TECHNICAL REPORT STANDARD TITLE PAGE

1. Report No. FHWA/TX-86/52+421-2	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Golden River Traffic Counter Analysis and Data Strorage for the IBM-PC		5. Report Date May 1986
		6. Performing Organization Code
7. Author(s) Jae Y. Lee, Duk Jin Chang and Gene P. Ritch		8. Performing Organization Report No. Research Report 421-2
9. Performing Organization Name and Address Texas Transportation Institute The Texas A&M University System College Station, TX 77843		10. Work Unit No.
		11. Contract or Grant No. Study No. 2-18-84-421
12. Sponsoring Agency Name and Address State Department of Highways and Public Transportation Transportation Planning Division P.O. Box 5051 Austin, TX 78763		13. Type of Report and Period Covered Interim/September 1983 May 1986
		14. Sponsoring Agency Code

15. Supplementary Notes

Research performed in cooperation with FHWA, DOT.
Research Study Title: Golden River Traffic Counter Analysis and Data Storage for the IBM-PC.

16. Abstract

This report is an operations guide for inputting traffic volume data from the automatic Golden River counters into an IBM-PC or look-alike, storing data files on floppy diskettes and providing printed analysis output data. Common serial communication support program PC-TALK III is used to read and store traffic volume data. Turbo Pascal written programs HEADER and REPORT enable user to provide descriptive information for each data file. Data analysis output can be obtained in sub-hourly, hourly and daily form. Requires an IBM-PC or look-alike, dual diskettes, 256k, variable character size printer, and an unused asynchronous communication (serial) port. Golden River and Turbo Pascal documentation included in Appendices.

17. Key Words Microcomputer, PC-TALK III, Turbo Pascal Programs, Traffic Volume Data		18. Distribution Statement No Restrictions. This document is available to the public through the National Technical Information Service 5285 Port Royal Road, Springfield Virginia 22161	
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 86	22. Price

Form DOT F 1700.7 (8-69)

GOLDEN RIVER TRAFFIC COUNTER ANALYSIS AND DATA STORAGE FOR THE IBM-PC

By

Jae Y. Lee
Systems Analyst

Duk Jin Chang
Research Assistant

and

Gene P. Ritch
Associate Research Specialist

Research Report 421-2
Research Study 2-18-84-421
Developing a Freeway Data Base Model

Sponsored by

Texas State Department of Highways and Public Transportation

In Cooperation with
The U.S. Department of Transportation
Federal Highway Administration

TEXAS TRANSPORTATION INSTITUTE
The Texas A&M University System
College Station, TX 77843

May 1986

METRIC CONVERSION FACTORS

Approximate Conversions to Metric Measures

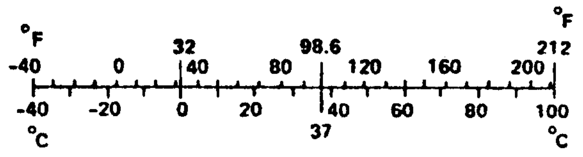
Symbol	When You Know	Multiply by	To Find	Symbol
		LENGTH		
in	inches	*2.5	centimeters	cm
ft	feet	30	centimeters	cm
yd	yards	0.9	meters	m
mi	miles	1.6	kilometers	km
		AREA		
in^2	square inches	6.5	square centimeters	cm^2
ft^2	square feet	0.09	square meters	m^2
yd^2	square yards	0.8	square meters	m^2
mi^2	square miles	2.6	square kilometers	km^2
	acres	0.4	hectares	ha
		MASS (weight)		
oz	ounces	28	grams	g
lb	pounds	0.45	kilograms	kg
	short tons (2000 lb)	0.9	tonnes	t
		VOLUME		
tsp	teaspoons	5	milliliters	ml
Tbsp	tablespoons	15	milliliters	ml
fl oz	fluid ounces	30	milliliters	ml
c	cups	0.24	liters	l
pt	pints	0.47	liters	l
qt	quarts	0.95	liters	l
gal	gallons	3.8	liters	l
ft^3	cubic feet	0.03	cubic meters	m^3
yd^3	cubic yards	0.76	cubic meters	m^3
		TEMPERATURE (exact)		
°F	Fahrenheit temperature	5/9 (after subtracting 32)	Celsius temperature	°C

*1 in = 2.54 (exactly). For other exact conversions and more detailed tables, see NBS Misc. Publ. 286, Units of Weights and Measures, Price $2.25, SD Catalog No. C13.10:286.

inches
1
2
3
4
5
6
7
8
9
cm
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23

Approximate Conversions from Metric Measures

Symbol	When You Know	Multiply by	To Find	Symbol
		LENGTH		
mm	millimeters	0.04	inches	in
cm	centimeters	0.4	inches	in
m	meters	3.3	feet	ft
m	meters	1.1	yards	yd
km	kilometers	0.6	miles	mi
		AREA		
cm^2	square centimeters	0.16	square inches	in^2
m^2	square meters	1.2	square yards	yd^2
km^2	square kilometers	0.4	square miles	mi^2
ha	hectares (10,000 m^2)	2.5	acres	
		MASS (weight)		
g	grams	0.035	ounces	oz
kg	kilograms	2.2	pounds	lb
t	tonnes (1000 kg)	1.1	short tons	
		VOLUME		
ml	milliliters	0.03	fluid ounces	fl oz
l	liters	2.1	pints	pt
l	liters	1.06	quarts	qt
l	liters	0.26	gallons	gal
m^3	cubic meters	35	cubic feet	ft^3
m^3	cubic meters	1.3	cubic yards	yd^3
		TEMPERATURE (exact)		
°C	Celsius temperature	9/5 (then add 32)	Fahrenheit temperature	°F



ABSTRACT

This report is an operations guide for inputting traffic volume data from the automatic Golden River counters into an IBM-PC or look-alike, storing data files on floppy diskettes and providing printed analysis output data. Common serial communication support program PC-TALK III is used to read and store traffic volume data. Turbo Pascal written programs HEADER and REPORT enable user to provide descriptive information for each data file. Data analysis output can be obtained in sub-hourly, hourly and daily form. Requires an IBM-PC or look-alike, dual diskettes, 256K, variable character size printer, and an unused asynchronous communication (serial) port. Golden River and Turbo Pascal documentation included in Appendices.

Key Words: Microcomputer, PC-TALK III, Turbo Pascal Programs, Traffic Volume Data

SUMMARY

The Golden River MARKSMAN counter/classifier is used to collect traffic volume data on a roadway using either pneumatic tubes or inductive loops as sensors. Once the data is collected, it is transferred to a floppy diskette for processing and storage or printed out directly onto a serial driven printer in two possible formats: (1) the data transfer between the Retriever and an IBM-PC is done using the PC-TALK utility which is a general purpose communications software designed for IBM-PC; or (2) the IBM-PC system must have an asynchronous communications card to support the serial communications with external devices.

Several software communication parameters must be properly set in both the Retriever and PC-TALK program. Once correctly established, each file in the Retriever can be passed into the PC for storage.

The data analysis programs are HEADER and REPORT. The HEADER program interacts with the user to obtain information (site location, weather, comments, etc.) and appends this information to the original data file. REPORT reduces the data and produces daily and weekly summary reports for each sensor.

Emphasis is placed on the descriptive logic flow in each subprogram of HEADER and REPORT. The Turbo Pascal Program listings are included in the appendix.

IMPLEMENTATION STATEMENT

For those users of the Golden River traffic counters that have access to an IBM-PC (or look alike), the programs described in this report will enable faster data analysis than through conventional methods as well as retaining non-hard copy of data on diskette(s).

DISCLAIMER

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

In the planning, design and operation of a transportation system, timely traffic data provides traffic engineers with the factual information needed to plan, design, maintain, and manage the highway systems. The Golden River traffic counter/classifier system is used to collect traffic volume counts on a roadway. These traffic counter/classifier devices can be configured to collect up to six channels of traffic volume counts of various count interval periods. More frequently used count intervals are 5, 10, 15 and 60 minutes. The summary of the volume counts per channel can be obtained from the counter/classifier according to the time interval periods. However, if the output format as presented by the Retriever is needed in another format, time consuming manual effort is required to transcribe from one form to another. This report documents the development of a microcomputer based software that will automate the previous manual efforts required to transcribe the raw data from the Retriever to a useful summary report. The methods used in this software to reduce the raw volume counts has been developed as a result of many hours of manual data reduction efforts.

This report documents a method to transfer the data from the Retriever directly onto a floppy diskette for processing and permanent storage. The processing software reduces the data and produces two major reports, daily traffic volume count summary according to the count interval and the weekly traffic volume for one hour count interval.

The procedures and programs discussed in this report were developed using an IBM-PC operating under the MS-DOS, version 2.1. The system requirements for this software include an IBM-PC or a compatible system with two floppy drives and minimum of 256k RAM. The data analysis software was developed in Turbo PASCAL by the request from the SDHPT.

SITUATION

The Golden River counter/classifier traffic data collection system is used by traffic engineers and researchers to obtain traffic volume counts on a roadway system. The counter/classifier may be used with either pneumatic tubes or inductive loops to collect traffic volume data. Once the pneumatic tubes are placed onto the roadway and connected (or inductive loops connected), the counter can be programmed to provide sensory information for data collection in several configurations. Data collection configurations, time, date, etc. are all established by connecting the Retriever to the counter. Once the data is collected, it can be stored into the Retriever.

The data is then transferred to the office via the Retriever and can be either printed out directly onto a printer or transferred onto a floppy disk for processing and storage. The data in output form as presented from Golden River is shown in Figure 1. From this data the hourly volume, peak hour volume, daily ADT, AM peak volume, PM peak volume, and other information is derived. Until recently, this data reduction process was carried out manually which introduced human errors and was very time consuming. The software described in this documentation may be used to analyze the data from the Retriever to produce the reports. The data transfer between the Retriever and an IBM-PC is done using the PC-TALK software utility which is a general purpose communications software designed for the IBM-PC. The first portion of this report describes the data transfer and storage procedures for both hardware and software configuration. The data reduction and analysis procedures are described in the following sections.

DATA TRANSFER AND STORAGE

I. Hardware Configuration

The IBM-PC microcomputer system must have an asynchronous commumications card to support the serial commumications with external devices. This RS-232 serial port is generally configured as communications port 1 (COM1:) in IBM-PC device assinments. The Retriever can be connected directly to the IBM-PC

```
*BEGIN 15 09 03100017 0015 00
840908 1015  0120 0202 0209 0130 0136 0205 0232 0160 0137 0199 0195 0128
840908 1145  0156 0212 0208 0154 0171 0246 0246 0176 0173 0230 0255 0179
840908 1315  0196 0252 0244 0173 0172 0245 0232 0184 0203 0244 0263 0184
840908 1445  0257 0302 0277 0186 0271 0294 0289 0202 0266 0307 0276 0210
840908 1615  0411 0407 0354 0242 0490 0434 0383 0265 0557 0491 0433 0303
840908 1745  0546 0485 0404 0327 0531 0456 0394 0297 0454 0441 0381 0269
840908 1915  0210 0253 0236 0155 0195 0213 0213 0145 0168 0221 0178 0135
840908 2045  0079 0127 0123 0084 0096 0121 0137 0073 0083 0127 0135 0082
840908 2215  0064 0117 0115 0051 0097 0100 0067 0100 0132 0105 0068 0107
840908 2345  0081 0115 0095 0057 0061 0087 0097 0045 0066 0105 0090 0040
840909 0115  0045 0083 0062 0045 0027 0072 0078 0051 0034 0057 0069 0034
840909 0245  0022 0051 0051 0028 0015 0047 0039 0025 0012 0028 0039 0011
840909 0415  0009 0020 0017 0013 0007 0014 0020 0019 0001 0019 0017 0014
840909 0545  0007 0022 0029 0007 0010 0026 0020 0015 0006 0032 0031 0012
840909 0715  0017 0041 0028 0032 0016 0049 0058 0037 0011 0037 0040 0028
840909 0845  0011 0037 0061 0040 0017 0063 0059 0043 0023 0071 0086 0059
840909 1015  0047 0089 0083 0059 0042 0084 0092 0068 0050 0093 0102 0069
840909 1145  0061 0112 0113 0082 0067 0129 0112 0086 0068 0119 0123 0074
840909 1315  0095 0162 0146 0100 0095 0181 0156 0107 0105 0178 0159 0110
840909 1445  0105 0158 0154 0115 0112 0176 0143 0103 0126 0173 0173 0110
840909 1615  0107 0164 0146 0104 0175 0158 0103 0116 0174 0135 0098 0122
840909 1745  0121 0182 0160 0114 0110 0167 0166 0102 0123 0189 0180 0122
840909 1915  0156 0215 0182 0127
*END   15 09 03100017 0015 00
```

Fig 1. Sample of Raw Data File

or data can be transferred through the GR-0308 interface unit. If the GR-0308 interface unit is used, the RS-232 cable is connected from the COM1: port and to the OFF-LINE TERMINAL connector port of the interface unit (Figure 2). During data transfers from Retriever to PC, the CTS indicator on front of the interface unit will remain on, and the DATA indicator will be activated by the data transfer. Figure 3 delineates the hardware connections between the IBM-PC, GR-0308 interface unit and the Retriever.

II. Software Configuration

There are several software commumications parameters that must be set properly to ensure correct data transfer between the Retriever and the IBM-PC. The following sections document the software configuration procedure for both the Retriever and the IBM-PC. The Retriever supports the XON-XOFF handshaking protocol to ensure the data commumications with the external devices. The PC-TALK utility program is a public domain communications software designed for the IBM-PC and has been used to communicate with the Retriever and receive the raw data. (Freeware is the term used by the supplier to denote a program(s) that requests a one time donation from each new user. This is different from public domain software which requires no formal donation request.)

A. Interface Signals in the Retriever

The interface signals transmitted during the data transfer from the Retriever are an important factor of the data transfer activity. These signals control the line feed and carriage return as well as other signals that can be programmed to be transmitted at the end of each line and at the end of each file. To make the proper signal settings, the following steps should be used on the Retriever. See Figure 4 for the detailed diagram of the Retriever control panel.

