TECHNICAL REPORT STANDARD TITLE PAGE

1. Report No.
FHWA/TX-87/31+496-1F

2. Government Accession No.

3. Recipient's Catalog No.

4. Title and Subtitle
ACR Scheduling and Information Management System

5. Report Date
January 1987

6. Performing Organization Code

7. Author(s)
Jae Y. Lee, Duk-Jin Chang and Daniel J. Vitello

8. Performing Organization Report No.
Research Report 496-1F

9. Performing Organization Name and Address
Texas Transportation Institute
The Texas A&M University System
College Station, Texas 78763

10. Work Unit No.

11. Contract or Grant No.
Study No. 2-10-85-496

13. Type of Report and Period Covered
Final - September 1984
January 1987

12. Sponsoring Agency Name and Address
Texas State Department of Highways and Public
Transportation; Transportation Planning Division
P. O. Box 5051
Austin, Texas 78763

14. Sponsoring Agency Code

15. Supplementary Notes
Research performed in cooperation with DOT, FHWA.
Research Study Title: ACR Scheduling and Information Management System

16. Abstract

This report documents two different microcomputer based ACR scheduling software packages that can be used by the SDHPT personnel to efficiently manage the Department's annual ACR scheduling and data collection activities.

First method uses the dBASE-III microcomputer database management system to setup a master database files that are similar to the current year's schedule. Programs were written using the dBASE-III programing language to access the master database to generate ACR data collection schedule on daily, weekly or entire District.

Second method uses a modified Travelling Salesman algorithm to generate optimal schedule for individual county's ACR data collection activities. A network of linknode diagrams representing each county was developed and entered into a data base. Programs were written in the Microsoft FORTRAN-77 language to implement the Travelling Salesman algorithm to generate an optimal ACR schedule for individual county.

Both systems can be used to setup the entire State ACR data collection stations for future year's scheduling activities.

17. Key Words
Database Management Systems, Network Optimization and Scheduling, dBASE-III programs, FORTRAN-77 programs, Microcomputer

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
81

22. Price

ACR SCHEDULING AND INFORMATION MANAGEMENT SYSTEM

by

Jae Y. Lee
Systems Analyst

Duk-Jin Chang
Research Assistant

and

Daniel J. Vitello
Research Associate

Research Report 496-1F
Research Study 2-10-85-496
ACR Scheduling and Information Management System

Sponsored by

Texas State Department of Highways and Public Transportation

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

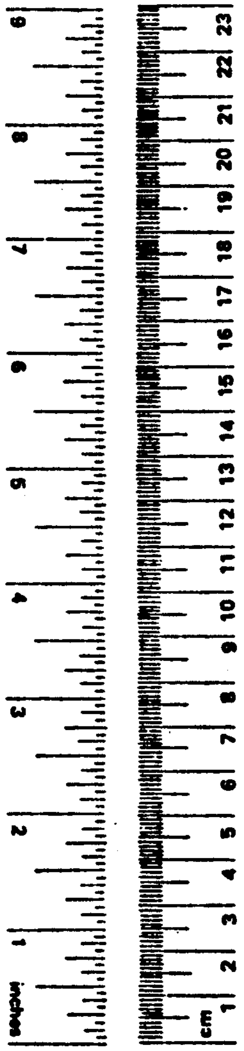
TEXAS TRANSPORTATION INSTITUTE
The Texas A&M University System
College Station, Texas

January 1987

METRIC CONVERSION FACTORS

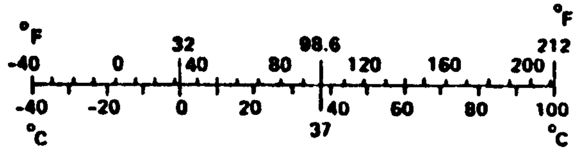
Approximate Conversions to Metric Measures

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



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



*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.

ACKNOWLEDGEMENT

This study was sponsored by the Texas State Department of Highways and Public Transportation (SDHPT) and Federal Highways Administration. Jae Y. Lee was the Study Supervisor and Curtis Goss was the SDHPT Contact Representative. Jack Crumley of D-10R provided guidance and expertise in the Department's current ACR scheduling activities.

ABSTRACT

This report documents two different microcomputer based ACR scheduling software packages that can be used by the SDHPT personnel to efficiently manage the Department's annual ACR scheduling and data collection activities.

First method uses the dBASE-III microcomputer database management system to setup a master database files that are similar to the current year's schedule. Programs were written using the dBASE-III programming language to access the master database to generate ACR data collection schedule on daily, weekly or entire District level.

Second method uses a modified Travelling Salesman algorithm to generate optimal schedule for individual county's ACR data collection acitvities. A network of link-node diagrams representing each county was developed and entered into a data base. Programs were written in the Microsoft FORTRAN-77 language to implement the Travelling Salesman algorithm to generate an optimal ACR schedule for individual county.

Both systems can be used to setup the entire State ACR data collection stations for future year's scheduling activities.

KEY WORDS : Database Management Systems, Network Optimization and Scheduling, dBASE-III programs, FORTRAN-77 programs, Microcomputer.

SUMMARY

In the planning, design and maintenance of the highway systems, accurate traffic data collection system provides the needed information to the transportation engineers. This information formulates the foundation for any decision-making process for effective highway management system.

Presently, the project supervisor manually schedules all the ACR data collection stations throught the State. Schedule is determined for each member of the field data collection crew for one week at a time. This manual scheduling process involves looking at the map of each county's ACR station locations, picking the paths which appear to be the shortest and then estimating the travelled milage for each member. This manual scheduling procedure is time consuming and very human dependent. Also, since the data collection stations remain virtually the same from one year to the next, with only minor additions or deletion of a few stations, the scheduling procedure should not be repeated each year with the same degree of difficulties.

Two different microcomputer based methods were developed to aid the SDHPT personnel in the ACR scheduling activities. First, a dBASE-III database management system was used to generate a master database which contains all the station information for the District 17. Any future ACR data collection schedule can be generated directly from this master database. Secondly, a network optimization program was developed in FORTRAN-77 to generate an optimal schedule that can be used for ACR data collection activities.

IMPLEMENTATION STATEMENT

The software product developed under this study is designed to be a tool to aid in the ACR scheduling activities. Database was setup only for District 17 and should be extended to incorporate the entire State.

The software described in theis report will enable faster ACR schedule generation, efficient management of ACR data collection activities, as well as other managerial activities.

DISCLAIMER

The contents of this report reflect the views of the authors who are responsible for the facts and the accuracy of the programs presented within. The usage of dBASE-III and Microsoft FORTRAN-77 compiler does not show any approval of these software packages over any other similar systems. The contents do not necessarily reflect the official views or policies of the Texas State Department of Highways and Public Transportation or the Fedral Highway Administration.

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. Every year the State Department of Highways and Public Transportaion (SDHPT) installs Automatic Cumulative Recorders (ACR's) at several thousand locations (stations) throughout the state of Texas. These ACR equipments use pneumatic tubes to collect traffic volume data for a 24-hour period. Installation of the ACR equipment is done by a SDHPT field personnel who follows a predetermined schedule for installation. There are several people who travel around the State installing the ACR's, each in a different region. These Automatic Cumulative Recorders are installed on what is actually a two-day schedule. The first day is spent installing the equipments on the roadway and on the second day, 24-hours later, they are picked up and the data is retrieved from the ACR equipment. On the average, 45 ACR equipments are installed and removed during each two-day period.

Presently, the project supervisor manually schedules all the ACR data collection stations throughout the State. Schedule is determined for each member of the field data collection crew for one week at a time. This manual scheduling involves looking at the map of each county's ACR station locations, picking the paths which appear to be shortest and then determining, (by using the measuring wheel), the distance that will be travelled by each member. This process is repeated for each person and for every district where the SDHPT collects the traffic volume data. This manual scheduling procedure is time consuming and very human dependent. Also, since the data collection stations remain virtually the same from one year to the next, with only minor additions or deletions of a few stations, the schedule producing process should not be repeated every year with the same degree of difficulties.

A better method of scheduling and managing the data collection efforts is needed to assist the SDHPT personnel. This report documents two such methods developed using the microcomputer system to assist the SDHPT personnel in ACR scheduling and overall management of the ACR data collection activities.

SITUATION

Current manual ACR data collection scheduling procedure is very human dependent and time consuming. The supervisor is totally responsible for making up the schedule for each field personnel for the entire State. He must be fully aware of the current status of the data collection activities at all times for the entire State to make further manpower adjustments if necessary. Any stations that were left out or missed due to bad conditions must be rescheduled at a later date. The supervisor must be aware of any such stations for further considerations and rescheduling. The scheduling procedure itself is very time consuming since the schedule must be created for each field personnel every year. For each schedule, the supervisor must determine not only the sequence of ACR equipment installation but also the milage travelled for each day. This estimated milage is determined using a distance wheel by tracing the route followed by the field personnel. The supervisor also uses different coloring schemes to distinguish those stations which have already been scheduled, data collected, scheduled but missed, etc. The situation described so far excludes any special urban studies which may increase the data collection activities tremendously depending on the urban area selected. Clearly, the supervisor needs some assistance in order to better manage the data collection activities and the field personnel.

This report documents two different procedures developed using the IBM-PC compatible microcomputer that can be used to assist the SDHPT personnel in generating schedule, maintaining the current status of the data collection activities, determining the mileage travelled by each field member, and printing out the weekly schedule for the field personnel. They are designed to be used as a tool for future scheduling and information management activities.

First method approaches the ACR scheduling process as a database management applications. A master database containing all the necessary information for a county schedule is generated and entered into a master database file using the dBASE-III microcomputer database management software package. Data is entered into a seperate file for each District in the same sequence as they appear on the actual schedule, including the number of setups, estimated miles of travel for each schedule week. Once this master database file is setup, individual weekly schedule can be produced easily by entering the desired range of station identification numbers. Since the majority of the data collection stations remain fixed from one year to the next, only minor updates have to be made to the master database file each year to accomodate new stations or deleted stations. Any future deletion of stations or insertion of new stations are easily updated on the master database file through the dBASE-III editing and updating procedures. After the modifications are made, another new schedule can be printed out incorporating the new updated database file. Other management information such as current data collection status for the District or the list of stations that were skipped can also be readily obtained through the usage of dBASE-III query commands. It is clearly a tool designed to assist the supervisor in maintaining the overall data collection activities, as well as other in-office management duties.

The second method approaches the ACR scheduling process as a network optimization and scheduling application. The entire State ACR stations can be viewed as a network of links and nodes where each node represents one ACR station and each link represents the roadway that conntcts the ACR stations. Once the network of link-node diagram is established, there are several proven methods that can be applied to determine the optimal schedule to cover the entire network. Separate data files are setup for each county that describes the link-node diagram for that county. The original link-node relationships are determined manually once for each county. A county is then represented by multiple links or blocks of continuous paths which contains all the ACR stations for that particular county. The number of required ACR equipment setups and the distance for each link is also stored in the data file. Based on these information, the scheduling program then generates an optimal routing sequence and prints out the daily schedule. For this study, a modified Travelling Salesman algorithm was developed to produce an optimal routing schedule. The algorithm was then implemented using the Microsoft FORTRAN-77 language to generate the routing schedule, print out the daily schedule. Data entry assistance program was also developed to be aid in the data file creation and data entry process.

District 17 was used in this study to develop the database used to determine the schedules. Initially, the database was to be setup for the entire State, but due to limitations in time, efforts, and cost, only District 17 was selected for prototyping and system validation. Procedures to setup the other District data files are included in this software package which can be used to incorporate the entire State in this system. Initially, it may require much time and effort to develop the data files for the entire State, but this would be a one-time task. Once completed, the same data file can be accessed repeatedly with only minor updates to incorporate new stations.

The following sections describe in full detail both of these systems and the necessary procedures to use the programs.

I. DATABASE MANAGEMENT SYSTEM APPLICATION

Some of the most powerful applications for the microcomputer are through the use of databases. A database is simply a collection of related data. Typically, a program called a database management system (DBMS) is used to maintain the data and generate reports and other statistics. The more capable DBMS's provide programming language to create programs, query language to directly access the database files for ad hoc operations, and even utilities to generate programs.

There are numerous ways to structure a database. Perhaps the simplest, though still quite powerful, method is the "relational" database structure. They are easy to design, maintain and modify, and information retrieval can be done relatively quickly. Because of their simplicity, relational database packages have come to dominate the field for microcomputers.

For this study, dBASE-III database management system was used to develop the master database files and the programs to generate the weekly schedules. dBASE-III is a relational DBMS by Ashton-Tate that has gained wide acceptance among the IBM-PC compatible system users. It has many user-friendly yet power features that are specifically designed for use by even the novice computer users. Specifically, the user assistance program called ASSIST can be used to access the database files without any previous knowledge of the database structure nor the database command language. Many utilities are incorporated in this ASSIST to allow the user to walk-through the database files, edit or update individual fields of records, and make direct queries against the database.

MASTER DATABASE STRUCTURE DESIGN

As described previously, the ACR scheduling process is an ideal application for the database management system. Each data collection station becomes an individual record in the master database and the different fields of the record then describes each station. The master database structure design and layout is the foremost important step. The database structure must include sufficient number of fields to completely encompass all the information for each record. Detailed structure design must be done initially to avoid any major reconstruction of the database formats. Any later reconstruction of the database structure is not only time consuming but also costly since the programs are dependent on the database structure. Any redesign of the database structure implies that the depending programs must also be updated to accomodate the changes in the database structure. Therefore, the master database structure must be designed initially with great detail and rigorous validation.

Current year's ACR schedules for District 17 were obtained from Mr. Jack Crumley's office of D-10R department. The RECORDER'S DAILY WORK SHEET shown in Figure 1 was examined along with the schedule to determine the master database structure. The individual fields of the final database structure is described in Figure 2. Some of these fields are only used in the case of start of a new schedule day. These fields however are necessary during the schedule printing process which determines the end of schedule day from these fields. The main objective in this dBASE-III application system was to replicate the current ACR schedule as much as possible so that the system may be used immediately without much modification to the existing scheduling procedures. There are total 15 fields for each record in this database which include the following information.

DAY1	- First day's county location
DAY2	- Second day's county location
MILE1	- First day's estimated travel milage
MILE2	- Second day's estimated travel milage
DAY_SETUP	- Total number of machine setups for this day
STATION	- Station Identification Number
COUNTY	- County name of the station

SETUP	- Number of setups for this station
NEWDAY	- Flag used to identify a new day
ARRIVAL	- Arrival time at the station
DEPART	- Departure time at the station
LASTCOUNT	- Last year's volume count
THISCOUNT	- This year's volume count
REMARKS	- Operator's comment field
MACHINE	- Machine Identification Number

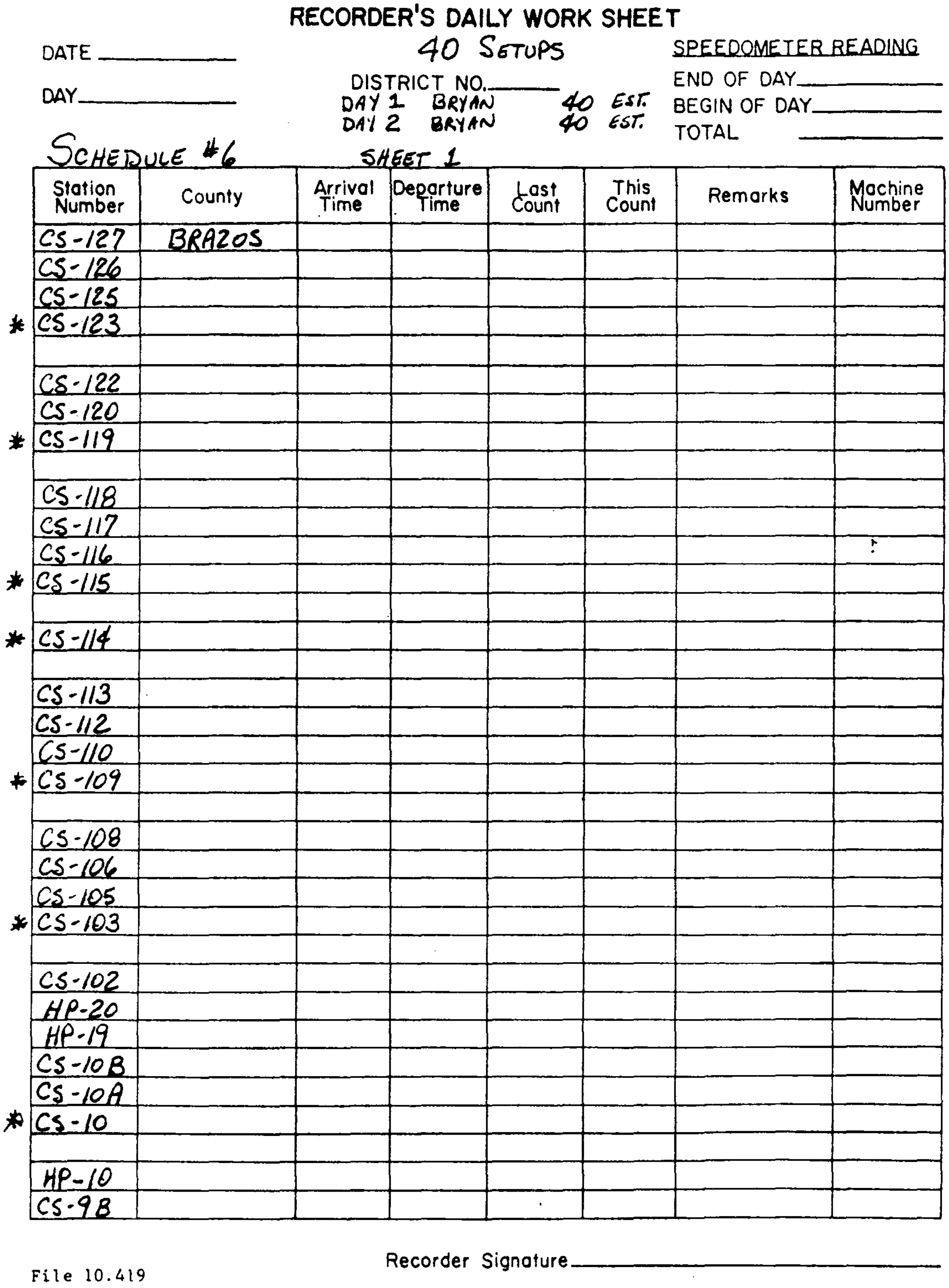
This structure was used to setup the actual database file for the District 17. The same structure will be used to setup any other District database files in the future.

