RMIS View/Print Document Cover Sheet

This document was retrieved from the Boeing ISEARCH System.

Accession #: D196071694

Document #: SD-WM-CSWD-076

Title/Desc:

DESIGN LAYOUT FOR GAS MONITORING SYSTEM II
COMPUTER SYSTEM

<i>Oct. 3, 1995</i> <i>Station 35</i>					ENGINEERING DATA TRANSMITTAL					Page 1 of <u>1</u> 1. EDT 607633					
2. To: (Receiving Organization) Characterization Monitoring Systems				3. From: (Originating Organization) Control Systems Engineering				4. Related EDT No.: N/A							
5. Proj./Prog./Dept./Div.: Waste Management				6. Cog. Engr.: B. L. Philipp				7. Purchase Order No.: N/A							
8. Originator Remarks: This EDT is for the release of WHC-SD-WM-CSWD-076, Rev. 0.								9. Equip./Component No.: N/A							
11. Receiver Remarks:								10. System/Bldg./Facility: N/A							
								12. Major Assm. Dwg. No.: N/A							
								13. Permit/Permit Application No.: N/A							
14. Required Response Date: N/A								14. Required Response Date: N/A							
								N/A							
15. DATA TRANSMITTED										(F)	(G)	(H)	(I)		
(A) Item No.	(B) Document/Drawing No.	(C) Sheet No.	(D) Rev. No.	(E) Title or Description of Data Transmitted				Approval Designator	Reason for Transmittal	Originator Disposition	Receiver Disposition				
1	WHC-SD-WM-CSWD-076		0	Design Layout for Gas Monitoring System II (GMS-2) Computer Systems				S,Q	1,2	1	1				
16. KEY										Approval Designator (F) E, S, Q, D or N/A (see WHC-CM-3-5, Sec. 12.7)		Reason for Transmittal (G) 1. Approval 2. Release 3. Information 4. Review 5. Post-Review 6. Dist. (Receipt Acknow. Required)		Disposition (H) & (I) 1. Approved 2. Approved w/comment 3. Disapproved w/comment 4. Reviewed no/comment 5. Reviewed w/comment 6. Receipt acknowledged	
(G)	(H)	17. SIGNATURE/DISTRIBUTION (See Approval Designator for required signatures)								(G)	(H)				
Reason	Disp.	(J) Name	(K) Signature	(L) Date	(M) MSIN	(J) Name	(K) Signature	(L) Date	(M) MSIN	Reason	Disp.				
1,2	1	Cog. Eng. EK Straalsund	<i>[Signature]</i>	9/14/95	E6-37										
1,2	1	Cog. Mgr. RE Bauer	<i>[Signature]</i>	9/22/95	E6-37										
1,2	1	QA M. L. McElroy	<i>[Signature]</i>	8-31-95	E6-37										
1,2	1	Safety L. S. Krogsrud	<i>[Signature]</i>	8/11/95	SR3-08										
		Env.													
1,2		CV Vo	<i>[Signature]</i>	9/22/95	E6-37	Karon Tate	<i>[Signature]</i>	9/22/95		1,2					
1,2		BL Philipp	<i>[Signature]</i>	8/2/95	E6-21										
18. BL Philipp <i>[Signature]</i> Signature of EDT Originator		19. <i>[Signature]</i> Authorized Representative for Receiving Organization		20. RE Bauer <i>[Signature]</i> Cognizant Manager		21. DOE APPROVAL (if required) Ctrl. No. ☐ Approved ☐ Approved w/comments ☐ Disapproved w/comments									

THIS PAGE INTENTIONALLY
LEFT BLANK

RELEASE AUTHORIZATION

Document Number: WHC-SD-WM-CSWD-076, Rev. 0

Document Title: Design Layout for Gas Monitoring System II (GMS-2)
Computer System.

Release Date: 10/3/95

**This document was reviewed following the
procedures described in WHC-CM-3-4 and is:**

APPROVED FOR PUBLIC RELEASE

WHC Information Release Administration Specialist:

Chris Willingham
C. Willingham

10/3/95

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.

This report has been reproduced from the best available copy. Available in paper copy. Printed in the United States of America. To obtain copies of this report, contact:

Westinghouse Hanford Company - Document Control Services
P.O. Box 1970, Mailstop H6-08, Richland, WA 99352
Telephone: (509) 372-2420; Fax: (509) 376-4989

LABNET and CHROMNET are Trademarks of Thermal Separation Product, California

THIS PAGE INTENTIONALLY
LEFT BLANK

SUPPORTING DOCUMENT

1. Total Pages ¹⁴⁸ 118 *ew*

2. Title

Design Layout for Gas Monitoring System II (GMS-2)
Computer System

3. Number

WHC-SD-WM-CSWD-076

4. Rev No.

0

5. Key Words

Design Layout, Gas Monitoring System II, GMS-2,
Computer System

6. Author

Name: B. L. Philipp

Barbara L. Philipp 8/2/95
Signature

Organization/Charge Code 57C00/N2B2F

7. Abstract

This document provides a general overview of the computer systems software that perform the data acquisition and control for the 241-SY-101 Gas Monitoring System II (GMS-2). It outlines the system layout, and contains descriptions of components and the functions they perform. The GMS-2 system was designed and implemented by Los Alamos National Laboratory and supplied to Westinghouse Hanford Company.

8. RELEASE STAMP

OFFICIAL RELEASE 22
EX-114

DATE Oct. 3, 1995

Station 35

THIS PAGE INTENTIONALLY
LEFT BLANK

**DESIGN LAYOUT
FOR
GAS MONITORING SYSTEM II (GMS-2)
COMPUTER SYSTEMS**

August 2, 1995

C. V. Vo

B. L. Philipp

M. P. Manke

Westinghouse Hanford Company
P.O. Box 1970
Richland, Washington 99352

THIS PAGE INTENTIONALLY
LEFT BLANK

CONTENTS

1.0	DESIGN STRATEGY	1
1.1	BACKGROUND	1
1.1.1	Site Environment	2
2.0	DESIGN DESCRIPTION	2
2.1	COMPONENT LAYOUT	2
2.1.1	FTIR Spectrometer System	4
2.1.2	RGA3 Gas Chromatograph System	4
2.1.3	Ammonia Monitoring System	4
2.1.4	GMS-2 Host Computer	4
2.1.5	RGA5 Dual Reduction Gas Detector H2 Host System	5
2.1.6	BlackBox - GC Inlet Gas line Valve Control System	5
2.2	PRINCIPAL OF OPERATION	6
2.2.1	SPFTIR - FTIR Spectrometer System Control	6
2.2.2	FTRANS - FTIR Spectrometer System File Transfer	6
2.2.3	FTAN-B - Infrared Data Analysis Codes	6
2.2.4	PHOTO - Ammonia Monitoring System	6
2.2.5	DACS - GMS-2 Host Computer	7
2.2.6	GC3 - RGA3 Reduction Gas Detector H2 Host System	7
2.1.7	BlackBox System	7
3.0	FUNCTION AND DESIGN REQUIREMENTS	8
3.1	FUNCTIONAL FLOW	8
3.1.1	SPFTIR - FTIR Spectrometer System Control	8
3.1.2	FTRANS - FTIR Spectrometer System File Transfer	8
3.1.3	FTAN-B - Infrared Data Analysis Codes	9
3.1.4	PHOTO - Ammonia Monitoring System	10
3.1.5	DACS - GMS-2 Host Computer	11
3.1.6	GC3 - RGA3 Reduction Gas Detector H2 Host System	12
3.2	SYSTEM INTERFACES	13
3.2.1	SPFTIR - FTIR Spectrometer System Control	13
3.2.1.1	Input Requirements	13
3.2.1.2	Output Requirements	13
3.2.2	FTRANS - FTIR Spectrometer System File Transfer	15
3.2.2.1	Input Requirements	15
3.2.2.2	Output Requirements	16
3.2.3	FTAN-B - Infrared Data Analysis Codes	17
3.2.3.1	Input Requirements	17
3.2.3.2	Output Requirements	17
3.2.4	PHOTO - Ammonia Monitoring System	18
3.2.4.1	Input Requirements	18
3.2.4.2	Output Requirements	18
3.2.5	DACS - GMS-2 Host Computer	19
3.2.5.1	Input Requirements	19
3.2.5.2	Output Requirements	20
3.2.6	GC3 - Dual Reduction Gas Detector H2 Host System	20
3.2.6.1	Input Requirements	20
3.2.6.2	Output Requirements	21
3.2.7	BlackBox System	22
3.2.7.1	Input Requirements	22
3.2.7.2	Output Requirements	22

CONTENTS (cont)

4.0	OPERATION	23
4.1	OVERVIEW	23
4.2	OPERATING PROCEDURES	23
4.2.1	System Startup, Normal Operation, Shut Down	23
5.0	DISCREPANCY REPORTING AND CORRECTIVE ACTIONS	23
6.0	CONFIGURATION CONTROL	23
6.1	CONFIGURATION IDENTIFICATION	24
6.2	CONFIGURATION CHANGE CONTROL	24
6.3	CONFIGURATION STATUS ACCOUNTING	24
6.4	MEDIA CONTROL	24
7.0	SAFETY, RELIABILITY, AND QUALITY ASSURANCE	25
7.1	SAFETY CLASSIFICATION AND SIGNATURE DESIGNATION	25
7.2	QUALITY ASSURANCE AND QUALITY CONTROL	25
8.0	REFERENCES	26
9.0	BIBLIOGRAPHY	27
9.1	U.S. GOVERNMENT	27
9.2	WESTINGHOUSE HANFORD COMPANY	27
9.3	CODES AND STANDARDS	27

APPENDICES

APPENDIX A	- COMPUTER SYSTEM DESCRIPTIONS	A-1
APPENDIX B	- OPERATION INSTRUCTIONS	B-1
B.1	PURPOSE	B-1
B.2	PREREQUISITES	B-1
B.3	OS/2 OPERATING SYSTEM	B-2
B.4	INITIAL ASSESSMENT AND RESTART FROM DACS	B-3
B.5	COMPLETE GC SYSTEM REBOOT	B-6
B.6	KENETICS COMPUTER RESTART FROM LOCKUP CONDITION	B-12
B.7	FTIR RESTART FROM DACS	B-15

CONTENTS (cont)

APPENDIX C - SYSTEM CODE LISTINGS	C-1
C.1 "C" COMPILER	C-1
C.2 SPFTIR - FTIR Spectrometer System Control	C-1
SOURCE LISTING : SPFTIR Ver. 1.0	C-1
C.3 FTRANS - FTIR Spectrometer System File Transfer	C-4
SOURCE LISTING : FTANS Ver. 1.0	C-4
SOURCE LISTING : DET_FILE Ver. 1.0	C-6
SOURCE LISTING : GET_FILE Ver. 1.0	C-8
SOURCE LISTING : TRANS F Ver. 1.0	C-9
SOURCE LISTING : CLEAN Ver. 1.0	C-10
C.4 FTAN-B - Infrared Data Analysis Codes	C-12
SOURCE LISTING : FTANB Ver. 2.02	C-12
SOURCE LISTING : HEXTOLO Ver. 1.0	C-18
SOURCE LISTING : HEXTODBL Ver. 1.0	C-19
SOURCE LISTING : RN (CONTINUOUS OPERATION CODE)	C-26
SOURCE LISTING : OFFLN	C-27
SOURCE LISTING : BATCHA (OFFLINE ANALYSIS CODE)	C-29
C.5 PHOTO - Ammonia Monitoring system	C-34
SOURCE LISTING : PHOTO Ver. 1.0	C-34
SOURCE LISTING : COMOPEN Ver. 1.0	C-39
SOURCE LISTING : COMREAD Ver. 1.0	C-42
SOURCE LISTING : COMWRITE Ver. 1.0	C-43
SOURCE LISTING : COMFLUSH Ver. 1.0	C-44
C.6 DACS - GMS-2 Host computer	C-45
SOURCE LISTING : DACS Ver. 1.5	C-45
C.7 GC3 - RGA3 Reduction Gas Detector H2 Host System	C-59
SOURCE LISTING : GCF3 Ver. 1.5	C-59
SOURCE LISTING : FIL_NAME Ver. 1.0	C-66
SOURCE LISTING : CRE_NAME Ver. 1.0	C-68
SOURCE LISTING : STRIP Ver. 1.0	C-69
C.8 Black Box - Programmable Logic Controller Code	C-71
APPENDIX D - FTIR CALIBRATION DESCRIPTIONS	D-1

LIST OF FIGURES

2.1 - GMS-2 Computer System Layout Diagram	3
--	---

THIS PAGE INTENTIONALLY
LEFT BLANK

DESIGN LAYOUT FOR GAS MONITORING SYSTEM II (GMS-2) COMPUTER SYSTEMS

1.0 DESIGN STRATEGY

This document provides a general overview of the computer systems that perform the data acquisition and control for the 241-SY-101 Gas Monitoring System II (GMS-2). It outlines the system layout, and contains descriptions of components and the functions they perform. The GMS-2 system was designed and implemented by Los Alamos National Laboratory (LANL) and supplied to Westinghouse Hanford Company (WHC).

The minimum set of documentation required for contract-developed, quality affecting software (WHC-CM-4-2, QR 19.0) is software source code on machine-readable media and the following documentation:

- System requirements specifications
- System design description
- User documentation
- Verification and validation records, including test procedures, test case specifications, and test results.

This document includes the Software Requirements Specification, Software Design Description, User Documentation, and the Software Configuration Management Plan. It does not include the verification and validation records (test specifications, acceptance test procedures, or the validation report).

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. Hydrogen build up in the 241-SY-101 waste tank was identified within the Department of Energy (DOE) complex as an Unreviewed Safety Question. Hydrogen monitoring is currently established on the 241-SY-101 waste tank. Additionally, for personnel safety and environmental monitoring, the amount of ammonia released from the SY tank farm exhaust stack is monitored. Currently, two technologies are employed to measure the presence and amount of Ammonia. The technologies are an on-line Fourier Transform Infrared Spectrometer (FTIR) and a multi-gas Photoacoustic Infrared Spectrometer. With both instruments, the data analysis provides good discrimination between the measured gas (ammonia) and water vapor.

The Gas Monitoring System II (GMS-2) Computer System consists of several individual computer components necessary to address the control and data acquisition from the specific types of instrumentation used to monitor the vapor space on the 241-SY-101 waste tank. These computers are networked, using the IBM OS/2 multi-tasking operating system and networking software. This allows the system to consist of several components that function in parallel to perform instrument control, data collection, analysis, computer interfacing, and archiving functions.

1.1.1 Site Environment

The environment to which the Gas Monitoring System GMS-2 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 shelter 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 is less than 0.5 mR/hr.

2.0 DESIGN DESCRIPTION

2.1 COMPONENT LAYOUT

The general organization of the GMS-2 computer system components are shown in the GMS-2 Computer System Layout Diagram (Figure 2.1). The initial tier of the GMS-2 computer system consists of several individual computer components necessary to address the control and data acquisition from specific types of instrumentation. These computers are networked, using the IBM OS/2 multi-tasking operating system and networking software.

- The Fourier Transform Infrared Spectrometer system (FTIR) component is used to control the analytical instrument and transfer the raw data necessary to calculate nitrous oxide and ammonia concentration.
- The RGA3 Gas Chromatograph with reduction gas detector component records accurate measurements of trace of hydrogen (0-300ppm).
- The RGA5 dual reduction gas detector gas chromatograph system component is also used to measure and record accurate measurements of trace levels of hydrogen up to 1%.
- The Bruel&Kjaer Multi-gas Monitor (B&K) component is also used to measure ammonia.

Each component system was chosen based on its abilities to perform a specific task. Information is exchanged between components to integrate all of the data and control signals into a complete system. These components are networked together with the GMS-2 computer via Ethernet. The GMS-2 computer analyzes, archives, and sends data that were generated from the instruments to the SY-farm DACS's computer.

The networked instrument/computer system is composed of the GMS-2 main host computer, the FTIR Spectrometer system, the RGA3 Gas Chromatograph system, and the B&K Ammonia Monitoring system. The GMS-2 main host computer communicates the calculated gas percentage data to the 241-SY-101 Data Acquisition Computer System's (DAC) ASCII module for display. The RGA5 system host communicates directly with the DAC's system using the ASCII module.

Figure 2.1 - GMS-2 Computer System Layout Diagram

2.1.1 FTIR Spectrometer System

The FTIR spectrometer system is used to measure nitrous oxide and ammonia concentration. It consists of a BioRad FTIR System Computer: SPC 3200, and a FTIR Spectrometer Optics Bench: BIO-RAD FTS-25. The SPC 3200 computer is a pseudo-UNIX based system that uses the BioRad IDRIS operating system. It has networking capability and is controlled remotely from the GMS-2 host computer via an RS-232 serial communication link. The custom script SPFTIR, developed by LANL, runs on the FTIR SPC 3200 computer. Its main function is to operate the infrared scanning optics bench, and create a unique raw data file for each scan performed. Each raw data file created is subsequently transferred to the GMS-2 host computer where it is evaluated to obtain final results.

2.1.2 RGA3 Gas Chromatograph System

Gas Chromatography (GC) technology is used to measure parts-per-million trace concentrations of hydrogen. The GC system consists of the GC computer, the Chromjet Integrator (Spectra Physics SP4400), and a single column Reduction Gas Chromatograph (Trace Analytical RGA3). The GC computer has networking capability using IBM OS/2 multi-tasking operating system to perform parallel functions: control of the integrator/RGA3 and interface with the system host.

Two commercial software packages that run in DOS windows from OS/2, LABNET and CHROMNET, are supplied by Spectra Physics to run the GC computer. LABNET and the associated hardware provides the communication interface between CHROMNET and the Chromjet Integrator. CHROMNET runs the Chromjet integrator and, indirectly, the Reduction Gas Chromatograph. The chromatogram (RAW data file) created at the GC computer is then available to transfer for the GMS-2 computer.

2.1.3 Ammonia Monitoring System

A Bruel&Kjaer Multi-gas Monitor (B&K) is used to accurately measure the concentration of ammonia. Its measurement principle is based on the photoacoustic infrared detection method. It is configured with both ammonia and water vapor optical filters.

The monitor interfaces with the GMS-2 computer via RS-232 serial communications for data transfer and remote control. The custom program PHOTO runs on the GMS-2 computer to acquire data and control the B&K.

2.1.4 GMS-2 Host Computer

The GMS-2 host computer uses the multi-tasking operating system IBM OS/2. It collects the data gathered from the FTIR computer system, the GC computer system, and the B&K instrument. Other functions consist of sending the resulting data to DACS's computer system and archiving data to the appropriate location at the local hard disk. All of the programs that run on the GMS-2 computer are customized to fit specific applications. FTRANS, FTAN-B, GC3F, PHOTO, and DACS program functions are listed as follows:

- FTRANS copies raw data files from the FTIR computer to the GMS-2 computer for analysis. After the original file has been transferred, FTRANS removes it from the FTIR computer.
- FTAN-B performs peak picking and peak integration functions to convert raw interferograms to concentration data (analysis codes).
- GC3F transfers data files that were created on the GC computer by LABNET software. Other functions of GC3F are analyzing data, formatting data, and storing data to the appropriate location.
- PHOTO primary function is to accumulate each data point that was generated from the B&K instrument and store it locally at the GMS-2 computer.
- DACS collects the FTIR, GC, and B&K gas concentration data, formats it, and sends it via serial line to the DACS's computer system.

2.1.5 RGA5 Dual Reduction Gas Detector H2 Host System

The RGA5 Process gas analyzer host control system is configured for dual reduction gas detector (RGD) hydrogen analysis. The process script uses a subset of the capabilities of the RGA5 host protocol to perform routine storage of Hydrogen concentration data collected using two reduction gas detectors in a single RGA5. This application is designed to function in a Microsoft Windows Version 3.1 operating environment on a 386, 486, or Pentium based Intel computer.

The hardware used in this application consists of a dual column, Trace Analytical RGA5 configured solely for hydrogen monitoring. The internal software of the RGA5 is used for selecting the event program. It controls the sequence of events necessary to measure the amount of hydrogen in the normal sample stream using both detectors. Additionally, it periodically runs one of two calibration gases independently on each detector.

The host computer software is used to capture the resultant data and store it to appropriate files on the host computer's hard disk drive. The host monitors the RGA5 software so it knows when the program has completed, then the host downloads data. The current data is presented on a display in either graphic or tabular form. The previously saved data files can be viewed in the same manner. The status of the RGA5 is internally monitored. Flags to document appropriate instrument operation are provided for operator viewing. Any alarms are automatically displayed and saved.

Note: The data required by the host computer software is only available at the end of an RGA5 sequencer run. Therefore, the RGA5 must be running in the SEQ mode.

2.1.6 BlackBox - GC Inlet Gas line Valve Control System

The Blackbox is an independent interface system between the gas chromatographs and their integrator that is used to control the calibration verification runs. It is composed of an Allen Bradley Programmable Logic Controller (PLC) running a ladder logic program with a notebook computer to change parameters in the program.

2.2 PRINCIPAL OF OPERATION

2.2.1 SPFTIR - FTIR Spectrometer System Control

The program SPFTIR runs on the BIO-RAD SPC 3200 computer. Its main function is to control the infra-red scanning instrument, the FTIR Spectrometer. SPFTIR creates a unique file identified by date and index number for the raw data of each "scan" performed. The raw data stored in the file is actually the average of 128 scans performed by the instrument. The raw data file created is subsequently copied to the KINETICS computer where it is operated upon by the FTRANS program. SPFTIR scans and produces raw data files every 15 minutes continuously. The most recent raw data file is named in another file created by SPFTIR, called "scanlog". SPFTIR writes output to the console (standard output) for information to the operator. No user input is required.

- SPFTIR consists of one file with four functions: main, fname, ita, and scan (OS/2 ANSI standard C code).

2.2.2 FTRANS - FTIR Spectrometer System File Transfer

The FTRANS program copies FTIR data files from the SPC 3200 computer to the KINETICS OS/2 computer for analysis. After the original file has been transferred, FTRANS deletes it from the SPC 3200 computer.

- FTRANS consists of five files (FTANS, DET FILE, GET_FILE, TRANS_F, CLEAN) with six functions: main, det_file, get_file, trans_f, clean, tim_get (OS/2 ANSI standard C code).

2.2.3 FTAN-B - Infrared Data Analysis Codes

The FTAN-B program is operated in conjunction with FTANS to perform analysis. It performs peak picking and peak integration functions to convert raw interferograms to concentration data. See Appendix D for a Summary of Calibrations for the FTIR.

- FTAN-B consists of three files (FTANB, HEXTOLO, HEXTOBL) with three functions (OS/2 ANSI standard C code).

2.2.4 PHOTO - Ammonia Monitoring System

The program PHOTO runs on the OS/2 Kenetics computer. Its function is to log data from the B&K 1302 Gas Analyzer installed in GMS-2. PHOTO creates an ASCII data file with a unique name for each date. Typically data from the B&K is appended to this data file once every two minutes. The naming convention is YMMDBK.PRN.

- PHOTO consists of five files (PHOTO, COMOPEN, COMREAD, COMFLUSH, COMWRITE) with five functions (OS/2 ANSI standard C code).

2.2.5 DACS - GMS-2 Host Computer

The DACS program runs on the KINETICS OS/2 computer. Its primary function is to collect the GC and FTIR data, format it and send it via serial line to the Genesis computer system.

- DACS consists of one file with two functions: main and tim_get (OS/2 ANSI standard C code).

2.2.6 GC3 - RGA3 Reduction Gas Detector H2 Host System

The program GC3 is used for transferring, analyzing and formatting data. File Transfer Protocol (FTP) operations are performed through system command files generated by GC3, to transfer the files that were created on the BLACKADDER computer by Spectra Physics' LABNET software.

Files located in the d:\gc3 directory on BLACKADDER have names of the form "GHnnnnnn.RPT" and "GHnnnnnn.RAW". The ".raw" and ".rpt" files are transferred from the remote system (BLACKADDER) via FTP to the working directory on KINETICS where GC3F executes.

Once the files are transferred, the ".raw" file is renamed. The new name is keyed to the current date and has the form "ymmddnnn.ra3". The file is copied to a directory (also keyed to the date) named "ymmdd". For example, the file "\30726\30726123.ra3" is the 123rd data file created on 26 July 1993.

The ".rpt" file is opened and the time, date, area, and retention time data is read from the file. A gas concentration value is computed from the area value and a calibration factor. The resulting data is stored in two files: "ymmdd.pr3" and "gc3.dat".

The ".dat" file is used by the DACS program to send data to the Genesis system. The ".pr3" file is opened in append mode so that the data from all files transferred in one day is accumulated in one file. At the beginning of the next day (after midnight) the previous day's ".pr3" file is copied to the directory for the previous day, and then deleted from the working directory. The ".ra3" and ".rpt" files are deleted from the working directory. The ".raw" and the ".rpt" files on the remote system (BLACKADDER) are also deleted via a command file and FTP operations.

- GC3F consists of four files (GC3F, FIL_NAME, STRIP, CRE_NAME) with five functions: main, fil_name, strip, behind, cre_name (OS/2 ANSI standard C code).

2.1.7 BlackBox System

The Blackbox is an independent system composed of an Allen Bradley Programmable Logic Controller (PLC) running a ladder logic program with a notebook computer to change parameters in the program. The controller is set up to receive a fixed number of cycle pulse from the GC integrators. Each time the number is attained, the controller sends signals to the GC to change inlet ports. The calibration verification cycle will run for a pre-set number of cycle pulses, then the controller will send signals to the GC to change the inlet port back to the sample line.

3.0 FUNCTION AND DESIGN REQUIREMENTS

3.1 FUNCTIONAL FLOW

3.1.1 SPFTIR - FTIR Spectrometer System Control

Call Tree:

3.1.2 FTRANS - FTIR Spectrometer System File Transfer

Call Tree:

3.1.3 FTAN-B - Infrared Data Analysis Codes

Call Tree:

3.1.4 PHOTO - Ammonia Monitoring System

Call Tree:

3.1.5 DACS - GMS-2 Host Computer

Call Tree:

3.1.6 GC3 - RGA3 Reduction Gas Detector H2 Host System

3.2 SYSTEM INTERFACES

3.2.1 SPFTIR – FTIR Spectrometer System Control

3.2.1.1 Input Requirements

User Input Requirements: None.

Other Input Requirements:

Function main() reads a file called "scanlog" in the default directory. "Scanlog" contains a 8-character string which is the name of the last file into which raw IR data was scanned by the FTIR instrumentation. The 8-character string is of the form:

<u>Char Pos</u>	<u>Description</u>
0	Last digit of the current year
1 - 2	Two characters to identify the month, i.e. first and third characters of the month name, except if month is January, then first and second characters of the name: if January then "JA" if February then "FB" if March then "MR", ... etc.
3 - 4	Integer day number (1 - 31)
5 - 7	Index (or number) id of the last scan performed (0-999)

"Scanlog" must exist prior to initial execution time.

