JUN 09 1997 (20) Sta. 37

ENGINEERING DATA TRANSMITTAL

Page 1 of 1

1. EDT 619510

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

15. DATA TRANSMITTED

(A) Item No.	(B) Document/Drawing No.	(C) Sheet No.	(D) Rev. No.	(E) Title or Description of Data Transmitted	(F) Approval Designator	(G) Reason for Transmittal	(H) Originator Disposition	(I) Receiver Disposition
1	HNF-SD-WM-CSWD-084		O	Computer Systems and Software Description for Standard-E+ Hydrogen Monitoring System (SHMS-E+)	SQ	1	1	

16. KEY

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

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

(G) Reason	(H) Disp.	(J) Name (K) Signature (L) Date (M) MSIN	(G) Reason	(H) Disp.	(J) Name (K) Signature (L) Date (M) MSIN
		Design Authority	4	6	Other T.C. Schneider 5-6-97
1	1	Design Agent D.D.Tate 5/5/97			
1	2	Cog.Eng. M.F. Erhart 5/28/97			
4	4	Cog. Mgr. R.L. Schlosser 5/29/97			
1	1	QA L.D. Salsbury 5/6/97			
1	1	Safety S.U. Zaman 5/8/97			
		Env.			

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

BD-7400-172-2 (05/96) GEF097

COMPUTER SYSTEMS AND SOFTWARE DESCRIPTION FOR STANDARD-E+ HYDROGEN MONITORING SYSTEM (SHMS-E+)

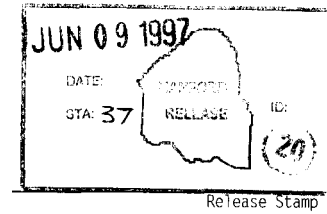
C.V. Vo
SGN Eurisys Serivices Corporation, Richland, WA 99352
U.S. Department of Energy Contract DE-AC06-96RL13200

EDT/ECN: No. 619510 UC: 2030
Org Code: S1200 Charge Code: N2022
B&R Code: EW3120072 Total Pages: 112

Key Words: Standard Hydrogen Monitoring System (SHMS), computer systems

Abstract:

Computer system design layout description for the Standard-E+ Hydrogen Monitoring System (SHMS-E+).



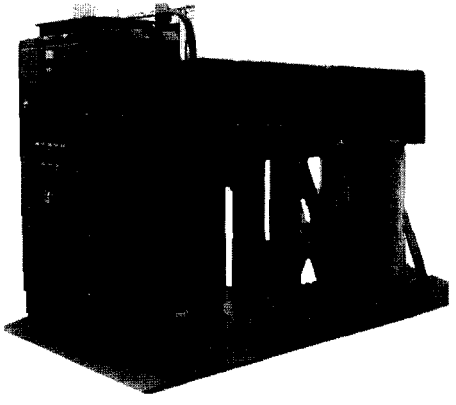
James Bishop 6-9-97
Release Approval Date

Release Stamp

Approved for Public Release

COMPUTER SYSTEMS AND SOFTWARE DESCRIPTION FOR STANDARD-E + HYDROGEN MONITORING SYSTEM (SHMS-E +)

(HNF-SD-WM-CSWD-084)



Prepared by

D. D. Tate
C. V. Vo
&
B. L. Philipp

May 1997

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

CONTENTS

LIST OF FIGURES

COMPUTER SYSTEMS AND SOFTWARE DESCRIPTION FOR STANDARD-E+ HYDROGEN MONITORING SYSTEM (SHMS-E+)

1.0 INTRODUCTION

The primary function of the Standard-E+ Hydrogen Monitoring System (SHMS-E+) is to determine tank vapor space gas composition and gas release rate, and to detect gas release events. Characterization of the gas composition is needed for safety analyses. The lower flammability limit, as well as the peak burn temperature and pressure, are dependent upon the gas composition. If there is little or no knowledge about the gas composition, safety analyses utilize compositions that yield the worst case in a deflagration or detonation. Knowledge of the true composition could lead to reductions in the assumptions and therefore there may be a potential for a reduction in controls and work restrictions. Also, knowledge of the actual composition will be required information for the analysis that is needed to remove tanks from the Watch List. Similarly, the rate of generation and release of gases is required information for performing safety analyses, developing controls, designing equipment, and closing safety issues.

To determine release rate, both the gas concentrations and the dome space ventilation rates (exhauster flow rate or passive dome/atmosphere exchange rate) are needed. Therefore, a flexible gas continuous monitoring system is needed that can be expanded to measure gas compositions at both high and low sensitivities. For these reasons, a modified version of the SHMS (entitled SHMS-E or -E+) has been developed. The SHMS-E is similar to the current SHMS-C system, with modular, expandable characterization capabilities. The SHMS-E+ measures gas concentrations in selectable ranges, nominally: Hydrogen (3-3,000 ppm and 0.2 - 10 Vol.%); Nitrous Oxide (10-4,000 ppm); Ammonia (10-10,000 ppm): and Methane (10-4,000 ppm). The SHMS-E has the same basic cabinet, wiring, tubing, and layout design as SHMS-E+. The SHMS-E will monitor Hydrogen using electro-chemical cells to monitor hydrogen nominally (0.2 - 10 Vol.%) like a basic SHMS, but will not have the gas chromatograph, the photoacoustic infrared monitor, and associated computers installed though they may be installed in the future if needed, as "plug-in" features.

The Standard-E+ Hydrogen Monitoring System Computer System consists of several individual computer components necessary to address the control and data acquisition functions of the several specific types of instruments used to monitor the vapor space. These computers are networked, using the Microsoft Windows 95 multi-tasking operating system and networking software. This allows the systems components to function in parallel and perform instrument control, data collection, analysis, computer interfacing, and archiving functions.

1.1 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, methane, and ammonia. A list of tanks with the potential for approaching the lower flammability limit (LFL) was created, and called the flammable gas watch-list tanks. Waste tank 241-SY-101 was identified as presenting the most serious safety concern, which prompted the design of the Standard Hydrogen Monitoring System (SHMS) for continuous hydrogen monitoring. The original SHMS design was implemented in 1992 in support of the Hydrogen Mitigation Project.

1.2 Site Environment

The environment to which the Standard-E+ Hydrogen Monitoring System (SHMS-E+) computer systems will be exposed is classified as an outdoor environment. All environmentally sensitive monitoring components shall be protected by the appropriate enclosure. Although the equipment will be protected by enclosed shelter, 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 up to 130 kilometers per hour (80 mph) with blowing dust.

2.0 SCOPE

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

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

3.0 SOFTWARE REQUIREMENTS SPECIFICATION

The SHMS-E+ requires computers to control and sequence the many instruments, sensors, and electric valves. Additionally, the computers must perform mathematical functions (ie. integrate the area under the curve) to produce "Calculated Concentrations". Control and sequencing of the valves and instruments, with these microcomputers, requires not only vendor supplied software, but also configuration, automation, and integration programming.

3.1 Criteria

This section will address the requirements, functions, inputs, outputs, format and desirables associated with the full SHMS-E+ product. The following criteria was established as necessary in relation to the SHMS-E+ software: SHMS-E Basic Design Features: Digital data logging plus TMACS connection, and SHMS-E+ Additional Design Features: Network data transmission/archiving capabilities. The sample gas analytical system will provide local display of analyzed sample gases, as well as both local and remote data logging capabilities through a high-speed data link.

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

- **H_2** @ 3 to 3,000 ppm ($\pm$3ppm)
  H_2 @ 0.2 to 10 Vol.% ($\pm$10% of reading)
- **NH_3** @ 10 to 10,000 ppm ($\pm$10% of reading)
- **N_2O** @ 10 to 4,000 ppm ($\pm$10% of reading)
- **CH_4** @ 10 to 4,000 ppm ($\pm$10% of reading)

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

3.1.3 The system CPU shall be network connected for:

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

3.1.4 The Gas Chromatograph shall sample at a variable rate that is normally set to 6 minutes.

3.1.5 The Photoacoustic Infrared Spectrometer shall sample at a variable rate that is normally set based on the length of the sample line.

3.1.6 The computer operating systems shall be compatible with the site standard and shall provide for preemptive multi-tasking. Windows[1] 95 is recommended. The software shall be Windows 95 compatible.

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

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

3.1.9 The Electrochemical Cell's data shall be logged at a variable rate that is normally set to 1 minute.

[1] Windows is a trademark of Microsoft Corporation.

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

3.1.11 The host computer (local to the instruments) shall preserve un-archived data for a minimum of one week, in the event of a failure of the central archival system or related telecommunications.

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

3.2 Software Functions

The SHMS-E+ data system provides the control and data acquisition functions for all instruments housed in the SHMS-E+ cabinet, and consists mainly of one Host computer and one Gas Chromatograph computer. In this document, the Host computer and the GC computer will be referred to as (*-HOST) and (*-GC), respectively.

Note: Several SHMS-E+ systems will be installed in different flammable gas watch list tanks. Since each SHMS-E+ contains one Host and one GC computers, these SHMS-E+ system network computer designators are uniquely defined as following:

HOST COMPUTER	XX-05-N-HOST
GC COMPUTER	XX-05-N-GC

where: XX is Tank Farm Identifier
05 SHMS system designator
N is Tank Number (i.e. 101=A 102=B 104=C etc..)
HOST is for Host Computer in the cabinet
GC is for GC Computer in the cabinet

The Host computer's functions are to interface and acquire data from the Bruel&Kjaer Multigas instrument (photo acoustic spectrometer), currently Ammonia content in the tank vapor space. It also provides data collection for two Hydrogen Whittaker[2] sensors and several other process instruments. The GC computer provides control and data acquisition for the gas chromatograph instrument currently used to measure Hydrogen, Nitrous Oxide, and Methane gases in the same sample gas stream.

Both computers, the Host and the GC computer, are networked with HLAN using the Microsoft Windows 95 multi-tasking operating system. This computer network system allows the components to function in parallel to control the analytical instruments, sensors, and electric valves. Additionally, the computers must perform mathematical functions (i.e. integrate the area under the curve) to produce "Calculated Concentrations". Controlling the instruments with these computers requires not only vendor supplied software, but also configuration, automation, and integration software.

[2] Whittaker is a trademark of the Whittaker Corporation.

Integration software consists of several systems. The SHMS-E+ Virtual Instrument (SHMS-E+ VI) provides process data acquisition and control of the automatic calibration feature for the gas chromatograph and the B&K Multi gas instruments. The GC software system provides an automated process for data configuration and handling. The B&K Virtual Instrument (B&K VI) software system allows the B&K data to transfer to the remote Host computer. The B&K VI is operated at the remote Host computer.

3.2.1 SHMS-E+ VIRTUAL INSTRUMENT

3.2.1.1 Operator Interface

- Provide real time display on SHMS-E+ front panel for both Whittaker hydrogen sensors, pressure and system flow.
- Interface with the B&K Multigas and the MTI gas chromatograph data systems to provide real time display on SHMS-E+ front panel.
- Accept control inputs from front panel to log data for Whittaker hydrogen sensors, system pressure, system flow and Whittaker calibration signal. Details to these control function inputs will be described in section 3.2.1.2.
- Accept the B&K and MTI GC auto verification input parameters. Details to these control inputs will be described in section 3.2.1.4 and 3.2.1.5.

3.2.1.2 Record and Archive Data

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

  - Complete archive network path.
  - Data frequency.
  - Toggle button to Save or Not Save to file.

- The data file will be automatically generated once at the beginning of each day and uniquely named using a date stamp.
- At the beginning of each day, the data file of the previous day will be archived across the HLAN per the input network archive path.

3.2.1.3 Data Handling During HLAN Network Outage

- Data will be preserved by saving locally.
- Provide adequate data storage for a minimum of one week.

3.2.1.4 B&K Calibration Verification Control

- Accept calibration verification control inputs on front panel as follows to initiate cal verification sequences:

- Time start
- Duration

- Provide signals to control electric valves.

- Interface with B&K Virtual Instrument to identify calibration verification samples in the report file.

3.2.1.5 MTI GC Calibration verification Control

- Accept calibration verification control inputs on front panel for each channel as follows:

  - Time start
  - Duration

- Provide signals to control electric valves for each channel.

- Interface with GC data system to identify calibration verification samples in the data report file.

3.2.2 B&K MULTI-GAS VIRTUAL INSTRUMENT

The B&K Virtual Instrument (B&K VI) software primary function is to provide the data communication interface between the Host computer and the B&K instrument.

3.2.2.1 Operator Interface

- Provide real time display on front panel for Ammonia gas concentration.

- Accept control input from front panel for down load data to 3.5" diskette. Front panel also displays the status during this process.

- Base on the input network archived path and with archive enable, the B&K data file that generated from the previous day will be archived across the HLAN to a server computer at the beginning of each day.

- Accept control input on the front panel at the remote Host computer to either run or stop the B&K instrument that sample in continuous mode.

3.2.2.2 Record B&K Data

- New data file is generated once at the beginning of a new day and uniquely named using a date stamp.

- Subsequent data records will be appended to the current data file following instrument restart or outage.

3.2.2.3 Data Handling During HLAN Network Outage

- System stores archive data at Host computer in the event of a network failure or interruption.

3.2.2.4 Interface with SHMS-E+ Virtual Instrument

- The Ammonia concentration data from the B&K data system passes to and displays in the LabView SHMS-E+ Virtual Instrument.

- Calibration samples in the B&K data file are flagged based upon the electric valve operation that is controlled from the SHMS-E+ VI.

3.2.3 GAS CHROMATOGRAPH DATA SYSTEM

The GC data system consists of a dual channel MTI gas chromatograph that is controlled by the GC computer. The GC system is networked with the Host computer and HLAN. The MTI GC channel A is set up and configured to measure hydrogen gas, and channel B is set up and configured to measure methane and nitrous-oxide gases.

3.2.3.1 EZChrom 200 Data System

- Vendor supplied software, EZChrom 200, controls and acquires data from the MTI Gas Chromatograph.

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

3.2.3.2 Organize and Format EZChrom Data

- Custom software (GCAPP32.EXE) collects and organizes EZChrom data.
  - Organized by date
  - Organized by run or index number

3.2.3.3 Data Handling Following System Outage or Date Change

- Software automatically indexes filenames following a system outage.

- Software automatically generates new filenames at midnight when the date changes.

3.2.3.4 Interface with SHMS-E+ Virtual Instrument

- Data from GC is passed to and displayed in Labview.[3]

- Calibration data is flagged based upon external valve operation initiated from Labview SHMS-E+ Virtual Instrument.

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

3.2.3.5 Data Compression and HLAN Transfer

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

3.2.3.6 Data Handling During HLAN Network Outage

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

4.0 DESIGN DESCRIPTION & SYSTEM INTERFACES

4.1 Principles of Operation

The general organization of the SHMS-E+ computer system components are shown in the SHMS-E+ Computer System Layout Diagram (Figure 4-A). An individual processor (*-GC) is dedicated to the Gas Chromatograph (GC), as recommended by the MTI GC equipment vendor. This dedicated CPU runs software supplied with the GC instrument. Another CPU (*-HOST) is responsible for collecting the data from the *-GC computer and from the B&K photoacoustic spectrometer as well as reading the analog sensors (electrochemical cells, flow, pressure, temperature). The *-HOST reduces and compresses the data, then passes the calculated results with source data via HLAN to a central computer that shares the data real time to the networked data users and archives the data to CD-ROM.

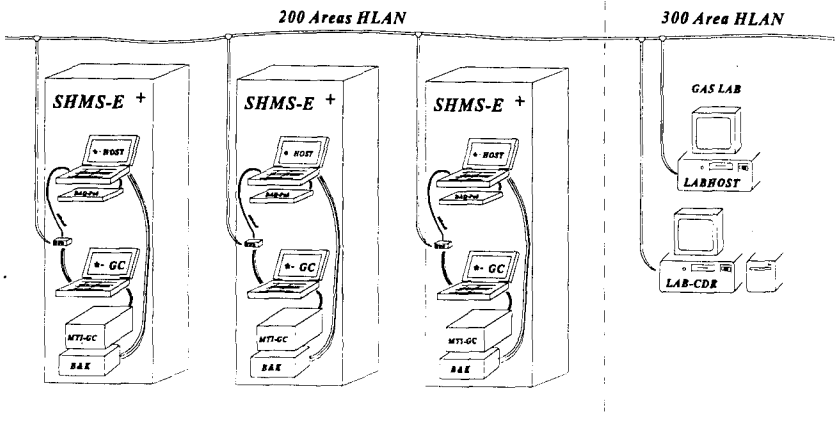


Figure 4A -- Computer System Layout Diagram

4.2 Computers & Software

This subsection defines the hardware and software composition of each of the two computers in each SHMS-E+ field unit.

4.2.1 GC Computer

The GC computer (*-GC) supports an MTI dual column Gas Chromatograph with a control and computational program which records accurate measurements of trace Hydrogen, Nitrous Oxide, and Methane (H_2 at 3-3,000ppm, $\mathbf{N_2O}$ at 10 to 4,000 ppm, and $\mathbf{CH_4}$ at 10 to 4,000 ppm).

4.2.1.1 GC Computer Hardware

Model:	486/66 min. Laptop computer
Memory:	16 megabytes of RAM min.
Disks:	A: - 3.5" HD floppy
	C: - 800 megabyte hard drive
Ethernet:	3COM Etherlink III PCMCIA card
Ports:	connected to dual column GC via RS-232 cable on COM1

4.2.1.2 GC Computer Software - (System only operates with commercial software)

Windows 95 (EZChrom 200 must run under a 32 bit operating system environment to allow the GCAPP32 call, from *-HOST, to check for the existence of the desired file)
NETBEUI/TCP/IP network protocol - connected to HLAN
EZChrom 200 Data System software, version 4.02 (supplied with GC)

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

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

C:\ (minimum 800 Meg. hard drive with one partition)
- GC
- MTI
- TEMP
- WINDOWS
- WINUTILS

4.2.1.4 GC Network Shares and Connections

Shares from GC include:

Share Name	Path	
C	C:\	(password protected)

4.2.2 The *-HOST Computer is the focal point for the instruments within the SHMS-E+ enclosure, performing 4 main functions.