ACR SCHEDULING SYSTEM DEVELOPMENT

The ACR scheduling system was designed as a menu driven system such that the user only selects an option from several levels of menu system. This is the simplest method used to design a software system for a common computer user since all the options are available through the menu selection. There are no other commands that the user needs to learn to operate this type of system.

The hierarchical system design diagram shown in Figure 3 describes the levels of system execution with each box representing a seperate program used in this integrated system. The system execution is controlled through the main driver program called **MENU.** This main menu driver in turn calls other routines in this system to create new database file, update existing file, print schedule, and determine the current status of the data collection activities. The main menu screen is shown in Figure 4.

The second step in developing the ACR scheduling system was to setup a database file for District 17. Once a new database file is setup through the **SETUP** procedure, the actual data entry process began through the **ENTRY** program. **ENTRY** is a full-screen data entry assistance program developed to aid the data entry process. The different fields of the record are entered through a full-screen editing program which allows the user to move around the screen using the arrow keys. Any of the fields may be edited while the record is still on the screen by using the arrow keys and typing over the previous



RECORDER'S DAILY WORK SHEET

DATE ____________

DAY ____________

40 SETUPS

DISTRICT NO. ______
DAY 1 BRYAN 40 EST.
DAY 2 BRYAN 40 EST.

SPEEDOMETER READING
END OF DAY ____________
BEGIN OF DAY ____________
TOTAL ____________

SCHEDULE #6

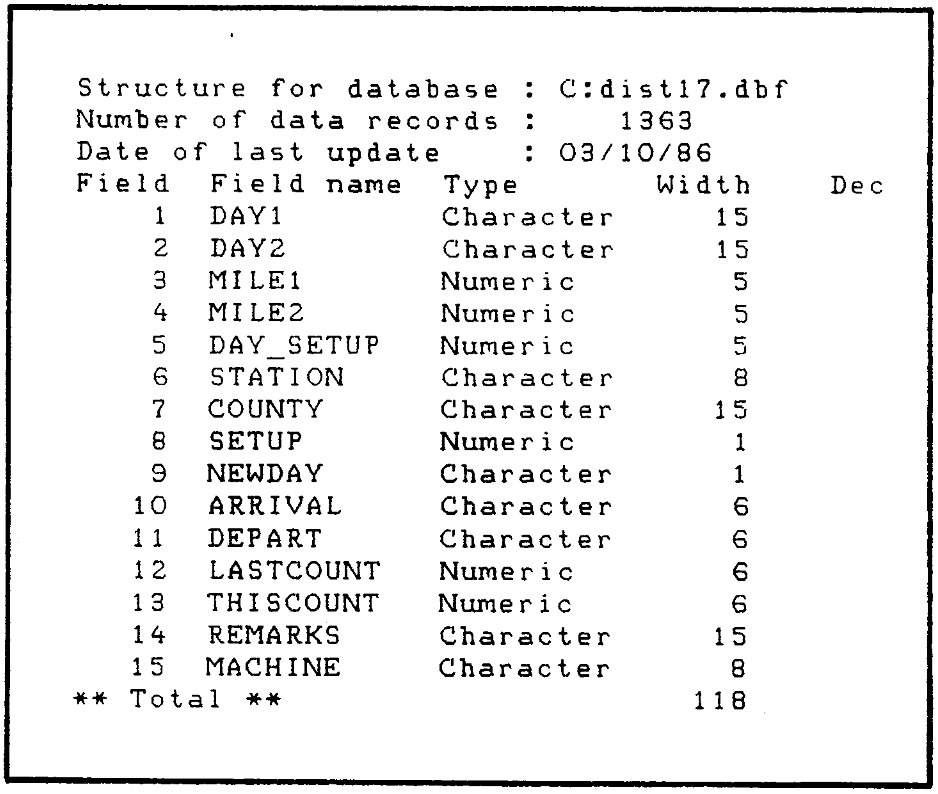
SHEET 1

Station Number	County	Arrival Time	Departure Time	Last Count	This Count	Remarks	Machine Number
CS-127	BRAZOS						
CS-126							
CS-125							
* CS-123							
CS-122							
CS-120							
* CS-119							
CS-118							
CS-117							
CS-116							
* CS-115							
* CS-114							
CS-113							
CS-112							
CS-110							
* CS-109							
CS-108							
CS-106							
CS-105							
* CS-103							
CS-102							
HP-20							
HP-19							
CS-10B							
CS-10A							
* CS-10							
HP-10							
CS-9B							

Recorder Signature ____________

File 10.419

Figure 1. RECORDER'S DAILY WORK SHEET



```
Structure for database : C:dist17.dbf
Number of data records :     1363
Date of last update    : 03/10/86
Field  Field name  Type       Width    Dec
    1  DAY1        Character     15
    2  DAY2        Character     15
    3  MILE1       Numeric        5
    4  MILE2       Numeric        5
    5  DAY_SETUP   Numeric        5
    6  STATION     Character      8
    7  COUNTY      Character     15
    8  SETUP       Numeric        1
    9  NEWDAY      Character      1
   10  ARRIVAL     Character      6
   11  DEPART      Character      6
   12  LASTCOUNT   Numeric        6
   13  THISCOUNT   Numeric        6
   14  REMARKS     Character     15
   15  MACHINE     Character      8
** Total **                     118
```

Figure 2. Master Database Structure

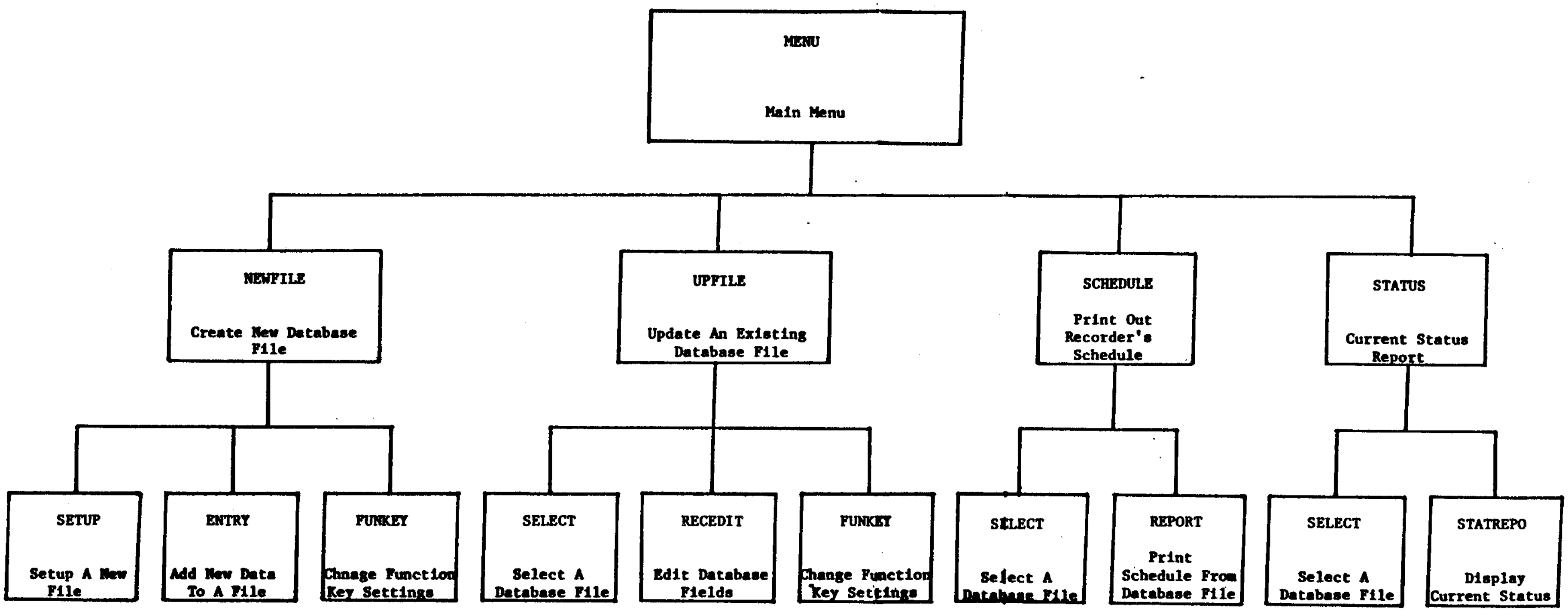
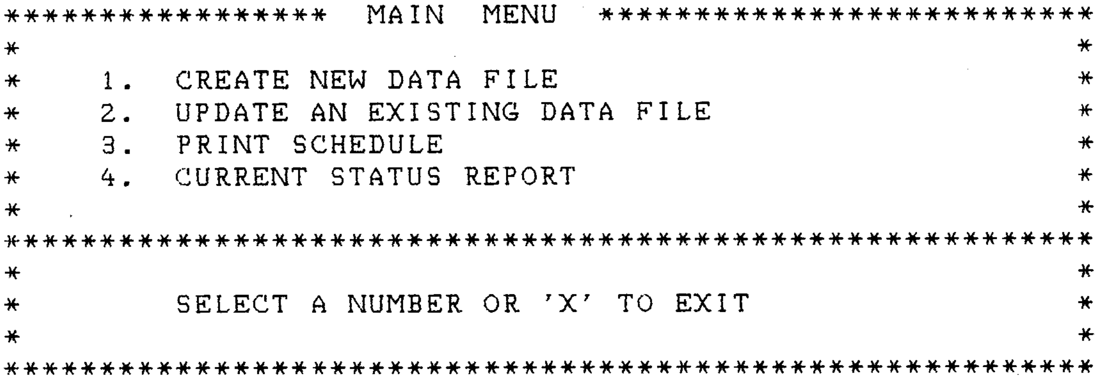


Figure 3. System Hierarchy Diagram



```
****************** MAIN  MENU  ***************************
*                                                        *
*    1.  CREATE NEW DATA FILE                            *
*    2.  UPDATE AN EXISTING DATA FILE                    *
*    3.  PRINT SCHEDULE                                  *
*    4.  CURRENT STATUS REPORT                           *
*                                                        *
**********************************************************
*                                                        *
*        SELECT A NUMBER OR 'X' TO EXIT                  *
*                                                        *
**********************************************************
```

Figure 4. Screen of MAIN Menu

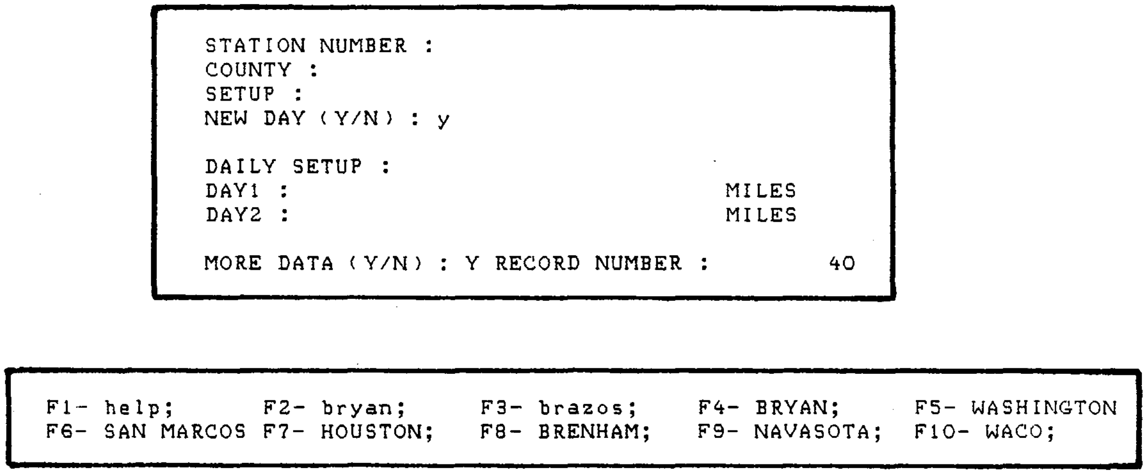
value. The function keys have been programmed to assist in the data entry by assigning the different county names to these nine function keys. Any of these function key values can be reprogrammed through the **FUNKEY** routine. This capability increases the data entry process by pressing a single key stroke for more frequently used county names. The full-screen editing screen is shown in Figure 5. This data entry screen is continued for each record until the operator enters "N" for the "MORE DATA (Y/N) :" field. The default value for this field is set to "Y" which automatically appends next record to the database file. Using this full-screen data entry program, the data for the District 17 ACR stations were entered. This same procedure can be used to setup database files for the remaining Districts in the State.

The next step in developing the ACR Scheduling system was to develop programs that generates the RECORDER'S DAILY WORK SHEET information. The program uses the database setup in the previous step to generate schedule and print out the daily schedule form. There are two options available to the schedule printing process as shown in Figure 6. First, the entire schedule for the District may be printed out at one time. It requires considerable time to print out the entire District's schedule but this may be used as a reference guide for individual weekly schedule and also for any other record keeping purposes. A sample output of the computer generated schedule form is shown in Figure 7. Second option allows the user to select the desired range of stations to be printed out. This option can be used to print out either daily or weekly schedule to be sent out to the field personnel. The user may invoke the **ASSIST** utility to determine the record numbers for the beginning and the ending record for individual schedule.

The current status determination program, **STATUS**, was developed to count the number of stations in the District where the data has been collected. This routine is used to determine the total percentage of the District that has been completed to make any future manpower utilization adjustment.

The strength of this dBASE-III ACR Scheduling system is the simplicity and ease of use. The entire program can be executed through simple menu selections at different levels of execution. It is virtually impossible to cause any catastrophical error that may destroy the database files or the programs. The user is encouraged to "play" with the system to discover the useful aspects of this system.

The source code listing of programs is included in Appendix A.



