU, reformats data, relabels the files, and stores data locally
GC2ZIP.EXE	Appendix "E" / C^{++}	Same as GC1ZIP.EXE except adds two fields of data
GC2APP32.EXE	Appendix "F" / C^{++}	Same as GC1APP32.EXE except adds two fields of data
FTIRZIP.EXE	Appendix "G" / C^{++}	Zips previous days interferograms & calculated data files and sends the result to the designated network directory across HLAN
GCSVI.EXE	Appendix "H" / C^{++}	Reads process variable data directly from the sensors (pressure, flow, and temperature) then displays and logs it

4.3.1 Gas Chromatograph

Both Gas Chromatographs function identically except for the extra two columns of data produced by GC2. From each GC, massive chromatogram files are produced on a continuous basis. GCS software converts each chromatogram into calculated concentrations of pre selected gasses. The software also serves many other functions:

- Place a date/time stamp on each dataset
- Identify each calibration
- Format the calculated results for readability
- After calculation, delete a pre-defined number of chromatograms showing no change from previous readings.
- Compress the data into a single calculated data file and a single chromatograms file per day. This minimizes storage requirements and reduces file clutter.
- Trigger instrument calibrations at defined intervals

4.3.1.1 GC1ZIP.CPP & GC2ZIP.CPP Program Description

The objective of using GC1ZIP and GC2ZIP programs is to reduce the large number of files generated by the chromatograms. Another

consideration is that a large amount of storage is consumed by the chromatograms, this storage allocation is reduced by joining all chromatograms from a single day into one file. The GCZIP data reduction and file handling algorithm is governed by the two initialization files known as INI files. The GC1_ZIP.INI and GC2_ZIP.INI files indicate the path of the executable of the pkzip program, the location of the files before they are to be zipped, their location after they are zipped, the number of the chromatogram to save(this condition depends on a number of parameters to be discussed later).

The GCZIP program receives an integer, in this case 1 and 2 to read the appropriate INI file, fills the string called date_name with the appropriate date. The integers for the day are then pulled out of the date string program, these integers are decremented by one day, in order to zip the files for the previous day. The next step in the GC1_ZIP and GC2_ZIP program is to build the strings for each of the commands executed above. The first string will be used to build the path for the executable for the pkzip file. The second string will be used to build the path for the location of the files to be zipped, the third and final string indicates the location of the files that are zipped. These locations are arbitrary and can be assigned to any drive, by changing the variables in the GC1_ZIP.INI and GC2_ZIP.INI files.

After the input/output path strings are built, the next step is to build the data compression routine. The data compression routine's main task is to selectively save chromatograms if any of the concentrations for the gases are zeros, or if any of the chromatograms are for calibration purposes. The routine skips the header file, reads the next set of concentrations into an array. A while loop is used to read in the number of chromatograms(same as number of lines in the program),convert each field of the array from an ascii character to a float.

A variable called crname is constructed to contain the base definition of the chromatogram name(i.e 50806. where the 5 stands for the year 1995, the 08 stands for the month and the 06 stands for the day and a period is attached for purposes of attaching an index to the end of this). An if statement is used to check whether any of the lines are used for calibration, or whether any of the gases have zero concentration, if this is true then these chromatograms are saved else they are deleted. After this deletion the variable crname is reset to its initial contents(i.e from 50806.123 to 50806.). As the compression routine is ended the next 10 to 15 lines of the gczip code are used to build a string to execute the pkzip command specifying the location of the files before and after the zipping operation and the actual executable location of the pkzip file. After the zipping operation is completed the source files that are stored locally are deleted and the directory is removed.

The resultant zipped GC file contains one days chromatograms and calculated data. Each is named using the following format.

year	month	day	instrument
5	08	28	GC1.ZIP

4.3.1.2 GC1APP32.CPP $ GC2APP32.CPP Summary

The GC1APP32 and GC2APP32 programs are used to retrieve data from the gas chromatograph, put the data into a file, then renumber the index and rename the report file consistent with the date. The program starts out with the opening of the dialogue box. An INI file exists for each application (GC1APP32.INI, GC2APP32.INI) which contains several different variables, including string identifiers for the backslash, the report and chromatogram output file root path etc. This INI file is read and the day name which is in the form 50901gc1 where the 5 stands for the year, the 09 stands for the month and the 01 stands for the day, the gc1 additive differentiates between each of the two gas chromatographs.

A directory is created for the day stamp of the form shown above. The file Header.H is copied to the report file for the day. In this case a search is made to find if the current day's file exists, if so then the last index is obtained from the report file else the reportindex is set to zero. A search then is conducted for the dif file, if the dif file is found all the files with the first three characters of gc1(or gc2) are deleted. Finally the notify thread is executed to indicate the last write time of the dif file. This thread is set so that it times out after 10 minutes.

After all the chromatograms and the dif files are completely written for the day, a variable called FindCal is used to search for the calibration file created by the labview VI. If the calibration does not occur the handle containing the calibration file is closed.

4.3.2 FTIR Software

4.3.2.1 FTIRZIP.CPP

The zip program for the ftir is similar to the program to zip the gc application programs. This is the main module of the ftir zip program. As is always the case some definitions are included in the beginning to signify the assignment of null terminators to all the variables which have nothing stored in them. The FTIRZIP.INI file is opened by the next statement (i.e. the ifstream statement). This INI file contains the original location of the files to be zipped and the final location of the files to be zipped. The lines from the FTIRZIP.INI file are copied onto the variables zip_path and zipped_path. The next few lines are used to create the string containing the directory where the files are to be zipped from.

The variable called zipped_path contains the destination directory where the files are zipped to. The next few lines are used to construct the string to make a destination directory for the .zip file to go to(this directory is of the form c:\ftir\50719\50719IR.zip). An "-m" is added to the above string to signify the deletion of all the files after the zipping process is completed. The final string is stored in the variable call which contains the string of the form "call c:\bin\pkzip.exe -m c:\ftir\50719\50719IR.zip d:\ftir\50719*.*". Two system calls are executed, one to copy the report file over to the destination directory and the other to execute the above string.

GC1_FTIR.CPP -- The beginning of this module contains definitions or each of the months of the year, in the variable space. The date and time stamp are obtained and put into the variable tms. The variable fname is then used to create the string in the form (50806 where 5 stands for the year 1995, 08 stands for the month and 06 stands for the day).

The next set of 20 or so lines is used compare the first three elements of the character array fname to various versions(versions means space0 space1 etc) of the variables space designating the months of the year. If there is a match between these two variables then the variable month is assigned an integer accordingly. The next few lines get the integer equivalent of the day from the time string, if the day is less than 10 a zero is assigned to the variable day. After the variable day is assigned correctly the variable i_day which is an integer is constructed from this variable.

An outer if statement is used to distinguish the first day of each month. Within this if statement is a switch statement which differentiates between each of the months(i.e if the month is May the new day is then set accordingly to Apr 30, if the month is September the new day is set to Aug 31 etc). If the current year happened to be a leap year the month of february gets assigned 29 days. An if statement is used to signal a leap year. Inside each of the case statements within the switch statement the variable fname is constructed to have the correct month, the variable day is also constructed to contain the last day of the month. Finally null terminators are assigned to each of these variables. After the proper integer is assigned to the variable month, this integer is decremented by one.

After the correct month is obtained an if statement is used to assign a zero to fname if the month is less than 10. If the month is greater than 10 fname gets assigned the proper value of the month depending on the variable temp. This variable temp is used to store the contents of the variable month as an ascii character. An else if statement is used to assign zero in front of the day when the day is less then 10. Finally the variable i_day is decremented by one and converted back to the ascii variable day with a null terminator.

A new variable called month1 is constructed based on the variable month, this variable is just the ascii equivalent of the month. If the current month is less than September a zero is added onto the beginning of this variable. If the current month is greater than September the variable month1 is copied onto the variable fname1. At this point fname1 contains the string in the form 50806 with the year, the month and the day attached. The phrase IR is then catenated onto the end of this phrase. The extension .zip is then catenated onto the end of this phrase to create the final expression for the file to be zipped.

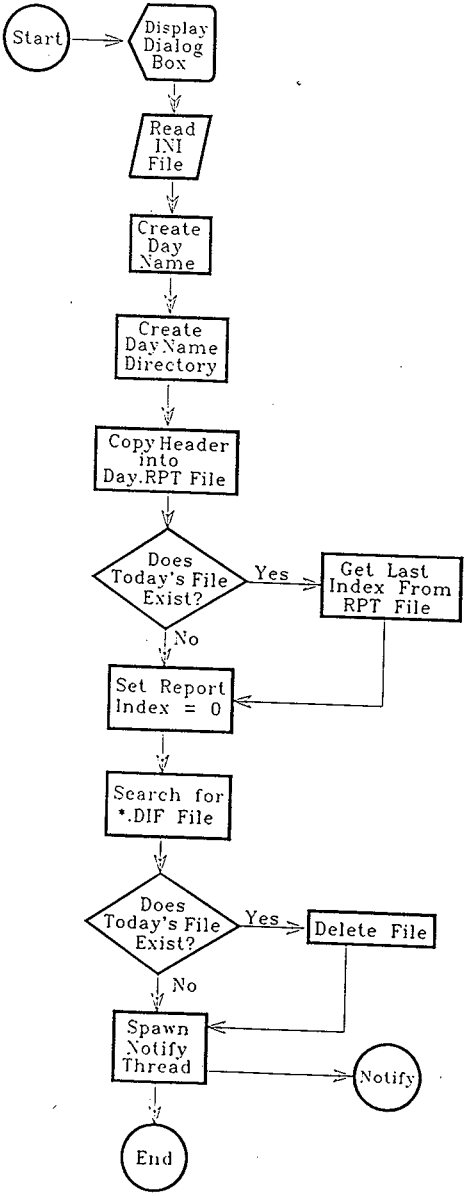


Figure 4E -- Interface Program "GCAPP32" Software Flow

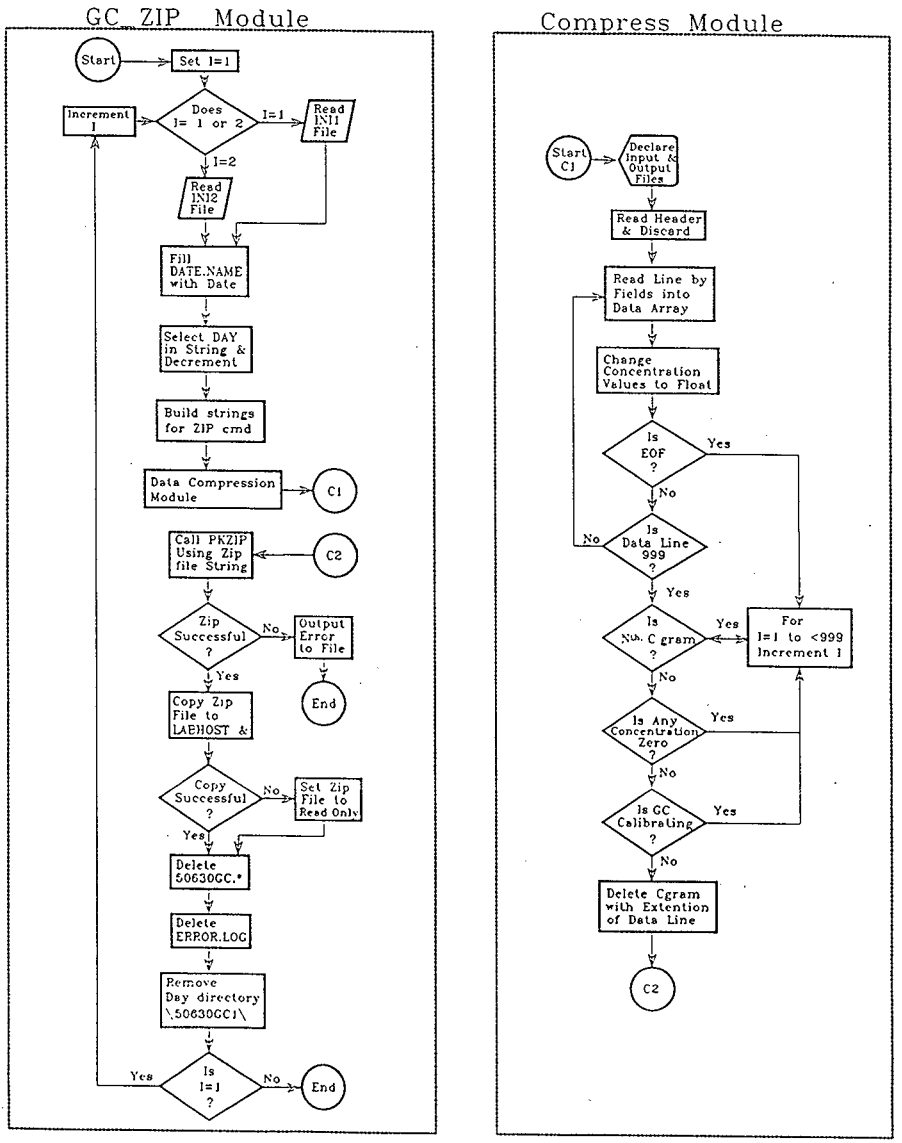
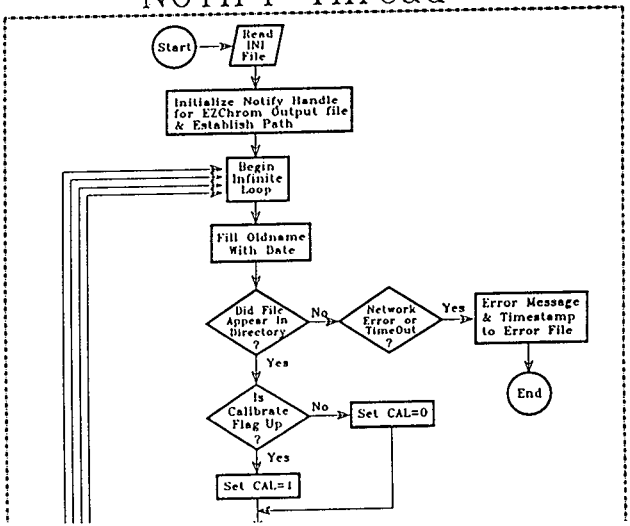


Figure 4F -- Interface Program "GC_ZIP" Software Flow

NOTIFY Thread



4.4 LABHOST Computer

The computer labeled LABHOST is not a component of the field located GCS. It resides in the development lab to serve several purposes. It's configuration is therefore flexible, outside of the hardware setup used by each field Host computer.

4.4.1 The primary function of LABHOST is to make the data from the field GCS modules accessible by the data users (Fluor Daniel Hanford, Battelle, and LANL Scientists and Engineers). This is accomplished by it's network accessibility (HLAN).

4.4.2 The second function of LABHOST is the archiving of all permanent data from each field GCS unit. Field GCS units transfer their data to LABHOST where it is formatted by month and written to a standard 5.25" CD-ROM. This form has long term stability (in excess of 10 years), and is convenient for transferring the 650 Megabytes of data written to each CD, when someone needs to review the data.

4.4.3 A third function of LABHOST is as the emergency spare for any GCSHOST in the field. It is identical to the field host CPU including internal port and network cards. This availability as emergency spare is significant on weekends and holidays, when other alternatives are few.

4.4.4 The fourth function of LABHOST is as a monitoring and diagnostic tool.

4.4.5 LABHOSTs fifth function is as a development and modification tool for the GCS. Any changes that may become necessary to accommodate measuring other gasses or multiple tanks from the same GCS will be developed and tested on LABHOST. Not only does LABHOSTs configuration as a spare enable it to duplicate the GCSHOST CPUs for testing, but the development software (i.e. Borland C++, LabView, & Grams386) also resides on it.

4.4.6 LABHOST Computer Hardware

Model:	AST Premmia GX P-100, EISA/ISA/PCI bus compatible
Processor:	Pentium at 100 MHz
Memory:	32 megabytes of RAM
Disks:	A: - 3.5" HD floppy
	C: - 1.2 gigabyte hard disk (internal)
	D: - 1.2 gigabyte hard drive (internal)
	E: - CD ROM drive (internal)
	F: - 2.1 gigabyte hard drive (external)
	G: - Writable CD ROM (external)
Ethernet:	AMD on the motherboard for HLAN access, AUI connector
	3COM Etherlink III network card on thin-net (BNC connector)
	(not bridged or forwarded to HLAN)
Video:	VGA

4.4.7 LABHOST Computer Software

Because LABHOST is a development computer, serving several functions, it's software configuration may change to accommodate the

purpose it is serving at the moment. Therefore the following configuration is only a recommendation for initial setup:

Operating System:	Windows NT version 3.5 (or later)
Network Protocol:	NETBEUI on 3COM card, for instrument subnet access TCP/IP on AMD net port, for HLAN access
Commercial:	LabView for Windows & NT, by National Instruments, version 3.1 Specifically: Borland C^{++}, LabView, Grams386, ...
Resource Tools:	SCHEDULE.EXE -- launches specific application routines at specific times

4.4.8 LABHOST Computer Hard Disk Organization

The first level directories essential to the GCS are dedicated to "D" hard disk.

LABHOSTs "C" drive may be reconfigured to serve other development tasks as long as it retains the ability to fully support the GCS field units (GCS1HOST, GCS2HOST, etc.). Drive "D" is dedicated to GCS development code and data from the GCS field units, for the purpose of archiving to CD-ROM. Bernouli drives may or may not be part of the LABHOST configuration.

```
D:\
  ├──CODE
  ├──FTIR
  ├──GC1
  └──GC2
```

H:\ (CDR hard drive for preparation of CD image only)

4.4.9 LABHOST Network Shares (HLAN) include:

Share Name		Path	
ADMIN$		C:\WINNT35	
C		C:\	(password protected)
D		D:\	(password protected)
GCS1DATA		D:\GCS1	(password protected)
GCS2DATA		D:\GCS2	(password protected)
Brnli-E	Φ	E:\	(password protected)
Brnli-F	Φ	F:\	(password protected)
CD		G:\	(no password)
LJ2	Φ	Printer (HP LaserJet II)	

4.5 Vendor Information File

Documentation and manuals for the commercial software used in the GCS are on file in CVI file # 22665.

Φ = Not a required part of the GCS configuration

5.0 OPERATION

5.1 Overview

The purpose of the operation procedure (Appendix B) is to provide instructions for the startup and operation of the computer in the data acquisition and control system and the following equipment in the GCS cabinet:

1) 486/33 or better host computer

2) GC1 Gas Chromatograph control computer

3) GC2 Gas Chromatograph control computer

4) FTIR control computer

5.2 Operating Procedures

5.2.1 System Startup, Normal Operation, Shut Down

See Appendix B for current operating procedure.

5.3 Discrepancy Reporting & Corrective Actions

Any person who identifies any problems or discrepancy at any time shall report it to the cognizant design engineer. The cognizant design engineer shall include a written description of the problem or discrepancy in the design file under "Problems". The cognizant design engineer shall provide a copy of this written description to the cognizant manager responsible for characterization and monitoring systems. The corrective action is recorded in an action-tracking database by the cognizant manager. The cognizant manager and the cognizant design engineer are responsible for assuring correction of the problem or discrepancy.

6.0 CONFIGURATION CONTROL

Each software file shall contain a functional description of the contents of the file. The contents of the file shall have the revision history. The revision history shall include a description of the initial release and all subsequent changes to the file. The description shall include the following:

- ♦♦ Revision number/revision date
- ♦♦ Nature of change
- ♦♦ Identity of requester
- ♦♦ Rationale for the change
- ♦♦ Identity of author making the change

6.1 Configuration Status Accounting

Characterization Monitoring Development shall prepare a status accounting file before software release. The file will be included in the release software description document. The status accounting shall include a list of all software files that make or produce a released product. The status accounting file shall include the following:

- -- Name and revision of released product
- -- Name and revision of all software files

Instructions on how to build the software product from the files shall include identification of support software. Support software includes compilers, assemblers, editors, operating systems. and other utility software.

6.2 Media Control

The form of media used to store approved release software shall be 12 cm. CD ROM disks. The software media shall be stored in at least two locations. One set of the software media shall be stored in a file cabinet drawer labeled "software records" in the file named "GCS software". This file cabinet will be located at a location specified by the manager of Characterization Monitoring Development. The other sets of software media shall be stored in each of the field GCS huts with the GCSHOST computers. Each approved release software CD shall bear the name "GCS", the version number, a disk number (sequential integer starting at 1), and the initials of the GCS Design Engineer, with date.

Each approved release software CD shall contain all software necessary to load and operate any computer in the system.

7.0 SAFETY, RELIABILITY, & QUALITY ASSURANCE

7.1 Safety Classification and Signature Designation

The monitoring instruments and systems used for the Hanford Waste Tank Gas Characterization System are classified as a non-safety related General Service system in accordance with the requirements of WHC-CM-4-46, "Safety Analysis Manual". Failure of the GCS does not adversely effect the environment or the health and safety of the personnel operating the enclosure. General design and quality assurance requirements for Non-Safety items shall be followed.

Although the GCS function does not require it to be safety class or safety significant, its connection to the tank vapor space may necessitate designation of protective equipment as Defense-in-Depth Safety Equipment List components.

At a minimum, documentation must satisfy an SQ Documentation Approval Designation Level with Environmental Assurance (E) as deemed appropriate. A ESQ is a designator for documentation that impacts occupational safety [including "as low as reasonably achievable" (ALARA) principles], environmental monitoring, and requires quality assurance verification of conformance to requirements. These have been determined in accordance with WHC-CM-6-1, *Engineering Practices*, EP-1.4, "Safety Classification", and WHC-CM-1-3, *Management Requirements and Procedures*, MRP 5.43, "Approval Designators".

7.2 Quality Assurance and Quality Control

All work is conducted in accordance with the relevant quality requirements of WHC-CM-6-1, EP-1.0, "Engineering Practices" (i.e. Design Control, Design Verification, Change Control, Interface Control, Instruction, Procedures, and Drawings), and WHC-CM-4-2, "Quality Assurance Manual."

Equipment that is deployed in Hanford's waste tank applications must be in compliance with National Environmental Protection Act requirements and have appropriate documentation to address safety and regulatory issues associated with the deployment activity.

8.0 REFERENCES

A) Payne, M. A.,1994, "*In-Tank Instrument Safety Classification*", (Internal Memo, 9307342B R1, to R. E. Gerton, October 14, 1993), Westinghouse Hanford Company, Richland, Washington.

B) WHC-CM-3-5, "*Approval of Environmental, Safety, and Quality Affecting Documents*", Fluor Daniel Hanford Company, Richland, Washington.

C) WHC-CM-4-46, "*Nonreactor Facility Safety Analysis Manual*", Fluor Daniel Hanford Company, Richland, Washington.

D) WHC-CM-4-2, "*Quality Assurance Manual*", QR-12, Fluor Daniel Hanford Company, Richland, Washington.

E) WHC-CM-6-1, "*Standard Engineering Practices*", Fluor Daniel Hanford Company, Richland, Washington.

9.0 BIBLIOGRAPHY

9.1 U.S. GOVERNMENT

DOE, 1989, DOE Order 6430.1A, *General Design Criteria*, U.S. Department of Energy, Washington, D.C.

DOE-RL, 1990, DOE-RL Order 6430.1C, *Hanford Plant Standards (HPS) Program*, U.S. Department of Energy, Richland Field Office, Richland, Washington.

9.2 FLUOR DANIEL HANFORD COMPANY

SD-WM-SAR-016 Safety Analysis Report Double Shell Tank Farms 241-AN, AW, AP & SY (OSR-T-152-00001)

OSD-T-151-00007 Unclassified Operating Specifications for 241-AN, AP, AW, AY, AZ, and SY Tank Farms

9.3 CODES AND STANDARDS

American National Standards Institute (ANSI)

ANSI/IEEE Std 100, *Standard Dictionary of Electrical and Electronics Terms*, 1988.

ANSI/UL 508, *Standard for Industrial Control Equipment*, 1988.

Institute of Electrical and Electronics Engineers (IEEE)

IEEE Standard 1016-1987.

National Electrical Manufacturers Association, (NEMA)

NEMA ICS-1, *General Standards for Industrial Control Systems*, 1988.

National Fire Protection Association (NFPA)

NFPA 70, *National Electrical Code*, 1990.

NFPA 493 - Standard for Intrinsically Safe Apparatus and Associated Apparatus for Use in Class I, II, and III, Division 1, Hazardous Locations

Underwriter's Laboratory

UL 698 - Standard for Industrial Control Equipment for Use in Hazardous Locations

UL 913 - Standard for Intrinsically Safe Apparatus and Associated Apparatus for Use in Class I, II, and III, Division 1, Hazardous (Classified) Locations

LIST OF APPENDICES

APPENDIX A -- DEFINITIONS & ACRONYMS

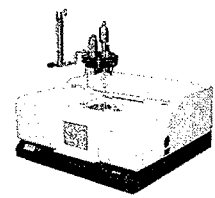
Background Interferrogram containing data from atmospheric air. Subtracted from sample interferrogram to yield components in tank vapor space.

Calibration Process of updating the response factor or linearization curve used to convert peak areas to concentration.

Centerburst Diagnostic interferrogram used to evaluate signal strength of FTIR

Chromatogram Raw data acquired by a gas chromatograph. Consists of binary or ASCII representation of detector signal versus acquisition time. Injection of sample or calibration standard results in peaks in the chromatogram. Areas under the peaks are set proportional to concentration.

FTIR Fourier Transform InfraRed Spectrometer -- This instrument uses a helium-neon laser to produce a mid-infra-red energy source, which is directed through the sample gas chamber. Various molecules absorb infrared energy at only specific frequencies. A graph of energy absorbed versus frequency is the absorption spectrum of the sample. Specific frequency absorption peaks represent specific molecules. The size of the peak (integrated area under the curve) represents the relative concentration of the molecule in the sample.



GC Gas Chromatograph -- This instrument uses the thermal conductivity of each gas to produce quantitative measurements. A molecular sieve slows down the larger molecules and allows quick passage of the small ones. As each gas passes into the detection chamber, after a specific delay time, the amount it cools a detector wire is plotted versus time. The size of the peak (integrated area under the curve) represents the relative concentration and the time defines the molecule in the sample stream.

Interferrogram Raw data acquired by an infrared spectrometer. Consists of binary or ASCII representation of detector signal versus wavelength. Injection of sample or calibration standards results in peaks in the interferrogram. Areas under the peaks are set proportional to concentration.

ppm parts per million -- standard unit of relative gas concentration

Response Factor Linear proportionality constant relating area under a chromatogram's peak to gas concentration. Response factor or RF is defined as Area/Concentration.

Training Spectra A series of interferrograms collected by an FTIR under known conditions (gas concentration, pressure, flow, path length, etc...). Used to generate linear or non-linear calibrations for FTIR. Training spectra may be generated in the laboratory or consist of electronic data base libraries such as the HITRAN data base.

Vapor Space The Vapor Space is the atmosphere between the Waste surface level and the dome of the waste tank.

Verification Verification refers to process of injecting standard calibration gases into analytical systems and comparing the instrument's response to these standards. Verification also refers to the process of comparing FTIR spectra with archived electronic training spectra.

ZIP A file compression and joining program that is used by the GCS to join the many spectrogram files from each day into one large file and reduce the size by data compression to the maximum possible.

APPENDIX B -- OPERATING PROCEDURES

OPERATING MANUAL FOR GAS CHARACTERIZATION SOFTWARE AND INSTRUMENT SYSTEMS

1.0 INTRODUCTION

The flammable gas watch-list (FGWL) tanks, which have demonstrated a Gas Release Event (GRE) exceeding 0.625% hydrogen by volume will require additional characterization. The purpose of this additional characterization is to accurately measure the flammable and hazardous gas compositions and resulting Lower Flammability Limit (LFL) of the tank vapor space during baseline and GRE emissions. Data from this characterization will help determine methods to resolve the Unreviewed Safety Question (USQ) for the FGWL tanks.

Currently, two Gas Characterization Systems (GCS) are being installed on tanks 105-AN and 101-AW. Each of the GCS systems will include two gas chromatographs (GC), a Fourier Transform Infrared Analyzer (FTIR), and a LabView data acquisition system. These instruments will provide the system with the capabilities to measure a wide range of gases from parts per million to percent volume concentration levels.

2.0 SCOPE

The objective of this document is to provide user desk instructions for the routine start-up, operation, and shutdown of the Gas Characterization Systems.

3.0 PREREQUISITE

Conduct of the procedure steps outlined in the document assumes reasonable proficiency with personal computers and the Windows operating system (Windows NT and Windows for Workgroups) including familiarity connecting network drives. Additionally, the Operator should be familiar with operation of the gas chromatographs and infrared spectrometer and associated software.

4.0 DESCRIPTION

The GCS instrument/data acquisition system consists of networked computers and analytical instruments. Vendor supplied software is used to control operation and analysis functions of the analytical instruments and custom developed software is used to collect, store, organize, and transfer the data. Figure attached shows the computer and instrument layout for the GCS. A brief description of the hardware software systems is given in the following sections. A more detailed description is detailed in Hardware Functional Design Description document.

4.1 GC1 Hardware/Software

The GC1 sub-system consists of an MTI gas chromatograph and a 486 66-MHz personal computer, GC1 networked to the Pentium host computer GCS1HOST (Note: currently two systems are being installed the Host computer for the first system will be designated GCS1HOST and the Host computer for the second system will be designated GCS2HOST - The remainder of this document will refer to the Host computer as GCS1HOST). Both computers run the Windows NT or Windows 95 operating system. The gas chromatograph is configured to measure hydrogen and is equipped with two molecular sieve and thermal conductivity detector/collum modules. The software system for running GC-1 consists of the following:

EZCHROM Version 4.0	Vendor supplied software for controlling the MTI gas chromatograph and analyzing data. Software resides on GC1 computer.
GC1APP32 Version 1.0	Custom developed software for transferring EZCHROM data from the GC1 computer to the GCS1HOST computer. Software also organizes and renames the data files using a standard date/run-number naming scheme. Software resides on the GCS1HOST computer.
GC1ZIP Version 1.0	Custom developed software for compressing and transferring GC1 data from GCS1HOST to the 300 area data server, LABHOST. Program is run at daily intervals using the Windows NT command scheduler.

4.2 GC2 Hardware/Software

The GC2 sub-system consists of an MTI gas chromatograph and a 486 66-MHz personal computer, GC2 networked to the Pentium host computer, GCS1HOST. Both computers run the Windows NT or Windows 95 operating system. The gas chromatograph is configured to measure methane and nitrous oxide and is equipped with two poraplot-Q and thermal conductivity detector/collum modules. The software system for running GC2 consists of the following:

EZCHROM Version 4.0	Vendor supplied software for controlling the MTI gas chromatograph and analyzing data. Software resides on GC2 computer.
GC2APPS Version 1.0	Custom developed software for transferring EZCHROM data from the GC2 computer to the GCS1HOST computer. Software also organizes and renames the data files using a standard date/run-number naming scheme. Software resides on the GCS1HOST computer.

GC2ZIP Version 1.0	Custom developed software for compressing and transferring GC2 data from GCS1HOST to the 300 area data server, LABHOST. Program is run at daily intervals using the Windows NT command scheduler.

4.3 FTIR Hardware/Software

The FTIR sub-system consists of a Bio-Rad FTS-175 IR spectrometer and a 486 66-MHz personal computer, FTIR networked to the Pentium host computer GCS1HOST. The FTIR computer runs the Windows for Workgroups operating system and the GCS1HOST runs Windows NT or Windows 95. The FTIR is configured to measure ammonia but can readily be configured to measure a wide variety of gases. The software system for running FTIR consists of the following:

WIN_IR Version 2.0	Vendor supplied software for controlling the FTIR and analyzing data. Software resides on FTIR computer. In addition to controlling the spectrometer the WIN_IR software uses user developed array basic programs to control analysis and reporting functions. These programs include ICOMPUTE.AB, JUANTBAS.AB and SPFTIRII.AB. These array basic programs are started automatically when executing the radio button labeled SPFTIRII in WIN_IR.
FTIRZIP Version 1.0	Custom developed software for compressing and transferring FTIR data from GCS1HOST to the 300 area data server, LABHOST. Program is run at daily intervals using the Windows NT command scheduler.

4.4 Process Instrumentation Data Logging Hardware/Software

The process data logging sub-system uses National Instruments data acquisition boards and software installed on the GCS1HOST computer to monitor and record system variables such as temperature and flow. In addition, this subsystem also controls automatic timed injections of calibration gases into the GC systems.

GCSVI Ver 1.0	Labview developed executable program to log system temperatures, flows, and pressures. Also controls automatic GC calibrations and performs data transfer and archive functions.

5.0 PROCEDURE

The following procedure describes steps for setting up, operating, and troubleshooting the GCS systems. Since all data is transferred to the 300 Area Data Server (LABHOST) a prerequisite to this procedure is to ensure that LABHOST is on and connected to the network. UNDER NO CIRCUMSTANCES SHOULD LABHOST BE TURNED OFF OR REMOVED FROM THE NETWORK WITHOUT THE SYSTEM ADMINISTRATORS ACKNOWLEDGMENT. Unplanned system and network outages have been factored into the software system design and in this event all data will be saved locally, however because of the large amount of data generated, local hard drives can not store more than 1 week of data.

5.1 Initial Setup

5.1.1 Ensure that all instrument (GC1, GC2, and FTIR) are powered up. Power up all computer systems GC1, GC2, GCS1HOST, and FTIR.

NOTE: SEQUENCE OF START-UP OF THESE COMPUTER SYSTEMS IS CRITICAL TO ENSURE PROPER ESTABLISHMENT OF NETWORK COMMUNICATIONS. With the exception of GCS1HOST all systems share a common monitor/keyboard/mouse which is activated by depressing the labeled button on the switching module. Passwords are required for all systems. The system administrator maintains the active user and Passwords list.

5.1.2 On the GCS1HOST computer, verify that the following shares and network drives are connected. NOTE: THE NAME AND LOCATION OF THESE DRIVES IS CRITICAL TO PROPER SYSTEM OPERATION.

SHARE	D:\FTIR	For FTIR computer
CONNECT	Drive F:	\\GC1\GC1
CONNECT	Drive G:	\\GC2\GC2
CONNECT	Drive H:	\\LABHOST\GCS1_HAN

If unable to establish network shares connect network drives please refer to Section 6.1 of this procedure - Troubleshooting Network Communications.

5.1.3 On the FTIR computer, connect the following network drive: NOTE: THE NAME AND LOCATION OF THIS DRIVE IS CRITICAL TO PROPER SYSTEM OPERATION.

CONNECT	Drive D:	\\GCS1HOST\FTIR

If unable to establish network shares connect network drives please refer to Section 6.1 of this procedure - Troubleshooting Network Communications.

5.1.4 On the GCS1HOST computer, start the Command Scheduler (in the GCSAPPS Group) and verify that the Interactive Mode is ON and that GC1ZIP.EXE, GC2ZIP.EXE, and FTIRZIP.EXE are set to run daily. Note: The time that these programs are launched is not critical, however in general it is preferable to run these during swing and grave-yard shift (4:00 PM to 5:00 AM) to ensure minimal network traffic. MINIMIZE BUT DO NOT CLOSE THE COMMAND SCHEDULER. For more information on using the Command Scheduler see Windows NT documentation.

5.2 Starting the FTIR

The default operating parameters for the FTIR instrument (analysis method, number of scans, scan type, etc...) are setup in the FTIR initialization file CONF001.FTS. The following procedure steps are in accordance with the parameters established in the CONF001.FTS. A complete description of the parameters of the file is contained in WHC-SD-WM-CSWD-077.

5.2.1 Activate the FTIR monitor/keyboard/mouse using the switching console "C".

5.2.2 Open the WIN_IR program group and launch the WIN_IR program.

5.2.3 Perform an initial alignment of the FTIR Optics. Select COLLECT - ALIGN from the WIN_IR Taskbar. This automatic alignment will take approximately 3 minutes.

5.2.4 Using **File**, Clear all slots for GRAM386 memory management.

5.2.5 Depress the SPFTIRII radio button in the upper right hand corner of the WIN_IR window. The WIN_IR display should change to read ..."Scanning 1 of 64....Scanning 2 of 64 etc..". If the display does not indicate scanning operation of the FTIR refer to Section 6.2 of this procedure Troubleshooting FTIR Operation.

5.3 Starting the GC1

The default operating parameters for the GC1 data transfer process(location of saved files, shared directories, file extensions, etc...) are located in the GC1_ZIP.INI file. The following procedure steps are in accordance with the parameters established in the GC1_ZIP.INI file. A complete listing and description of the contents of the INI file is contained in WHC-SD-WM-CSWD-077. Conduct of the following procedure steps assumes familiarity with using EZCHROM to control and operate and MTI GC. The MTI/EZCHROM users manual provides an excellent reference for all aspects of EZCHROM Operation.

5.3.1 Activate the GC1 monitor/keyboard/mouse using the switching console "A".

5.3.2 Open the MTI program group and launch the EZCHROM program.

5.3.3 Select METHOD - OPEN and open c:\MTI\EZCHROM\200\METHODS\GC1.MTD.

5.3.4 Select INSTRUMENT - SEND CURRENT METHOD to upload the selected method to the GC.

5.3.5 Select DATA-SAVE AS and enter C:\GC1\ in the dialogue box to establish default save path.

5.3.6 From the GCS1HOST computer, open the GCSAPPS program group and launch GC1APP32.

5.3.7 Select the RUN radio button in the GC1APPS program Window.

5.3.8 Select the External Standard Report.

5.3.9 From the EZCHROM application at the GC1 computer, select START. In the dialogue box enter the RUN ID as GC1, set number of runs to INFO, select time between injections as 360, ensure the SAVE and DIF SAVE options are selected. DO NOT SELECT THE EXTENDED OPTION.

5.3.10 Click on the START radio button to begin Run.

5.3.11 EZCHROM display should change to indicate operation of the Gas Chromatograph. If display does not indicate proper operation of the Gas Chromatograph refer to Section 6.3 of this procedure Troubleshooting MTI GC Operation.

NOTE: EZCHROM IS NOT MULTI-TASKING - SO AFTER IT'S RUNNING LEAVE IT ALONE!!!

5.4 Starting the GC2

The default operating parameters for the GC2 data transfer process (location of saved files, shared directories, file extensions, etc...) are located in the GC2_ZIP.INI file. The following procedure steps are in accordance with the parameters established in the GC2_ZIP.INI file. A complete listing and description of the contents of the INI file is contained in WHC-SD-WM-CSWD-077. Conduct of the following procedure steps assumes familiarity with using EZCHROM to control and operate and MTI GC. The MTI/EZCHROM users manual provides an excellent reference for all aspects of EZCHROM Operation.

5.4.1 Activate the GC2 monitor/keyboard/mouse using the switching console "B".

5.4.2 Open the MTI program group and launch the EZCHROM program.

5.4.3 Select METHOD - OPEN and open c:\MTI\EZCHROM\200\METHODS\GC2.MTD

5.4.4 Select INSTRUMENT - SEND CURRENT METHOD to upload the selected method to the GC.

5.4.5 Select DATA-SAVE AS and verify C:\GC2\ is in the dialogue box to establish default save path.

5.4.6 From the GCS1HOST computer, open the GCSAPPS program group and launch GC2APP32.

5.4.7 Select the RUN radio button in the GC2APPS program Window.

5.4.8 Select the External Standard Report.

5.4.9 From the EZCHROM application at the GC2 computer, select START. In the dialogue box enter the RUN ID as GC2, set number of runs to INFO, select time between injections as 360, ensure the SAVE and DIF SAVE options are selected. DO NOT SELECT THE EXTENDED OPTION.

5.4.10 Check the Run Time on GC-1 to verify that it is between 200 - 160 sec.

5.4.10 Click on the START radio button to begin run.

5.3.11 EZCHROM display should change to indicate operation of the Gas Chromatograph. If display does not indicate proper operation of the Gas Chromatograph refer to Section 6.3 of this procedure Troubleshooting MTI GC Operation.

NOTE: EZCHROM IS NOT MULTI-TASKING - SO AFTER IT'S RUNNING LEAVE IT ALONE!!!

5.5 Starting LabView Data Acquisition System

The LabView data acquisition system logs process variables and initiates automatic calibration verifications of the GCs. The following procedure assumes start-ups with the default settings. A complete description of user selectable settings is contained in WHC-SD-WM-CSWD-077.

5.5.1 From the GCS1HOST computer, open the GCSAPPS program group.

5.5.2 Launch the GCSVI application.

5.5.3 GCs auto calibration - Dial in the TIME and DURATION.

5.5.4 Depress the radio button.

5.5.5 Verify LOG TO FILE pushbutton.

5.6 Verify all systems are operating and communicating with GCS1HOST

5.6.1 From GCS1HOST maximize the GC1APP32 and GC2APP32 program window.

5.6.2 Verify that the run number index for both programs are incrementing. NOTE: GC run time is approximately 3 minutes, consequently the run number should be incremented about every 3 minutes. If the system does not update within 10 minutes the programs will "time out". The text in the window will indicate "PROGRAM TERMINATED".

5.6.3 If the GC1APP32 and/or GC2APP32 are not incrementing refer to section 6.3 of this procedure, Troubleshooting GC Operation

5.6.4 Open the Windows File Manager.

5.6.5 Browse the D drive and select the D:\FTIR directory.

5.6.6 Browse subdirectory under D:\FTIR\ymmdd NOTE: This directory is created automatically by the FTIR program. This directory will not exist until the FTIR has completed one complete analysis (64 scans).

5.6.7 Verify that the time stamp of the current data file corresponds with the current data and time.

5.6.8 If there is no current data file in D:\FTIR then refer to Section 6.2 "Troubleshooting FTIR Operation".

5.6.9 Browse the D drive and select the D:\NI directory.

5.6.10 Verify that the time stamp of the current data file corresponds with the current data and time.

5.6.11 If there is no current data file in D:\NI then refer to Section 6.4 "Troubleshooting Labview Data Acquisition Problems".

5.6.12 Close the Windows File Manager

NOTE: CLOSING THE WINDOWS FILE MANAGER IS A CRITICAL STEP - IF THE FILE MANAGER IS LEFT OPEN THERE COULD BE FILE TRANSFER PROBLEMS DUE TO FILE SHARING ISSUES

5.6.13 Verify the Command Scheduler Icon is present and minimized on the Windows desktop. If not start the Command Scheduler per procedure step 5.1.4

6.0 Troubleshooting

The following procedure steps present tips for troubleshooting problems that might be encountered in the field. As a general rule if problem is not listed in the trouble shooting section refer to the following individuals.