4.2.2.1 *-HOST handles the discrete sensors and valves associated with SHMS-E+ via the National Instruments I/O devices. Electrochemical cell hydrogen measurements and continuous temperature, pressure, and flow measurements are made, displayed, and logged. Additionally, periodic calibrations are automatically called for, which requires closing the sample line valve and opening the calibration gas valve.

4.2.2.2 The *-HOST computer supports a B&K Photoacoustic Multigas analyzer, which is used to record accurate measurements of Ammonia (NH_3 at 10 to 10,000 ppm) and water vapor.

4.2.2.3 *-HOST retrieves the chromatogram data files from the instrument computers and performs labeling, compression, and other functions and stores the data on LABHOST for network access by authorized data users and long term archiving.

4.2.2.4 *-HOST communicates with both the local instrument network and with HLAN using a network interface card.

4.2.2.5 *-HOST Computer Hardware

Model:	100 MHz min. Pentium[4] Laptop computer
Memory:	32 megabytes of RAM , min.
Disks:	A: - 3.5" HD floppy
	C: 850 megabyte hard drive
Ethernet:	3COM Etherlink III PCMCIA network card
Ports:	Parallel port EPP IEEE 1284.HTM std.
	Serial ports: Com1
Video:	SVGA 800x600 min.
Others:	National Instruments external "DAQPAD 1200", for analog and digital I/O

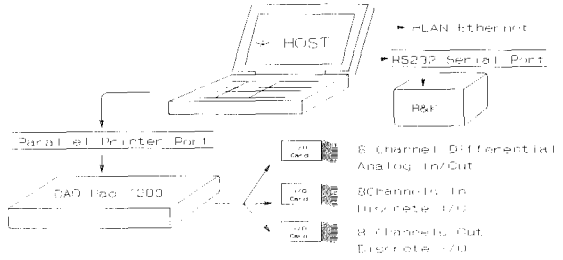


Figure 4B -- *-HOST I/O

[4]Pentium is a trademark of Intel.

4.2.2.6 *-HOST Computer Software

Operating System: Windows 95
Network Protocol: NETBEUI TCP/IP
Commercial Applications: NIDAQ for Windows 95, by National Instruments

Developed Interface Apps:

GC1ZIP.EXE -- Purges every N^{th} chromatogram unless it meets change or significance criteria and zips the previous day's data files into one, then sends the result to the designated network directory

GCAPP32.EXE -- Retrieves data from GC CPU, reformats data, relabels the files, and stores data locally

B&K.VI -- Labview developed executable program that interfaces with the B&K instrument and acquires data. It organizes data using a standard date naming scheme and transfers data from *-HOST to the 300 area data server, LABHOST, at daily intervals.

SHMS-E.VI -- Labview developed executable program to log electrochemical cell hydrogen sensors, system temperatures, flows, and pressures. Also controls automatic GC and B&K verifications and performs data transfer and archive functions.

4.2.2.7 *-HOST Computer Hard Disk Organization
(only the essential first level directories are shown)

```
C:\   (800 Meg.)
  ├──BIN
  ├──NETWORK
  ├──NIDAQ
  ├──USERS
  ├──TEMP
  ├──WINDOWS
  ├──WINUTIL
  └──DATA
     ├──ARCHIVE
     ├──B&K
     ├──B&K_DISK
     ├──GC
     └──NI
```

4.2.2.8 *-HOST Network Shares and Connections

Shares from *-HOST include:

Share Name	Path	
C	C:\	(password protected)
DATA	C:\DATA	(password protected)
GC	C:\DATA\GC	(password protected)

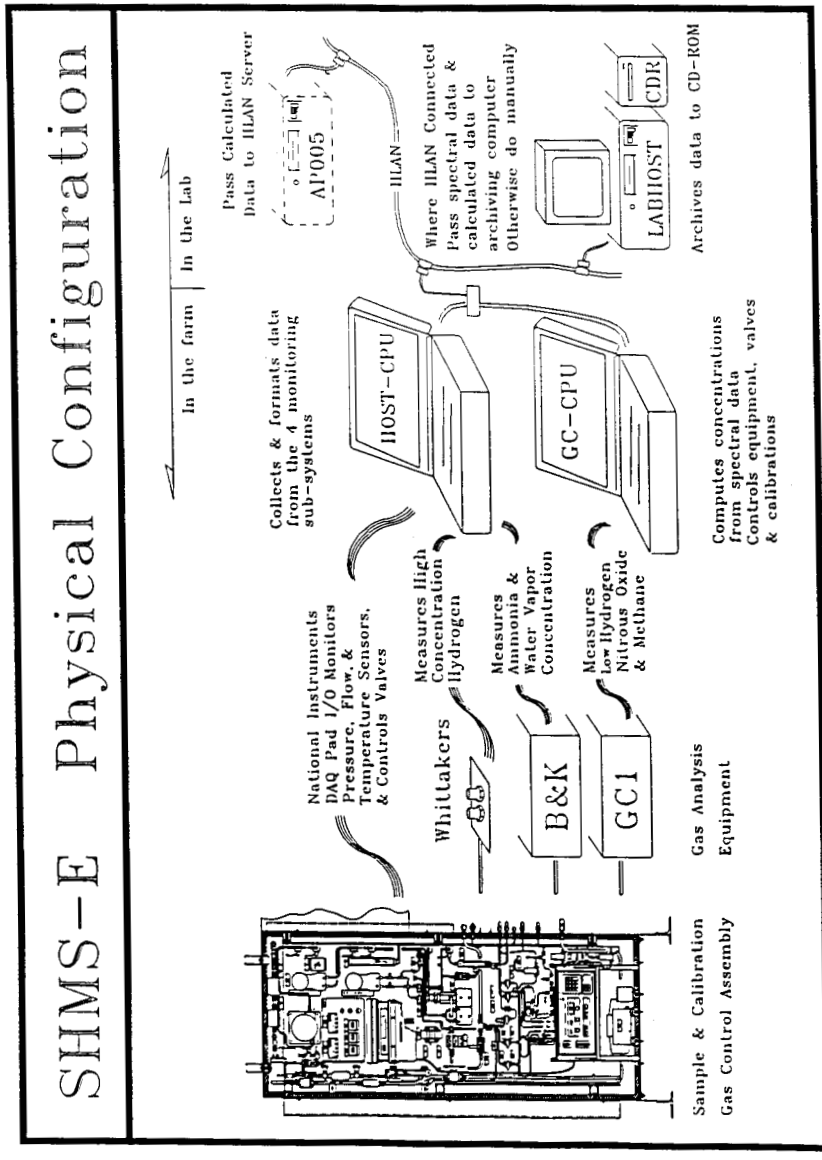


Figure 4C -- SHMS-E+ Physical Configuration

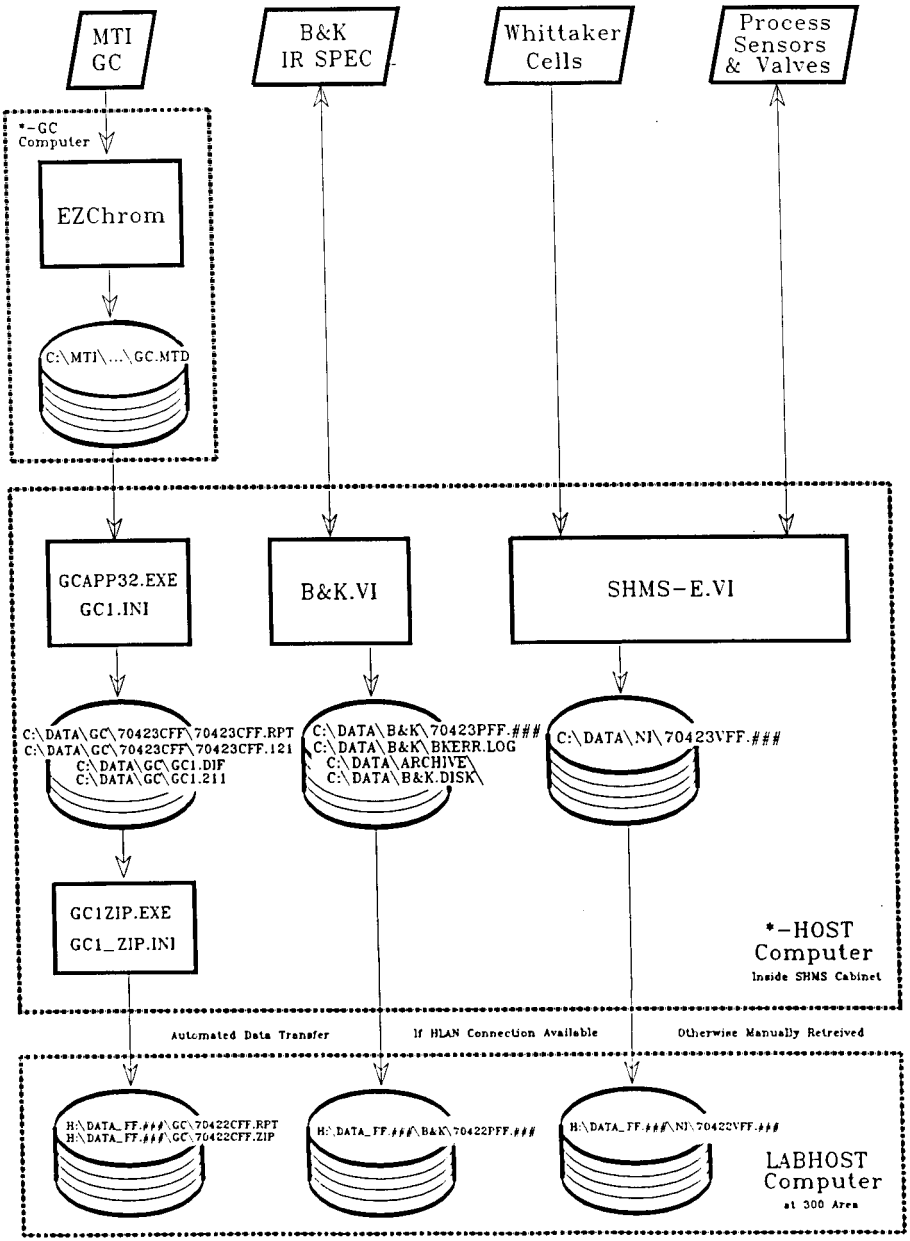


Figure 4D -- SHMS-E+ HIGH LEVEL DATA FLOW

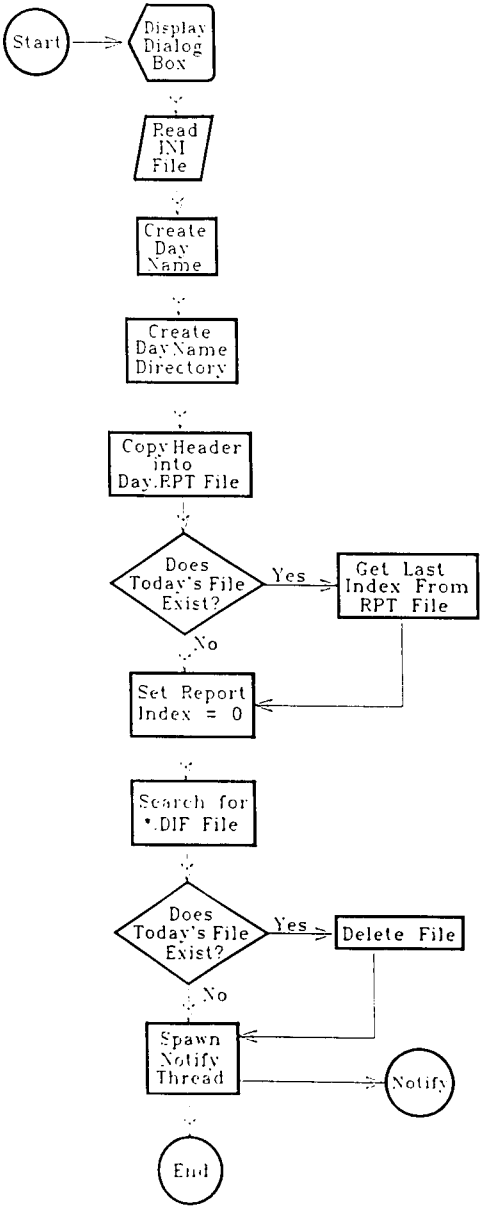


Figure 4E -- Interface Program "GCAPP32" Software Flow

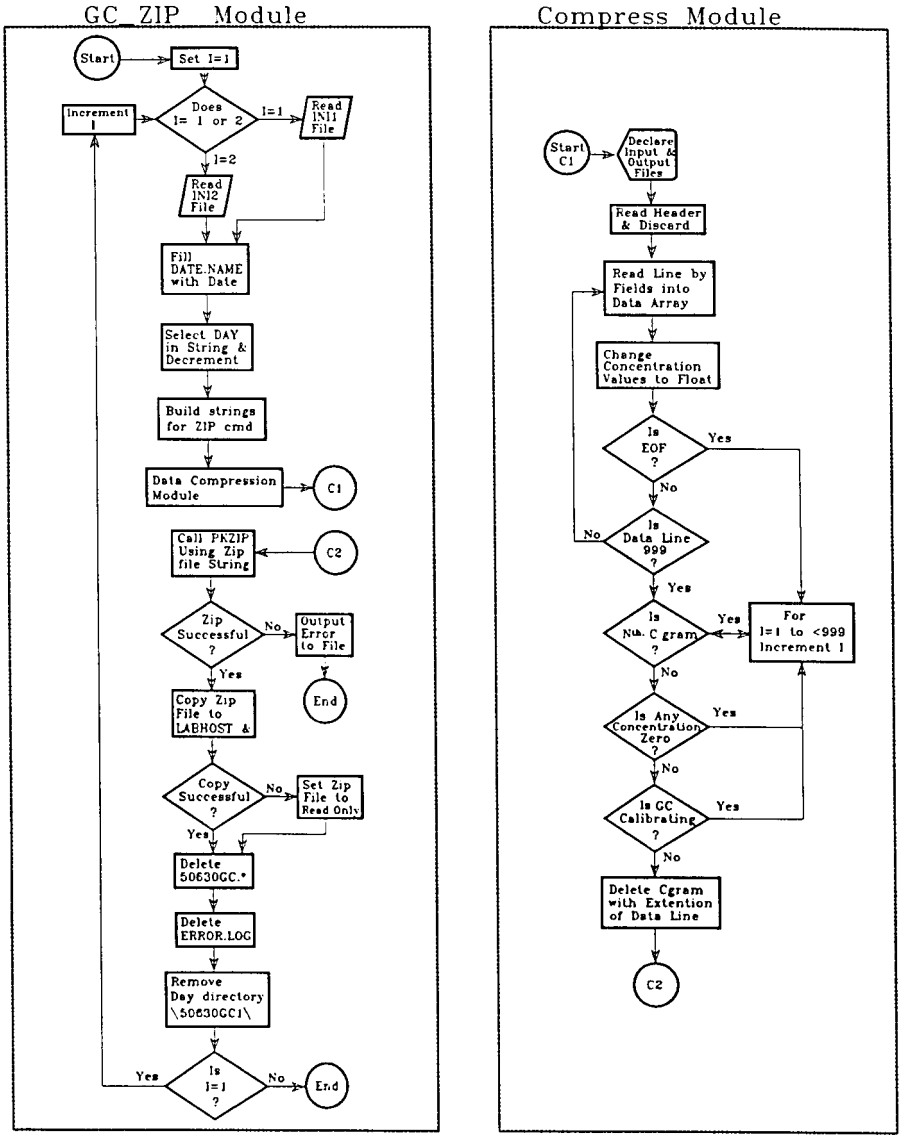


Figure 4F -- Interface Program "GC1ZIP" Software Flow

4.3 Functions and Flow of the SHMS-E+ Software

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

Software App Name	Source Code Listing/ Development Language	Description
GCAPP32.EXE	Appendix C / C++	Retrieves data from GC CPU, reformats data, relabels the files, and stores data locally.
GC1ZIP.EXE	Appendix D/C++	Purges every N^{th} chromatogram unless it meets change or significance criteria and zips the previous days data files into one, then sends the result to the designated network directory
SHMS-E.VI	Appendix E/Labview	Reads electrochemical cell data and process variable data directly from the sensors (temperature, pressure, & flow) then displays and logs it. It controls valves necessary for calibration verification.
B&K.VI	Appendix F/Labview	Interfaces with the B&K instrument, acquires and organizes data, transfers data from *-HOST to the 300 area data server, LABHOST, at daily intervals.

4.3.1 Gas Chromatograph

From the GC, massive chromatogram files are produced on a continuous basis. SHMS-E+ software converts each chromatogram into calculated concentrations of pre-selected gasses. The software also serves many other functions:

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

4.3.1.1 GCAPP32 Summary

The GCAPP32 program is used to retrieve data from the gas chromatograph, put the data into a file, then renumber the index and rename the report file consistent with the date. The program starts out with the opening of the console windows. The GC1.INI file containing several different variables which include string identifiers for the backslash, the report and chromatogram output file root path etc. This INI file is read and the day name which is in the form 50901cff where the 5 stands for the year, the 09 stands for the month and the 01 stands for the day, the cff additive differentiates between each of the tank farms.

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

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

4.3.1.2 GC1ZIP Program Description

The objective of using a GC1ZIP program is to reduce the large number of files generated by the chromatograms. Another consideration is that a large amount of storage is consumed by the chromatograms, this storage allocation is reduced by joining all chromatograms from a single day into one file. The GC1ZIP data reduction and file handling algorithm is governed by the initialization file known as an INI file. The GC1_ZIP.INI file indicates the path of the executable of the PKZIP program, the location of the files before they are to be zipped, their location after they are zipped, the number of the chromatogram to save (this condition depends on a number of parameters to be discussed later).