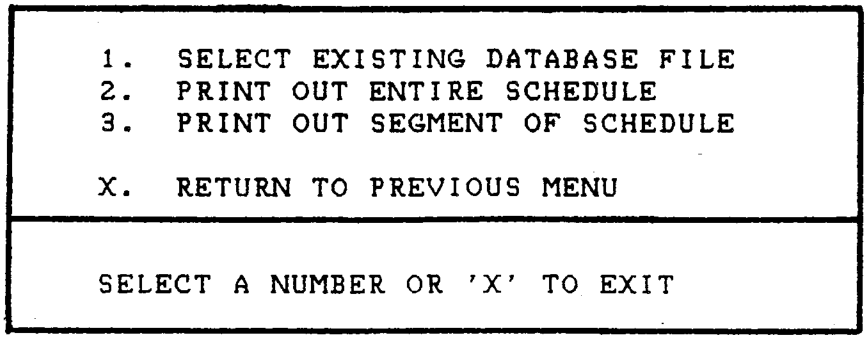
```
STATION NUMBER :
COUNTY :
SETUP :
NEW DAY (Y/N) : y

DAILY SETUP :
DAY1 :                                  MILES
DAY2 :                                  MILES

MORE DATA (Y/N) : Y RECORD NUMBER :         40
```

```
F1- help;       F2- bryan;      F3- brazos;     F4- BRYAN;      F5- WASHINGTON
F6- SAN MARCOS  F7- HOUSTON;    F8- BRENHAM;    F9- NAVASOTA;   F10- WACO;
```

Figure 5. Sample Data Entry Screen

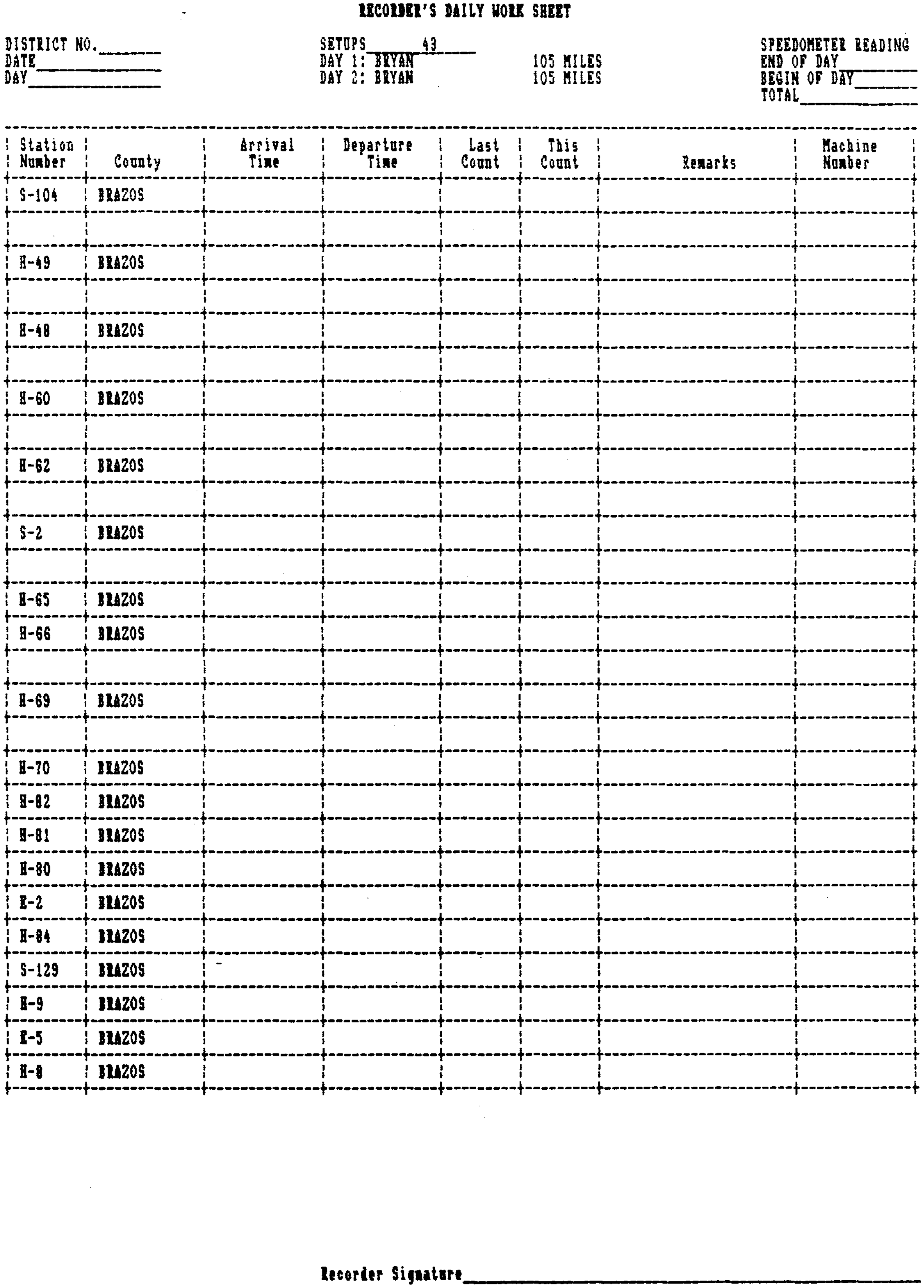


```
1.  SELECT EXISTING DATABASE FILE
2.  PRINT OUT ENTIRE SCHEDULE
3.  PRINT OUT SEGMENT OF SCHEDULE

X.  RETURN TO PREVIOUS MENU

SELECT A NUMBER OR 'X' TO EXIT
```

Figure 6. Menu for Schedule Printer



RECORDER'S DAILY WORK SHEET

DISTRICT NO.________ SETUPS 43 SPEEDOMETER READING
DATE________ DAY 1: BRYAN 105 MILES END OF DAY________
DAY________ DAY 2: BRYAN 105 MILES BEGIN OF DAY________
TOTAL________

Station Number	County	Arrival Time	Departure Time	Last Count	This Count	Remarks	Machine Number
S-104	BRAZOS						
H-49	BRAZOS						
H-48	BRAZOS						
H-60	BRAZOS						
H-62	BRAZOS						
S-2	BRAZOS						
H-65	BRAZOS						
H-66	BRAZOS						
H-69	BRAZOS						
H-70	BRAZOS						
H-82	BRAZOS						
H-81	BRAZOS						
H-80	BRAZOS						
E-2	BRAZOS						
H-84	BRAZOS						
S-129	BRAZOS						
H-9	BRAZOS						
E-5	BRAZOS						
H-8	BRAZOS						

Recorder Signature________________________________

Figure 7. Computer Generated Schedule

II. NETWORK OPTIMIZATION AND SCHEDULING APPLICATION

NETWORK SYSTEM OVERVIEW

The second methodology to implement the ACR Scheduling system is the Network Optimization procedure. The entire State can be modeled as a network system where each ACR station is represented by a node, and each link represents the network of roadways that interconnect these ACR stations. Each link is selected manually from the county maps to follow a given direction of flow. These links are delimited by major highway sigments or major roadway intersections. The sample link setup diagram of the Madison county is described in Figure 8. This manual process of setting up the link diagram for each county is very time consuming. Each circle in Figure 8 represents an ACR station and each link represents a segment of highway system in the Madison county. The number above the station indicates the number of machine setups required for that station and the number at the end of the link indicates the length of that particular link in miles. These information were abstracted from the individual ACR station county maps using the distance wheel. Setting up these link-node diagram for each county is the foremost necessary step in developing the network scheduling software.

The ACR Scheduling and Information Management System is composed of two FORTRAN Programs; Data Entry Program and Routing and Scheduling Program (See Fig. 9).

Data Entry Program gets the information about a county and creates the data base to be used by the Routing and Scheduling Program.

Routing and Scheduling Program finds the optimal routing among the blocks in a county and generates the schedules of data collection. The Routing Program is developed based on the Traveling Salesman Algorithm [1] with some modification to fit our application.

The description of subprograms of each programs and the instructions of running the programs are given below.

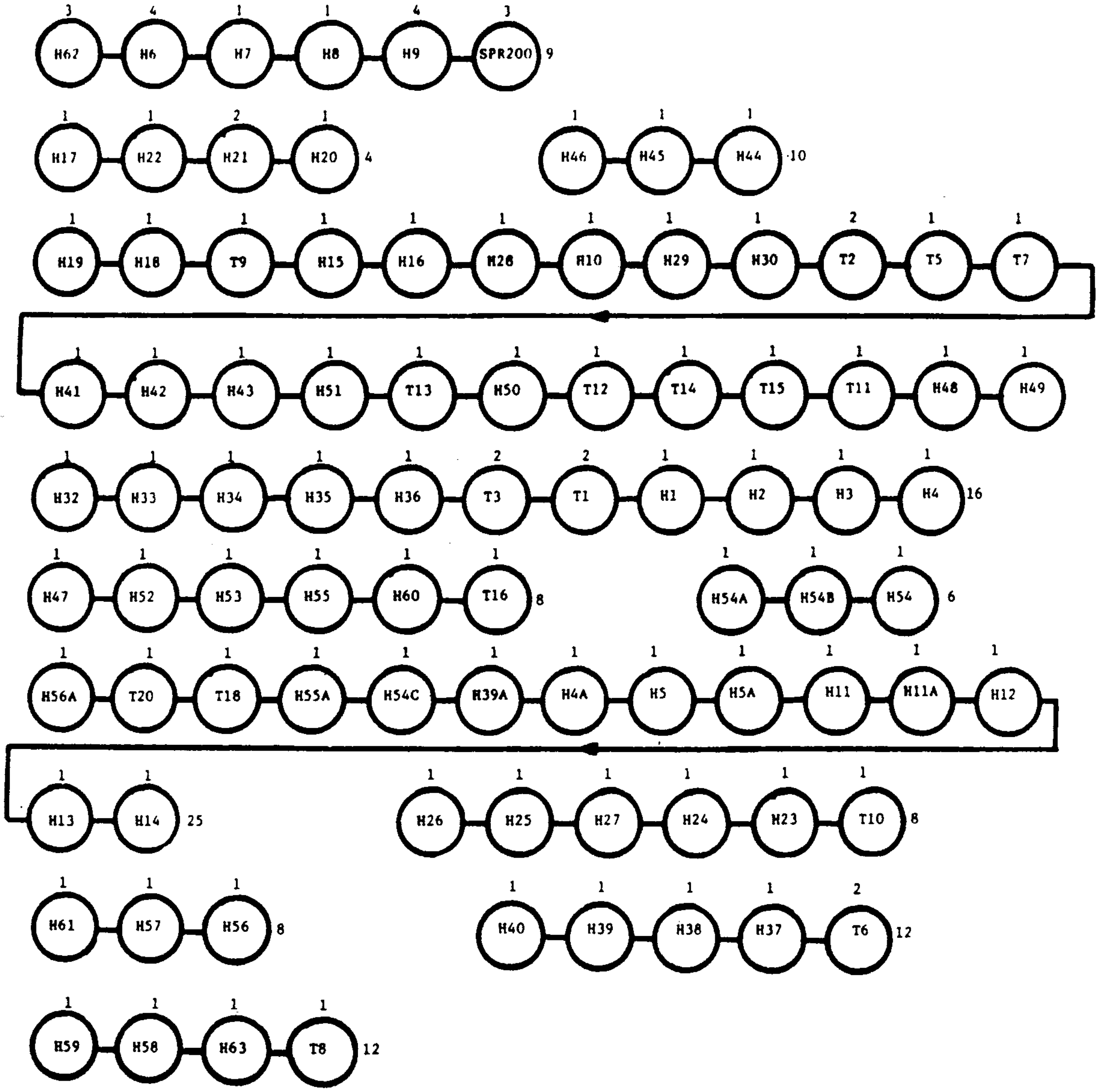


Figure 8. Sample Link Diagram For Madison Co.

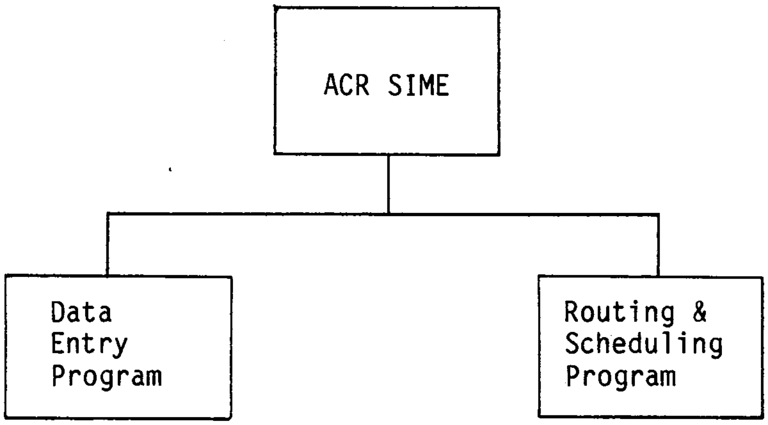


Figure 9. Hierarchy chart of ACR Scheduling and Information Management System.

PROGRAM DESCRIPTION

A. Data Entry Program

The Data Entry Program, **ENTRY** (Fig. 10), is designed to build the data base that contains the information needed to schedule the data collection in a county. There are five subroutines; GetCnst, NodeNam, GetDist, PrtMtx, Save.

1. GetCnst: The subroutine GetCnst asks the user for the constant values of a county. The items are county name, number of blocks in the county, name of the entering point to the county, and the exiting point of the county.

2. NodeNam: The subroutine NodeNam (Fig. 11) gets the informations of each blocks. These items of a block are the identification name of nodes, the number of ACR units to be assigned to each node, the total number of ACR units to be used in the block, and the length of the block.

3. GetDist: GetDist routine builds the distance matrix of the node network. The network is configured by two end nodes from each block and one dummy node which will force the start and end node to be the same to satisfy the requirement of the traveling salesman algorithm.

The resulted matrix size for a county of n blocks is (2n+1) by (2n+1). An arbitrary big number 999 is placed to the diagonal cells to represent that there is no connected way between two nodes. 999 is also placed to the cells either on the first row or column except the link to the start node and from the end node for the same reason. The distance between two end points of a block is set to zero so that the whole block should be scheduled once either one of the end nodes is selected.

4. PrtMtx: PrtMtx prints the distance matrix resulted from the GetDist. Due to the limited width of the paper, it prints up to 30 columns of matrix each time.

5. Save: Save creates two disk files. One is for the block distance and the other is for the site information. Block distance file has county name at the top to identify its contents, number of blocks in that county, start and end

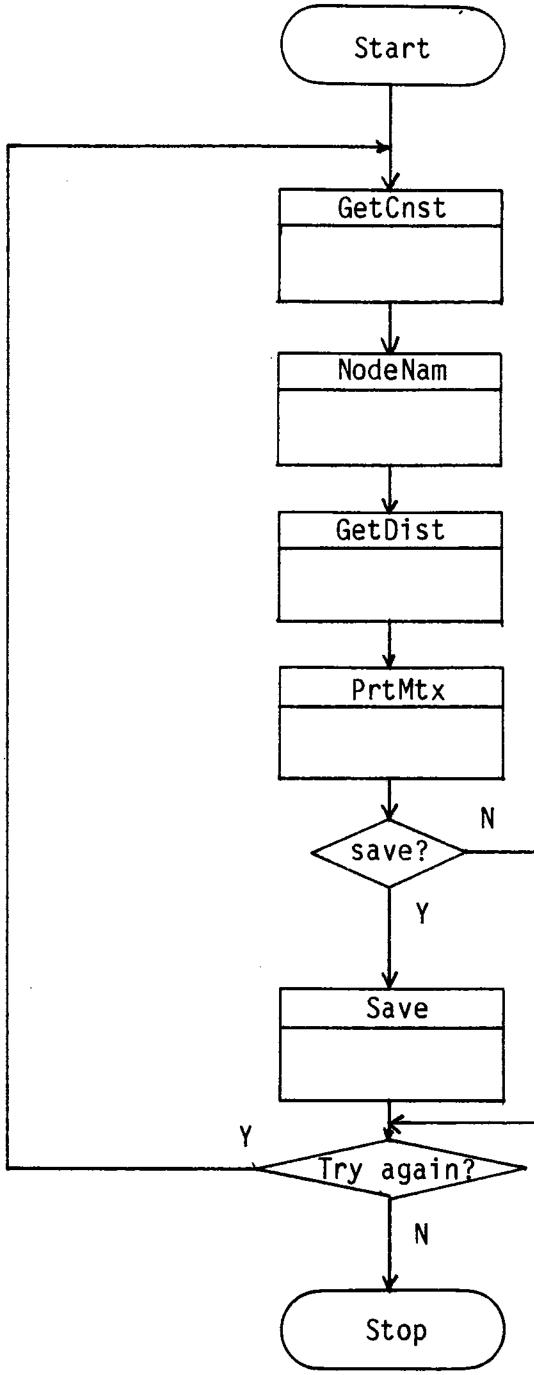


Figure 10. Flowchart of the Data Entry Program.

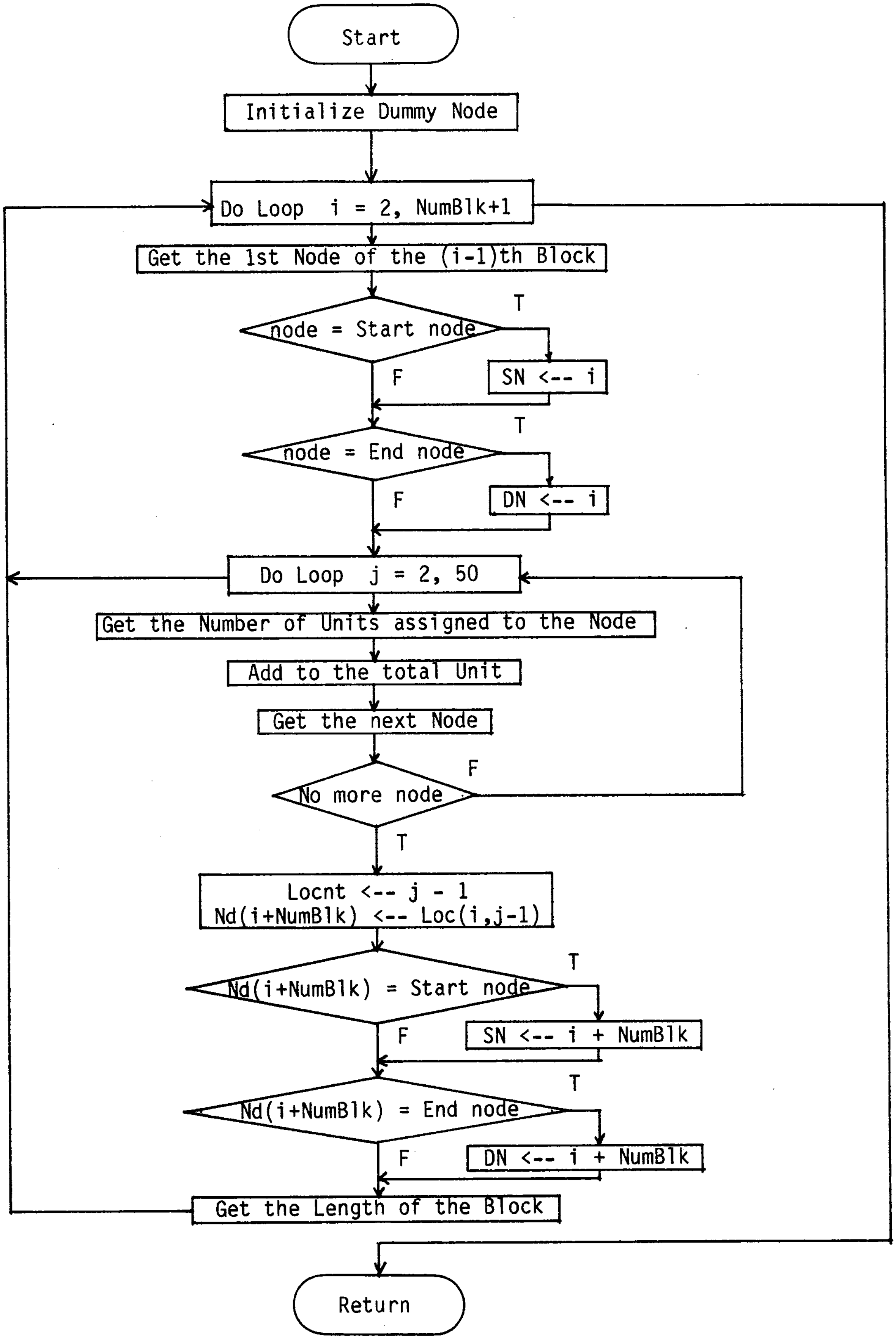


Figure 11. Flowchart of the NodeNam Routine.

nodes of the county. The distance matrix without the cells of value 999 or 0 are written next.