1. Using the arrow keys, align the bar mode indicator next to the INTERFACE mode.

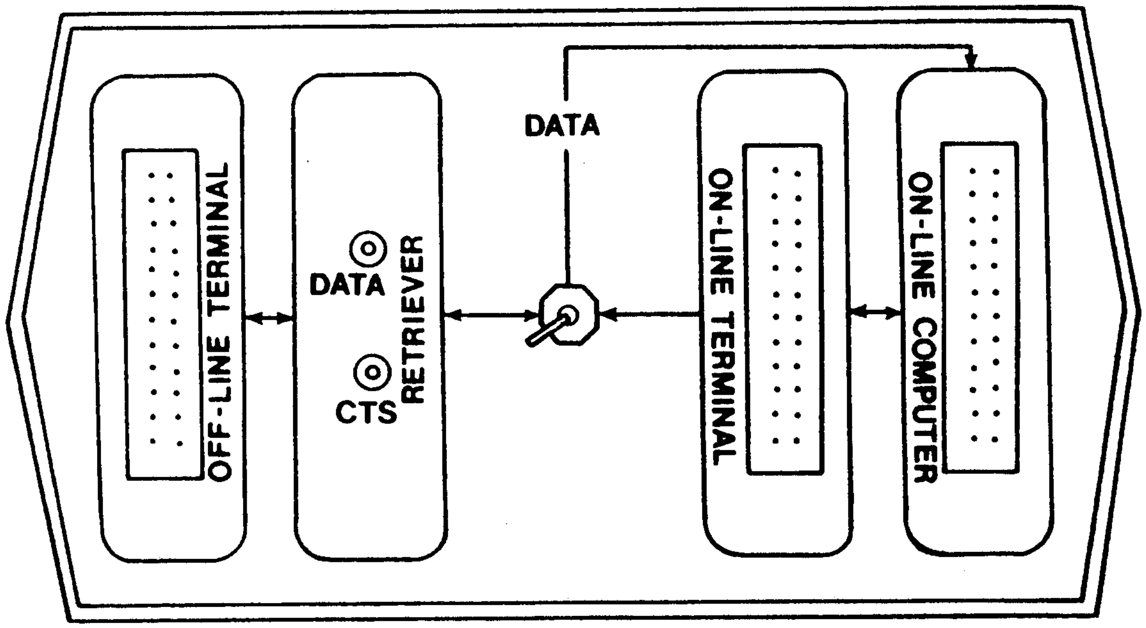


Fig.2 GR-0308 Interface/Charger Unit

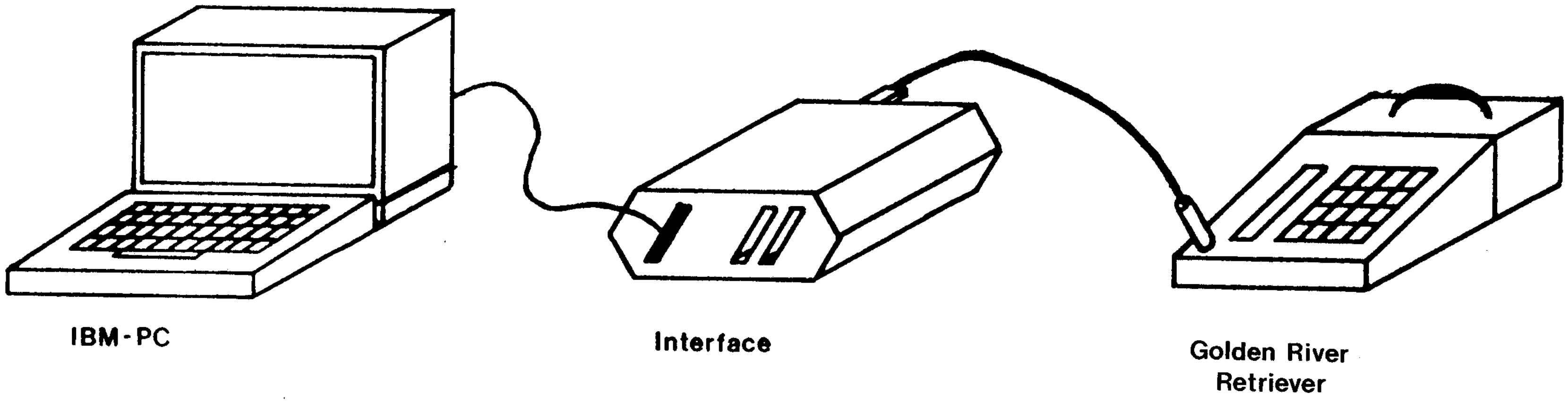


Fig. 3 Hardware Hook-up Diagram

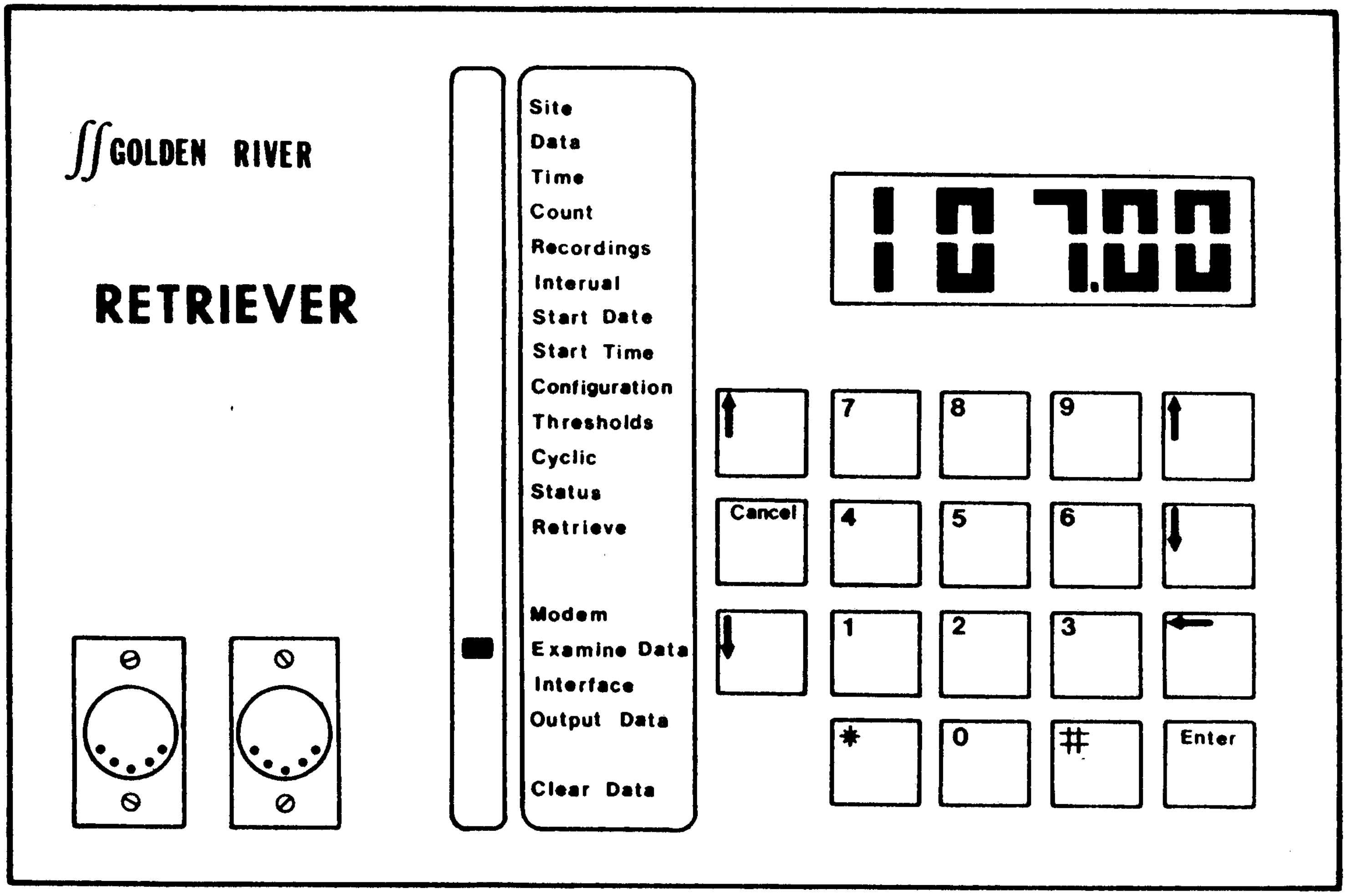


Fig.4 Retriever Control Panel

2. Push "2" to set the baud rate. The Retriever supports the baud rates between 110 to 19,200 bps. Press "#" and "*" simultaneously and the display will show "-----". Enter "01200" to set the baud rate at 1200.
3. Push "3" to set the end-of-line and end-of-file interface signals. Press "#" and "*" simultaneously and the display will show "---". Enter the following sequence of interface signals.

"013" + "ENTER" key	(Carriage Return)
"010" + "ENTER" key	(Line Feed)
"099" + "ENTER" key	
"099" + "ENTER" key	

4. Push "4" and set the interface 4 to "1" which enables the continuous data transmission.
5. Push "5" and set the interface 5 to "0" which disables the Status Line check.

These are the minimum interface signals required for proper data transfer and more details can be found in the Appendix A.

B. PC-TALK Communications Parameter Setting

PC-TALK is a general purpose communications program which supports a variety of data communication and storage capabilities. There are many parameters that may be set to be used with many different equipment and configurations. However, most of the parameters may be left at their initial default settings for the normal data transfer between the Retriever and the IBM-PC. Following are only the critical parameters that effect the data transfer activities and must be set to their proper values. These values may be set by pressing Alt-F from the main menu which will allow for parameter changes.

1. After the DOS prompt A>, enter "PC-TALK" to initiate the PC-TALK software.
2. After the start-up messages, press Alt-F to examine and modify the communications parameter settings.

3. The screen will display the current default settings and wait for any possible modifications.
4. Using the "RETURN" key to step through the parameters, make sure that the following parameter values are set accordingly.
    a. Baud Rate -- 1200
    b. Parity -- E
    c. Data Bits -- 7
    d. Stop Bits -- 1
    e. Pacing -- ''
    f. Comm. Port-- COM1:
5. Press ESC key to exit the parameter setting procedure.

III. Data Transfer Procedures

Once both the hardware connections and the software parameters are properly set, the actual data transfer and storage procedure can begin by following the steps outlined below. The Receive File function is used to receive the raw data from the Retriever and store it onto a disk file.

1. While in PC-TALK, press Alt-R to initiate Receive File function. The user will be prompted for the file name where the data will be stored. Enter the file name with the drive specification (e.g., "B:TEST.DAT" to store the data under a file named "TEST.DAT" on B: drive).
2. After the data file is created, PC-TALK enters a wait state to capture the incomming data through the specified communications port, COM1.
3. Prepare the Retriever for the OUTPUT DATA function by aligning the bar indicator next to the OUTPUT DATA label on the Retriever.
4. Press "#" and "*" simultaneously and the Retriever display will show "-----". Enter the file number to be output and press "ENTER". The Retriever will output the data within that file through the serial interface unit. PC-TALK will capture all the incomming data until the end of file has been reached and the Retriever ceases to send data.

5. Press Alt-R again to terminate the Receive File function which automatically closes the data file and save it on the diskette. The file must be properly closed using this second Alt-R command, otherwise it may be lost.
6. Repeat the above steps for each individual file stored in the Retriever.

This portion introduced only the minimal procedure necessary to store the data. Further information on the usage of the PC-TALK can be found in the PC-TALK III User's Guide (1).

DATA REDUCTION SOFTWARE

Until recently, all the data reduction has been manually carried out in the office after the data is printed out to the printer. This procedure was time consuming. Since the electronic data collection can be used in various configurations (up to the maximum of six channels), the manual data reduction process was further complicated by the number of channels used. The following section describes the software developed to automate the data reduction process and the simple operator's guide to execute the software.

I. Overview

The software for the data recudtion program ANALYZE may be divided into two major functional parts: HEADER and REPORT (see Figure 5 for the system hierarchy chart of ANALYZE). The HEADER program interacts with the user to obtain information about the data file and appends this information to the original data file. The REPORT program is used to reduce the data and produce the daily and weekly summary reports for each individual channel. The HEADER information obtained by the HEADER program is used to identify the data collection sites, locations, weather, etc. An example of the data file after the HEADER information has been appended is shown in Figure 6.

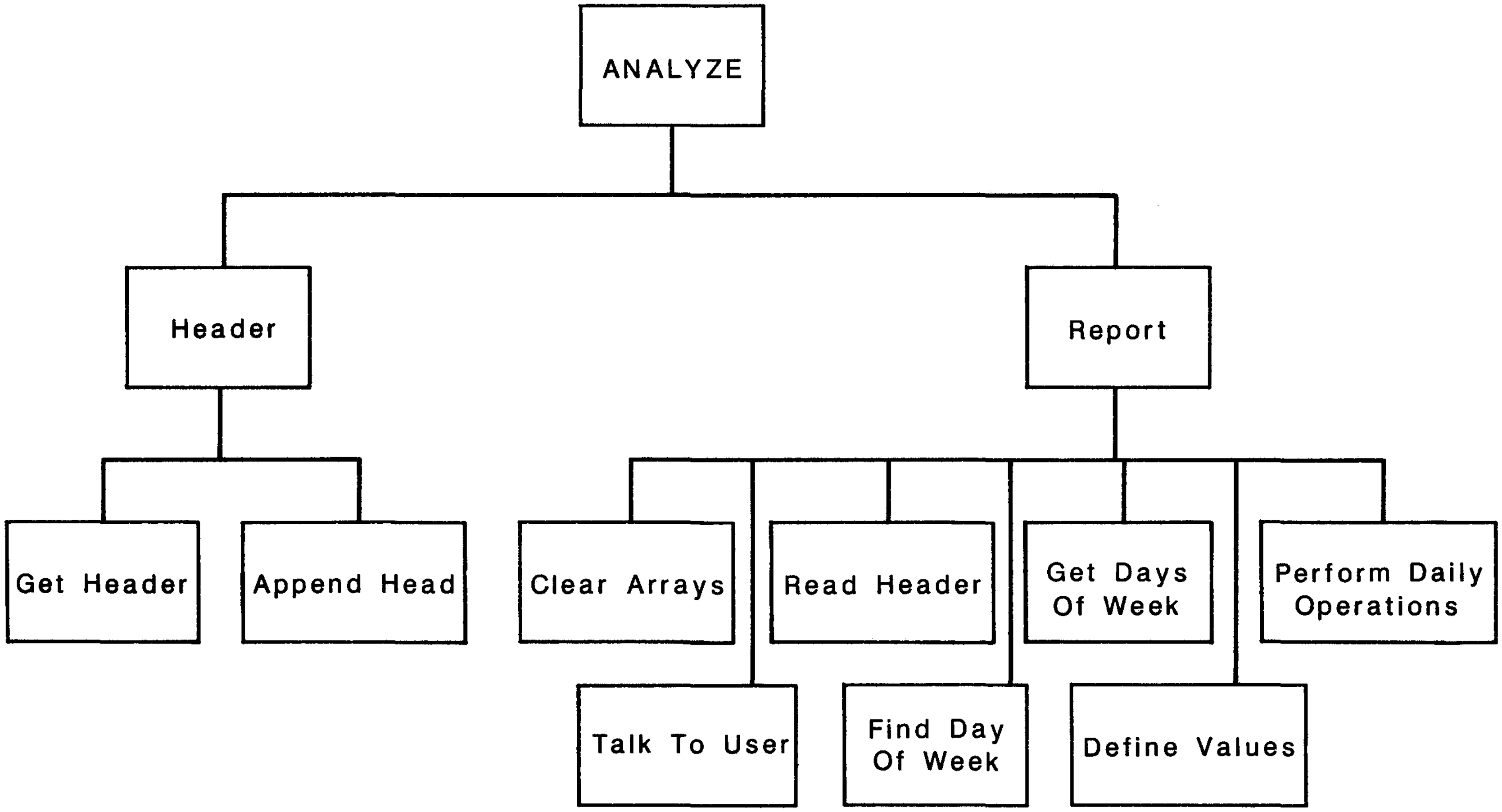


Fig. 5 Hierarchy Chart of System ANALYZE

```
  This is a sample data file.
COUNTER:                       John Doe
NUMBER OF WEATHERS:             2
WEATHER  1:                    Sunny
WEATHER  2:                    Cloudy
NUMBER OF LOCATIONS:           2
LOCATION  1:                   Loc #1
LOCATION  2:                   Loc #1
NUMBER OF CHANNELS:            2
CHANNEL  1 DIRECTION:          North
CHANNEL  2 DIRECTION:          South
*BEGIN 15 09 03100017 0015 00
840908 1015  0120 0202 0209 0130 0136 0205 0232 0160 0137 0199 0195 0128
840908 1145  0156 0212 0208 0154 0171 0246 0246 0176 0173 0230 0255 0179
840908 1315  0196 0252 0244 0173 0172 0245 0232 0184 0203 0244 0263 0184
840908 1445  0257 0302 0277 0186 0271 0294 0289 0202 0266 0307 0276 0210
840908 1615  0411 0407 0354 0242 0490 0434 0383 0265 0557 0491 0433 0303
840908 1745  0546 0485 0404 0327 0531 0456 0394 0297 0454 0441 0381 0269
840908 1915  0210 0253 0236 0155 0195 0213 0213 0145 0168 0221 0178 0135
840908 2045  0079 0127 0123 0084 0096 0121 0137 0073 0083 0127 0135 0082
840908 2215  0064 0117 0115 0051 0097 0100 0067 0100 0132 0105 0068 0107
840908 2345  0081 0115 0095 0057 0061 0087 0097 0045 0066 0105 0090 0040
840909 0115  0045 0083 0062 0045 0027 0072 0078 0051 0034 0057 0069 0034
840909 0245  0022 0051 0051 0028 0015 0047 0039 0025 0012 0028 0039 0011
840909 0415  0009 0020 0017 0013 0007 0014 0020 0019 0001 0019 0017 0014
840909 0545  0007 0022 0029 0007 0010 0026 0020 0015 0006 0032 0031 0012
840909 0715  0017 0041 0028 0032 0016 0049 0058 0037 0011 0037 0040 0028
840909 0845  0011 0037 0061 0040 0017 0063 0059 0043 0023 0071 0086 0059
840909 1015  0047 0089 0083 0059 0042 0084 0092 0068 0050 0093 0102 0069
840909 1145  0061 0112 0113 0082 0067 0129 0112 0086 0068 0119 0123 0074
840909 1315  0095 0162 0146 0100 0095 0181 0156 0107 0105 0178 0159 0110
840909 1445  0105 0158 0154 0115 0112 0176 0143 0103 0126 0173 0173 0110
840909 1615  0107 0164 0146 0104 0175 0158 0103 0116 0174 0135 0098 0122
840909 1745  0121 0182 0160 0114 0110 0167 0166 0102 0123 0189 0180 0122
840909 1915  0156 0215 0182 0127
*END   15 09 03100017 0015 00
```

Fig. 6 Sample of data file with HEADER information

For the purpose of the report generation, the main program REPORT requires information in addition to the data file initially created by the Retriever. The items needed are the location description, individual channel direction, weather conditions and any comment that may help identify the data file at a later date. The <u>**HEADER**</u> (Fig. 7) puts this header information at the top of a data file. By adding the header information, the HEADER information becomes a permanent part of the data file and the user does not have to key in such information every time a data analysis is performed on that data file. If the HEADER information has to be modified, the user can execute the HEADER program again on the same data file which will replace the old HEADER and append the new information at the top. It has two subroutines, GetHeader and AppendHead. GetHeader creates a temporary file of header information interactively with the user. The contents of the header information includes the counter's description, locations (up to a maximum of 6 locations) where the data was collected, direction of traffic flow at each location, and the weather of the days of data collection. AppendHead puts the header information in front of the temporary file. Then the temporary file is copied back to the original data file. Since all this procedure is done through batch processing, it is transparent to the user. The source code listings of the **HEADER** program is contained in Appendix B.

<u>REPORT</u> (Fig. 8) is used to generate daily and weekly reports. Its eight modules and the descriptions of their functions are:

1. **ClearArrays** initializes the arrays Nwtot, wFlag, dFlag, and days.
2. **TalkToUser** interacts with the user to obtain the data file name, and output options. Output options are 1) selecting the items such as input data, daily report on channel sum, daily report, or weekly report, 2) displaying or making a hard copy of the selected items. The peak hours are AM peak hour and PM peak hour. They are defined by default (06:30 - 09:30 for AM peak, 15:30 - 18:30 for PM peak), but the user has option to change them.