Network Communications	System Administrator Daren Tate
Computer Problems	System Administrator Daren Tate
Mechanical/Electrical	Lead Design Engineer Tom Schneider
Analytical Instruments	Analytical Instrument Lead Tom Schneider
Data Acquisition	Data Acquisition Lead Daren Tate
Software	Software Lead Daren Tate

Copies of the pertinent manuals for the FTIR, GC, and Labview Products. are archived in VI-22665.

6.1 Troubleshooting Network Communications

6.1.1 Verify all network cables are securely connected between various computer systems. Re-secure if necessary.

6.1.2 Verify the appropriate shares and connections are made per this procedure.

6.1.3 If network communications can not be established please contact the System Administrator for further guidance.

6.2 Troubleshooting FTIR Operation

6.2.1 Verify that the FTIR Bench is powered on.

6.2.2 Verify that the interface cable connecting the FTIR Bench with the FTIR computer is secure on both ends.

6.2.3 If problem persists contact the Analytical Instrument Lead for further guidance.

6.3 Troubleshooting MTI GC Operation.

6.3.1 Verify the MTI GC is powered on.

6.3.2 Verify the serial cable connecting the MTI GC with the GC computer is secure.

6.3.3 In EZCHROM verify the detectors are on

6.3.4 In EZCHROM verify that data is saved to c:\GC1\GC1 or c:\GC2\GC2

6.3.5 In EZCHROM verify that the DIF SAVE dialogue box is checked and that EXTENDED dialogue box in NOT checked.

6.3.6 If problem persists, contact the Analytical Instrument Lead.

6.4 Troubleshooting Labview Data Acquisition Problems

6.4.1 Verify that the interface cable between the computer and data acquisition modules is properly seated.

6.4.1 Verify that the Save to File radio button in the GCSVI application has been activated.

6.4.2 Verify that the Start radio button on the GCSVI application has been depressed.

6.4.3 If problem persists contact the Data Acquisition Lead.

APPENDIX C -- C++ SOURCE CODE LISTINGS for GC1 ZIP Routines

C.1 GC1_ZIP.INI

```
path = c:\bin\pkzip.exe
zippath = d:\gc1\
out_path = c:\gc1\
nth_c = 5

/*********************************************/
Parameter Definintions

path:      The path for the pkzip executable

zippath:   The path for the files to be zipped

out_path: The path for the zip file

nth_c:     The nth number of chromatograms to be saved
    (i.e. - 5 means every 5th c-gram is saved)
```

C.2 GC1ZIP.CPP

```
/*********************************************************************
// Gc1zip.cpp
// Russell Katz & Cuong Vo, June 1995
// This program uses PKZip(TM) to archive GC data and prepare it for
// being burned onto a CD.
// Version Beta 0.50
// Cuong Vo
// Turned over, August 1995
// Last Mod, Feb 1996
// Version Beta 1.1
//
*********************************************************************/


// System includes
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fstream.h>
#include <dos.h>
#include <iostream.h>

// User Includes
#include "gc1zip.h"          // This file includes function prototypes for zip's
                             // and compress modules.

void main(void)
{
  char err_file[30] = "D:\\GC1_ZIP.ERR";              // The zip error log file

  // Ini file pointer and title.            // The ini and error file
  fstream ini, err;                                   // pointers

  err.open(err_file, ios::app);              // Open the error file for
                                              // appending

  err << "\n";                            // Add empty line to error file
  char ini_file[30] = "c:\\bin\\GC1_ZIP.INI";  // The GC1_ZIP.INI ini file

    // Ini file input columns
    char col1[10]="\0";
    char col2[3]="\0";
    char col3[25]="\0";

    // program constants initialized from an ini file
    char exe_path[45]="\0";       // The pkzip.exe path
    char zip_path[45]="\0";       // The area defined for the files to be zipped


    // File extension strings
    char ext_star[5] = "*";          // File extension wildcard
    char ext_zip[5] = "ZIP";   // Zip file extension
    char ext_rpt[5] = "RPT";  // Report file extension
    char period[5] = ".";       // Period character
    char backs[5] = "\\";       // The "\" character
    char call[6] = "call";    // "Call" for calling pkzip.exe

    // File command strings
    char del[50] = "del";
```

```
// Misc. strings
char space[3] = " ";        // Space character
char out_path[100]="\0";   // The output path for the zip file
char nth_c_str[5]="\0";

// Date strings
char date_name[35]="\0";    // Holds file name from fil_name: "50616GC1."
char day_dir[35]="\0";          // Holds day's directory name: "50616GC1"
char day[5]="\0";               // Transfer variable to hold day digits of
                            // date_name "16"

char month[4]="\0";             // Transfer variable to hold month digits
char year[3]="\0";          // Transfer variable to hold year digit
char day_string[15]="\0";     // Contains day string alone
char del_dir[50]="\0";      // Contains directory path being zipped for
                            // removal

// Temp strings
char temp_path[100]="\0";       // Contains entire system call to zip files.
char temp_zip[50]="\0";         // Contains zip file path & name
char temp_zipped[75]="\0";          // Contains files to be zipped path & names
char temp_del[50]="\0";         // Contains zip file path & name to delete
                            // after file copy.
char temp_rpt[50]="\0";         // Report file char variable
char temp_cgram[50]="\0";       // C-gram file char variable
char temp[5]="\0";          // Temporary char variable
char out_rpt[30]="\0";              // Output report variable
char out_dir_temp[10]="D:\\"; // Temporary place for zip file befor transfer
int int_year;                   // Year digit
int int_month;                      // Month digits
int int_day;                    // Day digit to be decremented
int nth_c_int;

WIN32_FIND_DATA fileData;     // Find File Data Structure

HANDLE findFile;              // Find File Handle

// This reads the contents of the ini file
// and places it in the correct variables
ini.open(ini_file,ios::in);

ini >> col1 >> col2 >> col3;
strcpy(exe_path, col3);

ini >> col1 >> col2 >> col3;
strcpy(zip_path, col3);

ini >> col1 >> col2 >> col3;
strcpy(out_path, col3);

ini >> col1 >> col2 >> col3;
strcpy(nth_c_str, col3);

ini.close();          // closes ini file

nth_c_int = atoi(nth_c_str);

fil_name(date_name, 1); // File date_name with the date (i.e. - 50803GC!.)

day[0] = date_name[3];
day[1] = date_name[4]; // Set char day equal to the day of the month
                        // (i.e. - in May2595. , day = 25
```

```
int_day = atoi(day);   // convert the day string to an int.

if(int_day == 1)
{
  year[0] = date_name[0];
  month[0] = date_name[1];
  month[1] = date_name[2];

  int_year = atoi(year);
  int_month = atoi(month);

  switch(int_month)
  {
     case 3:
        date_name[1] = '0';
        date_name[2] = '2';
        date_name[3] = '2';
        date_name[4] = '8';

        if(int_year == 6 || int_year == 0)
        {
           date_name[1] = '0';
           date_name[2] = '2';
           date_name[3] = '2';
           date_name[4] = '9';
        }

        break;

     case 5:
        date_name[1] = '0';
        date_name[2] = '4';
        date_name[3] = '3';
        date_name[4] = '0';

        break;

     case 7:
        date_name[1] = '0';
        date_name[2] = '6';
        date_name[3] = '3';
        date_name[4] = '0';

        break;

     case 10:
        date_name[1] = '0';
        date_name[2] = '9';
        date_name[3] = '3';
        date_name[4] = '0';

        break;

     case 12:
        date_name[1] = '1';
        date_name[2] = '1';
        date_name[3] = '3';
        date_name[4] = '0';

        break;

     default:
        int_month --;
```

```
        if (int_month < 10)
        {
          if(int_month == 0)          // If it is january(1 -- is zero)
          {
            int_year --;
            itoa(int_year, temp, 10);
            date_name[0] = temp[0];
            date_name[1] = '1';
            date_name[2] = '2';
            date_name[3] = '3';
            date_name[4] = '1';
          }

          else
          {
            itoa(int_month, temp, 10);
            month[0]='0';           // If int_month is less than ten,
            month[1]=temp[0];         // convert it to char and place
                                    // a '0' in front of it.

            date_name[1] = month[0];
            date_name[2] = month[1];
            date_name[3] = '3';
            date_name[4] = '1';
          }

        }                           // Ends if loop

        else
        {
          itoa(int_month, month, 10);         // Convert int_month to char
          date_name[1] = month[0];
          date_name[2] = month[1];
          date_name[3] = '3';
          date_name[4] = '1';
        }                           // Ends else

        break;

      }                             // End of month switch

    }                               // End of 1st of month if

    else
    {
      int_day--;              // decrement the day integer by one.
      if (int_day < 10)
      {
        itoa(int_day,temp,10);
        day[0]='0';           // If int_day is less than ten,
        day[1]=temp[0];         // convert it to char and place
      } // Ends if loop        // a '0' in front of it.

      else
      {
        itoa(int_day,day,10);     // Convert int_day to char
      }                         // Ends else

      date_name[3] = day[0]; // put the decremented day string back into
      date_name[4] = day[1]; // date_name.  (date_name was: May2595.
                        //                 date_name now is: May2495.)
    }
```

```
strncpy(day_dir, date_name, 8);  // strip the period off of date_name and
                                 // place it in day_dir.

day_dir[8] = 0;                  // Null terminator for day_dir


strcpy(day_string, day_dir);

strcat(day_dir, backs);          // Concatenate "\" onto day_dir
                                 // day_dir = "50616GC1\"

strcpy(temp_path, call);         // Copy call onto temp_path
strcat(temp_path, space);        // Concatenate space onto temp_path
strcat(temp_path, exe_path);     // Concatenate exe_path onto temp_path
                                 // Temp_path = "call c:\bin\pkzip.exe"

strcat(temp_path, space);        // Concatenate a space onto temp_path
                                 // temp_path = "c:\bin\pkzip.exe "

strcpy(out_rpt, out_path);       // out_rpt = "g:\gc1\"
strcpy(temp_rpt, zip_path);
strcpy(temp_cgram, zip_path);
strcpy(temp_zip, out_dir_temp);  // Concatenate zip_path onto temp location
strcpy(temp_zipped, zip_path);   // Concatenate zip_path onto temp_zipped
strcpy(del_dir, zip_path);


strcat(temp_rpt, day_dir);
strcat(temp_cgram, day_dir);
strcat(temp_zipped, day_dir);    // Concatenate day_dir onto temp_zipped

strcat(del_dir, day_string);     // Copy temp_zipped into del_dir
                                 // del_dir = x:\50616GC1

strcat(out_rpt, day_string);     // out_rpt = "g:\GC1\50621GC1"
strcat(temp_rpt, day_string);    // temp_rpt = "D:\GC1\50621GC1\50621GC1"
strcat(temp_cgram, day_string);  // temp_cgram = "D:\GC1\50621GC1\50621GC1"

strcat(out_rpt, period);         // out_rpt = "g:\GC1\50621GC1."
strcat(temp_rpt, period);        // temp_rpt = "D:\GC1\50621GC1\50621GC1."
strcat(temp_cgram, period);      // temp_cgram = "D:\GC1\50621GC1\50621GC1."

strcat(out_rpt, ext_rpt);        // out_rpt = "g:\GC1\50621GC1.RPT"
strcat(temp_rpt, ext_rpt);       // temp_rpt = "D:\GC1\50621GC1\50621GC1.RPT"

strcpy(temp_del, temp_zipped);   // Copy temp_zipped into temp_del

strcat(temp_zip, date_name);     // Concatenate date_name onto temp_zip
strcat(temp_del, date_name);     // Concatenate date_name onto temp_del
strcat(temp_zipped, ext_star);   // Concatenate date_name onto temp_zipped

strcat(temp_zipped, period);     // Concatenate period onto temp_zipped

strcat(temp_zip, ext_zip);       // Concatenate ext_zip onto temp_zip
strcat(temp_zipped, ext_star);   // Concatenate ext_star onto temp_zipped
strcat(temp_del, ext_star);      // Concateneate ext_star onto temp_del

strcat(temp_path, temp_zip);     // Concatenate temp_zip onto temp_path

strcat(temp_path, space);        // Concatenate space onto temp_path

strcat(temp_path, temp_zipped);  // Concatenate temp_zipped onto temp_path
```

```
    compress(temp_rpt, temp_cgram, nth_c_int);        // Data compression algorithm

    printf("copy from %s to %s\n",temp_rpt,out_rpt);
    CopyFile(temp_rpt, out_rpt, FALSE);                          // Copy rpt file over to
    sleep(2);                                  // output server

    printf("command %s\n",temp_path);
    system(temp_path);                                   // Run pkzip on May2595.*
    sleep(2);
      //temp_zip = "d:\50616GC1.zip"
      //temp_zipped = "x:\gc1\50616GC1\50616GC1.*"
      //temp_path = "c:\bin\pkzip d:\50616GC1.zip
      //                                 x:\<dir>\50616GC1\50616GC1.*"
      //temp_del = "x:\<dir>\50616GC1\50616GC1.*"
      // where x:\<dir>\ are paths from the ini file

    strcat(out_path, date_name);      // Concatenate date_name onto out_path
    strcat(out_path, ext_zip);        // Concatenate ext_zip onto out_path
                                  // out_path = "x:\50616GC1.zip"
                                  // where x:\ is out_path from ini file



    printf("move from %s to %s\n",temp_zip, out_path);
    MoveFile(temp_zip, out_path);                  // Move zip file to
    sleep(2);                                // output_path

    findFile = FindFirstFile(out_path, &fileData);// Try to find temp_zip file

    err << "MoveFile ERROR: " << GetLastError() << "\n";// Print error code to log file

    if(findFile = INVALID_HANDLE_VALUE)
//  if(GetLastError() == 0 || GetLastError() == 6) // Not use if no error or
                                                   // bad parameter (which
    {                                              // we always seem to get)
      FindClose(findFile);                // close FindFile handle

      strcat(del, space);             // Concatenate space onto del
      strcat(del, temp_del);          // Concatenate temp_zipped on del
      printf("delete %s\n",del);
      system(del);                    // system call to delete 50616GC1.*


      sleep(2);                                // Sleep 5 sec to allow file
                                               // deletion to be updated by
                                               // file system.
      printf("Remove directory %s\n",del_dir);
      RemoveDirectory(del_dir);                        // remove the day directory

      sleep(2);                                // sleep 5 sec to allow
                                               // window read.
    }

    else          // If a GetLastError() not equal to 0 or 6...
    {
        FindClose(findFile);                   // Close FindFirstFile handle
        err << "Data to zip not found...leaving data for check\n";
    }
  err.close();
  sleep(5);
} //End of zip module
```

C.3 GC_FIL.CPP

```
// GC_FIL.CPP, based upon TDC.CPP by Cliff Hewitt(LANL)


//System include files
#include <iostream.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>

void fil_name(char *sdat, int gc)
{

  //local variables
  char *tms;
  char str_gc[5];
  int t;

  time_t tm;
  size_t len;

  // Get Time String
  time(&tm);
  tms=ctime(&tm);
  len=strlen(tms);
  tms[len-1]=0;

  //sum char in month
  t=tms[4]+tms[5]+tms[6];

  //switch on month sum and create temp date string
  //month day year
  switch(t){
    case 281: sdat[1]=48; sdat[2]=49; break; //Jan
    case 269: sdat[1]=48; sdat[2]=50; break; //Feb
    case 288: sdat[1]=48; sdat[2]=51; break; //Mar
    case 291: sdat[1]=48; sdat[2]=52; break; //Apr
    case 295: sdat[1]=48; sdat[2]=53; break; //May
    case 301: sdat[1]=48; sdat[2]=54; break; //Jun
    case 299: sdat[1]=48; sdat[2]=55; break; //Jul
    case 285: sdat[1]=48; sdat[2]=56; break; //Aug
    case 296: sdat[1]=48; sdat[2]=57; break; //Sep
    case 294: sdat[1]=49; sdat[2]=48; break; //Oct
    case 307: sdat[1]=49; sdat[2]=49; break; //Nov
    case 268: sdat[1]=49; sdat[2]=50; break; //Dec
    default: cout<<"ERROR IN SWITCH TDC.CCP\n";
    }


  itoa(gc, str_gc, 10);

  sdat[0] = tms[23];
  sdat[3] = tms[8];
  sdat[4] = tms[9];
  sdat[5] = 71;//
  sdat[6] = 67;//      Will be replacing these with data from
  sdat[7] = str_gc[0];//     *.ini file.
  sdat[8] = 46;//
  sdat[9] = 0;

  return;
}
```

C.4 EXP2.CPP

```
// Shawn R. Harden  July 14, 1995
// This program reads low and high concentrations from an rpt file,
// saves every Nth C-gram, compares the values read to a set value,
// and checks for calibration. It then deletes the corresponding
// C-gram depending on those comparisons.
// Cuong Vo, August 1995
// Last mod and turned over
//

#include <iostream.h>
#include <fstream.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <math.h>
#include <dos.h>

#include "function.h"

struct fileinfo
 {
    char col1[2],col2[6],col3[6],col4[6],col5[9],col6[7],
       col7[9],col8[9] /* ,col9[9],col10[9] */ ;

 } file[999]; // Declares 999 input structures.

void compress(char * filename, char crname[50], int N)
{
//   cout << "Filename: " << filename << "\n";
//   cout << "Crname: " << crname << "\n";
//   cout << "N: " << N << "\n";

    char index_ext[5];
    int max,count,n,i;
    float calibr[999];
    float lhydro[999];
//  float lnitro[999];
    float hhydro[999];
//  float hnitro[999];
    char base[21];

    int int_index;
    max = 999;
    count = 1;
    n = 0;


    ifstream infile(filename);

// Skips header.
       infile >> file[count].col1 >> file[count].col2 >> file[count].col3
              >> file[count].col4 >> file[count].col5 >> file[count].col6
              >> file[count].col7 >> file[count].col8 /* >> file[count].col9
              >> file[count].col10 */ ;

// Reads in the file by fields into an array.
   while (n <= max)
   {

       infile >> file[count].col1 >> file[count].col2 >> file[count].col3
              >> file[count].col4 >> file[count].col5 >> file[count].col6
```

```
            >> file[count].col7 >> file[count].col8 /* >> file[count].col9
            >> file[count].col10 */ ;

// Changes concentration values from character to float.
      calibr[count] = atof (file[count].col1);
      lhydro[count] = atof (file[count].col7);
      hhydro[count] = atof (file[count].col8);
//    lnitro[count] = atof (file[count].col9);
//    hnitro[count] = atof (file[count].col10);

// Checks for the end of file (EOF).

    if(infile.eof())
    {

      n = 1010;
    }

//  cout << "Reading line #" << count << "\n";
    count++;

  }// Ends the while statement.


  max = count - 1;
  strcpy(base,crname);
  for (i = 1; i <= max; i++)
  {
  // Append the index to the name of the c-gram.

      int_index = atoi(file[i].col6);  // convert to int
      index_str(int_index, index_ext); // make index_ext 3 char ext.
      strcat(crname, index_ext);          // concatenate on end of crname

      if ((i % N == 0) || lhydro[i] == 0.000 || hhydro[i] == 0.000
//                  || lnitro[i] == 0.000 || hnitro[i] == 0.000
                  || calibr[i] == 1)
      {
         cout << crname << " chromatogram was saved" << "\n";
      }
      else
      {
         cout << crname << " chromatogram was deleted" << "\n";
         remove(crname);
      }
       crname="\0";
      strcpy(crname,base); //Reset the characters in crname.
  }

// Close the input file.
  ifstream::close;
  cout << "End of Compress Section\n";
  sleep(5);
  return;
}
```

C.5 INDEX3.CPP

```
// This program creates a time stamp based on today's date & time

#include <stdlib.h>
#include <string.h>

void index_str(int int_index, char *ch_index)
{
  int hundreds;
  int tens;
  int ones;

  char hund[2];
  char tns[2];
  char ouns[2];
  char temp[20];

  hundreds = (int_index - ((int_index/1000)*1000))/100;
  tens = ((int_index - ((int_index/1000)*1000))- (hundreds * 100))/10;
  ones = (((int_index - ((int_index/1000)*1000))- (hundreds * 100)) - (tens * 10));

  itoa(hundreds, hund, 10);
  itoa(tens, tns, 10);
  itoa(ones, ouns, 10);

  strcpy(temp, hund);
  strcat(temp, tns);
  strcat(temp, ouns);
  strcpy(ch_index, temp);


  return;
}
```

C.6 GC1ZIP.H

```
// Defines of user-created functions

fil_name(char *, int);
compress(char *, char *, int);
```

C.7 FUNCTION.H

```
index_str(int, char *);
```

APPENDIX D -- C++ SOURCE CODE LISTINGS for GC1APP32

D.1 GC1.INI

```
backs = \
root = d:\gc1\
C_gram_old = gc1.
gcPath = F:\gc1.dif
outputPath = d:\gc1\
outputExt = rpt
gc_del = gc1.*
grampath = d:\gc1\
filepath = F:\
calPath = c:\gc1cal
```

****

Ini file descriptions

backs: Character for backslash on file paths

root: Report and chromatogram output file root path

C_gram_old: Chromatogram GC run identifier

gcPath: Dif file created by EZChrom path

outputPath: Report and chromatogram output file root path

outputExt: Report output file extension

gc_del: Wildcard definition of files to delete in input directory

grampath: Chromatogram file output path

filepath: Report and chromatogram input file path to watch for file changes

calPath: File to watch for for calibration events

D.2 GC1APP32.DEF

```
NAME            GC1APP32
DESCRIPTION     'GC Data Acquisition Software'
EXETYPE         WINDOWS

CODE      PRELOAD MOVEABLE DISCARDABLE
DATA      PRELOAD MOVEABLE MULTIPLE

HEAPSIZE     8192
STACKSIZE 8192
```

D.3 GC1APP32.CPP

```
/*****************************************************************************/
// Program GC1APP32.CPP                                    //
// Version Beta .68                                   //
/*****************************************************************************/

// May 18, 1995
//(0.51)
// Changed index char variable in appnd.cpp, gc1_trns.cpp, and gc1_indx.cpp
// to 4 characters long.
//(0.52)
// gc1 trans bug fixed by converting index to an integer before reaching end
// of file.
// (0.53)
// two variables are compared in gc1_trans (index0ut and index10ut)
// if either one is zero the other variable is returned
// (0.54)
// Path variables are loaded from gc1.ini file.
// an error of incorrectly skipping throught the while loop was detected
// and corrected as of 5/19/95. The problem was that the variable used in
// the for loop and the variable used in the while loop were the same.
// As a result the while loop was being ignored.
// (0.55)
// Corrected error from use of the gc1.ini file. Extra trigger resets
// for the dif file transfer were removed, since the directory where the
// dif and old c-grams vs. rpt file and new c-grams are now different.
// (0.56)
// Resized the executable window, buttons, and text box.
// (0.57)
// Added error log file. Error checking routines for file trip from something
// else other that *.dif and c-gram files, and for loss of communications
// with the gc computer.
// (0.58)
// Error in program due to inadvertent tripping of flag due to movefile
// command which renames the file and while doing this deletes the older
// version of the chromatogram. A reset has been put after the deletefile
// and the movefile commands. A module was added for the three character
// extensions for the chromatograms, a new getlasterror routine outputs
// the errors in text format.
// System command used for deleting gc1.* files upon startup of the program.
// (0.59)
// The error log file was expanded to include the GetLastError call
// results. A memory error occured due to the size of the char array known
// as time stamp. The array size was increased and the error was then
// corrected.
// (0.60)
// An automatic paging routine was added to the network timeout code.
// In the case of a network failure, gcs1host will page a specified pager.
// (0.61)
// The dos window appearing when calling system("del ...."), was removed.
// The WinExec command was used instead. Due to this change, the dif file
// appearance flag was changed to FILE_NOTIFY_CHANGE_LAST_WRITE, and
// a reset was eliminated after DeleteFile
// (0.62)
// Added gc_del to *.ini file.
// (0.63): June 1, 1995
// Complete commenting
// (0.64): June 8, 1995
// New version screens out first *.dif file from GC. Project file is
// updated to remove some Borland Warnings. Calibration column was added
// to the *.rpt file. The column contains a 1 when the gc is calibrating
// and a 0 when it is not. The grapical interface was modified also.
```

```
// (0.65): June 16, 1995
// Initial index transfer when the program began was updated to deal with
// the calibration column.  The output to the report file from an extra file
// trip was commented out, along with the file change output statement being
// moved farther down to only appear when the *.rpt file will be appended to.

// Another modification due to extra file trips was to delete a second
// "Beginning wait for file change" statement by deleting the line from the
// dialog box's text box.

// The C++ project was setup to remove the "Initialize code in header file"
// Borland warning.  This was done by placing all the included *.cpp files
// previous to the gc1app32.cpp file in the project, and creating a new
// gc1app32.h file to contain all the function prototypes for those modules.

// The updated program was also coded to screen out the first *.dif file of
// a run from the GC, since this file usually has a bad header that disrupts
// data handling.  If this file comes accross, it is deleted.

// The last change made was to make the file naming convention similar to
// that used for the FTIR.  GC1_fil_ was renamed GC_FIL, and its function
// changed to pass two parameters.  One parameter was the string to be
// filled with the file name, and the other as a gc identifier (1 or 2).
// The new file names are as follows: 50616GC1.<ext>.
//                  Year ----------^||^^---- Day
//                                  ||
//                                          Month

// (0.66): June 19, 1995
// Problems with the calibration-catching routine were fixed.  Switched the
// if and else loops for calibration and no dif file to have failure
// dealt with first.

// (0.67): June 20, 1995
// The previous effort was not successful.  An addition of the SetLastError
// function to set the error code to zero at the beginning of the infinite
// loop fixed the problem of the dif file not being found, and the problem
// of calibration not operating correctly.

// (0.68): July 11, 1995
// Addition of dif_fix function to strip out the header created by EZChrom
// on the first run of the GC.  First run data is no longer deleted.

// (0.69): July 12, 1995
// The program and modules were modified to accept 2 columns of data input
// from EZChrom versus the earlier 4.  GC1APP32 and GC2APP32 are now
// separate programs.

// (0.70): Jan 17, 1996
// Modified gc1app32.cpp to provide proper exit point when encounted network
// failure.

// (0.71); April 8, 1996
// Modified gc1app32.cpp to correct date change problem.

// Includes and Definitions
#define STRICT          // This is needed for the method of construction used for the dialog box.
#include <windows.h>    // Windows definitions include file.
#include <windowsx.h>   // Additional windows definitions include file
#include <stdio.h>      // Standard input and output for C++ include file.
#include <iostream.h>   // Input/Output streams include file.
#include <stdlib.h>     // Standard function library include file.
#include <fstream.h>    // File input/output stream include file.
```

```
#include <dos.h>          // Dos function include file.#include <windows.h>

// User Includes and Definitions
#include "Resource.H"   // ID specifications for dialog controls (Buttons, etc.)
#include "gc1app32.h"  // Function prototypes for gcapp modules.

#define header __TEXT("c:\\bin\\header1.h")       // file containing Header
                                                  // for *.rpt file.

// Ini and error file streams and titles
fstream ini, chk;                    // file streams for ini and error files
char ini_file[30]= "gc1.ini";         // Ini file name string
char col1[16],col2[3],col3[17];            // Input columns from ini file
char err_chk[30]="error.log";               // Error file name string
char gc_id_str[3];


// Temp character variables
char temp1[50];
char temp2[50];
char temp3[50];
char temp4[50];
char temp5[50];
char temp20[50];
char temp21[50];
char temp22[50];
char temp_command[70] = "command.com /c ";


// Global program constants initialized from an ini file
char backs[5];              // Contains "/" character
char root[10];              // Will contain root directory where output will go.
char C_gram_old[15];           // Contains EZChrom run ID "gc1."
char gcPath[20];            // Contains the path of the dif file "c:\gc1\gc1.dif"
char outputPath[50];           // Also contains output path for data
char outputExt[10];         // Contains the extension for the output file "rpt"
char gc_del[20];            // Contains the string to delete at the start of the
                       // program "gc1.*"


// Error codes...
char err_temp[50];
char err_temp2[50] = "WaitForSingleObject timed out...";
char err_temp4[40];
char error1[50];
char error2[50];
char win_exec_error_msg[80];

// Other Character variables
char dirName[50];
char Cgrampath[50];
char old_date[50];
char Path[50];
char new_date[50];
char del_path[20];
char del[20] = "del ";
char time_stamp[30];
char time_stamp1[30];

int reportIndex, difIndex;    // Index integers from *.rpt file and *.dif file
int gc_id;                    // Integer value taken from file name string
                       // to identify gc program is running with.
```

```
// WORD, DWORD, and Windows variables
DWORD win_exec_error;                   // WinExec error code container
WIN32_FIND_DATA fileData, calData;   // FindFirstFile file data container structures
LPSTR *FilePart, *Cpart, *Cnpart;   // SearchPath path container variables.
DWORD WINAPI NotifyThread 0;        // NotifyThread prototype

// Booleans
BOOL fFailIfExists = TRUE;                // CopyFile command parameter to have the
                                   // function fail if the file already exists
                                   // where it is trying to be copied to.

HWND g_hwndLB = NULL;                     // Global handle to dialog box's text output
                                   // box.

void AddStr (LPCTSTR szFmt, ...)   // AddString function from Advanced Windows
{                                  // By Jeffrey Richter.  It outputs
  char szBuf[150];                 // to the dialog text box, and takes input
  int nIndex;                            // essentially like printf0;
  va_list va_params;

  //Make va_params point to the first argument after szFmt
  va_start(va_params, szFmt);

  //Build the string to be displayed
  vsprintf(szBuf, szFmt, va_params);

  do
  {
    //Add the string to the end of the list box
    nIndex = ListBox_AddString(g_hwndLB, szBuf);

    //If the list box is full, delete the first item in it
    if(nIndex == LB_ERR)
    {
      ListBox_DeleteString(g_hwndLB, 0);
    }
  } while (nIndex == LB_ERR);

  //Select the newly added item
  ListBox_SetCurSel(g_hwndLB, nIndex);

  //Indicate that we're done referencing the variable arguments
  va_end(va_params);
}

void printError0                       // Print error function to output text
{                                  // from Windows GetLastError0 function.
  char *ErrorBuffer;               // Outputs to dialog text box.
  #define PERROR(X) printf(X);
  FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM,
          NULL, GetLastError0, MAKELANGID(LANG_ENGLISH, SUBLANG_ENGLISH_US),
          (LPTSTR)&ErrorBuffer, 0, NULL);  // Format message from system
                                           // GetLastError message into
                                           // english text and place in
                                           // ErrorBuffer variable
  AddStr(__TEXT("Error: %s"), ErrorBuffer);              // Print out Error Buffer to
                                           // dialog text box.
}

void runMain(void)          // Main run function
{
  // Main char variables
  char reportName[30];        // Will contain name of report file
```

```
// Main Booleans
BOOL copyMade = FALSE;  // Boolean indicating success of CopyFile function

// Main Handles
HANDLE hThread;         // Handle for NotifyThread executed in main program

// Main DWORDS
DWORD dwthreadID;       // System ID of NotifyThread.
DWORD cPath = 20;       // SearchPath parameter specifiying size of container
                        // variable the path is placed in.


// This reads the contents of the ini file
// and places it in the correct variables
    ini.open(ini_file,ios::in);         // open ini file for reading
    ini>> col1>>col2>>col3;
    strcpy(backs, col3);            // Copy from ini file into backs

    ini>> col1>>col2>>col3;
    strcpy(root, col3);             // Copy from ini file into root

    ini>> col1>>col2>>col3;
    strcpy(C_gram_old, col3);       // Copy from ini file into C_gram_old

    ini>> col1>>col2>>col3;
    strcpy(gcPath, col3);           // Copy from ini file into gcPath

    ini>> col1>>col2>>col3;
    strcpy(outputPath, col3);       // Copy from ini file into outputPath

    ini>> col1>>col2>>col3;
    strcpy(outputExt, col3);        // Copy from ini file into outputExt

    ini>> col1>>col2>>col3;
    strcpy(gc_del, col3);           // Copy from ini file into gc_del

    ini.close();            // close ini file

    gc_id_str[0] = ini_file[2];
    gc_id = atoi(gc_id_str);        // Set the gc_id variable equal to the
                                    // character in the ini file definition.
                                    // if gc1.ini, then gc_id = 1

    fil_name(dirName, gc_id);               // Fill dirName with date (May3095.)
                                    // (Developed function, gc1_fil_.cpp)

    strcpy(temp3, root);            // Copy root into temp3 (temp3 = g:\gc1\)

    strncpy(temp2, dirName, 8);     // Copy first 7 char of dirName into temp2
                                    // temp2 = "May3095"

    strcat(temp3, temp2);           // Concatenate temp2 onto temp3
                                    // Temp3 = "g:\gc1\May3095"

    CreateDirectory(temp3, NULL); // Create directory using temp3
                                    // (WIN32)

    strcat(temp3, backs);           // Concatenate backs onto temp3
                                    // Temp3 = "g:\gc1\May3095\"

    strcpy(outputPath, temp3);      // Copy temp3 into outputPath

    fil_name(reportName, gc_id);            // Fill reportName with date (May3095.)
```

```
                                // (Developed function, gc1_fil_.cpp)

    strcat(reportName, outputExt);// Concatenate outputExt onto reportName
                                // reportName = "May3095.rpt"

    strcat(temp3, reportName);   // Concatenate reportName onto temp3
                                // Temp3 = "g:\gc1\May3095\May3095.rpt"

    copyMade = CopyFile(header, temp3, fFailIfExists);
                                // Copy file from header (g:\code\header.h)
                                // into temp3 file name.
                                // (WIN32)


    if(copyMade == TRUE)        // If the copy succeeds.
    {
     AddStr(_TEXT("No file for today, %s is created"), outputPath);
     reportIndex = 0;           // Set reportIndex = 0
     AddStr(_TEXT("The first index in today's file is %d"), reportIndex);
    }

    else                        // If the copy failed
    {
       AddStr(_TEXT("Today's output file exists"));
       reportIndex = rpt_trns(temp3);          // Get the last index from the rpt file.
       AddStr(_TEXT("The transferred index is %d"), reportIndex);
    }


/*********************************************************************/
//Find gc1.dif to see if it currently exists
//If it does, delete it and all other gc1.* files
/*********************************************************************/

    SearchPath(NULL, gcPath, NULL, cPath, Path, FilePart);
                                    // Search for *.dif file

    if(GetLastError() == ERROR_FILE_NOT_FOUND)  // If no *.dif file...
    {
       AddStr(_TEXT("The file was not found"),0);
       AddStr(_TEXT("The *.dif and chromatogram files could not be deleted"),0);
    }

    else                        // If the *.dif file exists...
    {
       strncpy(del_path, gcPath, 3);    // Copy first 3 char of gcPath in to del_path
       strcpy(temp20, del);             // Copy del into temp20
       strcat(temp20, del_path);  // Concatenate del_path into temp20
                                // temp20 = del F:\
       strcpy(temp21, temp20);   // Copy temp20 into temp21
       strcat(temp21, gc_del);   // Concatenate gc_del into temp21
                                // temp21 = del F:\gc1.*
       strcpy(temp22, temp21);   // Copy temp21 into temp22
       strcat(temp_command, temp22);        // Concatenate temp22 into temp_command
                                // temp_command = command.com /c del F:\gc1.*

       win_exec_error = WinExec(temp_command, SW_HIDE);
                                // Execute temp_command string
                                // "command.com /c del F:\gc1.*"

       if(win_exec_error < 32)    // If there is an error in WinExec
       {
          sprintf(win_exec_error_msg, "WinExec failed; error code = %d", win_exec_error);
```

```
            AddStr(_TEXT("%s"), win_exec_error_msg);    // Output error
          }
          else
          {
            AddStr(_TEXT("The *.dif and chromatogram files were deleted"),0);
          }                              // Delete was successful
        }

        sleep(10);                       // Allow file deletion to complete before
                                         // beginning new thread.

/*****************************************************************************/
// Main program loop/thread
/*****************************************************************************/

        hThread = CreateThread ( 0, 0, (LPTHREAD_START_ROUTINE) NotifyThread,
                                                                    0, 0, &dwthreadID );
                                         // Create a thread called NotifyThread
                                         // and let it begin to execute.
                                         // This is based on code from the
                                         // Advanced Windows book: Pages 60-64
                                         // and Chapter 9(Thread Synchronization)
                                         // (WIN32)

        CloseHandle(hThread);            // Close the handle to the thread
                                         // (WIN32)

} // End of Main Program


BOOL DialogInitialize (HWND hwnd, HWND hwndFocus, LPARAM lParam)
{

  // Save the handle of the dialog's list box in a global
  // so that the threads can easily gain access to it.
  g_hwndLB = GetDlgItem(hwnd, IDC_LOG);


  // Associate an icon with the dialog box.
  SetClassLong(hwnd, GCL_HICON, (LONG)
    LoadIcon((HINSTANCE) GetWindowLong(hwnd, GWL_HINSTANCE),
    _TEXT("gc2app32")));
  return(TRUE);
}


void DialogCommand (HWND hwnd, int id, HWND hwndCtl, UINT codeNotify)
{

  switch (id)                            // Switch on the id of a pressed button
  {
    case IDOK:                           // If the RUN button is pushed
      ListBox_ResetContent(g_hwndLB);          // Reset the text window and
      runMain0;                          // Run the Main program
      break;

    case IDCANCEL:                       // If the QUIT button is pushed
      EndDialog(hwnd, id);               // Close the dialog box
      break;
  }
}
```

```
BOOL CALLBACK DialogProc (HWND hDlg, UINT uMsg, WPARAM wParam, LPARAM lParam)
{

  BOOL fProcessed = TRUE;                          // Set fProcessed to TRUE

  switch (uMsg) {                        // Switch on uMsg from the Windows system
    HANDLE_MSG(hDlg, WM_INITDIALOG, DialogInitialize);
                                         // If message is WM_INITDIALOG
                                         // run DialogInitialize function
    HANDLE_MSG(hDlg, WM_COMMAND, DialogCommand);
                                         // If message is WM_COMMAND
                                         // run DialogCommand function


    case WM_USER:                                // If message is WM_USER
      break;                             // break out.

    default:                               // If any other message
      fProcessed = FALSE;                        // reset fProcessed to FALSE
      break;                             // and continue executing
  }

  return(fProcessed);                            // Return fProcessed to function
                                         // that called DialogProc
}


int WINAPI WinMain (HINSTANCE hinstExe, HINSTANCE hinstPrev, LPSTR lpszCmdLine,
                                                                        int nCmdShow)

{

  DialogBox(hinstExe, MAKEINTRESOURCE(IDD_GC), NULL, DialogProc);
                                         // Create a Dialog Box with
                                         // dialog procedure function
                                         // DialogProc
  return(0);
}

DWORD WINAPI NotifyThread 0      // The Notify Thread started by main program
{
  // File Handles
  HANDLE hFile, findFile, findCal;

  // DWORDS and Booleans
  DWORD dw;                      // Will contain returned value of
                           // WaitForSingleObject.

  DWORD chPath = 20;             // SearchPath parameter specifing size of
                           // string to contain found path
                           // (filePath, line 644)

  BOOL notif = FALSE;            // Will contain returned value of
                           // FindFirstChangeNotification
  BOOL COPYGOOD = FALSE;               // Will contain returned value of CopyFile
                           // copying header into temp3 for new day's
                           // file name. (Line )
  BOOL Cal = FALSE;              // Calibration boolean.

  // Integers
```

```
int i = 1;                    // Constant value for infinite loop
int j = 0;                    // Counter for reading from ini file


// Character constants from ini file
char grampath[20];            // Path where c-grams will be placed.
char filePath[20];            // Path to search for dif and chromatogram files.
char calPath[50];             // Contains path to search for calibration
                              // flag file.

// Character Variables
char C_gram_new[50];          // Will contain variable string for new
                              // c-gram name. 50616GC1.003
char new_ext[50];             // New extension for c-gram  "003"
char old_ext[50];             // Old extension for c-gram  "256"
char reportName[50];          // Contains the *.rpt file name "50616GC1.rpt"
char zeros[7] = "0.000";      // String to compare for bogus dif file header

// Temporary Character Variables
char temp[50];
char temp6[50];
char temp7[50];
char ttemp[50];
char old_gram_temp[50];
char chrom_path_final[50];

ini.open(ini_file, ios::in);  // Open the ini file for reading

for (j = 1; j <= 8; j++)      // Skip the first seven lines processed
{                             // in the main program
  ini >> col1 >> col2 >> col3;
}

strcpy(grampath, col3);       // Read grampath from ini file
ini >> col1 >> col2 >> col3;
strcpy(filePath, col3);       // Read filePath from ini file
ini >> col1 >> col2 >> col3;
strcpy(calPath, col3);        // Read filePath from ini file
ini.close();                  // Close the ini file

AddStr(_TEXT("VIEW DIRECTORY %s"), filePath);

hFile = FindFirstChangeNotification(filePath, FALSE,
  FILE_NOTIFY_CHANGE_FILE_NAME || FILE_NOTIFY_CHANGE_LAST_WRITE);

                              // Initialize the handle for the
                              // FindFirstChangeNotification Object
                              // This flag tells us when any file's
                              // last write time has changed in the
                              // path specified in filePath
                              // This is based on code from the
                              // Advanced Windows book: Pages 663-666
                              // (WIN32)
  if (hFile == INVALID_HANDLE_VALUE)
  {
    AddStr(_TEXT("FindFirstChange problem"),0);
  }

fil_name(old_date, gc_id);    // Fill old_date with the date
                              // old_date = Jun0195.
                              // (Developed function, gc1_fil_.cpp)

while (i == 1)                // Infinite while loop for data collection
```

```
{

  SetLastError(0);            // Set Error code to zero
                            // to prevent carrythrough of
                            // any errors to other processes.


  /*****************************************************************/
  // Process the GC1.DIF file section
  /*****************************************************************/

  // Waiting for the gc1.dif file to be written....

  AddStr(_TEXT("Beginning Wait for File Change"),0);


  notif = FindNextChangeNotification(hFile); // Reset the waiting object flag

  if (notif == FALSE)
  {
    AddStr(_TEXT("FindNextChange problem"),0);
  }

  dw = WaitForSingleObject(hFile, 600000);    // Wait for 10 minutes for
                                              // flag to be triggered.
                                              // (Both functions WIN32)

  if (dw==258)                                          // If the wait times out...
  {
    strcpy(err_temp, outputPath);   // Copy outputPath into err_temp

    strcat(err_temp, err_chk);        // Concatenate err_chk onto
                                    // err_temp.
                                    // err_temp = d:\gc1\Jun0195\error.log

    chk.open(err_temp,ios::app);    // Open error.log file
    strcpy(error, err_temp2);       // Copy err_temp2 error code into
                                    // error.

    time_stp(time_stamp);             // Get the current time stamp.
                                    // (Developed function, time_stp.cpp)

    chk << time_stamp << "\t" << error << "\n";  // Output the timestamp and
                                                 // and the error to error.log

    chk.close();                  // Close error.log

    AddStr(_TEXT("Instrument Problem"),0);     // Inform the user
    AddStr(_TEXT("Program Terminated"),0);
//  system("page 85-7019 GC Problem on GCS1");        // Page gas group member
    break;
  } // End of timeout if loop

  sleep(10);                                            // Wait 10 seconds for the
                                              // c-gram and dif files to
                                              // complete writing.

  // Look here for whether calibration flag is up
  // if it is set Cal = TRUE
  findCal = FindFirstFile(calPath, &calData);   // Look for the calibration
                                              // file created by LabView
                                              // vi.

  if(GetLastError() == 2)
```

```
{
  Cal = FALSE;                                                    // Calibration is not occuring

  FindClose(findCal);                                                     // Close calibration FindFile
                                                                  // handle.
}


else
{
  Cal = TRUE;                        // Calibration is in progress

  FindClose(findCal);                                                     // Close calibration FindFile
                                                                  // handle.
}


findFile = FindFirstFile(gcPath, &fileData); // See if the dif file exists
                                             // as specified in gcPath
                                             // (WIN32)

// Network Problem check
if (findFile == INVALID_HANDLE_VALUE && GetLastError() != 2)
{
  AddStr(__TEXT("NetWork Problem"),0);          // Inform the user
  AddStr(__TEXT("Program Terminated"),0);
  FindClose(findFile);
//  system("page 85-7019 Network Problem on GCS1");          // Page gas group member
  break;
}

if(GetLastError() == 2)   // If the dif file is not found...
{
  ListBox_DeleteString(g_hwndLB, (ListBox_GetCount(g_hwndLB)-1));
                          // Delete the previous string from the
                          // dialog text box.  This would be a
                          // second line saying
                          // "Beginning Wait for file change..."
}

else    // Enters when the dif file does exist.
{
  AddStr(__TEXT("File Has Changed"),0);
  FindClose(findFile);                                  // Close the findFile handle
                                             // (WIN32)

  if(dif_chk(gcPath) == 0)          // If the dif file contains a
  {                                 // bogus header
     dif_fix(gcPath);               // Run the strip out header routine
                          // (Developed function, dif_fix.cpp)
  }


     difIndex = ind_trns(gcPath);                       // Get the current index from
                                             // the dif file and place it
                                             // in difIndex.
                                // (Developed function, gc1_trns.cpp)

     reportIndex++;                     // Increment reportIndex

     indx(gcPath, reportIndex);            // Replace the current index
                                             // in the dif file with
                                             // reportIndex.
```

```
                                // (Developed function, gc1_indx.cpp)

strcpy(temp7, temp3);                   // Copy temp3 into temp7

appnd(temp7, gcPath, Cal);                  // Append the line from the
                                            // dif file into the
                                            // rpt file.
                                // (Developed function, appnd.cpp)

AddStr(__TEXT("Finished Appending #%d"), reportIndex);

DeleteFile(gcPath);                 // When finished, delete the *.dif file
                                // (WIN32)

/*****************************************************************/
// Renaming of C-grams section
/*****************************************************************/

strcpy(C_gram_new, old_date);               // Copy old_date into C_gram_new

index_str(reportIndex, new_ext); // Convert reportIndex into the three-
                                // character extension for the new
                                // c-gram file.
                                // reportIndex = 3
                                // new_ext = "003"
                                // (Developed function, index3.cpp)

itoa(difIndex, old_ext, 10);    // Convert difIndex to a string for
                                // the old c-gram file.

strcat(C_gram_new,new_ext);     // Concatenate the new three-character
                                // extension onto C_gram_new
                                // C_gram_new = "Jun0195.003"

strcpy(old_gram_temp, C_gram_old);    // Copy C_gram_old into old_gram_temp
strcat(old_gram_temp,old_ext);  // Copy old_ext onto old_gram_temp
                                // old_gram_temp = "f:\gc1.256"


SearchPath(filePath, old_gram_temp, NULL, chPath, Cgrampath, Cpart);
                                // Get the full path for the old
                                // c-gram. (old_gram_temp)
                                // (WIN32)

strcpy(temp6, temp2);           // Copy temp2 into temp6
                                // temp6 = "Jun0195"

strcat(temp6, backs);           // Concatenate a backslash onto temp6
                                // temp6 = "Jun0195\"

strcpy(chrom_path_final, temp6);        // Copy temp6 into chrom_path_final
                                // chrom_path_final = "Jun0195\"

strcpy(ttemp, grampath);        // Copy grampath (from ini file)
                                // into ttemp.
                                // ttemp = "d:\gc1\"

strcat(ttemp, chrom_path_final); // Concatenate chrom_path_final
                                // onto ttemp.
                                // ttemp = "d:\gc1\Jun0195\"

strcpy(temp,ttemp);             // Copy ttemp into temp
                                // temp = "d:\gc1\Jun0195\"
```

```
strcat(temp, C_gram_new);       // Concatenate C_gram_new onto temp
                                // temp = "d:\gc1\Jun0195\Jun0195.003"

if (MoveFile(Cgrampath, temp) == 1)  // Move and rename c-gram from
{                               // F:\gc1.256 to
                                // d:\gc1\Jun0195\Jun0195.003"
                                // (WIN32)

}

else                            // if the MoveFile fails...
{
   AddStr(__TEXT("C-gram not generated"), 0);
}

/*****************************************************************/
// New Date Check Section
/*****************************************************************/

fil_name(new_date, gc_id);                  // Fill new_date with the current date
                                // (Developed function, gc1_fil_.cpp)

if(strncmp(old_date, new_date,5) != 0)      // If new_date and old_date
{                                           // are different...

   strcpy(old_date, new_date);          // assigned new_date to old_date
                                // when date change occurred

   fil_name(dirName, gc_id);                        // Fill dirName with the date
                                // (Developed function, gc1_fil_.cpp)

   strcpy(temp3, root);                 // Copy root into temp3
                                // temp3 = "d:\gc1\"

   strncpy(temp2, dirName, 7);          // Copy first 7 characters of
                                // dirName into temp2
                                // temp2 = "Jun0295"

   strcat(temp3, temp2);            // Concatenate temp2 onto temp3
                                // temp3 = "d:\gc1\Jun0295"

   CreateDirectory(temp3, NULL);    // Create a directory using
                                // temp3 string.
                                // (WIN32)

   strcat(temp3, backs);            // Concatenate a backslash onto
                                // temp3.
                                // temp3 = "d:\gc1\Jun0295\"

   strcpy(outputPath, temp3);       // Copy temp3 into outputPath
                                // outputPath = "d:\gc1\Jun0295\"

   fil_name(reportName, gc_id);         // Fill reportName with the new
                                // date.
                                // reportName = "Jun0295."
                                // (Developed function, gc1_fil_.cpp)

   strcat(reportName, outputExt);   // Concatenate outputExt onto
                                // reportName.
                                // reportName = "Jun0295.rpt"


   strcat(temp3, reportName);   // Concatenate reportName onto
```

```
                                // temp3.
                                // temp3 = "d:\gc1\Jun0295\Jun0295.rpt"

        COPYGOOD = CopyFile(header, temp3, fFailIfExists);
                                // Copy c:\bin\header.h file
                                // to create the new report file
                                // indicated by temp3.
                                // (WIN32)

        if(COPYGOOD == TRUE)        // If the copy succeeds...
        {
          ListBox_ResetContent(g_hwndLB); // Reset the text box.

          AddStr(__TEXT("No file for today, %s is created"), outputPath);
          AddStr(__TEXT("Copied header.h to today's file"), 0);

          reportIndex = 0;                          // Set reportIndex to zero for
                                        // the new day.

          AddStr(__TEXT("The first index in today's file is %d"), reportIndex);
        }

        else                        // If the copy fails because the file exists
        {
          ListBox_ResetContent(g_hwndLB); // Reset the text box

          AddStr(__TEXT("Today's output file exists"), 0);

          reportIndex = rpt_trns(outputPath);       // Get current index from the
                                                // existing report file.
                                        // (Developed function, gc1_trns.cpp)

          AddStr(__TEXT("The transferred index is %d"), reportIndex);
        }

      } //End of New Day if statment

    } //End of found *.dif file if statment


  } //end of infinite while loop

  FindCloseChangeNotification(hFile);                                   // Close the file change
                                                        // object handle. (flag)
                                                        // (WIN32)

  // Return thread
  return(0);
}
```