The site information file also identifies itself by the word 'SITE' followed by county name at the top. Then the block number, number of nodes, total number of ACR units, and the length of the block are written followed by the list of each node identification name with the number of units assigned to the node of that block.

The Save routine may be optionally called by the user after looking at the distance matrix from the main program.

B. Routing and Scheduling Program

The Routing and Scheduling Program (Fig. 12) is to find the optimal routing sequence and generate the schedule of the day by day data collection according to the optimal routes in a county. For the detailed explanation about the routing algorithm, consult the reference [1]. The modified portion will be explained in the section of REDCTN and IMRDTN routines.

The rules to break the days are 1) the maximum number of ACR units for a day is 50, 2) the daily operation can not be stopped in the middle of a block.

1. INIT: INIT routine reads the block distance file to recreate the distance matrix. The distance matrix is copied into two arrays. Another array with the same size as the distance matrix with all zeros is also created. INIT also initializes variables.

2. ROUTNG: ROUTNG routine (Fig. 13) is responsible to find the optimal routes of the given distance matrix. It applies the traveling salesman algorithm [1] with some modification in REDCTN and IMRDTN routines. It is able to reinitialize the system or to notify that the route can not be defined.

3. REDCTN: The routing algorithm reduces the size of distance matrix each time the candidate path link has been selected by eliminating that link from the matrix. The REDCTN routine (Fig. 14) is doing this matrix reduction. Since the traveling salesman algorithm used in this program is only applicable to the independent individual nodes network, we need to add some constraints so that it can recognize the existence of the path linkage of each block.

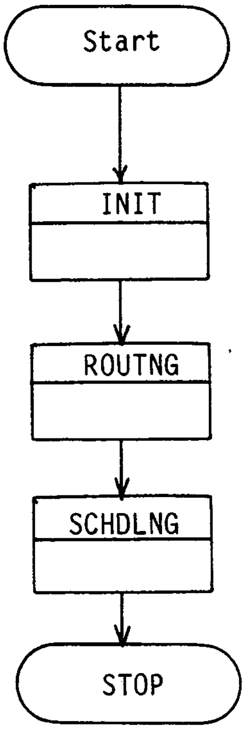


Figure 12. Flowchart of the Routing and Scheduling Program.

In other words, under the traveling salesman algorithm, any node can be selected from any one node as a next one to take while under the modified algorithm, only the other end node of that block can be selected once an end node is selected so that the routing can be arranged by block by block.

4. IMRDTN: IMRDTN is used whenever the one end of a block is selected as a candidate in a optimal routing to link it to the other end of the block automatically. By doing this, we can force a whole block can be on the routing schedule without the interference of other nodes outside the block.

5. ARCO: ARCO routine (Fig. 15) does the computations to find the best candidate path link to be added to the optimal route. It is the way to reduce the matrix by finding the candidate path link and removing these row and column from the matrix. In order to prevent the algorithm from not to go through a block once it reached to one end node of the block, some sequences of condition checking have been added.

6. REINIT: It is possible to find a better solution, so we must block the arc that was used at this level from being used in the new solution by setting the distance to infinity, which is 999 in our program. REINIT resets all initial values, and returns the control back to the start.

7. MINCOL: After the matrix is reduced its size by removing the candidate path link, columns are investigated to see if there is no zero on any column. If a column does not have a value of zero, the values on that column will be reduced by the smallest value on the column to make the column has at least one cell with the minimum value of zero. MINCOL is designed to check those columns and change their values if necessary.

8. MINROW: MINROW does the same type of operation as the MINCOL on the rows instead on the columns.

9. SCHDLNG: SCHDLNG routine generates the formatted ACR data collection schedule. The format is similar to the current form shown in Figure 1, used in highway department. SCHDLNG does not provide the space for district number, recorder's signature. SCHDLNG records the total mileage and number of

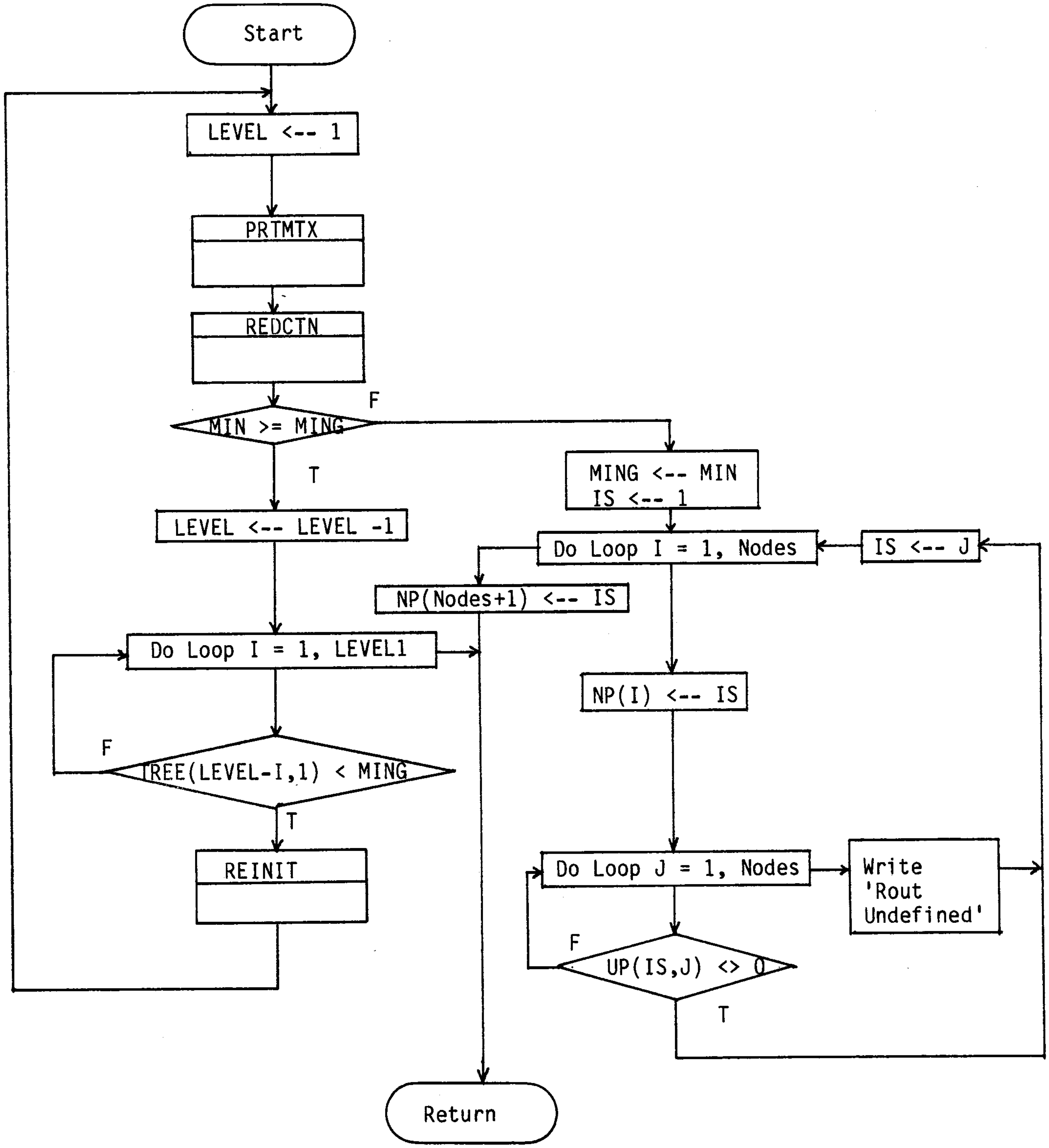


Figure 13. Flowchart of the Routing Routine.

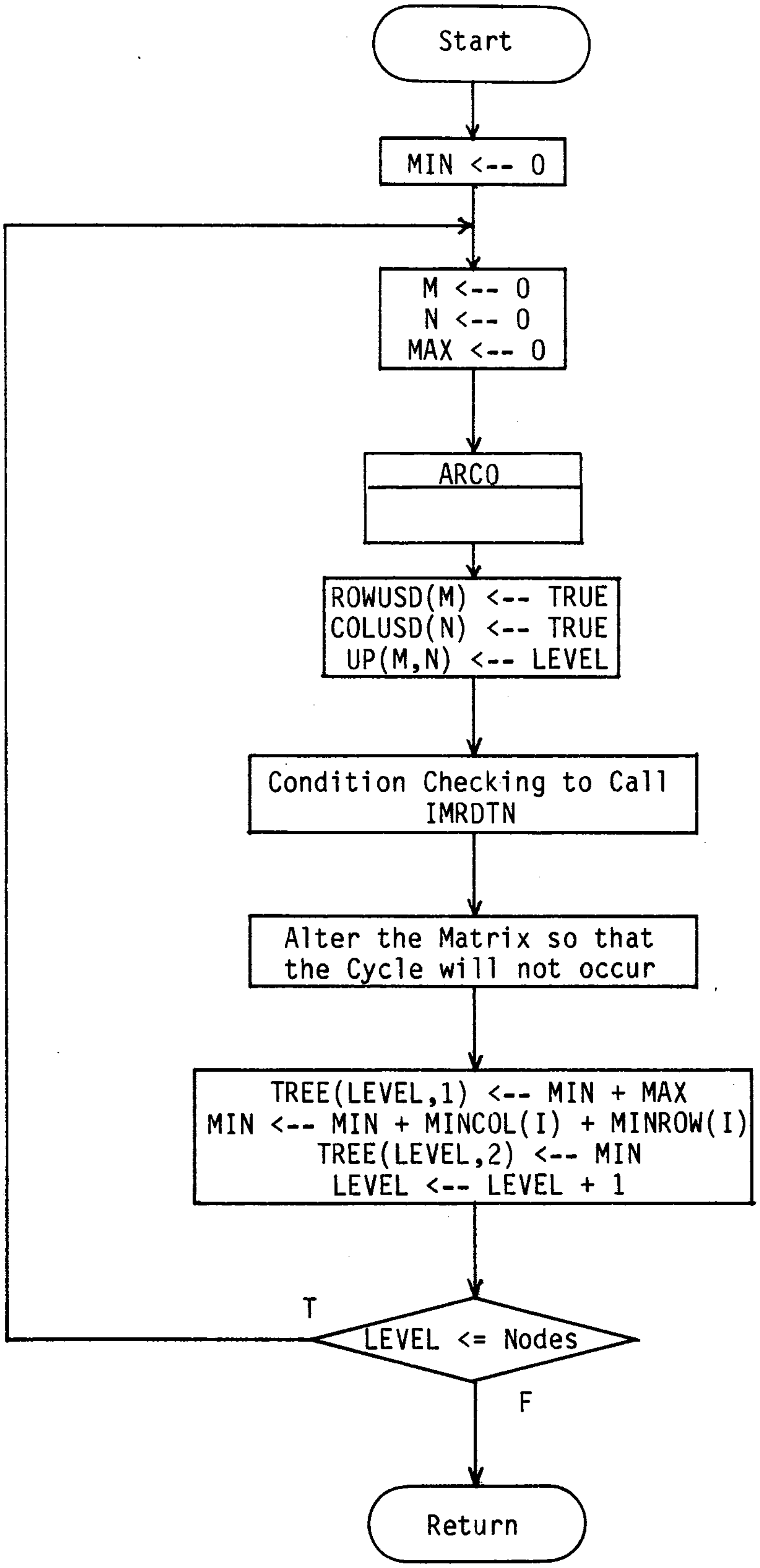


Figure 14. Flowchart of the REDCTN Routine.

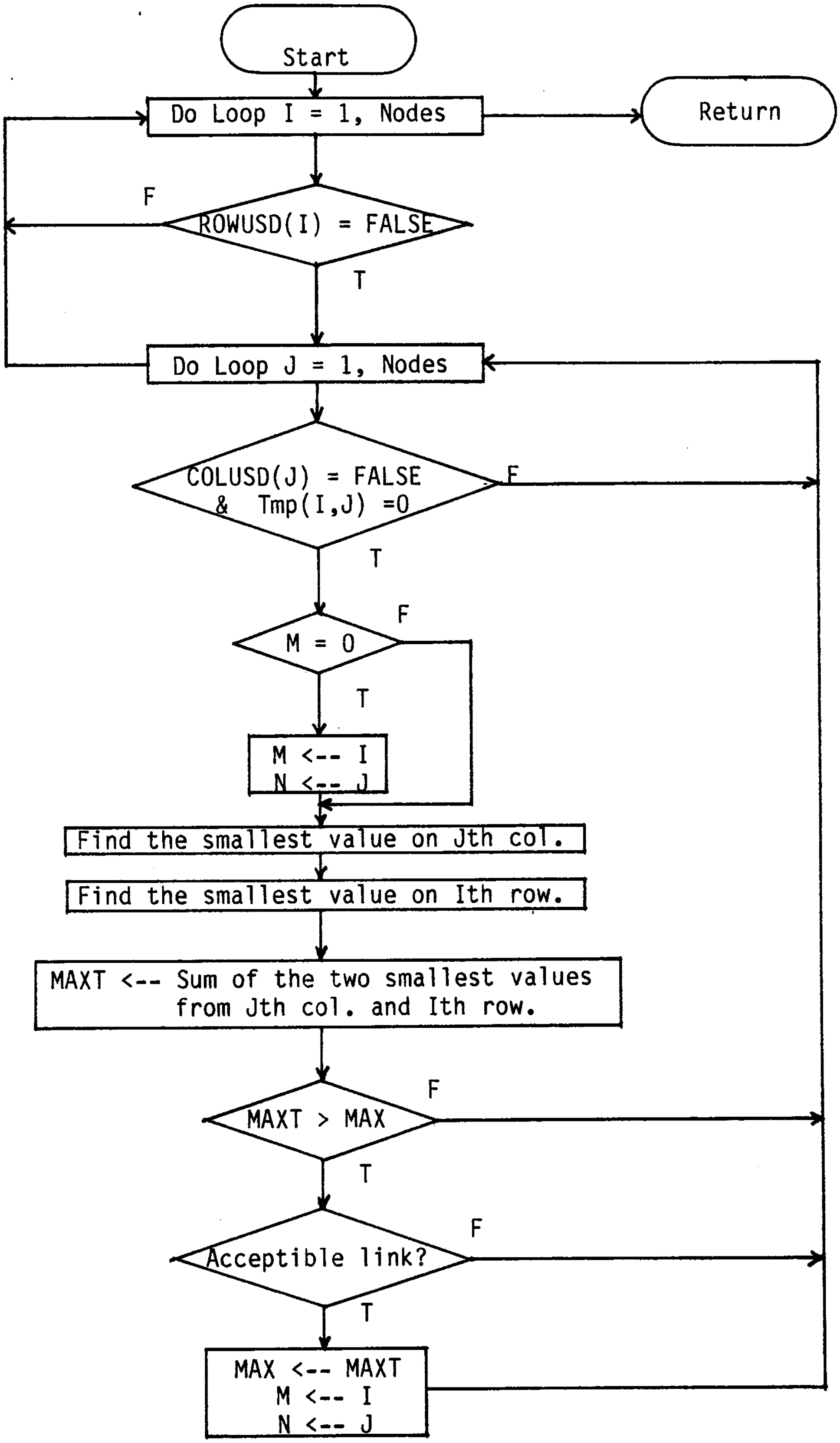


Figure 15. Flowchart of the ARCO Routine.

ACR units scheduled for a day at the end of the daily schedule.

The daily break occurs when the total number of ACR units will exceed 50 if the next block is included. Though up to 30 ACR units can be schduled on a page, the page break in the middle of a site which may have units assigned through the 30th line is started on the next page.

10. HEADING: HEADING routine resets the line count and prints the heading for the schedule on every page.

III. INSTRUCTIONS OF RUNNING THE PROGRAMS.

- **Bold** character strings are what you should type followed by pressing the return key.
- Underlined character strings are the system prompts.
- Press the Ctrl key and the PrtSc key simultaneously to start the printer to print the output on the screen at any time during the run.

A. Data Entry Program.

1. Call the Entry program.

   C>**entry**

2. Before the actual data is input, four questions must be answered.

   Enter the name of County: **xxxx**

   Here, xxxx is the full name of the county.

   Enter the number of blocks in the county: **b**

   The integer b must be between 1 and 25, and it represents the number of links from the link-node diagram of the county.

   Enter the name of the Start node: **S**

   S represents the name of the very first node of the county. This will be the first node of the diagram, or the first node on the matrix form.