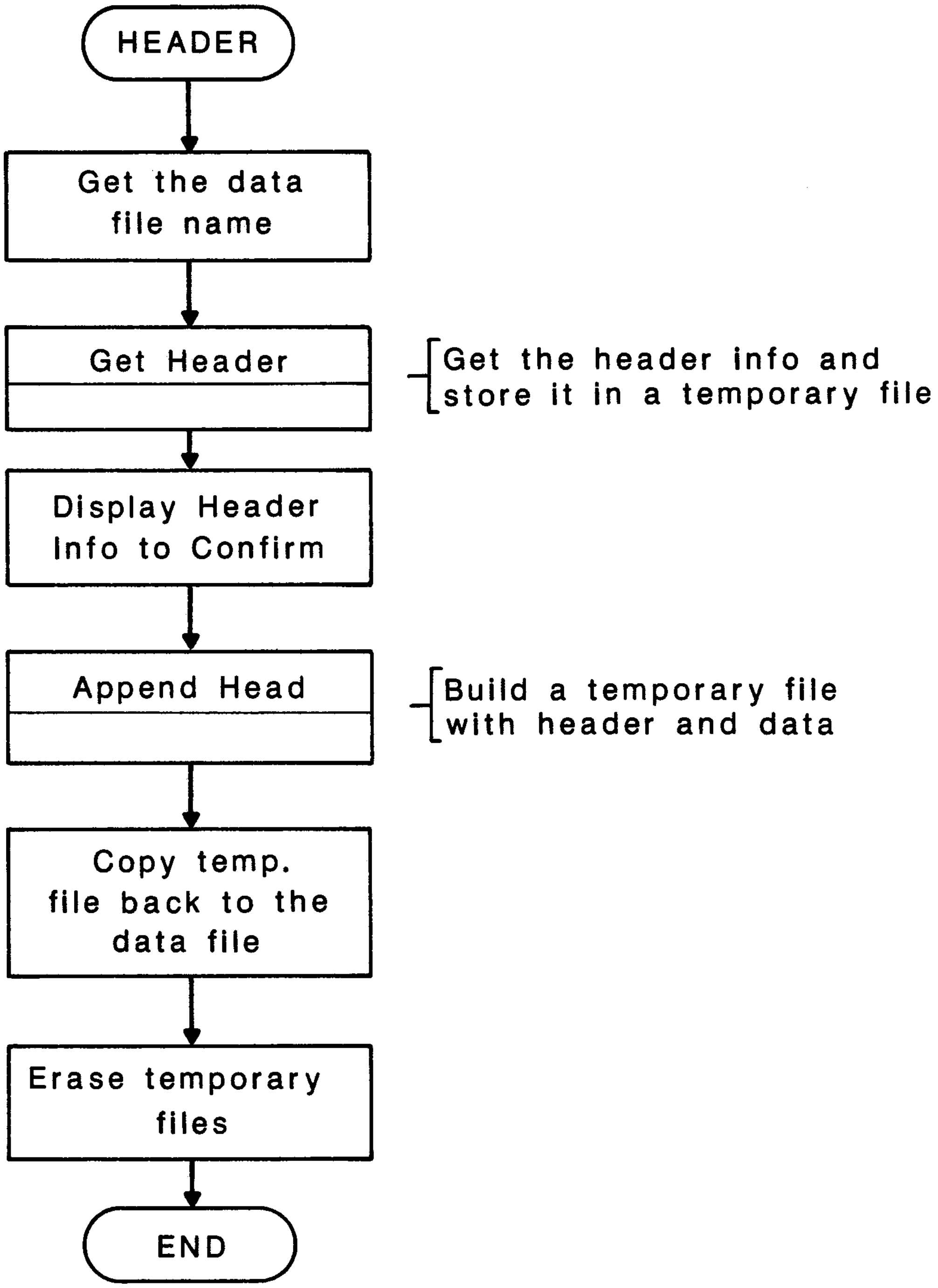
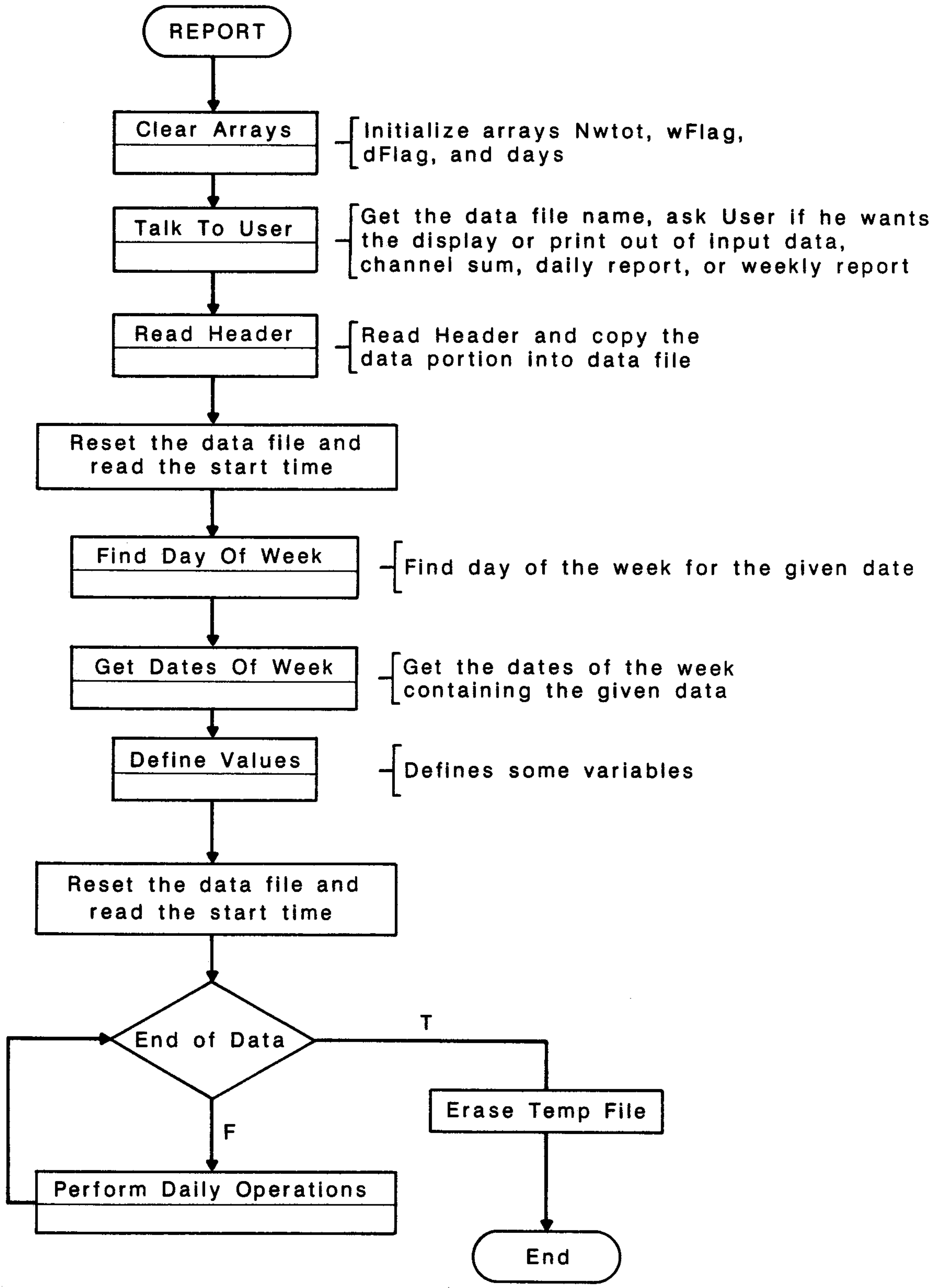

HEADER
Get the data file name
Get Header
Get the header info and store it in a temporary file
Display Header Info to Confirm
Append Head
Build a temporary file with header and data
Copy temp. file back to the data file
Erase temporary files
END

Fig. 7 Flowchart of HEADER



REPORT
Clear Arrays
Initialize arrays Nwtot, wFlag, dFlag, and days
Talk To User
Get the data file name, ask User if he wants the display or print out of input data, channel sum, daily report, or weekly report
Read Header
Read Header and copy the data portion into data file
Reset the data file and read the start time
Find Day Of Week
Find day of the week for the given date
Get Dates Of Week
Get the dates of the week containing the given data
Define Values
Defines some variables
Reset the data file and read the start time
End of Data
T
F
Erase Temp File
Perform Daily Operations
End

Fig. 8 Flowchart of REPORT

3. **ReadHeader** reads the data file and outputs data if requested by the operator. The data portion is copied into a temporary data file for a later use. The data file consists of header information and data. The first and last records are identical except the first field is '*BEGIN' for former and '*END' for latter. Other fields of this status record represent the number of channels, the data file number, the site number, the count interval time in minutes, and the machine status. A data record starts with the date and time of the first count. The date is recorded in 6 digits YYMMDD format. The time is recorded in 4 digits in HHMM format. One data record contains 12 traffic volume counts from all the channels in alternating manner. For instance, each column represents a consecutive reading for single channel, while every other column represents the consecutive reading for the double channel configuration. The header information is manually added to the file through the program HEADER. The items in that portion are well described in the following section, 'II. Directions for running HEADER.'
4. **FindDayOfWeek** finds the day of week of a given date. In this program the date is the first date of the data. It is necessary to know the day to generate a weekly report.
5. **GetDatesOfWeek** determines the dates of the current week for which a weekly report is generated. It takes care of the situation that the end of a month or year has occurred in the middle of the week.
6. **DefineValues** defines the working variables. First, the variable d3 is defined as the number of channels. The nch value is same as d3 unless the channel sum report is requested. The weathers are defined for the first week.

   The count is calculated to find the location of the first count in a daily report so the first day's counts are marked with '*' in front of them in a daily report. To do this, the corresponding elements (to the first day) of the array dFlag are assigned the value '*'.

   Some portions of the wFlag that reflects the hour slots of the time before the start of data collection started is assigned a value '-' which means the data is not available.

Since the peak hour definition is variable, the positions of the starting and ending counts of the two peak hours are calculated in the same manner as the starting count of a day was calculated.

7. **Perform_Daily_Operations** (Fig. 9) performs the day by day operations, generating daily and weekly reports, until the end of data is reached. Read_Data (Fig. 10) is used to read a 24-hour data. With new 24-hour data, the running subtotal and the hourly totals are computed.

   If the end of data is reached, it checks to see if the last day's data is more than three hours long in order to generate a daily report. Generate_Daily_Report routine will be called either when the end of data has not been reached or when the end of data has been reached and the data is longer than three hours.

   For each day's data, daily reports are generated by calling the procedure Generate_Daily_Report (Fig. 11). Generate_Daily_Report calls Peak_vol three times for AM peak hour, PM peak hour, and daily peak hour (00:00 - 24:00) respectively. The AM peak hour (06:30 - 09:30), PM peak hour (15:30 - 18:30) can be altered by the user at the beginning of the program. Peak_vol calculates the peak highest hour volume, peak hour start time, and peak volume during the given period of time.

   The peak highest hour volume is defined as the highest hourly volume count (start from any discrete point of time determined by the interval between counts and last one hour in the given period of time) among the hourly volume counts in the same period of time such as AM peak hour, PM peak hour or daily peak hour. The start time of the peak highest hour becomes the peak hour start time. Now it is ready to produce a daily report.

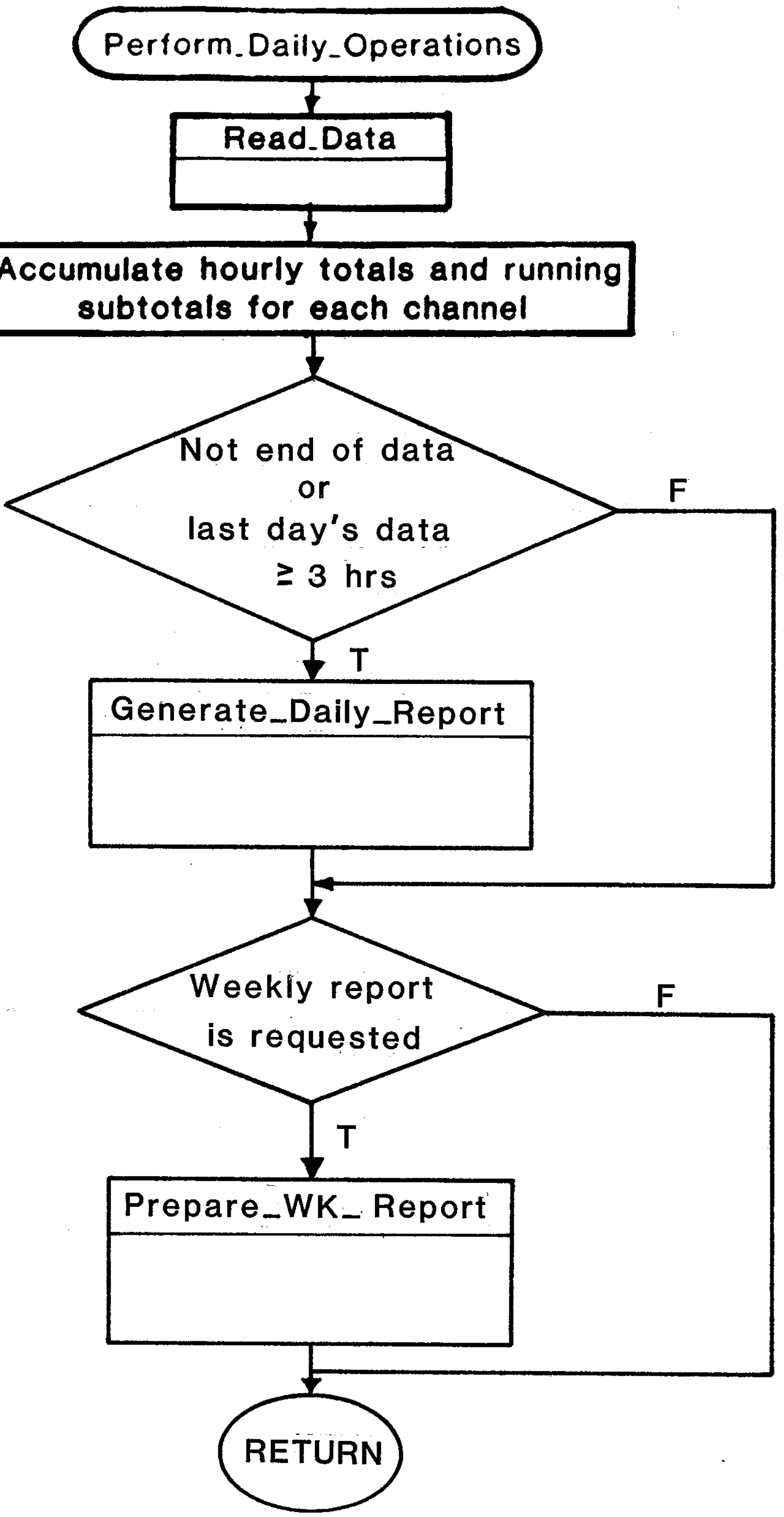

Perform_Daily_Operations
Read_Data
Accumulate hourly totals and running subtotals for each channel
Not end of data or last day's data ≥ 3 hrs
F
T
Generate_Daily_Report
Weekly report is requested
F
T
Prepare_WK_ Report
RETURN

Fig.9 Flowchart of Perform_Daily_Operations

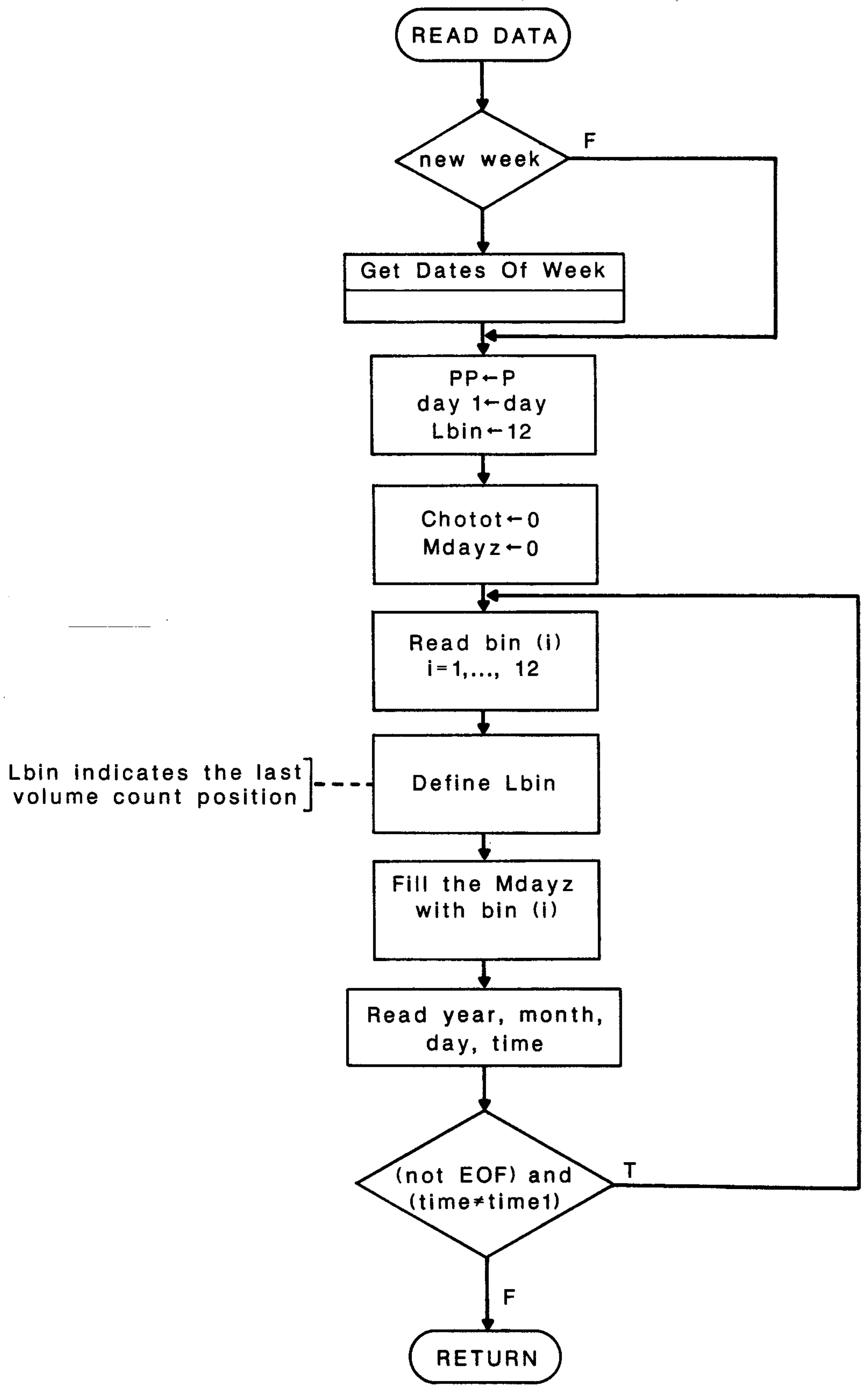

READ DATA
new week
F
Get Dates Of Week
PP←P
day 1←day
Lbin←12
Chotot←0
Mdayz←0
Read bin (i)
i=1,..., 12
Lbin indicates the last volume count position
Define Lbin
Fill the Mdayz with bin (i)
Read year, month, day, time
(not EOF) and (time≠time1)
T
F
RETURN

Fig. 10 Flowchart of READ DATA

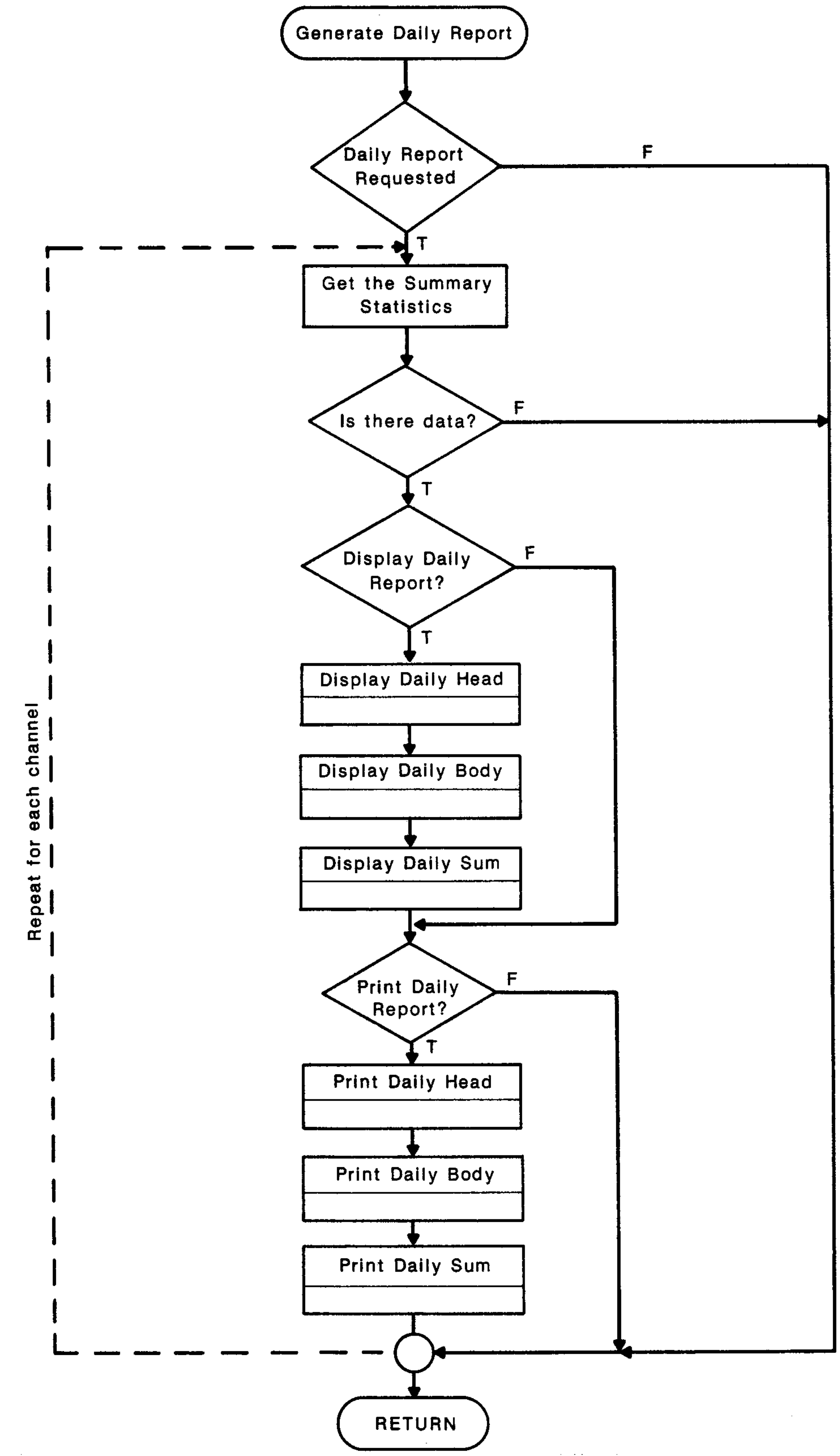


Fig. 11 Flowchart of GENERATE DAILY REPORT

If the total count of the day of the channel is not zero, then the output option is checked for daily report output. If the display was requested, the Display_Daily_Head, Display_Daily_Body, and Display_Daily_Sum is executed to produce daily report head, body, summary accordingly in sequence. Daily report may be printed on the paper (if requested) by calling Print_Daily_Head, Print_Daily_Body, Print_Daily_Sum.