D.4 GC1_INDX.CPP

```
/*****************************************************************************/
// Module: Gc1_indx.cpp
// This module replaces the index in the *.dif file with the updated one
// computed by the main program
// By  Shawn Harden
//    Saikat Kanjilal
//    Russell S. Katz
//    Cuong Vo
// Version 1.0
// June 23, 1995
/*****************************************************************************/

// System Includes
#include <fstream.h>
#include <stdlib.h>
#include <iomanip.h>

int indx(char *difName, int intIndex)
{
  char  col1[5],            // Month column char variable
        col2[5],            // Day column char variable
        col3[6],            // Year column char variable
        col4[11],           // Time column char variable
        index[4],           // Report index column char variable
        col6[10],           // First data column char variable
        col7[10],           // Second data column char variable
//      col8[10],           // Third data column char variable
//      col9[10],           // Fourth data column char variable
        space[3] = " ";     // Empty space char variable

  fstream iofile;           // Input/Output file stream pointer

  iofile.open(difName, ios::in);  // Open the *.dif file for reading


  iofile >> col1 >> col2 >> col3 >> col4 >> index >> col6 >>
          col7 /* >> col8 >> col9 */ ;
                        // Reads in one line of the file in fields

  iofile.close();       // Close the *.dif file for reading

  itoa(intIndex, index, 10);   // Convert intIndex to char and place in index

  iofile.open(difName, ios::out);   // Open *.dif file for writing

// Writes the data to the *.dif file in fields
  iofile << col1 << space << col2
              << space << col3
              << space << col4
              << space << setw(7) << index          // setw() sets the width
              << space    << setw(8) << col6        // of the positon the char
              << space << setw(8)<< col7                  // variable is written to,
/*                << space    << setw(8) << col8            // allowing spacing.
              << space << setw(8) << col9 */ << "\n";

    iofile.close();         // Close the *.dif file for writing

  return intIndex;                  // return intIndex to main program

} // End of gc1_indx module
```

D.5 GC1_TRNS.CPP

```
/*****************************************************************************/
// Module: Gc1_trns.cpp
// This module reads in the entire *.dif file and retrieves the last index.
// By Shawn Harden
//   Saikat Kanjilal
//   Russell Katz
//   Cuong Vo
// Version 1.0
// June 23, 1995
/*****************************************************************************/

// System Includes
#include <stdlib.h>
#include <fstream.h>

int ind_trns(char fname[50])
{
  int indexOut,indexOut1;          // Output integer index variables

                              // Each of these variables will contain
                              // a field from the *.dif file.

  char  col1[5],              // Month column char variable
        col2[5],            // Day column char variable
        col3[6],              // Year column char variable
        col4[11],             // Time column char variable
        index[4],             // Report index column char variable
        col6[10],             // First data column char variable
        col7[10];             // Second data column char variable
//      col8[10],             // Third data column char variable
//      col9[10];             // Fourth data column char variable

  // Declare the file pointer.
  fstream infile;                // *.dif file input stream pointer

  //open the file
  infile.open(fname, ios::in);    // Open the *.dif file for reading

  // Start the infinite loop.
  while(!infile.eof())            // While we have not reached the end-of-file...
  {
    infile >>  col1 >> col2 >> col3 >> col4 >> index >> col6 >>
    col7 /* >> col8 >> col9 */ ;       // Read in a line of the file by fields.

    if(!infile.eof())            // If not the end-of-file yet
    {
     indexOut = atoi(index);    // indexOut is equal to the integer
                              // value of index
    }

    else                      // If it is the end-of-file
    {

     indexOut1 = atoi(index);  // indexOut1 is equal to the integer
                              // value of index
     ifstream::close;         // close the report file pointer
    }

  }

  if (indexOut==0)                // If indexOut is 0 (not read correctly)
```

```
  {
    return indexOut1;              // Return the value in indexOut1
  }

  else if (indexOut1 = =0)       // Else if indexOut1 is 0 (not read correctly)
  {
    return indexOut;               // Return the value in indexOut
  }

  else                           // Else if an error with both...
  {
    return 0;                    // Return a zero to indicate error
  }

} // End of gc1_trns module
```

D.6 APPND.CPP

```
/*****************************************************************************/
// Module: appnd.cpp
// This module takes one line from the *.dif file and appends it to
// the report output file. It's also generated LabView interface file.
// By  Shawn Harden
//    Saikat Kanjilal
//    Russell S. Katz
//    Cuong Vo
// Version 1.0
// June 23, 1995
/*****************************************************************************/

// System Includes
#include <fstream.h>
#include <stdlib.h>
#include <iomanip.h>

void appnd(char *rptName, char *gc1_nam, int cal)
{                                  //rptName = output report file
                                   // gc1_nam = gc*.dif
                                   // cal = 1 or 0, depending on cal time

  int int_cal;                 // Integer value for holding cal boolean

  char VI_interface_filename[30] = "C:\\GC1_VI.DAT";// The LabView interface filename

  char  col0[3],                 // Calibration column char variable
        col1[5],             // Month column char variable
        col2[5],             // Day column char variable
        col3[6],               // Year column char variable
        col4[11],              // Time column char variable
        index[4],              // Report index column char variable
        col6[10],              // First data column char variable
        col7[10],              // Second data column char variable
//      col8[10],              // Third data column char variable
//      col9[10],              // Fourth data column char variable
        space[3] = " ";        // Empty space char variable
```

```
// Declare the input and output files.

  if(cal == 1)                  // If there is a calibration in progress
  {
    int_cal = 1;               // Int_cal is 1
    itoa(int_cal, col0, 10);     // Convert int_cal to char and put in col0
  }
  else                          // Otherwise
  {
    int_cal = 0;               // Int_cal is 0
    itoa(int_cal, col0, 10);  // Convert int_cal to char and put in col0
  }

  ifstream ifile;               // The input file stream pointer
  ofstream ofile;               // The output file stream pointer
  ofstream Vlfile;              // The output LabView interface file stream pointer

  ifile.open(gc1_nam, ios::in);   // Open the input file for reading

  ifile >> col1 >> col2 >> col3 >> col4 >> index >> col6 >>
        col7 /* >> col8 >> col9 */ ;
                                // Reads in one line of the file by fields

  ifile.close();                // Close input file pointer

  Vlfile.open(Vl_interface_filename, ios::out); // Open LabView file for writing only

  // Writes the concentrations to the io file by fields
  Vlfile << col6 << "\n"
       << col7 << "\n";

  Vlfile.close();               // Close output file pointer

  ofile.open(rptName, ios::app);   // Open the output file for appending

 // Writes the data to the io file by fields
 ofile << space << col0 << space << col1 << space << col2
            << space << col3 << space << col4
            << space << setw(7) << index
            << space     << setw(8) << col6
            << space << setw(8) << col7
/*             << space    << setw(8) << col8
            << space << setw(8) << col9 */ << "\n";

// Close the output file.
    ofile.close();              // Close output file pointer

} // End of appnd module
```

D.7 DIF_CHK.CPP

```
/*****************************************************************************/
// Module: Dif_chk.cpp
// This module takes a line from *.dif file and checks for whether it is the
// first *.dif of a run.  If it is, the *.dif file is deleted.
// By Russell S. Katz
// Version 1.0
// June 23, 1995
/*****************************************************************************/

// System Includes
#include <fstream.h>
#include <stdlib.h>

int dif_chk(char * dif_name)            // dif_name = *.dif file
{
  char  col1[5],            // Month column char variable
        col2[5],            // Day column char variable
        col3[6],            // Year column char variable
        col4[11],           // Time column char variable
        index[4],           // Report index column char variable
        col6[10],           // First data column char variable
        col7[10];           // Second data column char variable
//      col8[10],           // Third data column char variable
//      col9[10];           // Fourth data column char variable

// Declare the input file.
  ifstream ifile;           // *.dif file input stream pointer

  ifile.open(dif_name, ios::in);  // Open the *.dif file for reading

  ifile >> col1 >> col2 >> col3 >> col4 >> index >> col6 >>
        col7 /* >> col8 >> col9 */ ;
                            // Reads in one line of the file by fields

  ifile.close();            // Close the *.dif file pointer

  if(col6[0] > 57 || col6[0] < 48)   // If the first character of the first
  {                         // data column is not a numerical character...
    return(0);                  // Return a 0, indicating that the *.dif
  }                         // file included a header and should be deleted.

  else                          // Else...
  {
    return(1);              // Return a 1, indicating a correct *.dif file
  }

} // End of dif_chk module
```

D.8 DIF_FIX.CPP

```
/*****************************************************************************/
// Module: Dif_fix.cpp
// This module takes the first *.dif file from a GC run and strips off
// the first header line.
// By Russell S. Katz
// Version 1.0
// July 11, 1995
/*****************************************************************************/

// System Includes
#include <fstream.h>
#include <stdlib.h>

void dif_fix(char * dif_name)          // dif_name = *.dif file
{
  char  headerLine[90];      // Container variable for header line
  char  dataLine[90];            // Container variable for data line


// Declare the input file.
  ifstream ifile;            // *.dif file input stream pointer
  ofstream ofile;            // *.dif file output stream pointer

  ifile.open(dif_name, ios::in);  // Open the *.dif file for reading

  ifile.getline(headerLine, 89);  // Get first line of dif file (header)
  ifile.getline(dataLine, 89);    // Get second line of dif file (data line)

  ifile.close();             // Close the *.dif file pointer

  ofile.open(dif_name, ios::out);   // Open the *.dif file pointer for output

  ofile << dataLine << "\n";          // Overwrite the bogus header with data line

  ofile.close();                 // Close the dif file

} // End of dif_fix module
```

D.9 GC_FIL.CPP

```
/*****************************************************************************/
// Module: gc_fil.cpp, based upon TDC.CPP by Cliff Hewitt(LANL)
// This module takes an empy char array and an integer gc identifier
// and constructs a file/directory day string. (i.e. - 50623GC1.)
// By  Shawn Harden
//    Saikat Kanjilal
//    Russell S. Katz
//    Cuong Vo
// Version 1.0
// June 23, 1995
/*****************************************************************************/

//System include files
#include <stdlib.h>
#include <time.h>
#include <string.h>


void fil_name(char *sdat, int gc)       // sdat = empty date string
{                                       // gc = integer gc identifier

  //local variables
  char *tms;                            // Time string char array
  char str_gc[5];                       // Character variable holder for gc id
  int t;                                // Integer for holding month addition

/*****************************************************************************/
  time_t tm;                            // Time time variable
  size_t len;                           // Time size variable

  // Get Time String
  time(&tm);
  tms=ctime(&tm);
  len=strlen(tms);
  tms[len-1]=0;

  //sum char in month
  t=tms[4]+tms[5]+tms[6];

  //switch on month sum and create temp date string
  //month day year
  switch(t){
    case 281: sdat[1]=48; sdat[2]=49; break; //Jan
    case 269: sdat[1]=48; sdat[2]=50; break; //Feb
    case 288: sdat[1]=48; sdat[2]=51; break; //Mar
    case 291: sdat[1]=48; sdat[2]=52; break; //Apr
    case 295: sdat[1]=48; sdat[2]=53; break; //May
    case 301: sdat[1]=48; sdat[2]=54; break; //Jun
    case 299: sdat[1]=48; sdat[2]=55; break; //Jul
    case 285: sdat[1]=48; sdat[2]=56; break; //Aug
    case 296: sdat[1]=48; sdat[2]=57; break; //Sep
    case 294: sdat[1]=49; sdat[2]=48; break; //Oct
    case 307: sdat[1]=49; sdat[2]=49; break; //Nov
    case 268: sdat[1]=49; sdat[2]=50; break; //Dec
    default:  sdat[1]=48; sdat[2]=48; break; //ERR;
    }
/*****************************************************************************/
// The code above in the comment lines was taken from TDC.CPP by Cliff Hewitt,
// LANL

  itoa(gc, str_gc, 10);                 // Convert gc to char and place
```

```
                                        // in str_gc

  sdat[0] = tms[23];                    // sdat = " 06"
  sdat[3] = tms[8];                        // sdat = "5062"
  sdat[4] = tms[9];                        // sdat = "50623"
  sdat[5] = 71;                         // sdat = "50623G"
  sdat[6] = 67;                         // sdat = "50623GC"
  sdat[7] = str_gc[0];                     // sdat = "50623GC1"
  sdat[8] = 46;                         // sdat = "50623GC1."
  sdat[9] = 0;                          // sdat = "50623GC1." with null
                                     //                                        terminator

  return;                               // Return filled sdat to main program

} // End of gc_fil module
```

D.10 INDEX3.CPP

```
/*****************************************************************************/
// Module: index3.cpp
// This module takes one filled integer and one empty character parameter, and
// converts the integer value to a three-character character value.
// (i.e. - int = 1 becomes char = "003"
// By Russell S. Katz
// Version 1.0
// June 23, 1995
/*****************************************************************************/

// System Includes
#include <stdlib.h>
#include <string.h>

void index_str(int int_index, char *ch_index)          // int_index: filled int param.
{                    // ch_index: empty char param.
  int hundreds;                  // Integer hundreds digit holder.
  int tens;                  // Integer tens digit holder
  int ones;                  // Integer ones digit holder

  char hund[2];              // Character hundreds digit holder
  char tns[2];               // Character tens digit holder
  char ouns[2];              // Character ones digit holder
  char temp[20];                 // Character temporary variable to hold partial
                     // string while it is constructed

  hundreds = (int_index - ((int_index/1000)*1000))/100;                    // Mathmatically
                                                                                  // strip off
                                                                                  // hundreds digit

  tens = ((int_index - ((int_index/1000)*1000))- (hundreds * 100))/10;
                                                                          // Mathmatically
                                                                          // strip off tens
                                                                          // digit

  ones = (((int_index - ((int_index/1000)*1000)) - (hundreds * 100))
                                                                                  - (tens * 10));
                     // Mathmatically
                     // strip off
                     // ones digit

  itoa(hundreds, hund, 10);          // Convert hundreds digit to char
  itoa(tens, tns, 10);               // Convert tens string to char
  itoa(ones, ouns, 10);              // Convert ones string to char

  strcpy(temp, hund);                // Copy hundreds char into temp
  strcat(temp, tns);                 // Concatenate tens char into temp
  strcat(temp, ouns);                // Concatenate ones char into temp
  strcpy(ch_index, temp);                // Copy temp into ch_index

  return;                            // return to main program

} // End of index3 module
```

D.11 RPT_TRNS.CPP

```
/*****************************************************************************/
// Module: Rpt_trns.cpp
// This module reads in the entire *.rpt for the current day, if it exists,
// file  and get the last index.  This module is used if the gcapp has been
// run earlier in the day, stopped, and then restarted.
// By Russell Katz, adopted from gc1_trns.cpp
// Version 1.0
// June 23, 1995
/*****************************************************************************/

// Systems Includes
#include <stdlib.h>
#include <fstream.h>

int rpt_trns(char fname[50])
{
  int indexOut,indexOut1;          // Output integer index variables

                              // Each of these variables will contain
                              // a field from the *.rpt file.

  char  col0[3],                  // Calibration column char variable
      col1[5],            // Month column char variable
      col2[5],            // Day column char variable
      col3[6],                // Year column char variable
      col4[11],               // Time column char variable
      index[4],               // Report index column char variable
      col6[10],               // First data column char variable
      col7[10];               // Second data column char variable
//    col8[10],               // Third data column char variable
//    col9[10];               // Fourth data column char variable

  // Declare the file pointer.
  fstream infile;             // Report file input stream pointer

  //open the file
  infile.open(fname, ios::in);    // Open the report file for reading

  // Start the infinite loop.
  while(!infile.eof())            // While we have not reached the end-of-file...
  {
    infile >>  col0 >> col1 >> col2 >> col3 >> col4 >> index >> col6 >>
    col7 /* >> col8 >> col9 */ ;       // Read in a line of the file by fields.

    if(!infile.eof())           // If not the end-of-file yet
    {
      indexOut = atoi(index);    // indexOut is equal to the integer value of index
    }

    else                      // If it is the end-of-file
    {

      indexOut1 = atoi(index);  // indexOut1 is equal to the integer value of index
      ifstream::close;        // close the report file pointer
    }

  }

  if (indexOut==0)                // If indexOut is 0 (not read correctly)
  {
    return indexOut1;             // Return the value in indexOut1
```

```
  }

  else if (indexOut1 = =0)        // Else if indexOut1 is 0 (not read correctly)
  {
    return indexOut;                // Return the value in indexOut
  }

  else                            // Else if an error with both...
  {
    return 0;                     // Return a zero to indicate error
  }

} // End of rpt_trns module
```

D.12 TIME_STP.CPP

```
/*****************************************************************************/
// Module: Time_stp.cpp
// This module creates a time stamp based on today's date & time
// By Russell S. Katz, based on code TDC.cpp from Cliff Hewitt, LANL
// Version 1.0
// June 23, 1995
/*****************************************************************************/

// System Includes
#include <string.h>
#include <time.h>

void time_stp(char *tm_stp)
{
  // Local Variables
  char * tms;                    // Time string char variable
  time_t tm;                     // Time time variable
  size_t len;                 // Time size variable

  // get time string
  time(&tm);
  tms=ctime(&tm);
  len=strlen(tms);
  tms[len-1]=0;

  //create output file name for the day
  tm_stp[0]=tms[0];              // "F"
  tm_stp[1]=tms[1];           // "Fr"
  tm_stp[2]=tms[2];           // "Fri"
  tm_stp[3]=tms[3];              // "Fri "
  tm_stp[4]=tms[4];              // "Fri J"
  tm_stp[5]=tms[5];           // "Fri Ju"
  tm_stp[6]=tms[6];           // "Fri Jun"
  tm_stp[7]=tms[7];              // "Fri Jun "
  tm_stp[8]=tms[8];              // "Fri Jun 2"
  tm_stp[9]=tms[9];              // "Fri Jun 23"
  tm_stp[10]=tms[10];            // "Fri Jun 23 "
  tm_stp[11]=tms[11];            // "Fri Jun 23 0"
  tm_stp[12]=tms[12];            // "Fri Jun 23 08"
  tm_stp[13]=tms[13];            // "Fri Jun 23 08:"
  tm_stp[14]=tms[14];            // "Fri Jun 23 08:3"
  tm_stp[15]=tms[15];            // "Fri Jun 23 08:32"
  tm_stp[16]=tms[16];            // "Fri Jun 23 08:32:"
  tm_stp[17]=tms[17];            // "Fri Jun 23 08:32:0"
  tm_stp[18]=tms[18];            // "Fri Jun 23 08:32:06"
  tm_stp[19]=tms[19];            // "Fri Jun 23 08:32:06 "
  tm_stp[20]=tms[20];            // "Fri Jun 23 08:32:06 1"
  tm_stp[21]=tms[21];            // "Fri Jun 23 08:32:06 19"
  tm_stp[22]=tms[22];            // "Fri Jun 23 08:32:06 199"
  tm_stp[23]=tms[23];            // "Fri Jun 23 08:32:06 1995"
  tm_stp[24] = 0;             // "Fri Jun 23 08:32:06 1995" with null
                           //                                          terminator

  return;                  // Return to main program

} // End of time_stp module
```

D.13 GC1APP32.H

```
// Program functions
void dif_fix(char *);
int rpt_trns(char[]);
int dif_chk(char *);
void index_str(int, char[]);
void time_stp(char *);
void AddStr (LPCTSTR, ...);
void appnd(char *, char *, BOOL);
void fil_name(LPSTR, int);
int indx(char *, int);
int ind_trns(char[]);
```

D.14 HEADER1.H

```
Cal Mo Day Yr    TIME       INDEX   H2Hi   H2Low
```

D.15 RESOURCE.H

```
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by AlertIO.rc
//
#define IDD_GC                  1
#define IDC_LOG                     103
```

D.16 GC1APP32.RC

```
/*****************************************************************************/
// Resource File: Gc1app32.rc
// This file defines the graphical interface for the gc1app32 application
// By Russell Katz
// Version 1.0
// June 23, 1995
/*****************************************************************************/

#include "resource.h"

gc1app32 ICON "gc1app32.ICO"

IDD_GC DIALOG DISCARDABLE 6, 15, 161, 75
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU | WS_MINIMIZEBOX
CAPTION "GC1 Data Acquisition Application"
FONT 8, "MS Sans Serif"
{
 DEFPUSHBUTTON "&Run", IDOK, 13, 2, 50, 14
 DEFPUSHBUTTON "&Quit", IDCANCEL, 98, 2, 50, 14
 LTEXT "Execution &Log:", IDC_STATIC, 5, 21, 60, 8
 LISTBOX IDC_LOG, 5, 32, 150, 38, NOT LBS_NOTIFY | WS_BORDER |
        LBS_NOINTEGRALHEIGHT | WS_BORDER | WS_VSCROLL | WS_TABSTOP

}
```

APPENDIX E -- C++ SOURCE CODE LISTINGS for GC2 ZIP Routines

E.1 GC2_ZIP.INI

```
path = c:\bin\pkzip.exe
zippath = d:\gc2\
out_path = h:\gc2\
nth_c = 1


/*********************************************/
Parameter Definintions

path:      The path for the pkzip executable

zippath:   The path for the files to be zipped

out_path: The path for the zip file

nth_c:     The nth number of chromatograms to be saved
    (i.e. - 5 means every 5th c-gram is saved)
```

E.2 GC2ZIP.CPP

```
/*****************************************************************************
// Gc2zip.cpp
// Russell Katz & Cuong Vo, June 1995
// This program uses PKZip(TM) to archive GC data and prepare it for
// being burned onto a CD.
// Version Beta 0.50
// Cuong Vo
// Turned over, August 1995
// Last Mod, Feb 1996
// Version Beta 1.1
//
*****************************************************************************/


// System includes
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fstream.h>
#include <dos.h>
#include <iostream.h>

// User Includes
#include "gc2zip.h"          // This file includes function prototypes for zip's
                             // and compress modules.

void main(void)
{
  char err_file[30] = "D:\\GC2_ZIP.ERR";     // The zip error log file

  // ini file pointer and title.          // The ini and error file
  fstream ini, err;                                  // pointers

  err.open(err_file, ios::app);            // Open the error file for
                                           // appending

  err << "\n";                          // Add empty line to error file
```

```
char ini_file[30] = "c:\\bin\\GC2_ZIP.INI";  // The GC2_ZIP.INI file

  // Ini file input columns
  char col1[10]="\0";
  char col2[3]="\0";
  char col3[25]="\0";

  // program constants initialized from an ini file
  char exe_path[45]="\0";        // The pkzip.exe path
  char zip_path[45]="\0";        // The area defined for the files to be zipped


  // File extension strings
  char ext_star[5] = "*";        // File extension wildcard
  char ext_zip[5] = "ZIP";   // Zip file extension
  char ext_rpt[5] = "RPT";  // Report file extension
  char period[5] = ".";      // Period character
  char backs[5] = "\\";      // The "\" character
  char call[6] = "call";    // "Call" for calling pkzip.exe

  // File command strings
  char del[50] = "del";

  // Misc. strings
  char space[3] = " ";       // Space character
  char out_path[100]="\0";  // The output path for the zip file
  char nth_c_str[5]="\0";

  // Date strings
  char date_name[35]="\0";   // Holds file name from fil_name: "50616GC2."
  char day_dir[35]="\0";        // Holds day's directory name: "50616GC2"
  char day[5]="\0";             // Transfer variable to hold day digits of
                            // date_name "16"

  char month[4]="\0";           // Transfer variable to hold month digits
  char year[3]="\0";          // Transfer variable to hold year digit
  char day_string[15]="\0";    // Contains day string alone
  char del_dir[50]="\0";      // Contains directory path being zipped for
                            // removal

  // Temp strings
  char temp_path[100]="\0";      // Contains entire system call to zip files.
  char temp_zip[50]="\0";        // Contains zip file path & name
  char temp_zipped[75]="\0";        // Contains files to be zipped path & names
  char temp_del[50]="\0";        // Contains zip file path & name to delete
                            // after file copy.
  char temp_rpt[50]="\0";        // Report file char variable
  char temp_cgram[50]="\0";     // C-gram file char variable
  char temp[5]="\0";           // Temporary char variable
  char out_rpt[30]="\0";            // Output report variable
  char out_dir_temp[10]="D:\\"; // Temporary place for zip file befor transfer
  int int_year;                 // Year digit
  int int_month;                    // Month digits
  int int_day;                  // Day digit to be decremented
  int nth_c_int;

  WIN32_FIND_DATA fileData;     // Find File Data Structure

  HANDLE findFile;              // Find File Handle

  // This reads the contents of the ini file
  // and places it in the correct variables
  ini.open(ini_file,ios::in);
```

```
ini >> col1 >> col2 >> col3;
strcpy(exe_path, col3);

ini >> col1 >> col2 >> col3;
strcpy(zip_path, col3);

ini >> col1 >> col2 >> col3;
strcpy(out_path, col3);

ini >> col1 >> col2 >> col3;
strcpy(nth_c_str, col3);

ini.close();          // closes ini file

nth_c_int = atoi(nth_c_str);

fil_name(date_name, 2); // File date_name with the date (i.e. - 50803GC2.)

day[0] = date_name[3];
day[1] = date_name[4];  // Set char day equal to the day of the month
                        // (i.e. - in May2595. , day = 25

int_day = atoi(day);    // convert the day string to an int.

if(int_day == 1)
{
  year[0] = date_name[0];
  month[0] = date_name[1];
  month[1] = date_name[2];

  int_year = atoi(year);
  int_month = atoi(month);

  switch(int_month)
  {
     case 3:
        date_name[1] = '0';
        date_name[2] = '2';
        date_name[3] = '2';
        date_name[4] = '8';

        if(int_year == 6 || int_year == 0)
        {
           date_name[1] = '0';
           date_name[2] = '2';
           date_name[3] = '2';
           date_name[4] = '9';
        }

        break;

     case 5:
        date_name[1] = '0';
        date_name[2] = '4';
        date_name[3] = '3';
        date_name[4] = '0';

        break;

     case 7:
        date_name[1] = '0';
        date_name[2] = '6';
        date_name[3] = '3';
```

```
        date_name[4] = '0';

        break;

    case 10:
        date_name[1] = '0';
        date_name[2] = '9';
        date_name[3] = '3';
        date_name[4] = '0';

        break;

    case 12:
        date_name[1] = '1';
        date_name[2] = '1';
        date_name[3] = '3';
        date_name[4] = '0';

        break;

    default:
        int_month --;

        if (int_month < 10)
        {
            if(int_month == 0)          // If it is january(1 -- is zero)
            {
                int_year --;
                itoa(int_year, temp, 10);
                date_name[0] = temp[0];
                date_name[1] = '1';
                date_name[2] = '2';
                date_name[3] = '3';
                date_name[4] = '1';
            }

            else
            {
                itoa(int_month, temp, 10);
                month[0]='0';           // If int_month is less than ten,
                month[1]=temp[0];           // convert it to char and place
                                        // a '0' in front of it.

                date_name[1] = month[0];
                date_name[2] = month[1];
                date_name[3] = '3';
                date_name[4] = '1';
            }

        }                               // Ends if loop

        else
        {
            itoa(int_month, month, 10);         // Convert int_month to char
            date_name[1] = month[0];
            date_name[2] = month[1];
            date_name[3] = '3';
            date_name[4] = '1';
        }                               // Ends else

        break;

}                                       // End of month switch
```

```
}                                   // End of 1st of month if

else
{
  int_day--;              // decrement the day integer by one.
  if (int_day < 10)
  {
     itoa(int_day,temp,10);
     day[0]='0';          // If int_day is less than ten,
     day[1]=temp[0];        // convert it to char and place
  } // Ends if loop       // a '0' in front of it.

  else
  {
     itoa(int_day,day,10);     // Convert int_day to char
  }                        // Ends else

  date_name[3] = day[0]; // put the decremented day string back into
  date_name[4] = day[1]; // date_name.  (date_name was: May2595.
                         //                 date_name now is: May2495.)
}

strncpy(day_dir, date_name, 8);  // strip the period off of date_name and
                                 // place it in day_dir.

day_dir[8] = 0;                  // Null terminator for day_dir


strcpy(day_string, day_dir);

strcat(day_dir, backs);          // Concatenate "\" onto day_dir
                                 // day_dir = "50616GC2\"

strcpy(temp_path, call);         // Copy call onto temp_path
strcat(temp_path, space);          // Concatenate space onto temp_path
strcat(temp_path, exe_path);       // Concatenate exe_path onto temp_path
                                 // Temp_path = "call c:\bin\pkzip.exe"

strcat(temp_path, space);          // Concatenate a space onto temp_path
                                 // temp_path = "c:\bin\pkzip.exe "

strcpy(out_rpt, out_path);         // out_rpt = "g:\gc2\"
strcpy(temp_rpt, zip_path);
strcpy(temp_cgram, zip_path);
strcpy(temp_zip, out_dir_temp);  // Concatenate zip_path onto temp location
strcpy(temp_zipped, zip_path);   // Concatenate zip_path onto temp_zipped
strcpy(del_dir, zip_path);


strcat(temp_rpt, day_dir);
strcat(temp_cgram, day_dir);
strcat(temp_zipped, day_dir);      // Concatenate day_dir onto temp_zipped

strcat(del_dir, day_string);       // Copy temp_zipped into del_dir
                                 // del_dir = x:\50616GC2

strcat(out_rpt, day_string);       // out_rpt = "g:\GC2\50621GC2"
strcat(temp_rpt, day_string);      // temp_rpt = "D:\GC2\50621GC2\50621GC2"
strcat(temp_cgram, day_string);  // temp_cgram = "D:\GC2\50621GC2\50621GC2"

strcat(out_rpt, period);       // out_rpt = "g:\GC2\50621GC2."
strcat(temp_rpt, period);        // temp_rpt = "D:\GC2\50621GC1\50621GC2."
strcat(temp_cgram, period);    // temp_cgram = "D:\GC2\50621GC1\50621GC2."
```

```
strcat(out_rpt, ext_rpt);      // out_rpt = "g:\GC2\50621GC2.RPT"
strcat(temp_rpt, ext_rpt);       // temp_rpt = "D:\GC2\50621GC2\50621GC2.RPT"

strcpy(temp_del, temp_zipped);  // Copy temp_zipped into temp_del

strcat(temp_zip, date_name);     // Concatenate date_name onto temp_zip
strcat(temp_del, date_name);      // Concatenate date_name onto temp_del
strcat(temp_zipped, ext_star); // Concatenate date_name onto temp_zipped

strcat(temp_zipped, period);       // Concatenate period onto temp_zipped

strcat(temp_zip, ext_zip);      // Concatenate ext_zip onto temp_zip
strcat(temp_zipped, ext_star); // Concatenate ext_star onto temp_zipped
strcat(temp_del, ext_star);       // Concateneate ext_star onto temp_del

strcat(temp_path, temp_zip);     // Concatenate temp_zip onto temp_path

strcat(temp_path, space);       // Concatenate space onto temp_path

strcat(temp_path, temp_zipped);  // Concatenate temp_zipped onto temp_path

compress(temp_rpt, temp_cgram, nth_c_int);         // Data compression algorithm

printf("\n copy from %s to %s\n",temp_rpt,out_rpt);
CopyFile(temp_rpt, out_rpt, FALSE);                              // Copy rpt file over to
sleep(2);                                  // output server

printf("command %s\n",temp_path);
system(temp_path);                                  // Run pkzip on May2595.*
sleep(2);
  //temp_zip = "d:\50616GC2.zip"
  //temp_zipped = "x:\gc2\50616GC2\50616GC2.*"
  //temp_path = "c:\bin\pkzip d:\50616GC2.zip
  //                                  x:\<dir>\50616GC2\50616GC2.*"
  //temp_del = "x:\<dir>\50616GC2\50616GC2.*"
  // where x:\<dir>\ are paths from the ini file

strcat(out_path, date_name);      // Concatenate date_name onto out_path
strcat(out_path, ext_zip);          // Concatenate ext_zip onto out_path
                                  // out_path = "x:\50616GC2.zip"
                                  // where x:\ is out_path from ini file



printf("move from %s to %s\n",temp_zip, out_path);
MoveFile(temp_zip, out_path);                    // Move zip file to
sleep(2);                                  // output_path

findFile = FindFirstFile(out_path, &fileData);// Try to find temp_zip file

err << "MoveFile ERROR: " << GetLastError() << "\n";// Print error code to log file

if(findFile = INVALID_HANDLE_VALUE)
//  if(GetLastError() == 0 || GetLastError() == 6) // Not use if no error or
                                                   // bad parameter (which
{                                                  // we always seem to get)
  FindClose(findFile);                  // close FindFile handle

  strcat(del, space);              // Concatenate space onto del
  strcat(del, temp_del);           // Concatenate temp_zipped on del
  printf("delete %s\n",del);
  system(del);                     // system call to delete 50616GC2.*
```

```
      sleep(2);                         // Sleep 5 sec to allow file
                                        // deletion to be updated by
                                        // file system.
      printf("Remove directory %s\n",del_dir);
      RemoveDirectory(del_dir);                 // remove the day directory

      sleep(2);                         // sleep 5 sec to allow
                                        // window read.
    }

   else          // If a GetLastError() not equal to 0 or 6...
   {
       FindClose(findFile);             // Close FindFirstFile handle
       err << "Data to zip not found...leaving data for check\n";
   }
  err.close();
  sleep(10);
} //End of zip module
```

E.3 GC_FIL.CPP

```
// GC_FIL.CPP, based upon TDC.CPP by Cliff Hewitt(LANL)


//System include files
#include <iostream.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>


void fil_name(char *sdat, int gc)
{

  //local variables
  char *tms;
  char str_gc[5];
  int t;

  time_t tm;
  size_t len;

  // Get Time String
  time(&tm);
  tms=ctime(&tm);
  len=strlen(tms);
  tms[len-1]=0;

  //sum char in month
  t=tms[4]+tms[5]+tms[6];

  //switch on month sum and create temp date string
  //month day year
  switch(t){
    case 281: sdat[1]=48; sdat[2]=49; break; //Jan
    case 269: sdat[1]=48; sdat[2]=50; break; //Feb
    case 288: sdat[1]=48; sdat[2]=51; break; //Mar
    case 291: sdat[1]=48; sdat[2]=52; break; //Apr
    case 295: sdat[1]=48; sdat[2]=53; break; //May
    case 301: sdat[1]=48; sdat[2]=54; break; //Jun
    case 299: sdat[1]=48; sdat[2]=55; break; //Jul
    case 285: sdat[1]=48; sdat[2]=56; break; //Aug
    case 296: sdat[1]=48; sdat[2]=57; break; //Sep
    case 294: sdat[1]=49; sdat[2]=48; break; //Oct
    case 307: sdat[1]=49; sdat[2]=49; break; //Nov
    case 268: sdat[1]=49; sdat[2]=50; break; //Dec
    default: cout<<"ERROR IN SWITCH TDC.CCP\n";
    }

  itoa(gc, str_gc, 10);

  sdat[0] = tms[23];
  sdat[3] = tms[8];
  sdat[4] = tms[9];
  sdat[5] = 71;//
  sdat[6] = 67;//      Will be replacing these with data from
  sdat[7] = str_gc[0];//     *.ini file.
  sdat[8] = 46;//
  sdat[9] = 0;

  return;
}
```