   Enter the name of the End node: **E**

   E is the name of the very last node of the county, which is the last node on the last link. This is also the last node on the matrix form.

3. This is where the data entry actually begins.

   The data for the first part of this procedure is taken from the link-node diagram.

Enter the first node of block 1: **n1**
How many units are assigned: **atr 1**
Enter the next node: **n2**

⋮

n1, n2 are node names. They are taken in the order that they appear on the diagram. The number of units are above each node on the diagram. After an entire link has been entered, just press RETURN without typing anything to the third prompt for next node.

Enter the length of the block: **L**

L is the length of the block and it appears after the last node of the link in the diagram.

The next prompt will ask for data for the next block. Continue the above procedure for all of the remaining blocks.

4. The entering of the block distance matrix is much easier. Just enter the value going across the matrix in answer to the prompts:

Distance from x1 to x2? **L1**
Distance from x1 to x3? **L2**

⋮

L1 and L2 represent the distances in miles.
Value must be a whole number.
Do not enter the infinity diagonal values.
After the last distance is keyed in, the distance matrix will be shown to you.

5. Now it is time to save the data.

Would you like to SAVE it (y/n)?

Answer y if the distance matrix has been produced correctly.

Enter the file name for block distances: **d:name.blk**

d is the disk driver name where the data will be saved in.

name is the abreviated county name with up to 8 letters.

.blk is the required extension.

Enter the file name for site information: **d:name.st**

d and name are exactly the same as above.

.st is the required extension.

6. Would you like to try again (y/n)?

Here just type answer whether or not you would like to enter data for another. If so, repeat the previous steps but if not, the program will end.

B. The **ROUT** Program.

1. Set the printer to compressed mode:

   mode lpt1:132,8

2. Execute the program ROUT:

   rout

3. Enter the file name of block distance: **d:name.blk**

4. The system will read the data then generate the distance matrix.

   It takes a few seconds to find the optimal route.

   If the optimal route is found:

   Enter the file name of site information: **d:name.st**

5. Enter the file name of site list: **d:name.dt**

   This file is to be used for maintenance and statistics in the future.

6. The system will start to generate the work schedule.

REFERENCE

Phillips, Don T., and Alberto Garcia-Diaz. Fundamentals of Network Analysis. Englewood Cliffs, NJ: Prentice-Hall, Inc. 1981.

APPENDIX A
PROGRAM LISTING OF dBASE-III SYSTEM

```
************************************************************
*                                                          *
*                       MENU.PRG                           *
*                                                          *
*  THIS PROGRAM IS THE CONTROL MENU DRIVER FOR THE ACR     *
*  SCHEDULING PROCEDURES.  THE USER SELECTS THE DESIRED    *
*  FUNCTION AND THE PROGRAM CONTROL IS PASSED TO THE       *
*  SELECTED PROGRAM SEGMENT.  AT THE COMPLETION OF THE     *
*  SELECTED FUNCTION, THE PROGRAM RETURNS TO THIS MAIN     *
*  MENU FOR FURTHER PROCESSING.                            *
*                                                          *
************************************************************
SET TALK OFF
CLEAR
DO WHILE .T.
*
CLEAR
@ 5,9 SAY "*****************  MAIN  MENU  ************************"
@ 5,64 SAY "**"
@ 6,9 SAY "*"
@ 6,65 SAY "*"
@ 7,9 SAY "*    1.  CREATE NEW DATA FILE"
@ 7,65 SAY "*"
@ 8,9 SAY "*    2.  UPDATE AN EXISTING DATA FILE"
@ 8,65 SAY "*"
@ 9,9 SAY "*    3.  PRINT SCHEDULE"
@ 9,65 SAY "*"
@ 10,9 SAY "*    4.  CURRENT STATUS REPORT"
@ 10,65 SAY "*"
@ 11,9 SAY "*"
@ 11,65 SAY "*"
@ 12,9 SAY "********************************************************"
@ 12,64 SAY "**"
@ 13,9 SAY "*"
@ 13,65 SAY "*"
@ 14,9 SAY "*        SELECT A NUMBER OR 'X' TO EXIT"
@ 14,65 SAY "*"
@ 15,9 SAY "*"
@ 15,65 SAY "*"
@ 16,9 SAY "********************************************************"
@ 16,64 SAY "**"
*
*
WAIT ' ' TO CHOICE
*
*  BRANCH TO PROPER PROGRAM SEGMENT DEPENDING ON THE CHOICE
*
DO CASE CHOICE

   CASE CHOICE = '1'
      DO NEWFILE
   CASE CHOICE = '2'
      DO UPFILE
   CASE CHOICE = '3'
      DO SCHEDULE
   CASE CHOICE = '4'
      DO STATUS

   CASE UPPER(CHOICE) = 'X'
      ? "           RETURNING TO DBASE-III ENVIRONMENT"
      ? "           ENTER 'QUIT' TO EXIT TO MS-DOS"
      DO RESETKEY
      CLOSE DATABASES
      RETURN
*
ENDCASE CHOICE
ENDDO     /* MENU.PRG */
*
************************************************************
*                                                          *
*                   NEWFILE.PRG                            *
*                                                          *
*                                                          *
* MENU DRIVER TO CREATE A NEW DATABASE FILE                *
*                                                          *
************************************************************

CLEAR
 DO WHILE .T.
 DO DISPFUNC
*
* DISPLAY MENU SCREEN
*
    @ 2,12 SAY "__________________________________________________"
    @ 3,11 SAY "¦                                                  ¦"
    @ 4,11 SAY "¦   1.  SETUP NEW DATABASE STRUCTURE               ¦"
    @ 5,11 SAY "¦   2.  CHANGE FUNCTION KEY SETTINGS               ¦"
    @ 6,11 SAY "¦   3.  BEGIN DATA ENTRY                           ¦"
    @ 7,11 SAY "¦                                                  ¦"
    @ 8,11 SAY "¦   X.  EXIT TO PREVIOUS MENU                      ¦"
    @ 9,11 SAY "¦                                                  ¦"
    @ 10,11 SAY "¦__________________________________________________¦"
    @ 11,11 SAY "¦                                                  ¦"
    @ 12,11 SAY "¦       SELECT A NUMBER OR 'X' TO EXIT             ¦"
    @ 13,11 SAY "¦                                                  ¦"
    @ 14,11 SAY "¦__________________________________________________¦"
*  WAIT FOR THE CHOICE AND BRANCH
*
   WAIT ' ' TO CHOICE
*
   DO CASE CHOICE
      CASE CHOICE='1'
        DO SETUP

      CASE CHOICE='2'
        DO FUNKEY
*
      CASE CHOICE='3'
        DO ENTRY
*
```

```
      CASE UPPER(CHOICE)='X'
        CLEAR
        RETURN
   ENDCASE
 ENDDO
************************************************************
*                                                          *
*                       SETUP.PRG                          *
*                                                          *
*  PROCEDURE TO SETUP A NEW DATABASE STRUCTURE FILE        *
*  CALLED FROM THE MAIN MENU AND USES THE SKELETON         *
*  FILE TO COPY THE STRUCTURE FOR THE NEW FILE             *
*                                                          *
************************************************************

CLEAR

   STORE "             " TO NEWFILE
   @ 5,10 SAY "Enter the new database file name: " GET NEWFILE
   READ

   @ 10,10 SAY "Creating new database file structure....."

   USE SKELETON
   COPY STRUCTURE TO &NEWFILE
   USE &NEWFILE
   CLEAR
RETURN
*************************************************************
*                                                           *
*                   ENTRY.PRG                               *
*                                                           *
*  PROGRAM TO ASSIST DATA ENTRY FOR A NEW DATABASE FILE     *
*  DISPLAYS THE FULL-SCREEN EDITING TEMPLATE FOR THE DATA   *
*  ENTRY PROCEDURE                                          *
*                                                           *
*************************************************************
  CLEAR
  DO DISPFUNC

  SET ESCAPE ON
  SET TALK OFF

*  INITIALIZE VARIABLES USED IN THE PROGRAM
*
  STORE '                                           ' TO BLANKS
  STORE 'Y' TO MORE
*
* DISPLAY THE FULL-SCREEN DATA ENTRY TEMPLATE
*
```

```
        @ 3,12 SAY "__________________________________________________"
        @ 4,11 SAY "¦                                                  ¦"
        @ 5,11 SAY "¦   STATION NUMBER :"
        @ 5,62 SAY "¦"
        @ 6,11 SAY "¦   COUNTY :"
        @ 6,62 SAY "¦"
        @ 7,11 SAY "¦   SETUP :"
        @ 7,62 SAY "¦"
        @ 8,11 SAY "¦   NEW DAY (Y/N) :"
        @ 8,62 SAY "¦"
        @ 9,11 SAY "¦                                                  ¦"
        @ 10,11 SAY "¦"
        @ 10,62 SAY "¦"
        @ 11,11 SAY "¦"
        @ 11,62 SAY "¦"
        @ 12,11 SAY "¦"
        @ 12,62 SAY "¦"
        @ 13,11 SAY "¦                                                  ¦"
        @ 14,11 SAY "¦   MORE DATA (Y/N) :"
        @ 14,35 SAY "RECORD NUMBER :"
        @ 14,62 SAY "¦"
        @ 15,11 SAY "¦__________________________________________________¦"

*  CONTINOUS LOOP UNTIL THE END OF DATA ENTRY
*
   DO WHILE (UPPER(MORE)='Y')

      STORE 'N' TO XNEWDAY
      APPEND BLANK

        @ 14,50 SAY RECNO( )
        @ 5,32 GET STATION
        @ 6,24 GET COUNTY
        @ 7,23 GET SETUP
        @ 8,31 GET XNEWDAY
        READ

        IF (UPPER(XNEWDAY)='Y')

           @ 10,15 SAY "DAILY SETUP :"
           @ 10,29 GET DAY_SETUP
           @ 11,15 SAY "DAY1 :"
           @ 11,22 GET DAY1
           @ 11,51 SAY "MILES"
           @ 11,42 GET MILE1
           @ 12,15 SAY "DAY2 :"
           @ 12,22 GET DAY2
           @ 12,51 SAY "MILES"
           @ 12,42 GET MILE2

           READ
        ENDIF

        @ 14,33 GET MORE
        READ
```

```
*
     REPLACE NEWDAY WITH XNEWDAY
     SKIP
*
*  ERASE THE MIDDLE PORTION OF THE TEMPLATE
*
        @ 10,15 SAY BLANKS
        @ 11,15 SAY BLANKS
        @ 12,15 SAY BLANKS

   ENDDO
RETURN
************************************************************
*                                                          *
*                       FUNKEY.PRG                         *
*                                                          *
*  PROGRAM TO CHANGE THE FUNCTION KEY SETTINGS             *
*  FUNCTION KEYS F2-F9 ARE REPROGAMMABLE TO HOLD ANY       *
*  CHARACTER STRING.                                       *
*                                                          *
************************************************************
*
CLEAR
SET TALK OFF
RESTORE FROM FUNKEY.MEM ADDITIVE
STORE 'N' TO YESNO
*
  DO WHILE (UPPER(YESNO) ='N')

     @ 5,10 SAY "F1-  &FKEY1"
     @ 6,10 SAY "F2- " GET FKEY2 PICTURE 'XXXXXXXXXXXXXXX'
     @ 7,10 SAY "F3- " GET FKEY3 PICTURE 'XXXXXXXXXXXXXXX'
     @ 8,10 SAY "F4- " GET FKEY4 PICTURE 'XXXXXXXXXXXXXXX'
     @ 9,10 SAY "F5- " GET FKEY5 PICTURE 'XXXXXXXXXXXXXXX'
     @ 5,40 SAY "F6- " GET FKEY6 PICTURE 'XXXXXXXXXXXXXXX'
     @ 6,40 SAY "F7- " GET FKEY7 PICTURE 'XXXXXXXXXXXXXXX'
     @ 7,40 SAY "F8- " GET FKEY8 PICTURE 'XXXXXXXXXXXXXXX'
     @ 8,40 SAY "F9- " GET FKEY9 PICTURE 'XXXXXXXXXXXXXXX'
     @ 9,40 SAY "F10- " GET FKEY10 PICTURE 'XXXXXXXXXXXXXXX'
     @ 11,10 SAY "ALL VALUES OK (Y/N)? " GET YESNO
    READ

 ENDDO
*
* RESET THE FUNCTION KEY SETTINGS
*
     SET FUNCTION 2 TO TRIM('&FKEY2')
     SET FUNCTION 3 TO TRIM('&FKEY3')
     SET FUNCTION 4 TO TRIM('&FKEY4')
     SET FUNCTION 5 TO TRIM('&FKEY5')
     SET FUNCTION 6 TO TRIM('&FKEY6')
     SET FUNCTION 7 TO TRIM('&FKEY7')
     SET FUNCTION 8 TO TRIM('&FKEY8')
     SET FUNCTION 9 TO TRIM('&FKEY9')
     SET FUNCTION 10 TO TRIM('&FKEY10')
```

```
*
*  SAVE FKEY VALUES TO FUNKEY.MEM FILE
*
    SAVE TO FUNKEY.MEM
    CLEAR
RETURN
*
************************************************************
*                                                          *
*                       UPFILE.PRG                         *
*                                                          *
*   MENU DRIVER TO UPDATE EXISTING DATABASE FILES          *
*                                                          *
************************************************************
   CLEAR
*  CONTINUOUS LOOP OR EXIT TO PREVIOUS MENU
*
   DO WHILE .T.

     DO DISPFUNC
*
*  DISPLAY MENU SCREEN
*
        @ 4,16 SAY "____________________________________________"
        @ 5,15 SAY "¦                                            ¦"
        @ 6,15 SAY "¦     1.  SELECT EXISTING DATABASE FILE      ¦"
        @ 7,15 SAY "¦     2.  UPDATE FIELDS IN SELECTED FILE     ¦"
        @ 8,15 SAY "¦     3.  CHANGE FUNCTION KEY SETTINGS       ¦"
        @ 9,15 SAY "¦                                            ¦"
        @ 10,15 SAY "¦     X.  RETURN TO PREVIOUS MENU            ¦"
        @ 11,15 SAY "¦--------------------------------------------¦"
        @ 12,15 SAY "¦                                            ¦"
        @ 13,15 SAY "¦     SELECT A NUMBER OR 'X' TO EXIT         ¦"
        @ 14,15 SAY "¦____________________________________________¦"
*
*  WAIT FOR USER SELECTION
*
    WAIT ' ' TO CHOICE

        DO CASE CHOICE

          CASE CHOICE='1'
                DO SELECT
                * SELECT AN EXISTING FILE

           CASE CHOICE='2'
                DO RECEDIT
                * UPDATE A RECORD

           CASE CHOICE='3'
                DO FUNKEY
                * CHANGE FUNCTION KEY SETTINGS
           CASE UPPER( CHOICE )='X'
                CLEAR
                * IF 'X' THEN RETURN TO PREVIOUS MENU
                RETURN
        ENDCASE


  ENDDO

*
************************************************************
*                                                          *
*                        SELECT.PRG                        *
*                                                          *
*  PROCEDURE TO SELECT AN EXISTING DATABASE FILE           *
*  OBTAINS THE NEW DATABASE FILE NAME FROM THE USER        *
*  CHECK THE DIRECTORY FOR EXISTING FILE NAMES TO AVOID    *
*  DUPLICATE FILE NAMES OR OVERWRITING THE PREVIOUS FILE   *
*                                                          *
************************************************************

   DO WHILE .T.

        CLEAR
        STORE "               " TO NEWFILE
        @ 5,10 SAY "Enter the new database file name: " GET NEWFILE
        READ

        CLOSE DATABASES
        IF ( FILE( '&NEWFILE' ) )
        * CHECK TO SEE IF THE FILE EXISTS
           EXIT
          ELSE
            @ 7,10 SAY "FILE DOES NOT EXIST"
            DIR
            WAIT
        ENDIF

    ENDDO

        USE &NEWFILE
        CLEAR
RETURN