The GC1ZIP program reads the GC1_ZIP.INI file and fills the string called date_name with the appropriate date. The integers for the day are then pulled out of the date string program, these integers are decremented by one day, in order to ZIP the files for the previous day. The next step in the GC1ZIP program is to build the strings for each of the commands executed above. The first string will be used to build the path for the executable for the PKZIP file. The second string will be used to build the path for the location of the files to be zipped, the third and final string indicates the location of the files that are zipped. These locations are arbitrary and can be assigned to any drive, by changing the variables in the GC1_ZIP.INI file.

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

A variable called crname is constructed to contain the base definition of the chromatogram name(i.e 50806. where the 5 stands for the year 1995, the 08 stands for the month and the 06 stands for the day and a period is attached for purposes of attaching an index to the end of this). An if statement is used to check whether any of the lines are used for calibration, or whether any of the gases have zero concentration, if this is true then these chromatograms are saved else they are deleted. After this deletion the variable crname is reset to its initial contents(i.e from

50806.123 to 50806.). As the compression routine is ended the next 10 to 15 lines of the GC1ZIP code are used to build a string to execute the PKZIP command specifying the location of the files before and after the zipping operation and the actual executable location of the PKZIP file. After the zipping operation is completed the source files that are stored locally are deleted and the directory is removed. The resultant zipped GC file contains one days chromatograms and calculated data. Each is named using the following format.

year	month	day	Tank Farm Designator
5	08	28	CFF

4.4 LABHOST Computer

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

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

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

4.4.3 The third function of LABHOST is as a monitoring and diagnostic tool.

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

4.4.6 LABHOST Computer Hardware

Model:	AST[5] Premmia GX P-100, EISA/ISA/PCI bus compatible
Processor:	Pentium at 100 MHz min.
Memory:	32 megabytes of RAM min.
Disks:	A: - 3.5" HD floppy C: - 1.2 gigabyte hard disk (internal) D: - 1.2 gigabyte hard drive (internal) E: - CD ROM drive (internal) F: - 2.1 gigabyte hard drive (external) G: - Writable CD ROM (external)
Ethernet:	AMD on the motherboard for HLAN access, AUI connector 3COM Etherlink III network card on thin-net (BNC connector) (not bridged or forwarded to HLAN)
Video:	VGA

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

4.4.7 LABHOST Computer Software

Because LABHOST is a development computer, serving several functions, it's software configuration may change to accommodate the purpose it is serving at the moment. Therefore the following configuration is only a recommendation for initial setup:

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

4.4.8 LABHOST Computer Hard Disk Organization

The first level directories essential to the SHMS-E+ are dedicated to "D" hard disk.

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

```
D:\
  ├──CODE
  └──GC
H:\  (CDR hard drive for preparation of CD image only)
```

4.4.9 LABHOST Network Shares (HLAN) include:

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

4.5 Vendor Information File

Documentation and manuals for commercial software used in the SHMS-E+ are on file in CVI file # 22192.

Φ = Not a required part of the SHMS-E+ configuration

5.0 OPERATION

5.1 Overview

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

1) 486/33 or better host computer

2) Gas Chromatograph control computer

and 3) photoacoustic monitor control computer

5.2 Operating Procedures

5.2.1 System Startup, Normal Operation, Shut Down

See Appendix B for current operating procedure.

5.3 Discrepancy Reporting & Corrective Actions

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

6.0 CONFIGURATION CONTROL

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

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

6.1 Configuration Status Accounting

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

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

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

6.2 Media Control

The form of media used to store approved release software shall be CD ROM disks. The software media shall be stored in at least two locations. One set of the software media shall be stored in a file cabinet drawer labeled "software records" in the file named "SHMS-E+ software". This file cabinet will be located at a location specified by the manager of Characterization Monitoring Development. The other sets of software media shall be stored in each of the field SHMS-E+ cabinets with the HOST computers. Each approved release software CD shall bear the name "SHMS-E+", the version number, a disk number (sequential integer starting at 1), and the initials of the SHMS-E+ Design Engineer, with date. Each approved release software CD shall contain all software necessary to load and operate any computer in the system.

7.0 SAFETY, RELIABILITY, QUALITY ASSURANCE

7.1 Safety Classification and Signature Designation

The monitoring instruments and systems used for the Hanford Waste Tank Standard-E+ Hydrogen Monitoring System are classified as non-safety related General Service in accordance with the requirements of WHC-CM-4-46, "Safety Analysis Manual". Failure of the SHMS does not adversely effect the environment or the health and safety of the personnel operating the enclosure. General design and quality assurance requirements for Non-Safety items shall be followed. Although the SHMS function does not require it to be safety class or safety significant, its connection to the tank vapor space may necessitate designation of protective equipment as Defense-in-Depth Safety Equipment List components.

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

7.2 Quality Assurance and Quality Control

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

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

8.0 REFERENCES

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

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

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

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

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

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

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

9.3 Codes and Standards

American National Standards Institute (ANSI)

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

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

Institute of Electrical and Electronics Engineers (IEEE)

IEEE Standard 1016-1987.

National Electrical Manufacturers Association, (NEMA)

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

National Fire Protection Association (NFPA)

NFPA 70, *National Electrical Code*, 1990.

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

Underwriter's Laboratory

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

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

APPENDIX A -- DEFINITIONS & ACRONYMS

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

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

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

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

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

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

Verification Verification refers to process of injecting standard calibration gases into analytical systems and comparing the instrument's response to these standards.

APPENDIX B -- OPERATING MANUAL for SHMS-E+ SOFTWARE AND INSTRUMENT SYSTEMS

B.1.0 INTRODUCTION

Flammable gas watch-list (FGWL) tanks, which have a potential of producing Gas Release Events (GRE) exceeding 0.625% hydrogen by volume, require characterization. The purpose of this characterization is to accurately measure the flammable and hazardous gas compositions during baseline and GRE emissions. Additionally, the information is used to monitor any excursions that approach or exceed the Lower Flammability Limit (LFL) of the tank vapor space. Data from this characterization will help determine methods to resolve the Unreviewed Safety Question (USQ) for the FGWL tanks.

Currently, several Standard Hydrogen Monitoring System model E+ (SHMS-E+) are being installed on tank farms. Each of the SHMS-E+ systems will include two hydrogen Whittaker sensors, a Bruel&Kjaer Multi-gas Analyzer (B&K), a Gas Chromatograph instrument (GC), and a National Instrument data acquisition system. These instruments will provide the system with the capability to measure a wide range of gases from parts per million to percent volume concentration levels.

B.2.0 SCOPE

The objective of this document is to provide user desk instructions for the routine start-up, operation, and shutdown of the SHMS-E+.

B.3.0 PREREQUISITE

Conduct of the procedure steps outlined in the document assumes reasonable proficiency with personal computers and the Windows 95 operating system - including familiarity connecting network drives. Additionally, the Operator should be familiar with operation of the MTI gas chromatograph and the B&K infrared photo-acoustic spectrometer and associated software.

B.4.0 DESCRIPTION

The SHMS-E+ instrument/data acquisition system consists of networked computers, analytical instruments, gas sensors, and support sensors. Vendor supplied software is used to control operation and analysis functions of the analytical instruments and custom developed software is used to collect, store, organize, and transfer the data. Figure 1 attached shows the computer and instrument layout for the SHMS-E+. A brief description of the hardware software systems is given in the following sections. A more detailed description is detailed in Hardware Functional Design Description document.

B.4.1 GC Hardware/Software

The MTI gas chromatograph is currently configured to measure Hydrogen, Methane, and Nitrous Oxide. It is equipped with a molecular sieve column, poraplot-Q column and thermal conductivity detector/column modules.

The GC sub-system consists of an MTI gas chromatograph and a Pentium laptop computer networked to the second Pentium laptop host computer. Both computers operate with the Windows 95 operating system. In this document, the GC computer and the HOST computer will be referred to as (*-GC) and (*-HOST), respectively.

Note:

Several SHMS-E+ systems will be installed in different flammable gas watch list tanks. Since each SHMS-E+ contains one Host and one GC computer, these SHMS-E+ system network computer designators are uniquely defined as following:

GC COMPUTER	XX-05-N-GC
HOST COMPUTER	XX-05-N-HOST

where: XX is Tank Farm Identifier
05 SHMS system designator
N is Tank Number (i.e. 101=A 102=B 104=C etc..)
HOST is for Host Computer in the cabinet
GC is for GC Computer in the cabinet

The software system for running GC consists of the following:

EZCHROM, Version 4.0:	Vendor supplied software for controlling the MTI gas chromatograph and analyzing data. Software resides on +-GC computer.
GCAPP32, Version 1.0:	Custom developed software for transferring EZCHROM data from the *-GC computer to the *-HOST computer. Software also organizes and renames the data files using a standard date/run-number naming scheme. Software resides on the *-HOST computer.
GC1ZIP, Version 1.0:	Custom developed software for compressing and transferring GC data from *-HOST to the 300 area data server, LABHOST. Program is run at daily intervals using SHMS-E.VI scheduler.

B.4.2 Bruel&Kjaer Multi-gas Analyzer Hardware/Software

The B&K sub-system consists of a B&K photoacoustic spectrometer Multi-gas Analyzer which interfaces directly to the *-HOST laptop computer via RS-232 communication link. The *-HOST computer operates with Windows 95 operating system. Currently, the B&K Multi-gas Analyzer is configured to measure ammonia and water vapor but can readily be configured to measure a wide variety of gases.

The software system for running B&K consists of the following:

B&K.VI, Version 1.0:	Custom developed software to provide interfacing and data acquisition from the B&K instrument. Software also organizes data using a standard date naming scheme. It also transfers data from *-HOST to the 300 area data server, LABHOST, at daily intervals. Software resides on the *-HOST computer.

B.4.3 Whittaker Cells and Process Instrumentation Data Logging Hardware/Software

The data logging sub-system consists of several National Instruments data acquisition boards and software installed on the *-HOST computer. It monitors and records Whittaker hydrogen sensors and system variables such as temperature and flow. In addition, this subsystem also controls automatic timed injections of calibration gases into the GC systems.

SHMS-E.VI, Ver 1.0:	LabView developed executable program to log Whittaker hydrogen sensor signals, system temperatures, flows, and pressures. Also controls automatic GC and B&K calibration verifications and performs data transfer and archive functions.

B.5.0 PROCEDURE

The following procedure describes steps for setting up, operating, and limited troubleshooting for the SHMS-E+ systems. Since all data is transferred to the 300 Area Data Server (LABHOST), a prerequisite to this procedure is to ensure that LABHOST is on and connected to the network. NEVER SHOULD LABHOST BE TURNED OFF OR REMOVED FROM THE NETWORK WITHOUT THE SYSTEM ADMINISTRATOR'S ACKNOWLEDGMENT. Unplanned system and network outages have been factored into the software system design and in this event all data will be saved locally. However, because of the large amount of data generated, local hard drives can not store more than 1 week of data.

B.5.1 Initial Setup

5.1.1 Ensure that the SHMS-E+ and all instruments (B&K, GC, Process instruments and all other support equipments) are powered. Power up all computer systems *-HOST and *-GC. Passwords are required for all systems. The system administrator maintains the active user and Passwords list.

NOTE: THE START-UP SEQUENCE OF THESE COMPUTER SYSTEMS IS CRITICAL TO ENSURE PROPER ESTABLISHMENT OF NETWORK COMMUNICATIONS.

5.1.2 On the *-HOST computer, verify that the following shares and network drives are connected. If unable to establish network shares or connect network drives, please refer to Section 6.1 of this procedure - Troubleshooting Network Communications.

```
SHARE     C:\DATA\GC     AS  GC
CONNECT   Drive H:  TO  \\LABHOST\DATA_FF.###
```

NOTE: THE NAMES AND LOCATIONS OF THESE DRIVES ARE CRITICAL FOR PROPER SYSTEM OPERATION.

5.1.3 On the *-GC computer, connect the following network drive:

```
CONNECT   Drive D:  TO  \\*-HOST\GC
```

NOTE: THE NAME AND LOCATION OF THIS DRIVE IS CRITICAL FOR PROPER SYSTEM OPERATION.

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

5.1.4 Verify that the *-HOST, *-GC computer systems and the B&K instrument synchronize to the current time and date.

5.1.5 Visually verify that all support gas bottles are connected to the system and the pressures are properly regulated.

B.5.2 Starting SHMS-E+ Virtual Instrument (VI) Data Acquisition System

The SHMS-E+ VI data acquisition system logs hydrogen sensor output signals, process variables and initiates automatic calibration verifications of the B&K and GC. The following procedure assumes a start-up with the default settings. A complete description of user selectable settings is contained in HNF-SD-WM-CSWD-???.

5.2.1 From the *-HOST computer, **CLOSE** all active applications and then **START** the SHMS-E+ VI application by selecting the "SHMS-E" icon located on the desk top.

5.2.2 Verify that the SHMS-E.VI default control parameters as follows:

FILE CONFIGURATION:			
* ARCHIVE PATH:	**H:\NI**		
* LOG INTERVAL SEC:	**60**		
* RECORD DATA:	**LOG TO FILE**		
* TANK ID:	**VFF.###**		(note: 1)
B&K NIT--*52:	**OFF**		
GC MON--*60			
* ARCHIVE TIME:	**1:00**	**OFF**	
B&K CAL			
CALIBRATION			
* TIME:	**19:00**		(note: 2)
* DURATION:	**15**		
CAL CH A			
CALIBRATION			
* TIME:	**20:00**		(note: 3)
* DURATION:	**15**		
CAL CH B			
CALIBRATION			
* TIME:	**21:00**		(note: 3)
* DURATION:	**15**		

5.2.3 **RUN** the SHMS-E+ application with default parameters setting in the previous step by clicking on the continuous run button.

5.3.5 Toggle the ON/OFF switch on the SHMS-E front panel to **ON** position.

NOTE: 1. TANK ID designator definition is V-FF.### where V refers to Virtual Instrument and FF refers to tank farm designator. The extension .### is the tank identification number (i.e. VAW.101 is the virtual instrument for tank 101 in AW farm).

2. B&K auto calibration verification - Dial in the desired TIME and DURATION.

3. GC auto calibration verification - Dial in the desired TIME and DURATION for each channel.

B.5.3 Starting the B&K system

The following procedure steps are in accordance with the parameters established in the calibration for each instrument. The configuration parameters for the B&K instrument (i.e filters, calibrations, environment, communication, etc...) are setup and stored in the B&K's EEPROM. (Note: Refer to the B&K 1302 Operation & Maintenance manual for configuration details.) A unique configuration parameter settings for each B&K instrument is maintained in the 6-TF-143 procedure, BRUEL & KJAER 1302 AMMONIA MONITOR CALIBRATION.

5.3.1 Place the B&K instrument in continuous sampling mode by toggling the NIT-*52 switch on the SHMS-E front panel to **ON**. The B&K front panel window should be displayed in RUN mode.

NOTE: If the B&K "local" display does not indicate that the spectrometer is resetting operation refer to Section 6.2 of this procedure Troubleshooting B&K Operation.

5.3.2 Verify that the B&K.VI control parameters as follows:

FILE CONFIGURATION:			
* DOWN LOAD DATA TO DISK:	**OFF**		(note: 1)
* ARCHIVE PATH:	**H:\B&K**	**ON**	(note: 2)
* TANK ID:	**PFF.###**		(note: 3)
B&K SHUT-OFF SWITCH:	**ON**		(note: 4)

NOTE: 1. To down load data to diskette, insert a blank floppy disk into drive A of *-HOST computer then momentarily press the radio button to "SAVE TO DISK".

2. The "ARCHIVE PATH" parameter transfers B&K data to a network server. To activate this function, connect *-HOST computer to the network server then provide the completed network path as a parameter and then toggle the function to **ON**.

3. TANK ID designator definition is P-FF.### where P refers to the B&K Photo-acoustic Instrument identification and FF refers to the tank farm designator. The extension .### is the tank identification number (i.e. PAW.101 is the B&K instrument on tank 101 in AW farm).

4. Cycling the "B&K SHUT-OFF SWITCH" provides full reset to the B&K instrument.

B.5.4 Starting the GC system

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

5.4.1 From the *-GC computer, close all active applications.

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

5.4.3 Select METHOD/OPEN open c:\MTI\EZCHROM\METHODS\GC.MTD.

NOTE: If GC.MTD does NOT exist, generate a method file for the GC using certified standard calibration gases with the system engineer's direction and MTI vendor instructions.

5.4.4 To upload the selected method to the GC,
Select INSTRUMENT/SEND CURRENT METHOD

5.4.5 Select DATA/SAVE AS enter D:\GC\ in the dialogue box to establish default save path.

5.4.6 Under REPORT menu, select the External Standard for both channels A and B.

5.4.7 From the *-HOST computer, launch the GCAPP32 application icon.

NOTE: To shut down the GCAPP32 use <Ctrl><C> keys.

5.4.8 From the EZCHROM application at the *-GC computer, select the START menu. A run window should appear. Set the following parameters and click on START to begin the sampling process.

Run ID	**GC1**
Number of Runs:(1-999,inf)	**inf**
Time Between Injections:(secs)	**360**
( ) Wait for External Start	
(**X**) Save ( ) Print	
(**X**) DIF Save ( ) PRN Save ( ) Extended	
User Program:	

5.4.9 EZCHROM display should change to indicate operation of the Gas Chromatograph. If display does not indicate proper operation of the Gas Chromatograph refer to Section 6.4 of this procedure Troubleshooting MTI GC Operation.

5.4.10 Click on the START radio button to begin Run.

NOTE: AFTER IT'S RUNNING, LEAVE *-GC ALONE!!! EZCHROM IS NOT MULTI-TASKING.