After 24-hour data have been read and if weekly report generation was requested, the preparation of the weekly reports (Prepare_Wk_Report) is performed by the Build_Weekly__Table. It fills the hourly count slots of the 24-hour 7-day table. At this point, the end of data should be checked. If it is the end of data and the daily report spans over two days and the second day is Monday, weekly report is generated and the dates of the new week are obtained. No matter what the day of the second day is, and if the daily report contains two partial days, the remaining hours of the second day should be marked as data unavailable. After the wFlag table has been properly marked, the weekly report is generated.

If it is not the end of data but the Sunday's daily report has been generated, then the weekly report is generated. The Generate_Weekly_Report (Fig. 12) calls the Calc_Wk_Avg_Vol to calculate the 5-day and 7-day average volume counts that appear on the rightmost 2 columns of a weekly report. Then the Extract_Daily_Stats are called to compute the daily statistics to be shown at the bottom of the weekly report. After that, the Display_Wk_Head and Display_Wk_Body with Display_Sum_line for displaying the weekly report head and body or the Print_Wk_Head and Print_Wk_Body with Print_Sum_line for printing weekly report head and body may be called if they were requested. The last thing to do before leaving the Generate_Weekly_Report is to reset the arrays Nwtot and wFlag and get the next week's weather for the preparation of processing the next week's data.

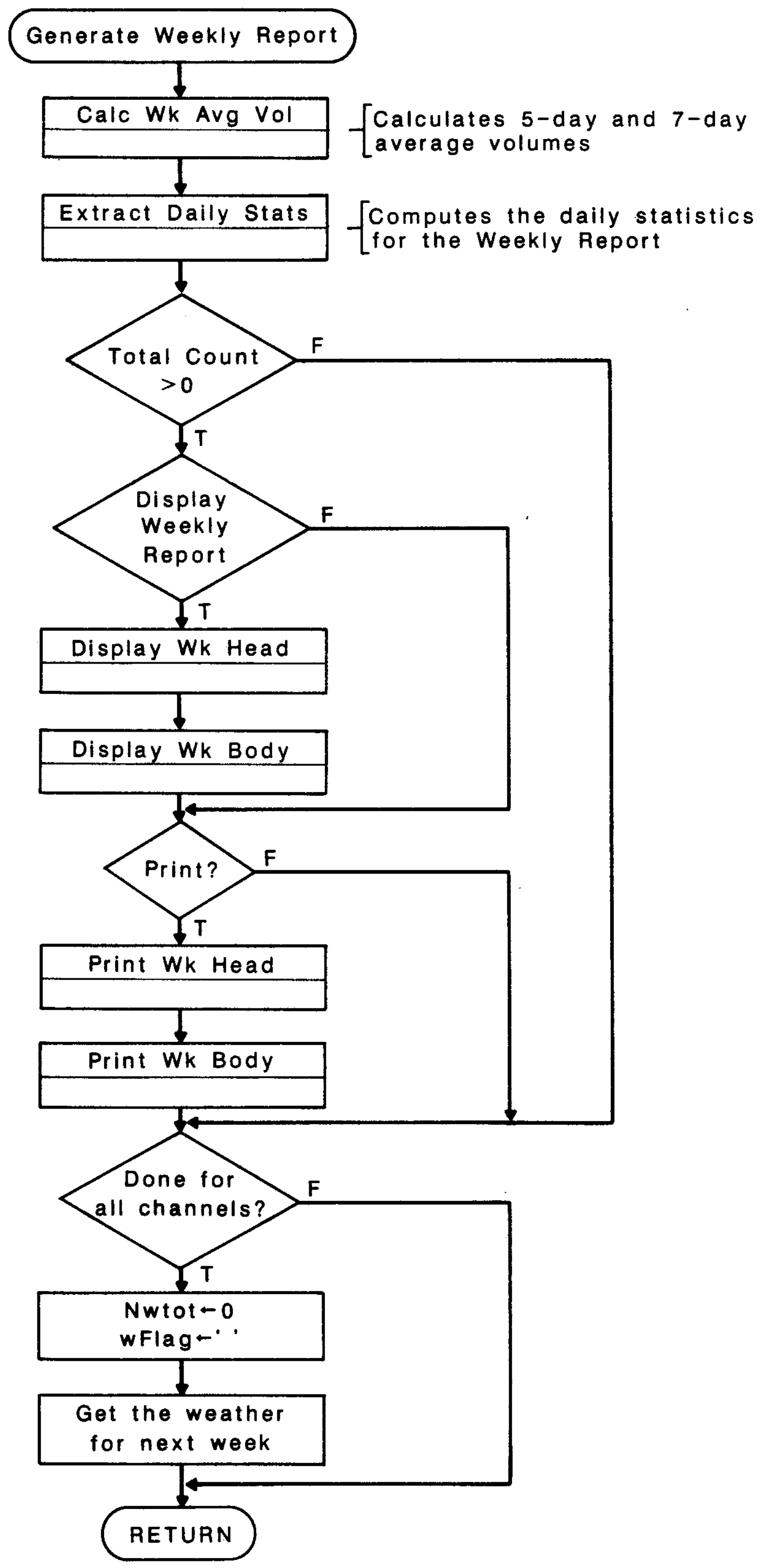


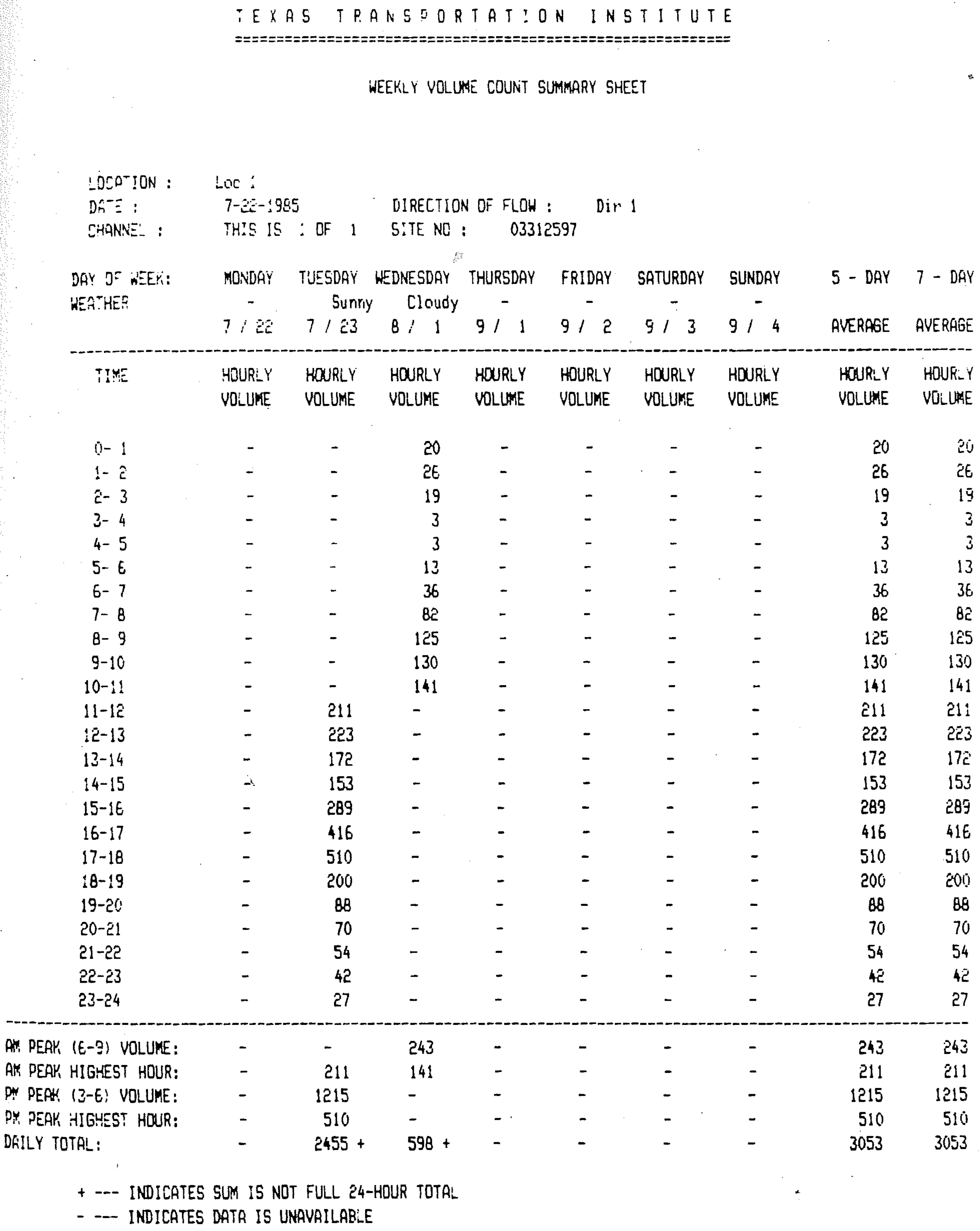
Fig. 12 Flowchart of GENERATE WEEKLY REPORT

Figure 13 describes the DAILY Report generated by the program. The raw data file shown in Figure 1 was used to generate this report. Figure 14 shows the WEEKLY Report generated using the same data file. The original Turbo PASCAL source code of the software is listed in the Appendix C.

Itek

```
                         T E X A S   T R A N S P O R T A T I O N   I N S T I T U T E
                         =========================================================
                                               VOLUME COUNT

                         985      SITE NO:   03312597             COUNTER:   John Doe
                                               DIRECTION OF FLOW:   Dir 1

                         S IS 1 OF 1

                         ------------------------------------------------------------------------------------------
                    HOURLY                          HOURLY                            HOURLY                         HOURLY
    TIME     VOLUME VOLUME        TIME      VOLUME  VOLUME       TIME       VOLUME    VOLUME       TIME      VOLUME  VOLUME
=========================================================================================================================
  0.00- 0.15    6                6.00- 6.15    5               *12.00-12.15    64                *18.00-18.15    71
  0.15- 0.30    7                6.15- 6.30    7               *12.15-12.30    50                *18.15-18.30    45
  0.30- 0.45    5                6.30- 6.45   13               *12.30-12.45    57                *18.30-18.45    43
  0.45- 1.00    2      20        6.45- 7.00   11      36       *12.45-13.00    52      223       *18.45-19.00    41     200
                     (  20)                         (  120)                          ( 1032)                          ( 2772)
-------------------------------------------------------------------------------------------------------------------------
  1.00- 1.15    9                7.00- 7.15   21               *13.00-13.15    36                *19.00-19.15    32
  1.15- 1.30    7                7.15- 7.30   14               *13.15-13.30    55                *19.15-19.30    21
  1.30- 1.45    3                7.30- 7.45   19               *13.30-13.45    36                *19.30-19.45    19
  1.45- 2.00    7      26        7.45- 8.00   28      82       *13.45-14.00    45      172       *19.45-20.00    16      88
                     (  46)                         (  202)                          ( 1204)                          ( 2860)
-------------------------------------------------------------------------------------------------------------------------
  2.00- 2.15   12                8.00- 8.15   22               *14.00-14.15    35                *20.00-20.15    24
  2.15- 2.30    4                8.15- 8.30   26               *14.15-14.30    37                *20.15-20.30    25
  2.30- 2.45    2                8.30- 8.45   46               *14.30-14.45    49                *20.30-20.45     9
- 2.45- 3.00    1      19        8.45- 9.00   31     125       *14.45-15.00    32      153       *20.45-21.00    12      70
                     (  65)                         (  327)                          ( 1357)                          ( 2930)
-------------------------------------------------------------------------------------------------------------------------
  3.00- 3.15    1                9.00- 9.15   37               *15.00-15.15    64                *21.00-21.15    10
  3.15- 3.30    0                9.15- 9.30   28               *15.15-15.30    57                *21.15-21.30    18
  3.30- 3.45    0                9.30- 9.45   31               *15.30-15.45    61                *21.30-21.45    16
  3.45- 4.00    2       3        9.45-10.00   34     130       *15.45-16.00   107      289       *21.45-22.00    10      54
                     (  68)                         (  457)                          ( 1646)                          ( 2984)
-------------------------------------------------------------------------------------------------------------------------
  4.00- 4.15    0               10.00-10.15   31               *16.00-16.15    83                *22.00-22.15     7
  4.15- 4.30    1               10.15-10.30   32               *16.15-16.30    84                *22.15-22.30    13
  4.30- 4.45    1               10.30-10.45   36               *16.30-16.45   115                *22.30-22.45     8
  4.45- 5.00    1       3       10.45-11.00   42     141       *16.45-17.00   134      416       *22.45-23.00    14      42
                     (  71)                         (  598)                          ( 2062)                          ( 3026)
-------------------------------------------------------------------------------------------------------------------------
  5.00- 5.15    6              *11.00-11.15   43               *17.00-17.15   140                *23.00-23.15     9
  5.15- 5.30    0              *11.15-11.30   45               *17.15-17.30   154                *23.15-23.30     3
  5.30- 5.45    4              *11.30-11.45   65               *17.30-17.45   143                *23.30-23.45     8
  5.45- 6.00    3      13      *11.45-12.00   58     211       *17.45-18.00    73      510       *23.45-24.00     7      27
                     (  84)                         (  809)                          ( 2572)                          ( 3053)
-------------------------------------------------------------------------------------------------------------------------

     AM PEAK (6:30 - 9:30) VOLUME:         296      PM PEAK (3:30 - 6:30) VOLUME:       1210        DAILY
     AM PEAK HIGHEST HOUR VOLUME :         142      PM PEAK HIGHEST HOUR VOLUME :        571        TOTAL:      3053
     (NEAREST 15 MINUTE COUNT)                      (NEAREST 15 MINUTE COUNT)                                ----------
     AM PEAK HOUR START :                 8:30      PM PEAK HOUR START :               16:45

     DAILY PEAK HOUR START :             16:45      DAILY PEAK VOLUME :                  571

    ( ) -- INDICATES CUMULATIVE HOURLY TOTAL        * -- INDICATES THE START DAY
```

Fig. 13 Sample of DAILY REPORT



TEXAS TRANSPORTATION INSTITUTE

WEEKLY VOLUME COUNT SUMMARY SHEET

LOCATION :	Loc 1		
DATE :	7-22-1985	DIRECTION OF FLOW :	Dir 1
CHANNEL :	THIS IS 1 OF 1	SITE NO :	03312597

DAY OF WEEK:	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY	SUNDAY	5 - DAY	7 - DAY
WEATHER	-	Sunny	Cloudy	-	-	-	-		
	7 / 22	7 / 23	8 / 1	9 / 1	9 / 2	9 / 3	9 / 4	AVERAGE	AVERAGE
TIME	HOURLY VOLUME	HOURLY VOLUME	HOURLY VOLUME	HOURLY VOLUME	HOURLY VOLUME	HOURLY VOLUME	HOURLY VOLUME	HOURLY VOLUME	HOURLY VOLUME
0- 1	-	-	20	-	-	-	-	20	20
1- 2	-	-	26	-	-	-	-	26	26
2- 3	-	-	19	-	-	-	-	19	19
3- 4	-	-	3	-	-	-	-	3	3
4- 5	-	-	3	-	-	-	-	3	3
5- 6	-	-	13	-	-	-	-	13	13
6- 7	-	-	36	-	-	-	-	36	36
7- 8	-	-	82	-	-	-	-	82	82
8- 9	-	-	125	-	-	-	-	125	125
9-10	-	-	130	-	-	-	-	130	130
10-11	-	-	141	-	-	-	-	141	141
11-12	-	211	-	-	-	-	-	211	211
12-13	-	223	-	-	-	-	-	223	223
13-14	-	172	-	-	-	-	-	172	172
14-15	-	153	-	-	-	-	-	153	153
15-16	-	289	-	-	-	-	-	289	289
16-17	-	416	-	-	-	-	-	416	416
17-18	-	510	-	-	-	-	-	510	510
18-19	-	200	-	-	-	-	-	200	200
19-20	-	88	-	-	-	-	-	88	88
20-21	-	70	-	-	-	-	-	70	70
21-22	-	54	-	-	-	-	-	54	54
22-23	-	42	-	-	-	-	-	42	42
23-24	-	27	-	-	-	-	-	27	27
AM PEAK (6-9) VOLUME:	-	-	243	-	-	-	-	243	243
AM PEAK HIGHEST HOUR:	-	211	141	-	-	-	-	211	211
PM PEAK (3-6) VOLUME:	-	1215	-	-	-	-	-	1215	1215
PM PEAK HIGHEST HOUR:	-	510	-	-	-	-	-	510	510
DAILY TOTAL:	-	2455 +	598 +	-	-	-	-	3053	3053

+ --- INDICATES SUM IS NOT FULL 24-HOUR TOTAL
- --- INDICATES DATA IS UNAVAILABLE

Fig. 14 Sample of WEEKLY REPORT

II. Directions for running HEADER

The user is assumed to have hands-on experience with a PC operation.