E.4 EXP2.CPP

```
// Shawn R. Harden  July 14, 1995
// This program reads low and high concentrations from an rpt file,
// saves every Nth C-gram, compares the values read to a set value,
// and checks for calibration. It then deletes the corresponding
// C-gram depending on those comparisons.
// Cuong Vo, August 1995
// Last mod and turned over
//

#include <iostream.h>
#include <fstream.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <math.h>
#include <dos.h>

#include "function.h"

struct fileinfo
 {
    char col1[2],col2[6],col3[6],col4[6],col5[9],col6[7],
       col7[9],col8[9],col9[9],col10[9];

 } file[999]; // Declares 999 input structures.

void compress(char * filename, char crname[50], int N)
{
//    cout << "Filename: " << filename << "\n";
//    cout << "Crname: " << crname << "\n";
//    cout << "N: " << N << "\n";

    char index_ext[5];
    int max,count,n,i;
    float calibr[999];
    float lhydro[999];
    float lnitro[999];
    float hhydro[999];
    float hnitro[999];
    char base[21];

    int int_index;
    max = 999;
    count = 1;
    n = 0;


    ifstream infile(filename);

// Skips header.
       infile >> file[count].col1 >> file[count].col2 >> file[count].col3
              >> file[count].col4 >> file[count].col5 >> file[count].col6
              >> file[count].col7 >> file[count].col8 >> file[count].col9
              >> file[count].col10;

// Reads in the file by fields into an array.
   while (n <= max)
   {

       infile >> file[count].col1 >> file[count].col2 >> file[count].col3
```

```
            >> file[count].col4 >> file[count].col5 >> file[count].col6
            >> file[count].col7 >> file[count].col8 >> file[count].col9
            >> file[count].col10 ;

// Changes concentration values from character to float.
      calibr[count] = atof (file[count].col1);
      lhydro[count] = atof (file[count].col7);
      hhydro[count] = atof (file[count].col8);
      lnitro[count] = atof (file[count].col9);
      hnitro[count] = atof (file[count].col10);

// Checks for the end of file (EOF).

    if(infile.eof()
    {

      n = 1010;
    }

//    cout << "Reading line #" << count << "\n";
    count++;

  }// Ends the while statement.


  max = count - 1;
  strcpy(base,crname);
  for (i = 1; i <= max; i++)
  {
  // Append the index to the name of the c-gram.

      int_index = atoi(file[i].col6);  // convert to int
      index_str(int_index, index_ext); // make index_ext 3 char ext.
      strcat(crname, index_ext);           // concatenate on end of crname

      if ((i % N == 0) || lhydro[i] == 0.000 || hhydro[i] == 0.000
                       || lnitro[i] == 0.000 || hnitro[i] == 0.000
                       || calibr[i] == 1)
      {
         cout << crname << " chromatogram was saved" << "\n";
      }
      else
      {
         cout << crname << " chromatogram was deleted" << "\n";
         remove(crname);
      }
      crname="\0";
      strcpy(crname,base); //Reset the characters in crname.
  }

// Close the input file.
  ifstream::close;
  cout << "End of Compress Section\n";
  sleep(5);
  return;
}
```

E.5 INDEX3.CPP

```
// This program creates a three digits index

#include <stdlib.h>
#include <string.h>

void index_str(int int_index, char *ch_index)
{
  int hundreds;
  int tens;
  int ones;

  char hund[2];
  char tns[2];
  char ouns[2];
  char temp[20];

  hundreds = (int_index - ((int_index/1000)*1000))/100;
  tens = ((int_index - ((int_index/1000)*1000))- (hundreds * 100))/10;
  ones = (((int_index - ((int_index/1000)*1000))- (hundreds * 100)) - (tens * 10));

  itoa(hundreds, hund, 10);
  itoa(tens, tns, 10);
  itoa(ones, ouns, 10);

  strcpy(temp, hund);
  strcat(temp, tns);
  strcat(temp, ouns);
  strcpy(ch_index, temp);


  return;
}
```

E.6 GC2ZIP.H

```
// Defines of user-created functions

fil_name(char *, int);
compress(char *, char *, int);
```

E.7 FUNCTION.H

```
index_str(int, char *);
```

APPENDIX F -- C++ SOURCE CODE LISTINGS for GC2APP32

F.1 GC2.INI

```
backs = \
root = d:\gc2\
C_gram_old = gc2.
gcPath = H:\gc2.dif
outputPath = d:\gc2\
outputExt = rpt
gc_del = gc2.*
grampath = d:\gc2\
filepath = H:\
calPath = c:\gc2cal
```

****

ini file descriptions

backs: Character for backslash on file paths

root: Report and chromatogram output file root path

C_gram_old: Chromatogram GC run identifier

gcPath: Dif file created by EZChrom path

outputPath: Report and chromatogram output file root path

outputExt: Report output file extension

gc_del: Wildcard definition of files to delete in input directory

grampath: Chromatogram file output path

filepath: Report and chromatogram input file path to watch for file changes

calPath: File to watch for for calibration events

F.2 GC2APP32.DEF

```
NAME        OWL1
DESCRIPTION 'TEST WINDOW IN OWL FORMAT'
EXETYPE     WINDOWS

CODE    PRELOAD MOVEABLE DISCARDABLE
DATA    PRELOAD MOVEABLE MULTIPLE

HEAPSIZE    8192
STACKSIZE 8192
```

F.3 GC2APP32.CPP

```
/*****************************************************************************/
// Program GC2APP32.CPP                                    //
// Version Beta .68                                   //
/*****************************************************************************/

// May 18, 1995
//(0.51)
// Changed index char variable in appnd.cpp, gc1_trns.cpp, and gc1_indx.cpp
// to 4 characters long.
//(0.52)
// gc1 trans bug fixed by converting index to an integer before reaching end
// of file.
// (0.53)
// two variables are compared in gc1_trans (index0ut and index10ut)
// if either one is zero the other variable is returned
// (0.54)
// Path variables are loaded from gc1.ini file.
// an error of incorrectly skipping throught the while loop was detected
// and corrected as of 5/19/95. The problem was that the variable used in
// the for loop and the variable used in the while loop were the same.
// As a result the while loop was being ignored.
// (0.55)
// Corrected error from use of the gc1.ini file. Extra trigger resets
// for the dif file transfer were removed, since the directory where the
// dif and old c-grams vs. rpt file and new c-grams are now different.
// (0.56)
// Resized the executable window, buttons, and text box.
// (0.57)
// Added error log file. Error checking routines for file trip from something
// else other that *.dif and c-gram files, and for loss of communications
// with the gc computer.
// (0.58)
// Error in program due to inadvertent tripping of flag due to movefile
// command which renames the file and while doing this deletes the older
// version of the chromatogram. A reset has been put after the deletefile
// and the movefile commands. A module was added for the three character
// extensions for the chromatograms, a new getlasterror routine outputs
// the errors in text format.
// System command used for deleting gc1.* files upon startup of the program.
// (0.59)
// The error log file was expanded to include the GetLastError call
// results. A memory error occured due to the size of the char array known
// as time stamp. The array size was increased and the error was then
// corrected.
// (0.60)
// An automatic paging routine was added to the network timeout code.
// In the case of a network failure, gcs1host will page a specified pager.
// (0.61)
// The dos window appearing when calling system("del ...."), was removed.
// The WinExec command was used instead. Due to this change, the dif file
// appearance flag was changed to FILE_NOTIFY_CHANGE_LAST_WRITE, and
// a reset was eliminated after DeleteFile
// (0.62)
// Added gc_del to *.ini file.
// (0.63): June 1, 1995
// Complete commenting
// (0.64): June 8, 1995
// New version screens out first *.dif file from GC. Project file is
// updated to remove some Borland Warnings. Calibration column was added
// to the *.rpt file. The column contains a 1 when the gc is calibrating
// and a 0 when it is not. The grapical interface was modified also.
```

```
// (0.65): June 16, 1995
// Initial index transfer when the program began was updated to deal with
// the calibration column.  The output to the report file from an extra file
// trip was commented out, along with the file change output statement being
// moved farther down to only appear when the *.rpt file will be appended to.

// Another modification due to extra file trips was to delete a second
// "Beginning wait for file change" statement by deleting the line from the
// dialog box's text box.

// The C++ project was setup to remove the "Initialize code in header file"
// Borland warning.  This was done by placing all the included *.cpp files
// previous to the gc1app32.cpp file in the project, and creating a new
// gc1app32.h file to contain all the function prototypes for those modules.

// The updated program was also coded to screen out the first *.dif file of
// a run from the GC, since this file usually has a bad header that disrupts
// data handling.  If this file comes accross, it is deleted.

// The last change made was to make the file naming convention similar to
// that used for the FTIR.  GC1_fil_ was renamed GC_FIL, and its function
// changed to pass two parameters.  One parameter was the string to be
// filled with the file name, and the other as a gc identifier (1 or 2).
// The new file names are as follows: 50616GC1.<ext>.
//                Year ---------^||^^---- Day
//                              ||
//                                   Month

// (0.66): June 19, 1995
// Problems with the calibration-catching routine were fixed.  Switched the
// if and else loops for calibration and no dif file to have failure
// dealt with first.

// (0.67): June 20, 1995
// The previous effort was not successful.  An addition of the SetLastError
// function to set the error code to zero at the beginning of the infinite
// loop fixed the problem of the dif file not being found, and the problem
// of calibration not operating correctly.

// (0.68): July 11, 1995
// Addition of dif_fix function to strip out the header created by EZChrom
// on the first run of the GC.  First run data is no longer deleted.

// (0.69); Jan 17, 1996
// Modified  gc2app32.cpp to provide proper exit point when encounter network
// failure.

// (0.70); April 8, 1996
// Modified gc2app32.cpp to correct the date change problem occurred at midnight
// by allow shorter ellapse time between newtime and oldtime variable.

// Includes and Definitions
#define STRICT          // This is needed for the method of construction used for the dialog box.
#include <windows.h>    // Windows definitions include file.
#include <windowsx.h>   // Additional windows definitions include file
#include <stdio.h>      // Standard input and output for C++ include file.
#include <iostream.h>   // Input/Output streams include file.
#include <stdlib.h>     // Standard function library include file.
#include <fstream.h>    // File input/output stream include file.
#include <dos.h>        // Dos function include file.#include <windows.h>

// User Includes and Definitions
#include "Resource.H"   // ID specifications for dialog controls (Buttons, etc.)
```

```
#include "gc2app32.h"  // Function prototypes for gcapp modules.

#define header __TEXT("c:\\bin\\header.h")        // file containing Header
                                                  // for *.rpt file.

// Ini and error file streams and titles
fstream ini, chk;                    // file streams for ini and error files
char ini_file[30]= "gc2.ini";        // Ini file name string
char col1[16],col2[3],col3[17];          // Input columns from ini file
char err_chk[30]="error.log";            // Error file name string
char gc_id_str[3];


// Temp character variables
char temp1[50];
char temp2[50];
char temp3[50];
char temp4[50];
char temp5[50];
char temp20[50];
char temp21[50];
char temp22[50];
char temp_command[70] = "command.com /c ";


// Global program constants initialized from an ini file
char backs[5];              // Contains "/" character
char root[10];              // Will contain root directory where output will go.
char C_gram_old[15];            // Contains EZChrom run ID "gc1."
char gcPath[20];            // Contains the path of the dif file "c:\gc1\gc1.dif"
char outputPath[50];            // Also contains output path for data
char outputExt[10];         // Contains the extension for the output file "rpt"
char gc_del[20];            // Contains the string to delete at the start of the
                        // program "gc1.*"


// Error codes...
char err_temp[50];
char err_temp2[50] = "WaitForSingleObject timed out...";
char err_temp4[40];
char error[50];
char error2[50];
char win_exec_error_msg[80];

// Other Character variables
char dirName[50];
char Cgrampath[50];
char old_date[50];
char Path[50];
char new_date[50];
char del_path[20];
char del[20] = "del ";
char time_stamp[30];
char time_stamp1[30];

int reportIndex, difIndex;    // Index integers from *.rpt file and *.dif file
int gc_id;                    // Integer value taken from file name string
                        // to identify gc program is running with.

// WORD, DWORD, and Windows variables
DWORD win_exec_error;                 // WinExec error code container
WIN32_FIND_DATA fileData, calData;   // FindFirstFile file data container structures
LPSTR *FilePart, *Cpart, *Cnpart;   // SearchPath path container variables.
```

```
DWORD WINAPI NotifyThread 0;      // NotifyThread prototype

// Booleans
BOOL fFailIfExists = TRUE;                // CopyFile command parameter to have the
                                  // function fail if the file already exists
                                  // where it is trying to be copied to.

HWND g_hwndLB = NULL;                     // Global handle to dialog box's text output
                                  // box.

void AddStr (LPCTSTR szFmt, ...)   // AddString function from Advanced Windows
{                                 // By Jeffrey Richter.  It outputs
  char szBuf[150];                // to the dialog text box, and takes input
  int nIndex;                             // essentially like printf0;
  va_list va_params;

  //Make va_params point to the first argument after szFmt
  va_start(va_params, szFmt);

  //Build the string to be displayed
  vsprintf(szBuf, szFmt, va_params);

  do
  {
    //Add the string to the end of the list box
    nIndex = ListBox_AddString(g_hwndLB, szBuf);

    //If the list box is full, delete the first item in it
    if(nIndex == LB_ERR)
    {
      ListBox_DeleteString(g_hwndLB, 0);
    }
  } while (nIndex == LB_ERR);

  //Select the newly added item
  ListBox_SetCurSel(g_hwndLB, nIndex);

  //Indicate that we're done referencing the variable arguments
  va_end(va_params);
}

void printError0                          // Print error function to output text
{                                 // from Windows GetLastError0 function.
  char *ErrorBuffer;                      // Outputs to dialog text box.
  #define PERROR(X) printf(X);
  FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM,
          NULL, GetLastError0, MAKELANGID(LANG_ENGLISH, SUBLANG_ENGLISH_US),
          (LPTSTR)&ErrorBuffer, 0, NULL);  // Format message from system
                                           // GetLastError message into
                                           // english text and place in
                                           // ErrorBuffer variable
  AddStr(__TEXT("Error: %s"), ErrorBuffer);                 // Print out Error Buffer to
                                           // dialog text box.
}


void runMain(void)          // Main run function
{
  // Main char variables
  char reportName[30];       // Will contain name of report file

  // Main Booleans
```

```
BOOL copyMade = FALSE;  // Boolean indicating success of CopyFile function

// Main Handles
HANDLE hThread;         // Handle for NotifyThread executed in main program

// Main DWORDS
DWORD dwthreadID;       // System ID of NotifyThread.
DWORD cPath = 20;       // SearchPath parameter specifiying size of container
                        // variable the path is placed in.


// This reads the contents of the ini file
// and places it in the correct variables
    ini.open(ini_file,ios::in);       // open ini file for reading
    ini>> col1>>col2>>col3;
    strcpy(backs, col3);          // Copy from ini file into backs

    ini>> col1>>col2>>col3;
    strcpy(root, col3);           // Copy from ini file into root

    ini>> col1>>col2>>col3;
    strcpy(C_gram_old, col3);     // Copy from ini file into C_gram_old

    ini>> col1>>col2>>col3;
    strcpy(gcPath, col3);         // Copy from ini file into gcPath

    ini>> col1>>col2>>col3;
    strcpy(outputPath, col3);     // Copy from ini file into outputPath

    ini>> col1>>col2>>col3;
    strcpy(outputExt, col3);      // Copy from ini file into outputExt

    ini>> col1>>col2>>col3;
    strcpy(gc_del, col3);         // Copy from ini file into gc_del

    ini.close0;          // close ini file

    gc_id_str[0] = ini_file[2];
    gc_id = atoi(gc_id_str);      // Set the gc_id variable equal to the
                                  // character in the ini file definition.
                                  // if gc1.ini, then gc_id = 1

    fil_name(dirName, gc_id);             // Fill dirName with date (May3095.)
                                  // (Developed function, gc1_fil_.cpp)

    strcpy(temp3, root);          // Copy root into temp3 (temp3 = g:\gc1\)

    strncpy(temp2, dirName, 8);   // Copy first 7 char of dirName into temp2
                                  // temp2 = "May3095"

    strcat(temp3, temp2);         // Concatenate temp2 onto temp3
                                  // Temp3 = "g:\gc1\May3095"

    CreateDirectory(temp3, NULL); // Create directory using temp3
                                  // (WIN32)

    strcat(temp3, backs);         // Concatenate backs onto temp3
                                  // Temp3 = "g:\gc1\May3095\"

    strcpy(outputPath, temp3);    // Copy temp3 into outputPath

    fil_name(reportName, gc_id);        // Fill reportName with date (May3095.)
                                  // (Developed function, gc1_fil_.cpp)
```

```
strcat(reportName, outputExt);// Concatenate outputExt onto reportName
                              // reportName = "May3095.rpt"

strcat(temp3, reportName);   // Concatenate reportName onto temp3
                              // Temp3 = "g:\gc1\May3095\May3095.rpt"

copyMade = CopyFile(header, temp3, fFailIfExists);
                              // Copy file from header (g:\code\header.h)
                              // into temp3 file name.
                              // (WIN32)


if(copyMade == TRUE)         // If the copy succeeds.
{
 AddStr(__TEXT("No file for today, %s is created"), outputPath);
 reportIndex = 0;            // Set reportIndex = 0
 AddStr(__TEXT("The first index in today's file is %d"), reportIndex);
}

else                         // If the copy failed
{
   AddStr(__TEXT("Today's output file exists"));
   reportIndex = rpt_trns(temp3);          // Get the last index from the rpt file.
   AddStr(__TEXT("The transferred index is %d"), reportIndex);
}


/*********************************************************************/
//Find gc1.dif to see if it currently exists
//If it does, delete it and all other gc1.* files
/*********************************************************************/

SearchPath(NULL, gcPath, NULL, cPath, Path, FilePart);
                                  // Search for *.dif file

if(GetLastError() == ERROR_FILE_NOT_FOUND)  // If no *.dif file...
{
   AddStr(__TEXT("The file was not found"),0);
   AddStr(__TEXT("The *.dif and chromatogram files could not be deleted"),0);
}

else                         // If the *.dif file exists...
{
   strncpy(del_path, gcPath, 3);    // Copy first 3 char of gcPath in to del_path
   strcpy(temp20, del);             // Copy del into temp20
   strcat(temp20, del_path);  // Concatenate del_path into temp20
                              // temp20 = del F:\
   strcpy(temp21, temp20);   // Copy temp20 into temp21
   strcat(temp21, gc_del);   // Concatenate gc_del into temp21
                              // temp21 = del F:\gc1.*
   strcpy(temp22, temp21);   // Copy temp21 into temp22
   strcat(temp_command, temp22);         // Concatenate temp22 into temp_command
                              // temp_command = command.com /c del F:\gc1.*

   win_exec_error = WinExec(temp_command, SW_HIDE);
                              // Execute temp_command string
                              // "command.com /c del F:\gc1.*"

   if(win_exec_error < 32)    // If there is an error in WinExec
   {
      sprintf(win_exec_error_msg, "WinExec failed; error code = %d", win_exec_error);
      AddStr(__TEXT("%s"), win_exec_error_msg);   // Output error
   }
```

```
        else
        {
           AddStr(_TEXT("The *.dif and chromatogram files were deleted"),0);
        }                          // Delete was successful
      }

      sleep(10);                   // Allow file deletion to complete before
                                   // beginning new thread.

/*****************************************************************************/
// Main program loop/thread
/*****************************************************************************/

      hThread = CreateThread ( 0, 0, (LPTHREAD_START_ROUTINE) NotifyThread,
                                                                  0, 0, &dwthreadID );
                                   // Create a thread called NotifyThread
                                   // and let it begin to execute.
                                   // This is based on code from the
                                   // Advanced Windows book: Pages 60-64
                                   // and Chapter 9(Thread Synchronization)
                                   // (WIN32)

      CloseHandle(hThread);          // Close the handle to the thread
                                   // (WIN32)

} // End of Main Program


BOOL DialogInitialize (HWND hwnd, HWND hwndFocus, LPARAM lParam)
{

  // Save the handle of the dialog's list box in a global
  // so that the threads can easily gain access to it.
  g_hwndLB = GetDlgItem(hwnd, IDC_LOG);


  // Associate an icon with the dialog box.
  SetClassLong(hwnd, GCL_HICON, (LONG)
    LoadIcon((HINSTANCE) GetWindowLong(hwnd, GWL_HINSTANCE),
    _TEXT("gc2app32")));
  return(TRUE);
}


void DialogCommand (HWND hwnd, int id, HWND hwndCtl, UINT codeNotify)
{

  switch (id)                          // Switch on the id of a pressed button
  {
    case IDOK:                         // If the RUN button is pushed
      ListBox_ResetContent(g_hwndLB);          // Reset the text window and
      runMain();                   // Run the Main program
      break;

    case IDCANCEL:                 // If the QUIT button is pushed
      EndDialog(hwnd, id);           // Close the dialog box
      break;
  }
}
```

```
BOOL CALLBACK DialogProc (HWND hDlg, UINT uMsg, WPARAM wParam, LPARAM lParam)
{

  BOOL fProcessed = TRUE;                          // Set fProcessed to TRUE

  switch (uMsg) {                          // Switch on uMsg from the Windows system
    HANDLE_MSG(hDlg, WM_INITDIALOG, DialogInitialize);
                                        // If message is WM_INITDIALOG
                                        // run DialogInitialize function
    HANDLE_MSG(hDlg, WM_COMMAND, DialogCommand);
                                        // If message is WM_COMMAND
                                        // run DialogCommand function


    case WM_USER:                                  // If message is WM_USER
      break;                            // break out.

    default:                                 // If any other message
      fProcessed = FALSE;                          // reset fProcessed to FALSE
      break;                            // and continue executing
  }

  return(fProcessed);                              // Return fProcessed to function
                                        // that called DialogProc
}


int WINAPI WinMain (HINSTANCE hinstExe, HINSTANCE hinstPrev, LPSTR lpszCmdLine,
                                                                          int nCmdShow)

{

  DialogBox(hinstExe, MAKEINTRESOURCE(IDD_GC), NULL, DialogProc);
                                        // Create a Dialog Box with
                                        // dialog procedure function
                                        // DialogProc
  return(0);
}

DWORD WINAPI NotifyThread 0      // The Notify Thread started by main program
{
  // File Handles
  HANDLE hFile, findFile, findCal;

  // DWORDS and Booleans
  DWORD dw;                     // Will contain returned value of
                            // WaitForSingleObject.

  DWORD chPath = 20;            // SearchPath parameter specifing size of
                            // string to contain found path
                            // (filePath, line 644)

  BOOL notif = FALSE;           // Will contain returned value of
                            // FindFirstChangeNotification
  BOOL COPYGOOD = FALSE;            // Will contain returned value of CopyFile
                            // copying header into temp3 for new day's
                            // file name. (Line )
  BOOL Cal = FALSE;             // Calibration boolean.

  // Integers
  int i = 1;                    // Constant value for infinite loop
  int j = 0;                    // Counter for reading from ini file
```

```
// Character constants from ini file
char grampath[20];          // Path where c-grams will be placed.
char filePath[20];        // Path to search for dif and chromatogram files.
char calPath[50];           // Contains path to search for calibration
                        // flag file.

// Character Variables
char C_gram_new[50];            // Will contain variable string for new
                        // c-gram name. 50616GC1.003
char new_ext[50];         // New extension for c-gram  "003"
char old_ext[50];           // Old extension for c-gram  "256"
char reportName[50];        // Contains the *.rpt file name "50616GC1.rpt"
char zeros[7] = "0.000";    // String to compare for bogus dif file header

// Temporary Character Variables
char temp[50];
char temp6[50];
char temp7[50];
char ttemp[50];
char old_gram_temp[50];
char chrom_path_final[50];

ini.open(ini_file, ios::in);          // Open the ini file for reading

for (j = 1; j <= 8; j++)                  // Skip the first seven lines processed
{                                     // in the main program
  ini >> col1 >> col2 >> col3;
}

strcpy(grampath, col3);             // Read grampath from ini file
ini >> col1 >> col2 >> col3;
strcpy(filePath, col3);             // Read filePath from ini file
ini >> col1 >> col2 >> col3;
strcpy(calPath, col3);              // Read filePath from ini file
ini.close();                    // Close the ini file

hFile = FindFirstChangeNotification(filePath, FALSE,
                                    FILE_NOTIFY_CHANGE_LAST_WRITE);

                                // Initialize the handle for the
                                // FindFirstChangeNotification Object
                                // This flag tells us when any file's
                                // last write time has changed in the
                                // path specified in filePath
                                // This is based on code from the
                                // Advanced Windows book: Pages 663-666
                                // (WIN32)

fil_name(old_date, gc_id);        // Fill old_date with the date
                                // old_date = Jun0195.
                                // (Developed function, gc1_fil_.cpp)

while (i == 1)                          // Infinite while loop for data collection
{

  SetLastError(0);            // Set Error code to zero
                        // to prevent carrythrough of
                        // any errors to other processes.
```

```
/*****************************************************************/
// Process the GC1.DIF file section
/*****************************************************************/

// Waiting for the gc1.dif file to be written....

AddStr(_TEXT("Beginning Wait for File Change"),0);


notif = FindNextChangeNotification(hFile); // Reset the waiting object flag
dw = WaitForSingleObject(hFile, 600000);    // Wait for 10 minutes for
                                            // flag to be triggered.
                                            // (Both functions WIN32)

if (dw==258)                                        // If the wait times out...
{
  strcpy(err_temp, outputPath);   // Copy outputPath into err_temp

  strcat(err_temp, err_chk);        // Concatenate err_chk onto
                                  // err_temp.
                                  // err_temp = d:\gc1\Jun0195\error.log

  chk.open(err_temp,ios::app);    // Open error.log file
  strcpy(error, err_temp2);       // Copy err_temp2 error code into
                                  // error.

  time_stp(time_stamp);             // Get the current time stamp.
                                  // (Developed function, time_stp.cpp)

  chk << time_stamp << "\t" << error << "\n";  // Output the timestamp and
                                            // and the error to error.log

  chk.close();                    // Close error.log

  AddStr(_TEXT("Instrument Problem"),0);        // Inform the user
  AddStr(_TEXT("Program Terminated"),0);
//  system("page 85-7019 GC problem on GCS");            // Page gas group member
  break;
} // End of timeout if loop

sleep(10);                                          // Wait 10 seconds for the
                                            // c-gram and dif files to
                                            // complete writing.

// Look here for whether calibration flag is up
// if it is set Cal = TRUE
findCal = FindFirstFile(calPath, &calData);   // Look for the calibration
                                            // file created by LabView
                                            // vi.

if(GetLastError() == 2)
{
  Cal = FALSE;                              // Calibration is not occuring

  FindClose(findCal);                       // Close calibration FindFile
                                  // handle.
}
```

```
else
{
  Cal = TRUE;                    // Calibration is in progress

  FindClose(findCal);                                          // Close calibration FindFile
                                                          // handle.
}


findFile = FindFirstFile(gcPath, &fileData); // See if the dif file exists
                                             // as specified in gcPath
                                             // (WIN32)
// Network problem check and exit
  if (findFile == INVALID_HANDLE_VALE && GetLastError() != 2)
  {
     AddStr(_TEXT("NetWork Problem"),0);          // Inform the user
     AddStr(_TEXT("Program Terminated"),0);
     FindClose(findFile);
//   system("page 85-7019 NetWork problem on GCS");      // Page gas group member
     break;
  }

if(GetLastError() == 2)   // If the dif file is not found...
{
  ListBox_DeleteString(g_hwndLB, (ListBox_GetCount(g_hwndLB)-1));
                          // Delete the previous string from the
                          // dialog text box.  This would be a
                          // second line saying
                          // "Beginning Wait for file change..."
}

else    // Enters when the dif file does exist.
{
  AddStr(_TEXT("File Has Changed"),0);
  FindClose(findFile);                         // Close the findFile handle
                                             // (WIN32)

  if(dif_chk(gcPath) == 0)        // If the dif file contains a
  {                             // bogus header
     dif_fix(gcPath);           // Run the strip out header routine
                          // (Developed function, dif_fix.cpp)
  }


     difIndex = ind_trns(gcPath);                  // Get the current index from
                                             // the dif file and place it
                                             // in difIndex.
                             // (Developed function, gc1_trns.cpp)

     reportIndex++;                  // Increment reportIndex

     indx(gcPath, reportIndex);          // Replace the current index
                                             // in the dif file with
                                             // reportIndex.
                             // (Developed function, gc1_indx.cpp)

     strcpy(temp7, temp3);               // Copy temp3 into temp7

     appnd(temp7, gcPath, Cal);              // Append the line from the
                                             // dif file into the
                                             // rpt file.
                           // (Developed function, appnd.cpp)
```

```
AddStr(__TEXT("Finished Appending #%d"), reportIndex);

DeleteFile(gcPath);             // When finished, delete the *.dif file
                        // (WIN32)

/*****************************************************************/
// Renaming of C-grams section
/*****************************************************************/

strcpy(C_gram_new, old_date);           // Copy old_date into C_gram_new

index_str(reportIndex, new_ext); // Convert reportIndex into the three-
                        // character extension for the new
                        // c-gram file.
                        // reportIndex = 3
                        // new_ext = "003"
                        // (Developed function, index3.cpp)

itoa(difIndex, old_ext, 10);    // Convert difIndex to a string for
                        // the old c-gram file.

strcat(C_gram_new,new_ext);     // Concatenate the new three-character
                        // extension onto C_gram_new
                        // C_gram_new = "Jun0195.003"

strcpy(old_gram_temp, C_gram_old);    // Copy C_gram_old into old_gram_temp
strcat(old_gram_temp,old_ext);  // Copy old_ext onto old_gram_temp
                        // old_gram_temp = "f:\gc1.256"

SearchPath(filePath, old_gram_temp, NULL, chPath, Cgrampath, Cpart);
                        // Get the full path for the old
                        // c-gram. (old_gram_temp)
                        // (WIN32)

strcpy(temp6, temp2);           // Copy temp2 into temp6
                        // temp6 = "Jun0195"

strcat(temp6, backs);           // Concatenate a backslash onto temp6
                        // temp6 = "Jun0195\"

strcpy(chrom_path_final, temp6);        // Copy temp6 into chrom_path_final
                        // chrom_path_final = "Jun0195\"

strcpy(ttemp, grampath);        // Copy grampath (from ini file)
                        // into ttemp.
                        // ttemp = "d:\gc1\"

strcat(ttemp, chrom_path_final); // Concatenate chrom_path_final
                        // onto ttemp.
                        // ttemp = "d:\gc1\Jun0195\"

strcpy(temp,ttemp);             // Copy ttemp into temp
                        // temp = "d:\gc1\Jun0195\"

strcat(temp, C_gram_new);       // Concatenate C_gram_new onto temp
                        // temp = "d:\gc1\Jun0195\Jun0195.003"

if (MoveFile(Cgrampath, temp) == 1)  // Move and rename c-gram from
{                       // F:\gc1.256 to
                                // d:\gc1\Jun0195\Jun0195.003"
                                // (WIN32)

}
```

```
else                                // if the MoveFile fails...
{
   AddStr(__TEXT("C-gram not generated"), 0);
}

/*****************************************************************/
// New Date Check Section
/*****************************************************************/

fil_name(new_date, gc_id);                          // Fill new_date with the current date
                                // (Developed function, gc1_fil_.cpp)

if(strncmp(old_date, new_date,5) != 0)              // If new_date and old_date
{                                                   // are different...

   strcpy(old_date, new_date);              // assign new_date to old_date
                                    // when date change


   fil_name(dirName, gc_id);                                // Fill dirName with the date
                                // (Developed function, gc1_fil_.cpp)

   strcpy(temp3, root);                     // Copy root into temp3
                                    // temp3 = "d:\gc1\"

   strncpy(temp2, dirName, 7);              // Copy first 7 characters of
                                    // dirName into temp2
                                    // temp2 = "Jun0295"

   strcat(temp3, temp2);                // Concatenate temp2 onto temp3
                                    // temp3 = "d:\gc1\Jun0295"

   CreateDirectory(temp3, NULL);    // Create a directory using
                                    // temp3 string.
                                    // (WIN32)

   strcat(temp3, backs);            // Concatenate a backslash onto
                                    // temp3.
                                    // temp3 = "d:\gc1\Jun0295\"

   strcpy(outputPath, temp3);       // Copy temp3 into outputPath
                                    // outputPath = "d:\gc1\Jun0295\"

   fil_name(reportName, gc_id);             // Fill reportName with the new
                                    // date.
                                    // reportName = "Jun0295."
                                // (Developed function, gc1_fil_.cpp)

   strcat(reportName, outputExt);   // Concatenate outputExt onto
                                    // reportName.
                                    // reportName = "Jun0295.rpt"


   strcat(temp3, reportName);   // Concatenate reportName onto
                                // temp3.
                                // temp3 = "d:\gc1\Jun0295\Jun0295.rpt"

   COPYGOOD = CopyFile(header, temp3, fFailIfExists);
                                // Copy c:\bin\header.h file
                                // to create the new report file
                                // indicated by temp3.
                                // (WIN32)
```

```
        if(COPYGOOD == TRUE)        // If the copy succeeds...
          {
            ListBox_ResetContent(g_hwndLB); // Reset the text box.

            AddStr(__TEXT("No file for today, %s is created"), outputPath);
            AddStr(__TEXT("Copied header.h to today's file"), 0);

            reportIndex = 0;                        // Set reportIndex to zero for
                                        // the new day.

            AddStr(__TEXT("The first index in today's file is %d"), reportIndex);
          }

          else                          // If the copy fails because the file exists
          {
            ListBox_ResetContent(g_hwndLB); // Reset the text box

            AddStr(__TEXT("Today's output file exists"), 0);

            reportIndex = rpt_trns(outputPath);       // Get current index from the
                                                // existing report file.
                                        // (Developed function, gc1_trns.cpp)

            AddStr(__TEXT("The transferred index is %d"), reportIndex);
          }

        } //End of New Day if statment

      } //End of found *.dif file if statment


    } //end of infinite while loop

    FindCloseChangeNotification(hFile);                                    // Close the file change
                                                        // object handle. (flag)
                                                        // (WIN32)

    // Return thread
    return(0);
}
```

F.4 APPND.CPP

```
/*****************************************************************************/
// Module: appnd.cpp
// This module takes one line from the *.dif file and appends it to
// the report output file.  It's also generated LabView interface file.
// By  Shawn Harden
//    Saikat Kanjilal
//    Russell S. Katz
//    Cuong Vo
// Version 1.0
// June 23, 1995
/*****************************************************************************/

// System Includes
#include <fstream.h>
#include <stdlib.h>
#include <iomanip.h>
```

```
void appnd(char *rptName, char *gc1_nam, int cal)
{                                   //rptName = output report file
                                    // gc1_nam = gc*.dif
                                    // cal = 1 or 0, depending on cal time

  int int_cal;              // Integer value for holding cal boolean

  char VI_interface_filename[30] = "C:\\GC2_VI.dat";  // LabView interface filename

  char  col0[3],                // Calibration column char variable
      col1[5],            // Month column char variable
      col2[5],            // Day column char variable
      col3[6],              // Year column char variable
      col4[11],             // Time column char variable
      index[4],             // Report index column char variable
      col6[10],             // First data column char variable
      col7[10],             // Second data column char variable
      col8[10],             // Third data column char variable
      col9[10],             // Fourth data column char variable
      space[3] = " ";       // Empty space char variable

// Declare the input and output files.

  if(cal == 1)              // If there is a calibration in progress
  {
    int_cal = 1;            // Int_cal is 1
    itoa(int_cal, col0, 10);  // Convert int_cal to char and put in col0
  }
  else                      // Otherwise
  {
    int_cal = 0;            // Int_cal is 0
    itoa(int_cal, col0, 10);  // Convert int_cal to char and put in col0
  }

  ifstream ifile;           // The input file stream pointer
  ofstream ofile;           // The output file stream pointer
  ofstream VIfile;          // The output VI_interface_filename stream pointer

  ifile.open(gc1_nam, ios::in);   // Open the input file for reading

  ifile >> col1 >> col2 >> col3 >> col4 >> index >> col6 >>
        col7 >> col8 >> col9;
                            // Reads in one line of the file by fields

  ifile.close();            // Close input file pointer

  VIfile.open(VI_interface_filename, ios::out); //Open file for writing only

  // Writes the dat to the io file by fields
  VIfile << col6 << "\n"
      << col7 << "\n"
      << col8 << "\n"
      << col9 << "\n";

  VIfile.close();           // Close ouput file pointer

  ofile.open(rptName, ios::app);   // Open the output file for appending

 // Writes the data to the io file by fields
 ofile << space << col0 << space << col1 << space << col2
          << space << col3 << space << col4
          << space << setw(7) << index
          << space    << setw(8) << col6
```

```
            << space << setw(8) << col7
            << space    << setw(8) << col8
            << space << setw(8) << col9 <<"\n";

// Close the output file.
    ofile.close();                  // Close output file pointer

} // End of appnd module
```

F.5 DIF_CHK.CPP

```
/*****************************************************************************/
// Module: Dif_chk.cpp
// This module takes a line from *.dif file and checks for whether it is the
// first *.dif of a run.  If it is, the *.dif file is deleted.
// By Russell S. Katz
// Version 1.0
// June 23, 1995
/*****************************************************************************/

// System Includes
#include <fstream.h>
#include <stdlib.h>

int dif_chk(char * dif_name)            // dif_name = *.dif file
{
  char  col1[5],              // Month column char variable
        col2[5],              // Day column char variable
        col3[6],              // Year column char variable
        col4[11],             // Time column char variable
        index[4],             // Report index column char variable
        col6[10],             // First data column char variable
        col7[10],             // Second data column char variable
        col8[10],             // Third data column char variable
        col9[10];             // Fourth data column char variable

// Declare the input file.
  ifstream ifile;             // *.dif file input stream pointer

  ifile.open(dif_name, ios::in);  // Open the *.dif file for reading

  ifile >> col1 >> col2 >> col3 >> col4 >> index >> col6 >>
        col7 >> col8 >> col9;
                              // Reads in one line of the file by fields

  ifile.close();              // Close the *.dif file pointer

  if(col6[0] > 57 || col6[0] < 48)    // If the first character of the first
  {                           // data column is not a numerical character...
    return(0);                    // Return a 0, indicating that the *.dif
  }                           // file included a header and should be deleted.

  else                            // Else...
  {
    return(1);                    // Return a 1, indicating a correct *.dif file
  }

} // End of dif_chk module
```

F.6 DIF_FIX.CPP

```
/*****************************************************************************/
// Module: Dif_fix.cpp
// This module takes the first *.dif file from a GC run and strips off
// the first header line.
// By Russell S. Katz
// Version 1.0
// July 11, 1995
/*****************************************************************************/

// System Includes
#include <fstream.h>
#include <stdlib.h>

void dif_fix(char * dif_name)          // dif_name = *.dif file
{
  char  headerLine[90];
  char  dataLine[90];


// Declare the input file.
  ifstream ifile;              // *.dif file input stream pointer
  ofstream ofile;              // *.dif file output stream pointer

  ifile.open(dif_name, ios::in);  // Open the *.dif file for reading

  ifile.getline(headerLine, 89);
  ifile.getline(dataLine, 89);

  ifile.close();               // Close the *.dif file pointer

  ofile.open(dif_name, ios::out);  // Open the *.dif file pointer for output

  ofile << dataLine << "\n";          // Overwrite the bogus header with data line

  ofile.close();                 // Close the dif file

} // End of dif_fix module
```

F.7 GC_FIL.CPP

```
/*****************************************************************************/
// Module: gc_fil.cpp, based upon TDC.CPP by Cliff Hewitt(LANL)
// This module takes an empy char array and an integer gc identifier
// and constructs a file/directory day string. (i.e. - 50623GC1.)
// By  Shawn Harden
//    Saikat Kanjilal
//    Russell S. Katz
//    Cuong Vo
// Version 1.0
// June 23, 1995
/*****************************************************************************/

//System include files
#include <stdlib.h>
#include <time.h>
#include <string.h>


void fil_name(char *sdat, int gc)      // sdat = empty date string
{                                      // gc = integer gc identifier
```

```
  //local variables
  char *tms;                          // Time string char array
  char str_gc[5];                     // Character variable holder for gc id
  int t;                              // Integer for holding month addition

/*****************************************************************************/
  time_t tm;                          // Time time variable
  size_t len;                         // Time size variable

  // Get Time String
  time(&tm);
  tms=ctime(&tm);
  len=strlen(tms);
  tms[len-1]=0;

  //sum char in month
  t=tms[4]+tms[5]+tms[6];

  //switch on month sum and create temp date string
  //month day year
  switch(t){
    case 281: sdat[1]=48; sdat[2]=49; break; //Jan
    case 269: sdat[1]=48; sdat[2]=50; break; //Feb
    case 288: sdat[1]=48; sdat[2]=51; break; //Mar
    case 291: sdat[1]=48; sdat[2]=52; break; //Apr
    case 295: sdat[1]=48; sdat[2]=53; break; //May
    case 301: sdat[1]=48; sdat[2]=54; break; //Jun
    case 299: sdat[1]=48; sdat[2]=55; break; //Jul
    case 285: sdat[1]=48; sdat[2]=56; break; //Aug
    case 296: sdat[1]=48; sdat[2]=57; break; //Sep
    case 294: sdat[1]=49; sdat[2]=48; break; //Oct
    case 307: sdat[1]=49; sdat[2]=49; break; //Nov
    case 268: sdat[1]=49; sdat[2]=50; break; //Dec
    default:  sdat[1]=48; sdat[2]=48; break; //ERR;
    }
/*****************************************************************************/
// The code above in the comment lines was taken from TDC.CPP by Cliff Hewitt,
// LANL

  itoa(gc, str_gc, 10);                     // Convert gc to char and place
                                            // in str_gc

  sdat[0] = tms[23];                  // sdat = " 06"
  sdat[3] = tms[8];                   // sdat = "5062"
  sdat[4] = tms[9];                   // sdat = "50623"
  sdat[5] = 71;                       // sdat = "50623G"
  sdat[6] = 67;                       // sdat = "50623GC"
  sdat[7] = str_gc[0];                // sdat = "50623GC1"
  sdat[8] = 46;                       // sdat = "50623GC1."
  sdat[9] = 0;                        // sdat = "50623GC1." with null
                                      //                                  terminator

  return;                             // Return filled sdat to main program

} // End of gc_fil module
```