B.5.5 Verify all systems are operating and communicating with *-HOST

5.5.1 Verify that the Whittaker and process instruments' local displays match the SHMS-E.VI front panel read out.

5.5.2 Use Explore or Windows File Manager to browse the C drive and select the C:\DATA\NI directory.

5.5.3 Browse the C:\DATA\NI directory. A new report file is generated for each day by SHMS-E VI software. The file naming convention is specified as Y-MM-DD-V-FF.### where Y-MM-DD refers to the year, month and day of today's date. Following are the VI instrument identification V and tank farm designator FF (i.e. 60211VAW stands for the Feb 11, 1996 report file. VAW is the SHMS-E+ Virtual Instrument for AW farm). The .### is the tank identification number (i.e. 60211VAW.101 is for tank 101 in AW farm).

5.5.4 If there is no current data file in C:\DATA\NI then refer to Section 6.2 "Troubleshooting National Instrument Data Acquisition Problems".

5.5.5 Verify that the Ammonia concentrations displayed on the B&K instrument local display, SHMS-E.VI front panel, and the pop up B&K front panel match.

5.5.6 Browse the C drive and select the C:\DATA\B&K directory. A new report file for each day is generated by B&K VI interface software. The report's file naming convention is specified as Y-MM-DD-P-FF.### where Y-MM-DD refers to the year, month and day of today's date. Following the date are the B&K instrument identification P and tank farm designator FF (i.e. 60211PAW stands for the Feb 11, 1996 report file. PAW refers to the Photo acoustic instrument for AW farm). The extension .### is the tank identification number (i.e. 60211PAW.101 is for tank 101 in AW farm).

5.5.7 If there is no current data file in C:\DATA\B&K then refer to Section 6.3 "Troubleshooting B&K Operation".

5.5.8 From *-HOST computer select GCAPP32 window and make it active.

5.5.9 Verify that the run number index is incrementing.

NOTE: GC run time is approximately 6 minutes, consequently the run number should be incremented about every 6 minutes. If the system does not update within 10 minutes the programs will "time out". The text in the window will indicate "PROGRAM TERMINATED".

5.5.10 If the GCAPP32 is not incrementing refer to section 6.4 of this procedure, Troubleshooting GC Operation

5.5.11 Browse the C drive and select the subdirectory under C:\DATA\GC. A new directory for the day was generated by GCAPP32 application. The directory naming convention is specified as Y-MM-DD-C-FF where Y-MM-DD refers to the year, month and day of today's date. Letter C is for chromatograph instrument identification and is followed by letters FF for tank farm designator (i.e. 60211CAW stands for the Feb 11, 1996. CAW refers to the gas chromatograph instrument in AW tank farm). All files in the directory also use the same naming convention with two different extensions. The extension .RPT is for the report file (i.e. 60211CFF.RPT) and the .### extension is the index number for the chromatogram (i.e. 60211CFF.020 is the 20th chromatogram).

5.5.12 If there is no current today data file in C:\DATA\GC\YMMDDCGG then refer to Section 6.4 "Troubleshooting GC Operation".

5.5.13 Close the Windows File Manager or Explore.

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

B.6.0 Troubleshooting

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

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

Copies of the pertinent manuals for the B&K, GC, and Labview Products. are archived in CVI-22192.

B.6.1 Troubleshooting Network Communications

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

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

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

B.6.2 Troubleshooting National Instrument Data Acquisition Problems

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

6.2.1 Verify that the **Save to File** radio button in the SHMS-E VI application has been activated.

6.2.2 Verify that the Start switch on the SHMS-E VI application has been toggled to **ON**.

6.2.3 If problem persists contact the Data Acquisition Lead.

B.6.3 Troubleshooting B&K Operation

6.3.1 Verify that the B&K instrument is powered on.

6.3.2 Verify that the interface cable connecting the B&K with the *-HOST computer serial port is secure on both ends.

6.3.3 Verify that the communication parameters are set properly. (refer to 6-TF-143)

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

B.6.4 Troubleshooting MTI GC Operation.

6.4.1 Verify the MTI GC is powered on.

6.4.2 Verify the serial cable connecting the MTI GC with the *-GC computer is secure.

6.4.3 In EZCHROM, verify the detectors are **ON**.

6.4.4 In EZCHROM, verify that data is saved to D:\.

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

6.4.6 If problem persists, contact the Analytical Instrument Lead.

APPENDIX C -- C++ SOURCE CODE LISTINGS for GCAPP32.EXE

C.1 GC1.INI

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

************************************************
Ini file descriptions

backs:  Character for backslash on file paths

root:   Report and chromatogram output file root path

C_gram_old:     Chromatogram GC run identifier

gcPath:                 Dif file created by EZChrom path

outputPath:     Report and chromatogram output file root path

outputExt:      Report output file extension

gc_del:         Wildcard definition of files to delete in input directory

grampath:       Chromatogram file output path

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

calPath:        File to watch for for calibration events
```

C.2 GCAPP32.CPP

```
/*****************************************************************************/
// Program GCAPP32.CPP (Consol Application)                                   //
// Version Beta                                                               //
/*****************************************************************************/


// Modified gc1app32.cpp to correct date change problem.

// Includes and Definitions

#include <windows.h>          // Windows definitions include file.
#include <windowsx.h>         // Additional windows definitions include file
#include <stdio.h>            // Standard input and output for C++ include file.
#include <iostream.h>         // Input/Output streams include file.
#include <stdlib.h>           // Standard function library include file.
#include <fstream.h>    // File input/output stream include file.
#include <dos.h>              // Dos function include file.#include <windows.h>
#include <signal.h>           // Signal function to handle Ctrl+C
#include <string.h>

// User Includes and Definitions
#include "gc1app32.h"  // Function prototypes for gcapp modules.

char  *header ="c:\\bin\\header1.h";     // file containing Header
                                                                          //
for *.rpt file.

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


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


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

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

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

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

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

// Booleans
BOOL fFailIfExists = TRUE;              // CopyFile command parameter to have the
                                                                    // function fail if
the file already exists
                                                                    // where it is
trying to be copied to.
// Main Handles
HANDLE hThread;                 // Handle for NotifyThread executed in main program


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

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


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

// This signal function will provided exit route
// hot key Ctrl+C
        if(signal(SIGINT, Exit_handler))
        {
                perror("Could not set SIGINT");
                exit(0);
```

```
        }

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

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

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

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

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

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

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

                    ini>> col1>>col2>>col3;
                        strcpy(grampath, col3);         // Copy from ini file into EZChrom output path

                        ini.close();            // close ini file

                        gc_id_str[0] = ini_file[9];
                        gc_id = atoi(gc_id_str);        // Set the gc_id variable equal to the
                                                                                        //
character in the ini file definition.
                                                                                        // if
gc1.ini, then gc_id = 1

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

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

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

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

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

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

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

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

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

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

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


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

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


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

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

                    if(GetLastError() == ERROR_FILE_NOT_FOUND)  // If no *.dif file...
                    {
                            printf("The file was not found\n");
                            printf("The *.dif and chromatogram files could not be deleted\n");
                    }

                    else                                                    // If the *.dif file
exists...
                    {
                            strcpy(del_path, grampath);         // Copy first 3 char of gcPath in to
del_path
                            strcpy(temp20, del);                // Copy del into temp20
                            strcat(temp20, del_path);  // Concatenate del_path into
temp20,temp20 = del F:
```

```
                        strcpy(temp21, temp20);   // Copy temp20 into temp21
                        strcat(temp21, gc_del);   // Concatenate gc_del into temp21
                                                                        // temp21
= del F:\gc1.*
                        strcpy(temp22, temp21);   // Copy temp21 into temp22
                        strcat(temp_command, temp22);  // Concatenate temp22 into
temp_command
                                                                        //
temp_command = command.com /c del F:\gc1.*

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

                        if(win_exec_error < 32)      // If there is an error in WinExec
                        {
                                sprintf(win_exec_error_msg, "WinExec failed; error code =
%d", win_exec_error);
                                printf(("%s"), win_exec_error_msg);      // Output error
                        }
                        else
                        {
                                printf("The *.dif and chromatogram files were deleted\n");
                        }                       // Delete was successful
                }

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

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

                hThread = CreateThread ( 0, 0, (LPTHREAD_START_ROUTINE) NotifyThread,
                                                                                0,
0, &dwthreadID );
                                                                        // Create a
thread called NotifyThread
                                                                        // and let
it begin to execute.
                                                                        // This is
based on code from the
                                                                        //
Advanced Windows book: Pages 60-64
                                                                        // and
Chapter 9(Thread Synchronization)
                i=0;                                                    // (WIN32)
                while (i ==0)
                {
                  Sleep(6000);
                }


}       // End of Main Program
```

```
DWORD WINAPI NotifyThread ()          // The Notify Thread started by main program
{
        // File Handles
        HANDLE hFile, findFile;

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

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

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


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


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

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

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

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

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

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

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

                                                                                  // Initialize
the handle for the
                                                                                  //
FindFirstChangeNotification Object
                                                                                  // This flag
tells us when any file's
                                                                                  // last
write time has changed in the
                                                                                  // path
specified in filePath
                                                                                  // This is
based on code from the
                                                                                  //
Advanced Windows book: Pages 663-666
                                                                                  // (WIN32)

                if (hFile == INVALID_HANDLE_VALUE)
                {
                        printf("FindFirstChange problem\n");
                }

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

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

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

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

                printf("Beginning Wait for File Change\n");

                SetLastError(0);                         // Set Error code to zero
                                                                  // to prevent carrythrough of
                                                                  // any errors to other
processes.
                notif = FindNextChangeNotification(hFile);       // Reset the waiting object flag for
dif file
                notif = FindNextChangeNotification(hFile);       // Reset the waiting object flag for
C-gram

                if (notif == FALSE)
                {
                        printf("FindNextChange problem\n");
```

```
		}

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

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

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

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

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

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

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

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

		Sleep(1000);                                                    //
Wait 10 seconds for the
                                                                                //
c-gram and dif files to
                                                                                //
complete writing.

		// Look here for whether calibration flag is up
		// If it is set Cal = TRUE
		if (FindFirstFile(calPathA, &calData) == INVALID_HANDLE_VALUE)	// Look for the
calibration
                                                                                // file
created by LabView
		{
			if (FindFirstFile(calPathB, &calData) == INVALID_HANDLE_VALUE)	//
vi.GC\Channel-A
			{
				Cal = "0 ";
			}
```

```
                else
                {
                        Cal = "B ";
                }
        }
        else
        {
                if (FindFirstFile(calPathB, &calData) == INVALID_HANDLE_VALUE)          //
vi.GC\Channel-A
                {
                        Cal = "A ";
                }
                else
                {
                        Cal = "AB";
                }
        }


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

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

        else    // If the dif file is not found...

        {
                printf("File Has Changed\n");
                FindClose(findFile);                                        // Close
the findFile handle
                                                                            //
(WIN32)

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


                        difIndex = ind_trns(gcPath);                        // Get the
current index from
                                                                            //
the dif file and place it
```

```
                                                                    // in
difIndex.
                                                                    //
(Developed function, gc1_trns.cpp)
                    reportIndex++;              // Increment reportIndex
                    indx(gcPath, reportIndex);  // Replace the current index
                                                                    // in
the dif file with
                                                                    //
reportIndex.
                                                                    //
(Developed function, gc1_indx.cpp)
                    strcpy(temp7, temp3);       // Copy temp3 into temp7
                    appnd(temp7, gcPath, Cal);      // Append the line from the
                                                                    //
dif file into the
                                                                    //
rpt file.
                                                                    //
(Developed function, appnd.cpp)
                    printf("Finished Appending #%d\n", reportIndex);
                    DeleteFile(gcPath);                 // When finished, delete the
*.dif file
                                                                    // (WIN32)

/*********************************************************************/
                    // Renaming of C-grams section
/*********************************************************************/
                    strcpy(C_gram_new, old_date);       // Copy old_date into
C_gram_new


                    index_str(reportIndex, new_ext); // Convert reportIndex into the three-
                                                                    //
character extension for the new
                                                                    //
c-gram file.
                                                                    //
reportIndex = 3
                                                                    //
new_ext = "003"
                                                                    //
(Developed function, index3.cpp)
                    itoa(difIndex, old_ext, 10);    // Convert difIndex to a string for
                                                                    //
the old c-gram file.
                    strcat(C_gram_new,new_ext);     // Concatenate the new
three-character
```

```
                                                                    //
extension onto C_gram_new
                                                                    //
C_gram_new = "Jun0195.003"

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


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

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

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

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

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

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

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

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

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

```
                                                                    // if
the MoveFile fails...
                    {
                        printf("C-gram not generated\n");
                    }

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

                    fil_name(new_date, gc_id);          // Fill new_date with the
current date
                                                                    //
(Developed function, gc1_fil_.cpp)
                    if(strncmp(old_date, new_date,5) != 0)      // If new_date and
old_date
                    {                                               //
are different...
                        strcpy(old_date, new_date);     // assigned new_date
to old_date
                                                                    //
when date change occurred
                        fil_name(dirName, gc_id);                   //
Fill dirName with the date
                                                                    //
(Developed function, gc1_fil_.cpp)
                        strcpy(temp3, root);                        // Copy
root into temp3
                                                                    //
temp3 = "d:\gc1\"
                        strncpy(temp2, dirName, 7);     // Copy first 7
characters of
                                                                    //
dirName into temp2
                                                                    //
temp2 = "Jun0295"
                        strcat(temp3, temp2);       // Concatenate temp2 onto
temp3
                                                                    //
temp3 = "d:\gc1\Jun0295"
                        CreateDirectory(temp3, NULL);   // Create a directory using
                                                                    //
temp3 string.
                                                                    //
(WIN32)
                        strcat(temp3, backs);       // Concatenate a backslash onto
                                                                    //
temp3.
                                                                    //
temp3 = "d:\gc1\Jun0295\"
```

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

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

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


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

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

                    if(COPYGOOD == TRUE)        // If the copy succeeds...
                    {

                        printf("No file for today, %s is created\n"), outputPath);
                        printf("Copied header.h to today's file\n");

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

                        printf("The first index in today's file is %d\n"),
reportIndex);
                    }

                    else                                            // If
the copy fails because the file exists
                    {

                        printf("Today's output file exists\n");

                        reportIndex = rpt_trns(outputPath);    // Get current
index from the
```

```
                                                                                    //
existing report file.
                                                                                    //
(Developed function, gc1_trns.cpp)

                                                printf("The transferred index is %d\n", reportIndex);
                                        }

                                } //End of New Day if statment

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


        } //end of infinite while loop

        FindCloseChangeNotification(hFile);                                 // Close the file change
                                                                                    //
object handle. (flag)
                                                                                    //
(WIN32)
        // Return thread
        return(0);
}


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

        char hund[2];                                           // Character hundreds digit holder
        char tns[2];                                            // Character tens digit holder
        char ounsl2];                                           // Character ones digit holder
        char temp[20];                                          // Character temporary variable to
hold partial
                                                                                    // string while it
is constructed

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

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

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

```
                                                                        //
strip off
                                                                        //
ones digit

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

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


        return;                                                 // return to main
program

} // End of index3 module


void Exit_handler(int)
{
int c;
signal(SIGINT, SIG_IGN);
printf("Interrupted. Quit?");
c= getchar();
if(c=='y' || c=='Y')
{
        CloseHandle(hThread);       // Close the handle to the thread
                                                                        // (WIN32)
        exit(0);
}
else signal(SIGINT, Exit_handler);
}
```

C.3 APPND.CPP

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

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

void appnd(char *rptName, char *gc1_nam, char *cal)
{
//rptName = output report file
```

```
                                                                        //
gc1_nam = gc*.dif
                                                                        //
cal = 1 or 0, depending on cal time


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

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

// Declare the input and output files.
        ifstream ifile;                                 // The input file stream pointer
        ofstream ofile;                                 // The output file stream pointer
        ofstream VIfile;                                // The output LabView interface file stream
pointer

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

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

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

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

        // Writes the concentrations to the io file by fields
        VIfile << col6 << "\n"
                << col7 << "\n"
                << col8 << "\n";

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

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

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

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

}       // End of appnd module
```

C.4 DIF_CHK.CPP

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

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

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

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

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

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

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

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

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

} // End of dif_chk module
```

C.5 DIF_FIX.CPP

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

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

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


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

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

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

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

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

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

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

} // End of dif_fix module
```

C.6 GC_FIL.CPP

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

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

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

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

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

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

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

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

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

        sdat[0] = tms[23];                          // sdat = " 06"
        sdat[3] = tms[8];                                  // sdat = "5062"
        sdat[4] = tms[9];                                  // sdat = "50623"
        sdat[5] = 67;                                      // sdat = "50623G"
        sdat[6] = 70;                                      // sdat = "50623GC"
        sdat[7] = 70;                           // sdat = "50623GC1"
        sdat[8] = 46;                                      // sdat = "50623GC1."
        sdat[9] = 0;                            // sdat = "50623GC1." with null
                                                //        terminator

        return;                                                 // Return filled sdat to
main program

} // End of gc_fil module
```

C.7 GC1_INDX.CPP

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

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

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

        fstream iofile;                             // Input/Output file stream pointer

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


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

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


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

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

// Writes the data to the *.dif file in fields
        iofile << col1 << space << col2
                                        << space << col3
                                        << space << col4
                                        << space << setw(7) << index              // setw()
sets the width
                                        << space << setw(8) << col6         // of the positon
the char
                                        << space << setw(8) << col7               // variable
is written to,
```

```
                                        << space << setw(8) << col8          // allowing
spacing.
/*                                      << space << setw(8) << col9 */ << "\n";

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

    return intIndex;                                    // return intIndex to main program