Function ita() reads a temporary file called "temp.tem" in the default directory. "Temp.tem" contains an integer which is the index id of the current scan. Ita() reads the index in string format to be used later in construction of the scan data file name mentioned above. "Temp.tem" does not need to exist prior to execution, in fact is created by ita() (see output requirements section).

3.2.1.2 Output Requirements

User Output:

SPFTIR outputs a record to standard output (console) which records its scan activity. The output record is of the form:

```
FIL_STAT: 0    FILE_NAME:    3SP01001
```

One record is generated for each scan. The record identifies the *current* scan to be performed. The record is generated before the scan is performed. Additional records are output to standard output if the scan number index exceeds 999, indicating an abnormal situation. The records generated are of the form:

```
Reset Data Sample Number
Changed To Alternate Dir
```

The records occur before the record which identifies the current scan activity.

Other Output:

Each scan operation writes raw scan data to a new file with an 8-character name of the form described below. This is the primary output of the SPFTIR program.

Function main() writes the file called "scanlog" in the default directory. "Scanlog" is written with an 8-character string which is the name of the last file into which raw IR data was scanned by the FTIR instrumentation. The 8-character string is of the form:

Char Pos Description

0	Last digit of the current year
1 - 2	Two characters to identify the month, i.e. first and third characters of the month name, except if month is January, then first and second characters of the name: if January then "JA" if February then "FB" if March then "MR", ... etc.
3 - 4	Integer day number (1 - 31)
5 - 7	Index (or number) id of the last scan performed (0-999)

"Scanlog" is written after the scan is performed.

Function ita() writes a temporary file called "temp.tem" in the default directory. "Temp.tem" is written with an integer which is the index id of the current scan. ita() writes the index in integer (%d) format to be read in a subsequent operation for construction of the scan data file name.

3.2.2 FTRANS - FTIR Spectrometer System File Transfer

3.2.2.1 Input Requirements

User Input Requirements: None.

Other Input Requirements:

File "paths" in the default directory is an ascii file containing 3 lines. The first two lines specify the directory paths for analysis data on the IBM and FT-IR computers, respectively. Line 3 is a long integer delay value. "Paths" must exist prior to execution. "Paths" is read by function main.

File "getlog" in the default directory is an ascii file containing 1 line. The line has the name of the most recent file acquired from the FT-IR computer. The file name is an 8 character string with the following form:

<u>Char Pos</u>	<u>Description</u>
0	Last digit of the current year
1 - 2	Two characters to identify the month, i.e. first and third characters of the month name, except if month is January, then first and second characters of the name: if January then "JA" if February then "FB" if March then "MR", ... etc.
3 - 4	Integer day number (1 - 31)
5 - 7	Index (or number) id of the last scan performed (0-999)

"getlog" must exist prior to execution. "Getlog" is read by functions: main, get_file, det_file, clean_file, and trans_file.

File "scanlog" in the default directory is an ascii file of the same form as "getlog". "Scanlog" is read by function det_file. "Scanlog" does not need to exist prior to execution.

File "ftanb.dat" in the default directory is checked for existence by function main.

File "temp.dat" in the default directory is checked for existence by function main.

Function clean_file creates a file name (dynamic) which has a form like the filename in the files "getlog" or "scanlog", but with the characters ".dt" appended to the name. The file is checked for existence in the data analysis directory which was specified in the "paths" file.

3.2.2.2 Output Requirements

User Output:

Function main console output:

"DosSleep ERROR nn"

Function get_file console output:

"GET FILE"

"getLog file is not open"

Function det_file console output:

"DET FILE"

"SCAN NAME: %s GET NAME %s"

"DosSleep ERROR nn"

Function clean_file console output:

"CLEAN FILE"

"PROBLEM IN GETTING SCAN_FILE"

Function trans_file console output:

"TRANS_FILE"

Other Output:

File "mkfile.cmd", "1st.cmd", "prn_make.cmd", "dacs_te.cmd" are opened and written by function main. Each is a system command procedure file.

File "ftanb.epm" is opened and written by function main. "Ftanb.epm" is an ascii file containing 2 lines, a directory spec and a file spec, respectively.

File "ftir.inp" is opened and written by function get_file. "Ftir.inp" is a command file which contains the commands to get the data file from the FT-IR computer.

File "ftir.inp" is opened and written by function det_file. "Ftir.inp" is a command file which contains the command to copy the "scanlog" file from the FT-IR computer.

File "getlog" is opened and written by function det file as soon as the names in "scanlog" and "getlog" are different. "Getlog" and "scanlog" are defined in the input requirements section.

File "ftir.inp" is opened and written by function clean_file. "Ftir.inp" is a command file which contains the commands to delete the data file on the FT-IR computer.

File "clef.cmd" is opened and written by function clean file.
 "Clef.cmd" is a command file which contains the commands to delete the data file on the IBM computer.

File "trans_cp.cmd" is opened and written by function trans file.
 "Trans cp.cmd" is a command file which contains the commands to copy the FT-IR data file and the "getlog" file to the data analysis directory.

3.2.3 FTAN-B - Infrared Data Analysis Codes

3.2.3.1 Input Requirements

User Input Requirements: None.

Other Input Requirements:

Files "ftanb.epm", "fname", and "peak.tab" in the default directory are read by function main. Two string items are read from "ftanb.epm" file and used to assemble each successive "fname". The "n" character items (raw data file length), each on separate lines, are read from the "fname" data file. A long integer (the number of peaks for calculation), and six long floating point items, each on separate lines, for each of the number of peaks are read from the "peak.tab" data file. The files must exist at execution time.

Function main() reads a file called "ftanb.epm" in the default directory. "Ftanb.epm" is an ascii file containing 2 lines, a directory spec and a file spec, respectively. This file is used to assemble the current data file name in "fname". "Fname" contains the raw IR data scanned by the FTIR instrumentation.

3.2.3.2 Output Requirements

User Output:

One analysis record is generated for each scan and placed in "ftanb.dat". The record identifies the name, date, ammonia peak area, nitrous oxide peak area, ammonia concentration, nitrous oxide concentration, methane concentration, and methane peak area.

Other Output:

Additional records are output to standard output as needed, indicating an abnormal situation. The records generated are of the form:

DosOpen ERROR nn in line 44

Each scan operation writes raw scan data to a new file with an 8-character name of the form described below. This is the primary output of the SPFTIR program.

3.2.4 PHOTO - Ammonia Monitoring System

3.2.4.1 Input Requirements

User Input Requirements:

When the program is initialized, it will print the following to the screen.

```
"THE CURRENT SETUP IS NUMBER nn"
"IS THIS THE CORRECT SETUP Y or N"
```

If the answer is "n" the program will print the following to the screen.

```
"ENTER THE NEW SETUP NUMBER"
```

After the operator enters the new setup number, the program will print the following to the screen.

```
"THE CURRENT SETUP IS NUMBER nn"
```

After the setup number is verified, the program begins continuous operation.

Other Input Requirements:

Function "comopen" requires information on the baud rate for verification the port is open and connected. Function "comread" reads the value of the data buffer.

Function "main" opens and reads a file "bkcmd".

3.2.4.2 Output Requirements

User Output:

Function main console output:

```
"Could not set SIGINT"
"Measurement Status nn"
```

Function comopen console output:

```
"Cannot open the Com Port to write RC=nn"
"Cannot write SETLINECTRL to the Com Port RC=nn"
"Cannot write SETBAUDRATE to the Com Port RC=nn"
"Cannot write DBD to the Com Port RC=nn"
"Cannot get baud rate from the Com Port RC=nn"
"BAUD RATE nn"
"Cannot get line ctrl from the Com Port RC=nn"
"DATA BIT nn"
"PARITY nn"
```

"STOP BITS nn"
 "BREAK nn"
 "Cannot get DCB from the Com Port RC=nn"
 "Write Time Out nn Sec"
 "Read Time Out nn Sec"
 "Flag1 nn"
 "Flag2 nn"
 "Flag3 nn"
 "Error Replacement Char nn"
 "Break Replacement Char nn"
 "XON Char nn"
 "XOFF Char nn"

Function comflush console output:

"Cannot write get read buff size to the Com Port"

Function comwrite console output:

"Error in writing the string to the Com Port"

Other Output:

PHOTO creates an ASCII data file with an unique name for each date. Typically, data from the B&K is appended to this data file once every two minutes. The naming convention is YMMDDBK.PRN. If there is an error reading the buffer it is appended to "bkerrlog".

3.2.5 DACS - GMS-2 Host Computer

3.2.5.1 Input Requirements

User Input Requirements: None.

Other Input Requirements:

File "temp.prc" in the directory "c:\cprog\gc3" is read by function main. The file "temp.prc" contains a single line of 5 ascii strings, tim, area, rt, con, and file.

File "temp.dat" in the directory "c:\cprog\ftir" is read by function main. "Temp.dat" contains a data string to be sent to the DACS computer.

3.2.5.2 Output Requirements

User Output:

DACS writes several console error messages under various conditions:

"DosSleep ERROR nn"

At the beginning of the operation which sends information out the COM port to the DACS computer, the DACS program sends the following messages to the console:

"***** SENDING GC FILE filename TO DACS *****"

"***** SENDING FTIR FILE filename TO DACS *****"

Various console error messages are displayed if the send operation to DACS encounters a difficulty:

"Cannot open the Com Port to write RC=nn"

"Cannot write pram to the Com Port RC=nn"

"Error in writing the string to the Com Port RC=nn"

"Error in closing the Com Port RC=nn"

Other Output:

Primary DACS output consists of sending ascii character data out the COM1 port to the DACS computer. The output operation is performed by a library of functions which access the port outside of normal C I/O. The functions are named DosOpen, DosWrite, etc. The DosWrite function takes a character string argument and its length (number of characters) among other parameters, and copies the string to the COM port, returning the actual number of characters written successfully.

3.2.6 GC3 - Dual Reduction Gas Detector H2 Host System

3.2.6.1 Input Requirements

User Input Requirements: None.

Other Input Requirements:

File "gc3.dat" in the default directory is read by function main. Four items, each on separate lines, are read from the file, first a string, then an int, and finally a long float. "Gc3.dat" must exist at execution time.

File "ti.dat" in the default directory is read by function main. A single int is read. The file must exist at execution time.

Functions cre_name and fil_name both read a file "itoa". Both files are used as temporary storage during an integer-to-ascii conversion operation. The files do not need to exist at execution time, in fact, are created as needed (see output requirements section).

Function main checks for existence of two files. The file names are dynamic, however, the names are of the form "GHnnnnnn.RPT" and "GHnnnnnn.RAW". The files themselves are not read, rather their names are used in several file transfer (FTP) operations (see output requirements). Function strip reads the first line of "GHnnnnnn.RPT"

Function main checks for existence of two additional files. Again, the names are dynamic, and of the forms "ymmddnnn.ra3" and "ymmdd.pr3". The files are not read but used in subsequent FTP operations.

3.2.6.2 Output Requirements

User Output:

Function main writes numerous console messages, most indicating error conditions:

```
"ERROR IN OPENING GC3.DAT"
"ERROR IN OPENING TI.DAT"
"DosCreateMutexSem: nn"
"flag1: nn"
"Dos Sleep ERROR nn in line 89"
"OFF: nn"
"Dos Sleep ERROR nn in line 96"
"DosRequestMutexSem ERROR: nn"
"f_name sss"
"r_name sss"
"File sss was not found GC3 may have problem"
"There is a problem with GC3 no RPT files have been"
"transferred in 7.5 minutes exiting program"
"There is a problem with GC3 no RAW files have been"
"transferred in 7.5 minutes exiting program"
"ERROR1 nn"
"ERROR2 nn"
"FILE nn WAS BAD"
"c_name: sss"
"Dos Copy ERROR nn in line 166"
"Dos Copy ERROR nn in line 172"
"Dos Delete ERROR nn in line 186"
"Dos Delete ERROR nn in line 224"
"Dos Move ERROR nn in line 241"
"Dos Copy ERROR nn in line 247"
"Dos Copy ERROR nn in line 252"
"Dos Sleep ERROR nn in line 228"
"TRANSFER COMPLETED IN nn SECONDS"
```

Function behind writes the following console messages:

```
"LTIM: nn"
"CTIM: nn"
"DIFF: nn"
"line 331"
```

Function cre_name writes one console message:

"Dos Create Dir error nn in line 29 of cre_name.c"

Function strip writes several console messages:

"Bad Time"
 "Bad Date"
 "Bad Area"
 "Bad RT"

Other Output:

The primary output of the GC3F program is several system command files that are subsequently used to execute FTP operations to transfer, delete, and rename data files.

File "tran.inp" in the default directory is written by function main. "Tran.inp" is a command file which transfers the "ghnnnnnn.rpt" and "ghnnnnnn.raw" files from the remote system (BLACKADDER) to the default directory. Subsequently, "tran.inp" is written again by function main, this time with the commands which cause an FTP delete of the original ".rpt" and ".raw" files on the remote system.

Temporary file "itoe" is written by functions cre_name and fil_name in int-to-ascii conversion operations.

Function main writes a file "temp.prc" in the default directory. "Temp.prc" contains a line of data which includes the values of time, area, etc. Function main also writes a file in the default directory named "ymmdd.pr3". This last file is opened for *append* to accumulate data regarding each file transfer operation during the current day.

3.2.7 BlackBox System

3.2.7.1 Input Requirements

User Input Requirements: None.

Other Input Requirements:

Cycle pulses from GC system integrator.

3.2.7.2 Output Requirements

User Output: None.

Other Output Requirements:

Control signal connection to the gas chromatograph used to change gas inlet port connections. The gas chromatograph inlet port can be set on the sample gas, the calibration standard gas, and on the back flush, each for a different, preset number of cycles.

4.0 OPERATION

4.1 OVERVIEW

The purpose of the operation procedure (Appendix B) is to provide instructions for the startup and operation of the Kenetics computer in the data acquisition and control system (DACS) trailer and the following equipment on the SY tank farm:

- 1) AST Premium II 486/33 computer (also called the "blackadder"),
- 2) Trace Analytical Gas Chromatograph Model RGA3 (designated GC-3),
- and 3) BioRad FTS 25 Fourier Infrared Spectrometer (FTIR).

The gas chromatograph measures hydrogen concentration in the 241-SY-101 ventilation exhaust header. The fourier infrared spectrometer measures ammonia and nitrous oxide.

4.2 OPERATING PROCEDURES

4.2.1 System Startup, Normal Operation, Shut Down

See Appendix B for current operating procedures.

5.0 DISCREPANCY REPORTING AND CORRECTIVE ACTIONS

Any person who identifies any problems or discrepancy at any time shall report it to the cognizant engineer. The cognizant engineer shall include a written description of the problem or discrepancy in the design file under "Problems". The cognizant 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 engineer are responsible for assuring correction of the problem or discrepancy.

6.0 CONFIGURATION CONTROL

Each phase of the gas monitoring system life cycle shall have configuration control. The approval, release, and control of the supporting documents shall follow WHC-CM-6-1, EP-1.7, and EP-2.2, "Engineering Document Control".

The software custodian responsible for maintaining configuration control of the software covered by this document shall be the cognizant organization responsible for Characterization and Monitoring Systems. The cognizant organization responsible for Characterization and Monitoring Systems shall control software code listings and configuration records. The cognizant organization responsible for Characterization and Monitoring Systems may use an automated approach with electronic record maintenance.

6.1 CONFIGURATION IDENTIFICATION

A unique descriptive name and revision number shall be assigned to each software file. The supporting document number, title, and revision number assigned to the software design description shall be used for labeling each floppy diskette or CD rom disk. The software design description will include the software listing and will be updated with each software change (including error corrections). The content of each software file will contain the name and revision number. Each revision will include the revision date. The initial revision, Rev. 0, shall be the baseline. The descriptive label shall appear as shown below:

- WHC-SD-XXX-XXX-XXX (supporting document number)
- GMS-2 software disk (title)
- Revision number/revision date
- Diskette number

6.2 CONFIGURATION CHANGE 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

6.3 CONFIGURATION STATUS ACCOUNTING

Characterization and Monitoring Systems 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.

6.4 MEDIA CONTROL

The form of media used to store the software shall be floppy disks or CD rom disks. The software media shall be stored in two locations. One set of the software media shall be stored in a locked file cabinet labeled "software records" in the file named "GMS-2 software". This file cabinet will be located at a location specified by the manager of Characterization and Monitoring Systems. The other set of software media shall be stored in a file that is maintained by the software developer.

7.0 SAFETY, RELIABILITY, AND QUALITY ASSURANCE

7.1 SAFETY CLASSIFICATION AND SIGNATURE DESIGNATION

The monitoring instruments and systems used for the Westinghouse Hanford Waste Tank Gas Monitoring System 2 are designate Safety Class 3 (Payne). The Safety Class 3 Equipment Classification level is for Systems and components that address the health and safety of facility workers and radioactive, chemical, or thermal releases to the environment.

At a minimum, documentation at WHC must satisfy an SQ Documentation Approval Designation Level for Safety Class 3 systems 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.

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

- 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.
- WHC, 1993, QR 19.0, *Software Quality Assurance Requirements*, WHC-CM-4-2, *Quality Assurance Manual*, Westinghouse Hanford Company, Richland Washington.
- WHC, 1988, WHC-CM-6-1, *Engineering Practices*, Westinghouse Hanford Company, Richland Washington.
- WHC, 1987, WHC-CM-1-3, *Management Requirements and Procedures*, Westinghouse 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 WESTINGHOUSE 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

THIS PAGE INTENTIONALLY
LEFT BLANK

APPENDICES

THIS PAGE INTENTIONALLY
LEFT BLANK

APPENDIX A - COMPUTER SYSTEM DESCRIPTIONS

CONFIGURATION DATA : BLACKADDER

HARDWARE:

Model: AST Premmia 4/33, AT bus compatible

Processor: 486, 33 MHz

Memory: 16MB

Disks: A: - 3.5" floppy
320MB internal hard drive
C: - 160MB internal partition
D: - 160MB internal partition

Ethernet: 3Com Etherlink II

Video: VGA

Instrumentation:

Model: SpectraPhysics Chromjet - CH2 SP4400 Integrator Assembly

Model: Trace Analytical RGA3 Reduction Gas Analyzer

SOFTWARE:

O/S: IBM OS/2 V2.0

Network: TCP/IP for OS/2 V1.2.1

Applications:

Two programs run on BLACKADDER: CHROMNET and LABNET. Each runs in a DOS window from OS/2. Both are commercial packages supplied by Spectra Physics.

CHROMNET - Runs the Chromjet integrator and, indirectly, the RGA3 reduction gas analyzer; creates D:\GC3\GHnnnnnn.RAW files which are available to and used by the GC3F program running on KINETICS.

LABNET - Acts as the communication interface between CHROMNET and the integrator.

CONFIGURATION DATA : KINETICS**HARDWARE:**

Model: AST Premium 4/66

Processor: 486, 16 MHz

Memory: 16 MB

Disks: A: - 3.5" floppy
 1GB internal hard drive
 C: - 500MB internal partition ESDI for operating system software
 D: - 500MB internal partition for data archiving
 E: - 90MB Bernoulli
 F: - 90MB Bernoulli

Ethernet: 3Com Etherlink II, MicroChannel

Video: IBM 8514 adapter, IBM 8515 display

Instrumentation:

NONE

SOFTWARE:

O/S: IBM OS/2 V2.0

Network: TCP/IP for OS/2 V1.2.1

Applications:

The following applications run on KINETICS: FTRANS, GC3F, and DACS. All are applications developed by CLS-2.

FTRANS - The FTRANS program copies FTIR data files from the SPC 3200 computer to the KINETICS OS/2 computer for analysis. After the original file has been transferred, FTRANS deletes it from the SPC 3200 computer.

GC3F - The program GC3F is used for transferring, analyzing and formatting data. FTP (File Transfer Protocol) operations are performed through system command files generated by GC3F, to transfer the files that were created on the BLACKADDER computer by Spectra Physics' LABNET software.

Files located in the d:\gc3 directory on BLACKADDER have names of the form "GHnnnnnn.RPT" and "GHnnnnnn.RAW". The ".raw" and ".rpt" files are transferred from the remote system (BLACKADDER) via FTP to the working directory on KINETICS where GC3F executes.

Once the files are transferred, the ".raw" file is renamed. The new name is keyed to the current date and has the form "ymmddnnn.ra3". The file is copied to a directory (also keyed to the date) named "ymmdd". For example, the file "\30726\30726123.ra3" is the 123rd data file created on 26 July 1993.

The ".rpt" file is opened and the time, date, area, and retention time data is read from the file. A gas concentration value is computed from the area value and a calibration factor. The resulting data is stored in two files: "ymmdd.pr3" and "gc3.dat". The ".dat" file is used by the DACS program to send data to the Genesis system. The ".pr3" file is opened in append mode so that the data from all files transferred in one day is accumulated in one file. At the beginning of the next day (after midnight) the previous day's ".pr3" file is copied to the directory for the previous day, and then deleted from the working directory. The ".ra3" and ".rpt" files are deleted from the working directory. The ".raw" and the ".rpt" files on the remote system (BLACKADDER) are also deleted via a command file and FTP operations.

DACS - The DACS program runs on the KINETICS OS/2 computer. Its primary function is to collect the GC (gas chromatograph) and FTIR (Fourier Transform InfraRed spectrometer) data, format it and send it via serial line to the Genesis computer system.

CONFIGURATION DATA : SPC3200**HARDWARE:**

Model: BIO-RAD SPC 3200
Processor: Motorola 68030, 16 MHz
Memory: 1 MB
Disks: A: - 3.5" floppy
C: - 105MB internal

Instrumentation:

Model: BIO-RAD ETS-60 FTIR Spectrometer Bench

SOFTWARE:

O/S: Idris
Network: TCP/IP V

Applications:

SPFTIR - The program SPFTIR runs on the BIO-RAD SPC 3200 computer. Its main function is to control the infra-red scanning instrument, the FTIR Spectrometer. SPFTIR creates a unique file identified by date and index number for the raw data of each "scan" performed. The raw data stored in the file is actually the average of 128 scans performed by the instrument. The raw data file created is subsequently copied to the KINETICS computer where it is operated upon by the FTRANS program. SPFTIR scans and produces raw data files every 15 minutes continuously. The most recent raw data file is named in another file created by SPFTIR, called "scanlog". SPFTIR writes output to the console (standard output) for information to the operator. No user input is required.

LOGICAL NETWORK DIAGRAM: WHC

By: M. P. Manke, EG&G/EM

**THIS PAGE INTENTIONALLY
LEFT BLANK**

**APPENDIX B - OPERATION INSTRUCTIONS
GC-3 GAS CHROMATOGRAPH
FTIR SPECTROMETER AND COMPUTERS**

B.1 PURPOSE

The purpose of this procedure is to provide instructions for the startup and operation of the Kenetics computer in the data acquisition and control system (DACS) trailer and the following equipment on the SY tank farm:

- 1) AST Premium II 486/33 computer (also called the "blackadder"),
- 2) Trace Analytical Gas Chromatograph Model RGA3 (designated GC-3),
- and 3) BioRad FTS 25 Fourier Infrared Spectrometer (FTIR).

The gas chromatograph measures hydrogen concentration in the 241-SY-101 ventilation exhaust header. The fourier infrared spectrometer measures ammonia and nitrous oxide.

The instructions are divided into four sections as follows:

- ~ 4.0, Section 1 - Initial Assessment and Restart from DACS, if Possible
- ~ 5.0, Section 2 - Complete System Reboot
- ~ 6.0, Section 3 - Kenetics Only Restart from Lock-up Condition
- ~ 7.0, Section 4 - FTIR Restart from DACS

B.2 PREREQUISITES

- 2.1 Equipment is located in the DACS trailer and in Building GMS-2 adjacent to Tank 241-SY-101. Access to GMS-2 is controlled by radiation work procedure and 241-SY Operations supervision. Obtain the GMS-2 building key (and other keys, if required) from the 241-SY Operations supervisor located in Building 272WA, Room 4.
- 2.2 Instrument power is supplied from JB2 distribution panel (power), H-2-815228, on the north wall of GMS-2.
- 2.3 Nitrogen carrier gas, prepurified grade, 99.997 percent minimum purity, is supplied from a gas distribution manifold located outside GMS-2.
- 2.4 Hydrogen calibration gas is supplied from a pressurized cylinder located outside GMS-2.
- 2.5 A Metal Bellows pump, Model MB-302XP, is located outside GMS-2 and delivers sample gas to the chromatographs.
- 2.6 The OS/2 operating system is automatically loaded when the computer is initially powered up.
- 2.7 Gas chromatographs were installed and started up per instrument manufacturer's instructions.

B.3 OS/2 OPERATING SYSTEM

3.1 There are a few things to understand prior to utilizing the system. The system runs OS/2 Version 2.0 and uses a graphical user interface (similar to Windows).

3.1.1 To bring up a particular window (make it the active window), you can do one of three things:

- a) Click the left trackball button with the cursor inside the window (this assumes you can see the window).
- b) Hold down CTRL and ESC at the same time. This will bring up the "Window List." Click (left button) on the desired window title in the listing.
- c) Press down BOTH the left and right trackball buttons at the same time. This will bring up the "Window List." Click (left button) on the desired window title in the listing.

3.1.2 The procedure will also ask you to minimize a window. This means to click on the box just to the left of the box in the upper right-hand corner of the active window. The upper right-hand corner box is a square (the maximize button); the minimize button is just to the left of it and has only the corner outlines of a box. If you minimize a program, it is still running; it is just not visible on the screen. You can easily bring it up by calling up the "Window List" as described above in Step 3.1.1.

B.4 INITIAL ASSESSMENT AND RESTART FROM DACS, IF POSSIBLE

SECTION 1

4.1 Initial Assessment

- 4.1.1 In the event of a GC-3 failure (no updates to DACS), please perform the following steps.

4.2 GC-3 Restart

- 4.2.1 At the GMS-2 computer (in the back of Rack 9 in DACS), look at the window with the title bar labeled "DACS." If the variables "areagc3, rtgc3, and congc3" are all zeros, then the problem is with the gc3f program or with the GC-3 integrator in the GMS-2 shack (continue with the procedure). If the variables are NOT zeros, then there is a problem with the ASCII Basic Module and DACS EG&G personnel will have to be notified. No further actions are required for GC-3.

- 4.2.2 Look at the window with the title bar labeled "gc3f." The gc3f program may be stopped or it may be looping--looking for a file (the most likely cause). If looking for a file, you would most likely see something like, "File GH000765.RAW not found." Write the filename down for later use by the GMS-2 team.