F.8 GC1_INDX.CPP

```
/*****************************************************************************/
// Module: GC1_indx.cpp
// This module replaces the index in the *.dif file with the updated one
// computed by the main program
// By  Shawn Harden
//    Saikat Kanjilal
//    Russell S. Katz
//    Cuong Vo
// Version 1.0
// June 23, 1995
/*****************************************************************************/

// System Includes
#include <fstream.h>
#include <stdlib.h>
#include <iomanip.h>

int indx(char *difName, int intIndex)
{
  char  col1[5],            // Month column char variable
        col2[5],            // Day column char variable
        col3[6],            // Year column char variable
        col4[11],           // Time column char variable
        index[4],           // Report index column char variable
        col6[10],           // First data column char variable
        col7[10],           // Second data column char variable
        col8[10],           // Third data column char variable
        col9[10],           // Fourth data column char variable
        space[3] = " ";     // Empty space char variable

  fstream iofile;           // Input/Output file stream pointer

  iofile.open(difName, ios::in);  // Open the *.dif file for reading


  iofile >> col1 >> col2 >> col3 >> col4 >> index >> col6 >>
         col7 >> col8 >> col9;
                    // Reads in one line of the file in fields

  iofile.close();       // Close the *.dif file for reading

  itoa(intIndex, index, 10);   // Convert intIndex to char and place in index

  iofile.open(difName, ios::out);    // Open *.dif file for writing

// Writes the data to the *.dif file in fields
  iofile << col1 << space << col2
             << space << col3
             << space << col4
             << space << setw(7) << index          // setw() sets the width
             << space    << setw(8) << col6        // of the positon the char
             << space << setw(8)<< col7                  // variable is written to,
             << space    << setw(8) << col8        // allowing spacing.
             << space << setw(8) << col9 <<"\n";

    iofile.close();         // Close the *.dif file for writing

  return intIndex;                  // return intIndex to main program

} // End of gc1_indx module
```

F.9 GC1_TRNS.CPP

```
/*****************************************************************************/
// Module: GC1_trns.cpp
// This module reads in the entire *.dif file and retrieves the last index.
// By Shawn Harden
//    Saikat Kanjilal
//    Russell Katz
//    Cuong Vo
// Version 1.0
// June 23, 1995
/*****************************************************************************/

// System Includes
#include <stdlib.h>
#include <fstream.h>

int ind_trns(char fname[50])
{
  int indexOut,indexOut1;          // Output integer index variables

                              // Each of these variables will contain
                              // a field from the *.dif file.

  char  col1[5],              // Month column char variable
        col2[5],              // Day column char variable
        col3[6],              // Year column char variable
        col4[11],             // Time column char variable
        index[4],             // Report index column char variable
        col6[10],             // First data column char variable
        col7[10],             // Second data column char variable
        col8[10],             // Third data column char variable
        col9[10];             // Fourth data column char variable

  // Declare the file pointer.
  fstream infile;             // *.dif file input stream pointer

  //open the file
  infile.open(fname, ios::in);     // Open the *.dif file for reading

  // Start the infinite loop.
  while(!infile.eof())        // While we have not reached the end-of-file...
  {
    infile >>  col1 >> col2 >> col3 >> col4 >> index >> col6 >>
    col7 >> col8 >> col9;     // Read in a line of the file by fields.

    if(!infile.eof())         // If not the end-of-file yet
    {
      indexOut = atoi(index);     // indexOut is equal to the integer
                              // value of index
    }

    else                      // If it is the end-of-file
    {

      indexOut1 = atoi(index);   // indexOut1 is equal to the integer
                              // value of index
      ifstream::close;        // close the report file pointer
    }

  }
```

```
  if (indexOut==0)              // If indexOut is 0 (not read correctly)
  {
    return indexOut1;           // Return the value in indexOut1
  }

  else if (indexOut1==0)       // Else if indexOut1 is 0 (not read correctly)
  {
    return indexOut;            // Return the value in indexOut
  }

  else                         // Else if an error with both...
  {
    return 0;                  // Return a zero to indicate error
  }

} // End of gc1_trns module
```

F.10 INDEX3.CPP

```
/*****************************************************************************/
// Module: index3.cpp
// This module takes one filled integer and one empty character parameter, and
// converts the integer value to a three-character character value.
// (i.e. - int = 1 becomes char = "003"
// By Russell S. Katz
// Version 1.0
// June 23, 1995
/*****************************************************************************/

// System includes
#include <stdlib.h>
#include <string.h>

void index_str(int int_index, char *ch_index)          // int_index: filled int param.
{                                                      // ch_index: empty char param.
  int hundreds;                  // Integer hundreds digit holder.
  int tens;                      // Integer tens digit holder
  int ones;                      // Integer ones digit holder

  char hund[2];                  // Character hundreds digit holder
  char tns[2];                   // Character tens digit holder
  char ouns[2];                  // Character ones digit holder
  char temp[20];                 // Character temporary variable to hold partial
                                 // string while it is constructed

  hundreds = (int_index - ((int_index/1000)*1000))/100;          // Mathmatically
                                                                 // strip off
                                                                 // hundreds digit

  tens = ((int_index - ((int_index/1000)*1000))- (hundreds * 100))/10;
                                                                 // Mathmatically
                                                                 // strip off tens
                                                                 // digit

  ones = (((int_index - ((int_index/1000)*1000)) - (hundreds * 100))
                                                                 - (tens * 10));
                                                                 // Mathmatically
                                                                 // strip off
                                                                 // ones digit

  itoa(hundreds, hund, 10);      // Convert hundreds digit to char
  itoa(tens, tns, 10);           // Convert tens string to char
  itoa(ones, ouns, 10);          // Convert ones string to char

  strcpy(temp, hund);            // Copy hundreds char into temp
  strcat(temp, tns);             // Concatenate tens char into temp
  strcat(temp, ouns);            // Concatenate ones char into temp
  strcpy(ch_index, temp);        // Copy temp into ch_index


  return;                        // return to main program

} // End of index3 module
```

F.11 RPT_TRNS.CPP

```
/*****************************************************************************/
// Module: Rpt_trns.cpp
// This module reads in the entire *.rpt for the current day, if it exists,
// file  and get the last index.  This module is used if the gcapp has been
// run earlier in the day, stopped, and then restarted.
// By Russell Katz, adopted from gc1_trns.cpp
// Version 1.0
// June 23, 1995
/*****************************************************************************/

// Systems Includes
#include <stdlib.h>
#include <fstream.h>

int rpt_trns(char fname[50])
{
  int indexOut,indexOut1;          // Output integer index variables

                              // Each of these variables will contain
                              // a field from the *.rpt file.

  char  col0[3],                  // Calibration column char variable
      col1[5],            // Month column char variable
      col2[5],            // Day column char variable
      col3[6],                // Year column char variable
      col4[11],               // Time column char variable
      index[4],               // Report index column char variable
      col6[10],               // First data column char variable
      col7[10],               // Second data column char variable
      col8[10],               // Third data column char variable
      col9[10];               // Fourth data column char variable

  // Declare the file pointer.
  fstream infile;                 // Report file input stream pointer

  //open the file
  infile.open(fname, ios::in);     // Open the report file for reading

  // Start the infinite loop.
  while(!infile.eof())            // While we have not reached the end-of-file...
  {
    infile >>  col0 >> col1 >> col2 >> col3 >> col4 >> index >> col6 >>
    col7 >> col8 >> col9;       // Read in a line of the file by fields.

    if(!infile.eof())            // If not the end-of-file yet
    {
      indexOut = atoi(index);     // indexOut is equal to the integer value of index
    }

    else                      // If it is the end-of-file
    {

      indexOut1 = atoi(index);   // indexOut1 is equal to the integer value of index
      ifstream::close;         // close the report file pointer
    }

  }
```

```
  if (indexOut==0)              // If indexOut is 0 (not read correctly)
  {
    return indexOut1;           // Return the value in indexOut1
  }

  else if (indexOut1==0)       // Else if indexOut1 is 0 (not read correctly)
  {
    return indexOut;            // Return the value in indexOut
  }

  else                         // Else if an error with both...
  {
    return 0;                   // Return a zero to indicate error
  }

} // End of rpt_trns module
```

F.12 TIME_STP.CPP

```
/*****************************************************************************/
// Module: Time_stp.cpp
// This module creates a time stamp based on today's date & time
// By Russell S. Katz, based on code TDC.cpp from Cliff Hewitt, LANL
// Version 1.0
// June 23, 1995
/*****************************************************************************/

// System Includes
#include <string.h>
#include <time.h>

void time_stp(char *tm_stp)
{
  // Local Variables
  char * tms;                   // Time string char variable
  time_t tm;                    // Time time variable
  size_t len;                   // Time size variable

  // get time string
  time(&tm);
  tms=ctime(&tm);
  len=strlen(tms);
  tms[len-1]=0;

  //create output file name for the day
  tm_stp[0]=tms[0];             // "F"
  tm_stp[1]=tms[1];             // "Fr"
  tm_stp[2]=tms[2];             // "Fri"
  tm_stp[3]=tms[3];             // "Fri "
  tm_stp[4]=tms[4];             // "Fri J"
  tm_stp[5]=tms[5];             // "Fri Ju"
  tm_stp[6]=tms[6];             // "Fri Jun"
  tm_stp[7]=tms[7];             // "Fri Jun "
  tm_stp[8]=tms[8];             // "Fri Jun 2"
  tm_stp[9]=tms[9];             // "Fri Jun 23"
  tm_stp[10]=tms[10];           // "Fri Jun 23 "
  tm_stp[11]=tms[11];           // "Fri Jun 23 0"
  tm_stp[12]=tms[12];           // "Fri Jun 23 08"
  tm_stp[13]=tms[13];           // "Fri Jun 23 08:"
  tm_stp[14]=tms[14];           // "Fri Jun 23 08:3"
```

```
  tm_stp[15]=tms[15];          // "Fri Jun 23 08:32"
  tm_stp[16]=tms[16];          // "Fri Jun 23 08:32:"
  tm_stp[17]=tms[17];          // "Fri Jun 23 08:32:0"
  tm_stp[18]=tms[18];          // "Fri Jun 23 08:32:06"
  tm_stp[19]=tms[19];          // "Fri Jun 23 08:32:06 "
  tm_stp[20]=tms[20];          // "Fri Jun 23 08:32:06 1"
  tm_stp[21]=tms[21];          // "Fri Jun 23 08:32:06 19"
  tm_stp[22]=tms[22];          // "Fri Jun 23 08:32:06 199"
  tm_stp[23]=tms[23];          // "Fri Jun 23 08:32:06 1995"
  tm_stp[24] = 0;            // "Fri Jun 23 08:32:06 1995" with null
                        //                                        terminator

  return;                    // Return to main program

} // End of time_stp module
```

F.13 GC2APP32.H

```
// Program functions
void dif_fix(char *);
int rpt_trns(char[]);
int dif_chk(char *);
void index_str(int, char[]);
void time_stp(char *);
void AddStr (LPCTSTR, ...);
void appnd(char *, char *, BOOL);
void fil_name(LPSTR, int);
int indx(char *, int);
int ind_trns(char[]);
```

F.14 HEADER.H

```
Cal Mo Day Yr    TIME       INDEX  CH4Low   N2OLow   CH4Hi    N2OHi
```

F.15 RESOURCE.H

```
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by Alertl0.rc
//
#define IDD_GC                  1
#define IDC_LOG                     103
```

F.16 GC2APP32.RC

```
//TEST2.RC

#include "resource.h"

gc2app32 ICON "gc2app32.ico"

IDD_GC DIALOG DISCARDABLE 6, 110, 161, 75
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU | WS_MINIMIZEBOX
CAPTION "GC2 Data Acquisition Application"
FONT 8, "MS Sans Serif"
{
 DEFPUSHBUTTON "&Run", IDOK, 13, 2, 50, 14
 DEFPUSHBUTTON "&Quit", IDCANCEL, 98, 2, 50, 14
 LTEXT "Execution &Log:", IDC_STATIC, 5, 21, 60, 8
 LISTBOX IDC_LOG, 5, 32, 150, 38, NOT LBS_NOTIFY | WS_BORDER |
        LBS_NOINTEGRALHEIGHT | WS_BORDER | WS_VSCROLL | WS_TABSTOP

}
```

APPENDIX G -- C++ SOURCE CODE LISTINGS for FTIR Routines

G.1 FTIRZIP.INI

```
data_files_path = d:\ftir\
zip_file_path   = H:\ftir\

-------------------------

This ftir_zip.ini is used with latest
FTIRZIP.EXE
```

G.2 FTIRZIP.CPP

```
// Beta Version 1.1 10/18/95
// Last Mode, Feb 1996
// Author: Saikat Kanjilal
//        Cuong Vo
// Main Program composes of four purposes:
// (1) Calls ftir function (i.e gc1_ftir module) to generate proper string for
// zipping purposes(typical string is of the form YMMDDIR.ZIP).
// (2) Send report file before zipping files to remote location.
// (3) Removes the local directory and files where the files that are to be
// zipped are initially located.
// (4) Saves data files locally when remote location is not available
// i.e. in the case of network failure occure.


// Header Files
#include <stdio.h>
#include <stdlib.h>
#include <iostream.h>
#include <dos.h>
#include <time.h>
#include <cstring.h>
#include <fstream.h>
#include <string.h>
#include <windows.h>

// declare external function which uses to generate day string (YMMDD) and
// complete zipped string (YMMDDIR.ZIP),  i.e. ftir(YMMDD,YMMDDIR.ZIP)
// where: Y    MM   DD     IR.          ZIP
//      Year Month Day  Instrument_name  Extention
void ftir(char *,char *);

void main(void)
{

// Define Local Variables
   char FTIR_ZIP_INI_path[30]="C:\\BIN\\FTIR_ZIP.INI \0";
   char pkzip_call[60]="CALL C:\\BIN\\PKZIP.EXE -m \0";
   char dir_day_filename[15];
   char zip_filename[15];

   char data_files_path[30];
   char zip_file_path[30];

   char ExistingFile1[30];
   char NewFile1[30];
   char ExistingFile2[30];
   char NewFile2[30];
```

```
  char Zip_local_area[30];

  char col1[10];
  char col2[3];
  char col3[25];

  BOOL CHECK;

  WIN32_FIND_DATA fileData; // FindFirstFile file data structure
  HANDLE checkFile;         // FindFirstFile file handle

// get string for dir_day_filename = YMMDD, zip_filename = YMMDDIR.ZIP
// from external function call ftir
  ftir(dir_day_filename,zip_filename);
  printf("dir_day_filename = %s\n",dir_day_filename);
  printf("zip_filename     = %s\n",zip_filename);

// get data_files_path string and zip_file_path string from FTIR_ZIP.INI file.
  ifstream FTIR_ZIP_INI_open(FTIR_ZIP_INI_path);
  //  FTIR_ZIP.INI file exist? error trapping
  if (!FTIR_ZIP_INI_open)
    printf("Error at FTIR_ZIP.INI file \n");
  FTIR_ZIP_INI_open >> col1 >> col2 >> col3;
  strcpy(data_files_path,col3);
  FTIR_ZIP_INI_open >> col1 >> col2 >> col3;
  strcpy(zip_file_path,col3);
  printf("data_files_path = %s\n",data_files_path);
  printf("zip_file_path = %s\n",zip_file_path);

// contruct the appropriate strings to copy report file
// from local area (i.e \localpath\YMMDDIR.RPT) to a remote
// network area (i.e. \remotepath\YMMDDIR.RPT)
  strcpy(ExistingFile1,data_files_path);
  strcat(ExistingFile1,dir_day_filename);
  strcat(ExistingFile1,"IR\\\0");
  strcat(ExistingFile1,dir_day_filename);
  strcpy(ExistingFile2,ExistingFile1);  // ExistingFile2 = ExistingFile1
                                        // preserved for future use
  strcat(ExistingFile1,"IR.RPT\0");
  strcpy(NewFile1,zip_file_path);
  strcat(NewFile1,dir_day_filename);
  strcpy(NewFile2,NewFile1);            // NewFile2 = NewFile1
                                        // preserved for future use
  strcat(NewFile1,"IR.RPT\0");
  printf("ExistingFile1 = %s\n",ExistingFile1);
  printf("NewFile1 = %s\n",NewFile1);
  CHECK    = CopyFile(ExistingFile1,NewFile1,FALSE);
  if (CHECK == FALSE)
    printf("Error at transfer report file \n");

// construct the appopriate strings to compress data, where the data located
// under the previous day's directory, to a local temporary location under
// the D:\ root directory. (i.e. pkzip to D:\YMMDDIR.ZIP from \localpath\*.*)
  strcpy(Zip_local_area,"D:\\\0");
  strcat(Zip_local_area,zip_filename);
  strcat(pkzip_call,Zip_local_area);
  strcat(pkzip_call," \0");
  strcat(pkzip_call,ExistingFile2);
  strcat(pkzip_call,"*.*");
  printf("pkzip_call = %s\n",pkzip_call);
  system(pkzip_call);

// construct the appropriate strings to transfer compressed files from
```

```
// local area (i.e. D:\YMMDDIR.ZIP) to a remote network area.
  strcat(NewFile2,"IR.ZIP\0");
  printf("Zip_local_area = %s NewFile2 = %s\n",Zip_local_area,NewFile2);
  CopyFile(Zip_local_area,NewFile2,FALSE);

// check remote network area for transfer completion of the compressed files
// (i.e. network awareness problems)
  checkFile = FindFirstFile(NewFile2,&fileData);
  if(checkFile != INVALID_HANDLE_VALUE)
  {
    DeleteFile(Zip_local_area);
    FindClose(checkFile);
    printf("Mission accomplish \n");
  }
  else
  {
    FindClose(checkFile);
    printf("Network problem, save file locally \n");
  }
  sleep(60);
  return;
}
```

G.3 FILENAME.CPP

```
// This module creates the string 50719IR.zip

// Header Files
#include <stdlib.h>
#include <stdio.h>
#include <iostream.h>
#include <dos.h>
#include <time.h>
#include <cstring.h>
#include <fstream.h>
#include <string.h>

void ftir(char day1i15], char fname1[10])

{
// Local Variables
char *tms;
char fname[5],fnm[5];
char month1[5];
char m1[5];
char sp[5],sp1[5],sp2[5],sp3[5];
char i1[5],i2[5];
char day[14],temp[10];
//int i_day;
int  month;


// The months of the year are stored in arrays
char space0[5]="Jan\0";
char space1[5]="Feb\0";
char space2[5]="Mar\0";
char space3[5]="Apr\0";
char space4[5]="May\0";
char space5[5]="Jun\0";
char space6[5]="Jul\0";
char space7[5]="Aug\0";
char space8[5]="Sep\0";
char space9[5]="Oct\0";
char space10[5]="Nov\0";
char space11[5]="Dec\0";

// A zero is defined now and later catenated onto the begining of the month
m1[0]='0';
m1[1]=0;

// get time string
time_t tm;
size_t len;
time(&tm);
tms=ctime(&tm);
len=strlen(tms);
tms[len-1]=0;

//The line below assigns the year to fname1;
//i.e if the year is 95 then 5 is assigned to fname1
fname1[0]=tms[23];
fname1[1]=0;

// The next four lines retrieve the string equivalent of the month
// i.e if the month is June the letters J,u,n are stored in fname.
// A null terminator is then added to the end of this string.
```

```
// fname is just an array of chars
fname[0]=tms[4];
fname[1]=tms[5];
fname[2]=tms[6];
fname[3]=0;


// The next set of int and if statements compare the fname
// functions defined earlier to the character arrays called space that
// were defined.  If there is a match the month is
// assigned its appropriate integer value.
int a=strncmp(space0,fname,3);
if (a==0) month=1;
int b=strncmp(space1,fname,3);
if   (b==0) month=2;
int c=strncmp(space2,fname,3);
if (c==0) month=3;
int d=strncmp(space3,fname,3);
if (d==0) month=4;
int e=strncmp(space4,fname,3);
if (e==0) month=5;
int f=strncmp(space5,fname,3);
if (f==0) month=6;
int g=strncmp(space6,fname,3);
if (g==0) month=7;
int h=strncmp(space7,fname,3);
if (h==0) month=8;
int i=strncmp(space8,fname,3);
if (i==0) month=9;
int j=strncmp(space9,fname,3);
if (j==0) month=10;
int k=strncmp(space10,fname,3);
if (k==0) month=11;
int l=strncmp(space11,fname,3);
if (l==0) month=12;


// The next three lines get the integer equivalent of the day from the time string.
// i.e if the day is less than 10 a zero is assigned to fname3
// automatically.
day[0]=tms[8];
day[1]=tms[9];
day[2]=0;
printf("day = %s\n",day);
int i_day = atoi(day);

// if day of month is the first day do the following

if(i_day == 1)
    {
      int int_year = atoi(fname1);  //  Convert the integer equivalent
                                    //  of the year to an ascii format.

      switch(month)                 // A switch statement is used to differentiate
                                     // between all the months of the year.
                                     // A break indicates the end of each case statement.

      {                         // Starting parenthesis for switch statement.
        case 1:                   // If starting day of month=1
        int_year--;               // Decrement the year by one
        month=12;                   // Change the month to December
        day[0] = '3';             // Change day to 31 and add a null terminator
        day[1] = '1';
```

```
day[2] = 0;
i_day=atoi(day);          // Convert the day from an ascii to an integer
itoa(int_year,fname1,10);  // Convert the year back to an ascii string
break;

case 3:                  // If the month is March do the following
   fnm[0] ='0';           // Decrement the month to February and add
   fnm[1] ='2';           // a null terminator to the end
   fnm[2]=0;
   day[0] ='2';           // Change the day to the last day of february
   day[1] ='8';           // and add a null terminator to the end.
   day[2]=0;
   i_day=atoi(day);        // Convert the day from an ascii to an integer
   month=2;               // Change the month t February

   if (((int_year+1990)-1996) % 4 = = 0)   // If its a leap year do the following
   {
    fnm[0] ='0';          // Change the month to February
    fnm[1] ='2';
    fnm[2]=0;

    day[0] ='2';          // Change the last day of the month to the 29th of Feb
    day[1] ='9';          // and add a null terminator to the end.
    day[2]=0;
    i_day=atoi(day);       // Change the day to an integer
    month=2;              // Change the month to February
   }

   break;

case 5:                  // If the month is May do the following
   fnm[0] = '0';          // Decrement the month to April
   fnm[1] = '4';          // and add a null terminator to the end
   fnm[2]=0;
   day[0] = '3';          // Change the date to the 30th and
   day[1] = '0';          // add a null terminator to the end since
   day[2]=0;              // April only has 30 days
   i_day=atoi(day);        // Convert the day back to an integer
   month=4;               // Change the month back to April
   break;
case 7:                  //  If the month is July
   fname[0] = '0';         //  Decrement the month to June
   fname[1] = '6';         //  and add a null terminator to the end
   fname[2]=0;
   day[0] = '3';          //  Change the date of the month back to 30th
   day[1] = '0';          //  and add a null terminator to the end
   day[2]=0;
   i_day=atoi(day);        // Change the day from an ascii to an integer
   month=6;               // Change the variable month to the proper
                           // integer value for June
   break;

case 10:                 // If the month is October do the foollowing
   fname[0] = '0';         // Change the month back to September
   fname[1] = '9';         // and add a null terminator to the end
   fname[2]=0;
   day[0] = '3';          // Change the date of the month back to 30th
   day[1] = '0';          // and add a null terminator to the end
   day[2]=0;
   i_day=atoi(day);        // Change the day from an ascii to an integer
   month=9;               // Change the variable month to the proper
                           // integer value for September
   break;
```

```
        case 12:                // If the month is December do the following
           fname[0]= '1';          // Change the month to November and add a
           fname[1]= '1';          // null terminator to the end
           fname[2]=0;
           day[0]='3';             // Change the date of the month back to 30th
           day[1]='0';             // and add a null terminator to the end
           day[2]=0;
           i_day=atoi(day);        // Change the day from an ascii to an integer equivalent
           month=11;               // Change the variable month to the proper
                                   // integer value for October
           break;

        default:                // In case month=any value other than the ones
                                   // mentioned above do the following
           month --;               // Decrement the month by one
           if (month <10)          // If the month is less than October do the following
           {
            fname[0]='0';          // Attach a zero to the first part of the
                                   // string of month
            itoa(month, temp, 10); // Convert the month to an ascii variable temp
            fname[1]= temp[0];     // The second part of the string of month
                                   // gets assigned an integer value here
            day[0] = '3';          // Day = 31 plus null terminator
            day[1] = '1';
            day[2] = 0;
            i_day=atoi(day);       // i_day equals the integer equivalent of day
           } // ends most recent if statement

           else       // If month is greater than October
           {
              itoa(month, temp, 10);  // Convert month to an ascii variable temp
              fname[0] = temp[0];     // Convert temp back into fname
              fname[1] = temp[1];
              day[0] = '3';           // Change day to 31 and add null terminator
              day[1] = '1';
              day[2] = 0;
           }
           break;

     }  // End of month switch

   } // End of 1st of month if



else if (i_day <= 10)              // If int_day is less than ten,
     {                        // convert it to char and place
        i_day--;                // a '0' in front of it.
        itoa(i_day,temp,10);
        day[0]='0';
        day[1]= temp[0];
        day[2]= 0;

     } // Ends if loop

else                           // If day of month is not equal to one
{                            // and greater than 10
         i_day--;              // Decrement the day by one
         itoa(i_day,day,10);  // Convert int_day to its ascii equivalent
         day[2]=0;             // Add a null terminator to day
} // Ends else
```

```
// The integer month is converted to a string.
// The new variable is termed month1
itoa(month,month1,10);


// The year, the month and the day are catenated together and
// a .spc is catenated onto the end of this expression.
if (month<=9)  {        // If month is less than or equal to September do the following
 // A zero is catenated onto the begining of the integer month obtained
 //  previously.
 strcat(m1,month1);
 strcat(fname1,m1);
 }
else  {                 // If month is greater than September do the following
 strcat(fname1,month1);  // stringcatenate month1 with fname1
 }                       // final string = 509 or 511 etc


 strcat(fname1,day);     // string (fname1)= 50921 etc
 strcpy(day1,fname1);    // day1 = fname1
 //printf("day1 in gc1_ftir=%s\n", day1);


// Attach the words IR to 50607
i1[0]='I';
i1[1]=0;
i2[0]='R';
i2[1]=0;
strcat(fname1,i1);
strcat(fname1,i2);

// Create the .zip extension
sp[0]='.';
sp[1]=0;
sp1[0]='z';
sp1[1]=0;
sp2[0]='i';
sp2[1]=0;
sp3[0]='p';
sp3[1]=0;

strcat(sp,sp1);
strcat(sp,sp2);
strcat(sp,sp3);
strcat(fname1,sp);  // attach the .zip extension  to the date


//printf("The final string in gc_ftir=%s\n", fname1);
//sleep(23);
return;
}
```

APPENDIX H -- ARRAY BASIC PROGRAM LISTINGS

H.1 ICOMPUT.AB

```
' Interferogram   compute with selectable    zero filling and apodization functions.
' Copyright (c)1993 Galactic Industries   Corp. Copy only          for use with GRAMS(tm).

' Box car, Triangular, Norton-Beer, Happ-Genzel            and Bessel apodization functions
' for computing   interferograms.    Must be applied to a raw interferogram  in the #S
' slot.    Asks if user wants to perform ratio with a background (#B) single beam
' trace   (%T, ABS, & K-Munk).

PROCEDURE
  free : pauseoff  : onpaint 0
  portout    -44,1 :   i=#s(0)          'test if trace in mem
  dim name(256),bname(256),temp(256),temp2(256),temp3(256)

string $temp3=^"^$[Sampling]"
  string $temp2="UDR-2"
  mc=sum(temp3(0,4)-temp2(0,4))
  if mc=0    then udr=0 else   udr=1
  if getxtype()<>255 then   goto 5

 'if bidir get gainrange stuff
  string $temp3=^"^$[Gain_Range_Radius]"
  string $temp3,-20
  GRR=v
  string $temp3=^"^$[Count1]"
  string $temp3,-20
  Count1=v
  string $temp3=^"^$[Count2]"
  string $temp3,-20
  Count2=v
  string $temp3=^"^$[Count3]"
  string $temp3,-20
  Count3=v
  string $temp3=^"^$[Count4]"
  string $temp3,-20
  Count4=v
  string $temp3=^"^$[Gain_Range_Slope]"
  string $temp3,-20
  Slope=v
  string $temp3=^"^$[Gain_Range_Offset]"
  string $temp3,-20
  Offset=v
  total=count2+count3+count4

5 string bname,-37          'get #B   background name
  if sum(bname)=0  then type=3
  fbeg=4000 : fend=400
  ffp=getffp() : flp=getflp()
  if ffp<flp then  mc=flp:flp=ffp:ffp=mc
  np=npts(#s):zf=0:ap=0

  type = 1
  ZF = 0
  ap = 0
  goto 50

'****************
'Main menu dialog
'****************
```

```
10  dialogon "Interferogram Compute"
  dialogloc 1,-130,2  : dialogtxt  3,280,50,"Spectral Data"
  dialogloc 1,-125,18 : dialogask     fbeg,0+256,flp,ffp,"Range (CM-1):  &First"
  dialogloc 1,-10,18  : dialogask fend,0+256,flp,ffp,"&Last "
  dialogloc 1,60,18   : dialogask type,3+128,0,0,"Type: Transmission Absorbance Kubelka-Munk Single_Beam"
  dialogloc 1,-125,34 : dialogask     bname,2,35,255,"Background"
  dialogloc 1,70,34   : dialogask mc,6+128,40,0,"Change_Background"
  dialogloc 1,-130,56 : dialogtxt     3,280,30,"Processing Parameters"
  dialogloc 1,-110,70 : dialogask     zf,3,0,0,"Zero_fill: None 2x 4x 8x 16x 32x 64x 128x"
  dialogloc 1,10,70   : dialogask ap,3,0,0,"Apodization: Triangular Boxcar Weak_Norton-Beer Medium_Norton-Beer
Strong_Norton-Beer Happ-Genzel Bessel Cosine"
  yl=88
  if nsubs(#s)=1 goto 20
  dialogloc 1,-130,yl : dialogtxt  3,280,30,"Multifile Options"
  dialogloc 1,-110,yl+16 : dialogask ow,3+64,0,0,"Save_multifile_to_disk: New Overwrite"
  yl=yl+34
20  dialogloc 3,0,yl+4 : dialogask mc,4,50,0,"&OK |Cancel"
25  dialogoff 11,0,30

'Grey #B name for SB type
30  if type=3 then dialogset 16,5 :    dialogset 16,6
  if type<>3 then   dialogset 8,5 :    dialogset 8,6
  goto 25

'Get new #B file name
40  openbgd    -1 : string bname,-37 :   goto 25

'Set up  last minute extras
  dialogoff -10
50  if type<>3 then     onerror     51 : open "O",#1,$bname      : close   #1 : onerror -1    'valid #B?
  string bname,-38          'set background        name
  if type<>3 then   openbgd  1      'load #B in mem, if possible
  if fbeg<fend then mc=fend:fend=fbeg:fbeg=mc
  setytype 1:if getffp0>getflp0    then xflip
  if (ap>=2 and ap<=4) then dim c(4)              'Norton-Beer coeffs
  if nsubs(#s)>1 goto 60
  goto 55

51  dialogon
  print "No reference file selected. Cannot compute."
  print " "
  dialogask mc,4,10,0,"__&OK__"
  dialogoff
  goto 10

'*******************
'Single  file compute
'*******************
'Does compute in place in #S mem slot. Replaces          the slot with computed
'data, but does not save to disk.
55  waiton
  dialogon "Interferogram Compute"
  print "Computing....."
  print " "
  dialogoff 20
  if getxtype0=255 then goto 59 '>>>>>testz
  gosub 100
  setleft   getffp0,getflp0 : autoscale
  if getleft0<getright0    then xflip
  if type<>3 then   openbgd  0      'close #B
  dialogoff -20
  waitoff
  if zf=0  then return            'no resolution change
```

```
  xres=rnd(xres)*2
  temp=0
  string temp=xres : string temp=$temp+"cm-1"
  'end
  return

'bidirectional background compute

58 if type<>3 then    openbgd  0:erroresc "Cannot ratio this type of file (bidirectional IGRAM)."

  string $temp2=^"^$[Igram_Symmetry]"
  string $temp3,"Double Sided"
  !mov a,#($temp2,$temp3)        'compare characters
  !if a<>#-1:!jmp 59           'double sided?

  fileslot=getsfile()
  fileslot=fileslot+1

  shalf=freq(#s(#((npts(#s)/2)-1)))
  shalf2=Freq(#s(#(index(#s(shalf))+1)))
  dim buff2(npts(#s)/2)
  buff2=#fileslot(flp,shalf)
  min=min(buff2)
  newspc ttt_(npts(#s)/2)  'temp=first half
  setffp flp,shalf
  #s=#fileslot(flp,shalf)
  setffp flp,ffp
  setfile   getsfile()+1  'data=second half
  #s=#s(shalf2,ffp)
  setffp flp,ffp
  reverse  #s
  gosub 100
  setfile   getsfile()-1    'temp=first
  gosub 100
  sffp=getffp()
  sflp=getflp()
  dim buff_(npts(#s))
  buff_=#s       'save temp=first in buffer
  noshow         'and cream old file
  newspc ttt_(npts(#s)*2)  'make new one big enough to hold both
  snp=(npts(#s)/2)
  enp=npts(#s)-1
  #s=buff_       'copy temp=first to new
  setfile   getsfile()+1  'data=second half
  setffp sffp,sflp
  buff_=#s       'get second half of data in buffer
  setfile   getsfile()-1    'temp
  #s(#snp,#enp)=buff_
  setffp sffp,sflp
  #fileslot=#s
  noshow
  end

' Bidirectional   doublesided compute
59 fileslot=getsfile()
  fileslot=fileslot+1

  slast=npts(#s)-1
  shalf=(slast+1)/2
  s1=int (shalf/2+134)
  s2=int (shalf/2-135)
  s3=(shalf+s1)
  s4=(shalf+s2)
```

```
  dim buff(npts(#s))
  buff=#s

  newspc ttt_(s1+1)      'first igram
  #s=buff(0,s1)
  reverse  #s
  setffp flp,ffp
  gosub 100
  sffp=getffp()
  sflp=getflp()
  ssize=npts(#s)*4
  dim buff2(ssize)       'buffer big enough to hold all data
  buff2=#s:ptr=ptr+npts(#s)

  noshow:newspc ttt_(s1+1)
  #s=buff(s2,shalf-1)        'second  igram
  setffp flp,ffp
  gosub 100
  buff2(ptr,ptr+npts(#s))=#s:ptr=ptr+npts(#s)

  noshow:newspc ttt_(s1+1)
  #s=buff(shalf,s3)         'third igram
  setffp flp,ffp
  reverse  #s
  gosub 100
  buff2(ptr,ptr+npts(#s)-1)=#s:ptr=ptr+npts(#s)

  noshow:newspc ttt_(s1+1)
  #s=buff(s4,slast)         'forth igram
  setffp flp,ffp
  gosub 100
  buff2(ptr,ptr+npts(#s))=#s
  noshow:free buff

'now stick back    together in on file
  newspc ttt_(ssize)
  setffp sffp,sflp
  #s=buff2:free buff2
  #fileslot=#s
  noshow
  setleft   sflp,sffp
  autoscale
  end


'*****************
'Multifile compute
'*****************
'Does compute to a multifile.  If NEW was selected, writes directly to
'new file name.     If OVERWRITE was selected, creates temporary multifile        in
'same directory, deletes original file,    and renames temporary to original.

60  ns=nsubs(#s)
  if ow=1    goto 70

'New multifile
62  menufile temp,"Name for New Multi-file",1  'Save as mode
  onerror 65 : open "O",#1,$temp : onerror -1 : close #1
  dialogon "Error"
  print "File ";$temp;" already exists."
  print " "
  dialogask mc,4,65,62,"Overwrite Retry ¦Cancel"
  dialogoff
```

```
65  string name,-1
  if sumd2(temp,name)=0 goto 70       'Same file name, Overwrite!
  kill $temp              'Kill temp name, just in case.
  goto 80

'Overwrite
70  string name,-1 : temp=name      'get #$      name
  reverse  temp : i=255-index0(temp-92)          'parse out path
  reverse  temp : temp(i+1,255)=0
  string temp=$temp+"_ZTMP.SPC"     'name for temp multifile
  kill $temp              'Kill temp name, just in case

'Do it!
80  waiton
  portin -140,flags
  copy=getsfile0 : orig=copy+1
  if getxtype0=255 then goto 81      'Bidirectional?
  noshow

  dialogon "Multifile Interferogram Compute"
  dialogask i,1+256,0,0,"Computing subfile number"
  dialogoff 20,0,0
  for i=1  to ns
    dialogoff 20              'update status counter
    loadspc $name,#i
    gosub  100
    savespc $temp,#i
    noshow
  next i
  goto 98

'bidirectional compute
81  string $temp2=^"^$!Igram_Symmetry!"
  string $temp3,"Double Sided"
  !mov a,#($temp2,$temp3)          'compare characters

  !if a<>#-1:!jmp 88            'double  sided?
  noshow
  dialogon "Multifile Interferogram Compute"
  dialogask j,1+256,0,0,"Computing subfile number"
  dialogoff 20,0,0
  digram=1
  loadspc  $name,#1
  sfileslot=getsfile0
  if type=3 then goto 82
  bfileslot=getbfile0
  setfile    bfileslot

  string $temp2=^"^$!Scanning!"
  string $temp3,"Bidirectional"
  !mov a,#($temp2,$temp3)          'compare characters
  !if a=#-1:openbgd 0:dialogoff -20:erroresc "Background is not a Bidirectional collect."
  string $temp2=^"^$!Igram_Symmetry!"
  string $temp3,"Single Sided"
  !mov a,#($temp2,$temp3)          'compare characters
  !if a=#-1:openbgd 0:dialogoff -20:erroresc "Background is not a Single Sided collect."

  bffp=getffp0
  bflp=getflp0
  bhalf=freq(#s(#((npts(#s)/2)-1)))
  bhalf2=Freq(#s(#(index(#s(bhalf))+1)))
  setfile    sfileslot
82  shalf=freq(#s(#((npts(#s)/2)-1)))
```

```
  shalf2=Freq(#s(#(index(#s(shalf))+1)))
  noshow
  for i=1  to ns
    loadspc $name,#i
    portin -143,t1
    portin -144,t2
    #s=#s(flp,shalf)
    setffp flp,ffp
    gosub  100
    if type=3 then goto 84
    openbgd
    setfile bfileslot
    #s=#s(bffp,bhalf)
    setffp bffp,bflp
    setfile sfileslot
    if type=0 then #s=transmit(#s,#b)
    if type=1 then #s=absorb(#s,#b)
    if type=2 then #s=kmunk(#s,#b)
84    if i=1 then j=i else j=i*2-1
    portout -143,t1
    portout -144,(t2-(t2-t1)/2)
    dialogoff 20
    savespc $temp,#j
    noshow
    loadspc $name,#i
    #s=#s(shalf2,ffp)
    setffp flp,ffp
    reverse #s
    gosub  100
    if type=3 then goto 86
    openbgd
    setfile bfileslot
    #s=#s(bhalf2,bflp)
    setffp bffp,bflp
    setfile sfileslot
    if type=0 then #s=transmit(#s,#b)
    if type=1 then #s=absorb(#s,#b)
    if type=2 then #s=kmunk(#s,#b)
86    j=i*2
    portout -143,(t2-(t2-t1)/2)
    portout -144,(t2)
    dialogoff 20
    savespc $temp,#j
    noshow
  next i
  goto 98

'bidirectional doublesided compute
88  noshow
  dialogon "Multifile Interferogram Compute"
  dialogask j,1+256,0,0,"Computing subfile number"
  dialogoff 20,0,0
  digram=1
  loadspc  $name,#1
  sfileslot=getsfile()
  if type=3 then goto 89
  bfileslot=getbfile()
  setfile    bfileslot

  string $temp2=^"^$[Scanning]"
  string $temp3,"Bidirectional"
  !mov a,#($temp2,$temp3)          'compare characters
  !if a=#-1:openbgd 0:dialogoff -20:erroresc "Background is not a Bidirectional collect."
```

```
  string $temp2=^"^$[Igram_Symmetry]"
  string $temp3,"Double Sided"
  !mov a,#($temp2,$temp3)          'compare characters
  !if a=#-1:openbgd 0:dialogoff -20:errorresc "Background is not a Double Sided collect."
  bffp=getffp()
  bflp=getflp()
  b12=freq(#s(#((npts(#s)/4)-1)))
  b21=freq(#s(#((npts(#s)/4))))
  b22=freq(#s(#((npts(#s)/2)-1)))
  b31=freq(#s(#((npts(#s)/2))))
  b32=freq(#s(#((npts(#s)*.75)-1)))
  b41=freq(#s(#((npts(#s)*.75))))

  setfile   sfileslot
89 slast=npts(#s)-1
  shalf2=(slast+1)/2
  shalf=shalf2-1
  s1=int (shalf2/2+134)
  s2=int (shalf2/2-135)
  s3=(shalf2+s1)
  s4=(shalf2+s2)
  shalf=freq(#s(#shalf))
  shalf2=freq(#s(#shalf2))
  s1=freq(#s(#s1))
  s2=freq(#s(#s2))
  s3=freq(#s(#s3))
  s4=freq(#s(#s4))
  noshow
  for i=1  to ns

   loadspc $name,#i        'first Igram
   portin -143,t1
   portin -144,t2
   #s=#s(flp,s1)
   setffp flp,ffp
   reverse #s
   gosub  100
   if type=3 then goto 90
   openbgd
   setfile bfileslot
   #s=#s(bffp,b12)
   setffp bffp,bflp
   setfile sfileslot
   if type=0 then #s=transmit(#s,#b)
   if type=1 then #s=absorb(#s,#b)
   if type=2 then #s=kmunk(#s,#b)
90   if i=1 then j=i else j=i*4-3
   portout -143,t1
   portout -144,(t1+(t2-t1)/4)
   dialogoff 20
   savespc $temp,#j
   noshow

   loadspc $name,#i     'second     Igram

   #s=#s(s2,shalf)
   setffp flp,ffp
   gosub  100
   if type=3 then goto 91
   openbgd
   setfile bfileslot
   #s=#s(b21,b22)
   setffp bffp,bflp
```

```
    setfile sfileslot
    if type=0 then #s=transmit(#s,#b)
    if type=1 then #s=absorb(#s,#b)
    if type=2 then #s=kmunk(#s,#b)
91    j=i*4-2
    portout -143,(t1+(t2-t1)/4)
    portout -144,(t1+(t2-t1)/2)
    dialogoff 20
    savespc $temp,#j
    noshow

    loadspc $name,#i        'third Igram
    #s=#s(shalf2,s3)
    setffp flp,ffp
    reverse #s
    gosub  100
    if type=3 then goto 92
    openbgd
    setfile bfileslot
    #s=#s(b31,b32)
    setffp bffp,bflp
    setfile sfileslot
    if type=0 then #s=transmit(#s,#b)
    if type=1 then #s=absorb(#s,#b)
    if type=2 then #s=kmunk(#s,#b)
92    j=i*4-1
    portout -143,(t1+(t2-t1)/2)
    portout -144,(t1+(t2-t1)*.75)
    dialogoff 20
    savespc $temp,#j
    noshow

    loadspc $name,#i        'forth Igram

    #s=#s(s4,ffp)
    setffp flp,ffp
    gosub  100
    if type=3 then goto 93
    openbgd
    setfile bfileslot
    #s=#s(b41,bflp)
    setffp bffp,bflp
    setfile sfileslot
    if type=0 then #s=transmit(#s,#b)
    if type=1 then #s=absorb(#s,#b)
    if type=2 then #s=kmunk(#s,#b)
93    j=i*4
    portout -143,(t1+(t2-t1)*.75)
    portout -144,(t2)
    dialogoff 20
    savespc $temp,#j
    noshow

  next i

98  dialogoff -20
  noshow
  if ow=0    goto 99
  kill $name            'wipe out original
  kill $temp,$name          'rename    temp to   original
  temp=name

'All done, show    results
```

```
99  loadspc  $temp
  setleft  getffp0,getflp0 : autoscale
  if getleft0<getright0   then xflip
  if type<>4 then  openbgd  0      'close #B
  waitoff
  if zf=0  then end          'no resolution change
  xres=rnd(xres)*2
  temp=0
  string temp=xres : string temp=$temp+"cm-1"
  savespc
  end

'****************************************
'Create apodization & compute subroutine
'****************************************
100  goto 110+(ap*10)

'Triangular
110  zfac=zf+1
  goto 1000

'Boxcar
120  zfac=neg(zf+1)
  goto 1000

'Weak Norton-Beer
130  c(0)= 0.5480
  c(1)=-0.0833
  c(2)= 0.5353
  c(3)= 0
  zfac=neg(zf+1)
  gosub 900 : goto 950

'Medium  Norton-Beer
140  c(0)= 0.152442
  c(1)=-0.136176
  c(2)= 0.983734
  c(3)= 0
  zfac=neg(zf+1)
  gosub 900 : goto 950

'Strong Norton-Beer
150  c(0)= 0.045335
  c(1)= 0
  c(2)= 0.554883
  c(3)= 0.399782
  zfac=neg(zf+1)
  gosub 900 : goto 950

'Happ-Genzel
160  gosub 900
  zfac=neg(zf+1)
  tt_=cos(fill(tt_)*3.14159) * 0.46 + 0.54
  #s=#s*tt_
  goto 1000

'Bessel
170  gosub 900
  zfac=neg(zf+1)
  tt_=squared(-(squared(neg(fill(tt_)))-1))
  #s=#s*tt_
  goto 1000
```

```
'Cosine
180  gosub 900
  zfac=neg(zf+1)
  tt_=(cos(fill(tt_)*pi_value)+1)/2
  #s=#s*tt_
  goto 1000

'Find ZPD and set up arrays for     apodization functions
900  mx=max(#s) : mn=min(#s)
  if abs(mx)>abs(mn) then  zpd=index0(#s-mx) else zpd=index0(#s-mn)
  np=npts(#s) : fp=getffp() : lp=getflp()
  pd=np-zpd
  fillbeg   -(1/(np-zpd-1))*zpd
  dim tt_(np)
  fillinc    1/(np-zpd-1)
  return

'Build and apply Norton-Beer function
950  tt_=tt_+c(0)
  if ap<>4 then tt_=tt_+(-(fill(tt_)^2)+1)*c(1)
  tt_=tt_+((-(fill(tt_)^2)+1)^2)*c(2)
  if ap=4 then tt_=tt_+((-(fill(tt_)^2)+1)^3)*c(3)
  #s=#s*tt_

'Do the compute  & ratio
1000 onerror    1100 : compute #s,zfac : onerror -1
  xres=abs(getffp()-getflp())/(npts(#s)-1)
  portout    addmod,3                'audit trail (C)
  #s=#s(fbeg,fend)
  portout    addmod,20               'audit trail (T)
  setytype 0 : setxtype 1
  if digram=1 then return                'bidirectional
  if type=0 then #s=transmit(#s,#b)
  if type=1 then #s=absorb(#s,#b)
  if type=2 then #s=kmunk(#s,#b)
  if type<>4 then  portout    addmod,25                'audit trail (Y)
  return

'**********************
'Error - cannot  compute
'**********************
1100 dialogon "ERROR!":beep
  dialogloc 1,0,4    : print "Cannot compute interferogram to single beam spectrum:"
  dialogloc 1,0,14 : print "1.) Interferogram must have a minimum of 128 data points on both sides of center burst."
  dialogloc 1,0,24 : print "2.) The X values of the interferogram must span zero to the laser wavenumber."
  dialogloc 3,0,44 : input >mc,4,1200,1109,"Fix_data_file Quit"
  dialogoff
1109 end

1200 mx=max(abs(#s))
  pmx=index(#s(mx)) : np=npts(#s)
  if (mx>=127 and  abs(np-mx)>=127) goto 1210
  beep
  dialogon "ERROR!"
  print "Cannot fix file.  Less than 128 data points on one side (or both) of center burst."
  print " "
  dialogoff
1210 rx=7899
  dialogon "Reset X Value of Last Data Point"
  input >rx,0,0,25000,"Enter laser frequency (cm-1): "
  dialogoff 1
  setffp 0,rx : setleft 0,rx : see
  goto 10
```