}       // End of gc1_indx module
```

C.8 GC1_TRNS.CPP

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

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

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

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

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

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

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

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

        if(!infile.eof())                   // If not the end-of-file yet
```

```
            {
              indexOut = atoi(index);       // indexOut is equal to the integer
                                                                  // value of index
            }

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

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

        }

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

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

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

        return indexOut1;

}       // End of gc1_trns module
```

C.9 INDEX3.CPP

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

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

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

        char hund[2];                                   // Character hundreds digit holder
        char tns[2];                                    // Character tens digit holder
        char ouns[2];                                   // Character ones digit holder
```

```
        char temp[20];                                  // Character temporary variable to
hold partial
                                                                    // string while it
is constructed

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

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

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

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

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


        return;                                                     // return to main
program

}
```

C.10 RPT_TRNS.CPP

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

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

int rpt_trns(char fname[50])
```

```
{
        int indexOut,indexOut1;                         // Output integer index variables

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

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

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

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

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

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

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

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

        }

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

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

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

        return indexOut1;

} // End of rpt_trns module
```

C.11 TIME_STP.CPP

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

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

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

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

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

        return;                                         // Return to main program

} // End of time_stp module
```

C.12 GC1APP32.H

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

C.13 HEADER.H

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

APPENDIX D -- C++ SOURCE CODE LISTINGS for GC1_ZIP.EXE

D.1 GC1_ZIP.INI

```
path = c:\bin\pkzip.exe
zippath = C:\DATA\GC\
out_path = C:\NI\
nth_c = 2


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

path:          The path for the pkzip executable

zippath:       The path for the files to be zipped

out_path:      The path for the zip file

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

D.2 GC1ZIP.CPP

```
/*****************************************************************************
// Gc1zip.cpp
// Cuong Vo, Dec 1996
// This program uses PKZip(TM) to archive GC data and prepare it for
// being burned onto a CD.
// Version Beta 0.10
//
*****************************************************************************/


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

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

void main(void)
{
        // Ini file pointer and title.          // The ini and error file
        fstream ini;                                                          // pointers

        char ini_file[30] = "c:\\bin\\GC_ZIP.INI";  // The GC1_ZIP.INI ini file

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

```
char col3[25] = "\0";

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


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

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

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

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

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

// Temp strings
char temp_path[100] = "\0";                  // Contains entire system call to zip files.
char temp_zip[50] = "\0";                          // Contains zip file path & name
char temp_zipped[75] = "\0";               // Contains files to be zipped path & names
char temp_del[50] = "\0";                          // Contains zip file path & name to delete
                                                                        // after file copy.
char temp_rpt[50] = "\0";                          // Report file char variable
char temp_cgram[50] = "\0";                // C-gram file char variable
char temp[5] = "\0";           // Temporary char variable
char out_rpt[30] = "\0";                   // Output report variable
char out_dir_temp[40] = "C:\\DATA\\GC\\"; // Temporary place for zip file befor transfer
int int_year;                                      // Year digit
int int_month;                                     // Month digits
int int_day;                                 // Day digit to be decremented
int nth_c_int;

WIN32_FIND_DATA fileData;        // Find File Data Structure

HANDLE findFile;                                   // Find File Handle

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

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

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

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

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

ini.close();          // closes ini file

nth_c_int = atoi(nth_c_str);

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

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

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

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

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

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

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

                        break;

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

                        break;

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

```
                break;

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

                break;

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

                break;

        default:
                int_month --;

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

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

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

                }                                                       //
Ends if loop

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

```
                                        date_name[4] = '1';
                                }                                                       //
Ends else

                                break;

                        }                                                               //
End of month switch

                }                                                                       //
End of 1st of month if

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

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

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

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

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


                strcpy(day_string, day_dir);

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

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

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

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

```
                strcpy(del_dir, zip_path);


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

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

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

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

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

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

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

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

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

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

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

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

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

                printf("copy from %s to %s\n",temp_rpt,out_rpt);
                CopyFile(temp_rpt, out_rpt, FALSE);                  // Copy rpt file over to
                Sleep(1000);                                                        //
output server

                printf("command %s\n",temp_path);
                system(temp_path);                                                  // Run pkzip on
May2595.*
                Sleep(1000);
                        //temp_zip = "d:\50616GC1.zip"
                        //temp_zipped = "x:\gc1\50616GC1\50616GC1.*"
                        //temp_path = "c:\bin\pkzip d:\50616GC1.zip
                        //
x:\<dir>\50616GC1\50616GC1.*"
                        //temp_del = "x:\<dir>\50616GC1\50616GC1.*"
                        // where x:\<dir>\ are paths from the ini file

                strcat(out_path, date_name);           // Concatenate date_name onto out_path
                strcat(out_path, ext_zip);                   // Concatenate ext_zip onto out_path
```

```
                                                                        //
out_path = "x:\50616GC1.zip"
                                                                        // where
x:\ is out_path from ini file


                printf("Copy from %s to %s\n",temp_zip, out_path);
                CopyFile(temp_zip, out_path, FALSE);            // Copy zip file to
                Sleep(30000);                                           // output_path

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

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

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


                        Sleep(1000);                                            // Sleep 1
sec to allow file
                                                                                //
deletion to be updated by
                                                                                //
file system.
                        printf("Remove directory %s\n",del_dir);
                        RemoveDirectory(del_dir);               // remove the day directory

                        Sleep(1000);                                            // sleep 1
sec to allow
                                                                                //
window read.

                        remove(temp_zip);
                        printf("Remove %s\n",temp_zip);

                        Sleep(1000);                                            // Sleep 1
sec to allow file

                }

                else                    // If a GetLastError() not equal to 0 or 6...
                {
                        FindClose(findFile);                    // Close FindFirstFile handle

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


                        Sleep(1000);                                            // Sleep 1
sec to allow file
```

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

                    Sleep(1000);                                    // sleep 1
sec to allow
                                                                    //
window read.
                }
        Sleep(5000);
} //End of zip module
```

D.3 GC_FIL.CPP

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


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


void fil_name(char *sdat, int gc)
{

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

        time_t tm;
        size_t len;

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

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

        //switch on month sum and create temp date string
        //month day year
        switch(t){
                case 281: sdat[1]=48; sdat[2]=49; break; //Jan
                case 269: sdat[1]=48; sdat[2]=50; break; //Feb
                case 288: sdat[1]=48; sdat[2]=51; break; //Mar
                case 291: sdat[1]=48; sdat[2]=52; break; //Apr
                case 295: sdat[1]=48; sdat[2]=53; break; //May
                case 301: sdat[1]=48; sdat[2]=54; break; //Jun
                case 299: sdat[1]=48; sdat[2]=55; break; //Jul
                case 285: sdat[1]=48; sdat[2]=56; break; //Aug
```

```
                case 296: sdat[1]=48; sdat[2]=57; break; //Sep
                case 294: sdat[1]=49; sdat[2]=48; break; //Oct
                case 307: sdat[1]=49; sdat[2]=49; break; //Nov
                case 268: sdat[1]=49; sdat[2]=50; break; //Dec
                default: cout<<"ERROR IN SWITCH TDC.CCP\n";
                }

        itoa(gc, str_gc, 10);

        sdat[0] = tms[23];
        sdat[3] = tms[8];
        sdat[4] = tms[9];
        sdat[5] = 67;//
        sdat[6] = 70;//          Will be replacing these with data from
        sdat[7] = 70;//     *.ini file.
        sdat[8] = 46;//
        sdat[9] = 0;

        return;
}
```

D.4 EXP2.CPP

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

#include <iostream.h>
#include <fstream.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <windows.h>
#include "GC1ZIP.h"

struct fileInfo
 {
        char col1[3],col2[6],col3[6],col4[6],col5[9],col6[7],
                        col7[9],col8[9],col9[9] /*,col10[9] */ ;

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

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

        char index_ext[5];
        int max,count,n,i;
        char  * calibr[999];
```

```
        float H2[999];
        float CH4[999];
        float N2O[999];
//      float hnitro[999];
        char base[50];

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


        ifstream infile(filename);

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

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

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

// Changes concentration values from character to float.
                calibr[count] = (file[count].col1);
                H2[count]  = atof (file[count].col7);
                CH4[count] = atof (file[count].col8);
                N2O[count] = atof (file[count].col9);
//              hnitro[count] = atof (file[count].col10);

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

                if(infile.eof())
                {

                        n = 1010;
                }

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

        }// Ends the while statement.


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

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

                if ((i % N == 0) || H2[i] == 0.000 || CH4[i] == 0.000
```

```
                                                    || N2O[i] == 0.000 || calibr[iii[0] != 79)
                        {
                                cout << crname << " chromatogram was saved \n";
                                cout << "\n";
                                Sleep(500);
                        }
                        else
                        {
                                cout << crname << " chromatogram was deleted \n";
                                remove(crname);
                                cout << "\n";
                                Sleep(500);
                        }
        crname="\0";
                        strcpy(crname,base); //Reset the characters in crname.
          }

// Close the input file.
        ifstream::close;
        cout << "End of Compress Section\n";
        cout << "\n";
        Sleep(1000);
        return;
}
```

D.5 INDEX3.CPP

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

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

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

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

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

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

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

        return;
}
```