- 4.2.3 Open an OS/2 window (double-click on the OS/2 window icon at the top of the screen). You will see an OS/2 command line prompt. At the prompt, type the following:

```
[C:]ftp blackadder <ENTER>
```

- 4.2.4 The system will display a couple of lines of information, followed by a prompt. Type "anonymous" after the prompt and press ENTER (as below):

```
Name (blackadder): anonymous <ENTER>
```

- 4.2.5 You will then be logged in. An "ftp" prompt will eventually be displayed. To check the GC3 directory on the GMS-2 computer on the farm, type the following at the prompt:

```
ftp> dir gc3 <ENTER>
```

- 4.2.6 You will now see a directory of files for GC-3 that are still out on the farm and have not been brought over to DACS. If there are no files listed or if the only file is 3A.TMP, then nothing more can be done from DACS. A farm entry is required to restart the integrators and network communications. Proceed to 6.0, Section 2 - Complete GC System Reboot, of this procedure.

- 4.2.7 If there are files on the farm, you will then need to determine the first PAIR of files that are out there. The files should be in pairs of the same filename with the extensions RAW and RPT. You may find a lone file; if so, write down the filename for later use by the GMS team. The filename of the lowest numerical pair of files (numbered in sequence, they are of the form GH003456.RAW, etc.) is extremely important for restarting GC-3 updates. Write down the number from the first pair (i.e., if the pair is GH004321.RAW and GH004321.RPT, the number is 4321).
-

Note: You can use the CTRL-S and CTRL-Q key combinations to stop (CTRL-S) and restart (CTRL-Q) the directory listing if there are too many files to view at one time.

- 4.2.8 Type "quit" at the prompt (as below):

```
ftp> quit <ENTER>
```

- 4.2.9 This will place you back at the [C:\] prompt. Now type the following:

```
[C:\]type c:\gc3\*.pr3 <ENTER>
```

- 4.2.10 This will give you the latest listing for GC-3 data brought over to DACS. When the listing is done, the last line will be of the form:

```
09/04/93 04:21:44 553 0.370000 46.818778 30904098.ra3,
```

and you will be back at the [C:\] prompt. The filename with the ra3 extension (30904098 in the above case) is important here for restarting GC-3. The leading "3" stands for 1993, the next two numbers are the month (09 in this example), the next two numbers are the day (04 in this case), and the last three digits are the file number for the day (they start at "001" after midnight). Record the last three numbers here (without leading zeros; in our example, it would be 98).

- 4.2.11 At the OS/2 command prompt, type "exit" as shown below:

```
[C:\]exit <ENTER>
```

This will cause the OS/2 window to disappear.

- 4.2.12 Move the mouse pointer into the center of the gc3f window and click the left button. This should cause the gc3f window title bar to become highlighted.
- 4.2.13 Press Ctrl-Break a few times, until the gc3f program exits and you see the [C:\GC3] prompt.
- 4.2.14 Again, move the mouse pointer into the center of the gc3f window and click the left button. This should cause the gc3f window title bar to become highlighted.
- 4.2.15 This step will restart the GC-3 updates to DACS. You will need the numbers from Steps 4.2.7 and 4.2.10 for this step. At the [G:\GC3] prompt, type the following:

[C:\GC3]gc3f [Step 4.2.7 number] [Step 4.2.10 number + 1] <ENTER>

In other words, if the number from Step 4.2.7 is 4321 and the number from Step 4.2.10 is 98, then the line would read:

[C:\GC3]gc3f 4321 99 <ENTER>

Make sure you type gc3f, NOT gc3. Make sure there is a space after gc3f and another space between the two numbers. Also make sure the last number is the number from Step 4.2.10 PLUS ONE.

- 4.2.16 This should restore the GC-3 updates to DACS. The program will wait 3 minutes before starting ANY updates, so be patient!

B.5 COMPLETE GC SYSTEM REBOOT

SECTION 2

5.1 DACS Kenetics Prerequisites

- 5.1.1 Make the gc3f window active. Press Ctrl-Break a few times until the gc3f program exits and you see the [C:\GC3] prompt (unless the program is stopped already and is currently at the [C:\GC3] prompt). If the program does not stop looping, repeat the Ctrl-Break key strokes.
- 5.1.2 Notify Shift Operations supervisor that access to the tank farm is needed, obtain keys, and WRAM in.

5.2 Computer (Blackadder) Shutdown Procedure

- 5.2.1 The blackadder computer in GMS-2 should have the blue "Labnet" screen active. If not, make Labnet active. Close out of Labnet by pressing the F10 key and responding to the prompt.
- 5.2.2 Press the right mouse button. Highlight the "shutdown" option and enter yes to all system queries.
- 5.2.3 At the system prompt, turn off the computer.

5.3 Computer (Blackadder) Startup Procedure

- 5.3.1 Verify that a floppy disk is not loaded in Disk Drive A.
- 5.3.2 Turn on power to the monitor by operating the switch located under the lower right corner of the monitor. If no image is seen, check the brightness control.
- 5.3.3 Turn on power to the computer by operating the switch located on the front panel of the computer.
- 5.3.4 Wait until automatic loading of programs is complete and [C:\] is displayed in the STARTUP.CMD window.
- 5.3.5 Minimize the multiple windows by positioning the arrow on the small square in the upper right corner of each window and clicking once on the left mouse button.
- 5.3.6 Verify that two rows of OS/2 icons are displayed across the top of the screen.

5.4 Gas Chromatograph GC-3 Load Method Files

- 5.4.1 Locate the power switch on the back of the GC-3 integrator. The switch is located next to a large cable connector near the right side and is difficult to find.

CAUTION

DO NOT TURN OFF INTEGRATOR POWER UNLESS "READY" IS DISPLAYED ON THE INTEGRATOR READOUT PANEL.

- 5.4.2 Turn the integrator power switch OFF and then ON to erase method files which were previously stored in integrator RAM.
- 5.4.3 Respond to the prompt on the integrator screen asking which program to run by pressing 2 <ENTER>. This identifies the method of data collection as "Labnet Data Capture."
- 5.4.4 At the GC-3 integrator keyboard, type the command POKE #1FFE9,1 <ENTER> which will turn the integrator printer OFF. (Changing the last 1 to 0 turns the printer ON.)
- 5.4.5 At the integrator keyboard, press the LCD status key. The LCD status command can be entered at any time.
- 5.4.6 Verify that the GC-3 integrator clock and the Kenetics computer clock are synchronized within 5 seconds.
- 5.4.7 Locate the manual control panel on the front of the Trace Analytical gas chromatograph and press ZERO.
- 5.4.8 Return to the computer keyboard and move the pointer arrow to the LABNET icon and double-click on the left mouse button.
- 5.4.9 Simultaneously depress the CTRL and ESC keys to open the PC menu selection window.
- 5.4.10 Verify that LABNET is listed on the "Windows List" screen as one of the menu items.
- 5.4.11 Move the pointer arrow to the CHROMNET icon and double-click on the left mouse button to open CHROMNET.
- 5.4.12 Simultaneously depress the CTRL and ESC keys to open the PC menu selection window.
- 5.4.13 Verify that CHROMNET-CN.BAT is listed as one of the menu items.
- 5.4.14 Move the pointer arrow to the OS/2 WINDOW icon and open an OS/2 window by double-clicking on the left mouse button. If an OS/2 window is already running, then open it by using the arrow keys to highlight OS/2 WINDOW from the menu selection and press <ENTER>.
- 5.4.15 Type D: <ENTER> at the prompt.
- 5.4.16 Type CD GC3 <ENTER> at the D: prompt. This is the directory that all data from the integrator will be sent to.

- 5.4.17 Type DIR <ENTER> to display the contents of the D:/GC3 directory.
- 5.4.18 If more than 50 files are displayed as *.RPT or *.RAW files, then these should be copied to disk and then erased by the command DEL GH*.R* <ENTER>. If less than 50 files are displayed as *.RPT or *.RAW files, then these files should be erased by the command DEL GH*.R* <ENTER>. This erasure will ensure that there will be no error messages due to an attempt by the PC to overwrite existing files. Minimize the OS/2 window by positioning the arrow on the small square in the upper right corner of the window and clicking once on the left mouse button.
- 5.4.19 Simultaneously depress the CTRL and ESC keys to open the PC menu selection window.
- 5.4.20 Use the arrow keys to select the CHROMNET program and press <ENTER>.
- 5.4.21 Use the arrow keys to move to SYSTEM SELECTION <ENTER>.
- 5.4.22 The window entitled "Configuration Selection" should be displayed. Use the arrow keys to move to LOAD FROM DISK <ENTER>.
- 5.4.23 Use the arrow keys to select GC3 <ENTER>. This command specifies that the integrator communicating with GC3 will be loaded with the correct method files. If a message appears on the screen, then select "proceed" or the option which allows you to continue.
- 5.4.24 Use the arrow keys to select METHOD SETUP <ENTER>. This command allows the operator to select the method files to load into GC3.
- 5.4.25 From the "Method Selection" window, use the arrow keys to select LOAD FROM DISK <ENTER>. The method file to load has been created and must be identified.
- 5.4.26 Use the arrow keys to identify GC3ACAL.MTD <ENTER>. The method file GC3ACAL.MTD has been identified to load into the integrator.
- 5.4.27 Use the arrow keys to select PREPARE A SYSTEM DOWNLOAD <ENTER>. Download options will be selected in the windows which follow.
- 5.4.28 The window entitled "System Preparation Options" should be displayed with SEND THE METHOD ONLY highlighted <ENTER>.
- 5.4.29 The window entitled "System Send Options" should be displayed with DEVICE FILE #1 highlighted <ENTER>.
- 5.4.30 The window entitled "Data Capture Options" should be displayed. Selections are as follows:

CAPTURE DATA YES/NO <ENTER> YES option to save Chromatogram,
 {Note: last line in GC3.DAT on
 KENETICS must be 1}.
 NO option to ignore Chromatogram,
 {Note: last line in GC3.DAT on
 KENETICS must be 0}.

CAPTURE REPORT YES <ENTER>
DISK FILENAME GH000001 <ENTER>
PATH NAME d:\gc3 <ENTER>
LNET UNIT NUMBER 1 <ENTER>
SAMPLE NAME SAME AS FILENAME NO <ENTER>

- 5.4.31 The PC will take a few minutes to download the method file into File #1, Channel A of the #3 integrator. Note the status message which is displayed on the second line from the bottom of the screen. Wait until the text on the screen displays PREPARE A SYSTEM DOWNLOAD highlighted on a blue background, then proceed to the next step.
- 5.4.32 Use the arrow keys to select ABANDON METHOD <ENTER>. Because there were not changes to the file, this can be closed without saving.
- 5.4.33 Type YES at the prompt. This will return you to the CHROMNET screen with METHOD SETUP highlighted with a blue background.
- 5.4.34 Press the "F10" key to close CHROMNET.
- 5.4.35 The red window entitled "Exit to DOS?" should be displayed. Use the arrow key to select "YES" and press <ENTER>.

5.5 GC-3 Calibrator Setup for Selectable Start Time

- 5.5.1 The GC calibrator or "blackbox" is used to program automatic calibrations of GC-3. The calibrator is an interface between the gas chromatograph and the integrator that controls operation of the internal GC 4-port valves and external solenoid valves. The interface is comprised of an Allan Bradley controller and a IBM laptop PC. Accumulator settings described in this procedure are inputted through an IBM laptop PC. The program is written in ladder logic. The GC calibrator is located below the GC-3 integrator. The IBM laptop is located in the equipment rack.

The following steps provide instructions to program GC calibrations for one calibration cycle (six injections) per day. Starting times for the GC-3 calibration cycles must be selected to calculate ACCUMULATOR values.

- 5.5.2 Verify PRESET values on laptop PC.

- a) GC-3 integrator counter: C5:7 PRESET = 570
- b) GC-3 calibration counter: C5:8 PRESET = 7

5.5.3 Reprogram GC-3 ACCUMULATOR values.

- a) Record time of GC startup (Inject A)
- b) Record start time of desired calibration cycle
- c) Calculate delta time in hours (Step 5.5.3a - Step 5.5.3b)
- d) Multiply by 24 (Step 5.5.3c * 24)
- e) Subtract value from Step 5.5.3d from 570 as follows:
(570 - Step 5.5.3d)
- f) Move the cursor with the arrow keys and highlight the C5:7 ACCUMULATOR value.
- g) Enter resultant value from Step 5.5.3e into C5:7 ACCUMULATOR
- h) Press the <ENTER> key.

5.5.4 To disable automatic calibrations.

Note: Perform this step only to terminate automatic calibration.

- a) For GC-3: Set C5:8 PRESET to 1

5.5.5 To initiate a manual calibration sequence.

Note: This step to be performed only for manual calibration.

- a) For GC-3: Press "Calibrate #3" button on GC calibrator

5.6 Gas Chromatograph GC-3 Startup

- 5.6.1 Verify that "A" is displayed on the integrator screen as the channel (CH) value.
- 5.6.2 Press the INJECT A key on the integrator to start continuous operation of GC3. Monitor the RUNTIME display on the integrator. It should zero and start counting up in hundredths of seconds.
- 5.6.3 Verify that the RUNTIME re-zeros at 2.50 minutes and continues to count up on a 2.50 minute cycle. The integrator should display "RUN" from 0.00-1.00 minutes, "REPORT" from 1.14-1.40 minutes, and "READY" from 1.41-2.50 minutes (these values can vary).
- 5.6.4 When Channel "A" re-zeros, verify that the indicator light on the GC-3 changes from LOAD to INJECT.

- 5.6.5 Verify that the RUNTIME re-zeros at 2.50 minutes and continues to count up on a 2.50-minute cycle. This completes startup of the GC-3 chromatograph.
- 5.7 DACS Procedure (Kenetics Computer) for Shutdown/Restart: Blackadder and GC-3.
 - 5.7.1 Return to the Kenetics computer in DACS.
 - 5.7.2 Highlight the GC3F window. From the [C:\GC3] prompt, enter a command from the keyboard in the following format for the current file:

TYPE 30604.PR3 <ENTER>,

where the first digit (3) represents the year, the next two digits represent the month (and therefore can range from 01 to 12), and the next two digits (04) represent the day of the month.
 - 5.7.3 A listing of files in the *.PR3 file will scroll down the screen. The file number consists of the first five digits of the *.PR3 file plus 3 more digits ranging from 001 to 999 with a *.RA3 extension. For example, the 510th filename would be 30604510.RA3.
 - 5.7.4 For future reference, copy the name of the last file in the *.RA3 list. Note the 3 digits which follow the first 5 digits of the last listed *.RA3 file.
 - 5.7.5 To restart transfer of GC3 data from the blackadder to the Kenetics computer, enter at the [C:\GC3] prompt:

GC3F n j where n=1 and j is the number plus 1 noted in Step 5.9.4. For example: [C:\GC3] GC3F 1 511 <ENTER>.

After 2.5 minutes, transfers from the blackadder to the Kenetics computer should begin.

Note: *n may not equal 1 if the first file number to be transferred from the blackadder is greater than 1.*

B.6 KENETICS COMPUTER RESTART FROM LOCKUP CONDITION

SECTION 3.0

6.1 Kenetics Computer Shutdown/Restart

- 6.1.1 In the event of a Kenetics lockup, turn off the power to the computer (switch on front).
- 6.1.2 After waiting at least 5 seconds, turn the computer power back on.
- 6.1.3 After approximately 5 minutes, the system will complete all required initialization. There should be ~~many~~ open windows on the screen.
- 6.1.4 Ensure that the PROCOMM window has a blue background and that it is counting scans on-screen. This will require bringing up PROCOMM and making it the active window. If it is solid black with either an error message or a blinking cursor, then go back and repeat steps starting at Step 6.1.1 (power off).

Note: PROCOMM utilizes interrupts and may appear to cause the system to halt or lock up when it is made active. It should clear up and respond again within 30-60 seconds. Don't give up unless it has remained locked up for at least 5 minutes or so. If it does, go back to Step 6.1.1 again.

- 6.1.5 Minimize ALL windows except for the following:

PHOTO
 GC3F
 DACS
 PROCOMM
 FTIR (you may also see this at FTANB, depending on timing)
 CLOCK (optional)

Note: These also may have an .EXE extension attached to them (i.e., GC3F.EXE or DACS.EXE).

Note: This procedure covers the restart of GC3. The FTIR data transfers should begin automatically following a Kenetics computer restart. This involves the programs PROCOMM and FTIR. The DACS.EXE program should also restart automatically. If it is not listed in the "Windows List," then double-click (with the left trackball button) on the DACS icon on the top of the screen. DACS test engineers should be able to confirm FTIR data transfers shortly following a computer restart.

- 6.1.6 Open an OS/2 window (double-click on the "OS/W Window" icon at the top of the screen). You will see an OS/2 command line prompt. At the prompt, type the following:

[C:]ftp blackadder <ENTER>

- 6.1.7 The system will display a couple of lines of information, followed by a prompt. Type "anonymous" after the prompt and press ENTER (as below):

Name (blackadder): anonymous <ENTER>

- 6.1.8 You will then be logged in. An "ftp" prompt will eventually be displayed. To check the GC3 directory on the GMS-2 computer on the farm, type the following at the prompt:

ftp> dir gc3 <ENTER>

- 6.1.9 You will now see a directory of files for GC-3 that are still out on the farm and have not been brought over to DACS. If there are not files listed or if the only file is 3A.TMP, then nothing more can be done from DACS. A farm entry is required to restart the integrators and network communications. Proceed to 5.0, Section 2 - Complete System Reboot, of this procedure.
- 6.1.10 If there are files on the farm, you will then need to determine the first PAIR of files that are out there. The files should be in pairs of the same filename with the extensions .RAW and .RPT.

Note: The files may or may not be in order.

You may find a lone file; if so, write down the filename for later use by the GMS team. The filename of the lowest numerical pair of files (numbered in sequence, they are of the form GH003456.RAW, etc.) is extremely important for restarting GC-3 updates. Write down the number from the first pair (i.e., if the pair is GH004321.RAW and GH004321.RPT, the number is 4321).

Number _____.

Note: You can use the CTRL-S and CTRL-Q key combinations to stop (CTRL-S) and restart (CTRL-Q) the directory listing if there are too many files to view at one time.

- 6.1.11 Type "quit" at the prompt (as below):

ftp> quit <ENTER>

- 6.1.12 This will place you back at the [C:\] prompt. Now type the following:

[C:\]type c:\gc3*.pr3 <ENTER>

This will give you the latest listing for GC-3 data brought over to DACS. When the listing is done, the last line will be of the form:

09/04/93 04:21:44 553 0.370000 46.818778 30904098.ra3,

and you will be back at the [C:\] prompt. The filename with the .ra3 extension (30904098 in the above case) is important here for restarting GC-3. The leading "3" stands for 1993, the next two numbers are the month (09 in this example), the next two numbers are the day (04 in this case), and the last three digits are the file number for the day (they start at "001" after midnight).

- 6.1.13 Record the last three numbers here (without leading zeros; in our example, it would be 98).

_____.

- 6.1.14 This will place you back at the [C:\] prompt.

- 6.1.15 At the OS/2 command prompt, type "exit," as shown below:

[C:\]exit <ENTER>

This will cause the OS/2 window to disappear.

- 6.1.16 Again, move the mouse pointer into the center of the gc3f window and click the left button. This should cause the gc3f window title bar to become highlighted.

- 6.1.17 This step will restart the GC-3 updates to DACS. You will need the numbers from Steps 6.1.10 and 6.1.13 for this step. At the [C:\GC3] prompt, type the following:

[C:\GC3]gc3f [Step 6.1.10 number][Step 6.1.13 number + 1] <ENTER>

In other words, if the number from Step 6.1.10 is 4321 and the number from Step 6.1.13 is 98, then the line would read:

[C:\GC3]gc3f 4321 99 <ENTER>

Make sure you type gc3f , NOT gc3. Make sure there is a space after gc3f and another space between the two numbers. Also make sure the last number is the number from Step 6.1.13 PLUS ONE.

- 6.1.22 This should restore the GC-3 updates to DACS. The program will wait 3 minutes before starting ANY updates, however, so be patient!
- 6.1.23 After ensuring that updates are now underway for GC-3, make gc3f the active window. Turn the monitor brightness down before leaving DACS.

B.7 FTIR RESTART FROM DACS

SECTION 4.0

7.1 Initial Assessment

- 7.1.1 Determine if FTIR is updating to DACS and operating properly by performing the following steps.
- 7.1.2 Make the PROCOMM window active.
- 7.1.3 Screen should display "Scanning xx of 64" with the xx value updating approximately every 10 seconds. If that is not occurring or if the message "error while collecting data" is observed, then a warm reboot of the FTIR as outlined in the next section may be required.

7.2 Warm Reboot of the FTIR from DACS

- 7.2.1 Make FTIR/FTANB the active window.
- 7.2.2 Break out of the program (CTRL-C).
- 7.2.3 Make PROCOMM the active window.
- 7.2.4 Press "CTRL" and "\ " simultaneously to halt the scanning process.
- 7.2.5 Press the <ENTER> key. This will return the "%" prompt.
- 7.2.6 At the "%" prompt, type "sync" (in lower case), followed by the <ENTER> key. After a few seconds, the "%" prompt will return.
- 7.2.7 Repeat the previous step one more time as follows: At the "%" prompt, type "sync" (in lower case), followed by the <ENTER> key. After a few seconds, the "%" prompt will return.
- 7.2.8 At the "%" prompt, type "reboot" (in lower case), followed by the <ENTER> key.
- 7.2.9 It will take approximately 2 to 3 minutes to reboot the FTIR, so be patient; you will be looking at a blank screen. After boot is complete, PROCOMM will display the "User:" prompt.
- 7.2.10 At the "User:" prompt, type "IR4" (upper case), followed by the <ENTER> key, PROCOMM will return with the familiar "%" prompt.
- 7.2.11 At the "%" prompt, type "spftir" (in lower case), followed by the <ENTER> key. The display should now indicate that the FTIR is scanning "scanning xx of 64."
- 7.2.12 After the FTIR has completed 64 scans and computed the absorbance, start FTIR by double-clicking the "FTIR" icon.

**THIS PAGE INTENTIONALLY
LEFT BLANK**

APPENDIX C - SYSTEM CODE LISTINGS

C.1 "C" COMPILER

The programs listed in Appendix C.2 through C.7 use the following

IBM C SET++ V2.1
for OS/2

1. IBM C/C++ Tools V2.01
2. IBM Workframe V2.1
3. IBM Developer's Toolkit for OS/2 V2.1

C.2 SPFTIR - FTIR Spectrometer System Control

SPFTIR consists of one file with four functions: main, fname, ita, and scan (OS/2 ANSI standard C code).

SOURCE LISTING : SPFTIR Ver. 1.0

```

/*FTIR.C PROGRAM TO RUN THE FTS-25 V1.0*/
/*CLIFFORD HEWITT*/
/*LOS ALAMOS NATIONAL LAB*/
/*PO BOX 1663*/
/*MS J565*/
/*LOS ALAMOS NM 87544*/
/*VOICE (505)667-7100*/
/*FAX (505)665-3504*/
/*VERSION 1.0 JULY 13 1992*/

/*MSC INCLUDE FILES*/
#include <stdio.h> /*NEEDED BY ALL ROUTINES*/
#include <time.h> /*NEEDED BY FNAME ROUTINE*/
#include <string.h> /*NEEDED BY FNAME ROUTINE*/
#include <stdlib.h> /*NEEDED BY FNAME ROUTINE*/
#include <signal.h>

int kap=0;
int sam_num=0;

main()
{
    int fil_stat,sig;
    char fil_name[9],log_name[9],fil[5],loger[5],new_num[4];
    FILE *fp;

    fp=(fopen("scanlog","r"));
    fscanf(fp,"%s",log_name);
    fname(fil_name,&sam_num);
    fclose(fp);

    fil[0]=fil_name[1];
    fil[1]=fil_name[2];
    fil[2]=fil_name[3];
    fil[3]=fil_name[4];
    fil[4]=0;

    loger[0]=log_name[1];
    loger[1]=log_name[2];
    loger[2]=log_name[3];
    loger[3]=log_name[4];
    loger[4]=0;

    if(strcmp(loger,fil)==0)
    {
        new_num[0]=log_name[5];
        new_num[1]=log_name[6];
        new_num[2]=log_name[7];
    }
}

```

```

        new_num[3]=0;

        sam_num=atoi(new_num);
        sam_num++;
    }

    else
        sam_num=0;

    while(sig!=27)
    {

        signal(sig,SIG_DFL);

        /*GET FILE NAME*/
        fil_stat=fname(fil_name);

        if (fil_stat==1)
            fil_stat=fname(fil_name);

        printf("\nFIL_STAT:\t%d\tFILE_NAME:\t%s",fil_stat,fil_name);

        /*TAKE FT-IR SCAN*/
        scan(fil_name);

        fp=fopen("scanlog","w");
        fprintf(fp,"%s",fil_name);
        fclose(fp);

        /*INCREMENT DATA SET NUMBER*/
        sam_num++;

    }
}

/*FNAME.C TAKES DATE INFORMATION AND CREATES FILE NAME*/
fname(char fil_name[])
{
    int j;
    time_t now;
    char temp_time[30],time_day[5],temp[4],mon[4];

    time(&now);

    temp[3]=0;

    strcpy(temp_time,ctime(&now));

    time_day[0]=temp_time[11];
    time_day[1]=temp_time[12];
    time_day[2]=temp_time[14];
    time_day[3]=temp_time[15];
    time_day[4]=0;

    j=atoi(time_day);

    if(sam_num==0)
        kap=j;

    if(j<kap)
    {
        sam_num=0;
        return(1);
    }

    ita(temp);

    if(sam_num<=9)
    {
        temp[2]=temp[0];
        temp[0]='0';
        temp[1]='0';
    }
    else if((sam_num>=10)&&(sam_num<=99))
    {
        temp[2]=temp[1];

```

```

        temp[1]=temp[0];
        temp[0]='0';
    }
    else if(sam_num>999)
    {
        printf("\nReset Data Sample Number");
        printf("\nChanged To Alternate Dir");
    }

    mon[0]=temp_time[4];
    mon[1]=temp_time[5];
    mon[2]=temp_time[6];
    mon[3]=0;

    if(strcmp("Jan",mon)==0)
    {
        fil_name[0]=temp_time[23];
        fil_name[1]=temp_time[4];
        fil_name[2]=temp_time[5];
        fil_name[3]=temp_time[8];
        fil_name[4]=temp_time[9];
        fil_name[5]=temp[0];
        fil_name[6]=temp[1];
        fil_name[7]=temp[2];
        fil_name[8]=0;
    }

    else
    {
        fil_name[0]=temp_time[23];
        fil_name[1]=temp_time[4];
        fil_name[2]=temp_time[6];
        fil_name[3]=temp_time[8];
        fil_name[4]=temp_time[9];
        fil_name[5]=temp[0];
        fil_name[6]=temp[1];
        fil_name[7]=temp[2];
        fil_name[8]=0;
    }

    return(0);
}

/*ITA.C CONVERTS INTEGERS TO ASCII*/
ita(char temp[])
{
    FILE *fp;

    fp=fopen("temp.tem","w+");

    fprintf(fp,"%d",sam_num);
    rewind(fp);
    fscanf(fp,"%s",temp);
    rewind(fp);

    fclose(fp);

    return(0);
}

/*dummy function for scan*/
scan(char fil_name[])
{
    char ftirscan[14];

    ftirscan[13]=0;

    strcpy(ftirscan,"scan ");
    strcat(ftirscan,fil_name);
    system(ftirscan);
    return(0);
}

```


C.3 FTRANS - FTIR Spectrometer System File Transfer

FTRANS consists of five files (FTANS, DET_FILE, GET_FILE, TRANS_F, CLEAN) with six functions (OS/2 ANSI standard C code).