1. Insert the system diskette in drive A, data diskette in B.
2. Under the A> prompt, type HEADER then return.
3. System will ask you for the following information. The questions and possible responses are :

   Enter the data file name : Type B: followed by the data file name.

   Comments? Put any comments you want to have on the data file up to maximum of 70 characters.

   Counter? Describe the counter in 20 characters.

   Location 1? Location of the data collection can be described in 30 characters.

   Location 2? Up to six locations will be asked. If you have no more location to describe, enter a "-" to stop the question about location.

   Direction 1? The direction of the traffic flow at the location also can be described in 30 characters. The number of directions to be answered is equal to the number of locations.

   Weather of the day 1? Description of the firdt day's weather can be answered in 9 characters (DRY, WET, CLOUDY, etc.).

   Weather of the day 2? Up to 30 days of weather will be asked unless you stop them by entering a "-".
4. In a few seconds, the screen is cleared and the new header information you have just entered will be displayed with a check point notice. At this point, you can decide whether to keep that information with the data file by hitting the RETURN key or cancel it by pressing the CTRL key with Break key if you do not like the header information. If you canceled it, then the system will go back to A> which is at the beginning of step 2.
5. If you hit RETURN in step 4, the header routine will be completed in a short period of time. Then the system will return to the DOS with A>.

III. Directions for running REPORT

1. Insert the system diskette to drive A and the data diskette to drive B.
2. Under A>, type REPORT.
3. Upon the prompt **Enter the data file name.** On the top of the screen, type B: followed by the data file name.
4. The program will ask for the following running options. If you wanted to have a option to be performed, type either 'y' or 'Y'. If you do not, hit any key other than those 'y' and 'Y'. The lists of the options and the meanings are followed.

   Want to see the input data? (y/n) Asking if you want to see the contents of the data file on the screen before the operation starts.

   Print out, too? (y/n) Asking if you want to have the contents of the data file to be printed.

   Want to see the channel sum? (y/n) Asking if you want to see the daily report on the overall channel on the screen.

   Print out, too? (y/n) Asking if you want to have the daily report on the overall channel to be printed.

   Want to see the daily report? (y/n) Asking if you want to see the daily reports on each channel on the screen.

   Print out, too? (y/n) Asking if you want to have the daily reports on each channel to be printed.

   Want to see the weekly report? (y/n) Asking if you want to see the weekly report on the screen.

   Print out, too? (y/n) Asking if you want to have the weekly report to be printed.

   Want to change the peak hours? (y/n) Asking if you want to define the peak hours other than the default values. (The defaut of the AM peak hours are 06:30 - 09:30, and the PM peak are 15:30 - 18:30). If the answer was yes, then the following prompts will be generated to get the new definition.

   Enter the AM peak start time (hhmm) Asking the start time of the AM peak hours in four digits of hour followed by minute.

   Enter the AM peak end time (hhmm) Asking the ending time of the AM peak hours.

Enter the PM peak start time (hhmm) Asking the start time of the PM peak hours.
Enter the PM peak end time (hhmm) Asking the ending time of the PM peak hours.

Now, the system will start running.

NOTE: Before you run the report, make sure the printer is set in compressed mode to print 132 characters on a line.

REFERENCES

1. Fluegelman, Andrew, PC-TALK, Tiburon, CA 94920
2. MK3 TRAFFIC COUNTERS & CLASSIFIERS USER MANUAL - ISSUE 5, Golden River Corporation, 3930 Knowles Avenue, Kensington Maryland 20895.

Appendix A

The following pages are taken from the MK3 AUTOMATIC TRAFFIC COUNTING EQUIPMENT USER MANUAL(2). This section describes the procedures to set up interface signals. The user may find more ASCII codes from the IBM System/370 Reference card. Any ASCII character set can be used as EOF and EOL signals. The user simply enters the decimal equivalent of the ASCII character set during the interface signal setting procedure.

3.3 Interface

This mode is used to programme the RS-232 interface to be compatible with the computer or off-line terminal which is to be used with the Retriever.

There are 4 functions available to initialise the interface parameters and there is one function available to allow the user to test the interface. These five functions correspond to 5 code numbers. The code number is displayed on the left hand digit of the display. The 6 right hand digits are used as a parameter field. To change the current code number, press the appropriate key in the range 1-5. Pressing a key outside this range will result in Interface 1 being selected.

The parameters displayed for each code are:

Interface Code	Parameters Displayed	Format
1	Retriever Status; Battery Voltage	C S V.VV
2	Baud Rate	C BBBBB
3	No. of End of Line and End of File Characters	C LL FF
4	Code for Autostop option setting	C A
5	Code for Status Line option setting	C S

C = Interface Code Number
S = Status Code
V = Battery Voltage
B = Baud rate
L = End of Line characters
F = End of File characters
A = Autostop Code
S = Status Line Code

Following is a description of the 5 Interface codes.

3.3.1 Interface 1 - Display Retriever status and battery voltage;

This function has three characteristics - it displays the present Retriever Status and battery voltage, and it can be instructed to output a test message to a computer or off-line terminal via the RS-232 Interface.

Select Interface 1 by pressing key "1", and the display will show 3 numbers. The left hand digit shows the Interface code selected, in this case 1. The digit in the centre of the display indicates the status of the Retriever. If this digit is 0, the Retriever is O.K., otherwise a fault condition is indicated by a non-zero number. Status is reset when the data files are cleared.

The battery voltage is shown on the right hand side of the display. When the battery is fully charged this should be in excess of 6.30 volts with the charger disconnected. As the battery discharges, this voltage will reduce towards 5.70 volts over a period of 6 to 8 days. At 5.50 volts a cut-out circuit will switch the power to the Retriever off (see section 1.3.1).

WARNING: When the power to the Retriever is switched off, ALL Data will be lost. For this reason it is suggested that the GR 0308 Charger/Interface unit is connected to the Retriever whenever possible. (It should be noted that the

Interface parameters, as described in the following sections, will not be lost when power to the Retriever is switched off).

Once the Interface parameters 2-5 have been initialised, (see the following sections),the user can check the RS-232 interface is matched to the computer or off-line terminal being used.

With Interface 1 selected, and the terminal or computer connected, depress "*" and "#" simultaneously. The Retriever will output repeatedly a test message of numbers and characters. Any End of Line characters (as described in Section 3.3.3) will follow each line, and transmission will stop at the end of each line if the Autostop option has been enabled (see Section 3.3.4) When the "*" and "#" are released, the Retriever will complete output of the current test message and End of Line characters, and it will output the End of File characters (see Section 3.3.3) followed by the End of Line characters again. During this output the Retriever will respond to any status line control as described in Section 3.3.5.

3.3.2 Interface 2 - Baud Rate

Select Interface code number 2. The right hand digits show the current Retriever baud rate. To alter the baud rate press the "*" and "#" simultaneously, and when the display shows 5 dashes enter the required baud rate, (with leading zeros), followed by the 'enter' key. The Retriever will confirm the baud rate selected on the display.

The available baud rates are:-

19200   9600   4800   2400   1200   600   300   110

Entry of a baud rate other than shown above will result in the nearest available baud rate being selected.

The average transmission rate will be less than the baud rate, whilst the Retriever calculates data for output. This delay will be approximately 5-500 milliseconds at start of each line of output, proportional to the recording interval of the data file being output. In format 2, longer delays may occur (up to 15 seconds) whilst the output is being compiled.

3.3.3 Interface 3 - End of Line and End of File characters

The operator can specify up to 20 characters to be added to the end of each line of output in the Output Data mode, and up to 20 characters to be added at the end of each File in the Output Data mode.

Select Interface Code Number 3; the centre two digits show the present number of characters transmitted at the end of each line, and the right hand two digits show the present number of characters transmitted at the end of each file. To alter the current E.O.L. (End of Line) and E.O.F. (End of File) character strings, depress "*" and "#" simultaneously, and when:

Display shows	Load into Retriever
---	ASCII code for first EOL character + "ENTER"
---	ASCII code for second EOL character + "ENTER"
...... etc.	
---	ASCII code for last EOL character + "ENTER"
---	999 + "ENTER" (terminating code)
---	ASCII code for first EOF character + "ENTER"
---	ASCII code for second EOF character + "ENTER"
...... etc.	
---	ASCII code for last EOF character + "ENTER"
---	999 + "ENTER" (terminating code)

For example:

For the E.O.L. characters to be CR, LF and six nulls, and the E.O.F. characters to be four asterisks then:

Display shows	Press Keys	Description
1 0 6.40	3	Select Interface 3 function
3 x y	* & #	To change stored characters (x and y are previous values)
---	013 + ENTER	Load ASCII Carriage Return
---	010 + ENTER	Load ASCII Line Feed
---	000 + ENTER	Load ASCII Null
---	000 + ENTER	Load ASCII Null
---	000 + ENTER	Load ASCII Null
---	000 + ENTER	Load ASCII Null
---	000 + ENTER	Load ASCII Null
---	000 + ENTER	Load ASCII Null
---	999 + ENTER	Terminate EOL / Commence EOF
---	042 + ENTER	Load ASCII *
---	042 + ENTER	Load ASCII *
---	042 + ENTER	Load ASCII *
---	042 + ENTER	Load ASCII *
---	999 + ENTER	Terminate loading EOF chars
3 8 4	- Indicates 8 EOL and 4 EOF characters have been accepted.	

All ASCII codes in the range 000 to 127 are accepted, any code in excess of 127 will terminate that part of the loading. Attempted entry of more than 20 E.O.L. or 20 E.O.F. characters will be ignored, but the first 20 characters loaded will be stored. If no E.O.F. characters are required then enter the terminating code "999" twice after the last E.O.L. character code.

All numbers entered and displayed on the Retriever are in the normal decimal numbering system. If you wish to enter a character code whose value is known in binary, octal, or hexadecimal, conversion to a decimal number must be done before entry of decimal equivalent into the Retriever.

For example:

	Decimal	Binary	Octal	Hexadecimal
ASCII Line Feed	010	0001010	012	0A

ASCII Codes for Control Characters

Code	Name	Description
000	NULL	A null character usually to fill time
001	SQH	Start of Heading
002	STX	Start of Text
003	ETX	End of Text
004	EOT	End of Transmission
005	ENQ	Enquiry
006	ACK	Acknowledge
007	BELL	Bell
008	BS	Backspace
009	HT	Horizontal Tab
010	LF	Line Feed
011	VT	Vertical Tab
012	FF	Form Feed
013	CR	Carriage Return
016	DLE	Data Link Escape
017	DC1	Device Code 1
018	DC2	Device Code 2
019	DC3	Device Code 3
020	DC4	Device Code 4
027	ESC	Escape
127	DEL	Delete

Only commonly used control characters have been listed, but the user can input any valid ASCII codes required.

3.3.4 Interface 4 - Autostop Option Code

When enabled, this option halts all output from the Retriever to the computer or terminal after completion of each line of data output in the Output Data mode (including programmed end of line characters). The computer or terminal must then send a DC1 code to recommence transmission.

Select Interface code number 4. The right hand digit shows the current option as follows:

Autostop Option Code	Resulting Operation
0	Normal operation, no stop at end of line
1	Stops transmission at end of each line of output in Output Data mode (and Interface 1 test mode)

To alter the Autostop Option Code, depress "*" and "#" simultaneously, and when the display shows 1 dash, enter the appropriate one digit code.

3.3.5 Interface 5 - Status Line Input Option

Some terminals and/or communications devices present a "device ready" signal as an indication that serial data (such as output from the Retriever) can now be sent, or must now be terminated. On an RS232 device this will usually be on pin 20 of the 25 way connecting plug and normally has high or positive level to indicate "device ready" for receipt of data.

Status Line (Pin 20 of 25 way RS232 "D" connector)

Input Code	Result
0	Ignore Status Input
1	Initialize transmission of next character only whilst Status input is high
2	Initialize transmission of next character only whilst Status input is low

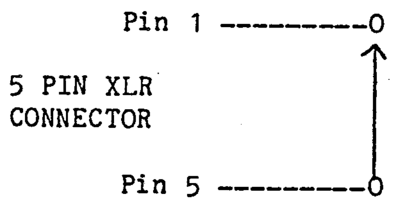
In this context Status Line input is defined as:

-15 to 0 volts	- Status Low
5 to +15 volts	- Status High

(0 to 5 volts undefined).

Select Interface Code Number "5". The right hand display will show the current value of Status Line Control.

To alter, press "*" and "#" simultaneously and when display shows one dash enter one digit number 0 through 2 as indicated above.
This facility could be used as follows with an external switch:



(Switch open equivalent to Low status input, switch closed equivalent to high status input)
Diagram shows switch closed

Note that if Status Line Input is left disconnected it assumes a low level, i.e. no transmission would occur with Status Line Control = 1, but continuous transmission with 0 or 2.

Appendix B

```
PROGRAM Header;

{~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~}
{                                                       }
{                  H E A D E R                          }
{                 ==============                        }
{                                                       }
{               Author: Duk-Jin Chang                   }
{               Date:   Aug. 16, 1985                   }
{                                                       }
{  This program is written in Turbo Pascal to build a   }
{  Header information file to append in front of the    }
{  data file.                                           }
{                                                       }
{~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~}

VAR
  f,              { = data file }
  t,              { = header file }
  x:              { = header + data file }
     TEXT;

  ch: CHAR;

  fname:          { = data file name }
         STRING[14];

  line:           { each line of data is read into 'line' }
        STRING[73];



  PROCEDURE GetHeader;
  { Builds a temporary header information file through  }
  { the communication with a user.                      }

  LABEL
    L1, L2, L3;

  TYPE
    str32 = STRING[32];

  VAR
    i, j, nch {= # of channels}, nwthr {= # of weathers}: INTEGER;
    counter: str32;
    dir, loc: ARRAY [1..7] of str32;
    weather: ARRAY [1..30] of str32;
    comment: STRING[70];

  BEGIN
    { Get the header informations }
    WRITE (' Comments?    ');
    READLN (comment);
    WRITE (' Counter?    ');
    READLN (counter);
```

```
    loc[7] := ' ';
    FOR nch := 1 TO 6 DO BEGIN
      WRITE (' Location', nch:2, '? ');
      READLN (loc[nch]);
      IF loc[nch] = '-' THEN BEGIN
        nch := nch - 1;
        GOTO L1
      END;
    END;

L1: dir[7] := 'Overall Channel';
    FOR i := 1 TO nch DO BEGIN
      WRITE (' Direction', i:2, '? ');
      READLN (dir[i])
    END;

L2: FOR nwthr := 1 TO 30 DO BEGIN
      WRITE (' Weather of the day', nwthr:2, '? ');
      READLN (weather[nwthr]);
      IF weather[nwthr] = '-' THEN BEGIN
        nwthr := nwthr - 1;
        GOTO L3
      END;
    END;

L3: FOR i := nwthr+1 TO 30 DO
      weather[i] := '    -     ';

    { Build a temporary file with the header information }
    ASSIGN (t, 'Head.dat');
    REWRITE (t);
    WRITELN (t, '  ', comment);
    WRITELN (t, 'COUNTER:                    ', counter);

    WRITELN (t, 'NUMBER OF WEATHERS:         ', nwthr:2);
    IF nwthr <> 0 THEN
      FOR i := 1 TO nwthr DO
        WRITELN (t, 'WEATHER ', i:2, ':                    ', weather[i]);

    WRITELN (t, 'NUMBER OF LOCATIONS:        ', nch);
    FOR i := 1 TO nch DO
      WRITELN (t, 'LOCATION ', i:2, ':                   ', loc[i]);

    WRITELN (t, 'NUMBER OF CHANNELS:         ', nch);
    FOR i := 1 TO nch DO
      WRITELN (t, 'CHANNEL ', i:2, ' DIRECTION:        ', dir[i]);

    CLOSE (t);

  END;
  PROCEDURE AppendHead;
  { Preffixes the header to the data file. }

  BEGIN
    ASSIGN (x, 'tmp.dat');
    RESET (f);
    RESET (t);
    REWRITE (x);

    WRITELN; WRITELN; WRITELN; WRITELN;
    WRITELN ('        .......... Wait for a few seconds ......');
    WHILE NOT EOF(t) DO BEGIN
      READLN (t, line);
      WRITELN (x, line);
      END;

    REPEAT
      READLN (f, ch, line)
    UNTIL ch = '*';

    WRITELN (x, ch, line);

    WHILE NOT EOF(f) DO
    BEGIN
      READLN (f, line);
      WRITELN (x, line);
    END;

    CLOSE (f);
    CLOSE (t);
    CLOSE (x);

  END;



BEGIN
  WRITE ('Enter the data file name :   ');
  READLN (fName);
  ASSIGN (f, fName);
  WRITELN; WRITELN; WRITELN;

  GetHeader;

  ClrScr;
  RESET (t);
  WHILE NOT EOF(t) DO BEGIN
    READLN (t, line);
    WRITELN (line);
    END;
  WRITELN; WRITELN; WRITELN;
  WRITELN ('*** Check point ***     Use ctrl-brk to reenter the data');
  WRITELN ('                                Or');
  WRITELN ( '                                Hit return to continue' );
  WRITELN; WRITELN; WRITELN;
  READ ( ch );
  ClrScr;

  AppendHead;

  REWRITE ( f );
  RESET ( x );
  WHILE NOT EOF( x ) DO BEGIN
    READLN ( x, line );
    WRITELN ( f, line );
    END;
  CLOSE ( f );

  ERASE ( x );
  ERASE ( t );

  WRITELN; WRITELN; WRITELN;
  WRITELN ( '                                           --- Header Completed ---' );

END.
```