D.6 GC1ZIP.H

```
// Defines of user-created functions

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

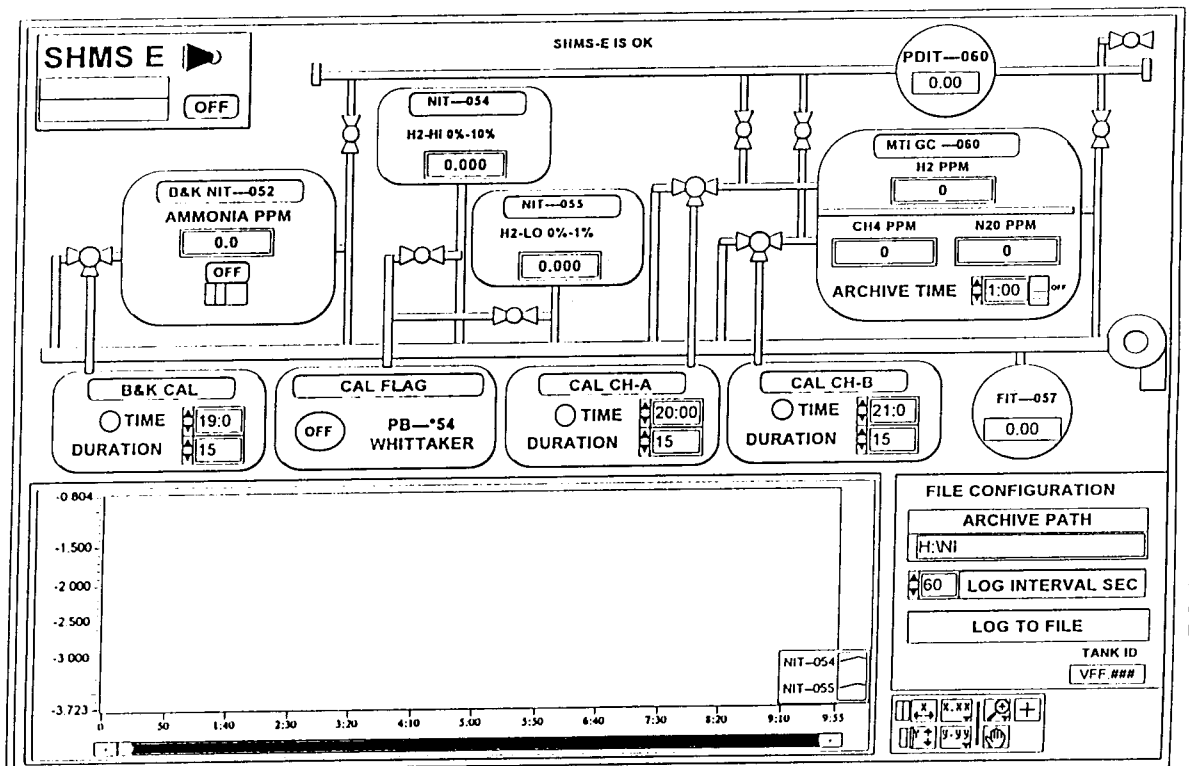
D.7 FUNCTION.H

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

Shms-e.vi
Last modified on 4/23/97 at 10:40 AM

SHMS

Front Panel



Connector Pane

E.1 SHMS-E.VI

[redacted]

C:\LABVIEW\SHMS-E\Shms-e.vi

Simple data acquisiton and display VI that makes use of handy subVIs. Acquires data with a DA optional multiplexers or SCXI modules) and scales data to Volts.

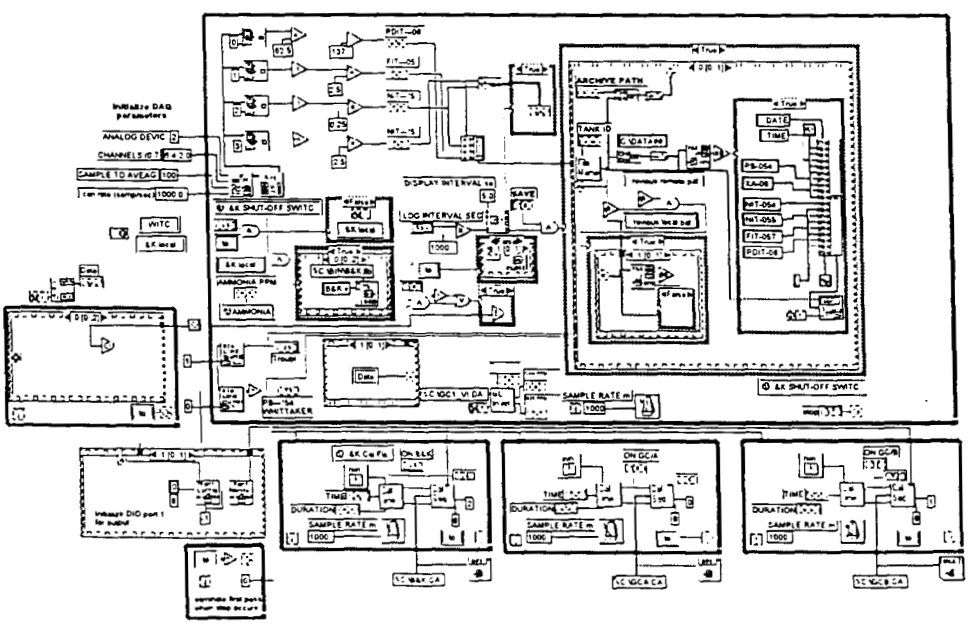
To cut down on overhead, the subVI Interval Timer controls how often the stripchart is updated a data are appended to the data file. You can add more displays to the Select structure that contains you like.

Data are saved in tab-delimited text format. Headers are written first. The first column is a time s are in decimal hours format, where 3:15PM would be 15.250 hours, with a resolution of 1 ms (o current version of LabVIEW supplies).

Performance: Making the display interval very long speeds things up by reducing graphics overh averaging is somewhat of a limiting factor, as is the conversion of numeric data to strings and the VI. Therefore, this VI is really at its best for sampling rates of less than about 5 Hz.

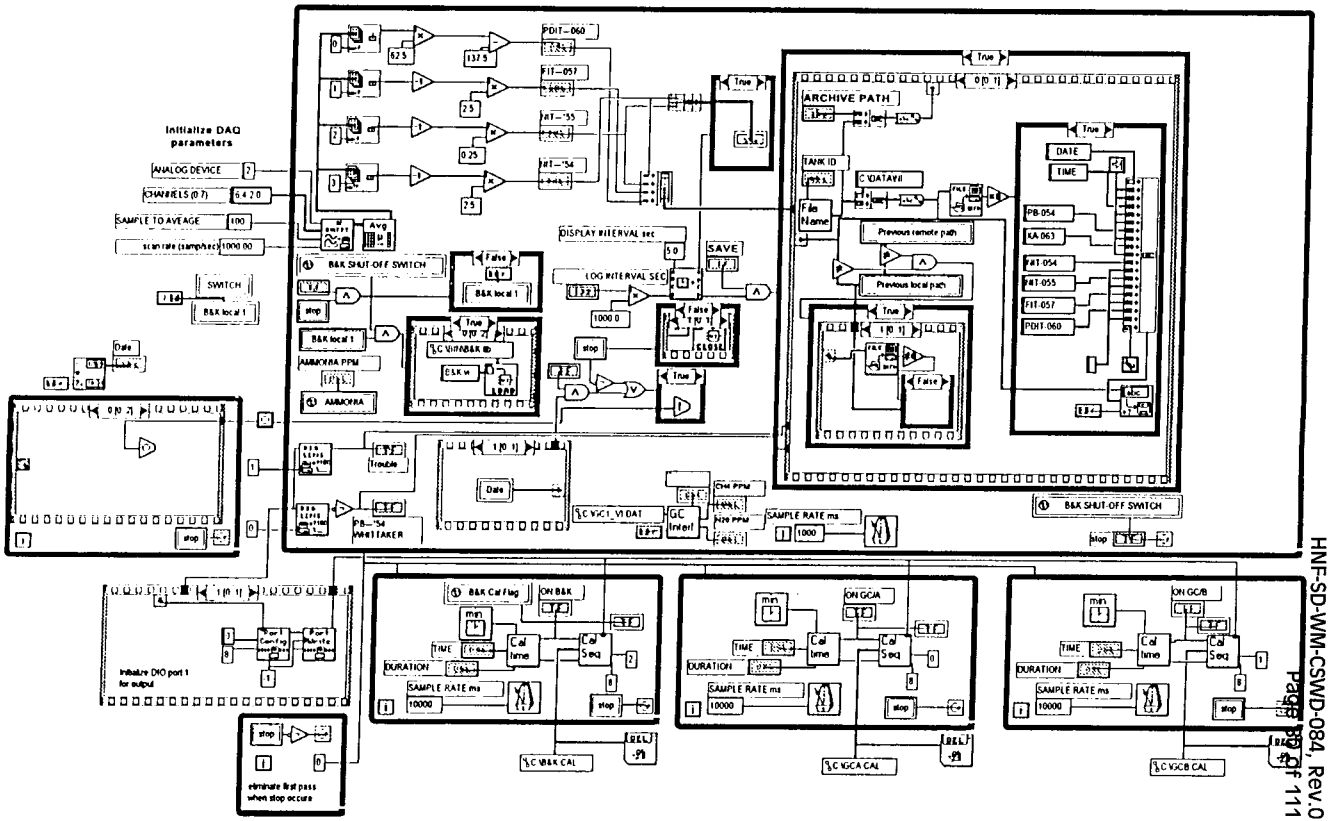
GW Johnson, 3-93

Block Diagram

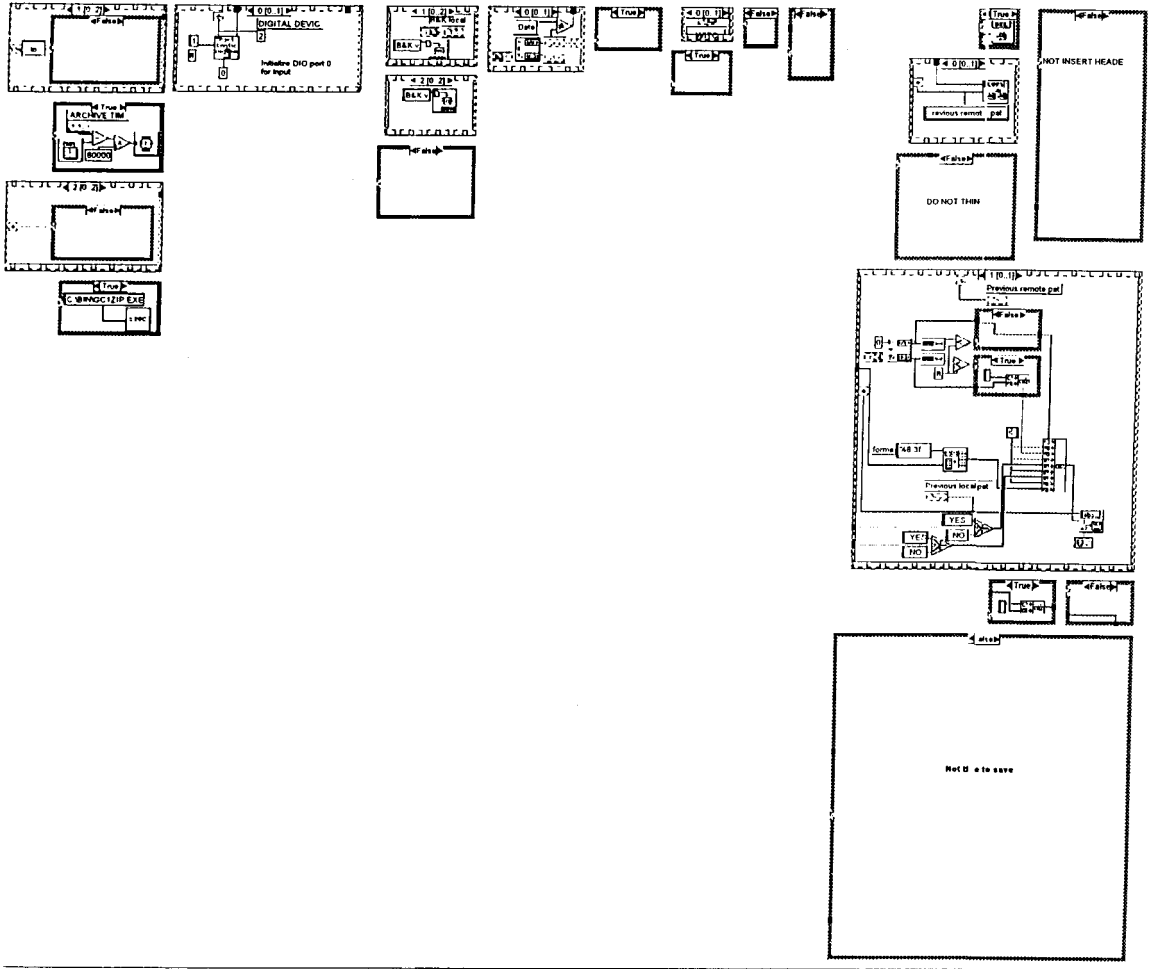


Shms-e.vi
Last modified on 4/23/97 at 10:40 AM

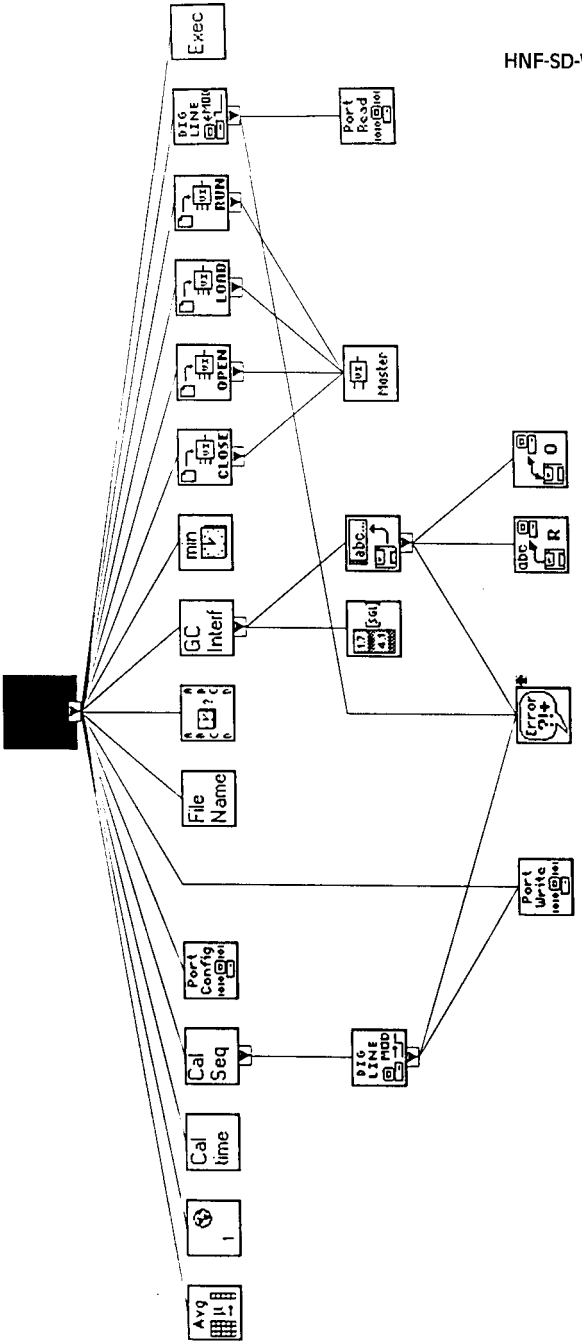
Block Diagram



Shms-e.vi
Last modified on 4/23/97 at 10:40 AM



Position in Hierarchy


Exec
DIG LINE
Port Read
RUN
LOAD
OPEN
Master
CLOSE
min
GC Interf
File Name
Port Config
Cal Seq
Cal time
Avg
Error
Port Write
abc
R
0

Write Characters To File.vi
C:\LABVIEW\vi.lib\Utility\FILE.LLB\Write Characters To File.vi

AI Acquire Waveforms.vi
C:\LABVIEW\vi.lib\DAQ\1easyio.llb\AI Acquire Waveforms.vi

Average Voltage.vi
C:\LABVIEW\SHMS-E\Average Voltage.vi

Interval Timer.vi
C:\LABVIEW\SHMS-E\Interval Timer.vi

GC INTERFACE.vi
C:\LABVIEW\SHMS-E\GC INTERFACE.vi

CAL SEQUENCE 1.VI
C:\LABVIEW\SHMS-E\CAL SEQUENCE 1.VI

Minutes now.vi
C:\LABVIEW\SHMS-E\Minutes now.vi

DIO Port Config.vi
C:\LABVIEW\SHMS-E\DIO Port Config.vi

DIO Port Write.vi
C:\LABVIEW\SHMS-E\DIO Port Write.vi

filename modify.vi
C:\LABVIEW\SHMS-E\filename modify.vi

Read from Digital Line modify.vi
C:\LABVIEW\SHMS-E\Read from Digital Line modify.vi

BK.gbl
C:\LABVIEW\SHMS-E\BK.gbl

CAL COMPARE TIME.VI
C:\LABVIEW\SHMS-E\CAL COMPARE TIME.VI

System Exec.vi
C:\LABVIEW\SHMS-E\System Exec.vi

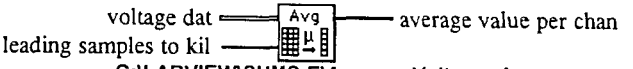
Preload Instrument.vi
C:\LABVIEW\SHMS-E\Preload Instrument.vi

Run Instrument.vi
C:\LABVIEW\SHMS-E\Run Instrument.vi

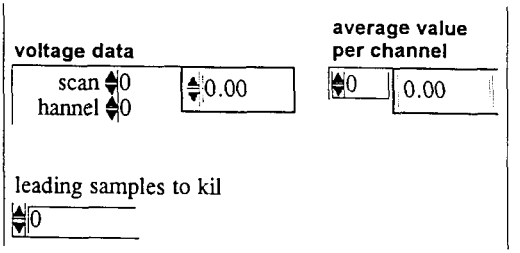
Open Panel.vi
C:\LABVIEW\SHMS-E\Open Panel.vi

Close Panel.vi
C:\LABVIEW\SHMS-E\Close Panel.vi

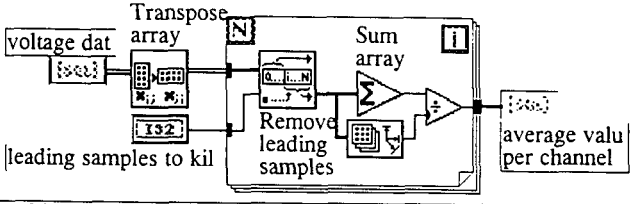
Connector Pane



Front Panel

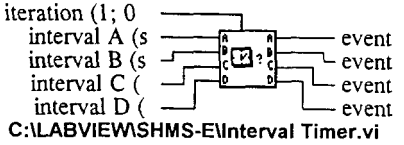


Block Diagram

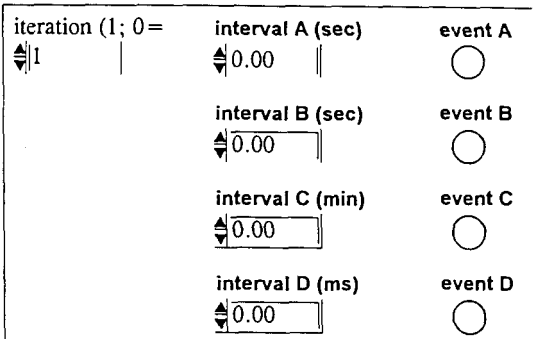