SOURCE LISTING : FTANS Ver. 1.0

```
#define INCL_DOSPROCESS

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <os2.h>

int det_file(int);
void get_file();
void trans_file(char[]);
int clean_file(char[]);
void tim_ge(int);

main()
{
    /*local variables*/
    int i,j,k,file_det;
    ULONG delay;
    char stop;
    char data_dir[75],data_dir1[75],fname[12],dir[75],dir2[75];
    char sfilename[12],dir_name[12],comd[75],comd1[75];
    char comd2[75];
    size_t lenght;
    FILE *fp;
    FILE *epm file;
    APIRET RC;

    /*Get paths for FT-IR and IBM directories*/
    fp=fopen("paths","r");
    fscanf(fp,"%s",dir);
    fscanf(fp,"%s",dir2);
    fscanf(fp,"%ld",&delay);
    fclose(fp);

    stop=1;
    printf("\t\t*****FTRANS PROGRAM*****\n");
    printf("\t\t*****VERSION 1.00*****\n");

    /*main program loop*/
    while(stop!=0)
    {
        /*Check to see if new FT-IR file is available*/
        /*If file_det equals one get new file from FT-IR computer*/
        while(file_det!=1)
            file_det=det_file(delay);

        /*Turn off FT-IR file detect flag*/
        file_det=0;

        /*Get file from FT-IR computer*/
        get_file();

        fp=fopen("getlog","r");
        fscanf(fp,"%s",fname);
        strcpy(sfilename,fname);
        strcat(fname,".dt");
        sfilename[5]=0;
        fclose(fp);
        if(strcmp(dir_name,sfilename)!=0)
        {
            strcpy(comd1,"md ");
            strcat(comd1,dir);
            strcat(comd1,sfilename);
            fp=fopen("mkfile.cmd","w");
            fprintf(fp,"%s",comd1);
            fclose(fp);
        }
    }
}
```

```

        system("mkfile");
        strcpy(dir_name,sfilename);
    }

    strcpy(data_dir,dir);
    strcat(data_dir,dir_name);
    strcpy(data_dir1,data_dir);
    strcat(data_dir1,"\\");

    /*Transfer file to analysis dirrectory*/
    trans_file(data_dir);

    /*Check to make sure files were tranfered correctly and*/
    /* then delete file from local directory and FT-IR computer*/
    j=clean_file(data_dir1);

    /* If j does not equal zero file file was either not transfered*/
    /*from the FT-IR computer correctly or to the analysis directory*/
    /*correctly try once more.*/
    if(j!=0)
    {
        get_file();
        trans_file(data_dir);
        j=clean_file(data_dir1);
    }

    /*Open and write out the path and filename in file ftanb.epm*/
    epm_file=fopen("ftanb.epm","w");
    fprintf(epm_file,"%s\n%s",data_dir1,fname);
    fclose(fp);
    fclose(epm_file);

    /*Make directory listing*/
    strcpy(cmd,"dir /n ");
    strcat(cmd,data_dir1);
    strcat(cmd,fname);
    strcat(cmd," >ftanb.lst");
    lenght=strlen(cmd);
    cmd[lenght+1]=0;
    fp=fopen("lst.cmd","w");
    fprintf(fp,"%s",cmd);
    fclose(fp);
    system("lst");

    /*Run FTIR analysis program*/
    system("ftanb");

    /*Get FTIR analysis*/
    if((fp=fopen("ftanb.dat","r"))!=0)
    {
        fclose(fp);
        fp=fopen("prn_make.cmd","w");
        strcpy(cmd2,"type ftanb.dat >> ");
        strcat(cmd2,dir2);
        strcat(cmd2,dir_name);
        strcat(cmd2,".prn");
        lenght=strlen(cmd2);
        cmd2[lenght+1]=0;
        fprintf(fp,"%s\n",cmd2);
        fclose(fp);
        system("prn_make");
        fp=fopen("dacs_te.cmd","w");
        fprintf(fp,"copy ftanb.dat temp.dat\n");
        fclose(fp);
        while((fp=fopen("temp.dat","r"))==0)
        {
            RC=DosSleep(5000);

            if(RC != 0)
                printf("DosSleep ERROR %ld",RC);
            system("dacs_te");
        }
        fclose(fp);
    }
}

```

SOURCE LISTING : DET_FILE Ver. 1.0

```

#define INCL_DOSPROCESS

/*Subroutine that detects new files as they are created by the */
/*FT-IR computer*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <signal.h>
#include <math.h>
#include <time.h>
#include <os2.h>

void tim_get(int);
void handler(int);

int det_file(ULONG delay)
{
    /*local variables*/
    int i;
    char scan_name[9],get_name[9];
    FILE *slget;
    FILE *gel;
    FILE *sal;
    APIRET RC;

    printf("\nDET_FILE");

    if (signal(SIGINT, handler)==SIG_ERR)
    {
        perror("Could not set SIGINT");
        exit(-1);
    }

    /*Open FT-IR input command file*/
    slget=fopen("ftir.inp","w");

    /*Put the following commands in the*/
    /*FT-IR command file*/

    /*User login on the FT-IR computer*/
    fprintf(slget,"IR4\n");
    /*Set FTP to ASCII mode*/
    fprintf(slget,"ascii\n");
    /*Change directory comand and*/
    /*path on the FT-IR computer*/

    /*Get the scanlog file*/
    fprintf(slget,"get scanlog\n");
    /*Quit FTP*/
    fprintf(slget,"quit\n");

    /*Close FT-IR input command file*/
    fclose(slget);

    /*FTP to FT-IR computer using the input*/
    /*from the FT-IR input command file*/
    system("ftir");

    /*Open scanlog file*/
    /*The scanlog file contains the most resent */
    /*file aquired by the FT-IR computer*/
    if((sal=fopen("scanlog","r"))!=NULL)
    {
        /*Open getlog file*/
        /*The getlog file contains the most resent */
        /*file aquired from the FT-IR computer*/
        gel=fopen("getlog","r");

        /*Read in the file name from the scanlog file*/
        fscanf(sal,"%s",scan_name);

        /* Read in the file name from the getlog file*/
        fscanf(gel,"%s",get_name);

        printf("\nSCAN_NAME:\t%s\tGET_NAME:\t%s",scan_name,get_name);
    }
}

```

```

/*Close the getlog and the scanlog files*/
fclose(gel);
fclose(sal);

/*Compare the file names from the getlog and scanlog files*/
/*If they are change the name in the getlog file to the name in*/
/*scanlog file and set the file_det flag */
if((strcmp(scan_name,get_name))!=0)
{
    gel=fopen("getlog","w");
    fprintf(gel,"%s",scan_name);
    fclose(gel);
    return(1);
}
else
{
    RC=DosSleep(delay);

    if(RC != 0)
        printf("DosSleep ERROR %ld",RC);

    return(0);
}
}

void tim_get(int delay)
{
    static flag=0;
    static long mastime;
    time_t now;
    long time1;

    if(flag==0)
    {
        time1=time(&now);
        mastime=time1;
    }

    mastime=mastime+delay;
    time1=time(&now);

    while((mastime-time1)>0)
    {
        time1=time(&now);

        /*printf("%ld\n",(mastime-time1));*/
    }

    if(flag==0)
        flag=1;
}

void handler(int sig)
{
    printf("EXITING PROGRAM");
    exit(0);
}

```

SOURCE LISTING : GET_FILE Ver. 1.0

```

/*Subroutine that gets new files as they are created by the */
/*FT-IR computer*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void get_file()
{
    /*local variables*/
    char get_name[9],comd[75];
    FILE *slget;
    FILE *get;

    printf("\nGET_FILE");

    /*Open FT-IR input command file*/
    slget=fopen("ftir.inp","w");

    /*create the command to get the file from the FT-IR*/
    if((gel=fopen("getlog","r"))==0)
        printf("\ngetlog file is not open\n");
    fscanf(gel,"%s",get_name);
    strcpy(comd,"get ");
    strcat(comd,get_name);
    strcat(comd,".dt");
    comd[15]=0;
    fclose(gel);
    /*Put the following commands in the*/
    /*FT-IR command file*/

    /*User login on the FT-IR computer*/
    fprintf(slget,"IR4\n");
    /*Set FTP to binary mode*/
    fprintf(slget,"binary\n");
    /*Change directory comand and*/
    /*path on the FT-IR computer*/

    /*Get command*/
    fprintf(slget,"%s\n",comd);
    /*Quit FTP*/
    fprintf(slget,"quit\n");

    /*Close FT-IR input command file*/
    fclose(slget);

    /*FTP to FT-IR computer using the input*/
    /*from the FT-IR input commnd file*/
    system("ftir");
}

```

}

SOURCE LISTING : TRANS_F Ver. 1.0

```

/*Subroutine that transfers the data into the data analysis directory*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void trans_file(char data_dir[])
{
    /*local variables*/
    char get_name[9],comd[75],comd1[75],comd2[75];
    size_t lenght;

    FILE *gel;
    FILE *tcopy;

    printf("\nTRANS_FILE");

    /*Open getlog file read in getlog file name*/
    /*close getlog file*/
    gel=fopen("getlog","r");
    fscanf(gel,"%s",get_name);
    fclose(gel);

    /*Create command to copy FT-IR file to the data*/
    /*analysis directory*/
    strcpy(comd,"copy ");
    strcat(comd,get_name);
    strcat(comd,".dt ");
    strcat(comd,data_dir);
    lenght=strlen(comd);
    comd[lenght+1]=0;

    /*Create command to copy getlog file to the data*/
    /*analysis directory*/
    strcpy(comd2,"copy getlog ");
    strcat(comd2,data_dir);
    lenght=strlen(comd2);
    comd2[lenght+1]=0;

    /*Copy files to data analysis directory*/
    tcopy=fopen("trans_cp.cmd","w");
    fprintf(tcopy,"%s\n",comd);
    fprintf(tcopy,"%s\n",comd2);
    fclose(tcopy);
    system("trans_cp");
}

```

SOURCE LISTING : CLEAN Ver. 1.0

```

/*Subroutine that checks that all data files have been transfered*/
/*correctly and then cleans up the disk*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int clean_file(char data_dir1[])
{
    /*local variables*/
    static int clean_count=0;
    int i=0;
    size_t lenght;
    char get_name[9],comd[75],comd1[75],gfile_name[75];
    FILE *slget;
    FILE *gel;
    FILE *gfile;
    FILE *cle;

    printf("\nCLEAN_FILE");

    /*Create check file name and path*/
    gel=fopen("getlog","r");
    fscanf(gel,"%s",get_name);
    strcpy(gfile_name,data_dir1);
    strcat(gfile_name,get_name);
    strcat(gfile_name,".dt");
    lenght=strlen(gfile_name);
    gfile_name[lenght+1]=0;
    fclose(gel);

    /*Check for file in the data analysis directory*/
    /*If there proceed with clean-up*/
    if((gfile=fopen(gfile_name,"r"))!=NULL)
    {

        fclose(gfile);

        /*Create command to clean-up FT-IR disk*/
        strcpy(comd,"delete ");
        strcat(comd,get_name);
        strcat(comd,".dt");
        comd[18]=0;

        /*Create command to clean-up IBM disk*/
        strcpy(comd1,"del ");
        strcat(comd1,get_name);
        strcat(comd1,".dt");
        comd1[18]=0;

        /*Open FT-IR input command file*/
        slget=fopen("ftir.inp","w");

        /*Put the following commands in the*/
        /*FT-IR command file*/

        /*User login on the FT-IR computer*/
        fprintf(slget,"IR4\n");
        /*Set FTP to ASCII mode*/
        fprintf(slget,"ascii\n");
        /*Change directory comand and*/
        /*path on the FT-IR computer*/

        /*Clean-up command*/
        fprintf(slget,"%s\n",comd);
        /*Quit FTP*/
        fprintf(slget,"quit\n");

        /*Open FT-IR input command file*/
        fclose(slget);

        /*Clean-up FT-IR disk*/
        system("ftir.cmd");

        /*Clean-up IBM disk*/
        cle=fopen("clef.cmd","w");
        fprintf(cle,"%s\n",comd1);
    }
}

```

```
fclose(cle);
system("clef");

/*Reset clean_count*/
clean_count=0;

/*Return to main*/
return(0);
}

else
{
    clean_count++;
    if(clean_count>=9)
    {
        printf("\n\n\n\tPROBLEM IN GETTING SCAN_FILE");
        /*Exit program*/
        exit(0);
    }

    /*Return to main with clean-up error*/
    return(1);
}
}
```


C.4 FTAN-B - Infrared Data Analysis Codes

FTAN-B consists of three files (FTANB, HEXTOLO, HEXTODBL) with three functions (OS/2 ANSI standard C code).

SOURCE LISTING : FTANB Ver. 2.02

```

/*****
/*FTANB Version 2.02*/
*****/

#define INCL_DOSPROCESS
#define INCL_DOSFILEMGR

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <os2.h>

long hextolo(char[]);
double hextodbl(char[]);

main(int argc, char *argv[], char *envp[])
{

    int x_exp,y_exp,quadnum;
    long i,j,k,l,sp,st,index;
    long bast1[20],basp1[20],bast2[20],basp2[20];
    long run[100],pkst[20],pksp[20];
    long lenght,ptl,d,di;
    long head_stl,data_stl,head_sts;
    long data_size,n_points,f_point;
    long n_of_peaks;
    ULONG fil,fibs,act;
    double peakst[20],peaksp[20],peakcon[20];
    double backst1[20],backsp1[20],backst2[20],backsp2[20];
    double back1[20],back2[20],slope[20],rise[20],area[20];
    double y_data,k0,k1,k2,tarea[20];
    double cal,nh3area,nh3con,n20area,n20con,ch4con,ch4area,temp,y,m;
    double p_sep,fd_point,fac,xfac,yfac,yvalue;
    char head_stc[5],fname[100];
    char data[100000],dir[100];
    char text[2500];
    char db[8],hb[4];
    char dat[20],name[12];
    FILE *fp;
    APIRET rc;
    HFILE fh;
    FILESTATUS3 fib;
    struct fib;

    head_stc[4]=0;

    for(i=0;i<=2499;i++)
        text[i]=0;

    fil=1;
    fibs=sizeof(FILESTATUS3);

    fp=fopen("ftanb.epm","r");
    fscanf(fp,"%s",dir);
    fscanf(fp,"%s",name);
    strcpy(fname,dir);
    strcat(fname,name);
    printf("%s\n",fname);
    rc=DosOpen(fname,

                                &fh,&act,
                                0,
                                FILE_NORMAL,
                                FILE_OPEN,
                                OPEN_ACCESS_READONLY | OPEN_SHARE_DENYNONE,
                                NULL);

```

```

if(rc != 0)
{
    printf("DosOpen ERROR %ld\n in line 44",rc);
    exit(-1);
}

rc=DosQueryFileInfo(fh,fil,&fib,fibs);
if(rc != 0)
    printf("DosQuery ERROR %ld in line 47\n",rc);

DosClose(fh);

fp=fopen(fname,"rb");

for(lenght=0;lenght<=(fib.cbFile-1);lenght++)
    fscanf(fp,"%c",&data[lenght]);

fclose(fp);

fp=fopen("peak.tab","r");
fscanf(fp,"%ld",&n_of_peaks);
for(i=0;i<=(n_of_peaks-1);i++)
{
    fscanf(fp,"%lf\n%lf",&backst1[i],&backsp1[i]);
    fscanf(fp,"%lf\n%lf",&backst2[i],&backsp2[i]);
    fscanf(fp,"%lf\n%lf",&peakst[i],&peaksp[i]);
}
fclose(fp);

hb[0]=data[80];
hb[1]=data[81];
hb[2]=data[82];
hb[3]=data[83];

head_stl=hextolo(hb);

hb[0]=data[84];
hb[1]=data[85];
hb[2]=data[86];
hb[3]=data[87];

head_sts=hextolo(hb);

hb[0]=data[88];
hb[1]=data[89];
hb[2]=data[90];
hb[3]=data[91];

data_stl=hextolo(hb);

hb[0]=data[92];
hb[1]=data[93];
hb[2]=data[94];
hb[3]=data[95];

data_size=hextolo(hb);

hb[0]=data[96];
hb[1]=data[97];
hb[2]=data[98];
hb[3]=data[99];

n_points=hextolo(hb);

x_exp=data[142];
y_exp=data[143];

db[0]=data[102];
db[1]=data[103];
db[2]=data[104];
db[3]=data[105];
db[4]=data[106];
db[5]=data[107];
db[6]=data[108];
db[7]=data[109];

```

```

p_sep=hextodbl(db);

hb[0]=data[110];
hb[1]=data[111];
hb[2]=data[112];
hb[3]=data[113];

f_point=hextolo(hb);

fd_point=p_sep*f_point;

di=data_stl+16;

xfac=1.0;

if(x_exp>128)
{
    x_exp=256-x_exp;
    for(i=0;i<(long)x_exp;i++)
        xfac=xfac/2.0;
}
else
{
    for(i=0;i<(long)x_exp;i++)
        xfac=xfac*2.0;
}

yfac=1.0;

if(y_exp>128)
{
    y_exp=256-y_exp;
    for(i=0;i<(long)y_exp;i++)
        yfac=yfac/2.0;
}
else
{
    for(i=0;i<(long)y_exp;i++)
        yfac=yfac*2.0;
}

fac=1.0;
for(i=0;i<=30;i++)
    fac=fac/2.0;

/*    printf("TEXT HEADER START: %ld\n",head_stl);
printf("NUMBER OF BYTES IN TEXT HEADER: %ld\n",head_sts);
printf("DATA HEADER START: %ld\n",data_stl);
printf("DATA SEGMENT SIZE: %ld\n",data_size);
printf("NUMBER OF DATA POINTS: %ld\n",n_points);
printf("RAW X EXP: %d\n",x_exp);
printf("RAW Y EXP: %d\n",y_exp);
printf("DATA POINT SEPERATION: %lf\n",p_sep);
printf("FIRST DATA POINT: %lf\n",fd_point);
printf("MULTI FACTOR: %e\n",fac);
printf("X REAL FAC: %e\n",xfac);
printf("Y REAL FAC: %e\n",yfac);*/

lenght=head_stl;
j=0;
while(lenght<=(head_stl+head_sts))
{
    text[j]=data[lenght];
    j++;
    lenght++;
}

for(l=0;l<=j;l++)
{
    if(text[l]==67)
    {
        if(text[l+1]==82)
        {
            if(text[l+2]==69)
            {
                if(text[l+3]==65)
                {
                    if(text[l+4]==84)

```

```

        {
            if(text[l+5]==69)
            {
                if(text[l+6]==68)
                {
                    dat[0]=text[l+12];
                    dat[1]=text[l+13];
                    dat[2]=text[l+14];
                    dat[3]=text[l+15];
                    dat[4]=text[l+16];
                    dat[5]=text[l+17];
                    dat[6]=text[l+18];
                    dat[7]=text[l+30];
                    dat[8]=text[l+31];
                    dat[9]=text[l+18];
                    dat[10]=text[l+19];
                    dat[11]=text[l+20];
                    dat[12]=text[l+21];
                    dat[13]=text[l+22];
                    dat[14]=text[l+23];
                    dat[15]=text[l+24];
                    dat[16]=text[l+25];
                    dat[17]=text[l+26];
                    dat[18]=0;
                }
            }
        }
    }
}

/*    printf("%s\n",dat);*/

/*calculate the point values of the wave numbers*/
for(i=0;i<=n_of_peaks-1;i++)
{
    temp=(backst1[i]/p_sep)-(fd_point/p_sep);
    y=modf(temp,&m);
    if (y>.5)
        m=m+1.0;
    bast1[i]=(long)m;

    temp=(backsp1[i]/p_sep)-(fd_point/p_sep);
    y=modf(temp,&m);
    if (y>.5)
        m=m+1;
    basp1[i]=(long)m;

    temp=(backst2[i]/p_sep)-(fd_point/p_sep);
    y=modf(temp,&m);
    if (y>.5)
        m=m+1;
    bast2[i]=(long)m;

    temp=(backsp2[i]/p_sep)-(fd_point/p_sep);
    y=modf(temp,&m);
    if (y>.5)
        m=m+1;
    basp2[i]=(long)m;

    temp=(peakst[i]/p_sep)-(fd_point/p_sep);
    y=modf(temp,&m);
    if (y>.5)
        m=m+1;
    pkst[i]=(long)m;

    temp=(peaksp[i]/p_sep)-(fd_point/p_sep);
    y=modf(temp,&m);
    if (y>.5)
        m=m+1;
    pksp[i]=(long)m;
}

/*    for(i=0;i<=(n_of_peaks-1);i++)
{
    printf("PEAK[%d]: %d\t%d\n",i+1,pkst[i],pksp[i]);
}

```

```

printf("BACK1[%d]: %d\t%d\n", i+1, bast1[i], basp1[i]);
printf("BACK2[%d]: %d\t%d\n", i+1, bast2[i], basp2[i]);
}*/

for(j=0; j<=(n_of_peaks-1); j++)
{
    back1[j]=0.0;
    for(i=0; i<=(basp1[j]-bast1[j]); i++)
    {
        index=di+((bast1[j]+i)*4);

        for(l=0; l<=3; l++, index++)
            hb[l]=data[index];

        y_data=hextolo(hb);
        y_data=y_data*fac*yfac;

        back1[j]=back1[j]+y_data;
        /*printf("%lf\t%lf\n", back1[j], y_data);*/
    }
    back1[j]=back1[j]/(basp1[j]-bast1[j]+1);

    back2[j]=0.0;
    for(i=0; i<=(basp2[j]-bast2[j]); i++)
    {
        index=di+((bast2[j]+i)*4);

        for(l=0; l<=3; l++, index++)
            hb[l]=data[index];

        y_data=hextolo(hb);
        y_data=y_data*fac*yfac;

        back2[j]=back2[j]+y_data;
        /*printf("%lf\t%lf\n", back2[j], y_data);*/
    }

    back2[j]=back2[j]/(basp2[j]-bast2[j]+1);

    rise[j]=back2[j]-back1[j];
    st=bast1[j]+((basp1[j]-bast1[j])/2);
    sp=bast2[j]+((basp2[j]-bast2[j])/2);
    run[j]=(sp-st);
    slope[j]=rise[j]/run[j];

    /*printf("Back1[%d]: %lf\n", j+1, back1[j]);
    printf("Back2[%d]: %lf\n", j+1, back2[j]);
    printf("Run[%d]: %ld = %ld - %ld\n", j+1, run[j], sp, st);
    printf("Slope[%d]: %.15lf = %.15lf/%ld\n", j+1, slope[j], rise[j], run[j]);*/

    area[j]=0.0;

    for(i=(pkst[j]-st), l=0; i<=(pksp[j]-st); i++, l++)
    {
        index=di+((pkst[j]+l)*4);

        for(d=0; d<=3; d++, index++)
            hb[d]=data[index];

        y_data=hextolo(hb);
        y_data=y_data*fac*yfac;

        yvalue=slope[j]*(double)i+back1[j];
        area[j]=area[j]+(y_data-yvalue);
        /*printf("YVALUE[%d]: %e\n", i, yvalue[l]);
        printf("Y_DATA: %e\n", pkst[j]+l, y_data);
        printf("ASORBANCE: %e\n", y_data-yvalue);*/
    }

    area[j]=area[j]/(1.0/p_sep);
}

fp=fopen("ftan.cal", "r");

for(i=0; i<=n_of_peaks-1; i++)
{

```

```

        fscanf(fp,"%lf\n%lf\n%lf",&k0,&k1,&k2);
        tarea[i]=area[i];
        peakcon[i]=k2*tarea[i]*tarea[i]+k1*tarea[i]+k0;
/*      printf("K0: %lf\tK1: %lf\tK2: %lf\n",k0,k1,k2);*/
        printf("CON[%ld]: %lf\tAREA[%ld]:%lf\n",i+1,peakcon[i],i+1,area[i]);
    }
    fclose(fp);

    if(area[0]<2.209)
    {
        nh3con=peakcon[0];
        nh3area=area[0];
    }

    if((area[0]>=2.209)&&(area[1]<5.1637))
    {
        nh3con=peakcon[1];
        nh3area=area[1];
    }

    if((area[1]>=5.1637)&&(area[2]<.2700))
    {
        nh3con=peakcon[2];
        nh3area=area[2];
    }

    if(area[2]>=.2700)
    {
        nh3con=peakcon[3];
        nh3area=area[3];
    }

    if(area[4]>=11.31)
    {
        n20con=peakcon[1];
        n20area=area[1];
    }

    if((area[4]>=11.31)&&(area[5]<57.453))
    {
        n20con=peakcon[5];
        n20area=area[5];
    }

    if((area[5]>=57.453)&&(area[6]<64.917))
    {
        n20con=peakcon[6];
        n20area=area[6];
    }

    if(area[6]>=64.917)
    {
        n20con=peakcon[7];
        n20area=area[7];
    }

    name[8]=0;
    fp=fopen("ftanb.dat","w+");
    fprintf(fp,"%s %s %lf %lf %lf %lf %lf %lf\n",
        name,dat,nh3area,n20area,nh3con,n20con,ch4con,ch4area);
    fclose(fp);
    printf("%s\n%s\n",name,dat);
}

```

SOURCE LISTING : HEXTOLO Ver. 1.0

```
#define INCL_DOSPROCESS
#define INCL_DOSFILEMGR

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <os2.h>

long hextolo(char data[])
{
    long hb1,hb2,hb3,hb4,value;

    hb1=data[0];
    hb2=data[1];
    hb3=data[2];
    hb4=data[3];

    hb1=hb1<<24;
    hb2=hb2<<16;
    hb3=hb3<<8;
    value=(hb1+hb2+hb3+hb4);

    return(value);
}
```