*
```

```
************************************************************
*                                                          *
*                        RECEDIT.PRG                       *
*                                                          *
*  PROCEDURE TO ASSIST IN FULL-SCREEN RECORD EDITING       *
*  OF THE EXISTING DATABASE FILES                          *
*  EACH FIELDSOF THE ACTIVE DATABASE ARE DISPLAYED ON      *
*  THE SCREEN FOR USER VIEWING.  ANY OF THESE FIELDS       *
*  CAN BE MODIFIED BY ENTEING THE NEW VALUES ON THE SCREEN *
*                                                          *
************************************************************

  CLEAR
   SET TALK OFF
   STORE 'N' TO YESNO

*
   @ 5,7 SAY "Do you wish to use ASSIST to locate a record ? (Y/N)" GET YESNO
   READ
*
* IF 'YES' THEN CALL "ASSIST" ELSE DO FULL-SCREEN EDITING
*
     IF (UPPER(YESNO)='Y')
        ASSIST
      ENDIF
*
*  POSITION TO STARTING RECORD AND START FSE
*

     CLEAR
*
*  DISPLAY FUNCTION KEYS
*
     DO DISPFUNC
     STORE 'Y' TO NEXT
*
*  CONTINUS LOOP AND RECORD EDITING
*
     DO WHILE ( .NOT. EOF( )  .AND. UPPER(NEXT)='Y')

        * CREATE TEMPORARY VARIABLES FOR FSE

        STORE DAY1 TO XDAY1
        STORE DAY2 TO XDAY2
        STORE MILE1 TO XMILE1
        STORE MILE2 TO XMILE2
        STORE DAY_SETUP TO XDAY_SETUP
        STORE STATION TO XSTATION
        STORE COUNTY TO XCOUNTY
        STORE NEWDAY TO XNEWDAY
        STORE SETUP TO XSETUP
        STORE NEWDAY TO XNEWDAY
        STORE ARRIVAL TO XARRIVAL
        STORE DEPART TO XDEPART
        STORE LASTCOUNT TO XLASTCOUNT
```

```
         STORE THISCOUNT TO XTHISCOUNT
         STORE REMARKS TO XREMARKS
         STORE MACHINE TO XMACHINE

*   DISPLAY EDITING TEMPLATE AND READ NEW VALUES
*


        @ 2,9 SAY "_______________________________________________________"
        @ 2,64 SAY "__"
        @ 3,8 SAY "¦"
        @ 3,66 SAY "¦"
        @ 4,8 SAY "¦    DAILY SETUP :"
        @ 4,27 GET XDAY_SETUP
        @ 4,45 SAY "NEW DAY : "
        @ 4,55 GET XNEWDAY
        @ 4,66 SAY "¦"
        @ 5,8 SAY "¦    DAY 1 :"
        @ 5,21 GET XDAY1
        @ 5,44 GET XMILE1
        @ 5,56 SAY "MILES     ¦"
        @ 6,8 SAY "¦    DAY 2 :"
        @ 6,21 GET XDAY2
        @ 6,44 GET XMILE2
        @ 6,56 SAY "MILES     ¦"
        @ 7,8 SAY "¦"
        @ 7,66 SAY "¦"
        @ 8,8 SAY "¦    STATION NUMBER :"
        @ 8,30 GET XSTATION
        @ 8,66 SAY "¦"
        @ 9,8 SAY "¦    COUNTY :"
        @ 9,22 GET XCOUNTY
        @ 9,43 SAY "SETUP :"
        @ 9,51 GET XSETUP
        @ 9,66 SAY "¦"
        @ 10,8 SAY "¦"
        @ 10,66 SAY "¦"
        @ 11,8 SAY "¦    ARRIVAL :"
        @ 11,23 GET XARRIVAL
        @ 11,43 SAY "DEPART :"
        @ 11,52 GET XDEPART
        @ 11,66 SAY "¦"
        @ 12,8 SAY "¦    LAST COUNT :"
        @ 12,26 GET XLASTCOUNT
        @ 12,43 SAY "THISCOUNT :"
        @ 12,55 GET XTHISCOUNT
        @ 12,66 SAY "¦"
        @ 13,8 SAY "¦    MACHINE NUMBER :"
        @ 13,30 GET XMACHINE
        @ 13,66 SAY "¦"
        @ 14,8 SAY "¦"
        @ 14,66 SAY "¦"
        @ 15,8 SAY "¦    NEXT RECORD :"
        @ 15,27 GET NEXT
        @ 15,40 SAY "RECORD NUMBER : "
```

```
       @ 15,55 SAY RECNO( )
       @ 15,66 SAY "¦"
       @ 16,8 SAY "¦______________________________________________________"
       @ 16,63 SAY "___¦"

       READ

*  UPDATE RECORD WITH NEW VALUES
*


        REPLACE DAY1 WITH XDAY1
        REPLACE DAY2 WITH XDAY2
        REPLACE MILE1 WITH XMILE1
        REPLACE MILE2 WITH XMILE2
        REPLACE DAY_SETUP WITH XDAY_SETUP
        REPLACE STATION WITH XSTATION
        REPLACE COUNTY WITH XCOUNTY
        REPLACE NEWDAY WITH XNEWDAY
        REPLACE SETUP WITH XSETUP
        REPLACE NEWDAY WITH XNEWDAY
        REPLACE ARRIVAL WITH XARRIVAL
        REPLACE DEPART WITH XDEPART
        REPLACE LASTCOUNT WITH XLASTCOUNT
        REPLACE THISCOUNT WITH XTHISCOUNT
        REPLACE REMARKS WITH XREMARKS
        REPLACE MACHINE WITH XMACHINE

        SKIP

    ENDDO

RETURN
*************************************************************
*                                                           *
*                       SCHEDULE.PRG                        *
*                                                           *
*  MENU DRIVER TO PRINT OUT RECORDER'S DAILY SCHEDULE       *
*  OPTIONS ARE TO PRINT OUT THE ENTIRE SCHEDULE FOR ONE     *
*  DISTRICT OR SELECT ONLY A PORTION TO BE PRINTED.         *
*                                                           *
*************************************************************
*

CLEAR

*  SELECT AN ITEM  OR EXIT TO PREVIOUS MENU
*

DO WHILE .T.

* DISPLAY MENU SCREEN
*

@ 6,16 SAY "___________________________________________"
@ 7,15 SAY "¦                                           ¦"
@ 8,15 SAY "¦    1.  SELECT EXISTING DATABASE FILE      ¦"
@ 9,15 SAY "¦    2.  PRINT OUT ENTIRE SCHEDULE          ¦"
@ 10,15 SAY "¦    3.  PRINT OUT SEGMENT OF SCHEDULE      ¦"
@ 11,15 SAY "¦                                           ¦"
@ 12,15 SAY "¦    X.  RETURN TO PREVIOUS MENU            ¦"
@ 13,15 SAY "¦ -----------------------------------------¦"
@ 14,15 SAY "¦                                           ¦"
@ 15,15 SAY "¦    SELECT A NUMBER OR 'X' TO EXIT         ¦"
@ 16,15 SAY "¦___________________________________________¦"
*
*  WAIT FOR USER SELECTION
*
WAIT ' ' TO CHOICE

        DO CASE CHOICE

                CASE CHOICE='1'
                        DO SELECT
                        * SELECT AN EXISTING FILE

                CASE CHOICE='2'
                        DO REPORT
                        *  PRINT OUT ENTIRE SCHEDULE TO PRINTER

                CASE CHOICE='3'
                        DO SEGMENT
                        *  PRINT OUT SEGMENT OF SCHEDULE TO PRINTER

                CASE UPPER( CHOICE )='X'
                        CLEAR
                        *  RETURN TO PREVIOUS MENU
                        RETURN

        ENDCASE

ENDDO

************************************************************
*                                                          *
*                          REPORT.PRG                      *
*                                                          *
*  THIS PROGRAM WILL GENERATE THE DAILY RECORDER'S REPORT  *
*  THIS IS THE MAIN WORK HORSE PROGRAM THAT GENERATES THE  *
*  DAILY SCHEDULE FOR THE DATA COLLECTION FIELD CREW.      *
*                                                          *
************************************************************
SET ESCAPE ON
SET TALK OFF
CLEAR
@ 10,10 SAY "PRINTING ENTIRE SCHEDULE TO PRINTER......"
@ 11,10 SAY "PRESS  [ESC] KEY TO ABORT PRINTING"

SET DEVICE TO PRINT
```

```
*  SET STRING CONSTANTS USED IN THE PROGRAM
*
STORE "------------------------------" TO DASHES
STORE "                                                   " TO BLANKS
STORE "______________________________" TO UNDERLINE
STORE "¦" TO BAR
*
*  INITIALIZE VARIABLES USED IN THE PROGRAM
*
STORE 10 TO LEFTCOL
STORE 50 TO MIDCOL
STORE 106 TO RIGHTCOL
store 75 to milepost
*
        STORE 10 TO COL1
        STORE 20 TO COL2
        STORE 35 TO COL3
        STORE 50 TO COL4
        STORE 65 TO COL5
        STORE 75 TO COL6
        STORE 85 TO COL7
        STORE 110 TO COL8
        store 125 to col9
*
     DO WHILE .NOT. EOF( )
*
     STORE 5 TO LINECNT
*
        @ LINECNT,55 SAY "RECORDER'S DAILY WORK SHEET"
        @ LINECNT,55 SAY "RECORDER'S DAILY WORK SHEET"

        STORE LINECNT+2 TO LINECNT
        @ LINECNT,LEFTCOL SAY SUBSTR("DISTRICT NO."+UNDERLINE,1,20)
        @ LINECNT,MIDCOL+10 SAY DAY_SETUP
        @ LINECNT,MIDCOL SAY SUBSTR("SETUPS"+UNDERLINE,1,20)
        @ LINECNT,RIGHTCOL SAY "SPEEDOMETER READING"

        STORE LINECNT+1 TO LINECNT
        @ LINECNT,LEFTCOL SAY SUBSTR("DATE"+UNDERLINE,1,20)
        @ LINECNT,MIDCOL SAY "DAY 1: " + DAY1 + "   "

        @ LINECNT,milepost SAY MILE1
        @ LINECNT,milepost+6 say "MILES"
        @ LINECNT,RIGHTCOL SAY SUBSTR("END OF DAY"+UNDERLINE,1,20)

        STORE LINECNT+1 TO LINECNT
        @ LINECNT,LEFTCOL SAY SUBSTR("DAY"+UNDERLINE,1,20)
        @ LINECNT,MIDCOL SAY "DAY 2: " + DAY2 + "   "
        @ LINECNT,milepost SAY MILE2
        @ LINECNT,milepost+6 say "MILES"
        @ LINECNT,RIGHTCOL SAY SUBSTR("BEGIN OF DAY"+UNDERLINE,1,20)

        STORE LINECNT+1 TO LINECNT
        @ LINECNT,RIGHTCOL SAY SUBSTR("TOTAL"+UNDERLINE,1,20)
*
```

```
*  THIS PORTION PRINTS THE COLUMN HEADINGS
*
        STORE LINECNT+2 TO LINECNT
        @ LINECNT,COL1 SAY DASHES + DASHES + DASHES +DASHES
        STORE LINECNT+1 TO LINECNT
*
        @ LINECNT,COL1 SAY bar + " Station"
        @ LINECNT,col2 say bar
        @ LINECNT,col3 say bar + "    Arrival"
        @ LINECNT,col4 say bar + "  Departure"
        @ LINECNT,col5 say bar + "   Last"
        @ LINECNT,col6 say bar + "   This"
        @ LINECNT,col7 say bar
        @ LINECNT,col8 say bar + "   Machine"
        @ LINECNT,col9 say bar
*
        store LINECNT+1 to LINECNT
*
*
        @ LINECNT,COL1 SAY BAR + " Number"
        @ LINECNT,COL2 SAY BAR + "   County"
        @ LINECNT,col3 say bar + "     Time"
        @ LINECNT,col4 say bar + "     Time"
        @ LINECNT,col5 say bar + "  Count"
        @ LINECNT,col6 say bar + "  Count"
        @ LINECNT,col7 say bar + "          Remarks"
        @ LINECNT,col8 say bar + "   Number"
        @ LINECNT,col9 say bar
*
*
        store LINECNT+1 to LINECNT
        @ LINECNT,col1 say DASHES + DASHES + DASHES + DASHES
*
        @ LINECNT,COL1 SAY BAR
        @ LINECNT,COL2 SAY BAR
        @ LINECNT,COL3 SAY BAR
        @ LINECNT,COL4 SAY BAR
        @ LINECNT,COL5 SAY BAR
        @ LINECNT,COL6 SAY BAR
        @ LINECNT,COL7 SAY BAR
        @ LINECNT,COL8 SAY BAR
        @ LINECNT,COL9 SAY BAR
*
*
        STORE LINECNT+1 TO LINECNT
*




*
```

```
*  PRINT OUT THE FIRST RECORD OF EACH PAGE
*

        @ LINECNT,col1 say bar
        @ LINECNT,col1+2 say station
        @ LINECNT,col2 say bar
        @ LINECNT,col2+2 say county
        @ LINECNT,col3 say bar
        @ LINECNT,col4 say bar
        @ LINECNT,col5 say bar
        @ LINECNT,col6 say bar
        @ LINECNT,col7 say bar
        @ LINECNT,col8 say bar
        @  LINECNT,col9 say bar
*
        STORE LINECNT+1 TO LINECNT

        @ LINECNT,COL1 SAY  DASHES + DASHES + DASHES + DASHES
        @ LINECNT,COL1 SAY BAR
        @ LINECNT,COL2 SAY BAR
        @ LINECNT,COL3 SAY BAR
        @ LINECNT,COL4 SAY BAR
        @ LINECNT,COL5 SAY BAR
        @ LINECNT,COL6 SAY BAR
        @ LINECNT,COL7 SAY BAR
        @ LINECNT,COL8 SAY BAR
        @ LINECNT,COL9 SAY BAR

        STORE SETUP TO XSETUP
        STORE LINECNT+1 TO LINECNT

        IF (XSETUP>1)

           DO WHILE (XSETUP>1)
              *
             * PRINT BLANK LINES HERE
             *
             *

                     @ LINECNT,COL1 SAY BAR
                     @ LINECNT,COL2 SAY BAR
                     @ LINECNT,COL3 SAY BAR
                     @ LINECNT,COL4 SAY BAR
                     @ LINECNT,COL5 SAY BAR
                     @ LINECNT,COL6 SAY BAR
                     @ LINECNT,COL7 SAY BAR
                     @ LINECNT,COL8 SAY BAR
                     @ LINECNT,COL9 SAY BAR

                   STORE LINECNT+1 TO LINECNT
                   @ LINECNT,COL1 SAY  DASHES + DASHES + DASHES + DASHES
                   @ LINECNT,COL1 SAY BAR
                   @ LINECNT,COL2 SAY BAR
                   @ LINECNT,COL3 SAY BAR
                   @ LINECNT,COL4 SAY BAR
                   @ LINECNT,COL5 SAY BAR
                   @ LINECNT,COL6 SAY BAR
                   @ LINECNT,COL7 SAY BAR
                   @ LINECNT,COL8 SAY BAR
                   @ LINECNT,COL9 SAY BAR

               STORE LINECNT+1 TO LINECNT
               STORE XSETUP-1 TO XSETUP

           ENDDO
        ELSE

             @ LINECNT,COL1 SAY BAR
             @ LINECNT,COL2 SAY BAR
             @ LINECNT,COL3 SAY BAR
             @ LINECNT,COL4 SAY BAR
             @ LINECNT,COL5 SAY BAR
             @ LINECNT,COL6 SAY BAR
             @ LINECNT,COL7 SAY BAR
             @ LINECNT,COL8 SAY BAR
             @ LINECNT,COL9 SAY BAR

        ENDIF

        SKIP
*
*   LOOP THROUGH THE FILE AND PRINT OUT SCHEDULE
*

        DO WHILE ( (NEWDAY<>'Y') .AND. (LINECNT+SETUP)<=70 .AND. (.NOT. EOF()))

           STORE SETUP TO XSETUP

*
*  PRINT CURRENT RECORD
*
        @ LINECNT,col1 say bar
        @ LINECNT,col1+2 say station
        @ LINECNT,col2 say bar
        @ LINECNT,col2+2 say county
        @ LINECNT,col3 say bar
        @ LINECNT,col4 say bar
        @ LINECNT,col5 say bar
        @ LINECNT,col6 say bar
        @ LINECNT,col7 say bar
        @ LINECNT,col8 say bar
        @ LINECNT,col9 say bar
*
        STORE LINECNT+1 TO LINECNT
```

```
@ LINECNT,COL1 SAY  DASHES + DASHES + DASHES + DASHES
@ LINECNT,COL1 SAY BAR
@ LINECNT,COL2 SAY BAR
@ LINECNT,COL3 SAY BAR
@ LINECNT,COL4 SAY BAR
@ LINECNT,COL5 SAY BAR
@ LINECNT,COL6 SAY BAR
@ LINECNT,COL7 SAY BAR
@ LINECNT,COL8 SAY BAR
@ LINECNT,COL9 SAY BAR

  DO WHILE (XSETUP>1)

    *
    *   PRINT OUT THE BLANK LINES HERE
    *
      STORE LINECNT+1 TO LINECNT

      @ LINECNT,COL1 SAY BAR
      @ LINECNT,COL2 SAY BAR
      @ LINECNT,COL3 SAY BAR
      @ LINECNT,COL4 SAY BAR
      @ LINECNT,COL5 SAY BAR
      @ LINECNT,COL6 SAY BAR
      @ LINECNT,COL7 SAY BAR
      @ LINECNT,COL8 SAY BAR
      @ LINECNT,COL9 SAY BAR


      STORE LINECNT+1 TO LINECNT

      @ LINECNT,COL1 SAY  DASHES + DASHES + DASHES + DASHES
      @ LINECNT,COL1 SAY BAR
      @ LINECNT,COL2 SAY BAR
      @ LINECNT,COL3 SAY BAR
      @ LINECNT,COL4 SAY BAR
      @ LINECNT,COL5 SAY BAR
      @ LINECNT,COL6 SAY BAR
      @ LINECNT,COL7 SAY BAR
      @ LINECNT,COL8 SAY BAR
      @ LINECNT,COL9 SAY BAR

   STORE XSETUP-1 TO XSETUP
  ENDDO

  SKIP
  STORE LINECNT+1 TO LINECNT

ENDDO

* DRAW UNDERLINE TO COMPLETE A DAY
* PRINT SIGNATURE LINE

  @ 80,50 SAY "Recorder Signature" + UNDERLINE + UNDERLINE
```

```
        EJECT

        * GOTO NEXT PAGE
        *
     ENDDO
*
        @ LINECNT,1 SAY BLANKS + BLANKS + BLANKS + BLANKS
  SET DEVICE TO SCREEN
  CLEAR
  EJECT
  RETURN

************************************************************
*                                                          *
*                        STATUS.PRG                        *
*                                                          *
*    MENU DRIVER PROGRAM TO DETERMINE THE CURRENT STATUS   *
*    OF THE DATA COLLECTION ACTIVITIES.                    *
*                                                          *
************************************************************
*
CLEAR

*  SELECT AN ITEM  OR EXIT TO PREVIOUS MENU
*

DO WHILE .T.

* DISPLAY MENU SCREEN
*

@ 6,16 SAY "___________________________________________"
@ 7,15 SAY "¦                                          ¦"
@ 8,15 SAY "¦    1.  SELECT EXISTING DATABASE FILE     ¦"
@ 9,15 SAY "¦    2.  CURRENT STATUS REPORT             ¦"
@ 10,15 SAY "¦                                          ¦"
@ 11,15 SAY "¦    X.  RETURN TO PREVIOUS MENU           ¦"
@ 12,15 SAY "¦ -----------------------------------------¦"
@ 13,15 SAY "¦                                          ¦"
@ 14,15 SAY "¦    SELECT A NUMBER OR 'X' TO EXIT        ¦"
@ 15,15 SAY "¦__________________________________________¦"
*
*  WAIT FOR USER SELECTION
*
WAIT ' ' TO CHOICE

        DO CASE CHOICE

                CASE CHOICE='1'
                        DO SELECT
                        * SELECT AN EXISTING FILE
                CASE CHOICE='2'
                        DO STATREPO
                        *  DISPLAY CURRENT STATUS

                CASE UPPER( CHOICE )='X'
                        CLEAR
                        *  RETURN TO PREVIOUS MENU
                        CLOSE DATABASES
                        RETURN

        ENDCASE

ENDDO

*
*
*************************************************************
*                                                           *
*                        STATREPO.PRG                       *
*                                                           *
*  PROGRAM TO DISPLAY CURRENT STATUS                        *
*  COUNT THE NUMBER OF STATIONS THAT HAVE BEEN COLLECTED    *
*                                                           *
*************************************************************
*
UNDONE=0
TOTALREC=0
GO TOP
@ 17,30 SAY "Counting..."
COUNT FOR THISCOUNT=0 TO UNDONE
GO BOTTOM
TOTALREC=RECNO( )
PERCENT=INT( UNDONE / TOTALREC *100 )

CLEAR
@  5,10 SAY UNDONE
@  5,25 SAY " STATIONS NOT COUNTED "
@  6,10 SAY PERCENT
@  6,25 SAY " % OF DISTRICTED NOT COUNTED"
WAIT
CLEAR
*
*
```

```
************************************************************
*                                                          *
*                   DISPFUNC.PRG                           *
*                                                          *
*  PROGRAM TO DISPLAY THE CURRENT FUNCTION KEY SETTINGS    *
*  READ FROM THE FUNCKEY.MEM                               *
*                                                          *
************************************************************
*
  CLEAR
  RESTORE FROM FUNKEY
*
   @ 18,2 SAY "______________________________________________________"
   @ 18,57 SAY "______________________"
   @ 19,1 SAY "¦"
   @ 19,79 SAY "¦"
   @ 20,1 SAY "¦  F1-"
   @ 20,8 SAY SUBSTR(FKEY1,1,10)
   @ 20,19 SAY "F2-"
   @ 20,23 SAY SUBSTR(FKEY2,1,10)
   @ 20,34 SAY "F3-"
   @ 20,38 SAY SUBSTR(FKEY3,1,10)
   @ 20,49 SAY "F4-"
   @ 20,53 SAY SUBSTR(FKEY4,1,10)
   @ 20,64 SAY "F5-"
   @ 20,68 SAY SUBSTR(FKEY5,1,10)
   @ 20,79 SAY "¦"
   @ 21,1 SAY "¦  F6-"
   @ 21,8 SAY SUBSTR(FKEY6,1,10)
   @ 21,19 SAY "F7-"
   @ 21,23 SAY SUBSTR(FKEY7,1,10)
   @ 21,34 SAY "F8-"
   @ 21,38 SAY SUBSTR(FKEY8,1,10)
   @ 21,49 SAY "F9-"
   @ 21,53 SAY SUBSTR(FKEY9,1,10)
   @ 21,64 SAY "F10-"
   @ 21,69 SAY SUBSTR(FKEY10,1,10)
   @ 21,79 SAY "¦"
   @ 22,1 SAY "¦_____________________________________________________"
   @ 22,56 SAY "_______________________¦"
*
* SET THE FUNCTION KEYS ACCORDING TO THE FUNKEY.MEM
*
        SET FUNCTION 2 TO TRIM('&FKEY2')
        SET FUNCTION 3 TO TRIM('&FKEY3')
        SET FUNCTION 4 TO TRIM('&FKEY4')
        SET FUNCTION 5 TO TRIM('&FKEY5')
        SET FUNCTION 6 TO TRIM('&FKEY6')
        SET FUNCTION 7 TO TRIM('&FKEY7')
        SET FUNCTION 8 TO TRIM('&FKEY8')
        SET FUNCTION 9 TO TRIM('&FKEY9')
        SET FUNCTION 10 TO TRIM('&FKEY10')
RETURN
*
```