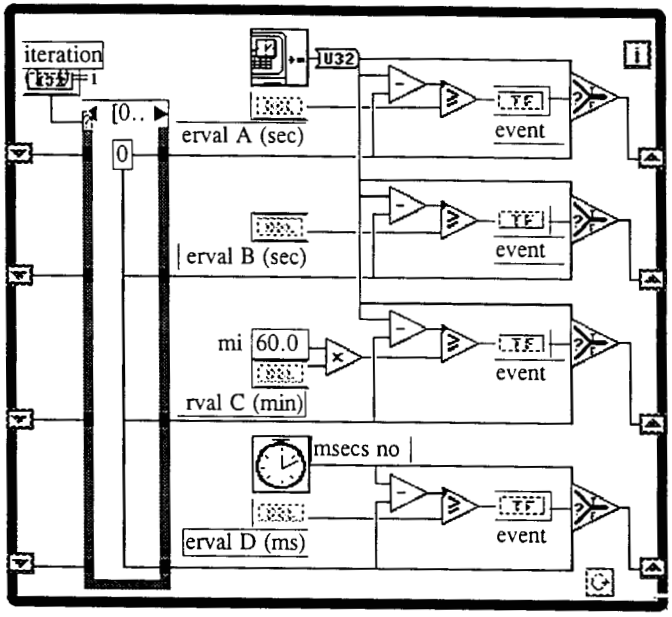
List of SubVIs

Connector Pane



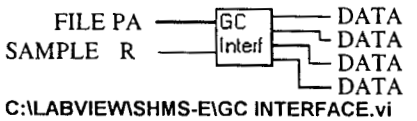
Front Panel



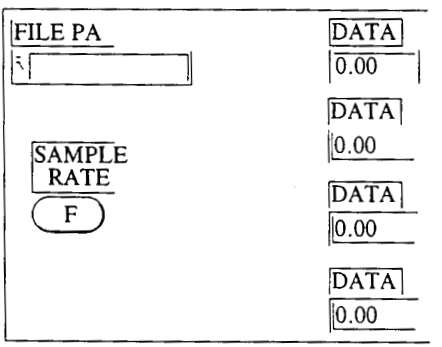

iteration
[0..
0
erval A (sec)
U32
event
erval B (sec)
mi
60.0
rval C (min)
msecs no
erval D (ms)


[0..

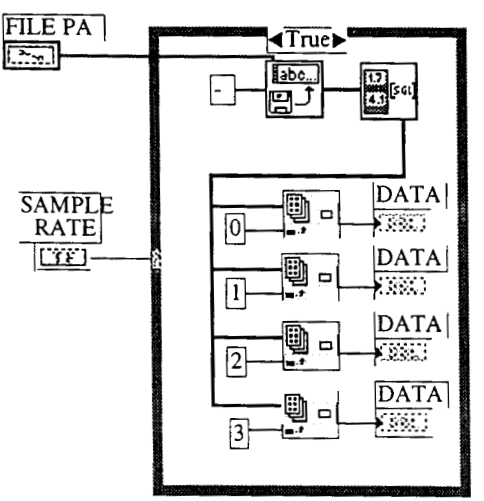
Connector Pane

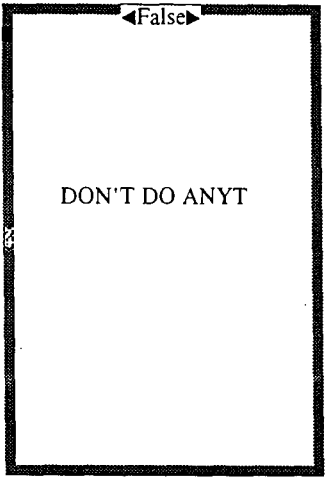


Front Panel



Block Diagram



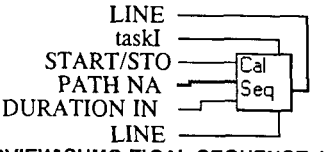


List of SubVIs

Read Characters From Filecvv.vi
C:\LABVIEW\SHMS-E\Read Characters From Filecvv.vi

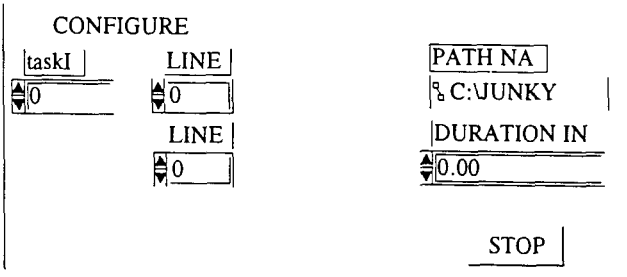
Extract Numbers.vi
C:\LABVIEW\SHMS-E\Extract Numbers.vi

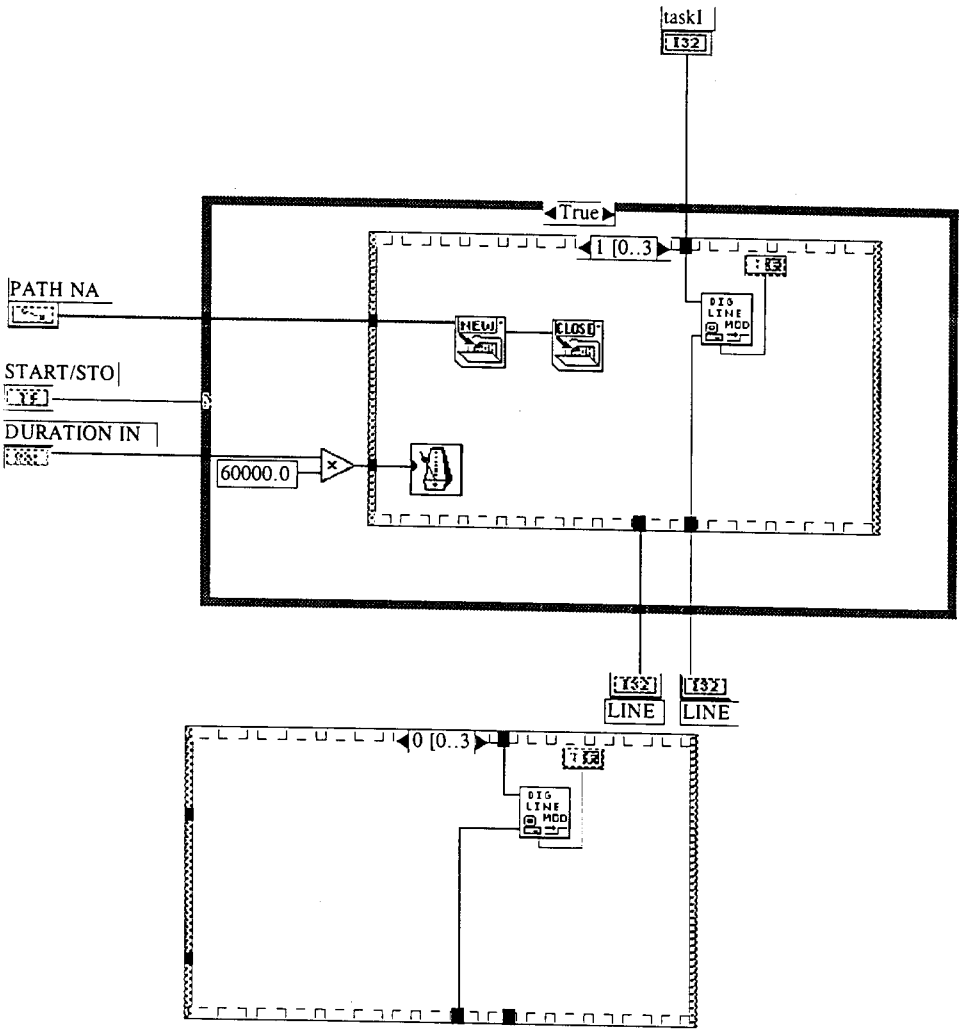
Connector Pane

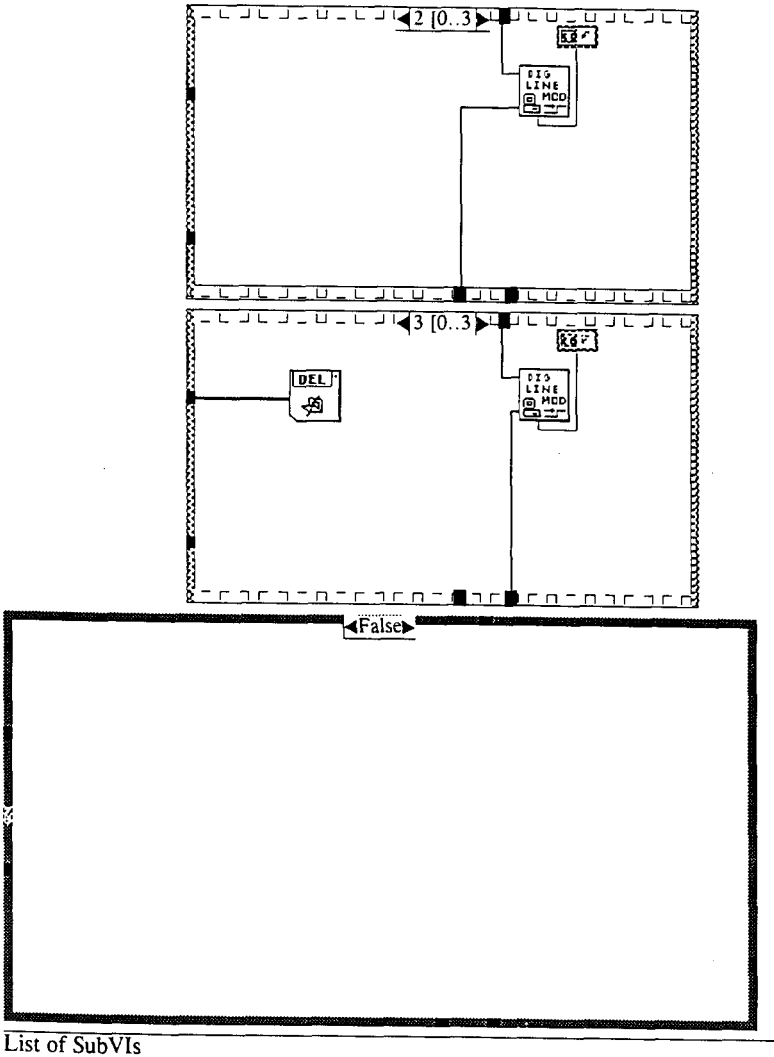


C:\LABVIEW\SHMS-E\CAL SEQUENCE 1.VI

Front Panel




task1
I32
True
1 [0..3]
I32
DIG LINE MOD
NEW
CLOSE
PATH NA
START/STO
TF
DURATION IN
60000.0
I32
LINE
I32
LINE
0 [0..3]
I32
DIG LINE MOD



List of SubVIs

Write to Digital Line modify.vi
C:\LABVIEW\SHMS-E\Write to Digital Line modify.vi

Connector Pane

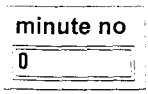
:\LABVIEW\SHMS-E\Minutes now.

This VI returns the decimal hour equivalent of the current time. For instance, if it is now 3:45 PM, 'decimal 15.75.

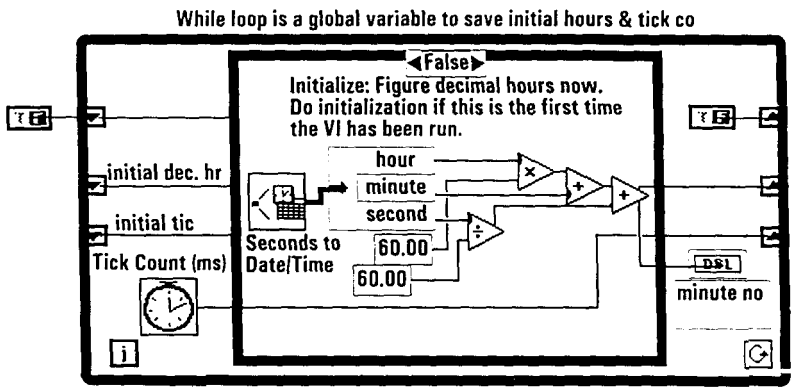
Resolution is 1 ms, or whatever the limitation of this implementation of LabVIEW might be.

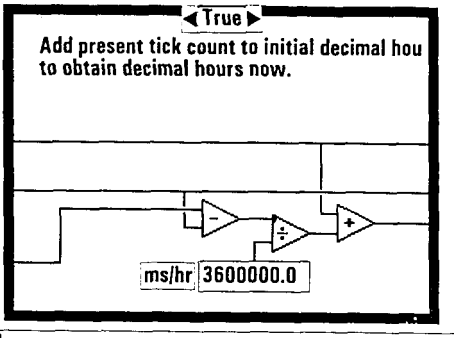
Time overflows after 2^32 ms, which is 50 days.

Front Panel



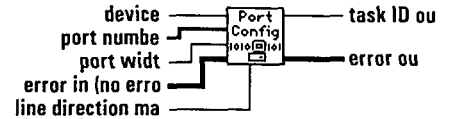
Block Diagram





List of SubVIs

Connector Pane



C:\LABVIEW\SHMS-E\DIO Port Config.vi

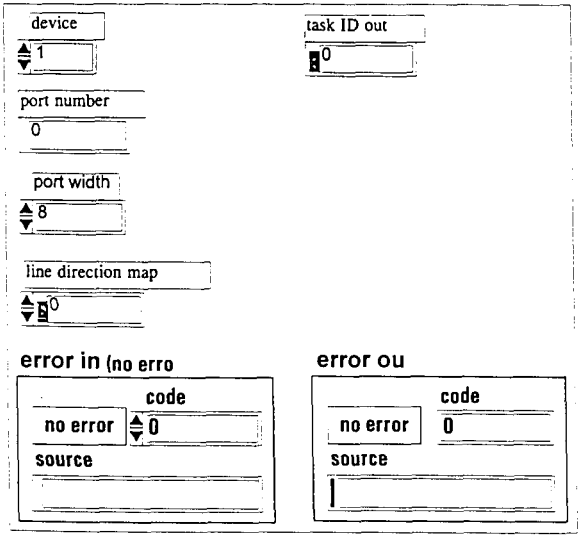
This VI establishes a port configuration. You can use the task ID that this VI returns only in digital port VIs.

NOTE: When you call this VI on a digital I/O port that is part of an 8255 PPI, all the ports within the same P to logic low regardless of the data direction. The data directions on other ports, however, are maintained.

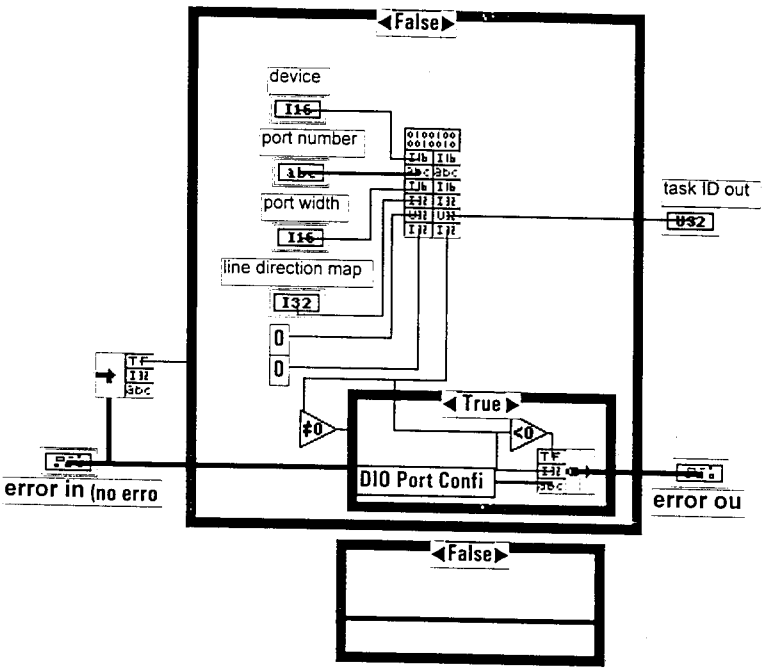
Port Width for all digital ports on the Lab boards, SCXI-1200, DAQCard-1200, DAQPad-1200, DIO-24, DIO-3 and ports 2, 3 and 4 on the AT-MIO-16DE-10 and AT-MIO-16D should be at least 8.

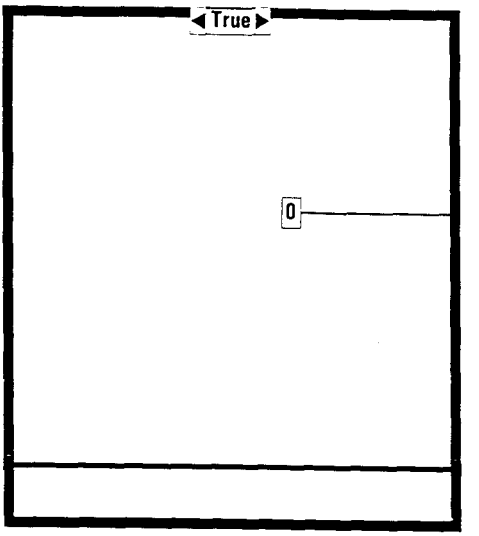
See the Data Acquisition VI Reference Manual for your platform for tables listing the default settings and r for the boards with which it will work.

Front Panel



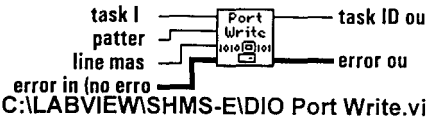
Block Diagram





List of SubVIs

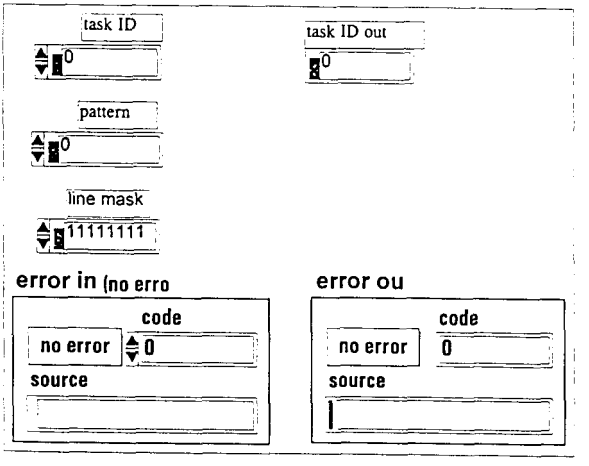
Connector Pane



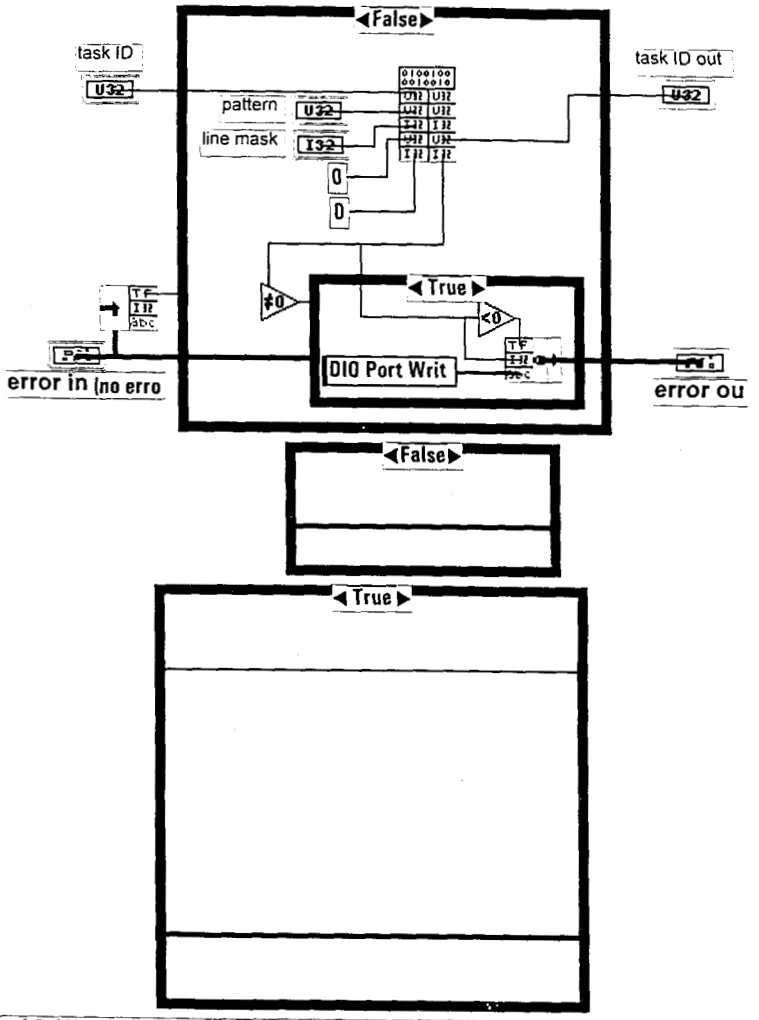
This VI writes the value in pattern to the port identified by task ID.

See the Data Acquisition VI Reference Manual for your platform for tables listing the default settings and r for the boards with which it will work.

Front Panel

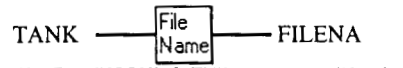


Block Diagram



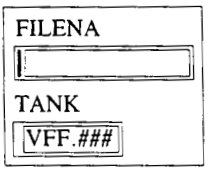
List of SubVIs

Connector Pane

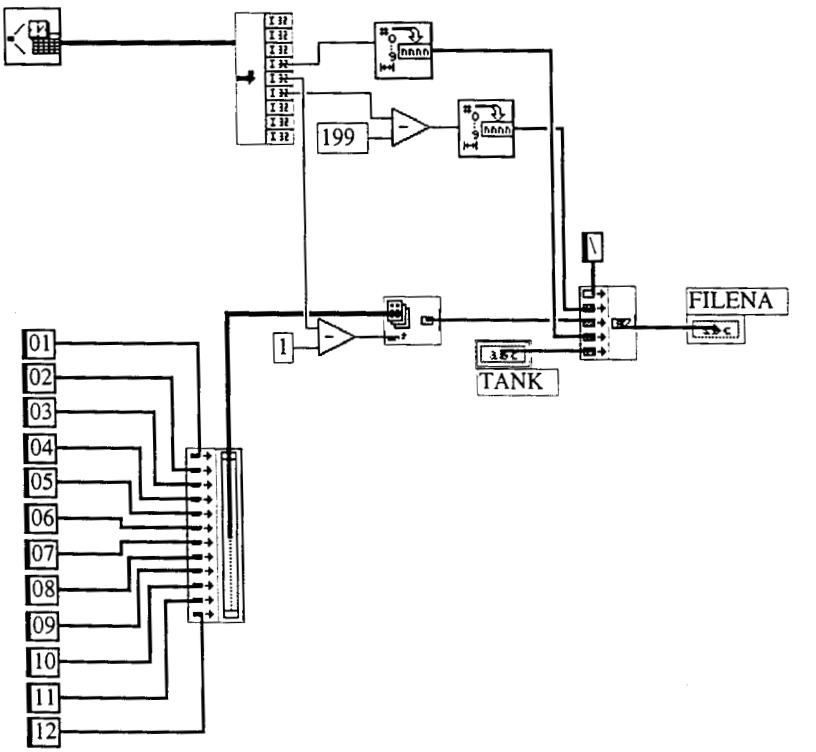


C:\LABVIEW\SHMS-E\filename modify.vi