SOURCE LISTING : HEXTODBL Ver. 1.0

```

#define INCL_DOSPROCESS
#define INCL_DOSFILEMGR

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <os2.h>

double hextodbl(char data[])
{
    int i,j,k,l;
    int sign,sb,exp,exp1,exp2,exp3;
    int base1,base2,base3,base4,base5,base6,base7;
    int temp;
    double fac,fac1;
    double num,flp[52];
    double num1,num2,num3,num4,num5,num6,num7;

    sign=data[0];

    sign=sign>>7;
    if(sign==1)
        sb=-1;
    if(sign==0)
        sb=1;

    exp1=data[0];
    exp2=data[1];

    exp1=exp1&127;
    exp1=exp1<<4;
    exp2=exp2>>4;
    exp=exp1^exp2;

    exp=exp-1023;

    fac1=1.0;

    if (exp>0)
    {
        for(i=1;i<=exp;i++)
            fac1=fac1*2.0;
    }

    if(exp<0)
    {
        exp1=exp*-1;
        for(i=1;i<=exp1;i++)
            fac1=fac1/2.0;
    }

    base1=data[1];
    base2=data[2];
    base3=data[3];
    base4=data[4];
    base5=data[5];
    base6=data[6];
    base7=data[7];

    fac=2.0;

    temp=base1&8;
    if(temp==8)
        flp[0]=1.0/fac;
    else
        flp[0]=0.0;

    fac=fac*2.0;

    temp=base1&4;
    if(temp==4)
        flp[1]=1.0/fac;
    else
        flp[1]=0.0;

```



```

fac=fac*2.0;

temp=base1&2;
if(temp==2)
    flp[2]=1.0/fac;
else
    flp[2]=0.0;

fac=fac*2.0;

temp=base1&1;
if(temp==1)
    flp[3]=1.0/fac;
else
    flp[3]=0.0;

fac=fac*2.0;

temp=base2&128;
if(temp==128)
    flp[4]=1.0/fac;
else
    flp[4]=0.0;

fac=fac*2.0;

temp=base2&64;
if(temp==64)
    flp[5]=1.0/fac;
else
    flp[5]=0.0;

fac=fac*2.0;

temp=base2&32;
if(temp==32)
    flp[6]=1.0/fac;
else
    flp[6]=0.0;

fac=fac*2.0;

temp=base2&16;
if(temp==16)
    flp[7]=1.0/fac;
else
    flp[7]=0.0;

fac=fac*2.0;

temp=base2&8;
if(temp==8)
    flp[8]=1.0/fac;
else
    flp[8]=0.0;

fac=fac*2.0;

temp=base2&4;
if(temp==4)
    flp[9]=1.0/fac;
else
    flp[9]=0.0;

fac=fac*2.0;

temp=base2&2;
if(temp==2)
    flp[10]=1.0/fac;
else
    flp[10]=0.0;

fac=fac*2.0;

temp=base2&1;
if(temp==1)
    flp[11]=1.0/fac;
else

```

```

    flp[11]=0.0;
    fac=fac*2.0;
    temp=base3&128;
    if(temp==128)
        flp[12]=1.0/fac;
    else
        flp[12]=0.0;
    fac=fac*2.0;
    temp=base3&64;
    if(temp==64)
        flp[13]=1.0/fac;
    else
        flp[13]=0.0;
    fac=fac*2.0;
    temp=base3&32;
    if(temp==32)
        flp[14]=1.0/fac;
    else
        flp[14]=0.0;
    fac=fac*2.0;
    temp=base3&16;
    if(temp==16)
        flp[15]=1.0/fac;
    else
        flp[15]=0.0;
    fac=fac*2.0;
    temp=base3&8;
    if(temp==8)
        flp[16]=1.0/fac;
    else
        flp[16]=0.0;
    fac=fac*2.0;
    temp=base3&4;
    if(temp==4)
        flp[17]=1.0/fac;
    else
        flp[17]=0.0;
    fac=fac*2.0;
    temp=base3&2;
    if(temp==2)
        flp[18]=1.0/fac;
    else
        flp[18]=0.0;
    fac=fac*2.0;
    temp=base3&1;
    if(temp==1)
        flp[19]=1.0/fac;
    else
        flp[19]=0.0;
    fac=fac*2.0;
    temp=base4&128;
    if(temp==128)
        flp[20]=1.0/fac;
    else
        flp[20]=0.0;
    fac=fac*2.0;
    temp=base4&64;
    if(temp==64)

```

```

    flp[21]=1.0/fac;
else
    flp[21]=0.0;

fac=fac*2.0;

temp=base4&32;
if(temp==32)
    flp[22]=1.0/fac;
else
    flp[22]=0.0;

fac=fac*2.0;

temp=base4&16;
if(temp==16)
    flp[23]=1.0/fac;
else
    flp[23]=0.0;

fac=fac*2.0;

temp=base4&8;
if(temp==8)
    flp[24]=1.0/fac;
else
    flp[24]=0.0;

fac=fac*2.0;

temp=base4&4;
if(temp==4)
    flp[25]=1.0/fac;
else
    flp[25]=0.0;

fac=fac*2.0;

temp=base4&2;
if(temp==2)
    flp[26]=1.0/fac;
else
    flp[26]=0.0;

fac=fac*2.0;

temp=base4&1;
if(temp==1)
    flp[27]=1.0/fac;
else
    flp[27]=0.0;

fac=fac*2.0;

temp=base5&128;
if(temp==128)
    flp[28]=1.0/fac;
else
    flp[28]=0.0;

fac=fac*2.0;

temp=base5&64;
if(temp==64)
    flp[29]=1.0/fac;
else
    flp[29]=0.0;

fac=fac*2.0;

temp=base5&32;
if(temp==32)
    flp[30]=1.0/fac;
else
    flp[30]=0.0;

fac=fac*2.0;

```

```

temp=base5&16;
if(temp==16)
    flp[31]=1.0/fac;
else
    flp[31]=0.0;

fac=fac*2.0;

temp=base5&8;
if(temp==8)
    flp[32]=1.0/fac;
else
    flp[32]=0.0;

fac=fac*2.0;

temp=base5&4;
if(temp==4)
    flp[33]=1.0/fac;
else
    flp[33]=0.0;

fac=fac*2.0;

temp=base5&2;
if(temp==2)
    flp[34]=1.0/fac;
else
    flp[34]=0.0;

fac=fac*2.0;

temp=base5&1;
if(temp==1)
    flp[35]=1.0/fac;
else
    flp[35]=0.0;

fac=fac*2.0;

temp=base6&128;
if(temp==128)
    flp[36]=1.0/fac;
else
    flp[36]=0.0;

fac=fac*2.0;

temp=base6&64;
if(temp==64)
    flp[37]=1.0/fac;
else
    flp[37]=0.0;

fac=fac*2.0;

temp=base6&32;
if(temp==32)
    flp[38]=1.0/fac;
else
    flp[38]=0.0;

fac=fac*2.0;

temp=base6&16;
if(temp==16)
    flp[39]=1.0/fac;
else
    flp[39]=0.0;

fac=fac*2.0;

temp=base6&8;
if(temp==8)
    flp[40]=1.0/fac;
else
    flp[40]=0.0;

```

```

fac=fac*2.0;

temp=base6&&4;
if(temp==4)
    flp[41]=1.0/fac;
else
    flp[41]=0.0;

fac=fac*2.0;

temp=base6&&2;
if(temp==2)
    flp[42]=1.0/fac;
else
    flp[42]=0.0;

fac=fac*2.0;

temp=base6&&1;
if(temp==1)
    flp[43]=1.0/fac;
else
    flp[43]=0.0;

fac=fac*2.0;

temp=base7&128;
if(temp==128)
    flp[44]=1.0/fac;
else
    flp[44]=0.0;

fac=fac*2.0;

temp=base7&&64;
if(temp==64)
    flp[45]=1.0/fac;
else
    flp[45]=0.0;

fac=fac*2.0;

temp=base7&&32;
if(temp==32)
    flp[46]=1.0/fac;
else
    flp[46]=0.0;

fac=fac*2.0;

temp=base7&&16;
if(temp==16)
    flp[47]=1.0/fac;
else
    flp[47]=0.0;

fac=fac*2.0;

temp=base7&&8;
if(temp==8)
    flp[48]=1.0/fac;
else
    flp[48]=0.0;

fac=fac*2.0;

temp=base7&&4;
if(temp==4)
    flp[49]=1.0/fac;
else
    flp[49]=0.0;

fac=fac*2.0;

temp=base7&&2;
if(temp==2)
    flp[50]=1.0/fac;
else

```

```

    flp[50]=0.0;
    fac=fac*2.0;
    temp=base7&1;
    if(temp==1)
        flp[51]=1.0/fac;
    else
        flp[51]=0.0;

    num=flp[0]+flp[1]+flp[2]+flp[3]+flp[4]+flp[5]+flp[6]+flp[7]+flp[8];
    num1=flp[9]+flp[10]+flp[11]+flp[12]+flp[13]+flp[14]+flp[15];
    num2=flp[16]+flp[17]+flp[18]+flp[19]+flp[20]+flp[21]+flp[22];
    num3=flp[23]+flp[24]+flp[25]+flp[26]+flp[27]+flp[28]+flp[29];
    num4=flp[30]+flp[31]+flp[32]+flp[33]+flp[34]+flp[35]+flp[36];
    num5=flp[37]+flp[38]+flp[39]+flp[40]+flp[41]+flp[42]+flp[43];
    num6=flp[43]+flp[44]+flp[45]+flp[46]+flp[47]+flp[48]+flp[49];
    num7=flp[50]+flp[51];
    num=num+num1+num2+num3+num4+num5+num6+num7;

    num=sb*(1.0+num)*fac1;

    return(num);
}

```

SOURCE LISTING : RN (CONTINUOUS OPERATION CODE)

```
/*routine that calls ftanb for contiuous operation*/
#include <stdio.h>

main(int argc, char *argv[], char *envp[])
{
    int x=0;

    while(x!=2)
        system("ftanb");
}
```

SOURCE LISTING : OFFLN

```

#define INCL_DOSPROCESS

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <os2.h>

int det_file(int);
void get_file();
void trans_file(char[]);
int clean_file(char[]);
void tim_ge(int);

main(int argc, char *argv[])
{
    /*local variables*/
    int i,j,k,l,sti,spi;
    ULONG delay;
    char stop;
    char data_dir[75],data_dir1[75],fname[12],dir[75],dir2[75];
    char sfilename[12],name[12],comd[75],comd1[75],fname1[75],prnname[15];
    char comd2[75],num[4],filest[12],filespl[12],st[4],sp[4];
    size_t lenght;
    FILE *fp;
    FILE *epm_file;
    APIRET RC;

    strcpy(filest,argv[1]);
    strcpy(filespl,argv[2]);
    strcpy(data_dir1,argv[3]);

    for(i=0,j=5;i<=2;i++,j++)
    {
        st[i]=filest[j];
        sp[i]=filespl[j];
    }
    st[3]=0;
    sp[3]=0;
    sti=atoi(st);
    spi=atoi(sp);
    printf("sti:\t%d\nstp:\t%d\n",sti,spi);
    strcpy(prnname,argv[4]);
    prnname[5]=0;
    strcat(prnname,".prn");
    fp=fopen(prnname,"a+");
    fprintf(fp,"%s\n%s\n",filest,filespl);
    fclose(fp);

    for(l=sti;l<=spi;l++)
    {
        restart;;
        fp=fopen("itoa","w");
        fprintf(fp,"%d",l);
        fclose(fp);

        fp=fopen("itoa","r");
        fscanf(fp,"%s",num);
        fclose(fp);

        if(l<=9)
        {
            num[2]=num[0];
            num[0]=48;
            num[1]=48;
        }
        if(l>9&&l<=99)
        {
            num[2]=num[1];
            num[1]=num[0];
            num[0]=48;
        }
        strcpy(fname,filest);

        for(i=0,j=5;i<=2;i++,j++)

```



```

    fname[j]=num[i];
    printf("%s\n",fname);
    strcpy(fname1,data_dir1);
    strcat(fname1,fname);
    if((fp=fopen(fname1,"r"))==0)
    {
        l++;
        goto restart;
    }

for(i=0;i<=4;i++)
    name[i]=fname[i];
name[5]=0;

/*Open and write out the path and filename in file ftanb.epm*/
epm_file=fopen("ftanb.epm","w");
fprintf(epm_file,"%s\n%s",data_dir1,fname);
fclose(fp);
fclose(epm_file);

/*Make directory listing*/
strcpy(cmd,"dir /n ");
strcat(cmd,data_dir1);
strcat(cmd,fname);
strcat(cmd," >ftanb.lst");
length=strlen(cmd);
cmd[length+1]=0;
fp=fopen("lst.cmd","w");
fprintf(fp,"%s",cmd);
fclose(fp);
system("lst");

/*Run FTIR analysis program*/
system("anal");

/*Get FTIR analysis*/
if((fp=fopen("anal.dat","r"))!=0)
{
    fclose(fp);
    fp=fopen("prn_make.cmd","w");
    strcpy(cmd2,"type ftanb.dat >> ");
    strcat(cmd2,name);
    strcat(cmd2,".prn");
    length=strlen(cmd2);
    cmd2[length+1]=0;
    fprintf(fp,"%s\n",cmd2);
    fclose(fp);
    system("prn_make");
}
}

```

SOURCE LISTING : BATCHA (OFFLINE ANALYSIS CODE)

```

#define INCL_DOSPROCESS
#define INCL_DOSFILEMGR

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <os2.h>

long hextolo(char[]);
double hextodbl(char[]);

main(int argc, char *argv[], char *envp[])
{
    int x_exp,y_exp,fnum;
    long i,j,k,l,sp,st,index;
    long bast1[20],basp1[20],bast2[20],basp2[20];
    long run[100],pkst[20],pksp[20];
    long lenght,ptl,d,di;
    long head_stl,data_stl,head_sts;
    long data_size,n_points,f_point;
    long n_of_peaks;
    ULONG fil,fibs,act;
    double peakst[20],peaksp[20],peakcon[20];
    double backst1[20],backsp1[20],backst2[20],backsp2[20];
    double back1[20],back2[20],slope[20],rise[20],area[20];
    double y_data;
    double cal,nh3area,nh3con,n20area,n20con,temp,y,m;
    double p_sep,fd_point,fac,xfac,yfac,yvalue;
    char head_stc[5],fname[100];
    char data[100000],dir[100];
    char text[2500];
    char db[8],hb[4];
    char dat[20],name[100][12];
    FILE *fp,*fp1;
    APIRET rc;
    HFILE fh;
    FILESTATUS3 fib;
    struct fib;

    head_stc[4]=0;

    for(i=0;i<=2499;i++)
        text[i]=0;

    fil=1;
    fibs=sizeof(FILESTATUS3);

    fp=fopen("file1.epm","r");
    fscanf(fp,"%d",&fnum);
    fscanf(fp,"%s",dir);
    for(i=0;i<=fnum-1;i++)
    {
        fscanf(fp,"%s",name[i]);
    }
    fclose(fp);

    for(z=0;z<=fnum-1;i++)
    {
        strcpy(fname,dir);
        strcat(fname,name[z]);
        printf("%s\n",fname);
        rc=DosOpen(fname,
                    &fh,&act,
                    0,
                    FILE_NORMAL,
                    FILE_OPEN,
                    OPEN_ACCESS_READONLY | OPEN_SHARE_DENYNONE,
                    NULL);

        if(rc != 0)
            printf("DosOpen ERROR %ld\n in line 44",rc);

        rc=DosQueryFileInfo(fh,fil,&fib,fibs);
        if(rc != 0)
            printf("DosQuery ERROR %ld in line 47\n",rc);
    }
}

```

```

DosClose(fh);

fp=fopen(fname,"rb");

for(lenght=0;lenght<=(fib.cbFile-1);lenght++)
    fscanf(fp,"%c",&data[lenght]);

fclose(fp);

fp=fopen("peak.tab","r");
fscanf(fp,"%ld",&n_of_peaks);
for(i=0;i<=(n_of_peaks-1);i++)
{
    fscanf(fp,"%lf\n%lf",&backst1[i],&backsp1[i]);
    fscanf(fp,"%lf\n%lf",&backst2[i],&backsp2[i]);
    fscanf(fp,"%lf\n%lf",&peakst[i],&peaksp[i]);
}
fclose(fp);

hb[0]=data[80];
hb[1]=data[81];
hb[2]=data[82];
hb[3]=data[83];

head_stl=hextolo(hb);

hb[0]=data[84];
hb[1]=data[85];
hb[2]=data[86];
hb[3]=data[87];

head_sts=hextolo(hb);

hb[0]=data[88];
hb[1]=data[89];
hb[2]=data[90];
hb[3]=data[91];

data_stl=hextolo(hb);

hb[0]=data[92];
hb[1]=data[93];
hb[2]=data[94];
hb[3]=data[95];

data_size=hextolo(hb);

hb[0]=data[96];
hb[1]=data[97];
hb[2]=data[98];
hb[3]=data[99];

n_points=hextolo(hb);

x_exp=data[142];
y_exp=data[143];

db[0]=data[102];
db[1]=data[103];
db[2]=data[104];
db[3]=data[105];
db[4]=data[106];
db[5]=data[107];
db[6]=data[108];
db[7]=data[109];

p_sep=hextodbl(db);

hb[0]=data[110];
hb[1]=data[111];
hb[2]=data[112];
hb[3]=data[113];

f_point=hextolo(hb);

fd_point=p_sep*f_point;

```

```

di=data_stl+16;

xfac=1.0;

if(x_exp>128)
{
    x_exp=256-x_exp;
    for(i=0;i<(long)x_exp;i++)
        xfac=xfac/2.0;
}
else
{
    for(i=0;i<(long)x_exp;i++)
        xfac=xfac*2.0;
}

yfac=1.0;

if(y_exp>128)
{
    y_exp=256-y_exp;
    for(i=0;i<(long)y_exp;i++)
        yfac=yfac/2.0;
}
else
{
    for(i=0;i<(long)y_exp;i++)
        yfac=yfac*2.0;
}

fac=1.0;
for(i=0;i<=30;i++)
    fac=fac/2.0;

printf("TEXT HEADER START: %ld\n",head_stl);
printf("NUMBER OF BYTES IN TEXT HEADER: %ld\n",head_sts);
printf("DATA HEADER START: %ld\n",data_stl);
printf("DATA SEGMENT SIZE: %ld\n",data_size);
printf("NUMBER OF DATA POINTS: %ld\n",n_points);
printf("RAW X EXP: %d\n",x_exp);
printf("RAW Y EXP: %d\n",y_exp);
printf("DATA POINT SEPERATION: %lf\n",p_sep);
printf("FIRST DATA POINT: %lf\n",fd_point);
printf("MULTI FACTOR: %e\n",fac);
printf("X REAL FAC: %e\n",xfac);
printf("Y REAL FAC: %e\n",yfac);

lenght=head_stl;
j=0;
while(lenght<=(head_stl+head_sts))
{
    text[j]=data[lenght];
    j++;
    lenght++;
}

for(l=0;l<=j;l++)
{
    if(text[l]==67)
    {
        if(text[l+1]==82)
        {
            if(text[l+2]==69)
            {
                if(text[l+3]==65)
                {
                    if(text[l+4]==84)
                    {
                        if(text[l+5]==69)
                        {
                            if(text[l+6]==68)
                            {
                                dat[0]=text[l+12];
                                dat[1]=text[l+13];
                                dat[2]=text[l+14];
                                dat[3]=text[l+15];
                                dat[4]=text[l+16];
                                dat[5]=text[l+17];
                                dat[6]=text[l+18];
                            }
                        }
                    }
                }
            }
        }
    }
}

```

```

        dat[7]=text[l+30];
        dat[8]=text[l+31];
        dat[9]=text[l+18];
        dat[10]=text[l+19];
        dat[11]=text[l+20];
        dat[12]=text[l+21];
        dat[13]=text[l+22];
        dat[14]=text[l+23];
        dat[15]=text[l+24];
        dat[16]=text[l+25];
        dat[17]=text[l+26];
        dat[18]=0;
    }
}
}
}
}
}
}
}
/*      printf("%s\n",dat);*/

/*calculate the point values of the wave numbers*/
for(i=0;i<=n_of_peaks-1;i++)
{
    temp=(backst1[i]/p_sep)-(fd_point/p_sep);
    y=modf(temp,&m);
    if (y>.5)
        m=m+1.0;
    bast1[i]=(long)m;

    temp=(backsp1[i]/p_sep)-(fd_point/p_sep);
    y=modf(temp,&m);
    if (y>.5)
        m=m+1;
    basp1[i]=(long)m;

    temp=(backst2[i]/p_sep)-(fd_point/p_sep);
    y=modf(temp,&m);
    if (y>.5)
        m=m+1;
    bast2[i]=(long)m;

    temp=(backsp2[i]/p_sep)-(fd_point/p_sep);
    y=modf(temp,&m);
    if (y>.5)
        m=m+1;
    basp2[i]=(long)m;

    temp=(peakst[i]/p_sep)-(fd_point/p_sep);
    y=modf(temp,&m);
    if (y>.5)
        m=m+1;
    pkst[i]=(long)m;

    temp=(peaksp[i]/p_sep)-(fd_point/p_sep);
    y=modf(temp,&m);
    if (y>.5)
        m=m+1;
    pksp[i]=(long)m;
}

for(i=0;i<=(n_of_peaks-1);i++)
{
    printf("PEAK[%d]: %d\t%d\n",i+1,pkst[i],pksp[i]);
    printf("BACK1[%d]: %d\t%d\n",i+1,bast1[i],basp1[i]);
    printf("BACK2[%d]: %d\t%d\n",i+1,bast2[i],basp2[i]);
}

for(j=0;j<=(n_of_peaks-1);j++)
{
    back1[j]=0.0;
    for(i=0;i<=(basp1[j]-bast1[j]);i++)
    {
        index=di+((bast1[j]+i)*4);

        for(l=0;l<=3;l++,index++)
            hb[l]=data[index];
    }
}

```

```

    y_data=hextolo(hb);
    y_data=y_data*fac*yfac;

    back1[j]=back1[j]+y_data;
    /*printf("%lf\t%lf\n",back1[j],y_data);*/
}
back1[j]=back1[j]/(basp1[j]-bast1[j]+1);

back2[j]=0.0;
for(i=0;i<=(basp2[j]-bast2[j]);i++)
{
    index=di+((bast2[j]+i)*4);

    for(l=0;l<=3;l++,index++)
        hb[l]=data[index];

    y_data=hextolo(hb);
    y_data=y_data*fac*yfac;

    back2[j]=back2[j]+y_data;
    /*printf("%lf\t%lf\n",back2[j],y_data);*/
}

back2[j]=back2[j]/(basp2[j]-bast2[j]+1);

rise[j]=back2[j]-back1[j];
st=bast1[j]+((basp1[j]-bast1[j])/2);
sp=bast2[j]+((basp2[j]-bast2[j])/2);
run[j]=(sp-st);
slope[j]=rise[j]/run[j];

/*printf("Back1[%ld]: %lf\n",j+1,back1[j]);
printf("Back2[%ld]: %lf\n",j+1,back2[j]);
printf("Run[%ld]: %ld = %ld - %ld\n",j+1,run[j],sp,st);
printf("Slope[%ld]: %.15lf = %.15lf / %ld\n",j+1,slope[j],rise[j],run[j]);*/

area[j]=0.0;

for(i=(pkst[j]-st),l=0;i<=(pksp[j]-st);i++,l++)
{
    index=di+((pkst[j]+l)*4);

    for(d=0;d<=3;d++,index++)
        hb[d]=data[index];

    y_data=hextolo(hb);
    y_data=y_data*fac*yfac;

    yvalue=slope[j]*(double)i+back1[j];
    area[j]=area[j]+(y_data-yvalue);
    /*printf("YVALUE[%ld]: %e\n",i,yvalue[l]);
    printf("Y_DATA: %e\n",pkst[j]+l,y_data);
    printf("ASORBANCE: %e\n",y_data-yvalue);*/
}

area[j]=area[j]/(1.0/p_sep);
}

fp=fopen("ftan.cal","r");
fp1=fopen("anal.dat","w+");
fprintf(fp1,"%s\n%s\n",name,dat);
for(i=0;i<=n_of_peaks-1;i++)
{
    fscanf(fp,"%lf",&cal);
    peakcon[i]=cal*area[i]*(1/p_sep);
    fprintf(fp1,"CON[%ld]: %lf AREA[%ld]: %lf ",
            i+1,peakcon[i],i+1,area[i]*(1.0/p_sep));
    fprintf(fp1,"PEAK[%ld]: %lf %lf\n",i+1,peakst[i],peaksp[i]);
}
fprintf(fp1,"\n");
fclose(fp);
fclose(fp1);
printf("%s\n%s\n",name,dat);
}

```

}

C.5 PHOTO - Ammonia Monitoring system

PHOTO consists of five files (PHOTO, COMOPEN, COMREAD, COMFLUSH, COMWRITE) with five functions (OS/2 ANSI standard C code).