APPENDIX B
PROGRAM LISTING OF FORTRAN-77 SYSTEM

```
**************************************************************************
*                                                                        *
*                        Data Entry Program                              *
*                        ==================                              *
*                                                                        *
*                Author: Duk-Jin Chang                                   *
*                Date: Aug. 27, 1986                                     *
*                Language: MS FORTRAN                                    *
*                                                                        *
*  This program is designed to build two data files for the traffic     *
*  volume count scheduling program.  One of the file is about the block *
*  layout and the other is about the sites on each block and the        *
*  counter assignments on each site.                                     *
*                                                                        *
**************************************************************************


      COMMON Dist( 50,50 ), DN, Nd( 100 ), NumBlk, SN
      COMMON /ChCom/ Ans, CntyNam, DName, Loc( 50,50 ), SName

      CHARACTER Ans
      CHARACTER*10 Loc, DName, Nd, SName
      CHARACTER*20 CntyNam

      INTEGER Dist, DN, Length( 50 ), Locnt( 50 ), NumBlk, SN, TotUnt( 50 ),
     *        Unit( 50,50 )

   10 CALL GetCnst
      CALL NodeNam ( Length,Locnt,TotUnt,Unit )
      CALL GetDist
      CALL PrtMtx

      WRITE ( *,'( ///,A\ )' ) ' Would you like to SAVE it ( y/n )?   '
      READ ( *,'( A )' ) Ans
      IF (( Ans.EQ.'y' ) .OR. ( Ans.EQ.'Y' )) THEN
        CALL Save ( Length,Locnt,TotUnt,Unit )
      ENDIF

      WRITE ( *,'( ///,A\ )' ) ' Would you like to try again ( y/n )?   '
      READ ( *,'( A )' ) Ans
      IF (( Ans.EQ.'y' ) .OR. ( Ans.EQ.'Y' )) GOTO 10

      STOP
      END
```

```
*     ******************
      SUBROUTINE GetCnst
*     ******************
*     { GetCnst gets the constant values for a county. }

      COMMON Dist( 50,50 ), DN, Nd( 100 ), NumBlk, SN
      COMMON /ChCom/ Ans, CntyNam, DName, Loc( 50,50 ), SName

      CHARACTER Ans
      CHARACTER*10 Loc, DName, Nd, SName
      CHARACTER*20 CntyNam

      INTEGER Dist, DN, NumBlk, SN

      WRITE ( *,'( A\ )' ) ' Enter the name of County:   '
      READ ( *,'( A )' ) CntyNam

      WRITE ( *,'( A\ )' ) ' Enter the number of blocks in the county:   '
      READ ( *,* ) NumBlk

      WRITE ( *,'( A\ )' ) ' Enter the name of the Start Node:   '
      READ ( *,'( A )' ) SName

      WRITE ( *,'( A\ )' ) ' Enter the name of the End Node:   '
      READ ( *,'( A )' ) DName

      RETURN
      END
```

```
*     *********************************************
      SUBROUTINE NodeNam ( Length,Locnt,TotUnt,Unit )
*     *********************************************
*     ( NodeNam gets the site IDs and the number of counter to be assigned
*       on those sites in each block and the length of the block.  }

      COMMON Dist( 50,50 ), DN, Nd( 100 ), NumBlk, SN
      COMMON /ChCom/ Ans, CntyNam, DName, Loc( 50,50 ), SName

      CHARACTER Ans
      CHARACTER*10 Loc, DName, Nd, SName
      CHARACTER*20 CntyNam

      INTEGER Dist, DN, Length( 50 ), Locnt( 50 ), NumBlk, SN, TotUnt( 50 ),
     *        Unit( 50,50 )

      Nd( 1 ) = ' '
      Locnt( 1 ) = 0
      DO 20 i = 2, NumBlk + 1
        TotUnt( i ) = 0
        WRITE (*,'( //,A,I2,A\ )' ) ' Enter the First Node of the block ',
     *                        i-1, ':    '
        READ (*,'( A )' ) Nd( i )
        Loc( i,1 ) = Nd( i )
        IF ( Nd( i ) .EQ. SName )  SN = i
        IF ( Nd( i ) .EQ. DName )  DN = i

        DO 10 j = 2, 50
          WRITE (*,'( A\ )' ) ' How many units are assigned?    '
          READ (*,'( i1 )' ) Unit( i,j-1 )
          TotUnt( i ) = TotUnt( i ) + Unit( i,j-1 )
          WRITE (*,'( A\ )' ) ' Enter the next node:    '
          READ (*,'( A )' ) Loc( i,j )
          IF ( Loc( i,j ) .EQ. ' ' ) THEN
            Locnt( i ) = j - 1
            Nd( i+NumBlk ) = Loc( i,j-1 )
            IF ( Nd( i+NumBlk ) .EQ. SName )  SN = i+NumBlk
            IF ( Nd( i+NumBlk ) .EQ. DName )  DN = i+NumBlk
            WRITE (*,'( A\ )' ) ' Enter the length of the block:   '
            READ (*,*) Length( i )
            GOTO 20
          ENDIF
   10   CONTINUE
   20 CONTINUE

      WRITE (*,'( ///// )' )

      RETURN
      END
```

```
*     ******************
      SUBROUTINE GetDist
*     ******************
*     { GetDist builds the distance matrix among the blocks. }

      COMMON Dist( 50,50 ), DN, Nd( 100 ), NumBlk, SN
      COMMON /ChCom/ Ans, CntyNam, DName, Loc( 50,50 ), SName

      CHARACTER Ans
      CHARACTER*10 Loc, DName, Nd, SName
      CHARACTER*20 CntyNam

      INTEGER Dist, DN, NumBlk, SN

      DO 20 i = 2, 2*NumBlk
        DO 10 j = i+1, 2*NumBlk + 1
          IF ( i .EQ. j+NumBlk ) THEN
            Dist( i,j ) = 0
            Dist( j,i ) = 0
          ELSE
            WRITE ( *,'( A,A,A,A,A\ )' ) ' Distance from ', Nd( i ),
     *                               ' to ', Nd( j ), '?   '
            READ ( *,* ) Dist( i,j )
            Dist( j,i ) = Dist( i,j )
          ENDIF
   10   CONTINUE
   20 CONTINUE

      DO 30 i = 1, 2*NumBlk + 1
        Dist( i,i ) = 999
   30 CONTINUE

      DO 40 i = 2, 2*NumBlk + 1
        Dist( 1,i ) = 999
        Dist( i,1 ) = 999
   40 CONTINUE

      Dist( 1,SN ) = 0
      Dist( DN,1 ) = 0

      RETURN
      END
```

```
*     *****************
      SUBROUTINE PrtMtx
*     *****************
*     { PrtMtx prints out the resulting distance matrix. }

      COMMON Dist( 50,50 ), DN, Nd( 100 ), NumBlk, SN
      COMMON /ChCom/ Ans, CntyNam, DName, Loc( 50,50 ), SName

      CHARACTER Ans
      CHARACTER*10 Loc, DName, Nd, SName
      CHARACTER*20 CntyNam

      INTEGER Dist, DN, NumBlk, SN

      WRITE ( *,1000 )
 1000 FORMAT ( ////,10X,'** ORIGINAL MATRIX **'/ )

      N = 2*NumBlk + 1
      K = 30
      N1 = 1
      N2 = K
   10 IF ( N2 .GT. N ) N2 = N
      WRITE ( *,2000 ) ( II,II = N1,N2 )
 2000 FORMAT ( //6X,30I4 )
      WRITE ( *,'( A )' ) ' '

      DO 20 L = 1,N
        WRITE ( *,3000 ) L,( Dist( L,M ),M = N1,N2 )
 3000   FORMAT ( I3,3X,30I4 )
   20 CONTINUE

      IF ( N2 .LT. N ) THEN
        N1 = N1 + K
        N2 = N2 + K
        GO TO 10
      ENDIF

      RETURN
      END
```

```
*     ******************************************
      SUBROUTINE Save ( Length,Locnt,TotUnt,Unit )
*     ******************************************
*     { Save creates data files on the floppy disk. }

      COMMON Dist( 50,50 ), DN, Nd( 100 ), NumBlk, SN
      COMMON /ChCom/ Ans, CntyNam, DName, Loc( 50,50 ), SName

      CHARACTER Ans
      CHARACTER*10 Loc, DName, Nd, SName
      CHARACTER*12 FN
      CHARACTER*20 CntyNam

      INTEGER Dist, DN, Length( 50 ), Locnt( 50 ), NumBlk, SN,
     *        TotUnt( 50 ), Unit( 50,50 )


      WRITE ( *,'( A\ )' ) ' Enter the file name for block distance:   '
      READ ( *,'( A )' ) FN

      OPEN ( 6, FILE=FN, STATUS='NEW' )

      WRITE ( 6,'( A20 )' ) CntyNam
      WRITE ( 6,'( I10 )' ) NumBlk
      WRITE ( 6,'( 2I10 )' ) SN, DN

      DO 20 i = 2, 2*NumBlk
        DO 10 j = i+1, 2*NumBlk + 1
          IF ( i+NumBlk .NE. j ) THEN
            WRITE ( 6,'( 3I10 )' ) i, j, Dist( i,j )
          ENDIF
10      CONTINUE
20    CONTINUE

      WRITE ( *,'( A\ )' ) ' Enter the file name for site information:   '
      READ ( *,'( A )' ) FN

      OPEN ( 7, FILE=FN, STATUS='NEW' )

      WRITE ( 7,'( A4,A20 )' ) 'SITE', CntyNam
      DO 40 i = 2, NumBlk+1
        WRITE ( 7,'( I2,3I3 )' ) i, Locnt( i ), TotUnt( i ), Length( i )
        DO 30 j = 1, Locnt( i )
          WRITE ( 7,'( A10,I2 )' ) Loc( i,j ), Unit( i,j )
30      CONTINUE
40    CONTINUE

      RETURN
      END
```

```
************************************************************************
*                                                                      *
*                         Scheduling Program                           *
*                         ==================                           *
*                                                                      *
*                 Author: Duk-Jin Chang                                *
*                 Date: Aug. 27, 1986                                  *
*                 Language: MS FORTRAN                                 *
*                                                                      *
*  The Scheduling Program is designed to find the feasible routs for   *
*  the traffic volume count in a county and produce the schedules.     *
*                                                                      *
************************************************************************

      COMMON Dist( 50,50 ),Tmp( 50,50 ),ROWUSD( 50 ),COLUSD( 50 ),TREE( 50,2 ),
     *       UP( 50,50 ),NP( 50 ),INFSTC( 50,3 ),Nodes,INF,LEVEL,M,N,MAX

      INTEGER MIN, MING, NumBlk, up


*     ( Mainline Control )

      CALL INIT (MING, NumBlk)

      CALL ROUTNG (NumBlk, MIN, MING)

      CALL SCHDLNG (MIN, MING, NumBlk)


      STOP
      END
```

```
*     ****************************
      SUBROUTINE INIT (MING,NumBlk)
*     ****************************

      COMMON Dist( 50,50 ),Tmp( 50,50 ),ROWUSD( 50 ),COLUSD( 50 ),TREE( 50,2 ),
     *       UP( 50,50 ),NP( 50 ),INFSTC( 50,3 ),Nodes,INF,LEVEL,M,N,MAX

      INTEGER Dist, DN, NumBlk, SN, Tmp, TREE, UP
      CHARACTER*20 CntyNam, FN
      LOGICAL ROWUSD,COLUSD

*     { Data I/O and initializations }
      INF = 999
      MING = INF

      WRITE (*,'(A\)') ' Enter the file name of block distance:  '
      READ (*,'(A)') FN
      OPEN (5, FILE = FN)

      READ (5,'(A20)') CntyNam
      READ (5,'(I10)') NumBlk
      READ (5,'(2I10)') SN, DN

      Nodes = 2*NumBlk + 1
      DO 20 I = 1,Nodes
        INFSTC( I,1 ) = INF
        ROWUSD( I ) = .FALSE.
        COLUSD( I ) = .FALSE.
        TREE( I,1 ) = 0
        TREE( I,2 ) = 0
        DO 10 j = 1, Nodes
          Dist( i,j ) = INF
   10   CONTINUE
   20 CONTINUE


   30 READ( 5,'( 3I10 )', END=40 ) i, j, Dist( i,j )
      Dist( j,i ) = Dist( i,j )
      GOTO 30

   40 Dist( 1,SN ) = 0
      Dist( DN,1 ) = 0
      DO 80 i = 1, Nodes
        DO 70 j = 1, Nodes
          IF (( i.GE.2 ) .AND. ( i+NumBlk.EQ.j )) THEN
            Dist( i,j ) = 0
            Dist( j,i ) = 0
          ENDIF
          Tmp( i,j ) = Dist( i,j )
          UP( i,j ) = 0
   70   CONTINUE
   80 CONTINUE

      RETURN
      END
```

```
*     *************************************
      SUBROUTINE ROUTNG (NumBlk, MIN, MING)
*     *************************************

      COMMON Dist( 50,50 ),Tmp( 50,50 ),ROWUSD( 50 ),COLUSD( 50 ),TREE( 50,2 ),
     *       UP( 50,50 ),NP( 50 ),INFSTC( 50,3 ),Nodes,INF,LEVEL,M,N,MAX

      INTEGER NumBlk,TREE,UP

*     { Beginning of the TRAVELING SALESMAN ALGORITHM }
   10 LEVEL = 1

      CALL PRTMTX

      CALL REDCTN (NumBlk,MIN)

      IF (MIN .GE. MING) THEN
        LEVEL1 = LEVEL - 1
        DO 20 I = 1,LEVEL1
          IF ( TREE( LEVEL-I,1 ) .LT. MING) THEN
            CALL REINIT ( I )
            GOTO 10
          ENDIF
   20   CONTINUE
      ELSE
        MING = MIN
        IS = 1

        DO 50 I = 1,Nodes