H.2 ICOMPUT.AB

```
' Univariant quantitative analysis V1.01G
' (c) 1993 Galactic Industries Corp. - All rights reserved.

'Header for quant file
'uni flag id
'number of components=m
'ref = false no reference true reference
'um is main matrix it has 20 columns
'0  1    2  3  4  5  6  7  8 9
'lb1  lb2  rbt  rb1  rb2  rbt  lre    rre        rt        Qt
'10 11    12     13     14     15     16     17     18     19     20
'zi  coeff1 etc
'cnames is m,20
1 free
  portout -44,1
  gosub 5
  goto 8

'+ + + + + + + +Allocate strings and arrays
5 onesc 62800      'cover all errors
  pauseoff
  dim title(256),temp(256),text(256),name(256),memo(256),message(256),directory(256)
  dim
temp2(256),temp3(256),temp4(256),temp5(256),mtext(5),ctext(5),WHC_name(12),WHC_run(5),WHC_time(10),WHC_temp(8)
  dim methname(256):string $methname="method.q"
  dim cfiname(256):string $cfiname="cal.cfi"
  viewstore
  return

8 verbose=1       'report all errors on internal use
  onerror -1
  dialogon "QuantBasic"
  dialogask mc,4,10,15,"Predict Calibrate"
  dialogoff

'predict
10  if getsfile0<>0 then goto 60050
  dialogon "Error"
  print "No spectrum is loaded in memory to quantitate."
  print "Use the File / Open menu to load a spectrum."
  print ""
  dialogask mc,4,12,0,"___&OK___"
  dialogend
12  end

'calibrate
15  v=getsfile0            'SPC in memory?
  if v<>0 then goto 20
  gosub 1900            'spc or method
  if methloaded=1 then viewonly "quantbas.vwh":see:goto 100
  if methloaded=0 and getsfile0=0 then goto 8          'double check

'initialize variables
20  viewonly "quantbas.vwh"
  m=1        'number of components
  free um:dim UM(1,20)
  transpose um
  um(0)=-999999
  um(1)=-999999
  um(3)=-999999
```

```
  um(4)=-999999
  um(6)=-999999
  um(7)=-999999
  transpose um
  free cname:dim cname(1,15)
'um(0,8)=2
  ref=false
  comp=0          'first component
  gosub 57190     'autoscale and orient
  see

100  onpaint 44000

200  onerror -1
  dlv=4       'initial vertical
  dlh=-100    'initial horizontal right
  dlvs=12     'vertical spacing
  dlhs=80     'horizontal spacing
  dlhc=dlh    'current horizontal
  dlvc=dlv    'current vertical

'=======
' Calibration Dialog

300  dialogon "QuantBasic Region Editor"
  dlhc=dlhc+dlhs*.3:dialogloc 2,dlhc,dlvc:dialogask cname(comp),2,15,15,"Component Name: "
  dialogloc 2,dlhc+dlhs*.95,dlvc-4:dialogtxt 7,1,31,""
  dlhc=dlhc+dlhs*1.35:dialogloc 2,dlhc,dlvc:dialogtxt 0,0,0,"Region"
  dlhc=dlhc+dlhs*.2:dialogloc 2,dlhc,dlvc-2:dialogask $ctext,2,3,3,""
  dlhc=dlhc+dlhs*.1:dialogloc 2,dlhc,dlvc:dialogtxt 0,0,0,"of"
  dlhc=dlhc+dlhs*.2:dialogloc 2,dlhc,dlvc-2:dialogask $mtext,2,3,3,""
  dlhc=dlhc+dlhs*1.2:dialogloc 2,dlhc,dlvc:dialogask  mc,6,1000,1010,"Add Del Next"
  dlvc=dlvc+dlvs+3:dlhc=dlh+dlhs*.3
  dialogloc 2,dlhc,dlvc:dialogask caltyp,3+128,0,0,"Calibration_Type: Raw Single_Point Multi_Point"
  dlhc=dlhc+dlhs*1.75:dialogloc 2,dlhc,dlvc:dialogask mref,3+128,0,0,"Reference_Band:"
  dlhc=dlhc+dlhs*1.15:dialogloc 2,dlhc,dlvc:dialogask mc,6,2300,0,"Update_Display"
  dlvc=dlvc+dlvs+3:dlhc=dlh
  dialogloc 2,dlhc-dlhs*.5,dlvc-3:dialogtxt 7,300,1,""
  dialogloc 2,dlhc,dlvc+2:print "Left Baseline:"
  dlhc=dlhc+dlhs*.3:dialogloc 2,dlhc,dlvc+2:dialogask um(comp,2),3+128,0,0,"Type: Single Minimum_ Average None"
  dlhc=dlhc+dlhs*.92:dialogloc 2,dlhc,dlvc+2:dialogask mc,6,2000,0,"Set"
  dlhc=dlhc+dlhs*1.05
  if fp>lp then dialogloc 2,dlhc,dlvc:dialogask UM(comp,0),0,lp,fp,""
  if fp<lp then dialogloc 2,dlhc,dlvc:dialogask UM(comp,0),0,fp,lp,""
  dlhc=dlhc+dlhs
  if fp>lp then dialogloc 2,dlhc,dlvc:dialogask UM(comp,1),0,lp,fp,""
  if fp<lp then dialogloc 2,dlhc,dlvc:dialogask UM(comp,1),0,fp,lp,""

  dlhc=dlh:dlvc=dlvc+dlvs
  dialogloc 2,dlhc,dlvc+2:print "Right Baseline:"
  dlhc=dlhc+dlhs*.3:dialogloc 2,dlhc,dlvc+2:dialogask um(comp,5),3+128,0,0,"Type: Single Minimum_ Average None"
  dlhc=dlhc+dlhs*.92:dialogloc 2,dlhc,dlvc+2:dialogask mc,6,2001,0,"Set"
  dlhc=dlhc+dlhs*1.05
  if fp>lp then dialogloc 2,dlhc,dlvc:dialogask UM(comp,3),0,lp,fp,""
  if fp<lp then dialogloc 2,dlhc,dlvc:dialogask UM(comp,3),0,fp,lp,""
  dlhc=dlhc+dlhs
  if fp>lp then dialogloc 2,dlhc,dlvc:dialogask UM(comp,4),0,lp,fp,""
  if fp<lp then dialogloc 2,dlhc,dlvc:dialogask UM(comp,4),0,fp,lp,""

  dlhc=dlh:dlvc=dlvc+dlvs
  dialogloc 2,dlhc,dlvc+2:print "Region:"
  dlhc=dlhc+dlhs*.3:dialogloc 2,dlhc,dlvc+2:dialogask um(comp,8),3+128,0,0,"Type: Single Maximum Area"
  dlhc=dlhc+dlhs*.92:dialogloc 2,dlhc,dlvc+2:dialogask mc,6,2002,0,"Set"
```

```
  dlhc=dlhc+dlhs*1.05
  if fp>lp then dialogloc 2,dlhc,dlvc:dialogask UM(comp,6),0,lp,fp,""
  if fp<lp then dialogloc 2,dlhc,dlvc:dialogask UM(comp,6),0,fp,lp,""
  dlhc=dlhc+dlhs
  if fp>lp then dialogloc 2,dlhc,dlvc:dialogask UM(comp,7),0,lp,fp,""
  if fp<lp then dialogloc 2,dlhc,dlvc:dialogask UM(comp,7),0,fp,lp,""
  dlvc=dlvc+dlvs+7:dlhc=dlh
  dialogloc 2,dlhc-dlhs*.5,dlvc-5:dialogtxt 7,300,1,""
  dialogloc 3,0,dlvc:dialogask  mc,6,1050,1060,"Zoom Autoscale Calibrate Load Save Exit"

'  dialogloc 3,0,dlvc:dialogask  mc,6,2200,2300,"Next Update Add Delete Zoom Autoscale"
'  dlvc=dlvc+dlvs:dlhc=dlh
'  dialogloc 3,0:dialogask  mc,6,2900,3000,"Reference Calibrate Save Save_As Load Exit"

400  dialogoff 11,0,500

' Dialog management
500  if mref=1 and ref=0 then gosub 2900
  if mref=0 and ref=1 then goto 2500
  dialogset 16,4:mt=comp+1:string $ctext=mt:dialogset 1,4,0,$ctext
  mm=m+ref
  dialogset 16,6:string $mtext=mm:dialogset 1,6,0,$mtext
  if ref=1 and comp=m then dialogset 16,1 else dialogset 8,1
  if um(comp,2)=3 then dialogset 256,15:dialogset 256,16:dialogset 16,14
  if um(comp,5)=3 then dialogset 256,20:dialogset 256,21:dialogset 16,19
  if um(comp,2)=1 or um(comp,2)=2 then dialogset 128,15:dialogset 128,16:dialogset 8,14
  if um(comp,5)=1 or um(comp,5)=2 then dialogset 128,20:dialogset 128,21:dialogset 8,19
  if um(comp,2)=0 then dialogset 128,15:dialogset 256,16:dialogset 8,14
  if um(comp,5)=0 then dialogset 128,20:dialogset 256,21:dialogset 8,19
  if um(comp,8)=0 then dialogset 256,26 else dialogset 128,26
  if caltyp=0 then dialogset 16,28,3 else dialogset 8,28,3
  if mm=1 then dialogset 16,7,2:dialogset 16,7,3
  if mm<>1 then dialogset 8,7,2:dialogset 8,7,3
  gosub 41000
  gosub 43000
  goto 400

' Jump table for dialog
  1000  goto 2400    'add
  1010  goto 2500    'delete
  1020  goto 2200    'next

  1050  goto 2600    'zoom
  1060  goto 2700    'autoscale
  1070  goto 3000    'calibrate

  1080  dialogoff -10    'load
    gosub 1900
    goto 200
  1090  gosub 41000
    goto 3200   'save
  1100  goto 62700   'exit

'+++++++ load method or spc subroutine
1900 onerror 1930
  dialogon "Load What?"
  dialogask mc,4,1910,1920,"Method Spectrum"
  dialogoff
1910 onerror -1
  gosub 3300
  return

1920 onerror 1930
```

```
  temp=0
  menufile $temp,"Choose Spectrum to Evaluate  *.spc"
  onerror -1
  if methloaded=1 then gosub 57130      'get current method range etc.
  gosub 57180     'load
  if spcOK=0 then goto 1930
  if methloaded=0 then goto 1928
  gosub 57110
  if spcmatch=1 then goto 1928
  beep
  dialogon "Warning"
  print "The spectrum you have chosen to load does not match"
  print "the current method in the following way:"
  print ""
  print $message
  print ""
  print "If you choose OK to continue Quantbasic may not be able to"
  print "calculate the spectrum height or area"
  print ""
  dialogoff

1928 onerror -1
  if methloaded<>0 then mc=getsfile0:setfile mc+1:noshow:setfile mc
  methloaded=2
  gosub 57190     'orient
  see
1930 return

'*******
' Select Regions
2000 i=0:goto 2005
2001 i=3:goto 2005
2002 i=6

2005 dialogoff -10
  onerror 2015
  temp2=0
  temp3=0
  if um(comp,2+i)=3 then goto 400
  if um(comp,2+i)=0 then goto 2010
2006 string temp2="Use F9 or Right mouse for Left edge.  F10 or Right mouse again for Right edge"
  string Temp3="Press Enter or Left mouse button to accept."
  gosub 43100
  pointer cx,cy,le,re
  um(comp,0+i)=freq(#S(le)):um(comp,1+i)=freq(#S(re))
  dialogoff -20
  goto 2020
2010 string temp2="Place the cursor at the desired position."
  string Temp3="Press Enter or Left mouse button to accept."
  gosub 43100
  pointer cx,cy,le,re
  um(comp,0+i)=freq(#S(cx)):um(comp,1+i)=freq(#S(cx))
2015 dialogoff -20
2020 onerror -1
  goto 200

'*******
'Next component
2200 comp=comp+1
  if comp>nrows(um)-1 then comp=0
  goto 500

'*******
```

```
'Update button
2300 goto 500

'*******
' Add a compnent
2400 mc=nrows(um)
  dim tum(mc+1,20),tcname(mc+1,15)
  if ref=1 then goto 2410
  tcname=cname
  tum=um
  free um,cname:dim um(mc+1,20),cname(mc+1,15):um=tum:cname=tcname:free tum,tcname
  um(mc)=-999999:um(mc,9)=0:um(mc,10)=0
  um(mc,2)=0:um(mc,5)=0:um(mc,8)=0
  comp=mc
  m=m+1
  goto 2420

2410 for i=0 to mc-2
    tum(i)=um(i)
    tcname(i)=cname(i)
  next i
  tum(mc)=um(mc-1)
  tcname(mc)=cname(mc-1)
  free um,cname:dim um(mc+1,20),cname(mc+1,15):um=tum:cname=tcname:free tum,tcname
  um(mc-1)=-999999:um(mc,9)=0:um(mc,10)=0
  um(mc-1,2)=0:um(mc-1,5)=0:um(mc-1,8)=0
  comp=mc-1
  m=m+1
2420  if cfl=1 then gosub 56505   'handle mp
  goto 500

'*******
'Delete current component
2500 mc=nrows(um)
  if mc=1 then beep:goto 500   'can't delete last peak
  if mc>2 then goto 2505
  if m=1 and comp=0 then beep:goto 500
2505 dim tum(mc-1,20),tcname(mc-1,15)
  if comp=0 then goto 2510
  for i=0 to comp-1
    tum(i)=um(i)
    tcname(i)=cname(i)
  next i
2510 if i=mc-1 then goto 2520
  for i=comp+1 to mc-1
    tum(i-1)=um(i)
    tcname(i-1)=cname(i)
  next
2520 if ref=1 and comp=mc-1 then ref=false:m=m+1:mref=false:goto 2525
  if cfl=1 then gosub 56605
2525 comp=comp-1:if comp<0 then comp=0
  free um,cname:dim um(mc-1,20),cname(mc-1,15):um=tum:cname=tcname:free tum,tcname
  m=m-1
  goto 500

'*******
' Zoom
2600 dialogoff -10
  onerror 2655
  string temp2="Click and hold the left mouse button to zoom a region."
  temp3=0
  gosub 43100
  pointer cx,cy,le,re,cfx,cfy
```

```
  dialogoff -20
  onerror 2650   'if use le,re action cx~=cfx
  setleft cx,cfx
  setmin cy,cfy
  setleft cx,cfx
  setmin cy,cfy
  gosub 43000
  goto 2655

'must be using le,re actions
2650 onerror 2655
  setleft le,re
  gosub 43000
2655 dialogoff -20
  onerror -1
  goto 200

'*******
' Autoscale
2700 setleft getffp0,getflp0
  autoscale
  goto 500

'*******
' Add a reference
2900  '  if ref=1 then goto 2910
     '  onerror 400
     '  dialogon "Create a Reference Peak"
     '  print "A reference peak has not been defined."
     '  print "would you like to do so?"
     '  print ""
     '  dialogask mc,4,2905,400,"OK Cancel"
     '  print ""
     '  dialogoff
2905   ' onerror -1
  mc=nrows(um)
  dim tum(mc+1,20)
  tum=um
  free um:dim um(mc+1,20):um=tum:free tum
  um(mc)=-999999:um(mc,9)=0:um(mc,10)=0
  um(mc,2)=0:um(mc,5)=0:um(mc,8)=0
  dim tcname(mc+1,15):tcname=cname:string $tcname(mc)="Reference"
  free cname:dim cname(mc+1,15):cname=tcname:free tcname
  ref=1
  comp=mc
2910 'onerror 400
  'dialogon "Reference Peak"
  'dialogask showref,3,0,0,"Show_Reference_peak:"
  'dialogoff
  'return

'Add rows to um

  return

' Calibration menu
3000 dialogoff -10
  if caltyp=0 then goto 3010
  if caltyp=1 then goto 3020
  if caltyp=2 then goto 3030

'++++++++
'Raw Values calibration: should never get here
```

```
'++++++++

3010 dialogon "Calibration"
  print "No calibration needed for reporting Raw values"
  dialogoff
  goto 3090

'+++++++++
'Single point calibration values
'+++++++++

3020 dlvc=dlv:dlhc=dlh
  onerror 3090
  dialogon "Enter Quantities for each component"
  for i=0 to m-1
    dialogloc 2,dlhc,dlvc:dialogask cname(i),2,15,15,"Component: "
    dlhc=dlhc+dlhs*1.8:dialogloc 2,dlhc,dlvc:dialogask UM(i,11),32+0,-999999,999999,"Concentration: "
    dlvc=dlvc+dlvs:dlhc=dlh
  next i
  dialogoff
  goto 3090

'++++++++
'Multi-point claibration
'++++++++

3030 gosub 50000
3090 onerror -1
  gosub 41000
  gosub 43000
  onpaint 44000
  goto 200

'*******
' SaveAs
3200 goodcal=1      'good calibration flag
  if methname=0 then string $methname="method.q"
  string temp5,-7     'get current directory
  onerror 3290
  menufile methname, "Method Filename   *.q",8+1
  onerror -1
  string temp5,-8     'restore current directory
  strname 7,$temp=$methname
  methname=temp
  kill "methname"
  if caltyp=2 and cfl=0 then string message="Calibration data not defined.":goto 3295
  gosub 57000      'Get X and Y values
  if dataOK=0 then goto 3295
  if caltyp=2 then goto 3210
  if caltyp=0 then goto 3220
  noerror=0
  for i=0 to m-1
    if um(i,11)=0 then noerror=99
    um(i,12)=Xk(i)
  next i
  if noerror=99 then string message="One or more calibration factors is set to zero.":goto 3295
  goto 3220

3210 noerror=0
  for sc=0 to m-1
    gosub 57200
    if calibOK=0 then noerror=99:goto 3215
    for jj=0 to sorder-1
```

```
    um(sc,jj+11)=coeffmatrix(jj,0)
   next jj
  um(sc,11+sorder+1)=rsquared
3215 next sc

3220 open #1,$methname
  temp=0
  v=69
  write #1,v
  write #1,m,ref,caltyp,um,cname,goodcal
  if goodcal=1 then gosub 57130:write #1,mfp,mlp,minterval,mxtype,mytype
  if caltyp=2 and cfl=1 then write #1,cfl,ns,na,y,sname,ck,Yk,Xk else write #1,cfl
  close #1
3290 onerror -1
  if goodcal=0 then goto 200
  goto 500

3295 goodcal=0   'flag calibration as not suitable for prediction
  onerror 3296
  dialogoff -10
  dialogon "Warning"
  print "The method being saved is incomplete and cannot be"
  print "used for predicting unknown concentration due to the"
  print "following problem:"
  print ""
  print $message
  print ""
  print "You may save the method as it stands by clicking on"
  print "Continue or you can abort the save and return to the "
  print "method editor"
  print ""
  dialogask mc,4,3220,3290,"Continue Cancel"
  dialogoff
3296 goto 200

'*******
'Load Method
3300 spcOK=0
  onerror 3365
  menufile methname,"Load Method   *.q"
3310 onerror 3312
  open #1,$methname
  read #1,id,m,ref,caltyp
  if id=69 then goto 3320
3312 if verbose=0 then goto 3315
  dialogon "Error"
  print "Bad or invalid QuantBasic method file."
  print ""
  dialogoff
3315 close #1
  methloaded=0
  return
3320 free um,cname,sname,ns,na,y,ck,yk,xk
  mc=m
  if ref=1 then mc=mc+1
  dim um(mc,20)
  read #1,um
  dim cname(m+ref,15)
  read #1,cname,goodcal
  if goodcal=1 then read #1,mfp,mlp,minterval,mxtype,mytype
  mm=m+ref
  mref=ref
  comp=0
```

```
  if caltyp<2 then goto 3350
  read #1,cfl
  if cfl=0 then goto 3350
  read #1,ns,na
  dim y(na,m):read #1,y
  dim sname(na,13),ck(na)dim yk(m,ns),xk(m,ns):read #1,sname,ck,yk,xk
  onerror -1
  if call=1 then close #1:methloaded=1:return
  temp=0:temp=sname(0)
  gosub 57180     'load first spc from training
  if spcOK=0 then goto 3370
  if methloaded<>0 then mc=getsfile0:setfile mc+1:noshow:setfile mc          'remove old spc

  gosub 57190     'orient and scale
3350 close #1
3360 methloaded=1
3365 onerror -1
  return

'Error loading spc
3370 close #1
  dialogon "Warning"
  print "Could not load first training spectrum."
  print ""
  dialogoff
  return

' ********SUBROUTINE
' Calculate region paramaters
' Calculate baseline X,Y values
' First check for valid settings
41000   if um(comp,0)<>-999999 then um(comp,0)=freq(#S(um(comp,0)))        'round to nearest point
  if um(comp,1)<>-999999 then um(comp,1)=freq(#S(um(comp,1)))
  if um(comp,3)<>-999999 then um(comp,3)=freq(#S(um(comp,3)))
  if um(comp,4)<>-999999 then um(comp,4)=freq(#S(um(comp,4)))
  if um(comp,6)<>-999999 then um(comp,6)=freq(#S(um(comp,6)))
  if um(comp,7)<>-999999 then um(comp,7)=freq(#S(um(comp,7)))

  if um(comp,2)=0 then um(comp,1)=UM(comp,0)                    'dup values if single<dk>06/07/94
  if um(comp,5)=0 then um(comp,4)=um(comp,3)
  if um(comp,8)=0 then um(comp,7)=um(comp,6)

  if um(comp,0)=-999999 or um(comp,1)=-999999 then b1=false else b1=true          'leftbase set?
  if um(comp,2)=3 then b1=true
  if um(comp,3)=-999999 or um(comp,4)=-999999 then b2=false else b2=true          'rightbase set?
  if um(comp,5)=3 then b2=true
  if b1=true and b2=true then ba=true else ba=false                    'Both base set?
  if um(comp,6)=-999999 or um(comp,7)=-999999 then r=false else r=true     'region set?
  if ba=true and r=true then rdefined=true else rdefined=false         'Aok of all?
  if b1=false then goto 41020          'if baseline values not set skip
  if um(comp,2)=3 then goto 41020                  'no baseline
  if um(comp,2)=0 then lby=#s (um(comp,0)):lbx=um(comp,0)   'Single left

  if um(comp,2)<>1 then goto 41010  'Minimum left
  lby=min(#s(um(comp,0),um(comp,1)))          'y left position
  mc=index0(#s(um(comp,0),um(comp,1))-lby)
  lbx=um(comp,0)+interval*mc
  lbx=freq #s(lbx)

41010   if um(comp,2)<>2 then goto 41020     'average left
  if (npts(#s(um(comp,0),um(comp,1))))=1 then lby=#s (um(comp,0)):lbx=um(comp,0):goto 41020     'need at least 2
pts
  lby=(sum(#s(um(comp,0),um(comp,1))))/(npts(#s(um(comp,0),um(comp,1))))
```

```
  lbx=um(comp,0)+((um(comp,1)-um(comp,0))/2)

41020  if b2=false then goto 41040                    'if baseline values not set skip
  if um(comp,5)=3 then goto 41040                     'no baselin
  if um(comp,5)=0 then rby=#s (um(comp,3)):rbx=um(comp,4) 'single right

  if um(comp,5)<>1 then goto 41030  'Minimum right
  rby=min(#s(um(comp,3),um(comp,4)))          'y left position
  mc=index0(#s(um(comp,3),um(comp,4))-rby)
  rbx=um(comp,3)+interval*mc
  rbx=freq #s(rbx)

41030  if um(comp,5)<>2 then goto 41040     'Average right
  if (npts(#s(um(comp,3),um(comp,4))))=1 then rby=#s (um(comp,3)):rbx=um(comp,4):goto 41040 ' can't /0
  rby=(sum(#s(um(comp,3),um(comp,4))))/(npts(#s(um(comp,3),um(comp,4))))
  rbx=um(comp,3)+((um(comp,4)-um(comp,3))/2)

'Calculate region x and y values and height x and y values

41040  if r=false then goto 41050
  lrx=um(comp,6)        'left region x
  rrx=um(comp,7)        'right region x
  if um(comp,2)=3 and um(comp,5)=3 then lry=0:rry=0:slope=0:offset=0:goto 41060          'no baselines
  if um(comp,2)=3 then lry=rby:rry=rby:slope=0:offset=rry:goto 41060                    'only right baseline
  if um(comp,5)=3 then lry=lby:rry=lby:slope=0:offset=lry:goto 41060                    'only left baseline

'Calculate slope and offset of baseline
41050  if ba=false then goto 41070
  slope=(lby-rby)/(lbx-rbx)
  offset=lby-slope*lbx

'region y values
41060  if r=false then goto 41500
  lry=slope*lrx+offset
  rry=slope*rrx+offset
41070  if r=false then goto 41500
  if um(comp,8)=2 then goto 41500
'peakbase value
  if r=false then goto 41500
  if um(comp,8)=0 then px=lrx:py=lry:goto 41500
  px=max(#s(lrx,rrx))       'y left position
  mc=index0(#s(lrx,rrx)-px)
  px=lrx+interval*mc
  px=freq #s(px)
  py=slope*px+offset


41500  if ba=false or r=false then goto 41900
  if um(comp,8)=2 then goto 41600
  height =#s(px)-py

  goto 41900
'area
41600  np=npts #s(lrx,rrx)
  if np=1 then area=0:goto 41900        'can't calculate 0 area!
  dim base(np)
  fillbeg lry              'strt baseline function at value of left pnt
  fillinc (rry-lry)/(np-1)     'slpe of function to delta of edges
  base = fill(base)              'create baseline correcting straight line
  base=#S(lrx,rrx)-base
  area=(sum(base)-(base(0)+base(np-1))/2)*abs(lrx-rrx)/(np-1)      'baseline corr area
  free np,base
```

```
41900   return

'Redraw display
43000   setcolor red          'Draw SPC and axis
  blankbox
  axes 1
  display


43010   plotset 1
  le=getleft0:re=getright0
  mx=getmax0:mn=getmin0


'left baseline ticks for comp
  if um(comp,2)=3 then goto 43020
  if b1=false then goto 43020
  penup
  sy=#s(um(comp,0))
  sy=abs((sy-mn)/(mx-mn))
  plot abs((le-um(comp,0))/(le-re)),sy+.025,black,2
  plot abs((le-um(comp,0))/(le-re)),sy-.025,black,2

  penup
  sy=#s(um(comp,1))
  sy=abs((sy-mn)/(mx-mn))
  penup
  plot abs((le-um(comp,1))/(le-re)),sy+.025,black,2
  plot abs((le-um(comp,1))/(le-re)),sy-.025,black,2

'right baselne ticks for comp
43020   if um(comp,5)=3 then goto 43030
  if b2=false then goto 43030
  penup
  sy=#s(um(comp,3))
  sy=abs((sy-mn)/(mx-mn))
  plot abs((le-um(comp,3))/(le-re)),sy+.025,black,2
  plot abs((le-um(comp,3))/(le-re)),sy-.025,black,2

  penup
  sy=#s(um(comp,4))
  sy=abs((sy-mn)/(mx-mn))
  penup
  plot abs((le-um(comp,4))/(le-re)),sy+.025,black,2
  plot abs((le-um(comp,4))/(le-re)),sy-.025,black,2

'draw the region lines
43030   if r=false then goto 43040

  penup
  plot abs((le-lrx)/(le-re)),abs((#s(lrx)-mn)/(mx-mn)),khaki,2
  plot abs((le-lrx)/(le-re)),abs((lry-mn)/(mx-mn)),khaki,2
  penup
  plot abs((le-rrx)/(le-re)),abs((#s(rrx)-mn)/(mx-mn)),khaki,2
  plot abs((le-rrx)/(le-re)),abs((rry-mn)/(mx-mn)),khaki,2

  if um(comp,8)<>2 then goto 43035
  np=npts #s(rrx,lrx)
  if np=1 then goto 43035  'can't calculate area of single pt
  mc=(rrx-lrx)/(np-1):t=mc
  t2=(rry-lry)/(np-1):t3=t2
  if ref=1 and comp+1=nrows(um) then color=brown else color=khaki
  for i=0 to np-3
```

```
    penup
    plot abs((le-lrx-(t))/(le-re)),abs((#s(lrx+t)-mn)/(mx-mn)),color,2
    plot abs((le-lrx-(t))/(le-re)),abs((lry+t3-mn)/(mx-mn)),color,2
    t=t+mc
    t3=t3+t2
  next

'draw the peak lines
43035   if um(comp,8)=2 then goto 43040
  penup
  plot abs((le-px)/(le-re)),abs((#s(px)-mn)/(mx-mn)),navy,2
  plot abs((le-px)/(le-re)),abs((py-mn)/(mx-mn)),navy,2

'draw baseline

43040   if ba=false then goto 43090
  if r=true then goto 43050
  if um(comp,2)=3 or um(comp,5)=3 then goto 43090
  tlbx=lbx:tlby=lby
  trbx=rbx:trby=rby
  goto 43065
43050   if getxtype0=1 then goto 43060
  if lbx<=lrx then tlbx=lbx:tlby=lby
  if lbx>lrx then tlbx=lrx:tlby=slope*lrx+offset
  if rbx>=rrx then trbx=rbx:trby=rby
  if rbx<rrx then trbx=rrx:trby=slope*rrx+offset
  if um(comp,2)=3 then tlbx=lrx:tlby=lry
  if um(comp,5)=3 then trbx=rrx:trby=rry

  goto 43065
43060   if lbx>=lrx then tlbx=lbx:tlby=lby
  if lbx<lrx then tlbx=lrx:tlby=slope*lrx+offset
  if rbx<=rrx then trbx=rbx:trby=rby
  if rbx>rrx then trbx=rrx:trby=slope*rrx+offset
  if um(comp,2)=3 then tlbx=lrx:tlby=lry
  if um(comp,5)=3 then trbx=rrx:trby=rry
43065   penup
  plot abs((le-tlbx)/(le-re)),abs((tlby-mn)/(mx-mn)),blue,2
  plot abs((le-trbx)/(le-re)),abs((trby-mn)/(mx-mn)),blue,2

  if ba=false or r=false then goto 43090
  printfont 2,2,8,0,"Arial":printhue black,silver
  if um(comp,8)=2 then px=lrx+(rrx-lrx)/2
  cx=neg(abs((le-px)/(le-re)))
  cy=neg(abs((#s(px)-mn)/(mx-mn)))
  printloc cx,cy:if um(comp,8)=2 then print area else print height

43090   if showref=1 and comp<>nrows(um-1) then showref=3:ocomp=comp:comp=nrows(um)-1:gosub
41000:goto 43010
  if showref=3 then comp=ocomp:showref=1
  return

43100   dialogon ""
  print $temp2
  print $temp3
  dialogoff 22
  return

'+++++++Onpaint subroutine for Region screen
44000   see
  gosub 43000
  return
```

```
' A spreadsheet-style editor for building method files "*.q"
' Spectral file names are displayed as rows, the components are the columns.
50000   onpaint 59500
  viewonly "QBfit.vwh"
  see
50001   v=0
  portout -44,1

'Setup status dialog
  dialogon ""
  dialogloc 0,-60
  dialogask message,2+256,35,35,"Messages"
  dialogloc 0,0
  dialogask a,0+256,0,0,"Intercept"
  dialogask b,0+256,0,0,"Linear Coefficent"
  dialogask c,0+256,0,0,"Quadratic Coefficent"
  dialogask d,0+256,0,0,"Cubic Coefficent"
  dialogask rsquared,0+256,0,0,"R Squared"
  dialogoff 20+9,0,0
  onpaint 0
  'goto 56800
  if cfl=1 then goto 50100
  na=1
  n=1
  free sname,ck,y
  dim sname(1,13)
  dim ck(1)
  dim y(1,m)
  string $temp,getsname
  strname 3,$sname(0),$temp
  cfl=1

'Main dialog editor
50100   onerror -1
  SC=0 : SR=0              'Starting column & row
  if comp>m-1 then comp=m-1       '~
  sc=comp              '~
  'if m<3 then LC=m-1 else LC=SC+2   'Last column
  'lc=m-1
  if na<13 then LR=na-1 else LR=SR+13          'First column

50200   onerror -1
  dlv=4       'initial vertical
  dlh=-100   'initial horizontal right
  dlvs=12    'vertical spacing
  dlhs=80    'horizontal spacing
  dlhc=dlh   'current horizontal
  dlvc=dlv   'current vertical

  string $title="Component Editor"

  dialogon $title
  if m>3 then k=70 else k=30+(20*(m-1))
  dialogloc 1,dlhc,dlvc:dialogask $sname(sc),32+2,13,13,"Component"
  dlvc=dlvc+dlvs+4
  dialogloc 1,dlhc,dlvc  :print "Sample Names"
  dlhc=dlhc+dlhs
  dialogloc 1,dlhc,dlvc :print "Exclude"
  dlhc=dlhc+dlhs*.45
  dialogloc 1,dlhc,dlvc      :print "Concentrations"
  dlvc=dlvc+dlvs

'Component names
```

```
  dlhc=dlh

'Sample names and concentrations
  for i=SR to LR
    dialogloc 1,dlhc,dlvc   :print i+1
    dlhc=dlhc+dlhs*.1
    dialogloc 1,dlhc,dlvc      :dialogask $sname(i),32+2,13,13,""
    dialogloc 1,dlhc+52,dlvc+2   :dialogask mc,6,(52230+i-sr),0,"S"
    dlhc=dlhc+dlhs
    dialogloc 1,dlhc,dlvc   :dialogask CK(i),32+3+128,0,0,""
    dlhc=dlhc+dlhs*.7
  dialogloc 2,dlhc,dlvc:dialogask Y(i,sc),32+0+256,0,0,""
    dialogloc 2,dlhc+20,dlvc+2   :dialogask mc,6,(52260+i-sr),0,"I"
    dialogloc 2,dlhc+32,dlvc+2   :dialogask mc,6,(52290+i-sr),0,"D"
    dlvc=dlvc+dlvs
    dlhc=dlh
  next i

'Conditional table movement buttons
  dlvc=dlvc+dlvs*2 : dlhc=dlh+dlhs*.65
  dialogloc 0,dlhc,dlvc
  dlhc=dlhc:dialogloc 2,dlhc,dlvc:dialogask mc,6,51010,0,"_&Up_"
  dlhc=dlhc:dialogloc 2,dlhc,dlvc+dlvs:dialogask mc,6,51000,0,"&Down"
  dialogloc 2,dlhc-24,dlvc+dlvs/2:dialogask mc,6,51030,0,"&Back"
  dialogloc 2,dlhc+24,dlvc+dlvs/2:dialogask mc,6,51020,0,"&Over"

'Quant buttons
  dlvc=dlvc-dlvs*1.5:dlhc=dlh
  dialogloc 1,dlhc,dlvc:dialogask um(sc,9),3+128,0,0,"Fit_to:  Linear Quadratic Cubic"
  dlhc=dlhc+dlhs:dialogloc 1,dlhc,dlvc:dialogask um(sc,10),3+128,0,0,"Forced_intercept:"


'Editing/file buttons
  dlhc=dlh+dlhs*1.6:dlvc=dlvc+dlvs*2
  dialogloc 0,dlhc,dlvc:dialogask mc,6,50600,53000,"ReCalc Options"
  dlvc=dlvc+dlvs*1.1
  dialogloc 0,dlhc,dlvc:dialogask mc,6,52400,0,"_____E&xit_____"

50400   dialogoff 14,0,50500

50500  for i=0 to LR-SR
      dialogset 16,i*7+6
  next
  pos=12+(lr-sr)*7
  if SR>0 then dialogset 8,pos else dialogset 16,pos
  if SR+13<na then dialogset 8,pos+1 else dialogset 16,pos+1
  if SC>0 then dialogset 8,pos+2 else dialogset 16,pos+2
  if SC+1<m then dialogset 8,pos+3 else dialogset 16,pos+3
  gosub 57000      'get X and Y values
  gosub 57200      'and calc coeff etc.
  gosub 57205      'set up for display
  gosub 57500      'and repaint display
  dialogoff 20
  onerror 52400
  goto 50400

' Recalc
50600   gosub 57000
  gosub 57200
  gosub 57205
  gosub 57500
  goto 50500
```

```
' Down button
51000   dialogoff -10
  SR=SR+14 : LR=LR+14
  if LR>Na-1 then LR=Na-1
  goto 50200

'Up button
51010   dialogoff -10
  SR=SR-14
  LR=SR+13
  goto 50200

'Over button
51020   dialogoff -10
  sc=sc+1
  if sc>m-1 then sc=m-1
  goto 50200

' Back button
51030   dialogoff -10
  SC=SC-1
  goto 50200

'Choose file for row
52230   cr=sr:goto 52250
52231   cr=sr+1:goto 52250
52232   cr=sr+2:goto 52250
52233   cr=sr+3:goto 52250
52234   cr=sr+4:goto 52250
52235   cr=sr+5:goto 52250
52236   cr=sr+6:goto 52250
52237   cr=sr+7:goto 52250
52238   cr=sr+8:goto 52250
52239   cr=sr+9:goto 52250
52240   cr=sr+10:goto 52250
52241   cr=sr+11:goto 52250
52242   cr=sr+12:goto 52250
52243   cr=sr+13:goto 52250


'Select training spectrum from file selector
52250   temp=0:temp2=0
  ONERROR 52258
  MENUFILE $temp, "Training Spectrum File Name?  ",2
  ONERROR 55200
  LOADSPC $temp
  onerror -1
  gosub 52600          'scale and orient
  gosub 57110:if spcmatch=0 then dialogoff -10:dialogkill=1:gosub 57150
  strname 3,temp2,$temp      'basename
  temp=0                'lower case
  strname 7,temp,$temp2
  sname(cr)=temp
  if nsubs(#s)>1 then gosub 52500
  noshow
  onerror 52258
52252   temp=0:temp2=0
  menufile $temp,"",3
  onerror 55200
  loadspc $temp
  onerror -1
  gosub 52600          'scale and orient
```

```
  gosub 57110:if spcmatch=0 then dialogoff -10:dialogkill=1:gosub 57150
  onerror -1
  mc=1:i=cr
  gosub 56550
  cr=cr+1
  strname 3,temp2,$temp      'basename
  temp=0                     'lower case
  strname 7,temp,$temp2
  sname(cr)=temp
  if nsubs(#s)>1 then gosub 52500
  noshow
  onerror 52254
  goto 52252
52254  onerror -1
  dialogoff -10
  goto 50200

52258  onerror -1
  if dialogkill=1 then dialogkill=0:goto 50200  'if dialog killed restart from scratch
  goto 50500

'INsert
52260  cr=sr:goto 52280
52261  cr=sr+1:goto 52280
52262  cr=sr+2:goto 52280
52263  cr=sr+3:goto 52280
52264  cr=sr+4:goto 52280
52265  cr=sr+5:goto 52280
52266  cr=sr+6:goto 52280
52267  cr=sr+7:goto 52280
52268  cr=sr+8:goto 52280
52269  cr=sr+9:goto 52280
52270  cr=sr+10:goto 52280
52271  cr=sr+11:goto 52280
52272  cr=sr+12:goto 52280
52273  cr=sr+13:goto 52280

52280  dialogoff -10
  mc=1:i=cr
  gosub 56550
  goto 50200

'Delete at current location
52290  cr=sr:goto 52310
52291  cr=sr+1:goto 52310
52292  cr=sr+2:goto 52310
52293  cr=sr+3:goto 52310
52294  cr=sr+4:goto 52310
52295  cr=sr+5:goto 52310
52296  cr=sr+6:goto 52310
52297  cr=sr+7:goto 52310
52298  cr=sr+8:goto 52310
52299  cr=sr+9:goto 52310
52300  cr=sr+10:goto 52310
52301  cr=sr+11:goto 52310
52302  cr=sr+12:goto 52310
52303  cr=sr+13:goto 52310

52310  dialogoff -10
  mc=1:i=cr
  gosub 56650
  goto 50200
```

```
' Exit from main dialog
52400   dialogoff -10
  dialogoff -20
  viewonly "Quantbas.vwh"
  if showplot=1 then noshow
  showplot=0
  viewtrace 1
  gosub 57190  'autoscale and reorient
  see
  comp=sc       '~
  return

' +++++++Subroutine to loading multifile and ask for subfile #
52500   dialogoff -10
  dialogkill=1
  dialogon "Multi-file detected"
  print "The selected spectrum is a multi-file and contains"
  print "multiple spectra.  Choose the sub-file number to use."
  print ""
  print "Sub-file range: 1 - ";nsubs(#s)
  print ""
  mc=1
  dialogask mc,1,1,nsubs(#s),"Sub-file to use"
  print ""
  dialogoff
  string $sname(cr)=$sname(cr)+" #"+mc
  return

'orient and auto
52600   if getxtype0=1 and getleft0<getright0 then xflip