SOURCE LISTING : PHOTO Ver. 1.0

```

/*Photo.exe*/
/*Author Clifford J. Hewitt*/
/*Los Alamos National Lab*/
/*MS J565*/
/*Phone 505 667 7100*/

#define INCL_DOSPROCESS
#define INCL_BASE
#define INCL_DOSERRORS
#define INCL_BDCALLS
#define INCL_DOSFILEMGR
#define INCL_DOSDEVIOTCL
#define INCL_DOSDATETIME
#define SERIAL "COM3"

#include <string.h>
#include <os2.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <signal.h>

UCHAR fhd;

void handler(int);
void comopen();
void comwrite(UCHAR [],int);
int comread(UCHAR []);
void comflush();

INT main()
{
    APIRET RC=0;
    HFILE hfileFileHandle;
    UCHAR readbuf1[2048];
    int i,j,x,z,inum,step,xx,ret;
    int hbwto,lbwto,hbrto,lbrto;
    int cmdnum,len;
    char file_name[13],cmd1[50],cmd[1000][50];
    char text[50],num[3],c;
    char str_tim[25],moni[6],set_num[25];
    char dat[50],dati[50],monf[50],out[100],nh3[10],h2o[10];
    char tim[50],tim1[50],day[50];
    size_t ltext;
    ULONG lenght;
    FILE *fp;
    struct tm *newtime;
    time_t ltime;

    if (signal(SIGINT, handler)==SIG_ERR)
    {
        perror("Could not set SIGINT");
        exit(-1);
    }

    system("cls");

    printf("\n\n\n\n\t\t\tPHOTO.EXE VERSION 1.00\n\n");
    DosSleep(200);

    comopen();

    printf("\n\n\n\n\t\t\tTHE B&K IS BEING RESET\n\n");

```

```

strcpy(set_num,"RESET_SY FUL");
len=strlen(set_num);
set_num[len]=10;
set_num[len+1]=0;
len=strlen(set_num);
comwrite(set_num,len);
DosSleep(15000);

strcpy(set_num,"CU_SE?");
len=strlen(set_num);
set_num[len]=10;
set_num[len+1]=0;
len=strlen(set_num);
comwrite(set_num,len);
DosSleep(200);
comread(readbuf1);
system("cls");
printf("\n\n\n\tTHE CURRENT SETUP IS NUMBER %s\n",readbuf1);
printf("\n\tIS THIS THE CORRECT SETUP Y or N\n\n\t");
scanf("%c",&c);
if(c=='n' || c=='N')
{
    printf("ENTER THE NEW SETUP NUMBER\n\n");
    scanf("%d",&z);
    fp=fopen("setch","r");

    for(i=0;i<=2;i++)
    {
        fgets(set_num,50,fp);
        len=strlen(set_num);
        set_num[len]=10;
        set_num[len+1]=0;
        len=strlen(set_num);
        comwrite(set_num,len);
        DosSleep(2000);
    }

    len=strlen(readbuf1);

    if(readbuf1[len-1]==20)
    {
        num[0]=readbuf1[len];
        num[1]=0;
    }

    if(readbuf1[len-1]!=20)
    {
        num[0]=readbuf1[len-1];
        num[1]=readbuf1[len];
        num[2]=0;
    }

    inum=atoi(num);
    step=inum-z;

    if(step<0)
    {
        for(i=step;i!=0;i++)
        {
            strcpy(set_num,"P_K S_3");
            set_num[7]=10;
            set_num[8]=0;
            comwrite(set_num,8);
            DosSleep(200);
        }
    }

    if(step>0)
    {
        for(i=step;i!=0;i--)
        {
            strcpy(set_num,"P_K S_1");
            set_num[7]=10;
            set_num[8]=0;
            comwrite(set_num,8);
            DosSleep(200);
        }
    }
}

```



```

for(i=0;i<=1;i++)
{
    fgets(set_num,50,fp);
    len=strlen(set_num);
    set_num[len]=10;
    set_num[len+1]=0;
    len=strlen(set_num);
    comwrite(set_num,len);
    DosSleep(200);
}

fclose(fp);
strcpy(set_num,"CU_SE?");
len=strlen(set_num);
set_num[len]=10;
set_num[len+1]=0;
len=strlen(set_num);
comwrite(set_num,len);
DosSleep(200);
comread(readbuf1);
printf("\n\n\n\tTHE CURRENT SETUP IS NUMBER %s\n",readbuf1);
DosSleep(4000);
}

fp=fopen("bkcmd","r");
fscanf(fp,"%d\n",&cmdnum);
for(i=1;i<=cmdnum;i++)
{
    fgets(cmd[i],50,fp);
    /*printf("%s",cmd[i]);*/
    len=strlen(cmd[i]);
    cmd[i][len]=10;
    cmd[i][len+1]=0;
    len=strlen(cmd[i]);
    comwrite(cmd[i],len);
    DosSleep(200);
}
fclose(fp);

system("cls");

text[0]=83;
text[1]=84;
text[2]=65;
text[3]=95;
text[4]=77;
text[5]=10;

comwrite(text,6);
DosSleep(200);
while(x!=27)
{
    ret=comread(readbuf1);
    if(ret!=-1)
        DosSleep(2000);

    DosSleep(2000);
    comflush();

    time(&lttime);
    newtime=localtime(&lttime);
    strcpy(str_tim,asctime(newtime));

    dati[0]=str_tim[23];
    dati[1]=str_tim[4];
    dati[2]=str_tim[5];
    dati[3]=str_tim[6];
    dati[4]=str_tim[8];
    dati[5]=str_tim[9];
    dati[6]=0;

    if(dati[4]==32)
        dati[4]=48;

    strcpy(file_name,dati);
    strcat(file_name,"bk.prn");
}

```

```

moni[0]=str_tim[4];
moni[1]=str_tim[5];
moni[2]=str_tim[6];
moni[3]=0;

if(str_tim[8]==32)
    day[0]=48;
else
    day[0]=str_tim[8];

day[1]=str_tim[9];
day[2]=0;

if(strcmp(moni,"Jan")==0)
    strcpy(monf,"01");
if(strcmp(moni,"Feb")==0)
    strcpy(monf,"02");
if(strcmp(moni,"Mar")==0)
    strcpy(monf,"03");
if(strcmp(moni,"Apr")==0)
    strcpy(monf,"04");
if(strcmp(moni,"May")==0)
    strcpy(monf,"05");
if(strcmp(moni,"Jun")==0)
    strcpy(monf,"06");
if(strcmp(moni,"Jul")==0)
    strcpy(monf,"07");
if(strcmp(moni,"Aug")==0)
    strcpy(monf,"08");
if(strcmp(moni,"Sep")==0)
    strcpy(monf,"09");
if(strcmp(moni,"Oct")==0)
    strcpy(monf,"10");
if(strcmp(moni,"Nov")==0)
    strcpy(monf,"11");
if(strcmp(moni,"Dec")==0)
    strcpy(monf,"12");

printf("%s\n",readbuf1);

if(readbuf1[0]==32||readbuf1[0]==88)
{
    fp=fopen(file_name,"a+");

    strcpy(dat,monf);
    strcat(dat,day);

    xx=12;
    i=0;

    while(readbuf1[xx]==32||readbuf1[xx]==46)
        xx++;

    while(readbuf1[xx]!=32)
    {
        nh3[i]=readbuf1[xx];
        i++;
        xx++;
    }
    nh3[i]=0;

    while(readbuf1[xx]==32||readbuf1[xx]==46)
        xx++;

    i=0;
    while(readbuf1[xx]!=0)
    {
        h20[i]=readbuf1[xx];
        i++;
        xx++;
    }
    h20[i]=0;

    tim[0]=readbuf1[4];
    tim[1]=readbuf1[5];
    tim[2]=readbuf1[7];

```

```

    tim[3]=readbuf1[8];
    tim[4]=0;

    tim1[0]=readbuf1[4];
    tim1[1]=readbuf1[5];
    tim1[2]=readbuf1[6];
    tim1[3]=readbuf1[7];
    tim1[4]=readbuf1[8];
    tim1[5]=readbuf1[9];
    tim1[6]=readbuf1[10];
    tim1[7]=readbuf1[11];
    tim1[8]=0;

    fprintf(fp,"%s %s %s %s\n",dati,tim1,nh3,h20);
    fclose(fp);

    fp=fopen("bk.dat","w");
    fprintf(fp,"%s\n%s\n%s",dat,tim,nh3);
    fclose(fp);
}

if(readbuf1[0]==13)
{
    fp=fopen("bkerrlog","a+");
    fprintf(fp,"%s\n",readbuf1);
    fclose(fp);
    text[0]=65;
    text[1]=95;
    text[2]=77;
    text[3]=10;
    commwrite(text,4);
}
}

void handler(int sig)
{
    APIRET RC;
    HFILE hfileFileHandle;
    ULONG ulBytesWritten;
    char text[8];

    hfileFileHandle=fhd;

    text[0]=83;
    text[1]=84;
    text[2]=79;
    text[3]=80;
    text[4]=95;
    text[5]=77;
    text[6]=10;
    text[7]=0;

    RC = DosWrite(hfileFileHandle,
        text,
        7,
        &ulBytesWritten);

    if (RC)
    {
        printf ("Error in writting the string to the Com Port RC=%ld\n",RC);
        exit(1);
    }
    printf("EXITING PROGRAM");
    exit(0);
}

```

SOURCE LISTING : COMOPEN Ver. 1.0

```

#define INCL_DOSPROCESS
#define INCL_BASE
#define INCL_DOSERRORS
#define INCL_BDCALLS
#define INCL_DOSFILEMGR
#define INCL_DOSDEVIOTCL
#define SERIAL "COM3"

#include <string.h>
#include <os2.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <signal.h>

UCHAR fhd;

INT comopen()
{
    APIRET RC=0;
    HFILE hfileFileHandle;
    USHORT usBPS=9600;
    ULONG ulLength=8L;
    ULONG ulAction = 0;
    ULONG ulParmLen;
    ULONG ulBytesWritten;
    UCHAR baud[2];
    UCHAR lchar[4];
    UCHAR ldc[11],rq[2];
    int i,j,x,hbbaud,lbbaud,ibaud;
    int hbwto,lbwto,hbrto,lbrto;
    int iwto,irto;
    int hbrq,lbrq,irq;
    char rs2[3];
    char dcb[11];
    size_t ltext;
    ULONG lenght;

    rs2[0]=8;
    rs2[1]=0;
    rs2[2]=0;

    baud[0]=0;
    baud[1]=0;

    dcb[0]=112;
    dcb[1]=23;
    dcb[2]=112;
    dcb[3]=112;
    dcb[4]=0;
    dcb[5]=3;
    dcb[6]=12;
    dcb[7]=0;
    dcb[8]=0;
    dcb[9]=17;
    dcb[10]=19;

    RC = DosOpen(SERIAL,
        &hfileFileHandle,
        &ulAction,
        0L,
        FILE_SYSTEM,
        OPEN_ACTION_OPEN_IF_EXISTS,
        OPEN_ACCESS_READWRITE | OPEN_SHARE_DENYREADWRITE,
        (PEADP2)NULL);
    if (RC)
    {
        printf ("Cannot open the Com Port to write RC=%ld\n",RC);
        exit (-1);
    }
    fhd=hfileFileHandle;

```

```

ulParamLen=3;
RC = DosDevIOctl(hfileFileHandle,
    IOCTL_ASYNC,
    ASYNC_SETLINECTRL,
    rs2,
    3,
    &ulParamLen,
    NULL,
    0,
    NULL);
if (RC)
{
    printf ("Cannot write SETLINECTRL to the Com Port RC=%ld\n",RC);
    exit (-1);
}

ulParamLen=2;
RC = DosDevIOctl(hfileFileHandle,
    IOCTL_ASYNC,
    ASYNC_SETBAUDRATE,
    (PULONG)&usBPS,
    sizeof(usBPS),
    &ulParamLen,
    NULL,
    0,
    NULL);
if (RC)
{
    printf ("Cannot write SETBAUDRATE to the Com Port RC=%ld\n",RC);
    exit (-1);
}

ulParamLen=11;
RC = DosDevIOctl(hfileFileHandle,
    IOCTL_ASYNC,
    0x53,
    dcb,
    11,
    &ulParamLen,
    NULL,
    0,
    NULL);
if (RC)
{
    printf ("Cannot write DBD to the Com Port RC=%ld\n",RC);
    exit (-1);
}

RC = DosDevIOctl(hfileFileHandle,
    IOCTL_ASYNC,
    0x61,
    NULL,
    0,
    NULL,
    (PULONG)baud,
    sizeof(baud),
    &lenght);
if (RC)
{
    printf ("Cannot get baud rate from the Com Port RC=%ld\n",RC);
    /*exit (-1);*/
}

hbbaud=baud[1];
lbbaud=baud[0];
hbbaud=hbbaud<<8;
ibaud=hbbaud+lbbaud;
/*printf("BAUD RATE %d\n",ibaud);*/

RC = DosDevIOctl(hfileFileHandle,
    IOCTL_ASYNC,
    0x62,
    NULL,
    0,
    NULL,
    (PULONG)lchar,
    sizeof(lchar),
    &lenght);

```

```

if (RC)
{
    printf ("Cannot get line ctrl from the Com Port RC=%ld\n",RC);
    exit (-1);
}

/*printf("DATA BIT %d\n",lchar[0]);
printf("PARITY %d\n",lchar[1]);
printf("STOP BITS %d\n",lchar[2]);
printf("BREAK %d\n",lchar[3]);*/

RC = DosDevIOctl(hfileFileHandle,
    IOCTL_ASYNC,
    0x73,
    NULL,
    0,
    NULL,
    (PULONG)ldcb,
    sizeof(ldcb),
    &lenght);

if (RC)
{
    printf ("Cannot get DCB from the Com Port RC=%ld\n",RC);
    exit (-1);
}

hbwto=ldcb[1];
lbwto=ldcb[0];
hbwto=hbwto<<8;
iwto=(hbwto+lbwto)*.01;
hbrto=ldcb[3];
lbrto=ldcb[2];
hbrto=hbrto<<8;
irto=(hbrto+lbrto)*.01;
/*printf("Write Time Out %d Sec\n",iwto);
printf("Read Time Out %d Sec\n",irto);
printf("Flag1 %d\n",ldcb[4]);
printf("Flag2 %d\n",ldcb[5]);
printf("Flag3 %d\n",ldcb[6]);
printf("Error Replacement Char %d\n",ldcb[7]);
printf("Break Replacement Char %d\n",ldcb[8]);
printf("XON Char %d\n",ldcb[9]);
printf("XOFF Char %d\n",ldcb[10]);*/
return;
}

```

SOURCE LISTING : COMREAD Ver. 1.0

```

#define INCL_DOSPROCESS
#define INCL_BASE
#define INCL_DOSERRORS
#define INCL_BDCALLS
#define INCL_DOSFILEMGR
#define INCL_DOSDEVIOCTL

#include <string.h>
#include <os2.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <signal.h>

INT comread(UCHAR readbuf1[])
{
    extern UCHAR fhd;

    APIRET RC=0;
    HFILE hfileFileHandle;
    ULONG ulParmLen;
    ULONG ulBytesWritten;
    UCHAR rq[4];
    UCHAR readbuf,dump[2048];
    int i,j,x;
    int hbrq,lbrq,irq;
    ULONG lenght;

    hfileFileHandle=fhd;

    i=0;

    while(readbuf!=10)
    {
        RC = DosRead(hfileFileHandle,
                    &readbuf,
                    1,
                    &ulBytesWritten);

        if (readbuf==10)
            readbuf1[i-1]=0;
        else
            readbuf1[i]=readbuf;

        i++;
    }

    return(0);
}

```

SOURCE LISTING : COMWRITE Ver. 1.0

```

#define INCL_DOSPROCESS
#define INCL_BASE
#define INCL_DOSERRORS
#define INCL_BDCALLS
#define INCL_DOSFILEMGR
#define INCL_DOSDEVICTL

#include <string.h>
#include <os2.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <signal.h>

VOID commwrite(char text[],int len)
(
    extern UCHAR fhd;

    APIRET RC=0;
    HFILE hfileFileHandle;
    ULONG ulBytesWritten;

    hfileFileHandle=fhd;

    RC = DosWrite(hfileFileHandle,
        text,
        len,
        &ulBytesWritten);
    if (RC)
    {
        printf ("Error in writting the string to the Com Port RC=%ld\n",RC);
        exit(1);
    }
    return;
}

```


SOURCE LISTING : COMFLUSH Ver. 1.0

```

#define INCL_DOSPROCESS
#define INCL_BASE
#define INCL_DOSERRORS
#define INCL_BDCALLS
#define INCL_DOSFILEMGR
#define INCL_DOSDEVOCTL

#include <string.h>
#include <os2.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <signal.h>

VOID comflush()
{
    extern UCHAR fhd;

    APIRET RC=0;
    HFILE hfileFileHandle;
    ULONG ulParmLen;
    ULONG ulBytesWritten;
    UCHAR rq[4];
    UCHAR readbuf,dump[2048];
    int i,j,x;
    int hbrq,lbrq,irq,hbrq1,lbrq1,irq1;
    ULONG lenght;

    hfileFileHandle=fhd;

    RC = DosDevIOCtl(hfileFileHandle,
        IOCTL_ASYNC,
        0x68,
        NULL,
        0L,
        NULL,
        (PULONG)rq,
        sizeof(rq),
        &lenght);
    if (RC)
    {
        printf ("Cannot write get read buff size to the Com Port RC=%ld\n",RC);
    }

    hbrq=rq[1];
    lbrq=rq[0];
    hbrq=hbrq<<8;
    irq=hbrq+lbrq;
    hbrq1=rq[3];
    lbrq1=rq[2];
    hbrq1=hbrq1<<8;
    irq1=hbrq1+lbrq1;
    /*printf("buffer size = %d %d\n",irq,irq1);*/
    RC = DosRead(hfileFileHandle,
        dump,
        irq,
        &ulBytesWritten);

    return;
}

```

C.6 DACS - GMS-2 Host computer

DACS consists of one file with two functions: main and tim_get (OS/2 ANSI standard C code).

SOURCE LISTING : DACS Ver. 1.5

```

/*****
* File Name      : COMM.C
*
*
* Required files: To build and run this sample code the following files
*                  are needed:
*                  - COMM.C
*                  - COMM.DEF
*                  - COMM.MAK
*                  - COMM.LNK
*
*****/
#define INCL_DOSPROCESS
#define INCL_BASE
#define INCL_DOSERRORS
#define INCL_BDCALLS
#define INCL_DOSFILEMGR
#define INCL_DOSDEVIOTCL
#define PRINTER "COM1"

#include <string.h>
#include <os2.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <signal.h>

void setgc3zero();
void setphotozero();
void setftirzero();

/* Define the parameter to Hide and Restore the cursor */
#define HIDE -1
#define RESTORE 0

char rs2[3];
char tim[20],area[20],rt[20],con[20],file[20];
char timgc3[7],areagc3[6],rtgc3[6],congc3[6],filegc3[6],data_str[200];
char photo_tim[6],photo_dat[6],photo_con[12],conhb[6],conlb[6];
char filef[6],timf[6],n2area[6],n2con[6],nharea[6],nhcon[6];
char filex[6],file2[6],file3[6],temp[6];

INT main()
{
    APIRET RC=0;
    HFILE hfileFileHandle;
    ULONG ulLength=8L;
    ULONG ulAction = 0;
    ULONG ulParmLen=3;
    ULONG ulBytesWritten;
    int i,x,gc3flag,photoflag,ftirflag,inx,next,lstflag;
    float xx;
    long yy;
    size_t lenght;
    time_t timergc3,timerphoto,timerftir,timer,lst;
    FILE *fp;
    int dumm=1;

    gc3flag=0;
    photoflag=0;
    ftirflag=0;

    rs2[0]=8;
    rs2[1]=0;

```

```

rs2[2]=0;
x=0;

setgc3zero();
setphotozero();
setftirzero();

time(&timergc3);
time(&timerphoto);
time(&timerftir);
time(&tst);

printf("\t\t*****DACS PROGRAM*****\n");
printf("\t\t*****VERSION 1.50*****\n");
DosSleep(10000);

while(x==0)
{
/*****
/*****GET GC3 DATA*****/
/*****
if ((fp=fopen("c:\\gc3\\temp.prc","r"))!=0)
{
fscanf(fp,"%s\n%s\n%s\n%s\n%s",tim,area,rt,con,file);
fclose(fp);
DosDelete("c:\\gc3\\temp.prc");

timgc3[0]=2;
timgc3[1]=tim[0];
timgc3[2]=tim[1];
timgc3[3]=tim[3];
timgc3[4]=tim[4];
timgc3[5]=13;
timgc3[6]=0;

rtgc3[0]=48;
rtgc3[1]=48;
rtgc3[2]=rt[2];
rtgc3[3]=rt[3];
rtgc3[4]=13;
rtgc3[5]=0;

if ((length=strlen(area))==5)
{
areagc3[0]=48;
areagc3[1]=48;
areagc3[2]=area[0];
areagc3[3]=area[1];
areagc3[4]=13;
areagc3[5]=0;
}
if ((length=strlen(area))==6)
{
areagc3[0]=48;
areagc3[1]=area[0];
areagc3[2]=area[1];
areagc3[3]=area[2];
areagc3[4]=13;
areagc3[5]=0;
}
if ((length=strlen(area))==7)
{
areagc3[0]=area[0];
areagc3[1]=area[1];
areagc3[2]=area[2];
areagc3[3]=area[3];
areagc3[4]=13;
areagc3[5]=0;
}
if (((length=strlen(area))<5) || ((length=strlen(area))>7))
{
areagc3[0]=48;
areagc3[1]=48;
areagc3[2]=48;
areagc3[3]=48;
areagc3[4]=13;
areagc3[5]=0;
}
}

```

```

    )
    i=0;
    while(con[i]!=46)
        i++;
    if((i<0) || (i>4))
    {
        congc3[0]=48;
        congc3[1]=48;
        congc3[2]=48;
        congc3[3]=48;
        congc3[4]=13;
        congc3[5]=0;
    }
    if(i==1)
    {
        congc3[0]=48;
        congc3[1]=48;
        congc3[2]=48;
        congc3[3]=con[0];
        congc3[4]=13;
        congc3[5]=0;
    }
    if(i==2)
    {
        congc3[0]=48;
        congc3[1]=48;
        congc3[2]=con[0];
        congc3[3]=con[1];
        congc3[4]=13;
        congc3[5]=0;
    }
    if(i==3)
    {
        congc3[0]=48;
        congc3[1]=con[0];
        congc3[2]=con[1];
        congc3[3]=con[2];
        congc3[4]=13;
        congc3[5]=0;
    }
    if(i==4)
    {
        congc3[0]=con[0];
        congc3[1]=con[1];
        congc3[2]=con[2];
        congc3[3]=con[3];
        congc3[4]=13;
        congc3[5]=0;
    }

    filegc3[0]=file[5];
    filegc3[1]=file[6];
    filegc3[2]=file[7];
    filegc3[3]=file[11];
    filegc3[4]=13;
    filegc3[5]=0;

    time(&timergc3);
    gc3flag=1;
}

```

```

DosSleep(1000);

```

```

/*****
/*****GET FTIR DATA*****/
/*****

```

```

if((fp=fopen("c:\\ftir\\temp.dat","r"))!=0)
{
    for(i=0;i<99;i++)
        fscanf(fp,"%c",&data_str[i]);
    fclose(fp);

    DosDelete("c:\\ftir\\temp.dat");

    filef[0]=48;
    filef[1]=data_str[5];
}

```

```

filef[2]=data_str[6];
filef[3]=data_str[7];
filef[4]=13;
filef[5]=0;

timf[0]=data_str[19];
timf[1]=data_str[20];
timf[2]=data_str[22];
timf[3]=data_str[23];
timf[4]=13;
timf[5]=0;

inx=0;
next=27;

while(data_str[next]==32)
    next++;

while(data_str[next]!=46)
{
    temp[inx]=data_str[next];
    inx++;
    next++;
}

if((inx-1)==0)
{
    nharea[0]=48;
    nharea[1]=48;
    nharea[2]=48;
    nharea[3]=temp[0];
    nharea[4]=13;
    nharea[5]=0;
}

if((inx-1)==1)
{
    nharea[0]=48;
    nharea[1]=48;
    nharea[2]=temp[0];
    nharea[3]=temp[1];
    nharea[4]=13;
    nharea[5]=0;
}

if((inx-1)==2)
{
    nharea[0]=48;
    nharea[1]=temp[0];
    nharea[2]=temp[1];
    nharea[3]=temp[2];
    nharea[4]=13;
    nharea[5]=0;
}

if((inx-1)==3)
{
    nharea[0]=temp[0];
    nharea[1]=temp[1];
    nharea[2]=temp[2];
    nharea[3]=temp[3];
    nharea[4]=13;
    nharea[5]=0;
}

if((inx-1)>3)
{
    nharea[0]=48;
    nharea[1]=48;
    nharea[2]=48;
    nharea[3]=48;
    nharea[4]=13;
    nharea[5]=0;
}

while(data_str[next]!=32)
    next++;

```

```

inx=0;

while(data_str[next]==32)
    next++;

while(data_str[next]!=46)
{
    temp[inx]=data_str[next];
    inx++;
    next++;
}

if((inx-1)==0)
{
    n2area[0]=48;
    n2area[1]=48;
    n2area[2]=48;
    n2area[3]=temp[0];
    n2area[4]=13;
    n2area[5]=0;
}

if((inx-1)==1)
{
    n2area[0]=48;
    n2area[1]=48;
    n2area[2]=temp[0];
    n2area[3]=temp[1];
    n2area[4]=13;
    n2area[5]=0;
}

if((inx-1)==2)
{
    n2area[0]=48;
    n2area[1]=temp[0];
    n2area[2]=temp[1];
    n2area[3]=temp[2];
    n2area[4]=13;
    n2area[5]=0;
}

if((inx-1)==3)
{
    n2area[0]=temp[0];
    n2area[1]=temp[1];
    n2area[2]=temp[2];
    n2area[3]=temp[3];
    n2area[4]=13;
    n2area[5]=0;
}

if((inx-1)>3)
{
    n2area[0]=48;
    n2area[1]=48;
    n2area[2]=48;
    n2area[3]=48;
    n2area[4]=13;
    n2area[5]=0;
}

while(data_str[next]!=32)
    next++;

inx=0;

while(data_str[next]==32)
    next++;

while(data_str[next]!=46)
{
    temp[inx]=data_str[next];
    inx++;
    next++;
}

if((inx-1)==0)

```

```

{
    nhcon[0]=48;
    nhcon[1]=48;
    nhcon[2]=48;
    nhcon[3]=temp[0];
    nhcon[4]=13;
    nhcon[5]=0;
}

if((inx-1)==1)
{
    nhcon[0]=48;
    nhcon[1]=48;
    nhcon[2]=temp[0];
    nhcon[3]=temp[1];
    nhcon[4]=13;
    nhcon[5]=0;
}

if((inx-1)==2)
{
    nhcon[0]=48;
    nhcon[1]=temp[0];
    nhcon[2]=temp[1];
    nhcon[3]=temp[2];
    nhcon[4]=13;
    nhcon[5]=0;
}

if((inx-1)==3)
{
    nhcon[0]=temp[0];
    nhcon[1]=temp[1];
    nhcon[2]=temp[2];
    nhcon[3]=temp[3];
    nhcon[4]=13;
    nhcon[5]=0;
}

if((inx-1)>3)
{
    nhcon[0]=48;
    nhcon[1]=48;
    nhcon[2]=48;
    nhcon[3]=48;
    nhcon[4]=13;
    nhcon[5]=0;
}

while(data_str[next]!=32)
    next++;

inx=0;

while(data_str[next]==32)
    next++;

while(data_str[next]!=46)
{
    temp[inx]=data_str[next];
    inx++;
    next++;
}

if((inx-1)==0)
{
    n2con[0]=48;
    n2con[1]=48;
    n2con[2]=48;
    n2con[3]=temp[0];
    n2con[4]=13;
    n2con[5]=0;
}

if((inx-1)==1)
{
    n2con[0]=48;
    n2con[1]=48;

```

```

n2con[2]=temp[0];
n2con[3]=temp[1];
n2con[4]=13;
n2con[5]=0;
}

```

```

if((inx-1)==2)
{
n2con[0]=48;
n2con[1]=temp[0];
n2con[2]=temp[1];
n2con[3]=temp[2];
n2con[4]=13;
n2con[5]=0;
}