Appendix C

```
PROGRAM Report;

{:~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~:
 :                                                                           :
 :                     R E P O R T                                           :
 :                     ===========                                           :
 :                                                                           :
 :                Author : Duk-Jin Chang                                     :
 :                Date : May 27, 1986                                        :
 :                Language : Turbo Pascal                                    :
 :                Operating System : MS DOS 2.10                             :
 :                                                                           :
 :    This program is designed to read a traffic-count data                  :
 :  and produce Daily and Weekly reports with analyzed figures.              :
 :                                                                           :
 :                                                                           :
 :      Variable Definitions                                                 :
 :                                                                           :
 :  am_high ... the largest hourly volume in the am peak hours.              :
 :  am_hour ... the beginning hour of the am_high.                           :
 :  am_min .... the beginning minute      "      .                           :
 :  am_peak ... the total volume of the am peak hours.                       :
 :  a_p_s ..... the starting time of the AM peak hour (default = 6:30).      :
 :  a_p_e ..... the ending      "          "          (default = 9:30).      :
 :  a_p_s_c ... the position of the AM peak start count in Mdays.            :
 :  a_p_e_c ...         "            "     end        "         .            :
 :  a_e ....... string conversion of the a_p_e.                              :
 :  a_s .......           "                a_p_s.                            :
 :  arday[i] .. the date of the ith day in a current week.                   :
 :  armon[i] .. the month of               "            .                    :
 :  aryr[i] ... the year of       "          "          .                    :
 :  astrik[i,j] .. marks ' +' if the ith day doesn't have 24-hour data.      :
 :  a_hi[i] ... the highest hourly volume of am in ith day of the            :
 :              weekly report.                                               :
 :  a_pk[i] ... the total volume count between 06:00 and 09:00 of the        :
 :              ith day of the weekly report.                                :
 :  begn ...... the first 6 characters read in from a data record.           :
 :  bin[i] .... the ith volume count of a data record.                       :
 :  blank[i,j] .. a dummy argument passed to Display_Sum_Line.               :
 :  ch ........ represents the channel number.                               :
 :  ChoTot[i,j] .. an hourly total of the ith channel, jth hour.             :
 :  ChSum ..... tells if the channel sum daily report is asked.              :
 :  comment ... the comments on the data file.                               :
 :  conf ...... the channel configuration number.                            :
 :  count ..... the location of the first data of the day in the             :
 :              array of 24 hour data slots.                                 :
 :  counter ... identification of the data counter.                          :
 :  date1 ..... the ending date & time of the first data count.              :
 :  day ....... the day of the data record created.                          :
 :  days[i] ... the number of days in the ith month.                         :
 :  day1 ...... a start date of a daily report.                              :
 :  day_total . a total volume count of a daily report.                      :
 :  delta ..... 100th of the interval between the counts.                    :
 :  dFlag[i,j] .. tells if the ith row, jth column count on the              :
 :                daily report is of the first date.                         :
```

```
:  dir[i] .... the direction of the traffic of the ith channel.         :
:  dOut ...... the flag for daily report display.                       :
:  dRpt ......                ''             print out.                 :
:  dummy ..... holds the unnecessary data read in.                      :
:  d_high .... the highest hourly volume of the day.                    :
:  d_hour .... the beginning hour of the d_high.                        :
:  d_min .....        ''      minute      ''      .                     :
:  d3 ........ the number of channels.                                  :
:  f ......... the original data file.                                  :
:  FileNo .... the data file number.                                    :
:  fName ..... the original data file name.                             :
:  galf ...... tells if the daily total is full 24 hours'.              :
:  id_tot[i] . the ith daily total of the week.                         :
:  iflag[i] .. is set to 1 if the ith day's data is not full.           :
:  intval .... the time between the counts.                             :
:  iw  ....... the number of days the weather has been assigned.        :
:  k3 ........ row count in the daily report body generation.           :
:  Lbin ...... the column number of the last count in the data record. :
:  loc[i] .... the ith location of the data collection.                 :
:  Mdays[i,j] .. the count of the ith channel, jth time slot.           :
:                One quarter of Mdays array reflects one column  of     :
:                a Daily report body.                                   :
:  month ..... the current month.                                       :
:  multpl .... the number of counts per hour.                           :
:  nch ....... the number of channels.                                  :
:  nwthr ..... the number of weather descriptions.                      :
:  Nwtot[i,j,k] .. the hourly total count of the ith channel, jth day, :
:                  kth hour.                                            :
:  p ......... Number of days from Monday to the daily report start.    :
:  pp ........               ''                ''             end.      :
:  pk_flag[i,j] .. tells if the data in the summary line of the weekly :
:                  report is not meaningful.                            :
:  pm_high ... the highest hourly volume count of pm.                   :
:  pm_hour ... the starting hour of the pm_high.                        :
:  pm_min ....        ''     minute      ''      .                      :
:  pm_peak ... the total count during the pm peak hour.                 :
:  p_ch_sum .. is set to 1 if daily reports on channel sum is asked to :
:              be printed.                                              :
:  p_day ..... is set to 1 if daily report is to be printed.            :
:  p_dt ......         ''        data is to be printed.                 :
:  p_hi ...... the hourly highest count during pm in weekly report.     :
:  p_pk ...... the total count during pm peak hour         ''       .   :
:  p_wk ...... is set to 1 if weekly report is to be printed.           :
:  p_p_s ..... pm peak hour start time (15:30).                         :
:  p_p_e .....      ''       ending time (18:30).                       :
:  p_p_s_c ... the position of the p_p_s in the Mdays.                  :
:  p_p_e_c ...          ''          p_p_e       ''     .                :
:  p_e ....... string conversion of the p_p_e.                          :
:  p_s .......              ''               p_p_s.                     :
:  run_sub[i,j] .. the running subtotal of ith channel at jth hour.     :
:  SiteNo .... identification number of the site of data collection.    :
:  status .... unused value in this program.                            :
:  t ......... temporary data file.                                     :
:  time ...... the collection time of the current data.                 :
:  time1 .....               ''               first data.               :
```

```
:  tot8[i,j] .. saves the next Monday's ith channel, jth hour count.  :
:  wFlag[i,j] .. tells the validity of the ith channel, jth hour data. :
:  WkRpt ..... is set to 1 if weekly report is to be generated.        :
:  wt[i] ..... the weather of the ith day from the start.              :
:  wthr[i] ...              "              of the current week.        :
:  year ...... current year.                                           :
:                                                                      :
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~}
```

```
TYPE
  real_ar_9 = ARRAY [1..9] of REAL;
  str_2 = STRING[2];
  str_3 = STRING[3];
  str_21 = ARRAY [1..21] of CHAR;
  str_2_ar_9 = ARRAY [1..9, 1..2] of CHAR;

VAR
  am_high, a_p_s, a_p_e, a_p_s_c, a_p_e_c, ch, conf,
    count, d, day, day1, d_high, d3, FileNo, galf,
    i, intval, iw, j, k, k3, Lbin, month, multpl, nch, nwthr, p,
    pm_high, pp, p_p_s, p_p_e, p_p_s_c, p_p_e_c,
    status, time, time1, year: {................................} INTEGER;
  date1, delta, am_peak, pm_peak, day_total: {......................} REAL;
  ChSum, dOut, dRpt, p_ch_sum, p_day, p_dt, p_h, p_wk, WkRpt: {......} CHAR;

  arday, armon, aryr: {..........................} ARRAY [1..7] of INTEGER;
  a_hi, a_pk, id_tot, iflag, p_hi, p_pk: {.....................} real_ar_9;
  bin, days: {.................................} ARRAY [1..12] of INTEGER;
  ChoTot, tot8: {.........................} ARRAY [1..7, 1..24] of INTEGER;
  Mdays: {...............................} ARRAY [1..7, 1..288] of INTEGER;
  Nwtot: {.........................} ARRAY [1..7, 1..9, 1..24] of INTEGER;

  run_sub: {..............................} ARRAY [1..7, 1..24] of REAL;
  wFlag: {................................} ARRAY [1..7, 1..24] of CHAR;
  dFlag: {................................} ARRAY [1..72, 1..4] of CHAR;
  pk_flag: {...............................} ARRAY [1..5, 1..9] of CHAR;

  am_min, d_min, pm_min: {.......................................} str_2;
  am_hour, d_hour, pm_hour: {....................................} str_3;
  a_e, a_s, p_e, p_s: {.....................................} STRING[4];
  begn: {...................................................} STRING[6];
  SiteNo: {.................................................} STRING[8];
  fName: {.................................................} STRING[14];
  dummy: {.................................................} STRING[29];
  counter: {...............................................} STRING[32];
  comment: {...............................................} STRING[70];
  astrik, blank: {........................................} str_2_ar_9;
  dir, loc, wthr: {.........................} ARRAY [1..7] of STRING[32];
  wt: {.....................................} ARRAY [1..30] of STRING[32];

  f, t: {..........................................................} TEXT;
```

```
PROCEDURE ClearArrays;
{ Clears the arrays Nwtot, wFlag, dFlag and
  defines the days of each month. }


BEGIN  { ClearArrays }
  FillChar (Nwtot, SizeOf (Nwtot), 0);
  FillChar (wFlag, SizeOf(wFlag), ' ');
  FillChar (dFlag, SizeOf(dFlag), ' ');
  FillChar (days, 12, 31);
  days[4] := 30; days[6] := 30;  days[9] := 30;  days[11] := 30;
END;  { ClearArrays }
```

```
PROCEDURE TalkToUser;
{ Communicates with the user to get the data file name and the output
  options.  Output options are to display or make a hard copy of input
  data, daily report on channel sum, daily report, or weekly report.}


BEGIN  { TalkToUser }
  WRITE (' Enter the data file name.    ');
  READLN ( fName );

  WRITE (' Want to see the input data? ( y/n )  ' );
  READLN ( dOut );
  WRITE (' Print out, too? ( y/n )  ' );
  READLN ( p_dt );

  WRITE (' Want to see the channel sum? ( y/n )  ' );
  READLN ( ChSum );
  WRITE (' Print out, too? ( y/n )  ' );
  READLN ( p_ch_sum );

  WRITE (' Want to see the daily report? ( y/n )  ' );
  READLN ( dRpt );
  WRITE (' Print out, too? ( y/n )  ' );
  READLN ( p_day );

  WRITE (' Want to see the weekly report? ( y/n )  ' );
  READLN ( WkRpt );
  WRITE (' Print out, too? ( y/n )  ' );
  READLN ( p_wk );

  WRITELN; WRITELN;
  WRITE (' Want to change the peak hours? ( y/n )  ' );
  READLN ( p_h );
  IF ( p_h = 'Y' ) OR ( p_h = 'y' ) THEN BEGIN
    WRITE (' Enter the AM peak start time ( hhmm )  ' );
    READLN ( a_p_s );
    WRITE (' Enter the AM peak end time ( hhmm )    ' );
    READLN ( a_p_e );
    WRITE (' Enter the PM peak start time ( hhmm )  ' );
    READLN ( p_p_s );
    WRITE (' Enter the PM peak end time ( hhmm )    ' );
    READLN ( p_p_e );
    END
  ELSE BEGIN
    a_p_s := 630;
    a_p_e := 930;
    p_p_s := 1530;
    p_p_e := 1830;
    END;

  WRITELN; WRITELN; WRITELN; WRITELN; WRITELN; WRITELN
END;  { TalkToUser }
```

```
PROCEDURE ReadHeader;
{ Reads the data file and outputs data if asked.  The data portion is
  copied into a temporary data file for the later use. }


VAR
  comnt: STRING[71];


BEGIN  { ReadHeader }
  ASSIGN ( f, fName );
  ASSIGN ( t, 'scratch.dat' );
  RESET ( f );
  REWRITE ( t );

  { Read and write header information }
  READLN ( f, comment );
  IF ( dOut = 'y' ) THEN WRITELN ( comment );
  IF ( p_dt = 'y' ) THEN WRITELN ( LST, ^L, comment );

  READLN ( f, dummy,counter );
  IF ( dOut = 'y' ) THEN WRITELN ( dummy, counter );
  IF ( p_dt = 'y' ) THEN WRITELN ( LST, dummy, counter );

  READLN ( f, dummy,nwthr );
  IF ( nwthr > 0 ) THEN
    FOR i := 1 TO nwthr DO BEGIN
      READLN ( f, dummy,wt[i] );
      IF ( dOut = 'y' ) THEN WRITELN ( dummy,wt[i] );
      IF ( p_dt = 'y' ) THEN WRITELN ( LST, dummy, wt[i] );
    END;
  FOR i := nwthr+1 TO 30 DO
    wt[i] := '    -                           ';

  READLN ( f, dummy,nch );
  IF ( dOut = 'y' ) THEN WRITELN ( dummy,nch );
  IF ( p_dt = 'y' ) THEN WRITELN ( LST, dummy,nch );
  FOR i := 1 TO nch DO BEGIN
    loc[i] := ' ';
    READLN ( f, dummy,loc[i] );
    IF ( dOut = 'y' ) THEN WRITELN ( dummy,loc[i] );
    IF ( p_dt = 'y' ) THEN WRITELN ( LST, dummy,loc[i] );
  END;
  loc[7] := 'Overall Channels';

  READLN ( f, dummy,nch );
  IF ( dOut = 'Y' ) THEN WRITELN ( dummy, nch );
  IF ( p_dt = 'y' ) THEN WRITELN ( LST, dummy, nch );
  FOR i := 1 TO nch DO BEGIN
    dir[i] := ' ';
    READLN ( f, dummy,dir[i] );
    IF ( dOut = 'y' ) THEN WRITELN ( dummy,dir[i] );
    IF ( p_dt = 'y' ) THEN WRITELN ( LST, dummy,dir[i] );