  if getxtype0<>1 and getleft0>getright0 then xflip
  setleft getffp0,getflp0:autoscale
  return

'++++++++++++++++
' Options
'++++++++++++++++
53000   dialogoff -10
  onerror 53300
  dialogon "Load or Save PLS Style Concentration File (CFL)"
  dialogask mc,4,53100,53200,"Save Load Exit"
  dialogoff 10

'Save to new name
53100   onerror 53110
  menufile $cflname,"Save to CFL As   ",1+8
  onerror -1
  gosub 56700
53110   onerror -1
  dialogoff -10
  goto 50200

'Load button
53200   onerror 50200
  menufile $cflname,"Choose CFL file to load   *.CFL"
  onerror -1
  gosub 56800
53210   onerror -1
  dialogoff -10
  goto 50200
'Exit
53300   onerror -1
  dialogoff -10
```

```
  goto 50200

'Save to new name
'53210   temp=0:string temp="Current_File_"+$TEXT+"  *.CFL"
  onerror 50200
  menufile name,$temp
  onerror -1
  reverse name : mc=index0(name-92) : reverse name
  mc=256-mc : temp=0 : temp=name(mc,255) : mc=index0(temp)
  if temp(mc-4)<>46 then string $temp=$temp+".CFL"
  TEXT=temp
  onerror -1
  gosub 56700
  goto 50200

'Save button
'53320   onerror -1
  gosub 56700
  goto 59010

'Abort edits button
'53340   onerror -1
  goto 59010

'Return to editor button
'53360   onerror -1
  goto 50200

'Error message - CFL file name
55200   dialogoff -10
  onerror 50200
  dialogon "WARNING!"
  dialogloc 1,-100
  PRINT "File bad or does not exist.  If you choose to use this filename"
  print "QuanBasic may not be able to find it when calibrating or running"
  print "diagnostics."
  dialogloc 3,0
  dialogask mc,4,52255,50200,"OK Cancel"
  dialogoff

'New Component
56505   'M=M+1
  DIM YTEMP(M,Na)
  TRANSPOSE Y
  ytemp=y
  free y
  dim y(m,Na)
  y=ytemp
  transpose y
  free ytemp
  return

'New sample
56550   Na=Na+1
  DIM SNTEMP(Na,13), YTEMP(Na,M), cktemp(na)
  FOR K = 0 TO I
    SNTEMP(K) = SNAME(K)
    YTEMP(K) = Y(K)
    cktemp(k)=ck(k)
  NEXT K
56555   SNTEMP(I+1)=32:Ytemp(i+1)=0:cktemp(i+1)=0
  IF I=Na-2 GOTO 56560
  FOR K = I+1 TO Na-1
```

```
    SNTEMP(K+1) = SNAME(K)
    YTEMP(K+1) = Y(K)
    cktemp(k+1)=ck(k)
  NEXT K
56560  FREE SNAME, Y, ck
  DIM SNAME(Na,13), Y(Na,M), ck(na)
  SNAME = SNTEMP : Y = YTEMP: ck=cktemp
  FREE SNTEMP, YTEMP,cktemp
  sr=int(sr/14)*14
  lr=sr+13:if lr>na-1 then lr=na-1
  return

'Delete component
56605  DIM YTEMP(M,Na)
  TRANSPOSE Y
  FOR K = 0 TO comp-1
    YTEMP(K) = Y(K)
  NEXT K
  IF comp=M GOTO 56610
  FOR K =comp+1 TO M
    YTEMP(K-1) = Y(K)
  NEXT K
56610  FREE Y
  DIM Y(M-1,Na)
  Y = YTEMP
  FREE YTEMP
  TRANSPOSE Y
  return

'Delete sample
56650  if na>1 then  Na=Na-1 else return
  DIM SNTEMP(Na,13), YTEMP(Na,M),cktemp(na)
  FOR K = 0 TO I-1
    SNTEMP(K) = SNAME(K)
    YTEMP(K) = Y(K)
    cktemp(k)=ck(k)
  NEXT K
  IF I=Na GOTO 56660
  FOR K = I+1 TO Na
    SNTEMP(K-1) = SNAME(K)
    YTEMP(K-1) = Y(K)
    cktemp(k-1)=ck(k)
  NEXT K
56660  FREE SNAME, Y, ck
  DIM SNAME(Na,13), Y(Na,M),ck(na)
  SNAME = SNTEMP : Y = YTEMP:ck=cktemp
  FREE SNTEMP, YTEMP, cktemp
  if sr>na-1 then sr=na-1
  sr=int(sr/14)*14
  lr=sr+13:if lr>na-1 then lr=na-1
  return

'+++++++
'Save to filename in CFLNAME Subroutine
56700  n=na-sum(ck) : i=index0(memo)
  if sum(neg(memo(0,i-1)=32))=i or sum(memo)=0 then string memo="PLSPlus Calibration File"
  OPEN "C",#1,$cflname
  PRINT #1,M;"   =NUMBER OF COMPONENTS"
  PRINT #1,N;"   =NUMBER OF SAMPLES"
  PRINT #1,$memo
  FOR I=0 TO M-1
    PRINT #1,$CNAME(I)
  NEXT I
```

```
  TEMP=32
  FOR I=0 TO 8
    TEMP(I*8)=49+I
  NEXT I
  TEMP(72)=49:TEMP(73)=48
  temp(80)=0
  PRINT #1,$TEMP
  PORTOUT -30,8
  FOR I = 0 TO Na-1
    If ck(i)=1 then temp=0:temp(1,256)=sname(i):temp(0)=42
    if ck(i)=1 then print #1,$temp
    if ck(i)=0 then PRINT #1,$SNAME(I)
    IF M=1 THEN J=0:GOTO 56706
    FOR J = 0 TO M-2
      PRINT #1, Y(I,J),"";
    NEXT J
56706     PRINT #1, Y(I,J)
  NEXT I
  CLOSE #1
  return

'+++++++
'Load CFL from TEXT routine
56800   free m,n,na,cname,sname,ck,y,sc,sr:v=0
      ' ONERROR 52010
  OPEN "O",#1,$Cfiname
  ONERROR -1

'   temp=0 : temp=TEXT                'remove path from CFL file
'   reverse temp : mc=index0(temp-92) : reverse temp
'   mc=256-mc : i=index0(temp) : TEXT=0 : TEXT=temp(mc,i-1)

'   plotbox .03,0.03,.97,.09,3,5
'   printhue yellow : printloc .05,.045 : printfont 6,2,10,0,"Arial"
'   PRINT "Reading calibration information - please wait."
      ' ONERROR 52020

' First calculate the actual number of calibration samples
  INPUT #1," ", M
  INPUT #1," ", N
  INPUT #1," ",$temp
  DIM Cname(M,15) 'COMPONENT NAMES
  FOR I = 0 TO M-1
    INPUT #1," ",$Cname(I)
  NEXT
  INPUT #1," ",$TEMP
  I = 0
  dim tck(10000)          'no more than 10,000 samples!
   onerror 56850
56820     INPUT #1," ",$temp
    if temp(0)=42 then tck(i)=1     'If started, set check flag
    FOR J = 0 TO M-1
      INPUT #1, DUMMY;
    NEXT J
  i=i+1
  goto 56820

56850   onerror -1
  NA=i          'Actual samples
  'if n>na or na<1 then goto 52100
  OPEN "O",#1,$cfiname    'Reset file pointer
  dim ck(NA)
  ck=tck          'Check array
```

```
  free tck

'Now read real data in
  INPUT #1," ", M
  INPUT #1," ", N
  INPUT #1," ",$memo
  ONERROR -1
  DIM Sname(NA,13)  'TRAINING SPECTRA FILE NAMES
  DIM Y(NA,M)

  FOR I = 0 TO M-1
    INPUT #1," ",$Cname(I)
  NEXT
  INPUT #1," ",$TEMP
  FOR I = 0 TO NA-1
    INPUT #1," ",$Sname(I)
    if ck(i)=1 then temp=sname(i):sname(i)=temp(1,255)
    FOR J = 0 TO M-1
      INPUT #1, DUMMY;
      Y(I,J)=DUMMY
    NEXT J
  NEXT I
  CLOSE #1
  if na<13 then LR=na-1 else LR=SR+13
  ref=0
  mm=m+ref
  mref=ref
  comp=0
'Kill um when loading cfl
  free um
  dim UM(m,20)
  transpose um
  um(0)=-999999
  um(1)=-999999
  um(3)=-999999
  um(4)=-999999
  um(6)=-999999
  um(7)=-999999
  transpose um
  noshow:noshow
  showplot=0

56870   temp=0:temp=sname(0)
  gosub 57180      'loadspc
  if spcOK=0 then return
  gosub 57190      'orient and autoscale
  return


'++++++++++++++++++++++++++++++
' Will get all spectrum X values and concentration y values and stick them in
' Xk and Yk
'++++++++++++++++++++++++++++++
57000   DataOK=0
  free yk,xk
  if caltyp=2 then ns=na-sum(ck) else ns=1
  if ns=0 then string message,"No Data Specified":goto 57050
  dim yk(m,ns),xk(m,ns)

  crow=0            'row counter
  if caltyp<2 then goto 57010
  for i=0 to na-1
    if ck(i)=1 then goto 57030
```

```
   temp=0:temp=sname(i)          'load spc and abort if error
   gosub 57180
   if spcOK=0 then goto 57280
   onerror -1
57010    if ref=false then goto 57020
   comp=m
   gosub 41000
   if rdefined=false then goto 57290
   if um(comp,8)>1 then refval=area else refval=height
57020     for j=0 to m-1
     comp=j
     if caltyp=2 then Yk(j,crow)=Y(i,j)
     gosub 41000
     if rdefined=false then goto 57300
     if um(j,8)>1 then samval=area else samval=height
     if ref=1 then samval=samval/refval
     Xk(j,crow)=samval
   next j
   if caltyp<2 then goto 57040
   crow=crow+1
   noshow
57030    next i
57040    DataOK=1
57050    return


'++++++++Verify current and previous spectral range and point spacint subroutine
57110   spcmatch=0
  if fp<>getffp0 then string message="Frequency of first data point":goto 57120
  if lp<>getflp0 then string message="Frequency of last data point":goto 57120
  if xtype<>getxtype0 then string message="X unit type":goto 57120
  if ytype<>getytype0 then string message="Y unit type":goto 57120
  if interval<>(lp-fp)/(npts(#s)-1) then string message="Data point spacing":goto 57120
  spcmatch=1
57120   return

'++++++++save range and spacing info to verify pred spectra cover same range and res and dtypes
57130   transpose um
  cmax=max(um(0))
  cmin=min(um(0))
  for i=1 to 7
    if i=2 then next
    if i=5 then next
    if max(um(i))>cmax then cmax=max(um(i))
    if min(um(i))<cmin then cmin=min(um(i))
  next
  transpose um
  cmax=freq(#S(cmax))
  cmin=freq(#s(cmin))
  if fp>lp then mfp=cmax:mlp=cmin
  if lp>fp then mfp=cmin:mlp=cmax
  minterval=(mlp-mfp)/((npts(#s(mfp,mlp))-1))
  mxtype=xtype
  mytype=ytype
  return

'++++++++verify spectral range and spacing for prediction
57140   mthmatch=0
  if fp>lp then goto 57144
  if mfp<fp then string message="Spectral range of method exceeds spectral range of spectrum":goto 57148
  if mlp>lp then string message="Spectral range of method exceeds spectral range of spectrum":goto 57148
  goto 57146
57144   if mfp>fp then string message="Spectral range of method exceeds spectral range of spectrum":goto 57148
```

```
  if mlp<lp then string message="Spectral range of method exceeds spectral range of spectrum":goto 57148
57146   if mxtype<>xtype then string message="X unit type":goto 57148
  if mytype<>ytype then string message="Y unit type":goto 57148
  if abs((interval-minterval)/minterval)>.01 then string message="Datapoint spacing does not match method":goto
57148
  mthmatch=1
57148   return

'++++++++Warn the user that range/spacing of new spc does not match
57150   onerror 57152
  dialogon "Warning"
  print "The spectrum you have just chosen: ";$temp
  print "does not match the original model spectrum in the following way:"
  print ""
  print $message
  print ""
  print "Be aware this may cause errors in quantitation or prevent"
  print "the use of this spectrum in the quantitation."
  print ""
  dialogask mc,4,57152,57152,"__&OK__"
  dialogoff
57152   onerror -1
  return

'++++++++Load spc file subroutine
' pass filename in temp
57180   spcOK=0
  mc=index0(temp-35)    'old LC multifile convention?
  if mc<256 then temp(mc-1)=0  'if so cut out the mf number
  onerror 57185
  LOADSPC $TEMP         'load the file
  if mc<256 then temp2=0:temp2=temp(mc+1,255):string $temp2,-20:loadspc #v  'goto sub#
  spcOK=1
57185   onerror -1
  return

' ++++++++orient and autoscale spc subroutine
57190   if getsfile0=0 then goto 62700          'if no spc disaster
  if getxtype0=1 and getleft0<getright0 then xflip
  if getxtype0<>1 and getleft0>getright0 then xflip
  setleft getffp0,getflp0:autoscale
  fp=getffp0:lp=getflp
  interval=(lp-fp)/(npts(#s)-1)
  xtype=getxtype0
  ytype=getytype0
  return


'++++++++++++++++++++++++++++++
' Takes Yk and Xk values and calculates coeff matrix and regression statistics
' for component sc
'++++++++++++++++++++++++++++++
' Update graph
57200   CalibOK=0
  if DataOK=0 then return
  sorder=um(sc,9)+2
  if um(sc,10)=1 then sorder=sorder-1
  if ns<=sorder then string message,"Not Enough Data Points":goto 57202
  'Create local matrix variables for performing the regression.
    free coeffMatrix, tempMatrix1, xMatrix, yActual,yestimate
    dim coeffMatrix(sorder, 1)        'column matrix of coefficients
    dim tempMatrix1(sorder, sorder)  'temporary square matrix
    dim xMatrix(sorder, ns)        'matrix of X-values raised to varying powers (X^0, X^1, X^2, ...)
```

```
    dim yActual(ns,1)          'column matrix of actual Y-values
    dim yEstimate(ns,1)        'column matrix of estmate y values

  'Fill the X-matrix with X-values raised to varying powers (X^0, X^1, X^2, ...).
    for power = 0 to (sorder - 1)
      if um(sc,10)=0 then xMatrix(power) = Xk(sc) ^ power
      if um(sc,10)=1 then xMatrix(power) = Xk(sc) ^ (power + 1)
      if um(sc,10)=0 > 1 and power = 0 then xMatrix(power) = xMatrix(power) + neg(xMatrix(power) = 0)

    next power

  'Perform the regression.
    setzero 0
    tempMatrix1 = xMatrix [#T] xMatrix
    onerror 57202
    string message,"Non Invertable Matrix"
    invertdp tempMatrix1
    onerror -1
    yActual = Yk(sc)
    coeffMatrix = tempMatrix1 [#] (xMatrix [#] yActual)

     'Calculate r-squared.
     rSquared = -999999
  yEstimate = xMatrix [T#] coeffMatrix
  yAverage = sum(yActual) / ns
  string message,"Invalid Concentration Values"
  onerror 57202
  rSquared = sumd2(yEstimate, yAverage) / sumd2(yActual, yAverage)
  onerror -1
  string message,"OK"
  CalibOK=1
57202   onerror -1
  return

'Set up for display
    'Store peak calibration quantities and responses into temporary arrays.
57205   if calibOK=0 then return
  maxQuantity = max(yActual)      'largest calibration quantity
  minQuantity = min(yActual)      'smallest calibration quantity
  maxResponse = max(Xk(sc))       'largest calibration response
  minResponse = min(Xk(sc))       'smallest calibration response

    'Calculate ranges for the quantities and responses from their maximum and minimum values.
  responseRange = maxResponse - minResponse
  quantityRange = maxQuantity - minQuantity
  if responseRange = 0 then String message,"Intensity =0":goto 57250          'response range is not allowed to
be 0
  if quantityRange = 0 then quantityRange = 1          'quantity range is not allowed to be 0

    'Set up the axis limits so that each range covers 71% of it's corresponding axis.
'   leftX = minResponse - responseRange * 0.2
'   rightX = maxResponse + responseRange * 0.2
'   topY = maxQuantity + quantityRange * 0.2
'   bottomY = minQuantity - quantityRange * 0.2

bottomy = minResponse - responseRange * 0.2
topy = maxResponse + responseRange * 0.2
rightx = maxQuantity + quantityRange * 0.2
leftx = minQuantity - quantityRange * 0.2

    'Create a temporary array of X-values for calculating the calibration curve.
  fillbeg bottomy
  fillinc (topy-bottomy)/1023        'allow for 1024 points or X-values
```

```
  free temparray1,temparray2:dim temparray1(1024),temparray2(1024)
  tempArray1 = fill(tempArray1)

    'Create a trace in memory for storing the calibration curve.
  if showplot=1 then goto 57210
  newspc calCurve(1024,128)
  newslot=getsfile()
  oldslot=newslot+1
  showplot=1
57210  if getffp() > getflp() then xflip  'flip trace around if necessary
  setffp bottomy, topy
  string "Absorbance", -18
  string "Concentration", -16

    'Based on the calibration model, jump to the appropriate code and calculate the curve.
57220   'goto (57220 + sorder)

    'Calculate normal regression function.

if um(sc,10)=1 then goto 57222
57221   temparray2 = coeffmatrix( 0)
  for coefficient = 1 to (sorder - 1)
    temparray2 = temparray2 + (fill(temparray2) ^ coefficient) * coeffmatrix(coefficient)
  next coefficient
  goto 57250

    'Calculate regression function using a forced zero intercept.
57222   temparray2 = 0
  for coefficient = 0 to (sorder - 1)
    temparray2 = temparray2 + (fill(temparray2) ^ (coefficient + 1)) * coeffmatrix(coefficient)
  next coefficient

57250  onerror -1
  #s.x=temparray2
  #s.y=temparray1
  return

'+++++++Some error handling

' Error on loading spectrum
57280  i=na-1     'clear loop
  next i
  string message="Invalid Spectrum"
  return

' Undefined reference
57290  i=na-1     'clear loop
  next i
  if caltyp=2 then noshow
  string message="Undefined Reference Region"
  return

'Undefined region
57300  j=m-1      'clear loops
  next j
  i=na-1
  if caltyp=2 then noshow:next i  '<dk> added conditional next i 06/08/94
  string message="Undefined Region"
  return

'*********************************************
'Display/Repaint Routine for Calibration Curve
'*********************************************
```

```
    'Create empty display box with grids (i.e. no calibration curve).
57500   'blank NAVY       'clear the display and set the background color to navy
  'blankbox       'blank the display box
  'axes 1       'draw the blank display box and the axes
  if CalibOK=0 then blankbox:return

    'Scale and display calibration curve.
57520   viewtrace 1
  setleft leftX, rightX
  setmin bottomY, topY
  see
'  edisplay
'  edisplay
  axes 1

'Display Calibration points for calib

  crow=0
  xrng=abs(getleft0-getright0)          'Plot numbers
  yrng=abs(getmax0-getmin0)
  printhue black,grey,0
  printfont 6,2,9,0,"Arial"
  for i=0 to na-1
    if ck(i)=1 then goto 57522
    cx=neg((yactual(crow)-getleft0)/xrng)+0.005 :  cy=neg((xk(sc,crow)-getmin0)/yrng)+0.022
    if i+1>=10 then cx=cx+0.005
    if i+1>=100 then cx=cx+0.005
    printloc cx,cy : print i+1
    crow=crow+1
57522    next

    'Print the first 4 calibration coefficients in Arial 9 pt boldface.
  a=0:b=0:c=0:d=0
  mc=0
  if um(sc,10)=1 then goto 57525
  a=coeffmatrix(mc)
  mc=mc+1
  if mc>sorder-1 then goto 57530
57525   b=coeffmatrix(mc)
  mc=mc+1
  if mc>sorder-1 then goto 57530
  c=coeffmatrix(mc)
  mc=mc+1
  if mc>sorder-1 then goto 57530
  d=coeffmatrix(mc)

57530   return

'Display calibration points for predict
57600   crow=0
  viewtrace 1
  setleft leftX, rightX
  setmin bottomY, topY
  viewedit 0,2,4,getsfile0
  see
  axes 1
  xrng=abs(getleft0-getright0)          'Plot numbers
  yrng=abs(getmax0-getmin0)
  printhue black,grey,0
  printfont 6,2,9,0,"Arial"
  for i=0 to na-1
    if ck(i)=1 then goto 57622
    cx=neg((yactual(crow)-getleft0)/xrng)+0.005 :  cy=neg((xk(comp,crow)-getmin0)/yrng)+0.022
```

```
    if i+1>=10 then cx=cx+0.005
    if i+1>=100 then cx=cx+0.005
    printloc cx,cy : print i+1
    crow=crow+1
57622   next

  printhue blue,grey,0

  cx=-.01:cy=-.9
  printloc cx,cy:print $cname(comp)
  printfont 6,2,9,0,"wingdings"
  cx=neg((yp(comp)-getleft0)/xrng)+.015:cy=neg((x(comp)-getmin0)/yrng)+.022
  printhue blue,grey,0
  printloc cx,cy:print "u"

  plotset 1
  cx=(yp(comp)-getleft0)/xrng:cy=(x(comp)-getmin0)/yrng
  penup:plot 0,cy,blue,2:plot cx,cy,blue,2:plot cx,cy,blue,2:plot cx,0,blue,2
  return

' If error exit through here
59000   v=99

' If returning ok, exit through here
59010   free title,name,sc,sr,lc,na,dlv,dlh,dlvs,dlhs,dlhc,dlvc,ckcnt
  free ofp,olp,fp,lp,onp,np,j,k,m,n,cname,sname,ck,y,tck,dummy,memo
  return

'++++++Onpaint routine for calibration
59500   see
  gosub 57200
  gosub 57500
  return

'*******External Procedure call
' Take current #s and return raw values for all components in Yp
' Arguments methodname(string), ReportOption,FileOption,PrintOption
'ReportOption=0 display results and data to screen, 1=results only, 2=nothing to screen
'FileOption=0 don't save, 1=save to SPCNAME.RPT
'PrintOption=0 no print 1=print to printer
'verbose=1 give all errors in a dialog box

'External entry point
60000   procedure tmethname(256),ReportOption,FileOption,PrintOption,verbose
  _v=0        'initialize error
  gosub 5        'allocate
  methname=tmethname
  ExtCall=1       'call was external
'Internal entry point
60050   onerror -1
  onpaint 0
  if ReportOption<>0 and PrintOption=0 then goto 60100
  viewonly "QBpred.vwh"
60100   call=1
  if getsfile0=0 then _v=4:goto 62700
  if extCall=1 then gosub 3310     'load method
  if extCall=1 and methloaded=0 then _v=2:goto 62700
  if extCall=1 goto 60160

'Load spectrum if not in memory
  if getsfile0<>0 then goto 60150
  temp=0
  menufile temp,"Choose New Spectrum  *.spc"
```

```
  gosub 57180
  if spcOK=0 then goto 62700
  if nsubs(#s)>1 then gosub 63000    'multi-file?

'Choose method
60150   gosub 3300       'choose method
  if methloaded=0 then goto 62700
60160   gosub 57190        'get spc range/orinetation
  gosub 57140      'does in match
  if caltyp<>2 and ReportOption=0 then viewedit -1,1
  if caltyp<>2 and PrintOption=1 then viewedit -1,1
  see
  if verbose=0 and mthmatch=0 then _v=3:goto 62700
  if verbose=1 and mthmatch=0 then gosub 57150:goto 62700
  if goodcal=0 then beep:beep:goto 62700
  gosub 62000      'extract values from current #s
'Now get predicted results in yp
  free Yp:dim yp(m)
  if caltyp=2 then gosub 62100
  if caltyp=1 then gosub 62200
  if caltyp=0 then yp=x
'Do report
  if fileoption=1 then gosub 62600
  if printoption =1 then gosub 62550
  if reportoption<2 then comp=0:goto 62400
  _v=1
  goto 62700

'++++++++Get spectrum X values for all components and return in X
62000   free x:dim x(m)
  if ref=false then goto 62020
    comp=m
    gosub 41000
    if rdefined=false then beep:goto 62700
    if um(comp,8)>1 then refval=area else refval=height
62020     for j=0 to m-1
      comp=j
      gosub 41000
      if rdefined=false then beep:62700
      if um(j,8)>1 then samval=area else samval=height
      if ref=1 then samval=samval/refval
      X(j)=samval
    next j
      return

'++++++++for multipoint calibrations predict all components return in Yp
62100   free rsquaredp:dim rsquaredp(m)
  free xmatrix,coeffMatrix
  for i=0 to m-1
    sorder=um(i,9)+2
    if um(i,10)=1 then sorder=sorder-1
    dim xMatrix(sorder, 1)
    dim coeffMatrix(sorder, 1)

    for j=0 to sorder-1
      coeffmatrix(j,0)=um(i,j+11)
      if um(i,10)=0 then xMatrix(j) = X(i) ^ j
      if um(i,10)=1 then xMatrix(j) = X(i) ^ (j + 1)
      if um(i,10)=0 > 1 and j = 0 then xmatrix(j) = xmatrix(j) + neg(xMatrix(j) = 0)
    next j
    yp(i) = xmatrix [T#] coeffmatrix
    rsquaredp(i)=um(i,11+sorder+1)
    free xmatrix,coeffmatrix
```

```
  next i
  return

'+++++++get coeff and xmatrix of selected comp to view plot
62150    free xmatrix,coeffMatrix,yactual
   i=comp
   sorder=um(i,9)+2
   if um(i,10)=1 then sorder=sorder-1
   dim xMatrix(sorder, 1)
   dim coeffMatrix(sorder, 1)
   for j=0 to sorder-1
     coeffmatrix(j,0)=um(i,j+11)
     if um(i,10)=0 then xMatrix(j) = X(i) ^ j
     if um(i,10)=1 then xMatrix(j) = X(i) ^ (j + 1)
     if um(i,10)=0 > 1 and j = 0 then xmatrix(j) = xmatrix(j) + neg(xMatrix(j) = 0)
   next j
   dim yactual(ns)
   yactual=yk(i)
   return

'+++++++for sp calibration prediction
62200   transpose um
  Yp=X*(um(11)/um(12))
  transpose um
  return

'+++++++Routine to display report
62400   onpaint 62900
  if showplot=1 then setfile oldslot 'make sure we point to data when entering
  dlv=4       'initial vertical
  dlh=-100    'initial horizontal right
  dlvs=12     'vertical spacing
  dlhs=80     'horizontal spacing
  dlhc=dlh    'current horizontal
  dlvc=dlv    'current vertical
  gosub 62480  'fill in header
  if caltyp=2 then sp=dlhs*.55 else sp=0
  if showplot=1 then setfile newslot
' Main Predict report dialog

  v=keyboard(2)      '?????

  dialogon  "QuantBasic Report"
  dialogloc 1,dlhc,dlvc
  print "Quant Type: ";$temp
  dialogloc 1,dlhc+dlhs*1+sp,dlvc
  print $temp3
  dlvc=dlvc+dlvs:dialogloc 1,dlhc,dlvc
  print "Method: ";$Methname
  dlvc=dlvc+dlvs:dialogloc 1,dlhc,dlvc
  print "Sample: ";$temp2

  dlvc=dlvc+dlvs*1.2:dialogloc 0,dlhc+20,dlvc
  'dialogask name,2,20,20,"Analyst"
  dlvc=dlvc+dlvs*1.2:dialogloc 0,dlhc+20,dlvc
  'dialogask temp5,2,20,20,"Sample ID"

  dlvc=dlvc+dlvs*1.5
  dlhc=dlh:dialogloc 1,dlhc,dlvc+dlvs*.7:print "Component"
  dialogloc 1,dlhc+dlhs*1.1,dlvc+dlvs*.7:print "Concentrations"
  if caltyp=2 then dialogloc 1,dlhc+dlhs*1.8,dlvc+dlvs*.7:print "R-Squared"
  dlvc=dlvc+dlvs*1.8
  for i=0 to m-1
```

```
    dialogloc 1,dlhc,dlvc:print $cname(i)
    dialogloc 1,dlhc+dlhs*1.1,dlvc:print yp(i)
    if caltyp=2 then dialogloc 1,dlhc+dlhs*1.8,dlvc:print rsquaredp(i)
    dlvc=dlvc+dlvs
  next i
  dlvc=dlvc-(12*(m+1)+10)
  dlvc=dlvc+dlvs:dlhc=dlhc-5:dialogloc 1,dlhc,dlvc-dlvs
  dialogtxt 3,145+sp,(12*(m+1)+10),"Results"

  dlvc=dlvc+dlvs*.6:dialogloc 1,dlhc,dlvc
  dialogtxt 5,145+sp,1,""
  dlvc=dlvc-dlvs*1.2
  dlhc=dlhc+dlhs*1.1:dialogloc 1,dlhc,dlvc
  dialogtxt 5,1,(12*(m+1)+6),""
  if caltyp=2 then dialogloc 1,dlhc+dlhs*.7,dlvc:dialogtxt 5,1,(12*(m+1)+6),""
  print ""
  dlvc=dlvc+(12*(m+1)+16):dlhc=dlh:dialogloc 3,0

  if ReportOption=0 and m>1 then dialogask mc,4,62420,62430,"Print Save View_Next Exit"
  if ReportOption=1 or M=1 then goto 62421
  'if ReportOption=1 or m=1 then dialogask mc,4,62420,62421,"Print Save Exit"
  if extcall=0 then dialogloc 3,0:dialogask mc,6,62460,62470,"New_&Spectrum New_&Method"

62410   if ReportOption=0 then dialogoff 10+4,0,62412 else dialogoff 10+4

62412   if caltyp<>2 then goto 62414
  viewtrace 1
  calibOK=1:gosub 62150:gosub 57205:gosub 57600
  viewtrace 2
  setfile oldslot
  gosub 57190:gosub 41000:gosub 43000
  setfile newslot
  goto 62410
62414   viewtrace 1
  gosub 57190:gosub 41000:gosub 43000
  goto 62410

' Print
62420   gosub 62550
  goto 62400
62421   goto 62430
62422   goto 62450
' Save
62430   gosub 62600
  goto 62422
  'goto 62400

'View next
62440   comp=comp+1:if comp>m-1 then comp=0
  sc=comp
  goto 62412
'End
62450   _v=1
  goto 62700

' new spc
62460   dialogoff -10
  onpaint 0
  if showplot=1 then setfile newslot:noshow:showplot=0
  temp=0
  onerror 62468
  menufile temp,"Choose_New_Spectrum  *.spc"
  onerror -1
```

```
  gosub 57180
  if spcOK=0 then goto 62468
  if nsubs(#s)>1 then gosub 63000   'multi-file
  setfile getsfile0+1:noshow        'kill old
  viewtrace 1
62468   onerror -1
  viewonly "qbpred.vwh"
  goto 60160

' new method
62470   dialogoff -10
  onpaint 0
  if showplot=1 then setfile newslot:noshow:showplot=0
  goto 60050

'-------SUB fill in header
62480   if caltyp=0 then string temp="Raw Values"
  if caltyp=1 then string temp="Single Point"
  if caltyp=2 then string temp="Multipoint"
  string temp3,getsname
  strname 7,$temp2=$temp3
  for i=0 to m-1
    temp3=cname(i)
    mc=index0(temp3)
    temp3(mc,13)=32
    temp3(14)=0
    cname(i)=temp3
  next
  v=clock(-7)*(2^20)+clock(-6)*(2^16)+clock(-5)*(2^11)+clock(-4)*(2^6)+clock(-3)
  string temp3,-55
  return

'+++++++Print Report
' Just text
62500   dialogoff -10
  printline 1
  onerror 62590
  lprinton
  onerror -1
  print "QuantBasic Report"
  print "Quant Type: ";$temp,$temp3
  print "Method: ";$Methname
  print "Sample: ";$temp2
  print "Analyst: ";$name
  print "Sample ID: ";$temp5
  if caltyp <> 2 then print "Component","Concentration"
  if caltyp=2 then print "Component","Concen","R-Squared"
  for i=0 to m-1
    if caltyp<> 2 then print $cname(i),yp(i)
    if caltyp=2 then print $cname(i),yp(i),rsquaredp(i)
  next i
  lprintoff
  return

' Print all
62550   onpaint -1
  dialogoff -10
  onerror -1
'loop
  for comp=0 to m-1
    sc=comp
    if caltyp<>2 then goto 62570
    viewtrace 1
```

```
    calibOK=1:gosub 62150:gosub 57205:gosub 57600
    viewtrace 2
    setfile oldslot
    gosub 57190:gosub 41000:gosub 43000
    setfile newslot
    goto 62570
62570     printline 1
    printhue black
    printfont arial,normal,10
    gosub 62480      'fill in header
    xloc=.05
    yloc=.9
    yinc=.025
    xinc=.1
    onerror 62595
    if comp>0 then lprinton -1:goto 62575
    if extcall=0 then lprinton else lprinton -1
62575     onerror -1
    portout -41,1
    gosub 62900
    portout -41,0
    printloc xloc,yloc:yloc=yloc-yinc
    print "QuantBasic Report"
    printloc xloc,yloc:yloc=yloc-yinc
    print ""
    printloc xloc,yloc:yloc=yloc-yinc
    print "Quant Type: ";$temp,$temp3
    printloc xloc,yloc:yloc=yloc-yinc
    print "Method: ";$Methname
    printloc xloc,yloc:yloc=yloc-yinc
    print "Sample: ";$temp2
    printloc xloc,yloc:yloc=yloc-yinc
    print "Analyst: ";$name
    printloc xloc,yloc:yloc=yloc-yinc
    print "Sample ID: ";$temp5
    printloc xloc,yloc:yloc=yloc-yinc
    print ""
    printloc xloc,yloc:yloc=yloc-yinc
    if caltyp <> 2 then print "Component","Concentrations"
    if caltyp=2 then print "Component","Concen","R-Squared"
    for i=0 to m-1
      printloc xloc,yloc:yloc=yloc-yinc
      if caltyp<> 2 then print $cname(i),yp(i)
      if caltyp=2 then print $cname(i),yp(i),rsquaredp(i)
    next i
  lprintoff
62590   next comp
  comp=0:sc=0
  onpaint 62900
  onerror -1
  return

62595   onerror -1
  comp=m-1       'clear loop on error
  goto 62590

'+++++++Save Report
62600   dialogoff -10
  string directory, GETDIR
  string _path, SETDIR

  string $WHC_name = $_name + "IR.RPT"
  strname 3,$temp4 = $temp2
```

```
  string $WHC_run = $_name1

  gosub 62480  'fill in header

  onerror 62607
  open "O",#1,$WHC_name
  close #1
  onerror -1

62606   open #1,$WHC_name
  seek #1,-0
  WHC_time = temp3(10,17)
  if WHC_time(4) = 32 then string WHC_time = " " + $WHC_time
  portout -30, 12
  print #1,$WHC_time," ";$WHC_run," ";
  for i=0 to m-1
    print #1,yp(i)," ";
  next i
  print #1," "
  close #1
  portout -30, 16
  goto 62610

62607   onerror -1
  portout -30, 12
  open "C",#1, $WHC_name
  print #1,"TIME"," ";"RUN ID"," ";
  for i=0 to m-1
    WHC_temp = cname(i)
    print #1,$WHC_temp," ";
  next i
  print #1," "
  close #1
  portout -30, 16
  goto 62606

62610   string directory, SETDIR
  return

'+++++++Exit through here
62700   onesc -1
  onpaint -1
  onerror 62710
  if showplot=1 then setfile newslot:noshow
  dialogoff -10
  viewtrace 1:autoscale
  onerror -1
62710   if ExtCall=0 then end else free:return

'+++++++onesc global error handler
62800   dialogoff -10:dialogoff -20     'just in case
  if v=-27 then goto 62700
  if ExtCall=1 then goto 62700
  onesc -1
  dialogon "Error!"
  print "An unrecoverable error has occurred"
  print "Error Number: ";v
  print ""
  print "Click OK to restart QuantBasic or Quit to end"
  print ""
  dialogask mc,4,62850,62700,"OK Quit"
  dialogoff
62850   if showplot=1 then setfile newslot:noshow:showplot=0
```

```
  goto 1

' Repaint
62900   see
  if reportoption<>0 and printoption=0 then return
  if caltyp<>2 then goto 62914
  viewtrace 1
  gosub 57600
  viewtrace 2
  setfile oldslot
  setleft getffp0,getflp0
  if getffp0<getflp0 and getxtype0=1 then xflip
  gosub 43000
  setfile newslot
  return
62914   viewtrace 1
  gosub 43000
  return

' Handle multifile load
63000   dialogon "Multi-file detected"
  print "The selected spectrum is a multi-file and contains"
  print "multiple spectra.  Choose the sub-file number to analyze."
  print ""
  print "Sub-file range: 1 - ";nsubs(#s)
  print ""
  mc=1
  dialogask mc,1,1,nsubs(#s),"Sub-file to analyze"
  print ""
  dialogoff
  string $temp=$temp+" #"+mc
  loadspc #mc
  return

'*********************************
'QUANTBASIC.AB internal documentation
'*********************************
65000   'Code beyond here ignored by interpreter

MRU History
-----------
3/15/94 V1.00G  Intitial release
06/08/94 V1.01G <DK>fixed problem when manually defining regions values for baseline
    did not get calculated as values in um were not geting intialized
    Fixed problem when saving method got a no for when encountered a next
    for caltyp<2 when encountering error
```