```

```

if((inx-1)==3)
{
n2con[0]=temp[0];
n2con[1]=temp[1];
n2con[2]=temp[2];
n2con[3]=temp[3];
n2con[4]=13;
n2con[5]=0;
}

```

```

if((inx-1)>3)
{
n2con[0]=48;
n2con[1]=48;
n2con[2]=48;
n2con[3]=48;
n2con[4]=13;
n2con[5]=0;
}

```

```

time(&timerftir);
ftirflag=1;
}

```

```

DosSleep(1000);

```

```

/*****
/*****GET Photo DATA*****/
/*****

```

```

if((fp=fopen("c:\\photo\\bk.dat","r"))!=0)
{
fscanf(fp,"%s\n%s\n%s",photo_dat,photo_tim,photo_con);
fclose(fp);
DosDelete("c:\\photo\\bk.dat");

```

```

photo_dat[4]=13;
photo_dat[5]=0;

```

```

photo_tim[4]=13;
photo_tim[5]=0;

```

```

xx=atof(photo_con);
yy=(long)xx;

```

```

fp=fopen("dat","w");
fprintf(fp,"%ld",yy);
fclose(fp);

```

```

fp=fopen("dat","r");
fscanf(fp,"%s",photo_con);
fclose(fp);

```

```

if ((length=strlen(photo_con))==1)
{
conlb[0]=48;
conlb[1]=48;
conlb[2]=48;
conlb[3]=photo_con[0];
conlb[4]=13;
conlb[5]=0;
conhb[0]=48;

```



```

        conhb[1]=48;
        conhb[2]=48;
        conhb[3]=48;
        conhb[4]=13;
        conhb[5]=0;
    }
    if ((lenght=strlen(photo_con))==2)
    {
        conlb[0]=48;
        conlb[1]=48;
        conlb[2]=photo_con[0];
        conlb[3]=photo_con[1];
        conlb[4]=13;
        conlb[5]=0;
        conhb[0]=48;
        conhb[1]=48;
        conhb[2]=48;
        conhb[3]=48;
        conhb[4]=13;
        conhb[5]=0;
    }
    if ((lenght=strlen(photo_con))==3)
    {
        conlb[0]=48;
        conlb[1]=photo_con[0];
        conlb[2]=photo_con[1];
        conlb[3]=photo_con[2];
        conlb[4]=13;
        conlb[5]=0;
        conhb[0]=48;
        conhb[1]=48;
        conhb[2]=48;
        conhb[3]=48;
        conhb[4]=13;
        conhb[5]=0;
    }
    if ((lenght=strlen(photo_con))==4)
    {
        conlb[0]=photo_con[0];
        conlb[1]=photo_con[1];
        conlb[2]=photo_con[2];
        conlb[3]=photo_con[3];
        conlb[4]=13;
        conlb[5]=0;
        conhb[0]=48;
        conhb[1]=48;
        conhb[2]=48;
        conhb[3]=48;
        conhb[4]=13;
        conhb[5]=0;
    }
    if ((lenght=strlen(photo_con))==5)
    {
        conlb[0]=photo_con[1];
        conlb[1]=photo_con[2];
        conlb[2]=photo_con[3];
        conlb[3]=photo_con[4];
        conlb[4]=13;
        conlb[5]=0;
        conhb[0]=48;
        conhb[1]=48;
        conhb[2]=48;
        conhb[3]=photo_con[0];
        conhb[4]=13;
        conhb[5]=0;
    }
    if ((lenght=strlen(photo_con))==6)
    {
        conlb[0]=photo_con[2];
        conlb[1]=photo_con[3];
        conlb[2]=photo_con[4];
        conlb[3]=photo_con[5];
        conlb[4]=13;
        conlb[5]=0;
        conhb[0]=48;
        conhb[1]=48;
        conhb[2]=photo_con[0];
        conhb[3]=photo_con[1];
    }

```

```

        conhb[4]=13;
        conhb[5]=0;
    }
    if ((length=strlen(photo_con))==7)
    {
        conlb[0]=photo_con[3];
        conlb[1]=photo_con[4];
        conlb[2]=photo_con[5];
        conlb[3]=photo_con[6];
        conlb[4]=13;
        conlb[5]=0;
        conhb[0]=48;
        conhb[1]=photo_con[0];
        conhb[2]=photo_con[1];
        conhb[3]=photo_con[2];
        conhb[4]=13;
        conhb[5]=0;
    }
    if ((length=strlen(photo_con))==8)
    {
        conlb[0]=photo_con[4];
        conlb[1]=photo_con[5];
        conlb[2]=photo_con[6];
        conlb[3]=photo_con[7];
        conlb[4]=13;
        conlb[5]=0;
        conhb[0]=photo_con[0];
        conhb[1]=photo_con[1];
        conhb[2]=photo_con[2];
        conhb[3]=photo_con[3];
        conhb[4]=13;
        conhb[5]=0;
    }
    if ((length=strlen(photo_con))==0)
    {
        conlb[0]=48;
        conlb[1]=48;
        conlb[2]=48;
        conlb[3]=48;
        conlb[4]=13;
        conlb[5]=0;
        conhb[0]=48;
        conhb[1]=48;
        conhb[2]=48;
        conhb[3]=48;
        conhb[4]=13;
        conhb[5]=0;
    }
    if ((length=strlen(photo_con))>8)
    {
        conlb[0]=48;
        conlb[1]=48;
        conlb[2]=48;
        conlb[3]=48;
        conlb[4]=13;
        conlb[5]=0;
        conhb[0]=48;
        conhb[1]=48;
        conhb[2]=48;
        conhb[3]=48;
        conhb[4]=13;
        conhb[5]=0;
    }

    time(&timerphoto);
    photoflag=1;
}

DosSleep(1000);

time(&timer);

if(((timer-timergc3)>450)&&(gc3flag!=1))
    setgc3zero();

if(((timer-timerphoto)>1800)&&(photoflag!=1))
    setphotozero();

```

[illegible]

```

        strlen(rtc3),
        &ulBytesWritten);

if (RC)
{
    printf ("Error in writting the string to the Com Port RC=%ld\n",RC);
    exit(1);
}
RC = DosWrite(hfileFileHandle,
cong3,
strlen(cong3),
&ulBytesWritten);

if (RC)
{
    printf ("Error in writting the string to the Com Port RC=%ld\n",RC);
    exit(1);
}
RC = DosWrite(hfileFileHandle,
filegc3,
strlen(filegc3),
&ulBytesWritten);

if (RC)
{
    printf ("Error in writting the string to the Com Port RC=%ld\n",RC);
    exit(1);
}

RC = DosWrite(hfileFileHandle,
timf,
strlen(timf),
&ulBytesWritten);

if (RC)
{
    printf ("Error in writting the string to the Com Port RC=%ld\n",RC);
    exit(1);
}

RC = DosWrite(hfileFileHandle,
n2area,
strlen(n2area),
&ulBytesWritten);

if (RC)
{
    printf ("Error in writting the string to the Com Port RC=%ld\n",RC);
    exit(1);
}

RC = DosWrite(hfileFileHandle,
n2con,
strlen(n2con),
&ulBytesWritten);

if (RC)
{
    printf ("Error in writting the string to the Com Port RC=%ld\n",RC);
    exit(1);
}

RC = DosWrite(hfileFileHandle,
nharea,
strlen(nharea),
&ulBytesWritten);

if (RC)
{
    printf ("Error in writting the string to the Com Port RC=%ld\n",RC);
    exit(1);
}

RC = DosWrite(hfileFileHandle,
nhcon,
strlen(nhcon),
&ulBytesWritten);

```

```

    if (RC)
    {
        printf ("Error in writting the string to the Com Port RC=%ld\n",RC);
        exit(1);
    }

    RC = DosWrite(hfileFileHandle,
        filef,
        strlen(filef),
        &ulBytesWritten);

    if (RC)
    {
        printf ("Error in writting the string to the Com Port RC=%ld\n",RC);
        exit(1);
    }

    RC = DosWrite(hfileFileHandle,
        photo_tim,
        strlen(photo_tim),
        &ulBytesWritten);

    if (RC)
    {
        printf ("Error in writting the string to the Com Port RC=%ld\n",RC);
        exit(1);
    }

    RC = DosWrite(hfileFileHandle,
        conhb,
        strlen(conhb),
        &ulBytesWritten);

    if (RC)
    {
        printf ("Error in writting the string to the Com Port RC=%ld\n",RC);
        exit(1);
    }

    RC = DosWrite(hfileFileHandle,
        conlb,
        strlen(conlb),
        &ulBytesWritten);

    if (RC)
    {
        printf ("Error in writting the string to the Com Port RC=%ld\n",RC);
        exit(1);
    }

    RC = DosClose(hfileFileHandle);

    if (RC)
    {
        printf ("Error in closing the Com Port RC=%ld\n",RC);
        exit (1);
    }
}

void setgc3zero()
{
    timgc3[0]=2;
    timgc3[1]=48;
    timgc3[2]=48;
    timgc3[3]=48;
    timgc3[4]=48;
    timgc3[5]=13;
    timgc3[6]=0;

    areagc3[0]=48;
    areagc3[1]=48;
    areagc3[2]=48;
    areagc3[3]=48;
    areagc3[4]=13;
    areagc3[5]=0;
}

```

```

    rtgc3[0]=48;
    rtgc3[1]=48;
    rtgc3[2]=48;
    rtgc3[3]=48;
    rtgc3[4]=13;
    rtgc3[5]=0;

    cong3[0]=48;
    cong3[1]=48;
    cong3[2]=48;
    cong3[3]=48;
    cong3[4]=13;
    cong3[5]=0;

    filegc3[0]=48;
    filegc3[1]=48;
    filegc3[2]=48;
    filegc3[3]=48;
    filegc3[4]=13;
    filegc3[5]=0;
}

void setphotozero()
{
    photo_tim[0]=48;
    photo_tim[1]=48;
    photo_tim[2]=48;
    photo_tim[3]=48;
    photo_tim[4]=13;
    photo_tim[5]=0;

    photo_tim[0]=48;
    photo_tim[1]=48;
    photo_tim[2]=48;
    photo_tim[3]=48;
    photo_tim[4]=13;
    photo_tim[5]=0;

    conlb[0]=48;
    conlb[1]=48;
    conlb[2]=48;
    conlb[3]=48;
    conlb[4]=13;
    conlb[5]=0;

    conhb[0]=48;
    conhb[1]=48;
    conhb[2]=48;
    conhb[3]=48;
    conhb[4]=13;
    conhb[5]=0;
}

void setftirzero()
{
    timf[0]=48;
    timf[1]=48;
    timf[2]=48;
    timf[3]=48;
    timf[4]=13;
    timf[5]=0;

    n2area[0]=48;
    n2area[1]=48;
    n2area[2]=48;
    n2area[3]=48;
    n2area[4]=13;
    n2area[5]=0;

    n2con[0]=48;
    n2con[1]=48;
    n2con[2]=48;
    n2con[3]=48;
    n2con[4]=13;
    n2con[5]=0;

    nharea[0]=48;
    nharea[1]=48;

```

```
nharea[2]=48;  
nharea[3]=48;  
nharea[4]=13;  
nharea[5]=0;
```

```
nhcon[0]=48;  
nhcon[1]=48;  
nhcon[2]=48;  
nhcon[3]=48;  
nhcon[4]=13;  
nhcon[5]=0;
```

```
filef[0]=48;  
filef[1]=48;  
filef[2]=48;  
filef[3]=48;  
filef[4]=13;  
filef[5]=0;
```

)

C.7 GC3 – RGA3 Reduction Gas Detector H2 Host System

GC3F consists of four files with five functions (OS/2 ANSI standard C code).

SOURCE LISTING : GCF3 Ver. 1.5

```
#define INCL_DOSPROCESS
#define INCL_DOSFILEMGR
#define INCL_DOSSEMAPHORES

#include <stdio.h>
#include <signal.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include <os2.h>
#include <string.h>

FILE *fp;
int num;
int num1;
HMTX semh;

void handler(int);
void fil_name(char[],char[]);
int strip(char[],char[],char[],long*,double*,long*,long*);
void cre_name(char[],char[],char[]);
int behind(long,long*,int,char[]);

int main( int argc, char *argv[] )
{
    extern num;
    extern num1;

    int i,j,k,dl,err,flag1=0;
    ULONG delay,opmode;
    APIRET RC;
    int ex,inx,xman;
    int flag=-1;
    int raw_flag=0;
    char tim[9];
    char dat [9];
    char f_name[13];
    char r_name[13];
    char c_name[13];
    char gc3str[1000];
    char cdat[6];
    char pdat[6];
    char prn_file[13];
    char dat_path[50];
    char path[50];
    char comm1[50];
    char comm2[50];
    char comm3[50];
    char comm4[50];
    char comm5[50];
    char comm6[50];
    char comm7[50];
    long larea,lpat,ltim,off;
    double frt,con,fac;
    time_t time1,time2;
    FILE *fp1;
    HFILE fh;
    ULONG fil,fibs,act;
    FILESTATUS3 fib;
    size_t lengt;
    struct fib;

    tim[8]=0;
    dat[8]=0;
    f_name[12]=0;
    k=0;

    num=atoi(argv[1]);
```



```

num1=atoi(argv[2]);
if((fp1=fopen("gc3.dat","r"))==0)
{
    printf("ERROR IN OPENING GC3.DAT\n");
    exit(-1);
}
fscanf( fp1,"%s\n%d\n%lf\n%d",dat_path,&delay,&fac,&raw_flag);
fclose(fp1);

if((fp1=fopen("ti.dat","r"))==0)
{
    printf("ERROR IN OPENING TI.DAT\n");
    exit(-1);
}
fscanf( fp1,"%d",&dl);
fclose(fp1);

if (signal(SIGINT, handler)==SIG_ERR)
{
    perror("Could not set SIGINT");
    exit(-1);
}

printf("\n\n\n*****GC3F VERSION 1.50*****\n\n\n");

while(k!=27)
{
    bad_file_restart;;
    printf("flag1:\t%d\n",flag1);

    if(flag1==0)
    {
        RC=DosSleep(delay);
        if(RC!=0)
            printf("Dos Sleep ERROR %ld in line 89",RC);
    }

    if(flag1==1)
    {
        printf("OFF:\t%d\n",off);
        RC=DosSleep(delay+off);
        if(RC!=0)
            printf("Dos Sleep ERROR %ld in line 96",RC);
    }

    time(&time1);

    fil_name(f_name,r_name);

    printf("\nf_name\t%s\nr_name\t%s\n",f_name,r_name);

    fp1=fopen("tran.inp","w");
    fprintf(fp1,"anonymous\n");
    fprintf(fp1,"binary\n");
    fprintf(fp1,"prompt off\n");
    fprintf(fp1,"cd gc3\n");
    fprintf(fp1,"get %s\n",f_name);
    if(raw_flag==1)
        fprintf(fp1,"get %s\n",r_name);
    fprintf(fp1,"quit\n");
    fclose(fp1);

    system("tran");

    opmode=1;

    ex=0;
    while((fp1=fopen(f_name,"r"))==0)
    {
        printf("File %s was not found GC3 may have problem\n",f_name);
        DosSleep(150000);

        ex++;
        if(ex>0)
        {
            num1++;

```

```

        flag1=0;
        printf("There is possible problem with GC3 no RPT files have been\n");
        printf("transferred in 7.5 minutes will try next file number\n");
        goto bad_file_restart;
    }
    system("tran");
}
fclose(fp1);

if(raw_flag==1)
{
    ex=0;
    while((fp1=fopen(r_name,"r"))==0)
    {
        printf("File %s was not found GC3 may have problem\n",r_name);
        DosSleep(150000);
        ex++;
        if(ex>0)
        {
            num1++;
            flag1=0;
            printf("There is possible problem with GC3 no RAW files have been\n");
            printf("transferred in 7.5 minutes will try next file number\n");
            goto bad_file_restart;
        }
        system("tran");
    }
    fclose(fp1);
}

fp1=fopen("tran.inp","w");
fprintf(fp1,"anonymous\n");
fprintf(fp1,"cd gc3\n");
fprintf(fp1,"delete %s\n",f_name);
if(raw_flag==1)
    fprintf(fp1,"delete %s\n",r_name);
fprintf(fp1,"quit\n");
fclose(fp1);

err= strip(f_name,tim,dat,&larea,&firt,&ldat,&ltim);

if(err!=-1)
{
    num1++;

    system("tran");

    flag1=0;
    fil=1;
    fibs=sizeof(FILESTATUS3);

    RC=DosOpen(f_name,
        &fh,&act,
        0,
        FILE_NORMAL,
        FILE_OPEN,
        OPEN_ACCESS_READONLY | OPEN_SHARE_DENYNONE,
        NULL);
    if(RC != 0)
        printf("ERROR1 %ld\n",RC);

    RC=DosQueryFileInfo(fh,fil,&fib,fibs);
    if(RC != 0)
        printf("ERROR2 %ld\n",RC);

    DosClose(fh);

    if(fib.cbFile<250)
    {
        printf("FILE %d WAS BAD\n",(num-1));
        strcpy(gc3str,"type ");
        strcat(gc3str,f_name);
        system(gc3str);
    }
    if(fib.cbFile>=250)
    {
        printf("FILE %d WAS BAD\n",(num-1));
        strcpy(gc3str,"type ");

```

```

        strcat(gc3str,f_name);
        system(gc3str);
    }

    goto bad_file_restart;
}

flag1=behind(lmat,&off,dl,tim);

cdat[0]=dat[7];
cdat[1]=dat[0];
cdat[2]=dat[1];
cdat[3]=dat[3];
cdat[4]=dat[4];
cdat[5]=0;

cre_name(cdat,c_name,dat_path);
printf("\nc_name:\t%s\n",c_name);

strcpy(comm3,dat_path);
strcat(comm3,cdat);
strcat(comm3,"\\");
strcat(comm3,c_name);

if(raw_flag==1)
{
    RC=DosCopy(r_name,comm3,1);
    if(RC!=0)
        printf("Dos Copy ERROR %ld in line 166\n",RC);

    while((fp1=fopen(comm3,"r"))==0)
    {
        RC=DosCopy(r_name,comm3,1);
        if(RC!=0)
            printf("Dos Copy ERROR %ld in line 172\n",RC);
            system("pause");
    }
    fclose(fp1);
}
system("tran");

RC=DosDelete(f_name);
if(RC!=0)
    printf("Dos Delete ERROR %ld in line 186\n",RC);

if(raw_flag==1)
{
    RC=DosDelete(r_name);
    if(RC!=0)
        printf("Dos Delete ERROR %ld in line 224\n",RC);
}

if(flag == -1)
    strcpy(pdat,cdat);

if(strcmp(cdat,pdat)!=0)
{
    strcpy(comm4,pdat);
    strcat(comm4,".pr3 ");

    strcpy(comm5,dat_path);
    strcat(comm5,pdat);
    strcat(comm5,"\\");
    strcat(comm5,comm4);

    RC=DosCopy(comm4,comm5,1);
    if(RC!=0)
        printf("Dos Move ERROR %ld in line 241",RC);

    while((fp1=fopen(comm5,"r"))==0)
    {
        RC=DosCopy(comm4,comm5,1);
        if(RC!=0)
            printf("Dos Copy ERROR %ld in line 247",RC);
    }
    fclose(fp1);
    RC=DosDelete(comm4);
}

```

```

    if(RC!=0)
        printf("Dos Copy ERROR %ld in line 252",RC);

    strcpy(pdat,cdat);
}

flag=0;
con=larea/fac;
strcpy(prn_file,cdat);
strcat(prn_file,"_pr3");
fp=fopen(prn_file,"a+");
fprintf(fp,"%s %s %ld %lf %lf %s\n",dat,tim,larea,frt,con,c_name);
fclose(fp);

while((fp=fopen("temp.prc","w"))==0)
{
    RC=DosSleep(5000);
    if(RC!=0)
        printf("Dos Sleep ERROR %ld in line 228",RC);
}

fprintf(fp,"%s\n%ld\n%lf\n%lf\n%s",tim,larea,frt,con,c_name);
fclose(fp);
time(&time2);
printf("\nTRANSFER COMPLETED IN %ld SECONDS\n\n",(time2-time1));
}

}

int behind(long ldat, long *off, int dl, char tim[])
{
    char str_tim[25],yearf[3],dayf[4],moni[4],monf[4];
    char timef[9],datf[9],timec1[3],timec2[3],timec3[3];
    char timel1[3],timel2[3],timel3[3];
    struct tm *newtime;
    time_t ltime;
    long cdat,ctim,ctim1,ctim2,ctim3,ltim1,ltim2,ltim3,ltim;

    time(&ltime);
    newtime=localtime(&ltime);
    strcpy(str_tim,asctime(newtime));

    moni[0]=str_tim[4];
    moni[1]=str_tim[5];
    moni[2]=str_tim[6];
    moni[3]=0;
    monf[3]=0;

    if(strcmp(moni,"Jan")==0)
        strcpy(monf,"01");
    if(strcmp(moni,"Feb")==0)
        strcpy(monf,"02");
    if(strcmp(moni,"Mar")==0)
        strcpy(monf,"03");
    if(strcmp(moni,"Apr")==0)
        strcpy(monf,"04");
    if(strcmp(moni,"May")==0)
        strcpy(monf,"05");
    if(strcmp(moni,"Jun")==0)
        strcpy(monf,"06");
    if(strcmp(moni,"Jul")==0)
        strcpy(monf,"07");
    if(strcmp(moni,"Aug")==0)
        strcpy(monf,"08");
    if(strcmp(moni,"Sep")==0)
        strcpy(monf,"09");
    if(strcmp(moni,"Oct")==0)
        strcpy(monf,"10");
    if(strcmp(moni,"Nov")==0)
        strcpy(monf,"11");
    if(strcmp(moni,"Dec")==0)
        strcpy(monf,"12");

    dayf[0]=str_tim[8];
    dayf[1]=str_tim[9];

```

```

dayf[2]=47;
dayf[3]=0;
if(dayf[0]==32)
    dayf[0]=48;

yearf[0]=str_tim[22];
yearf[1]=str_tim[23];
yearf[2]=0;

datf[0]=monf[0];
datf[1]=monf[1];
datf[2]=dayf[0];
datf[3]=dayf[1];
datf[4]=yearf[0];
datf[5]=yearf[1];
datf[6]=0;

timec1[0]=str_tim[11];
timec1[1]=str_tim[12];
timec1[2]=0;

timec2[0]=str_tim[14];
timec2[1]=str_tim[15];
timec2[2]=0;

timec3[0]=str_tim[17];
timec3[1]=str_tim[18];
timec3[2]=0;

timel1[0]=tim[0];
timel1[1]=tim[1];
timel1[2]=0;

timel2[0]=tim[3];
timel2[1]=tim[4];
timel2[2]=0;

timel3[0]=tim[6];
timel3[1]=tim[7];
timel3[2]=0;

ctim1=atol(timec1)*3600;
ctim2=atol(timec2)*60;
ctim3=atol(timec3);
ctim=ctim1+ctim2+ctim3;

ltim1=atol(timel1)*3600;
ltim2=atol(timel2)*60;
ltim3=atol(timel3);
ltim=ltim1+ltim2+ltim3;

cdat=atol(datf);
printf("LTIM:\t%d\nCTIM:\t%d\nDIFF:\t%d\n",ltim,ctim,(ctim-ltim));
if((ctim-ltim)<150&&ldat==cdat)
{
    *off=(150-(ctim-ltim))*1000;
    return(-1);
}

*off=0;

if(ltim<(ctim-dl) && ldat==cdat)
{
    /*printf("line 323\n");*/
    return(1);
}

if(ltim>ctim && ldat < cdat)
{
    if(((86399-ltim)+ctim)>dl)
    {

```

```
        printf("line 331\n");
        return(1);
    }
    return(0);
}

if(ldat < cdat)
{
    /*printf("line 338\n");*/
    return(1);
}

return(0);
}

void handler(int sig)
{
    printf("EXITING PROGRAM");
    DosCloseMutexSem(semh);
    fclose(fp);
    exit(0);
}
```

SOURCE LISTING : FIL_NAME Ver. 1.0

```

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

void fil_name(char f_name[],char r_name[])
{
    extern num;
    char cnum[61];
    int i;
    FILE *fp1;

    f_name[0]=71;
    f_name[1]=72;
    f_name[8]=46;
    f_name[9]=82;
    f_name[10]=80;
    f_name[11]=84;

    fp1=fopen("itoa","w");
    fprintf(fp1,"%d",num);
    fclose(fp1);
    fp1=fopen("itoa","r");
    fscanf(fp1,"%s",cnum);
    fclose(fp1);

    if(num<=9)
    {
        f_name[2]=48;
        f_name[3]=48;
        f_name[4]=48;
        f_name[5]=48;
        f_name[6]=48;
        f_name[7]=cnum[0];
    }
    if(num>9&&num<=99)
    {
        f_name[2]=48;
        f_name[3]=48;
        f_name[4]=48;
        f_name[5]=48;
        f_name[6]=cnum[0];
        f_name[7]=cnum[1];
    }
    if((num>99)&&(num<=999))
    {
        f_name[2]=48;
        f_name[3]=48;
        f_name[4]=48;
        f_name[5]=cnum[0];
        f_name[6]=cnum[1];
        f_name[7]=cnum[2];
    }
    if(num>999&&num<=9999)
    {
        f_name[2]=48;
        f_name[3]=48;
        f_name[4]=cnum[0];
        f_name[5]=cnum[1];
        f_name[6]=cnum[2];
        f_name[7]=cnum[3];
    }
    if(num>9999&&num<=99999)
    {
        f_name[2]=48;
        f_name[3]=cnum[0];
        f_name[4]=cnum[1];
        f_name[5]=cnum[2];
        f_name[6]=cnum[3];
        f_name[7]=cnum[4];
    }
    if(num>99999&&num<=999999)
    {
        f_name[2]=cnum[0];
        f_name[3]=cnum[1];

```

```
f_name[4]=cnum[2];  
f_name[5]=cnum[3];  
f_name[6]=cnum[4];  
f_name[7]=cnum[5];  
}  
  
strcpy(r_name,f_name);  
r_name[9]=82;  
r_name[10]=65;  
r_name[11]=87;  
  
num++;  
}
```