  END;
  dir[7] := 'Overall Channels';


  { Start to read the real data portion and
    write into the temporary data file t. }
  READLN ( f, begn,conf,FileNo,SiteNo,intval,status );
  IF ( dOut = 'y' ) THEN
    WRITELN ( begn,conf:3,FileNo:3,' ',SiteNo,intval:5,status:2 );
  IF ( p_dt = 'y' ) THEN
    WRITELN ( LST, begn,conf:3,FileNo:3,' ',SiteNo,intval:5,status:2 );

  READLN ( f, begn,comnt );
  REPEAT
    WRITELN ( t, COPY( begn,1,2 ),' ',COPY( begn,3,2 ),' ',COPY( begn,5,2 ),comnt );
    IF ( dOut = 'y' ) THEN WRITELN ( begn,comnt );
    IF ( p_dt = 'y' ) THEN WRITELN ( LST, begn,comnt );
    READLN ( f, begn,comnt )
  UNTIL ( COPY( begn,1,1 ) = '*' );
  IF ( dOut = 'y' ) THEN WRITELN ( begn,comnt );
  IF ( p_dt = 'y' ) THEN WRITELN ( LST, begn,comnt );

  CLOSE ( t );

END;  { ReadHeader }
```

```
PROCEDURE FindDayOfWeek;
{ Finds the day of week of the given date. }


VAR
  v1, v2, v3, v4, v5, v6, v7: INTEGER;


BEGIN  { FindDayOfWeek }
  year := 1900 + year;

  { Find the index for the array dates to convert dates }
  IF (month <= 2) THEN
    v1 := 1
  ELSE
    v1 := 0;

  v2 := year - v1;
  v3 := month + 12*v1;
  v4 := v2 DIV 100;
  v5 := v4 DIV 4;
  v6 := 5*v2 DIV 4;
  v7 := 13*( v3+1 ) DIV 5;

  p := v7 + v6 - v4 + v5 + day + 5;
  p := p - ( 7*( p DIV 7 )) + 1;

END;  { FindDayOfWeek }
```

```
PROCEDURE GetDatesOfWeek;
{ Gets the dates of the week to where the given day is belong. }


VAR
  dd, mm: INTEGER;


BEGIN  { GetDatesOfWeek }
  IF ((year MOD 4) = 0) THEN
    days[2] := 29
  ELSE
    days[2] := 28;

  mm := month;
  dd := day;

  FOR i := p DOWNTO 1 DO BEGIN
    armon[i] := mm;
    arday[i] := dd;
    aryr[i] := year;
    dd := dd - 1;
    IF (dd = 0) THEN BEGIN
      IF (mm = 1) THEN BEGIN
        mm := 13;
        year := year - 1
        END;
      mm := mm-1;
      dd:= days[mm]
      END
    END;

  mm := month;
  dd := day;

  FOR i := p TO 7 DO BEGIN
    armon[i] := mm;
    arday[i] := dd;
    aryr[i] := year;
    dd := dd + 1;
    IF (dd > days[mm]) THEN BEGIN
      IF (mm = 12) THEN BEGIN
        mm := 0;
        year := year + 1
        END;
      mm := mm + 1;
      dd := 1
      END
    END;

END;  { GetDatesOfWeek }
```

```
PROCEDURE DefineValues;
{ Defines the working variables. }


BEGIN  { DefineValues }
  IF (conf <= 3) THEN
    d3 := 1
  ELSE IF (conf <= 15) THEN
    d3 := 2
  ELSE
    d3 := 6;

  galf := 0;
  multpl := 60 DIV intval;
  delta := intval / 100;

  IF (ChSum = 'y') THEN
    nch := 7
  ELSE
    nch := d3;

  FOR i := 1 to p-1 DO
    wthr[i] := '    -                            ';
  iw := 1;
  FOR i := p TO 7 DO BEGIN
    wthr[i] := wt[iw];
    iw := iw + 1
    END;

  count := (time1 DIV 100)*multpl + (time1 MOD 100) DIV intval;

  { Initialize the tables for special character use in reports }
  k := 0;
  FOR i := 1 TO 4 DO
    FOR j := 1 TO 6*multpl DO BEGIN
      k := k + 1;
      IF (k < count) THEN
        dFlag[j,i] := ' '
      ELSE
        dFlag[j,i] := '*';
      END;

  FillChar(wFlag, (p-1)*24, '-');

  IF (intval = 60) THEN
    j := time1 DIV 100 - 1
  ELSE
    j := time1 DIV 100;

  FOR k := 1 TO j DO
    wFlag[p,k] := '-';

  IF ((time1 MOD 100) > intval) THEN
```

```
    wFlag[p,k+1] := '-';

  p := p - 1;

  { Define the start and end time of the am, pm peak hour. }
  a_p_s_c := ((a_p_s DIV 100) - 1)*multpl +
               (a_p_s MOD 100) DIV intval + 1;
  a_p_e_c := ((a_p_e DIV 100) - 1)*multpl +
               (a_p_e MOD 100) DIV intval;
  p_p_s_c := ((p_p_s DIV 100) - 1)*multpl +
               (p_p_s MOD 100) DIV intval + 1;
  p_p_e_c := ((p_p_e DIV 100) - 1)*multpl +
               (p_p_e MOD 100) DIV intval;

  Str (a_p_s:4, a_s);
  Str (a_p_e:4, a_e);
  Str (p_p_s:4, p_s);
  Str (p_p_e:4, p_e);
END;  { DefineValues }
```

```
PROCEDURE Read_Data ( knt: INTEGER );
{ Reads the data until the end of 24-hour data or file is reached. }


LABEL
  1, 2, 3;

VAR
  k: INTEGER;


BEGIN  { Read_Data }
  { Advance the date. }
  p := p + 1;
  IF ( p = 8 ) THEN BEGIN
    p := 1;
    GetDatesOfWeek;
    END;

  pp := p;
  day1 := day;
  Lbin := 12;
  { Clear the tables for the new day }
  FillChar (ChoTot, SizeOf( ChoTot ), 0 );
  FillChar (Mdays, SizeOf( Mdays ), 0 );

  { Read the data and add to the 24-hour table (Mdays) }
1:
  FOR i := 1 TO 12 DO BEGIN
    bin[i] := -1;
    READ ( t, bin[i] );
    IF ( bin[i] = -1 ) THEN BEGIN
      Lbin := i - 1;
      GOTO 2
      END
    END;

2:
  FOR i := 1 TO ( 12 DIV d3 ) DO BEGIN
    FOR j := 1 TO d3 DO BEGIN
      k := ( i-1 )*d3 + j;
      IF ( k > Lbin ) THEN GOTO 3;
      Mdays[j,knt] := bin[k];
      Mdays[7,knt] := Mdays[7,knt] + bin[k]
      END;

   knt := knt + 1;
   IF ( knt > ( 24*multpl ) ) THEN BEGIN
     knt := 1;
     pp := pp + 1
     END
  END;
  READ ( t, year,month,day,time );
  year := 1900 + year;
  IF ((NOT EOF( t )) AND ( time <> time1 )) THEN GOTO 1;

3:
 END;  { Read_Data }
```

```
PROCEDURE Build_Weekly_Table;
{ Called by Prepare_Wk_Report to assign the hourly volume
  counts in the slots of Nwtot after Read_Data subroutine. }


VAR
  d_from, d_to, hr, knt, new_hr: INTEGER;


BEGIN  { Build_Weekly_Table }
  d := p;
  hr := time1 DIV 100 + 1;
  IF ( intval = 60 ) THEN hr := hr - 1;
  knt := ( time1 MOD 100 ) DIV intval;

  IF (( knt=1 ) OR ( intval=60 )) THEN BEGIN
    FOR j := hr TO 24 DO Nwtot[i,d,j] := ChoTot[i,j];
    IF ( hr > 1 ) THEN BEGIN
      d := d + 1;
      FOR j := 1 TO hr-1 DO Nwtot[i,d,j] := ChoTot[i,j];
      END;
    END
  ELSE BEGIN
    IF ( knt < 1 ) THEN BEGIN
      hr := hr - 1;
      IF ( hr = 0 ) THEN BEGIN
        hr := 24;
        d := d - 1;
        END;
      d_to := count;
      d_from := count - multpl + 1;
      END
    ELSE BEGIN
      d_to := count + ( multpl - knt );
      d_from := count - knt + 1;
      END;

    IF ( Nwtot[i,d,hr] > 0 ) THEN
      FOR k := count TO d_to DO
        Nwtot[i,d,hr] := Nwtot[i,d,hr] + Mdays[i,k];

    new_hr := hr + 1;
    IF ( new_hr > 24 ) THEN BEGIN
      new_hr := 1;
      d := d + 1;
      END;
    FOR j := new_hr TO 24 DO Nwtot[i,d,j] := ChoTot[i,j];

    IF ( new_hr > 1 ) THEN BEGIN
      d := d + 1;
      FOR j := 1 TO hr-1 DO Nwtot[i,d,j] := ChoTot[i,j];
      FOR k := d_from TO count-1 DO
        Nwtot[i,d,j+1] := Nwtot[i,d,j+1] + Mdays[i,k];
      END
    ELSE BEGIN
      Nwtot[i,d,24] := Nwtot[i,d,24] - ChoTot[i,24];
      FOR k := d_from TO count-1 DO
        Nwtot[i,d,24] := Nwtot[i,d,24] + Mdays[i,k];
    END;
  END;

  { Care for the partial data }
  FOR j := 1 TO 24 DO BEGIN
    IF (wFlag[d,j] = '-') THEN Nwtot[i,d,j] := 0;
    IF (wFlag[d-1,j] = '-') THEN Nwtot[i,d-1,j] := 0;
    END;

END;  { Build_Weekly_Table }
```

```
PROCEDURE Peak_vol (option,first,last:INTEGER; VAR high:INTEGER;
                    VAR hr:str_3; VAR min:str_2; VAR pk_vol:REAL);
{ Calculates the one-hour peak volume and the starting time of that
  hour in the given period of time. }

VAR
  h, hi_index, m, new_hi, p1, p2: INTEGER;

BEGIN  { Peak_vol }
  high := 0;
  pk_vol := 0;

  CASE option OF
    1: BEGIN
         p1 := 1;
         p2 := 11*multpl;
         END;
    2: BEGIN
         p1 := 12*multpl;
         p2 := 23*multpl;
         END;
    3: BEGIN
         p1 := 1;
         p2 := 23*multpl;
         END;
    END;

  FOR j := p1 TO p2 DO BEGIN
    new_hi := 0;
    FOR k := 1 TO multpl DO new_hi := new_hi + Mdays[i,j+k];
    IF (new_hi > high) THEN BEGIN
      high := new_hi;
      hi_index := j;
      END;
    END;

  IF (first <> 1) THEN first := first + multpl;
  FOR j := first TO last+multpl DO
    pk_vol := pk_vol + Mdays[i,j];

  IF (high > 0) THEN BEGIN
    h := hi_index DIV multpl;
    m := (hi_index MOD multpl) * intval;
    Str (h:3, hr);
    Str (m:2, min);
    IF m < 10 THEN
      min[1] := '0';
    END
  ELSE BEGIN
    hr := ' 00';
    min := '00';
    END;

END;  { Peak_vol }
```

```
PROCEDURE Display_Daily_Head;
{ Outputs the heading of the daily report to the screen. }


BEGIN  { Display_Daily_Head }
  FOR j := 1 TO 51 DO WRITE (' ');
  WRITELN ('T E X A S   S D H P T');
  FOR j := 1 TO 51 DO WRITE (' ');
  FOR j := 1 TO 21 DO WRITE ('='); WRITELN;
  FOR j := 1 TO 55 DO WRITE (' ');
  WRITELN ('VOLUME COUNT'); WRITELN; WRITELN;

  WRITE ('          START DATE:    ', month:2,'-',day1:2,'-',year:4,
        '      SITE NO:    ', SiteNo);
  FOR j := 1 TO 18 DO WRITE (' ');
  WRITELN ('COUNTER:    ', counter);

  WRITE ('          WEATHER:    ', wthr[p]);
  FOR j := 1 TO (32 - Length(wthr[p])) DO WRITE (' ');
  WRITELN ('  DIRECTION OF FLOW:    ', dir[i]);

  WRITELN ('          LOCATION:    ', loc[i]);
  WRITELN ('          CHANNEL :          THIS IS ', i, ' OF ', d3);

  WRITELN;
  WRITE ('     ');
  FOR k := 1 TO 125 DO WRITE ('-');
  WRITELN;

  FOR j := 1 TO 4 DO BEGIN
    FOR k := 1 TO 26 DO WRITE (' ');
    WRITE ('HOURLY');
    END;
  WRITELN;

  FOR j := 1 TO 4 DO
    WRITE ('          TIME     VOLUME  VOLUME');
  WRITELN;

  WRITE ('     ');
  FOR j := 1 TO 125 DO WRITE ('='); WRITELN;

END;  { Display_Daily_Head }
```

```
PROCEDURE Display_Daily_Body;
{ Produces the body portion of the daily report on the screen. }


VAR
  l: INTEGER;
  col: ARRAY [1..4, 1..2] of REAL;


BEGIN  { Display_Daily_Body }
  k3 := 1;
  FOR j := 1 TO 4 DO col[j,1] := (j-1)*6.0;

  FOR j := 1 TO 6 DO BEGIN
    FOR k := 1 TO multpl DO BEGIN
      FOR l := 1 TO 4 DO col[l,2] := col[l,1] + delta;
      WRITE ('     ');
      IF (k = multpl) THEN
        FOR l := 1 TO 4 DO BEGIN
          col[l,2] := col[l,2] + 0.4;
          WRITE (dFlag[k3,l], col[l,1]:5:2, '-', col[l,2]:5:2,
                 Mdays[i,k3+(l-1)*6*multpl]:6, ChoTot[i,j+(l-1)*6]:9,
                 '     ');
          END
      ELSE
        FOR l := 1 TO 4 DO
          WRITE (dFlag[k3,l], col[l,1]:5:2, '-', col[l,2]:5:2,
                 Mdays[i,k3+(l-1)*6*multpl]:6, '               ');
      WRITELN;

      k3 := k3 + 1;
      FOR l := 1 TO 4 DO col[l,1] := col[l,2];
      END;

    WRITE ('   ');
    FOR k := 0 TO 3 DO BEGIN
      FOR l := 1 TO 24 DO WRITE (' ');
      WRITE ('(', run_sub[i,j+k*6]:5:0, ')');
      END;
    WRITELN;

    WRITE ('     ');
    FOR k := 1 TO 125 DO WRITE ('-');
    WRITELN;
    END;

END;  { Display_Daily_Body }
```

```
PROCEDURE Display_Daily_Sum;
{ Produces the daily statistics on the screen. }


BEGIN  { Display_Daily_Sum }
  WRITELN; WRITELN;
  WRITELN ('           AM PEAK (', (a_p_s DIV 100), ':',
           Copy (a_s, 3, 2), ' - ', (a_p_e DIV 100), ':',
           Copy (a_e, 3, 2), ') VOLUME:', am_peak:10:0,
           '           PM PEAK (', ((p_p_s - 1200) DIV 100), ':',
           Copy (p_s, 3, 2), ' - ', ((p_p_e - 1200) DIV 100), ':',
           Copy (p_e, 3, 2), ') VOLUME:', pm_peak:10:0,
           '        DAILY ');

  WRITE ('          AM PEAK HIGHEST HOUR VOLUME :', am_high:10,
         '           PM PEAK HIGHEST HOUR VOLUME :', pm_high:10,
         '        TOTAL:', day_total:10:0);

  IF (galf = 1) THEN
    WRITELN ('   +')
  ELSE
    WRITELN;

  WRITELN ('          (NEAREST', intval:3, ' MINUTE COUNT)',
           '                          (NEAREST', intval:3, ' MINUTE COUNT)',
           '                              ----------');

  WRITE ('          AM PEAK HOUR START :             ',
         am_hour, ':', am_min);
  WRITELN ('            PM PEAK HOUR START :             ',
           pm_hour, ':', pm_min);
  WRITELN;

  WRITE ('          DAILY PEAK HOUR START :          ',
         d_hour, ':', d_min);
  WRITELN ('            DAILY PEAK VOLUME :                 ', d_high:4);
  WRITELN;

  IF (galf = 1) THEN
    WRITELN ('          + -- INDICATES SUM IS NOT FULL 24-HOUR TOTAL');

  WRITELN ('         (  ) -- INDICATES CUMULATIVE HOURLY TOTAL',
           '          * -- INDICATES THE START DAY');

END;  { Display_Daily_Sum }
```

```
PROCEDURE Print_Daily_Head;


BEGIN  { Print_Daily_Head }
  WRITE (LST, ^L);
  FOR j := 1 TO 51 DO WRITE (LST, ' ');
  WRITELN (LST, 'T E X A S   S D H P T');
  FOR j := 1 TO 51 DO WRITE (LST, ' ');
  FOR j := 1 TO 21 DO WRITE (LST, '='); WRITELN(LST);
  FOR j := 1 TO 55 DO WRITE (LST, ' ');
  WRITELN (LST, 'VOLUME COUNT',^J,^J);

  WRITE (LST, '          START DATE:   ', month:2,'-',day1:2,'-',year:4,
         '        SITE NO:   ', SiteNo);
  FOR j := 1 TO 18 DO WRITE (LST, ' ');
  WRITELN (LST, 'COUNTER:   ', counter);

  WRITE (LST, '          WEATHER:   ', wthr[p]);
  FOR j := 1 TO (32 - Length(wthr[p])) DO WRITE (LST, ' ');
  WRITELN (LST, '  DIRECTION OF FLOW:   ', dir[i]);

  WRITELN (LST, '          LOCATION:   ', loc[i]);
  WRITELN (LST, '          CHANNEL :          THIS IS ', i, ' OF ', d3);
  WRITELN (LST);

  WRITE (LST, '    ');
  FOR k := 1 TO 125 DO WRITE (LST, '-');
  WRITELN (LST);

  FOR j := 1 TO 4 DO BEGIN
    FOR k := 1 TO 26 DO WRITE (LST, ' ');
    WRITE (LST, 'HOURLY');
    END;
  WRITELN (LST);

  FOR j := 1 TO 4 DO
    WRITE (LST, '         TIME     VOLUME  VOLUME');
  WRITELN (LST);
  WRITE (LST, '    ');
  FOR j := 1 TO 125 DO WRITE (LST, '='); WRITELN (LST);

END;  { Print_Daily_Head }
```

```
PROCEDURE Print_Daily_Body;


VAR
  l: INTEGER;
  col: ARRAY [1..4, 1..2] of REAL;


BEGIN  { Print_Daily_Body }
  k3 := 1;
  FOR j := 1 TO 4 DO col[j,1] := (j-1)*6.0;

  FOR j := 1 TO 6 DO BEGIN
    FOR k := 1 TO multpl DO BEGIN
      FOR l := 1 TO 4 DO col[l,2] := col[l,1] + delta;
      WRITE (LST, '    ');
      IF (k = multpl) THEN
        FOR l := 1 TO 4 DO BEGIN
          col[l,2] := col[l,2] + 0.4;
          WRITE (LST, dFlag[k3,l], col[l,1]:5:2, '-', col[l,2]:5:2,
                 Mdays[i,k3+(l-1)*6*multpl]:6, ChoTot[i,j+(l-1)*6]:9,
                 '      ');
          END
      ELSE
        FOR l := 1 TO 4 DO
          WRITE (LST, dFlag[k3,l], col[l,1]:5:2, '-', col[l,2]:5:2,
                 Mdays[i,k3+(l-1)*6*multpl]:6, '               ');
      WRITELN (LST);

      k3 := k3 + 1;
      FOR l := 1 TO 4 DO col[l,1] := col[l,2];
      END;

    WRITE (LST, '  ');
    FOR k := 0 TO 3 DO BEGIN
      FOR l := 1 TO 24 DO WRITE (LST, ' ');
      WRITE (LST, '(', run_sub[i,j+k*6]:5:0, ')');
      END;
    WRITELN (LST);

    WRITE (LST, '    ');
    FOR k := 1 TO 125 DO WRITE (LST, '-');
    WRITELN (LST);
    END;

END;  { Print_Daily_Body }
```

```
PROCEDURE Print_Daily_Sum;

BEGIN  { Print_Daily_Sum }
  WRITELN ( LST, ^J, ^J, '          AM PEAK ( ', ( a_p_s DIV 100 ), ':',
           Copy ( a_s, 3, 2 ), ' - ', ( a_p_e DIV 100 ), ':',
           Copy ( a_e, 3, 2 ), ' ) VOLUME:', am_peak:10:0,
           '            PM PEAK ( ', (( p_p_s - 1200 ) DIV 100 ), ':',
           Copy ( p_s, 3, 2 ), ' - ', (( p_p_e - 1200 ) DIV 100 ), ':',
           Copy ( p_e, 3, 2 ), ' ) VOLUME:', pm_peak:10:0,