Front Panel



Block Diagram



Connector Pane

C:\LABVIEW\SHMS-E\Read from Digital Line modify.vi

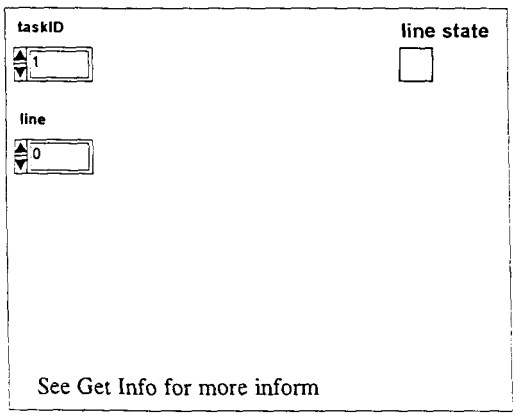
Most applications need only connect wires to the controls and indicators displayed in bold. Conn controls and indicators are useful only if specifically needed.

The Read from Digital Line VI reads the logical state of a digital line on a user configured port. option of not reconfiguring the port. A boolean indicates the value read from the line as either lo low.

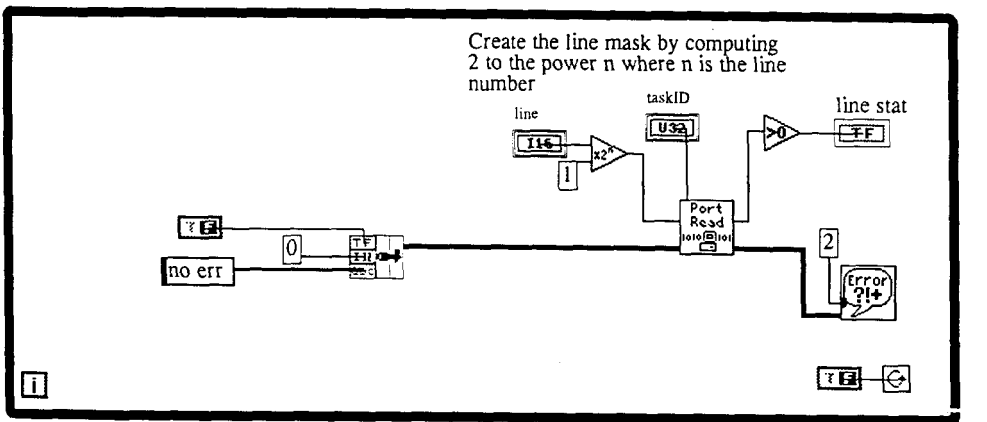
Port Width for the Lab boards, SCXI-1200, DAQBox-1200, DIO-24, DIO-32F, DIO-96, and AT and 4 should be at least 8.

Please read the DAQ VI Reference Manual for descriptions of Digital Port Configure and Digital information.

Front Panel



Block Diagram



List of SubVIs

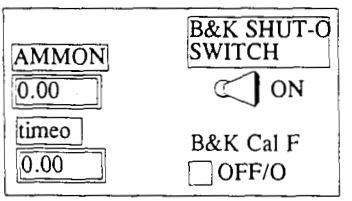
General Error Handler.vi
C:\LABVIEW\SHMS-E\General Error Handler.vi

DIO Port Read.vi
C:\LABVIEW\SHMS-E\DIO Port Read.vi

Connector Pane

C:\LABVIEW\SHMS-E\BK.gbl

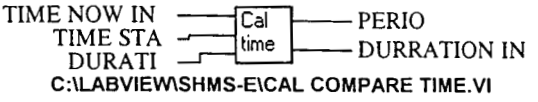
Front Panel



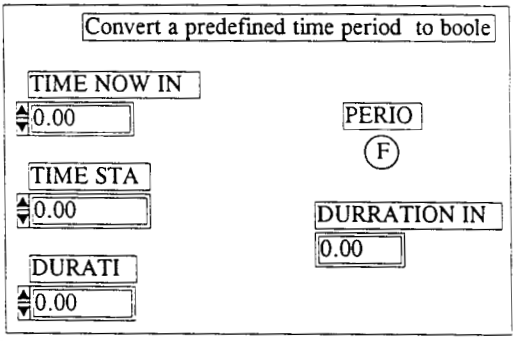
Block Diagram

List of SubVIs

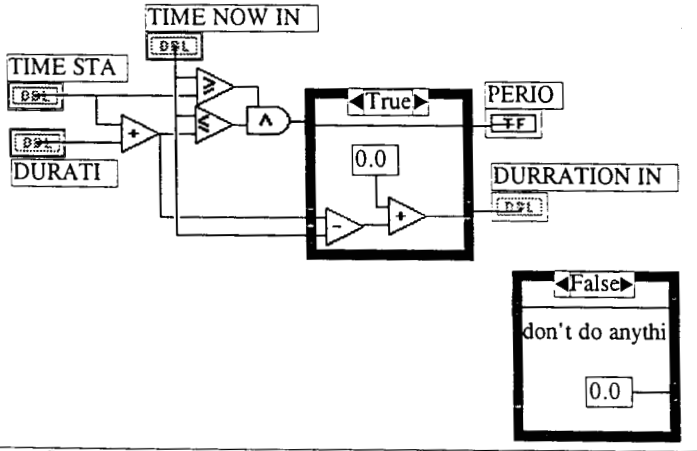
Connector Pane



Front Panel

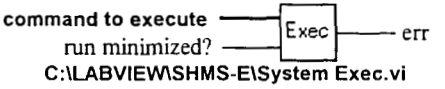


Block Diagram



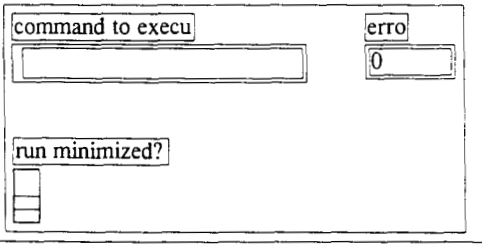
List of SubVIs

Connector Pane

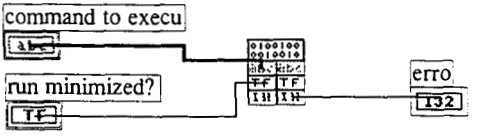


This VI calls the winexec command, allowing you to execute a system level command.

Front Panel



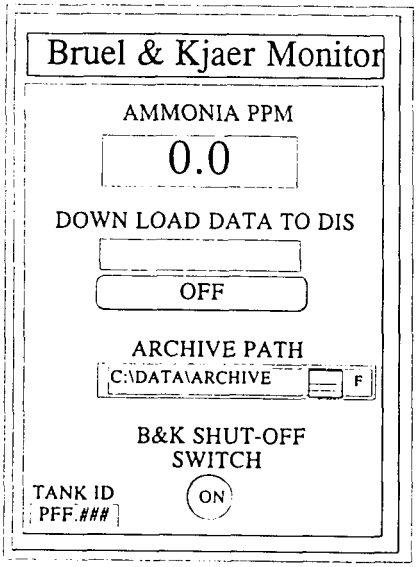
Block Diagram



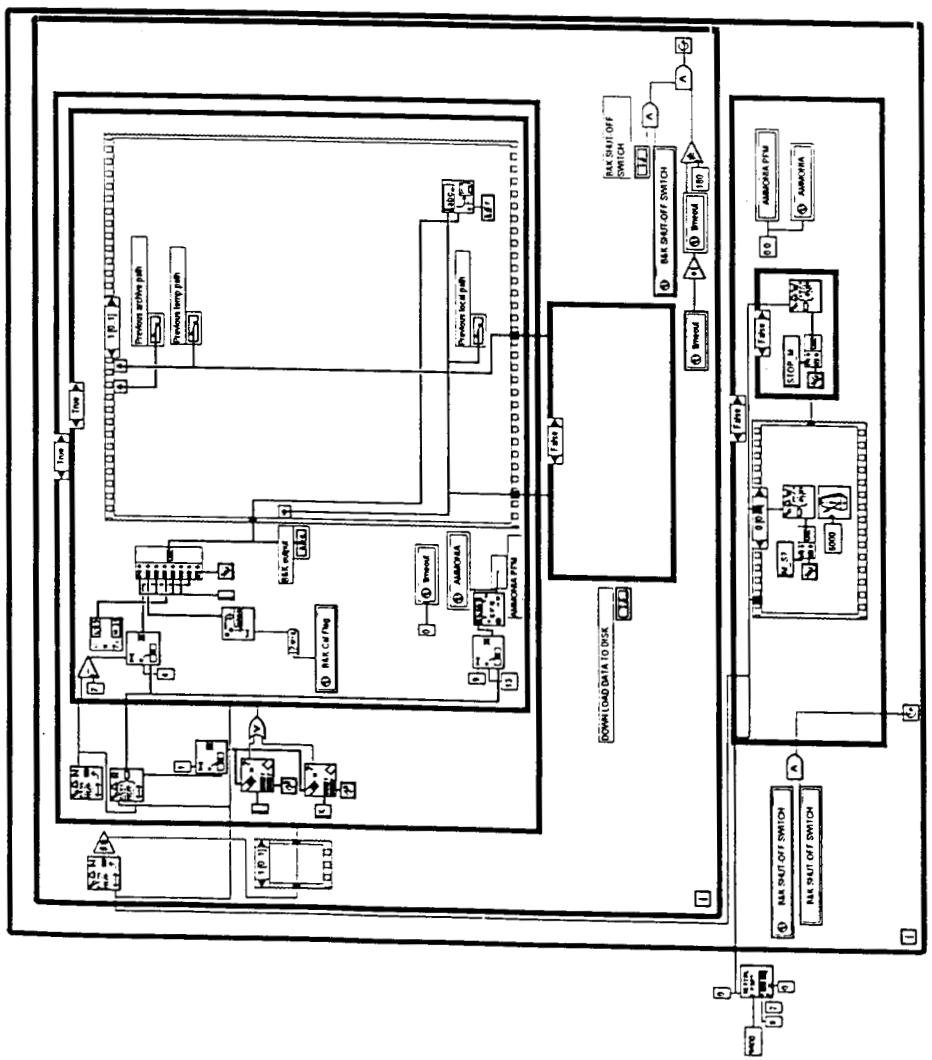
List of SubVIs

E.2 B&K.VI

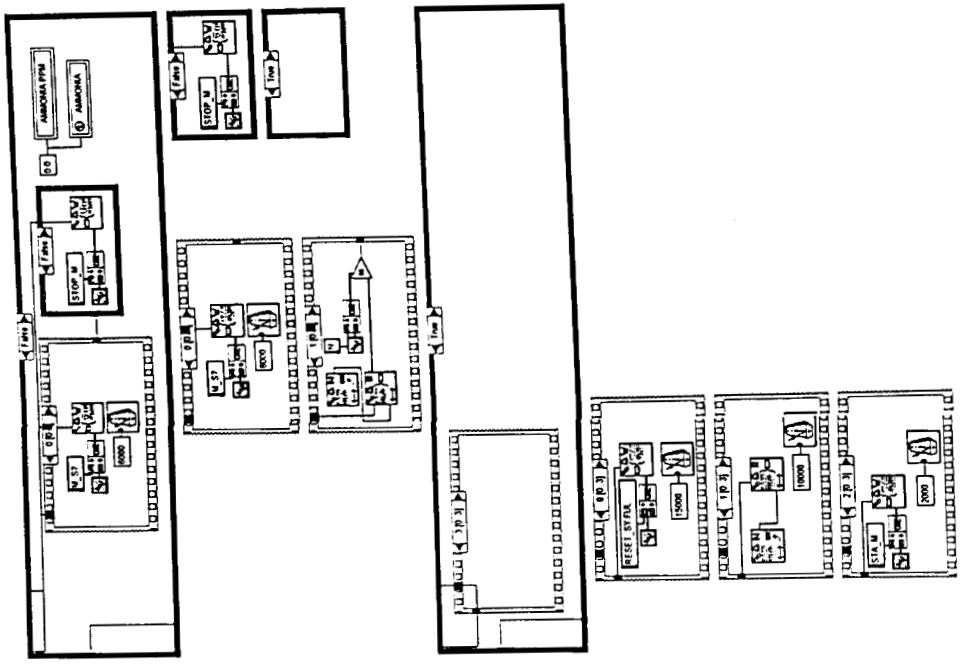
Front Panel

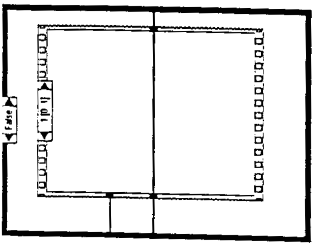


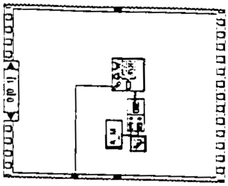
Block Diagram

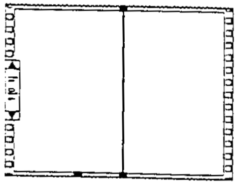


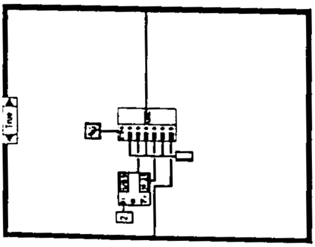
Frame
Time
DOWN LOADING
SAVE LABLE
Frame
DOWN LOADING
SAVE LABLE
Frame
SAVE LABLE
Frame
C:\DATA\AK_DISK
Frame
C:\DATA\AK_DISK
SAVE LABLE
DONE
10000
Frame
SAVE LABLE
Previous temp path
Previous test path

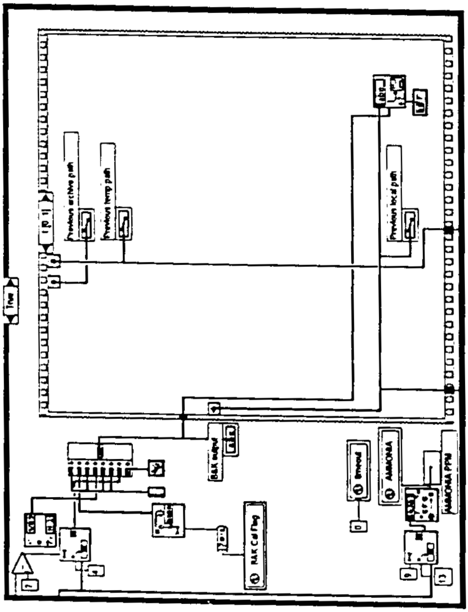

AMMONIA PPM
AMMONIA
STOP M
M-57
10000
RESET SYNC
15000
STA M
2000


False

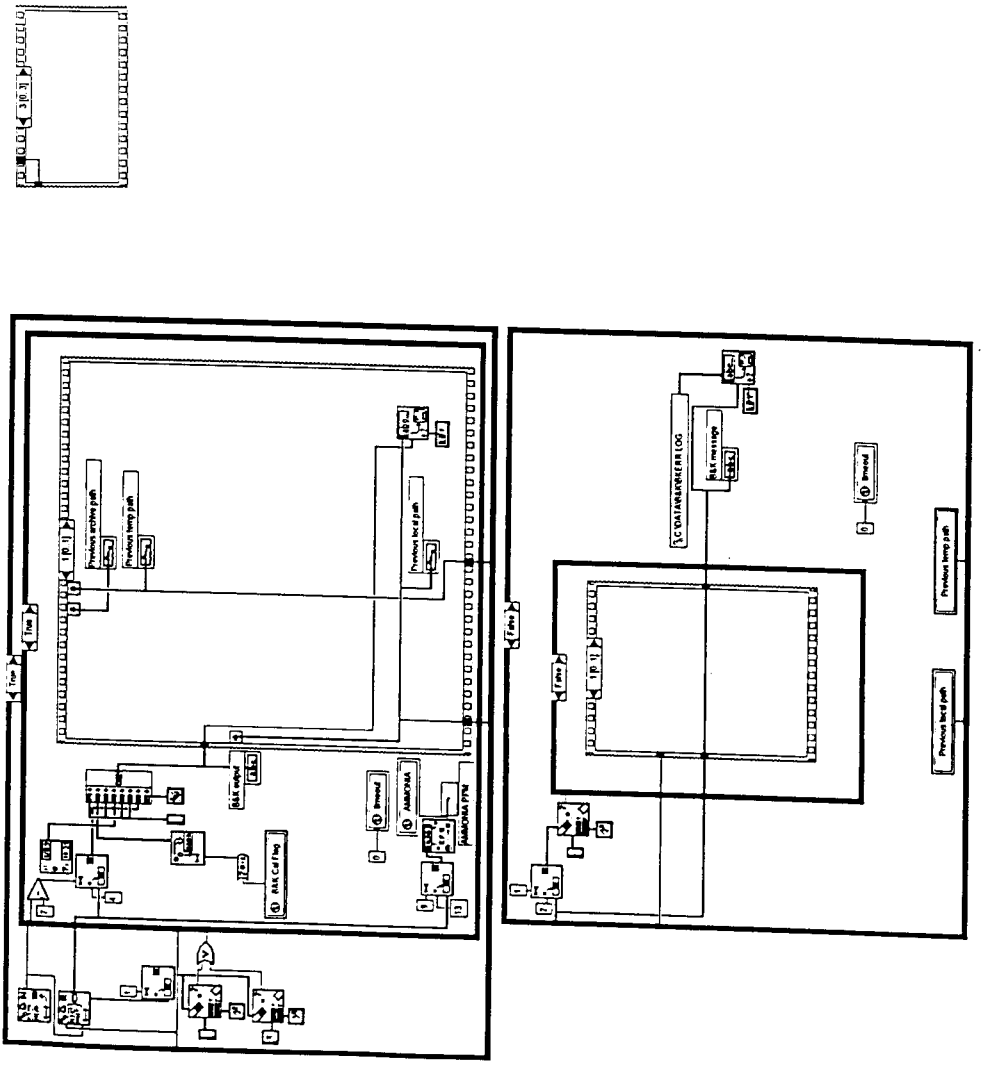





True


True
Previous archive path
Previous temp path
Previous local path
B&K output
B&K Cal Flag
Timeout
AMMONIA
AMMONIA PPM

Previous archive path
Previous temp path
Previous local path
CAL
DATE
TIME
AMMONIA
WATER
ARCHIVE PATH
DATABASE DISK
DATABASE
File Name
TANK ID
NOT INSERT HEADER



DISTRIBUTION SHEET

To	From	Page 1 of 1
DISTRIBUTION	B.L. Philipp SESC	Date 5/29/97
Project Title/Work Order		EDT No. 619510
Computer Systems and Software Description for Standard-E+ Hydrogen Monitoring System (SHMS-E+)		ECN No.

Name	MSIN	Text With All Attach.	Text Only	Attach. / Appendix Only	EDT/ECN Only
T.C. Schneider	L6-37	X			
D.D. Tate	L6-37	X			
M.F. Erhart	R1-51	X			
R.L. Schlosser	R1-56				X
L.D. Salsberry	B7-41	X			
S.U. Zaman	R3-08	X			
B.L. Philipp (3)	L6-37	X			
Central Files	A3-88	X			