SOURCE LISTING : CRE_NAME Ver. 1.0

```

#define INCL_DOSFILEMGR

#include <stdio.h>
#include <fcntl.h>
#include <stdlib.h>
#include <string.h>
#include <os2.h>

void cre_name(char cdat[],char c_name[],char datapath[])
{
    extern num1;
    char cnum[4];
    static char cmpdat[6];
    UCHAR comm[50];
    APIRET RC;
    PEAOP2 EAbuf=0;

    int i=1;
    FILE *fp1;
    static int flag=-1;

    cmpdat[5]=0;
    i = (strcmp(cdat,cmpdat));
    if(i!=0)
    {
        strcpy(comm,datapath);
        strcat(comm,cdat);
        printf("%s\n",comm);
        RC=DosCreateDir(comm,EAbuf);
        if(RC != 0)
        {
            if(RC!=5)
                printf("\nDos Create Dir error %ld in line 29 of cre_name.c\n",RC);
        }
        strcpy(cmpdat,cdat);
        if (flag!=-1)
            num1=1;
        flag=0;
    }

    strcpy(c_name,cdat);
    fp1=fopen("itoea","w");
    fprintf(fp1,"%d",num1);
    fclose(fp1);
    fp1=fopen("itoea","r");
    fscanf(fp1,"%s",cnum);
    fclose(fp1);

    if(num1<=9)
    {
        c_name[5]=48;
        c_name[6]=48;
        c_name[7]=cnum[0];
    }
    if(num1>9&&num1<=99)
    {
        c_name[5]=48;
        c_name[6]=cnum[0];
        c_name[7]=cnum[1];
    }
    if((num1>99)&&(num1<=999))
    {
        c_name[5]=cnum[0];
        c_name[6]=cnum[1];
        c_name[7]=cnum[2];
    }

    c_name[8]=46;
    c_name[9]= 114;
    c_name[10]=97;
    c_name[11]=51;
    c_name[12]=0;

    num1++;
}

```

SOURCE LISTING : STRIP Ver. 1.0

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int strip(f_name, tim, dat, larea, frt, ldat, ltim)

char f_name[];
char tim[];
char dat[];
long *larea;
double *frt;
long *ldat;
long *ltim;
{
    int i;
    char carea[9],ldats[8],ltims[8];
    char crt[51];
    char stop,c[500];
    FILE *fp1;

    dat[8]=0;
    tim[8]=0;
    crt[4]=0;
    carea[8]=0;

    fp1=fopen(f_name,"r");
    while(fp1!=0)
    {
        fp1=fopen(f_name,"r");
    }

    for(i=0;i<=249;i++)
        fscanf(fp1,"%c",&c[i]);

    fclose(fp1);

    for(i=0;i<=7;i++)
        tim[i]=c[41+i];

    for(i=0;i<=7;i++)
        dat[i]=c[32+i];

    for(i=0;i<=3;i++)
        crt[i]=c[185+i];

    for(i=0;i<=7;i++)
        carea[i]=c[191+i];

    ldats[0] = dat[0];
    ldats[1] = dat[1];
    ldats[2] = dat[3];
    ldats[3] = dat[4];
    ldats[4] = dat[6];
    ldats[5] = dat[7];
    ldats[6] = 0;

    ltims[0] = tim[0];
    ltims[1] = tim[1];
    ltims[2] = tim[3];
    ltims[3] = tim[4];
    ltims[4] = tim[6];
    ltims[5] = tim[7];
    ltims[6] = 0;

    if((*ltim=atoi(ltims))==0)
    {
        printf("\nBad Time\n");
        return(-1);
    }

    if((*ldat=atoi(ldats))==0)
    {
        printf("\nBad Date\n");
        return(-1);
    }
}

```

```
if((*larea=atol(carea))==0)
{
    printf("\nBad Area\n");
    return(-1);
}

if((*frt=atof(crt))==0)
{
    printf("\nBad RT\n");
    return(-1);
}

return(0);
}
```

C.8 Black Box - Programmable Logic Controller Code

```
+-----+
:
: Allen-Bradley Co.
: Allen-Bradley SLC 5/01 CPU
: 1747 Series Software
: SLC500 Advanced Programming Software
: APS Release 3.2
: Documentation Utility
: Program Listing
:
+-----+
```

```
OFFLINE File: GCCAL3
```

Program Listing

Processor File: GCCAL3

Rung 2:0

Rung 2:0

OPEN COLLECTOR #1

```

: I:1.0                                0:2.0 :
:--]/[------( )--:
:      12                                4 :

```

Rung 2:1

OPEN COLLECTOR #2

```

: I:1.0                                0:2.0 :
:--]/[------( )--:
:      2                                5 :

```

Rung 2:2

OPEN COLLECTOR #3

```

: I:1.0                                0:2.0 :
:--]/[------( )--:
:      3                                9 :

```

Rung 2:3

OPEN COLLECTOR #4

```

: I:1.0                                0:2.0 :
:--]/[------( )--:
:      4                                10 :

```

Rung 2:4

OPEN COLLECTOR #5

```

: I:1.0                                0:2.0 :
:--]/[------( )--:
:      5                                11 :

```

Rung 2:5

OPEN COLLECTOR #6

```

: I:1.0                                0:2.0 :
:--]/[------( )--:
:      6                                12 :

```

Rung 2:6

START #1

```

: I:1.0                                B3 :
:--] [------(L)--:
:      9                                0 :

```

Rung 2:7

```

: I:1.0                                +TON-----+ :
:--] [-----+TIMER ON DELAY +- (EN)--:
:      7                                :Timer    T4:5+- (DN) :
:                                         :Time Base  0.01: :
:                                         :Preset      1: :
:                                         :Accum       0: :
:                                         +-----+ :

```

Rung 2:8

```

: T4:5                                +CTU-----+ :
:--] [-----+COUNT UP +- (CU)--:
:      TT                                :Counter   C5:0+- (DN) :
:                                         :Preset    289: :
:                                         :Accum     0: :
:                                         +-----+ :

```

Program Listing

Processor File: GCCAL3

Rung 2:9

```

Rung 2:9
: T4:5      C5:0      +CTU-----+
: --] [---+---] [---+-----+COUNT UP+- (CU)---:
:      TT      DN      :Counter      C5:1+- (DN) :
:      :      :Preset      7:      :
:      :      :Accum      0:      :
:      :      +-----+
:      B3
:      +---] [---+
:      0

```

```

Rung 2:10
: C5:1      C5:0
: --] [----- (RES)---:
:      DN

```

```

Rung 2:11
: C5:1      C5:1
: --] [----- (RES)---:
:      DN

```

```

Rung 2:12
: C5:1      B3
: --] [----- (U)---:
:      DN      0

```

```

Rung 2:13
: C5:1      B3
: --] [----- (U)---:
:      DN      3

```

```

Rung 2:14
: C5:1      B3
: --] [----- (L)---:
:      CU      3

```

```

Rung 2:15
: B3      0:2.0
: --] [----- ( )---:
:      3      0

```

```

Rung 2:16
: B3      0:2.0
: --] [----- ( )---:
:      3      1

```

```

Rung 2:17
START #2
: I:1.0      B3
: --] [----- (L)---:
:      10      1

```

Program Listing

Processor File: GCCAL3

Rung 2:16

Rung 2:18

```

: I:1.0                                     +TON-----+ :
:--] [-----+-----+TIMER ON DELAY +--(EN)-- :
:      8                                     :Timer    T4:6+--(DN) :
:                                           :Time Base  0.01: :
:                                           :Preset      1: :
:                                           :Accum       0: :
:-----+-----+ :

```

Rung 2:19

```

: T4:6                                     +CTU-----+ :
:--] [-----+-----+COUNT UP      +--(CU)-- :
:      TT                                     :Counter    C5:5+--(DN) :
:                                           :Preset     139: :
:                                           :Accum      77: :
:-----+-----+ :

```

Rung 2:20

```

: T4:6  C5:5                             +CTU-----+ :
:--] [---+---] [---+-----+COUNT UP    +--(CU)-- :
:      TT      DN                                     :Counter    C5:6+--(DN) :
:                                           :Preset      7: :
:                                           :Accum       0: :
:-----+-----+ :
:      B3 :
:      +---] [---+ :
:      1 :

```

Rung 2:21

```

: C5:6                                     B3 :
:--] [-----+-----+-----+-----+ (U)--- :
:      DN                                     1 :

```

Rung 2:22

```

: C5:6                                     B3 :
:--] [-----+-----+-----+-----+ (U)--- :
:      DN                                     4 :

```

Rung 2:23

```

: C5:6                                     C5.5 :
:--] [-----+-----+-----+-----+ (RES)--- :
:      DN                                     :

```

Rung 2:24

```

: C5:6                                     C5.6 :
:--] [-----+-----+-----+-----+ (RES)--- :
:      DN                                     :

```

Rung 2:25

```

: C5:6                                     B3 :
:--] [-----+-----+-----+-----+ (U)--- :
:      DN                                     4 :

```

Rung 2:26

```

: C5.6                                     B3 :
:--] [-----+-----+-----+-----+ (L)--- :
:      CU                                     4 :

```

Program Listing

Processor File: GCCAL3

Rung 2:27

```

Rung 2:27
: B3                                     0:2.0 :
:--] [-----] ( )--
:      4                                     2 :

```

Rung 2:28

```

START #3
: B3                                     0:2.0 :
:--] [-----] ( )--
:      4                                     3 :

```

Rung 2:29

```

: I:1.0
:--] [-----] B3 :
:      11         (L)--
:                                     2 :

```

Rung 2:30

```

: I:1.0
:--] [-----] +TON-----+ :
:      0         +TIMER ON DELAY +-(EN)-- :
:                                     :Timer T4:7+-(DN) :
:                                     :Time Base 0.01: :
:                                     :Preset 1: :
:                                     :Accum 0: :
:                                     +-----+ :

```

Rung 2:31

```

: T4:7
:--] [-----] +CTU-----+ :
:      TT         +COUNT UP +-(CU)-- :
:                                     :Counter C5:7+-(DN) :
:                                     :Preset 570: :
:                                     :Accum 294: :
:                                     +-----+ :

```

Rung 2:32

```

: T4:7 C5:7
:--] [---+---] [---+ :
:      TT      DN :
:                                     +CTU-----+ :
:                                     +COUNT UP +-(CU)-- :
:                                     :Counter C5:8+-(DN) :
:                                     :Preset 8: :
:                                     :Accum 0: :
:                                     +-----+ :
:                                     :
:      B3 :
:      +---] [---+ :
:      2 :

```

Rung 2:33

```

: C5:8
:--] [-----] C5.7 :
:      DN         (RES)--

```

Rung 2:34

```

: C5:8
:--] [-----] B3 :
:      DN         (U)--
:                                     2 :

```

Rung 2:35

```

: C5:8
:--] [-----] B3 :
:      DN         (U)--
:                                     5 :

```


Program Listing

Processor File: GCCAL3

Rung 2:36

```

Rung 2:36
: C5:8                                     B3 :
:--] [----- (U)-----:
:   DN                                     6 :

```

```

Rung 2:37
: C5.8                                     C5.8 :
:--] [----- (RES)-----:
:   DN                                     :

```

```

Rung 2:38
: C5.8                                     B3 :
:--] [----- (L)-----:
:   CU                                     6 :

```

```

Rung 2:39
: B3                                     +TON-----+ :
:--] [-----+TIMER ON DELAY +- (EN)---:
:   6                                     :Timer      T4:0+- (DN) :
:                                     :Time Base   0.01: :
:                                     :Preset      200: :
:                                     :Accum        0: :
:                                     +-----+ :

```

```

Rung 2:40
: B3                                     +TON-----+ :
:--] [-----+TIMER ON DELAY +- (EN)---:
:   6                                     :Timer      T4:1+- (DN) :
:                                     :Time Base   0.01: :
:                                     :Preset      10: :
:                                     :Accum        0: :
:                                     +-----+ :

```

```

Rung 2:41
: B3                                     +TON-----+ :
:--]/[-----+TIMER ON DELAY +- (EN)---:
:   6                                     :Timer      T4:2+- (DN) :
:                                     :Time Base   0.01: :
:                                     :Preset      10: :
:                                     :Accum       10: :
:                                     +-----+ :

```

```

Rung 2:42
: B3                                     +TON-----+ :
:--]/[-----+TIMER ON DELAY +- (EN)---:
:   6                                     :Timer      T4:3+- (DN) :
:                                     :Time Base   0.01: :
:                                     :Preset      200: :
:                                     :Accum      200: :
:                                     +-----+ :

```

```

Rung 2:43
: C5.8                                     B3 :
:--] [----- (L)-----:
:   CU                                     5 :

```

Program Listing

Processor File: GCCAL3

Rung 2:44

Rung 2:44
: C5.8 : B3 :
:--] [----- ()-- :
: 5 : 8 :
:

Rung 2:45
: T4.3 T4.0 : 0:2.0 :
:--+]/[--+]/[----- ()-- :
: : TT : TT : 6 :
: : T4.2 : : :
: +--+]/[--+ : : :
: : TT : : :
:

Rung 2:46
: B3 : 0:2.0 :
:--+]/[----- ()-- :
: : 5 : 7 :
: : T4.1 : : :
: +--+]/[--+ : : :
: : DN : : :
:

Rung 2:47
:
:-----+ END +----- :
:

Program Listing

Processor File: GCCAL3

Summary

REPORT OPTIONS SUMMARY

Insure Valid X-Ref Info:	YES
Graphics Mode:	No
Page Width:	80
Page Length:	66
Starting File:	2
Ending File:	2
Power Rail:	YES
Address Comments:	NO
Address Display:	NO
Rung Comments:	YES
Ladder Cross Reference:	NONE

Los Alamos ~~APPENDIX D - FTIR CALIBRATION DESCRIPTIONS~~

Los Alamos National Laboratory

Los Alamos, New Mexico 87545

memorandum

To: R. Baucz/Westinghouse Hanford Date: May 7, 1994

From: John R. Stephens/CST-2 *JS* MS/Phone: J 565/7-7363

Symbol: CST-2:94

Subject: Summary of Calibrations for FTIR on tank 101SY

INTRODUCTION

This report updates and summarizes the calibration history of the Bio-Rad Transform Infrared Spectrometer (FTIR), used to monitor nitrous oxide (N_2O) and ammonia gases (NH_3) on Hanford waste tank 101SY. Included are a brief description of the FTIR instrument as implemented at Hanford, a summary of the original purpose of the instrument and validation procedures, and a history of calibration changes for the system. Other memos concerning the calibration of the FTIR were issued on 8/3/93, 10/28/93, and 2/22/94 when changes were made in the FTIR calibration factors. This document is part of the Los Alamos and Westinghouse/Hanford program to verify and recalculate all of the FTIR data obtained on tank 101SY and provide this data to users in a convenient form. This report covers the calibration history of the FTIR. Documentation of the software used to calculate the reported concentrations of NH_3 and N_2O and analysis of the accuracy and precision of the FTIR on tank 101SY, including the acceptance tests, will be discussed in other documents.

INSTRUMENT DESCRIPTION

A Bio-Rad Fourier Transform Infrared Spectrometer (FTIR), Model FTS-60 is used to monitor N_2O and NH_3 on tank 101SY. The FTIR is operated by a Bio-Rad SPC 3200 Data Station running the Iris operating system, a Unix look-alike. The instrument system was developed by Herb Fry and Cliff Hewitt, of CST-2 in conjunction with personnel at Hanford. The FTIR is presently operating on tank 101SY under the day-to-day supervision of Westinghouse and EG&G personnel. The FTIR is connected via Ethernet and RS-232 computer links to an IBM-PC computer (Kinetics) located in the DACS trailer.

The Kinetics computer queries each instrument, determines if new files are present, transfers the raw data files via the Ethernet link, analyzes the analytical data to calculate gas concentrations, and sends concentration data to the DACS computer via an RS-232 link. It should be noted that the entire FTIR spectrum is stored which has allowed us to post-calibrate the FTIR data.

The FTIR was originally installed on tank 101SY to monitor baseline concentrations (< 100 ppm) of NH_3 and N_2O gases during intervals between "burp" events to establish a baseline for comparison to gas concentrations occurring during mixing pump insertion and operation. Subsequently, the FTIR has provided data on NH_3 and N_2O concentrations during burp event I (June 26, 1993) and subsequent times, which included concentrations up to percent levels.

The FTIR is installed in the Gas Monitoring System 2 (GMS-2) shack above the 101SY tank. Gas from the tank header space is drawn through a heated $1/2$ " diameter stainless tube into the (GMS-2) shack using a sampling pump with a $1 \text{ ft}^3/\text{min}$ flow rate. The entire gas sample travels through a multipath (5 m) gas cell in the FTIR and exhausts back into the vent header. The high sample flow rate decreases the response time of the FTIR to rapid tank events, such as a burp or pump-induced gas release. The FTIR on tank 101SY is currently run on a 64 scan (~ 7.5 minute) cycle. Shorter cycle times would result in a faster time response but would sacrifice signal to noise in the resulting spectra.

CALIBRATION HISTORY

Calibration prior to and during event I (June 26, 1993)

The major task in recalibrating the FTIR is to extend the range of the FTIR analysis from 100 ppm as originally developed to concentrations up to about 5% levels that are seen during a burp and pump operations. Originally, to monitor NH_3 and N_2O below 100 ppm a group of lines was chosen for analysis that exhibited strong absorption in the low ppm range. For NH_3 a group of lines at 965 cm^{-1} were used. N_2O analysis used a line group at 2200 cm^{-1} . These line groups, as well as all line groups used for calibrations, are shown in Table 1. These lines provide good accuracy and precision in this concentration range using a linear relationship between peak area and concentration. The calibration factors for all

Table 1 Spectral Areas used for FTIR Calibrations (cm^{-1})

NH ₃		lower baseline	upper baseline	peak area
	965	954 -	973 -	963 -
	area	956	975	968
	1158	1156-	1160-	1158-
	area	1157	1161	1159
	1212	1208-	1214-	1211-
	area	1210	1216	1214
N ₂ O				
	2200	2154-	2263-	2195-
	area	2159	2268	2243
	2509	2504-	2622-	2509-
	area	2507	2625	2621
	2797	2790-	2802-	2794-
	area	2793	2803	2799

2243

.315
1051 old

.003

- .04

TABLE 2 - N₂O Calibration Factors

Date Installed	Line (cm ⁻¹)	Conc. Range (ppm)	A*	B	C
11/6/92	2200	All	0	4.48	0
8/2/93	2200	0 - 100	0	4.48	0
	2797	> 100	0	3142	0
2/26/94	2200	0 - 80	-0.114	5.44	0.140
	2200	80 - 671	-25.0	8.52	0.063
	2509	671 - 22641	-140.8	229.5	1.87
	2797	> 22641	1180.3	3533	2041

* Concentration (ppm) = A + B(peak area) + C(peak area²)
 [Peak areas in Abs/cm-1]

TABLE 3 - ~~N₂O~~ Calibration Factors

11/2/92	965	All	0	56.9	0
8/2/93	965	0 - 100	0	56.9	0
	1212	> 100	0	5727	0
10/26/93	965	0 - 500	0	59.56	10.602
	1158	500 - 3000	111.67	177.71	6022.5
	1212	> 3000	-2333	13961	10428
2/26/94	965	0 - 178	0.0	71.7	4.09
	965	178 - 565	1.44	57.89	9.92
	1158	565 - 13488	852	-3673	9665
	1212	> 13488	-819	14034	10780

* Concentration (ppm) = A + B(peak area) + C(peak area²)
 [Peak areas in Abs/cm-1]

Table 4 - Results of GMS2 Qualification Tests
January 19 and 20, 1993*

Name of Standard	Calculated concentration (ppm)	Measured Concentration (ppm)
N ₂ O mix one	31.6	29.7
N ₂ O mix two	64.3	49.9
N ₂ O standard	26	26.5
NH ₃ mix one	19.2	18.0
NH ₃ mix two	39.7	37.3
NH ₃ standard	30	28.1

* Extracted from Herb Fry's Qualification Test Report

Figure 1

Figure 2

Figure 4

Figure 5

Figure 6

line groups are shown in Tables 2 and 3. Validation of the calibrations based on these factors was carried out using the FTIR on tank 101SY on January 19 and 20, 1993 using two gas standards prepared by mixing and one certified gas standard for each gas. The results are shown in Table 4.

Calculation of NH₃ and N₂O concentrations during event I burp

In anticipation of burp event I high concentration (5%) gas standards of N₂O and NH₃ were measured using the FTIR on tank 101SY prior to the event to allow estimation of the peak concentrations of these gases during the event. After event I Herb Fry provided a "first look" of the peak gas concentrations for the NH₃ and N₂O gases based on the 5% gas standards calibration. The results were reported for five FTIR measurements during the burp event based on a linear relationship between the peak area of the 1212 cm⁻¹ peak for NH₃ and 2797 cm⁻¹ peak for N₂O. The peak areas in each case were ratioed to the 5% standards to determine the concentration in each analysis. The results of this analysis were reported in a memo by Herb Fry to Jack Lentsch on June 30, 1994.

Subsequent to Herb Fry's quick look at NH₃ and N₂O concentrations during event I, John Stephens re-analyzed the concentrations for event I. The detailed procedure and results were delivered in a memo to R. Bauer at Hanford on July 9, 1993. For N₂O below 100 ppm the peak group at 2200 cm⁻¹ was employed using the linear calibration factor listed in table 2. Above 100 ppm a quadratic calibration factor based on calculated factors provided by a vendor (IR Analysis, Anaheim, CA) were used. The non-linear calibration factors were:

$$\text{Concentration (ppm)} = 2691(\text{area}^2) + 5406(\text{areea}) + 1.41$$

over the peak area 2799.4 to 2794.1 cm⁻¹

For NH₃, analysis of concentrations during event I were calculated using the 965 cm⁻¹ peak group to 100 ppm. For ranges 100 ppm to 1000 ppm linear calibration factors were employed using the 1212 cm⁻¹ peak group relative to a 40 ppm NH₃ spectrum. Above this value the ratio of the peak area of the 1212 cm⁻¹ peak to the 5% NH₃ spectra was used to determine the concentration. The results of this analysis were discussed in a memo dated 7/9/93.

FTIR calibration methods and transfer of calibration to FTIR on 101SY

To provide more accurate on-line NH_3 and N_2O analyses it is necessary to perform calibrations using several peak groups, each optimized for a particular concentration range. Using several spectral peak groups is necessary because the areas of spectral peaks do not show a linear dependence with increasing concentration but falls off, a phenomena known as saturation. To provide accurate data from about 5 ppm to 5% requires using three peak areas for analysis. The concentration range over which each peak group can be used is increased by using non-linear (quadratic) calibration factors. This is the procedure we have followed in our calibration of the FTIR on tank 101SY.

Calibrations were performed using certified gas standards, provided by AIRCO Incorporated and using a Kin Tek Model 491 Precision Standards Generator (permeation system) that enabled us to deliver to the FTIR gas mixtures from 5 ppm to nearly 3% N_2O or NH_3 . In both cases, concentrations reported by the vendors were used for the calibrations. The calibrations, performed at Los Alamos, were compared to calibrations performed with the FTIR on tank 101SY and the Hanford gas monitoring laboratory and the data agreed within the precision and accuracy of the calibrations. This agreement allowed the calibrations, performed at Los Alamos, to be transferred to the FTIR instrument on tank 101SY.

To calibrate the FTIR at Los Alamos, a calibration system similar to the Hanford gas handling system on tank 101SY was set up at Los Alamos. The gas from a certified gas standard or permeation system flowed through the FTIR cell and spectra were taken using the same software used for on-line analysis at Hanford. During calibrations of the FTIR, the cell pressure and temperature were monitored. Calibrations were performed between 22 and 28 degrees C at a cell pressure of 645 torr corresponding to the FTIR cell pressure on tank 101SY. No temperature or pressure corrections were therefore required to convert the Los Alamos calibration data to conditions present during gas analysis on tank 101SY.

Calibration changes outlined in 8/3/94 memo

After pump insertion, the background levels of NH_3 and N_2O have been in the 100 ppm range with higher concentrations occurring during pump bumps and particularly during "pump induced mini burps". To meet the need for gas analysis between 100 ppm and about 1000 ppm calibrations for each gas were changed to analyze the original spectral peaks (965 cm^{-1} for NH_3 and 2200 cm^{-1} for N_2O) below 100 ppm but switch to the spectral peaks used for high concentration analysis (1212 cm^{-1} for NH_3 and 2797 cm^{-1} for N_2O) above 100 ppm. Linear calibration factors were still employed for all peaks. The calibration factors are listed in Table 2 and 3 for the spectral ranges listed in Table 1. These calibrations increased the accuracy of concentrations for the two gases, but traded off signal to noise, particularly for NH_3 , where the 1212 cm^{-1} peak exhibits low signal to noise ratio at concentrations near 100 ppm.

Calibration Changes outlined in 10/28/93 memo

Following further calibration measurements, a three peak, non-linear calibration scheme was developed for NH_3 and delivered to Hanford on 10/28/93. Three peaks are now used for analysis, the original peak at 965 cm^{-1} to about 500 ppm, a peak at 1153 cm^{-1} up to about 3000 ppm, and the 1212 cm^{-1} peak beyond this range. No change was made in the N_2O calibration factors at this time. Calibration factors and spectral ranges for analyzed peaks are listed in Tables 1, 2, and 3.

Calibrations delivered to Hanford on February 22, 1994

Final calibration factors were delivered to Hanford on February 22, 1994 for both NH_3 and N_2O shown in Tables 1, 2, and 3. For both gases the peak group used for the lowest concentration calculations was divided into two regions with a quadratic fit to each region. For NH_3 the crossover between regions is 178 ppm, for N_2O the crossover is 80 ppm. Switchover to other peak groups occur at 565 and 13488 ppm for NH_3 and 671 and 22641 for N_2O respectively.

Fits of concentration vs peak area for the two gases for each of the three peak groups are shown in Figures 1 - 6. Shown in each

plot are $\pm 10\%$ limits to the quadratic fits. Virtually all of the calibration points fall within the $\pm 10\%$ limits. In the figures, both Los Alamos calibration points using certified standard gases and gas mixtures produced by the permeation system fall within these limits, giving independent verification of the calibration scheme. Also shown are data derived from spectra taken at the Hanford gas monitoring laboratory. These points also fall within the limits shown.

SUMMARY

This report discusses the calibration history of the FTIR installed on Hanford waste tank 101SY through calibration procedures performed at Los Alamos and transferred to the Hanford FTIR spectrometer. Work is ongoing to perform acceptance tests for gas calibrations on the FTIR on tank 101SY. At that time a report will be written discussing the final accuracy and precision of the NH_3 and N_2O data delivered from that instrument. All of the gas concentration data will then be reanalyzed and made available to those interested in the results.

Distribution:

R. Bauer/Hanford/FAX 509-376-4945
J. N. Edwards/N-6/K-555
H. Fry/CLS-2/J-565
C. Hewitt/CLS-2/J-565
H. Hoida/N-1/E-340
N. Kirch/Hanford/ FAX 509-373-4095
J. W. Lentsch/Hanford/FAX 509-376-6521
M. McElroy/Hanford/FAX 509-373-5835
L. Steve Krugsrund/Hanford/FAX 509-372-3522

**THIS PAGE INTENTIONALLY
LEFT BLANK**