          NP( I ) = IS
          DO 30 J = 1,Nodes
            IF( UP( IS,J ) .NE. 0 ) GO TO 40
   30     CONTINUE

          WRITE (*,1000) I,J,IS
 1000     FORMAT( 10X,'ROUTE UNDEFINED',3I10/ )
   40     IS = J
   50   CONTINUE

        NP( Nodes+1 ) = IS
      ENDIF

      RETURN
      END
```

```
*     *****************
      SUBROUTINE PRTMTX
*     *****************

      COMMON Dist( 50,50 ),Tmp( 50,50 ),ROWUSD( 50 ),COLUSD( 50 ),TREE( 50,2 ),
     *       UP( 50,50 ),NP( 50 ),INFSTC( 50,3 ),Nodes,INF,LEVEL,M,N,MAX

      INTEGER Tmp

      WRITE ( *,1000 )
 1000 FORMAT( 10X,'** ORIGINAL MATRIX **'/ )

      N = Nodes
      K = 30
      N1 = 1
      N2 = K
   10 IF ( N2 .GT. N ) N2 = N
      WRITE ( *,2000 ) ( II,II = N1,N2 )
 2000 FORMAT ( //6X,30I4 )
      WRITE ( *,'( A )' ) ' '

      DO 20 L = 1,N
        WRITE ( *,3000 ) L,( Tmp( L,M ),M = N1,N2 )
 3000   FORMAT ( I3,3X,30I4 )
   20 CONTINUE

      IF ( N2 .LT. N ) THEN
        N1 = N1 + K
        N2 = N2 + K
        GO TO 10
      ENDIF

      RETURN
      END
```

```
*     ******************************
      SUBROUTINE REDCTN (NumBlk, MIN)
*     ******************************

      COMMON Dist(50,50),Tmp(50,50),ROWUSD(50),COLUSD(50),TREE(50,2),
     *       UP(50,50),NP(50),INFSTC(50,3),Nodes,INF,LEVEL,M,N,MAX

      INTEGER NumBlk, Tmp, TREE,UP
      LOGICAL ROWUSD,COLUSD

      MIN = 0
   10 M = 0
      N = 0
      MAX = 0

      CALL ARCO (NumBlk)

      ROWUSD(M) = .TRUE.
      COLUSD(N) = .TRUE.
      UP(M,N) = LEVEL

      L = M
      K = N
      IF (M .EQ. 1) THEN
        CALL IMRDTN (NumBlk, L, N, 1)
      ELSE
        IF (N .EQ. 1) THEN
          CALL IMRDTN (NumBlk, K, M, 2)
        ELSE
          IF (ABS(M-N) .NE. NumBlk) THEN
            IF (.NOT. ROWUSD(N)) CALL IMRDTN (NumBlk, L, N, 1)
            IF (.NOT. COLUSD(M)) CALL IMRDTN (NumBlk, K, M, 2)
          ENDIF
        ENDIF
      ENDIF

      IF (LEVEL .LT. (Nodes-1)) THEN
        L = N
   20   DO 30 I = 1,Nodes
          IF (UP(L,I) .NE. 0) THEN
            L = I
            GOTO 20
          ENDIF
   30   CONTINUE

        K = M
   40   DO 50 I = 1,Nodes
          IF (UP(I,K) .NE. 0) THEN
            K = I
            GO TO 40
          ENDIF
   50   CONTINUE

        Tmp(L,K) = INF
      ENDIF
      TREE( LEVEL,1 ) = MIN + MAX
      MIN = MIN + MINCOL( I ) + MINROW( I )
      TREE( LEVEL,2 ) = MIN
      LEVEL = LEVEL + 1

      IF ( LEVEL .LE. Nodes ) GO TO 10

      RETURN
      END




*     ***************************************
      SUBROUTINE IMRDTN (NumBlk, N1, N2, N3 )
*     ***************************************

      COMMON Dist( 50,50 ),Tmp( 50,50 ),ROWUSD( 50 ),COLUSD( 50 ),TREE( 50,2 ),
     *       UP( 50,50 ),NP( 50 ),INFSTC( 50,3 ),Nodes,INF,LEVEL,M,N,MAX

      INTEGER LEVEL, NumBlk, N1, N2, UP
      LOGICAL ROWUSD, COLUSD

        LEVEL = LEVEL + 1
        N1 = N2
        IF ( N1 .LE. NumBlk+1 ) THEN
          N2 = N1 + NumBlk
        ELSE
          N2 = N1 - NumBlk
        ENDIF

        IF ( N3.EQ.1 ) THEN
          ROWUSD( N1 ) = .TRUE.
          COLUSD( N2 ) = .TRUE.
          UP( N1,N2 ) = LEVEL
        ELSE
          ROWUSD( N2 ) = .TRUE.
          COLUSD( N1 ) = .TRUE.
          UP( N2,N1 ) = LEVEL
        ENDIF

      RETURN
      END
```

```
*     ************************
      SUBROUTINE ARCO ( NumBlk )
*     ************************

      COMMON Dist( 50,50 ),Tmp( 50,50 ),ROWUSD( 50 ),COLUSD( 50 ),TREE( 50,2 ),
     *       UP( 50,50 ),NP( 50 ),INFSTC( 50,3 ),Nodes,INF,LEVEL,M,N,MAX

      INTEGER NumBlk, Tmp, UP
      LOGICAL Accpt, ROWUSD,COLUSD

      DO 40 I = 1,Nodes
        IF ( .NOT. ROWUSD( I )) THEN
          DO 30 J = 1,Nodes
            IF (( .NOT. COLUSD( J )) .AND. ( Tmp( I,J ) .EQ. 0 )) THEN
              IF ( M .EQ. 0 ) THEN
                M = I
                N = J
              ENDIF
              MT = INF

              DO 10 K = 1,Nodes
                IF (( .NOT. ROWUSD( K ))
     *          .AND. ( Tmp( K,J ) .LT. MT )
     *          .AND. ( K .NE. I )) MT = Tmp( K,J )
  10          CONTINUE

              MAXT = MT
              MT = INF
              DO 20 K = 1,Nodes
                IF (( .NOT. COLUSD( K ))
     *          .AND. ( Tmp( I,K ) .LT. MT )
     *          .AND. ( K .NE. J )) MT = Tmp( I,K )
  20          CONTINUE

              MAXT = MAXT + MT
              Accpt = .FALSE.
              IF ( MAXT.GT.MAX ) THEN
                IF (( I.EQ.1 ).OR.( J.EQ.1 )) THEN
                  Accpt = .TRUE.
                ELSE
                  IF ( ABS( I-J ).EQ.NumBlk ) THEN
                    Accpt = .TRUE.
                  ELSE
                    IF ( I.LE.NumBlk ) THEN
                      IF ( UP( I+NumBlk,I ).NE.0 ) THEN
                        Accpt = .TRUE.
                      ENDIF
                    ELSE
                      IF ( UP( I-NumBlk,I ).NE.0 ) THEN
                        Accpt = .TRUE.
                      ENDIF
                    ENDIF
                  ENDIF
                ENDIF
              ENDIF
            IF (Accpt) THEN
              MAX = MAXT
              M = I
              N = J
            ENDIF

          ENDIF
 30     CONTINUE
      ENDIF
 40 CONTINUE

    RETURN
    END
```

```
*     *********************
      SUBROUTINE REINIT ( I )
*     *********************

      COMMON Dist( 50,50 ),Tmp( 50,50 ),ROWUSD( 50 ),COLUSD( 50 ),TREE( 50,2 ),
     *       UP( 50,50 ),NP( 50 ),INFSTC( 50,3 ),Nodes,INF,LEVEL,M,N,MAX

      INTEGER Dist, Tmp, TREE,UP
      LOGICAL ROWUSD,COLUSD

      LEVEL = LEVEL - I
      DO 20 I = 1,Nodes
        ROWUSD( I ) = .FALSE.
        COLUSD( I ) = .FALSE.
        TREE( I,1 ) = 0
        TREE( I,2 ) = 0
        DO 10 J = 1,Nodes
          Tmp( I,J ) = Dist( I,J )
          IF ( UP( I,J ) .NE. LEVEL ) THEN
            UP( I,J ) = 0
          ELSE
            M = I
            N = J
            UP( I,J ) = 0
          ENDIF
   10   CONTINUE
   20 CONTINUE

      PASS = 0.
      DO 30 I = 1,Nodes
        IF ( INFSTC( I,1 ) .GT. LEVEL ) THEN
          IF ( PASS .NE. 0. ) THEN
            INFSTC( I,1 ) = INF
            GOTO 30
          ELSE
            PASS = 1.
            INFSTC( I,1 ) = LEVEL
            INFSTC( I,2 ) = M
            INFSTC( I,3 ) = N
          ENDIF
        ENDIF
        II = INFSTC( I,2 )
        IJ = INFSTC( I,3 )
        Tmp( II,IJ ) = INF
   30 CONTINUE

      RETURN
      END
```

```
*     ********************
      FUNCTION MINCOL( KKK )
*     ********************

      COMMON Dist( 50,50 ),Tmp( 50,50 ),ROWUSD( 50 ),COLUSD( 50 ),TREE( 50,2 ),
     *       UP( 50,50 ),NP( 50 ),INFSTC( 50,3 ),Nodes,INF,LEVEL,M,N,MAX

      INTEGER Tmp
      LOGICAL ROWUSD,COLUSD


      MINCOL = 0
      DO 30 I = 1,Nodes
        IF ( .NOT. COLUSD( I )) THEN
          MIN = INF
          DO 10 J = 1,Nodes
            IF (( .NOT. ROWUSD( J )) .AND. ( Tmp( J,I ).LE.MIN )) THEN
              MIN = Tmp( J,I )
            ENDIF
   10     CONTINUE

          IF ( MIN .NE. 0 ) THEN
            DO 20 J = 1,Nodes
              Tmp( J,I ) = Tmp( J,I ) - MIN
   20       CONTINUE
          ENDIF
          MINCOL = MINCOL + MIN
        ENDIF

   30 CONTINUE

      RETURN
      END
```

```
*     ********************
      FUNCTION MINROW( KKK )
*     ********************

      COMMON Dist( 50,50 ),Tmp( 50,50 ),ROWUSD( 50 ),COLUSD( 50 ),TREE( 50,2 ),
     *       UP( 50,50 ),NP( 50 ),INFSTC( 50,3 ),Nodes,INF,LEVEL,M,N,MAX

      INTEGER Tmp
      LOGICAL ROWUSD,COLUSD


      MINROW = 0
      DO 30 I = 1,Nodes
        IF ( .NOT. ROWUSD( I )) THEN
          MIN = INF
          DO 10 J = 1,Nodes
            IF (( .NOT. COLUSD( J )) .AND. ( Tmp( I,J ).LE.MIN)) THEN
              MIN = Tmp( I,J )
            ENDIF
 10       CONTINUE

          IF ( MIN .NE. 0 ) THEN
            DO 20 J = 1,Nodes
              Tmp( I,J ) = Tmp( I,J ) - MIN
 20         CONTINUE
          ENDIF
          MINROW = MINROW + MIN
        ENDIF

 30   CONTINUE

      RETURN
      END
```

```
*     ***************************************
      SUBROUTINE SCHDLNG ( MIN, MING, NumBlk )
*     ***************************************

      COMMON Dist( 50,50 ),Tmp( 50,50 ),ROWUSD( 50 ),COLUSD( 50 ),TREE( 50,2 ),
     *       UP( 50,50 ),NP( 50 ),INFSTC( 50,3 ),Nodes,INF,LEVEL,M,N,MAX

      INTEGER BN, by, Dist, from, Length( 50 ), LnCnt, LN, Locnt( 50 ),
     *        Miles, NumBlk, to, TotUnt( 50 ), Unit( 50,50 ), UNs
      CHARACTER*10 Comment, Loc( 50,50 )
      CHARACTER*20 CntyNam, FN

      WRITE ( *,'( ////,A\ )' ) ' Enter the file name of site information: '
      READ ( *,'( A )' ) FN  .
      OPEN ( 5,FILE=FN )

      WRITE ( *,'( ////,A\ )' ) ' Enter the file name of site list: '
      READ ( *,'( A )' ) FN
      OPEN ( 6,FILE=FN, STATUS='NEW' )

      READ ( 5, '( A )' ) Comment

      DO 110 BN = 2, NumBlk+1
        READ ( 5,'( I2,3I3 )' ) BN, Locnt( BN ), TotUnt( BN ), Length( BN )
        DO 100 LN = 1, Locnt( BN )
          READ ( 5,'( A10,I2 )' ) Loc( BN,LN ), Unit( BN,LN )
  100   CONTINUE
  110 CONTINUE

      CALL HEADING ( CntyNam, LnCnt )

      DO 160 i = 2, Nodes, 2
        BN = NP( i )
        IF ( BN .LE. NumBlk+1 ) THEN
          from = 1
          to = Locnt( BN )
          by = 1
        ELSE
          BN = BN - NumBlk
          from = Locnt( BN )
          to = 1
          by = -1
        ENDIF

        IF (( UNs+TotUnt( BN )) .LE. 50 ) THEN
          UNs = UNs + TotUnt( BN )
          Miles = Miles + Dist( NP( i-1 ),NP( i )) + Length( BN )
        ELSE
          WRITE ( *,1100 ) UNs, Miles
 1100     FORMAT ( //60X, I2, ' SETUPS', 10X, I3, ' MILES' )
          CALL HEADING ( CntyNam, LnCnt )
          UNs = TotUnt( BN )
          Miles = Length( BN )
        ENDIF
        DO 130 j = from, to, by
          IF ( LnCnt .GE. 30 ) THEN
            CALL HEADING ( CntyNam, LnCnt )
          ENDIF
          WRITE ( *,1000 ) Loc( BN,j )
 1000     FORMAT ( T9,'¦',T20,'¦',T40,'¦',T53,'¦',T66,'¦',T76,'¦',
     *              T85,'¦',T121,'¦',T130,'¦',/,
     *              T9,'¦ ',A10,T20,'¦',T40,'¦',T53,'¦',T66,'¦',T76,'¦',
     *              T85,'¦',T121,'¦',T130,'¦',/,'+',T10,120( '_' ))
          WRITE ( 6,'( A10 )' ) Loc( BN,j )
          DO 120 k = 2, Unit( BN,j )
            WRITE ( *,1000 ) ' '
            WRITE ( 6,'( A )' ) ' '
  120     CONTINUE
          LnCnt = LnCnt + Unit( BN,j )
  130   CONTINUE

  160 CONTINUE

      WRITE ( *,1100 ) UNs, Miles

      RETURN
      END



*     **********************************
      SUBROUTINE HEADING ( CntyNam, LnCnt )
*     **********************************

      INTEGER LnCnt
      CHARACTER*20 CntyNam

      LnCnt = 0
      WRITE ( *,1000 )
 1000 FORMAT ( /,'1',T40,'R E C O R D E R '' S   D A I L Y   W O R K',
     *          '   S H E E T', /, T40, 53( '=' ), //,
     *        T10, 'DATE :', T80,'SPEEDOMETER READING :', /,
     *        T10,'DAY :', T81,'END OF DAY',/,
     *        T81,'BEGIN OF DAY',/,T81,'TOTAL', //,
     *        T10,120( '_' ),/,
     *        T9,'¦ Station', T20,'¦  County',T40,'¦    Arrival',
     *        T53,'¦ Departure',T66,'¦  Last', T76,'¦  This  ¦',
     *        T106,'Remarks', T121,'¦ Machine¦', /,
     *        T9,'¦ Number',T20,'¦',T40,'¦     Time', T53,'¦   Time',
     *        T66,'¦  Count',T76,'¦  Count ¦', T121,'¦ Number ¦',
     *        /, '+',T10,120( '_' ))

      RETURN
      END
```