           '         DAILY ' );

  WRITE ( LST, '          AM PEAK HIGHEST HOUR VOLUME :', am_high:10,
         '            PM PEAK HIGHEST HOUR VOLUME :', pm_high:10,
         '         TOTAL:', day_total:10:0 );

  IF ( galf = 1 ) THEN
    WRITELN ( LST, '    +' )
  ELSE
    WRITELN ( LST );

  WRITELN ( LST, '          ( NEAREST', intval:3, ' MINUTE COUNT )',
           '                    ( NEAREST', intval:3,
           ' MINUTE COUNT )', '                            ----------' );

  WRITELN ( LST, '          AM PEAK HOUR START :            ',
           am_hour, ':', am_min,
           '            PM PEAK HOUR START :            ',
           pm_hour, ':', pm_min, ^J, ^J );

  WRITELN ( LST, '          DAILY PEAK HOUR START :         ',
           d_hour, ':', d_min,
           '            DAILY PEAK VOLUME :                 ',
           d_high:4, ^J, ^J );

  IF ( galf = 1 ) THEN
    WRITELN ( LST, '        + -- INDICATES SUM IS NOT FULL 24-HOUR TOTAL' );

  WRITELN ( LST, '          (   ) -- INDICATES CUMULATIVE HOURLY TOTAL',
           '            * -- INDICATES THE START DAY' );

END;  { Print_Daily_Sum }
```

```
PROCEDURE Generate_Daily_Report;
{ Generates the daily reports for all channels per day. }


BEGIN  { Generate_Daily_Report }
  IF ((dRpt = 'y') OR (p_day = 'y')) THEN
    FOR i := 1 TO nch DO
      IF ((i <= d3) OR (i = 7)) THEN BEGIN
        Peak_vol (1, a_p_s_c,a_p_e_c, am_high,am_hour,am_min,am_peak);
        Peak_vol (2, p_p_s_c,p_p_e_c, pm_high,pm_hour,pm_min,pm_peak);
        Peak_vol (3, 1,23*multpl, d_high,d_hour,d_min,day_total);

        IF (day_total > 0) THEN BEGIN
          IF (dRpt = 'y') THEN BEGIN
            Display_Daily_Head;
            Display_Daily_Body;
            Display_Daily_Sum;
            END;
          IF (p_day = 'y') THEN BEGIN
            Print_Daily_Head;
            Print_Daily_Body;
            Print_Daily_Sum;
            END;
          END;
        END;

END;  { Generate_Daily_Report }
```

```
PROCEDURE Calc_Wk_Avg_Vol;
{ Calculate the 5-day and 7-day average volumes of each hour in a week. }


VAR
  cnt_5d, cnt_7d: ARRAY [1..24] of INTEGER;


BEGIN  { Calc_Wk_Avg_Vol }
  FOR k := 1 TO 24 DO BEGIN
    cnt_5d[k] := 0;
    cnt_7d[k] := 0;
    Nwtot[i,8,k] := 0;
    Nwtot[i,9,k] := 0;
    wFlag[8,k] := '-';
    wFlag[9,k] := '-';

    FOR j := 1 TO 7 DO
      IF (wFlag[j,k] = ' ') THEN BEGIN
        wFlag[8,k] := ' ';
        wFlag[9,k] := ' ';
        Nwtot[i,9,k] := Nwtot[i,9,k] + Nwtot[i,j,k];
        cnt_7d[k] := cnt_7d[k] + 1;
        IF ( j <= 5 ) THEN BEGIN
          Nwtot[i,8,k] := Nwtot[i,8,k] + Nwtot[i,j,k];
          cnt_5d[k] := cnt_5d[k] + 1;
          END;
        END;

    IF (cnt_7d[k] > 0) THEN
      Nwtot[i,9,k] := Nwtot[i,9,k] DIV cnt_7d[k]
    ELSE
      wFlag[9,k] := '-';

    IF (cnt_5d[k] > 0) THEN
      Nwtot[i,8,k] := Nwtot[i,8,k] DIV cnt_5d[k]
    ELSE
      wFlag[8,k] := '-';

    END;

END;  { Calc_Wk_Avg_Vol }
```

```
PROCEDURE Extract_Daily_Stats;
{ Calculates the daily statistics for a weekly report. }


BEGIN  { Extract_Daily_Stats }
  FOR j := 1 TO 9 DO BEGIN
    a_pk[j] := 0;
    p_pk[j] := 0;
    a_hi[j] := 0;
    p_hi[j] := 0;
    iflag[j] := 0;
    id_tot[j] := 0;
    FOR k := 1 TO 2 DO astrik[j,k] := ' ';
    FOR k := 1 TO 5 DO pk_flag[k,j] := ' ';

    { Get the Peak volume stats }
    FOR k := 1 TO 24 DO BEGIN
      IF (wFlag[j,k] = '-') THEN iflag[j] := 1
      ELSE id_tot[j] := id_tot[j] + Nwtot[i,j,k];

      IF (k <= 12) THEN BEGIN
        IF ((k > 6) AND (k <= 9)) THEN a_pk[j] := a_pk[j] + Nwtot[i,j,k];
        IF (Nwtot[i,j,k] > a_hi[j]) THEN a_hi[j] := Nwtot[i,j,k];
        END
      ELSE BEGIN
        IF (Nwtot[i,j,k] > p_hi[j]) THEN p_hi[j] := Nwtot[i,j,k];
        IF ((k > 15) AND (k <= 18)) THEN p_pk[j] := p_pk[j] + Nwtot[i,j,k];
        END;
      END;

    { Check the availability of data }
    IF (a_pk[j] = 0) THEN pk_flag[1,j] := '-';
    IF (a_hi[j] = 0) THEN pk_flag[2,j] := '-';
    IF (p_pk[j] = 0) THEN pk_flag[3,j] := '-';
    IF (p_hi[j] = 0) THEN pk_flag[4,j] := '-';
    IF (id_tot[j] = 0) THEN pk_flag[5,j] := '-'
    ELSE
      IF (iflag[j] = 1) THEN BEGIN
        pk_flag[5,j] := ' ';
        astrik[j] := ' +';
        END;
    END;

END;  { Extract_Daily_Stats }
```

```
PROCEDURE Display_Sum_line (p1:str_21; p2:real_ar_9;
                            p3:INTEGER; p4:str_2_ar_9);


BEGIN  { Display_Sum_line }
  WRITE ('      ', p1, '  ');
  FOR j := 1 TO 7 DO
    IF (pk_flag[p3,j] = '-') THEN
      WRITE ('     ', pk_flag[p3,j], '    ')
    ELSE
      WRITE ('  ', p2[j]:6:0, p4[j]);

    IF (pk_flag[p3,8] = '-') THEN
      WRITE ('          ', pk_flag[p3,8], '  ')
    ELSE
      WRITE ('     ', p2[8]:6:0, p4[8]);

    IF (pk_flag[p3,9] = '-') THEN WRITE ('       ', pk_flag[p3,9])
    ELSE WRITE ('  ', p2[9]:6:0, p4[9]);
    WRITELN;

END;  { Display_Sum_line }
```

```
PROCEDURE Display_Wk_Head;


BEGIN  { Display_Wk_Head }
  FOR j := 1 TO 51 DO WRITE (' ');
  WRITELN ('T E X A S   S D H P T');

  FOR j := 1 TO 52 DO WRITE (' ');
  FOR j := 1 TO 21 DO WRITE ('='); WRITELN; WRITELN;

  FOR j := 1 TO 48 DO WRITE (' ');
  WRITELN ('WEEKLY VOLUME COUNT SUMMARY SHEET'); WRITELN; WRITELN; WRITELN;

  FOR j := 1 TO 15 DO WRITE (' ');
  WRITELN ('LOCATION :      ', loc[i]);

  FOR j := 1 TO 15 DO WRITE (' ');
  WRITELN ('DATE :          ', armon[1]:2, '-', arday[1]:2, '-', aryr[1]:4,
           '              DIRECTION OF FLOW :      ', dir[i]);

  FOR j := 1 TO 15 DO WRITE (' ');
  WRITELN ('CHANNEL :       THIS IS ', i:2, ' OF ', d3:2,
           '    SITE NO :      ', SiteNo:8); WRITELN;

  FOR j := 1 TO 13 DO WRITE (' ');
  WRITELN ('DAY OF WEEK:      MONDAY   TUESDAY  WEDNESDAY  ',
           'THURSDAY   FRIDAY   SATURDAY   SUNDAY      5 - DAY   7 - DAY');

  WRITE ('             WEATHER          ');
  FOR j := 1 TO 7 DO WRITE (Copy(wthr[j],1,9):9, ' '); WRITELN;

  FOR j := 1 TO 30 DO WRITE (' ');
  FOR J := 1 TO 7 DO WRITE (armon[j]:2, ' / ', arday[j]:2, '    ');
  WRITELN ('   AVERAGE   AVERAGE');

  WRITE ('             ');
  FOR j := 1 TO 107 DO WRITE ('-'); WRITELN;

  WRITE ('                TIME           ');
  FOR j := 1 TO 7 DO WRITE ('HOURLY     ');
  WRITELN ('   HOURLY    HOURLY ');

  FOR j := 1 TO 31 DO WRITE (' ');
  FOR j := 1 TO 7 DO WRITE ('VOLUME     ');
  WRITELN ('   VOLUME    VOLUME '); WRITELN;

END;  { Display_Wk_Head }
```

```
PROCEDURE Display_Wk_Body;


BEGIN  { Display_Wk_Body }
  { Body }
  FOR k := 1 TO 24 DO BEGIN
    WRITE ('                ', k-1:2, '-', k:2, '            ');
    FOR j := 1 TO 9 DO
      IF (wFlag[j,k] = '-') THEN
        IF (j = 8) THEN WRITE ('        ', wFlag[j,k], '     ')
        ELSE WRITE ('   ', wFlag[j,k], '      ')
      ELSE
        IF (j = 8) THEN WRITE ('   ', Nwtot[i,j,k]:6, '    ')
        ELSE WRITE (Nwtot[i,j,k]:6, '    ');

    WRITELN;
    END;

  { Summary }
  WRITE ('       ');
  FOR j := 1 TO 114 DO WRITE ('-');
  WRITELN;
  FillChar (blank, SizeOf(blank), ' ');

  Display_Sum_line ('AM PEAK (6-9) VOLUME:', a_pk, 1, blank);
  Display_Sum_line ('AM PEAK HIGHEST HOUR:', a_hi, 2, blank);
  Display_Sum_line ('PM PEAK (3-6) VOLUME:', p_pk, 3, blank);
  Display_Sum_line ('PM PEAK HIGHEST HOUR:', p_hi, 4, blank);
  Display_Sum_line ('DAILY TOTAL:         ', id_tot, 5, astrik);

  WRITELN;
  WRITELN ('               + --- INDICATES SUM IS NOT FULL 24-HOUR TOTAL');
  WRITELN ('               - --- INDICATES DATA IS UNAVAILABLE');

END;  { Display_Wk_Body }
```

```
PROCEDURE Print_Sum_line (p1:str_21; p2:real_ar_9;
                          p3:INTEGER; p4:str_2_ar_9);


BEGIN  { Print_Sum_line }
  WRITE (LST, '      ', p1, '  ');
  FOR j := 1 TO 7 DO
    IF (pk_flag[p3,j] = '-') THEN
      WRITE (LST, '     ', pk_flag[p3,j], '    ')
    ELSE
      WRITE (LST, '  ', p2[j]:6:0, p4[j]);

    IF (pk_flag[p3,8] = '-') THEN
      WRITE (LST, '          ', pk_flag[p3,8], '  ')
    ELSE
      WRITE (LST, '     ', p2[8]:6:0, p4[8]);

    IF (pk_flag[p3,9] = '-') THEN
      WRITE (LST, '       ', pk_flag[p3,9])
    ELSE
      WRITE (LST, '  ', p2[9]:6:0, p4[9]);

    WRITELN (LST);

END;  { Print_Sum_line }
```

```
PROCEDURE Print_Wk_Head;

BEGIN
  WRITE ( LST, ^L );
  FOR j := 1 TO 52 DO WRITE ( LST, ' ' );
  WRITELN ( LST, 'T E X A S   S D H P T' );

  FOR j := 1 TO 52 DO WRITE ( LST, ' ' );
  FOR j := 1 TO 21 DO WRITE ( LST, '=' ); WRITELN ( LST, ^J );

  FOR j := 1 TO 48 DO WRITE ( LST, ' ' );
  WRITELN ( LST, 'WEEKLY VOLUME COUNT SUMMARY SHEET' );
  WRITELN ( LST, ^J, ^J );

  FOR j := 1 TO 15 DO WRITE ( LST, ' ' );
  WRITELN ( LST, 'LOCATION :      ', loc[i] );

  FOR j := 1 TO 15 DO WRITE ( LST, ' ' );
  WRITELN ( LST, 'DATE :          ', armon[1]:2, '-', arday[1]:2, '-',
           aryr[1]:4, '           DIRECTION OF FLOW :     ', dir[i] );

  FOR j := 1 TO 15 DO WRITE ( LST, ' ' );
  WRITELN ( LST, 'CHANNEL :        THIS IS ', i:2, ' OF ', d3:2,
           '    SITE NO :     ', SiteNo:8, ^J );

  FOR j := 1 TO 13 DO WRITE ( LST, ' ' );
  WRITELN ( LST, 'DAY OF WEEK:      MONDAY   TUESDAY  WEDNESDAY  ',
           'THURSDAY   FRIDAY   SATURDAY   SUNDAY      5 - DAY   7 - DAY' );

  WRITE ( LST, '             WEATHER          ' );
  FOR j := 1 TO 7 DO WRITE ( LST, Copy( wthr[j],1,9 ):9, ' ' ); WRITELN ( LST );

  FOR j := 1 TO 30 DO WRITE ( LST, ' ' );
  FOR J := 1 TO 7 DO WRITE ( LST, armon[j]:2, ' / ', arday[j]:2, '    ' );
  WRITELN ( LST, '   AVERAGE   AVERAGE' );

  WRITE ( LST, '             ' );
  FOR j := 1 TO 107 DO WRITE ( LST, '-' ); WRITELN ( LST );

  WRITE ( LST, '                TIME          ' );
  FOR j := 1 TO 7 DO WRITE ( LST, 'HOURLY    ' );
  WRITELN ( LST, '   HOURLY    HOURLY ' );

  FOR j := 1 TO 31 DO WRITE ( LST, ' ' );
  FOR j := 1 TO 7 DO WRITE ( LST, 'VOLUME    ' );
  WRITELN ( LST, '   VOLUME    VOLUME ', ^J );

END;
```

```
PROCEDURE Print_Wk_Body;

BEGIN
  { Body }
  FOR k := 1 TO 24 DO BEGIN
    WRITE ( LST, '               ', k-1:2, '-', k:2, '           ' );
    FOR j := 1 TO 9 DO
      IF ( wFlag[j,k] = '-' ) THEN
        IF ( j = 8 ) THEN WRITE ( LST, '        ', wFlag[j,k], '    ' )
        ELSE WRITE ( LST, '    ', wFlag[j,k], '      ' )
      ELSE
        IF ( j = 8 ) THEN WRITE ( LST, '   ', Nwtot[i,j,k]:6, '     ' )
        ELSE WRITE ( LST, Nwtot[i,j,k]:6, '    ' );

    WRITELN ( LST );
    END;

  { Summary }
  WRITE ( LST, '      ' );
  FOR j := 1 TO 114 DO WRITE ( LST, '-' );
  WRITELN ( LST );
  FillChar ( blank, SizeOf( blank ), ' ' );

  Print_Sum_line ( 'AM PEAK ( 6-9 ) VOLUME:', a_pk, 1, blank );
  Print_Sum_line ( 'AM PEAK HIGHEST HOUR:', a_hi, 2, blank );
  Print_Sum_line ( 'PM PEAK ( 3-6 ) VOLUME:', p_pk, 3, blank );
  Print_Sum_line ( 'PM PEAK HIGHEST HOUR:', p_hi, 4, blank );
  Print_Sum_line ( 'DAILY TOTAL:         ', id_tot, 5, astrik );

  WRITELN ( LST );
  WRITELN ( LST, '               + --- INDICATES SUM IS NOT FULL ',
                 '24-HOUR TOTAL' );
  WRITELN ( LST, '               - --- INDICATES DATA IS UNAVAILABLE' );

END;
```

```
PROCEDURE Generate_Weekly_Report;

BEGIN
  Calc_Wk_Avg_Vol;
  Extract_Daily_Stats;

  IF ( id_tot[9] > 0 ) THEN BEGIN
    IF (WkRpt = 'y' ) THEN BEGIN
      Display_Wk_Head;
      Display_Wk_Body;
      END;

    IF ( p_wk = 'y' ) THEN BEGIN
      Print_Wk_Head;
      Print_Wk_Body;
      END;
    END;

  { Reinitialize the arrays for the next week. }
  IF ( i = nch ) THEN BEGIN
    FillChar (Nwtot, SizeOf( Nwtot ), 0 );
    FillChar (wFlag, SizeOf( wFlag ), ' ' );

    FOR J := 1 TO 7 DO BEGIN
      wthr[j] := wt[iw];
      iw := iw + 1;
      END;

    END;
END;
```

```
PROCEDURE Perform_Daily_Operations;
{ Controls all the routines for the daily operations until the end of
  data file is reached. }

VAR
  end_time, next_end_time, start_time, time_diff, total_min: INTEGER;

  PROCEDURE Prepare_Wk_Report;
  { Builds weekly table and generates weekly report. }

  BEGIN
    FOR i := 1 TO nch DO
      IF ((i<=d3) OR (i=7)) THEN
        Build_Weekly_Table;

    IF EOF(t) THEN BEGIN
      { Mark the remaining hour slots not available for data }
      total_min := time DIV 100 * 60 + time MOD 100 +
                   ((Lbin DIV d3) - 1) * intval;
      next_end_time := (total_min MOD 1440) DIV 60;
      IF (next_end_time > 0) THEN BEGIN
        IF (pp > 7) THEN BEGIN
          pp := 1;
          FOR i := 1 TO nch DO
            IF ((i <= d3) OR (i = 7)) THEN BEGIN
              FOR j := 1 TO 24 DO Tot8[i,j] := Nwtot[i,8,j];
              Generate_Weekly_Report;
              FOR j := 1 TO 24 DO Nwtot[i,1,j] := Tot8[i,j];
              END;
          p := 1;
          GetDatesOfWeek;
          END;
        FOR i := 1 to nch DO
          IF ((i <= d3) OR (i=7)) THEN Nwtot[i,pp,next_end_time+1] := 0;
        FOR i := next_end_time+1 TO 24 DO wFlag[pp,i] := '-';
        END;

      FOR i := pp+1 TO 7 DO
        FOR j := 1 TO 24 DO wFlag[i,j] := '-';

      FOR i := 1 TO nch DO
        IF ((i <= d3) OR (i = 7)) THEN
          Generate_Weekly_Report;
      END

    ELSE IF (p = 7) THEN
      FOR i := 1 to nch DO
        IF ((i <= d3) OR (i = 7)) THEN BEGIN
          FOR j := 1 TO 24 DO Tot8[i,j] := Nwtot[i,8,j];
          Generate_Weekly_Report;
          FOR j := 1 TO 24 DO Nwtot[i,1,j] := Tot8[i,j];
          END;

    END; { Prepare_Wk_Report }


BEGIN { Perform_Daily_Operations }
  Read_Data ( count );

  { Compute the hourly totals and running subtotals }
  FOR ch := 1 TO 7 DO
    FOR i := 1 TO 24 DO BEGIN
      run_sub[ch,i] := 0;
      FOR j := 1 TO multpl DO
        ChoTot[ch,i] := ChoTot[ch,i] + Mdays[ch,( i-1 )*multpl+j];
      IF ( i > 1 ) THEN
        run_sub[ch,i] := run_sub[ch,i-1] + ChoTot[ch,i]
      ELSE
        run_sub[ch,i] := ChoTot[ch,i];
      END;

  IF EOF( t ) THEN BEGIN
    { Calculate the length of the last day's data. }
    end_time := ( time DIV 100 * 60 ) + ( time MOD 100 ) +
                ( ( Lbin-1 ) DIV d3 * intval );
    start_time := ( time1 DIV 100 * 60 ) + ( time1 MOD 100 );
    IF ( start_time <> ( end_time+intval ) ) THEN galf := 1;
    IF ( start_time > end_time ) THEN end_time := end_time + 24*60;
    time_diff := ( end_time - start_time ) DIV 60;
    END;

  IF ( NOT EOF( t ) ) OR ( time_diff >= 3 ) THEN
    Generate_Daily_Report;


  IF ( ( WkRpt='y' ) OR ( p_wk = 'y' ) ) THEN
    Prepare_Wk_Report;

END; { Perform_Daily_Operations }
```

```
BEGIN { Main }
  ClearArrays;
  TalkToUser;
  ReadHeader;

  RESET ( t );
  READ ( t, year,month,day,time1 );
  FindDayOfWeek;
  GetDatesOfWeek;
  DefineValues;

  RESET ( t );
  READ ( t, year,month,day,time1 );

  WHILE ( NOT EOF( t ) ) DO
    Perform_Daily_Operations;

  { Delete the temporary data file. }
  ERASE ( t );

END.
```