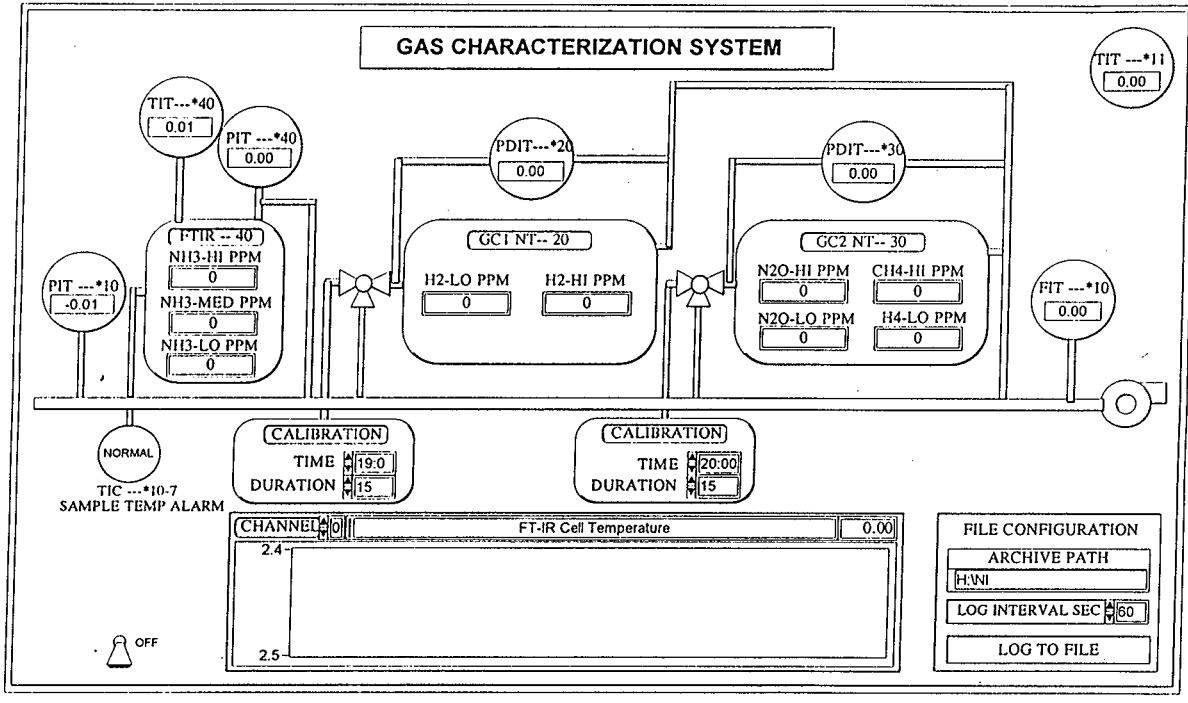
H.3 SPFTIRII.AB

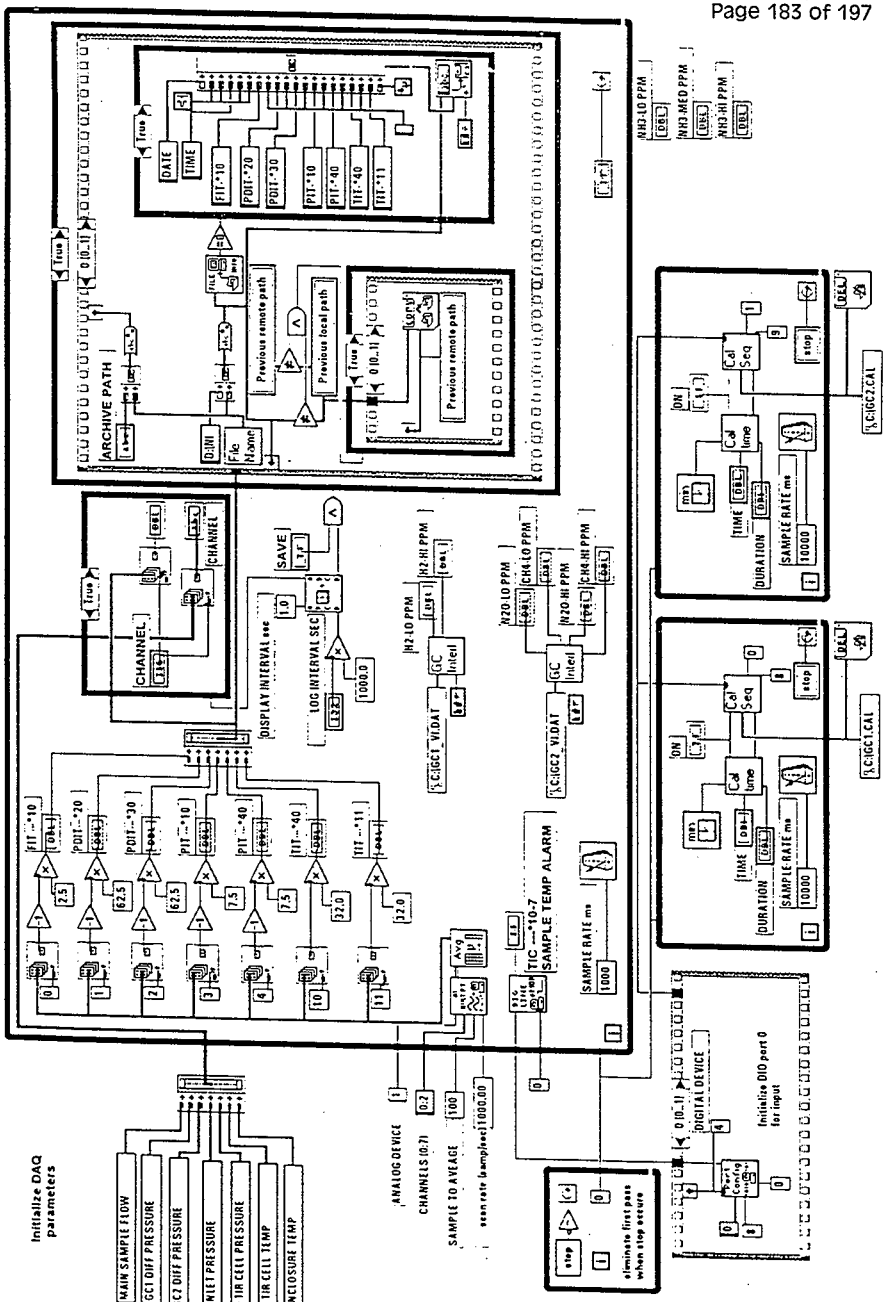
```
'Author: Cuong Vo
'Simple continuous autoscan program w/ quantbas analyses
'note quantbas.ab has been modified for continouse operation
'and renamed to juanbas.ab.
'Saves .SPC trace files and *.RPT files to date name coded under
'FTIR directory.
'8/28/95 User now inputs # of igrams to save in the last line of
'param.inp user file.
'8/30/95 Network awareness, saves trace files and report file locally
'(i.e. C:\FTIR) when network drive is not available.
'11/3/95 Remove drive letter dialog promp. Remote drive letter
'is now assigned to D:

  free *
  free _path,_name,drive,_name1
  dim FTIR_name(72), chk_rpt(60), _path(60), _name(5), _name1(3), id (8)
  dim command1(40), command2(40), command3(80), fname(60), save_igram(2)
  dim conpath(40), cfile(12), bfile(8), metfile(10), drive(2)
  dim local_path(80), host_path(80), command4(100), command5(100)
rem  open the configuration file
  open #1,"c:\win_ir\conf\param.inp"
  input #1," ",$conpath
  input #1," ",$cfile
  input #1," ",$bfile
  input #1," ",$metfile
  input #1," ",$save_igram
  close #1

  del_igram = save_igram(0) - 48
  CNT = 1

  string fname = $conpath + $cfile
  string command3 = "cparms " + $fname
  driver command3
  command3 = 0

  string command3 = "cparms back=" + "C:\ftest\back\" + $bfile
  driver command3
  command3 = 0

rem  The next five lines were rem out to remove dialog box.
rem  The dialog box is used to promp for DRIVE letter.
'   onerror 5
'   dialogon "ENTER DRIVE LETTER TO SAVE DATA"
'   dialogask $drive,2,2,2,"DRIVE TO SAVE AS "
'   dialogoff
'  5  onerror -1
rem  The privious five lines replace with the next line.
  string drive = "D"

rem  Create X:\FTIR dir at host X's drive (X = D)
  string command1 = "command.com /c md "+ $drive + ":\FTIR"
  dos $command1
  see
  command1 = 0

rem  8/30/95
rem  Create C:\FTIR dir at local drive
  string command1 = "command.com /c md C:\FTIR"
```

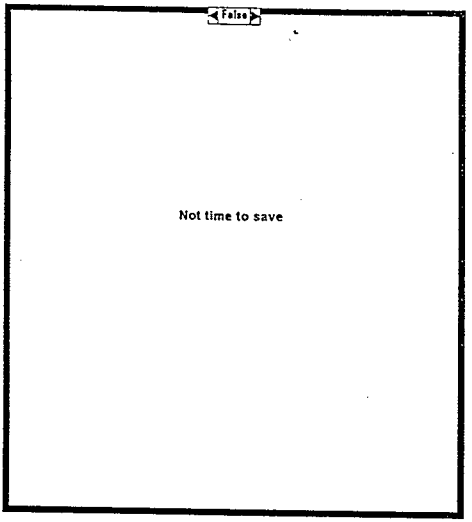
```
  dos $command1
  see
  command1 = 0
rem  end

10  _name=0
  run = 0
  YEAR = clock(-7)
  YEAR = YEAR - 1990
  _name(0) = YEAR + 48
  MONTH = clock(-6)
  month2 = int(MONTH/10)
  month1 = MONTH - (month2*10)
  _name(1) = month2 + 48
  _name(2) = month1 + 48
  DAY = clock(-5)
  day2 = int(DAY/10)
  day1 = DAY - (day2*10)
  _name(3) = day2 + 48
  _name(4) = day1 + 48
  PREDAY = DAY

rem  create today's dir at host drive
  string command1 = "command.com /c md "+ $drive + ":\FTIR\"+ $_name + "IR"
  dos $command1
  see
  command1 = 0

rem  check to see where today.rpt file exist
  string _path= "C:\FTIR\"
  string    chk_rpt = $_path + $_name + "IR.RPT"
  onerror 100
  open "O", #1, $chk_rpt
  close #1
  onerror -1

rem  if today.rpt file exist then acquire run number from RUN.ID file
  string command1 = "C:\FTIR\RUN.ID"
  open #1, $command1
  input #1,run
  close #1
  command1 = 0

100  onerror -1
  for w = 1 to 1000
  run = run + 1
  numb1=int(run/100)
  numb2=int((run-100*numb1)/10)
  numb3=run-numb1*100-numb2*10
  _name1(0)= numb1 + 48
  _name1(1)= numb2 + 48
  _name1(2)= numb3 + 48

  DAY = clock(-5)
  if not(DAY = PREDAY) then 120 else 140

120  dos "net time \\GCS1HOST /set /yes"

rem  network connection OK?
  string command4 = 0
  string command4 = $drive +":\FTIR\" + $_name + "IR\" + $_name + "IR.RPT"
  onerror 130
  open "O", #1, $command4
```

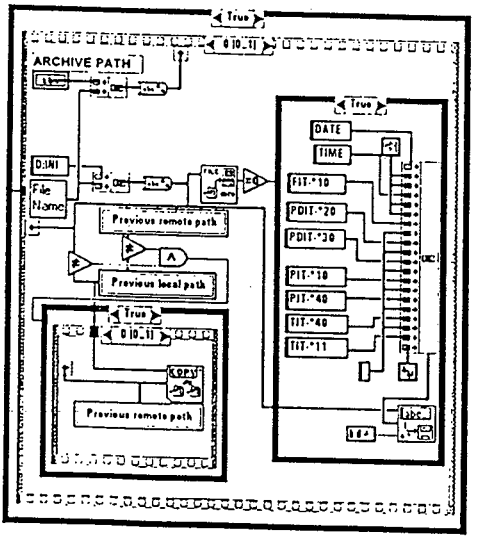
```
  close #1
  onerror -1
  string command5 = 0
  string command5 = "C:\FTIR\" + $_name + "IR.RPT"
  kill $command5
  CNT = 1

130  w=1001
  goto 200

140  string FTIR_name = $_path + $_name + $_name1
  command2 = 0
  string command2 = "scan " + $FTIR_name
  driver $command2

  chain "ICOMPUT"

  onerror 150
  string command1 ="C:\FTIR\RUN.ID"
  kill $command1
150  onerror -1
  open "C",#1, $command1
  print #1,run
  close #1
  string command3 = $conpath + $metfile
  chain "juantbas",60000,($command3,2,1,0,0)
  command3 = 0
  onerror -1

rem  construct host and local based path
  string host_path = $drive + ":\FTIR\" + $_name + "IR"
  string local_path = "C:\FTIR\" + $_name

  if (CNT = del_igram) then 160 else 170
160  CNT = 1
  command4 = 0
  command5 = 0
  string command4 ="command.com /c move /Y " + $FTIR_name + ".spc " + $host_path
  dos $command4
  onerror -1
  see
  goto 190

170  string command3 = $FTIR_name + ".spc"
  kill $command3
  CNT = CNT +1
190  command4 = 0
  command5 = 0
  string command4 ="command.com /c copy " + $local_path + "IR.RPT "
  string command5 = $command4 + $host_path + "\" + $_name + "IR.RPT"
  dos $command5
  see
  command3 = 0
  FTIR_name = 0
  _name1 = 0
  noshow
200  next w

  kill $command1
  command1 = 0
  goto 10
```

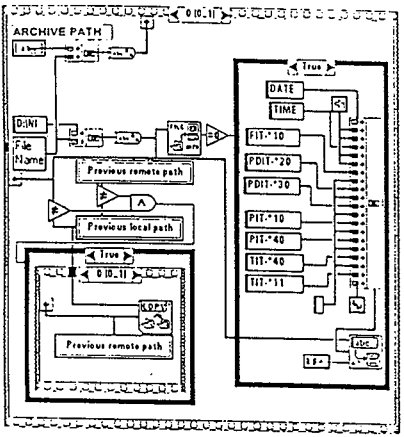
APPENDIX I -- LABVIEW SCREEN AND BLOCK DIAGRAM LISTINGS

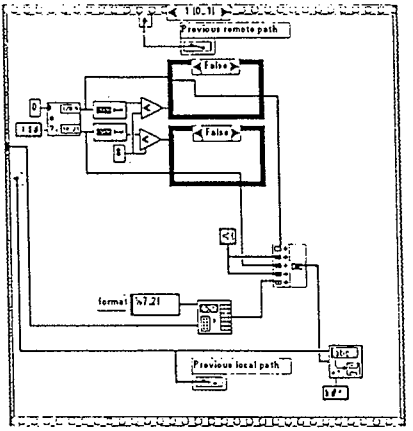

GAS CHARACTERIZATION SYSTEM
TIT ---*11
0.00
TIT---*40
0.01
PIT ---*40
0.00
PDIT---*20
0.00
PDIT---*30
0.00
FTIR -- 40
NH3-HI PPM
0
NH3-MED PPM
0
NH3-LO PPM
0
GC1 NT-- 20
H2-LO PPM
0
H2-HI PPM
0
GC2 NT-- 30
N2O-HI PPM
0
CH4-HI PPM
0
N2O-LO PPM
0
H4-LO PPM
0
PIT ---*10
-0.01
FIT ---*10
0.00
NORMAL
TIC ---*10-7
SAMPLE TEMP ALARM
CALIBRATION
TIME
19:0
DURATION
15
CALIBRATION
TIME
20:00
DURATION
15
CHANNEL
0
FT-IR Cell Temperature
0.00
2.4
2.5
OFF
FILE CONFIGURATION
ARCHIVE PATH
H:\NI
LOG INTERVAL SEC
60
LOG TO FILE

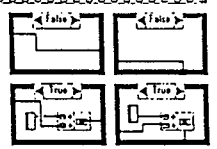

Initialize DAQ parameters
MAIN SAMPLE FLOW
GC1 DIFF PRESSURE
GC2 DIFF PRESSURE
INLET PRESSURE
FTIR CELL PRESSURE
FTIR CELL TEMP
ENCLOSURE TEMP
ANALOG DEVICE
CHANNELS (0:7)
SAMPLE TO AVERAGE
scan rate (sampleset)
1000.00
DIGITAL DEVICE
Initialize DIO port 0 for input
eliminate first pass when stop secure
SAMPLE RATE ms
1000
SAMPLE TEMP ALARM
TIC_*10_7
GC Intel
%CIGC1_VI.DAT
%CIGC2_VI.DAT
N2-LO PPM
N2-HI PPM
N2O-LO PPM
N2O-HI PPM
CH4-LO PPM
CH4-HI PPM
NH3-LO PPM
NH3-MED PPM
NH3-HI PPM
DISPLAY INTERVAL sec
LOG INTERVAL SEC
1000.0
SAVE
CHANNEL
ARCHIVE PATH
Previous remote path
Previous local path
DATE
TIME
FIT_*10
PDIT_*20
PDIT_*30
PIT_*10
PIT_*40
TIT_*11
Cal Seq
Cal time
DURATION
SAMPLE RATE ms
10000
%CIGC1.CAL
%CIGC2.CAL


False
Not time to save

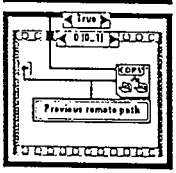

True
0 10_11
ARCHIVE PATH
D:\NI
File Name
Previous remote path
Previous local path
True
0 10_11
COPY
Previous remote path
True
DATE
TIME
FIT-*10
PDIT-*20
PDIT-*30
PIT-*10
PIT-*40
TIT-*40
TIT-*11

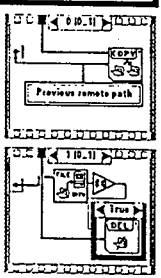

ARCHIVE PATH
D:\N1
File Name
Previous remote path
Previous local path
True
DATE
TIME
FIT-*10
PDIT-*20
PDIT-*30
PIT-*10
PIT-*40
TIT-*40
TIT-*11
Previous remote path

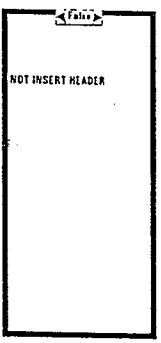

Previous remote path
False
False
format
%7.2f
Previous local path

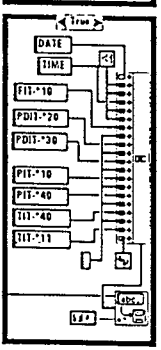

False
False
True
True

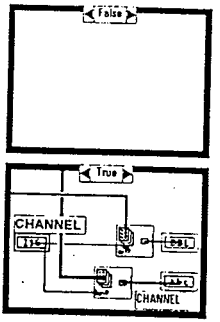
False
DO NOT THING

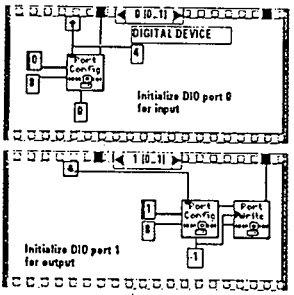

True
Previous remote path


Previous remote path
True
False
True

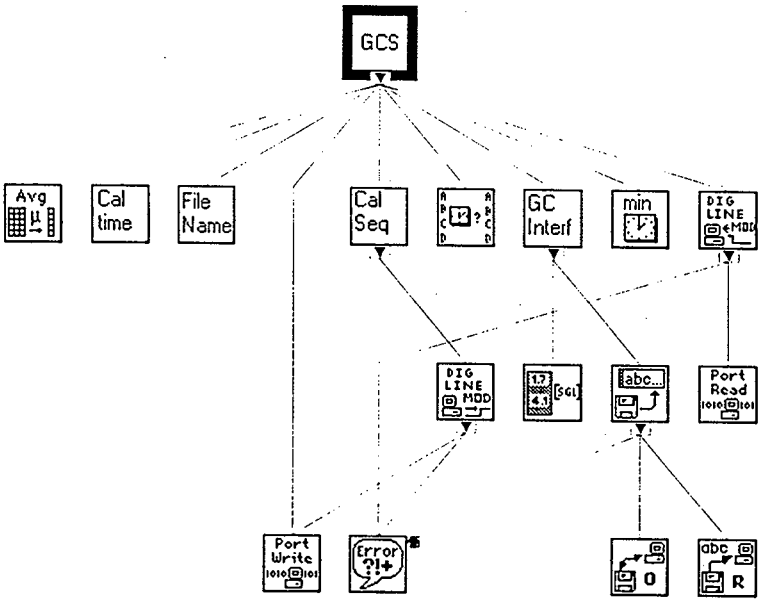

False
NOT INSERT HEADER


True
DATE
TIME
FIT-*10
PDIT-*20
PDIT-*30
PIT-*10
PIT-*40
TIT-*40
TIT-*11

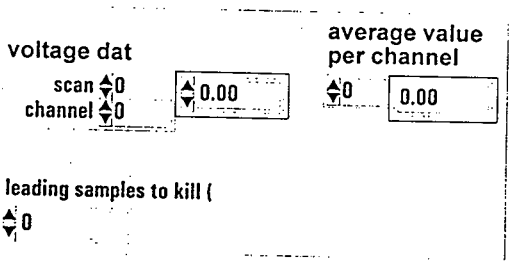

False
True
CHANNEL
CHANNEL


DIGITAL DEVICE
Port Config
Initialize DIO port 0
for input
Port Config
Port Write
Initialize DIO port 1
for output

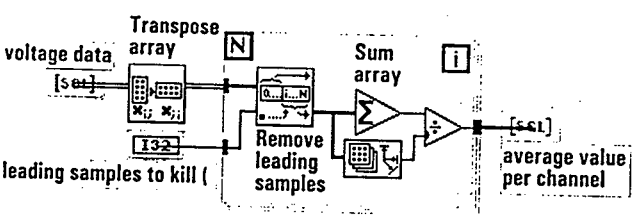
APPENDIX J.1 VI STRUCTURE



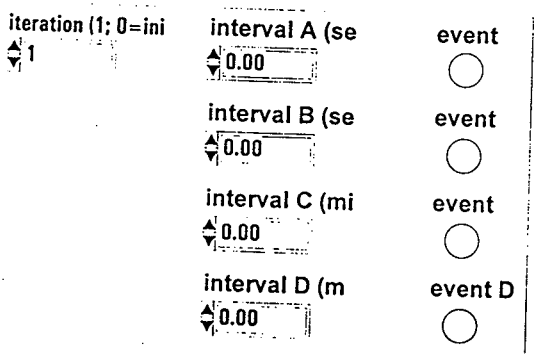
Average Voltage.vi
Last modified on 8/19/96 at 10:11 AM

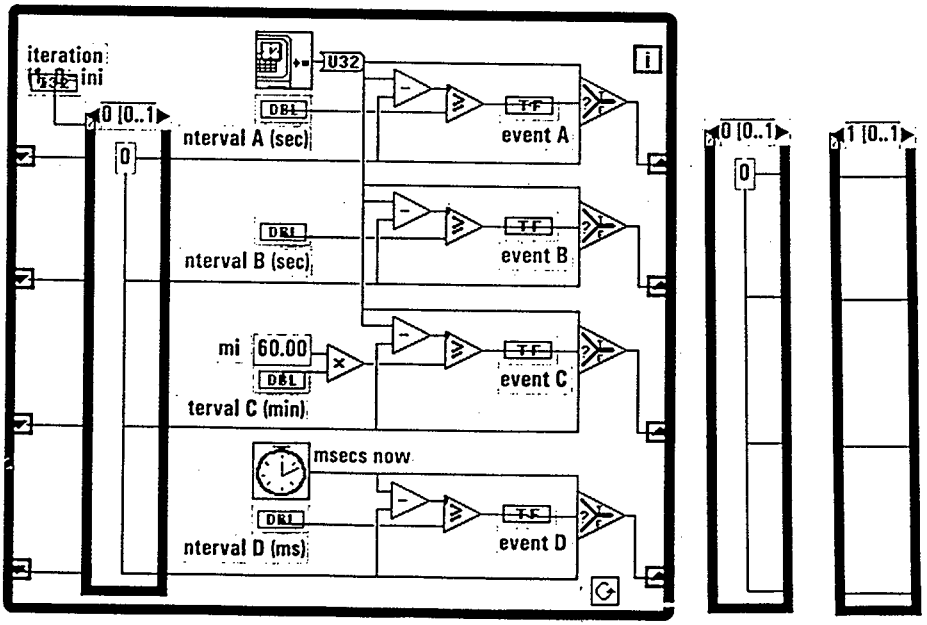


Average Voltage.vi
Last modified on 8/19/96 at 10:11 AM

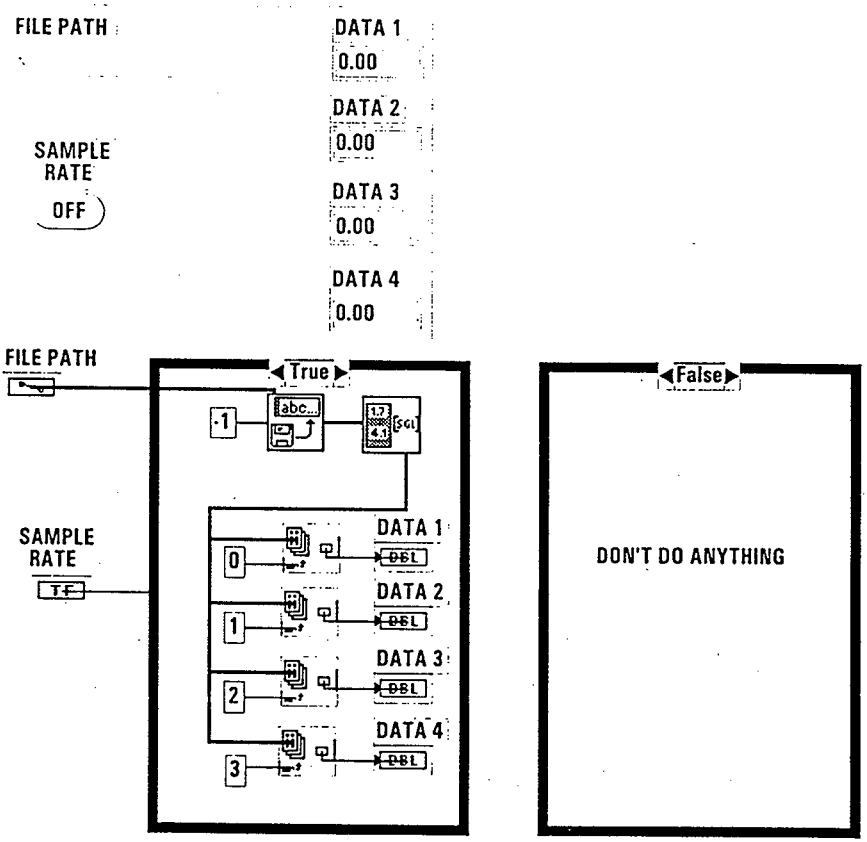


Interval Timer.vi
Last modified on 8/19/96 at 10:11 AM





GC INTERFACE.vi
Last modified on 1/1/97 at 6:05 AM



Cal
time

CAL COMPARE TIME.VI
Last modified on 11/15/96 at 11:49 AM

Convert a predefined time period to boolean o

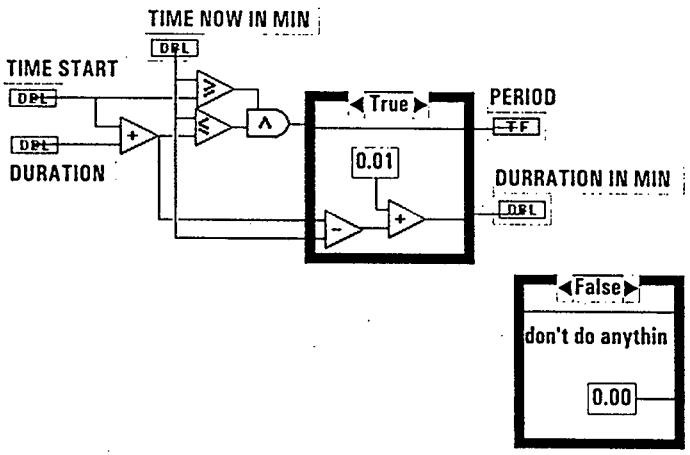
TIME NOW IN MIN
0.00

TIME START
0.00

DURATION
0.00

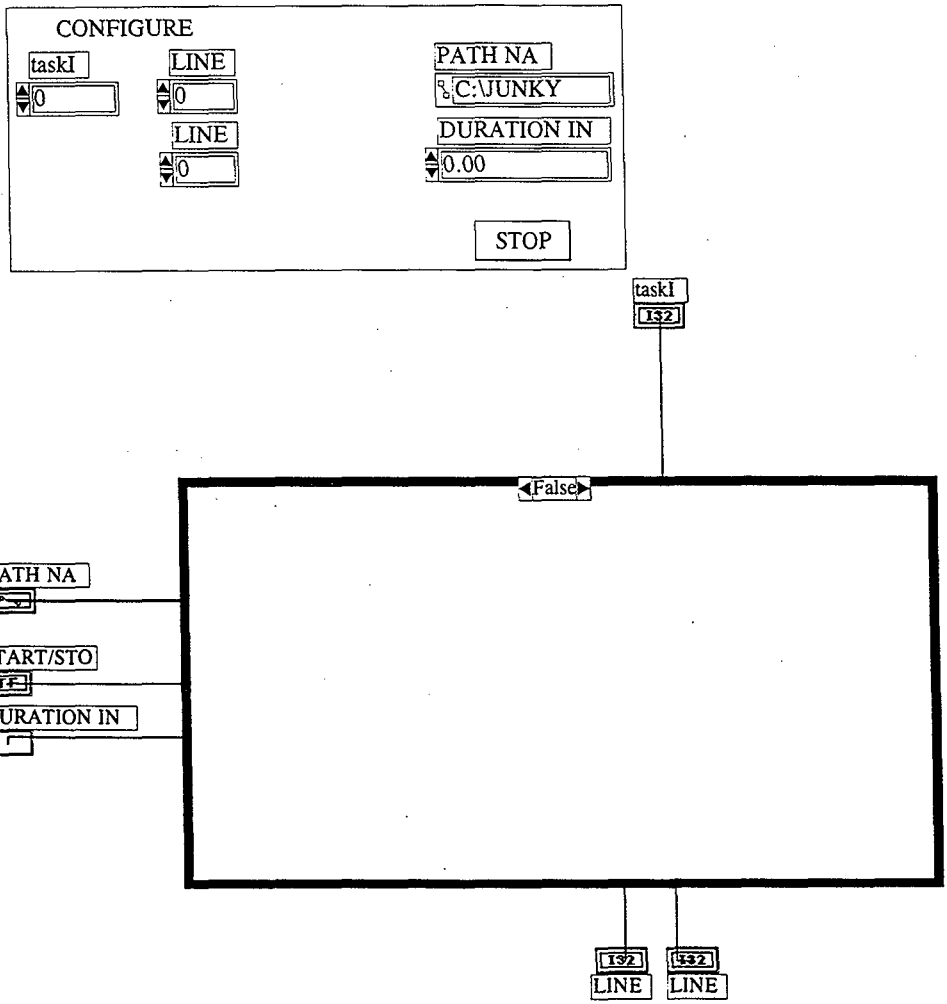
PERIOD
OFF

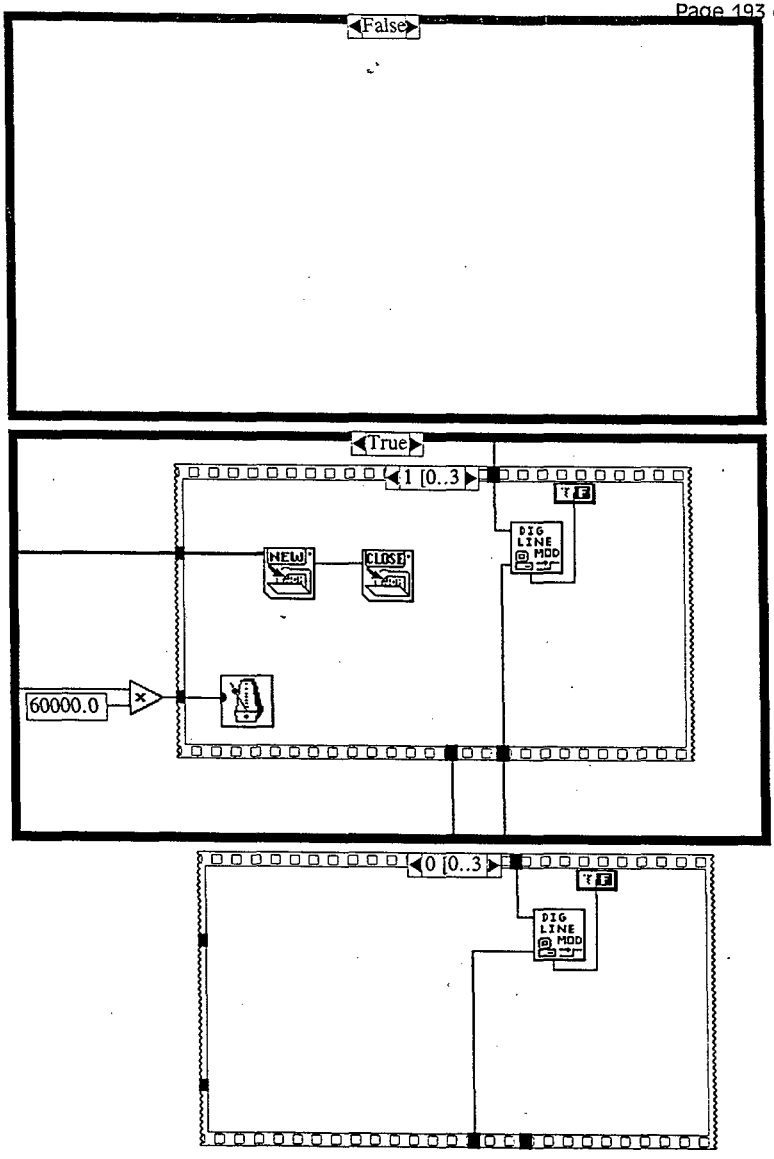
DURRATION IN MIN
0.00

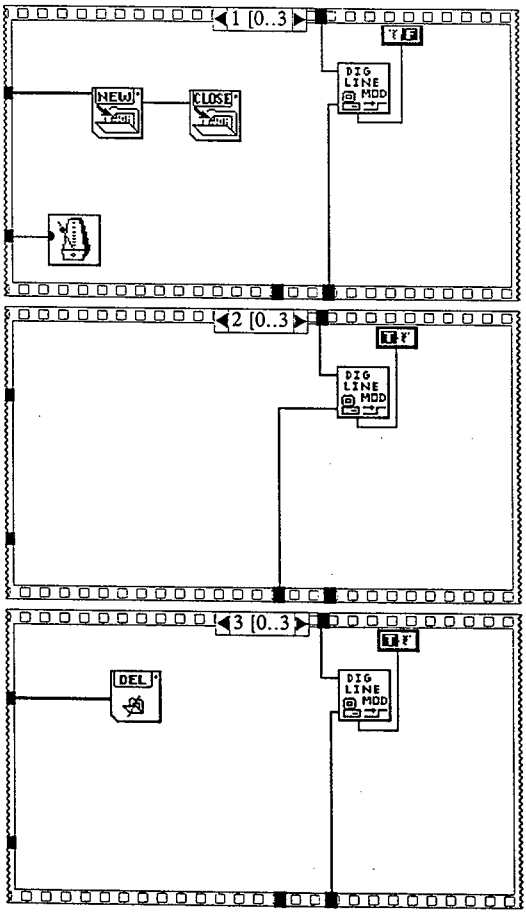


Cal
Seq

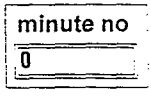
CAL SEQUENCE 1.VI
Last modified on 8/19/96 at 10:11 AM



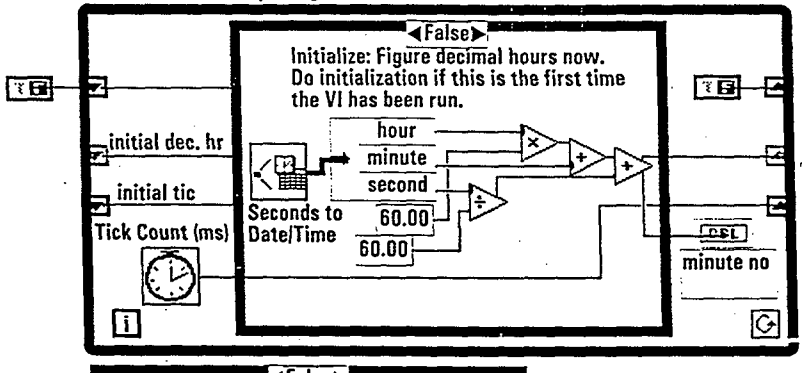

False
True
1 [0..3]
NEW
CLOSE
DIG LINE MOD
60000.0
0 [0..3]
DIG LINE MOD

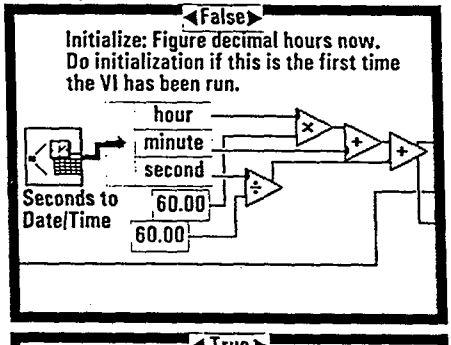

1 [0..3]
NEW
CLOSE
DIG LINE MOD
2 [0..3]
DIG LINE MOD
3 [0..3]
DEL
DIG LINE MOD

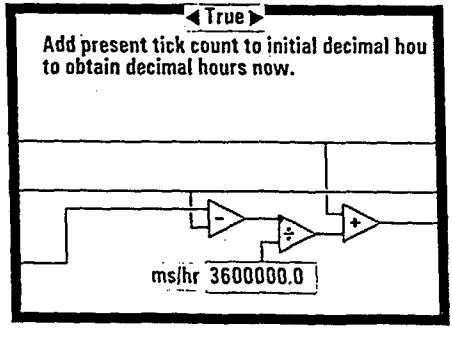
Minutes now.vi
Last modified on 8/19/96 at 10:11 AM



While loop is a global variable to save initial hours & tick co

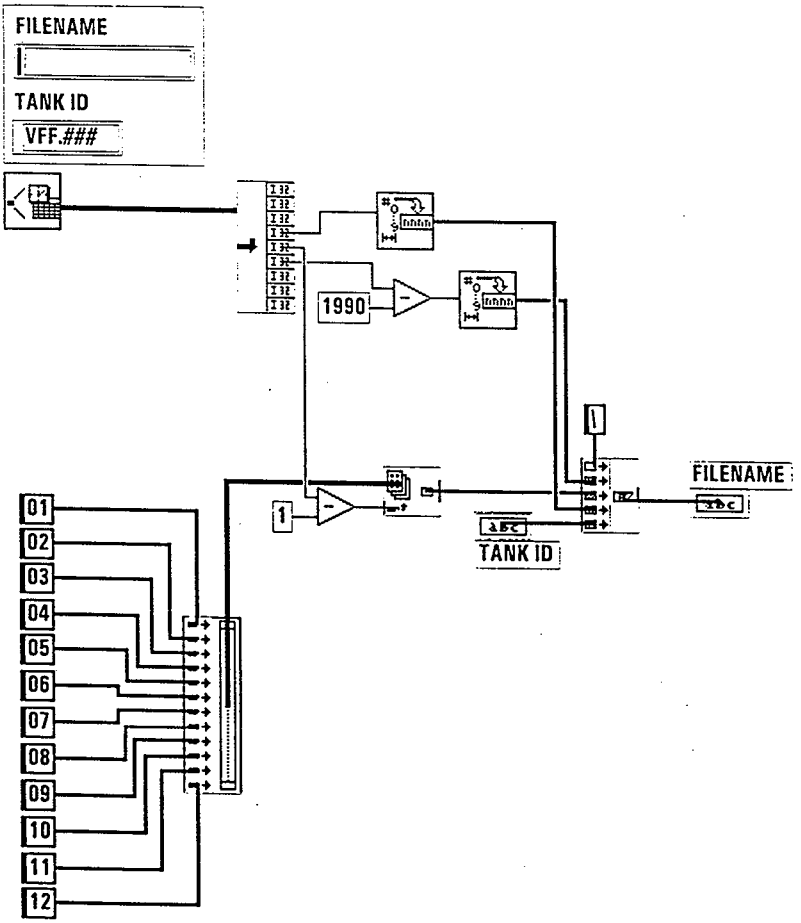




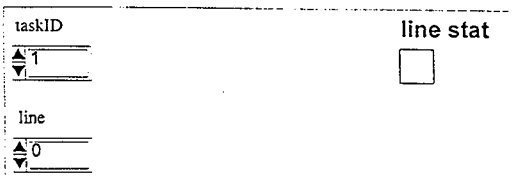


File Name

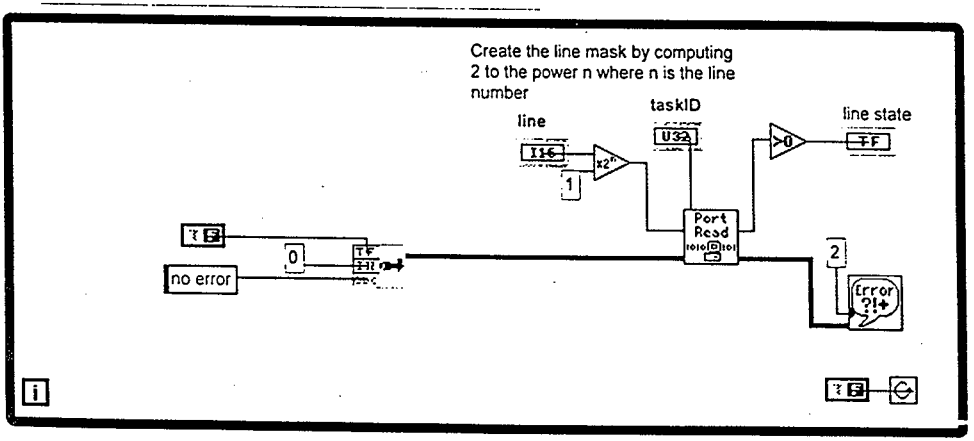
filename modify.vi
Last modified on 12/19/96 at 3:36 PM



Read from Digital Line modify.vi
Last modified on 1/1/97 at 6:05 AM



See Get Info for more informatio



DISTRIBUTION SHEET

To	From	Page 1 of 1
DISTRIBUTION	B.L. Philipp SESC	Date 5/29/97
Project Title/Work Order		EDT No. 619509
Computer Systems and Software Description for Gas Characterization System		ECN No.

Name	MSIN	Text With All Attach.	Text Only	Attach. / Appendix Only	EDT/ECN Only
T.C. Schneider	L6-37	X			
D.D. Tate	L6-37	X			
M.F. Erhart	R1-51	X			
R.L. Schlosser	R1-56				X
L.D. Salsberry	B7-41	X			
S.U. Zaman	R3-08	X			
R.E. Bauer	S7-14		(Pgs. 1-35) +	A&B +	X
G.D. Johnson	S7-14		(Pgs. 1-35) +	A&B +	X
B.L. Philipp (3)	L6-37	X			
Central Files	A3-88	X			